МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

**М. А. Новотарський**

# ДИСКРЕТНА МАТЕМАТИКА

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського*
*як навчальний посібник для студентів,*
*які навчаються за спеціальністю*
*123 «Комп'ютерна інженерія», спеціалізацією*
*«Комп'ютерні системи та мережі»*

Київ
КПІ ім. Ігоря Сікорського
2020

Рецензенти: *Романюк А. С.,* д. ф.-м. н., проф., зав. відділу теорії функцій Інституту математики НАН України
*Чемерис О. А*., д. т. н., с. н. с., заступник директора з наукової роботи Інституту проблем моделювання в енергетиці ім. Г. Є. Пухова НАН України

Відповідальний редактор *Стіренко С. Г.*, д. т. н., проф.

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 3 від 5.11.2020 р.) за поданням Вченої ради Факультету інформатики та обчислювальної техніки (протокол № 7 від 17.02.2020 р.)*

Електронне мережеве навчальне видання
*Новотарський Михайло Анатолійович*, д-р техн. наук, с.н.с.

# ДИСКРЕТНА МАТЕМАТИКА

Дискретна математика [Електронний ресурс] : навч. посіб. для студ. спеціальності 123 «Комп'ютерна інженерія», спеціалізації «Комп'ютерні системи та мережі» / М. А. Новотарський; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 11500 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 278 с.

Навчальний посібник створений за матеріалами лекцій з курсу «Дискретна математика». Він містить основні положення теорії множин, комбінаторики та теорії графів. Зокрема розглянуті основи теорії множин, відповідностей та відношень на множинах. Представлені відношення еквівалентності та порядку, детально описані їх властивості. Основні положення комбінаторики представлені законами комбінаторики, комбінаторними вибірками та типовими комбінаторними алгоритмами. Значна увага приділена теорії графів. Розглянуті базові визначення теорії графів, способи створення та властивості графів, відношення та відображення на графах, магічні числа графів. Окремо розглянуто дерева, їх властивості та ліс. Описані основні алгоритми на графах, призначені для обходу графів, визначення найкоротших шляхів у них та побудови мінімальних остовних дерев. Наведені базові поняття та теореми, пов'язані з правильним мінімальним розфарбуванням графів. Дано код або псевдокод основних алгоритмів розфарбування.

Посібник може бути корисним для інженерів та студентів технічних спеціальностей.

# ЗМІСТ

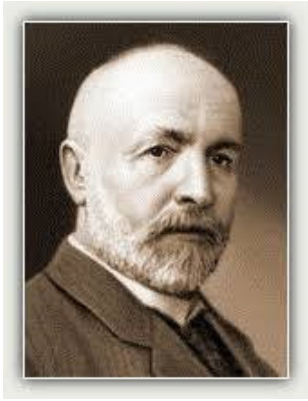
## Розділ 1. Теорія множин

## Розділ 3. Теорія графів

# Розділ 1. Теорія множин

## 1.1. Основні положення теорії множин

### 1.1.1. Основні визначення



Базові поняття теорії множин створені математиками XIX ст., які працювали над основами математичного аналізу. Основу теорії множин заклав німецький математик Георг Кантор (1845–1918). Йому належить висловлення, яке сьогодні розглядається як інтуїтивне визначення множини: «Множина – це розмаїття, мислиме як єдине».

Однак пізніше виявилося, що таке визначення множини має внутрішнє протиріччя. Прикладом може служити парадокс Рассела, що одержав у популярній літературі назву парадокса перукаря. Суть його полягає в нерозв'язності питання про те, чи повинен голитися перукар, який дав обіцянку голити всіх у його селі, хто не голиться сам?

*Визначення множини*

**Множина** – сукупність різноманітних об'єктів, що мають певну спільну властивість, яка об'єднує їх у єдине ціле.

**Приклади множин:** множина студентів, присутніх на лекції, множина парних чисел, множина громадян України. Елементом множини може бути інша множина.

Таке визначення множини вимагає введення наступних позначень:

1. *Множина* позначається великою літерою будь-якого алфавіту.

Наприклад: $A$, $B$, $C$, …, $X$, $Y$, …, $Z$.

2. *Елемент* множини здебільшого позначається малою літерою будь-якого алфавіту.

**Приклад 1.1.** Можливі позначення елементів множини: $a$, $b$, $c$,…, $x$, $y$,…, $z$.

Елементом множини може бути інша множина. У цьому випадку елементом множини є множина, яка представлена своєю назвою або елементами у фігурних дужках.

**Приклад 1.2.** Нехай $B = \{d, c, n\}$. Тоді $A = \{a, b, c, B\}$ або $A = \{a, b, c, \{d, c, n\}\}$.

3. *Приналежність* елемента множині позначають символом $\in$ – «належить». Якщо елемент не належить множині, то використовують позначення $\notin$ – «не належить». Елемент може належати множині, якщо він є одним з об'єктів цієї множини.

**Приклад 1.3.** Позначення $x \in X$ показує, що елемент $x$ належить множині $X$, тобто $x$ є одним з елементів множини $X$. Позначення $\{d,c,n\} \in X$ показує, що множина $\{d,c,n\}$ є елементом множини *X.*

Позначення $a \notin X$ показує, що елемент $a$ не належить множині $X$.

**Вправа 1.**

Нехай *A* – множина рослин, що ростуть у парку КПІ, *B* – множина квітів, *C* – множина дерев.

a) Назвіть два елементи множини *B*, що не є елементами множини *A*.

b) Назвіть два елементи множини *C*, що не є елементами множини *A*.

*Способи задавання множини*

1. Задавання множини *в явній формі* шляхом перерахування її елементів: $X = \{x_1, x_2, x_3, ..., x_n\}$, де $n$ – кількість елементів множини.

Такий спосіб задавання множини громіздкий і не використовується при великій кількості елементів.

2. Задавання множини *предикатом*.

У цьому випадку множину задають шляхом задавання умови або властивості $P(x)$, якій повинні задовольняти всі елементи $x$ множини. Властивість $P(x)$ називають предикатом. Предикат дозволяє із сукупності об'єктів виділити ті, що належать множині. У цьому випадку множину записують у такий спосіб:

$$X = \{x \mid P(x)\}.$$

Читають цей вираз так:

Множина $X$ складається з таких елементів $x$, що $P(x)$.

**Приклад 1.4.** Нехай предикат задано висловлюванням «$x$ є парним числом». Тоді множина $X$ складається з таких елементів $x$, що $x$ є парним числом, тобто елементи множини $X$ – парні числа.

3. Задавання множини *рекурсивною процедурою.* Рекурсивна процедура дозволяє визначити наступні елементи множини через попередні.

**Приклад 1.5.** Множину натуральних чисел $N = \{1,2,3,....\}$ можна задати в такий спосіб:

$$N = \{\, i \mid \text{якщо ціле } i \in N, \text{ то } i+1 \in N,\ i \geq 1 \,\}$$

**Приклад 1.6.** Задати множину *A*={*x*| $x \in$ *N*, *x* – дільник числа 20}.

*Розв'язок.*

*A*={*x*| $x \in$ *N*, *x* – дільник числа 20} = {1, 2, 4, 5, 10, 20}.

*Визначення підмножини*

Нехай кожний елемент множини має не одну властивість, а деякий набір властивостей. Однак для приналежності елемента до даної множини достатньо того, щоб тільки одна властивість із даного набору була спільною для всіх елементів.

У той же час, ніщо не заважає нам згрупувати елементи відповідно до будь-якої іншої властивості. У такий спосіб ми можемо утворювати **підмножини** даної множини, тобто сукупність елементів, що відрізняються від інших елементів даної множини за деякою властивістю.

**Множина $A$ є підмножиною множини $X$**, якщо кожний елемент множини $A$ є елементом множини $X$, тобто якщо $x \in A$, то $x \in X$. Зокрема, будь-яка множина є підмножиною самої себе.

Для визначення співвідношення множини й підмножини використовують позначення:

$\subseteq$ – «входить у»;

$\subset$ – «строго входить у».

$A \subseteq X$ – множина $A$ входить у множину $X$. При цьому не виключається випадок, коли всі елементи множини $A$ й множини $X$ збігаються.

$B \subset X$ – множина $B$ строго входить у множину $X$. У цьому випадку підкреслюється той факт, що в множині $X$ обов'язково існують елементи, що не належать множині $B$.

**Приклад 1.7.** Визначити, чи є множина $A$ підмножиною множини $X$ за умови, що $X=\{0,1,2,3,4,5,6,7,8,9\}$, $A=\{3,7,9\}$.

*Розв'язок.* За визначенням підмножини перевіримо приналежність усіх елементів множини $A$.

1. $3\in A$, $3\in X$.
2. $7\in A$, $7\in X$.
3. $9\in A$, $9\in X$.

Однак, $0\notin A$, $0\in X$. Тому $A\subset X$.

Будемо користуватися наступними позначеннями, які полегшують запис співвідношень, пов'язаних з підмножинами:

$\forall$ – символ, називаний квантором (лат. Quantum – скільки). Він означає «будь-який», «довільний»;

$\Rightarrow$ – символ наслідку (імплікації), що означає «спричиняє», «якщо..., то ...»;

$\Leftrightarrow$ – символ, що означає еквівалентність (у розумінні «те ж саме, що»).

*Визначення рівності множин*

Множина $X$ дорівнює множині $Y$ у випадку, якщо будь-який елемент $a$ належить множині $X$ $\left(a \in X\right)$ тоді й тільки тоді, коли $a \in Y$. Іншими словами $X = Y$ ($X$ дорівнює $Y$) тоді й тільки тоді, коли $X \subseteq Y$ й $Y \subseteq X$.

**Приклад 1.8.** Визначити, чи є рівними множини $X$ та $Y$ за умови, що $X=\{a,b,c,d\}$, $Y=\{c,a,b,d\}$.

*Розв'язок.* Перевіримо приналежність елементів множин $X$ та $Y$. Будь-який елемент, що належить множині $X$, також належить і множині $Y$. Отже, $X=Y$.

*Визначення порожньої множини*

**Порожня множина** – це множина, яка не містить елементів.

Позначення порожньої множини: $\varnothing$ або $\{\ \}$.

*Основна властивість порожньої множини*: порожня множина є підмножиною будь-якої множини.

**Вправа 2.**

Чи є множина, що складається з числа 0, порожньою множиною?

*Визначення універсальної множини*

**Універсальна множина** $U$ – це множина, властивістю якої є те, що всі розглянуті множини є її підмножинами.

*Властивості універсальної множини.*

***Властивість 1***. Будь-який об'єкт, яка б не була його природа, є елементом універсальної множини. $\forall x \to x \in U$ , зокрема $U \in U$ .

***Властивість 2***. Будь-яка множина є підмножиною універсальної множини. $\forall A \to A \subseteq U$ , зокрема $U \subseteq U$ .

*Універсальна множина має прикладний характер*

Множина $U$ , незважаючи на те, що названа універсальною, не може бути однозначно визначена, якщо не названа предметна область, тобто не зазначена властивість об'єктів, за якою дана множина формується.

1. Універсальна множина не є множиною всіх множин.
2. Універсальна множина є єдиною для певної чітко окресленої загальної ознаки її елементів.
3. Універсальна множина має прикладний характер.

**Приклад. 1.9.**

1. Універсальна множина в теорії чисел – множина цілих чисел.
2. Універсальна множина в математичному аналізі – множина дійсних чисел.

*Скінченні й нескінченні множини*

*Скінченна множина* – це множина, кількість елементів якої скінченна, тобто, існує невід'ємне ціле число *k*, яке дорівнює кількості елементів цієї множини. Якщо такого числа не існує, то множину називають нескінченною.

**Приклад 1.10.** Навести приклад скінченної множини.

*Розв'язок.*

Задамо скінченну множину перерахуванням $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

У цій множині $k = 10$.

*Нескінченна множина* – множина, що включає нескінченну кількість елементів. Якщо не існує невід'ємного цілого числа *k*, то множину називають нескінченною.

**Приклад 1.11.** Навести приклад нескінченної множини.

*Розв'язок.*

Задамо нескінченну множину предикатом $B = \{b \mid "b - \textit{ціле число}"\}$.

Для цієї множини не існує цілого невід'ємного числа *k*.

*Зліченна множина* – нескінченна множина, елементи якої принципово можливо пронумерувати натуральними числами.
*Незліченна множина* – така нескінченна множина, яка не є зліченною.

**Приклад 1.12.** Дати приклади нескінченних множин, до яких входять зліченні множини.
*Розв'язок.*
Множини натуральних чисел $N$, цілих чисел $Z$.
Множини раціональних чисел $Q$ і дійсних чисел $R$.
Множина комплексних чисел $C$.
**Зліченна множина** – нескінченна множина, елементи якої можливо пронумерувати натуральними числами.
**Зліченна множина** – нескінченна множина $X$, у якій існує взаємо-однозначна відповідність $X \leftrightarrow N$, де $N$ позначає множину всіх натуральних чисел.
**Зліченна множина** є «найменшою» нескінченною множиною, тобто в будь-якій нескінченній множині знайдеться зліченна підмножина.
**Властивість 1.** Будь-яка підмножина зліченної множини скінченна.
**Властивість 2.** Множина всіх скінченних підмножин зліченної множини зліченна.

**Приклад 1.13.** Приклад нескінченної зліченної множини.
Зліченна множина – зірки, які принципово можливо порахувати.
Скінченна множина, яка є підмножиною зліченної множини – це зірки, які занесені до зіркового каталогу.
**Незліченна множина** – така нескінченна множина, яка не є зліченною.


Рис. 1.1. Класифікація множин

Таким чином, будь-яка множина або скінченна, або зліченна, або незліченна.
*$n$-множина* – це множина, яка має $n$ елементів.
*Потужність множини* визначають як *кількість елементів* множини.
Потужність множини $X$ позначають $|X|$ або $\# X$.

Якщо | A | = | B |, то множини A и B *називають рівнопотужними.*

**Булеан**

*Визначення.* Множину всіх підмножин множини $M$ називають ***булеаном*** і позначають $2^M$

$$2^M = \{A | A \subseteq M\}.$$

Для скінченної множини $M$ з $n = |M|$ кількість її підмножин (потужність булеана) дорівнює $2^n = 2^{|M|}$. Отже, $|2^M| = 2^{|M|}$.

**Вправа 3.**

Нехай *A* – множина студентів вашої групи, яка її потужність? Яка потужність булеана вашої групи?

### 1.1.2. Операції над множинами

Операції над множинами дозволяють будувати нові множини, використовуючи вже існуючі.

*Об'єднання*

**Об'єднанням** множин $X$ і $Y$ називають множину, що складається із усіх тих і тільки тих елементів, які належать хоча б одній із множин $X, Y$, тобто належать $X$ або належать $Y$. (Об'єднання множин іноді називають сумою або додаванням множин).

Визначення об'єднання множин $X$ і $Y$ може бути записане в такий спосіб:

$$x \in X \cup Y \leftrightarrow x \in X \text{ або } x \in Y.$$

$\leftrightarrow$ – символ, що означає еквівалентність (у змісті «те ж саме, що»).

У літературі часто використовують ще й таке визначення:

$$X \cup Y = \{ x \mid x \in X \text{ або } x \in Y \}$$

Тут «або» відіграє не розділову (або-або), а об'єднавчу роль, тобто об'єднанню належать і спільні елементи згаданих множин.

**Приклад 1.14.** Знайти об'єднання множин $X$ і $Y$ за умови, що $X = \{1,2,3,4,5\}$, $Y = \{2, 5, 8, 9\}$.

*Розв'язок.* $X \cup Y = \{1,2,3,4,5,8,9\}$

**Об'єднання** множин у загальному випадку визначають у такий спосіб:

Нехай $I = \{1,2,3,\ldots,n\}$. Тоді

$$\bigcup_{i \in I} X_i = X_1 \cup X_2 \cup X_3 \cup \ldots \cup X_n = \{x \mid \text{існує } i \in I \text{ таке, що } x \in X_i\}.$$

*Перетин*

**Перетином** множин $X$ і $Y$ називають множину, що складається із усіх тих і тільки тих елементів, які належать як множині $X$, так і множині $Y$. (Перетин множин іноді називають добутком множин).

Перетин множин $X$ і $Y$ позначають так: $X \cap Y$ «перетин $X$ і $Y$».

Визначення перетину множин $X$ і $Y$ може бути записане в такий спосіб:

$$x \in X \cap Y \leftrightarrow x \in X \text{ і } x \in Y.$$

Часто використовують також визначення:

$$X \cap Y = \{ x \mid x \in X \text{ і } x \in Y \}.$$

**Приклад 1.15.** Знайти перетин множин $X$ і $Y$ за умови, що $X = \{1,2,3,4,5\}$ $Y = \{2, 5, 8, 9\}$.

*Розв'язок.* $X \cap Y = \{2,5\}$.

**Перетин** множин у загальному випадку визначають у такий спосіб:
Нехай $I = \{1,2,3,\ldots,n\}$. Тоді

$$\bigcap_{i \in I} X_i = X_1 \cap X_2 \cap X_3 \cap \ldots \cap X_n = \{x \mid x \in X_i \text{ для всіх } i \in I\}.$$

**Вправа 4.**

1. Найстарший математик серед шахістів і найстарший шахіст серед математиків – це та сама людина чи (можливо) різні?
2. Найкращий математик серед шахістів і найкращий шахіст серед математиків – це та сама людина чи (можливо) різні?

*Різниця*

Ця операція, на відміну від операцій об'єднання і перетину, може бути застосована тільки для двох множин.

**Різницею** множин $X$ і $Y$ називають множину, що складається з усіх тих і тільки тих елементів, які належать $X$, але не належать $Y$.

Різницю множин $X$ і $Y$ позначають $X \setminus Y$ або $X - Y$.

Визначення різниці множин $X$ і $Y$ може бути записане так:

$$x \in X \setminus Y \leftrightarrow x \in X \text{ і } x \notin Y$$

Різницю множин можна також визначити в такий спосіб:

$$X \setminus Y = \{x \mid x \in X \, i \, x \notin Y\}.$$

**Приклад 1.16.** Знайти $X \setminus Y$ і $Y \setminus X$, якщо $X = \{1,2,3,4,5\}$ і $Y = \{2,4,6,7\}$.

*Розв'язок.* $X \setminus Y = \{1,3,5\}$, $Y \setminus X = \{6,7\}$.

*Симетрична різниця*

**Симетричну різницю** множин $X$ і $Y$ визначають виразом
$X \Delta Y = (X \setminus Y) \cup (Y \setminus X)$.

**Приклад 1.17.** Знайти $X \Delta Y$ за умови, що $X = \{1,2,4,6,7\}$ і $Y = \{2,3,4,5,6\}$

*Розв'язок.* $X \setminus Y = \{1,7\}$, $Y - X = \{3,5\}$, $X \Delta Y = \{1,3,5,7\}$.

*Доповнення*

**Доповнення** множини $X$, позначуване як $\bar{X}$, визначають як множину елементів універсальної множини, що не належать множині $X$. Отже,

$$\bar{X} = U \setminus X = \{x \mid x \in U \text{ і } x \notin X\}.$$

Множина $\bar{X}$ (потрібно читати як «НЕ *X*») включає всі елементи універсальної множини, що не ввійшли в множину *X*.

Часто використовуваним еквівалентним позначенням множини $\bar{X}$ є позначення $\neg X$. Отже $\bar{X} = \neg X$.

### 1.1.3. Діаграми Венна (Ейлера)

***Діаграми Венна (Ейлера)*** – зручний інструмент, що дозволяє зображувати множини й ілюструвати операції над ними. Множини в діаграмах Венна зображують внутрішніми частинами кіл, їх перетинами, об'єднаннями і т. ін. На рис. 1.2 наведена діаграма Венна для множини $X$, яка зображена внутрішньою частиною кола. Зовнішня частина кола зображує $\overline{X}$.

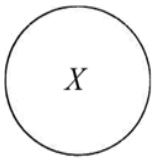



Рис. 1.2. Діаграма Венна для множини $X$

На рис. 1.3 наведена діаграма Венна для двох множин: $X$ і $Y$. Кожна множина зображена колом, і кола перетинаються.

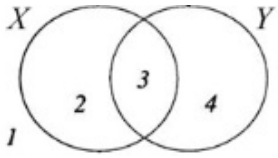



Рис. 1.3. Діаграма для множин $X$ і $Y$

На рис. 1.3 можна бачити 4 області: 1 – область універсальної множини, 2 – область, що належить тільки множині $X$, 3 – область, що належить спільно множинам $X$ і $Y$, 4 – область, що належить тільки множині $Y$.

На рис. 1.4 наведені ілюстрації операцій над множинами.

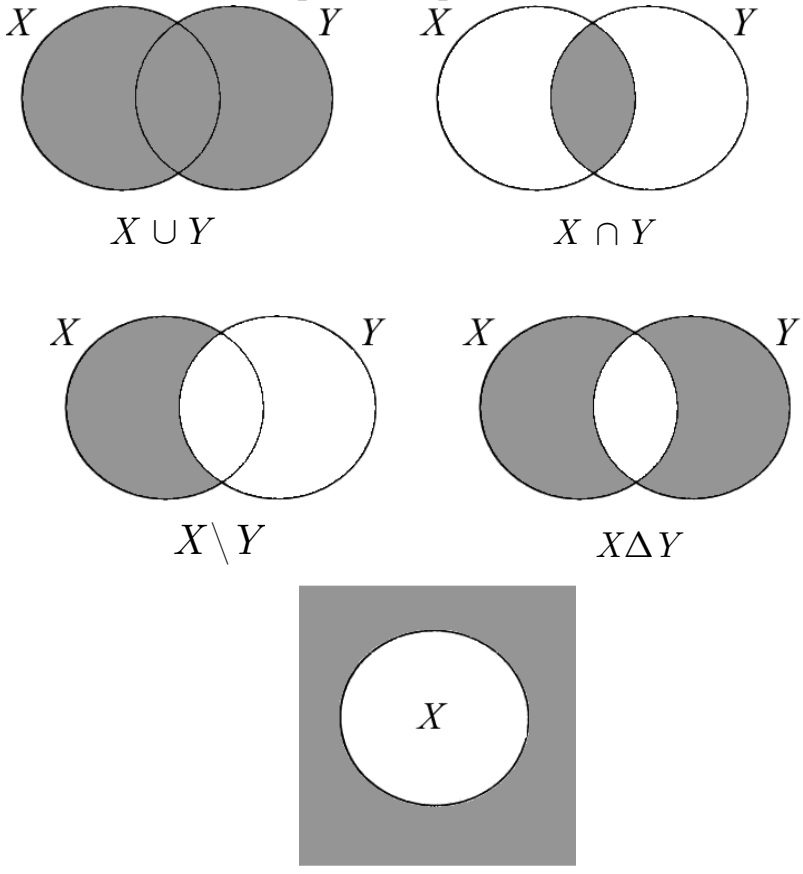



Рис. 1.4. Діаграми Венна операцій над множинами

### 1.1.4. Тотожності алгебри множин

**Тотожності алгебри множин**, сформульовані в наведених нижче теоремах, можуть бути перевірені шляхом формальних доведень або на діаграмах Венна.

**Таблиця 1.1.**

Тотожності алгебри множин

| | |
|---|---|
| 1. Комутативність об'єднання<br>$X \cup Y = Y \cup X$ | 1. Комутативність перетину<br>$X \cap Y = Y \cap X$ |
| 2. Асоціативність об'єднання<br>$X \cup (Y \cup Z) = (X \cup Y) \cup Z$ | 2. Асоціативність перетину<br>$X \cap (Y \cap Z) = (X \cap Y) \cap Z$ |
| 3. Дистрибутивність об'єднання відносно перетину<br>$X \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$ | 3. Дистрибутивність перетину відносно об'єднання<br>$X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z)$ |
| 4. Закони дії з порожніми і універсальними множинами<br>$X \cup \varnothing = X$<br>$X \cup \bar{X} = U$<br>$X \cup U = U$ | 4. Закони дії з порожніми і універсальними множинами<br>$X \cap U = X$<br>$X \cap \bar{X} = \varnothing$<br>$X \cap \varnothing = \varnothing$ |
| 5.Закон ідемпотентності об'єднання<br>Термін **ідемпотентність** означає властивість математичного об'єкта, яка проявляється в тому, що **повторна дія** над об'єктом *не змінює* його<br>$X \cup X = X$ | 5. Закон ідемпотентності перетину<br>$X \cap X = X$ |
| 6. Закон де Моргана<br>$\overline{X \cup Y} = \bar{X} \cap \bar{Y}$ | 6. Закон де Моргана<br>$\overline{X \cap Y} = \bar{X} \cup \bar{Y}$ |
| 7. Закон поглинання<br>$X \cup (X \cap Y) = X$ | 7. Закон поглинання<br>$X \cap (X \cup Y) = X$ |
| 8. Закон склеювання<br>$(X \cap Y) \cup (X \cap \bar{Y}) = X$ | 8. Закон склеювання<br>$(X \cup Y) \cap (X \cup \bar{Y}) = X$ |
| 9. Закон Порецького<br>$X \cup (\bar{X} \cap Y) = X \cup Y$ | 9. Закон Порецького<br>$X \cap (\bar{X} \cup Y) = X \cap Y$ |
| 10. Закон подвійного доповнення $\bar{\bar{X}} = X$ | |

У справедливості перерахованих властивостей можна переконатися у різний спосіб. Наприклад, намалювати діаграми Ейлера для лівої й правої частин тотожностей й переконатися, що вони збігаються, або ж провести формальне міркування для кожної тотожності. Розглянемо для прикладу першу тотожність: $A \cup A = A$. Візьмемо довільний елемент $x$, що належить до лівої

частини тотожності, $x \in A \cup A$. За визначенням операції об'єднання маємо: $x \in A$ або $x \in A$. У кожному разі $x \in A$. Візьмемо знову довільний елемент з множини в лівій частині тотожності. Виявляється, що він належить множині в правій частині. Звідси за визначенням включення множин одержуємо, що $A \cup A \subseteq A$ . Нехай тепер $x \in A$. Тоді, очевидно, вірно, що $x \in A$ або $x \in A$. Звідси за визначенням операції об'єднання маємо $x \subseteq A \cup A$. Таким чином, $A \subseteq A \cup A$ .

Отже, за визначенням тотожності множин: $A \cup A = A$. Аналогічні міркування неважко провести й для інших тотожностей.

Для доведення можна використовувати тотожності алгебри логіки.

Доведемо в такий спосіб властивість дистрибутивності множин.

**Теорема 1.1.** Для множин $X$ і $Y$ справджується тотожність:

$$X \cap \left(Y \cup Z\right) = \left(X \cap Y\right) \cup \left(X \cap Z\right)$$

*Доведення.*

При доведенні скористаємося операціями алгебри логіки, оскільки для доведення співвідношень на множинах необхідно довести, що ці співвідношення справедливі для всіх його елементів.

$x \in X \cap \left(Y \cup Z\right) \leftrightarrow \left(x \in X\right) \wedge \left(x \in \left(Y \cup Z\right)\right) \leftrightarrow$ Визначення перетину

$\leftrightarrow \left(x \in X\right) \wedge \left(x \in \left(\left(x \in Y\right) \vee \left(x \in Z\right)\right)\right) \leftrightarrow$ Визначення об'єднання

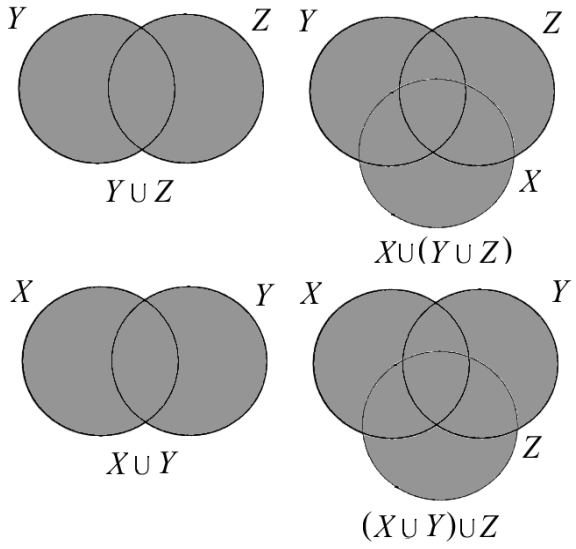
$\leftrightarrow \left(\left(x \in X\right) \wedge \left(x \in Y\right)\right) \vee \left(\left(x \in X\right) \wedge \left(x \in Z\right)\right) \leftrightarrow$ Закон логіки де Моргана

$\leftrightarrow \left(x \in \left(X \cap Y\right)\right) \vee \left(x \in \left(X \cap Z\right)\right) \leftrightarrow$ Визначення перетину

$\leftrightarrow x \in \left(\left(X \cap Y\right) \cup \left(X \cap Z\right)\right)$ Визначення об'єднання

Для доведення також використовують діаграми Венна (Ейлера).

Доведення властивості асоціативності за допомогою діаграм Венна показано на рис. 1.5.




Будуємо $\left(Y \cup Z\right)$ й потім $X \cup \left(Y \cup Z\right)$

Будуємо $\left(X \cup Y\right)$ й потім $\left(X \cup Y\right) \cup Z$

Рис 1.5. Графічне доведення властивості асоціативності

Для доведення тотожностей алгебри множин можна використовувати інші тотожності алгебри множин.

**Теорема 1.2.** Для множин $X$ і $Y$ справедлива тотожність (закон склеювання)

$$\left(X \cap Y\right) \cup \left(X \cap \bar{Y}\right) = X$$

*Доведення.* Доведемо цю тотожність двома способами: аналітично (використовуючи алгебру множин) і конструктивно (використовуючи діаграми Ейлера-Венна).

| | |
|---|---|
| $(X \cap Y) \cup \left(X \cap \bar{Y}\right) =$ | початковий вираз |
| $= \left(X \cup \left(X \cap \bar{Y}\right)\right) \cap \left(Y \cup \left(X \cap \bar{Y}\right)\right) =$ | застосували закон дистрибутивності відносно $\left(X \cap \bar{Y}\right)$ |
| $= \left(X \cup \left(X \cap \bar{Y}\right)\right) \cap \left(Y \cup X\right) =$ | застосували закон Порецького |
| $= X \cap \left(Y \cup X\right) =$ | застосували закон склеювання для об'єднання |
| $= X$ | застосували закон склеювання для перетину |

2. На рис. 1.6 показано діаграми Ейлера-Венна для доведення закону склеювання.

$X \cap Y$ $X \cap \bar{Y}$ $\left(X \cap Y\right) \cup \left(X \cap \bar{Y}\right) = X$

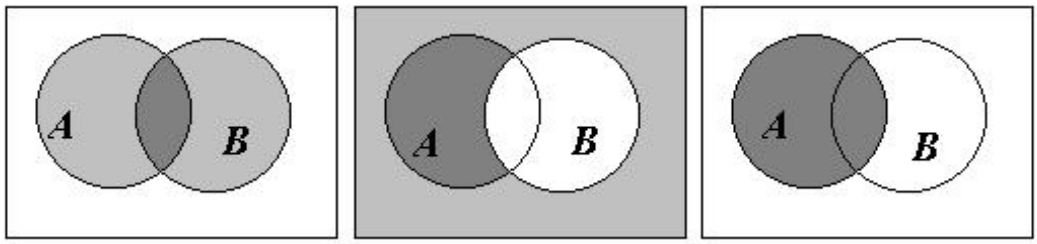



Рис. 1.6. Графічне доведення закону склеювання

**Приклад 1.18**. Доведемо тотожність:

$$A \setminus (B \cup C) = (A \setminus B) \setminus C$$

*Доведення.*

1 *спосіб*

1) Доведемо, що $A \setminus (B \cup C) \subseteq (A \setminus B) \setminus C$. Розглянемо довільний елемент множини $A \setminus (B \cup C)$:

$x \in A \setminus (B \cup C) \Rightarrow x \in A$ і $x \notin B \cup C \Rightarrow x \in A$ і $x \notin B$ і $x \notin C \Rightarrow x \in A \setminus B$ й $x \notin C \Rightarrow x \in (A \setminus B) \setminus C$.

2) Доведемо, що $(A \setminus B) \setminus C \subseteq A \setminus (B \cup C)$. Нехай $x \in (A \setminus B) \setminus C$:

$x \in (A \setminus B) \setminus C \Rightarrow x \in A \setminus B$ і $x \notin C \Rightarrow x \in A$ і $x \notin B$ і $x \notin C \Rightarrow x \in A$ й $x \notin B \cup C \Rightarrow$

$\Rightarrow x \in A \setminus (B \cup C)$.

2 *спосіб*

Перетворимо ліву частину тотожності в праву за допомогою властивостей операцій над множинами:

$$A \setminus (B \cup C) = A \cap \overline{(B \cup C)} = A \cap (\overline{B} \cap \overline{C}) = (A \cap \overline{B}) \cap \overline{C} = (A \setminus B) \cap \overline{C} = (A \setminus B) \setminus C$$

На рис. 1.7 зображені обидві частини тотожності за допомогою діаграм Ейлера-Венна:

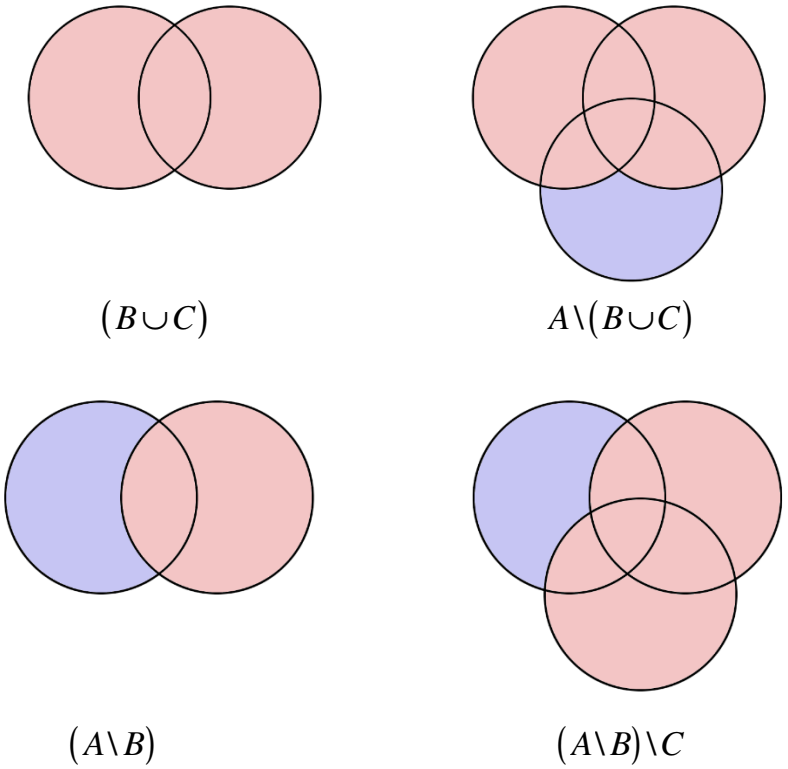



Рис. 1.7. Графічне доведення тотожності $A \setminus (B \cup C) = (A \setminus B) \setminus C$

### 1.1.5. Розбиття множин

Множина $X$ може бути розбита на класи множин $X_j$, які не перетинаються, якщо:

- об'єднання всіх підмножин $X_j$ збігається з множиною $X$: $X = \bigcup_{j \in J} X_j$
- перетин двох різних підмножин порожній, тобто для будь-яких двох $i \in J$ і $j \in J$ при $i \neq j$ виконується умова: $X_i \cap X_j = \varnothing$.

**Приклад 1.19.** Розбиття множини на підмножини.

1. Довільна множина $X$ може бути розбита на дві підмножини, які доповнюють одна одну $X_1$ і $X_2 = X \setminus X_1$. Для цих підмножин справедливі співвідношення: $X_1 \cup X_2 = X$ і $X_1 \cap X_2 = \varnothing$.

2. Множину двозначних чисел $X = \{10,11,12,...,98.99\}$ можна розбити на класи за ознакою остачі від ділення на 4:
клас, породжений остачею 0 – $X_0 = \{12,16,20,...,96\}$;
клас, породжений остачею 1 – $X_1 = \{13,17,21,...,97\}$;
клас, породжений остачею 2 – $X_2 = \{10,14,18,...,98\}$;
клас, породжений остачею 3 – $X_3 = \{11,15,19,...,99\}$.

### 1.1.6. Покриття множин

Покриттям множини $X$ називають сімейство множин $C = \{Y_j\}_{j \in J}$ таких, що їх об'єднання містить множину $X$:

$$X \subset \bigcup_{j \in J} Y_j$$

Якщо $C$ – покриття множини $X$, то будь-яку множину $D \subset C$, що також є покриттям множини $X$, називають **підпокриттям** множини $C$.

**Приклад 1.20.** Нехай $X = \{i \mid i = 2n+1, n = 0,1,2,....\}$. Побудувати покриття множини $X$.

*Розв'язок.*

$J = \{1,2\}, C = \{Y_1, Y_2\}, Y_1 = \{-k \mid k = 1,2,...\}$, $Y_2 = \{k \mid k = 0,1,2,...\}$.

Тоді $X \subset Y_1 \cup Y_2$, а, отже, сімейство множин $C$ є покриттям множини $X$.

### 1.1.7. Упорядкований набір або кортеж

**Упорядкованим** набором (або кортежем) називають таку сукупність елементів, у послідовності яких кожний елемент займає певне місце.

Самі елементи при цьому називають компонентами кортежу.

Приклади кортежів:

1) Множина людей, що стоять у черзі;

2) множина букв у слові;

3) числа, що виражають довготу й широту точки на місцевості;

4) координатна пара (або трійка) в аналітичній геометрії.

Число елементів кортежу називають його **довжиною.** Таким чином, корте́ж або *n-ка* (**упорядкована *n-ка***) – упорядкований скінченний набір елементів довжини *n* (де *n* – будь-яке натуральне число або 0). Кожний з елементів набору $x_i, 1 \le i \le n$ належить деякій множині $X$.

Для позначення впорядкованого набору (або кортежу) використовують круглі дужки (на відміну від фігурних дужок для позначення довільної множини):

$$X = (x_1, x_2, x_3, ..., x_n)$$

Відповідно до визначення, кортежі довжини 2 називають парами або впорядкованими парами, кортежі довжини 3 – трійками, 4 – четвірками і т. д.

Окремі випадки кортежу:

1) $\left(x_1\right)$ кортеж з одного елемента;

2) $\left(\ \right)$ порожній кортеж, тобто кортеж з кількістю елементів 0.

**На відміну** від довільної множини елементи кортежу можуть повторюватися.

Багато математичних об'єктів формально визначають як кортежі.

**Приклад спеціальних кортежів:**

1. Орієнтований граф визначають як кортеж (*V*, *E*), де *V* – це набір вершин, а *E* – підмножина $V \times V$, що позначає ребра.

2. Точку в *n*-вимірному просторі дійсних чисел визначають як кортеж довжини *n*, що складається з елементів множини дійсних чисел.

**Упорядкована пара** $\left(a,b\right)$ – часто вживаний математичний об'єкт. Основна її властивість – **єдиність**. Ця властивість виражена в наступному:

якщо $\left(a,b\right)$ та $\left(x,y\right)$ – упорядковані пари і стверджується, що $\left(a,b\right)=\left(x,y\right)$, то $a=x$ і $b=y$.

Будь-яку множину можна зробити впорядкованою, якщо, наприклад, переписати всі елементи множини в деякий список $\left(a,b,c,\ldots\right)$, а потім поставити у відповідність кожному елементу номер місця, на якому він розміщений у списку. Очевидно, що довільну множину, яка містить більше одного елемента, можна впорядкувати не єдиним способом. Упорядковані множини вважають різними, якщо вони відрізняються або своїми елементами, або їх порядком.

Різні впорядковані множини, що відрізняються лише порядком елементів (тобто можуть бути отримані з тієї ж самої множини), називають **перестановками** цієї множини.

Число різних способів, якими може бути впорядкована дана множина, дорівнює числу перестановок цієї множини. Число перестановок множини з $n$ елементів дорівнює $P_n = n!$

Наприклад. $X=\left\{a,b,c\right\}, n=3, P_3=3!=6.$

Перестановки мають вигляд: $\left(a,b,c\right),\left(a,c,b\right),\left(b,a,c\right),\left(b,c,a\right),\left(c,a,b\right),\left(c,b,a\right).$

### 1.1.8. Алгоритм упорядкування множини

Нехай є неупорядкована множина $A=\left\{a_1,a_2,a_3,\ldots,a_n\right\}$, елементами якої є цілі числа. У деяких програмах потрібно впорядкувати елементи множини $A$, наприклад, у порядку зростання значень цих чисел. Таку операцію називають *сортуванням*, а множину $A$ визначають *як масив.*

Одним з методів сортування масиву чисел є «Швидке сортування» (Quicksort).

У наведеному алгоритмі використані наступні позначення:

`a[k]` – масив чисел, у якому проводиться сортування.

Процедура дозволяє сортувати довільну підмножину масиву `a[k]`

`g` – номер мінімального елемента масиву, з якого починається сортування.

`r` – номер максимального елемента масиву, на якому закінчується сортування.

*Суть методу*

1. На кожній ітерації виділяють частину масиву чисел – робочий масив.

На першій ітерації – це весь масив чисел `a[k]`, у якому проводиться сортування. Встановлюємо границі робочого масиву $g = 1$ і $r = n$.

2. Вибирають елемент x:=a[(g+r)div2], який розміщений посередині робочого масиву.

3. Далі, починаючи з $i = 1$, послідовно збільшуємо значення $i$ на одиницю й порівнюємо кожний елемент $a_i$ з $x$, поки не буде знайдено елемент $a_i$ такий, що $a_i > x$.

4. Потім, починаючи з $j = r$, послідовно зменшуємо значення $j$ на одиницю, поки не буде знайдений елемент $a_j$ такий, що $x > a_j$.

5. Якщо для знайдених елементів $a_i$ і $a_j$ виконується умова $i \le j$, то ці елементи в робочому масиві міняються місцями.

6. Описана процедура закінчиться знаходженням головного елемента й поділом масиву на дві частини: одна частина буде містити елементи, менші за $x$, а інша – більші за $x$, як показано на рис. 1.8.

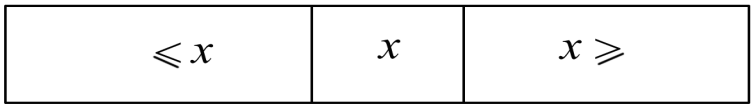



Рис. 1.8. Розподіл елементів масиву відносно головного елемента

Далі для кожної такої частини знову застосовується описана вище процедура.

Програма, що реалізує даний алгоритм для масиву $L$, показана двома варіантами реалізації функції сортування мовою програмування Python:

```
def qsort(L):
  if len(L)<2: return L
  k= len(L)//2
  pivot_element = L[k]
  small = [i for i in L if i< pivot_element]
  medium = [i for i in L if i==pivot_element]
  large = [i for i in L if i> pivot_element]
  return qsort(small) + medium + qsort(large)
L=[5,3,4,1,2]
qsort(L)
```

```
def qsort(L):
  small= filter(lambda x: x < L[0] ,L[1:])
  large= filter(lambda x: x >= L[0], L[1:])
  if L: return qsort(small) + L[0:1]+qsort(large)
  return []
L=[5,3,4,1,2]
qsort(L)
```

### 1.1.9. Декартовий добуток множин

*Визначення декартового добутку*

*Декартовым (прямим) добутком* множин $A$ і $B$ називають множину $C = A \times B$, що складається із усіх упорядкованих пар $\left(a, b\right)$ таких, що $a \in A$, $b \in B$, тобто

$$C = A \times B = \left\{ \left(a,b\right) \middle| a \in A, b \in B \right\}.$$

**Приклад 1.21.** Побудувати $C = A \times B$, якщо $A = \left\{x, y, z\right\}$, $B = \left\{1, 2\right\}$.

*Розв'язок.*

$$C = \left\{ \left(x,1\right), \left(x,2\right), \left(y,1\right), \left(y,2\right), \left(z,1\right), \left(z2\right) \right\}.$$

*Графічна інтерпретація декартового добутку*

Існує графічна інтерпретація прямого добутку множин. Нехай множина $A = \left\{ x \middle| a \le x \le b \right\}$ – це інтервал значень змінної $x$ і $B = \left\{ y \middle| c \le y \le d \right\}$ – це інтервал значень змінної $y$. Ясно, що множини $A$ і $B$ мають нескінченне число елементів. Тоді прямий декартовий добуток $A \times B$ – це множина точок прямокутника, зображеного на рис. 1.9.

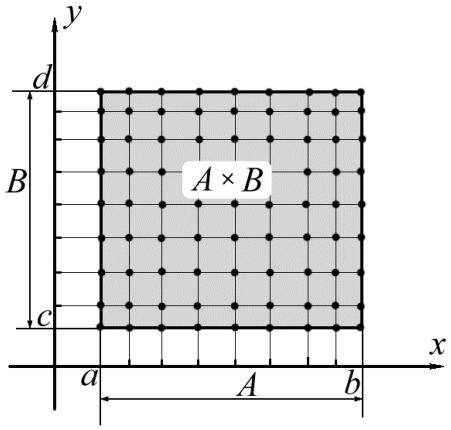



Рис. 1.9. Геометричне місце точок, що входять до декартовго добутку множин $A$ і $B$

Отже,

$$C = A \times B = \left\{ \left(x,y\right) \middle| x \in A, y \in B \right\}.$$

У випадку декартового добутку декількох множин використовуємо показник степеня:

$$A \times A = A^2, A \times A \times A = A^3, \underbrace{A \times A \times A \times \ldots \times A}_{n} = A^n.$$

Таким чином, $n = 2,3,\ldots$

Однак таке представлення добутку множини може бути розширене на $A^1 = A, A^0 = \{\Lambda\}$, де $\Lambda$ – проєкція кортежу на порожню множину осей, тобто порожній кортеж.

*Зворотний декартовий добуток*

Нехай $C = A \times B$ – прямий декартовий добуток множин.

Тоді $C^{-1} = B \times A$ буде називатися **зворотним** декартовим добутком до прямого добутку $C$.

**Приклад 1.22.** Побудувати прямий та зворотний декартові добутки для множин $A = \{1, 2, 3\}$ і $B = \{x, y, z\}$.

*Розв'язок.*

$$C = A \times B = \{(1,x),(1,y),(1,z),(2,x),(2,y),(2,z),(3,x),(3,y),(3,z)\}$$

$$C^{-1} = B \times A = \{(x,1),(y,1),(z,1),(x,2),(y,2),(z,2),(x,3),(y,3),(z,3)\}$$

### 1.1.10. Проєктування

У математиці прийнято позначати через $R$ множину дійсних чисел. Тоді $R^2 = R \times R$ є площина дійсних чисел, а $R^3 = R \times R \times R$ представляє тривимірний простір дійсних чисел.

На рис. 1.10 показані двовимірний та тривимірний простори дійсних чисел.

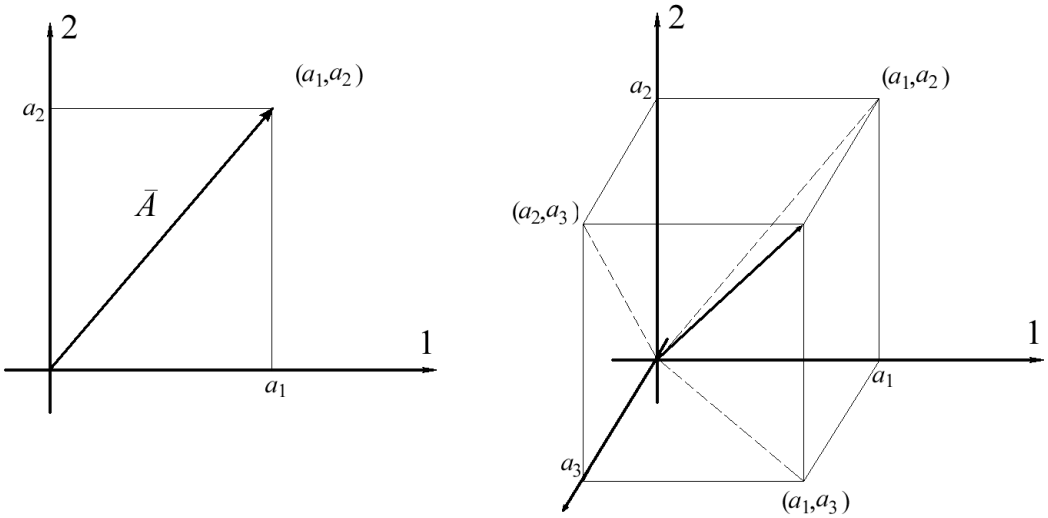



Рис. 1.10. Вектор $\overline{A}$ у двовимірному та тривимірному просторі

Кортеж $(a_1, a_2)$ – це точка на площині або вектор, проведений з початку координат у дану точку. Компоненти $a_1$ і $a_2$ – це **проєкції** вектора $\overline{A} = (a_1, a_2)$ на осі 1 і 2. Цей факт скорочено записують так:

$$proj_1 \overline{A} = proj_1 (a_1, a_2) = a_1,$$

$$proj_2\overline{A} = proj_2\left(a_1, a_2\right) = a_2.$$

Кортеж $\left(a_1, a_2, a_3\right)$ – це точка в тривимірному просторі або тривимірний вектор, проведений з початку координат у точку з координатами $\left(a_1, a_2, a_3\right)$

Проєкції вектора на осі координат у цьому випадку записують так:

$$proj_1\overline{A} = proj_1\left(a_1, a_2, a_3\right) = a_1,$$
$$proj_2\overline{A} = proj_2\left(a_1, a_2, a_3\right) = a_2,$$
$$proj_3\overline{A} = proj_3\left(a_1, a_2, a_3\right) = a_3.$$

Проєктувати можна не тільки на осі, але й на координатну площину. У цьому випадку проєкція є двоелементним кортежем:

$$proj_{1,2}\overline{A} = proj_{1,2}\left(a_1, a_2, a_3\right) = \left(a_1, a_2\right),$$
$$proj_{1,3}\overline{A} = proj_{1,3}\left(a_1, a_2, a_3\right) = \left(a_1, a_3\right),$$
$$proj_{2,3}\overline{A} = proj_{2,3}\left(a_1, a_2, a_3\right) = \left(a_2, a_3\right).$$

Узагальнюючи поняття проєкції на $n$-вимірний простір, можна $n$-елементну впорядковану множину $\left(a_1, a_2, a_3, ..., a_n\right)$ розглядати як точку в $n$-вимірному просторі. У цьому випадку

$$proj_i\overline{A} = proj_i\left(a_1, a_2, a_3, ..., a_i, ..., a_n\right) = a_i,$$

$$proj_{i,j}\overline{A} = proj_{i,j}\left(a_1, a_2, a_3, ..., a_i, ..., a_j, ..., a_n\right) = \left(a_i, a_j\right),$$

$$proj_{i,j,k}\overline{A} = proj_{i,j,k}\left(a_1, a_2, a_3, ..., a_i, ..., a_j, ..., a_k, ..., a_n\right) = \left(a_i, a_j, a_k\right).$$

Очевидно, що загальна кількість осей, на які припустиме проєктування, дорівнює $n-1$.

Нехай множина $D$ складається з кортежів довжини $m$. Тоді проєкцією множини $D$ називають множину проєкцій кортежів з $D$.

**Приклад 1.23.** Нехай дано множину кортежів

$$D = \left\{\left(1,2,3,4,5\right),\left(3,2,1,5,4\right),\left(2,3,6,7,1\right),\left(8,1,1,4,6\right)\right\}.$$

Побудувати проєкції множини $D$ на одну, дві та три осі.

*Розв'язок.* Проєктування кортежів на одну вісь:

$$proj_1D = \left\{\left(1\right),\left(3\right),\left(2\right),\left(8\right)\right\},\ \ proj_2D = \left\{\left(2\right),\left(2\right),\left(3\right),\left(1\right)\right\},$$

$$proj_3D = \left\{\left(3\right),\left(1\right),\left(6\right),\left(1\right)\right\},\ \ proj_4D = \left\{\left(4\right),\left(5\right),\left(7\right),\left(4\right)\right\},$$

$$proj_5D = \left\{\left(5\right),\left(4\right),\left(7\right),\left(6\right)\right\}.$$

Проєктування кортежів на дві осі:

$proj_{1,2}D = \{(1,2),(3,2),(2,3),(8,1)\}$,

$proj_{1,3}D = \{(1,3),(3,1),(2,6),(8,1)\}$,

……..

$proj_{2,3}D = \{(2,3),(2,1),(3,6),(1,1)\}$,

$proj_{1,3}D = \{(1,3),(3,1),(2,6),(8,1)\}$,

…….

Проєктування кортежів на три осі:

$proj_{1,2,3}D = \{(1,2,3),(3,2,1),(2,3,6),(8,1,1)\}$

……..

$proj_{3,4,5}D = \{(3,4,5),(1,5,4),(6,7,7),(1,4,6)\}$

……

Операція проєктування може застосовуватися тільки до тих множин, які містять кортежі однакової довжини.

**Контрольні запитання**

1. Нехай дано множину $A = \{a,\{b,c\},d\}$. Визначити, яке з тверджень є правильним:

a) $\{b,c\} \in A$, b) $\{b,c\} \subset A$, c) $\{a,d\} \subset A$ d) $a \in A$, e) $\{d\} \subset A$.

2. Записати булеан множини $A = \{\{1,2\},3,\{4,5,6\},7\}$. Визначити потужність створеного булеана.

3. Нехай дано множини $A = \{1,2,8,10\}$, $B = \{3,2,9,8\}, C = \{1,8,10,3\}$. Записати множини, які є результатами операцій об'єднання, перетину та різниці.

4. Доведіть тотожність $X \cup (X \cap Y) = (X \cup Y) \cap (X \cup \overline{Y})$.

5. Побудувати прямий та зворотний декартові добутки для множин $A = \{2,4,6,8\}$, $B = \{1,3,5,7\}$.

6. Нехай дано множину кортежів

$A = \{(1,3,5,7,9),(2,4,6,8,10),(a,b,c,d,e),(\alpha,\beta,\gamma,\delta,\lambda)\}$

Побудувати проєкції на осі 1 і 2, і на три осі.

7. Наведіть приклади розбиття множин.

## 1.2. Відповідності та відношення

### 1.2.1. Відповідність. Основні поняття

Розглянемо множини $X$ і $Y$. Елементи цих множин можуть зіставлятися один з одним яким-небудь чином, утворюючи впорядковані пари $\left(x, y\right)$.

Якщо спосіб такого зіставлення визначений, тобто для кожного елемента $x \in X$ вказано елемент $y \in Y$, з яким зіставляється елемент $x$, то говорять, що між множинами $X$ та $Y$ установлена відповідність.

Для того, щоб задати відповідність, необхідно вказати:

1) Множину $X$, елементи якої зіставляють з елементами іншої множини;

2) Множину $Y$, елементи якої ставлять у відповідність елементам першої множини;

3) Множину $Q \subseteq X \times Y$, що визначає закон (правило), за яким здійснюють відповідність, тобто таке правило, що перераховує всі пари $\left(x, y\right)$, які беруть участь у зіставленні.

Таким чином, відповідність (позначимо її через $q$) є трійкою множин

$$q = \left\langle X, Y, Q \right\rangle$$

де $Q \subseteq X \times Y$ – підмножина декартового добутку множин $X$ і $Y$, яку ще називають графіком відповідності; $X$ – множина відправлення відповідності; $Y$ – множина прибуття відповідності.

Крім того, з кожною відповідністю зв'язані нерозривно ще дві множини:

1. множина $proj_x Q$, яку називають **областю визначення** відповідності. В цю множину входять елементи множини $X$, що беруть участь у зіставленні;

2. множина $proj_y Q$, яку називають **областю значень** відповідності. В цю множину входять елементи множини $Y$, що беруть участь у зіставленні.

Якщо $\left(x, y\right) \in Q$, то говорять, що елемент $y$ відповідає елементу $x$. Геометрично це зображають у вигляді стрілки, спрямованої від $x$ до $y$:

На рис. 1.11 показано дві множини $X$ і $Y$ з установленими відповідностями між їх елементами. При цьому графік відповідності має вигляд:

$$Q = \left\{\left(x_1, y_6\right), \left(x_2, y_7\right), \left(x_3.y_1\right), \left(x_4, y_2\right), \left(x_5, y_3\right), \left(x_6, y_4\right), \left(x_7, y_5\right)\right\}$$

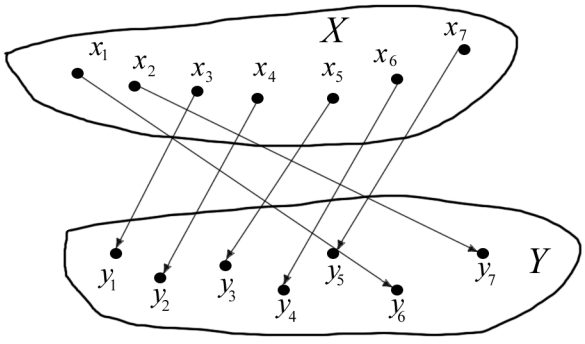



Рис. 1.11. Графічне представлення відповідності множин $X$ і $Y$

Для кожної відповідності $q = \langle X, Y, Q \rangle$, $Q \subseteq X \times Y$ існує зворотна відповідність, яка утворюється у випадку, коли дану відповідність розглядають у зворотному напрямку, тобто визначають елементи $x \in X$, з якими зіставляються елементи $y \in Y$.

Зворотна відповідність позначається: $q^{-1} = \langle X, Y, Q^{-1} \rangle$, де $Q^{-1} = Y \times X$.

Геометрично зворотну відповідність можна одержати шляхом зміни напрямку стрілок прямої відповідності.

### 1.2.2. Типи відповідностей

Відповідності можуть бути різних типів: одно-однозначні, одно-багатозначні, багато-однозначні, багато-багатозначні.

*Одно-однозначна (або взаємно-однозначна) відповідність*

**Одно-однозначна** (або **взаємно-однозначна)** відповідність – це така попарна відповідність між елементами двох множин $X$ і $Y$, коли один елемент з $X$ зіставлено з єдиним елементом з $Y$ і навпаки.

**Приклад 1.24.** Нехай існує множина натуральних чисел $N$ і множина квадратів натуральних чисел $P$. Побудувати одно-однозначну відповідність.
*Розв'язок.*

Кожному натуральному числу можна поставити у відповідність його квадрат і навпаки – кожному квадрату цілого числа відповідає саме натуральне число. Тому між множинами $N$ й $P$ існує взаємно-однозначна відповідність, як показано на рис. 1.12.

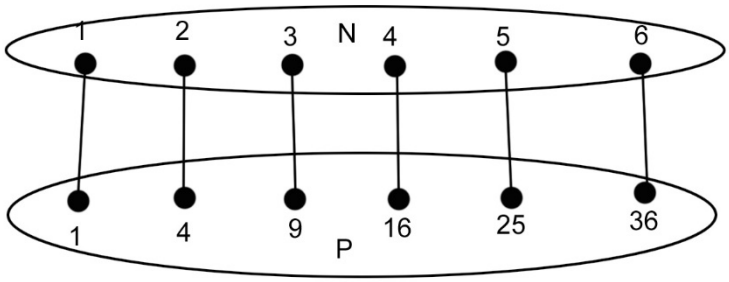



Рис. 1.12. Графічне представлення відповідності елементів множини натуральних чисел $N$ та множини квадратів цих чисел $P$.

*Одно-багатозначна відповідність*

**Одно-багатозначна** відповідність – це така відповідність між елементами двох множин $X$ і $Y$, коли з одним елементом першої множини $X$ зіставлено більше одного елемента другої множини $Y$, але кожний елемент другої множини відповідає тільки одному елементу першої множини.

**Приклад 1.25.** Нехай існує множина квадратних коренів $G = \{1,2,3,4,5\}$ і множина цілих чисел $N = \{1,2,3,4,5,6,7,8,9,10,11,12,13,15,16,\ldots,25\}$. Побудувати відповідність між елементами цих множин.

*Розв'язок.*

Кожному елементу множини $G$ однозначно відповідає один елемент множини $N$. Зворотна відповідність може бути багатозначною за умови, що ми будемо виділяти цілу частина від взяття квадратного кореня для кожного елемента множини $N$, як показано на рис. 1.13.

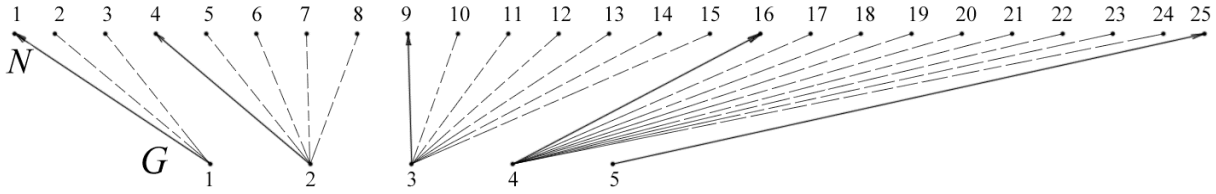



Рис. 1.13. Графічне представлення одно-багатозначної відповідності між елементами множини $N$ та $G$

*Багато-однозначна відповідність*

**Багато-однозначна** відповідність – це така відповідність між елементами двох множин X і Y, коли з елементом першої множини зіставлено тільки один елемент другої множини, але кожний елемент другої множини відповідає більше ніж одному елементу першої множини.

**Приклад 1.26.** Нехай $X = \{1,2,3,4,5\}$ – це припустима множина оцінок, а $Y = \{1,2,3,4,5,6,\ldots,25\}$ – множина студентів у групі. Побудувати відповідність між елементами цих множин.

*Розв'язок.*

Кожний студент під час здачі іспиту може одержати тільки одну оцінку. У той же час, та сама оцінка може бути поставлена деякій підмножині студентів. Тому між цими множинами можна побудувати багато-однозначну відповідність, як показано на рис. 1.14.

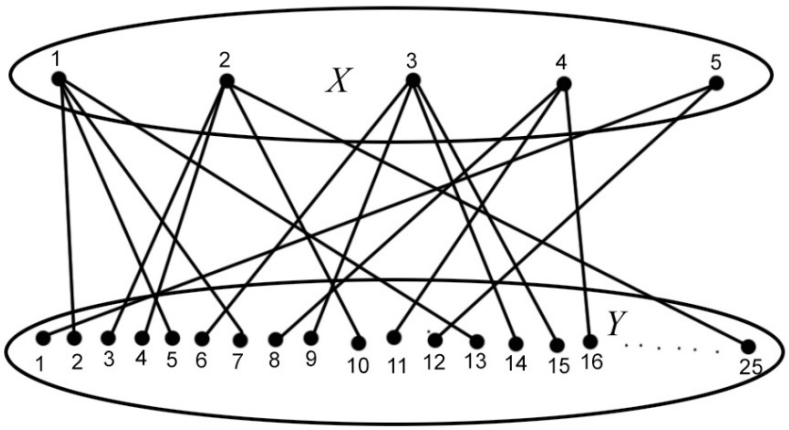



Рис. 1.14. Графічне представлення багато-однозначної відповідності між елементами множини $X$ та $Y$

*Багато-багатозначна відповідність*

**Багато-багатозначна** відповідність – це така відповідність між елементами двох множин $X$ і $Y$, коли з одним елементом першої множини зіставлено більш ніж один елемент другої множини і навпаки.

**Приклад 1.27.** Нехай $X$ – множина театральних постановок, а $Y$ – множина глядачів. Кожний глядач може подивитися деяку підмножину театральних постановок. У той же час, кожну з театральних постановок відвідує деяка підмножина глядачів. Графічне представлення такої відповідності показане на рис. 1.15.

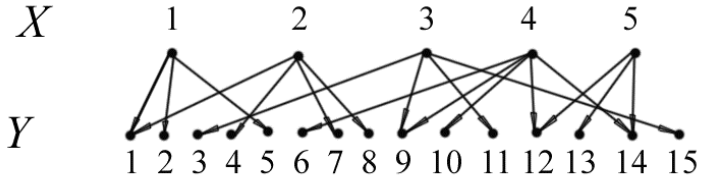



Рис. 1.15. Графічне представлення багато-багатозначної відповідності між елементами множини $X$ та $Y$

### 1.2.3. Поняття відношення

Відношення між парою об'єктів називають бінарним. Бінарне відношення використовують для того, щоб вказати характер виду зв'язку між парою об'єктів, які розглядають у певному порядку. При цьому відношення дає критерій для визначення відмінності одних упорядкованих пар від інших. Таким чином, поняття «відношення» є подальшим розвитком понять упорядкованої множини, «відповідності» і «відображення».

У математиці для позначення зв'язку між об'єктами або поняттями часто користуються терміном «відношення».

**Приклад 1.28.** Правило формування відношення може бути задане за допомогою предиката. Такі неповні речення (або так звані предикати, твердження) можуть розглядатися як відношення:

- $X$ менше (або більше), ніж $Y$,
- $X$ вище (або нижче), ніж $Y$,
- $X$ ділиться на $Y$,
- $X$ відбувається раніше (або пізніше), ніж $Y$,
- $X$ включено (або входить) в $Y$,
- $X$ паралельно (або перпендикулярно) до Y,
- $X$ дорівнює (або еквівалентне) $Y$,
- $X$ є братом $Y$,
- $X$ зв'язаний (електрично або у інший спосіб) з $Y$ и т. ін.

### 1.2.4. Визначення відношення

**Відношенням** $R$ множин $X$ і $Y$ називають довільну *підмножину* $X \times Y$ Якщо $(x, y) \in R$, то записують $xRy$; при цьому говорять, що $x$ і $y$ перебувають у відношенні $R$, або просто, що $x$ знаходиться у відношенні з $y$. Якщо $X = Y$, то відношення є підмножиною $X \times X$. Таке відношення називають ***бінарним відношенням*** на $X$.

**Приклад 1.29.** Навести приклади бінарних відношень.

*Розв'язок.*

1. Вся множина $X \times Y$ є відношенням множин $X$ і $Y$.
2. Якщо $X$ – множина дійсних чисел, то відношення

$$\left\{(a,b) \in X \times X \middle| a^2 + b^2 = 4\right\}$$ є бінарним відношенням на $X$.

3. Нехай $X$ – множина товарів у магазині, а $Y$ – множина дійсних чисел. Тоді $\left\{(a,b) \in X \times Y \middle| a \, price \, b\right\}$ – відношення множин $X$ і $Y$.
4. Нехай $X$ – множина жінок, а $Y$ – множина чоловіків, тоді $\{(a,b) | b$ є чоловіком $a\}$ є відношення множин $X$ і $Y$.
5. Якщо $A$ – множина людей, то відношення

$\{(a,b) \in A^2 | b$ є родичом $a\}$ є бінарним відношенням на $A$.

### 1.2.5. Область визначення й множина значень

***Область визначення*** відношення $R$ на $X$ і $Y$ **–** це множина всіх $x \in X$ таких, що для деяких $y \in Y$ маємо $(x, y) \in R$. Інакше кажучи, область визначення $R$ є множиною всіх перших координат упорядкованих пар з $R$.

***Множина значень*** відношення $R$ на $X$ і $Y$ – це множина всіх $y \in Y$ таких, що $(x, y) \in R$ для деяких $x \in X$. Інакше кажучи, множина значень $R$ є множиною всіх других координат упорядкованих пар з $R$.

З кожним відношенням $R$ на $X \times Y$ зв'язане відношення $R^{-1}$ на $Y \times X$.

### 1.2.6. Способи задавання бінарних відношень

*Спосіб* 1.

Бінарне відношення можна задати, перераховуючи всі пари, які в нього входять (якщо відношення складається зі скінченної кількості пар) або вказавши загальну властивість пар, що належать цьому відношенню, тобто предикатом (згадайте способи задавання множин).

**Приклад 1.30.** Нехай дана множина $X = \{p, r, s, q\}$. Задати відношення $R \subseteq X \times X$ перерахуванням пар.

*Розв'язок.* $R = \{(p,r),(s,q),(r,p),(p,p),(s,r),(p,s)\}$

**Приклад 1.31.** Нехай дано множину натуральних чисел $N$. Задати відношення, вказавши загальну властивість пар, що належать відношенню.

*Розв'язок.* $R_1 = \{(n,m) \in N \times N \mid n \text{ є дільником } m\}$

*Спосіб 2.*

Бінарне відношення може бути задане за допомогою графа. Нехай $R$ – бінарне відношення на множині $X$. Зобразимо елементи множини $X$ у вигляді точок на площині (їх називають вершинами графа). Для двох точок $x_i, x_j$ проводимо стрілку з $x_i$ у $x_j$ тоді й тільки тоді, коли $(x_i, x_j) \in R$. При цьому, якщо одночасно $(x_i, x_j) \in R$ та $(x_j, x_i) \in R$, то точки $x_i$ і $x_j$ з'єднують стрілкою $\longleftrightarrow$, а якщо $(x_j, x_j) \in R$, то в точці $x_j$ зображують петлю.

**Приклад 1.32.** Нехай існує відношення $R$, яке задано перерахуванням: $R = \{(p,r),(s,q),(r,p),(p,p),(s,r),(p,s)\}$. Представити дане відношення у вигляді графа.

*Розв'язок.*

На рис. 1.16 зображено граф бінарного відношення:

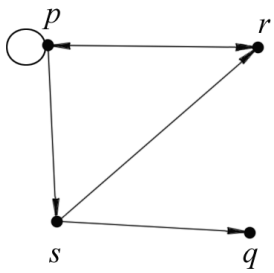



Рис. 1.16. Граф бінарного відношення $R$

*Спосіб* 3. Бінарне відношення може бути задане за допомогою булевих матриць.

Нехай $R \subseteq X \times Y$, де $X = \{x_1, x_2, x_3, ..., x_n\}$; $Y = \{y_1, y_2, y_3, ..., y_m\}$. Тоді відношення $R$ у вигляді матриці – це таблиця з $n$ рядками і $m$ стовпцями.

$$|X| = n \ , \ |Y| = m .$$

| | $y_1$ | $y_2$ | $\cdots$ | $y_m$ |
|---|---|---|---|---|
| $x_1$ | 1 | 0 | $\cdots$ | 1 |
| $x_2$ | 0 | 1 | $\cdots$ | 0 |
| $x_3$ | 1 | 0 | $\cdots$ | 1 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $x_n$ | 0 | 1 | $\cdots$ | 0 |

1. В перший стовпець виписані елементи множини $X$.
2. В перший рядок виписані елементи множини $Y$.
3. На перетині рядка елемента $x_i$ й стовпця елемента $y_j$ записують 1, якщо пари $(x_i, y_j) \in R$, і 0 – якщо $(x_i, y_j) \notin R$.

Таку таблицю називають **булевою матрицею відношення.**

**Приклад 1.33.** Нехай існує відношення $R$, яке задано перерахуванням: $R = \{(p,r),(s,q),(r,p),(p,p),(s,r),(p,s)\}$. Представити дане відношення у вигляді матриці.

*Розв'язок.* Булева матриця відношення $R = \{(p,r),(s,q),(r,p),(p,p),(s,r),(p,s)\}$ показана на рис. 1.17.

| $R$ | $p$ | $q$ | $r$ | $s$ |
|---|---|---|---|---|
| p | 1 | 0 | 1 | 1 |
| $q$ | 0 | 0 | 0 | 0 |
| $r$ | 1 | 0 | 0 | 0 |
| $s$ | 0 | 1 | 1 | 0 |

Рис. 1.17. Булева матриця відношення $R$

### 1.2.7. Зріз відношення через елемент

Нехай $R$ – довільне бінарне відношення між елементами множин $X$ і $Y$, $x \in X$. Множину тих елементів, з якими елемент $x$ перебуває у відношенні $R$, називають **зрізом** (або **перетином**) відношення $R$ через елемент $x$ і позначають $R(x)$. Якщо бінарне відношення R представлено за допомогою графа, то $R(x)$ складається з тих вершин, у які з вершини $x$ іде стрілка. Підкреслимо, що *зріз відношення* через елемент – це *деяка множина*, яка може містити кілька елементів, один елемент і жодного елемента (бути порожньою).

**Приклад 1.34.** Нехай дано множини $X = \{x_1, x_2, x_3, x_4\}$ та $Y = \{y_1, y_2, y_3, y_4, y_5, y_6\}$ і відношення $R \subset X \times Y$, яке задане графом на рис. 1.18:

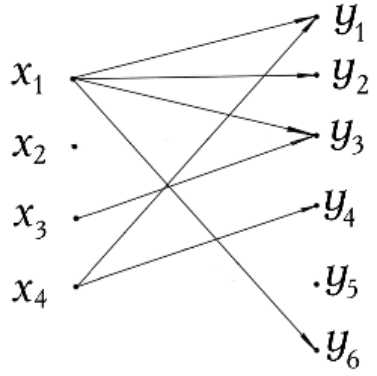



Рис. 1.18. Граф відношення $R$

Побудувати зрізи через кожний з елементів множини $X$

*Розв'язок.*

Зріз відношення $R$ через елемент $x_1$: $R(x_1) = \{y_1, y_2, y_3, y_6\}$

Зріз відношення $R$ через $x_2$: $R(x_2) = \{\varnothing\}$

Зріз відношення $R$ через $x_3$: $R(x_3) = y_3$

Зріз відношення $R$ через $x_4$: $R(x_4) = \{y_1, y_4\}$

### 1.2.8. Операції над відношеннями

Оскільки бінарні відношення представляють множини (пар), то до них застосовні поняття рівності, включення, а також операції об'єднання, перетину і доповнення.

**Включення** $R \subset S$ розуміють таким чином, що будь-яка впорядкована пара елементів, яка належить відношенню $R$, належить і відношенню $S$.

**Визначення.** Нехай дано множини $X = \{x_i\}_{i=1}^{n}$, $Y = \{y_i\}_{i=1}^{m}$ та відношення $R \subset X \times Y$, $S \subseteq X \times Y$. Тоді відношення $R$ строго включене у відношення $S$, $R \subset S$, якщо кожний елемент $(x_i, y_j)$, $i = \overline{1,n}$, $y_j = \overline{1,m}$, який належить $R$ одночасно належить відношенню $S$. Але не всі елементи $(x_i, y_j) \in S$ належать відношенню $R$.

**Приклад 1.35.** Дано множини $X = \{1,2,3\}$, $Y = \{a,b,c\}$ та існують відношення на даних множинах $R \subset X \times Y$, $S = X \times Y$, які задано перерахуванням:

$S = \{(1,a),(1,b),(1,c),(2,a),(2,b),(2,c),(3,a),(3,b),(3,c)\}$,

$R = \{(1,a),(2,a),(3,a)\}$. Визначити, чи входить відношення $R$ у відношення $S$

*Розв'язок.*

Розглянемо елементи відношення $R$.

Ми можемо стверджувати, що $R \subset S$, оскільки справедливі вирази, наведені в таблиці 1.2.

**Таблиця 1.2.**

Співвідношення елементів відношень $S$ та $R$

| | |
|---|---|
| $(1,a) \in R \ \wedge \ (1,a) \in S$, | $(1,b) \notin R \wedge (1,b) \in S$ |
| $(2,a) \in R \ \wedge \ (2,a) \in S$, | $(1,c) \notin R \wedge (1,c) \in S$ |
| $(3,a) \in R \ \wedge \ (3,a) \in S$ | …… |
| | $(3,c) \notin R \wedge (3,c) \in S$ |

**Рівність** $R = S$ означає, що відношення $R$ і $S$ складаються з тих самих упорядкованих пар.

**Визначення.** Нехай дано множини $X = \{x_i\}_{i=1}^{n}$, $Y = \{y_i\}_{i=1}^{m}$ та відношення: $R \subset X \times Y$, $S \subset X \times Y$. Тоді відношення $R$ дорівнює відношенню $S$, тобто $R = S$, якщо

$\forall (x_i, y_j) \to (x_i, y_j) \in R \wedge (x_i, y_j) \in S$, $|R| = |S|$

**Приклад 1.36.** Нехай дано множини $X = \{1,2,3\}$, $Y = \{a,b,c\}$, що утворюють відношення $R \subset X \times Y$, $S \subset X \times Y$, які задано перерахуванням елементів: $S = \{(1,a),(2,a),(3,a)\}$, $R = \{(1,a),(2,a),(3,a)\}$. Довести, що відношення $S$ дорівнює відношенню $R$.

*Доведення.*

$R = S$, оскільки за визначенням рівності $(1,a) \in R \ \wedge \ (1,a) \in S$, $(2,a) \in R \ \wedge \ (2,a) \in S$, $(3,a) \in R \ \wedge \ (3,a) \in S$, а також $|R| = |S|$.

**Приклад 1.37.** Нехай дано дві множини студентів

$$X = \{Іван(28\,років), Василь(29\,років), Петро(30\,років)\},$$
$$Y = \{Марія(21\,років), Оксана(19\,років), Світлана(20\,років)\}.$$

Множини $X$ і $Y$ утворюють відношення $|R| = |S|$, $R \subset X \times Y$, $S \subset X \times Y$, які задано перерахуванням пар:

$$S = \{(Іван, Марія),(Петро, Оксана),(Василь, Світлана)\},$$
$$R = \{(Іван, Марія),(Петро, Оксана),(Василь, Світлана)\},$$

які утворені предикатами:

$S = \{(a,b) | \texttt{вік a + вік b < 50} \ \}$

$R = \{(a,b) | a \ i \ b - \ \texttt{студенти}\}$

Чи дорівнює відношення $S$ відношенню $R$?

*Розв'язок.*

$R = S$, оскільки за визначенням рівності

$(Іван, Марія) \in R \ \wedge (Іван, Марія) \in S$,

$(Петро, Оксана) \in R \ \wedge (Петро, Оксана) \in S$,

$(Василь, Світлана) \in R \ \wedge \ (Василь, Світлана) \in S$,

а також $|R| = |S|$.

**Об'єднання** $R \cup S$ відношень $R$ і $S$ складається з упорядкованих пар, що належать хоча б одному із цих відношень.

**Визначення.** Нехай дано множини $X = \{x_i\}_{i=1}^{n}$, $Y = \{y_i\}_{i=1}^{m}$ та відношення: $R \subset X \times Y$, $S \subset X \times Y$. Тоді відношення $Z$ є об'єднанням відношень $R$ і $S$, тобто $Z = R \cup S$, якщо $\forall (x_i, y_j) \in Z \to (x_i, y_j) \in R \vee (x_i, y_j) \in S$.

**Приклад 1.38.** Нехай дано множини $X=\{1,2,3\}$, $Y=\{a,b,c\}$ та відношення $R\subset X\times Y$, $S\subset X\times Y$, які задані перерахуванням елементів:

$S=\{(1,a),(1,b),(1,c)\}$, $R=\{(1,a),(2,a),(3,a)\}$. Знайти об'єднання відношень.

*Розв'язок.*

В об'єднання входять всі унікальні елементи відношень та по одному екземпляру елементів, якщо вони входять в обидва відношення. Отже, отримуємо: $R\cup S=\{(1,a),(1,b),(1,c),(2,a),(3,a)\}$.

**Перетин** $R\cap S$ відношень $R$ і $S$ є новим відношення, що складається з упорядкованих пар, які належать одночасно обом відношенням.

**Визначення.** Нехай дано множини $X=\{x_i\}_{i=1}^{n}$, $Y=\{y_i\}_{i=1}^{m}$ та відношення: $R\subset X\times Y$, $S\subset X\times Y$. Тоді відношення $Z$ є перетином відношень $R$ і $S$, тобто $Z=R\cap S$, якщо $\forall(x_i,y_j)\in Z\to(x_i,y_j)\in R\wedge(x_i,y_j)\in S$.

**Приклад 1.39.** Нехай дано множини $X=\{1,2,3\}$, $Y=\{a,b,c\}$ та відношення $R\subset X\times Y$, $S\subset X\times Y$, які задані перерахуванням елементів

$S=\{(1,a),(1,b),(1,c)\}$, $R=\{(1,a),(2,a),(3,a)\}$.

*Розв'язок.*

В перетин входять тільки ті елементи, які присутні в обох відношеннях. Отже, отримуємо $R\cap S=\{(1,a)\}$

**Різниця** $R-S$ відношень $R$ і $S$ є множиною впорядкованих пар, що належать відношенню $R$ і не належать відношенню $S$.

**Визначення.** Нехай дано множини $X=\{x_i\}_{i=1}^{n}$, $Y=\{y_i\}_{i=1}^{m}$ та відношення: $R\subset X\times Y$, $S\subset X\times Y$. Тоді відношення $Z$ є різницею відношень $R$ і $S$, тобто $Z=R-S$, якщо $\forall(x_i,y_j)\in Z\to(x_i,y_j)\in R\wedge(x_i,y_j)\notin S$

**Приклад 1.40.** Нехай дано множини $X=\{1,2,3\}$, $Y=\{a,b,c\}$ та відношення $R\subset X\times Y$, $S\subset X\times Y$, які задані перерахуванням елементів

$S=\{(1,a),(1,b),(1,c)\}$, $R=\{(1,a),(2,a),(3,a)\}$.

*Розв'язок.*

В різницю входять тільки ті елементи, які входять у відношення $R$, але не входять в $S$. Отже, отримуємо $Z=R-S=\{(2,a),(3,a)\}$.

**Доповнення.** Якщо $R$ – бінарне відношення між елементами множин $X$ і $Y$, то його **доповненням** (відносно $X\times Y$) називають різницю $(X\times Y)-R$.

**Визначення.** Нехай дано множини $X=\{x_i\}_{i=1}^{n}$, $Y=\{y_i\}_{i=1}^{m}$ та відношення:

$R \subset X \times Y$. Тоді відношення $\overline{R}$ є доповненням відношення $R$, якщо $\forall\left(x_i, y_j\right) \in \overline{R} \to \left(x_i, y_j\right) \in \left(X \times Y\right) - R$.

**Приклад 1.41.** Нехай дано множини $X = \{1,2,3\}$, $Y = \{a,b,c\}$ та відношення $R \subset X \times Y$,
де $X \times Y = \{(1,a),(1,b),(1,c),(2,a),(2,b),(2,c),(3,a),(3,b),(3,c)\}$
Знайти доповнення $\overline{R}$.
*Розв'язок.* В доповнення входять ті пари елементів, що утворюють декартовий добуток множин $X$ та $Y$, але не належать відношенню $R$.
Отже, отримуємо $\overline{R} = \{(1,b),(1,c),(2,b),(2,c),(3,b),(3,c)\}$

*Операції об'єднання й перетину довільних сімейств відношень*

**Об'єднання довільних сімейств відношень.** Нехай $\{R_i\}_{i \in I}$ – сімейство відношень, яке утворює множину відношень, де $I$ – *додатне натуральне число*. Тоді відношення $Z$ є об'єднанням сімейства $\{R_i\}_{i \in I}$, якщо $Z = \bigcup_{i \in I} R_i$, тобто відношення $Z$ складається з упорядкованих пар, які належать хоча б одному з відношень $R_i$.

**Приклад 1.42.** Побудувати об'єднання сімейств відношень
$X = \{a,b,c\}, Y = \{k,l,m\}$,
$X \times Y = \{(a,k),(a,l),(a,m),(b,k),(b,l),(b,m),(c,k),(c,m),(c,l)\}$
$R_1 = \{(a,k),(c,l),(a,m)\}$, $R_2 = \{(a,m),(b,k),(c,k)\}$ $R_3 = \{(a,m),(c,k),(c,l)\}$.

*Розв'язок.* $Z = \bigcup_{i=1}^{3} R_i = \{(a,k),(c,l),(a,m),(b,k),(c,k)\}$

**Перетин довільних сімейств відношень.** Перетин сімейства $(R_i)_{i \in I}$ – це відношення $Z = \bigcap_{i \in I} R_i$, що складається з упорядкованих пар, які належать одночасно усім відношенням $R_i$.

**Приклад 1.43.** Побудувати перетин сімейств відношень
$X = \{a,b,c\}, Y = \{k,l,m\}$,
$X \times Y = \{(a,k),(a,l),(a,m),(b,k),(b,l),(b,m),(c,k),(c,m),(c,l)\}$
$R_1 = \{(a,k),(c,l),(a,m)\}$, $R_2 = \{(a,m),(b,k),(c,k)\}$ $R_3 = \{(a,m),(c,k),(c,l)\}$

*Розв'язок.* $Z = \bigcap_{i=1}^{3} R_i = \{(a,m)\}$

### 1.2.9. Додаткові операції

Для відношень задають деякі **додаткові операції**, які пов'язані з їх специфічною структурою, яка проявляється в тому, що **всі елементи відношень є упорядкованими парами**. Розглянемо дві такі операції.

*Обернене відношення*

Якщо в кожній упорядкованій парі, що належить відношенню *R*, поміняти місцями перший і другий компонент, то одержимо нове відношення, яке називають **оберненим до** відношення R і позначають через $R^{-1}$.

**Приклад 1.44.** Для відношення

$$R=\{(p,r),(s,q),(r,p),(p,p),(s,r),(p,s)\},$$

заданого перерахуванням, знайти обернене відношення $R^{-1}$.

*Розв'язок.* $R^{-1}=\{(r,p),(q,s),(p,r),(p,p),(r,s),(s,p)\}$.

Представлення $R^{-1}$ графом, матрицею та предикатом.

**Граф.** Граф відношення $R^{-1}$ одержують із графа відношення $R$ шляхом переорієнтації всіх стрілок.

**Матриця.** Відношення $R$, задане за допомогою булевої матриці, перетворюємо у відношення $R^{-1}$, міняючи місцями рядки і стовпці.(ТРАНСПОНУВАННЯ)

**Предикат.** Нехай $R\subseteq X\times Y$ є відношенням на $X\times Y$. Тоді відношення $R^{-1}$ на $Y\times X$ визначають у такий спосіб:

$$R^{-1}=\{(y,x)\,|\,(x,y)\in R\}.$$

Інакше кажучи, $(y,x)\in R^{-1}$ тоді й тільки тоді, коли $(x,y)\in R$ або, що рівнозначно, $yR^{-1}x$ тоді й тільки тоді, коли $xRy$.

Відношення $R^{-1}$ називають ***оберненим відношенням*** до даного відношення $R$.

**Приклад 1.45.** Приклади прямих та обернених відношень перерахуванням. Нехай $R=\{(1,r),(1,s),(3,s)\}$, тоді $R^{-1}=\{(r,1),(s,1),(s,3)\}$.

**Приклад 1.46.** Задавання прямих та обернених відношень предикатом.

Нехай $R=\{(a,b)\,|\,b\ є\ чоловіком\ a\}$, тоді $R^{-1}=\{(b,a)\,|\,a\ є\ дружиною\ b\}$

**Приклад 1.47.** Задавання прямих та обернених рефлексивних відношень

Нехай $R=\{(a,b)\,|\,b\ є\ родичем\ a\}$, тоді $R=R^{-1}$

Нехай $R$ – відношення $\{(a,b)\,|\,a^2+b^2=4\}$, тоді також $R^{-1}=R$.

**Теорема 2.1.** Теорема про двічі обернене відношення. Обернене відношення від оберненого відношення дорівнює прямому відношенню, тобто $\left(R^{-1}\right)^{-1} = R$.

*Доведення.*

Нехай існують дві множини: $X$ та $Y$. На декартовому добутку цих множин задано відношення $R \subset X \times Y$. Припустимо, що $\left(x, y\right) \in \left(R^{-1}\right)^{-1}$. Тоді відповідно до означення оберненого відношення $\left(y, x\right) \in R^{-1}$. Знову застосуємо означення оберненого відношення: $\left(x, y\right) \in R$.

Отже $\left(x, y\right) \in \left(R^{-1}\right)^{-1} \Leftrightarrow \left(x, y\right) \in R$

*Композиція відношень* (*Добуток відношень*)

Нехай $R \subseteq X \times Y$ – відношення на $X \times Y$, а $S \subseteq Y \times Z$ – відношення на $Y \times Z$. ***Композицією*** відношень $S$ і $R$ називають відношення $T \subseteq X \times Z$, визначене в такий спосіб:

$$T = \left\{\left(x, z\right) \middle| \textit{існує такий елемент } y \in Y, \left(x, y\right) \in R \textit{ і } \left(y, z\right) \in S\right\}$$

Цю множину позначають $T = S \circ R$.

**Приклад 1.48.** Нехай $X = \{1,2,3\}$, $Y = \{a,b\}$ і $Z = \{\alpha, \beta, \lambda, \mu\}$.

Також задані відношення $R = X \times Y$ та $S = Y \times Z$

$R = \{(1,a),(2,b),(3,b)\}$, $S = \{(a,\alpha),(a,\beta),(b,\lambda),(b,\mu)\}$

Побудувати композицію відношень.

*Розв'язок.*

$S \circ R = \{(1,\alpha),(1,\beta),(2,\lambda),(2,\mu),(3,\lambda),(3,\mu)\}$ оскільки

з $(1,a) \in R$ і $(a,\alpha) \in S$ випливає, що $(1,\alpha) \in S \circ R$,

з $(1,a) \in R$ і $(a,\beta) \in S$ випливає, що $(1,\beta) \in S \circ R$,

з $(3,b) \in R$ і $(b,\mu) \in S$ випливає, що $(3,\mu) \in S \circ R$.

**Властивості композиції відношень**

Композиція відношень **асоціативна**; тобто, якщо $X, Y, Z$, D – множини і якщо $R \subseteq X \times Y$, $S \subseteq Y \times Z$ і $T \subseteq Z \times D$, тоді $R \circ \left(S \circ T\right) = \left(R \circ S\right) \circ T$.

### 1.2.10. Спеціальні властивості відношень

*Рефлексивність*

**Визначення.** Відношення $R_{ref}$ на множині $X$ називають ***рефлексивним***, якщо для будь-якого $x \in X$ наявним є $xR_{ref}x$, тобто, кожний елемент $x \in X$ перебуває у відношенні $R_{ref}$ до самого себе.

**Приклад 1.49.** Нехай $R_1 \subset A \times A$. $A = \{1,2,3\}$. Відношення $R_1$ задано предикатом $R_1 = \{(a,b) | a \le b - \text{на множині натуральних чисел}\}$.

Визначити, чи є відношення $R_1$ рефлексивним.

*Розв'язок.*

Користуючись предикатом, побудуємо відношення $R_1$.

$$R_1 = \{(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)\}$$

В цьому відношенні присутні всі елементи типу $xR_1x$.

$$1R_1 1 \equiv (1,1) \in R_1,$$
$$2R_1 2 \equiv (2,2) \in R_1,$$
$$3R_1 3 \equiv (3,3) \in R_1.$$

Отже, відношення $R_1$ є рефлексивним.

**Приклад 1.50.** Нехай $R_2 \subseteq A \times A$. $A = \{1,2,3,4\}$. Відношення $R_2$ задано предикатом

$$R_2 = \{(a,b) | a \ i \ b - \text{мають спільний дільник на множині цілих чисел}\}.$$

Визначити, чи є відношення $R_2$ рефлексивним.

*Розв'язок.*

Користуючись предикатом, побудуємо відношення $R_2$.

$(1,1) \to 1,\ (1,2) \to 1,\ (1,3) \to 1\ (1,4) \to 1$

$(2,1) \to 1,\ (2,2) \to 2\,\mathbf{i}\,1,\ (2,3) \to 1,\ (2,4) \to 1\,\mathbf{i}\,2$

$(3,1) \to 1,\ (3,2) \to 1\ (3,3) \to 1\,\mathbf{i}\,3,\ (3,4) \to 1$

$(4,1) \to 1,\ (4,2) \to 2\,\mathbf{i}\,1,\ (4,3) \to 1,\ (4,4) \to 1\,\mathbf{i}\,4$

$$R_2 = \{(1,1),(1,2),(1,3),(1,4),(2,1),(2,2),(2,3),(2,4),$$
$$(3,1),(3,2),(3,3),(3,4),(4,1),(4,2),(4,3),(4,4)\}.$$

В цьому відношенні присутні всі елементи типу $xR_2x$.

$$1R_2 1 \equiv (1,1) \in R_2,$$
$$2R_2 2 \equiv (2,2) \in R_2,$$
$$3R_2 3 \equiv (3,3) \in R_2,$$
$$4R_2 4 \equiv (4,4) \in R_2.$$

Отже, відношення $R_2$ є рефлексивним.

**Визначення.** Для рефлексивного відношення всі діагональні елементи *матриці* дорівнюють 1.

**Приклад 1.51.** Розглянемо відношення $R_{ref} \subset A \times A$ на множині $A = \{a_1, a_2, a_3, a_4, a_5\}$, яке задане перерахуванням:

$R_{ref} = \{(a_1,a_1),(a_1,a_2),(a_2,a_1),(a_2,a_2),(a_3,a_3),(a_4,a_1),(a_4,a_2),(a_4,a_3),(a_4,a_4),(a_5,a_2),(a_5,a_3),(a_5,a_5)\}$

Побудувати матрицю відношення та визначити, чи є відношення $R$ рефлексивним.

*Розв'язок.*

Побудуємо таблицю відношення $R_{ref}$, позначаючи рядки першими елементами двійок, а стовпці – другими елементами.

**Таблиця 1.3.**

Таблиця відношення $R_{ref}$

| $R_{ref}$ | Другі елементи у двійках | | | | | |
|---|---|---|---|---|---|---|
| Перші елементи у двійках | | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
| | $a_1$ | 1 | 1 | | | |
| | $a_2$ | 1 | 1 | | | |
| | $a_3$ | | | 1 | | |
| | $a_4$ | 1 | 1 | 1 | 1 | |
| | $a_5$ | | 1 | 1 | | 1 |

На основі таблиці 1.3 запишемо матрицю відношення $R_{ref}$, зберігаючи порядок рядків і стовпців, як показано на рис. 1.19:

$$R_{ref} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Рис. 1.19. Матриця відношення $R_{ref}$

З рисунка 1.19 видно, що головна діагональ матриці повністю заповнена одиницями. Отже, за визначенням, дане відношення є рефлексивним.

**Визначення.** При задаванні відношення *графом* кожний елемент має петлю – дугу $(x, x)$.

**Приклад 1.52.** Розглянемо відношення $R_{ref} \subset A \times A$ на множині, задане перерахуванням: $A = \{a_1, a_2, a_3, a_4, a_5\}$

$R_{ref} = \{(a_1,a_1),(a_1,a_2),(a_2,a_1),(a_2,a_2),(a_3,a_3),(a_4,a_1),(a_4,a_2),(a_4,a_3),(a_4,a_4),(a_5,a_2),(a_5,a_3),(a_5,a_5)\}$

Побудувати граф відношення та визначити, чи є відношення $R_{ref}$ рефлексивним.

*Розв'язок.*

Для побудови графа відношення $R_{ref}$ елементи множини $A$ позначимо кружками, як вершини графа. Кожній двійці відношення $R_{ref}$ відповідає дуга, яка виходить з вершини, позначеної першим елементом двійки, і входить у вершину, позначену другим елементом двійки. Двійки з однаковими першим та другим елементом зображуються петлями, як показано на рис. 1.20.

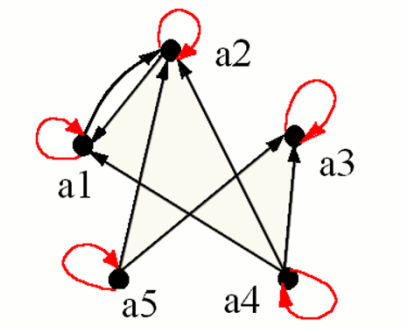



Рис. 1.20. Граф рефлексивного відношення $R_{ref}$

З рисунка 1.20 видно, що всі вершини графа мають петлі. Отже, за визначенням, дане відношення є рефлексивним.

*Антирефлексивність*

**Визначення.** Нехай задане відношення $R_{ar} \subseteq X \times X$. Відношення $R_{ar}$ на множині $X$ називають ***антирефлексивним***, якщо з $x_i R_{ar} x_j$ випливає, що $x_i \neq x_j$.

**Приклад 1.53.** Розглянемо відношення $R_1 \subset A \times A$ $A = \{1,2,3,4\}$. Елементи відношення задані предикатом $R_1 = \{(a,b) | a < b - \text{на множині цілих чисел}\}$.

Визначити, чи є відношення $R_1$ антирефлексивним.

*Розв'язок.*

Користуючись предикатом, побудуємо відношення $R_1$.

$R_1 = \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$

В цьому відношенні всі елементи типу $(x_i, x_i) \notin R_1, (x_j, x_j) \notin R_1$, тобто

$(1,1) \notin R_1, (2,2) \notin R_1, (3,3) \notin R_1, (4,4) \notin R_1$.

Для решти пар з $x_i R_1 x_j$ випливає, що $x_i \neq x_j$.

$$1R_1 2 \equiv (1,2) \in R_1 \rightarrow 1 \neq 2 \quad 2R_1 3 \equiv (2,3) \in R_1 \rightarrow 2 \neq 3$$

$$1R_1 3 \equiv (1,3) \in R_1 \to 1 \neq 3 \quad 2R_1 4 \equiv (2,4) \in R_1 \to 2 \neq 4$$
$$1R_1 4 \equiv (1,4) \in R_1 \to 1 \neq 4 \quad 3R_1 4 \equiv (3,4) \in R_1 \to 3 \neq 4$$

Отже, відношення $R_1$ є антирефлексивним.

**Приклад 1.54.** Розглянемо відношення $R_2 \subset A \times A$, задане на множині $A = \{\texttt{Іван}, \texttt{Марія}, \texttt{Петро}, \texttt{Оксана}, \texttt{Максим}, \texttt{Ольга}\}$ за допомогою предиката $R_2 = \{(a,b) \mid \texttt{a є сином b на множині людей.}\}$. Визначити, чи є це відношення антирефлексивним.

*Розв'язок.*

Граф відношення показано на рис. 1.21.

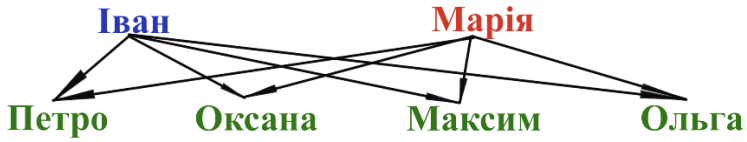



Рис. 1.21. Граф антирефлексивного відношення $R_2$

Відношення $R_2$, що задане перерахуванням елементів, має вигляд:

$$R_2 = \{(\texttt{Петро, Іван}), (\texttt{Петро, Марія}), (\texttt{Максим, Іван}), (\texttt{Максим, Марія})\}$$

Перевіримо елементи на відповідність визначенню антирефлексивності:

$(\texttt{Петро, Іван}) \in R_2 \to \texttt{Петро} \neq \texttt{Іван}$

$(\texttt{Максим, Іван}) \in R_2 \to \texttt{Максим} \neq \texttt{Іван}$

$(\texttt{Петро, Марія}) \in R_2 \to \texttt{Петро} \neq \texttt{Марія}$

$(\texttt{Максим, Марія}) \in R_2 \to \texttt{Максим} \neq \texttt{Марія}$

Отже, з $x_i R_2 x_j$ випливає, що $x_i \neq x_j$. Тому відношення $R_2$ антирефлексивне.

*Представлення антирефлексивного відношення булевою матрицею:*

**Визначення.** Для антирефлексивного відношення всі діагональні елементи *матриці* дорівнюють 0.

**Приклад 1.55.** Розглянемо відношення $R_{ar} \subset A \times A$ на множині $A = \{a_1, a_2, a_3, a_4, a_5\}$, задане перерахуванням елементів:

$$R_{ar} = \{(a_1, a_2), (a_2, a_1), (a_4, a_1), (a_4, a_2)(a_4, a_3), (a_5, a_2), (a_5, a_3)\}.$$

Побудувати матрицю відношення та визначити, чи є відношення $R_{ar}$ антирефлексивним.

*Розв'язок.*

Побудуємо таблицю відношення $R_{ar}$, позначаючи рядки першими елементами двійок, а стовпці – другими елементами.

**Таблиця 1.4.**

Таблиця відношення $R_{ar}$

| $R_{ar}$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
|---|---|---|---|---|---|
| $a_1$ | 0 | 1 | | | |
| $a_2$ | 1 | 0 | | | |
| $a_3$ | | | 0 | | |
| $a_4$ | 1 | 1 | 1 | 0 | |
| $a_5$ | | 1 | 1 | | 0 |

На основі таблиці 1.4 запишемо матрицю відношення $R_{ar}$, зберігаючи порядок рядків і стовпців, як показано на рис. 1.22:

$$R_{ar} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Рис. 1.22. Матриця відношення $R_{ar}$

З рис 1.22 видно, що на головній діагоналі матриці немає жодної одиниці. Отже, за визначенням, дане відношення є антирефлексивним.

*Представлення антирефлексивного відношення графом:*

**Визначення.** При задаванні відношення *графом* жодна з **вершина не має петлі** – немає дуг виду $(x_i, x_i)$.

**Приклад 1.56**. Розглянемо відношення $R_{ar} \subset A \times A$ на множині $A = \{a_1, a_2, a_3, a_4, a_5\}$, задане перерахуванням елементів:

$R_{ar} = \{(a_1, a_2), (a_2, a_1), (a_4, a_1), (a_4, a_2)(a_4, a_3), (a_5, a_2), (a_5, a_3)\}$.

Побудувати граф відношення та визначити, чи є відношення $R_{ar}$ антирефлексивним.

*Розв'язок.*

Для побудови графа відношення $R_{ar}$ елементи множини $A$ позначимо кружками, як вершини графа. Кожній двійці відношення $R_{ar}$ відповідає дуга, яка виходить з вершини, позначеної першим елементом двійки, і входить у вершину, позначену другим елементом двійки, як показано на рис. 1.23.

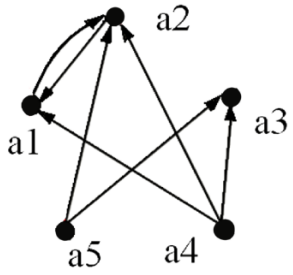



Рис. 1.23. Граф антирефлексивного відношення $R_{ar}$

З рисунка 1.23 видно, що жодна з вершин графа не має петлі. Отже, за визначенням, дане відношення є антирефлексивним.

*Симетричність*

**Визначення.** Нехай задане відношення $R_{sym} \subseteq X \times X$. Відношення $R_{sym}$ на множині $X = \{x_1, ..., x_i, ..., x_j, ...\}$ називається симетричним, якщо для пари $(x_i, x_j) \in R_{sym}$ з $x_i R_{sym} x_j$ випливає $x_j R_{sym} x_i$ (інакше кажучи, для будь-якої пари відношення $R_{sym}$ виконується або в обидва боки, або не виконується взагалі).

**Приклад 1.57.** Розглянемо відношення $R_1 \subset A \times A$ на множині $A = \{1,2,3,4\}$ задане предикатом $R_1 = \{(a,b) | a \neq b - \text{на множині цілих чисел}\}$. Визначити, чи є це відношення симетричним.

*Розв'язок.*

$R_1 = \{(1,2),(2,1),(1,3),(3,1),(1,4),(4,1),(2,3),(3,2),(4,2),(2,4),(3,4)(4,3)\}$

За визначенням симетричності

$$(1,2) \in R_1 \to (2,1) \in R_1 \quad (2,3) \in R_1 \to (3,2) \in R_1$$
$$(1,3) \in R_1 \to (3,1) \in R_1 \quad (2,4) \in R_1 \to (4,2) \in R_1$$
$$(1,4) \in R_1 \to (4,1) \in R_1 \quad (3,4) \in R_1 \to (4,3) \in R_1$$

**Приклад 1.58.** Розглянемо відношення $R_2 \subset A \times A$ на множині $A = \{\texttt{Іван}, \texttt{Марія}, \texttt{Петро}, \texttt{Оксана}, \texttt{ Максим}, \texttt{Ольга}\}$, задане предикатом $R_2 = \{(a,b) | a \text{ є родичем } b\}$.

*Розв'язок.*

Застосуємо визначення симетричності:

$(\texttt{Петро}, \texttt{Іван}) \in R_2 \to (\texttt{Іван}, \texttt{Петро}) \in R_2$

$(\texttt{Максим}, \texttt{Іван}) \in R_2 \to (\texttt{Іван}, \texttt{Максим}) \in R_2$

$(\texttt{Петро}, \texttt{Марія}) \in R_2 \to (\texttt{Марія}, \texttt{Петро}) \in R_2$

$(\texttt{Максим}, \texttt{Марія}) \in R_2 \to (\texttt{Марія}, \texttt{Максим}) \in R_2$

$(\texttt{Петро}, \texttt{Ольга}) \in R_2 \rightarrow (\texttt{Ольга}, \texttt{Петро}) \in R_2$

$(\texttt{Максим}, \texttt{Оксана}) \in R_2 \rightarrow (\texttt{Оксана}, \texttt{Максим}) \in R_2$

**Визначення.** Матриця симетричного відношення є симетричною відносно головної діагоналі.

**Приклад 1.59.** Розглянемо відношення $R_{sym} \subset A \times A$ на множині $A = \{a_1, a_2, a_3, a_4, a_5\}$, яке задане перерахуванням елементів:

$R_{sym} = \{(a_1, a_2), (a_2, a_1), (a_1, a_4), (a_4, a_1)(a_3, a_4), (a_4, a_3), (a_3, a_5), (a_5, a_3)\}$.

Побудувати матрицю відношення та визначити, чи є відношення $R_{sym}$ симетричним.

*Розв'язок.*

Побудуємо таблицю відношення $R_{sym}$, позначаючи рядки першими елементами двійок, а стовпці – другими елементами.

**Таблиця 1.5.**

Таблиця відношення $R_{sym}$

| $R_{sym}$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
|---|---|---|---|---|---|
| $a_1$ | x | 1 | | 1 | |
| $a_2$ | 1 | x | | | |
| $a_3$ | | | x | 1 | 1 |
| $a_4$ | 1 | 0 | 1 | x | |
| $a_5$ | | 0 | 1 | | x |

На основі таблиці 1.5 запишемо матрицю відношення $R_{sym}$, зберігаючи порядок рядків і стовпців, як показано на рис. 1.24.

$$R_{sym} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Рис. 1.24. Матриця відношення $R_{sym}$

З рис 1.24 видно, що матриця є симетричною відносно головної діагоналі. Отже, за визначенням, дане відношення є симетричним.

**Визначення.** У графі для кожної дуги з $x_i$ в $x_j$ існує протилежно спрямована дуга з $x_j$ в $x_i$.

**Приклад 1.60.** Розглянемо відношення $R_{sym} \subset A \times A$ на множині $A = (1, 2, 3, 4, 5)$, задане перерахуванням

$$R_{sym}=\{(a_1,a_4),(a_2,a_2),(a_2,a_3),(a_2,a_5),(a_3,a_5),(a_3,a_2),(a_4,a_4),(a_4,a_1),(a_5,a_2),(a_5,a_3)\}$$

Побудувати граф відношення $R_{sym}$ та визначити, чи є дане відношення симетричним.

*Розв'язок.*

Для побудови графа відношення $R_{sym}$ елементи множини $A$ позначимо кружками, як вершини графа. Кожній двійці відношення $R_{sym}$ відповідає дуга, яка виходить з вершини, позначеної першим елементом двійки, і входить у вершину, що позначена другим елементом двійки, як показано на рис. 1.25.

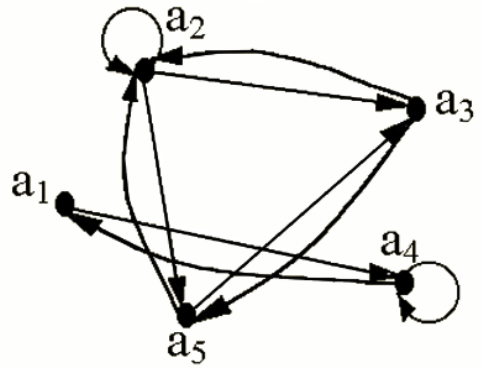



Рис. 1.25. Граф симетричного відношення $R_{sym}$

З рисунка 1.25 видно, що кожні дві суміжні вершини зв'язані двома протилежними дугами. Отже, за визначенням, відношення $R_{sym}$ є симетричним.

*Антисиметричність*

**Визначення.** Відношення $R_{ans}$ на множині $X=\{x_1,\dots,x_i,\dots,x_j,\dots\}$ називається **антисиметричним**, якщо з існування $x_iR_{ans}x_j$ і $x_jR_{ans}x_i$ випливає, що $x_i=x_j$ .

**Приклад 1.61.** Розглянемо відношення $R_1\subset A\times A$ на множині $A=\{1,2,3\}$, задане предикатом $R_1=\{(a,b)\,|\,a\le b-\text{на множині } A\}$. Визначити, чи є відношення $R_1$ антисиметричним.

*Розв'язок.* Використовуючи предикат, запишемо відношення $R_1$ перерахуванням елементів.

$$R_1=\{(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)\}$$

У цьому відношенні з $aR_1b$ і $bR_1a$ випливає, що $a=b$.

$$(1,1)\in R_1\to(1,1)\in R_1,\qquad (1,2)\in R_1\to(2,1)\notin R_1$$
$$(2,2)\in R_1\to(2,2)\in R_1,\qquad (1,3)\in R_1\to(3,1)\notin R_1$$
$$(3,3)\in R_1\to(3,3)\in R_1,\qquad (2,3)\in R_1\to(3,2)\notin R_1$$

Отже, за визначенням, відношення $R_1$ є антисиметричним.

**Приклад 1.62.** Розглянемо відношення $R_2 \subset A \times A$ на множині $A = \{1,2,3,4\}$, задане предикатом

$$R_2 = \{(a,b) \mid a \text{ є дільником } b \text{ на множині } A\}.$$

Визначити, чи є відношення $R_2$ антисиметричним.

*Розв'язок.* Визначимо дільники для кожної двійки $(a,b)$ у відношенні $R_2$.

$(1,1) \to 1,\ (1,2) \to 1,\ (1,3) \to 1\ (1,4) \to 1,$
$(2,2) \to 2,\ \ (2,4) \to 2,$
$(3,3) \to 3, (4,4) \to 4.$

В результаті сформуємо елементи відношення $R_2$:

$$R_2 = \{(1,1),(1,2),(1,3),(1,4),(2,2),(2,4),(3,3),(4,4)\}.$$

У цьому відношенні з $aR_1b$ і $bR_1a$ випливає, що $a = b$.

$$(1,1) \in R_2 \to (1,1) \in R_2, \qquad (1,2) \in R_2 \to (2,1) \notin R_2$$
$$(2,2) \in R_2 \to (2,2) \in R_2, \qquad (1,3) \in R_2 \to (3,1) \notin R_2$$
$$(3,3) \in R_2 \to (3,3) \in R_2, \qquad (2,4) \in R_2 \to (4,2) \notin R_2$$
$$(4,4) \in R_2 \to (4,4) \in R_2$$

Отже, за визначенням, відношення $R_2$ є антисиметричним.

**Визначення.** Матриця антисиметричного відношення – це матриця, яка характеризується такими властивостями:

1. Матриця може мати одиниці на головній діагоналі.
2. Повністю відсутня симетрія відносно головної діагоналі.

**Приклад 1.63.** Розглянемо відношення $R_{ans} \subset A \times A$ на множині $A = \{a_1, a_2, a_3, a_4, a_5\}$, задане перерахуванням елементів:

$R_{ans} = \{(a_1,a_1),(a_2,a_1),(a_4,a_4)(a_4,a_1)(a_3,a_4),(a_3,a_5)\}$

Побудувати матрицю відношення $R_{ans}$ та визначити, чи є дане відношення антисиметричним.

*Розв'язок.*

Побудуємо таблицю відношення $R_{ans}$, позначаючи рядки першими елементами двійок, а стовпці – другими елементами.

**Таблиця 1.6.**

Таблиця відношення $R_{ans}$

| $R_{ans}$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
|---|---|---|---|---|---|
| $a_1$ | 1 | | | | |
| $a_2$ | 1 | x | | | |
| $a_3$ | | | x | 1 | 1 |
| $a_4$ | 1 | | | 1 | |
| $a_5$ | | | | | x |

Використовуючи таблицю 1.6, створимо матрицю відношення $R_{ans}$, зберігаючи порядок рядків і стовпців, як показано на рис. 1.26.

$$R_{ans} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Рис. 1.26. Матриця відношення $R_{ans}$

З рис 1.26 видно, що матриця несиметрична відносно головної діагоналі. На головній діагоналі знаходяться нулі і одиниці. Отже, за визначенням матриці, дане відношення є антисиметричним.

**Визначення.** Граф антисиметричного відношення – це граф, у якому для кожної дуги з $x_i$ в $x_j$ не існує протилежно спрямованої дуги з $x_j$ в $x_i$ при $x_i \neq x_j$. Вершини графа можуть мати петлі.

**Приклад 1.64.** Розглянемо відношення $R_{ans} \subset A \times A$ на множині $A = \{a_1, a_2, a_3, a_4, a_5\}$, задане перерахуванням елементів:

$$R_{ans} = \{(a_1, a_1), (a_2, a_1), (a_4, a_4)(a_4, a_1)(a_3, a_4), (a_3, a_5)\}$$

Побудувати граф відношення $R_{ans}$ та визначити, чи є дане відношення антисиметричним.

*Розв'язок.*

Для побудови графа відношення $R_{ans}$ елементи множини $A$ позначимо кружками, як вершини графа. Кожній двійці відношення $R_{ans}$ відповідає дуга, яка виходить з вершини, позначеної першим елементом двійки, і входить у вершину, позначену другим елементом двійки, як показано на рис. 1.27.

З рисунка 1.27 видно, що для жодної дуги у графі не існує протилежної дуги. Дві вершини з п'яти мають петлі. Отже, за визначенням графа, відношення $R_{ans}$ є антисиметричним.

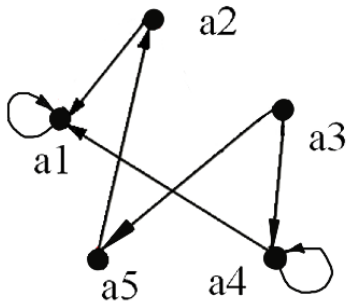



Рис. 1.27. Граф антисиметричного відношення $R_{ans}$

*Асиметричність*

**Визначення.** Відношення $R_{as}$ на множині $X = \{x_1,...,x_i,...,x_j,...\}$ називають ***асиметричним***, якщо для пари $(x_i, x_j) \in R_{as}$ з того, що $x_i R_{as} x_j$, випливає, що не виконується $x_j R_{as} x_i$ (інакше кажучи, для будь-якої пари відношення $R_{as}$ виконується або в одну сторону, або не виконується взагалі).

**Приклад 1.65.** Розглянемо відношення $R_1 \subset A \times A$ на множині $A = \{1,2,3,4\}$, яке задане предикатом

$R_1 = \{(a,b) \mid a > b - \text{на множині цілих чисел}\}$.

Визначити, чи є відношення $R_1$ асиметричним.

*Розв'язок.* Побудуємо за предикатом елементи відношення $R_1$:

$R_1 = \{(2,1),(3,1),(3,2),(4,1),(4,2)\}$

Перевіримо кожну двійку на властивість асиметричності

$(2,1) \in R_1 \rightarrow (1,2) \notin R_1$
$(3,1) \in R_1 \rightarrow (1,3) \notin R_1$
$(3,2) \in R_1 \rightarrow (2,3) \notin R_1$
$(4,1) \in R_1 \rightarrow (1,4) \notin R_1$
$(4,2) \in R_1 \rightarrow (2,4) \notin R_1$

$(1,1) \notin R_1, (2,2) \notin R_1, (3,3) \notin R_1, (4,4) \notin R_1$

Оскільки для всіх елементів відношення $R_1$ справджується властивість асиметричності, то дане відношення є асиметричним.

**Приклад 1.66.** Розглянемо відношення $R_2$ на множині $A = \{\texttt{Іван}, \texttt{Марія}, \texttt{Петро}, \texttt{Оксана}, \texttt{ Максим}, \texttt{Ольга}\}$, задане предикатом

$R_2 = \{(a,b) \mid a \text{ є сином } b \quad \text{на множині } A\}$.

Визначити, чи є відношення $R_2$ асиметричним.

*Розв'язок.*

Представимо відношення $R_2$ графом, як показано на рис. 1.28.

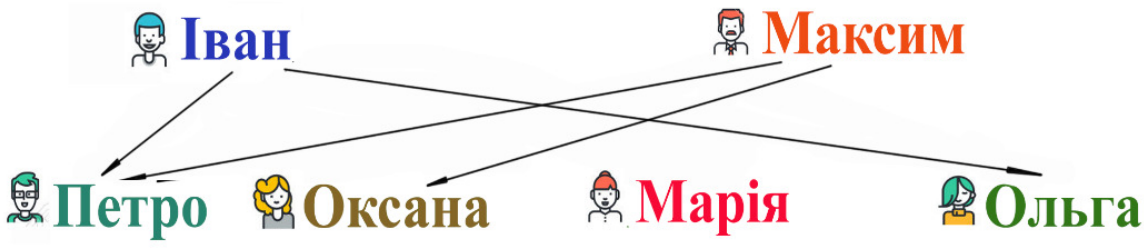



Рис. 1.28. Граф відношення $R_2$

Перевіримо властивість асиметричності для кожної пари суміжних вершин графа:

$(\texttt{Іван,Петро}) \in R_2 \to (\texttt{Петро,Іван}) \notin R_2$

$(\texttt{Іван, Ольга}) \in R_2 \to (\texttt{Ольга,Іван}) \notin R_2$

$(\texttt{Максим,Петро}) \in R_2 \to (\texttt{Петро,Максим}) \notin R_2$

$(\texttt{Максим,Оксана}) \in R_2 \to (\texttt{Оксана,Максим}) \notin R_2$

Оскільки властивість асиметричності виконується для кожної пари суміжних вершин, які відповідають двійкам відношення $R_2$, то дане відношення є асиметричним.

**Визначення.** Матриця асиметричного відношення – це матриця, яка характеризується такими властивостями:

1. Матриця не містить одиничних елементів, симетричних відносно головної діагоналі.
2. Відсутня симетрія.

**Приклад 1.67.** Розглянемо відношення $R_{as} \subset A \times A$ на множині $A = \{a_1, a_2, a_3, a_4, a_5\}$, задане перерахуванням елементів:

$R_{as} = \{(a_2, a_1)(a_4, a_1)(a_3, a_4), (a_3, a_5), (a_2, a_5)\}$.

Побудувати матрицю відношення $R_{as}$ та визначити, чи є дане відношення асиметричним.

*Розв'язок.*

Побудуємо таблицю відношення $R_{as}$, позначаючи рядки першими елементами двійок, а стовпці – другими елементами.

**Таблиця 1.7.**

Таблиця відношення $R_{as}$

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
|---|---|---|---|---|---|
| $a_1$ | x | | | | |
| $a_2$ | 1 | x | | | 1 |
| $a_3$ | | | x | 1 | 1 |
| $a_4$ | 1 | | | x | |
| $a_5$ | | | | | x |

Використовуючи таблицю 1.7, створимо матрицю відношення $R_{as}$, зберігаючи порядок рядків і стовпців, як показано на рис. 1.29.

$$R_{as} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Рис. 1.29. Матриця відношення $R_{as}$

**Визначення.** Граф асиметричного відношення – це граф, у якому для кожної дуги з $x_i$ в $x_j$ не існує протилежно спрямованої дуги з $x_j$ в $x_i$.

**Приклад 1.68.** Розглянемо відношення $R_{as} \subset A \times A$ на множині $A = \{a_1, a_2, a_3, a_4, a_5\}$, задане перерахуванням елементів $R_{as} = \{(a_2, a_1), (a_4, a_1)(a_3, a_4), (a_3, a_5), (a_5, a_2)\}$. Побудувати граф відношення $R_{as}$ та визначити, чи є дане відношення асиметричним.

*Розв'язок.*

Побудуємо граф відношення $R_{as}$, позначивши вершини графа елементами множини $A$. Кожна двійка відношення $R_{as}$ представлена дугою, яка виходить з вершини, позначеної першим елементом двійки, і входить у вершину, що позначена другим елементом двійки, як показано на рис. 1.30.

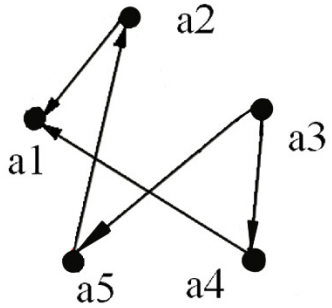



Рис. 1.30. Граф орієнтований, без петель

З рисунка 1.30 видно, що для жодної дуги у графі не існує протилежної дуги. Жодна з вершин графа не має петель. Отже, за визначенням графа, відношення $R_{as}$ є асиметричним.

*Транзитивність*

**Визначення.** Відношення $R_{tr} \subseteq X \times X$ на множині $X = \{x_1, \ldots, x_i, \ldots, x_j, \ldots, x_k, \ldots\}$ називають ***транзитивним***, якщо для будь-яких $x_i, x_j, x_k$ з $x_i R_{tr} x_j$ і $x_j R_{tr} x_k$ випливає $x_i R_{tr} x_k$.

**Приклад 1.69.** Розглянемо відношення $R_1 \subset A \times A$ на множині $X = \{1,2,3,4\}$ задане предикатом $R_1 = \{(a,b) | a \le b - \text{на множині } X\}$. Визначити, чи є відношення $R_1$ транзитивним.

*Розв'язок.*

Користуючись предикатом, сформуємо відношення $R_1$, задане перерахуванням елементів:

$$R_1 = \{(2,3),(1,1),(2,4),(1,2),(2,2),(1,3),(1,4),(3,4),(3,3),(4,4)\}$$

Перевіримо властивості відношення $R_1$, використовуючи визначення транзитивності.

$$\{1,1,2\} - (1,1),(1,2) \to (1,2),$$
$$(1,2,3) - (1,2),(2,3) \to (1,3)$$
$$(1,3,4) - (1,3),(3,4) \to (1,4)$$
$$(3,4,4) - (3,4)(4,4) \to (3,4)$$

Звідси можна зробити висновок, що відношення $R_1$ є транзитивним.

**Приклад 1.70.** Розглянемо відношення $R_2 \subseteq A \times A$ на множині $A = \{\texttt{Іван}, \texttt{Петро}, \texttt{Василь}, \texttt{Максим}\}$, задане предикатом $R_2 = \{(a,b) | a \text{ нащадок } b \text{ на множині } A\}$. Визначити, чи є відношення $R_2$ транзитивним.

*Розв'язок.*

Представимо відношення $R_2$ графом (рис. 1.31):

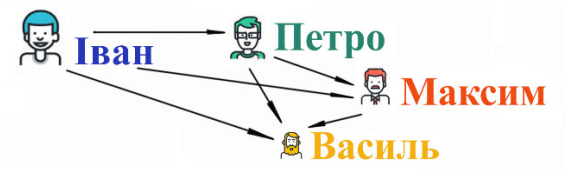



Рис. 1.31. Граф відношення $R_2$

та перерахуванням елементів:

$R_2 = \{(\text{Іван, Петро}),(\text{Іван,Максим}),(\text{Іван,Василь}),(\text{Петро,Максим}),(\text{Петро,Василь}),(\text{Максим,Василь})\}$

Перевіримо властивості відношення $R_2$, використовуючи визначення транзитивності.

$$\left.\begin{array}{l}(\text{Іван, Петро}) \in R_2 \\ (\text{Петро, Максим}) \in R_2\end{array}\right\} \Rightarrow (\text{Іван, Максим}) \in R_2$$

$$\left.\begin{array}{l}(\text{Іван, Максим}) \in R_2 \\ (\text{Максим, Василь}) \in R_2\end{array}\right\} \Rightarrow (\text{Іван, Василь}) \in R_2$$

$$\left.\begin{array}{l}(\text{Петро, Максим}) \in R_2 \\ (\text{Максим, Василь}) \in R_2\end{array}\right\} \Rightarrow (\text{Петро, Василь}) \in R_2$$

Отже, відношення $R_2$ є транзитивним.

**Визначення.** Граф транзитивного відношення $R_{tr}$ – це граф, у якому для всякої пари дуг таких, що кінець першої збігається з початком другої, існує третя дуга, що має початок у спільній вершині з першою і кінець у спільній вершині з другою.

**Приклад 1.71.** Розглянемо відношення $R_{tr} \subseteq A \times A$ на множині $A = \{a_1, a_2, a_3, a_4\}$, задане перерахуванням елементів

$$R_{tr} = \{(a_1, a_2),(a_2, a_3),(a_1, a_3),(a_3, a_4),(a_1, a_4),(a_2, a_4)\}$$

Побудувати граф відношення $R_{tr}$ та визначити, чи є дане відношення транзитивним.

*Розв'язок.*

Використовуючи правила побудови графа відношення, побудуємо граф відношення $R_{tr}$, як показано на рис. 1.32.

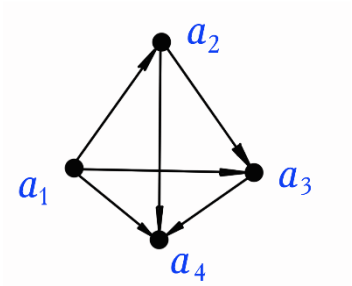



Рис. 1.32. Граф відношення $R_{tr}$

Умови існування транзитивності даного відношення

$$\left.\begin{array}{l}(a_1, a_2) \in R_3 \\ (a_2, a_3) \in R_3\end{array}\right\} \Rightarrow (a_1, a_3) \in R_3$$

$$\left.\begin{array}{l}(a_1, a_3) \in R_3 \\ (a_3, a_4) \in R_3\end{array}\right\} \Rightarrow (a_1, a_4) \in R_3$$

$$\begin{matrix}(a_2,a_3)\in R_3\\(a_3,a_4)\in R_3\end{matrix}\Rightarrow(a_2,a_4)\in R_3$$

*Антитранзитивність*

**Визначення.** Відношення $R_{at}\subseteq X\times X$ на множині $X=\{x_1,\ldots,x_i,\ldots,x_j,\ldots,x_k,\ldots\}$ називають ***антитранзитивним***, якщо для будь-яких $x_i,x_j,x_k$ з $x_iR_{tr}x_j$ і $x_jR_{tr}x_k$ не виконується $x_iR_{tr}x_k$.

**Приклад 1.72.** Розглянемо відношення $R_1\subseteq A\times A$ на множині $A=\{2010,2011,2012,2013,2014,2015,2016\}$, задане предикатом: $R_1=\{(a,b)\,|\,a$ є наступним роком за $b$ на множині $A\}$. Визначити, чи є відношення $R_1$ антитранзитивним.

*Розв'язок.*

Використовуючи предикат, побудуємо відношення, задане перерахуванням елементів:

$$R_1=\{(2010,2011),(2011,2012),(2012,2013),(2013,2014),(2014,2015),(2015,2016)\}$$

Перевіримо властивості відношення $R_1$, використовуючи визначення антитранзитивності.

$$\begin{matrix}(2010,2011)\in R_1\\(2011,2012)\in R_1\end{matrix}\Rightarrow(2010,2012)\notin R_1$$

$$\begin{matrix}(2014,2015)\in R_1\\(2015,2016)\in R_1\end{matrix}\Rightarrow(2014,2016)\notin R_1$$

Отже, відношення $R_1$ є антитранзитивним.

**Приклад 1.73.** Розглянемо відношення $R_2\subseteq A\times A$ на множині $A=\{Іван,Петро,Василь,Максим\}$, задане предикатом

$R_2=\{(a,b)\,|\,a$ є батьком $b\}$.

Визначити, чи є відношення $R_2$ антитранзитивним.

*Розв'язок.* Представимо відношення $R_2$ графом, як показано на рис. 1.33:

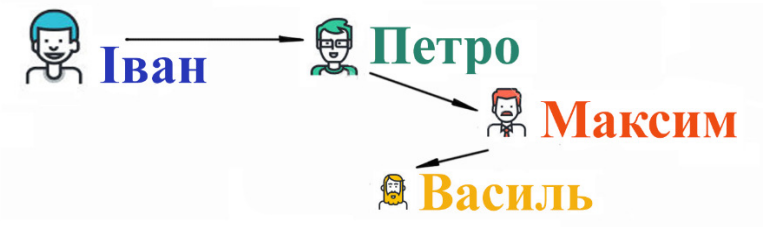



Рис. 1.33. Граф відношення $R_2$

та перерахуванням елементів:

$$R_2 = \{(\textit{Іван,Петро}),(\textit{Петро,Максим}),(\textit{Максим,Василь})\}$$

Перевіримо властивості відношення $R_2$, використовуючи визначення антитранзитивності.

$$\begin{matrix}(\textit{Іван,Петро}) \in R_2 \\ (\textit{Петро,Максим}) \in R_2\end{matrix} \Rightarrow (\textit{Іван,Максим}) \notin R_2$$

$$\begin{matrix}(\textit{Петро,Максим}) \in R_2 \\ (\textit{Максим,Василь}) \in R_2\end{matrix} \Rightarrow (\textit{Петро,Василь}) \notin R_2$$

Отже, відношення $R_2$ є антитранзитивним.

**Приклад 1.74.** Розглянемо відношення $R \subseteq X \times X$ на множині $X = \{\alpha, \beta, \gamma, \delta\}$, задане перерахуванням:

$$R = \{(\alpha,\alpha),(\alpha,\beta),(\alpha,\delta),(\beta,\alpha),(\delta,\alpha),(\delta,\delta),(\gamma,\delta),(\gamma,\gamma)\}.$$

Визначити спеціальні властивості відношення $R$.

*Розв'язок.*

1. $R$ не є рефлексивним, оскільки $\beta \in X$, але $(\beta,\beta) \notin R$.
2. $R$ не є симетричним, оскільки $(\gamma,\delta) \in R$, але $(\delta,\gamma) \notin R$.
3. $R$ не є антисиметричним, оскільки $(\alpha,\beta) \in R$ й $(\beta,\alpha) \in R$, але $\alpha \neq \beta$.
4. $R$ не є транзитивним, оскільки $(\beta,\alpha) \in R, (\alpha,\delta) \in R$, але $(\beta,\delta) \notin R$.

**Контрольні запитання**

1. Скільки і які типи відповідностей вам відомі. Дайте визначення кожного з типів відповідності.
2. Дайте визначення відношення. Поясніть, чим відрізняється відповідність від відношення.
3. Розглянемо відношення $R \subset A \times A$ на множині $A = \{1,2,3,4,5\}$, задане перерахуванням: $R = \{(1,4),(2,4),(3,4),(1,5),(2,5),(3,5),(5,4)\}$. Побудуйте матрицю та граф відношення $R$, сформулюйте правила побудови матриць та графів відношення.
4. Розглянемо відношення $R \subset A \times B$ на декартовому добутку множин $A = \{1,3,5,7,9,11,13,15\}$ і $B = \{b_1, b_2, b_3, b_4, b_5, b_6\}$, задане графом, представленим на рис. 1.34.

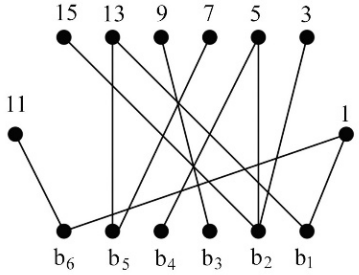



Рис. 1.34. Граф відношення $R$

Побудувати зрізи через кожний з елементів множини $A$.

5. Дайте визначення рефлексивності відношення. Властивості матричного представлення рефлексивного відношення.

6. Дайте визначення симетричності та транзитивності відношень.

## 1.3. Відношення еквівалентності

### 1.3.1. Визначення відношення еквівалентності

Деякі елементи множини можна розглядати як еквівалентні в тому випадку, коли кожний із цих елементів при деякому розгляді може бути замінений іншим. У цьому випадку говорять, що дані елементи перебувають у відношенні еквівалентності.

**Визначення.** Відношення $R$ на множині $X$ є відношенням еквівалентності, якщо воно рефлексивне, симетричне й транзитивне.

### 1.3.2. Властивості еквівалентних відношень

1. Властивість **рефлексивності** проявляється в тому, що кожний елемент еквівалентний самому собі або $x \equiv x$.

2. Висловлювання про те, що два елементи є еквівалентними, не вимагає уточнення, який з елементів розглядається першим, який другим, тобто наявною є $x \equiv y \to y \equiv x$ – властивість **симетричності.**

3. Два елементи, еквівалентні третьому, еквівалентні між собою, або наявною є $x \equiv y$ і $y \equiv z \to z \equiv z$ – властивість **транзитивності.**

Як загальний символ відношення еквівалентності використовується символ «$\equiv$» (іноді символ «$\sim$»). Для окремих відношень еквівалентності використовуються інші символи:

«=» – для позначення рівності;

«$\parallel$» –для позначення паралельності;

«$\leftrightarrows$» або «$\rightleftarrows$» – для позначення логічної еквівалентності.

**Приклад 1.75.** Розглянемо відношення $R \subset A \times A$ на множині $A = \{1,2,3,4,5,6\}$, задане перерахуванням:

$$R = \{(1,1),(2,2),(3,3),(4,4),(5,5),(6,6),(1,2),(1,4),$$
$$(2,1),(2,4),(3,5),(5,3),(4,1),(4,2)\}$$

Визначити, чи є відношення еквівалентним.

*Розв'язок.*

Легко перевірити, що відношення $R$ є рефлексивним, симетричним і транзитивним. Тому воно є відношенням еквівалентності на множині $A$.

### 1.3.3. Класи еквівалентності

**Визначення.** Відношення еквівалентності $R_e$ – це відношення на множині $A$, яке розбиває дану множину на підмножини, елементи яких еквівалентні один одному, але не еквівалентні елементам інших підмножин.

**Визначення. Класами еквівалентності** називають підмножини, що не перетинаються та отримані в результаті розбиття множини $A$ відношенням еквівалентності $R_e$

**Визначення**. Множину класів еквівалентності множини $A$ відносно $R_e$ називають **фактор-множиною** і позначають $\left[A\right]_R$ .

**Приклад 1.76.** Розбиття множини на підмножини.
Нехай множина $B$ – це набір різнокольорових повітряних кульок.
1. Відношення $R_1$ задамо умовою: $\left(a,b\right) \in R_1$ якщо «$a$ одного кольору з $b$». *Одержимо класи еквівалентності з кульок одного кольору.*

Рис. 1.35. Множина кульок $B$

2. Відношення $R_2$ задамо умовою:
$\left(a,b\right) \in R_2$ якщо «$a$ одного розміру з $b$»
*Одержимо класи еквівалентності з кульок одного розміру.*
3. Відношення $R_3$ задамо умовою:
$\left(a,b\right) \in R_3$ якщо «$a$ однакової форми з $b$»
*Одержимо класи еквівалентності з кульок однакової форми.*

**Визначення.** Нехай $a_i \in A$ *– елемент множини* $A = \left\{a_1, a_2, \ldots, a_i, \ldots, a_n\right\}$. Тоді $\left[a_i\right]$ позначає множину $\left\{x \middle| xRa_i\right\} = \left\{x \middle| \left(x, a_i\right) \in R\right\}$, *яку* називають ***класом еквівалентності***, що містить $a_i$. Символ $\left[A\right]_R$ позначає множину всіх класів еквівалентності множини $A$ по відношенню $R$. Таким чином, $\left[A\right]_R$ – фактор-множина.

**Приклад 1.77.** Нехай $A = \left\{1,2,3,4,5,6\right\}$ і дано відношення еквівалентності:
$R = \left\{\left(1,1\right),\left(2,2\right),\left(3,3\right),\left(4,4\right),\left(5,5\right),\left(6,6\right),\left(1,2\right),\left(1,4\right),\right.$
$\left.\left(2,1\right),\left(2,4\right),\left(3,5\right),\left(5,3\right),\left(4,1\right),\left(4,2\right)\right\}$

Класи еквівалентності відношення $R$ були отримані шляхом визначення класу еквівалентності кожного елемента множини $A$:

$[1] = \{x \mid (x,1) \in R\} = \{x \mid xR1\} = \{1,2,4\}$, де

$1 \in [1]$, оскільки $(1,1) \in R$,

$2 \in [1]$, оскільки $(2,1) \in R$,

$4 \in [1]$, оскільки $(4,1) \in R$,

не існує жодного іншого $x \in A$ такого, що $(x,1) \in R$.

Так само одержуємо

$[2] = \{x \mid (x,2) \in R\} = \{x \mid xR2\} = \{2,1,4\}$

$[3] = \{x \mid (x,3) \in R\} = \{x \mid xR3\} = \{3,5\}$

$[4] = \{x \mid (x,4) \in R\} = \{x \mid xR4\} = \{4,1,2\}$

$[5] = \{x \mid (x,5) \in R\} = \{x \mid xR5\} = \{5,3\}$

$[6] = \{x \mid (x,6) \in R\} = \{x \mid xR6\} = \{6\}$

**Приклад 1.78.** Нехай $Q$ – множина раціональних чисел.

Розіб'ємо $Q$ на класи еквівалентності, для яких $a/b$ – раціональний дріб, де $a \in Z$, $b \in N$ .

Будь-який дріб $c/d$ буде віднесений до одного класу еквівалентності з $a/b$ тоді й тільки тоді, коли $ad = bc$.

(Наприклад: 2/4 ≡ 3/6 бо $3 \cdot 4 = 6 \cdot 2 = 12$).

Властивості такого відношення.

1. **Рефлексивність.** Для будь-якого дробу $a/b$ виконується рівність $ab = ba$. Отже, $a/bRa/b$.

2. **Симетричність.** Якщо $a/bRc/d$, то $ad=bc$, у той же час $bc = ad$. Звідси $c/d\ Ra/b$.

3. **Транзитивність.** Нехай $a/bRc/d$ і $c/dRm/n$. Доведемо, що $a/bRm/n$, тобто $an = bm$. Дійсно, оскільки $a/bRc/d$, то $ad = bc$ і $c/dRm/n$, то $cn = dm$. Домножимо першу рівність на $n$, а другу на $b$, одержимо $and = bcn$ і $bcn = bmd$. В обох рівностях присутнє $bcn$. Тому $and = bmd$ або $an = bm$.

### 1.3.4. Замикання множин

**Визначення.** Множину $A$ називають замкнутою відносно деякої операції, якщо результатом виконання даної операції над елементами множини $A$ завжди буде елемент, який належить множині $A$.

**Приклад 1.79.** Нехай множина $N$ – множина натуральних чисел. Розглянемо операцію «+» на множині $N$. Чи є множина $N$ замкнутою відносно операції «+»?

*Розв'язок.* Нехай $n \in N$ і $m \in N$. Тоді $n + m = k \in N \ \forall\ n,m \in N$

**Приклад 1.80.** Нехай множина $Z$ є множиною цілих чисел, а множина $N$ – множина натуральних чисел.
Довести, що замиканням для множини $N$ відносно операції «-» є множина $Z$.
*Розв'язок.* Нехай $n \in N$ і $m \in N$.

$$\text{Тоді } \begin{cases} n-m=k \in N \ \forall \ n>m \to k \in Z \\ n-m=k \in N \ \forall \ n=m \to k \in Z \ \forall \ n,m \in N \\ n-m=k \in N \ \forall \ n<m \to k \in Z \end{cases}$$

**Контрольні запитання**

1. Дайте визначення відношення еквівалентності. Сформулюйте властивості еквівалентних відношень.
2. Розглянемо відношення $R \subset A \times A$ на множині $A = \{a,b,c,d,e,f\}$, задане перерахуванням:

$$R = \{(a,a),(b,b),(c,c),(d,d),(e,e),(f,f),(a,b),(a,d),$$
$$(b,a),(b,d),(c,e),(e,c),(d,a),(d,b)\}$$

Визначити, чи є відношення $R$ еквівалентним.
3. Дайте визначення класу еквівалентності та фактор-множини.
4. Нехай дано множину

$$A = \{\textit{береза}, \textit{клен}, \textit{дуб}, \textit{яблуня}, \textit{груша}, \textit{вишня}, \textit{сосна}, \textit{ялина}, \textit{ніхта}\}$$

Побудувати всі можливі класи еквівалентності на множині $A$.
5. Розглянемо множину

$$A = \{\textit{хліб}, \textit{булка}, \textit{печиво}, \textit{пилосос}, \textit{смартфон}, \textit{телевізор}, \textit{картопля}, \textit{морква}, \textit{буряк}\}$$

Побудувати всі можливі класи еквівалентності на множині $A$.
6. Дайте визначення замкнутої множини. Дайте приклади замкнутих множин.

## 1.4. Відношення порядку

### 1.4.1. Приклади відношень порядку

Існують відношення, які визначають порядок розташування елементів множини.
1. Умову відношення «$t_i$ раніше $t_j$» і «$t_j$ пізніше $t_i$»
використовуємо у випадках, коли елементами множини є стани динамічної системи

$$T = \{t_0, t_1, t_2, t_3, ..., t_{n-1}\},$$

де $t_0 < t_1 < t_2 < t_3 < ... < t_{n-1}$
Приклад процесу, який послідовно розвивається в часі, показано на рис. 1.36.


Рис. 1.36. Процес, представлений послідовністю кроків

Символи «<» «>» використовують для порівняння величин відрізків часу, вимірюваних від початку відліку.

Запишемо ці відношення у вигляді предиката:

$$R_1 = \left\{ \left( t_i, t_j \right) \middle| t_i < t_j \; при \; i < j \right\}$$

$$R_2 = \left\{ \left( t_j, t_i \right) \middle| t_j > t_i \; при \; j > i \right\}$$

2. Умову відношення «$a_j$ більше $a_i$» або «$a_i$ менше $a_j$» використовуємо, коли елементами множини є числа або об'єкти, що мають властивість, виражену числом.

$$A = \left\{ a_0, a_1, a_2, ..., a_i, ..., a_j, ..., a_n \right\},$$

де $a_0 < a_1 < a_2 < ... < a_i < ... < a_j < ... < a_n$.

Приклад порівняння об'єктів за розміром показаний на рис. 1.37.

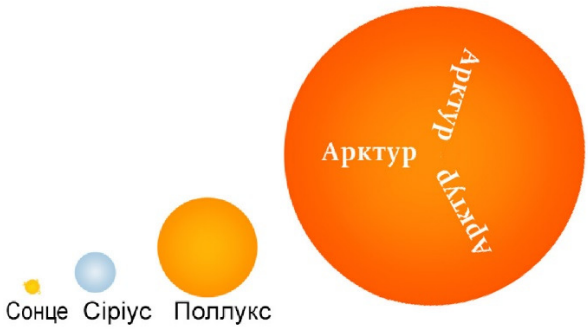



Рис. 1.37. Відношення об'єктів за величиною

Символами «>» або «<» користуються для порівняння чисел.

Відношення $R$ у вигляді предиката, задане на $A \times A$.

$$R_1 = \left\{ \left( a_i, a_j \right) \middle| a_i < a_j \; при \; i < j \right\}$$

$$R_2 = \left\{ \left( a_j, a_i \right) \middle| a_j > a_i \; при \; j > i \right\}$$

Умова відношення: «$A_i$ *входить в* $A_j$», «$A_i$ *строго входить в* $A_j$» використовуємо, коли елементами множини є множини.

$$A = \left\{ A_0, A_1, ..., A_i, ..., A_j, ..., A_n \right\}.$$

де $A_0 \subseteq A_1 \subseteq ... \subseteq A_i \subseteq ... \subseteq A_j \subseteq ... \subseteq A_n$ або

$A_0 \subset A_1 \subset ... \subset A_i \subset ... \subset A_j \subset ... \subset A_n$ .

Приклад відношення множин показано на рис. 1.38.

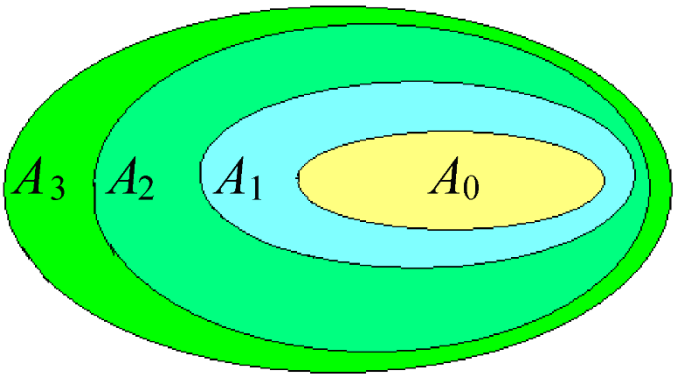



Рис. 1.38. Відношення множин

Відношення $R$ у вигляді предиката, задане на $A \times A$.

$$R_1 = \left\{ \left( A_i, A_j \right) \middle| A_i \subseteq A_j \ при \ i < j \right\}$$

$$R_2 = \left\{ \left( A_i, A_j \right) \middle| A_i \subset A_j \ при \ i < j \right\}$$

У всіх випадках можна розташувати елементи множин у деякому порядку або, інакше кажучи, ввести відношення порядку на множині.

### 1.4.2. Визначення відношень порядку

Відношення порядку на множині $A$ поділяють на:
- відношення строгого порядку;
- відношення нестрогого порядку.

**Визначення.** Відношення $R$ називають **відношенням строгого порядку** на множині $A$, якщо воно має властивості:

- *антирефлексивності*, тобто якщо $xRy$ то $x \neq y$.
- *антисиметричності,* тобто, якщо $xRy$ і $yRx$, то $x = y$.
- *транзитивності,* тобто, якщо $xRy$ і $yRz$, то $xRz$.

**Визначення.** Відношення $R$ називають **відношенням нестрогого порядку** на множині $A$, якщо воно має властивості:

- *рефлексивності*, тобто, $xRx$.
- *антисиметричності,* тобто, якщо $xRy$ і $yRx$, то $x = y$
- *транзитивності,* тобто, якщо $xRy$ і $yRz$, то $xRz$.

### 1.4.3. Термінологія й позначення

1. **Відношення нестрогого порядку** позначають символом «$\leq$» за аналогією з відношенням «менше або дорівнює» на множині дійсних чисел. При цьому, якщо $a \leq b$, то говорять, що елемент $a$ не перевищує $b$ або $a$ підпорядкований $b$.

2. **Відношення строгого порядку**. Якщо $a \leq b$ і $a \neq b$, то пишуть $a < b$ і говорять, що $a$ менше $b$ або що елемент $a$ строго підпорядкований $b$.

1. **Загальний випадок відношень**.
Відношення порядку на множинах: «$\subseteq$» і «$\subset$»
Відношення порядку на числах: «$\leq$» і «$<$»
Відношення порядку в часі: «$\preceq$» і «$\prec$»

**Приклад 1.81.**

**Відношення порядку в кортежах, що задають координати в просторі** $R^n$
1. Відношення чисел «$\leq$» «$\geq$» задають нестрогий порядок.
$(a_1,...,a_{i-1},a_i,a_{i+1},...,a_n) \leq (b_1,...,b_{i-1},b_i,b_{i+1},...,b_n)$, якщо
$a_1 \leq b_1,..., a_{i-1} \leq b_{i-1}, a_i \leq b_i, a_{i+1} \leq b_{i+1},..., a_n \leq b_n$
2. Відношення чисел «$<$» «$>$» задають строгий порядок
$(a_1,...,a_{i-1},a_i,a_{i+1},...,a_n) < (b_1,...,b_{i-1},b_i,b_{i+1},...,b_n)$, якщо
$a_1 < b_1,...,a_{i-1} < b_{i-1},a_i < b_i,a_{i+1} < b_{i+1},...,a_n < b_n$
3. Однак для встановлення нестрогого порядку достатньо, щоб умова $a_i \leq b_i$ була виконана хоча б по одній координаті, тобто
$(a_1,...,a_{i-1},a_i,a_{i+1},...,a_n) \leq (b_1,...,b_{i-1},b_i,b_{i+1},...,b_n)$, якщо
$a_1 < b_1,..., a_{i-1} < b_{i-1}, a_i \leq b_i, a_{i+1} < b_{i+1},..., a_n < b_n$

**Приклад 1.82.** Відношення порядку в кортежах, що задають координати в просторі $R^2$. $(a_1,a_2) \leq (b_1,b_2)$

Це відношення справедливе, якщо $a_1 \leq b_1$ і $a_2 \leq b_2$

$(1,5) \leq (1,7) \rightarrow 1 = 1$ і $5 < 7$ порівнювані елементи кортежів;

$(1,5)$ і $(5,1)$ непорівнювані елементи кортежів.

$(a_1,a_2) < (b_1,b_2)$

$(1,5) < (2,7) \rightarrow (1 < 2)$ і $5 < 7$ порівнювані елементи кортежів;

$(1,5)$ і $(7,2)$ непорівнювані елементи кортежів.

**Приклад 1.83.** Відношення порядку в кортежах, що задають координати в просторі $R^3$.

$$(a_1,a_2,a_3) \leq (b_1,b_2,b_3)$$

Це відношення справедливе, якщо $a_1 \le b_1$, $a_2 \le b_2$ і $a_3 \le b_3$

$(1,2,3) \le (1,2,4) \rightarrow 1 = 1; 2 = 2; 3 < 4$

$$(a_1, a_2, a_3) < (b_1, b_2, b_3)$$

Це відношення справедливе, якщо $a_1 < b_1$, $a_2 < b_2$ і $a_3 < b_3$

$(1,2,3) < (2,3,4) \rightarrow 1 < 2;\ 2 < 3;\ 3 < 4$

### 1.4.4. Види відношень порядку

*Строгий повний порядок*

**Визначення.** Відношення строгого повного порядку – це відношення, яке має властивості *антирефлексивності*, *антисиметричності та транзитивності* і задане **на всіх** елементах упорядкованої множини.

*Строгий частковий порядок*

**Визначення.** Відношення строгого часткового порядку – це відношення, яке має властивості *антирефлексивності*, *антисиметричності* та *транзитивності* і задане **не на всіх** елементах упорядкованої множини.

*Нестрогий повний порядок*

**Визначення.** Відношення нестрогого повного порядку – це відношення, яке має властивості *рефлексивності*, *антисиметричності* та *транзитивності* і задане **на всіх** елементах упорядкованої множини.

*Нестрогий частковий порядок*

**Визначення.** Відношення нестрогого часткового порядку – це відношення, яке має властивості *рефлексивності*, *антисиметричності* та *транзитивності* і задане **не на всіх** елементах упорядкованої множини.

### 1.4.5. Основні поняття про впорядковані множини

Упорядковані множини утворюють один з фундаментальних типів математичних структур.

**Визначення.** *Упорядкованою множиною* називають непусту множину $X$ разом із заданим на ньому бінарним нестрогим «$\le$» *відношенням порядку,* яке за визначенням:

1) рефлексивне: $a \le a$;

2) антисиметричне: $a \le b \le a \Rightarrow a = b$ ( для будь-яких $a, b, X$ );

3) транзитивне: $a \le b \le c \Rightarrow a \le c$;

або строгим «$<$» *відношенням порядку,* яке за визначенням:

1) антирефлексивне: $a < b \Rightarrow a \ne b$;

2) антисиметричне: $a < b \wedge b < a \Rightarrow a = b$;

3) транзитивне: $a \le b \le c \Rightarrow a \le c$.

### 1.4.6. Лінійно впорядковані множини

**Визначення порівнюваності.** Елементи $a$ і $b$ упорядкованої множини називають *порівнюваними,* якщо $a < b$, $a = b$ або $a > b$. Знаки $<$, $=$ і $>$ мають звичайний зміст.

**Визначення лінійно впорядкованої множини**
**(через порівнюваність)**. Упорядковану множину $X$ називають *лінійно впорядкованою,* або *ланцюгом,* якщо будь-які два його елементи порівнювані.

**Визначення відношення лінійного порядку**. Бінарне відношення *R* на множині $X$ називають *відношенням лінійного порядку,* якщо для будь-яких $a \in X$ і $b \in X$ (для будь-яких двох елементів множини $X$) або $aRb$, або $bRa$.
*Іншими словами*, відношення порядку $R$ на множині $X$ називають відношенням лінійного порядку, якщо будь-які два елементи цієї множини перебувають у відношенні $R$.

**Визначення лінійно впорядкованої множини**
**(через відношення лінійного порядку)**. Упорядковану множину $X$ називають *лінійно впорядкованою,* або *ланцюгом,* якщо на ній задане відношення лінійного порядку.
Отже, відношення лінійного порядку – це відношення, яке має властивості рефлексивності, антисиметричності та транзитивності. Такий порядок завжди повний та нестрогий.
**Визначення ланцюга на множині.**
Ланцюг – лінійно впорядкована множина.
**Визначення ланцюга на підмножині.**
Ланцюг – це лінійно впорядкована підмножина частково впорядкованої множини

**Приклад 1.84.** Навести приклади лінійно впорядкованих множин (ланцюгів)
*Розв'язок.*
*Натуральні числа* – найменша лінійно впорядкована множина, що не має верхньої межі.
$A = \{1, 2, 3, 4, \ldots.\}$
*Цілі числа* – найменша лінійно впорядкована множина, що не має ні верхньої, ні нижньої межі.
$A = \{\ldots. - 4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$
**Визначення.** Антиланцюг – упорядкована множина, у якій жодні два різні елементи не є порівнюваними.

**Приклад 1.85.** Навести приклади антиланцюгів.

*Розв'язок.*

Антиланцюг непорівнюваних елементів: $A = \{a, 1, \exists\}$.

Антиланцюг непорівнюваних кортежів: $B = \{(1,2,3),(4,1,6)\}$

### 1.4.7. Властивості лінійно впорядкованих множин

*Покриття*

**Визначення.** Нехай $X$ – довільний ланцюг. Якщо $a < b$ в $X$ і не існує елемента $c \in X$ з умовою $a < c < b$ (який розміщений між $a$ і $b$), то співвідношення $a < b$ називають *покриттям.*

**Приклад 1.86.** Розглянемо відношення строгого порядку $R \subset A \times A$, яке задає предикатом $R = \{x < y \mid \textit{якщо } x \textit{ передує } y \textit{ в алфавітному порядку}\}$ порядок елементів у множині $A = \{a, b, c, d, e, f\}$. Визначити можливі ланцюги, які є покриттями.

*Розв'язок.*

Покриття: $a < b$, $b < c$, $c < d$, $d < e$, $e < f$

Не є покриттями: $a < c$, оскільки $a < b < c$; $b < d$, оскільки $b < c < d$

*Взаємне розташування елементів*

**Визначення.** Нехай $X$ – довільний ланцюг. Якщо $a < b$ в $X$, то елемент $a$ *називають попереднім* для $b$, а елемент $b$ називають *наступним* за $a$.

Елемент ланцюга, у якого немає попереднього елемента або наступного елемента, називають *граничним елементом.*

**Приклад 1.87.** Розглянемо відношення строгого порядку $R \subset A \times A$, яке задає предикатом $R = \{x < y \mid \textit{якщо } x \textit{ передує } y \textit{ в алфавітному порядку}\}$ порядок елементів у множині $A = \{a, b, c, d, e, f\}$. Для $c \in A$ визначити попередній і наступний елементи. Вказати граничні елементи для множини $A$.

*Розв'язок.*

Елемент $b \in A$ є попереднім для елемента $c \in A$.

Елемент $d \in A$ є наступним для елемента $c \in A$.

Елементи $a \in A$ і $f \in A$ є граничними елементами множини $A$.

*Щільний ланцюг*

**Визначення.** Ланцюг називають *щільним,* якщо в ньому немає покриттів. У щільних ланцюгах між будь-якими елементами $a < b$ лежить нескінченна кількість елементів.

**Приклад 1.88.** Навести приклад щільного ланцюга.

*Розв'язок.*

Прикладом щільного ланцюга є множина дійсних чисел в просторі $R$.

Доведення. Візьмемо два довільних дійсних числа: $1 < 1.4$.
Вони не є покриттям, оскільки $1 < 1.2 < 1.4$;
Нехай $1.4 < 2.345$.
Цей ланцюг також не є покриттям, оскільки $1.4 < 2 < 2.345$ і т. д.

*Повний зверху ланцюг*
**Визначення.** Ланцюг називають *повним зверху,* якщо його довільна непуста підмножина має sup (супремум).

*Повний знизу ланцюг*
**Визначення.** Ланцюг називають *повним знизу,* якщо його довільна непуста підмножина має inf (інфімум).

*Повний ланцюг*
**Визначення.** Ланцюг називають *повним,* якщо він повний зверху і знизу одночасно.

### 1.4.8. Цілком упорядкована множина

Найважливіший клас ланцюгів утворюють цілком упорядковані множини.
**Визначення.** Ланцюг називають *цілком упорядкованою множиною,* якщо будь-яка її непуста підмножина має найменший елемент.

**Приклад 1.89.** Навести приклади цілком упорядкованих множин.

*Розв'язок.*

1. Ланцюг усіх натуральних чисел $\mathrm{N} = \{1,2,3,4,5,6,\ldots\}$ є цілком упорядкованою множиною, оскільки її найменший елемент – 1.

2. Усі скінченні ланцюги є прикладами цілком упорядкованих множин.

3. Будь-яка непуста підмножина цілком упорядкованої множини цілком упорядкована.

### 1.4.9. Частково впорядкована множина

**Визначення (через відношення часткового порядку)**.
Упорядковану множину $X$ називають *частково впорядкованою,* якщо для неї задане відношення часткового порядку.

**Визначення.** Бінарне відношення $R$ на множині $X$ називають відношенням ***часткового порядку,*** якщо для деяких елементів $a \in X$ та $b \in X$ не виконується ні відношення $aRb$, ні відношення $bRa$.

**Приклад 1.90.** Нехай дано множину $X = \{x, y, z, 1, 2, 3\}$. Довести, що множина $X$ є частково упорядкованою.

*Розв'язок.*
У множині $X$ букви розміщені у алфавітному порядку, числа упорядковані за величиною, але букви і числа є непорівнянними. Отже $X$ – частково упорядкована множина.

*Властивості частково впорядкованих множин*
(*строгий та нестрогий порядок*)
**Якщо відношення $R$ на $X$ є відношенням** *нестрогого часткового порядку*, то воно

рефлексивне – $\forall a\,(aRa)$,

антисиметричне – $\forall a,b\,(aRb)\wedge(bRa)\Rightarrow a=b$

транзитивне – $\forall a,b,c\,(aRb)\wedge(bRc)\Rightarrow aRc$ .

**Якщо відношення $R$ на $X$ є відношенням** *строгого часткового порядку*, то воно

антирефлексивне – $\forall a,b(aRb)\Rightarrow a\neq b$,

антисиметричне – $\forall a,b(aRb)\wedge(bRa)\Rightarrow a=b$

транзитивне – $\forall a,b,c(aRb)\wedge(bRc)\Rightarrow aRc$.

**Приклад 1.91.** Нехай задано $A$ – множину натуральних чисел від 1 до 30:

$$A=\{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,\ldots,29,30\}.$$

Також задано відношення «$\leq$ », згідно якого *m* і *n є порівнюваними*: $m\leq n$ *за умови, що n* ділиться на *m* націло.

Нехай *n*=30 і *m*=5. Тоді $n$ і $m$ – є порівнюваними, оскільки 30 ділиться на 5 націло.

Нехай *n*=30 і *m*=7. Тоді $n$ і $m$ – непорівнювані, оскільки 30 не ділиться на 7 націло.

Це правило можна представити предикатом:

$$R=\{(m,30)\,|\,m\leq 30\wedge 30\%m==0\}$$

Довести, що задане відношення на множині $A$ є відношенням часткового порядку, а множина $A$ є частково упорядкованою множиною.

*Розв'язок.*
Задане відношення порядку «$\leq$ » на множині $A$ є відношенням часткового порядку, оскільки не всі елементи множини є порівнюваними.

Множина $A$ є *частково упорядкованою множиною* на заданому відношенні, оскільки тільки її підмножина $T_1$ містить порівнювані елементи.

$$T_1=\{1,2,3,5,6,10,15,30\}\quad T_2=\{4,7,8,9,11,12,13,14,16,\ldots 29\}.$$

### 1.4.10. Розбиття частково впорядкованої множини на ланцюзі

Нехай є деяка множина $A$. Говорять, що множина $A$ *розбита* на підмножини $A_1, A_2, A_3, \ldots A_m$, якщо:

1. $A_i \neq \varnothing, \ (i = 1,2,...,m)$;
2. $A_i \cap A_j = \varnothing$, якщо $i \neq j$ для всіх $i, j \in \{1,2,3,...,m\}$;
3. $A = \bigcup_{i=1}^{m} A_i$

Нехай $A$ є частково впорядкованою множиною. Розбиття множини $A$ на ланцюзі називають *найменшим*, якщо воно має найменше число елементів $m$ у порівнянні з іншими розбиттями $A$ на ланцюзі. Таке розбиття також називають *мінімальним ланцюговим розбиттям* (*МЛР*) множини $A$.

**Приклад 1.92.**

Нехай дана множина $A$: $A = \{1, a, \angle, б, 2, 7, в, \triangleleft, 1245, \square, д\}$,

на якій задані такі відношення часткового порядку:

$R_1 = \{(a,b) \mid a \leq b\}$, $R_2 = \{(a,b) \mid$ *"a слідує в алфавітному порядку за b"*$\}$,

$R_3 = \{(a,b) \mid$ *"a має більше кутів, ніж b"*$\}$. Побудувати МЛР на множині $A$.

*Розв'язок.*

Побудуємо спочатку довільне розбиття цієї множини на ланцюзі

$A_1 = \{1,2\}; A_2 = \{7,1245\}; A_3 = \{a,б\}; A_4 = \{в,д\}; A_5 = \{\angle, \triangleleft, \square\}$

$m = 5$, $A = \bigcup_{i=1}^{m} A_i$, $A_i \neq \varnothing \ (i = 1,...,5)$.

Це розбиття не є МЛР, оскільки існує можливість зменшити кількість підмножин розбиття.

$A_i \cap A_j = \varnothing$ якщо $i \neq j$ для всіх $i, j \in \{1,2,3,4,5\}$.

$m = 3 \quad A_1 = \{1,2,7,1245\}; A_2 = \{a,б,в,д\}; \ A_3 = \{\angle, \triangleleft, \square\}$

Отже, розбиття $A_1, A_2, A_3$ – МЛР.

### 1.4.11. Найбільший елемент множини

**Визначення. Найбільшим** елементом лінійно впорядкованої множини $X$ відносно строгого «$<$» або нестрогого «$\leq$» упорядкування будемо називати такий елемент $a \in X$, що для будь-якого $x \in X$ вірно $x < a$ або $x \leq a$.

**Теорема 4.1.** Про єдиність найбільшого елемента.

Якщо існує найбільший елемент лінійно впорядкованої множини, то він є єдиним.

*Доведення.* Припустимо, що $a$ – найбільший елемент і $a'$ – також найбільший елемент.

Тоді для будь-якого $x$ виконується $x \le a$ і $x \le a'$. Зокрема, $a \le a'$ або $a' \le a$. За властивістю антисиметричності, з $\left(aRa'\right) \wedge \left(a'Ra\right)$ випливає $a = a'$.

Оскільки $a = a'$, то якщо існує найбільший елемент, то він єдиний.
Тому, якщо говорять про найбільший елемент множини, то мають на увазі **цілком визначений** її елемент.

**Приклад 1.93.** Необхідно знайти найбільший елемент лінійно впорядкованої множини $X = \{1,2,15,18\}$, заданої на відношенні нестрогого порядку $a \le b$.
*Розв'язок.*
Згідно з визначенням:
1. Усі елементи даної множини повинні бути меншими або дорівнювати найбільшому.
2. Найбільший елемент – єдиний.
Порівняємо елементи множини $X$:

1) $1 \ge 1,\ 1 \not\ge 2,\ 1 \not\ge 15,\ 1 \not\ge 18.$
2) $2 \ge 1,\ 2 \ge 2,\ 2 \not\ge 15,\ 2 \not\ge 18.$
3) $15 \ge 1,\ 15 \ge 2,\ 15 \ge 15,\ 15 \not\ge 18.$
4) $18 \ge 1,\ 18 \ge 2,\ 18 \ge 15,\ 18 \ge 18.$

Необхідним умовам відповідає тільки елемент 18.

### 1.4.12. Максимальний елемент множини

**Визначення. Максимальним** елементом частково впорядкованої множини $X$ відносно строгого «<» (нестрогого «$\le$») впорядкування називають такий його елемент $a \in X$, для якого наявна одна із двох ситуацій:

- або $x < a$ ($x \le a$),
- або $a$ і $x$ – непорівнювані.

**Зауваження**

На одній і тій же множині можуть бути задані **різні відношення** порядку.

За одним з них множина може бути *лінійно впорядкованою*, а за іншим – *частково впорядкованою*.

Тоді за першим відношенням будемо говорити **про найбільший елемент**, а за другим – **про максимальний**.

**Приклад 1.94.** Розглянемо множину студентів групи
$A = \{Антонов, Борисенко, Вітренко, Загоруйко, Іванов, Яковенко, Повх\}$.
Задати різні відношення порядку на множині $A$.
*Розв'язок.*
1. Студенти групи упорядковані у алфавітному порядку за предикатом
$R_1 = \{(a,b) \mid якщо\ a \prec b\}$.
Це лінійне упорядкування, оскільки порівнюваними є всі елементи.

2. Студенти групи є частково упорядкованими за одержаними оцінками з дискретної математики за предикатом $R_2 = \left\{ a \in A_i \,\middle|\, якщо\ i \in \{1,2,3,4,5\} \right\}$.

Таке упорядкування є частковим, оскільки множину можна розбити на підмножини, наприклад, так:

$A_1 = \left\{ Антонов,\ Іванов \right\}$,

$A_2 = \left\{ Борисенко \right\}$,

$A_3 = \left\{ Вітренко \right\}$,

$A_4 = \left\{ Загоруйко \right\}$,

$A_5 = \left\{ Яковенко, Повх \right\}$.

### 1.4.13. Найменший і мінімальний елементи множини

**Визначення. Найменшим** елементом лінійно впорядкованої множини $X$ відносно строгого «<» (нестрогого «$\le$ ») впорядкування будемо називати такий елемент $a \in X$, що для всіх $x \in X$ вірно $a < x$ ($a \le x$).

**Визначення. Мінімальним** елементом частково впорядкованої множини $X$ відносно строгого «<» або нестрогого «$\le$ » впорядкування називають такий його елемент $a \in X$, для якого наявна одна із двох ситуацій:

- або $a < x$, ($a \le x$)
- або $a$ і $x$ – непорівнювані.

**Зауваження.** Якщо на множині існує найменший елемент, то він є єдиним мінімальним. Аналогічно, якщо на множині існує найбільший елемент, то він є єдиним максимальним.

**Приклад 1.95.** Розглянемо множину $X$ точок трикутника $OAB$ з наступним відношенням порядку: $(a,b) \le (c,d)$ тоді і тільки тоді, коли $a \le c$ і $b \le d$. Графічне представлення множини точок $X$ показано на рис. 1.39.

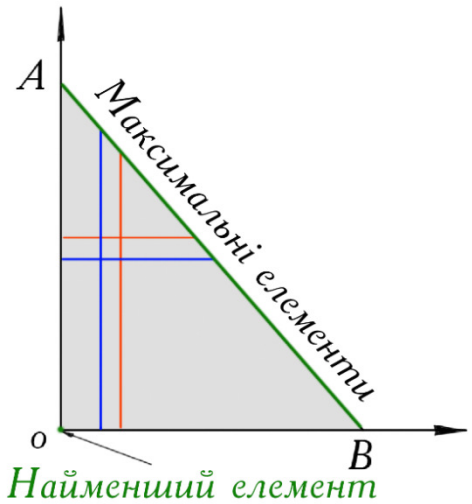



Рис. 1.39. Множина дійсних чисел, представлених координатами кортежів у трикутнику $OAB$

Визначити найбільший, найменший, мінімальний та максимальний елементи множини $X$.

*Розв'язок.*

Елементами множини $X$ є кортежі $(a,b)$, які задають координати точок на площині. Причому, елемент $a$ задає координату точки вздовж осі $OA$, а елемент $b$ – вздовж осі $OB$.

1. Точка з координатами (0,0) *є найменшим елементом* даної множини, оскільки всі координати точок множини $X$ порівнювані з кортежем $(0,0)$ та більші за нього.
2. *Мінімальний елемент* множини $X$ – єдиний і збігається з найменшим елементом, тобто елемент $(0,0)$ – це єдиний найменший елемент.
3. *Максимальними елементами* множини X є всі точки, що лежать на стороні AB трикутника OAB.
4. *Найбільший елемент* множини X не існує.

### 1.4.14. Верхні і нижні грані множин

*Верхня грань множини*

**Визначення.** Якщо $A$ є частково впорядкованою множиною і $B \subseteq A$, то елемент $a \in A$ називають **верхньою гранню** множини $B$, якщо для кожного $b \in B$ існує нерівність $b \le a$.

Елементи верхньої грані для множини $B$, що належать множині $A$, показані на рис. 1.40.


Рис. 1.40. Елементи верхньої грані множини $B$

**Приклад 1.96.** Розглянемо множину натуральних чисел $A = \{1,2,3,4,5,6,7,8,9\}$ та множину натуральних чисел $B = \{1,2,3,4,5\}$.

Визначити верхні грані для множини $B$ за умови, що $B \subseteq A$.

*Розв'язок.*

Верхніми гранями для множини $B$ є елементи множини $A$ за умови, що вони є більшими за елементи множини $B$ або дорівнюють найбільшому елементу множини $B$: 6, 7, 8, 9.

*Нижня грань множини*

**Визначення.** Якщо $A$ є частково впорядкованою множиною і $B \subseteq A$, то елемент $a \in A$ називають нижньою гранню множини $B$, якщо для кожного $b \in B$ існує нерівність $a \le b$.

Елементи нижньої грані для множини $B$, що належать множині $A$, показані на рис. 1.41.


Рис. 1.41. Елементи нижньої грані множини $B$

**Приклад 1.97.** Розглянемо множину натуральних чисел $A = \{1,2,3,4,5,6,7,8,9\}$ та множину натуральних чисел $B = \{5,6,7,8,9\}$.

Визначити нижні грані для множини $B$ за умови, що $B \subseteq A$.

*Розв'язок.*

Нижніми гранями для множини $B$ є елементи множини $A$ за умови, що вони є меншими за елементи множини $B$ або дорівнюють найменшому елементу множини $B$: 1,2,3,4.

*Точна верхня грань множини*

**Визначення.** Елемент $a \in A$ називають **точною верхньою гранню** множини $B$, якщо $a = \min\limits_i a_i$, де $a_i$ – довільна верхня грань множини $B$.

Точна верхня грань множини $B$, що належать множині $A$, показана на рис. 1.42.


Рис. 1.42. Точна верхня грань множини $B$

**Визначення.** Найменший елемент $a \in A$ множини всіх верхніх граней для множини $B$ називають точною верхньою гранню множини $B$ або *супремумом* і позначають як sup $B$.

**Приклад. 1.98.** Розглянемо множину натуральних чисел $A = \{1,2,3,4,5,6,7,8,9\}$ та множину натуральних чисел $B = \{3,4,5\}$.

Визначити точну верхню грань для множини $B$ за умови, що $B \subseteq A$.

*Розв'язок.*

Точною верхньою гранню для множини $B$ є елемент множини $A$ за умови, що він дорівнює найбільшому елементу множини $B$.

$$A = \{1,2,3,4,5,6,7,8,9\},\ B = \{3,4,5\},\ a = \min\{5,6,7,8,9\} = 5$$

*Іншими словами*, найменшою верхньою гранню є така верхня грань, яка є нижньою гранню множини всіх верхніх граней.

*Точна нижня грань множини*

**Визначення.** Елемент $a \in A$ називають **точною нижньою гранню**, якщо $a = \max\limits_i a_i$, де $a_i$ – довільна нижня грань множини $B$.

Точна нижня грань множини $B$, що належить множині $A$, показана на рис. 1.43.


Рис. 1.43. Точна нижня грань множини $B$

**Визначення.** Найбільший елемент $a \in A$ множини всіх нижніх граней для множини $B$ називають точною нижньою гранню множини $B$ або *інфімумом* і позначають як $\inf B$.

**Приклад. 1.99.** Розглянемо множину натуральних чисел $A = \{1,2,3,4,5,6,7,8,9\}$ та множину натуральних чисел $B = \{5,6,7\}$.

Визначити точну нижню грань для множини $B$ за умови, що $B \subseteq A$.

*Розв'язок.*

Точною нижньою гранню для множини $B$ є елемент множини $A$ за умови, що він дорівнює найменшому елементу множини $B$.

$$A = \{1,2,3,4,5,6,7,8,9\},\ B = \{5,6,7\},\ a = \max\{1,2,3,4,5\} = 5$$

$$\inf B = 5$$

*Іншими словами,* найбільшою нижньою гранню є така нижня грань, яка є верхньою гранню множини всіх нижніх граней.

**Приклад 1.100.** Розглянемо множину $A$ точок прямокутника з відношенням порядку $R = \left\{\left(\left(a,b\right),\left(c,d\right)\right)\middle| якщо \left(a,b\right) \le \left(c,d\right)\right\}$ на підмножині $B$. Знайти $\inf B$ та $\sup B$.

*Розв'язок.*

За правилами порівнювання кортежів умова $\left(a,b\right) \le \left(c,d\right)$ справедлива тоді і тільки тоді, коли $a \le c$ і $b \le d$. Тому точка $o$ є точною нижньою гранню $\inf B \in A$.

Точка $c$ є точною верхньою гранню $\sup B \in A$.

З рис. 1.44. видно, що точки $o$ і $c$ належать множинам $A$ і $B$.

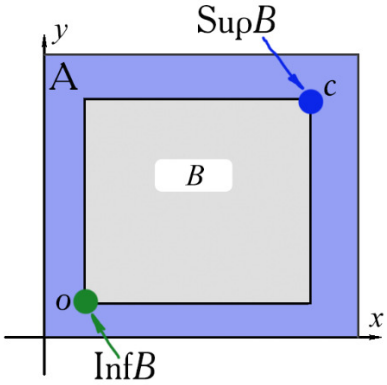



Рис. 1.44. Визначення точок $\sup B \in A$ та $\inf B \in A$

**Приклад 1.101.** Розглянемо множину $F \subset A$ точок трапеції $ABNM$ із заданим відношенням порядку: $\left(a,b\right) \le \left(c,d\right)$ тоді і тільки тоді, коли $a \le c$ і $b \le d$. Знайти $\inf F$ та $\sup F$ за умови взаємного розміщення множин, показаного на рис. 1.45.

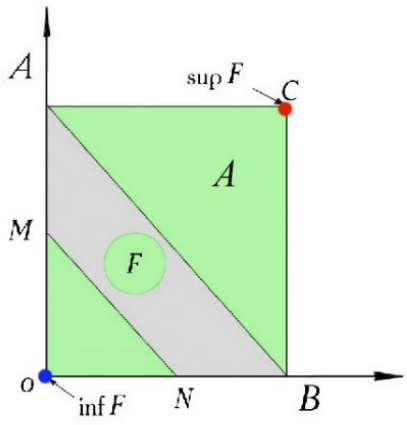



Рис. 1.45. Визначення точок $\sup F \in A$ та $\inf F \in A$

*Розв'язок.*

Виходячи з умови порівняння кортежів, можна зробити висновок, що існує точна верхня грань $\sup F \in A$, але $c \notin F$ і точна нижня грань $\inf F \in A$, але $o \notin F$.

## 4.15. Діаграма Хассе

Для графічного представлення впорядкованої множини $R$ використовують *діаграму Хассе.* Цю діаграму будують у такий спосіб.

Кожному елементу множини $X$ ставлять у відповідність точку (кружок) на площині, причому за умови існування $aRb$, точку $a$ розташовують нижче точки, яка відповідає елементу $b$. Точки $a \in X$ і $b \in X$ з'єднують лінією (ребром), якщо $aRb$ і не існує елемента $c \in X$ такого, що $aRc$ й $cRb$.

**Приклад 1.102.** Нехай дана множина $X = \{1,2,3,4,5,6,7,8\}$, на якій задано відношення

$$R = \{(1,2),(1,3),(1,4),(\mathbf{1,5}),(\mathbf{1,6}),(\mathbf{1,7}),(\mathbf{1,8}),$$
$$(2,5),(2,7),(\mathbf{2,8}),(3,5),(3,6),(\mathbf{3,8}),(4,6),(4,7),(\mathbf{4,8}),$$
$$(5,8),(6,8),(7,8)\}$$

Побудувати діаграму Хассе для даного відношення.

*Розв'язок.*

Діаграма Хассе даного відношення представлена на рис. 1.46.

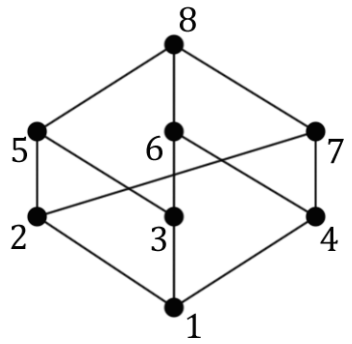



Рис. 1.46. Діаграма Хассе відношення $R$, заданого на множині $X$

**Приклад 1.103.** Нехай дано множину $C = \{x, y, z\}$ та $X$ – булеан множини $C$: $X = \{\varnothing, \{x\}, \{y\}, \{z\}, \{x, y\}, \{y, z\}, \{z, x\}, \{z, y, z\}\}$.

Визначимо відношення $R$ на $X$ наступним предикатом:

$R = \{(T, V) \mid T \subseteq V\}$. Наприклад, $(\{y\}, \{x, y\}) \in R$, оскільки $\{y\} \subseteq \{x, y\}$.

Однак $(\{y, z\}, \{z\}) \notin R$, оскільки $\{y, z\} \not\subset \{z\}$.

Побудувати діаграму Хассе для даного відношення $R$.

*Розв'язок.*

Побудувавши відношення $R$, можна легко перевірити, що $X$ – частково упорядкована множина на $R$. Тому діаграма Хассе є неповним графом, представленим на рис. 1.47.

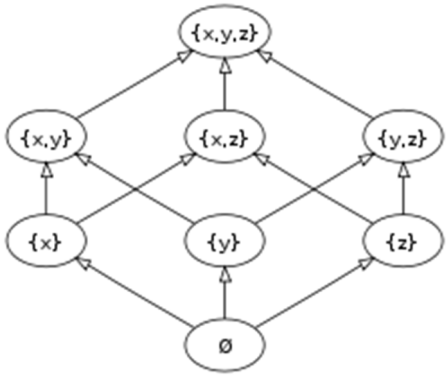



Рис. 1.47. Діаграма Хассе для булеана на множині $C$

**Приклад 1.104.** На рисунку представлений частковий порядок, породжений бінарним відношенням $R = \left\{(a_1, a_2), (a_1, a_3), (a_1, a_5), (a_4, a_2), (a_5, a_2)\right\}$ на множині $A = \left\{a_1, a_2, a_3, a_4, a_5\right\}$. Побудувати діаграму Хассе даного відношення.

*Розв'язок.*

Діаграма Хассе є неповним графом, представленим на рис. 1.48.

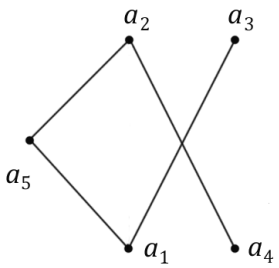



Рис. 1.48. Діаграма Хассе відношення $R$, заданого на множині $A$

Діаграма Хассе допомагає краще розуміти взаємозв'язок елементів, що належать одній і тій же впорядкованій множині (наприклад, приналежність одного і того ж ланцюга або одного і того ж антиланцюга).

**Контрольні запитання**

1. Поясніть відмінність відношення строгого порядку від відношення нестрогого порядку. Дайте приклади відношень строгого та нестрогого порядку.
2. Розглянемо відношення строгого порядку $R \subset A \times A$, яке задає предикатом $R = \left\{xRy \middle| \textit{якщо } y - x \le 3\right\}$ порядок елементів у множині $A = \left\{1, 2, 3, 4, 5, 6, 7, 8\right\}$. Визначити можливі ланцюги, які є покриттями.
3. Нехай дана множина $A$: $A = \left\{3, c, b, a, 2, k, 1, d, 4, 5, e\right\}$, на якій задані такі відношення часткового порядку: $R_1 = \left\{aRb \middle| a \le b\right\}$ та $R_2 = \left\{aRb \middle| \textit{"a слідує в алфавітному порядку за } b\textit{"}\right\}$. Побудувати МЛР на множині $A$.

4. Розглянемо упорядковані множини $A=\{1,2,3,4,5\}$ і $B=\{1,a,2,b,3,c\}$. Яка з цих множин має максимальні елементи, а яка характеризується найбільшим елементом? Чим відрізняється найбільший елемент від максимального елемента упорядкованої множини?

5. Розглянемо множину $A=\{а,б,в,г,д,е,ж,з\}$ та множину $B=\{в,г,д\}$. Визначити точну верхню та нижню грані для множини $B$ за умови, що $B\subseteq A$.

6. На рисунку представлений частковий порядок, породжений бінарним відношенням $R=\{(1,2),(1,3),(2,3),(4,5),(2,5),(3,4)\}$ на множині $A=\{1,2,3,4,5\}$. Побудувати діаграму Хассе даного відношення.

## 1.5. Функції та їхні властивості

### 1.5.1. Основні поняття про функцію

**Визначення. Функція** – математичне поняття, що відображає зв'язок між елементами множин. Можна сказати, що функція – це «закон», за яким кожному елементу однієї множини ставиться у відповідність деякий елемент іншої множини.

$$f=\{(x_1,y_1),(x_2,y_2),\ldots,(x_i,y_i),\ldots,(x_n,y_n),\}$$

Значення $y_i$ в кожній з пар $(x_i,y_i)\in f$ називають функцією від $x_i$, що у загальному випадку записується у вигляді $y=f(x)$.

Отже, функція – це множина, представлена у вигляді:

$$f=\{(x,y)\in X\times Y\,|\,y=f(x)\}.$$

*Відношення* $f$ на $X\times Y$ називають ***функцією*** з $X$ в $Y$ і позначають через $f:X\to Y$, якщо для кожного $x\in X$ існує єдиний елемент $y\in Y$ такий, що $(x,y)\in f$.

Якщо $f:X\to Y$ – функція, і $(x,y)\in f$, то говорять, що $y=f(x)$.

Як видно з визначення, символ $f$ використовують у двох значеннях:

1. $f$ – це множина, елементами якої є пари, які беруть участь у відношенні.
2. $f(x)$ – це позначення для $y\in Y$, яке відповідає даному $x\in X$.

*Область визначення й область значень. Образ множини*

Якщо задана функція $f:X\to Y$, то множину $X$ називають ***областю визначення*** функції $f$, а множину $Y$ називають ***областю потенційних значень або образом множини X***.

**Образ підмножини**

Нехай дана підмножина $E \subseteq X$. Образом підмножини $E$ називають множину всіх значень функції $f$ на всіх елементах підмножини $E$. Отже, образ підмножини $E$ позначається $f\left(E\right)$:

$f\left(E\right) = \left\{ f\left(x\right) \middle| x \in E \right\}$ або рівнозначно:

$f\left(E\right) = \left\{ y \in Y \middle| \left(x, y\right) \in f \text{ для деякого } x \in E \right\}$

**Образ елемента**

Елемент $f\left(x\right)$ називають **образом** елемента $x$.

**Визначення області значень через образ**

***Областю значень*** $Y$ функції $f$ називають образ множини $f\left(X\right)$ на усій множині $X$.

**Прообраз множини**. Прообразом **множини** $Y$ називають множину всіх елементів $x \in X$. Прообраз множини $Y$ позначають як $f^{-1}\left(Y\right)$

**Прообраз підмножини**. Прообразом **підмножини** $F \subseteq Y$ називають множину всіх елементів $x \in X$, для яких $f\left(x\right) \in F$. Прообраз позначають $f^{-1}\left(F\right)$:

$$f^{-1}\left(F\right) = \left\{ x \middle| f\left(x\right) \in F \right\}$$

**Елемент-прообраз**

Елемент $x$ називають **прообразом** $f\left(x\right)$.

## 1.5.2. Відображення, їхні властивості та види

Функцію $f : X \to Y$ називають також ***відображенням***; при цьому говорять, що $f$ відображає $X$ в $Y$.

Отже, *функція* та *відображення* – синоніми.

Однак термін «функція» частіше використовується для того, щоб вказати на відношення між елементами множин, а відображення – для визначення відношення між множинами.

*Властивості відображень множини*

**Властивість 1**. Якщо $A_1$ й $A_2$ – підмножини $X$, то образ об'єднання дорівнює об'єднанню образів:

$$f\left(A_1 \cup A_2\right) = f\left(A_1\right) \cup f\left(A_2\right).$$

**Приклад 1.105.** Розглянемо множини $A_1, A_2$ та їх образи $f\left(A_1\right), f\left(A_2\right)$:

$$\begin{cases} A_1 = \{1,2,3\}, f\left(A_1\right) = \{11,12,13\}, \\ A_2 = \{3,4,5\}, f\left(A_2\right) = \{13,14,5\} \end{cases}$$

Тоді об'єднання множин $A_3$ визначається відповідно до виразу:

$$A_3 = A_1 \cup A_2 = \{1,2,3\} \cup \{3,4,5\} = \{1,2,3,4,5\}.$$

Відповідно, об'єднання образів має вигляд:

$$f\left(A_3\right) = \{11,12,13,14,15\}$$

$$f\left(A_1\right) \cup f\left(A_2\right) = \{11,12,13\} \cup \{13,14,5\} = \{11,12,13,14,15\}$$

**Властивість 2**. Для взаємно-однозначного відображення образ перетину дорівнює перетину образів:

$$f\left(A_1 \cap A_2\right) = f\left(A_1\right) \cap f\left(A_2\right).$$

**Приклад 1.106.** Розглянемо множини $A_1, A_2$ та їх образи $f\left(A_1\right), f\left(A_2\right)$:

$$\begin{cases} A_1 = \{1,2,3\}, f\left(A_1\right) = \{11,12,13\}, \\ A_2 = \{2,3,4\}, f\left(A_2\right) = \{12,13,14\} \end{cases}$$

Перетин множин $A_3$ дорівнює:

$$A_3 = A_1 \cap A_2 = \{1,2,3\} \cap \{2,3,4\} = \{2,3\}$$

Перетин образів $f\left(A_3\right)$ має вигляд:

$$f\left(A_3\right) = \{12,13\}$$

$$f\left(A_1\right) \cap f\left(A_2\right) = \{11,12,13\} \cap \{12,13,14\} = \{12,13\}$$

**Властивість 3**. Для довільного образа відображення перетину входить у перетин образів:

$$f\left(A_1 \cap A_2\right) \subseteq f\left(A_1\right) \cap f\left(A_2\right)$$

**Узагальнення властивостей** 1 і 3:

$$f\left(\bigcup_{i=1}^{n} A_i\right) = \bigcup_{i=1}^{n} f\left(A_i\right), \quad f\left(\bigcap_{i=1}^{n} A_i\right) \subseteq \bigcap_{i=1}^{n} f\left(A_i\right).$$

*Композиція функцій*

Композицією двох функцій $f : A \to B$ і $g : B \to C$ називають функцію $h : A \to C$, задану співвідношенням

$$h\left(x\right) = g\left(f\left(x\right)\right)$$

Інакше кажучи, $h$ є множиною двійок $\left(a,c\right)$

$$h = \left\{\left(a,c\right) \middle| \left(a,b\right) \in f \text{ и } \left(b,c\right) \in g \text{ для деякого } b \in B\right\}$$

Композицію функцій позначають: $f \circ g$.

Нехай $f : A \to B$, $g : B \to C$ і $k : C \to D$

Композиція (як операція над функціями) *асоціативна*, тобто $f \circ \left(g \circ k\right) = \left(f \circ g\right) \circ k$.

Тому в композиції декількох функцій, які ідуть підряд, можна опускати дужки.

*Композиція відображень*

Нехай дані відображення $Q : X \to X$ і $G : X \to X$.

Композицією цих відображень називають відображення $Q \circ G$, обумовлене співвідношенням:

$$Q\left(G\right) = Q \circ G.$$

Дане співвідношення виражає відображення $Q$ відображення $G$.

У випадку, коли $Q = G$, можливо одержати відображення:

$$Q^2 = Q\left(Q\right), Q^3 = Q\left(Q^2\right), \ldots, Q_X^m = Q\left(Q_X^{m-1}\right) \quad .$$

Якщо $Q^0 = X$ то $Q^0 = Q\left(Q^{-1}\right) = X$.

Оскільки $Q^{-1}$ – зворотне відображення, то

$Q^{-1} = Q\left(Q^{-2}\right), Q^{-2} = Q\left(Q^{-3}\right)$, і т. ін.

**Приклад 1.107.** Нехай $X$ – множина людей. Для кожної людини $x$ із множини $X$ множину її дітей визначимо як $Q_X = Q\left(X\right)$.

Тоді $Q_X^2 = Q\left(Q\left(X\right)\right)$ буде представляти множину онуків,

$Q_X^3 = Q\left(Q\left(Q\left(X\right)\right)\right)$ – множина правнуків,

$Q_X^{-1} = Q^{-1}\left(X\right)$ – множина батьків.

Зобразимо множину людей точками на рис. 1.49, а стрілками представимо відповідності між $X, Q_X$, $Q_X^2$ і т. ін.

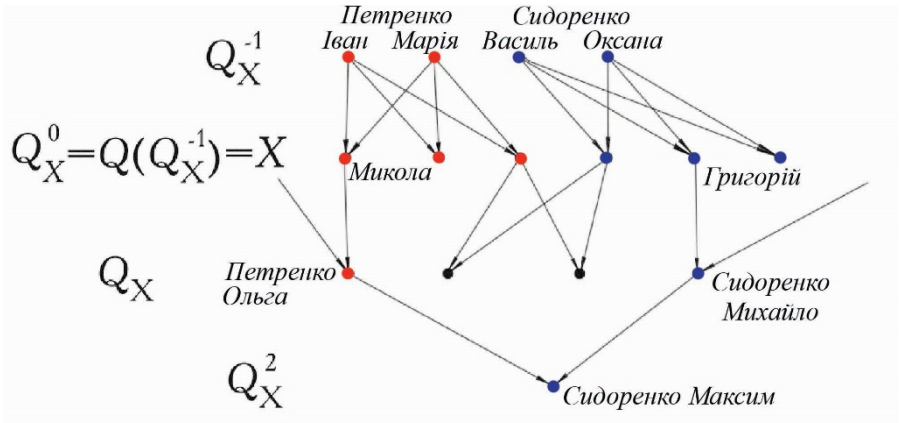



Рис. 1.49. Приклад генеалогічного дерева

Тоді одержуємо родовід або генеалогічне дерево для даної множини людей.

*Ін'єктивні відображення і функції*

**Визначення.** Відображення множини $X$ у множину $Y$ називають ін'єктивним, якщо образ $f\left(x\right)$ може мати лише один прообраз $x$.

Отже, наявною є *одно-однозначна* відповідність.

$$f : X \to Y, f = \left\{\left(1, D\right), \left(2, B\right), \left(3, A\right)\right\}$$

При цьому, не всі елементи $Y$ – образи.

**Визначення.** Функцію $f : X \to Y$ називають ***ін'єктивною***, або ***ін'єкцією***, якщо з $f\left(x\right) = f\left(x'\right)$ випливає $x = x'$.

Приклад ін'єктивного відображення показаний на рис. 1.50.

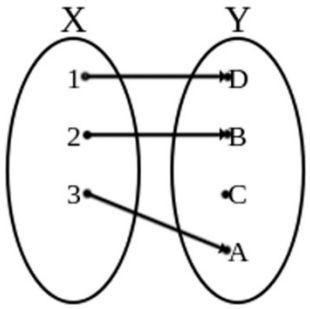



Рис. 1.50. Ін'єктивне відображення $f : X \to Y$

(Вікі) Ін'єкція (ін'єктивне відображення, ін'єктивна функція) – таке співвідношення між елементами двох множин, в якому двом різним елементам першої множини (області визначення) ніколи не співставляється один і той самий елемент другої множини (області значень).

**Приклад 1.108.** Нехай дано відображення $f : X \to Y$, $y = x^2$, де $X, Y \subseteq N$.

Визначити, чи є дане відображення ін'єктивним.

*Розв'язок.*

Розглянемо множину прообразів $X$. Кожному елементу множини $X$ (прообразу) відповідає елемент множини $Y$ (образ):

$$f\left(1\right) = 1,\ f\left(2\right) = 4,\ f\left(3\right) = 9,\ f\left(4\right) = 16, \ldots$$

Розглянемо множину образів $Y$. Не для всіх образів існують прообрази в даному відображенні. Для елементів множини $Y \subseteq N = \left\{3, 5, 6, 7, 8, 10, 11, \ldots\right\}$ немає прообразів. Отже, дане відображення ін'єктивне.

**Приклад 1.109.** Нехай дано відображення $f : X \to Y$, $y = x^2$, де $X, Y \subseteq Z$.

Визначити, чи є дане відображення ін'єктивним.

*Розв'язок.*

Розглянемо множину прообразів $X$. Двом різним елементам множини $X$ (прообразам) відповідає один і той же елемент множини $Y$ (образ):

$$f\left(-2\right) = f\left(2\right) = 4,\ f\left(-3\right) = f\left(3\right) = 9 \ldots$$

Відповідно до означення, таке відображення не може бути ін'єктивним.

*Сюр'єктивні відображення і функції*

**Визначення.** Відображення множини $X$ в множину $Y$ називають **сюр'єктивним,** якщо кожний елемент із $Y$ має принаймні один прообраз із $X$.

Отже, наявна багато-однозначна відповідність. Приклад сюр'єктивного відображення показаний на рис. 1.51.

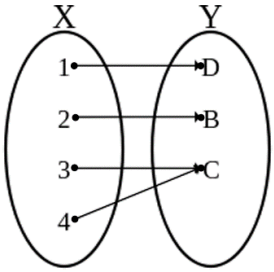



Рис. 1.51. Сюр'єктивне відображення $f : X \to Y$

Функцію $f : X \to Y$ називають сюр'єктивною функцією, або сюр'єкцією, якщо кожний елемент множини $Y$ є образом хоча б одного елемента множини $X$, тобто

$$\forall y \in Y \exists x \in X : y = f(x)$$

(Вікі) Сюр'єкція (сюр'єктивне відображення, сюр'єктивна функція) – співвідношення між двома множинами, в якій з кожним елементом другої множини асоціюється щонайменше один (або більше) елементів першої множини.

**Приклад 1.110.** Нехай дано відображення $f : X \to Y$, $y = \sin(x)$, де $X, Y \subseteq R$. Визначити, чи є дане відображення сюр'єктивним.

*Розв'язок.* Розглянемо множину прообразів $X$. Оскільки функція $\sin(x)$ є періодичною з періодом $2\pi$, то кожному прообразу відповідає більше, ніж один образ. Відповідно до означення таке відображення сюр'єктивне.

**Приклад 1.111.** Нехай дано відображення $f : X \to Y$, $y = \lfloor x \rfloor$ та $y = \lceil x \rceil$, де $X \subseteq R$, $Y \subseteq Z$. Визначити, чи є дане відображення сюр'єктивним.

*Розв'язок.*

Розглянемо множину прообразів $X$. Оскільки дані функції забезпечують округлення довільного дійсного числа до найближчого цілого, то кожному прообразу відповідає лише один образ. $\lceil 6.9 \rceil = 7$

Розглянемо множину образів $Y$. Для кожного образу існує один або більше прообразів у даному відображенні: $7 = \lceil 6.9 \rceil$, $7 = \lceil 6.9 \rceil$, $7 = \lceil 6.0001 \rceil$.

Отже, дане відображення – сюр'єктивне.

**Приклад 1.112.** Нехай дано відображення $f : X \to Y$, $y = \lfloor x^2 \rfloor$ де $X \subseteq R$, $Y \subseteq Z$. Визначити, чи є дане відображення сюр'єктивним.

*Розв'язок.*

Розглянемо множину прообразів $X$. Оскільки дані функції забезпечують округлення довільного дійсного числа до найближчого цілого, то кожному прообразу відповідає лише один образ. $\left\lfloor 6.9^2 \right\rfloor = 47$.

Розглянемо множину образів $Y$. Для від'ємних образів не існує прообразу $x \in R$ у даному відображенні. Отже, дане відображення не сюр'єктивне.

*Бієкція*

**Визначення.** Функцію, яка є одночасно ін'єктивною і сюр'єктивною, називають взаємно однозначною відповідністю, або бієкцією.

$f : X \to Y$

$f = \{(1, B), (2, C), (3, D), (4, G)\}$

Якщо $X = Y$ і $f : X \to X$ є взаємно однозначною відповідністю, то $f$ називають перестановкою множини $X$.

Приклад бієктивного відображення показаний на рис. 1.52.

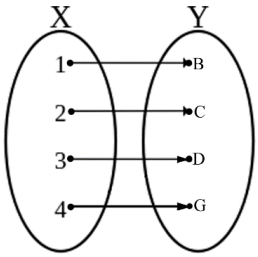



Рис 1.52. Бієктивне відображення $f : X \to Y$

(Вікі) Бієкція – відповідність, яка асоціює один елемент вхідної множини з одним і тільки одним елементом результуючої множини і навпаки, одному елементу результуючої множини співставляється один і лише один елемент вхідної множини.

**Приклад 1.113.** Нехай дано відображення $f : X \to Y$, $y = 2x + 1$, де $X, Y \subseteq R$. Визначити, чи є дане відображення бієктивним.

*Розв'язок.*

Ця функція є бієктивною, тому що для будь-якого $y \in R$ існує єдиний розв'язок рівняння $y = 2x + 1$ відносно $x$: $x = (y - 1)/2$.

**Приклад 1.114.** Нехай дано відображення $f : X \to Y$, $y = x^2$,де $X, Y \subseteq R$. Визначити, чи є дане відображення бієктивним.

*Розв'язок.*

Розглянемо множину прообразів $X$. Очевидно, що кожному елементу даної множини відповідає лише один прообраз.

Розглянемо множину образів $Y$. Кожному елементу даної множини відповідає два прообрази: від'ємний та додатний. Отже, дане відображення не бієктивне.

### 1.5.3. Способи задавання функцій

*Табличний спосіб задавання функції*

Функція може бути представлена у вигляді таблиці, рядки якої містять значення аргументів та відповідні значення відображення.

**Таблиця 1.8.**

Таблична функція

| $x$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $f(x)$ | 1 | 4 | 9 | 16 | 25 | 36 | 49 | 64 | 81 |

У даній таблиці стовпці є множиною впорядкованих пар:

$y = f(x) = \{(1,1),(2,4),(3,9),(4,16),(5,25),(6,36),(7,49),(8,64),(9,81)\}$

що відповідає визначенню функції, представленому раніше.

*Аналітичний спосіб задавання функції*

При аналітичному задаванні функція представлена у вигляді формули, тобто математичного виразу, що включає математичні операції, які необхідно виконати над $x \in X$, щоб одержати $y \in Y$:

$$y = f(x) = x^2$$

$$y = \left\{(x, y) \in R^2 \,\middle|\, y = x^2\right\}$$

*Графічний спосіб задавання функції*

Якщо $X \subseteq R$ і $Y \subseteq R$, тобто $X$ і $Y$ є підмножинами множини дійсних чисел, то пари $(x, y) \in R^2$ можливо представити у вигляді точок на площині. Повна сукупність точок буде графіком функції, як показано на рис. 1.53.

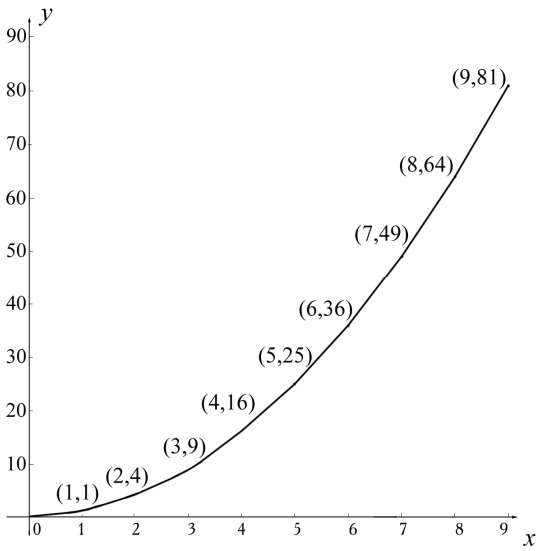



Рис. 1.53. Графік функції $y = x^2$

### 1.5.4. Спеціальні функції

Нехай $I : X \to X$ визначене співвідношенням $f\left(x\right) = x$ для всіх $x \in X$. Функцію $I$ називають **тотожною функцією** на $X$**.**

**Приклад 1.115.** Навести приклад тотожної функції.
*Розв'язок.* Тотожною є функція $y = 1 \cdot x$.

*Округлення до нижнього цілого*

Функцію $f : X \to Y$, де $X$ – множина дійсних чисел, а $Y$ – множина цілих чисел, називають **округленням до нижнього цілого** й позначають $f\left(x\right) = \left\lfloor x \right\rfloor$, якщо вона кожному $x \in X$ ставить у відповідність найбільше з цілих чисел, яке є меншим або дорівнює $x$.

**Приклад 1.116.** Навести приклади дії функції округлення до нижнього цілого для додатних і від'ємних прообразів.
*Розв'язок.*
Додатні образи:
$\left\lfloor 2,3 \right\rfloor = 2;\ \left\lfloor 3,899 \right\rfloor = 3;\ \left\lfloor 10 \right\rfloor = 10;$
Від'ємні образи:
$\left\lfloor -11,1 \right\rfloor = -12;\ \left\lfloor -10,99 \right\rfloor = -11$

*Округлення до верхнього цілого*

Функцію $f : F \to B$ називають округленням до верхнього цілого й позначають $f\left(x\right) = \left\lceil x \right\rceil$, якщо вона кожному $x \in X$ ставить у відповідність найменше з цілих чисел, яке більше або дорівнює $x$.

**Приклад 1.117.** Навести приклади дії функції **округлення до верхнього цілого для додатних і від'ємних прообразів.**
*Розв'язок.*
Для додатних образів: $\left\lceil 11,1 \right\rceil = 12;\ \left\lceil 45,9 \right\rceil = 46;\ \left\lceil 22 \right\rceil = 22;$
Для від'ємних образів: $\left\lceil -45 \right\rceil = -45\ \left\lceil -145,4 \right\rceil = -145.$

*Факторіал*

Нехай $X$ і $Y$ збігаються із множиною невід'ємних цілих чисел. **Факторіалом** назвемо функцію $f : X \to Y$, позначувану через $f\left(n\right) = n!$ і обумовлену наступними співвідношеннями:
$0!=1$
$1!=1$
$2! = 1 \cdot 2 = 2$
$k! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot k$

*Бінарна операція*

Нехай $X, Y, Z$ – трійка непустих множин. **Бінарною операцією** або **двомісною операцією** у парі $(x, y)$, $x \in X$ і $y \in Y$ зі значенням в $z \in Z$ називають функцію $b : P \to Z$, де $P \subset X \times Y$.

Бінарну операцію позначають знаком дії, який ставиться зазвичай між операндами.

Нехай $\bullet$ – довільна операція. Тоді існують види записів:

1. Інфіксна форма запису: $x \bullet y$. *Наприклад:* $x + y$
2. Префіксна форма запису: $\bullet xy$. *Наприклад:* $+xy$
3. Постфіксна форма запису: $xy \bullet$. *Наприклад:* $xy +$

**Приклад 1.118.** Приклади бінарних операцій на множині дійсних чисел: «+», «-», «$\times$»

*Послідовність*

Нехай дана множина $X = \{x_1, \ldots, x_i, \ldots, x_n\}$ довільної природи. Усяке відображення $f : N \to X$ множини натуральних чисел $N$ у множину $X$ називають **послідовністю** (елементів множини $X$).

Образ натурального числа $i$, а саме, елемент $x_i = f(i)$, називають $i$-м членом або **елементом послідовності**, а порядковий номер члена послідовності – її індексом.

***Позначення***

*Послідовність* $x_1, x_2, \ldots, x_i, \ldots$ *записують у вигляді* $\{x_i\}_{i=1}^{\infty}$, *якщо нестрогий порядок,* $(x_i)_{i=1}^{\infty}$ *якщо строгий.*

*Для скінченних послідовностей:* $(x_i)_{i=1}^{n}$ *або* $\{x_i\}_{i=1}^{n}$.

*Сума елементів послідовності:* $S = \sum_{i=1}^{n} x_i$.

**Приклад 1.119.** Побудувати послідовності, які задано предикатами.

*Розв'язок.*

1. Послідовність $\{x_i\}_{i=1}^{8} = \{x_i \mid x_i = i + 1,\ i = \overline{1,8}\}$

| $i \in N$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $x = i + 1$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

2. Послідовність $\{x_i\}_{i=1}^{8} = \{x_i \mid x_i = i^2,\ i = \overline{1,8}\}$

| $i \in N$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $x = i^2$ | 1 | 4 | 9 | 16 | 25 | 36 | 49 | 64 |

3. Послідовність $\left\{x_i\right\}_{i=1}^{8} = \left\{x_i \middle| x_i = \dfrac{i+2i}{2i-i},\ i = \overline{1,8}\right\}$

| $i \in N$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $x = \dfrac{i+2i}{2i-i}$ | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

### 1.5.5. Функція двох змінних

**Визначення.** Якщо **кожній парі** $(x, y)$ елементів деякої множини $D = X \times Y$ **відповідає єдиний елемент** $z \in Z$, а **кожному елементу** $z$ **відповідає хоча б одна пара** $(x, y)$, то ми говоримо, що $z$ є функція двох незалежних змінних $x$ і $y$, визначена в $D$.

Функція двох змінних $f : D \to Z$ є відображенням декартового добутку $D = X \times Y$ у множину $Z$.

Формальне визначення функції двох змінних має такий вид:

$$f = \left\{(x, y, z) \in X \times Y \times Z \middle| z = f(x, y)\right\}.$$

### 1.5.6. Матриці, операції над матрицями

Нехай є дві скінченні множини:

$$M = \{1, 2, \ldots, m\} \text{ і } N = \{1, 2, \ldots, n\},$$

де $m$ і $n$ – натуральні числа.

Функція

$$A : M \times N \to D$$

представляє матрицю розміру $m \times n$, або масив $m \times n$ ($m$ на $n$).

*Множина* $D$ – це, як правило, множина дійсних, комплексних, раціональних або цілих чисел.

Елементи $D$ називають **скалярами**.

Таким чином, для кожного $i, 1 < i < m$, і кожного $j, 1 < j < n$, є елемент $A(i, j) \in D$, який перебуває в $i$-му рядку і $j$-му стовпці відповідної прямокутної таблиці.

*Елемент матриці* $A(i, j)$ *представляє собою образ* елемента області визначення $(i, j)$ і скорочено позначається через $A_{i,j}$. Отже, $m \times n$ матриця $A$ зображується прямокутною таблицею, де образи впорядкованих пар $(i, j) \in \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$ можуть бути представлені в такому вигляді:

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & A_{m3} & \cdots & A_{mn} \end{bmatrix}$$

*Матриця $A$ містить $m$ рядків і $n$ стовпців і є матрицею розміру $m \times n$. Скорочено матрицю записують $A = \left[A_{ij}\right]$ або $A = \left[a_{ij}\right]$.*

Значення $a_{ij}$ називають **компонентом**, або **елементом** матриці А.

*Види матриць*

1. ***Матриця-стовпець.*** Матрицю розміром $m \times 1$ називають матрицею-***стовпцем або вектором-стовпцем***

$$A = \begin{bmatrix} a_{11} \\ a_{2,1} \\ \vdots \\ a_{m1} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$$

2. ***Матриця-рядок***. Матрицю розміром $1 \times n$ називають матрицею-рядком або ***вектором-рядком.***

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}$$

Якщо $A$ – матриця-рядок або матриця-стовпець, то індекс рядка або, відповідно, стовпця, зазвичай опускають.

3. ***Квадратна матриця.*** Якщо в матриці кількість рядків і кількість стовпців збігається: $m = n = k$, то її називають ***квадратною матрицею***.

$$A = \begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1k} \\ A_{12} & A_{22} & \ldots & A_{2k} \\ \ldots & \ldots & \ldots & \ldots \\ A_{k1} & A_{k2} & \ldots & A_{kk} \end{bmatrix}$$

4. ***Діагональна матриця.*** Це квадратна матриця, усі елементи якої, крім діагональних, нульові.

$$\forall\left(i \neq j\right) \Rightarrow A_{ij} = 0. \qquad A = diag\left(A_1, A_2, \ldots, A_k\right).$$

5. ***Одинична матриця.*** Це діагональна матриця з одиничними елементами на діагоналі.

$$\begin{cases} \forall\left(i \neq j\right) \Rightarrow A_{ij} = 0, \\ \forall\left(i = j\right) \Rightarrow A_{ij} = 1 \end{cases} \qquad A = diag\left(1, 1, \ldots, 1\right)$$

*Операції над матрицями*

**Рівність матриць**

Дві матриці $A=\left[A_{ij}\right]$ і $B=\left[B_{ij}\right]$ розміром $m\times n$ ***рівні***, якщо рівні їхні відповідні елементи; тобто $A=B$ тоді й тільки тоді, коли $A_{ij}=B_{ij}$ для всіх $i,\ 1<j<m$, і всіх $j,\ 1<j<n$.

**Множення матриці на скаляр**

Якщо $d$ – скаляр, а $A=\left[A_{ij}\right]$ – матриця $m\times n$, то $dA$ – це матриця $D=\left[D_{ij}\right]$ розміром $m\times n$, де $D_{ij}=dA_{ij}$, тобто кожний компонент є добуток відповідного компонента $A$ на $d$. Добуток числа $d$ й матриці $A$ називають множенням матриці на скаляр.

**Сума і різниця матриць**

*Додавати і віднімати можна тільки матриці одного розміру !!*

**Сума**

Якщо $A=\left[A_{ij}\right]$ і $B=\left[B_{ij}\right]$ – $m\times n$-матриці, тоді $A+B$ є $m\times n$ матрицею $C=\left[C_{ij}\right]$, де $C_{ij}=A_{ij}+B_{ij}$, інакше кажучи, матриці додаються ***покомпонентно***. Матрицю $C$ називають **сумою матриць** $A$ і $B$.

**Різниця**

Різницю двох матриць визначимо через їх суму.

Запис $A-B$ означає $A+\left(-1\right)\cdot B$.

Отже, якщо $A=\left[A_{ij}\right]$ й $B=\left[B_{ij}\right]$ – $m\times n$-матриці, тоді $A-B$ є $m\times n$-матриця $C=\left[C_{ij}\right]$, де $C_{ij}=A_{ij}-B_{ij}$.

**Добуток матриць**

Добуток матриць $A\cdot B$ – це операція обчислення такої матриці $C$, кожний елемент якої дорівнює сумі добутків у відповідному рядку першого співмножника та стовпці другого: $c_{ij}=\sum_{k=1}^{n}a_{ik}b_{kj}$.

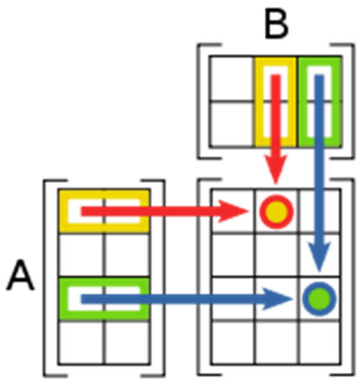



Рис. 1.54. Схема множення матриць

$$c_{12} = a_{11}b_{12} + a_{12}b_{22}$$
$$c_{33} = a_{31}b_{13} + a_{32}b_{23}$$

1. **Множення матриці на матрицю-стовпець**

Матриця повинна бути ліворуч, а матриця-стовпець – праворуч:

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots \\ A_{m1} & A_{m1} & \dots & A_{mn} \end{bmatrix} \times \begin{bmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{bmatrix} = \begin{bmatrix} A_{11}B_1 + A_{12}B_2 + \dots A_{1n}B_n \\ A_{21}B_1 + A_{22}B_2 + \dots A_{2n}B_n \\ \dots \\ A_{m1}B_1 + A_{m2}B_2 + \dots A_{mn}B_n \end{bmatrix}$$

2. **Множення матриці-рядка на матрицю**

Матриця-рядок повинна бути ліворуч, а матриця-праворуч:

$$\begin{bmatrix} A_1 & A_2 \dots A_m \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \dots & \dots & \dots & \dots \\ B_{m1} & B_{m1} & \dots & B_{mn} \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{m} A_k B_{k1}, & \sum_{k=1}^{m} A_k B_{k2}, & \dots, & \sum_{k=1}^{m} A_k B_{kn} \end{bmatrix}$$

Нехай $A$ матриця $m \times p : A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1p} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & A_{m3} & \cdots & A_{mp} \end{bmatrix}$

Нехай $B$ матриця $p \times n :$ $B = \begin{bmatrix} B_{11} & B_{12} & B_{13} & \cdots & B_{1n} \\ B_{21} & B_{22} & B_{23} & \cdots & B_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ B_{p1} & B_{p2} & B_{p3} & \cdots & B_{pn} \end{bmatrix}$

Тоді добутком матриць $A$ і $B$ називають матрицю $C = \left[C_{ij}\right]$ розміром $m \times n$, де $C_{ij}$ – це скалярний добуток $i$-го рядка матриці $A$ на $j$-й стовпець матриці $B$.

$C=AB$

$$C_{i,j} = \begin{bmatrix} A_{i1} & A_{i2} & A_{i3} & \cdots & A_{ip} \end{bmatrix} \bullet \begin{bmatrix} B_{1j} \\ B_{2j} \\ B_{3j} \\ \vdots \\ B_{pj} \end{bmatrix} = \sum_{k=1}^{p} A_{ik} B_{kj}$$

**Транспонована матриця**

Нехай $A$ – матриця $m \times n$.

Її **транспонованою матрицею** називають матрицю $A^t$ розміром $n \times m$ таку, що $A_{ij}^t = A_{ji}$, де $A_{ij}$ – елемент $i$-го рядка і $j$-го стовпця матриці $A$.

$$\begin{pmatrix} a & 1 \\ b & 2 \\ c & 3 \end{pmatrix}^t = \begin{pmatrix} a & b & c \\ 1 & 2 & 3 \end{pmatrix}$$

**Симетрична матриця**

Якщо $A$ – матриця $n \times n$ і $A_{ij} = A_{ji}$ для всіх $1 \le i,\ j \le n,$ то матрицю $A$ називають **симетричною.** Іншими словами, матриця $A$ симетрична тоді й тільки тоді, коли $A = A^t$.

*Матричне представлення відношень*

Нехай $A = \{a_1, a_2, a_3, \ldots, a_m\}$ і $B = \{b_1, b_2, b_3, \ldots, b_n\}$, і нехай $R$ – відношення на $A \times B$**.**

**Матричним представленням** $R$ називають матрицю $M = \left[M_{ij}\right]$ розміром $m \times n$, елементи якої визначають із співвідношення

$$M_{ij} = \begin{cases} 1, & (a_i, b_j) \in R, \\ 0, & (a_i, b_j) \notin R. \end{cases}$$

**Приклад 1.120.** Нехай $A = \{a_1, a_2, a_3, a_4\};\ B = (b_1, b_2, b_3, b_4),\ m = 4, n = 4$

Побудувати матрицю відношення $M$, якщо

$R = \{(a_1, b_1), (a_2, b_1), (a_2, b_3), (a_3, b_2), (a_4, b_4)\}$

*Розв'язок.*

Матриця відношення $M = \begin{pmatrix} & b_1 & b_2 & b_3 & b_4 \\ a_1 & 1 & 0 & 0 & 0 \\ a_2 & 1 & 0 & 1 & 0 \\ a_3 & 0 & 1 & 0 & 0 \\ a_4 & 0 & 0 & 0 & 1 \end{pmatrix}$

*Матриця перестановок*

Нехай $M$ – матриця розміром $n \times n$, у кожному рядку і у кожному стовпці якої тільки один елемент, який дорівнює 1, а всі інші дорівнюють 0. Таку матрицю $M$ називають матрицею перестановок.

$$M_{ij} = \begin{cases} 1, & \left(a_i, b_j\right) \in R, \\ 0, & \left(a_i, b_j\right) \notin R. \end{cases}$$

**Приклад 1.121.** Нехай дана перестановка 4-го порядку

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}.$$

Побудувати матрицю перестановок.

*Розв'язок.*

Відповідна матриця перестановок має вигляд:

$$P = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Множення перестановочної матриці на довільну міняє місцями рядки в довільній матриці.

Множення довільної матриці на перестановочну міняє місцями стовпці в довільній матриці.

### 1.5.7. Поняття функціонала

Поняття функціонала є більш широким, ніж поняття функції.

**Функція**

Функція у загальному випадку : $f : X \to Y$, де

$X -$ множина дійсних чисел.

$Y -$ множина дійсних чисел.

Кожна пара $\left(x, y\right) \in f$ ставить у відповідність одному дійсному числу $x$ інше дійсне число $y$.

**Функціонал**

Функція у загальному випадку : $F : \left\{f\left(x\right)\right\} \to Y$, де

$\left\{f\left(x\right)\right\} -$ множина функцій.

$Y -$ множина дійсних чисел.

Уявимо собі деякий набір кривих (траєкторій) $y = f_i\left(x\right)$, що з'єднують фіксовані точки А и В, як показано на рис. 1.55.

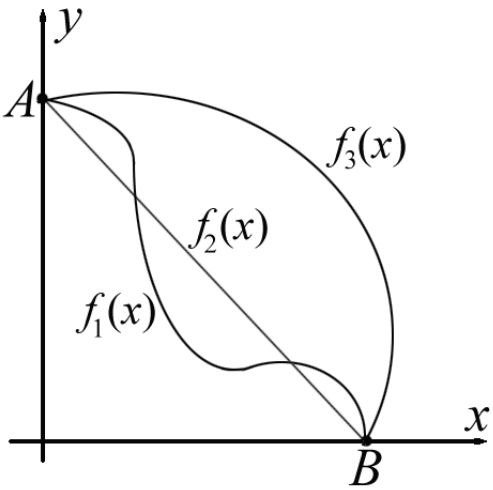


Рис. 1.55. Траєкторії руху

Нехай по кожній із цих траєкторій може відбуватися вільне переміщення точки. Позначимо через $t$ час, потрібний на переміщення із точки $A$ в точку $B$. Цей час очевидно залежить від характеру траєкторії $AB$, тобто від виду функції $f_i\left(x\right)$.

Позначимо через $F\left(x\right)$ множину з $n$ різних функцій, що зображують траєкторію $AB$,

$$F\left(x\right)=\left\{f_1\left(x\right),f_2\left(x\right),...,f_i\left(x\right),...,f_n\left(x\right)\right\},$$

а через $T-$ множину дійсних чисел $t\in T$, що визначають час переміщення точки, тоді залежність часу руху від виду функції може бути записана як відображення.
**Функціонал** – це відображення $J$, що має таке формальне представлення:

$$J:F\left(x\right)\to T,$$

або $J=\left\{\left(f\left(x\right),t\right)\middle|f\left(x\right)\in F\left(x\right),t\in T,t=J\left[f\left(x\right)\right]\right\}.$

### 1.5.8. Поняття оператора

**Поняття оператора.** Оператор представляє більш загальне поняття в порівнянні з функціоналом. Оператор у загальному випадку: $L:X\to Y$, де

$X=\left\{x\left(t\right)\middle|t-\textit{аргумент функції}\right\}$ – множина функцій,

$Y=\left\{y\left(t\right)\middle|t-\textit{аргумент функції}\right\}$ – множина функцій.

$x\left(t\right)\in X$ і $y\left(t\right)\in Y$ – функції, що є елементами множин.

Отже, елементами множини $L$ є пари$\left(x\left(t\right),y\left(t\right)\right)$, а оператор $L$ перетворить функцію $x\left(t\right)$ у функцію $y\left(t\right)$. $y\left(t\right)=L\left[x\left(t\right)\right]$.

Таким чином, оператор встановлює відповідність між двома множинами функцій так, що кожній функції з однієї множини відповідає функція з іншої множини.

**Приклад 1.122.** Позначимо через $p$ оператор диференціювання. Тоді зв'язок між похідною $f'\left(x\right)=\dfrac{df\left(x\right)}{dx}$ і функцією $f\left(x\right)$ може бути представлений в операторному вигляді: $f'\left(x\right)=p\left[f\left(x\right)\right]$.

**Контрольні запитання**

1. Розглянемо функцію $f:X\to Y$, $X,Y\subset R$, яка задана виразом $y=\sqrt{x}$ за умови, що $x\in X$ і $X=\left\{1,4,9,16,25\right\}$. Визначити область визначення та область значень даної функції.
2. Розглянемо функцію $f:X\to Y$, $X,Y\subset R$, яка задана виразом $y=\left|x+2x^3\right|$. Визначити, чи є дана функція ін'єктивною, сюр'єктивною або бієктивною.
3. Розглянемо множини $A_1,A_2$ та їхні образи $f\left(A_1\right),f\left(A_2\right)$:

$$\begin{cases}A_1=\left\{a,b,c\right\},f\left(A_1\right)=\left\{11,12,13\right\},\\ A_2=\left\{\alpha,\beta,\gamma\right\},f\left(A_2\right)=\left\{13,14,5\right\}\end{cases}$$

Визначити об'єднання та перетин образів та прообразів.
4. Розглянемо функцію $f:X\to Y$, яка задана відношенням $f=\left\{\left(1,B\right),\left(2,C\right),\left(3,D\right),\left(4,G\right),\left(5,B\right)\right\}$. Визначити, чи є дана функція бієктивною. Довести своє твердження.
5. Побудувати генеалогічне дерево своєї родини. Записати відповідні прямі та зворотні відображення на множині батьків, онуків та правнуків.
6. Нехай $A=\left\{1,2,3,4\right\}$; $B=\left(10,9,8,7\right)$, $m=4,n=4$. Побудувати матрицю відношення $M$, якщо $R=\left\{\left(1,10\right),\left(2,10\right),\left(2,8\right),\left(3,8\right),\left(4,7\right)\right\}$.

# Розділ 2. Комбінаторика

## 2.1. Теоретичні основи комбінаторики

### 2.1.1. Вступ в комбінаторику

Термін «***комбінаторика***» походить від латинського слова «combina», що українською означає – «поєднувати», «з'єднувати».
Термін ***«комбінаторика»*** був застосований у математиці вперше Лейбницем, який у 1666 році опублікував свою працю «Міркування про комбінаторне мистецтво».

**Готфрид Вільгельм фон Лейбниц**
*Gottfried Wilhelm Leibniz.*
(1.07.1646, Лейпциг – 14.11.1716, Ганновер)
німецький філософ, математик, юрист, дипломат

**Комбінаторика сьогодні**

*Комбінаторика* або *комбінаторний аналіз* або *комбінаторна математика* – це галузь математики, яка вивчає **способи побудови підмножин** деякої скінченної множини, причому таких, які відповідають заданим обмеженням.

Згадані підмножини часто називають **комбінаторними конфігураціями** або **вибірками**. Якщо, наприклад, враховувати порядок розміщення об'єктів, то одержимо різні комбінаторні конфігурації, як показано на рис. 2.1.

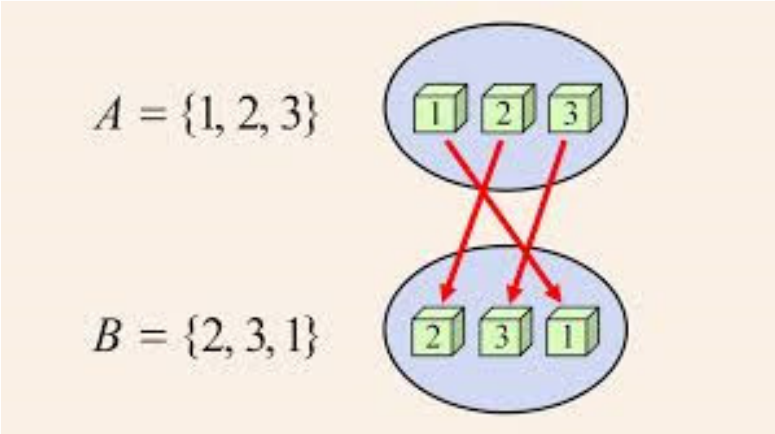



Рис. 2.1. Формування комбінаторної конфігурації

**Комбінаторні методи** лежать в основі розв'язку багатьох задач теорії ймовірностей і її додатків.

**Комбінаторика вивчає такі види задач:**

1. *Підрахунок числа комбінаторних конфігурацій.*

**Приклад 2.1.** При зміні порядку кольорових кульок у стовпцях одержимо скінченну кількість різних стовпців. Кожний стовпець відповідатиме одній комбінаторній конфігурації, як показано на рис. 2.2.

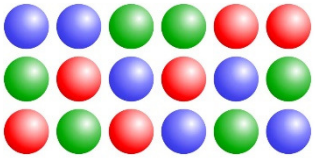


Рис. 2.2. Задача визначення кількості відмінних стовпців кульок

2. *Знаходження умов існування комбінаторної конфігурації.*

**Приклад 2.2.** Задача знаходження умов існування комбінаторної комбінації виникає при визначенні можливості існування певного розфарбування кубика Рубіка (рис. 2.3)

Рис. 2.3. Задача існування заданого розфарбування

3. *Розробка алгоритмів побудови комбінаторних конфігурацій*.

**Приклад 2.3.** Алгоритм побудови транспортних маршрутів, які відповідають заданим параметрам трафіку, дозволяє гнучко розподіляти транспортні потоки.


Рис. 2.4. Задача перерозподілу транспортних потоків

4. *Розв'язування оптимізаційних задач* (*екстремальних комбінаторних задач*).

**Приклад 2.4.** Перебір та порівняння варіантів вирішення задачі є одним із способів розв'язування оптимізаційної задачі. Така задача може виникнути при оптимізації WEB-сайту.

Рис. 2.5. Задача оптимізації технічного рішення

Підрахунок кількості комбінаторних конфігурацій часто зустрічається в програмних засобах. Такі задачі є предметом вивчення *рахункової комбінаторики.*

### 2.1.2. Основні поняття комбінаторики

У комбінаториці прийнято говорити про множину, вказуючи кількість її елементів. Наприклад, якщо є множина $A = \{a_1, a_2, a_3, ..., a_n\}$, яка складається з $n$ елементів, то в цьому випадку її називають $n$-множиною $A$.

**Визначення.** Множину $B$ називають *підмножиною* множини $A$ і позначають $B \subset A$, якщо всі елементи множини $B$ є також елементами множини $A$.

**Визначення.** Якщо множина $C$ має кілька екземплярів одного і того самого елемента, то таку множину називають *мультимножиною.*

*Вибірка*

**Визначення.** Вибіркою називають довільну мультимножину, елементи якої вибирають з елементів множини A, тобто таку множину, яка, у загальному випадку, може містити кілька екземплярів одного і того самого елемента множини A, як показано на рис. 2.3.

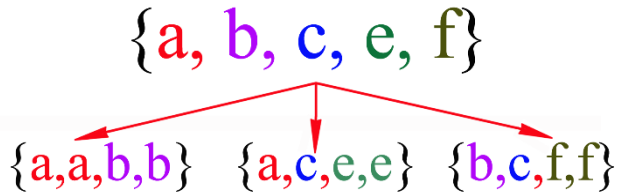



Рис. 2.3. Приклади 4-вибірок з 5-множини $A$

**Обсяг вибірки.** Кількість елементів $r$ у вибірці (таку вибірку називають також $r$-вибіркою) визначають як її ***обсяг.***

**Інший зміст поняття «вибірка».** Поняття «вибірка» використовують також для позначення **самого процесу відбору** елементів підмножини з початкової множини.

*Упорядкована вибірка*

**Визначення.** Вибірку називають *упорядкованою,* якщо порядок слідування елементів в ній заданий. Дві впорядковані вибірки, які різняться лише порядком проходження елементів, вважають різними.

**Визначення.** Нехай $A = \{a_1, a_2, ..., a_n\}$ – множина з $n$ елементів. Упорядкованою вибіркою обсягом $r$ з $n$ – множини $A$ називають будь-яку впорядковану підмножину з $r$ її елементів, як показано на рис. 2.4.

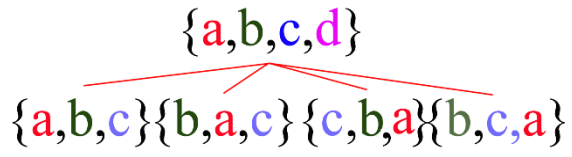



Рис. 2.4. Упорядковані 3-вибірки з 4-множини

**Висновок.** Упорядковані вибірки відрізняються порядком слідування одних і тих самих елементів.

*Неупорядкована вибірка*

Якщо порядок слідування елементів не є істотним, то таку вибірку називають *неупорядкованою*. Приклад такої вибірки показаний на рис. 2.5.

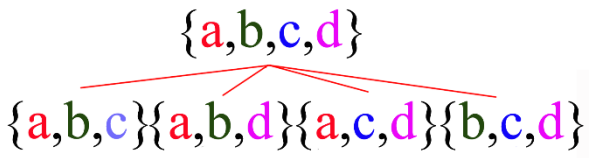



Рис. 2.5. Приклад неупорядкованої 3-вибірки з 4-множини

**Висновок.** Неупорядковані вибірки відрізняються елементами, а не порядком їх слідування.

*Вибірки з повтореннями та без повторень*

**Визначення.** *Вибірки з повтореннями* – це вибірки, які допускають повторення елементів. Приклад вибірки з повтореннями показаний на рис. 2.6.

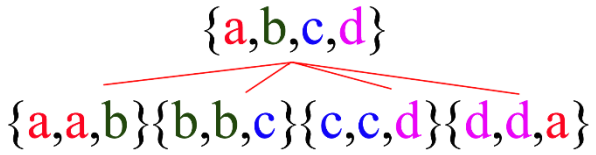



Рис. 2.6. Приклад 3-вибірок з повтореннями з 4-множини

**Визначення.** *Вибірки без повторень* – це вибірки, які не допускають повторення елементів. Приклад вибірки без повторень показаний на рис. 2.7.

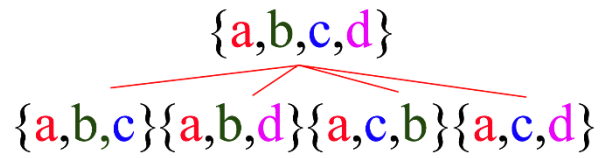



Рис. 2.7. Приклад 3-вибірок без повторень з 4-множини

*Загальноприйняті назви вибірок*

**1. Розміщення** ($\widehat{A}_n^k$ з повторенням, $A_n^k$ – без повторень)
(упорядкована вибірка з повтореннями або без повторень)

Набір елементів $\left\{x_{i_1}, x_{i_{12}}, \ldots, x_{i_k}\right\}$ з множини $X = \left\{x_1, x_2, \ldots, x_n\right\}$ називають $i-$ю вибіркою обсягом $k$ з $n$ елементів $k < n$ або, інакше, $\left(n,k\right)-$ розміщенням.

Наприклад: $\left\{1,2,3,4\right\} \Rightarrow \left\{1,1,2\right\}, \left\{2,1,1\right\}, \left\{1,2,3\right\} \ldots$

$\left\{1,2,3,4\right\} \Rightarrow \left\{1,2,3\right\}, \left\{1,2,4\right\}, \left\{4,2,1\right\} \ldots$

**2. Сполука** ($C_n^k$ без повторень, $\widehat{C}_n^k$ з повтореннями)
(неупорядкована вибірка з повтореннями або без повторень).

Вибірки, у яких не враховуються порядок запису елементів і які відрізняються між собою хоча б одним елементом, називають *сполуками або комбінаціями.*

Наприклад: $\left\{1,2,3,4\right\} \Rightarrow \left\{1,2,3\right\}, \left\{1,2,4\right\}, \left\{2,3,4\right\} \ldots$

$\left\{1,2,3,4\right\} \Rightarrow \left\{1,1,3\right\}, \left\{1,1,4\right\}, \left\{4,2,1\right\} \ldots$

**3. Перестановка** ($P_n$ без повторень, $P\left(k_1, \ldots, k_m\right)$ з повтореннями)
(упорядкована повна вибірка без повторень та з повтореннями)

Вибірки, у які складаються з одних і тих самих елементів і які відрізняються тільки порядком їх слідування, називають *перестановками.*

Наприклад: $\left\{1,2,3\right\} \Rightarrow \left\{1,2,3\right\}, \left\{1,3,2\right\}, \left\{3,2,1\right\} \ldots$

$\left\{1,2,3\right\} \Rightarrow \left\{1,1,3\right\}, \left\{3,1,1\right\}, \left\{3,3,1\right\} \ldots$

### 2.1.3. Основні правила комбінаторики

Розглянемо два основні правила, які використовують при розв'язуванні комбінаторних задач.

*Правило суми*

**Визначення.** Якщо елементи множини $A$ можна вибрати $n$ способами, а елементи множини $B$ можна вибрати $m$ способами, то за умови, що $A \cap B = \varnothing$, загальне число вибірок становить $n + m$.

На рисунку 2.8. показано умови застосування правила суми, у випадку, коли множини не перетинаються.

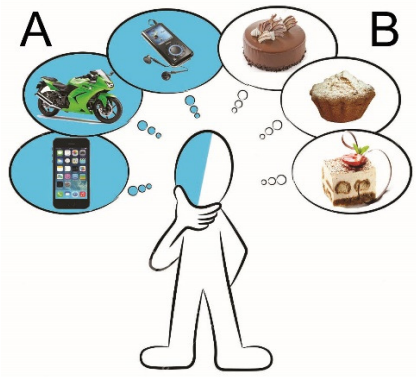



Рис. 2.8. Демонстрація правила суми

**Приклад 2.5.** На лекції з дискретної математики присутні 20 студентів. Квитки на концерт Лари Фабіан купило 15 студентів. Скільки всього студентів відвідали лекцію й концерт за умови, що вони відбуваються одночасно?

*Розв'язок.* Позначимо через $X$ множину студентів, які були присутні на лекції, а через $Y$ – множину студентів, які відвідали концерт. Оскільки $n = |X| = 20$, $m = |Y| = 15$ і $X \cap Y = \varnothing$, то за правилом суми: $m + n = 20 + 15 = 35$.

*Правило добутку*

**Визначення.** Кількість способів вибору елементів множини $A \cdot B$ дорівнює $n \cdot m$.

**Приклад 2.6.** Космонавт, що працює на орбітальній станції, може зв'язатися із центром керування двома способами: за допомогою радіозв'язку і передачі повідомлення космічним човником.

У той же час працівники центру керування польотом можуть подзвонити рідним космонавта по провідному телефону, по мобільному телефону, послати їм лист поштою, послати електронний лист, подзвонити по Skype, послати sms.

Скількома способами може потрапити інформація від космонавта його рідним?

*Розв'язок.* $|m| = 2$, $|n| = 6$.

Використовуючи правило множення, одержуємо $m \cdot n = 2 \cdot 6 = 12$.

*Правило включень і виключень для двох множин*

**Визначення.** Нехай $A$ і $B$ – скінченні множини. Визначимо, чому дорівнює $|A \cup B|$ , якщо відомі потужності $|A|$ і $|B|$.

Дотримуючись визначення операції об'єднання*:*

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

**Пояснення.** Сума $|A| + |B|$ включає всі елементи множини $A$ й множини $B$. При цьому, загальні елементи множин $A$ і $B$, а їх буде $|A \cap B|$, включаються в суму двічі. Тому для одержання суми об'єднання необхідно відняти один набір спільних елементів. В результаті одержуємо:
$|A \cup B| = |A| + |B| - |A \cap B|$

*Правило включень і виключень для трьох множин*

$$|A \cup B \cup C| = |A \cup (B \cup C)| = |A| + |B \cup C| - |A \cap (B \cup C)| =$$

| Застосували правило включень і виключень для множин $A$ і $(B \cup C)$ |
|---|

$$= |A| + |B| + |C| - |B \cap C| - |A \cap (B \cup C)| =$$

| Застосували правило включень і виключень для множин $(B \cup C)$ |
|---|

$$= |A| + |B| + |C| - |B \cap C| - |(A \cap B) \cup (A \cap C)| =$$

| Застосували дистрибутивний закон |
|---|

$$= |A| + |B| + |C| - |B \cap C| - (|A \cap B| + |A \cap C| - |(A \cap B) \cap (A \cap C)|) =$$

| Правило включень і виключень для множин $(A \cap B)$ і $(A \cap C)$ |
|---|

$$= |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

| Розкрили дужки |
|---|

*Правило включень і виключень у загальному вигляді*

Розглянемо правило включень і виключень при застосуванні до $n$ множин.

Нехай маємо $A_1, A_2, A_3, ..., A_i, ..., A_n$ – деякі множини. Тоді формула для визначення потужності множини об'єднання даних множин має такий вигляд:

$$\left|A_1 \cup A_2 \cup A_2 \cup ... \cup A_n\right| =$$

$$= \sum_{i=1}^{n} \left|A_i\right| \; -$$

$$- \sum_{1 \le i < j \le n} \left|A_i \cap A_j\right| +$$

$$+ \sum_{1 \le i < j < k \le n} \left|A_i \cap A_j \cap A_k\right| - ...$$

$$+ (-1)^{n-1} \cdot \sum_{1 \le i < j < k < ... < l \le n} \left|A_i \cap A_j \cap ... A_k \cap ... A_l \cap A_n\right|$$

Правило підрахунку за даною формулою полягає в послідовному виконанні **операцій додавання і віднімання**, які чергуються між собою.
Звідси випливає назва: **правило включень і виключень.**

**Приклад 2.7.** Обчислення за правилом включень і виключень
Нехай дані множини

$$A = \{1,2,3,4,9\},\ B = \{3,4,5,6,9\} \text{ і } C = \{5,6,7,8,9\}.$$

Обчислити *1)* $\left|A \cup B\right|$ *2)* $\left|B \cup C\right|$ *3)* $\left|A \cup C\right|$ *4)* $\left|A \cup B \cup C\right|$.

*Розв'язок.* Попередньо обчислимо об'єднання трьох множин

$$A \cup B \cup C = \{1,2,3,4,5,6,7,8,9\}$$

*1)* $A \cap B = \{3,4,9\}$, $\left|A \cap B\right| = 3$.

Тому $\left|A \cup B\right| = \left|A\right| + \left|B\right| - \left|A \cap B\right| = 5 + 5 - 3 = 7$

*2)* $B \cap C = \{5,6,9\}$, $\left|B \cap C\right| = 3$.

Тому $\left|B \cup C\right| = \left|B\right| + \left|C\right| - \left|B \cap C\right| = 5 + 5 - 3 = 7$

*3)* $A \cap C = \{9\}$, $\left|A \cap C\right| = 1$.

Тому $\left|A \cup C\right| = \left|A\right| + \left|C\right| = 5 + 5 - 1 = 9$

*4)* $(A \cap B \cap C) = \{9\}$, $\left|A \cap B \cap C\right| = 1$

$$\left|A \cup B \cup C\right| = \left|A\right| + \left|B\right| + \left|C\right| - \left|A \cap B\right| - \left|A \cap C\right| - \left|B \cap C\right| + \left|A \cap B \cap C\right| =$$

$$\left|A \cup B \cup C\right| = 5 + 5 + 5 - 3 - 1 - 3 + 1 = 9$$

**Приклад 2.8.** Записати правило включень і виключень для множин $A, B, C$ і $D$.

*Розв'язок.* Скористаємося загальною формулою:

$$\left|A_1 \cup A_2 \cup A_2 \cup \ldots \cup A_n\right| = \sum_{i=1}^{n}\left|A_i\right| - \sum_{1\le i<j\le n}\left|A_i \cap A_j\right| +$$

$$+ \sum_{1\le i<j<k\le n}\left|A_i \cap A_j \cap A_k\right| - \ldots + \left(-1\right)^{n-1} \sum_{1\le i<j<k<\ldots<l\le n}\left|A_i \cap A_j \cap \ldots A_k \cap \ldots A_l \cap A_n\right|$$

Перепишемо дану формулу, враховуючи, що $n = 4$. В результаті одержимо:

$$\begin{aligned}
&\left|A + B + C + D\right| = \left|A\right| + \left|B\right| + \left|C\right| + \left|D\right| - \\
&-\left|A \cap B\right| - \left|A \cap C\right| - \left|A \cap D\right| - \left|B \cap C\right| - \left|B \cap D\right| + \\
&+ \left|A \cap B \cap C\right| + \left|A \cap C \cap D\right| + \left|B \cap C \cap D\right| - \\
&-\left|A \cap B \cap C \cap D\right|
\end{aligned}$$

### 2.1.4. Розміщення з повтореннями (з поверненнями)

Упорядковану $(n,k)$-вибірку, у якій елементи можуть повторюватися, називають $(n,k)$-розміщенням з повтореннями.

Іншими словами, розміщеннями з повтореннями з $n$ елементів по $k$ називають упорядковані вибірки довжини $k$, складені з $n$ елементів множини $X$.

Кількість розміщень з повтореннями з $n$ елементів по $k$ визначають оцінкою відповідного декартового добутку $X^k$ $n$-елементної множини, позначають $\widehat{A}_n^k$ і обчислюють у такий спосіб:

$$\widehat{A}_n^k = n^k$$

**Приклад 2.9.** Нехай дано алфавіт із трьох букв $X = \{a,b,c\}$.

Побудувати всі розміщення $\widehat{A}_3^2$.

*Розв'язок.* Всі розміщення з повтореннями з цих трьох букв по дві $\widehat{A}_3^2 = 3^2$ утворюють множину пар, яка визначається декартовим добутком $X \times X$:

$X \times X = \{(a,a),(b,b),(c,c),(a,b),(b,a),(a,c),(c,a),(b,c),(c,b)\}$.

**Приклад 2.10.** У нашому розпорядженні є склад з комп'ютерами чотирьох типів: 1, 2, 3, 4. Поставлена задача комп'ютеризувати 3 лабораторії, шляхом передачі їм по одному комп'ютеру довільного типу. Скількома способами можна виконати цю задачу?

*Розв'язок.* Кожний спосіб оснащення – це вибірка з повтореннями $(4,3)$, тобто $\widehat{A}_4^3 = 4^3 = 64$.

Приведемо всі можливі способи оснащення кожної із трьох лабораторій одним із чотирьох типів комп'ютерів:

$\{1,1,1\},\{1,1,2\},\{1,1,3\},\{1,1,4\},\{1,2,1\},\{1,2,2\},\{1,2,3\},\{1,2,4\}$
$\{1,3,1\},\{1,3,2\},\{1,3,3\},\{1,3,4\},\{1,4,1\},\{1,4,2\},\{1,4,3\},\{1,4,4\},$
$\{2,1,1\},\{2,1,2\},\{2,1,3\},\{2,1,4\},\{2,2,1\},\{2,2,2\},\{2,2,3\},\{2,2,4\},$
$\{2,3,1\},\{2,3,2\},\{2,3,4\},\{2,3,4\},\{2,4,1\},\{2,4,2\},\{2,4,3\},\{2,4,4\},$
$\{3,1,1\},\{3,1,2\},\{3,1,3\},\{3,1,4\},\{3,2,1\},\{3,2,2\},\{3,2,3\},\{3,2,4\},$
$\{3,3,1\},\{3,3,2\},\{3,3,3\},\{3,3,4\},\{3,4,1\},\{3,4,2\},\{3,4,3\},\{3,4,4\},$
$\{4,1,1\},\{4,1,2\},\{4,1,3\},\{4,1,4\},\{4,2,1\},\{4,2,2\},\{4,2,3\},\{4,2,4\},$
$\{4,3,1\},\{4,3,2\},\{4,3,3\},\{4,3,4\},\{4,4,1\},\{4,4,2\},\{4,4,3\},\{4,4,4\}.$

Алгоритм формування розміщення з повтореннями:

Розміщення формуються послідовним рахунком в системі числення з основою 4.

### 2.1.5. Розміщення без повторень (без повернень)

У ряді задач необхідно визначити число вибірок довжини $k$ з $n$ елементів даної множини без повторення елементів.

*Якщо елементи впорядкованої $(n,k)$-вибірки попарно різні, то їх називають* $(n,k)$-розміщенням без повторень *або просто* $(n,k)$-розміщенням.

Кількість таких розміщень без повторень позначають $A_n^k$ .

*Вивід формули визначення кількості розміщень без повторень ( без повернень)*

Кожне $(n,k)$-розміщення без повторень є впорядкованою послідовністю довжини $k$ , елементи якої попарно різні і вибираються із множини з $n$ елементами.

Тоді перший елемент цієї послідовності може бути обраний $n$ способами, після кожного вибору першого елемента послідовності другий елемент може бути обраний $(n-1)$ способами і т. д., $k$ -й елемент вибирають $n-(k-1)$ способом:

$$A_n^k = n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot(n-(k-1)).$$

Перетворимо цю формулу, помноживши та поділивши її на добуток чисел $1\cdot 2\cdot\ldots\cdot(n-k)$:

$$A_n^k=\frac{n\cdot(n-1)\cdot\ldots\cdot\big(n-(n-k)\big)\cdot 1\cdot 2\cdot(n-k)}{1\cdot 2\cdot\ldots\cdot(n-k)}=$$

$$=\frac{1\cdot 2\cdot\ldots\cdot(n-k)\cdot\big(n-(n-k)\big)\cdot\ldots\cdot(n-1)\cdot n}{1\cdot 2\cdot\ldots\cdot(n-k)}=$$

$$=\frac{n!}{(n-k)!}$$

*Окремі випадки виразів для розміщень без повторень (без повернень):*

1. При $k=0$ одержуємо $A_n^0=\frac{n!}{(n-0)!}=1$.

2. При $k=n$ одержуємо $A_n^n=\frac{n!}{(n-n)!}=\frac{n!}{0!}=n!$

3. При $k>n$ $A_n^k=0$.

4. При $k<n$ $A_n^k=\frac{n!}{(n-k)!}$

**Приклад 2.11.** Скількома способами можна розподілити 20 студентів на 5 комп'ютерів, за умови, що один студент може бути розподілений тільки на один комп'ютер?

*Розв'язок*. Обчислимо розміщення без повторень, оскільки студент може бути розподілений тільки на один комп'ютер:

$$A_n^k=\frac{n!}{(n-k)!}$$

Для нашого випадку $n=20, k=5$: $A_{20}^5=\frac{20!}{15!}=1\,860\,480$.

### 2.1.6. Перестановки без повторень

Розглянемо задачу упорядкування $n$-елементної множини $A$ (формування впорядкованої вибірки довжини $n$, складеної з $n$-елементної множини). Отримані при цьому вибірки будуть відрізнятися лише порядком слідування елементів.

Такі вибірки називають *перестановками без повторень із $n$ елементів.*

Число перестановок без повторень із $n$ елементів позначається $P_n$. До перестановок без повторень можна прийти, вважаючи, що здійснюється розміщення без повторень із $n$ елементів по $n$.

$$P_n = A_n^n = \frac{n!}{(n-n)!} = n!$$

Нагадаємо, що $0! = 1$. Таким чином, перестановки без повторень – це окремий випадок розміщень без повторень (див. вище).

**Приклад 2.12.** Скільки існує послідовностей перевірки контрольних робіт 130 студентів?

*Розв'язок.* Обчислимо кількість перестановок без повторень:

$$P_3 = 130! = 6.46685548922047367257304395536 5e + 219$$

**Приклад 2.13.** Скільки існує послідовностей перевірки контрольних робіт трьох студентів?

*Розв'язок.* $P_3 = 3! = 6$

Ці послідовності мають вигляд:

$$(1,2,3),(2,3,1),(3,1,2),(2,1,3),(1,3,2),(3,2,1).$$

**Приклад 2.14.** Скільки існує можливих послідовностей відвідування туристом п'яти різних країн?

*Розв'язок.* $P_n = n!$ При $n = 5$ одержуємо $P_5 = 5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$

### 2.1.7. Перестановки з повтореннями

При визначенні **перестановок без повторень** ми розглядали ситуацію, коли у початковій $n$-множині $A$ всі елементи унікальні.

Однак існують ситуації, коли *множина може містити деяку кількість однотипних елементів*.

**Визначення.** Число різних перестановок, які можна побудувати з $n$ елементів, серед яких $k_1$ елементів першого типу, $k_2$ елементів другого типу,…, $k_m$ елементів $m$-го типу, дорівнює:

$$P(k_1, k_2, \ldots, k_m) = \frac{n!}{k_1! \cdot k_2! \cdot \ldots \cdot k_m!}.$$

*Приклад для перестановок з повтореннями*

**Приклад 2.15.** Скільки різних слів можна побудувати шляхом перестановки букв у слові «лаваш»?

*Розв'язок.* Слово «лаваш» включає по одному екземпляру букв «л», «в» і «ш», а також два екземпляри букви «а». Загальна кількість букв у слові дорівнює 5.

Використовуючи формулу $P\left(k_1,k_2,\ldots,k_m\right)=\dfrac{n!}{k_1!\cdot k_2!\cdot\ldots\cdot k_m!}$

одержимо $P\left(1,1,1,2\right)=\dfrac{5!}{1!1!1!2!}=5\cdot4\cdot3=60$

**Приклад 2.16.** Скільки слів з 8 букв можна побудувати з букв «а» і «б» за умови, що кількість букв «а» у цих словах не повинна перевищувати 3?

*Розв'язок.* Зазначеним умовам будуть задовольняти слова, які

- не мають жодної букви «а», $P\left(0,8\right)$ ,
- мають одну букву «а», $P\left(1,7\right)$,
- мають дві букви «а», $P\left(2,6\right)$,
- мають три букви «а», $P\left(3,5\right)$.

Тоді загальна кількість слів дорівнює:

$$P\left(0,8\right)+P\left(1,7\right)+P\left(2,6\right)+P\left(3,5\right)=$$

$$=\frac{8!}{0!8!}+\frac{8!}{1!7!}+\frac{8!}{2!6!}+\frac{8!}{3!5!}=1+8+28+56=93$$

### 2.1.8. Сполуки (комбінації) без повторень

**Сполуками без повторень з $n$ елементів по $k$** називають одмінні одна від одної хоча б одним елементом вибірки довжини $k$, складені з елементів $n$-множини.

Кількість сполук без повторень з $n$ елементів по $k$, позначуване як $C_n^k$, визначають виходячи з числа розміщень без повторень $A_n^k$ з урахуванням того, що різних неупорядкованих вибірок (підмножин вихідної множини) буде менше в число раз, яке дорівнює перестановці без повторень з $k$ елементів $P_k$:

$$C_n^k=\frac{A_n^k}{P_k}=\frac{n!}{k!\left(n-k\right)!}.$$

**Приклад 2.17.** Визначити кількість трьохелементних підмножин множини, яка складається з чотирьох елементів.

*Розв'язок.* Перераховуємо всі трьохелементні підмножини множини $A=\left\{a_1,a_2,a_3,a_4\right\}$:

$$\{a_1, a_2, a_3\}, \{a_2, a_3, a_4\}, \{a_1, a_3, a_4\}, \{a_1, a_2, a_4\}$$

Їх кількість можна одержати, обчисливши кількість сполук з 4-х по 3.

$$C_4^3 = \frac{4!}{3!(4-3)!} = \frac{24}{6} = 4$$

**Приклад 2.18.** Збірна команда університету по волейболу нараховує 15 осіб. Скільки варіантів повинен розглянути тренер перед грою, щоб заявити список гравців на гру, за умови, що в грі мають брати участь тільки 6 гравців?
*Розв'язок.* Число гравців волейбольної команди дорівнює 6. Виходить, число всіх можливих варіантів – це число різних підмножин, що складаються з шести елементів у множині з 15 елементів. Отже, використовуючи формулу

$$C_n^k = \frac{n!}{k!(n-k)!}, \text{ отримаємо}$$

$$C_{15}^6 = \frac{15!}{6!(15-6)!} = \frac{15!}{6!9!} = \frac{15 \cdot 14 \cdot 13 \cdot 12 \cdot 11 \cdot 10}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6} = 5 \cdot 7 \cdot 13 \cdot 11 = 5005$$

### 2.1.9. Сполуки (комбінації) з повтореннями

**Суму $k$ кратностей** усіх елементів називають порядком сполуки.

Сполуку з повтореннями $k$-го порядку, що складена з множини, яка містить $n$ елементів, називають також комбінацією з повторенням з $n$ елементів по $k$.

Якщо $k_1, k_2, \ldots, k_n$ – кратності елементів $a_1, a_2, \ldots, a_n$, то за визначенням $k_1 + k_2 + \ldots + k_n$ є порядок комбінації

$$\underbrace{a_1 a_1 a_1 \ldots a_1}_{k_1} \underbrace{a_2 a_2 a_2 \ldots a_2}_{k_2} \ldots \underbrace{a_n a_n a_n \ldots a_n}_{k_n}$$

Кількість вибірок з повтореннями з $n$ елементів по $k$ виражають формулою:

$$\widehat{C}_n^k = C_{n+k-1}^k = C_{n+k-1}^{n-1} = \frac{(n+k-1)!}{k!(n-1)!}$$

Позначаємо сполуки з повтореннями з $n$ елементів по $k$ як $\widehat{C}_n^k$.

**Приклад 2.19.** У кондитерському магазині продавалися 4 сорти тістечок: наполеони, еклери, пісковi і картопля. Скількома способами можна купити 7 тістечок?

*Розв'язок.* Покладемо тістечка в коробку, а щоб вони не переплуталися, розділимо їх картонними роздільниками. Потрібно 3 роздільника. Позначення: 0 (картонки-роздільники) і 1 – тістечка.

Приклад покупки: 1110101101 – три наполеона, 1 еклер, 2 пісковых і 1 картопля (рис. 2.9).

Отже, наявні два класи об’єктів: клас 1 – (7 штук) і клас 0 – (3 штуки) – покупка – 10 об’єктів.


Рис. 2.9. Приклад розташування тістечок в коробці

Два способи міркування:

**(1)** задача зводиться до вибору місць для 7 тістечок (або для 3 роздільників) серед 10 об’єктів. Результат такого вибору показаний у таблиці 2.1.

**Таблиця 2.1**.

Варіанти вибору з повтореннями

| № | Наполеони | | Еклери | | Бісквітні | | Картоплі | Усього |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1111 | 7 |
| 2 | 1 | 0 | 1 | 0 | 11 | 0 | 111 | 7 |
| 3 | 1 | 0 | 1 | 0 | 111 | 0 | 11 | 7 |
| 4 | 1 | 0 | 1 | 0 | 1111 | 0 | 1 | 7 |
| 5 | 1 | 0 | 11 | 0 | 111 | 0 | 1 | 7 |
| 6 | 1 | 0 | 111 | 0 | 11 | 0 | 1 | 7 |
| 7 | 1 | 0 | 1111 | 0 | 1 | 0 | 1 | 7 |
| 7 | 11 | 0 | 111 | 0 | 1 | 0 | 1 | 7 |
| 9 | 111 | 0 | 11 | 0 | 1 | 0 | 1 | 7 |
| 10 | 1111 | 0 | 1 | 0 | 1 | 0 | 1 | 7 |
| 11 | 111 | 0 | 1 | 0 | 1 | 0 | 11 | 7 |
| 12 | 11 | 0 | 1 | 0 | 1 | 0 | 111 | 7 |
| … | … | | … | | … | | … | |
| 210 | 11 | 0 | 1 | 0 | 111 | | 1 | 7 |

$$\widehat{C}_n^k = \widehat{C}_7^4 = \frac{(n+k-1)!}{k!(n-1)!} = \frac{(7+4-1)!}{4!(7-1)!} = \frac{10!}{4!6!} = \frac{10\cdot 9\cdot 8\cdot 7}{24} = 210$$

**(2)** інший спосіб міркування (еквівалентний). Треба розбити 10 місць на дві групи: для 7 тістечок і 3 роздільників.

*У чому особливість:* об’єкти повторюються, причому один еклер на смак не відрізнимо від іншого. Звідси назва: комбінації з повтореннями. Можна уявляти собі, що тістечка безупинно печуть, так що вони не переводяться, скільки не їж. Це зовсім інша ситуація, ніж у звичайних комбінаціях!!!

**Приклад 2.20.** Скількома способами 12 кульок можна розподілити по 3 урнах?

*Розв'язок.* Представимо, що три урни – це одна коробка з двома простінками. Тоді одержуємо два типи об'єктів:

Тип перший: кульки (12 штук) – кульку позначимо 1.

Тип другої: простінки (2 штуки) – простінок позначимо 0.

Приклади розміщення:

$$11111011101111\text{-}\{5,3,4\}$$

$$10110111111111\text{-}\{1,2,9\}$$

$$11101110111111\text{-}\{3,3,6\}$$

$$\widehat{C}_n^k = \widehat{C}_{12}^3 = \frac{(n+k-1)!}{k!(n-1)!} = \frac{(12+3-1)!}{3!(12-1)!} = \frac{14!}{3!11!} = \frac{12\cdot 13\cdot 14}{6} = 364$$

**Приклад 2.21.** Нехай дана множина $A = \{a,b,c,d\}$. Тоді підмножини комбінацій з повтореннями $\widehat{C}_4^2$ включають:

$\{a,a\},\{a,b\},\{a,c\},\{a,d\},\{b,b\},\{b,c\},\{b,d\},\{c,c\},\{c,d\},\{d,d\}$

Їхня кількість

$$\widehat{C}_n^k = \widehat{C}_4^2 = \frac{(n+k-1)!}{k!(n-1)!} = \frac{(4+2-1)!}{2!(4-1)!} = \frac{5!}{2!3!} = 10.$$

### 2.1.10. Розбиття множини на підмножини

Нехай дана $n$-множина $A$. Говорять, що множина $A$ розбита на $k$ підмножин $A_i$, де $(1,2,...,k)$, якщо:

1. $A_i \neq \varnothing, \quad i \in \{1,2,...,k\}$;
2. $A_i \cap A_j = \varnothing,\ i,j \in \{1,2,...,k\}$;
3. $\bigcup_{i=1}^{k} A_i = A$.

Позначимо число елементів у підмножині $A_i$ через $n(A_i) = n_i$ Очевидно, що $n_1 + n_2 + ... + n_k = n$

Кількість розбиттів множини на підмножини позначимо:

$$C(n; n_1, n_2, ..., n_k)$$

**Визначимо кількість розбиттів**

Кількість способів вибору елементів підмножини $A_1$ дорівнює кількості сполук $C_n^{n_1}$ .

Кількість способів вибору елементів підмножини $A_2$ дорівнює кількості сполук $C_{n-n_1}^{n_2}$ .

Кількість способів вибору цих двох підмножин дорівнює $C_n^{n_1} \cdot C_{n-n_1}^{n_2}$ і так далі. Таким чином, кількість вибору всіх розбиттів дорівнює

$$C_n^{n_1} \cdot C_{n-n_1}^{n_2} \cdot C_{n-n_1-n_2}^{n_3} \cdot \ldots \cdot C_{n-n_1-\ldots-n_{k-1}}^{n_k} =$$

$$= \frac{n!}{n_1!\left(n-n_1\right)!} \cdot \frac{\left(n-1\right)!}{n_2!\left(n-n_1-n_2\right)!} \cdot \ldots \cdot \frac{\left(n-n_1-\ldots-n_{k-1}\right)!}{n_k!\left(n-n_1-\ldots-n_{k-1}\right)!} =$$

$$= \frac{n!}{n_1!\cdot n_2!\cdot \ldots \cdot n_{k!}}.$$

$$C_n^{n_1} \cdot C_{n-n_1}^{n_2} \cdot C_{n-n_1-n_2}^{n_3} \cdot \ldots \cdot C_{n-n_1-\ldots-n_{k-1}}^{n_k} = \frac{n!}{n_1!\cdot n_2!\cdot \ldots \cdot n_{k!}}$$

Порівняємо даний вираз з формулою для перестановок з повтореннями.

$$P\left(n_1, n_2, \ldots, n_k\right) = \frac{n!}{n_1!\cdot n_2!\cdot \ldots \cdot n_{\ k}!}$$

Порівняємо даний вираз з формулою для перестановок з повтореннями. Можна зробити висновок, що **явище розбиття множини на підмножини** й **перестановки з повтореннями** – це та сама комбінаторна дія з різною інтерпретацією.

**Приклад 2.22.** Із пропорції $C_x^y : C_x^{y-1} : C_x^{y-2} = 3:3:2$ знайти $x$ й $y$.

*Розв'язок.* Записавши окремо відношення першого члена пропорції до другого й другого до третього, після скорочення одержимо:

$$\frac{x!}{y!(x-y)!} : \frac{x!}{\left(y-1\right)!(x-y+1)!} = \frac{\left(y-1\right)!\left(x-y+1\right)!}{y!\left(x-y\right)!} =$$

$$\frac{\left(y-1\right)!\left(x-y\right)!\left(x-y+1\right)}{y\left(y-1\right)!\left(x-y\right)!} = \frac{\left(x-y+1\right)}{y}$$

$$\frac{x!}{(y-1)!(x-y+1)!}:\frac{x!}{(y-2)!(x-y+2)!}=\frac{(y-2)!(x-y+2)!}{(y-1)!(x-y+1)!}=$$
$$=\frac{(y-2)!(x-y+1)!(x-y+2)}{(y-2)!(y-1)(x-y+1)!}=\frac{(x-y+2)}{(y-1)}$$

З умови задачі одержуємо систему рівнянь $\begin{cases}\dfrac{(x-y+1)}{y}=1\\ \dfrac{(x-y+2)}{(y-1)}=\dfrac{3}{2}\end{cases}$.

Розв'язуємо систему алгебраїчних рівнянь:

$x-y+1=y;\ \ x=2y-1,$

$2x-2y+4=3y-3;\ \ 2x=5y-7.$

$4y-2=5y-7;\ \ y=5;\ \ x=9$

### 2.1.11. Тотожності для сполук

Основна формула для кількості сполук

$$C_n^r=\frac{n!}{r!(n-r)!}$$

дозволяє одержати ряд простих тотожностей.
Розглянемо деякі з них.

**Теорема 2.1**. $C_n^r=C_n^{n-r}$

**Доведення.**

$$C_n^{n-r}=\frac{n!}{(n-r)!(n-(n-r))!}=\frac{n!}{(n-r)!r!}=C_n^r$$

**Теорема 2.2.** $C_n^r=C_{n-1}^r+C_{n-1}^{r-1}.$

**Доведення.**

$$C_{n-1}^r+C_{n-1}^{r-1}=\frac{(n-1)!}{r!(n-r-1)!}+\frac{(n-1)!}{(r-1)!(n-1-(r-1))!}=$$
$$\frac{(n-1)!}{r!(n-r-1)!}+\frac{(n-1)!}{(r-1)!(n-r)(n-r-1)!}=$$
$$\frac{(n-r)(n-1)!+r(n-1)!}{r(r-1)!(n-r)(n-r-1)!}=\frac{(n-r+r)(n-1)!}{r!(n-r)!}=\frac{n!}{r!(n-r)!}=C_n^r$$

**Теорема 2.3.** $C_n^iC_i^r=C_n^rC_{n-r}^{i-r}$

**Доведення.**

$$C_n^i C_i^r = \frac{n!}{i!(n-i)!} \cdot \frac{i!}{r!(i-r)!} = \frac{n!}{r!(i-r)!(n-i)!} =$$

$$\frac{n!(n-r)!}{r!(i-r)!(n-i)!(n-r)!} = \frac{n!}{r!(n-r)!} \cdot \frac{(n-r)!}{(i-r)!(n-i)!} = C_n^r \cdot C_{n-r}^{i-r}$$

**Теорема 2.4. Біном Ньютона:** $(x+y)^n = \sum_{r=0}^{n} C_n^r x^r y^{n-r}$.

**Наслідок 1:** $\sum_{r=0}^{n} C_n^r = 2^n$.

**Наслідок 2:** $\sum_{r=0}^{n} (-1)^r C_n^r = 0$.

**Теорема 2.5.** $\sum_{r=0}^{n} r C_n^r = n 2^{n-1}$

**Теорема 2.6.** $C_{n+r}^k = \sum_{i=0}^{k} C_n^i C_r^{k-i}$

**Контрольні запитання**

1. Скільки існує трицифрових непарних чисел?

2. В групі 25 студентів. Скількома способами можна вибрати керівництво групи у складі старости, заступника старости та профорга?

3. Скільки слів можна одержати шляхом перестановки букв у слові «математика»?

4. На іспиті з математики були запропоновані три задачі: одна з алгебри, одна з геометрії, одна з математичного аналізу. З 1000 студентів завдання з алгебри вирішили 800, з геометрії – 700, з математичного аналізу – 600. При цьому завдання з алгебри і геометрії вирішили 600 студентів, з алгебри та математичного аналізу – 500, з геометрії та математичного аналізу – 400, а 300 студентів вирішили всі завдання. Скільки студентів не вирішили жодної задачі?

5. Скількома способами можна розподілити 10 студентів на 5 груп, за умови, що один студент може бути розподілений тільки в одну групу?

6. Довести, що завжди справедлива така комбінаторна тотожність:

$P(i-1,j,k) + P(i,j-1,k) + P(i,j,k-1) = P(i,j,k)$. Доведення виконати шляхом безпосередньої перевірки та шляхом застосування комбінаторних перетворень.

## 2.2 Базові комбінаторні алгоритми

### 2.2.1. Алгоритми породження підмножин

Розглянемо задачу генерування підмножин довільної $n$-множини $A = \{a_1, a_1, ..., a_n\}$ .

*Існує кілька варіантів цієї задачі, які включають:*

- генерування всіх можливих підмножин даної множини,
- генерування підмножин з деякими умовами, які накладають на підмножини.

Відомо, що кожна $n$-множина $A$ має точно $2^n$ підмножин (згадайте булеан). Тому при створенні комп'ютерних алгоритмів породження підмножин доцільно кожну з отриманих підмножин представити у вигляді двійкової послідовності.

Деяка підмножина $B \subset A$ може бути зіставлена з двійковою послідовністю $b_1 b_2 ... b_j ... b_n$, яку задають у такий спосіб:

$$b_j = \begin{cases} 0, & a_j \notin B, \\ 1, & a_j \in B. \end{cases}$$

Такий підхід дозволяє встановити взаємно однозначну відповідність між усіма булевими векторами довжини $n$ і елементами множини $B$.

### 2.2.2. Генерування всіх підмножин

Для генерації всіх підмножин $n$-множини $A$, достатньо породити всі двійкові набори довжини $n$.

Легко побачити, що найбільш прямим способом їх породження є запис у системі числення з основою 2.

**Приклад 2.23.** Згенерувати всі підмножини множини $M = \{a_0, a_1, a_2\}$

*Розв'язок*. Позначимо підмножину множини $M$ через $B_i$, де $i = 1,2,...,2^{|M|}$

Кожній підмножині $B_i$ поставимо у відповідність двійкову послідовність $b_0 b_1 b_2 ... b_{|M|-1}$ з умовою, що

$$b_j = \begin{cases} 0, & a_j \notin B, \\ 1, & a_j \in B. \end{cases}$$

Результати відповідності представимо в таблиці 2.2.

**Таблиця 2.2.**

Генерація підмножин за двійковою послідовністю

| $i$ | $b_0b_1b_2$ | $B_i$ |
|---|---|---|
| 0 | 000 | $\varnothing$ |
| 1 | 001 | $a_2$ |
| 2 | 010 | $a_1$ |
| 3 | 011 | $a_1a_2$ |
| 4 | 100 | $a_0$ |
| 5 | 101 | $a_0a_2$ |
| 6 | 110 | $a_0a_1$ |
| 7 | 111 | $a_0a_1a_2$ |

$$2^M = \left\{\varnothing, a_0, a_1, a_2, \left\{a_0, a_1\right\}, \left\{a_1, a_2\right\}, \left\{a_0, a_1, a_2\right\}\right\}$$

$$2^{|M|=3},\ b_0 \ldots b_{|M|-1} = b_0b_1b_2$$

### 2.2.3. Алгоритм генерації всіх двійкових векторів довжини $n$ в лексикографічному порядку

Алгоритм породжує всі двійкові вектори $b = \left(b_n, b_{n-1}, \ldots, b_1, b_0\right)$ довжини $n$ в лексикографічному порядку, починаючи з найменшого елемента.

1. Будемо використовувати масив
$b\left[n\right], b\left[n-1\right], \ldots, b\left[1\right], b\left[0\right]$, установивши $b\left[n\right] = 0$.

2. Переглядаючи справа наліво, знаходимо першу позицію $b\left[i\right]$ таку, що $b\left[i\right] = 0$

3. Записуємо $b\left[i\right] = 1$, а всі елементи $b\left[j\right]$, $j < i$, що розміщені праворуч від $b\left[i\right]$, встановлюємо в 0.

4. Для всіх породжуваних послідовностей елемент $b\left[n\right]$ не змінюється, за винятком генерації останнього вектора $\left(1,1,\ldots,1\right)$, $i = n$. Рівність $b\left[n\right] = 1$ є умовою зупинки алгоритму.

*Python-програма генерації двійкових векторів довжини $n$*

```
b=[]
n=4
b = [0 for i in range(0,n+1)]
while not b[n]==1:
 for z in range(0,n): print(b[z], end=' ')
```

```
print()
i=0
while b[i]==1:
    b[i]=0
    i+=1
b[i]=1
```

*Псевдокод програми генерації підмножин*

Розглянемо генерацію підмножин множини $A = \{a_0, a_1, ..., a_{n-1}\}$.

$\mathrm{B} = \varnothing$;
**while** $a_n \notin \mathrm{B}$:
  **while**(B)
    $i = 0$
    **while** $a_i \in \mathrm{B}$:
      $\mathrm{B} = \mathrm{B} \setminus \{a_i\}$
      $i = i + 1$;
    $\mathrm{B} = \mathrm{B} \cup \{a_i\}$;

Введемо фіктивний елемент $a \notin A$. Генерація всіх двійкових векторів b довжини $n = 3$ й підмножин B множини $A = \{a_0, a_1, a_2\}$.

$b^1 = (0,0,0), \quad B^1 = \varnothing, \quad i = 1$

$b^2 = (0,0,1), \quad B^2 = \{a_2\}, \quad i = 2$

$b^3 = (0,1,0), \quad B^2 = \{a_1\}, \quad i = 0$

$b^4 = (0,1,1), \quad B^4 = \{a_1, a_2\}, \quad i = 2$

$b^5 = (1,0,0), \quad B^5 = \{a_0\}, \quad i = 0$

$b^6 = (1,0,1), \quad B^6 = \{a_0, a_2\}, \quad i = 1$

$b^7 = (1,1,0), \quad B^7 = \{a_0, a_1\}, \quad i = 0$

$b^8 = (1,1,1), \quad B^8 = \{a_0, a_1, a_2\}, \quad i = 3$

### 2.2.4. Генерування підмножин з умовою

Розглянемо генерацію підмножин з умовою мінімальної відмінності сусідніх породжуваних елементів. Для такої генерації будемо формувати підмножин, користуючись двійковим кодом Грея. Для цього розглянемо алгоритми формування послідовності за кодом Грея.

*Перший спосіб генерації коду Грея.*

Нехай $b_1 b_2 \ldots b_n$ – деяке двійкове число. Код Грея числа одержують шляхом зсуву його на один розряд вправо, і, відкинувши самий правий ($n$-й розряд), додають порозрядно по модулю два (без переносу у старші розряди) із цим же, але незсунутим числом.

$$\begin{array}{l} b_1 b_2 b_3 \ldots b_{n-1} b_n \\ \oplus\, b_1 b_2 b_3 \ldots\ldots b_{n-1} \not{b_n} \\ \hline c_1 c_2 c_3 \ldots . c_{n-1} c_n \end{array}$$

Правило додавання по модулю 2 двох розрядів проілюстровано в таблиці 2.3.

**Таблиця 2.3.**

Правило додавання

| $b_i \oplus b_{i-1}$ | |
|---|---|
| $0 \oplus 0$ | 0 |
| $0 \oplus 1$ | 1 |
| $1 \oplus 0$ | 1 |
| $1 \oplus 1$ | 0 |

Таким чином, кожний результуючий розряд $c_i$ одержують за формулою $c_i = b_i \oplus b_{i-1}$. Вважають, що $b_0 = 0$.

**Приклад 2.24.** Розглянемо порядок генерації коду Грея для трьохрозрядних двійкових чисел. Результати генерації відобразимо в таблиці 2.4.

**Таблиця 2.4.**

Генерація коду Грея першим способом

| $i$ | Двійкове число | Операція | Код Грея |
|---|---|---|---|
| 0 | 000 | $000 \oplus 00 = 000$ | 000 |
| 1 | 001 | $001 \oplus 00 = 001$ | 001 |
| 2 | 010 | $010 \oplus 01 = 011$ | 011 |
| 3 | 011 | $011 \oplus 01 = 010$ | 010 |
| 4 | 100 | $100 \oplus 10 = 110$ | 110 |
| 5 | 101 | $101 \oplus 10 = 111$ | 111 |
| 6 | 110 | $110 \oplus 11 = 101$ | 101 |
| 7 | 111 | $111 \oplus 11 = 100$ | 100 |

*Другий спосіб генерації коду Грея*

Другий спосіб генерації коду Грея складається спочатку розглянемо як алгоритм, що складається з фіксованої послідовності кроків:

1. Вибираємо дворозрядну послідовність 00,01,11,10 як базову.
2. Будуємо трирозрядну послідовність за наступним правилом:

2а. Приписуємо до 00,01,11,10 праворуч 0 і отримуємо послідовність:

000,010,110,100.

2б. Переставляєм елементи послідовності 00,01,11,10 у зворотному порядку:

10,11,01,00.

2в. До елементів послідовності 10,11,01,00 приписуємо праворуч 1 і отримуємо послідовність :

101,111,011,001.

2г. Об'єднаємо послідовності з п.2а й п.2в і отримуємо повну трирозрядну послідовність:

000, 010,110,100,101,111,011,001.

3. Для одержання чотирирозрядної послідовності перейдемо до п.1 алгоритму, замінивши дворозрядну послідовність послідовністю, отриманою в п. 2г.

4. Повторюючи дії $(n-2)$ раз, одержимо $n-$ розрядний код Грея.

Спираючись на даний алгоритм сформулюємо загальне правило генерації коду Грея другим способом.

Якщо $k-$розрядна послідовність $c_1, c_2, c_3, ..., c_k$ містить усі можливі двійкові числа довжини $k$ і кожне число відрізняється від сусіднього точно одним розрядом, то, приписуючи праворуч нуль до кожного числа цієї послідовності і одиницю також до кожного числа цієї послідовності, записаної у зворотному порядку, одержимо в результаті нову послідовність, яка містить усі числа довжини $k+1$, які відрізняються один від одного точно одним розрядом.

**Приклад 2.25.** Згенерувати всі підмножини множини $A=\{a_1, a_2, a_3\}$ з умовою мінімальної відмінності сусідніх породжуваних елементів.

*Розв'язок.* Результати представимо у вигляді таблиці:

| $i$ | $b_1b_2b_3$ | $B_i$ |
|---|---|---|
| 0 | 000 | $\varnothing$ |
| 1 | 010 | $a_2$ |
| 2 | 110 | $a_1, a_2$ |
| 3 | 100 | $a_1$ |
| 4 | 101 | $a_1, a_3$ |
| 5 | 111 | $a_1, a_2, a_3$ |
| 6 | 011 | $a_2, a_3$ |
| 7 | 001 | $a_3$ |

1. Початкова послідовність
00,01,11,10
2а. Дописали 0 справа
000,010,110,100
2б. Послідовність (1) перевернули
10,11,01,00
2в. До послідовності (2б) дописали 1
101,111,011,001
2г. Поєднали послідовності (2а) та (2в)
000, 010,110,100,101,111,011,001

*Python-програма, яка генерує код Грея першим способом*

```
m=8
Gr=[]
def bintogray(i):
    return i ^ (i >> 1)
for i in range(1,m+1):
```

```
    Gr.append(bintogray(i))
print([bin(k) for k in Gr])
```

*Псевдокод генерації коду Грея другим способом*

Розглянемо двовимірний масив `GCode[i, j]`, де `j` – це `j`–й розряд в `i`–му коді Грея. Змінна `p` містить поточну кількість вже створених кодів Грея.

```
buildCode(n):
    GrayCode[1, n] = false
    GrayCode[2, n] = true   // Побудова кода довжини 1
    p = 2
    for i = 2 to n
        t = p
        p = p * 2
        for k = (p / 2 + 1) to p
            GrayCode[k] = GrayCode[t]// Перевертання
            GrayCode[t, n + 1 – i] = false
            GrayCode[k, n + 1 – i] = true //Додавання 0 і1
            t--
    return GrayCode
```

### 2.2.5. Генерування $k$ -елементних підмножин

Розглянемо множину $A=\{1,2,...,n\}$. Довільне сполучення з $n$ по $k$ зручно представити у вигляді послідовності довжини $k$ з чисел, упорядкованих за зростанням зліва направо. Всі такі послідовності природно породжувати в лексикографічному порядку. Наприклад, при $n$ = 5 і $k$ = 3 послідовність всіх сполучень в лексикографічному порядку наступна:

123, 124, 125, 134, 135, 145, 234, 235, 245, 345.

Очевидно, що при генерації всіх можливих сполучень перший елемент у лексикографічному порядку є сполучення $(1,2,...,k)$, а останній- $(n-k+1,n-k....,n-1,n)$.

1. Розглянемо сполучення $(a_1,a_2,...,a_k)$.

2. Тоді наступне сполучення визначають з виразу:

$(b_1,b_2,...,b_k)=(a_1,...,a_{p-1},a_p+1,a_p+2,...,a_p+k-p+1)$, де

$p=\max\{i|a_i<n-k+1\}$

3. Наступне сполучення, яке генерується на основі сполучення $(b_1,b_2,...,b_k)$, має вигляд:

$(c_1,...,c_k)=(b_1,...,b_{p'-1},b_{p'}+1,b_{p'}+2,...,b_{p'}+k-p'+1)$, де

$$p'=\begin{cases} p-1, & при\, b_k=n, \\ k, & при\, b_k<n \end{cases}$$

Розглянемо застосування цих правил на прикладі множини $A=\{1,2,3,4,5,6\}$ з потужністю $n=|A|=6$. З даної множини будемо формувати всі можливі підмножини з потужністю $k=4$.

1. Нехай перша підмножина з елементами $\{a_1,a_1,\ldots,a_{k-1},a_k\}$ дорівнює $\{1,2,3,4\}$.

2. Якщо $(a_1<a_n-k+1)\,\&\ldots\&\,(a_{k-1}<a_n-1)\,\&\,(a_k<n)$, то збільшуємо на 1 останній елемент поточної послідовності $\{a_1,a_1,\ldots,a_{k-1},a'_k=a_k+1\}$ для формування наступного елемента послідовності goto 2.

*Наприклад*: якщо $\{a_1,a_1,\ldots,a_{k-1},a_k\}=\{12,3,4\}$ то $\{a_1,a_1,\ldots,a_{k-1},a'_k\}=\{12,3,5\}$

……

3. Якщо $(a_1<a_n-k+1)\,\&\ldots\&\,(a_{k-1}<a_n-1)\,\&\,(a_k=n)$ то наступний елемент послідовності формуємо у відповідності з виразом $(a'_1=a_1+1,a_2=a'_1+1,\ldots,a'_{k-1}=a'_1+k-2,a'_k=a'_1+k-1)$ і переходимо для формування наступного елемента послідовності goto 2.

*Наприклад*: якщо $\{1,4,5,6\}$ то $\{2,3,4,5\}$

4. Якщо $(a_1=a_n-k+1)\,\&\ldots\&\,(a_{k-1}=a_{n-1})\,\&\,(a_k=a_n)$ то Stop

*Наприклад*: якщо $\{3,4,5,6\}$ то Stop

*Python- програма, яка генерує всі $k$-елементні підмножини $n$-множини*

```
n=6
k=4
A=[]
for i in range(k+1): A.append(i)
p=k
while p:
 print(A[1:k+1])
 if A[k]==n: p-=1
 else: p=k
 if p:
    for i in range(k,p-1,-1):
       A[i]=A[p]+i-p+1
```

```
1234
1235
1236
1245
1246
1256
1345
1346
1356
1456
2345
2346
2356
2456
3456
```

Праворуч наведений приклад 4-елементних підмножин множини {1,…,6}, побудованих за даним алгоритмом.

Значно простіше можна одержати всі підмножини заданої множини, якщо скористатися ітераторами та функціями з модуля itertools.

*Python-програма, яка генерує всі $k$-елементні підмножини $n$-множини за допомогою модуля itertools*

```
import itertools
s = set((1, 2, 3, 4, 5, 6))
print(list(map(set, itertools.combinations(s, 4))))
```

### 2.2.6. Алгоритми перестановок

Розглянемо методи генерування послідовностей $n!$ перестановок множини, складеної з $n$ елементів. Для цього задану множину представимо у вигляді елементів масиву $P[1], P[2], ..., P[n]$.

Методи, які будуть нами розглядатися, базуються на застосуванні до масиву $P[i]$, $i = 1,2,...,n$ елементарної операції, яку називають **поелементною транспозицією**. Суть операції полягає в обміні даними між елементами масиву $P[i]$ і $P[j]$, $1 \le i, j \le n$ за такою схемою:

$$vrem = P[i],\ P[i] = P[j],\ P[j] = vrem,$$

де $vrem$ – деяка допоміжна змінна, яку використовують для тимчасового зберігання значення елемента масиву $P[i]$.

*Лексикографічний порядок*

**Визначення лексикографічного порядку.** Нехай існують перестановки у вигляді послідовностей $\{x_1, x_2, x_3, ..., x_n\}$, $\{y_1, y_2, y_3, ..., y_n\}$,... тієї самої множини $X$. Перестановки з елементів множини $X$ впорядковані в лексикографічному порядку, якщо $\{x_1, x_2, x_3, ..., x_n\} < \{y_1, y_2, y_3, ..., y_n\}$ тоді й тільки тоді, коли для довільного $k$: $x_k \le y_k$ і $x_i = y_i$ для всіх $i < k$.

**Визначення антилексикографічного порядку.** Нехай існують перестановки у вигляді послідовностей $\{x_1, x_2, x_3, ..., x_n\}$, $\{y_1, y_2, y_3, ..., y_n\}$,... тієї самої множини $X$. Перестановки з елементів множини $X$ впорядковані в антилексикографічному порядку, якщо $\{x_1, x_2, x_3, ..., x_n\} > \{y_1, y_2, y_3, ..., y_n\}$ тоді й тільки тоді, коли для довільного $k$: $x_k > y_k$ і $x_i = y_i$ для всіх $i < k$.

*Алгоритм побудови перестановок у лексикографічному порядку*

Початкова перестановка $(1,2,\ldots,n-1,n)$.

Завершальна перестановка $(n,n-1,\ldots,2,1)$.

Розглянемо перехід від $(x_1,x_2,\ldots,x_n)$ до $(y_1,y_2,\ldots,y_n)$

Як будуємо перестановку $(y_1,y_2,\ldots,y_n)$ ?

1. Розглядаємо перестановку справа наліво

$$x=\left(x_1,x_2,\ldots,x_i,\overset{\leftarrow}{x_{i+1},\ldots,x_n}\right)$$

і знайдемо таку позицію $i$, що $x_i < x_{i+1}$.

2. Якщо такої позиції немає, то тоді $x_1 > x_2 > \ldots > x_n$, тобто $x=(n,n-1,\ldots,2,1)$. Дана перестановка є завершальною перестановкою нашого алгоритму.

3. Якщо позиція $i$ знайдена, то $x_i < x_{i+1} > x_{i+2} > \ldots > x_n$.

4. Шукаємо першу позицію $j$ в діапазоні від позиції $n$ до позиції $i$ таку, що $x_i < x_j$. Тоді $i < j$.

$$x=\left(x_1,x_2,\ldots,x_i,\overset{\leftarrow}{x_{i+1},\ldots,x_j},\ldots,x_n\right)$$

5. Далі виконуємо операцію транспозиції над елементами $x_i$ й $x_j$

$$x=\left(x_1,x_2,\ldots,x_i,\overset{\leftarrow}{x_{i+1},\ldots,x_j},\ldots,x_n\right)$$

6. В отриманій перестановці частину елементів $x_{i+1},\ldots,x_{n-1},x_n$ перевертаємо, тобто міняємо порядок їх слідування на протилежний.

7. Отримана перестановка є перестановкою $y=(y_1,y_2,\ldots,y_n)$.

Саме ця перестановка є наступною в лексикографічному порядку слідування перестановок.

**Приклад 2.26.** Нехай $x=(2,6,5,8,7,4,3,1)$.

1. Відповідно до даного алгоритму $x_i=5$, а $x_j=7$.

2. Виконаємо транспозицію для елементів $i=3$ і $j=5$:

$$\tilde{x}=(2,6,7,8,5,4,3,1)$$

3. Перевернемо частину елементів $x_3,\ldots,x_8 \to x_8,\ldots,x_3$:

$$\left(8,5,4,3,1\right) \to \left(1,3,4,5,8\right)$$

У результаті одержимо наступну в лексикографічному порядку перестановку $y = \left(2,6,7,1,3,4,5,8\right)$.

*Python-програма*
*для генерування перестановок у лексикографічному порядку*

```
p=[]
n=4
for k in range(n+1): p.append(k)
k=1;
while k!=0:
 print(p)
 k=n-1
 while p[k]>p[k+1]: k-=1
 j=n;
 while p[k]>p[j]: j-=1
 p[k],p[j]=p[j],p[k]
 j=n
 m= k+1
 while j>m:
   p[j],p[m]=p[m],p[j]
   j-=1
   m=m+1
```

Елемент масиву a[0]=0 використовується тільки як ознака закінчення алгоритму.

*Python-програма, яка генерує перестановки за допомогою модуля itertools*

```
import itertools
print([x for x in itertools.permutations('1234')])
```

**Приклад 2.27.** Нехай дано множину $X = \left\{1,2,3\right\}$. Записати всі перестановки елементів даної множини у лесикографічному (а) і антилексикографічному (б) порядку.

*Розв'язок.*

| | (а) | (б) |
|---|---|---|
| 1 | 1 2 3 | 1 2 3 |
| 2 | 1 3 2 | 2 1 3 |
| 3 | 2 1 3 | 1 3 2 |
| 4 | 2 3 1 | 3 1 2 |
| 5 | 3 1 2 | 2 3 1 |
| 6 | 3 2 1 | 3 2 1 |

### 2.2.7. «Швидке сортування» (Quicksort)

Quicksort є одним з найбільш ефективних і найбільш часто використовуваних алгоритмів для сортування списку чисел. Quicksort може сортувати список на місці, заощаджуючи необхідність створення копії списку, і, таким чином, зменшуючи вимоги до оперативної пам'яті.

Основна ідея алгоритму полягає у знаходженні головного елемента з наступним поділом масиву на дві частини: одна частина буде містити елементи, менші за $x$, а інша — більші за $x$. Далі для кожної такої частини знову застосовується описана вище процедура.

Створимо функцію `partition` з трьома аргументами. Перший аргумент – це список `xs`, який необхідно відсортувати. Аргумент `start` містить індекс початкового елемента для сортування. Аргумент `end` є індексом останнього елемента для сортування.

```
1.def partition(xs, start, end):
2.    follower = leader = start
3.    while leader < end:
4.        if xs[leader] <= xs[end]:
5.            xs[follower], xs[leader] = xs[leader], xs[follower]
6.            follower += 1
7.        leader += 1
8.    xs[follower], xs[end] = xs[end], xs[follower]
9.    return follower
```

Ми задаємо початкові та кінцеві значення елементів, тому що не завжди необхідно сортувати весь список. Алгоритм в ході сортування буде працювати на підсписках, довжина яких зменшуватиметься. Для того, щоб не створювати нові копії списку, будемо визначати ці підсписки, використовуючи індекси даного початкового списку.

У *рядку* 2 ми починаємо з того, що змінні `follower` та `leader` посилаються на початок списку, який підлягає сортуванню. Змінна `leader`

буде змінюватися швидше, ніж `follower`, тому ми будемо виконувати цикл, поки `leader` не досягне з кінця сегмента списку.
*Рядок* 3:(**`while`** `leader < end`).

Ми можемо вибрати будь-який елемент, як опорний елемент, але для простоти ми просто виберемо **останній елемент**. Потім у *рядку* 4 ми порівнюємо елемент – `leader` з опорним елементом. Змінна `leader` проходить по кожному пункту в списку, тому після досягнення кінцевого елемента ми перевіряємо увесь сегмент списку.

Якщо елемент – `leader` є меншим або дорівнює опорному елементу, ми повинні перенести його далі ліворуч, а більший елемент, на який вказує `follower`, перемістити вправо. Ці дії виконуються в *рядках* 4–5. Якщо знайдемо випадок, коли `leader` менший або дорівнює опорному елементу, ми міняємо його місцями з `follower`. На цьому етапі `follower` вказує на менший елемент (той, на який тільки-но вказував `leader`), тому ми збільшуємо величину змінної `follower` на один для того, щоб почати відстежувати наступний елемент (*рядок* 6).

Незалежно від того, чи зробили ми транспозицію, ми розглядаємо наступний елемент по відношенню до нашого опорного елемента, тому в *рядку* 7 ми збільшуємо величину змінної `leader`.

Як тільки ми випадаємо з циклу (*рядок* 8), нам потрібно виконати транспозицію опорного елемента (все ще в кінці списку) з елементом на який вказує `follower` (який перемістився на один елемент вище для кожного елемента, який був меншим, ніж опорний). Якщо це все ще незрозуміло, перегляньте наш приклад знову, спираючись на простий приклад з початковим списком:

```
xs = [8, 4, 2, 2, 1, 7, 10, 5]
```

У `xs` є 4 елементи, які менші за опорний елемент. Кожен раз, коли ми знаходимо елемент, який є меншим, ніж опорний, ми робимо інкремент `follower` на одиницю. Це означає, що в кінці циклу `follower` збільшиться в 4 рази і вкаже на індекс 4 у початковому списку. Можна побачити, що це правильне місце для нашого опорного елемента (5).

Останнє, що ми робимо, це повернення індекса змінної `follower`, який тепер вказує на наш опорний елемент на своєму коректному місці. Далі ми розглядаємо дві менші підзадачі в нашому розділеному списку – тепер ми сортуємо `xs[0:4]` (перші 4 елементи, які утворюють несортований список) і `xs[5:]` (останні 3 елементи, які утворюють несортований список).

```
xs = [4, 2, 2, 1, 5, 7, 10, 8]
```

Оскільки ми вже створили алгоритм розділення, сортування стає легким. Спочатку ми визначимо допоміжну функцію `_quicksort`, щоб обробити рекурсію, а потім створимо кращу загальнодоступну функцію.

```
1.def _quicksort(xs, start, end):
2.     if start >= end:
3.         return
4.     p = partition(xs, start, end)
5.     _quicksort(xs, start, p-1)
6.     _quicksort(xs, p+1, end)
```

Щоб відсортувати список, розділяємо його (*рядок* 4), сортуємо лівий підсписок (*рядок* 5: від початку початкового списку до опорного елемента), а потім сортуємо правий підсписок (рядок 6: відразу після опорного елемента до на кінця початкового списку). Ми робимо це рекурсивно, починаючи з `end`, рухаючись вліво у напрямку до `start` для лівого підсписку і, починаючи зі `start`, рухаємся вправо до `end` для правих підсписків. Коли межі початку і кінця зустрічаються (*рядок* 2), ми закінчили!

Перший виклик `Quicksort` завжди буде з повним списком, який ми хочемо відсортувати, що означає, що `0` буде початком списку, а `len(xs) - 1` буде кінцем списку. Ми не хочемо пам'ятати про передачу цих додаткових аргументів кожного разу, коли потрібно викликати функцію `Quicksort` з іншої програми (тобто, у будь-якому випадку, коли вона не викликає себе), тому ми зробимо більш гарну функцію обгортки з тими значеннями за замовчуванням, щоб розпочати процес сортування.

Тепер ми, як користувачі функції сортування, можемо викликати функцію таким чином:

```
quicksort ([4,5,6,2,3,9,10,2,1,5,3,100,23,42,1]),
```

передаючи тільки список, який ми хочемо відсортувати. Це, у свою чергу, перейде до виклику функції `_quicksort`, яка буде в подальшому викликати себе в процесі сортування.

Повний код Python-програми, яка реалізує алгоритм `quicksort`:

```
def partition(xs, start, end):
    follower = leader = start
    while leader < end:
        if xs[leader] <= xs[end]:
            xs[follower], xs[leader] = xs[leader], xs[follower]
            follower += 1
        leader += 1
    xs[follower], xs[end] = xs[end], xs[follower]
```

```
    return follower

def _quicksort(xs, start, end):
    if start >= end:
        return
    p = partition(xs, start, end)
    _quicksort(xs, start, p - 1)
    _quicksort(xs, p + 1, end)

def quicksort(xs):
    _quicksort(xs, 0, len(xs) - 1)

import random

# create 100000 random numbers between 1 and 1000
xs = [random.randrange(100) for _ in range(1000)]

# look at the first few and last few
print(xs[:10], xs[-10:])
quicksort(xs)
print(xs[:10], xs[-10:])
```

### 2.2.8. Сортування злиттям

Ідея алгоритму сортування злиттям полягає в тому, щоб розділяти несортований список на менші групи елементів, поки не створимо групи лише один елемент у групі. Потім групують два елементи в сортованому порядку і поступово збільшують розмір групи. Кожного разу, коли відбувається злиття, елементи в групах повинні порівнюватися один за одним і об'єднуватися в єдиний список у відсортованому порядку. Цей процес триває доти, доки всі елементи не будуть об'єднані та відсортовані. Зверніть увагу, що коли відбувається перегрупування, сортований порядок повинен завжди зберігатися.

Розглянемо роботу алгоритму на прикладі. Початковий несортований список має такий вигляд: `alist = [5,9,1,2,7,0]`.

1. Етап розбиття списку на підсписки:

```
Крок 1: [5,9,1] [2,7,0]
Крок 2 : [5] [9,1] [2] [7,0]
Крок 3: [5] [9] [1] [2] [7] [0]
```

Оскільки кількість елементів у підсписках на кроці 2 є непарним, то розбиття цих підсписків може виконуватися довільним способом. Тому розбиття `[5,9] [1] [2,7] [0]` також можливе і не порушує хід алгоритму.

2. Етап злиття підсписків

```
Крок 4: [5,9] [1,2] [0,7]
Крок 5: [1,2,5,9][0,7]
Крок 6: [0,1,2,5,7,9]
```

Алгоритм сортування злиттям включає:

1. Поділ несортованого списку на підсписки рекурсивно, поки не одержимо підсписки з довжиною 1 елемент.
2. Порівняння кожного з елементів, а потім їх групування
3. Повторення кроку 2 поки весь список не буде об'єднано у відсортований список.

Код Python-програми для алгоритму сортування злиттям `mergeSort`:

```
def mergeSort(alist):

    print("Splitting ",alist)

    if len(alist)>1:
        mid = len(alist)//2
        lefthalf = alist[:mid]
        righthalf = alist[mid:]

        #recursion
        mergeSort(lefthalf)
        mergeSort(righthalf)

        i=0
        j=0
        k=0

        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                alist[k]=lefthalf[i]
                i=i+1
            else:
                alist[k]=righthalf[j]
                j=j+1
            k=k+1

        while i < len(lefthalf):
            alist[k]=lefthalf[i]
            i=i+1
            k=k+1
```

```
        while j < len(righthalf):
            alist[k]=righthalf[j]
            j=j+1
            k=k+1

    print("Merging ",alist)

alist = [54,26,93,17,77,31,44,55,20]
mergeSort(alist)
print(alist)
```

### 2.2.9. Двійковий (бінарний) пошук елемента в масиві

Якщо є масив, який містить упорядковану послідовність даних, то ефективним є бінарий пошук.

Ідея пошуку полягає в тому, щоб брати елемент посередині, між границями, і порівнювати його із шуканим. У випадку рівності повертати його, а якщо шукане більше (у випадку правобічного – не менше), ніж елемент порівняння, то звужуємо область пошуку так, щоб нова ліва границя дорівнювала індексу середини попередньої області. А якщо ні, то присвоюємо це значення правій границі. Пророблюємо цю процедуру доти, поки права границя більше лівої більш ніж на 1, або ж поки ми не знайдемо шуканий індекс.

Двійковий пошук – дуже потужний метод. Якщо, наприклад, довжина масиву дорівнює 1023, після першого порівняння область звужується до 511 елементів, а після другого – до 255. Легко порахувати, що для пошуку в масиві з 1023 елементів достатньо 10 порівнянь.

*Опис алгоритму*

1. Знаходимо середній елемент послідовності. Для цього перший і останній індекси зв’язуються з змінними, а індекс середнього елемента обчислюється.
2. Шукане значення порівнюємо зі значенням середнього елемента та визначаємо більше воно чи менше значення середнього елемента.
3. Подальший пошук будемо проводити тільки в лівій або тільки в правій половині списку в залежності від одержаного результату у п. 2. Якщо значення середнього елемента дорівнює шуканому, то пошук завершується.
4. У випадку вибору одного з підсписків одна з меж досліджуваної послідовності зсувається. Якщо шукане значення більше значення середнього елемента, то нижня межа зсувається за середній елемент на один елемент праворуч. Якщо шукане значення менше значення середнього елемента, то верхня межа зсувається на елемент перед середнім.
5. Знову знаходиться середній елемент тепер уже в обраній половині. Описаний вище алгоритм повторюється для даного підсписку.

*Код Python-програми для алгоритму бінарного пошуку.*

```
from random import randint

# Створення списку,
# сортування за зростанням
# та вивід на екран
a = []
for i in range(15):
    a.append(randint(1, 50))
a.sort()
print(a)

# шукане число
value = int(input())

mid = len(a) // 2
low = 0
high = len(a) - 1

while a[mid] != value and low <= high:
    if value > a[mid]:
        low = mid + 1
    else:
        high = mid - 1
    mid = (low + high) // 2

if low > high:
    print("No value")
else:
    print("ID =", mid)
```

**Контрольні запитання**

1. Студент здає $n$ заліків у кожну сесію. Якщо він отримує у поточній сесії $m$ раз оцінку «не зараховано», то підлягає автоматичному виключенню з університету. Знайдіть кількість послідовностей, при яких може бути отримано оцінку «не зараховано» $m$ або більше разів.

2. За довільним набором букв згенерувати всі можливі слова в лексикографічному порядку. Наприклад, зі слова «abc» можна отримати слова abc, acb, bac, bca, cab, cba. Букви в слові можуть повторюватися.
Букви верхнього і нижнього регістра вважати різними.
Необхідний вивід для програми: для кожного вхідного набору вивести всі можливі слова, які можна отримати із заданих літер, у зростаючому лексикографічному порядку. Кожне слово виводити в окремому рядку.

3. Сформувати та вивести на друк усі послідовності довжини $k$ з чисел 1, ..., $n$, $k \geq n$ у яких $i$-ий член послідовності не перевищує $i$.

4. За заданою перестановкою із чисел 1, ..., $n$ знайти її номер при лексикографічному впорядкуванні.

5. Сформувати та вивести на друк усі розбиття натурального числа $n$ на натуральні доданки. Розбиття, що відрізняються лише порядком доданків, вважаються однаковими. Наприклад, при $n = 4$ розбиттями будуть 1+1+1+1, 2+1+1, 2+2, 3+1, 4.

6. Потяг складається із $n$ вагонів, які пронумеровані від 1 до $n$. Вагони потягу стоять у деякому порядку, утворюючи своїми номерами перестановку чисел від 1 до $n$. Є пристрій, який дозволяє поміняти місцями два сусідні вагони. Знайти найменшу кількість застосувань такого пристрою для відсортування вагонів за їх номерами у порядку зростання.

# Розділ 3. Теорія графів

## 3.1. Основні положення теорії графів

### 3.1.1. Історія виникнення теорії графів

Історія теорії графів почалася із задачі про кенігсберзькі мости, придуманої і розв'язаної Ейлером у 1736 році. Ейлер поставив собі задачу, як обійти всі чотири частини суші, пройшовши по кожному з мостів рівно один раз, і повернутися у початкове місце. Мости, по яких ходив Ейлер, розташовувалися, як показано на рис. 3.1.

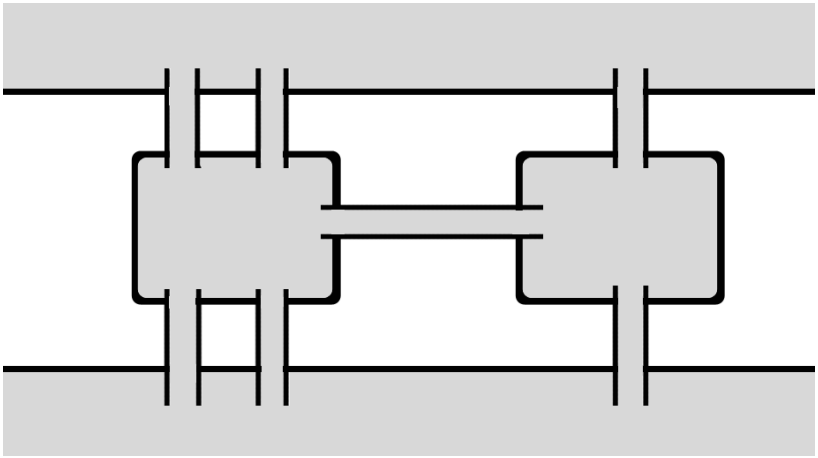


Рис. 3.1. Розташування кенігсберзьких мостів

Для розв'язування цієї задачі Ейлер позначив ділянки суші точками, а мости, що їх з'єднують, лініями, одержавши таким чином перше представлення графа.

Один з варіантів графа задачі про кенігсберзькі мости має такий вигляд (рис. 3.2):

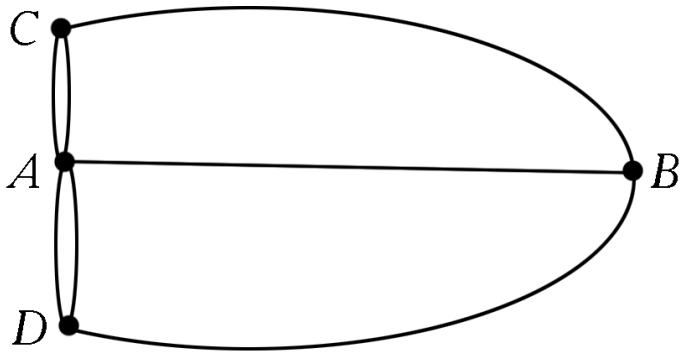



Рис. 3.2. Граф, який зображає розташування кенігсберзьких мостів

Як видно з рис. 3.2, граф складається з точок та ліній. Точки називають вершинами. Лінії, що з'єднують вершини, називають ребрами.

### 3.1.2. Основні визначення графів

*Неорієнтований граф*

**Визначення.** *Неорієнтованим графом* $G(V,E)$ називають сукупність двох множин:
– непустої множини $V$, яку називають множиною *вершин*;
– множини $E$ неупорядкованих пар елементів множини $V$, яку називають множиною *ребер*. Множина ребер у графі може бути пустою.

Умови існування неорієнтованого графа $G(V,E)$ задають наступними правилами:

1. $V \neq \varnothing$. Множина вершин графа непуста.
2. $E \subset V \times V$. Елементами множини ребер є відношення, елементами якого є двійки з вершин графа. Кількість двійок є підмножиною декартового добутку $V \times V$.
3. $E = E^{-1}$. Множина ребер неорієнтованого графа – це симетричне відношення.
   Приклад неорієнтованого графа показаний на рис. 3.3.

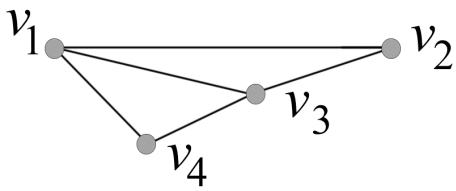



Рис. 3.3. Неорієнтований граф

Кількість вершин графа $G$ позначимо через $p$, а кількість ребер графа $G$ позначимо через $q$.

$$p = p(G) = |V|,\ q = q(G) = |E|,$$

$$V = \{v_1, v_2, v_3, v_4\},$$

$$E = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_3), (v_3, v_4)\} =$$

$$= E^{-1} = \{(v_2, v_1), (v_3, v_1), (v_4, v_1), (v_4, v_3), (v_3, v_2)\}.$$

*Орієнтований граф*

**Визначення.** Орієнтованим графом $G(V,E)$ називають сукупність двох множин:
– непустої множини $V$, яку називають множиною вузлів;
– множини $E$ упорядкованих пар елементів множини $V$, яку називають множиною дуг.

Умови існування орієнтованого графа $G(V,E)$ задають наступними правилами:

4. $V \neq \varnothing$. Множина вузлів графа непуста.
5. $E \subset V \times V$. Елементами множини дуг є відношення, елементи якого – це двійки з вузлів графа. Кількість двійок є підмножиною декартового добутку $V \times V$.
6. $E \neq E^{-1}$. Множина дуг орієнтованого графа – це антисиметричне відношення.

Якщо елементами множини $E$ є впорядковані пари, то граф називають *орієнтованим* (або орграфом).

Дуги зображують лініями зі стрілками, що вказують напрямок. Приклад орієнтованого графа показаний на рис. 3.4.

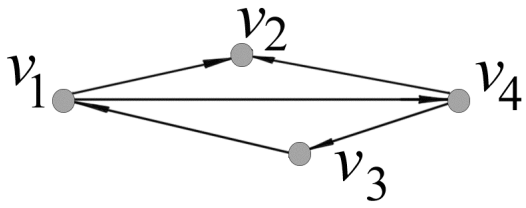



Рис. 3.4. Орієнтований граф

$$V = \{v_1, v_2, v_3, v_4\},$$

$$E = \{(v_1, v_2), (v_4, v_2), (v_1, v_4), (v_3, v_1), (v_4, v_3)\}.$$

**Приклад 3.1.** Дано орграф $G(V, E)$, зображений на рис. 3.5:

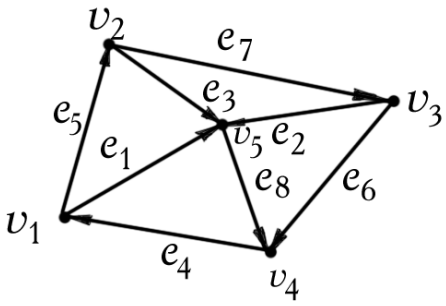



Рис. 3.5. Орграф $G(V, E)$

Визначити впорядковані пари множини

$$E = \{e_i \mid i = 1, 2, 3, \ldots, 7\},$$

що задає дуги даного орграфа.

*Розв'язок.*

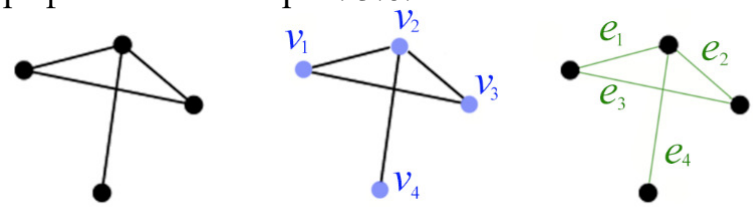
Запишемо відношення, яке відповідає множині впорядкованих пар вузлів $V$.

$$E = \{(v_1, v_5), (v_3, v_5), (v_2, v_5), (v_4, v_1), (v_1, v_2), (v_3, v_4), (v_2, v_3)\},$$

$$e_1 = (v_1, v_5),\ e_2 = (v_3, v_5),\quad e_3 = (v_2, v_5).$$

*Помічені графи*

**Визначення.** Якщо задана функція $F : V \to M$ або $F : E \to M$, то множину $M$ називають множиною *міток*, а граф називають *поміченим графом.* Такі графи показані на рис. 3.6.




А Б В

Рис. 3.6. Помічені (Б, В) і непомічені графи (А), де

А – непомічений граф,

Б – граф з поміченими вершинами $V = \{v_1, v_2, v_3, v_4\}$

(вершинно-помічений граф),

В – граф з поміченими ребрами $E = \{e_1, e_2, e_3, e_4\}$

(реберно-помічений граф).

*Граф з петлями*

Якщо серед елементів множини $E$ зустрічаються пари, які містять однакові вершини, то такий граф називають *графом з петлями*.

**Приклад 3.2.** На рис. 3.7 показаний граф з петлями і орграф з петлями.

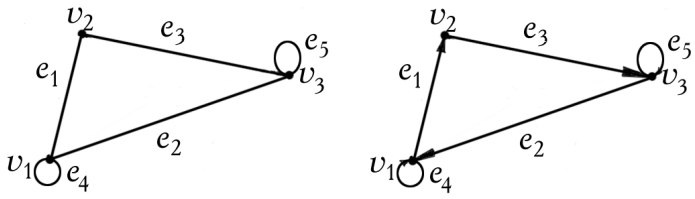



Рис. 3.7. Граф $G_1(V_1, E_1)$ та орграф $G_2(V_2, E_2)$ з петлями

Задати ці графи аналітично.

*Розв'язок.*

$G_1(V_1, E_1)$; $V_1 = \{v_1, v_2, v_3\}$,

$E_1 = \{(v_1, v_2), (v_2, v_1), (v_2, v_3), (v_3, v_2), (v_1, v_3)(v_3, v_1)(v_1, v_1), (v_3, v_3)\}$,

$G_2(V_2, E_2)$, $V_2 = \{v_1, v_2, v_3\}$, $E_2 = \{(v_1, v_2), (v_2, v_3), (v_3, v_1)(v_1, v_1), (v_3, v_3)\}$.

*Мультиграф*

**Визначення.** Якщо множина $E$ містить повторювані елементи, то відповідний граф $G(V, E)$ включає кратні ребра. Тоді його називають мультиграфом.

**Приклад 3.3.** Мультиграф з петлями $G(V, E)$ показаний на рис. 3.8.

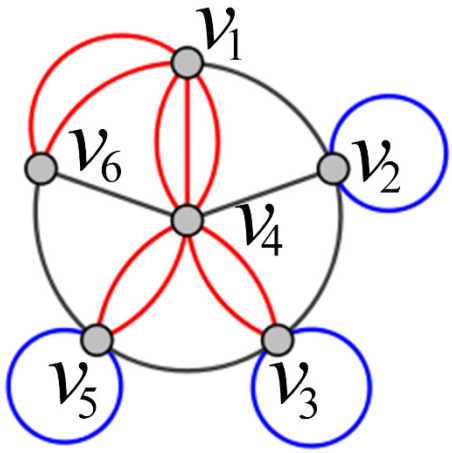



Рис. 3.8. Графічне представлення мультиграфа

Описати цей граф аналітично.

*Розв'язок.*

Множина вершин: $V = \{v_1, v_2, v_2, v_4, v_5, v_6\}$,

Множина ребер:

$E = \{(v_1, v_2),$
$(v_1, v_6), (v_1, v_6),$
$(v_1, v_4), (v_1, v_4), (v_1, v_4),$
$(v_2, v_2), (v_2, v_3), (v_2, v_4),$
$(v_3, v_3), (v_3, v_5),$
$(v_3, v_4), (v_3, v_4),$
$(v_4, v_5), (v_4, v_5),$
$(v_4, v_6), (v_5, v_6), (v_5, v_5)\}$

*Гіперграф*

**Визначення.** Гіпергра́ф – узагальнення графа, в якому ребром називають довільну підмножину вершин графа.

На рис. 3.9 наведено приклад графічного зображення гіперграфа.

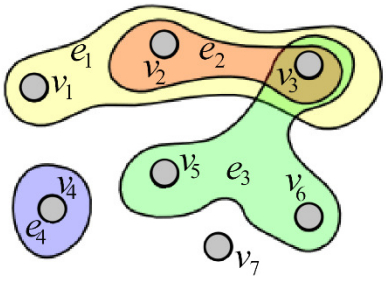



Рис. 3.9. Гіперграф $G(V, E)$

Аналітично гіперграф $G(V, E)$ також представляють кортежем, де $V$ – непорожня множина вершин гіперграфа, $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$;

$E$ – непорожня множина ребер гіперграфа, $E=\{e_1,e_2,e_3,e_4\}$;

Множини вершин, на які спираються ребра гіперграфа:

$e_1=\{v_1,v_2,v_3\}$, $e_2=\{v_1,v_2\}$, $e_3=\{v_3,v_5,v_6\}$, $e_4=\{v_4\}$.

Особливість гіперграфів полягає у тому, що вони можуть містити ребра, які спираються одночасно на кілька вершин (наприклад, ребро $e_3$) і ребра, які спираються тільки на одну вершину (наприклад, ребро $e_4$).

*Повний граф*

**Визначення.** Якщо кожна пара вершин графа $G(V,E)$ з'єднана ребром, то такий граф називають *повним*. Повний граф з $n$ вершин позначають $K_n$.

**Приклад 3.4.** На рис. 3.10 представлені повні графи:

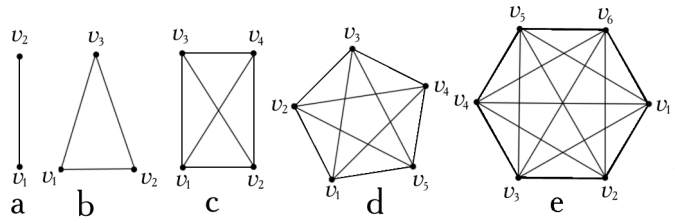



Рис. 3.10. Повні графи a) $K_2$, b) $K_3$, c) $K_4$, d) $K_5$, e) $K_6$.

*Дводольний граф*

**Визначення.** Граф $G(V,E)$ називають *дводольним*, якщо множину його вершин $V$ можна представити об'єднанням множин, які не перетинаються. Нехай $V=A\cup B$, $A\neq\varnothing$, $B\neq\varnothing$, $A\cap B=\varnothing$,

де $A=\{a_1,\ldots,a_i,\ldots,a_n\}$ і $B=\{b_1,\ldots,b_j,\ldots,b_m\}$.

Тоді в дводольному графі існують тільки ребра

$(a_i,b_j)$ або $(b_j,a_i)$, $1\le i\le n$, $1\le j\le m$.

Таким чином, кожне ребро зв'язує вершину, яка належить множині $A$, з вершиною, яка належить множині $B$, але жодні дві вершини з $A$ або дві вершини з $B$ не мають спільних ребер. Приклади графічного представлення дводольних графів показані на рис. 3.11.

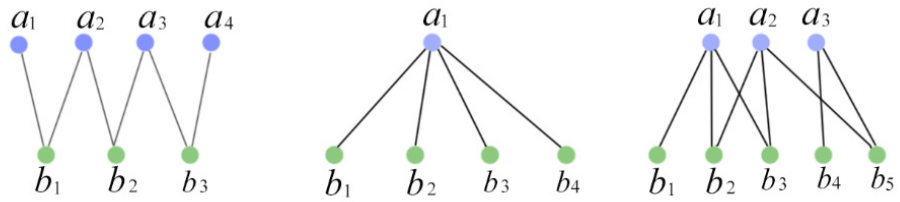


Рис. 3.11. Дводольні графи

*Повний дводольний граф*

**Визначення.** Дводольний граф називають *повним дводольним* графом $K_{m,n}$, якщо $A$ містить $m$ вершин, $B$ містить $n$ вершин і кожна вершина з множини $A$ з'єднана ребром з кожною вершиною з множини $B$.

$$K_{m,n} \to V = A \cup B,\ A \cap B = \varnothing,$$

$$\forall\ a \in A,\ b \in B\ \exists\ (a,b) \in E$$

На рис. 3.12 представлені повні дводольні графи $K_{1,2}, K_{2,3}, K_{2,2}, K_{3,3}$.

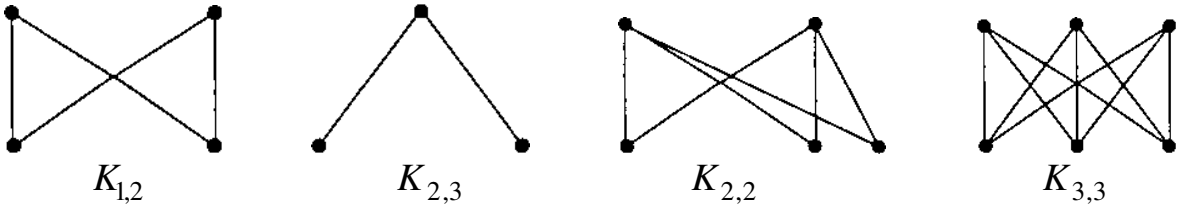



Рис. 3.12. Повні дводольні графи

### 3.1.3. Суміжність

Нехай $v_1 \in V$ і $v_2 \in V$ – вершини,

$e = (v_1, v_2)$ – ребро, що з'єднує вершини $v_1$ та $v_2$, $e \in E$.

Тоді вершина $v_1$ та ребро $e$ інцидентні.

Також вершина $v_2$ та ребро $e$ інцидентні.

**Два ребра**, які інцидентні одній вершині, називають *суміжними ребрами* (рис. 3.13 а).

**Дві вершини**, які інцидентні одному ребру, називають *суміжними вершинами* (рис. 3.13 б).


а б

Рис. 3.13. Суміжні ребра (а) та суміжні вершини (б)

**Визначення.** Множину вершин, суміжних з вершиною $v$, називають ***множиною суміжності вершини*** або відображенням вершини $v$, і позначають $\Gamma(v)$. На рис. 3.14 показано множину вершин $\{u_1, u_2, u_3, u_4, u_5\}$, суміжних з вершиною $v$.

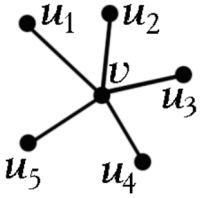



Рис. 3.14. Вершини $\{u_1, u_2, u_3, u_4, u_5\}$, суміжні з $v$

Вершини $\{u_1, u_2, u_3, u_4, u_5\}$ утворюють множину відображення $\Gamma(v)$ вершини $v$. Звідси

$$\Gamma(v) = \{u_i \in V \mid (u_i, v) \in E,\ 0 \le i \le p-1\},\ \text{де}\ p = |V|$$

**Приклад 3.5.** Нехай дано граф $G(V, E)$, показаний на рис. 3.15.

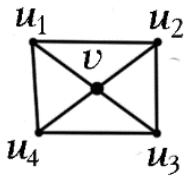



Рис. 3.15. Неорієнтований граф $G(V, E)$

Описати аналітично множини вершин та ребер даного графа та задати множину $\Gamma(v)$ предикатом.

*Розв'язок.*

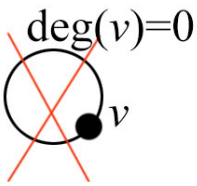
Множина вершин: $V = \{v, u_1, u_2, u_3, u_4\}$, $p = |V| = 5$.

Множина ребер:

$$E = \{(u_1, v), (u_2, v), (u_3, v), (u_4, v), (u_1, u_2), (u_1, u_4), (u_3, u_4), (u_3, u_2)\},\ q = |E| = 8.$$

Множина відображення вершини $v$: $\Gamma(v) = \{u_i \in V \mid (u_i, v) \in E,\ i = 1, \ldots, 4\}$.

### 3.1.4. Степінь вершини




**Визначення.** *Степенем* вершини $v$ називають кількість ребер, інцидентних цій вершині.

***Степінь*** вершини позначають $\deg(v)$ або $d(v)$,

$\forall v \in V\ \ 0 \le \deg(v) \le p-1$, де $p = |V|$.

**Зауваження.** Наявність петлі не збільшує степінь вершини.

***Степінь*** вершини дорівнює потужності множини суміжності: $\deg(v) = |\Gamma(v)|$.

Позначимо **мінімальний степінь** вершини графа $G$ через $\delta(G)$, а **максимальний** – через $\Delta(G)$.

Тоді $\delta(G) = \min\limits_{v \in V} \deg(v)$, $\Delta(G) = \max\limits_{v \in V} \deg(v)$.

Розглянемо характеристики неорієнтованого графа $G(V, E)$, показаного на рис. 3.16.

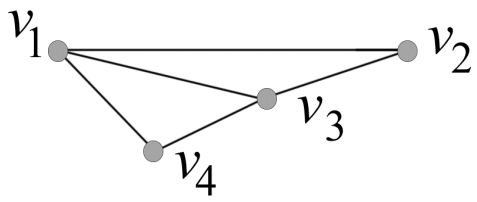



Рис. 3.16. Неорієнтований граф $G(V,E)$

$V=\{v_1,v_2,v_3,v_4\}, p=|V|=4.$

$E=\{(v_1,v_2),(v_1,v_3),(v_1,v_4),(v_2,v_3),(v_3,v_4),\}.$

Вершина $v_1$ характеризується відображенням $\Gamma(v_1)$ і потужністю відображення $|\Gamma(v_1)|$, яка дорівнює степеню даної вершини $\deg(v_1)$:

$$\Gamma(v_1)=\{v_2,v_3,v_4\},\quad |\Gamma(v_1)|=3,\quad \deg(v_1)=3.$$

Вершина $v_2$: $\Gamma(v_2)=\{v_1,v_3\}$, $|\Gamma(v_2)|=2$, $\deg(v_2)=2$.

Вершина $v_3$: $\Gamma(v_3)=\{v_1,v_2,v_4\}$, $|\Gamma(v_3)|=3$, $\deg(v_3)=3$.

Вершина $v_4$: $\Gamma(v_4)=\{v_1,v_3\}$, $|\Gamma(v_4)|=2$, $\deg(v_4)=2$.

Отже, $\delta(G)=\deg(v_2)=\deg(v_4)=2$, $\Delta(G)=\deg(v_1)=\deg(v_3)=3$.

*Регулярний граф*

**Визначення.** Якщо степені всіх вершин дорівнюють $k$, то граф називають ***регулярним графом*** зі степенем $k$.

Приклад регулярного графа $G(V,E)$, для якого $\delta(G)=\Delta(G)=4$, показаний на рис. 3.17.

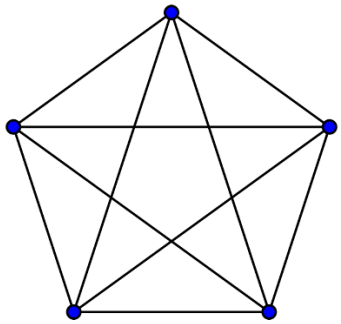


Рис. 3.17. Регулярний граф

Для регулярного $k$-графа справедливе співвідношення: $\delta(G)=\Delta(G)=k$.

**Приклад 3.6.** Визначити, чи є повним графом граф $K_5$.

*Розв'язок.*

Повний граф $K_5$ є регулярним графом, оскільки $\delta\left(K_5\right)=4,\ \ \Delta\left(K_5\right)=4$, як показано на рис. 3.17.

*Ізольована вершина*

**Визначення.** Вершину $v$, для якої $\deg\left(v\right)=0$, називають ***ізольованою.*** На рис. 3.18 наведено приклад графа з ізольованою вершиною $\deg\left(4\right)=0$.

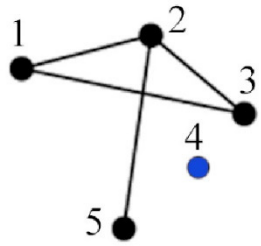



Рис. 3.18. Граф, який містить ізольовану вершину 4

*Висяча вершина*

**Визначення.** Вершину $v$, для якої $\deg\left(v\right)=1$, називають ***кінцевою*** або ***висячою.***

На рис. 3.19 показано граф, який містить ***висячу вершину*** $\deg\left(4\right)=1$.

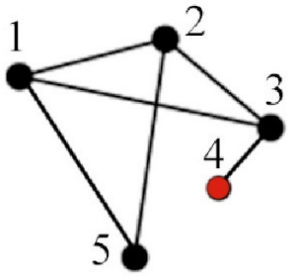



Рис. 3.19. Граф, який містить висячу вершину 4

*Напівстепені у орграфі*

**Визначення.** *Напівстепенем виходу вузла або прямим відображенням* вузла у орграфі називають кількість дуг, які виходять з даного вузла.

Напівстепінь виходу для вузла $v$ позначають як $\deg^+\left(v\right)=\left|\Gamma^+\left(v\right)\right|$.


**Визначення.** *Напівстепенем входу вузла або зворотним відображенням* вузла у орграфі називають кількість дуг, які входять у даний вузол.

Напівстепінь входу для вузла $v$ позначають як $\deg^-\left(v\right)=\left|\Gamma^-\left(v\right)\right|$.


Розглянемо характеристики орієнтованого графа $G\left(V,E\right)$, показаного на рис. 3.20.

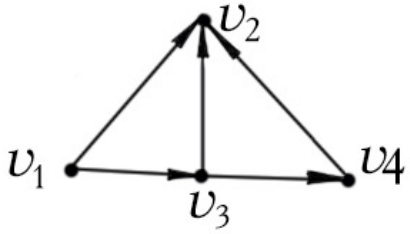



Рис. 3.20. Орієнтований граф $G\left(V,E\right)$

$V=\left\{v_1,v_2,v_3,v_4\right\},\ p=\left|V\right|=4,$

$E=\left\{\left(v_1,v_2\right),\left(v_1,v_3\right),\left(v_3,v_2\right),\left(v_3,v_4\right),\left(v_4,v_2\right),\right\},q=\left|E\right|=5,$

$\Gamma^+\left(v_1\right)=\left\{v_2,v_3\right\},\deg^+\left(v_1\right)=\left|\Gamma^+\left(v_1\right)\right|=2,$

$\Gamma^-\left(v_2\right)=\left\{v_1,v_3,v_4\right\},\ \deg^-\left(v_2\right)=\left|\Gamma^-\left(v_2\right)\right|=3,$

$\Gamma^+\left(v_3\right)=\left\{v_2,v_4\right\},\deg^+\left(v_3\right)=2,$

$\Gamma^-\left(v_3\right)=\left\{v_1\right\},\deg^-\left(v_3\right)=1,$

$\Gamma^+\left(v_4\right)=\left\{v_2\right\},\deg^+\left(v_4\right)=1,$

$\Gamma^-\left(v_4\right)=\left\{v_3\right\},\ \deg^-\left(v_4\right)=1.$

### 3.1.5. Теореми про степені вершин графа

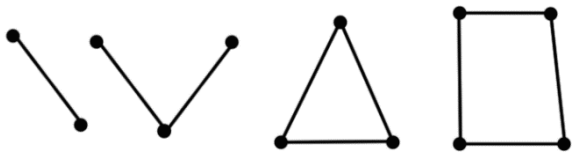
**Теорема 3.1.** Сума степенів вершин графа завжди парна.

*Доведення.* Кожне ребро графа має два кінці. Тому кожне ребро збільшує степінь кожної з 2-х інцидентних вершин на одиницю. Таким чином, кожне ребро збільшує суму степенів усіх вершин на 2. Отже, сума степенів усіх вершин завжди кратна 2, тобто, парна.

**Приклад 3.7.** Зобразити 4 довільних графи та визначити суму їх степенів.

*Розв'язок.*

Розглянемо 4 графи, показані на рис. 3.21, і визначимо суму їх степенів $S_i$, де $i=\overline{1,...,4}$.



$S_1=2,\ S_2=4,\ S_3=6,\ S_4=8.$

Рис. 3.21. Неорієнтовані графи для визначення суми степенів вершин

**Теорема 3.2.** У будь-якому графі кількість вершин непарного степеня парна.

*Доведення.* Доведення виконуватимемо методом від протилежного. Припустимо, що теорема не вірна.

1. Якщо теорема не вірна, то існує непарна кількість вершин, степені яких непарні.

2. Якщо в графі немає вершин з парними степенями як показано на рис. 3.22, то відразу виникає протиріччя з першою теоремою.

Рис. 3.22. Графи з вершинами, які мають тільки непарні степені

Протиріччя полягає в тому, що кількість вершин у цьому випадку повинна бути парною, оскільки сума степенів вершин графа завжди парна.

3. Якщо в графі є вершини з парними і непарними степенями як показано на рис. 3.23, то очевидно, що сума степенів вершин з парними степенями парна.

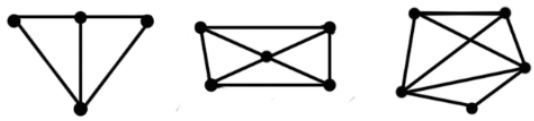


Рис. 3.23. Графи з вершинами, які мають непарні і парні степені

4. Однак, оскільки сума всіх степенів графа парна, то знову виникає протиріччя з початковим припущенням.

Протиріччя полягає в тому, що, оскільки сума непарної кількості і парної кількості є число непарне, то в цьому випадку сума степенів усіх вершин повинна бути непарною.

Але це суперечить теоремі, тому ми прийшли до протиріччя.

Отже, робимо висновок, що теорема доведена.

**Теорема 3.3. Теорема Ейлера.** Сума степенів вершин графа дорівнює подвоєній кількості ребер:

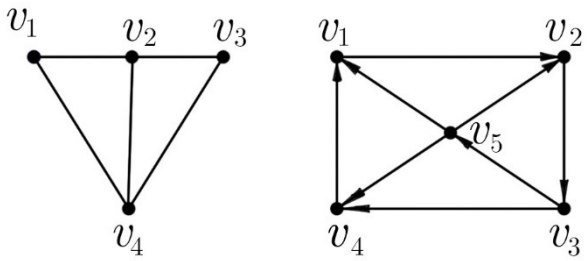
$$\sum_{v\in V}\deg\left(v\right)=2q \text{ – для неорієнтованого графа,}$$

$$\sum_{v\in V}\deg^{-}\left(v\right)+\sum_{v\in V}\deg^{+}\left(v\right)=2q \text{ – для орграфа,}$$

де $q=\left|E\right|$ – потужність множини ребер.

**Доведення**. При підрахунку суми степенів вершин кожне ребро враховується двічі: для одного кінця ребра і для іншого. Звідси випливає, що кількість ребер у графі має дорівнювати половині суми степенів вершин. Теорема доведена.

**Приклад 3.8**. Перевірити задане теоремою Ейлера співвідношення вершин на графах, показаних на рис. 3.24.




$G_1\left(V_1, E_1\right)$ $G_2\left(V_2, E_2\right)$

Рис. 3.24. Графи для визначення співвідношення вершин і ребер

Неорієнтований граф $G_1\left(V_1, E_1\right)$:

$$\sum_{i=1}^{3} \deg\left(v_i\right) = 10,\ q = \left|E\right| = 5,$$

Орієнтований граф $G_2\left(V_2, E_2\right)$:

$$\sum_{i=1}^{5} \deg^{-}\left(v_i\right) = 8,\ \sum_{i=1}^{5} \deg^{+}\left(v_i\right) = 8,\quad q = \left|E\right| = 8.$$

### 3.1.6. Графи з постійним і змінним степенем вершин

Якщо граф регулярний, то характеристикою такого графа є степінь графа, а не степені вершини.

У регулярному графі степінь регулярності є *інваріантом* (постійною властивістю) графа і позначається $r\left(G\right)$.

**Приклад 3.9.** На рис. 3.25 показаний $G\left(V, E\right)$ граф. Визначити регулярність та його степінь $r\left(G\right)$.

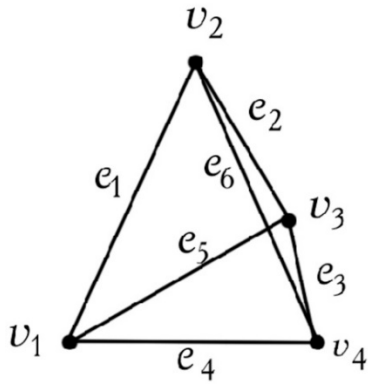



Рис. 3.25. Граф $G\left(V, E\right)$

*Розв'язок.*

$G\left(V, E\right)$, де $V = \left\{v_1, v_2, v_3, v_4\right\}$, $E = \left\{e_1, e_2, e_3, e_4, e_5, e_6\right\}$.

$r\left(G\right) = \deg\left(v_1\right) = \deg\left(v_2\right) = \deg\left(v_3\right) = \deg\left(v_4\right) = 3.$

Всі степені вершин графа є однаковими. Тому граф $G(V,E)$ – це регулярний граф. Степінь графа $r(G)=3$.

Для нерегулярних графів, тобто графів зі змінним степенем вершин, значення $r(G)$ **не визначене**.

Існують класичні приклади регулярних графів, що мають назви
a) 0-регулярний граф,
b) 1-регулярний граф,
c) 2-регулярний граф
d) 3-регулярний граф.
Зображення цих графів показані на рис. 3.26.

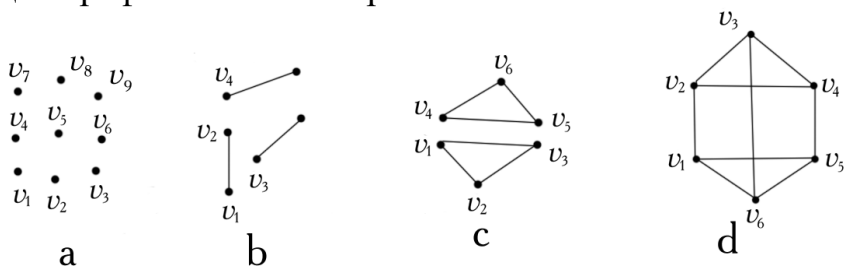



Рис. 3.26. Приклади регулярних графів

### 3.1.7. Підграф графа

**Визначення.** Граф $G'(V',E')$ називають ***підграфом*** графа $G(V,E)$ $G'(V',E')\preceq G(V,E)$, якщо $V'\subseteq V$ і $E'\subseteq E$.

Отже,
- кожна вершина в $G'$ є одночасно вершиною в $G$,
- кожне ребро в $G'$ є одночасно ребром в $G$.

**Визначення.** Граф $G'(V',E')$ називають ***остовним підграфом*** $G$ або **суграфом** графа $G$, якщо $V'=V$ і $E'\subseteq E$,

Отже,
- множини вершин графа $G'$ та графа $G$ співпадають,
- кожне ребро в $G'$ є одночасно ребром в $G$.

**Визначення.** Граф $G'(V',E')$ називають ***правильним підграфом*** графа $G$, якщо $V'\subset V$ і $G'$ містить усі можливі ребра $G$:

$$\forall u,v\in G'(u,v) \text{ таких, що } (u,v)\in E \Rightarrow (u,v)\in E'.$$

Правильний підграф утворюють шляхом виключення з графа певної кількості вершин та інцидентних до них ребер.

**Приклад 3.10.** На рисунку (**a**) показаний граф $G(V,E)$. Визначити, якими є його підграфи (b), (c) і (d).

*Розв'язок.*

(**b**). На рисунку представлений **підграф** $G_1(V_1,E_1)$ графа $G(V,E)$, оскільки $V_1 \subset V$ і $E_1 \subset E$.

(**c**). Граф $G_2(V_2,E_2)$ є **остовним графом** або суграфом графа $G(V,E)$, тому що $V_2 = V$ й $E_2 \subset E$.

(**d**). Граф $G_3(V_3,E_3)$ є **правильним підграфом** графа $G(V,E)$, оскільки містить усі його можливі ребра.

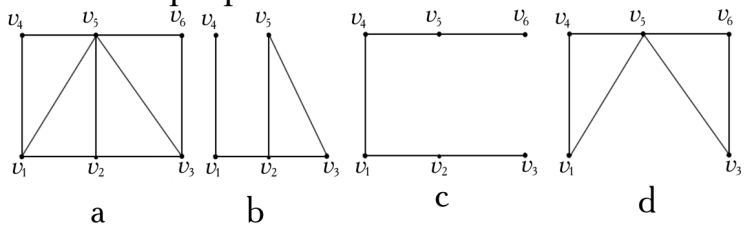



Рис. 3.27. Граф $G(V,E)$ (a) та його підграфи (b), (c) і (d)

### 3.1.8. Циркулянтні графи

*Циркулянтні графи* – це об'єкти, які широко застосовуються в сучасній комп'ютерній техніці і дискретній математиці.

Вони використовуються в *обчислювальних структурах, мережах передачі даних* і *розподілених обчисленнях*.

Циркулянтні графи були реалізовані вперше як комунікаційні мережі в таких легендарних обчислювальних системах як ILLIAC-IV, MPP, Cray T3D.

Зараз циркулянтні графи розглядають як основи конфігурації різного роду **кластерних систем.**

Циркулянтні графи також застосовують у **теорії кодування** при створенні кодів, які виправляють помилки.

*Визначення циркулянтного графа* $S$

Нехай $s_1, s_2, \ldots, s_m, \ldots, s_k, n$ – цілі числа, такі, що задовольняють умови:

$$1 \le s_1 < s_2 < \ldots < s_m < \ldots < s_k < n.$$

*Циркулянтним* графом будемо називати граф з множиною вершин

$$V = \{0,1,2,\ldots,n-1\}$$

і множиною ребер, яка сформована за таким правилом:

$$E = \left\{ (i,j) \middle| \left( |i-j| \bmod n = s_m \right), m = 1,2,\ldots,k \right\},$$

де $i = 0,1,\ldots,n-1$, $j = 0,1,\ldots,n-1$, $s_m \in S$.

Цей запис означає, що між вершиною $i$ та вершиною $j$ тільки тоді існує ребро, якщо справедливий вираз $\left(\left|i-j\right| \bmod n = s_m\right)$, де $s_m$ – це одна з заданих констант $s_m = \{1,2,\ldots\}$

$$E = \left\{(i,j)\middle|\left(\left|i-j\right| \bmod n = s_m\ \right), m = 1,2,\ldots,k\right\},$$

Число $n$ називають порядком циркулянтного графа. Число $k$ – розмірність циркулянтного графа. Елементи $s_m \in S$ – утворюючі циркулянтного графа (хорди). Циркулянтний граф прийнято задавати у вигляді параметричного опису

$$G\left(n;S\right) = G\left(n;s_1,s_2,\ldots,s_k\right),$$

порядок, що задає розмірність і значення утворюючих.

*Степінь циркулянтного графа*

Степінь вершин графа $G\left(n;s_1,s_2,\ldots,s_k\right)$ дорівнює:

- $2k$ у випадку, коли $s_k \neq \dfrac{n}{2}$
- $\left(2k-1\right)$ у випадку, коли $n$ – парне і $s_k = \dfrac{n}{2}$.

**Приклад 3.11.** Сформувати кільцевий циркулянтний граф $G\left(16;1,4\right)$, визначити степінь графа та записати множини його вершин і ребер.

*Розв'язок.*

Циркулянтний граф $G\left(16;1,4\right)$ має вигляд, як показано на рис. 3.28:

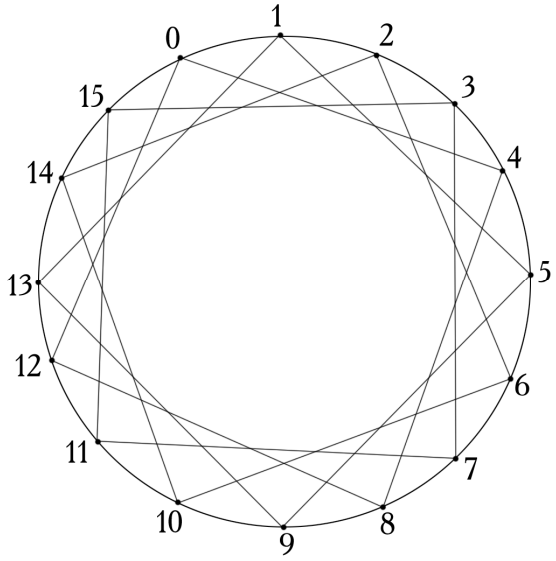



Рис. 3.28. Циркулянтний граф $G\left(16;1,4\right)$

Множина вершин: $V = \left\{0,1,2,\ldots,15\right\}$.

Основні константи графа: $s_1 = 1,\ s_2 = 4,\ n = 16$.

Множина ребер:

$E = \{(0,1),(1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,8),...,(14,15),(15,0),$
$(0,4),(1,5),(2,6),(3,7),(4,8),...,(11,15),(12,0)\}$.

Степінь графа: $2k = \dfrac{16}{2} = 8$. Звідси $k = 4$.

### 3.1.9. Структурні характеристики графів

*Маршрут (шлях)*

**Визначення.** *Маршрутом* або *шляхом* у графі $G(V,E)$ називають послідовність вершин і ребер, які чергується:

$$v_0, e_1, v_1, ..., v_{t-1}, e_t, v_t, \text{ де } e_i = (v_{i-1}, v_i) \text{ при } 1 \le i \le t.$$

Такий маршрут коротко називають $(v_0, v_t)$-маршрутом і говорять, що він з'єднує $v_0$ з $v_t$, які називають *кінцевими вершинами даного маршруту*.

Найчастіше маршрут зображують у вигляді послідовності вершин і ребер, як показано на рис. 3.29:

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \dots \xrightarrow{e_t} v_t$$

Рис. 3.29. Маршрут $(v_0, v_t)$

Відзначимо, що стрілки тут указують лише порядок проходження вершин у маршруті.

**Визначення.** ***Довжиною маршруту (шляху)*** називають *кількість ребер, що входять в нього.* Випадок, коли довжина маршруту **дорівнює нулю**, не виключається; у цьому випадку маршрут зводиться до однієї вершини.

Відзначимо, що **у звичайному графі маршрут** (шлях) **повністю визначається послідовністю** $v_0, v_1, ..., v_t$ своїх вершин.

Якщо $v_0 = v_t$, то $(v_0, v_t)$-маршрут називають ***замкненим***.

У **довільному маршруті** (шляху) **будь-яке ребро і будь-яка вершина можуть повторюватися**. Накладаючи обмеження на число повторень вершин або ребер, ми приходимо до наступних окремих видів маршрутів (шляхів).

*Ланцюг*

***Ланцюг*** – це шлях через ребра, які не повторюються.

Ланцюг називають ***простим ланцюгом***, якщо в ньому немає повторюваних вершин, крім, можливо, початкової і кінцевої вершин, які співпадають (рис. 3.30).

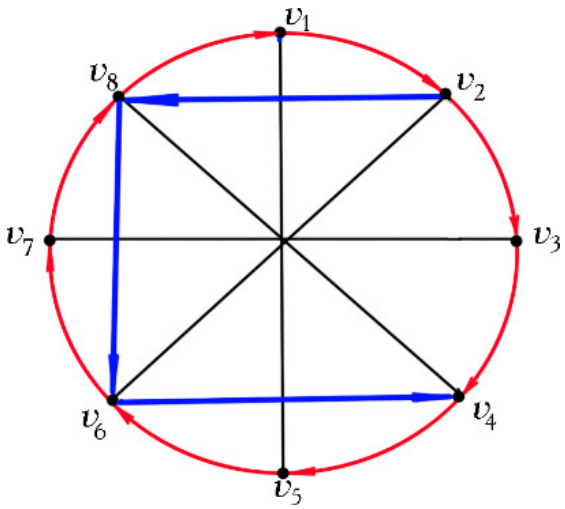



Рис. 3.30. Простий ланцюг у графі

*Цикл*

**Визначення.** Замкнений простий ланцюг називають ***циклом***.

**Цикл** повністю визначають множиною його ребер.

Тому часто під циклом ми будемо розуміти відповідну йому множину ребер.

**Петля** дає цикл довжини 1.

**Пара кратних ребер** утворює цикл довжини 2.

**Цикли довжини 3** зазвичай називають *трикутниками.*

а б в

Рис. 3.31. Графи з петлею (а), пара ребер (б), трикутник (в)

**Лема.** Якщо для деяких двох вершин $u$ і $v$ в графі існує $(u,v)$-маршрут, то існує і простий $(u,v)$-ланцюг.

**Доведення.** Розглянемо в графі (рис. 3.32) $(u,v)$-маршрут найменшої довжини.

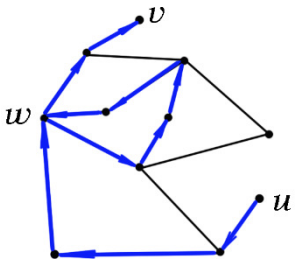


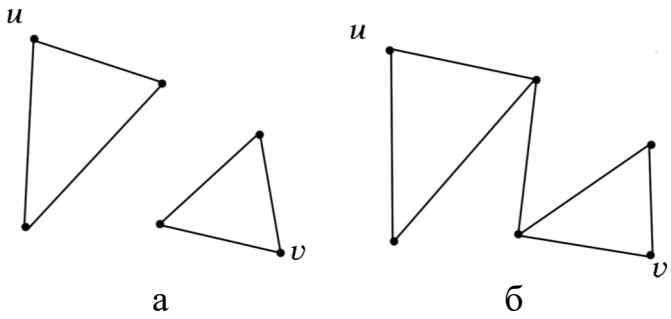

Рис. 3.32. Граф з маршрутом $(u,v)$

Покажемо, що цей маршрут є простим ланцюгом. Якщо в ньому є повторювана вершина $w$, то, заміняючи частину маршруту від першого входження вершини $w$ до її другого входження на одну вершину $w$, ми одержимо більш короткий $(u,v)$-маршрут.

### 3.1.10. Зв'язність графа

Граф $G$ називають ***зв'язним***, якщо для будь-яких його двох різних вершин $u$ і $v$ існує $(u,v)$-маршрут (рис. 3.33 а).

Якщо для графа $G$ можна вказати пари вершин $u$ і $v$, між якими не існує маршруту, то такий граф називають **незв'язним** (рис. 3.31 б).




а б

Рис. 3.33. Графи: незв'язний (а) та зв'язний (б)

*Теорема про незв'язний граф*

**Теорема 3.4.** Граф є незв'язним тоді і тільки тоді, коли множину його вершин $V$ можна розбити хоча б на дві непусті підмножини $V_1$ і $V_2$ так, щоб будь-яке ребро графа з'єднувало тільки ті вершини, які належать одній підмножині.

*Доведення.*

На множині вершин $V$ графа $G$ визначимо *відношення зв'язності* ~ вважаючи, що

$$u \sim v \Leftrightarrow \text{існує } (u,v)\text{-маршрут.}$$

Дане відношення є відношенням еквівалентності (рефлексивне, симетричне і транзитивне).

Позначимо через $G_i = G(V_i)$ підграфи, породжені множиною вершин $V_i$ при $1 \le i \le k$, як показано на рис. 3.34.

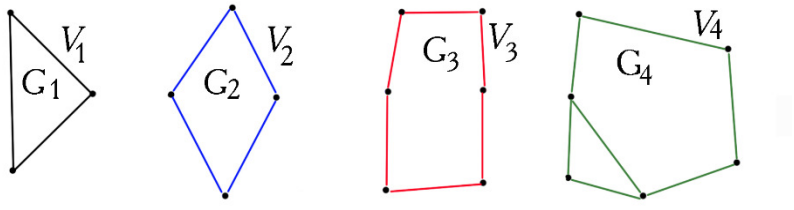



Рис. 3.34. Незв'язний граф $G$, який складається з підграфів $G_1, G_2, G_3, G_4$

Підграфи $G_1, G_2, \ldots, G_k$, називають *компонентами зв'язності* графа $G$. Ясно, що кожний компонент зв'язності $G_i$ є зв'язним підграфом. Тому множина компонентів зв'язності $G = \{G_1, G_2, \ldots, G_k\}$ – це множина всіх зв'язних підграфів даного графа, і будь-яке ребро належить деякому компоненту зв'язності. Таким чином справедливе наступне твердження:

**Кожний незв'язний граф є диз'юнктивним об'єднанням своїх компонентів зв'язності.**

*Властивості зв'язності графів*

1. Кожна вершина графа входить в один і тільки в один компонент зв'язності.
2. Будь-який скінченний граф має скінченну кількість компонентів зв'язності.
3. Граф, що складається з єдиного компонента зв'язності, є зв'язним.
4. Кожний компонент зв'язності графа є його підграфом.
5. Для будь-якого графа або він сам, або його доповнення є зв'язним.

При явному визначенні компонентів зв'язності граф описують трійкою, як $(p, q, k)$-граф, де $p$ – кількість вершин графа, $q$ – кількість ребер графа, а $k$ – кількість компонентів зв'язності.

**Приклад 3.12.** Визначити трійку, яка описує граф, показаний на рис. 3.34.

*Розв'язок.*

Для даного графа $G$ характерні такі параметри:

$p = |V_1| + |V_2| + |V_3| + |V_4| = 3 + 4 + 6 + 6 = 19$

$q = |E_1| + |E_2| + |E_3| + |E_4| = 3 + 4 + 6 + 7 = 20$

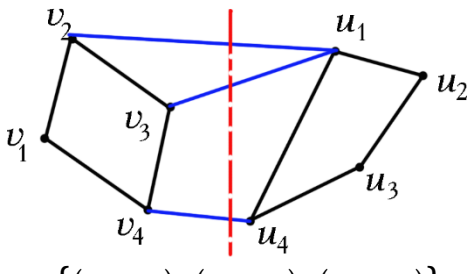
$k = 4$

Отже, $G = G(19, 20, 4)$.

### 3.1.11. Множина розрізання, розріз і міст

**Визначення.** *Множиною розрізання* називають множину ребер, видалення яких з графа приводить до збільшення компонентів зв'язності.

На рис. 3.35 вертикально розташований відрізок перетинає ті ребра графа, що входять в множину розрізання.




$\{(v_2, u_1), (v_3, u_1), (v_4, u_4)\}$

Рис. 3.35. Розрізання графа

**Визначення.** Мінімальну за включенням ребер множину називають *розрізом* графа.

**Визначення.** *Міст* – це розріз, що складається з єдиного елемента.

Показані на рис. 3.36 графи демонструють множину розрізання (a); розріз (b) і міст (c).

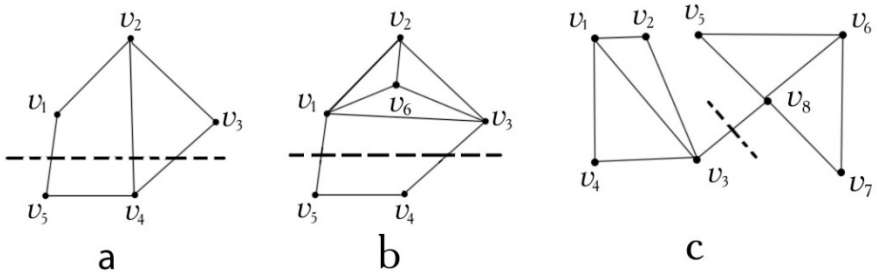



Рис. 3.36. Множина розрізання (a); розріз (b) і міст (c)

a) Множина розрізання графа складається з ребер $E_r = \{(v_1, v_5), (v_4, v_3), (v_2, v_4)\}$. Ця множина не є мінімальною за включенням, оскільки два рази включає ребро $(v_1, v_3)$.

b) Приклад розрізу графа, що містить множину розрізання, мінімальну за включенням $E_r = \{(v_1, v_5), (v_2, v_4), (v_3, v_4)\}$.

c) Міст графа представлений множиною розрізання з одного елемента: $E_r = \{(v_3, v_8)\}$.

**Контрольні запитання**

1. Нехай дано зв'язний граф $G(V, E)$. Визначити величину загального степеня вершин графа за умови, що $|V| = 5$ і $|E| = 8$. Побудувати варіант графа з максимально можливим степенем вершини.
2. Позначимо через $n$ – кількість будинків і $m$ – кількість доріг. Будинки пронумеровані від 1 до $n$. Кожна дорога визначається двійкою чисел – двома номерами будинків – кінців дороги. У кожному будинку живе по одній людині. Знайти точку – місце зустрічі всіх людей, від якої сумарна відстань до всіх будинків буде мінімальною за умови, що довжина усіх доріг однакова.
3. Побудувати циркулянтний граф $G(32; 2, 8)$ і знайти множину двійок вершин, відстань між якими є максимальною.
4. Побудувати підграф, правильний підграф та остовний підграф графа, показаного на рис. 3.37.

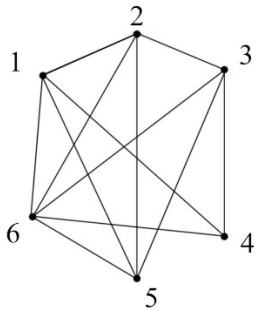


Рис. 3.37. Граф до задач 4 та 6

5. Побудувати незв'язний граф, який містить 12 вершин, 3 компоненти і мінімально можливу кількість вершин.
6. На графі, показаному на рис. 3.37, знайдіть всі можливі розрізи.

## 3.2. Способи задавання й властивості графів

### 3.2.1. Операції з елементами графів

1. *Операція видалення ребра*

Нехай $G = (V, E)$ – граф і $e \in E$ – деяке його ребро. Граф $G_1 = G - e$ одержуємо із графа $G$ шляхом видалення ребра $e$ за умови, що $G_1 = (V, E \setminus \{e\})$. Таким чином, видалення ребра графа не викликає зміни кількості його вершин.

**Властивості операції видалення ребра.** Нехай необхідно вилучити ребра $e_1 \in E$ і $e_4 \in E$. Тоді справедливий закон асоціативності: $(G - e_1) - e_4 = (G - e_4) - e_1$.

Якщо підряд виконується кілька операцій видалення ребер, то результат **не залежить від порядку видалення**. На рис. 3.38 показано 2 різні послідовності видалення ребер.

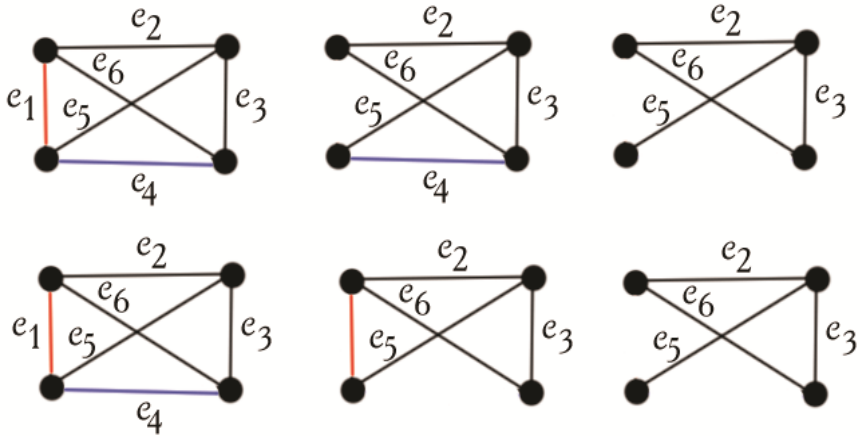


Рис. 3.38. Різні послідовності видалення ребер $e_1$ і $e_4$

2. *Операція видалення вершини*

Нехай $G=(V,E)$ і $v\in V$ – деяка вершина графа $G$. Граф $G_2=G-v$ одержуємо із графа $G$ шляхом видалення вершини $v$ із множини вершин $V$ і видалення всіх інцидентних з вершиною $v$ ребер з множини ребер $E$.

Таким чином, видалення вершин графа може викликати зміну кількості його ребер.

**Властивості операції видалення вершини.** Нехай необхідно вилучити вершини $v_1\in V$ й $v_5\in V$. Тоді виконується закон асоціативності: $(G-v_1)-v_5=(G-v_5)-v_1$. Якщо підряд виконується кілька операцій видалення вершин, то результат ***не залежить від порядку видалення***. На рис. 3.39 показано процес послідовного видалення вершин.

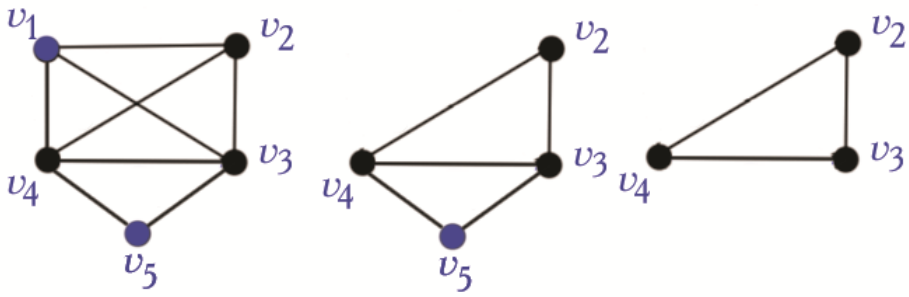



Рис. 3.39. Видалення вершин $v_1$ і $v_5$

3. *Операція введення ребра*

Нехай $G=(V,E)$ і існують дві вершини $u\in V$ і $v\in V$, і $(u,v)\notin E$. Тоді операція введення ребра може бути представлена виразом:

$G_3=G+e=(V,E\cup\{e\})$, де $e=(u,v)$.

**Властивості операції введення ребра.** Виходячи із властивостей комутативності та асоціативності операції об'єднання, можна стверджувати, що при введенні в граф декількох ребер результат не залежить від порядку їх додавання.

$$(G+e)+e_1=(G+e_1)+e,\ \text{де}\ e\in E\ \text{і}\ e_1\in E.$$

4. *Операція введення вершини в ребро*

Нехай дано граф $G=(V,E)$, який включає вершини $v\in V$ і $u\in V$, а також ребро $(v,u)\in E$, яке їх з'єднує. Операція введення вершини в ребро може бути представлена виразом:

$$G_4=\left(V\cup\{w\},\left(E\cup\{(v,w)\}\cup\{(w,u)\}\right)\setminus\{(v,u)\}\right).$$

До множини $V$ додають вершину $w$, до множини $E$ додають ребра $(v,w)$ і $(w,u)$, а ребро $(v,u)$ видаляють з множини $E$. На рис. 3.40 показано етапи введення вершини в ребро.

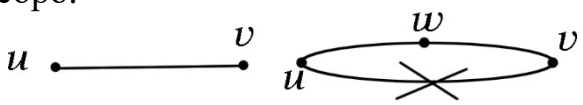



Рис. 3.40. Операція введення вершини $w$ в ребро $(u,v)$

5. *Ототожнення (злиття) вершин*

Нехай дано граф $G=(V,E)$, що включає вершини $v\in V$ і $u\in V$ з відповідними множинами суміжності $\Gamma(v)=\{v_1,v_2,...,v_m,u\}$ і $\Gamma(u)=\{u_1,u_2,...,u_k,v\}$.

Злиття вершин $v$ і $u$ виконують у два етапи:

1. Виключають вершини $v$ і $u$ з графа $G$: $G'=G-v-u$
2. Додають до отриманого графа вершину $u'$ з такою множиною суміжності: $\Gamma(u')=\Gamma(v)\setminus u\cup\Gamma(u)\setminus v$: $H=G'+u'$.

На рис. 3.41 показано процес злиття двох вершин у графі $G$.

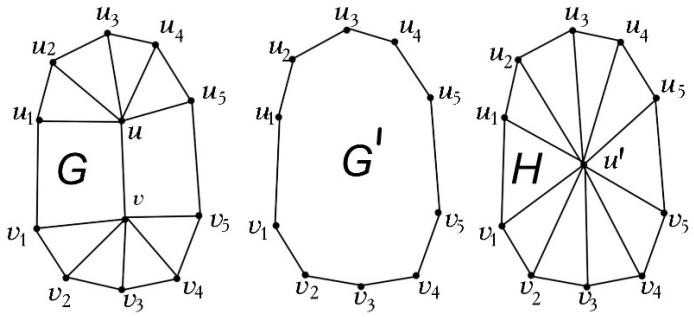



Рис. 3.41. Операція злиття вершин $u$ і $v$ у вершину $u'$

### 3.2.2. Задавання графів у математиці

У дискретній математиці прийнято розглядати три способи задавання графів:

1. *Аналітичний спосіб*

Аналітичний спосіб припускає представлення графа $G(V,E)$ у **вигляді множин** $V$ і $E$. Для задавання цих множин можуть використовуватися всі три способи задавання множин. На рис. 3.42 показано граф $G$ для математичного представлення.

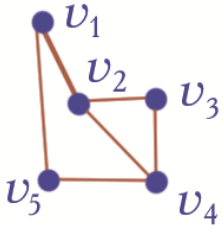



Рис. 3.42. Граф $G$

Явний спосіб у вигляді кортежу $G(V,E)$, де $V=\{v_1,v_2,v_3,v_4,v_5\}$ – множина вершин, $E=\{(v_1,v_2),(v_2,v_3),(v_2,v_4),(v_3,v_4),(v_4,v_5),(v_5,v_1)\}$ – множина ребер.

Задавання предикатом: $V=\{v_i \mid i=1,\ldots,n\}$,

$E=\{(v_i,v_j) \mid i=2k+1, j=2k, k=1,\ldots 2n-1\}$

Рекурсивною процедурою: $V=\{v_i \mid i=i+1, i<m\}$

$E=\{(v_i,v_j) \mid j=j+1, i=i+2, i,j<n\}$

2. *Графічний спосіб*

**Вершини** представлені **точками**, а **ребра – лініями**, які з'єднують ці точки. На рис. 3.43 показані способи графічного представлення різних графів.

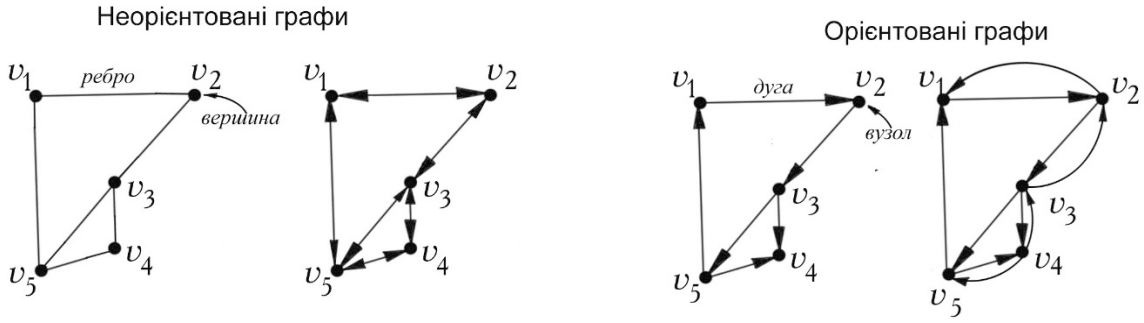



Рис. 3.43. Способи графічного представлення орієнтованих і неорієнтованих графів

3. *Матричний спосіб*

Граф задають у вигляді **матриці інцидентності** або **матриці суміжності**.

*Задавання неорієнтованого графа за допомогою матриці інцидентності*

Нехай $G$ – **неорієнтований** граф. Нехай $B$ – матриця, кожний **рядок** якої **відповідає вершині** графа, а кожний **стовпець відповідає ребру** графа.

$$B = \begin{pmatrix} & e_1 & e_2 & \dots & e_j & \dots & e_m \\ v_1 & 1 & 0 & 0 & 0 & 0 & 1 \\ v_2 & 0 & 1 & 0 & 1 & 0 & 0 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 \\ v_i & 0 & 0 & 0 & b_{ij} & 0 & 0 \\ \dots & 0 & 1 & 0 & 0 & 0 & 0 \\ v_n & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$G = (V, E),\ V = \{v_1, v_2, \dots, v_i, \dots, v_n\},\ E = \{e_1, e_2, \dots, e_j, \dots, e_m\}.$

Елемент $i$-го рядка та $j$-го стовпця матриці $B$ позначають $b_{ij}$. $b_{ij} = 1$, якщо $i$-та вершина інцидентна $j$-му ребру, і дорівнює 0 у протилежному випадку.

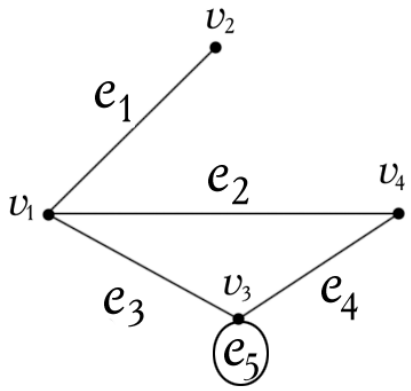



Рис. 3.44. Графічне представлення графа, заданого матрицею $B$

Матрицю $B$ називають *матрицею інцидентності* неорієнтованого графа $G$.

Отже, елементи матриці інцидентності $B = (b_{ij})$ задають формулою:

$$b_{ij} = \begin{cases} 1, & \text{якщо ребро } e_i \text{ інцидентне вершині } v_j, \\ 0, & \text{в протилежному випадку.} \end{cases}$$

**Приклад 3.13.** Граф $G = (V, E)$ задано аналітично множинами

$V = \{v_1, v_2, v_3, v_4\},$

$E = \{e_1, e_2, e_3, e_4, e_5\} = \{(v_1, v_2), (v_1, v_4), (v_1, v_3), (v_3, v_4), (v_3, v_3)\}.$

Сформувати матрицю інцидентності та зобразити даний граф графічно.

*Розв'язок.*

Матриця інцидентності графа $G = (V, E)$ має вигляд:

| | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ |
|---|---|---|---|---|---|
| $v_1$ | 1 | 1 | 1 | 0 | 0 |
| $v_2$ | 1 | 0 | 0 | 0 | 0 |
| $v_3$ | 0 | 0 | 1 | 1 | 1 |
| $v_4$ | 0 | 1 | 0 | 1 | 0 |

або у більш формалізованому вигляді:

$$B = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Графічне представлення графа, заданого матрицею $B$, відповідає зображенню графа на рис. 3.44.

*Властивості матриці інцидентності неорієнтованого графа*

1. Для вершин без петель **степінь вершини дорівнює сумі одиничних елементів** відповідного рядка матриці, оскільки кожна одиниця в цьому рядку представляє інцидентність цієї вершини ребру.

2. У **кожному стовпці**, який не представляє ребро петлі, **будуть дві одиниці**, тому що кожне таке ребро інцидентне двом вершинам.

3. У рядку матриці інцидентності, який відповідає вершині **з петлею**, сума **одиниць на одну більше** степеня даної вершини.

4. Стовпець, що відповідає **ребру петлі**, містить тільки **одну одиницю**.

*Задавання орієнтованого графа за допомогою матриці інцидентності*

Нехай $G$ – **орієнтований** граф. Тоді матриця інцидентності $B = \left(b_{ij}\right)$ включає елементи, які дорівнюють 1, якщо вершина інцидентна з початком ребра, дорівнюють -1, якщо вершина інцидентна з кінцем ребра, дорівнюють 0, якщо вершина і ребро не інцидентні, дорівнюють 2 або будь-якому числу, якщо вершина містить петлю.

$$b_{ij} = \begin{cases} 1, \text{ якщо вершина } v_j \text{ є початком ребра } e_i, \\ -1, \text{ якщо вершина } v_j \text{ є кінцем ребра } e_i, \\ 2, \text{ якщо вершина } v_j \text{ є початком і кінцем ребра } e_i, \\ 0, \text{ в інших випадках.} \end{cases}$$

**Приклад 3.14.** Нехай задано орієнтований граф $G=\left(V,E\right)$, де $V=\left\{v_1,v_2,v_3,v_4\right\}$ і $E=\left\{e_1,e_2,e_3,e_4,e_5,e_6\right\}$. Описати даний граф матрицею та зобразити графічно.

*Розв'язок.*

Графічне представлення даного орграфа показано на рис. 3.45:

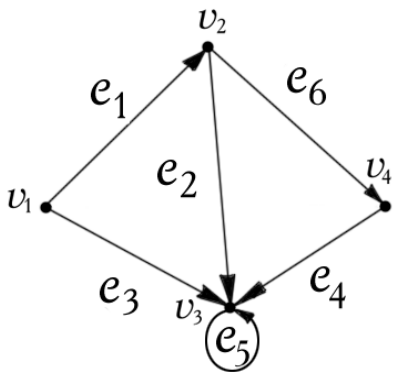



Рис. 3.45. Орграф $G=\left(V,E\right)$

Матриця інцидентності орграфа має вигляд:

| | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ |
|---|---|---|---|---|---|---|
| $v_1$ | 1 | 0 | 1 | 0 | 0 | 0 |
| $v_2$ | −1 | 1 | 0 | 0 | 0 | 1 |
| $v_3$ | 0 | −1 | −1 | −1 | 2 | 0 |
| $v_4$ | 0 | 0 | 0 | 1 | 0 | −1 |

або

$$B=\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}$$

*Властивості матриці інцидентності орграфа*

1. Для вершин без петель напівстепінь виходу **дорівнює сумі додатних одиничних елементів** відповідного рядка
2. Для вершин без петель напівстепінь входу **дорівнює сумі від'ємних одиничних елементів** відповідного рядка.
3. У **кожному стовпці**, який не представляє ребро петлі, сума елементів дорівнює нулю.
4. Якщо дуга – це петля, то в стовпці один елемент, який дорівнює 2.

*Задавання неорієнтованого графа за допомогою матриці суміжності*

Нехай $G$ – неорієнтований граф. Нехай $C$ – матриця, **рядки якої позначені вершинами графа** і **стовпці позначені тими ж вершинами** в тому ж самому порядку.

Елемент $i$-го рядка й $j$-го стовпця матриці $C$ позначається $c_{ij}$, і

- дорівнює 1, якщо існує одне ребро з $i$-ї вершини в $j$-у вершину,
- дорівнює числу ребер з $i$-ї вершини в $j$-у вершину за наявності декількох ребер,
- дорівнює 0, якщо ребер між вершинами не існує.

$$C = \begin{pmatrix} & v_1 & v_2 & v_i & v_j & v_n \\ v_1 & 0 & 0 & 1 & 0 & 1 \\ v_2 & 0 & 0 & 1 & 1 & 1 \\ v_i & 1 & 1 & 0 & c_{ij} & 1 \\ v_j & 0 & 1 & c_{ji} & 0 & 0 \\ v_n & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Матрицю $C$ називають матрицею суміжності графа G.

Формула для визначення елементів матриці суміжності графа:

$$c_{ij} = \begin{cases} 1, \textit{ якщо існує ребро } (v_i, v_j), \\ k, \textit{ якщо існують ребра } \{\overbrace{(v_i, v_j), (v_i, v_j), ..., (v_i, v_j)}^{k}\} \\ 0, \textit{ в інших випадках.} \end{cases}$$

**Приклад 3.15.** Розглянемо неорієнтований граф $G(V, E)$ показаний на рис. 3.46.

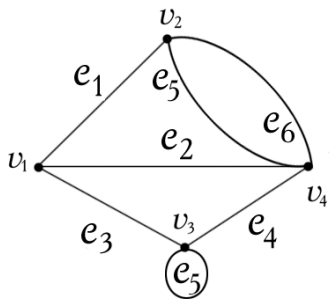



Рис. 3.46. Неорієнтований граф $G(V, E)$

Сформувати матрицю суміжності для даного графа.

*Розв'язок.*

Матриця суміжності даного графа має вигляд:

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 1 & 1 & 1 \\ v_2 & 1 & 0 & 0 & 2 \\ v_3 & 1 & 0 & 1 & 1 \\ v_4 & 1 & 2 & 1 & 0 \end{array} \quad \text{або } C = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 2 \\ 1 & 0 & 1 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix}$$

*Властивості матриці суміжності*

1. Матриця суміжності неорієнтованого графа симетрична відносно головної діагоналі.

2. Якщо вершина має петлі, то їх число розміщається на головній діагоналі матриці суміжності.

3. Якщо між двома вершинами графа існує кілька ребер, то на перетині рядків і стовпців проставляють їх кількість.

*Матриця суміжності орієнтованого графа*

Нехай $G$ – орієнтований граф. Нехай $C$ – матриця, **рядки якої позначені вершинами графа і стовпці позначені тими ж вершинами** в тому ж самому порядку.

Елемент $i$-ого рядка й $j$-го стовпця матриці $C$, позначається $c_{ij}$, і

- дорівнює 1, якщо ребро виходить із вершини $v_i$, представленої $i$-м рядком і входить у вершину $v_j$, представлену $j$-м стовпцем матриці.
- дорівнює числу ребер з $i$-ї вершини в $j$-у вершину за наявності декількох ребер,
- дорівнює 0, якщо ребер між вершинами не існує.

Матрицю $C$ називають *матрицею суміжності* орграфа $G$.

**Приклад 3.16.** Розглянемо орієнтований граф $G(V,E)$, що показаний на рис. 3.47:

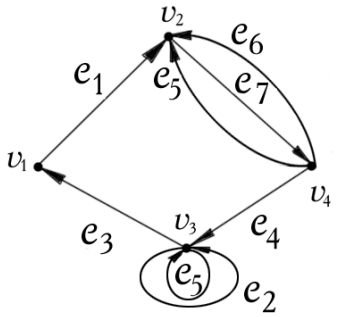



Рис. 3.47. Орієнтований граф $G(V,E)$

Сформувати матрицю суміжності для даного графа.

*Розв'язок.*

Його матриця суміжності має вигляд:

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 1 & 0 & 0 \\ v_2 & 0 & 0 & 0 & 1 \\ v_3 & 1 & 0 & 2 & 0 \\ v_4 & 0 & 2 & 1 & 0 \end{array} \text{ або } C = \begin{Vmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 2 & 1 & 0 \end{Vmatrix}.$$

*Властивості матриці суміжності орієнтованого графа*

1. Матриця суміжності несиметрична відносно головної діагоналі.
2. Сума чисел у рядку матриці суміжності без урахування чисел на головній діагоналі дозволяє визначити потужність *напівстепеня виходу* для кожної вершини орграфа: $\deg^+(v_i)$, де $1 \le i \le n$.
3. Сума чисел у стовпці матриці суміжності без урахування чисел на головній діагоналі дозволяє визначити потужність *напівстепеня входу* для кожної вершини орграфа: $\deg^-(v_i)$, де $1 \le i \le n$.

*Задавання графа за допомогою списку ребер*

Список ребер графа (орграфа) представлено двома стовпцями. Перший стовпець містить ребра, а другий – інцидентні з ними вершини. Для неорієнтованого графа порядок проходження вершин довільний. Для орграфа першим стоїть номер вершини, з якої ребро виходить. Список ребер графа (орграфа) може також бути представлений послідовністю елементів, кожний з яких містить ребро та інцидентну йому пару вершин.

**Приклад 3.17.** Розглянемо граф $G(V,E)$, показний на рис. 3.48.

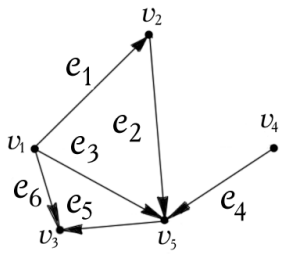



Рис. 3.48. Граф $G(V,E)$

Описати даний граф списком його ребер.

*Розв'язок.*

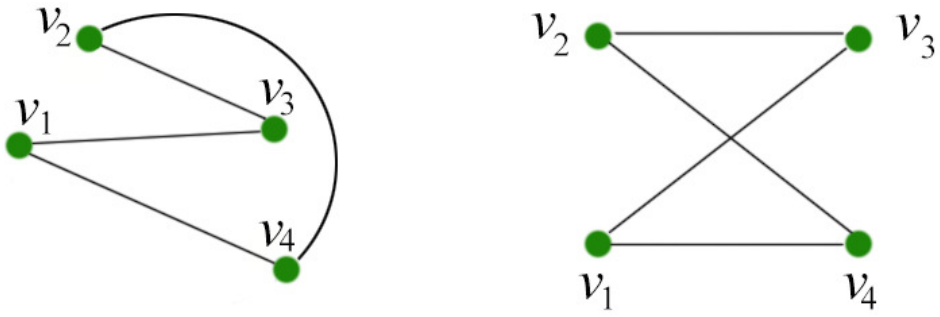
Орграф і список його ребер

$$e_1 \to (v_1, v_2), e_2 \to (v_2, v_3), e_3 \to (v_1, v_5),$$
$$e_4 \to (v_4, v_5), e_5 \to (v_5, v_3), e_6 \to (v_1, v_3)$$

### 3.2.3. Ізоморфізм графів

З попереднього ми дізналися про різні способи задавання графів. Отже, довільний граф можна задати графічно (рисунком), матрицею інцидентності, матрицею суміжності і списком ребер.

При графічному задаванні графа вигляд рисунка залежить від форми ліній і взаємного розташування вершин. Тому не завжди легко зрозуміти, чи однакові графи, зображені на різних рисунках. Наприклад, на рисунках 3.49 а та 3.49 б зображено один і той же граф.




а б

Рис. 3.49. Різні графічні представлення одного і того ж графа

При задаванні графа матрицями або списком ребер їх вигляд **залежить від нумерації** вершин і ребер графа. Будемо вважати, що граф повністю заданий, якщо нумерація його вершин зафіксована.

**Просте визначення.** Графи, що відрізняються тільки нумерацією вершин, називають *ізоморфними графами.*

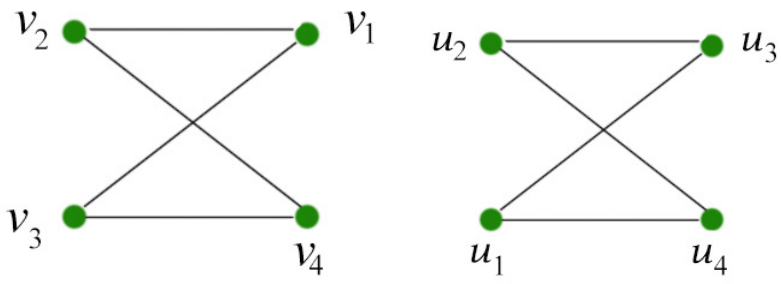
*Ізоморфізм графів*

Нехай $G=(V_1,E_1)$ і $H=(V_2,E_2)$ – графи. Між вершинами даних графів існує взаємно однозначна відповідність (бієкція) $R:V_1\to V_2$. Потужності множин $V_1$ та $V_2$ співпадають: $|V_1|=|V_2|$.

**Визначення.** Відображення $R$ називають *ізоморфізмом* графів $G$ і $H$, якщо для будь-яких суміжних вершин $v_i,v_j\in G$ їх образи $(u_k,u_m)\in H$ також суміжні.

**Точне визначення.** Якщо відображення $R$ існує на графах $G$ і $H$, які відповідають попереднім умовам, то такі графи називають *ізоморфними* графами.

**Приклад 3.18**. На рис. 3.50 показані графи $G(V,E)$(а) та $H(U,F)$ (б).




а б

Рис. 3.50. Графи $G(V,E)$(а) та $H(U,F)$ (б)

Визначити, чи дані графи є ізоморфними.

*Розв'язок.*

Порівняємо множини відображень для вершини $v_1$ у графі $G(V,E)$ та $u_1$ у графі $H(U,F)$.

$$\begin{cases}\Gamma(v_1)=\{(v_1,v_2),(v_1,v_3)\}\\ \Gamma(u_3)=\{(u_3,u_2),(u_3,u_1)\}\end{cases}$$

З даних відображень можна зробити висновок, що $R: v_1 \to u_3$.

Отже, відношення, яке задане перерахуванням, має вигляд:

$$R=\{(v_1,u_3),(v_2,u_2),(v_3,u_1),(v_4,u_4)\}$$

**Приклад 3.19**. На рис. 3.51 показані графи $G(V_1,E_1)$ та $H(V_2,E_2)$. Визначити, чи є дані графи ізоморфними.

*Розв'язок.*

1. Визначаємо потужність вершин графів: $|V_1|=8, |V_2|=8, \ |V_1|=|V_2|$
2. Знаходимо однозначне відображення вершин.

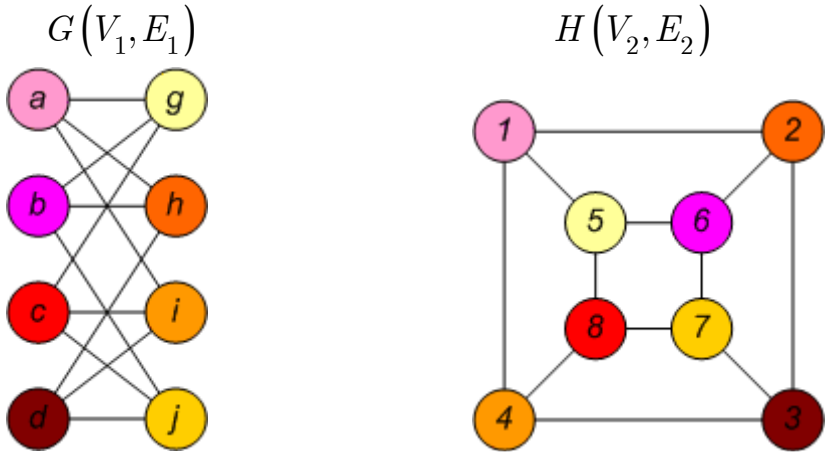



Рис. 3.51. Графи $G(V_1,E_1)$ та $H(V_2,E_2)$.

$f(a)=1$
$f(b)=6$
$f(c)=8$
$f(d)=3$
$f(g)=5$
$f(h)=2$
$f(i)=4$
$f(j)=7$

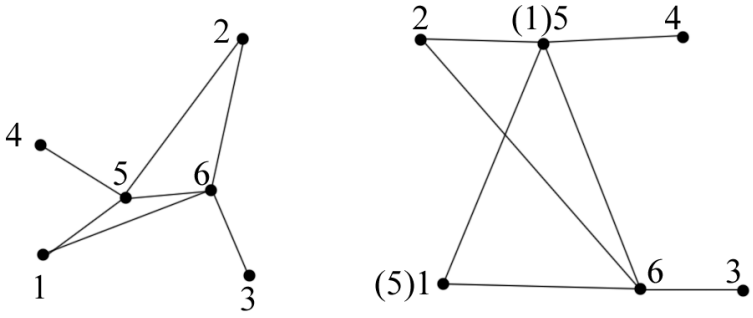
$(c,g)\to(8,5)$
$(c,i)\to(8,4)$
$(c,j)\to(8,7)$
$(d,h)\to(3,2)$
$(d,i)\to(3,4)$
$(d,j)\to(3,7)$

$(a,g)\to(1,5)$
$(a,h)\to(1,2)$
$(a,i)\to(1,4)$
$(b,g)\to(6,5)$
$(b,h)\to(6,2)$
$(b,j)\to(6,7)$

3. Одержуємо відношення:

$$R=\{(a,1),(b,6),(c,8),(d,3),(h,2),(g,5),(i,4),(j,7)\}$$

**Приклад 3.20**. На рис. 3.52 показані ізоморфні графи $G$ й $H$.




Граф G Граф H

Рис. 3.52. Ізоморфні графи $G$ і $H$

Побудувати матриці суміжності даних графів.

*Розв'язок.*

$G_{\text{сум}}$ **–** матриця суміжності графа $G$ й $H_{\text{сум}}$ – матриця суміжності графа $H$.

$$G_{\text{сум}} = \begin{Vmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{Vmatrix} \quad H_{\text{сум}} = \begin{Vmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{Vmatrix}$$

*Властивість матриць суміжності ізоморфних графів*

Якщо графи $G$ й $H$ **ізоморфні**, то з матриці суміжності $G_{\text{сум}}$ графа можна одержати матрицю суміжності $H_{\text{сум}}$ шляхом послідовної перестановки рядків і стовпців.

Для цього потрібно виконати максимально $n!$ перестановок, де $n$ – число вершин графа.

*Ізоморфізм орграфів*

**Визначення.** Орграфи, що відрізняються нумерацією вершин та мають з точністю до нумерації однаково направлені ребра, називають *ізоморфними орграфами.*

### 3.2.4. Алгоритм розпізнавання ізоморфізму графів

1. Перевіряємо умову $|V| = |W| = n$. Якщо кількість вершин графа $|V|$ не дорівнює кількості вершин графа $|W|$, то графи однозначно неізоморфні.

2. Сортуємо елементи множин $V=\{v_1,v_2,v_3,\ldots,v_n\}$ і $W=\{w_1,w_2,w_3,\ldots,w_n\}$ за критерієм величини степеня для кожної вершини.
3. В отриманих упорядкованих послідовностях вершин шукаємо вершини з однаковими значеннями критерія упорядкування, тобто шукані вершини повинні мати однакові степені.
4. Якщо такі вершини знайдені, то з'єднуємо їх ребром з метою побудови графа взаємно однозначної відповідності. Якщо такої відповідності немає, тобто одна зі знайдених вершин уже включена в граф відповідності, то початкові графи $G$ й $H$ неізоморфні.
5. Якщо граф взаємно однозначної відповідності побудований, то розглянуті графи ізоморфні, а його ребра вказують на перестановки, які потрібно зробити для ізоморфного перетворення графа $G$ в граф $H$.
На рис. 3.53 відповідні вершини ізоморфних графів з'єднані лініями.

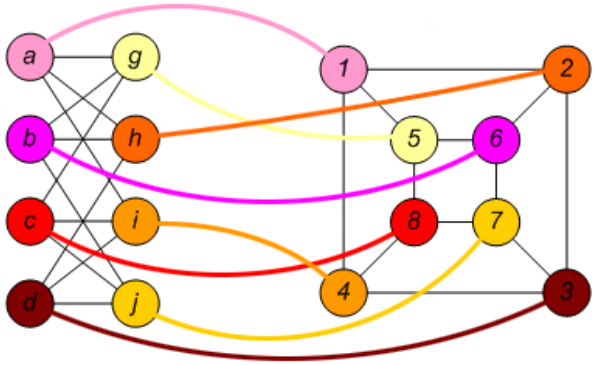



Рис. 3.53. Ізоморфні графи

### 3.2.5. Теоретико-множинні операції над графами

*Операція об'єднання графів*

Граф $F$ називають об'єднанням графів $G=(V,E)$ і $H=(V_1,E_1)$, якщо

$$F=G\cup H=(V\cup V_1,E\cup E_1).$$

Якщо $V\cap V_1=\varnothing$ та $E\cap E_1=\varnothing$, то об'єднання графів називають диз'юнктивним. З властивостей операції об'єднання випливає, що $G\cup H=H\cup G$.

Граф є зв'язним, якщо його не можна представити у вигляді диз'юнктивного об'єднання двох графів і незв'язним – у протилежному випадку. На рис. 3.54 показано об'єднання двох графів.

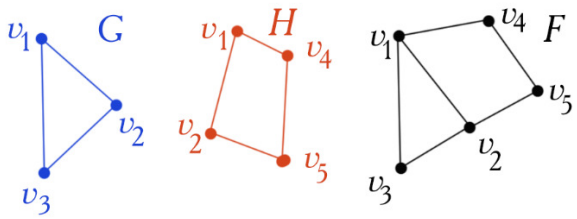



Рис. 3.54. Графи $G$ і $H$ та їх об'єднання $F$

*Операція перетину графів*

Граф $F$ називають перетином графів $G=(V,E)$ і $H=(V_1,E_1)$, якщо $F=G\cap H=(V\cap V_1, E\cap E_1)$.

*Операція доповнення*

Доповненням графа $G=(V,E)$ називають граф $\overline{G}=(V,\overline{E})$, множиною вершин якого є множина $V$, а множина ребер формується відповідно до правила $\overline{E}=\{e\in V\times V \mid e\notin E\}$.

*Декартовий добуток графів*

**Визначення.** *Декартовим добутком* графів $G_1=(V_1,E_1)$ і $G_2=(W_2,E_2)$ називають граф $G(\Omega,E)$, множиною вершин якого є декартовий добуток $\Omega=V_1\times V_2$, де $V_1=\{v_1,v_2,...,v_n\}$, $W_2=\{w_1,w_2,...,w_m\}$.

$$\Omega=\{(v_1,w_2),(v_1,(w_2)),...,(v_n,w_m)\}.$$

Причому вершина $(v_i,w_j)$ суміжна з вершиною $(v_a,w_b)$ при $1\le i,a\le n,\ 1\le j,b\le m$ тоді й тільки тоді, коли в графі $G_1$ суміжні відповідні вершини $v_i$ і $v_a$, а в графі $G_2$ суміжні вершини $w_j$ і $w_b$.

**Приклад 3.21.** Розглянемо два графа:

$G_1=(V_1,E_1)$, де $V_1=\{v_1,v_2\}$ й $E_1=\{(v_1,v_2)\}$.

$G_2=(W_2,E_2)$, де $W_2=\{w_1,w_2,w_3\}$ й $E_2=\{(w_1,w_2),(w_2,w_3)\}$.

Побудувати декартовий добуток даних графів $G=G_1\times G_2$. $G=(\Omega,E)$

*Розв'язок.*

Графічне представлення графів $G_1$, $G_2$ та їхнього декартового добутку $G$ показане на рис. 3.55.

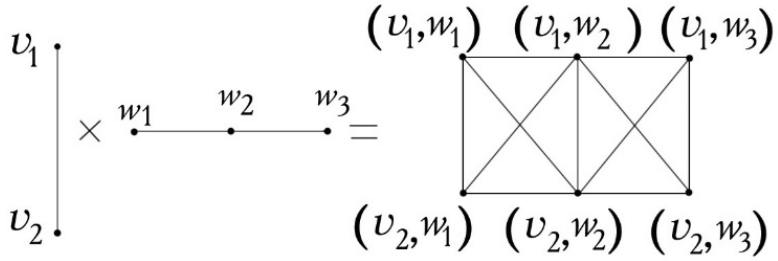



Рис. 3.55. Графи $G_1$, $G_2$ і $G$

$$\Omega = \{(v_1, w_1), (v_1, w_2)(v_1, w_3), (v_2, w_1), (v_2, w_2), (v_2, w_3)\}$$

$$E = \{((v_1, w_1), (v_2, w_1)), ((v_1, w_1), (v_1, w_2)), \ldots\}$$

$|V_1| = 2; |W_2| = 3 \Rightarrow$ матриця вершин має 2 рядки і 3 стовпці

**Приклад 3.22.** Знайти декартовий добуток орграфів, які показано на рис. 3.56.

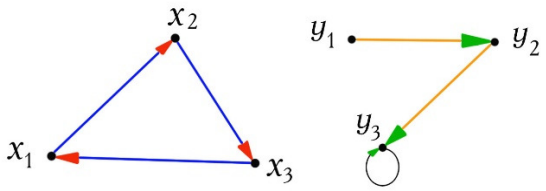



Рис. 3.56. Орграфи $G_1(X, E_1)$ і $G_2(Y, E_2)$

*Розв'язок.*

$G_1(X, E_1)$: $X = \{x_1, x_2, x_3\}$, $E_1 = \{(x_1, x_2), (x_2, x_3), (x_3, x_1)\}$

$G_2(Y, E_2)$: $Y = \{y_1, y_2, y_3\}$, $E_1 = \{(y_1, y_2), (y_2, y_3), (y_3, y_3)\}$

1. Побудуємо множину вершин декартового добутку графів:

$$\{(x_1, y_1), (x_1, y_2), (x_1, y_3), (x_2, y_1), (x_2, y_2), (x_2, y_3), (x_3, y_1), (x_3, y_2), (x_3, y_3)\}$$

2. Створюємо графічне представлення графа (рис. 3.57)

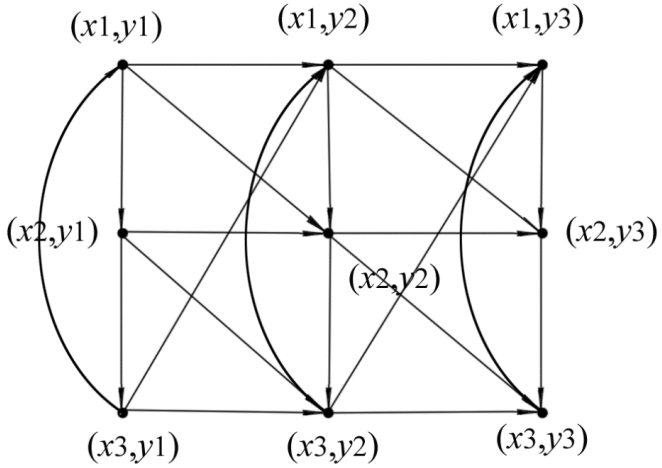



Рис. 3.57. Граф добутку графів $G_1(X, E_1)$ і $G_2(Y, E_2)$

### 3.2.6. Паросполучення ребер графа

**Визначення.** *Паросполучення* – це довільна підмножина попарно несуміжних ребер графа $G(V, E)$.

**Визначення.** *Досконале паросполучення* – це набір ребер, у якому кожна вершина графа інцидентна хоча б одному ребру з паросполучення.

**Приклад 3.23.** Дано граф $G\left(V,E\right)$, який задано:

множиною вершин $V=\left\{v_1,v_2,v_3,v_4,v_5,v_6\right\}$ та множиною ребер

$E=\left\{\left(v_1,v_4\right),\left(v_1,v_5\right),\left(v_1,v_6\right),\left(v_2,v_4\right),\left(v_2,v_5\right),\left(v_2,v_6\right),\left(v_3,v_4\right),\left(v_3,v_5\right),\left(v_3,v_6\right)\right\}$

Граф показаний на рис. 3.58:

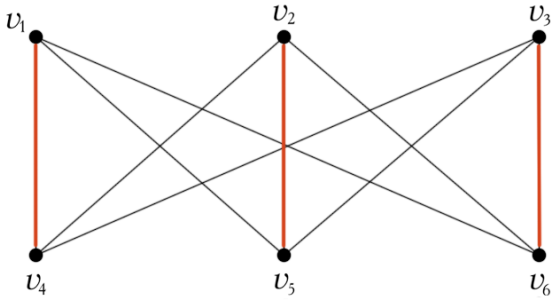



Рис. 3.58. Граф $G\left(V,E\right)$

Знайти досконале паросполучення.

*Розв'язок.*

Граф має досконале паросполучення $\left\{\left(v_1,v_4\right),\left(v_2,v_5\right),\left(v_3,v_6\right)\right\}$.

**Контрольні запитання**

1. Визначте мінімальну кількість та послідовність операцій над графом $G_1$ (рис. 3.59) для перетворення його у граф $G_2$.

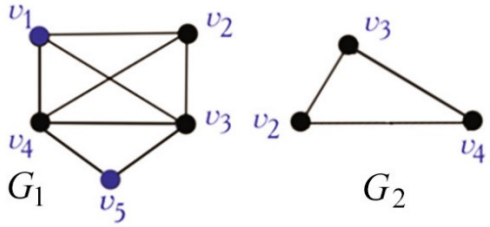



Рис. 3.59. Графи до задачі 1

2. Для графа, зображеного на рис. 3.60, побудувати матриці суміжності та інцидентності. Спираючись на властивості даних матриць, визначити основні характеристики графа.

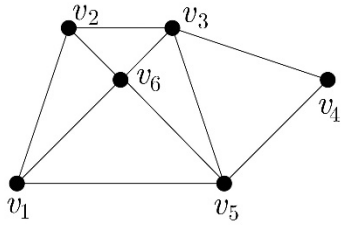



Рис. 3.60. Граф до задачі 2

3. Довести ізоморфізм графів, показаних на рис. 3.61.

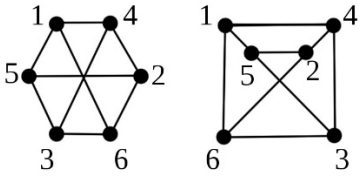



Рис. 3.61. Графи до задачі 3

4. Побудувати доповнення незв'язного графа, показаного на рис. 3.62.

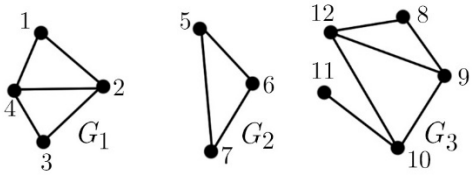



Рис. 3.62. Графи до задачі 4

5. Знайти декартовий добуток оргарафів $G(V,E)$ і $H(W,F)$, які задані такими параметрами:

$V=\{v_1,v_2,v_3\}$, $E=\{(v_1,v_2),(v_2,v_1),(v_2,v_3)\}$,

$W=\{w_1,w_2,w_3\}$ $F=\{(f_1,f_2),(f_2,f_3),(f_3,f_2)\}$.

6. Знайти досконале паросполучення у графі $G(V,E)$, якщо

$V=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8\}$,

$E=\{(v_1,v_2),(v_1,v_3),(v_1,v_4),(v_2,v_5),(v_3,v_5),(v_3,v_6),(v_4,v_7),(v_7,v_6),(v_5,v_8),(v_6,v_8),(v_7,v_8)\}$,

## 3.3. Відношення та відображення на графах

### 3.3.1. Графи й бінарні відношення

Відношенню $R$, заданому на множині $V$, взаємно однозначно відповідає орієнтований граф $G(R)$ без кратних ребер з множиною вершин $V$, у якому ребро $(v_i,v_j)$ існує тільки тоді, коли виконано $v_iRv_j$. Представимо на графах деякі бінарні відношення.

*Рефлексивність.*

Відношення $R$ на множині $V$ ***рефлексивне,*** якщо для кожного елемента $v\in V$ справедливе $(v,v)\in R$. На графі це зображується петлею, а матриця суміжності графа з рефлексивними відношеннями містить одиниці на головній діагоналі.

Іншими словами, якщо відношення $R$ рефлексивне, то граф $G(R)$ без кратних ребер має петлі у всіх вершинах.

**Приклад 3.24.** Зобразити графічно та задати матрицею суміжності граф $G\left(R\right)$, заданий рефлексивним відношенням $R$.

*Розв'язок.*

На рис. 3.63 показаний приклад графа рефлексивного відношення.

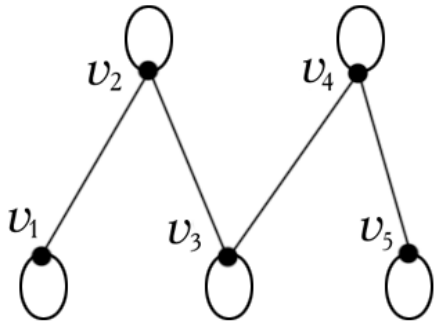



Рис. 3.63. Граф рефлексивного відношення

Головна діагональ матриці суміжності $G\left(R\right)$ складається з одиниць.

$$\mathbf{C} = \begin{Vmatrix} \mathbf{1} & 1 & 0 & 0 & 0 \\ 1 & \mathbf{1} & 1 & 0 & 0 \\ 0 & 1 & \mathbf{1} & 1 & 0 \\ 0 & 0 & 1 & \mathbf{1} & 1 \\ 0 & 0 & 0 & 1 & \mathbf{1} \end{Vmatrix}$$

*Антирефлексивність*

Якщо відношення $R$ на множині $V$ *антирефлексивне,* то для всіх елементів $v$ множини $V$ справедливе $(v,v)\notin R$.

Якщо $R$ антирефлексивне, то граф $G\left(R\right)$ без кратних ребер не має петель.

**Приклад 3.25**. Зобразити графічно та задати матрицею суміжності граф $G\left(R\right)$, заданий антирефлексивним відношенням $R$.

*Розв'язок.*

На рис. 3.64 показаний приклад графа антирефлексивного відношення.

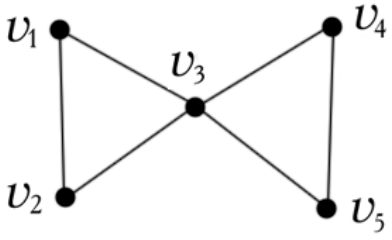



Рис. 3.64. Граф антирефлексивного відношення

Головна діагональ матриці суміжності $G(R)$ складається з нулів.

$$\mathbf{C} = \begin{Vmatrix} \mathbf{0} & 1 & 1 & 0 & 0 \\ 1 & \mathbf{0} & 1 & 0 & 0 \\ 1 & 1 & \mathbf{0} & 1 & 1 \\ 0 & 0 & 1 & \mathbf{0} & 1 \\ 0 & 0 & 1 & 1 & \mathbf{0} \end{Vmatrix}$$

*Симетричність*

Відношення $R$ на $V$ називають ***симетричним***, якщо з $(v_i, v_j) \in R$ випливає $(v_j, v_i) \in R$ при $v_i \neq v_j$. Матриця суміжності симетричного відношення симетрична щодо головної діагоналі.

**Приклад 3.26**. Зобразити графічно та задати матрицею суміжності граф $G(R)$, заданий симетричним відношенням $R$.

*Розв'язок.*

На рис. 3.65 показаний приклад графа симетричного відношення.

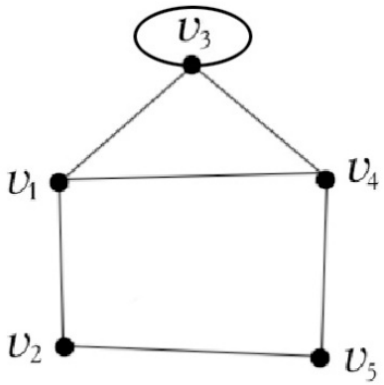



Рис. 3.65. Граф симетричного відношення

Матриця суміжності $C$ симетрична щодо головної діагоналі.

$$C = \begin{Vmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{Vmatrix}$$

Симетричність підтверджується наявністю симетричних пар.

$(v_1, v_2) \in R \to (v_2, v_1) \in R$, $(v_1, v_3) \in R \to (v_3, v_1) \in R$,

$(v_1, v_4) \in R \to (v_4, v_1) \in R$, $(v_2, v_5) \in R \to (v_5, v_2) \in R$,

$(v_3, v_4) \in R \to (v_4, v_3) \in R$, $(v_4, v_5) \in R \to (v_5, v_4) \in R$.

*Антисиметричність*

Відношення $R$ на $V$ називають ***антисиметричним***, якщо з $(v_i, v_j) \in R$ випливає $(v_j, v_i) \notin R$ при $v_i \neq v_j$. Матриця суміжності антисиметричного відношення несиметрична щодо головної діагоналі. Антисиметричне відношення завжди представлене орграфом з дугами без повторень.

**Приклад 3.27**. Зобразити графічно та задати матрицею суміжності граф $G(R)$, заданий антисиметричним відношенням $R$.

*Розв'язок.*

На рис. 3.66 показаний приклад графа антисиметричного відношення.

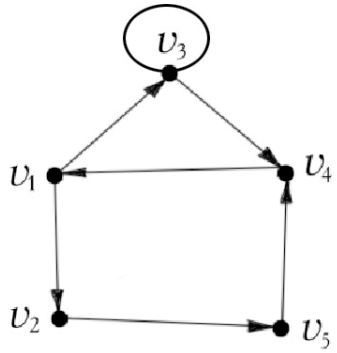



Рис. 3.66. Граф антисиметричного відношення

Матриця суміжності $C$ антисиметрична щодо головної діагоналі.

$$C = \begin{Vmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{Vmatrix}$$

Антисиметричність підтверджується відсутністю симетричних пар

$(v_1, v_2) \in R \to (v_2, v_1) \notin R$, $(v_1, v_3) \in R \to (v_3, v_1) \notin R$,

$(v_4, v_1) \in R \to (v_1, v_4) \notin R$, $(v_2, v_5) \in R \to (v_5, v_2) \notin R$,

$(v_3, v_4) \in R \to (v_4, v_3) \notin R$, $(v_5, v_4) \in R \to (v_4, v_5) \notin R$.

*Транзитивність*

Відношення $R$ на множині $V$ називають ***транзитивним***, якщо з $(v_i, v_j) \in R$, $(v_j, v_k) \in R$ випливає $(v_i, v_k) \in R$ при $v_i, v_j, v_k \in V$

і $v_i \neq v_j, v_j \neq v_k, v_i \neq v_k$. У графі, що задає транзитивне відношення для всякої пари дуг, таких, що кінець першої дуги збігається з початком другої, існує транзитивно замикаюча дуга, що має спільний початок з першою і спільний кінець з другою.

**Приклад 3.28.** Зобразити графічно та задати матрицею суміжності граф $G\left(R\right)$, заданий транзитивним відношенням $R$.

*Розв'язок.*

На рис. 3.67 показаний приклад графа транзитивного відношення.

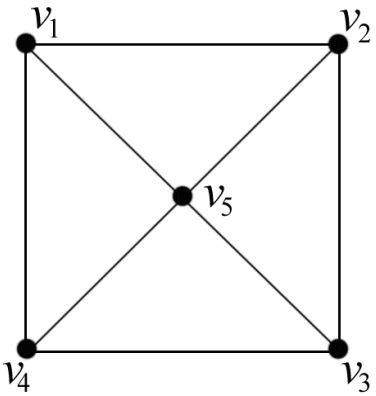



Рис. 3.67. Граф транзитивного відношення

$$C = \begin{Vmatrix} & v_1 & v_2 & v_3 & v_4 & v_5 \\ v_1 & 0 & 1 & 0 & 1 & 1 \\ v_2 & 1 & 0 & 1 & 0 & 1 \\ v_3 & 0 & 1 & 0 & 1 & 1 \\ v_4 & 1 & 0 & 1 & 0 & 1 \\ v_5 & 1 & 1 & 1 & 1 & 0 \end{Vmatrix}$$

Властивість транзитивності виконується на всіх ребрах.

$$\left(v_1, v_5\right) \in R, \left(v_5, v_2\right) \in R \to \left(v_1, v_2\right) \in R;$$

$$\left(v_1, v_5\right) \in R, \left(v_5, v_4\right) \in R \to \left(v_1, v_4\right) \in R;$$

$$\left(v_3, v_5\right) \in R, \left(v_5, v_4\right) \in R \to \left(v_3, v_4\right) \in R;$$

$$\left(v_2, v_5\right) \in R, \left(v_5, v_3\right) \in R \to \left(v_2, v_3\right) \in R;$$

$$\dots$$

$$\left(v_1, v_2\right) \in R, \left(v_2, v_5\right) \in R \to \left(v_1, v_5\right) \in R;$$

$$\left(v_5, v_1\right) \in R, \left(v_1, v_2\right) \in R \to \left(v_5, v_2\right) \in R.$$

Отже, відношення $R$ на множині вершин $V=\{v_1,v_2,...,v_5\}$ транзитивне, оскільки для довільного ребра в графі виконується умова транзитивності.

*Антитранзитивність*

Відношення $R$ на множині $V$ називають ***антитранзитивним***, якщо з $(v_i,v_j)\in R$, $(v_j,v_k)\in R$ випливає $(v_i,v_k)\notin R$ при $v_i,v_j,v_k\in V$ і $v_i\neq v_j, v_j\neq v_k, v_i\neq v_k$. У графі, що задає антитранзитивне відношення для будь-якої пари дуг, таких, що кінець першої дуги збігається з початком другої, не існує транзитивно замикаючої дуги, яка має спільний початок з першою і спільний кінець з другою.

**Приклад 3.29.** Зобразити графічно та задати матрицею суміжності граф $G(R)$, заданий антитранзитивним відношенням $R$.

*Розв'язок.*

На рис. 3.68 показаний приклад графа антитранзитивного відношення.

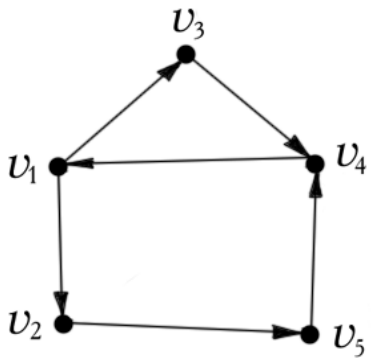



Рис. 3.68. Граф антитранзитивного відношення

$$C=\begin{Vmatrix}0&1&1&0&0\\0&0&0&0&1\\0&0&0&1&0\\1&0&0&0&0\\0&0&0&1&0\end{Vmatrix}$$

Властивість антитранзитивності виконується на всіх ребрах.

$$(v_1,v_3)\in R,(v_3,v_4)\in R\rightarrow(v_1,v_4)\notin R;$$
$$(v_4,v_1)\in R,(v_1,v_3)\in R\rightarrow(v_4,v_3)\notin R;$$
$$(v_3,v_4)\in R,(v_4,v_1)\in R\rightarrow(v_3,v_1)\notin R;$$
$$(v_4,v_1)\in R,(v_1,v_2)\in R\rightarrow(v_4,v_2)\notin R;$$

$$(v_2, v_5) \in R, (v_5, v_4) \in R \rightarrow (v_2, v_4) \notin R;$$
$$(v_5, v_4) \in R, (v_4, v_1) \in R \rightarrow (v_5, v_1) \notin R;$$
$$(v_1, v_2) \in R, (v_2, v_5) \in R \rightarrow (v_1, v_5) \notin R$$

Отже, відношення $R$ на множині вершин $V = \{v_1, v_2, ..., v_5\}$ антитранзитивне, оскільки для довільних пар ребер виконується умова антитранзитивності.

### 3.3.2. Зв'язок між операціями над графами й операціями над відношеннями

1. Нехай $\overline{R}$ – доповнення відношення $R \subset V \times V$:

$$\overline{R} = U \setminus R,$$

де $U$ – універсальне (повне) відношення $U = V \times V$, тобто відношення, яке існує між будь-якою парою елементів з $V$.

2. Граф $G(\overline{R})$ є доповненням графа $G(R)$ до повного графа $K$ з множиною вершин $V$ і множиною ребер $E(K) = V \times V$:

$$G(\overline{R}): E(\overline{R}) = E(K) \setminus E(R); \; V(\overline{R}) = V(R)$$

**Приклад 3.30.** Нехай дано множину вершин: $V = \{v_1, v_2, v_3, v_4\}$ та множину

$$R = \{(v_1, v_1), (v_1, v_2), (v_2, v_2), (v_2, v_3), (v_3, v_3), (v_3, v_4), (v_4, v_1), (v_4, v_4)\}$$

Побудувати множини ребер графів, які відповідають універсальному відношенню $U$ доповненню $\overline{R}$. Зобразити дані графи графічно.

*Розв'язок.*

$$U = \{(v_1, v_1), (v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_1), (v_2, v_2), (v_2, v_3), (v_2, v_4),$$
$$(v_3, v_1), (v_3, v_2), (v_3, v_3), (v_3, v_4), (v_4, v_1), (v_4, v_2), (v_4, v_3), (v_4, v_4)\}$$

$$\overline{R} = \{(v_1, v_3), (v_1, v_4), (v_2, v_1), (v_2, v_4), (v_3, v_1), (v_3, v_2), (v_4, v_2), (v_4, v_3)\}$$

На рис. 3.69 показані графи на відношеннях $R, U$ та $\overline{R}$

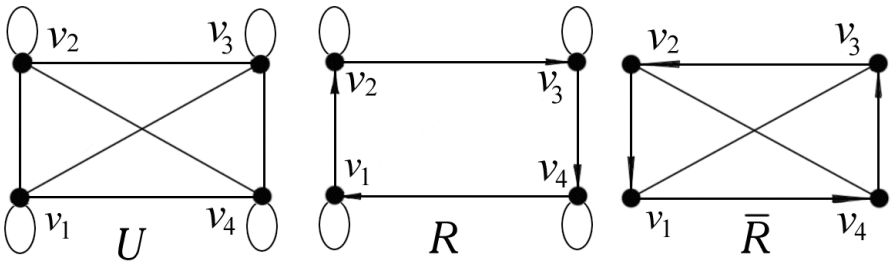



Рис. 3.69. Графи на відношеннях $R, \overline{R}, U$

3. Граф зворотного відношення $G(R^{-1})$ відрізняється від графа $G(R)$ тим, що напрямки всіх ребер замінені на зворотні.

**Приклад 3.31.** Нехай дано множину вершин: $V = \{v_1, v_2, v_3, v_4\}$ та множину

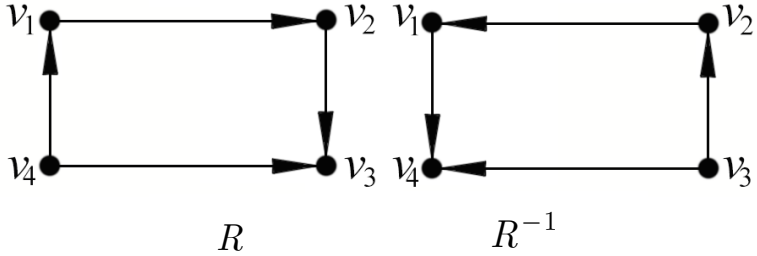
$R = \{(v_1, v_2), (v_2, v_3), (v_4, v_1), (v_4, v_3)\}$

Побудувати множину ребер графа, який відповідає зворотному відношенню $R^{-1}$. Представити графічно графи $G(R)$ та $G(R^{-1})$.

*Розв'язок.* Зворотне відношення $R^{-1}$ має вигляд:

$R^{-1} = \{(v_2, v_1), (v_3, v_2), (v_1, v_4), (v_3, v_4)\}$

Графи відношень $R$ та $R^{-1}$ показані на рис. 3.70.




$R$ $R^{-1}$

Рис. 3.70. Графи $G(R)$ та $G(R^{-1})$ відношень $R$ та $R^{-1}$

4. Граф об'єднання двох відношень, заданих на $V$, $G(R_1 \cup R_2)$, є графом об'єднання двох графів $G(R_1)$ і $G(R_2)$:

$$G(R_1 \cup R_2) = G(R_1) \cup G(R_2).$$

**Приклад 3.32.** Нехай дано множину вершин: $V = \{v_1, v_2, v_3, v_4\}$ та множину, на якій задано два відношення:

$R_1 = \{(v_1, v_2), (v_2, v_3), (v_4, v_1), (v_4, v_3), (v_4, v_2)\}$,

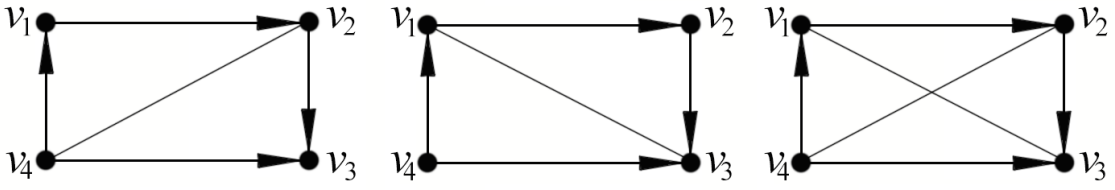
$R_2 = \{(v_1, v_2), (v_2, v_3), (v_4, v_1), (v_4, v_3), (v_1, v_3)\}$.

Побудувати множину ребер графа, задану на об'єднанні відношень $R_1$ та $R_2$. Представити графічно графи $G(R_1 \cup R_2)$, $G(R_1)$ та $G(R_2)$.

*Розв'язок.*

$R_1 \cup R_2 = \{(v_1, v_2), (v_2, v_3), (v_4, v_1), (v_4, v_3), (v_4, v_2), (v_1, v_3)\}$

Графи відношень $G(R_1 \cup R_2)$, $G(R_1)$ та $G(R_2)$ показані на рис. 3.71.




$G(R_1)$ $G(R_2)$ $G(R_1 \cup R_2)$

Рис. 3.71. Графи $G(R_1 \cup R_2)$, $G(R_1)$ та $G(R_2)$

5. Граф перетину відношень $R_1 \cap R_2$ на $V$ $G(R_1 \cap R_2)$ є графом перетину двох графів $G(R_1)$ і $G(R_2)$:

$$G(R_1 \cap R_2) = G(R_1) \cap G(R_2).$$

**Приклад 3.33.** Нехай дано множину вершин: $V = \{v_1, v_2, v_3, v_4\}$ та множину, на якій задано два відношення:

$R_1 = \{(v_1, v_2), (v_2, v_3), (v_4, v_1), (v_4, v_3), (v_4, v_2)\}$,

$R_2 = \{(v_1, v_2), (v_2, v_3), (v_4, v_1), (v_4, v_3), (v_1, v_3)\}$.

Побудувати множину ребер графа, задану на перетині відношень $R_1$ та $R_2$. Представити графічно графи $G(R_1 \cap R_2)$, $G(R_1)$ та $G(R_2)$.

*Розв'язок.*

$R_1 \cap R_2 = \{(v_1, v_2), (v_2, v_3), (v_4, v_1), (v_4, v_3)\}$

Графи відношень $G(R_1 \cap R_2)$, $G(R_1)$ та $G(R_2)$ показані на рис. 3.72.

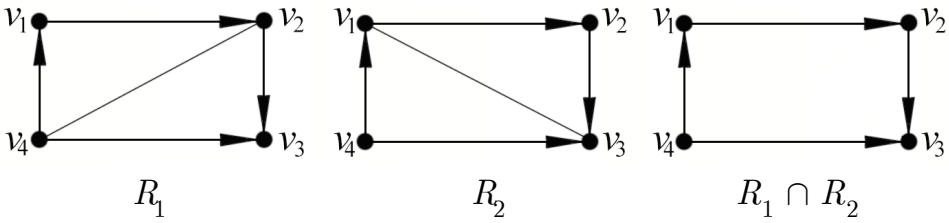



Рис. 3.72. Графи $G(R_1 \cap R_2)$, $G(R_1)$ та $G(R_2)$

### 3.3.3. Багатозначні відображення

**Визначення.** *Пряме відображення першого порядку* вершини $v_i$ – це множина таких вершин $v_j$ графа $G(V, E)$, для яких існує дуга $(v_i, v_j)$, тобто

$\Gamma^+(v_i) = \{v_j \mid (v_i, v_j) \in E, i, j = 1, 2, ..., n\}$, де $n = |V|$ – кількість вершин графа.

**Приклад 3.34.** Розглянемо неорієнтований граф $G(V, E)$,

де $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$,

$E = \{(v_1, v_2), (v_1, v_9), (v_2, v_3), (v_2, v_8), (v_3, v_4), (v_4, v_{11}), (v_5, v_6), (v_5, v_{11}),$

$(v_6, v_7), (v_7, v_{10}), (v_8, v_{10}), (v_9, v_{10})\}$.

Побудувати граф $G(V, E)$ та множину відображення $\Gamma^+(v_8)$.

*Розв'язок.*

Множина відображення $\Gamma^+(v_8)$ у неорієнтованому графі $G(V, E)$ – це множина вершин, які є суміжними з вершиною $v_8$. Шуканий граф $G(V, E)$ показаний на рис. 3.73.

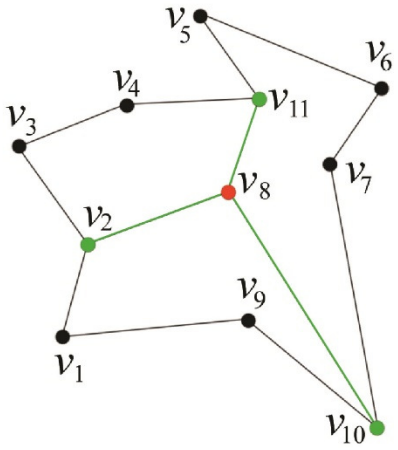



Рис. 3.73. Граф $G(V,E)$ з $\Gamma^+(v_8)=\{v_2,v_{11},v_{10}\}$

Множина відображення $\Gamma^+(v_8)$: $\Gamma^+(v_8)=\{v_2,v_{11},v_{10}\}$

**Визначення.** *Пряме відображення другого порядку* вершини $v_i$ – це пряме відображення від прямого відображення першого порядку

$$\Gamma^{+2}(v_i)=\Gamma^{+}\left(\Gamma^{+1}(v_i)\right).$$

**Приклад 3.35.** Розглянемо неорієнтований граф $G(V,E)$,
де $V=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{10},v_{11}\}$,
$E=\{(v_1,v_2),(v_1,v_9),(v_2,v_3),(v_2,v_8),(v_3,v_4),(v_4,v_{11}),(v_5,v_6),(v_5,v_{11}),$
$(v_6,v_7),(v_7,v_{10}),(v_8,v_{10}),(v_9,v_{10})\}$.
Побудувати граф $G(V,E)$ та множину прямого відображення від прямого відображення $\Gamma^+(v_8)$, тобто $\Gamma^{+2}(v_8)$.
*Розв'язок.*

Множина відображення другого порядку $\Gamma^{+2}(v_8)$ у неорієнтованому графі $G(V,E)$ – це множина вершин, які є суміжними з множиною суміжних вершин до вершини $v_8$. Шуканий граф $G(V,E)$ показаний на рис. 3.74.

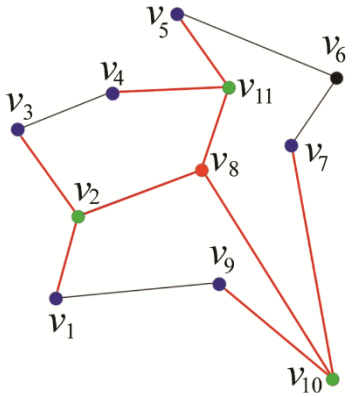



Рис. 3.74. Граф $G(V,E)$ з $\Gamma^{+2}(v_8)=\{v_1,v_3,v_4,v_5,v_7,v_9\}$

Множини прямого відображення першого і другого порядку для вершини $v_8$:

$\Gamma^{+}(v_8)=\{v_2, v_{11}, v_{10}\}$

$\Gamma^{+2}(v_8)=\{v_1, v_3, v_4, v_5, v_7, v_9\}$

**Визначення.** *Пряме відображення третього порядку* вершини $v_i$ – це пряме відображення від прямого відображення другого порядку.

$$\Gamma^{+3}(v_i)=\Gamma^{+}\left(\Gamma^{+2}(v_i)\right)=\Gamma^{+}\left(\Gamma^{+}\left(\Gamma^{+1}(v_i)\right)\right),$$

**Визначення.** *Пряме відображення четвертого порядку* вершини $v_i$ – це пряме відображення від прямого відображення третього порядку.

$$\Gamma^{+4}(v_i)=\Gamma^{+}\left(\Gamma^{+3}(v_i)\right)=\Gamma^{+}\left(\Gamma^{+}\left(\Gamma^{+}\left(\Gamma^{+1}(v_i)\right)\right)\right),$$

**Визначення.** *Пряме відображення p-го порядку* вершини $v_i$ – це пряме відображення від прямого відображення $p$–1 порядку.

…….

$$\Gamma^{+p}(v_i)=\Gamma^{+}\left(\Gamma^{+(p-1)}(v_i)\right)$$

**Приклад 3.36.** Знайдемо прямі багатозначні відображення для графа $G$, показаного на рис. 3.75:

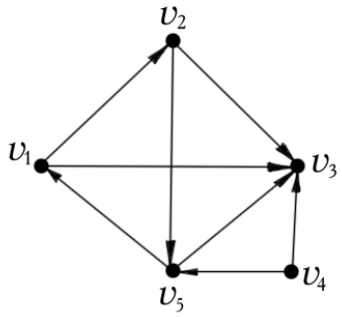



Рис. 3.75. Граф $G$ для визначення прямих відображень вершини $v_1$

*Розв'язок.*

$\Gamma^{+1}(v_1)=\{v_2, v_3\}$,

$\Gamma^{+2}(v_1)=\Gamma^{+}\left(\Gamma^{+}(v_1)\right)=\Gamma^{+}(v_2, v_3)=\{v_3, v_5\}$,

$\Gamma^{+3}(v_1)=\Gamma^{+}\left(\Gamma^{+2}(v_1)\right)=\Gamma^{+}(v_3, v_5)=\{v_3, v_1\}$,

$\Gamma^{+4}(v_1)=\Gamma^{+}\left(\Gamma^{+3}(v_1)\right)=\Gamma^{+}(v_3, v_1)=\{v_2, v_3\}$.

Далі легко помітити, що

$\Gamma^{+1}(v_1) = \Gamma^{+4}(v_1) = \Gamma^{+7}(v_1)....$

$\Gamma^{+2}(v_1) = \Gamma^{+5}(v_1) = \Gamma^{+8}(v_1)....$

$\Gamma^{+3}(v_1) = \Gamma^{+6}(v_1) = \Gamma^{+9}(v_1)....$

Аналогічно знаходимо відображення для інших вершин графа.

**Визначення.** *Зворотне відображення першого порядку* вершини $v_i$ – це множина таких вершин $v_j$ графа $G(V,E)$, для яких існує дуга $(v_j, v_i)$, тобто

$$\Gamma^{-}(v_i) = \left\{ v_j \middle| (v_j, v_i) \in E, i, j = 1,2,...,n \right\},$$

де $n = |V|$ – кількість вершин графа

**Визначення.** *Зворотне відображення другого й наступних порядків* вершини $v_i$ – це зворотне відображення від зворотного відображення попереднього порядку

$\Gamma^{-2}(v_i) = \Gamma^{-}\left(\Gamma^{-1}(v_i)\right).$

$\Gamma^{-3}(v_i) = \Gamma^{-}\left(\Gamma^{-2}(v_i)\right) = \Gamma^{-}\left(\Gamma^{-}\left(\Gamma^{-1}(v_i)\right)\right)$

……

$\Gamma^{-p}(v_i) = \Gamma^{-}\left(\Gamma^{-(p-1)}(v_i)\right)$

**Приклад 3.37.** Знайдемо зворотні багатозначні відображення для графа, показаного на рис. 3.76:

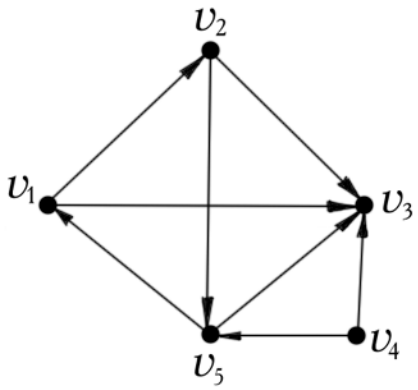



Рис. 3.76. Граф $G$ для визначення зворотних відображень вершини $v_1$

*Розв'язок.*

$\Gamma^{-}(v_1) = \{v_5\},$

$\Gamma^{-2}(v_1) = \Gamma^{-}\left(\Gamma^{-1}(v_1)\right) = \Gamma^{-}(v_5) = \{v_2, v_4\},$

$$\Gamma^{-3}\left(v_1\right)=\Gamma^{-}\left(\Gamma^{-2}\left(v_1\right)\right)=\Gamma^{-}\left(v_2,v_4\right)=\left\{v_1\right\},$$
$$\Gamma^{-4}\left(v_1\right)=\Gamma^{-}\left(\Gamma^{-3}\left(v_1\right)\right)=\Gamma^{-}\left(v_1\right)=\left\{v_5\right\}\ \text{і т. д.}$$

### 3.3.4. Відображення множини вершин

Якщо розглянуте раніше відображення застосовується одночасно до всіх вершин графа, то воно може бути отримане з виразу:

$$\Gamma^{+}\left(V\right)=\bigcup_{v\in V}\Gamma^{+}\left(v\right).$$

Якщо $V=\left\{V_1,V_2,...,V_n\right\}$, то вірні співвідношення:

$$\Gamma^{+}\left(V\right)=\Gamma^{+}\left(\bigcup_{i=1}^{n}V_i\right)=\bigcup_{i=1}^{n}\Gamma^{+}\left(V\right)_i$$

**Приклад 3.38.** Розглянемо незв'язний граф $G$, показаний на рис. 3.77. Побудувати множину прямих відображень $\Gamma^{+}\left(V\right)$ даного графа.

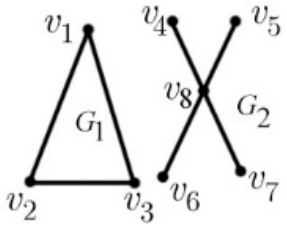



Рис. 3.77. Незв'язний граф $G=\left\{G_1,G_2\right\}$

*Розв'язок.*

$V_1=\left\{v_1,v_2,v_3\right\}$, $V_2=\left\{v_4,v_5,v_6,v_7,v_8\right\}$.

*Розглянемо підграф* $G_1$.

$\Gamma^{+}\left(v_1\right)=\left\{v_2,v_3\right\}$, $\Gamma^{+}\left(v_2\right)=\left\{v_1,v_3\right\}$, $\Gamma^{+}\left(v_3\right)=\left\{v_1,v_2\right\}$

Звідси

$$\Gamma^{+}\left(V_1\right)=\Gamma^{+}\left(v_1\right)\cup\Gamma^{+}\left(v_2\right)\cup\Gamma^{+}\left(v_3\right)=\left\{v_2,v_3\right\}\cup\left\{v_1,v_3\right\}\cup\left\{v_1,v_2\right\}=\left\{v_1,v_2,v_3\right\}$$

*Розглянемо підграф* $G_2$.

$\Gamma^{+}\left(v_4\right)=\left\{v_8\right\}$, $\Gamma^{+}\left(v_5\right)=\left\{v_8\right\}$, $\Gamma^{+}\left(v_6\right)=\left\{v_8\right\}$, $\Gamma^{+}\left(v_7\right)=\left\{v_8\right\}$,

$\Gamma^{+}\left(v_8\right)=\left\{v_4,v_5,v_6,v_7,v_8\right\}$

Звідси

$$\Gamma^{+}\left(V_2\right)=\Gamma^{+}\left(v_4\right)\cup\Gamma^{+}\left(v_5\right)\cup\Gamma^{+}\left(v_6\right)\cup\Gamma^{+}\left(v_7\right)\cup\Gamma^{+}\left(v_8\right)=$$
$$=\left\{v_8\right\}\cup\left\{v_8\right\}\cup\left\{v_8\right\}\cup\left\{v_8\right\}\cup\left\{v_4,v_5,v_6,v_7\right\}=\left\{v_4,v_5,v_6,v_7,v_8\right\}$$

Знаходимо відображення всіх вершин:

$$\Gamma^{+}\left(V\right)=\Gamma^{+}\left(V_1\right)\cup\Gamma^{+}\left(V_2\right)=\left\{v_1,v_2,v_3\right\}\cup\left\{v_4,v_5,v_6,v_7,v_8\right\}=\left\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8\right\}$$

### 3.3.5. Визначення графа і його властивостей з використанням відображень

**Визначення**. *Граф* $G$ – це функція двох змінних $G(V,\Gamma)$, де $V$ – непуста множина вершин; $\Gamma: V \to V$ – відображення множини вершин самої в себе.

Правила графічного представлення графа за його аналітичним описом $G(V,\Gamma)$ такі:

1. Кожну вершину з множини $V = (v_1,...,v_i,...,v_j,...)$ зображають у вигляді окремо розташованої точки.
2. Пари вершин $v_i$ і $v_j$ з'єднують ребром за умови, що $v_j \in \Gamma^+(v_i)$ .

**Визначення.** *Підграфом* графа $G(V,\Gamma)$ називають граф виду $G(A,\Gamma_A)$, де $A \subset V$, а відображення $\Gamma_A$ визначене в такий спосіб: $\Gamma_A^+(v) = \Gamma^+(v) \cap A$.

Правила графічного представлення підграфа $G(A,\Gamma_A)$ за його аналітичним описом такі:

1. Кожну вершину з множини $A = (v_1,...,v_i,...,v_j,...)$, де $A \subset V$ зображають у вигляді окремо розташованої точки.
2. Пари вершин $v_i$ і $v_j$ з'єднують ребром за умови, що $v_j \in \Gamma^+(v_i)$ і $v_i, v_j \in A$

*Компонент зв'язності графа*

**Визначення.** *Компонент зв'язності* – деяка множина вершин графа, у якій між довільними двома вершинами **існує шлях** з однієї в іншу, і не існує жодного шляху з вершини цієї множини у вершину не з цієї множини.

**Визначення.** *Компонент зв'язності* – це граф, породжений деякою множиною вершин $C_v$, де $C_v$ – множина вершин, що включає вершину $v$ і усі ті вершини графа, які з'єднані з нею ланцюгом.

*Теорема про розбиття графа.*

**Теорема 3.5.** Різні компоненти графа $G(V,\Gamma)$ утворюють розбиття множини $V$, тобто

1. $C_v \neq \varnothing$,
2. $v_i, v_j \in V, C_{v_i} \neq C_{v_j} \Rightarrow C_{v_i} \cap C_{v_j} = \varnothing$,
3. $\bigcup C_v = V$.

**Теорема про зв'язний граф.** Граф є зв'язним графом тоді й тільки тоді, коли він складається з одного компонента зв'язності.

Між будь-якою парою вершин зв'язного графа існує як мінімум один шлях.

### 3.3.6. Досяжність і контрдосяжність вершини в графах

**Визначення.** Вершину $w$ графа $D$ (або орграфа) називають **досяжною** з вершини $v$, якщо $w = v$, або існує шлях з $v$ у $w$ (маршрут від $v$ у $w$).


**Визначення.** Вершину $w$ графа $D$ (або орграфа) називають **контрдосяжною** з вершини $v$, якщо існує шлях з $w$ у $v$ (маршрут від $w$ у $v$).


*Матриця досяжності*

**Визначення.** *Матрицею досяжності* називають матрицю $n \times n$ $R = \left(r_{ij}\right), i, j = 1, 2, ..., n$, де $n$ – число вершин графа, а кожний елемент визначається в такий спосіб:

$$r_{ij} = \begin{cases} 1, \text{якщо існує шлях з вершини } v_i \text{ у вершину } v_j, \\ 0, \text{в протилежному випадку.} \end{cases}$$

Усі *діагональні елементи* $r_{ii}$ в матриці $R$ дорівнюють 1, оскільки кожна вершина досяжна з себе самої зі шляхом довжиною 0.

*Множина досяжних вершин* $D\left(v_i\right)$ графа $G$. Множина $D\left(v_i\right)$ вершин, досяжних із заданої вершини $v_i$, складається з таких елементів $v_j$, для яких елемент $r_{ij}$ в матриці досяжності дорівнює 1.

$$R = \left(D\left(v_1\right), ..., D\left(v_i\right), ..., D\left(v_n\right)\right)^T$$

Усі *діагональні елементи* $r_{ii}$ в матриці $R$ дорівнюють 1, оскільки кожна вершина досяжна з себе самої зі шляхом довжиною 0.

*Визначення множини досяжності через відображення*

Будь-яка вершина графа $G$, яка досяжна з $v_i$, повинна бути досяжна з використанням шляху (або шляхів) довжиною 0 або 1, або 2, ..., або $p$. Тоді множину вершин, досяжних з вершини $v_i$, можна представити у вигляді

$$D\left(v_i\right) = \left\{v_i\right\} \cup \Gamma^{+1}\left(v_i\right) \cup \Gamma^{+2}\left(v_i\right) \cup ... \cup \Gamma^{+p}\left(v_i\right).$$

*Побудова матриці досяжності* (рис. 3.78)

Будуємо матрицю по рядках.

1. Знаходимо досяжні множини $D\left(v_i\right)$ для всіх вершин $v_i \in V$.

2. Для $i-$го рядка $r_{ij} = 1$, якщо $v_j \in D\left(v_i\right)$, а якщо ж $v_j \notin D\left(v_i\right)$, то $r_{ij} = 0$.

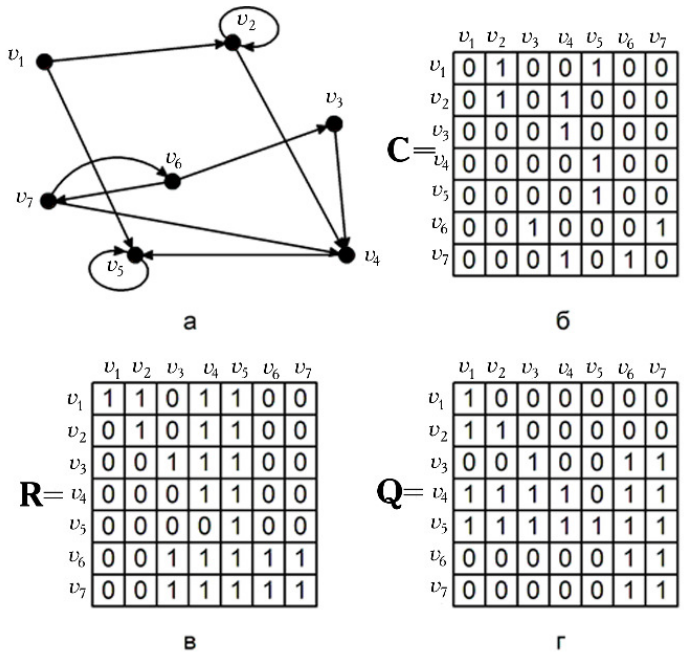



Рис. 3.78. Досяжність у графі: а – граф; б – матриця суміжності; в – матриця досяжності; г – матриця контрдосяжності.

Множини досяжностей знаходять у такий спосіб:

$$D(v_1)=\{v_1\}\cup\Gamma^{+1}(v_1)\cup\Gamma^{+2}(v_1)\cup\Gamma^{+3}(v_1)=$$
$$=\{v_1\}\cup\{v_2,v_5\}\cup\{v_2,v_4,v_5\}\cup\{v_2,v_4,v_5\}=\{v_1,v_2,v_4,v_5\}$$

$$D(v_2)=\{v_2\}\cup\Gamma^{+1}(v_2)\cup\Gamma^{+2}(v_2)=$$
$$=\{v_2\}\cup\{v_2,v_4\}\cup\{v_2,v_4,v_5\}=\{v_2,v_4,v_5\}$$

$$D(v_3)=\{v_3\}\cup\Gamma^{+1}(v_3)\cup\Gamma^{+2}(v_3)\cup\Gamma^{+3}(v_3)=$$
$$=\{v_3\}\cup\{v_4\}\cup\{v_5\}\cup\{v_5\}=\{v_3,v_4,v_5\}$$

$$D(v_4)=\{v_4\}\cup\Gamma^{+1}(v_4)\cup\Gamma^{+2}(v_4)=$$
$$=\{v_4\}\cup\{v_5\}\cup\{v_5\}=\{v_4,v_5\}$$

$$D(v_6)=\{v_6\}\cup\{v_3,v_7\}\cup\{v_4,v_6\}\cup\{v_3,v_5,v_7\}\cup\{v_4,v_5,v_6\}\cup\ldots$$
$$\cup\{v_4,v_5,v_6\}=\{v_3,v_4,v_5,v_6,v_7\},$$

$$D(v_7)=\{v_7\}\cup\{v_4,v_6\}\cup\{v_3,v_5,v_7\}\cup\{v_4,v_5,v_6\}=\{v_3,v_4,v_5,v_6,v_7\}.$$

*Матриця контрдосяжності*

Матриця контрдосяжності – це матриця $n\times n$ $Q=(q_{ij})$, $i,j=1,2,3,\ldots,n$, де $n$ – число вершин графа, визначається в такий спосіб:

$$q_{ij} = \begin{cases} 1, \text{ якщо з } v_j \text{ може бути досягнута } v_i, \\ 0, \text{ у протилежному випадку.} \end{cases}$$

*Контрдосяжною множиною* $K(v_i)$ називають множину вершин, з яких можна досягти вершину $v_i$. Контрдосяжну множину $K(v_i)$ визначають з виразу:

$$K(v_i) = \{v_i\} \cup \Gamma^{-1}(v_i) \cup \Gamma^{-2}(v_i) \cup ... \cup \Gamma^{-p}(v_i).$$

*Співвідношення між матрицями досяжності і контрдосяжності*

**Визначення.** *Матриця контрдосяжності* дорівнює транспонованій матриці досяжності $Q = R^T$.

Дане співвідношення випливає з визначення матриць, оскільки стовпець $v_i$ матриці $Q$ збігається з рядком $v_i$ матриці $R$.

Слід зазначити, що оскільки всі елементи матриць $R$ і $Q$ дорівнюють 1 або 0, то кожний рядок можна зберігати у двійковій формі, заощаджуючи витрати пам'яті комп'ютера. Матриці $R$ і $Q$ зручні для обробки на комп'ютері, оскільки з обчислювальної точки зору основними операціями є швидкодіючі логічні операції.

**Контрольні запитання**

1. Побудувати граф, який характеризується рефлексивністю, симетричністю і транзитивністю.
2. Зобразити графічно та задати матрицею суміжності граф $G(R)$, заданий антитранзитивним та антисиметричним відношенням $R$.
3. Нехай дано множину вершин: $V = \{v_1, v_2, v_3, v_4, v_5\}$ та множину, на якій задано два відношення:

$R_1 = \{(v_1, v_2), (v_2, v_3), (v_4, v_1), (v_4, v_3), (v_4, v_2), (v_3, v_5), (v_1, v_5)\}$,

$R_2 = \{(v_1, v_2), (v_2, v_3), (v_4, v_1), (v_4, v_3), (v_1, v_3), (v_2, v_5), (v_4, v_5)\}$.

Побудувати множину ребер графа, задану на об'єднанні відношень $R_1$ та $R_2$. Представити графічно графи $G(R_1 \cup R_2)$, $G(R_1)$ та $G(R_2)$.
4. Розглянемо неорієнтований граф $G(V, E)$,
де $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$,

$E=\{(v_1,v_2),(v_1,v_9),(v_2,v_3),(v_2,v_8),(v_3,v_4),(v_4,v_{11}),(v_5,v_6),(v_5,v_{11}),$

$(v_6,v_7),(v_7,v_{10}),(v_8,v_{10}),(v_9,v_{10}),(v_7,v_1),(v_7,v_3),(v_{10},v_7),\}.$

Побудувати граф $G(V,E)$ та множину відображення $\Gamma^+(v_7)$.

5. Розглянемо незв'язний граф $G$, показаний на рис. 3.79.

Побудувати множину прямих відображень $\Gamma^+(V)$ даного графа.

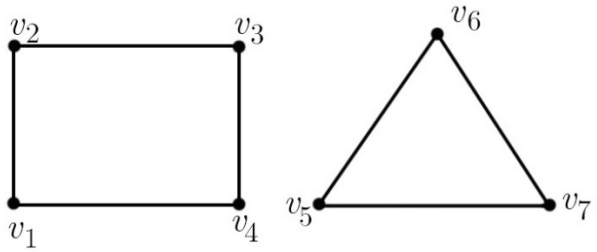



Рис. 3.79. Граф до задачі 5

6. Побудувати матрицю досяжності для графа, показаного на рис. 3.80.

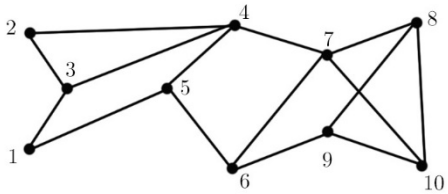



Рис. 3.80 Граф до задачі 6

## 3.4. Числа графа

### 3.4.1. Цикломатичне число

*Цикломатичним числом графа* $G=(V,E)$ називають число

$$m=N-n+p,$$

де $N=|E|$ – число ребер графа, $n=|V|$ – число вершин графа, $p$ – число компонентів зв'язності графа. Для зв'язного графа $m=N-n+1$.

**Теорема 3.6**.

Цикломатичне число графа дорівнює найбільшій кількості незалежних циклів.

*Цикли в графі*

*Циклом* називають шлях, у якому перша й остання вершини збігаються.

*Довжина циклу* **–** число складових його *ребер*.

*Простий цикл* **–** це цикл без повторюваних ребер.

*Елементарний цикл* **–** це простий цикл без повторюваних вершин.

*Наслідок.* Петля – елементарний цикл.

*Вектор-цикл, незалежні цикли*

Поставимо у відповідність циклу $\mu$ графа $G$ деякий вектор.

Для цього додамо кожному ребру графа довільну орієнтацію.

Якщо цикл $\mu$ проходить через ребро $e_k$, де $1 \le k \le N$, у напрямку його орієнтації $r_k$ разів і в протилежному напрямку $s_k$ разів, то вважаємо $c^k = r_k - s_k$.

Вектор $\mathrm{c} = \left(c^1, c^2, c^3, ..., c^k, ..., c^N\right)$ називають вектором-**циклом**, відповідним до циклу $\mu$.

Цикли $\mu_1$ й $\mu_2$ називають **незалежними**, якщо відповідні їм вектори $\mathrm{c}_1 = \left(c_1^1, c_1^2, c_1^3, ..., c_1^k, ..., c_1^N\right)$ та $\mathrm{c}_2 = \left(c_2^1, c_2^2, c_2^3, ..., c_2^k, ..., c_2^N\right)$ лінійно незалежні.

*Властивості циклів*

1. Зв'язний граф *G* не має циклів тоді й тільки тоді, коли цикломатичне число $m = 0$. Такий граф є деревом.

2. Зв'язний граф *G* має єдиний цикл тоді й тільки тоді, коли цикломатичне число $m = 1$.

*Визначення цикломатичного числа*

**Цикломатичне число** зв'язного графа можна визначити як число ребер, які потрібно вилучити, щоб граф перетворити на дерево.

*Визначення лінійної незалежності векторів-циклів*

З курсу лінійної алгебри випливає, що вектори $\mathrm{c}_1 = \left(c_1^1, c_1^2, c_1^3, ..., c_1^k, ..., c_1^N\right)$ й $\mathrm{c}_2 = \left(c_2^1, c_2^2, c_2^3, ..., c_2^k, ..., c_2^N\right)$ можна представити як вектори в просторі $R^N$. Нехай $\alpha$ – деяка константа $\alpha \in R$. Тоді $\alpha\mathrm{c}_1 = \left(\alpha c_1^1, \alpha c_1^2, \alpha c_1^3, ..., \alpha c_1^k, ..., \alpha c_1^N\right)$ і $\alpha\mathrm{c}_2 = \left(\alpha c_2^1, \alpha c_2^2, \alpha c_2^3, ..., \alpha c_2^k, ..., \alpha c_2^N\right)$.

$$\mathrm{c}_1 + \mathrm{c}_2 = \left(c_1^1 + c_2^1, c_1^2 + c_2^2, c_1^3 + c_2^3, ..., c_1^k + c_2^k, ..., c_1^N + c_2^N\right).$$

Деяку множину $E \subset R^N$ називають векторним підпростором, коли

1. $\alpha \in R$ , $\mathrm{c} \in E \Rightarrow \alpha\mathrm{c} \in E$.

2. $\mathrm{c}_1, \mathrm{c}_2 \in E \Rightarrow \mathrm{c}_1 + \mathrm{c}_2 \in E$.

Говорять, що вектори $\mathrm{c}_1, \mathrm{c}_2, \mathrm{c}_3, ....\mathrm{c}_i$ з $R^N$ лінійно незалежні, якщо

$$\alpha_1\mathrm{c}_1 + \alpha_2\mathrm{c}_2 + ... + \alpha_i\mathrm{c}_i = 0 \Rightarrow \alpha_1 = \alpha_2 = ... = \alpha_i = 0.$$

Навпаки, якщо при $\alpha_1\mathrm{c}_1 + \alpha_2\mathrm{c}_2 + ... + \alpha_i\mathrm{c}_i = 0$ деякі $\alpha_i$ одночасно не дорівнюють нулю, то говорять, що дані вектори лінійно залежні.

Якщо, наприклад, $\alpha_1 \ne 0$, то можна записати

$$\mathrm{c}_1 + \frac{\alpha_2}{\alpha_1}\mathrm{c}_2 + \frac{\alpha_3}{\alpha_1}\mathrm{c}_3 + ... + \frac{\alpha_i}{\alpha_1}\mathrm{c}_i = 0.$$

Звідси $\frac{\alpha_2}{\alpha_1}\mathrm{c}_2 + \frac{\alpha_3}{\alpha_1}\mathrm{c}_3 + ... + \frac{\alpha_i}{\alpha_1}\mathrm{c}_i = -\mathrm{c}_1$

У цьому випадку вектор $c_1$ лінійно виражений через вектори $c_2, c_3, ..., c_i$.
Для визначення факту лінійної залежності векторів необхідно розв'язати систему $\alpha_1 c_1 + \alpha_2 c_2 + ... + \alpha_i c_i = 0$

$$\alpha_1 \begin{pmatrix} c_1^1 \\ c_1^2 \\ \vdots \\ c_1^N \end{pmatrix} + \alpha_2 \begin{pmatrix} c_2^1 \\ c_2^2 \\ \vdots \\ c_2^N \end{pmatrix} + ... + \alpha_i \begin{pmatrix} c_i^1 \\ c_i^2 \\ \vdots \\ c_i^N \end{pmatrix} = 0, \text{ або } \begin{cases} \alpha_1 c_1^1 + \alpha_2 c_2^1 + ... + \alpha_i c_i^1 = 0, \\ \alpha_1 c_1^2 + \alpha_2 c_2^2 + ... + \alpha_i c_i^2 = 0, \\ ... \\ \alpha_1 c_1^N + \alpha_2 c_2^N + ... + \alpha_i c_i^N = 0. \end{cases}$$

**Приклад 3.39.** Визначити цикломатичне число графа $G(V,E)$, показаного на рис. 3.81.

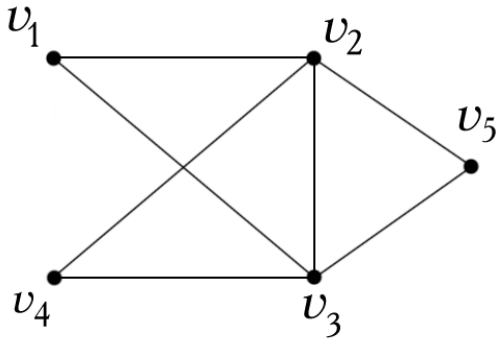



Рис. 3.81. Граф $G(V,E)$

*Розв'язок.*
У розглянутому графі число вершин $n=5$, число ребер $N=7$. Оскільки граф є зв'язним, то число компонентів зв'язності $p=1$. Таким чином, $m = N - n + p = 7 - 5 + 1 = 3$.

### 3.4.2. Число внутрішньої стійкості

Нехай дано граф $G(V,\Gamma)$. Множину $S \subset V$ називають внутрішньо стійкою, якщо ніякі дві вершини, що входять в $S$, не є суміжними. Іншими словами сформулюємо цю умову, використовуючи відображення першого порядку:

$$\Gamma^+(S) \cap S = \varnothing.$$

Якщо позначити через $\Phi$ сімейство всіх внутрішньо стійких множин графа, то для нього будуть справедливі співвідношення:
1. $\varnothing \in \Phi$, $S \in \Phi$.
2. Якщо $A \subset S$, то $A \in \Phi$.
**Визначення**. Числом *внутрішньої стійкості* графа $G$ є величина, яку визначають з виразу: $a = \max\limits_{S \in \Phi} |S|$.

**Визначення** $S \subset V$ *називають множиною внутрішньої стійкості, якщо всі вершини з* $S$ *не суміжні між собою. Потужність найбільшої множини внутрішньої стійкості називають числом внутрішньої стійкості.*

**Приклад 3.40.** Знайти число внутрішньої стійкості графа $G(V,E)$ (рис. 3.82).

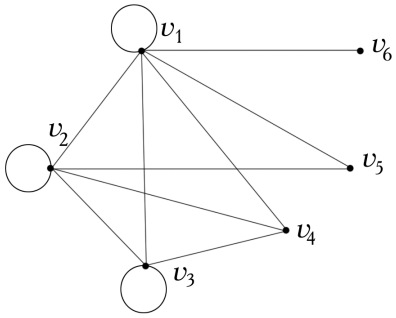



Рис. 3.82. Граф $G(V,E)$ для визначення внутрішньої стійкості

*Розв'язок.*

Найбільша множина внутрішньої стійкості для нашого графа має вигляд $S=\{v_4,v_5,v_6\}$ (при додаванні будь-яких інших вершин будемо одержувати суміжні вершини). Відповідно, *число внутрішньої стійкості* графа $G$ дорівнює $a=3$.

**Приклад 3.41.** Знайти число внутрішньої стійкості графа $G(V,E)$, показаного на рис. 3.83:

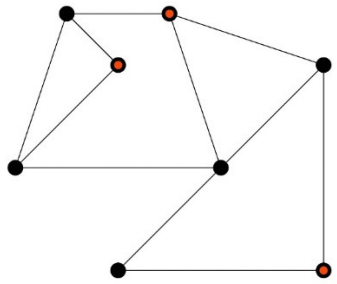


Рис. 3.83. Граф $G(V,E)$ для визначення внутрішньої стійкості графа

Всі відмічені вершини даного графа є несуміжними. Потужність множини відмічених несуміжних вершин є максимальною. Тому число внутрішньої стійкості для даного графа дорівнює 3.

### 3.4.3. Число зовнішньої стійкості

Нехай даний граф $G(V,\Gamma)$. Говорять, що множина вершин $T\subset V$ зовні стійка, якщо для кожної вершини $v\notin T$ маємо $\Gamma^{+}(v)\cap T\neq\varnothing$, інакше кажучи $V\setminus T\subset\Gamma^{-1}(T)$. Якщо $\Psi$ – сімейство всіх зовні стійких множин графа, то для нього слушні такі співвідношення:

1. $T\in\Psi$.

2. Якщо $T \subset A$, то $A \in \Psi$.

**Визначення.** Число *зовнішньої стійкості* $b$ графа $G$ є величина, яку одержують з виразу: $b = \min\limits_{T \in \Psi} |T|$.

**Зовні стійка множина** – множина вершин *T* таких, що будь-яка вершина графа або належить *T,* або суміжна з вершиною з *T*.

**Приклад 3.42.** Знайти число зовнішньої стійкості у графі (рис. 3.84).

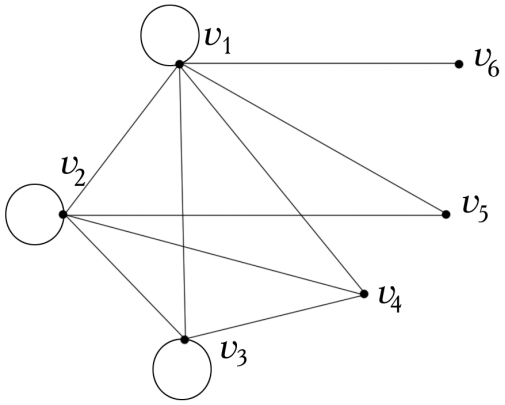



Рис. 3.84. Граф $G(V,E)$ для визначення зовнішньої стійкості графа

*Розв'язок.*

Для графа, показаного на рис. 3.84, найменша множина зовнішньої стійкості має вигляд $T = \{v_1\}$ (оскільки будь-яка інша вершина (не приналежна $T$) з'єднана з вершиною $v_1$ з $T$).

*Число зовнішньої стійкості* графа *G* дорівнює $b = 1$.

**Контрольні запитання**

1. Визначити цикломатичне число зв'язного графа $G(V,E)$, якщо граф характеризується множиною вершин $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, а потужність множини ребер $|E| = 12$.

2. Нехай дано граф $G(V,E)$, де $V = \{v_1, v_2, v_3, v_4, v_5\}$, $E = \{(v_1, v_2), (v_5, v_2), (v_3, v_4), (v_1, v_4), (v_1, v_5), (v_3, v_2), (v_4, v_5), (v_4, v_2)\}$

Скільки ребер потрібно видалити, щоб граф став деревом? Визначити підмножини ребер, які можуть бути видалені без втрати зв'язності графа.

3. Визначити число внутрішньої стійкості для графа, показаного на рис. 3.85.

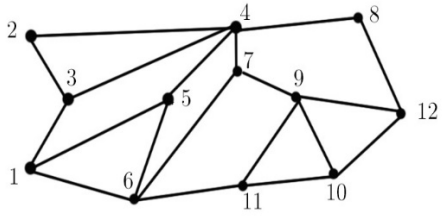



Рис. 3.85 Граф до задач 3 та 4

4. Визначити число зовнішньої стійкості для графа, показаного на рис. 3.85.

## 3.5. Дерева та їхні властивості, ліс, цикли

### 3.5.1. Визначення дерева. Властивості дерев

**Визначення.** *Неорієнтованим деревом* називають зв'язний неорієнтований граф без циклів (рис. 3.86).

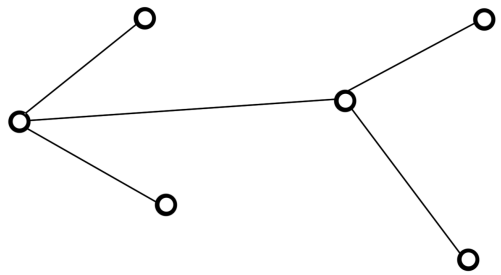


Рис. 3.86. Неорієнтоване дерево

*Кореневим деревом* називають таке дерево, у якому існує виділена вершина, що має назву *кореня.*
***Корінь*** у неорієнтованому графі – це одна з вершин, обрана за бажанням спостерігача.

**Орієнтованим деревом** називають зв'язний орієнтований граф без циклів, у якому напівстепінь входу кожної вершини, за винятком кореневої, дорівнює одиниці, а напівстепінь входу кореневої вершини дорівнює 0 (рис. 3.87).

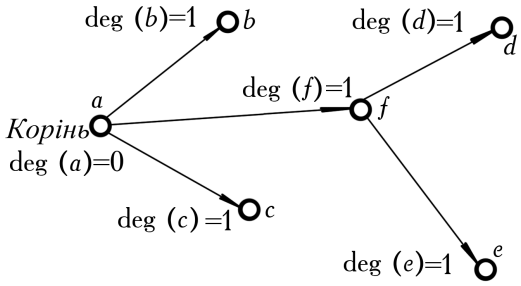
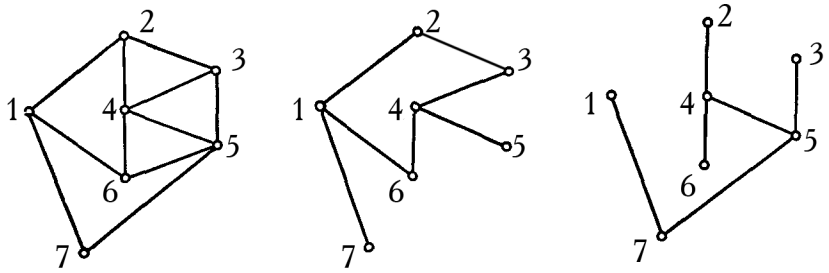



Рис. 3.87. Орієнтоване дерево

*Остовним підграфом* називають такий підграф, у якому множина його вершин збігається з множиною вершин самого графа.

*Остовним деревом* графа *G* називають ***остовний підграф*** графа *G*, який є деревом.

Початковий граф, остовний підграф та остовне дерево, які утворені шляхом послідовного видалення ребер, показані на рис. 3.88.




Початковий граф | Остовний підграф | Остовне дерево

Рис. 3.88. Утворення остовного підграфа та остовного дерева шляхом видалення ребер

**Теорема 3.7.** Граф є деревом тоді і тільки тоді, коли будь-які дві його вершини зв'язані єдиним ланцюгом.

*Доведення.*

Нехай граф є деревом. Якщо припустити існування більш ніж одного ланцюга, що зв'язує будь-які дві його вершини, то в такому графі існує цикл, тобто граф не може бути деревом (рис. 3.89).

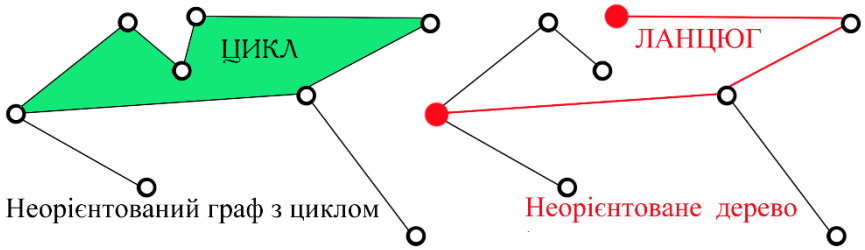



Рис. 3.89. Граф з циклом та неорієнтоване дерево

Навпаки, оскільки будь-які дві вершини графа з'єднані ланцюгом, то граф зв'язний, а в силу того, що цей ланцюг єдиний, він не має циклів. Отже, граф є деревом. *Теорема доведена*.

**Наслідок 3.7.1.** Якщо $T$ – дерево і $u$ – його кінцева вершина (листок), то граф $T-u$ ($T$ *мінус* $u$) – дерево.

Дійсно, граф $T-u$ – правильний підграф дерева $T$, для якого виконуються всі умови теореми (рис. 3.90).

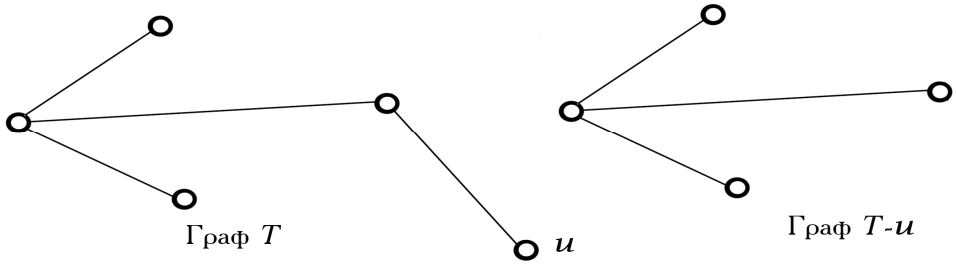



Рис. 3.90. Неорієнтовані дерева $T$ і $T-u$

**Наслідок 3.7.2.** Будь-яке непусте дерево має принаймні дві висячі вершини і одне висяче ребро.

1. *Висяча вершина в неорієнтованому графі* – це вершина степеня 1.

2. *Висяча вершина в орграфі* – вершина з напівстепенем входу, що дорівнює 1, і напівстепенем виходу, що дорівнює 0.
3. *Висяче ребро* – це ребро, інцидентне вершині зі степенем 1.

**Визначення.** Ребро зв'язного графа називають *істотним*, якщо його видалення веде до порушення зв'язності цього графа.

**Визначення.** У неорієнтованому графі істотним ребром є міст.

**Теорема 3.8.** У дереві кожне ребро істотне.

*Доведення.* Доведення випливає з того, що видалення ребра $e=(u,v)$ у дереві $T$ через наявність єдиного ланцюга, який з'єднує вершини $u$ і $v$, веде до появи двох компонентів зв'язності: один компонент містить вершину $u$, а інший – вершину $v$. Отже, граф $T-e$ не є зв'язним, як показано на рис. 3.91.

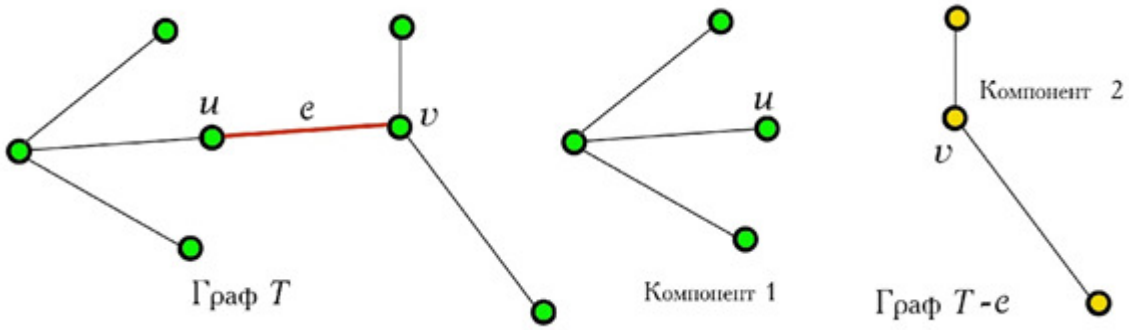



Рис. 3.91. Зв'язний граф $T$ і незв'язний граф $T-e$

**Теорема 3.9.** Якщо $T=(V,E)$ – дерево і $v\notin V$ – вершина, $u$ – довільна вершина $u\in V$, то граф $T'=\left(V\cup\{v\},E\cup\{(u,v)\}\right)$, теж є деревом.

*Доведення.* Оскільки $T$ – дерево, то існує єдиний ланцюг, що з'єднує будь-яку вершину $u'$ з вершиною $u$. Оскільки вершина $v\notin V$, то додавання одного кінцевого ребра $(u,v)$ приводить до того, що з кожної вершини $u'$ маємо лише єдиний ланцюг, який з'єднує вершини $u'$ і $v$. Виходячи з теореми 3.7, граф $T'$ є деревом.

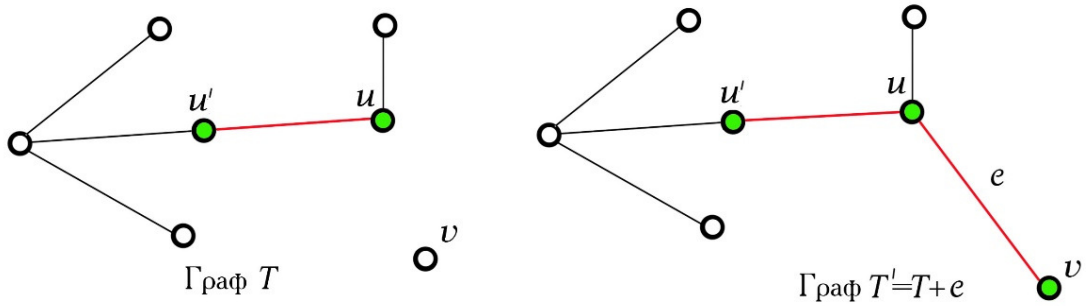



Рис. 3.92. Зв'язні графи-дерева $T$ і $T+e$

**Теорема 3.10.** Нехай дерево $T$ має $n$ вершин. Тоді еквівалентними є такі твердження:

1. $T$ не має циклів і має $n-1$ ребро.
2. $T$ – зв'язний граф і має $n-1$ ребро.

3. $T$ – зв'язний граф і кожне його ребро є мостом.

4. Будь-які дві вершини графа $T$ з'єднані тільки одним простим ланцюгом.

5. $T$ не має циклів, але додавання будь-якого нового ребра до $T$ сприяє виникненню тільки одного циклу.

**Визначення.** *Ліс* – незв'язний $n$ -граф без циклів.

1. Зв'язні компоненти лісу є деревами.
2. Будь-яка частина лісу або дерева також не має циклів.
3. Будь-яка частина лісу є лісом або деревом.

Приклад лісу показаний на рис. 3.93.

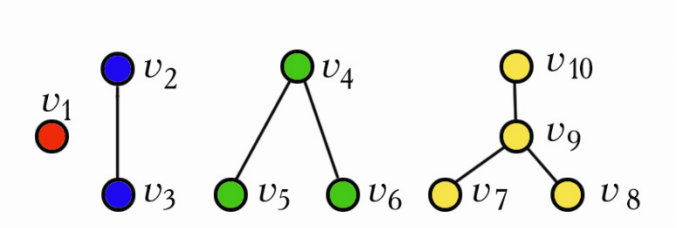



Рис. 3.93. Ліс з чотирьох дерев

**Теорема 3.11.** Нехай ліс $G$ містить $n$ вершин і $k$ компонентів. Тоді ліс $G$ має $n-k$ ребер.

*Доведення.* З умови 2 теореми 3.10 випливає, що кожний компонент $G_i$ має $(n_i-1)$ ребро. Але тоді число ребер у $G$ дорівнює

$$(n_1-1)+(n_2-1)+\ldots+(n_k-1)=n_1+n_2+\ldots+n_k-k=n-k,$$

що і потрібно довести.

**Теорема 3.12.** *Теорема Келі.* Число різних дерев, які можна побудувати на $n$ вершинах, дорівнює $n^{n-2}$.

### 3.5.2. Процедури побудови остовного дерева та лісу

*Процедура побудови остовного дерева*

1. Видалення із зв'язного графа $G$ одного ребра, що належить деякому циклу, не порушує зв'язності графа $G$.
2. Застосуємо процедуру видалення ребра до одного з циклів у графі $G$.
3. Будемо повторювати видалення доти, поки в $G$ не залишиться жодного циклу.
4. У результаті одержимо дерево, що містить усі вершини графа $G$. Це дерево називають *остовним деревом графа* $G$.

Приклад побудови остовного дерева шляхом почергового видалення ребер показано на рис. 3.94.

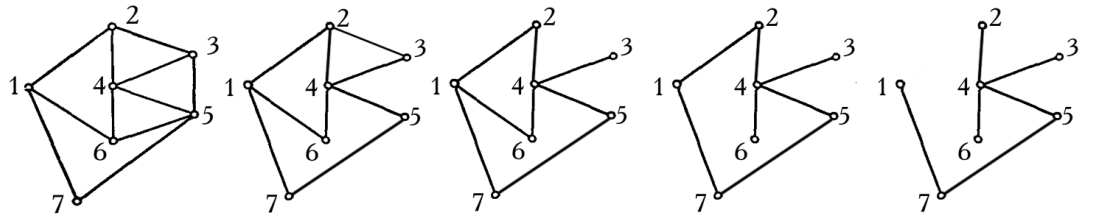



Рис. 3.94. Приклад побудови остовного дерева

*Процедура побудови лісу*

Нехай $G$ є графом з $n$ вершинами, $m$ ребрами і $k$ компонентами.

1. Застосуємо процедуру видалення ребра до одного з циклів у кожному компоненті зв'язності графа $G$.
2. Будемо повторювати видалення доти, поки в кожному компоненті $G$ не залишиться жодного циклу.
3. У результаті одержимо граф, який називають *остовним лісом.*
4. Число ребер, які при цьому видаляються, називають *цикломатичним числом* або *циклічним рангом графа* $G$ і позначають $C(G)$. Таким чином, цикломатичне число є мірою зв'язності графа.

Приклад побудови остовного лісу шляхом почергового видалення ребер показано на рис. 3.95.

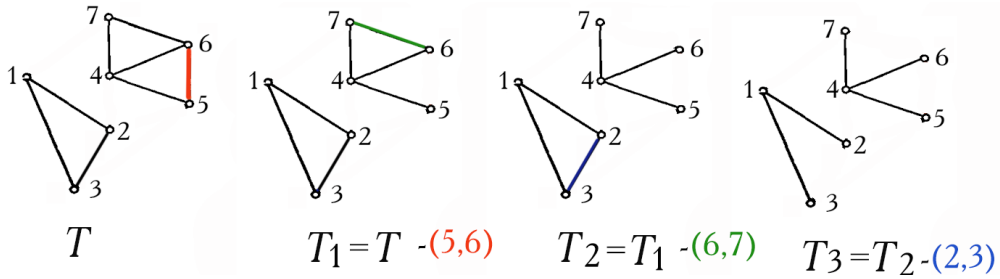



Рис. 3.95. Приклад побудови остовного лісу: $C(G)=3$

### 3.5.3. Властивості циклічного рангу

1. *Циклічний ранг дерева* дорівнює нулю.
2. *Циклічний ранг циклічного графа* дорівнює одиниці.

**Визначення.** *Циклічний граф* – зв'язний регулярний граф степеня 2, єдиний компонент зв'язності циклічного графа є простим циклом (рис. 3.96).

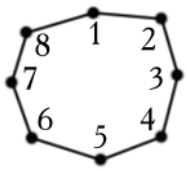
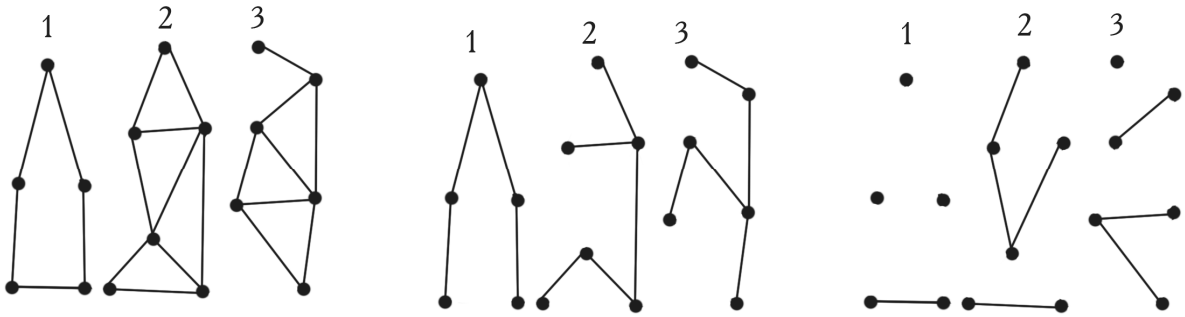


Рис. 3.96. Циклічний граф

**Визначення.** *Остовне дерево графа* – це дерево, що містить усі вершини графа.

**Визначення.** *Остовним лісом* називають незв'язний граф, що складається з компонентів, які є остовними деревами.

**Теорема 3.13.** Нехай $T$ – остовний ліс графа $G$. Граф $G'$, отриманий із графа $G$ шляхом видалення всіх ребер графа $T$, називають *доповненням* остовного лісу $T$ графа $G$ (рис. 3.97).



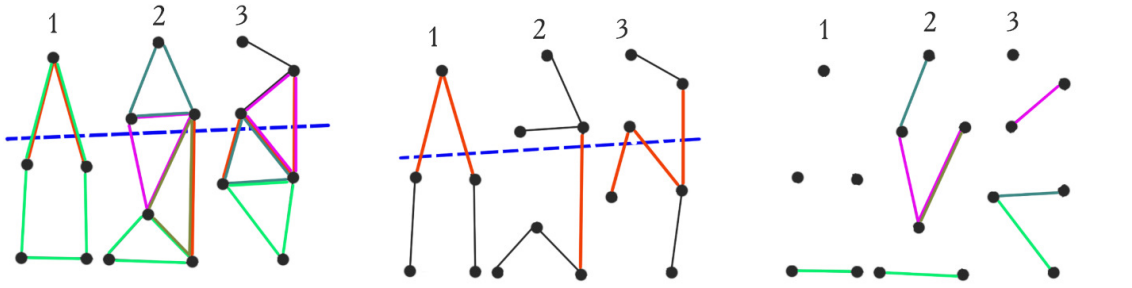

Незв'язний граф $G$ | Остовний ліс $T$ | $G'$-доповнення остовного лісу $T$

Рис. 3.97. Побудова доповнення остовного лісу $G'$

**Теорема 3.14.** Якщо $T$ – остовний ліс графа $G$, то

а) будь-який розріз в $G$ має загальне ребро з $T$;

б) будь-який цикл у $G$ має загальне ребро з доповненням $T$.




Незв'язний граф $G$ | Остовний ліс $T$ | Доповнення остовного лісу $G'$

Рис. 3.98. Властивості розрізу незв'язного графа

### 3.5.4. Фундаментальна система циклів графа

Нехай $T$ – остовний ліс графа $G$.

Якщо додати до $T$ будь-яке ребро графа $G$, що не входить до нього, то за п. 5 теореми 3.10 одержимо єдиний цикл.

**Визначення.** Множину всіх циклів, які одержують шляхом додавання окремо кожного ребра з $G$, що не входить до $T$, називають фундаментальною системою циклів, асоційованою з $T$.

Цикли даної фундаментальної системи будуть різними, але їх кількість дорівнює циклічному рангу графа $G$.

На рис. 3.99 графи показують фундаментальну систему циклів.

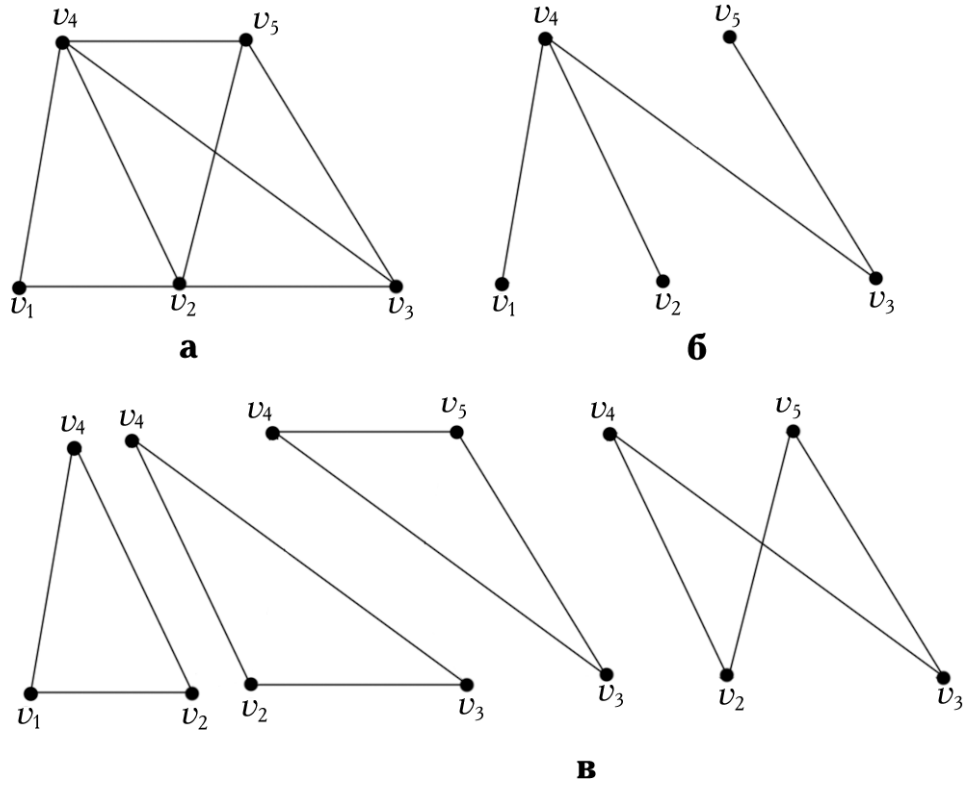



Рис. 3.99: Граф та його фундаментальні цикли: **а** – граф $G$; **б** – остовне дерево графа $G$; **в** – фундаментальна система циклів, асоційована з остовним деревом графа $G$

**Теорема 3.15.** Нехай $G=(V,E)$ – довільний скінченний граф. Число ребер графа *G*, які необхідно вилучити для одержання остовного лісу *T*, не залежить від порядку їх видалення і дорівнює $C(G)=|E|-|V|+k$, де $k$ – число компонентів зв'язності графа *G*.

*Наслідок* 3.15.1. Граф $G$ є остовним лісом тоді і тільки тоді, коли $C(G)=0$.

*Наслідок* 3.15.2. Граф $G$ має єдиний цикл тоді і тільки тоді, коли $C(G)=1$.

*Наслідок* 3.15.3. Граф G, у якого число ребер перевищує число вершин, має цикл.

*Наслідок* 3.15.4. Будь-яке дерево порядку $n\geq 2$ має принаймні дві кінцеві вершини (рис. 3.100).

Рис. 3.100. Мінімальний граф з двома кінцевими вершинами

### 3.5.5. Остовне дерево найменшої ваги

Нехай кожному ребру графа $G=(V,E)$ поставлена у відповідність деяка вага

$$d_i = d(e_i), \quad e_i \in E, \quad i = 1,2,...,|E|.$$

Необхідно в графі $G$ знайти остовне дерево, сума ваг $d_i$ у якому найменша.

$$S = \min \sum_{e_i \in E} \left(d(e_i)\right)$$

Число $d_i$ називають *вагою ребра* $e_i$, а сам граф $G$ – таким, що має *вагу (зваженим)*. Отже, завдання полягає в тому, щоб знайти остовне дерево з найменшою вагою.

*Практичні застосування задачі пошуку остовного дерева з мінімальною вагою.*

Ця задача виникає при проєктуванні доріг, ліній електропередач, трубопроводів та ін., коли необхідно з'єднати задані точки деякою системою зв'язку так, щоб загальна довжина ліній зв'язку була мінімальною. На рис. 3.101 показані об'єкти, при проєктуванні яких застосовують розв'язування задачі пошуку остовного дерева з мінімальною вагою.



Рис. 3.101. Приклади об'єктів, для яких необхідна мінімізація параметрів

Розглянемо кілька методів розв'язування цієї задачі.

*Розв'язування, що випливає з теореми Келі*

1. Розглянемо всі можливі остовні дерева повного графа $G$ з $n$ вершинами.

Згідно з теоремою Келі число *різних дерев, які можна побудувати на* $n$ *вершинах, дорівнює* $n^{n-2}$.

2. При виборі кожного дерева визначимо суму його ваг.

3. Застосувавши сортування за величиною суми ваг, на початку списку одержимо ті остовні дерева, які мають мінімальну суму ваг. Але оскільки число $n^{n-2}$ навіть при невеликому $n$ буде дуже великим, то такий шлях розв'язування даної задачі досить трудомісткий.

*Тому актуальним є використання більш ефективних алгоритмів.*

**Попередні зауваження**

1. **Порядок графа** – число, яке дорівнює кількості вершин графа.
2. **Порожній граф** – граф, що не містить ребер або регулярний граф степеня 0.

### 3.5.6. Алгоритм Краскала

*Алгоритм Краскала* полягає у виконанні такої послідовності дій:

1. Беремо порожній граф $O$ і будуємо граф $T_1 = O + e_1$,

де $e_1$ – ребро графа $G = (V, E)$ мінімальної ваги.

2. Якщо граф $T_k$ уже побудований і $k < n-1$, то будуємо граф $T_{k+1} = T_k + e_{k+1}$, де $e_{k+1}$ – ребро мінімальної ваги серед ребер графа $G$, що не ввійшли в граф $T_k$, яке не утворює циклу з ребрами графа $T_k$.

Цей алгоритм базується на спеціальній теоремі.

**Теорема 3.16.** Нехай $G$ – зв'язний граф порядку $n$ і $T$ – підграф графа $G$, отриманий у результаті виконання кроків 1 і 2 алгоритму Краскала. Тоді підграф $T$ – остовне дерево графа $G$ мінімальної ваги.

**Приклад 3.43.** Нехай дано граф $G$, показаний на рис. 3.102. Необхідно знайти остовне дерево мінімальної ваги цього графа за допомогою алгоритму Краскала.

*Розв'язок.*

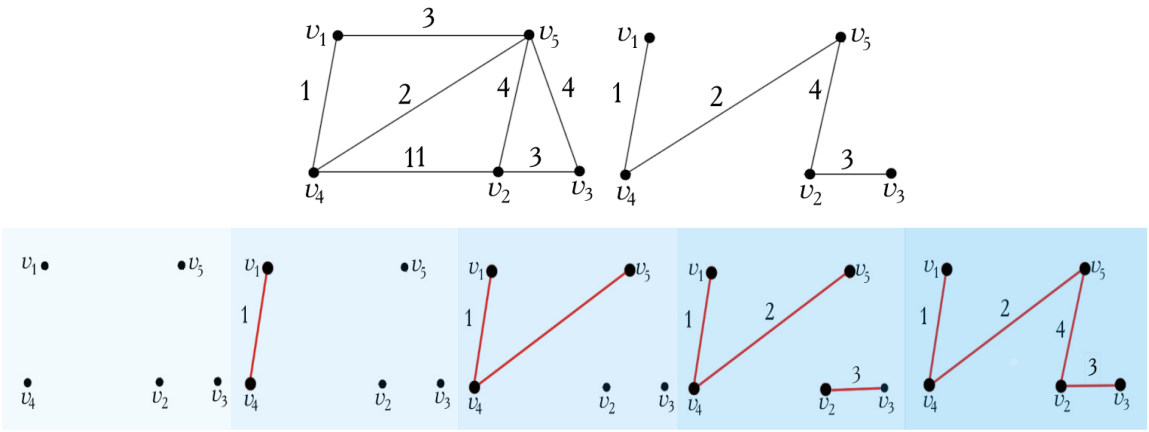



Рис. 3.102. Застосування алгоритму Краскала

*Властивості алгоритму Краскала*

1. Алгоритм використовується для зважених неорієнтованих графів.

2. Алгоритм може бути використаний для побудови остовного лісу в багатокомпонентних незв'язних графах.

У цьому випадку **структури даних**, що описують кожну з компонент зв'язності, **повинні бути окремими**.

3. Обчислювальна складність алгоритму Краскала $O(e \cdot \log e)$, де $e$ – кількість ребер у даному графі.

### 3.5.7. Алгоритм Прима

*Алгоритм Прима* полягає у виконанні такої послідовності дій:

1. У порожньому графі $O$ вибираємо довільну **вершину**.

2. Вибираємо ребро $e_1$ мінімальної ваги до суміжної вершини і будуємо підграф $T_1 = O + e_1,$.

3. Якщо граф $T_k$ уже побудований і $k < n-1$, то будуємо граф $T_{k+1} = T_k + e_{k+1}$, де $e_{k+1}$ – ребро мінімальної ваги, що з'єднує вершину графа $T_k$ із суміжною вершиною, яка не включена в множину вершин графа $T_k$.

**Приклад 3.44.** Нехай даний граф $G$. Необхідно знайти остовне дерево мінімальної ваги цього графа за допомогою алгоритму Прима.

*Розв'язок.*

1. Вибираємо довільну вершину $r$ і проводимо інцидентне ребро мінімальної ваги.

2. Потім, переглядаючи інцидентні ребра на кожній з кінцевих вершин, знаходимо ребро мінімальної ваги.

На рис. 3.103 показано процес формування мінімального остовного дерева за алгоритмом Прима.

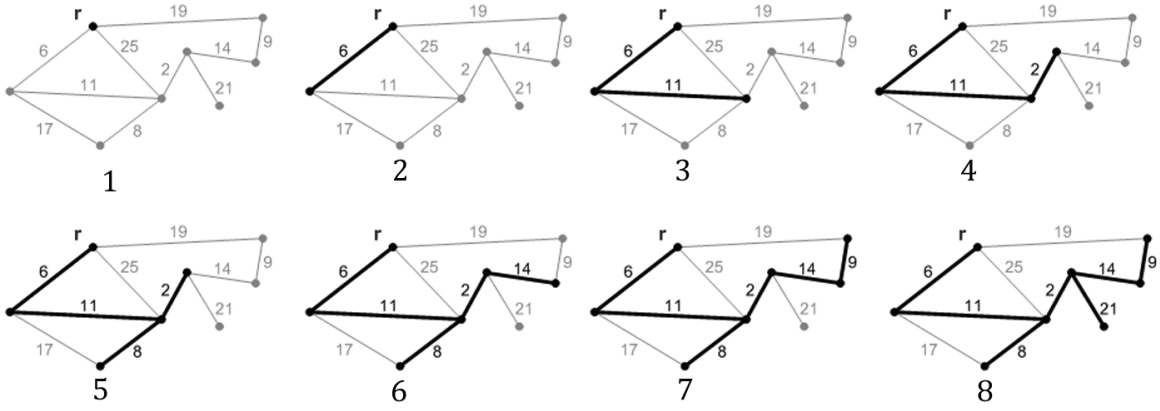



Рис. 3.103. Застосування алгоритму Прима

*Властивості алгоритму Прима*

1. Алгоритм використовується для зважених **неорієнтованих зв'язних** графів.

2. Обчислювальна складність алгоритму Прима $O\left(n^2\right)$, де $n$ – кількість вершин графа. Якщо значення $n$ достатньо **велике**, то використовувати цей алгоритм **не раціонально**.

3. Якщо кількість ребер $e$ **значно меншою за** $n^2$, то алгоритм Краскала кращий, але якщо $e$ **близька до** $n^2$, то рекомендують застосовувати алгоритм Прима.

*Структура алгоритму Прима*

Даний граф $G(V,E)$, де множина вершин $V=\{1,2,...,i,...,n\}$

$U=\varnothing$ –множина вершин остовного дерева.

$T=\varnothing$ – множина ребер остовного дерева.

```
def Prim (G):
  u_i = v_i  #задаємо довільну вершину дерева
  U = {u_i}
  while U ≠ V :
    Знаходимо ребро (u_i, v_j) найменшої ваги і
    таке, що u_i ∈ U і v_j ∈ V \ U
    T = T ∪ {(u_i, v_j)}
    U = U ∪ {v_j}
print(T)
```

**Контрольні запитання**

1. Розглянемо ліс, який складається з трьох компонентів. Множина вершин лісу $V=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{10}\}$. Знайти потужність множини ребер лісу.

2. Дано множину вершин графа $V=\{v_1,v_2,v_3,v_4\}$. Обчислити кількість дерев, які можна побудувати на множині вершин $V$. Побудувати всі можливі дерева.

3. Побудувати остовний ліс та доповнення остовного лісу для незв'язного графа, показаного на рис. 3.104.

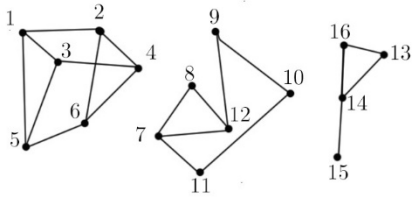



Рис. 3.104. Граф до задачі 3

4. Визначити кількість фундаментальних циклів у графі, показаному на рис. 3.105 та побудувати підграфи, кожен з яких містить тільки один фундаментальний цикл даного графа.

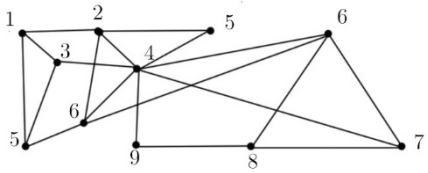



Рис. 3.105. Граф до задачі 4

5. Дано зв’язний неорієнтований зважений граф $G$, показаний на рис. 3.106. Побудувати послідовність створення остовного дерева мінімальної ваги за алгоритмом Краскала.

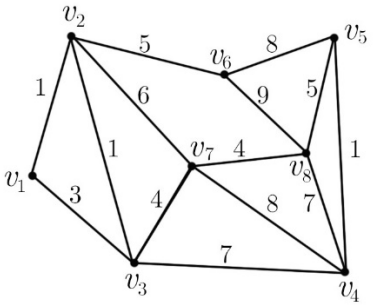



Рис. 3.106. Граф до задач 5 та 6

6. Побудувати остовне дерево мінімальної ваги за алгоритмом Прима, використовуючи граф, показаний на рис. 3.106.

## 3.6. Обхід графів. Основні положення

**Обійти граф** – означає побувати у всіх вершинах точно по одному разу.
**Алгоритми обходу** складаються з:
1. Послідовного відвідування вершин. При цьому:

1.1. Вершину, яка ще **не відвідана, називають новою**.

1.2. **Факт відвідування вершини запам’ятовується**, так що з моменту відвідування і до кінця роботи алгоритму вона вважається відвіданою.

1.3. **У результаті відвідування** вершина **стає відкритою** і залишається такою, поки не будуть досліджені всі інцидентні їй ребра.

1.4. **Після дослідження** всіх інцидентних **ребер** вона перетворюється на ***закриту***.
2. Дослідження ребер.

Які саме дії виконуються при відвідуванні вершини і дослідженні ребра – залежить від конкретної задачі
**Алгоритми обходу графа:** обхід графа у глибину, обхід графа у ширину.

### 3.6.1. Обхід у глибину

Обхід у глибину – це обхід графа за правилами, які наведено на прикладі графа, показаного на рис. 3.107.

1. Перебуваючи у вершині $x$, потрібно рухатися **в будь-яку іншу** $y$, раніше не відвідану вершину (якщо така знайдеться), одночасно **запам’ятовуючи ребро**, по якому ми вперше потрапили в дану вершину.

2. Якщо з вершини $z$ ми **не можемо потрапити** в раніше не відвідану вершину або такої немає, то ми **повертаємося у вершину** $y$, з якої вперше потрапили в $z$, і **продовжуємо обхід** у глибину з вершини $y$.

При виконанні обходу графа за цими правилами ми прагнемо проникнути «углиб» графа так далеко, як тільки це можливо, потім відступаємо на крок назад і знову прагнемо пройти вперед, і так далі.

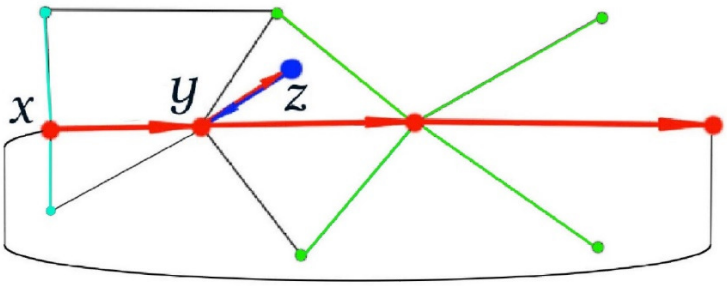



Рис. 3.107. Граф з обходом у глибину

### 3.6.2. Програма обходу графа у глибину

Пошук у глибину починаємо від будь-якої вершини. Рекурсивно застосуємо до всіх вершин, у які можна потрапити з поточної, таку послідовність дій.

1. Задамо граф з $n$ вершин матрицею суміжності $A$ розміром $n \times n$ у вигляді масиву $A[n,n]$.

2. Задамо одновимірний масив $Visited[n]$ і заповнимо його нулями, уважаючи, що жодна вершина до початку виконання алгоритму не була відвідана.

3. Задамо рекурсивну процедуру обходу графа в глибину.

```
def go(Curr):
    Visited[Curr]=1#Познач. поточну вершину як пройдену
    for i in range(1,n+1):
        If Visited[i]==0 AND (A[Curr,i]==1):
            Go(i)
Go(Start)
```

Приклад обходу графа у глибину показаний на рис. 3.108.

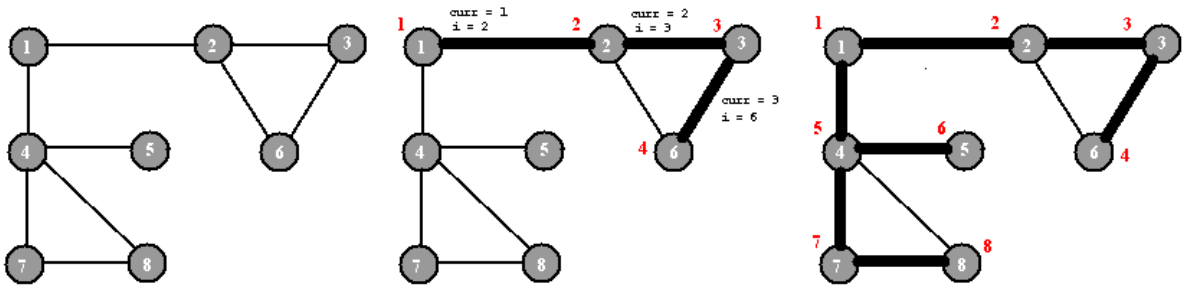



Рис. 3.108. Обхід графа у глибину

### 3.6.3. Обхід у ширину

Обхід у ширину – це обхід графа за правилами, які наведемо на прикладі графа, показаного на рис. 3.109.

1. Нехай ми перебуваємо у початковій вершині $a$.

2. Із цієї вершини відбувається перегляд відразу всіх ще не переглянутих вершин, суміжних вершині $a$. Таким чином, пошук ведеться у всіх можливих напрямках одночасно.
3. Потім відбувається перегляд вершин, що перебувають від $a$ на відстані 2, і т. д.

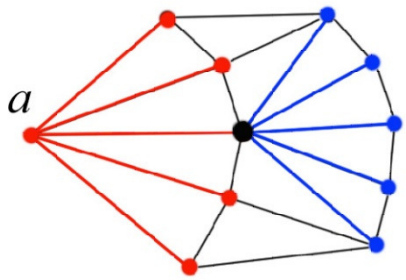



Рис. 3.109. Граф з обходом у ширину

Чим ближче вершина до стартової вершини, тем раніше вона буде відвідана. Обхід в ширину шукає найкоротший можливий шлях.

### 3.6.4. Програма обходу графа у ширину

**При пошуку в ширину** замість стека рекурсивних викликів **використовується черга**, у яку записуються вершини в порядку віддалення від початкової. $n = |V|$.

1. **Задамо масив** $Visited[n]$ і заповнимо його нулями, вважаючи, що жодна вершина до початку виконання алгоритму не була відвідана.
2. **Створимо чергу** для зберігання вершин у вигляді списку $queue[n]$. У початок черги запишемо початкову вершину. $queue[0] = a$.
3. **Змінна** $rd$ вказує на позицію в черзі, з якої ми **читаємо** дані.
4. **Змінна** $wt$ вказує на позицію в черзі, куди ми будемо **писати** дані.
5. Задамо процедуру обходу в ширину.

```
rd=0 # Read from 0
wt=1 # Write to 1
queue[0]=start # The Start vertex Number
while (rd < wt):
 Curr = queue[rd]# chose the active vertex
 for i in range(1,len(V)+1):# visit all the adjacent vertices
   if (not Visited[i]) and A[curr,i]:
      Visited[i]= 1 # mark the visited vertex
      queue[wt]= i  # put it in the queue of active vertices
        wt+=1
      rd+=1
```

Приклад обходу графа у ширину показаний на рис. 3.110.

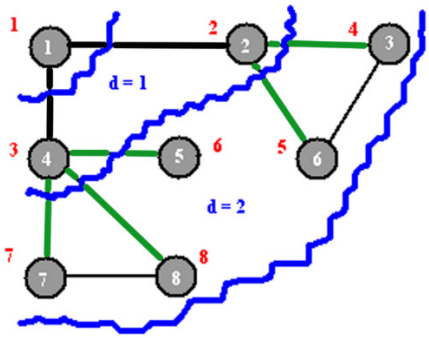


Рис. 3.110. Обхід графа у ширину

**Контрольні запитання**

1. Сформулюйте задачу обходу графа та базовий алгоритм обходу. Яку вершину називають закритою вершиною?
2. Опишіть принципи, на яких ґрунтується алгоритм обходу графа в глибину. Яка умова зупинки алгоритму обходу графа в глибину?
3. Написати програму та вивести на друк проміжні результати обходу графу (рис. 3.111) в глибину.

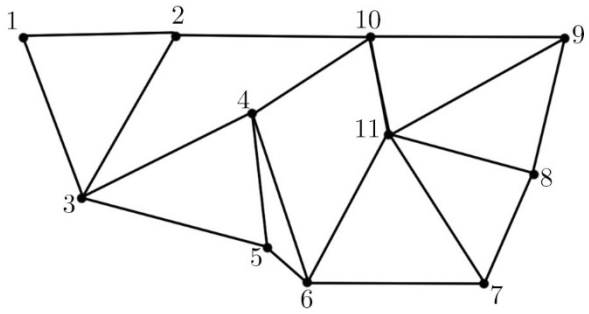


Рис. 3.111. Граф до задач 3 та 4

4. Створити програму обходу графа в ширину. Застосувати дану програму до графа, показаного на рис. 3.111.

## 3.7. Алгоритми пошуку найкоротших шляхів у графі

### 3.7.1. Пошук шляхів у графі за алгоритмом Террі

Виходячи з **початкової вершини** й здійснюючи послідовний перехід **від кожної досягнутої вершини до суміжної вершини**, слід додержуватися таких правил.

1. Позначати напрямок, у якому проходимо ребро, як у прямому, так і у зворотному напрямках.
2. З будь-якої вершини просуватися тільки по тому ребру, яке ще не було пройдено або було пройдено в протилежному напрямку.

3. Для кожної вершини позначати перше ребро, по якому до неї потрапили, якщо вершина зустрічається вперше.
4. Будь-яку вершину залишати по ребру, по якому прийшли в цю вершину (у зворотному напрямку) лише тоді, коли немає іншої можливості.

*V – множина вершин графа; m – кількість ребер графа;*
*M – матриця суміжності графа. P-орієнтований цикл, що проходить через кожне ребро графа по одному разу в кожному з двох напрямків.*
*Вибрати вершину* $a \in V$; P=0; $k = 2m$; $V' = V$
**while** $k \neq 0$ **:**
*вибрати ребро* $(a,b)$, *для якого* $M(a,b) = 1$,
причому в останню чергу вибирати ребро $(a,b)$,
для якого $M(b,a) = 0$;
**if** $b \in V'$:
*вилучити* $b$ *з* $V'$; $M(a,b) = 0$
**else:**
$M(a,b) = 0$
$P = P \cup \{(a,b)\}$
$k = k - 1$
$a = b$

*Опис послідовності дій в алгоритмі Террі*
1. Скласти матрицю суміжності M[i,j].
2. Задати початкову й кінцеву вершини vs і vf.
3. Задати дві змінні:
4. row – поточний рядок M[i,j];
5. column – поточний стовпець M[i,j].
6. row:= vs; column := 1.
7. Якщо (i,j) елемент матриці M[i,j] одиничний, то обнуляємо цей елемент і елемент (j,i) (якщо він одиничний).
8. Заносимо номер і-го рядка в стек. Переходимо в рядок з номером j.
9. Якщо (i,j) елемент нульовий, то переходимо до аналізу наступного стовпця і-го рядка.
10. Якщо виявилося, що всі елементи аналізованого рядка нульові, то дістаємо зі стека номер вершини й переходимо в рядок з цим номером.
11. Якщо ми переходимо з одного рядка в інший, то номер стовпця робимо одиничним.
12. Аналіз матриці закінчується, коли потрапляємо в рядок з номером vf.
13. Вміст стека – маршрут з *vs* у *vf*.

*Розв'язування задачі вручну*
Задамо граф.

На рис. 3.112 наведено граф для розгляду алгоритму Террі.

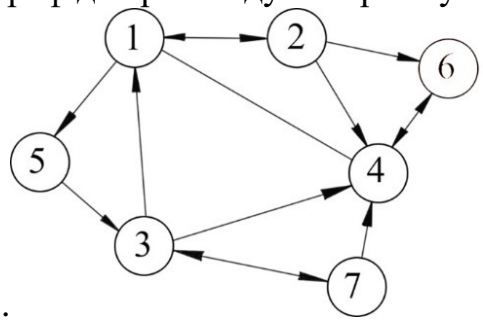
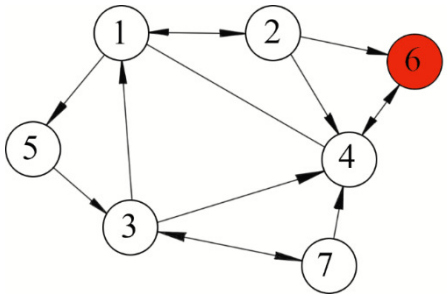



Рис. 3.112. Початковий граф

**ЗАВДАННЯ:** Знайти маршрут з початкової вершини 6 в кінцеву вершину 7. Уважно розглянувши граф, знайдемо відповідь: {6,4,1,5,3,7}.

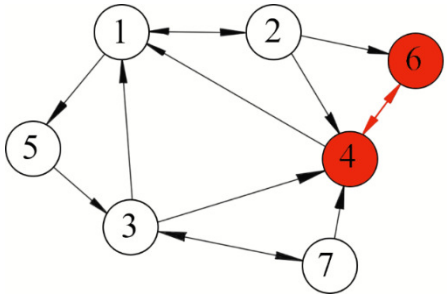
**РОЗВ'ЯЗОК: (1)**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | 1 | | | 1 | | |
| 2 | 1 | | | 1 | | 1 | |
| 3 | 1 | | | 1 | | | 1 |
| 4 | 1 | | | | | 1 | |
| 5 | | | 1 | | | | |
| 6 | | | | 1 | | | |
| 7 | | | 1 | 1 | | | |




1) Розглянемо перетин рядка 6 і стовпця 1.
1а) Оскільки елемент (6,1) нульовий, переходимо до елемента (6,2) і т. д. до знаходження одиничного елемента.
2) Знаходимо перший ненульовий елемент у рядку 6: (6,4).
3) Елемент (6,4) одиничний. Заповнюємо стек: {6}.
**Переписуємо матрицю** (2)

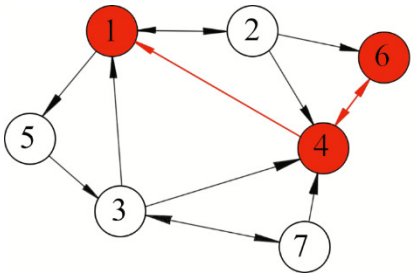
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | 1 | | | 1 | | |
| 2 | 1 | | | 1 | | 1 | |
| 3 | 1 | | | 1 | | | 1 |
| 4 | 1 | | | | | 0 | |
| 5 | | | 1 | | | | |
| 6 | | | | 0 | | | |
| 7 | | | 1 | 1 | | | |




1) Обнуляємо (6,4) і (4,6)
2) Переходимо до аналізу рядка 4 і стовпця 1.
Вміст стека: {6}.
3) Елемент (4,1) одиничний. Заповнюємо стек: {6,4}.

**Переписуємо матрицю (3)**

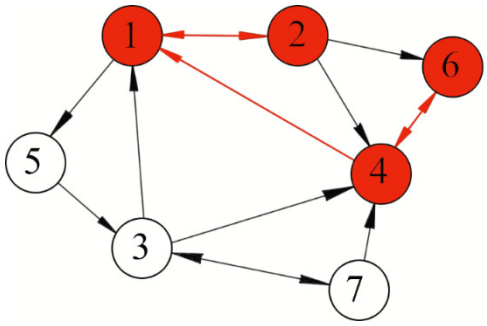
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | 1 | | | 1 | | |
| 2 | 1 | | | 1 | | 1 | |
| 3 | 1 | | | 1 | | | 1 |
| 4 | 0 | | | | | 0 | |
| 5 | | | 1 | | | | |
| 6 | | | | 0 | | | |
| 7 | | | 1 | 1 | | | |




1) Обнуляємо (4,1). Елемент (1,4) не обнуляємо, оскільки він нульовий.
2) Переходимо до аналізу рядка 1 і стовпця 1.
Вміст стека: {6,4}.
3) Елемент (1,1) нульовий, переходимо до елемента (1,2).
4) Елемент (1,2) одиничний.
Заповнюємо стек: {6,4,1}.

**Переписуємо матрицю (4)**

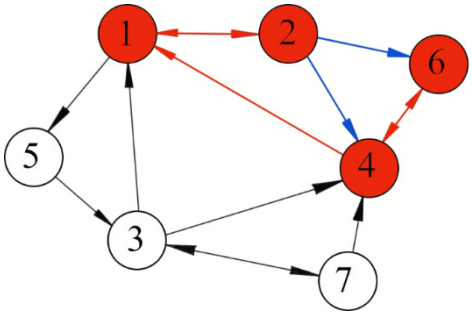
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | 0 | | | 1 | | |
| 2 | 0 | | | 1 | | 1 | |
| 3 | 1 | | | 1 | | | 1 |
| 4 | 0 | | | | | 0 | |
| 5 | | | 1 | | | | |
| 6 | | | | 0 | | | |
| 7 | | | 1 | 1 | | | |




1) Обнуляємо (1,2) і (2,1)
2) Переходимо до аналізу рядка 2 і стовпця 1.
Вміст стека: {6,4,1}.
3) Знаходимо перший ненульовий елемент у рядку 2: (2,4).
4) Елемент (2,4) одиничний. Заповнюємо стек: {6,4,1,2}.

**Переписуємо матрицю (5)**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | 0 | | | 1 | | |
| 2 | 0 | | | 0 | | 1 | |
| 3 | 1 | | | 1 | | | 1 |
| 4 | 0 | | | | | 0 | |
| 5 | | | 1 | | | | |
| 6 | | | | 0 | | | |
| 7 | | | 1 | 1 | | | |

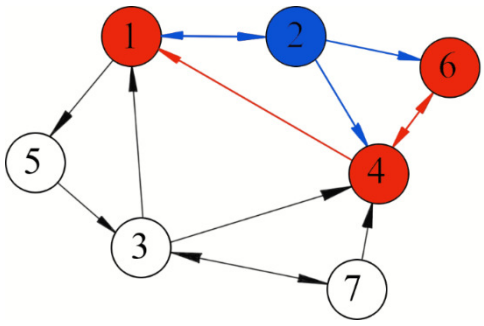



1) Обнуляємо (2,4).
2) Переходимо до аналізу рядка 4. Вміст стека: {6,4,1,2}.
Рядок 4 нульовий, виймаємо зі стека елемент. Вміст стека: {6,4,1}.
3) Переходимо до аналізу рядка 2 і стовпця 1.

4) Знаходимо перший ненульовий елемент у рядку 2: (2,6).
Елемент (2,6) одиничний. Заповнюємо стек: {6,4,1,2}.

**Переписуємо матрицю (6)**

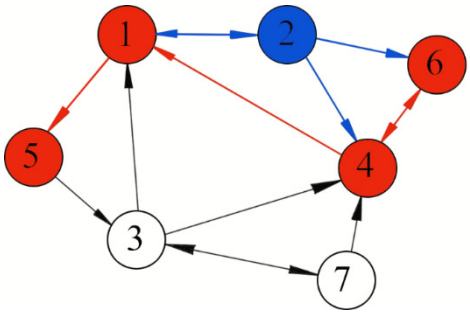
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | 0 | | | 1 | | |
| 2 | 0 | | | 0 | | 0 | |
| 3 | 1 | | | 1 | | | 1 |
| 4 | 0 | | | | | 0 | |
| 5 | | | 1 | | | | |
| 6 | | | | 0 | | | |
| 7 | | | 1 | 1 | | | |




1) Обнуляємо (2,6).
2) Переходимо до аналізу рядка 6. Вміст стека: {6,4,1,2}.
Рядок 6 нульовий, виймаємо зі стека елемент. Вміст стека: {6,4,1}.
3) Переходимо до аналізу рядка 2. Вміст стека: {6,4,1}.
Рядок 2 нульовий, виймаємо зі стека елемент 1. Вміст стека: {6,4}.
4) Переходимо до аналізу рядка 1 і стовпця 1. Вміст стека: {6,4}.
5) Знаходимо перший ненульовий елемент у рядку 1: (1,5).
6) Елемент (1,5) одиничний. Заповнюємо стек: {6,4,1}.

**Переписуємо матрицю (7)**

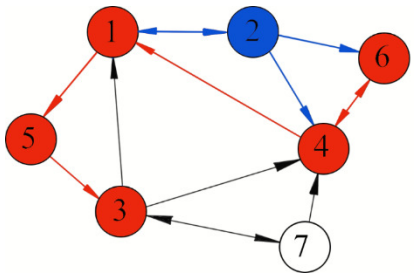
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | 0 | | | 0 | | |
| 2 | 0 | | | 0 | | 0 | |
| 3 | 1 | | | 1 | | | 1 |
| 4 | 0 | | | | | 0 | |
| 5 | | | 1 | | | | |
| 6 | | | | 0 | | | |
| 7 | | | 1 | 1 | | | |




1) Обнуляємо (1,5).
2) Переходимо до аналізу рядка 5 і стовпця 1.
Уміст стека: {6,4,1}.
5) Знаходимо перший ненульовий елемент у рядку 5: (5,3).
6) Елемент (5,3) одиничний. Заповнюємо стек: {6,4,1,5}.

**Переписуємо матрицю (8).**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | 0 | | | 0 | | |
| 2 | 0 | | | 0 | | 0 | |
| 3 | 1 | | | 1 | | | 1 |
| 4 | 0 | | | | | 0 | |
| 5 | | | 0 | | | | |
| 6 | | | | 0 | | | |
| 7 | | | 1 | 1 | | | |

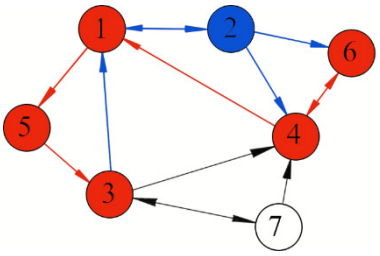



1) Обнуляємо (5,3).

2) Переходимо до аналізу рядка 3 і стовпця 1. Вміст стека: {6,4,1,5}.
3) Елемент (3,1) одиничний. Заповнюємо стек: {6,4,1,5,3}.

**Переписуємо матрицю (9).**

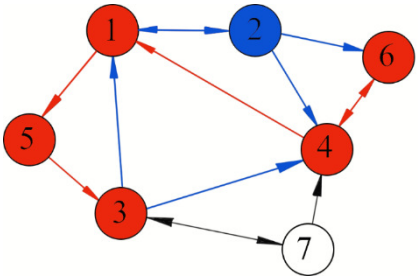
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | 0 | | | 0 | | |
| 2 | 0 | | | 0 | | 0 | |
| 3 | 0 | | | 1 | | | 1 |
| 4 | 0 | | | | | 0 | |
| 5 | | | 0 | | | | |
| 6 | | | | 0 | | | |
| 7 | | | 1 | 1 | | | |




1) Обнуляємо (3,1).
Переходимо до аналізу рядка 1. Вміст стека: {6,4,1,5,3}.
2) Рядок 1 нульовий, виймаємо елемент. Стек: {6,4,1,5}.
Переходимо до аналізу рядка 3 . Вміст стека: {6,4,1,5}.
3) Знаходимо перший ненульовий елемент у рядку 3: (3,4).
4) Елемент (3,4) одиничний. Заповнюємо стек: {6,4,1,5,3}.

**Переписуємо матрицю (10).**

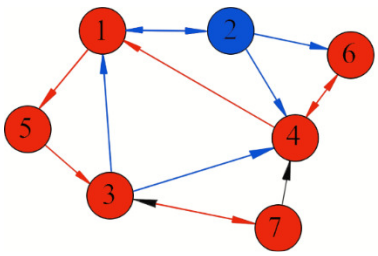
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | 0 | | | 0 | | |
| 2 | 0 | | | 0 | | 0 | |
| 3 | 0 | | | 0 | | | 1 |
| 4 | 0 | | | | | 0 | |
| 5 | | | 0 | | | | |
| 6 | | | | 0 | | | |
| 7 | | | 1 | 1 | | | |




1) Обнуляємо (3,4).
2) Переходимо до аналізу рядка 4. Стек: {6,4,1,5,3}.
Рядок 4 нульовий. Виймаємо елемент. Стек: {6,4,1,5}.
Переходимо до аналізу рядка 3. Стек: {6,4,1,5}.
3) Знаходимо перший ненульовий елемент у рядку 3: (3,7).
4) Елемент (3,7) одиничний. Заповнюємо стек: {6,4,1,5,3}.

*Результат роботи алгоритму Террі з пошуку найкоротшого шляху в незваженому графі*

Оскільки вершина 7 – кінцева за умовою, потрапляння в неї є ознакою закінчення роботи алгоритму Террі.
Отже, відповідь одержуємо шляхом занесення кінцевої вершини в стек: {6,4,1,5,3,7}.



### 3.7.2. Хвильовий алгоритм

Нехай $G = G(V, E)$ непустий граф. Потрібно знайти шлях між вершинами $s$ і $t$ графа ($s \neq t$), який містить мінімальну кількість проміжних вершин (ребер).

*Структура даних*

1. `Time[i]` – масив хвильових міток вершин графа $G$ .

Початкова установка:

```
for i in range(1,len(V)+1):Time[i]=-1
else Time[s]=0
```

2. `OldFront` – множина вершин старого фронту хвилі.

Початкова установка: `OldFront={s}`.

3. `NewFront` – множина вершин нового фронту хвилі.

Початкова установка: `NewFront={}`.

4. `T` – змінна поточного часу.

Початкова установка: `T=0` .

*Опис алгоритму*

Алгоритм полягає в наступному:

1. Для кожної з вершин, що входять в `OldFront`, переглядаємо суміжні вершини $u_i$)

2. `if Time[`$u_i$`]==-1:`

   `Time[`$u_i$`]=T+1`

   `NewFront=NewFront`$\cup\{u_i\}$;

3. Якщо `NewFront==`$\{\}$, то КІНЕЦЬ («немає розв'язку»);

4. Якщо `t`$\in$`NewFront` (тобто одна з вершин $u_i$ == `t` ), то знайдено найкоротший шлях між `s` і `t` з `Time[t]=T+1` проміжними ребрами; КІНЕЦЬ («розв'язок знайдений»).

Якщо жодна з вершин $u_i$ не співпадає з `t`, то `goto 5`

**5.** `OldFront=NewFront`

`NewFront={}; T=T+1; goto 1`

**Зауваження:** на кроці (1) «**сусідніми**» вершинами для **неорієнтованих графів** вважають всі **суміжні вершини**, а для **орграфів** – вершини, у які з даної вершини **ведуть дуги**.

*Відновлення найкоротшого шляху*

Якщо на кроці (4) була досягнута вершина `t`, то алгоритм побудови найкоротшого шляху такий:

1. Серед сусідів вершини `t` знайдемо будь-яку вершину з хвильовою міткою `Time[t]-1`.

2. Серед сусідів вже знайденої вершини знайдемо вершину з міткою `Time[t]-1`, і т. д., поки не досягнемо `s`.
Знайдена послідовність вершин визначає один з найкоротших шляхів з `s` у `t`. На практиці вигідно зберігати інформацію про те, з якої вершини «хвиля» прийшла у вершину $u_i$ – тоді відновлення шляху відбувається швидше.

На рис. 3.113 показана нумерація вершин, отримана в результаті роботи хвильового алгоритму.

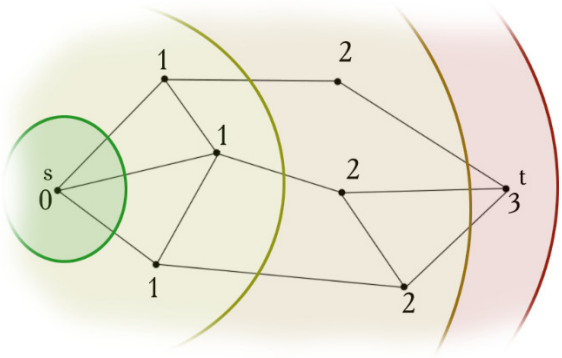



Рис. 3.113. Граф хвильового алгоритму

### 3.7.3. Пошук найкоротшого шляху у зваженому графі за алгоритмом Дейкстри

***Нехай дано граф*** $G$ з матрицею ваг $C=\left[c(i,j)\right]$.

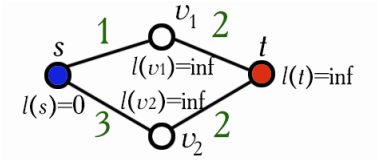
**Задача про найкоротший шлях** полягає в знаходженні найкоротшого шляху від заданої **початкової вершини** $s\in V$ до заданої **кінцевої вершини** $t\in V$ за умови, що такий шлях існує.
Алгоритм Дейкстри знаходить найкоротший шлях **для випадку** $c(i,j)$.

1. Вершинам приписують **тимчасові позначки**. Нехай $l(v_i)$ – позначка вершини $v_i$.
2. Кожна позначка вершини **дає верхню границю довжини шляху** від $s$ до цієї вершини.
3. Величини цих **позначок зменшують ітераційно**.
4. На кожному кроці ітерації одна з тимчасових позначок **стає постійною**.
5. Величина цієї позначки є точною довжиною найкоротшого шляху від $s$ до розглянутої вершини.

*Теоретичний опис алгоритму Дейкстри*

**Присвоєння початкових значень**

*Крок* 1. Встановимо $l(s)=0$ і вважатимемо цю позначку постійною.
Встановимо $l(v_i)=\infty$ для всіх $v_i\neq s$ і вважатимемо ці позначки тимчасовими.

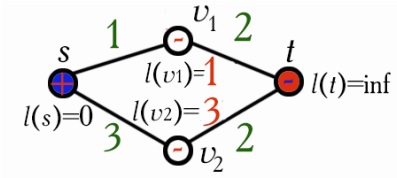



Встановимо $cur = s$.

**Відновлення позначок**

*Крок* 2. Для всіх $v_i \in \Gamma(cur)$, позначки яких тимчасові, змінимо позначки відповідно до такого виразу:

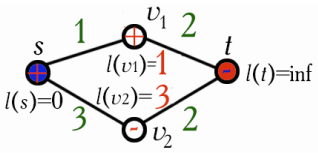
$$l(v_i) = \min\left[l(v_i), l(cur) + c(cur, v_i)\right]$$




**Перетворення позначки на постійну**

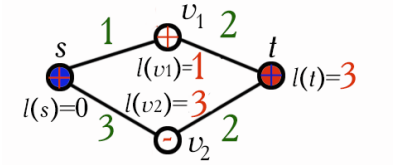
*Крок 3.* Серед вершин з тимчасовими позначками знайдемо таку, для якої

$$l\left(v_i^*\right) = \min\left(v_i\right),\ v_i \in \Gamma(cur)$$



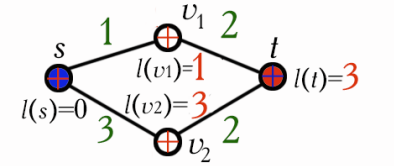

Будемо вважати позначку $l\left(v_i^*\right)$ постійною і встановлюємо $cur = v_i^*$.

*Крок 4.* **Якщо потрібно знайти шлях від** $s$ **до** $t$. Якщо $cur = t$, то $l(cur)$ є довжиною найкоротшого шляху від вершини $s$ до вершини $t$. Зупинка.




*Крок 5.* Якщо $cur \neq t$, перейти до кроку 2.

*Крок 6.* **Якщо потрібно знайти шляхи від** $s$ **до всіх інших вершин.** Якщо всі вершини позначені як постійні, то ці позначки дають довжини найкоротших шляхів. Зупинка.




*Крок 7.* Якщо деякі позначки залишаються тимчасовими, то перейти до кроку 2.

*Приклад ручного розв'язування задачі*

Розглянемо граф, зображений на рис. 3.114, у якому кожне неорієнтоване ребро вважаємо парою протилежно орієнтованих дуг рівної ваги. Ця вага виписана на рисунку, що зображує граф, поруч із ребром.

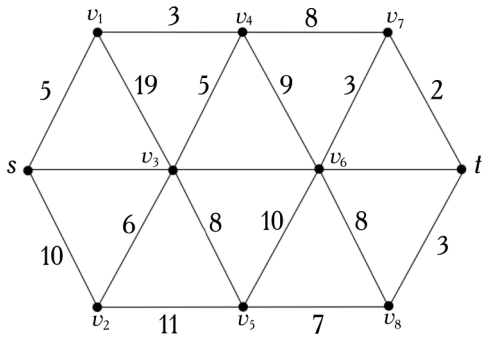



Рис. 3.114. Неорієнтований зважений граф

Потрібно знайти найкоротші шляхи від вершини $s$ до всіх інших вершин. Для цього використовуємо алгоритм Дейкстри. Постійні позначки позначаються знаком +, інші позначки є тимчасовими.

**Схема застосування алгоритму**

*Крок 1.* $l(s) = 0^+, l(v_i) = \infty,\ i = 1,\ldots,8,\ p = s.$

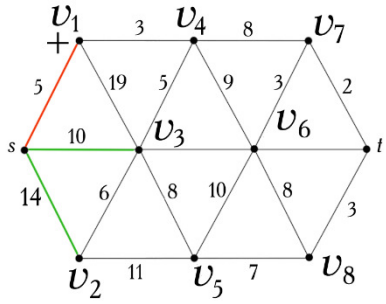
***Перша ітерація***

*Крок 2.* $\Gamma(s) = \{v_1, v_2, v_3\}$ – усі позначки тимчасові.

$$l(v_1) = \min[\infty, 0^+ + 5] = 5,$$

$$l(v_2) = \min[\infty, 0^+ + 14] = 14,$$

$$l(v_3) = \min[\infty, 0^+ + 10] = 10.$$




*Крок 3.* $l(v_1) = \min\limits_{i=1,2,3} l(v_i) = 5.$

*Крок 4.* $l(v_1) = 5^+ - v_1$ одержує постійну позначку; $p = v_1$.

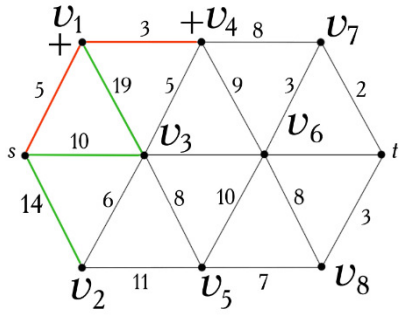
*Крок 5.* Не всі вершини мають постійні позначки, тому переходимо до кроку 2.

***Друга ітерація***

*Крок 2.* $\Gamma(p) = \Gamma(v_1) = \{s, v_3, v_4\}$. Вершина $s$ має постійну позначку,

$$l(v_3) = \min[10, 5^+ + 19] = 10,$$

$$l(v_4) = \min[\infty, 5^+ + 3] = 8.$$

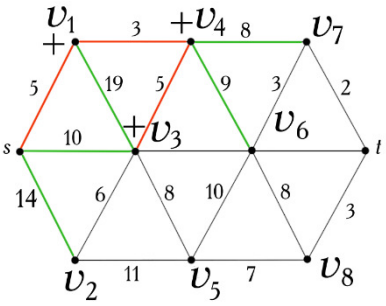



*Крок 3.* $l(v_4) = \min\limits_{i=3,4} l(v_i).$

*Крок 4.* Вершина $v_4$ одержує постійну мітку: $l(v_4) = 8^+;\ p = v_4$.

*Крок 5.* Не всі вершини мають постійні позначки, тому переходимо до кроку 2.

***Третя ітерація***




*Крок 2.* $\Gamma(p) = \Gamma(v_4) = \{v_1, v_3, v_6, v_7\}$.

Вершина $v_1$ має постійну позначку;

$$l(v_3) = \min\left[10, 8^+ + 5\right] = 10,$$

$$l(v_7) = \min\left[\infty, 8^+ + 8\right] = 16,$$

$$l(v_6) = \min\left[\infty, 8^+ + 9\right] = 17.$$

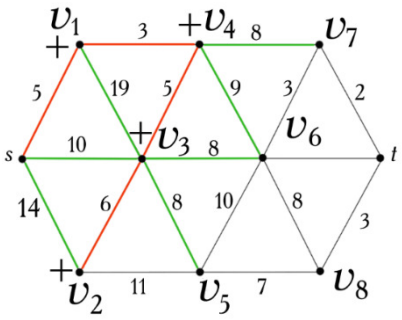
*Крок 3.* $l(v_3) = \min\limits_{i=3,6,7} l(v_i) = 10$.

*Крок 4.* Вершина $v_3$ одержує постійну мітку: $l(v_3) = 10^+$; $p = v_3$.

*Крок 5.* Не всі вершини мають постійні позначки, тому переходимо до кроку 2.

***Четверта ітерація***

*Крок 2.* $\Gamma(p) = \Gamma(v_3) = \{s, v_1, v_2, v_4, v_5, v_6\}$. Вершини $s, v_1, v_4$ мають постійні позначки,




$$l(v_2) = \min\left[14, 10^+ + 6\right] = 14,$$

$$l(v_5) = \min\left[\infty, 10^+ + 8\right] = 18,$$

$$l(v_6) = \min\left[17, 10^+ + 8\right] = 17.$$

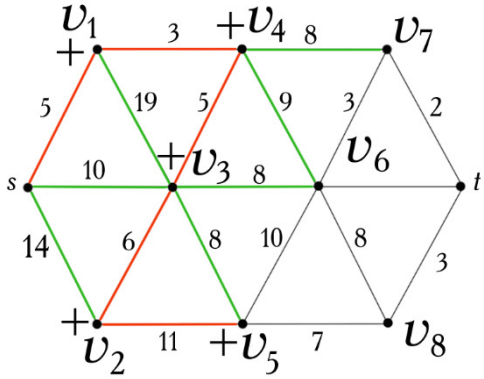
*Крок 3.* $l(v_2) = \min\limits_{i=2,5,6} l(v_i) = 14$.

*Крок 4.* Вершина $v_2$ одержує постійну мітку: $l(v_2) = 14^+$; $p = v_2$.

*Крок 5.* Не всі вершини мають постійні позначки, тому переходимо до кроку 2.

***П'ята ітерація***

*Крок 2.* $\Gamma(p) = \Gamma(v_2) = \{s, v_3, v_5\}$. Вершини $s, v_3$ мають постійні позначки;




$$l(v_5) = \min\left[18, 14^+ + 11\right] = 18.$$

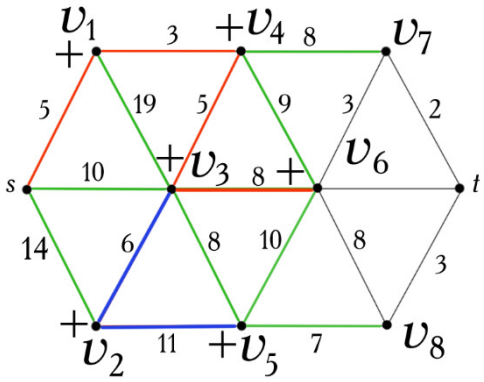
*Крок 3.* $l(v_5) = \min\limits_{i=5} l(v_i) = 18$.

*Крок 4.* Вершина $v_5$ одержує постійну мітку: $l(v_5) = 18^+$; $p = v_5$.

*Крок 5.* Не всі вершини мають постійні позначки, тому переходимо до кроку 2.

***Шоста ітерація***

*Крок 2.* $\Gamma(p)=\Gamma(v_5)=\{v_2,v_3,v_6,v_8\}$.




Вершини $v_2, v_3$ мають постійні позначки,

$l(v_6)=\min\left[17,18^{+}+10\right]=17,$

$l(v_8)=\min\left[\infty,18^{+}+7\right]=25.$

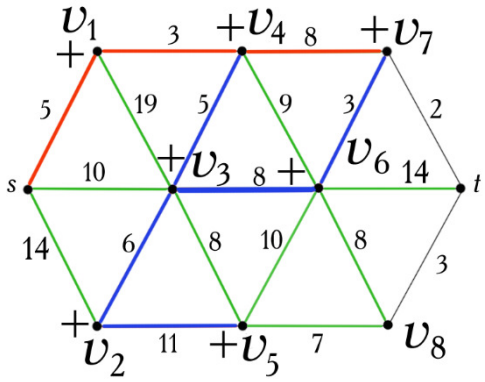
*Крок 3.* $l(v_6)=\min\limits_{i=6,8} l(v_i)=17$.

*Крок 4.* Вершина $v_6$ одержує постійну мітку: $l(v_6)=17^{+};\ p=v_6$.

*Крок 5.* Не всі вершини мають постійні позначки, тому переходимо до кроку 2.

***Сьома ітерація***

*Крок 2.* $\Gamma(p)=\Gamma(v_6)=\{v_3,v_4,v_4,v_7,v_8,t\}$. Вершини $v_3, v_4, v_5$ мають постійні позначки,




$l(v_7)=\min\left[16,17^{+}+3\right]=16,$

$l(v_6)=\min\left[25,17^{+}+8\right]=25,$

$l(t)=\min\left[\infty,17^{+}+14\right]=31.$

*Крок 3.* $l(v_7)=\min\limits_{i=7,8,t} l(v_i)=16$.

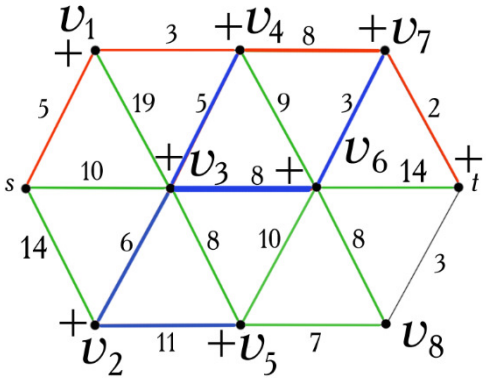
*Крок 4.* Вершина $v_7$ одержує постійну мітку: $l(v_7)=16^{+};\ p=v_7$.

*Крок 5.* Не всі вершини мають постійні позначки, тому переходимо до кроку 2.

***Восьма ітерація***

*Крок 2.* $\Gamma(p)=\Gamma(v_7)=\{v_4,v_6,t\}$. Вершини $v_4, v_6$ мають постійні позначки;

$l(t)=\min\left[31,16^{+}+2\right]=18.$



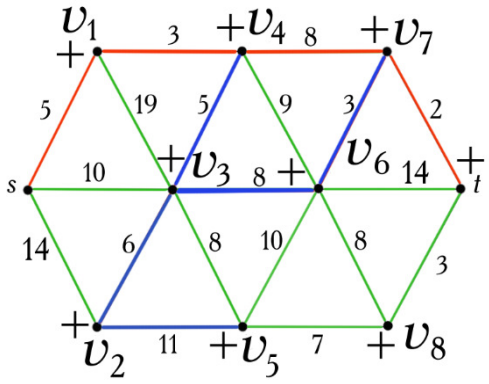

*Крок 3.* $l(v_t)=\min\limits_{i=t} l(v_i)=18$.

*Крок 4.* Вершина $t$ одержує постійну мітку: $l(t)=18^{+};\ p=t$.

*Крок 5.* Не всі вершини мають постійні позначки, тому переходимо до кроку 2.

***Дев'ята ітерація***

*Крок 2.* $\Gamma(p) = \Gamma(t) = \{v_6, v_7, v_8\}$.




Вершини $v_6, v_7$ мають постійні позначки;

$l(v_8) = \min\left[25, 18^+ + 3\right] = 21.$

*Крок 3.* $l(v_8) = \min\limits_{i=8} l(v_i) = 21.$

*Крок 4.* Вершина $v_8$ одержує постійну мітку: $l(v_8) = 21^+$; $p = v_8$.

*Крок 5*. Позначки всіх вершин постійні. Зупинка.

**Зауваження**

Як тільки всі позначки вершин графа стають постійними, тобто знайдені довжини найкоротших шляхів від вершини *s* до будь-якої іншої вершини, самі шляхи можна одержати, використовуючи співвідношення

$$l(v_i') + c(v_i', v_i) = l(v_i).$$

Тут $c(v_i', v_i)$ – вага дуги, що з'єднує вершину $v_i'$ з вершиною $v_i$. Це співвідношення дає можливість для кожної вершини $v_i$ знайти попередню до неї вершину $v_i'$ й відновити весь шлях від *s* до $v_i$.

У розглянутому прикладі найкоротший шлях від вершини *s* до вершини *t* можна відновити із співвідношень, отриманих в 7-й, 2-й та 1-й ітераціях.

$$l(t) = l(v_6) + c(v_6, t),$$
$$l(v_6) = l(v_4) + c(v_4, v_7),$$
$$l(v_4) = l(v_1) + c(v_1, v_4),$$
$$l(v_1) = l(s) + c(s, v_1).$$

На рис. 3.115 показано граф з відміченими вершинами, в яких вказано відстань від початкової вершини $s$ з позначкою 0+.

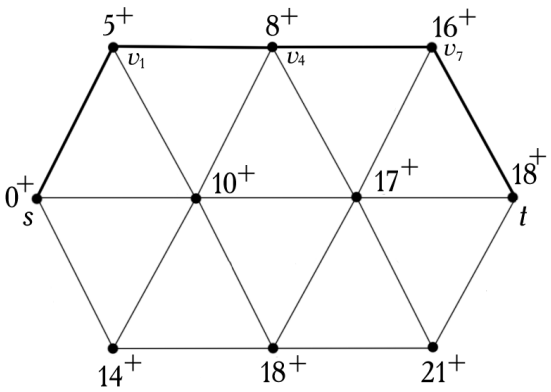



Рис. 3.115. Результуючий граф алгоритму Дейкстри

Найкоротшому шляху відповідає послідовність вершин $s, v_1, v_4, v_7, t$.

*Машинний алгоритм Дейкстри*

**Структура даних**

1. Логічний масив $visited[n]$ : $False$ – вершина не розглянута; $True$– вершина розглянута.

2. Масив $nlen[n]$ містить поточні найкоротші відстані від початкової до відповідної вершини.

3. Масив $path[n]$ містить номери вершин.

$path[k]$ містить номер передостанньої вершини на поточному найкоротшому шляху з початкової вершини у вершину $k$.

4. $matrix[i, j]$ матриця відстаней; $1 \le i \le n, 1 \le j \le n$.

*Кроки алгоритму Дейкстри:*

1. **Ініціалізація.**

Очистимо масив $visited$ для всіх вершин графа:

```
for i in range(1,n+1): visited[i]=False.
```

Виберемо стартову вершину `s`.

Заповнимо масив шляху стартовою вершиною `s`.

```
for i in range(1,n+1): path[i]=s.
```

Перепишемо рядок стартової вершини з матриці відстаней у масив `nlen`

```
for i in range(1,n+1): nlen[i]=matrix[s,i].
```

Виконаємо початкову установку для стартової вершини `s`.

```
visited[d]=True; path[s]=0.
```

2. **Загальний крок**

Виконуємо перевірку на те, чи залишилися невідмічені вершини.

```
def possible():
   r=True
   for i in range(1,n+1):
      if not visited[i]: exit
      r=False
   return=r
```

Знайдемо серед невідмічених ту вершину, що має мінімальну відстань від поточної вершини.

```
def min():
  minvalue=infinity
```

```
  for i in range(1,n+1):
       if not visited[i]:
           if nlen[i]<minvalue:
                 currentmin=i
                 minvalue=nlen[i]
  min=currentmin
```

Знайдену вершину $k$ з мінімальною відстанню від поточної позначаємо як позначену:

`visited[k]=True` (Позначка стає постійною)

Потім модифікуємо списки `nlen` й `path` з метою визначення суміжних вершин і відстані до них від стартової вершини.

```
for i in range(1,n+1):
    if nlen[i]>nlen[k]+mattr[i, k]:
        nlen[i]=nlen[k]+mattr[i, k]
        path[i]=k
```

3. **Завершальні дії**

Якщо всі елементи списку `visited` дорівнюють `True`, тобто всі вершини графа позначені, то довжина шляху від $v_i$ до $v_k$ дорівнює `nlen[k]`.

Вивід вершин, що входять у шлях.

```
input('Кінцева вершина: ', finish)
print(finish)
k=finish
finish=path[pinish]
while finish!=start:
  print(' <-  ', finish)
  finish=path[finish]
print(' <-  ', start)
print('Довжина шляху = ', nlen[k])
```

### 3.7.4. Алгоритм Форда – Беллмана знаходження мінімального шляху

Передбачається, що орієнтований граф не містить контурів від'ємної довжини.

Основними величинами, які потрібно обчислювати в цьому алгоритмі, є величини $\lambda_i(k)$, де $i = 1,2,\ldots,n$ ($n$ – число вершин графа); $k = 1,2,\ldots,n-1$.

Для фіксованих $i$ і $k$ величина $\lambda_i(k)$ дорівнює довжині мінімального шляху, що веде із заданої початкової вершини $v_1$ у вершину $v_i$ і складається з не більше ніж $k$ дуг.

*Крок 1.* Установка початкових умов. Ввести кількість вершин графа $n$ і матрицю ваг $C = \left| c_{ij} \right|$.

*Крок 2. Встановити* $k = 0$. Встановити $\lambda_i \left( 0 \right) = \infty$ для всіх вершин, крім $v_1$; встановити $\lambda_1 \left( 0 \right) = 0$.

*Крок 3.* У циклі по $k$, $k = 1, 2, \ldots, n-1$, кожній вершині $v_i$ на $k$ -му кроці приписати індекс $\lambda_i \left( k \right)$ за наступним правилом:

$$\lambda_i \left( k \right) = \min_{1 \le j \le n} \left\{ \lambda_j \left( k-1 \right) + c_{ji} \right\} \tag{1}$$

для всіх вершин, крім $v_1$, встановити $\lambda_1 \left( k \right) = 0$.

У результаті роботи алгоритму формується таблиця індексів

$$\lambda_i \left( k \right),\ i = 1,2,\ldots,n;\ k = 0,1,2,\ldots,n-1.$$

При цьому $\lambda_i \left( k \right)$ визначає довжину мінімального шляху з першої вершини у вершину $i$, що містить не більше ніж $k$ дуг.

*Крок 4.* Відновлення мінімального шляху. Для будь-якої вершини $v_s$ попередня до неї вершина $v_r$ визначається зі співвідношення:

$$\lambda_r \left( n-2 \right) + c_{rs} = \lambda_s \left( n-1 \right),\ v_r \in G^{-1} \left( v_s \right), \tag{2}$$

де $G^{-1} \left( v_s \right)$ – прообраз вершини $v_s$.

Для знайденої вершини $v_r$ попередня до неї вершина $v_q$ визначається із співвідношення:

$$\lambda_q \left( n-3 \right) + c_{qr} = \lambda_r \left( n-2 \right),\ v_q \in G^{-1} \left( v_r \right),$$

де $G^{-1} \left( v_r \right)$ – прообраз вершини $v_r$, і т. д.

Послідовно застосовуючи це співвідношення, починаючи від останньої вершини $v_i$, знайдемо мінімальний шлях.

*Макроструктура алгоритму*

Алгоритм послідовно уточнює значення функції $d\left( cur \right)$.

1. На початку задаємо значення

$d\left( s \right) = 0;$

$d\left( cur \right) = \infty \ \ \forall cur \ne s$.

2. Виконуємо $n-1$ ітерацій, під час яких виконуємо релаксацію всіх ребер графа.

*Вхідні дані:*

Граф: вершини $V$, ребра $\left( i, j \right) \in E$ з вагами $W\left( i, j \right)$

Початкова вершина $s$.

*Вихідні дані:* відстані $d(cur)$ до кожної вершини $cur \in V$ від вершини $s$.

*Код алгоритму Форда – Беллмана*

```
start=1
ML = 10 ** 9
d = [ML] * N
d[start] = 0
for k in range(1, N):
    for i in range(N):
        for j in range(N):
            if d[j] + W[j][i] < d[i]:
                d[i] = d[j] + W[j][i]
```

### 3.7.5. Алгоритм Флойда – Воршелла

Метод Флойда безпосередньо ґрунтується на тому факті, що в графі з додатними вагами ребер будь-який неелементарний (довжиною більше 1 ребра) найкоротший шлях складається з інших найкоротших шляхів.

Цей алгоритм більш загальний у порівнянні з алгоритмом Дейкстри, оскільки він знаходить найкоротші шляхи між будь-якими двома вершинами графа.

В алгоритмі Флойда використовується матриця $A$ розміром $n \times n$, у якій обчислюються довжини найкоротших шляхів. Елемент $A[i, j]$ дорівнює відстані від вершини $i$ до вершини $j$, яка має скінченне значення, якщо існує ребро $(i, j)$, і дорівнює нескінченності в протилежному випадку.

*Алгоритм Флойда*

Основна ідея алгоритму.

Нехай є три вершини $i, j, k$ і задані відстані між ними.

Якщо виконується нерівність $A[i,k] + A[k,j] < A[i,j]$, то доцільно замінити шлях $i \to j$ шляхом $i \to k \to j$.

Така заміна виконується систематично в процесі виконання даного алгоритму.

Дано зважений граф $G(V,E)$ з вершинами, що пронумеровані від 1 до $n$.

Вага ребра $w_{uv} = \begin{cases} k, (u,v) \in E, \\ \infty, (u,v) \notin E. \end{cases}$

Знаходимо матрицю найкоротших відстаней $D$, в якій елемент $d_{ij}$ або дорівнює найкоротшій відстані від вершини $i$ до вершини $j$, або дорівнює $\infty$, якщо вершина $j$ не досяжна з $i$.

На кожному кроці беремо чергову вершину (нехай з номером $i$) і для всіх пар вершин $u$ та $v$ будемо обчислювати $d_{uv}^{(i)} = \min\left(d_{uv}^{(i-1)}, d_{ui}^{(i-1)} + d_{iv}^{(i-1)}\right)$

*Псевдокод*

$d_{uv}^{(0)} = w$
*for* $i$ *in* $V$:
  *for* $u$ *in* $V$:
    *for* $v$ *in* $V$:
      $d_{uv}^{(i)} = \min\left(d_{uv}^{(i-1)}, d_{ui}^{(i-1)} + d_{iv}^{i-1}\right)$

У підсумку отримуємо, що матриця $D^{(n)}$ і є шуканою матрицею найкоротших шляхів, оскільки містить довжини найкоротших шляхів між усіма парами вершин, що мають вершини з множини $\{1,...,n\}$ як проміжні, що є множиною всіх вершин графа.

На кожній ітерації перебирають всі пари вершин і шлях між ними скорочується за допомогою проміжної $i$-ї вершини.

**Приклад 3.45.** Розглянемо роботу алгоритму Флойда на прикладі графа $G$ (рис. 3.116), що складається з чотирьох вершин.

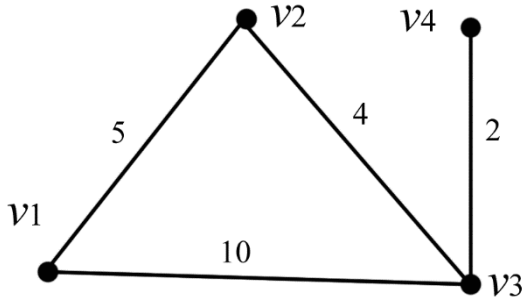



Рис. 3.116. Граф $G$

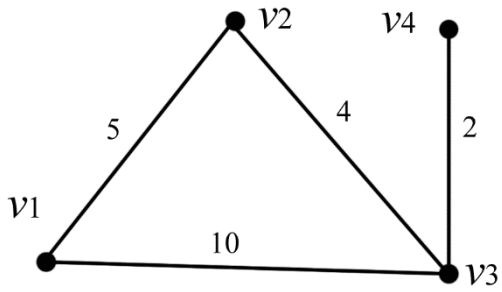
1. Будуємо початкову матрицю $T_0$:

$$T_0 = \begin{pmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 5 & 10 & \infty \\ v_2 & 5 & 0 & 4 & \infty \\ v_3 & 10 & 4 & 0 & 2 \\ v_4 & \infty & \infty & 2 & 0 \end{pmatrix}$$

2. Модифікуємо матрицю $T_0$ при $k=1$

```
for i in range(1,5):
 for j in range(1,5):
  T[i,j] = min(T[i,j], T[i,1] + T[1,j])
```

Замінюємо елемент матриці, якщо шлях між вершинами $i$ та $j$ виявиться коротшим через вершину $k=1$.



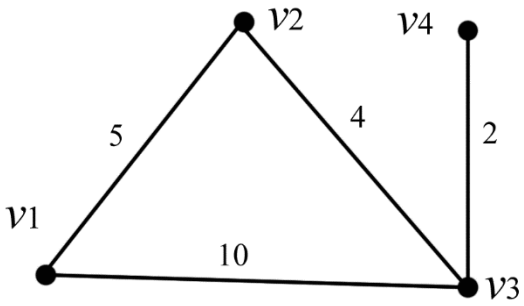

На першому етапі в матриці немає змін, оскільки не знайдено більш коротких шляхів між вершинами через вершину 1.

2. Модифікуємо матрицю $T_1$ при $k=2$

```
for i in range(1,5):
 for j in range(1,5):
  T[i,j] = min(T[i,j], T[i,2] + T[2,j])
```

Заміняємо елемент матриці, якщо шлях між вершинами $i$ та $j$ виявиться коротшим через вершину $k=2$

$$T_2 = \begin{pmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 5 & 9 & \infty \\ v_2 & 5 & 0 & 4 & \infty \\ v_3 & 9 & 4 & 0 & 2 \\ v_4 & \infty & \infty & 2 & 0 \end{pmatrix}$$

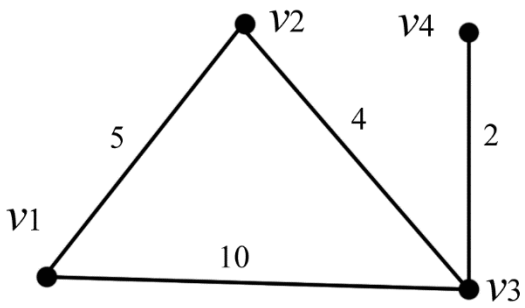



На другому етапі шлях з вершини 1 у вершину 3 виявився коротшим через вершину 2.

3. Модифікуємо матрицю $T_2$ при $k=3$

```
for i in range(1,5):
 for j in range(1,5):
  T[i,j] = min(T[i,j], T[i,3] + T[3,j])
```

Заміняємо елемент матриці, якщо шлях між вершинами $i$ та $j$ виявиться коротшим через вершину $k=3$.




$$T_3 = \begin{pmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 5 & 9 & 12 \\ v_2 & 5 & 0 & 4 & 6 \\ v_3 & 9 & 4 & 0 & 2 \\ v_4 & 12 & 6 & 2 & 0 \end{pmatrix}$$

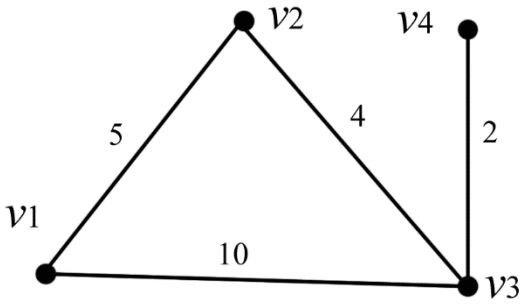
На 3-му етапі встановили шлях у вершину 4 через вершину 3.

4. Модифікуємо матрицю $T_2$ при $k=4$

```
for i in range(1,5):
 for j in range(1,5):
  T[i,j] = min(T[i,j], T[i,4] + T[4,j])
```

Заміняємо елемент матриці, якщо шлях між вершинами $i$ та $j$ виявиться коротшим через вершину $k=4$

$$T_4 = \begin{pmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 5 & 9 & 12 \\ v_2 & 5 & 0 & 4 & 6 \\ v_3 & 9 & 4 & 0 & 2 \\ v_4 & 12 & 6 & 2 & 0 \end{pmatrix}$$




На 4-му етапі не знайшли оптимальних шляхів через вершину 4, оскільки вона висяча.

**Контрольні запитання**

1. Покроково описати алгоритм пошуку шляху з вершини $s$ до вершини $t$ за алгоритмом Террі для графа, зображеного на рис. 3.117.

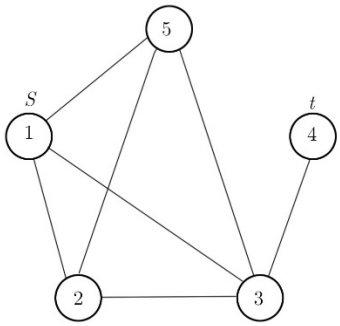



Рис. 3.117. Граф до задачі 1

2. Визначити мінімальний шлях з вершини $v_1$ до вершини $v_6$ за допомогою хвильового алгоритму у орграфі $G$, який задано його матрицею суміжності:

$$\begin{pmatrix} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ v_1 & 0 & 0 & 0 & 1 & 1 & 0 \\ v_2 & 1 & 0 & 0 & 0 & 0 & 1 \\ v_3 & 0 & 1 & 0 & 0 & 0 & 1 \\ v_4 & 0 & 1 & 0 & 0 & 1 & 0 \\ v_5 & 1 & 0 & 1 & 0 & 0 & 0 \\ v_6 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

3. Знайти найкоротший шлях з вершини $v_1$ до вершини $v_5$ з використанням алгоритму Дейкстри для орграфа $G$, який задано ваговою матрицею.

$$\begin{pmatrix} & v_1 & v_2 & v_3 & v_4 & v_5 \\ v_1 & 0 & 5 & 0 & 0 & 0 \\ v_2 & 0 & 0 & 0 & 4 & 2 \\ v_3 & 0 & 0 & 0 & 0 & 1 \\ v_4 & 0 & 0 & 2 & 0 & 0 \\ v_5 & 3 & 0 & 0 & 0 & 0 \end{pmatrix}$$

4. Визначити мінімальний шлях з вершини $v_1$ до вершини $v_6$ за допомогою алгоритму Форда – Беллмана у орграфі $G$, який показаний на рис. 3.118.

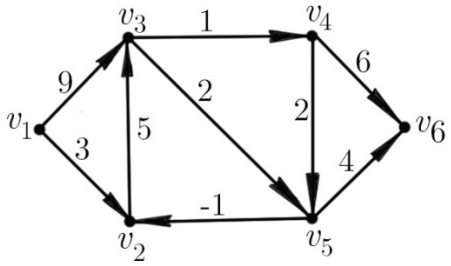



Рис. 3.118. Граф до задачі 4

5. Визначити найкоротший шлях між усіма вершинами за допомогою алгоритму Флойда – Воршелла у графі, заданому на рис. 3.119.

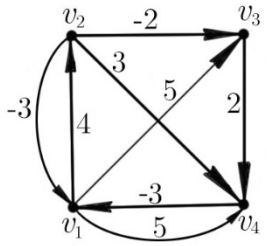



Рис. 3.119. Граф до задачі 5

## 3.8. Розфарбування графа

### 3.8.1. Задачі розфарбування

Задачі розфарбування вершин або ребер графа займають важливе місце в теорії графів.

До задачі розфарбування графа зводиться цілий ряд практичних задач. Одна з областей – складання розкладів:

- розкладу для освітніх закладів;
- розкладу в спорті;

- планування зустрічей, зборів, інтерв'ю;
- розкладу транспорту, у тому числі – авіатранспорту;
- розкладу для комунальних служб;
- інші.

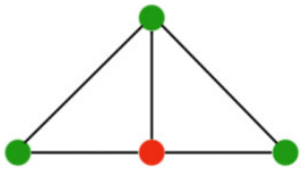
### 3.8.2. Основні визначення

Нехай $G=(V,E)$ – скінченний граф, а $k$ – деяке натуральне число.

*Вершинне розфарбування*

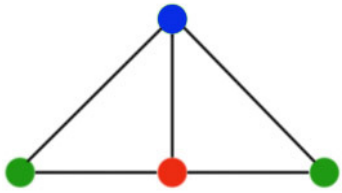
Довільну функцію виду

$f:V\to N_k$, де $N_k=\{1,2,...,k\}$, називають вершинним $k$-розфарбуванням, або просто $k$-розфарбуванням графа $G$.




*Правильне розфарбування*

Розфарбування називають правильним, якщо кольори суміжних вершин не співпадають, тобто $\forall\,(u,v)\in E \Rightarrow f(u)\neq f(v)$.




*Розфарбований граф*. Граф, для якого існує правильне $k$-розфарбування, називають розфарбованим графом.

*Базовий принцип оптимізації розфарбування*

- Якщо функція $f$ не взаємно однозначна, то при $|V|=k$ фактично може бути використано менше, ніж $k$ кольорів.
- Правильне розфарбування – це розбиття множини вершин.

  Правильне $k$-розфарбування можна розглядати як розбиття множини вершин $V$ графа $G$ на класи
  - $V_1\cup V_2\cup...\cup V_l=V$, де $l\le k$,
  - $V_i\neq\varnothing,\quad i=1,2,...,l$.

Кожний клас $V_i$ – це незалежна множина.

Такі класи називають *кольоровими класами.*

### 3.8.3. Хроматичне число

**Визначення.** Мінімальне число $k$, при якому існує правильне $k$-розфарбування графа $G$, називають *хроматичним числом* цього графа і позначають $X_p(G)$.

**Визначення.** Якщо $X_p(G)=k$, то граф $G$ називають $k$-*хроматичним.* Тобто його вершини можна розфарбувати $k$ різними кольорами так, що у будь-якого ребра інцидентні вершини матимуть різний колір.

**Визначення.** Правильне $k$-розфарбування графа $G$ при $k=X_p(G)$ називають *мінімальним.*

**Визначення**. Хроматичне число незв'язного графа дорівнює максимальному з хроматичних чисел його компонент зв'язності.

**Приклад 3.46.** Розглянемо граф $G$, зображений на рис. 3.120, на якому показано одне із правильних $k$-розфарбувань. Натуральними числами 1,2,3,4 позначені кольори відповідних вершин.

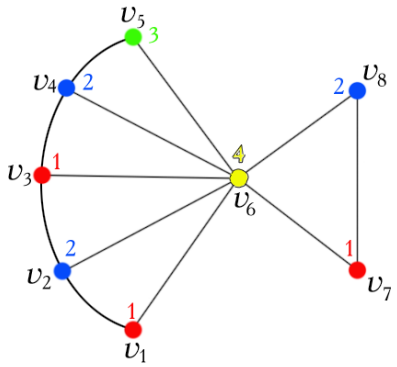



Рис. 3.120. Розфарбований граф $G$

*Хроматичні числа деяких графів*

Для деяких простих графів неважко знайти хроматичні числа.

1. Повний граф $K_n$, що складається з $n$ вершин, має хроматичне число $X_p\left(K_n\right) = n$

**Приклад 3.47.** Знайти хроматичні числа правильних графів $K_2, K_3, K_4, K_5$

*Розв'язок.*

Для знаходження хроматичних чисел необхідно виконати мінімальне правильне розфарбування відповідних графів, як показано на рис. 3.121.

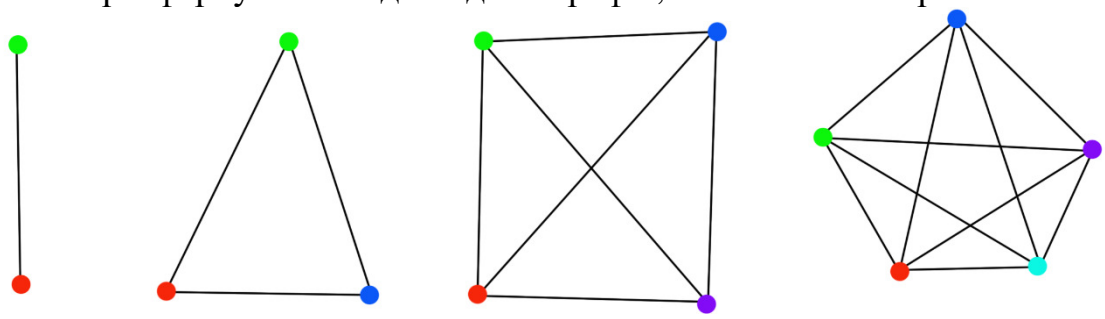


Рис. 3.121. Мінімальне правильне розфарбування повних графів

2. Повний граф $K_n - e$, який складається з $n$ вершин з одним відсутнім ребром, має хроматичне число $X_p\left(K_n - e\right) = n - 1$

**Приклад 3.48.** Знайти хроматичні числа графів $K_3 - 1, K_4 - 1, K_5 - 1$.

*Розв'язок.*

Для знаходження хроматичних чисел необхідно виконати мінімальне правильне розфарбування графів, як показано на рис. 3.122.

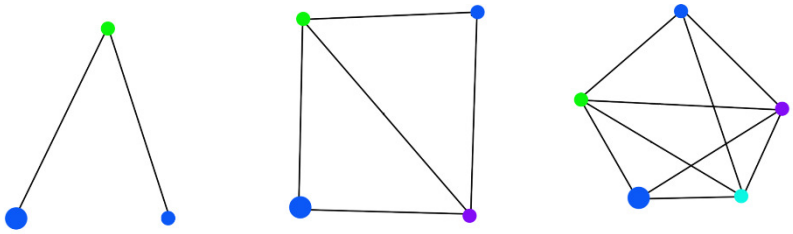


Рис. 3.122. Мінімальне правильне розфарбування графів

3. Повні дводольні графи $K_{m,n}$, що складаються з долей $|A| = m$ і $|B| = n$, мають хроматичне число $X_p\left(K_{m,n}\right) = 2$.

**Приклад 3.49.** Знайти хроматичні числа графів $K_{1,2}, K_{2,2}, K_{2,3}, K_{3,3}$.

*Розв'язок.*

Для знаходження хроматичних чисел необхідно виконати мінімальне правильне розфарбування біхроматичних графів, як показано на рис. 3.123.

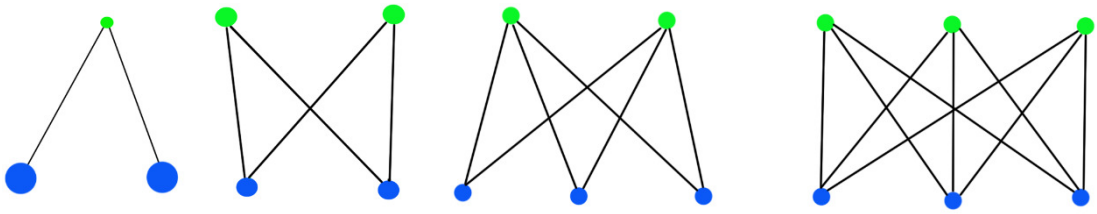


Рис. 3.123. Мінімальне правильне розфарбування біхроматичних графів

**Теорема 3.17.** Непустий граф є **біхроматичним** тоді й тільки тоді, коли він не має циклів непарної довжини.

**Приклад 3.50.** Навести приклади 1-хроматичного, 2-хроматичного і 3-хроматичного графів.

*Розв'язок.*

1-*хроматичний граф* – порожній граф, показаний на рис. 3.124.

Рис. 3.124. 1-хроматичний граф

2-*хроматичний граф* – дводольний непустий граф, показаний на рис. 3.125. 2-хроматичні графи, як правило, називають *біхроматичними.*

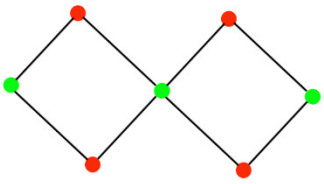


Рис. 3.125. 2-хроматичний граф

3-*хроматичний граф* – циклічний граф з непарним числом вершин у кожному з циклів, показаний на рис. 3.126.

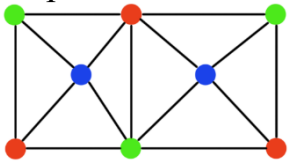


Рис. 3.126. 3-хроматичний граф

Визначення 1. Якщо граф має $n$ вершин, то його хроматичне число не перевищує $n$.

Визначення 2. Якщо граф має підграф $K_m$, то його хроматичне число не менше, ніж $m$.

### 3.8.4. Хроматичне число й стандартні характеристики

У загальному випадку хроматичне число графа не можна обчислити, знаючи тільки його стандартні числові характеристики: *число вершин, ребер, компонент зв'язності, розподіл степенів вершин.*

Розглянемо графи $G_1$ й $G_2$, показані на рис. 3.127.

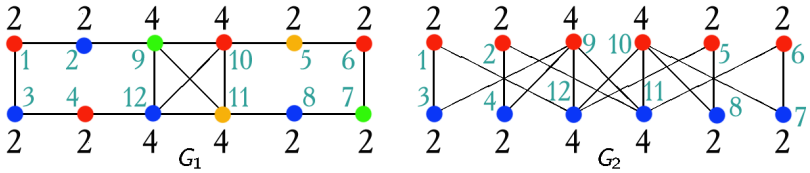



Рис. 3.127. Графи $G_1$ і $G_2$

Кожний з них має 12 вершин, у тому числі 4 вершини зі степенем 4 і 8 вершин зі степенем 2, 16 ребер, 1 компонент зв'язності. Але, як видно з рисунка, $X_p(G_1)=4$, а $X_p(G_2)=2$.

Оскільки $G_1$ містить $K_4$ підграф, *то* $X_p(G_1)=4$

Оскільки граф $G_2$ – дводольний, маємо $X_p(G_2)=2$.

Тому надалі вестимемо мову про оцінки, а не про точні значення хроматичного числа.

### 3.8.5. Хроматичне число й щільність графа. Три нижні оцінки хроматичного числа

Під **нижніми оцінками** хроматичного числа будемо розуміти нерівності виду $X(G)\geq c$, де $c$ – деяка константа, що обчислюється на графі $G$.

**Верхня оцінка** хроматичного числа – це нерівності виду $X(G)\leq c$, де $c$ – деяка константа, що обчислюється на графі $G$.

**Визначення**. Максимальне число вершин, що утворюють повний підграф у графі $G$, називають щільністю $G$ і позначають через $\omega(G)$.

**Повний підграф** деякого графа $G$ – це підграф, що складається з попарно суміжних вершин.

**Перша нижня оцінка** може застосовуватися у випадку, якщо підграфом деякого графа є повний підграф.

*Перша нижня оцінка*
Для довільного графа $G$ справедлива нерівність $X\left(G\right) \geq \omega\left(G\right)$.

**Визначення.** Будь-яку множину попарно несуміжних вершин графа $G$ називають *незалежною множиною.*
**Визначення.** Максимальне число вершин у незалежній множині називають *числом незалежності* графа $G$ й позначають через $\beta\left(G\right)$.

**Число незалежності графа** – це поняття, протилежне за змістом поняттю щільності графа. Якщо $G$ – звичайний граф, а $\overline{G}$ – його доповнення, то $\beta\left(G\right) = \omega\left(\overline{G}\right)$.

*Друга нижня оцінка*
Для довільного графа $G$ справедлива нерівність

$$X(G) \geq \frac{n(G)}{\beta(G)},$$

де $n = n(G)-$ кількість вершин графа $G$,

$\beta(G)-$ число незалежності, яке дорівнює числу внутрішньої стійкості графа $G$.

*Третя нижня оцінка хроматичного числа*

Існують **нижні оцінки** хроматичного числа, які використовують тільки ті характеристики графа, що легко обчислюються. Наведемо без доведення одну з них.

Якщо $G$ – звичайний граф і $n = n(G)-$ кількість вершин графа $G$,

$m = m(G)-$ кількість ребер графа $G$,

то хроматичне число $X(G) \geq \frac{n^2}{n^2 - 2m}$ .

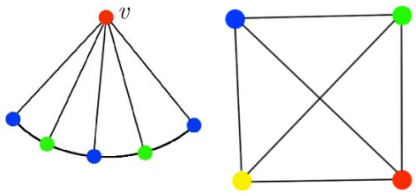
Легко зрозуміти, що в повному графі (як і в будь-якому звичайному графі) подвоєне число ребер менше квадрата числа вершин, і тому число, що стоїть в знаменнику в правій частині нерівності, завжди додатне.

Як видно з описаних вище результатів, **задачі визначення хроматичного числа** графа й побудови мінімального розфарбування довільного графа **досить складні**, а ефективні алгоритми їх розв'язування невідомі. Розглянемо простий алгоритм побудови правильного розфарбування, який у деяких випадках дає розфарбування, близькі до мінімальних.

### 3.8.6. Верхня оцінка хроматичного числа

**Теорема 3.18**. Для будь-якого графа $G$ наявна нерівність $X_p\left(G\right) \leq r+1$, де $r = \max\limits_{v \in V}\left(\deg\left(v\right)\right)$.

**Приклад 3.51.** Перевірити правильність верхньої оцінки хроматичного числа для графів, зображених на рис. 3.128.




а б

Рис. 3.128. Графи для визначення верхньої оцінки хроматичного числа

*Розв'язок.*

Розглянемо граф, що показаний на рис. 3.128 а.

1. Знаходимо вершину графа, яка має максимальний степінь.

$$r = \max_{v \in V} \left( \deg(v) \right) = 5$$

2. Визначаємо верхню оцінку хроматичного числа $X_p(G) \le r + 1$.

Оскільки $r = 5$, то $X_p(G) \le 6$. З рис. 3.128 а видно, що $X_p(G) = 3$. Оскільки $3 < 6$, то оцінка правильна. Граф, показаний на рис. 3.128 б, є регулярним графом. Тому $r = 3$. Звідси $X_p(G) \le 4$. З рис. 3.128 б видно, що $X_p(G) = 4$. Отже, оцінка правильна і у цьому випадку.

**Наслідок.** Будь-який кубічний граф розфарбовується за допомогою чотирьох фарб. На рис. 3.129 наведено приклад розфарбування кубічного графа.

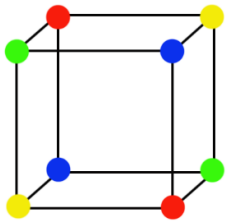


Рис. 3.129. Мінімальне правильне розфарбування куба

Теорема Брукса дає можливість більш точної верхньої оцінки хроматичного числа для спеціальних графів.

### 3.8.7. Теорема Брукса

Якщо $G$ – зв'язний неповний граф і

$$r \ge 3, \text{ де } r = \max_{v \in V} \left( \deg(v) \right), \text{ то } X_p(G) \le r \ .$$

**Приклад 3.52.** Перевірити правильність верхньої оцінки хроматичного числа для графа, зображеного на рис. 3.130.

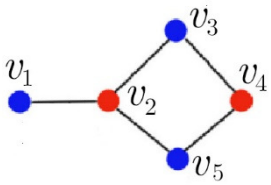



Рис. 3.130. Мінімальне правильне розфарбування графа

*Розв'язок.*
Розглянемо умови застосування теореми Брукса.

1. Вершина $v_2$ має максимальну степінь $\deg(v_2)=3$.
2. Граф є зв'язним та неповним.

Отже, $X_p(G)\le 3$ за теоремою Брукса. Звідси можна зробити висновок, що оцінка правильна.
Хоча обидві теореми й дають певну інформацію про хроматичне число графа, але їх оцінки досить неточні.

Дійсно, **зірковий граф** $K_{1n}$, який згідно з теоремою Брукса **розфарбовується** $n$ **фарбами**, насправді є **біхроматичним** (рис. 3.131).

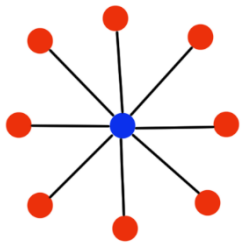


Рис. 3.131. Біхроматичний граф

Ця ситуація значно спрощується, якщо обмежитися ***планарними графами***. У цьому випадку легко довести такий досить загальний і важливий факт.

### 3.8.8. Теореми про шість, п'ять та чотири фарби

**Теорема про шість фарб**. Для будь-якого планарного (ізоморфного плоскому (у якому ребра перетинаються лише у вершинах)) графа $G$ вірна нерівність $X_p(G)\le 6$.

Більш детальний аналіз шляхів зниження верхньої границі хроматичного числа приводить до так званої *теореми про п'ять фарб.*

**Теорема про п'ять фарб.** Для будь-якого планарного графа $G$ вірна нерівність $X_p(G)\le 5$.

**Теорема про чотири фарби**. Кожний планарный граф без петель і кратних ребер є не більш ніж 4-хроматичним.

Проблема чотирьох фарб залишалася невирішеною протягом багатьох років. Стверджується, що ця теорема була доведена за допомогою певних міркувань і комп'ютерної програми в 1976 році (**Kenneth Appel and Wolfgang Haken. Every Planar Map is Four Colorable. Contemporary Mathematics 98, American Mathematical Society, 1980**).

### 3.8.9. Задача про розподіл устаткування

На підприємстві планують виконати 8 робіт $v_1, v_2, \ldots, v_8$. Для виконання цих робіт необхідні механізми $a_1, a_2, \ldots, a_6$. Використання механізмів для кожної з робіт визначається наступною таблицею 3.1.

**Таблиця 3.1.**

Таблиця використання механізмів

| Механізм | Робота | | | | | | |
|---|---|---|---|---|---|---|---|
| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ |
| $a_1$ | + | | + | | | | + |
| $a_2$ | | + | | + | | | |
| $a_3$ | | | + | | | + | + |
| $a_4$ | + | + | | + | + | | |
| $a_5$ | | | + | | + | | |
| $a_6$ | | | | | + | + | |

Жоден з механізмів не може бути використаний одночасно на двох роботах. Виконання кожної роботи займає 1 годину. Як розподілити механізми, щоб сумарний час виконання всіх робіт був мінімальним і який цей час?

*Розв'язок.* Розглянемо граф $G$, показаний на рис. 3.132. Вершинами даного графа є плановані роботи $v_1, v_2, \ldots, v_8$, а ребра з'єднують роботи, у яких бере участь хоча б один загальний механізм (і які, з цієї причини, не можна проводити одночасно).

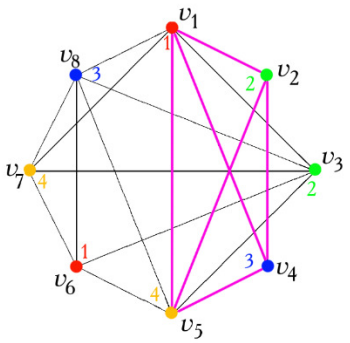



Рис. 3.132. Граф планування робіт

Вершини $v_1, v_2, v_4, v_5$ породжують підграф графа $G$, ізоморфний $K_4$. Отже, $X\{G\} \geq 4$. Отже, $X(G)$.

Таким чином, усі роботи можна виконати за 4 години.

Для цього, відповідно до знайденого розфарбування графа $G$, потрібно за першу годину виконувати роботи $v_1$ і $v_6$, за другу годину — роботи $v_2$ і $v_3$, за третю годину — роботи $v_4$ і $v_8$, за четверту годину — роботи $v_5$ й $v_7$.

У таблиці 3.2 кольорами показана послідовність виконання робіт, яка забезпечує максимальне завантаження механізмів.

**Таблиця 3.2.**

Таблиця визначення послідовності робіт

| Механізм | Робота | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ |
| $a_1$ | + | | + | | | | + | + |
| $a_2$ | | + | | + | | | | |
| $a_3$ | | | + | | | + | + | |
| $a_4$ | + | + | | + | + | | | |
| $a_5$ | | | + | | + | | | + |
| $a_6$ | | | | | + | + | | + |

### 3.8.10. Задача складання розкладу

**Умова задачі**

Потрібно прочитати лекції з чотирьох предметів двом групам студентів. Деякі з лекцій *не можуть бути прочитані одночасно* з ряду причин, а саме:

1. Лекцію з даного предмету в різних групах читає той самий лектор.
2. Дві лекції одній і тій же групі одночасно читатися не можуть.
3. Різні лекції повинні проходити в тому самому приміщенні.

Потрібно скласти розклад так, щоб читання всіх лекцій зайняло мінімально можливий час (за «одиницю часу» у даній задачі природно розглядати одну пару).

***Конкретизуємо постановку задачі***

1. Будемо розглядати дві групи студентів: 1 і 2.
2. Предмети:

- обчислювальні методи – читає викладач X,
- дискретна математика – читає викладач X,
- математичний аналіз – читає викладач Y,
- українська мова – читає викладач Z.

Знайти мінімальне число пар, у які можна «укласти» усі заняття, і скласти відповідний розклад.

***Тобто, створити максимально щільний розклад***

**Розв'язок.** Створимо граф, показаний на рис. 3.133. Вершини даного графа позначені такими символами: О1, О2, Д1, Д2, М1, М2, У1 і У2 (буква

відповідає предмету, а цифра – номеру групи). З'єднаємо ребрами вершини, відповідні до пар, які не можна проводити одночасно.

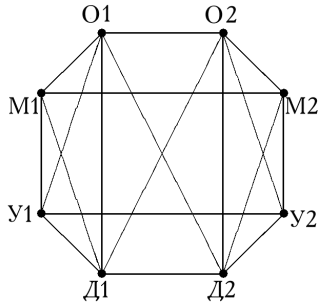



Рис. 3.133. Граф для складання розкладу

Вершини О1, О2, Д1 і Д2 цього графа породжують у ньому підграф, ізоморфний графу $K_4$. Отже, хроматичне число нашого графа не менше 4. На рис. 3.134 зазначене правильне розфарбування нашого графа в 4 фарби.

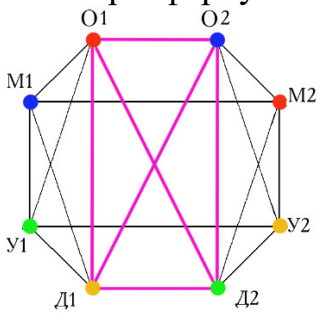



Рис. 3.134. Мінімальне правильне розфарбування графа розкладу

Отже, хроматичне число графа дорівнює 4, тобто всі заняття можна провести за 4 пари. Відповідний розклад зазначений у таблиці 3.3.

**Таблиця 3.3.**

Оптимальний розклад

| | Група 1 | Група 2 |
|---|---|---|
| 1 пара | Обч. методи | Матем. аналіз |
| 2 пара | Матем. аналіз | Обч. методи |
| 3 пара | Українська мова | Дискр. матем |
| 4 пара | Дискр. матем. | Українська мова |

**Контрольні запитання**

1. Для яких графів відомі точні значення хроматичних чисел?
2. Чому дорівнює хроматичне число графа $G$, показаного на рис. 3.135?

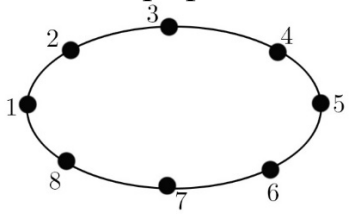



Рис. 3.135. Граф до задачі 2

3. Визначити першу нижню оцінку хроматичного числа для графа, показаного на рис. 3.136.

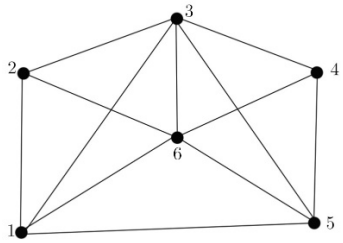



Рис. 3.136. Граф до задачі 3

4. Визначити другу та третю нижні оцінки хроматичного числа для графа, показаного на рис. 3.137.

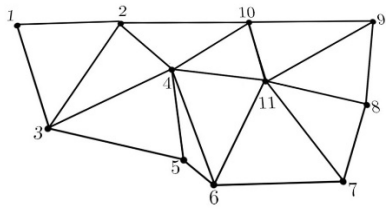



Рис. 3.137. Граф до задач 4 та 5

5. Застосувати теорему Брукса для визначення верхньої оцінки хроматичного числа для графа, зображеного на рис. 3.137.

6. Визначте точне значення хроматичного число повного графа з 15 вершин, після видалення одного ребра.

## 3.9. Основні алгоритми розфарбування графів

### 3.9.1. Базові відомості

Будемо розглядати алгоритми розфарбування на неорієнтованих графах без петель.

Граф $G$ називають $r$*-хроматичним*, якщо його вершини *можуть бути розфарбовані* з використанням $r$ кольорів (фарб) так, що не знайдеться двох суміжних вершин одного кольору. Найменше число $r$, таке, що граф $G$ є $r$ *хроматичним*, називають *хроматичним числом* графа $G$ і позначають $X(G)$.

Задачу знаходження хроматичного числа графа називають *задачею про розфарбовування* (або *завданням розфарбовування*) графа. Відповідно до хроматичного числа розфарбування вершин розбиває множину вершин графа на $r$ підмножин, кожна з яких містить вершини одного кольору. Ці множини є незалежними, оскільки в межах однієї множини немає двох суміжних вершин.

Завдання знаходження хроматичного числа довільного графа стало предметом багатьох досліджень наприкінці XIX та у XX столітті. З цього питання отримано багато цікавих результатів.

Хроматичне число графа не можна знайти, знаючи тільки кількість вершин і ребер графа. Недостатньо також знати степінь кожної вершини, щоб обчислити хроматичне число графа. При відомих величинах $n$ (кількість

вершин), $m$ (кількість ребер) і $\deg(x_1),...,\deg(x_n)$ (степені вершин графа) можна отримати тільки верхню і нижню оцінки для хроматичного числа графа.

### 3.9.2. Алгоритм неявного перебору

Алгоритм прямого неявного перебору є найпростішим алгоритмом вершинного розфарбування графів. Цей алгоритм дозволяє реалізувати правильне розфарбування графа з вибором мінімальної в рамках даного алгоритму кількості фарб.

Нехай множина вершин графа довільно упорядкована, тобто для довільної $i-$ї вершини графа існує певний індекс $v_i$. Тоді всі вершини графа $G$ утворюють множину $V=\{v_1,v_2,...,v_i,...,v_n\}$, де $n-$потужність множини вершин: $|V|=n$.

Тоді перше допустиме розфарбування може бути одержано так:

1. Розфарбуємо першу довільну вершину $v_i$ кольором 1.

2. Кожну наступну вершину будемо розфарбовувати, застосовуючи таке правило:

3. Вибираємо вершину $v_{i+1}$:

- намагаємося використати колір 1.
- якщо колір 1 використати не вдається, то вводимо колір 2 і розфарбовуємо вершину кольором 2.

4. Вибираємо вершину $v_{i+2}$

- намагаємося використати колір 1.
- якщо колір 1 використати не вдається, то намагаємося використати колір 2.
- якщо колір 2 використати не вдається, то вводимо колір 3 і розфарбовуємо вершину кольором 3.

5. Продовжуємо алгоритм, використовуючи такий порядок розфарбування до того моменту, коли всі вершини графа будуть розфарбовані.

Вибравши початковою іншу вершину або інакше упорядкувавши граф, можемо одержати інше розфарбування.

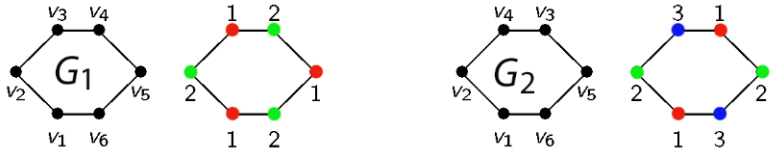
Кількість обчислень за даним алгоритмом може бути мінімальною, якщо вершини пронумерувати таким чином, щоб першими стояли ті вершини, які входять у *кліку* графа.

**Кліка** в неорієнтованому графі – це підмножина його вершин, яка утворює повний підграф.

Отже, якщо вершини $v_1,v_2,...,v_i$ розфарбовані $l$ кольорами $1,2,...,l;\ l\le i$, то новій довільно взятій вершині $v_{i+1}$ припишемо мінімальний колір, не використаний при розфарбуванні суміжних з нею вершин.

Розфарбування, до якого приводить описаний алгоритм, називають алгоритмом прямого неявного перебору або послідовним алгоритмом розфарбування.

Розглянемо кільцевий граф, вершини якого пронумеровані у різний спосіб (рис. 3.138). У першому випадку маємо граф $G_1$ (а), а у другому випадку – граф $G_2$ (б).




а б

Рис. 3.138. Розфарбування кільцевих графів з різною нумерацією

Отримане розфарбування завжди правильне, але не завжди оптимальне навіть для простих графів.

Починаючи з вершини $v_1$ та використовуючи алгоритм неявного перебору, одержимо розфарбування лише двома фарбами. У другому випадку, при розфарбуванні вершини $v_4$, виникає ситуація, коли необхідно вводити колір 3. Отже, основним недоліком алгоритму прямого неявного перебору є його залежність від нумерації вершин.

### 3.9.3. Програмний код алгоритму прямого неявного перебору

Програмний код написаний мовою програмування $Python$.

Він забезпечує генерацію випадкової симетричної матриці суміжності $a[r]$ та випадкового списку кольорів $colarr$[i]

```
from random import *
n = 5 # максимальна кількість вершин графа
colarr=[0 for i in range(n)]
# Генерація симетричної матриці
a = []
for r in range(n): # n рядків
    a.append([]) # створили рядок
    for c in range(n): # в кожному рядку по n елементів
        if r > c:
            # додаємо однонаправлене ребро
            a[r].append(randrange(0,2))
        else:
            a[r].append(0) # протилежний напрям
# Формування протилежних напрямів
for r in range(n):
    for c in range(n):
        if a[r][c]!=0:
            a[c][r]=a[r][c]
for r in range(n): # 6 рядків
```

```
    print(a[r])
def color(i):
#Функція вибору фарби для розфарбування вершини з номером i
  w = {0};
  for j in range(i):
     if a[j][i]>0: w.add(colarr[j])
  curcol=0
  while True:
      curcol+=1
      if curcol not in w: break
  return curcol

for i in range(n):
     colarr[i]=color(i)
print(colarr)
```

Наведемо приклад розфарбування графа за даним алгоритмом.

**Приклад 3.53.** Розглянемо граф $G(V,E)$, показаний на рис. 3.139.

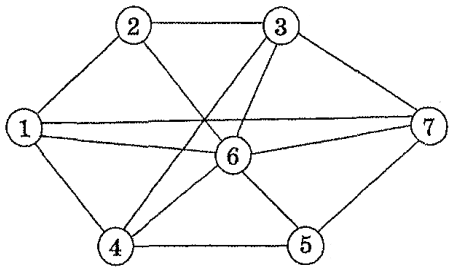



Рис. 3.139. Граф для розфарбування прямим неявним перебором

Множину вершин графа $V=\{1,2,3,4,5,6,7\}$ потрібно розфарбувати з використанням алгоритму послідовного розфарбування.

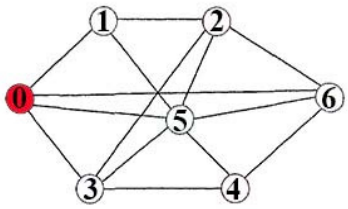
*Розв'язок.*

Сформуємо матрицю суміжності $A$ :

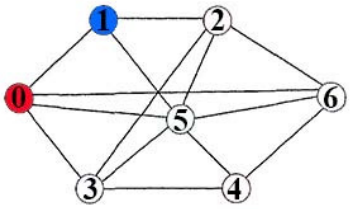
$$A=\begin{pmatrix} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 3 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 4 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 5 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 6 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Використовуючи дану матрицю, виконаємо алгоритм покроково.

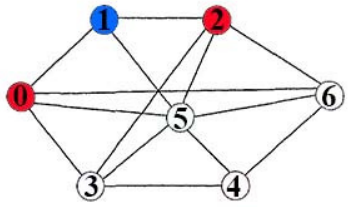
***Крок* 0**. Розглядаємо вершину 0. Множина розфарбованих суміжних вершин $w$ містить колір 0. Тому функція color(0) повертає 1. Нехай колір 1- червоний.



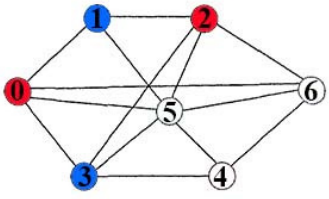

***Крок* 1**. Розглянемо вершину 1. Єдиною меншою за номером суміжною вершиною є вершина 0, яка уже червона. Тому множина w містить елементи {0,1}. Функція color(1) повертає наступну за номером фарбу синього кольору: curcol=1.



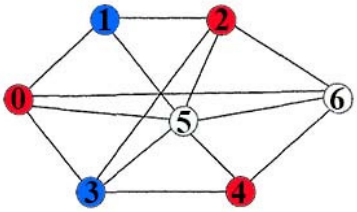

***Крок* 2**. Вершина 2 має єдину суміжну вершину 1 з меншим номером. Множина w містить елементи [0,2]. Функція color(2) повертає фарбу з номером 1 червоного кольору.



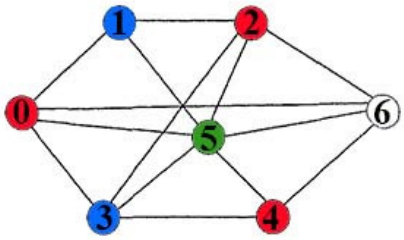

***Крок* 3**. Вершина 3 має дві суміжні вершини з меншими номерами: 0 і 2. Оскільки обидві вершини розфарбовані в колір 1, то множина w містить елементи {0,1}. Тому функція color(3) повертає наступну за номером фарбу 2 синього кольору.




***Крок* 4**. Вершина 4 має єдину суміжну вершину з меншим номером. Це вершина 3. Множина w містить елементи {0,2}. Тому функція color(4) повертає фарбу з номером 1 червоного кольору.

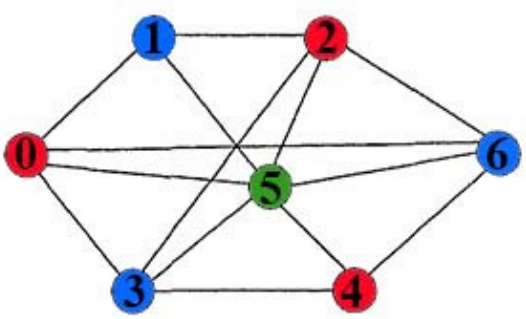


***Крок* 5**. Вершина 5 має такі суміжні вершини з меншими номерами:0,1,2 і 4. Ці вершини розфарбовані в колір 1 та колір 2. Отже, множина містить елементи: {0,1,2}. Тому функція color(5) повертає наступну за номером фарбу 3 зеленого кольору.




***Крок* 6**. Вершина 6 має такі суміжні вершини з меншими номерами: 0, 2, 4 і 5. Ці вершини розфарбовані в колір 1 та колір 3. Отже, множина w містить два елементи: {0,1,2}. Тому функція color(6) повертає фарбу 2 синього кольору.

В результаті роботи даного алгоритму одержуємо правильно розфарбований граф, що показаний на рисунку.




### 3.9.4. Евристичний алгоритм розфарбування

Точні методи розфарбовування графа складні для програмної реалізації. Однак існує багато евристичних процедур розфарбовування, які дозволяють знаходити хороші наближення для визначення хроматичного числа графа. Такі процедури також можуть з успіхом використовуватися при розфарбовуванні графів з великим числом вершин, де застосування точних методів не виправдане з огляду на високу трудомісткість обчислень.

З евристичних процедур розфарбовування слід зазначити послідовні методи, засновані на впорядкуванні множини вершин. В одному з найпростіших методів вершини спочатку розташовуються в порядку зменшення їх степенів. Перша вершина зафарбовується в колір 1, потім список вершин переглядається за зменшенням степенів, і в колір 1 зафарбовується кожна вершина, яка не є суміжною з вершинами, зафарбованими в той же колір. Потім повертаємося до першої в списку незафарбованої вершини, фарбуємо її в колір 2 і знову переглядаємо список вершин зверху вниз, зафарбовуючи в колір 2 будь-яку незафарбовану вершину, яка не з'єднана

ребром з іншою, вже пофарбованою в колір 2, вершиною. Аналогічно діємо із кольорами 3, 4 і т. д., допоки не будуть пофарбовані всі вершини. Кількість використаних кольорів буде тоді наближеним значенням хроматичного числа графа.

Послідовність дій в евристичному алгоритмі має вигляд:

1. Упорядкувати вершини за спаданням степеня.
2. Вибрати колір фарбування 1.
3. Розфарбувати першу вершину в колір 1.
4. Поки не пофарбовані всі вершини, повторювати п. 4.1–4.2:
   - 4.1. Розфарбувати в обраний колір кожну вершину, яка не суміжна з іншою вершиною, уже пофарбованою в цей колір.
   - 4.2. Вибрати наступний колір.
5. Повернутися до першої в списку нерозфарбованої вершини.

Число використаних кольорів буде тоді наближеним значенням хроматичного числа графа.

На рис. 3.140 показано приклад вибору кольору вершини $v_{i+1}$ за евристичним алгоритмом.

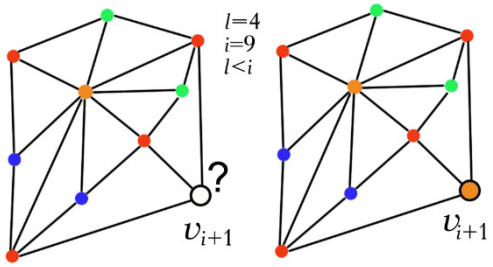



Рис. 3.140. Розфарбування графа за евристичним алгоритмом

Розглянемо кроки евристичного алгоритму детально:

***Крок* 1**. Сортувати вершини графа за степенями зменшення:

$$\deg(x_i) \geq \deg(x_j),\ \forall x_i, x_j \in G.$$

Встановити поточний колір $p = 1$ та $i = 1$.

***Крок* 2**. Вибрати чергову нерозфарбовану вершину зі списку і призначити їй новий колір $col(x_i) = p,\ X = \{x_i\}$.

***Крок* 3**. $i = i + 1$. Вибрати чергову не розфарбовану вершину $x_i$ і перевірити умову суміжності: $x_i \cap \Gamma(X) = \varnothing$, де $X$ – множина вершин, уже розфарбованих у колір $p$. Якщо вершина $x_i$ не є суміжною з даними вершинами, то також присвоїти їй колір $p$: $col[x_i] = p$.

***Крок* 4**. Повторювати крок 3 до досягнення кінця списку $(i == n)$.

***Крок* 5**. *Якщо* всі вершини графа розфарбовані, *то* – кінець алгоритму; *інакше*: $p = p + 1; i = 1$. Повторити крок 2.

Для роботи алгоритму можна використовувати довільну структуру даних, яка однозначно задає граф.

Розглянемо приклад програми реалізації евристичного алгоритму зі структурою графа, яка задана матрицею суміжності $a$.

### 3.9.5. Програмний код евристичного алгоритму

Як і у попередньому випадку, даний алгоритм передбачає початкову генерацію симетричної матриці суміжності та списку кольорів.

```
# Код евристичного алгоритму
from random import *
n = 5 # максимальна кількість вершин графа
# Початковий колір
curcol=1
# Список кольорів вершин
colarr=[0 for i in range(n)]
# Генерація симетричної матриці
a = [[0 for i in range(n)] for j in range(n)]
for r in range(n): # n рядків
    for c in range(n): # в кожному рядку по n елементів
        if r > c:
            a[r][c]= randrange(0,2) # додаємо ребро
            a[c][r]=a[r][c] # протилежний напрям

# Формування списку за степенями
def degforming():
    def getkey(item):
        return item[0]

    # Список кортежей (степінь, номер вершини)
    degarr=[[0 for i in range(2)] for j in range(n)]
    for i in range(n):
        for j in range(n):
            degarr[i][0] += a[i][j]
            degarr[i][1] = i

    # Сортуємо кортежі за спаданням степенів
    degarr.sort(key=getkey, reverse=True)
    return degarr

# Розфарбовка вершин
def dyer(curcol,node):
    for k in range(n):
        if a[node][k]==0:
            if colarr[k]==0:colarr[k]= curcol
# Основний код
```

```
sortarr=degforming()

for i in range(n):
    if not colarr[sortarr[i][1]]:
        colarr[sortarr[i][1]]=curcol
        dyer(curcol,sortarr[i][1])
        curcol+=1

            for r in range(n): print(a[r])
                    print(sortarr)
                    print(colarr)
```

### 3.9.6. Приклад евристичного алгоритму розфарбування

Розфарбуємо граф $G$, зображений на рис. 3.141.

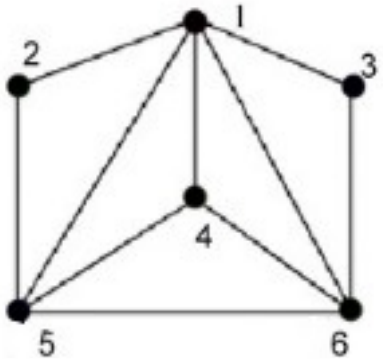



Рис. 3.141. Граф для розфарбування евристичним алгоритмом

Проміжні дані для вирішення завдання будемо записувати в таблицю 3.4.

Таблиця 3.4

Таблиця розфарбування

| Номери вершин SortArr | 1 | 5 | 6 | 4 | 2 | 3 |
|---|---|---|---|---|---|---|
| Степені вершин DegArr | *5* | *4* | *4* | *3* | *2* | *2* |
| CurCol = 1 | *1* | - | - | - | - | - |
| CurCol = 2 | *1* | *2* | - | - | - | *2* |
| CurCol = 3 | *1* | *2* | *3* | - | *3* | *2* |
| CurCol = 4 | *1* | *2* | *3* | *4* | *3* | *2* |

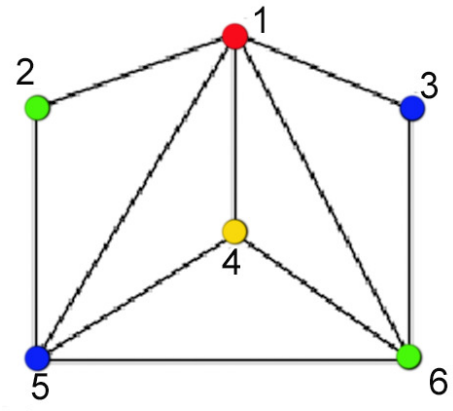
**Крок 1.** Першою SortArr стоїть вершина 1, яку фарбуємо червоним кольором 1. Несуміжних з 1 немає.

**Крок 2**. Другою в SortArr стоїть вершина 5, яку фарбуємо синім кольором 2. Несуміжна з 5 вершина 3, яку процедура dyer(curcol,5) фарбує також синім кольором 2.

**Крок 3.** Третьою в SortArr є вершина 6, яку фарбуємо зеленим кольором 3. Несуміжна з 6 вершина 2, яку процедура dyer(curcol,6) фарбує також зеленим кольором 3.

**Крок 4.** Четвертою в SortArr є вершина 4, яку фарбуємо жовтим кольором 4. Всі несуміжні вершини з 4 вже розфарбовані.

$$A = \begin{pmatrix} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 0 & 1 & 1 \\ 5 & 1 & 1 & 0 & 1 & 0 & 1 \\ 6 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$




### 3.9.7. Модифікований евристичний алгоритм розфарбування

**Попередні визначення**

**Визначення 1.** Відносний степінь – це степінь нерозфарбованих вершин у нерозфарбованому підграфі даного графа.

**Визначення 2.** Двокроковий відносний степінь – сума відносних степенів суміжних вершин у нерозфарбованому підграфі.

Проста модифікація описаної вище евристичної процедури базується на переупорядкуванні нерозфарбованих вершин по незростанню їх відносних степенів.

У даній модифікації передбачалося, що якщо дві вершини мають однакові степені, то порядок таких вершин випадковий. Їх можна впорядкувати за двокроковими степенями. Двокроковий степінь визначимо як суму відносних степенів інцидентних вершин. Аналогічно можна продовжувати далі.

***Крок* 1.** Сортуємо вершини графа за степенями зменшення:

$$\deg(x_i) \geq \deg(x_j),\ \forall x_i, x_j \in G.$$

У випадку $\deg(x_i) = \deg(x_j),\ \forall x_i, x_j \in G$ розглянемо множини суміжності $\Gamma(x_i)$ і $\Gamma(x_j)$.

Сортуємо вершини за ознакою:

$$[\deg(x_{i1}) + \deg(x_{i2}) + \ldots + \deg(x_{ik})] \geq [\deg(x_{j1}) + \deg(x_{j2}) + \ldots + \deg(x_{jn})],$$

де $x_{i1}, x_{i2}, \ldots, x_{ik}$ – нерозфарбовані вершини з множини суміжності $\Gamma(x_i)$;

$x_{j1}, x_{j2}, \ldots, x_{jn}$ – нерозфарбовані вершини з множини суміжності $\Gamma(x_j)$;

Встановити поточний колір $p=1, i=1$.

***Крок* 2**. Вибрати чергову нерозфарбовану вершину зі списку і призначити їй новий колір $col(x_i)=p$, $X=\{x_i\}$.

***Крок* 3**. $i=i+1$. Вибрати чергову нерозфарбовану вершину $x_i$ і перевірити умову суміжності: $x_i \cap \Gamma(X)=\varnothing$, де $X-$ множина вершин, вже розфарбованих у колір $p$. Якщо вершина $x_i$ не є суміжною з даними вершинами, то також присвоїти їй колір $p$ : $col(x_i)=p$.

***Крок* 4**. Повторювати крок 3 до досягнення кінця списку $(i=n)$ .

***Крок* 5**. *Якщо* всі вершини графа розфарбовані, *то* – кінець алгоритму;

*інакше*: $p=p+1$, $i=1$. Повторити крок 2.

Даний алгоритм від попереднього відрізняється ускладненням процедури сортування SortNodes, яка при сортуванні вершин з однаковими степенями враховує двокроковий степінь, тобто загальну кількість степенів вершин, суміжних з даною вершиною.

Як і в попередньому випадку, розглянемо програму, яка використовує граф, заданий матрицею суміжності $A$.

### 3.9.8. Приклад модифікованого евристичного алгоритму розфарбування

Розфарбуємо граф $G$, зображений на рис. 3.142.

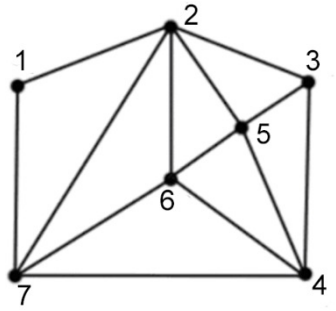



Рис. 3.142. Граф для розфарбування модифікованим евристичним алгоритмом

Множину вершин графа $V=\{1,2,3,4,5,6,7\}$ потрібно розфарбувати з використанням модифікованого евристичного алгоритму розфарбування.

Сформуємо матрицю суміжності $A$:

$$A=\begin{pmatrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 3 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 5 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 6 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 7 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

1. Модифікуємо процедуру формування списку степенів вершин `degarr` для врахування двокрокових степенів.

2. Відсортуємо вершини графа за незростанням їх перших та других степенів. Як результат, отримуємо вектор відсортованих вершин `sortarr`

Вектор степенів відсортованих вершин має наступний вигляд: $D=(5,4,4,4,4,3,2)$

У першому рядку таблиці запишемо вектор SortArr, у другому – вектор $D$, а у третьому – вектор $D^2$.

Четвертий рядок відповідає представленню степенів $D$ та $D^2$ в масиві DegArr.

Наступні рядки відображають вміст вектора розфарбування $\mathrm{col}(X^*)$.

Таким чином, даний граф можна розфарбувати не менш ніж у три кольори, тобто $X_p(G)=3$ .

Проміжні дані для вирішення завдання будемо записувати в таблицю 3.5.

Таблиця 3.5

Таблиця розфарбування модифікованим евристичним алгоритмом

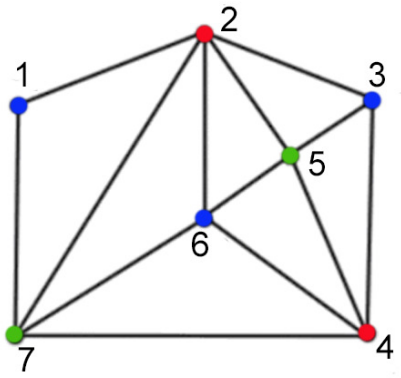
| Номери вершин $X^*$ | 2 | 6 | 5 | 4 | 7 | 3 | 1 |
|---|---|---|---|---|---|---|---|
| Степінь вершин $D$ | 5 | 4 | 4 | 4 | 4 | 3 | 2 |
| Двокроковий степінь $D^2$ | 17 | 17 | 16 | 15 | 15 | 13 | 9 |
| sortarr[i,0] | 517 | 417 | 416 | 415 | 415 | 313 | 209 |
| curcol=1 | 1 | - | - | 1 | - | - | - |
| curcol=2 | 1 | 2 | - | 1 | - | 2 | 2 |
| curcol=3 | 1 | 2 | 3 | 1 | 3 | 2 | 2 |

Робота модифікованого евристичного алгоритму по кроках:

**Крок 1.** sortarr[0][1]==2 фарбуємо червоним кольором 1. Несуміжна вершина 4. Також фарбуємо червоним.

**Крок 2**. sortarr[1][1]==6 фарбуємо синім кольором 2. Несуміжні 1 та 3 фарбуємо також синім кольором 2.

**Крок 3.** sortarr[2][1]==5 фарбуємо зеленим кольором 3. Несуміжна вершина 7. Фарбуємо також зеленим кольором 3.




### 3.9.9. Розфарбування графа методом А.П. Єршова

Андрій Петрович Єршов (1931–1988 рр.), видатний вчений в області теоретичного програмування, зробив великий внесок у розвиток інформатики. Зокрема, він створив алгоритм розфарбування графа, що вирізняється оригінальною евристичною ідеєю.

Уведемо ряд визначень.

Для даної вершини $v \in V$ графа $G(V,E)$ назвемо всі суміжні з нею вершини околом 1-го порядку – $R_1(v)$.

Усі вершини, що перебувають на відстані два від $v$, назвемо околом 2-го порядку – $R_2(v)$ .

Граф $G(V,E)$, у якого для вершини $v \in V$ всі інші вершини належать околу $R_1(v)$, назвемо граф-зіркою відносно вершини $v$.

*Ідея алгоритму*

Фарбування у фарбу $\alpha$ вершини $v$ утворює навколо неї в $R_1(v)$ «мертву зону» для фарби $\alpha$. Очевидно, при мінімальному розфарбуванні кожна фарба повинна розфарбувати максимально можливу кількість вершин графа. Для цього необхідно, щоб мертві зони, хоча б частково, перекривалися між собою. Перекриття мертвих зон двох несуміжних вершин $v_1$ і $v_2$ досягається тільки тоді, коли одна з них перебуває в околі $R_2(v_1)$ від іншої, $v_2 \in R_2(v_1)$.

Таким чином, суть алгоритму полягає в тому, щоб на черговому кроці вибрати для розфарбування фарбою $\alpha$ вершину $v_2 \in R_2(v_1)$. Цей процес повторювати доти, поки фарбою $\alpha$ не будуть пофарбовані всі можливі вершини графа.

Графічно фарбування вершин $v_1$ і $v_2$ однією фарбою можна відобразити як «склеювання» цих вершин, як показано на рис. 3.143.

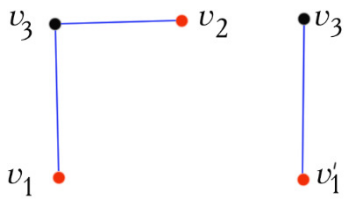



Рис. 3.143. Приклад об'єднання двох вершин: $v_1' = v_1 \cup v_2$

При цьому кількість вершин зменшується на одиницю у графі $G$, а також зменшується кількість ребер.

*Опис алгоритму*

1. Встановити $i = 0$ .
2. Вибрати в графі $G$ довільну незафарбовану вершину $v$.
3. Встановити $i = i + 1$ .
4. Пофарбувати вершину $v$ фарбою $i$.
5. Фарбувати фарбою $i$ незабарвлені вершини графа $G$, вибираючи їх з $R_2(v)$, поки граф не перетвориться в граф-зірку відносно $v$.
6. Перевірити, чи залишилися незабарвлені вершини в графі $G$. Якщо так, то перейти до п. 2, інакше – до п. 7.
7. Отриманий повний граф $K_i$. Хроматичне число графа $X(K_i) = i$.

Кінець алгоритму.

Як випливає з алгоритму, після того, як у першу обрану вершину стягнуті всі можливі й граф перетворився в граф-зірку відносно цієї вершини, вибирається довільним чином друга вершина й процес повторюється доти, поки граф не перетвориться в повний.

Повний граф – це граф-зірка відносно будь-якої вершини. Хроматичне число повного графа дорівнює числу його вершин.

### 3.9.10. Приклад розфарбування методом А.П. Єршова

На рис. 3.144 зображений граф $G$, на прикладі якого будемо розглядати роботу евристичного алгоритму Єршова.

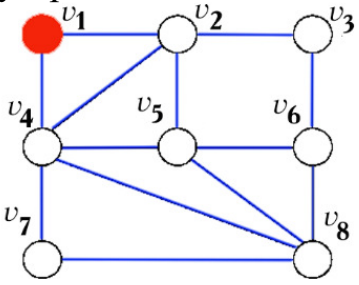



Рис. 3.144. Початковий граф $G$

Виберемо довільну вершину, наприклад, $v_1$. Окіл 2-го порядку $R_2(v_1) = \{v_3, v_5, v_7, v_8\}$. Склеїмо вершину $v_1$ наприклад, з вершиною $v_3$: $v_1' = v_1 \cup v_3$ $v_1' = v_1 \cup v_3$. Одержимо граф $G_1$, зображений на рис. 3.145.

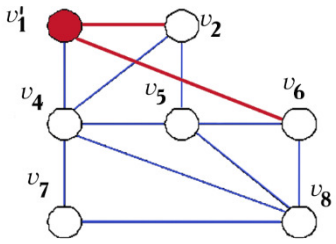



Рис. 3.145. Граф $G_1$. Склеєні вершини $v_1$ та $v_3$

Розглянемо тепер граф $G_1$. Окіл другого порядку вершини $v_1'$ визначається множиною $R_2(v_1') = \{v_5, v_7, v_8\}$. Склеїмо вершину $v_1'$, наприклад, з вершиною $v_5$ : $v_1'' = v_1' \cup v_5$. Одержимо граф $G_2$, зображений на рис. 3.146.

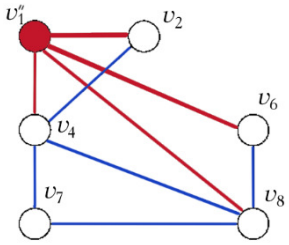



Рис. 3.146. Граф $G_2$. Склеєні вершини $v_1'$ й $v_5$

У графі $G_2$ визначимо окіл другого порядку для вершини $v_1''$: $R_2(v_1'') = \{v_7\}$. Склеїмо вершину $v_1''$ з вершиною $v_7$: $v_1''' = v_1'' \cup v_7$. Одержимо граф $G_3$, зображений на рис. 3.147. Цей граф є зіркою відносно вершини $v_1'''$.

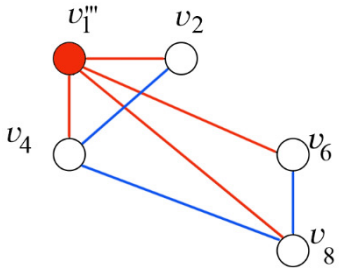



Рис. 3.147. Граф $G_3$. Склеєні вершини $v_1''$ й $v_7$

У графі $G_3$ виберемо, наприклад, вершину $v_2$ для фарбування другою фарбою. Окіл 2-го порядку $R_2(v_2) = \{v_6, v_8\}$. Склеїмо вершину $v_2$, наприклад, з вершиною $v_6$: $v_2' = v_2 \cup v_6$. Одержимо граф $G_4$, зображений на рис. 3.148.

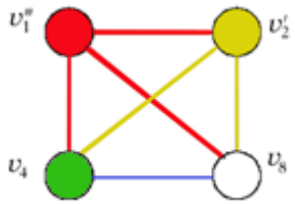



Рис. 3.148. Граф $G_4$ еквівалентний $K_4$. Склеєні вершини $v_1''$ й $v_6$

Граф $G_4$ є повним чотиривершинним графом $K_4$. Отже, граф $G_4$ на рис. 3.148 розфарбовується в чотири кольори. У перший (червоний) колір

розфарбовується вершина $v_1$ й склеєні з нею вершини: $v_3, v_5$ і $v_7$. У другий (жовтий) колір розфарбовується вершина $v_2$ й склеєна з нею вершина $v_6$. У третій (зелений) колір розфарбовується вершина $v_4$ й у четвертий (білий) колір розфарбовується вершина $v_8$.

У підсумку одержуємо правильно розфарбований граф, показаний на рис. 3.149.

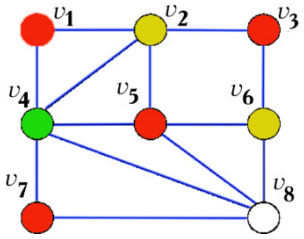



Рис. 3.149. Граф $G$, розфарбований за допомогою алгоритму розфарбування А.П. Єршова

*Програмна реалізація алгоритму А.П. Єршова*

Розглянемо матрицю суміжності графа $G$, представленого на рис. 3.149.

$$A = \begin{pmatrix} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \\ v_1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ v_2 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ v_3 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ v_4 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ v_5 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ v_6 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ v_7 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ v_8 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Операції склеювання відповідних вершин графа, які описані вище, виконуються на матриці суміжності шляхом виконання операції диз'юнкції між цими вершинами.

Наприклад, розглянемо склеювання вершин 1 та 3.

Результуюча матриця суміжності містить новий рядок та стовпець для вершини $v_1'$, але не містить $v_1$ та $v_3$.

$$A' = \begin{pmatrix} & v_1' & v_2 & v_4 & v_5 & v_6 & v_7 & v_8 \\ v_1' & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ v_2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ v_4 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ v_5 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ v_6 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ v_7 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ v_8 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Фінальна матриця суміжності має вигляд:

$$A''' = \begin{pmatrix} & v_1''' & v_2' & v_4 & v_8 \\ v_1''' & 0 & 1 & 1 & 1 \\ v_2' & 1 & 0 & 1 & 1 \\ v_4 & 1 & 1 & 0 & 1 \\ v_8 & 1 & 1 & 1 & 0 \end{pmatrix},$$

що відповідає матриці повного графа $K_4$.

```
Const n=10; {*максимальна кількість вершин графа*}
Type V=0..n;
    R2S=Set of V;
    TA = Array (1..n, 1..n) of Integer;
Var A:TA; {*матриця суміжності графа*}
   MainNode:Byte;

Procedure Glue(master,slave:Byte);
{*Процедура склеювання вершин *}
Begin
 {*Склеювання рядка master і рядка slave*}
 For i=1 to n do A[i,master]:= A[i,master] OR A[i,slave];
{*Склеювання стовпця master і стовпця slave*}
 For j=1 to n do A[master,j]:= A[master,j] OR A[slave,j];
End;

Procedure Reduce(master,slave:Byte);
{*Процедура видалення стовпця і рядка в матриці суміжності*}
Begin
 For i:=1 to n do
  For j:=1 to n-1 do
   {*Видалення рядка slave*}
   If (j≥ slave) then A[i,j]:= A[i,j+1];
   For j:=1 to n-1 do
   For i:=1 to n-1 do
   {*Видалення стовпця slave*}
    If (i≥ slave) then A[i,j]:= A[i+1,j];
    n:=n-1;
End;

Function Check_K:Byte;
{*Процедура перевірки наявності повного графа*}
Var Gh:Byte;
Begin
{*У повному графі всі елементи матриці, крім діагональних, повинні бути
одиничними*}
```

```
 Ch:=0;
 For i:=1 to n do
 For j:=1 to n do if (i≠ j ) AND (A[i,j]=0) then Ch:=j;
 Check_K:=Ch;
End;

Procrdure R2(master:Byte);
{*Процедура формування околу 2-го порядку*}
Begin
 For j:=1 to n do
 begin
  If (j≠master) AND (A[master,j]=1) then
  begin
   {*Вибрали суміжну вершину до master*}
   For i=1 to n do
   begin
    If (i≠master) AND (A[j,i]=1) then
    {* Вибрали  вершину околу 2-го порядку  до master*}
    R2S:=R2S+[i];  {*Додали вершину до множини вершин околу*}
   end;
  end;
 end;
End;

Begin
 MainNode:=1; {*Вибираємо першу вершину*}
 K_finded:=false; {*Ознака закінчення алгоритму*}
 While K_finded do
 begin
  R2S:=[]; {*Очистка множини околу 2-го порядку*}
  R2(MainNode); {*Формуємо окіл 2-го порядку для MainNode*}
  For k=1 to n do {*Цикл по вершинах графа*}
  begin
   If k in R2S then
   begin {*Вершина належить околу другого порядку*}
    {*Склеювання вершини MainNode з вершиною околу k *}
    Glue(MainNode,k);
    {*Модифікація матриці суміжності після склеювання вершин *}
    Reduce(MainNode,k);
   end;
  end;
  MainNode:= Check_K(MainNode);
  If MainNode=0 then K_finded:=true;
 end;
end;
```

### 3.9.11. Рекурсивна процедура послідовного розфарбування

1. Фіксуємо порядок обходу вершин.
2. Ідемо по суміжних вершинах, використовуючи такий найменший колір, який не створить конфліктів.
3. Якщо на черговому кроці колір вибрати не вдалось, то «відкочуємось» до попередньої вершини й вибираємо для неї наступний колір, який не створить конфліктів.

У процедурі використовується рекурсивний виклик процедури фарбування наступної вершини у випадку успішного фарбування попередньої вершини.

Код алгоритму має вигляд:

```
#Код рекурсивного алгоритму

from random import *
# Максимально допустима кількість кольорів
cmax=10

n = 5 # максимальна кількість вершин графа

#Список кольорів вершин
color=[0 for i in range(n)]

# Генерація симетричної матриці
a = [[0 for i in range(n)] for j in range(n)]
for r in range(n): # n рядків
    for c in range(n): # в кожному рядку по n елементів
        if r > c:
            a[r][c]= randrange(0,2) # додаємо ребро
            a[c][r]=a[r][c] # протилежний напрям

def visit (i):

# Функція вибору фарби для розфарбування вершини
з номером і
    def nicecolor():
        w = {0}
        newcol=0
        for j in range(n):
           if a[i][j] > 0: w.add(color[j])
        for cm in range(1,cmax):
           if cm not in w:
               newcol=cm
               break
        return newcol
```

```
 # код функції visit
    if  i == n:
 #Якщо всі вершини розфарбовані, те виводимо результат
        print("FINAL")
    else:
#Якщо поточна вершина не розфарбована
       if color[i]==0:
        curcol=nicecolor()
        if curcol >0:
#Якщо неконфліктний, то розфарб. вершину і фарбою c
            color [i] = curcol
#Рекурсивно викликаємо для наступної вершини
            visit (i + 1)
#Основний код програми
visit (0)
for r in range(n):
    print(a[r])
print()
print(color)
```

### 3.9.12. Приклад роботи рекурсивної процедури

Розглянемо граф $G$ та застосуємо рекурсивну процедуру для його розфарбування (рис. 3.150).

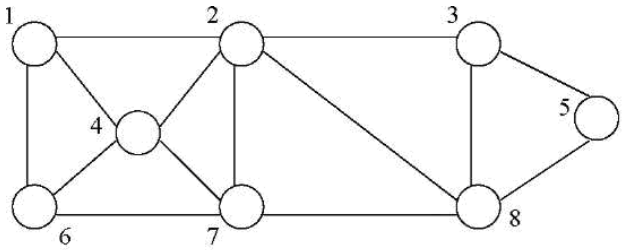



Рис. 3.150. Граф $G$ для рекурсивного розфарбування

Матриця суміжності $A$ має такий вигляд

$$A = \begin{pmatrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 3 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 4 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 6 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 7 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 8 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Робота алгоритму зведена в таблицю 3.6.

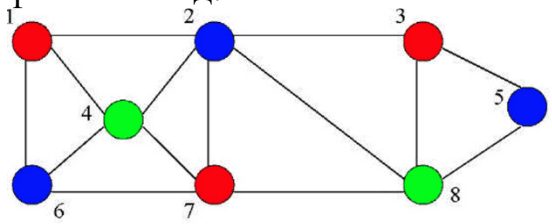
Перший стовпець містить виклики процедури Visit, а решта стовпців показує, яка фарба була прийнята, а яка відхилена.

Таблиця 3.6

Послідовність розфарбувань рекурсивним алгоритмом

| | Червоний | Синій | Зелений |
|---|---|---|---|
| Visit(1) | + | | |
| Visit(2) | - | + | |
| Visit(3) | + | | |
| Visit(4) | - | - | + |
| Visit(5) | - | + | |
| Visit(6) | - | + | |
| Visit(7) | + | | |
| Visit(8) | - | - | + |

Розфарбований граф має вигляд:




### 3.9.12. «Жадібний» алгоритм розфарбування

Нехай дано зв'язний граф $G(V,E)$ .

1. Задамо множину $monochrom = \varnothing$, куди будемо записувати всі вершини, які можна пофарбувати одним кольором.

2. Переглядаємо всі вершини й виконуємо наступний «жадібний» алгоритм:

```
Procedure Greedy
For (для кожної незафарбованої вершини v∈V)  do
If v не суміжна з вершинами з monochrom then
begin
   color(v)=колір;
   monochrom = monochrom ∪ {v}
end.
```

Розглянемо детальніше програмну реалізацію даного алгоритму за умови, що для представлення графа використовують матрицю суміжності.

```
# код жадібного алгоритму
from random import *
n=5
allcolored = False #Ознака того, що всі вершини не розфарбовані
color=0
#Список кольорів вершин
colarr=[0 for i in range(n)]
```

```
# Генерація симетричної матриці
a = [[0 for i in range(n)] for j in range(n)]
for r in range(n): # n рядків
    for c in range(n): # в кожному рядку по n елементів
        if r > c:
            a[r][c]= randrange(0,2) # додаємо ребро
            a[c][r]=a[r][c] # протилежний напрям
#Функція вибору фарби для розфарбування вершини з номером i
def  avid (i,color):
   def  check (i): #Перевірка кольору суміжних вершин
       ch = True
       for j in range(n):
#Якщо вершина j суміжна з тією, що підлягає перевірці
           if a [i][j] == 1:
               if (j in w): ch = False
       return ch
   w = set()  # Очищаємо множину одноколірних вершин
#Розфарбовуємо першу вершину новою фарбою
   colarr [i]=color
#Доповнюємо множину одноколірних вершин вершиною
   w.add(i)
   #Перевіряємо інші вершини на можливість розфарбування цією фарбою
   for k in range(n):
       if colarr[k] == 0:
          if check(k):
              colarr [k] = color
              w.add(k)
# Головний код
while not allcolored: #Цикл по вершинах графа
   allcolored = True
   for i in range(n):
       if colarr [i] == 0: #Знайшли не розфарбовану вершину
           color += 1  # Встановлюємо новий колір
           allcolored = False
           avid (i,color) #процедура жадібного розфарбування

for r in range(n):  # 6 рядків
print(a[r])
print()
print(colarr)
```

*Можливі результати роботи «жадібного» алгоритму розфарбування*

Розглянемо варіанти розфарбування графа, показаного на рис. 3.151.

Нехай зафарбуємо вершину 1 у синій колір, а потім, пропустивши вершину 2, зафарбуємо в синій колір вершини 3 і 4. Тоді можна одержати 2 фарби.

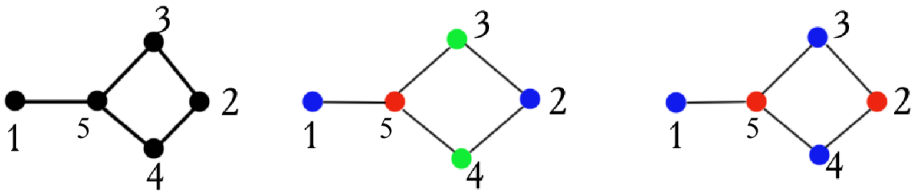



Рис. 3.151. Варіанти розфарбування графа

Але «жадібний» алгоритм, ґрунтуючись на нумерації вершин, зафарбує в синій колір вершини 1 і 2, для розфарбування графа тепер потрібно 3 фарби.

### 3.9.14. Приклад роботи «жадібного» алгоритму розфарбування

Розглянемо граф $G$ (рис. 3.152) та застосуємо до нього «жадібний» алгоритм розфарбування.

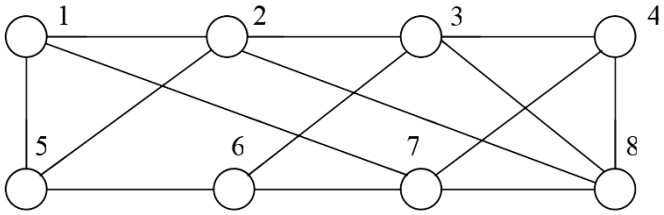



Рис. 3.152. Граф для розфарбування жадібним алгоритмом

Матриця суміжності $A$ має такий вигляд:

$$A = \begin{pmatrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 4 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 5 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 6 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 7 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 8 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Для початку розфарбування вибираємо вершину з номером 1 та розфарбовуємо її в колір 1 (червоний).

Далі відбувається пошук несуміжної вершини з вершиною 1. Якщо така вершина знайдена, то вона також розфарбовується в колір 1 (червоний).

Наступна знайдена для розфарбування кольором 1 вершина повинна бути не суміжною з двома попередніми. Процес продовжується до того часу, поки всі можливості розфарбувати вершини кольором 1 будуть вичерпані.

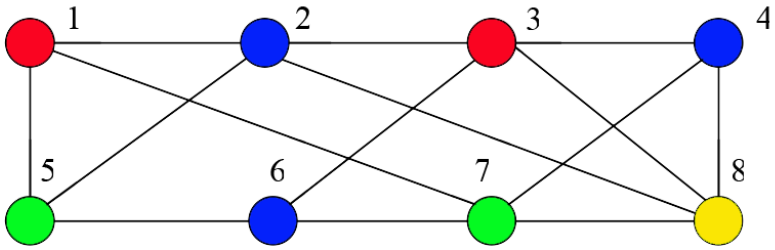
Після цього вибираємо фарбу кольору 2 (синя) і розфарбовуємо нею вершину з мінімальним номером, яка є не розфарбованою до цього часу.

Наступна придатна для розфарбування фарбою 2 вершина повинна бути не суміжною з вершиною, яка була розфарбована кольором 2 (синій) на попередньому кроці. Процес розфарбування фарбою 2 також продовжується до того часу, поки не будуть вичерпані всі можливості розфарбування вершин цією фарбою.

Перед вибором чергової фарби для розфарбування завжди перевіряємо, чи залишилися ще не розфарбовані вершини. Якщо такі вершини знайдено, то вибираємо чергову фарбу і продовжуємо процес розфарбування.

Якщо ж всі вершини графа розфарбовано, то процес розфарбування жадібним алгоритмом закінчується.

Результат розфарбування «жадібним» алгоритмом графа $G$ показано на рисунку.




**Контрольні запитання**

1. Які переваги і недоліки алгоритму прямого неявного перебору?
2. На яких базових принципах ґрунтується евристичний алгоритм розфарбування?
3. Чим відрізняється евристичний алгоритм від модифікованого евристичного алгоритму розфарбування графа.
4. Зобразіть графічно послідовність стягування графа, показаного на рис. 3.153, за алгоритмом А.П. Єршова.

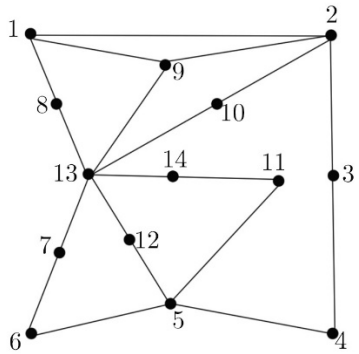



Рис. 3.153. Граф до задачі 4

5. Опишіть у вигляді псевдокоду принцип роботи «жадібного» алгоритму розфарбування графа.
6. Які переваги і недоліки рекурсивного алгоритму розфарбування графів?

## 3.10. Шляхи та цикли Ейлера. Плоскі та планарні графи

### 3.10.1. Шляхи та цикли Ейлера

**Визначення.** Нехай $G=(V,E)$ – граф. Цикл, який включає всі ребра і вершини графа $G$, називають *циклом Ейлера.* Якщо ця умова виконується, говорять, що граф $G$ має цикл Ейлера.

Якщо тепер повернутися до задачі про кенігсберзькі мости, легко переконатися, що вона зводиться до спроби визначити, чи містить граф, що ілюструє задачу, цикл Ейлера. Для цього нам буде потрібна наведена нижче теорема. Ця теорема справедлива також для мультиграфів. Більше того, доведення теореми залишається тим же. Для ясності будемо використовувати термін граф, розуміючи, що кожне твердження справедливе для мультиграфів.

**Теорема 3.19.** Граф з більше, ніж однією вершиною, має цикл Ейлера тоді і тільки тоді, коли він зв'язний і кожна його вершина має парний степінь.

**Приклад 3.54.** Наведений на рис. 3.154 граф має цикл Ейлера, оскільки степінь кожної його вершини парна.

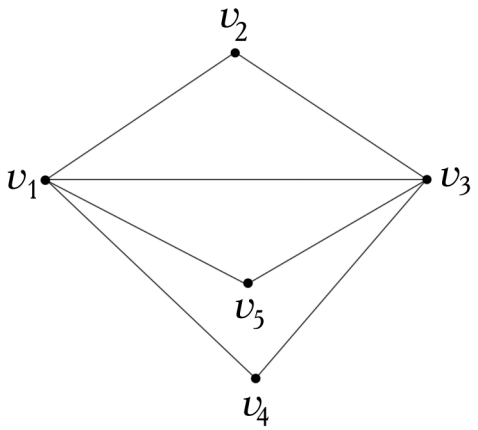



Рис. 3.154. Граф з циклом Ейлера

Приклади циклів Ейлера:
$(v_1,v_2,v_3,v_4,v_1,v_5,v_3,v_1,)$, $(v_4,v_3,v_2,v_1,v_3,v_5,v_1,v_4)$ і т. д.

Повертаючись до задачі про кенігсберзькі мости, виявляємо, що мультиграф, побудований Ейлером для опису кенігсберзьких мостів, має непарні степені всіх його вершин.

Отже, цей мультиграф не має циклу Ейлера, тому неможливо пройти кожний міст по одному разу і повернутися у початкову точку шляху у графі, показаному на рис. 3.155.

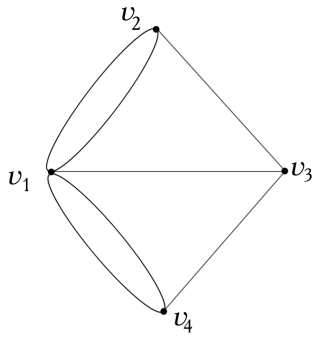


Рис. 3.155. Граф кенігсберзьких мостів

Як видно з рисунка, задача про кенігсберзькі мости була розв'язана з використанням мультиграфа. Однак цю задачу можна розв'язати, використовуючи простий граф. Для цього початковий мультиграф необхідно привести до простого графа шляхом введення додаткових вершин (рис. 3.156):

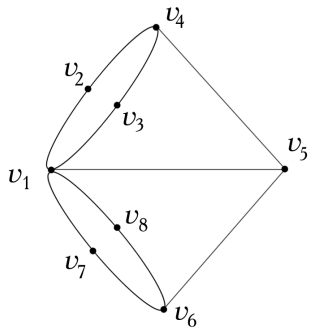


Рис. 3.156. Простий граф кенігсберзьких мостів

Якщо поставити задачу не повертатися у початкову точку, то попередня задача зводиться до пошуку шляху в графі.

**Визначення.** Нехай $G(V,E)$ – граф. Шлях, який включає кожне ребро графа $G$ тільки один раз, називають *шляхом Ейлера*. У цьому випадку говорять, що граф $G$ має шлях Ейлера.

У випадку, коли шлях Ейлера не є циклом Ейлера, його називають власним шляхом Ейлера.

**Теорема 3.20.** Граф або мультиграф має власний шлях Ейлера тоді і тільки тоді, коли він зв'язний і рівно дві його вершини мають непарний степінь.

Оскільки граф для кенігсберзьких мостів має чотири вершини з непарними степенями, можна зробити висновок про неможливість пройти кожний міст по одному разу, навіть якщо не потрібно повертатися у початкову точку маршруту.

**Приклад 3.55.** На рис. 3.157 показано граф, який має шлях Ейлера, оскільки дві його вершини мають непарний степінь.

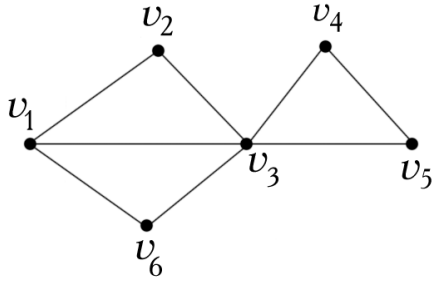



Рис. 3.157. Граф з двома непарними степенями

Приклад шляху Ейлера: $v_1, v_2, v_3, v_1, v_6, v_3, v_4, v_5, v_3$.

*Цикл Ейлера в орієнтованому графі*

**Визначення.** Нехай $G(V,E)$ – орієнтований граф. *Орієнтованим циклом* називають орієнтований шлях ненульової довжини з вершини в ту ж вершину без повторення ребер.

**Визначення.** Нехай $G(V,E)$ – орієнтований граф. Орієнтований цикл, який включає всі ребра і вершини графа $G$ , називають *циклом Ейлера.* У цьому випадку говорять, що орієнтований граф $G$ має цикл Ейлера.

**Теорема 3.21.** Орієнтований граф має цикл Ейлера тоді і тільки тоді, коли він зв'язний і напівстепінь входу кожної вершини дорівнює її напівстепені виходу.

*Алгоритм побудови циклу Ейлера*

1. Виходимо з довільно обраної вершини $v$ і кожне пройдене ребро закреслюємо.
2. Ніколи не йдемо по тому ребру $e$, яке в даний момент є мостом (тобто при видаленні якого граф, утворений незакресленими ребрами, розпадається на два компонента зв'язності, що мають хоча б по одному ребру).
3. Не вибираємо ребро, що веде в $v$, поки є інші можливості.

### 3.10.2. Цикли Гамільтона (основні визначення)

**Визначення.** *Гамільтоновим ланцюгом* графа називають його простий ланцюг, який проходить через кожну вершину графа точно один раз.

**Визначення.** Цикл графа, що проходить через кожну його вершину, називають *гамільтоновим циклом*.

**Визначення.** Граф називають *гамільтоновим*, якщо він має гамільтонів цикл.

**Визначення.** Граф, який містить простий шлях, що проходить через кожну його вершину, називають *напівгамільтоновим.*

Це визначення можна поширити на орієнтовані графи, якщо шлях вважати орієнтованим. Гамільтонів цикл не обов'язково містить усі ребра графа. Ясно, що гамільтоновим може бути тільки зв'язний граф і що будь-який гамільтонів граф є напівгамільтоновим. Помітимо, що гамільтонів цикл існує далеко не в кожному графі.

**Зауваження:** Будь-який граф $G$ можна перетворити в гамільтонів граф, додавши достатню кількість вершин. Для цього, наприклад, достатньо до вершин $v_1,...,v_p$ графа $G$ додати вершини $u_1,...,u_p$ і множини ребер $\{v_i,u_i\}$ та $\{u_i,v_{i+1}\}$.

Степенем вершини $v$ називають число ребер $\deg(v)$, інцидентних їй, при цьому петля не враховується. У випадку орієнтованого графа розрізняють степінь $\deg^+(v)$ по вихідних дугах і $\deg^-(v)$ – по вхідних.

**Теорема 3.22.** Нехай $G$ має $p \geq 3$ вершин. Якщо для будь-якого $n$, $1 \leq n \leq \frac{p-1}{2}$, кількість вершин зі степенями, що не перевищують $n$, менше ніж $n$, і для непарного $p$ кількість вершин степеня $\frac{p-1}{2}$ не перевищує $\frac{p-1}{2}$, то $G$ – гамільтонів граф.

**Наслідок 1.** Якщо $p \geq 3$ і $\deg(u)+\deg(v) \geq p$ для будь-якої пари $u$ і $v$ несуміжних вершин графа $G$, то $G$ – гамільтонів граф.

**Наслідок 2.** Якщо $p > 3$ і $\deg(v) \geq \frac{p}{2}$ для будь-якої вершини $v$ графа $G$, то $G$ – гамільтонів граф.

**Теорема 3.23.** У повному графі $G(V,E)$ завжди існує гамільтонів шлях.

### 3.10.3. Плоскі та планарні графи. Загальні поняття про плоский граф

У визначенні графа як геометричної фігури до цього ми не накладали жодних обмежень на розташування фігури в просторі. Тепер же будемо говорити, що граф зображений на поверхні (площини, сфері, і т. п.), якщо всі його **вершини і ребра належать цій поверхні**.

**Визначення.** Граф, зображений на площині, називають *плоским,* якщо його ребра не перетинаються в точках, відмінних від вершин графа.

*Відзначимо, що властивість графа* ***бути*** *або* ***не бути плоским –*** *це властивість геометричного зображення, а не алгебраїчного об'єкта.*

На рис. 3.158 показані графи $G_1$ і $G_2$, які є ізоморфними, але $G_1$ – плоский, а $G_2$ – неплоский.

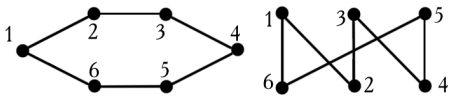


Рис. 3.158. Ізоморфні графи

*Загальні поняття про планарний граф*

Таким чином, термін «плоский граф» завжди відноситься до **конкретного** (одного з багатьох) геометричного зображення графа.

Той самий граф (як множина вершин + множина ребер) може мати **як плоскі, так і не плоскі зображення**.

У той же час, принципове питання, на яке потрібно відповідати при розв'язуванні задач типу прокладки комунікацій: «Чи має даний граф хоча б одне плоске зображення?». Визначимо клас графів, для яких відповідь на це питання позитивна.

**Визначення.** Граф називають ***планарним***, якщо він ізоморфний плоскому графу.

Іншими словами. *Планарним графом* називають граф, який може бути зображений на площині так, що його ребра не перетинаються.

*Укладання графа на поверхні*

Про планарні графи говорять, що вони мають **плоске укладання**, або що вони **укладаються на площині**.

Можна визначити укладання графів не тільки на площині, але і на **інших поверхнях** у просторі.

Властивість графа укладатися на поверхні безумовно **залежить від виду цієї поверхні**.

Однак багато поверхонь із огляду на укладання графів нічим **не відрізняються від площини**.

*Жорданова крива*

*Жордановою кривою* на площині називають неперервну криву лінію без самоперетинань.

*Замкненою жордановою кривою* називають жорданову криву, початок і кінець якої збігаються.

**Теорема Жордана.** Якщо $S$ – замкнена жорданова крива на площині, а $x$ та $y$ – дві різні точки цієї кривої, то будь-яка жорданова крива, що з'єднує точки $x$ і $y$, або повністю лежать всередині $S$, крім точок $x$ і $y$, або поза кривою $S$, окрім точок $x$ і $y$, або перетинає криву $S$ у деякій точці, відмінній від точок $x$ і $y$.

На рис. 3.159 показано три варіанти взаємного розташування жорданових кривих.

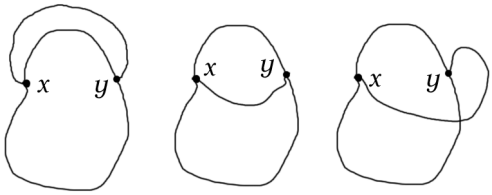



Рис. 3.159. Жорданові криві

Будемо говорити, що граф $G$ укладається в просторі $L$, якщо існує взаємно однозначне відображення вершин графа $G$ у точки і ребер у жорданові криві цього простору таке, що криві, які відповідають різним ребрам, перетинаються в інцидентних даним ребрам вершинах. Зображений у такий спосіб граф $G$ у просторі $L$ називають *укладанням графа* $G$.

*Теорема про укладання графа в тривимірному просторі*

**Теорема 3.24.** Будь-який граф укладається в тривимірному просторі.

**Доведення.** Розмістимо всі вершини графа $G=(V,E)$ на осі $0x$. З пучка площин, що проходять через цю вісь, виберемо $|E|$ різних площин. Далі кожне ребро $(u,v)\in E$ зобразимо у відповідній площині півколом, що проходить через вершини $u$ і $v$, як показано на рис. 3.160. Ясно, що в результаті одержимо укладання графа $G$ у тривимірний простір, оскільки всі ребра лежать у різних площинах, і тому не перетинаються ні в яких точках, крім вершин.

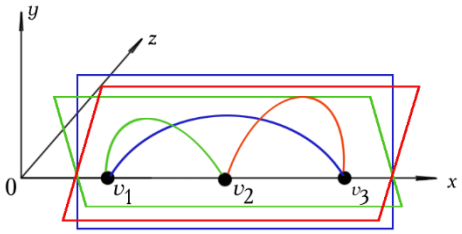



Рис. 3.160. Граф з ребрами у різних площинах

**Теорема 3.25.** Граф укладається на сфері тоді і тільки тоді, коли він планарний.

*Д о в е д е н н я. Необхідність.* Нехай граф $G$ укладений на сфері; побудуємо його ізоморфізм на плоский граф. Для цього виберемо на сфері точку $N$, що не належить $G$, а в діаметрально протилежній до точки $N$ точці $S$ проведемо до сфери дотичну площину $\pi$ (рис. 3.161). Розглянемо довільну пряму, що проходить через $N$ не паралельно $\pi$. Ця пряма не є дотичною до сфери, оскільки дотичні площини до сфери в точках $N$ і $S$ паралельні. Отже, така пряма має лише одну спільну точку $X$ зі сферою, відмінну від $N$, і лише одну спільну точку $Y$ із площиною. Визначимо функцію $\varphi$, яка переводить будь-яку точку $X$ сфери, що не співпадає з $N$, у точку $Y$ площини $\pi$, що лежить на прямій $NX$.

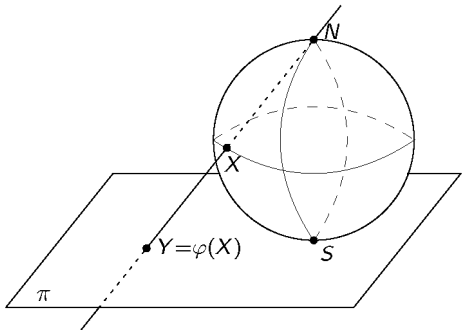



Рис. 3.161. Побудова ізоморфного графа

Функцію $\varphi$ називають *стереографічною проєкцією сфери на площину* $\pi$ *з точки* $N$.

Очевидно, що

1. $\varphi$ – бієкція (різні точки сфери переходять у різні точки площини, а для будь-якої точки $Y \in \pi$ можна знайти її прообраз, провівши пряму $YN$).

2. $\varphi$ неперервна (стандартними засобами математичного аналізу легко показати, що близькі точки на сфері переходять у близькі точки на площині), а отже, образом відрізка неперервної лінії на сфері є відрізок неперервної лінії на площині (рис. 3.162).

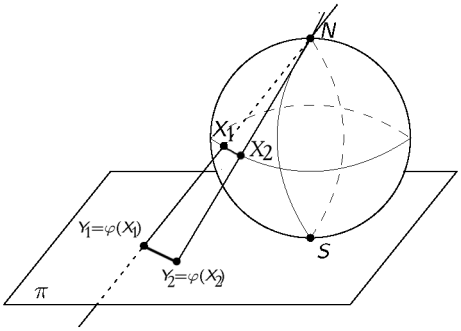



Рис. 3.162. Стереографічна проєкція сфери на площину

З неперервності $\varphi$ випливає, що геометрична фігура $\varphi(G)$, тобто образ графа $G$ при функції $\varphi$, сама є графом, як показано на рисунку (вершини і ребра $\varphi(G)$ є образами вершин і ребер $G$).

Граф $\varphi(G)$ зображений на площині (рис. 3.163). Перевіримо, чи він плоский.

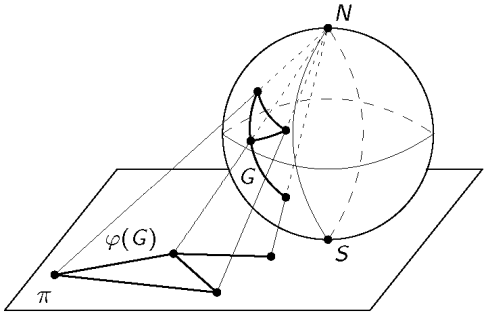



Рис. 3.163. Перевірка, чи граф $\varphi(G)$ – плоский

Нехай точка $Y$ належить двом ребрам $\varphi(G)$. Тоді точка $X=\varphi^{-1}(Y)$ належить двом відповідним ребрам графа $G$, тобто за умовою є вершиною в $G$. Але тоді $Y=\varphi(X)$ – вершина в $\varphi(G)$, що, власне і було потрібно. Залишилося відмітити, що функція $\varphi$, яка розглядається тільки на множині вершин графа $G$, є ізоморфізмом $G$ на $\varphi(G)$. Тим самим, ми довели, що граф $G$ – планарний. Отже, планарний і будь-який граф, ізоморфний $G$. Необхідність доведена.

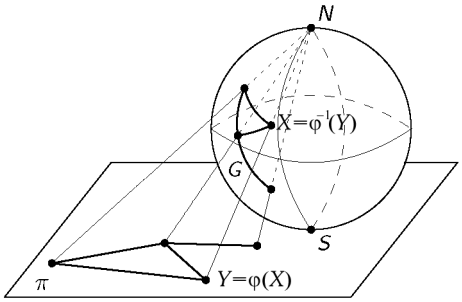



Рис. 3.164. Будь-який граф, ізоморфний графу $G$, є планарним

*Доведення теореми про укладання на сфері (достатність)*

*Достатність* легко доводиться за допомогою тієї ж самої бієкції $\varphi$ (точніше, за допомогою бієкції $\varphi^{-1}$). Змінюється тільки початок побудови: на площину, що містить заданий граф, довільним чином «ставимо» сферу, після чого за $N$ беремо точку сфери, протилежну точці дотику із площиною.

### 3.10.4. Непланарні графи

**Теорема 3.26.** Графи $K_5$ (рис. 3.165) і $K_{3,3}$ не є планарними.

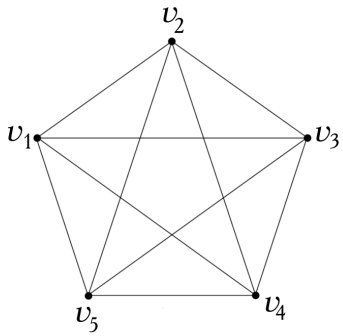



Рис. 3.165. Граф $K_5$

*Доведення*. Припустимо протилежне, що граф $K_5$ планарний. Оскільки він має цикл довжини 5, наприклад, $(v_1, v_2, v_3, v_4, v_5)$, то можна вважати, що при будь-якому укладанні цього графа на площину цей цикл зображується правильним п’ятикутником. За теоремою Жордана ребро $(v_1, v_3)$ має лежати або цілком усередині цього п’ятикутника, або цілком поза ним. Третю можливість (коли ребро має загальну точку з п’ятикутником) ми не розглядаємо, оскільки маємо справу з укладанням на площину.

1. Розглянемо спочатку випадок, коли $(v_1, v_3)$ лежить усередині п'ятикутника. Оскільки ребра $(v_2, v_4)$ і $(v_2, v_5)$ не мають перетинати ребро $(v_1, v_3)$, то обидва ці ребра лежать поза п'ятикутником. Ця ситуація зображена на рис. 3.166.

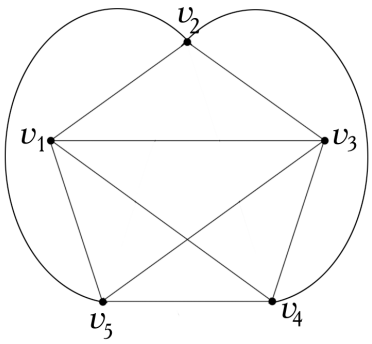



Рис. 3.166. Спроба уникнути перетину з ребром $(v_1, v_3)$

Ребро $(v_1, v_4)$ не має перетинати ребро $(v_2, v_5)$ і тому воно має лежати усередині п'ятикутника. Аналогічно ребро $(v_3, v_5)$ має лежати усередині п'ятикутника, оскільки не має перетинатися з ребром $(v_2, v_4)$. Однак ребра $(v_1, v_4)$ і $(v_3, v_5)$ обов'язково перетинаються, тому, згідно з теоремою Жордана, одне з них має лежати поза п'ятикутником.

Таким чином, ми приходимо до протиріччя з нашим припущенням. Наявність даного протиріччя доводить помилковість початкового припущення, тобто граф $K_5$ не є планарним.

Аналогічно доводять, що граф $K_{3,3}$ не є планарним, оскільки неможливо побудувати ребро $(v_1, v_4)$ без перетину з іншим ребром графа $K_{3,3}$ (рис. 3.167).

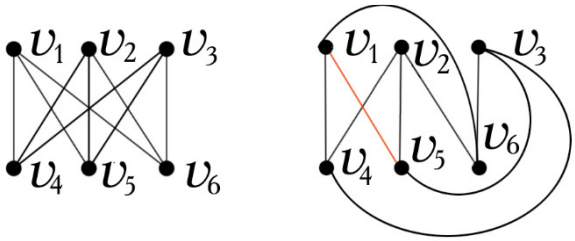



Рис. 3.167. Доведення того, що граф $K_{3,3}$ не є планарним

### 3.10.5. Грані плоского графа

Точку $x$ площини $S$, на якій розміщено граф $G$, називають *диз'юнктною* з $G$, якщо ця точка не відповідає жодній вершині графа $G$ і не лежить на жодному ребрі цього графа. Дві точки $x$ і $y$ площини $S$ називають еквівалентними, якщо вони диз'юнктні з $G$ і їх можна з'єднати такою жордановою кривою, усі точки якої диз'юнктні з $G$.

Введене відношення еквівалентності точок площини розбиває множину всіх точок площини $S$ на класи еквівалентності, які називають гранями графа

$G$. Наприклад, граф $G$, зображений на рис. 3.168, має 4 грані: $f_1, f_2, f_3, f_4$. Причому грань $f_1$ – нескінченна.

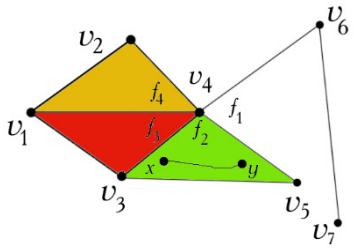



Рис. 3.168. Грані графа $G$

### 3.10.6. Теорема Ейлера

**Теорема 3.27.** Нехай $G$ – зв'язний плоский граф, у якому
$n$ – число вершин,
$m$ – число ребер
і $f$ – число граней.
Тоді справедливе співвідношення:

$$n + f = m + 2.$$

*Доведення.* План доведення ґрунтується на виконанні індукції за числом ребер у графі $G$.
1. Нехай $m = 0$, то $n = 1$
(оскільки за умовою теореми $G$ – зв'язний) і $f = 1$ (нескінченна грань).
У цьому випадку теорема вірна, оскільки $n + f = 1 + 1 = 2$ і $m + 2 = 0 + 2 = 2$.
Приклад даного графа показаний на рис. 3.169.

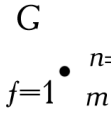



Рис. 3.169. Зв'язний граф $G$ з параметрами $n = 1, m = 0, f = 1$

2. Нехай граф $G$ представлений такими параметрами:
Ребро $e_1$ є петлею, і в цьому випадку число граней збільшується на одну, а число вершин залишається незмінним. Граф з однією вершиною і двома гранями показаний на рис. 3.170.

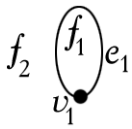



Рис. 3.170. Зв'язний граф $G$ з параметрами $n = 1, m = 1, f = 2$

Для розглянутого графа $G = (V, E)$ одержуємо

$$V = \{v_1\},\ E = \{e_1\},\ F = \{f_1, f_2\}.$$

Тому $n = |V| = 1$, $m = |E| = 1$, $f = |F| = 2$.
Звідси $n + f = 1 + 2 = 3$ і $m + 2 = 1 + 2 = 3$.

3. Нехай граф $G$ представлений такими параметрами:

ребро $e_1$ з'єднує дві різні вершини в $G$. Граф з двома вершинами і одним ребром показаний на рис. 3.171.

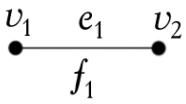



Рис. 3.171. Зв'язний граф $G$ з параметрами $n=2, m=1, f=1$

У даному графі $G=(V,E)$ одержуємо $V=\{v_1,v_2\}$, $E=\{e_1\}$, $F=\{f_1\}$.

Тому $n=|V|=1$, $m=|E|=1$, $f=|F|=1$.

Звідси $n+f=2+1=3$ і $m+2=1+2=3$.

4. Нехай зв'язний граф $G$ містить кілька вершин і ребер. Граф з трьома вершинами і трьома ребрами показаний на рис. 3.172.

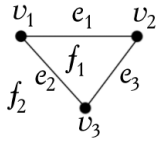



Рис. 3.172. Зв'язний граф $G$ з параметрами $n=3, m=3, f=2$

Граф $G=(V,E)$ характеризується параметрами

$V=\{v_1,v_2,v_3\}$, $E=\{e_1,e_2,e_3\}$, $F=\{f_1,f_2\}$.

Тому $n=|V|=3$, $m=|E|=3$, $f=|F|=2$.

Звідси $n+f=3+2=5$ і $m+2=3+2=5$.

5. До графа, показаного на рис. 3.172, додамо петлю. У результаті одержимо граф, показаний на рис. 3.173.

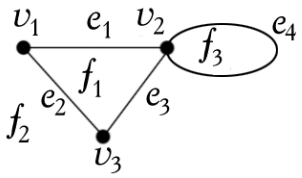



Рис. 3.173. Зв'язний граф $G$ з параметрами $n=3, m=3, f=3$

Граф $G=(V,E)$ характеризується параметрами

$$V=\{v_1,v_2,v_3\},\ E=\{e_1,e_2,e_3,e_4\},\ F=\{f_1,f_2,f_3\}$$

Тому $n=|V|=3$, $m=|E|=4$, $f=|F|=3$.

Отже, $n+f=3+3=6$ і $m+2=4+2=6$.

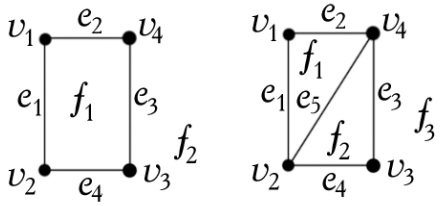
6. Нехай зв'язний граф $G_1$ містить 4 вершини і 4 ребра (рис. 3.174 а).

Даний граф має такі параметри:

$$V=\{v_1,v_2,v_3,v_4\},\ E=\{e_1,e_2,e_3,e_4\},\ F=\{f_1,f_2\}$$

Побудуємо граф $G_2$ (рис. 3.174 б), додавши ребро, що з'єднує дві його вершини.

$$V=\{v_1,v_2,v_3,v_4\},\ E=\{e_1,e_2,e_3,e_4,e_5\},\ F=\{f_1,f_2,f_3\}$$




а б

Рис. 3.174. Граф $G_1$ (а) і $G_2$ (б)

Параметри нового графа $n=4, m=5, f=3$.

Отже, $n+f=4+3=7$, $m+2=5+2=7$.

7. Якщо до зв'язного графа $G_2$ (рис. 3.174 б), який має такі параметри

$$V=\{v_1,v_2,v_3,v_4\},\ E=\{e_1,e_2,e_3,e_4,e_5\},\ F=\{f_1,f_2,f_3\}$$

додати ребро, інцидентне тільки з однією з його вершин, то одержимо граф з параметрами (рис. 3.175):

$$V=\{v_1,v_2,v_3,v_4,v_5\},\ E=\{e_1,e_2,e_3,e_4,e_5,e_6\},\ F=\{f_1,f_2,f_3\}$$

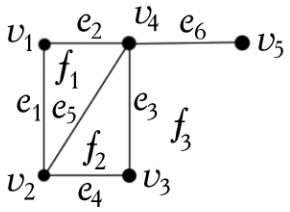



Рис. 3.175. Зв'язний граф $G$ з параметрами $n=5, m=6, f=3$

Одержуємо $n+f=5+3=8$, $m+2=6+2=8$.

**Наслідок.** Нехай $G$ – плоский граф з $n$ вершинами, $m$ ребрами, $f$ гранями і $k$ компонентами зв'язності; тоді

$$n+f=m+k+1$$

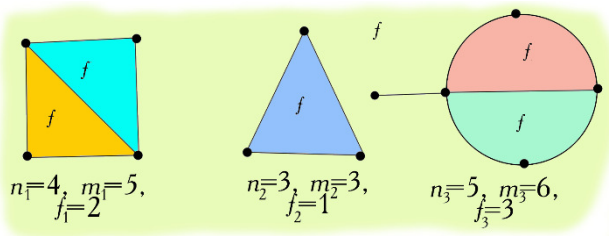



Рис. 3.176. Незв'язний граф $G$ з параметрами $n=12, m=14, f=6$

$$n=n_1+n_2+n_3=4+3+5=12$$

$$m = m_1 + m_2 + m_3 = 5 + 3 + 6 = 14$$
$$f = f_1 + f_2 + f_3 = 2 + 1 + 3 = 6$$
$$n + f = 12 + 6 = 18$$
$$m + k + 1 = 14 + 3 + 1 = 18$$

**Наслідок.** Якщо $G$ – зв'язний планарний граф (рис. 3.177) з $m$ ребрами і $n \geq 3$ вершинами, то $m \leq 3n - 6$.

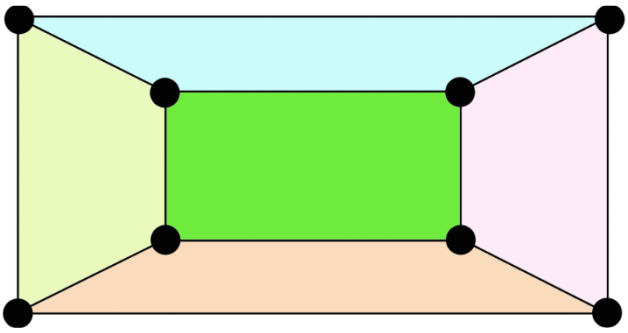


Рис. 3.177. Зв'язний граф з параметрами $n = 8, m = 12$

$$n = 8 > 3,\ m = 12$$
$$m < 3n - 6,\ m < 3 \cdot 8 - 6, m < 24 - 6, 121 < 18$$

**Наслідок.** У будь-якому планарному графі існує вершина, степінь якої не більше п'яти (рис. 3.178).

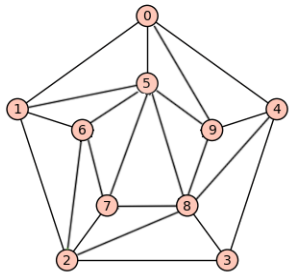



Рис. 3.178. Планарний граф

**Твердження.** Будь-який підграф планарного графа теж планарний.
**Твердження.** Якщо граф включає непланарний підграф, то і сам граф непланарний.

**Наслідок.** Будь-який підграф, що включає граф $K_5$ або $K_{3,3}$, не є планарним.

Виявляється, що графи $K_5$ (рис. 3.179) і $K_{3,3}$ (рис. 3.180) – **єдині непланарні** графи в тому розумінні, що будь-який непланарний граф включає один з них як підграф.

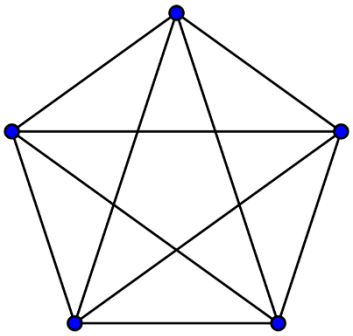


Рис. 3.179. Зв'язний граф $K_5$

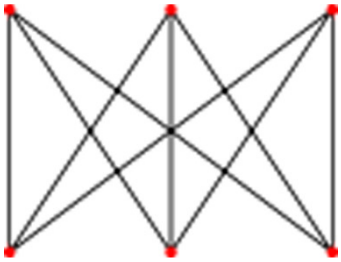


Рис. 3.180. Зв'язний граф $K_{3,3}$

Для формулювання цього результату введемо поняття *гомеоморфності графів*.

### 3.10.7. Гомеоморфні графи

**Визначення 1.** Два графа називають *гомеоморфними* або *тотожними* з точністю до вершин степеня 2, якщо вони можуть бути отримані з того самого графа за допомогою операції введення вершини степеня 2 у ребро.

Наприклад, графи, зображені на рис. 3.181, гомеоморфні.

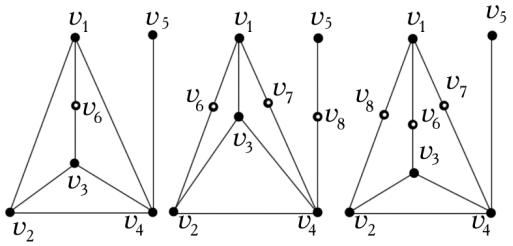



Рис. 3.181. Приклад гомеоморфних графів

Неважко переконатися, що гомеоморфізм є відношенням еквівалентності.

**Визначення 2.** Два графи називають гомеоморфними, якщо кожен з них може бути одержаний розбиттям певного графа.

**Визначення 3.** Два графа $G_1$ і $G_2$ є гомеоморфними, якщо можливо сформувати таке розбиття графа $G_1$ і розбиття графа $G_2$, які ізоморфні між собою.

**Визначення 4.** Розбиття графа – це граф, який отримуємо послідовним застосуванням до даного графа операції розбиття ребер.

При розбитті, ребро $e$, яке інцидентне вершинам $\{u,v\}$, замінюємо на вершину $w$ і два ребра $\{u,w\}$ і $\{w,v\}$ (рис. 3.182).

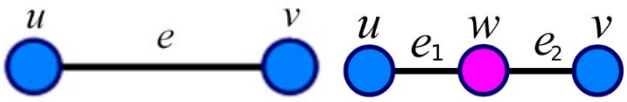



Рис. 3.182. Розбиття ребра $e$

### 3.10.8. Теорема Понтрягіна – Куратовського

Граф є планарним тоді і тільки тоді, коли він не має підграфів, гомеоморфних $K_5$ і $K_{3,3}$.

**Доведення.** З геометричної точки зору додавання вершини степеня 2 – це додавання точки на ребрі, а стирання такої вершини поєднує два ребра із спільним кінцем в одне. Очевидно, що кожна із цих операцій, застосована до плоского графа, знову дасть плоский граф. Отже, за наслідками з теореми Ейлера, жодний плоский (а, отже, і планарний) граф не гомеоморфний графам $K_5$ і $K_{3,3}$. З урахуванням зауваження про непланарні підграфи, теорема доведена.

### 3.10.9. Операція стягування

Нехай $(u,v)$ – ребро графа $G$. Вилучимо із графа $G$ вершини $u$ *і* $v$. Після цього додамо в граф $G$ нову вершину $w$ і з'єднаємо її ребрами з усіма вершинами, з якими була суміжна хоча б одна з вершин $u$ і $v$ (рис. 3.183).

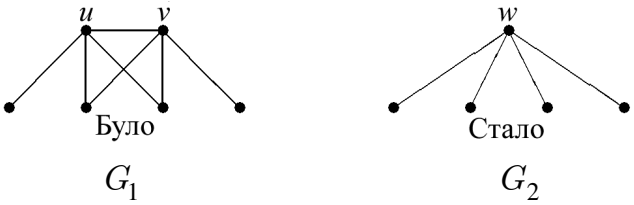



Рис. 3.183. Стягування графа $G_1$ до графа $G_2$

Позначимо отриманий граф через $G_2$. Говорять, що граф $G_2$ отримано з $G_1$ *стягуванням ребра* $(u,v)$.

Якщо скінченним (можливо нульовим) числом операцій стягування ребра із графа $G_1$ можна одержати граф $G_2$, то говорять, що $G_1$ *стягується* до $G_2$.

### 3.10.10. Теорема Вагнера

**Теорема.** Граф планарний тоді і тільки тоді, коли він не має підграфів, які стягуються графом $K_5$ і $K_{3,3}$.

**Доведення необхідності.** Якщо стягти ребро в плоскому графі, він залишиться плоским. Отже, за наслідками з теореми Ейлера, жоден плоский (а отже, і планарний) граф не стягується ні до графа $K_5$, ні до графа $K_{3,3}$. З урахуванням зауваження про непланарні підграфи, необхідність доведена.

Процес стягування до $K_{3,3}$ і $K_5$ (рис. 3.184).

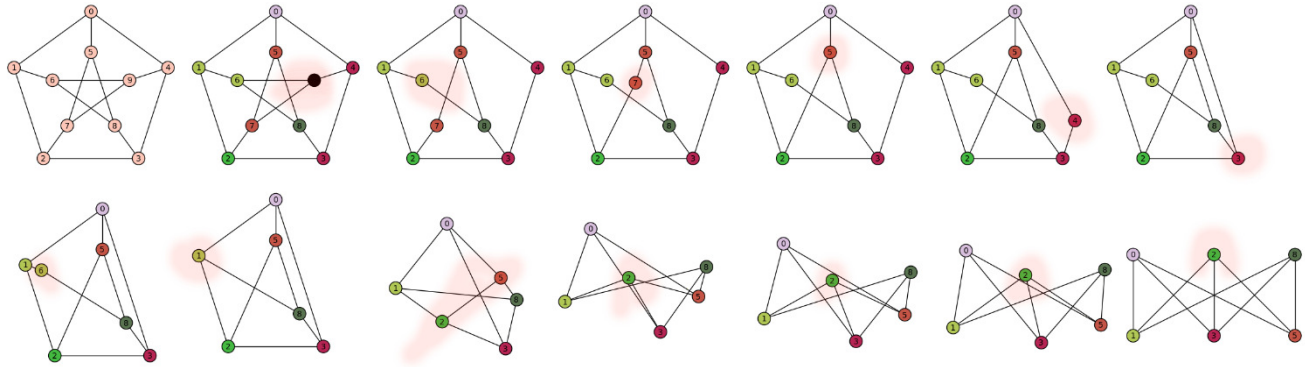


Рис. 3.184. Процес стягування

### Контрольні запитання

1. Коли існує шлях Ейлера в графі? Поясніть, чому граф для кенігсберзьких мостів не має шляху Ейлера.
2. Дайте визначення гамільтонового графа. Чи існує можливість перетворення негамільтонового графа в гамільтоновий?
3. Чи є повний граф з множиною вершин $V=\{v_1,v_2,v_3,v_4,v_5\}$ планарним графом? Якщо так, то виконайте його укладку на площині.
4. Перевірте, чи є планарним граф, який задано такою матрицею суміжності:

$$S=\begin{pmatrix} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 \\ v_1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ v_2 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ v_3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ v_4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ v_5 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_6 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ v_7 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ v_8 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ v_9 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Якщо даний граф планарний, то виконайте його укладку на площину.

5. Визначте число граней графа, заданого матрицею суміжності:

$$S = \begin{pmatrix} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ v_1 & 0 & 1 & 1 & 1 & 0 & 0 \\ v_2 & 1 & 0 & 1 & 1 & 1 & 1 \\ v_3 & 1 & 1 & 0 & 1 & 0 & 0 \\ v_4 & 1 & 1 & 1 & 0 & 1 & 1 \\ v_5 & 0 & 1 & 0 & 1 & 0 & 1 \\ v_6 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

6. Нехай дано граф, який містить 10 ребер і 6 граней. Визначте кількість вершин даного графа.