

# ТЕОРІЯ ГРАФІВ

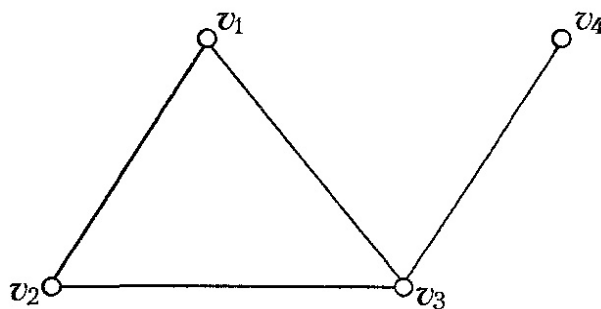
Теорія графів – одна з істотних частин математичного апарату інформатики та кібернетики. У термінах теорії графів можна сформулювати багато задач, пов'язаних із дискретними об'єктами. Такі задачі виникають у проектуванні інтегральних схем і схем управління, у дослідженні автоматів, в економіці й статистиці, теорії розкладів і дискретній оптимізації.

## Основні означення та властивості

Термін “граф” уперше з'явився в книзі видатного угорського математика Д. Кеніга 1936 р., хоча перші задачі теорії графів пов'язані ще з іменем Л. Ейлера (XVIII ст.).

*Простим графом* називають пару  $G = (V, E)$ , де  $V$  – непорожня скінченна множина елементів, названих *вершинами*,  $E$  – множина неупорядкованих пар різних елементів із  $V$ . Елементи множини  $E$  (неупорядковані пари різних вершин) називають *ребрами*.

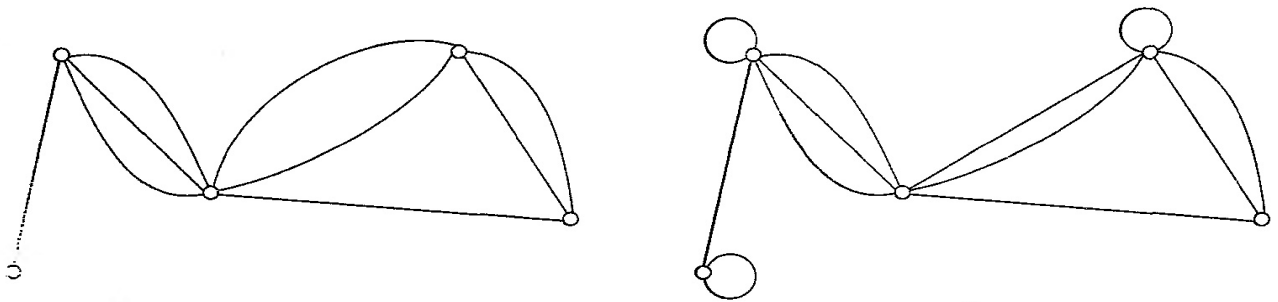
**Приклад.** На рисунку зображено простий граф  $G$  з множиною вершин  $V = \{v_1, v_2, v_3, v_4\}$  і множиною ребер  $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_3, v_4\}\}$ .



Говорять, що ребро  $\{u, v\}$  з'єднує вершини  $u$  та  $v$ . Оскільки  $E$  – множина, то з простому графі пару вершин може з'єднувати не більше ніж одне ребро. Іноді розглядають графи, у яких дві вершини можуть бути з'єднані більше ніж одним ребром. Так виникає поняття мультиграфа. *Мультиграфом* називають пару  $(V, E)$ , де  $V$  – скінченна непорожня множина вершин, а  $E$  –

сім'я неупорядкованих пар різних елементів множини  $V$ . Тут застосовано термін “сім'я” замість поняття “множина”, бо елементи в  $E$  (ребра) можуть повторюватись. Ребра, що з'єднують одну й ту саму пару вершин, називають *кратними* (або *паралельними*). Окрім кратних ребер розглядають також *петлі*, тобто ребра, які з'єднують вершину саму із собою. *Псевдографом* називають пару  $(V, E)$ , де  $V$  – скінченна непорожня множина вершин, а  $E$  – сім'я неупорядкованих пар не обов'язково різних вершин.

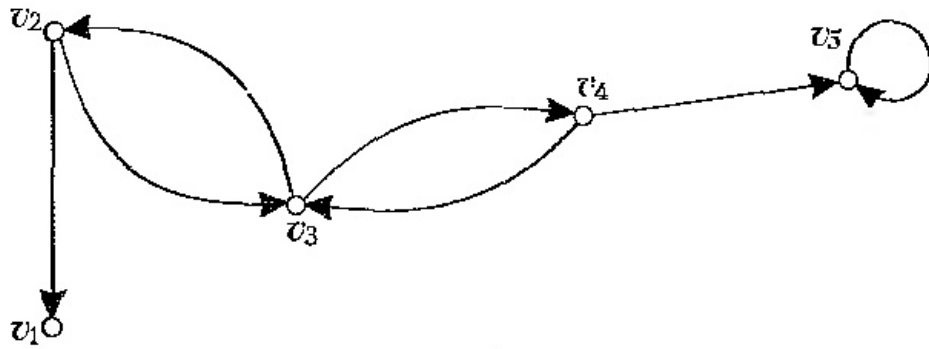
**Приклад.** На рисунку зображено мультиграф (зліва) і псевдограф (справа).



Розглянуті три типи графів називають *неорієнтованими*. Псевдограф – це найзагальніший тип неорієнтованого графа, бо він може містити петлі й кратні ребра. Мультиграф – це неорієнтований граф, який може містити кратні ребра, але не може містити петель. Нарешті, простий граф – це неорієнтований граф без кратних ребер і без петель.

Розглядають також орієнтовані графи. *Орієнтованим графом* називають пару  $(V, E)$ , де  $V$  – скінченна непорожня множина вершин, а  $E$  – множина впорядкованих пар елементів множини  $V$ . Елементи множини  $E$  в орієнтованому графі називають *дугами* (чи *орієнтованими ребрами*). Дугу  $(v, v)$  називають *петлею*.

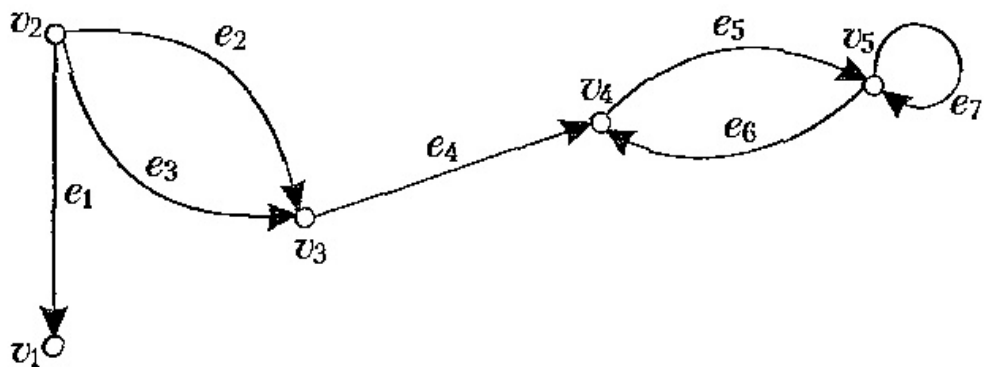
**Приклад.** На рисунку зображено орієнтований граф із множиною вершин  $V = \{v_1, v_2, v_3, v_4, v_5\}$  і множиною дуг  $E = \{(v_2, v_1), (v_2, v_3), (v_3, v_2), (v_3, v_4), (v_4, v_3), (v_4, v_5), (v_5, v_5)\}$



Зазначимо, що дуга – це впорядкована пара вершин (записують у круглих дужках), тому дуги  $(v_2, v_3)$  та  $(v_3, v_2)$  різні. На рисунках дуги позначають стрілками.

Орієнтованим мультиграфом називають пару  $(V, E)$ , де  $V$  – скінченна непорожня множина вершин, а  $E$  – сім'я впорядкованих пар елементів множини  $V$ . Отже, елементи (дуги) в  $E$  в разі орієнтованого мультиграфа можуть повторюватись; такі дуги називають *кратними*. Зауважимо, що кратні дуги з'єднують одну пару вершин і однаково напрямлені.

**Приклад.** На рисунку наведено приклад орієнтованого мультиграфа. Дуги  $e_2$  та  $e_3$  кратні, а дуги  $e_5$ ,  $e_6$  – ні.

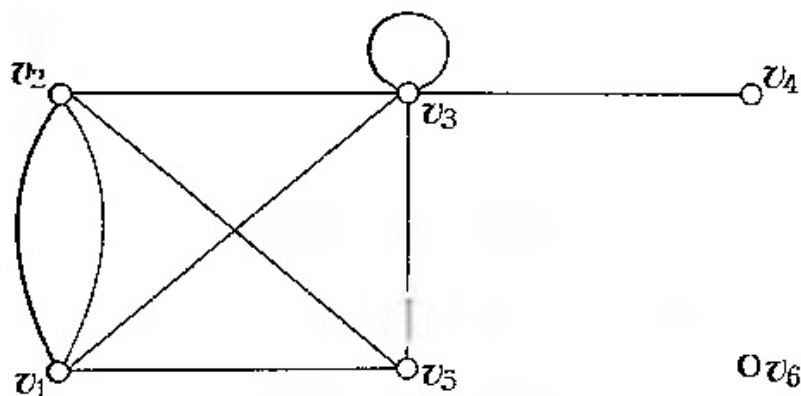


Надалі ми будемо використовувати термін “граф” для опису довільних графів – орієнтованих і неорієнтованих, із петлями та кратними ребрами чи без них, а термін “неорієнтований граф” або “псевдограф” – для довільного неорієнтованого графа, який може мати кратні ребра й петлі. Означення різних типів графів зведено в таблиці.

Тип графа	Редра	Чи дозволені кратні ребра?	Чи дозволені петлі?
Простий граф	Неорієнтовані	Ні	Ні
Мультиграф	Неорієнтовані	Так	Ні
Псевдограф	Неорієнтовані	Так	Так
Орієнтований граф	Орієнтовані (дуги)	Ні	Так
Орієнтований мультиграф	Орієнтовані (дуги)	Так	Так

Дві вершини  $u$  та  $v$  в неорієнтованому графі  $G$  називають *суміжними*, якщо  $\{u, v\}$  – ребро:  $\{u, v\} \in E$ . Якщо  $e = \{u, v\}$  – ребро, то вершини  $u$  та  $v$  називають його *кінцями*. Два ребра називають *суміжними*, якщо вони мають спільний кінець. Вершину  $v$  та ребро  $e$  називають *інцидентними*, якщо вершина  $v$  – кінець ребра  $e$ . Зазначимо, що суміжність – це зв'язок між однорідними елементами графа, а інцидентність – зв'язок між його різнорідними елементами. *Степінь вершини* в неорієнтованому графі – це кількість інцидентних їй ребер, причому петлю враховують двічі. Степінь вершини  $v$  позначають  $\deg(v)$ . Якщо  $\deg(v) = 0$ , то вершину  $v$  називають *ізолюваною*; якщо  $\deg(v) = 1$  – *висячою*, або *кінцевою*.

**Приклад.** У неорієнтованому графі на рисунку степені вершин такі:  $\deg(v_1) = 4$ ,  $\deg(v_2) = 4$ ,  $\deg(v_3) = 6$ ,  $\deg(v_4) = 1$ ,  $\deg(v_5) = 3$ ,  $\deg(v_6) = 0$ . Отже, вершина  $v_6$  – ізолювана, а  $v_4$  – висяча.



Зв'язок між степенями вершин неорієнтованого графа та кількістю його ребер дає така теорема.

**Теорема.** Нехай  $G = (V, E)$  – неорієнтований граф з  $m$  ребрами. Тоді

$$\sum_{v \in V} \deg(v) = 2m.$$

Це твердження стосується будь-якого неорієнтованого графа, зокрема з петлями та кратними ребрами.

**Доведення.** Додавання кожного нового ребра додає по одиниці до степенів двох вершин або двійку до степеня однієї вершини в разі петлі.

**Теорема.** Неорієнтований граф має парну кількість вершин непарного степеня.

**Доведення.** Позначимо як  $V_1$  множину вершин парного степеня, як  $V_2$  – непарного, а як  $m$  – кількість ребер графа. Тоді

$$2m = \sum_{v \in V} \deg(v) = \sum_{v \in V_1} \deg(v) + \sum_{v \in V_2} \deg(v).$$

Отже,  $\sum_{v \in V_2} \deg(v)$  – парне число. Сума непарних чисел парна тоді й лише тоді, коли кількість доданків парна.

Якщо в орієнтованому мультиграфі  $G = (V, E)$ ,  $(u, v) \in E$ , то вершину  $u$  називають *початковою* (ініціальною), а вершину  $v$  – *кінцевою* (термінальною) вершиною дуги  $e = (u, v)$ . Петля має початок і кінець в одній і тій самій вершині.

Для орієнтованого графа означення степеня вершини інше. В орієнтованому мультиграфі *напівстепенем входу* вершини  $v$  називають кількість дуг, для яких вершина  $v$  кінцева; позначають  $\deg^-(v)$ . *Напівстепенем виходу* вершини  $v$  називають кількість дуг, для яких вершина  $v$  початкова; позначають  $\deg^+(v)$ .

**Приклад.** Для графа, зображеного на рисунку, напівстепені вершин такі:  
 $\deg^-(v_1) = 0, \deg^+(v_1) = 1, \deg^-(v_2) = 2, \deg^+(v_2) = 0, \deg^-(v_3) = 1, \deg^+(v_3) = 2$ .



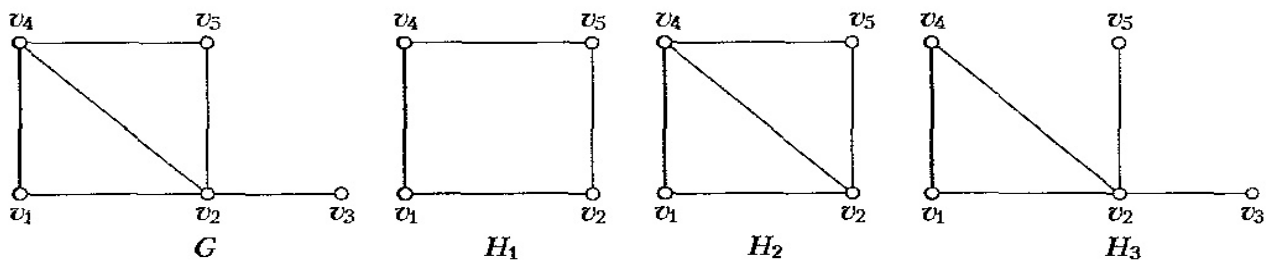
**Теорема.** Нехай  $G=(V,E)$  – орієнтований мультиграф, який має  $m$  дуг.

Тоді

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = m.$$

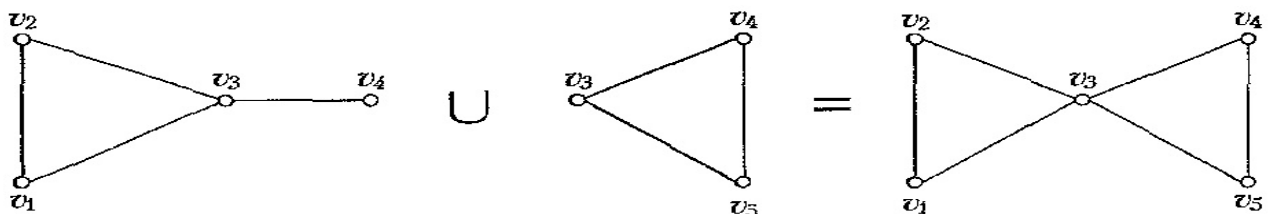
Простий граф  $H=(W,F)$  називають *підграфом* простого графа  $G=(V,E)$ , якщо  $W \subset V, F \subset E$ . Підграф  $H$  називають *каркасним* (або *фактором*), якщо  $W=V$ . Якщо  $W \neq V$ , а  $F$  – множина всіх ребер з  $E$ , які мають кінці в  $W$ , то підграф  $H$  називають *породженням* (або *індукованим*) *множиною*  $W$  і позначають як  $G(W)$ .

**Приклад.** На рисунку зображено граф  $G$  та три його підграфа  $H_1, H_2, H_3$ , серед яких  $H_2$  породжений, а  $H_3$  – каркасний.



Об'єднанням двох простих графів  $G_1=(V_1,E_1)$  та  $G_2=(V_2,E_2)$  називають такий простий граф  $G=(V,E)$  що  $V=V_1 \cup V_2, E=E_1 \cup E_2$ .

**Приклад.** На рисунку зображено приклад об'єднання двох простих графів. Знаки „ $\cup$ ” та „ $=$ ” тут мають символічний зміст.



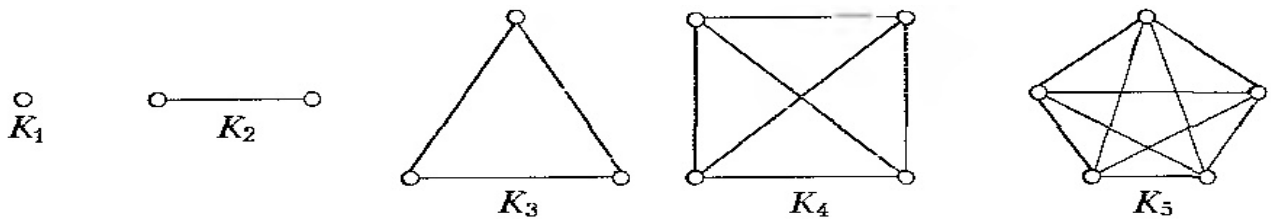
## Деякі спеціальні класи простих графів

Розглянемо деякі спеціальні класи простих графів, часто використовуваних як приклади й широко застосовуваних.

1. Повний граф з  $n$  вершинами (позначають як  $K_n$ ) - це граф, у якому будь-яку пару вершин з'єднано точно одним ребром.

Кількість ребер у графі  $K_n$  дорівнює  $C_n^2 = \frac{n(n-1)}{2}$ .

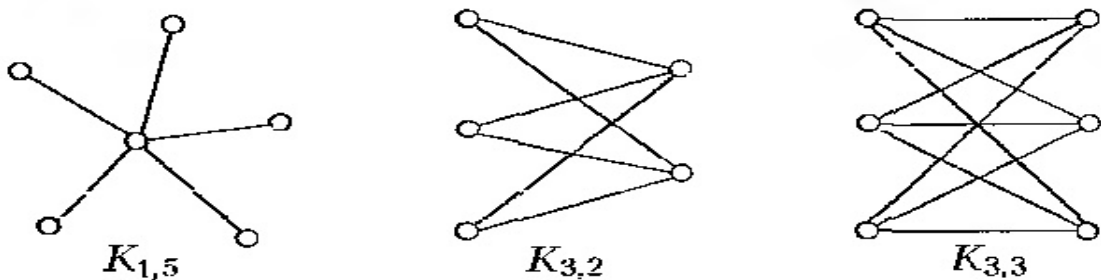
**Приклад.** На рисунку зображено графи  $K_n$  для  $n = 1, 2, 3, 4, 5$ .



2. Граф називають порожнім, якщо  $E = \emptyset$ , тобто такий граф не має ребер. Порожній граф з  $n$  вершинами позначають як  $O_n$ .

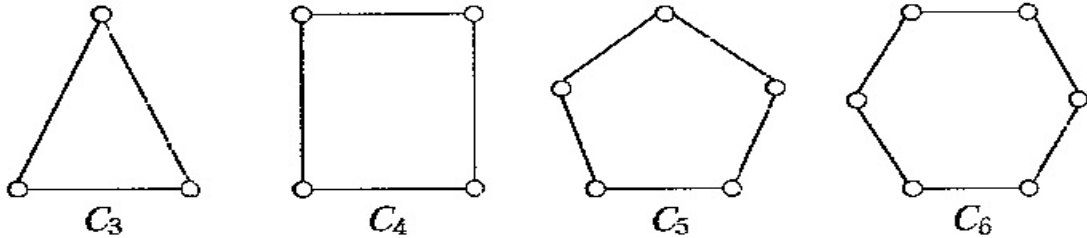
Граф  $G = (V, E)$  називають дводольним, якщо  $V$  можна розбити на дві підмножини  $V_1$  і  $V_2$ , що не перетинаються ( $V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset$ ), так, що кожне ребро з'єднує вершину з  $V_1$  і вершину з  $V_2$ . Дводольний граф називають повним, якщо кожну вершину з  $V_1$  з'єднано ребром із кожною вершиною з  $V_2$ . Повний дводольний граф позначають як  $K_{m,n}$ , де  $m = |V_1|$ ,  $n = |V_2|$ . Граф  $K_{1,n}$  називають зіркою. Граф  $K_{m,n}$   $n + m$  вершин та  $nm$  ребер.

**Приклад.** На рисунку наведено повні дводольні графи  $K_{1,5}$ ,  $K_{3,2}$  та  $K_{3,3}$ .



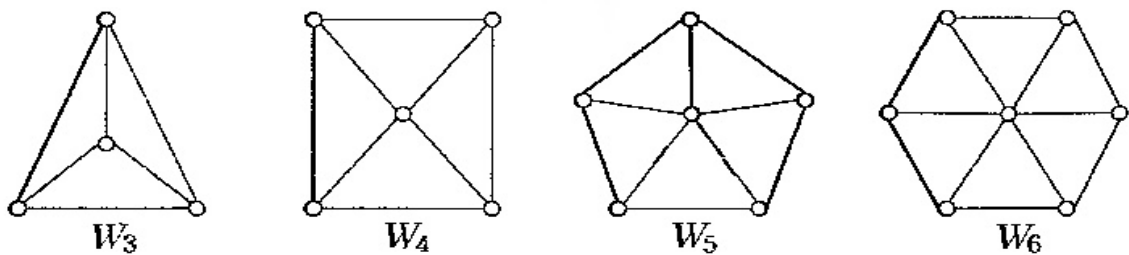
3. Циклом  $C_n, n \geq 3$ , називають граф із множиною вершин  $V = \{v_1, v_2, \dots, v_n\}$  і множиною ребер  $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$ .

**Приклад.** На рисунку зображено цикли  $C_3, C_4, C_5, C_6$ .



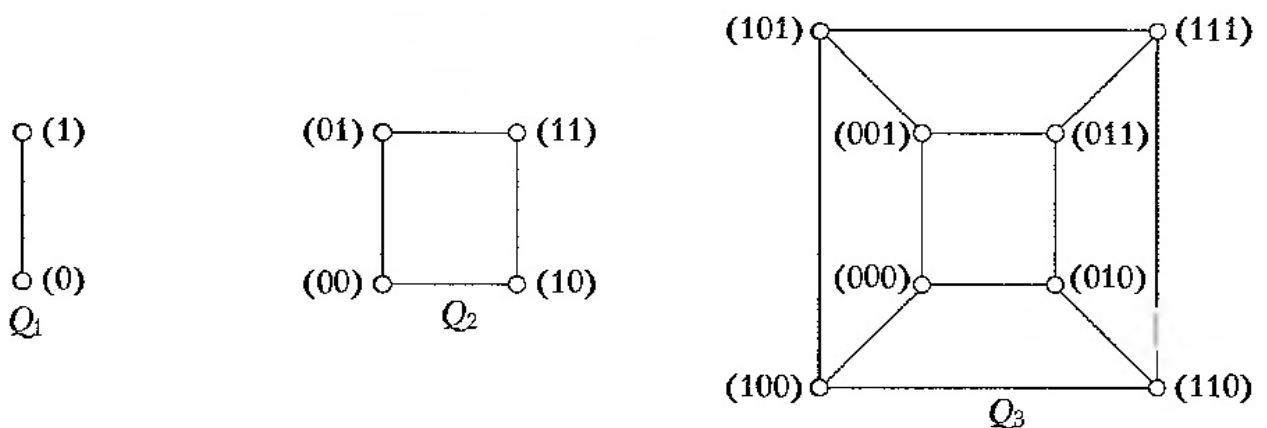
4. Колесом  $W_n$  називається граф, який можна одержати з циклу  $C_n$  додаванням ще однієї вершини, з'єднаної з усіма  $n$  вершинами в  $C_n$  новими ребрами.

**Приклад.** На рисунку зображено колеса  $W_3, W_4, W_5, W_6$ .



5. Простий граф, вершини якого зображають усі  $2^n$  бітові рядки довжиною  $n$ , називають  $n$ -вимірним кубом і позначають  $Q_n$ . Дві вершини в  $Q_n$  з'єднано ребром тоді й лише тоді, коли бітові рядки, які їх подають, відрізняються точно в одному біті.

**Приклад.** На рисунку зображено графи  $Q_1, Q_2$  та  $Q_3$ .



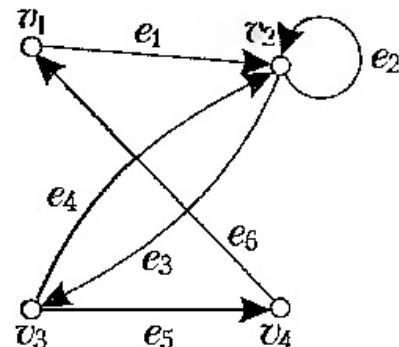
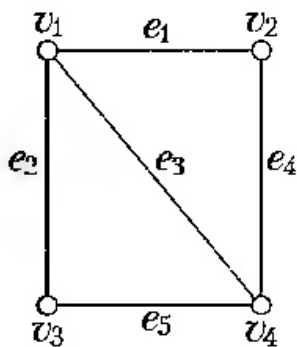


Граф  $Q_{n+1}$  можна отримати з двох графів  $Q_n$ , з'єднавши ребрами їхні однаково позначені вершини. Після цього до бітових рядків у вершинах одного з графів  $Q_n$  зліва дописують 0, а другого - 1.

### Способи подання графів

Найзрозуміліший і корисний для людини спосіб подання (зображення) графів – це рисунок на площині у вигляді точок і ліній, які з'єднують ці точки. Проте цей спосіб подання абсолютно непридатний, якщо потрібно розв'язувати на комп'ютері задачі з графами.

Розглянемо декілька інших способів подання графів для двох найважливіших типів графів: простого (лівий рисунок) й орієнтованого (правий рисунок).



Матрицю, кожний елемент якої дорівнює 0 чи 1, називають *булевою*.

#### *Матриця інцидентності*

Нехай  $G = (V, E)$  – простий граф із множиною вершин  $V = \{v_1, v_2, \dots, v_n\}$  і множиною ребер  $E = \{e_1, e_2, \dots, e_m\}$ .

*Матрицею інцидентності* графа  $G$ , яка відповідає заданій нумерації вершин і ребер, називають булеву  $n \times m$  матрицю  $M$  з елементами  $m_{ij}$  ( $i = 1, \dots, n; j = 1, \dots, m$ ), де

$$m_{ij} = \begin{cases} 1, & \text{якщо вершина } v_i \text{ та ребро } e_j \\ 0, & \text{у протилежному випадку.} \end{cases}$$

**Приклад.** Для графа, зображеного на рисунку зліва, матриця інцидентності має вигляд

$$\begin{matrix} & e_1 & e_2 & e_3 & e_4 & e_5 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{matrix}.$$

Отже, для простого графа в матриці інцидентності в кожному стовпці точно дві одиниці, і немає однакових стовпців. Матрицю інцидентності можна використовувати й для подання мультиграфа. Тоді з'являться однакові стовпці (вони відповідають кратним ребрам). Для подання псевдографа, якщо в ньому є петлі, у відповідних позиціях матриці ставимо 2 (у цьому разі матриця інцидентності не булева).

За допомогою матриці інцидентності можна подавати й орієнтовані графи. Для таких графів вона також не булева. Нехай  $G = (V, E)$  – орієнтований граф із множиною вершин  $V = \{v_1, v_2, \dots, v_n\}$  і множиною дуг  $E = \{e_1, e_2, \dots, e_m\}$ .

Матрицею інцидентності орієнтованого графа  $G$ , яка відповідає заданій нумерації вершин і дуг, називають  $n \times m$  матрицю  $M$  з елементами  $m_{ij}$  ( $i = 1, \dots, n; j = 1, \dots, m$ ), де

$$m_{ij} = \begin{cases} -1, & \text{якщо дуга } e_j \text{ входить у вершину } v_i, \\ 2, & \text{якщо дуга } e_j \text{ – це петля у вершині } v_i, \\ 1, & \text{якщо дуга } e_j \text{ виходить з вершини } v_i, \\ 0 & \text{в інших випадках.} \end{cases}$$

**Приклад.** Для графа, зображеного на рисунку справа, матриця інцидентності має вигляд

$$\begin{matrix} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 \\ -1 & 2 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \end{matrix}.$$

З алгоритмічного погляду матриця інцидентності – це, мабуть, найгірший

спосіб подання графа, який тільки можна собі уявити. По-перше, для цього потрібно  $nm$  комірок пам'яті, більшість із яких зайняті нулями. По-друге, доступ до інформації незручний. Щоб отримати відповідь на елементарні питання (наприклад, чи існує дуга  $(v_i, v_j)$ , до яких вершин ведуть дуги з  $v_i$ ), у найгіршому випадку потрібно перебрати всі стовпці матриці, тобто виконати  $m$  кроків.

### **Матриця суміжності**

Нехай  $G = (V, E)$  – простий граф,  $|V| = n$ . Припустимо, що вершини графа  $G$  занумеровані:  $v_1, v_2, \dots, v_n$ . Матрицею суміжності графа  $G$  (яка відповідає даній нумерації вершин) називають булеву  $n \times n$  матрицю  $A$  з елементами  $a_{ij}(i, j = 1, \dots, n)$ , де

$$a_{ij} = \begin{cases} 1, \text{ якщо } \{v_i, v_j\} \in E, \\ 0 \text{ в протилежному випадку.} \end{cases}$$

**Приклад.** Матриця суміжності для неорієнтованого графа, зображеного на рисунку, має вигляд

$$\begin{array}{c} v_1 \quad v_2 \quad v_3 \quad v_4 \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}.$$

Цілком очевидно, що для неорієнтованого графа  $a_{ij} = a_{ji}$ , тобто матриця суміжності симетрична. Більше того, позаяк у простому графі немає петель, то для нього з матриці суміжності  $a_{ii} = 0$  ( $i = 1, \dots, n$ ).

Матрицю суміжності можна використовувати також для подання псевдографа. Тоді це не булева матриця: елемент  $a_{ij}$  дорівнює кількості ребер, що з'єднують  $v_i$  та  $v_j$ . Петлю у вершині  $v_i$  подають значенням діагонального елемента  $a_{ii} = 1$ .

Для подання орієнтованих графів також використовують матрицю суміжності. Це булева  $n \times n$  матриця  $A$  з елементами  $a_{ij}(i, j = 1, \dots, n)$ , де

$$a_{ij} = \begin{cases} 1, \text{ якщо } (v_i, v_j) \in E, \\ 0 \text{ в протилежному випадку.} \end{cases}$$

**Приклад.** Матриця суміжності для орієнтованого графа, зображеного на рисунку, має вигляд

$$\begin{matrix} & v_1 & v_2 & v_3 & v_4 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{matrix}.$$

Зазначимо, що матриця суміжності орієнтованого графа, загалом кажучи, несиметрична.

Матрицю суміжності можна використовувати й для подання орієнтованого мультиграфа. У такому разі це не булева матриця: елемент  $a_{ij}$  дорівнює кількості дуг, які мають  $v_i$  початковою вершиною, а  $v_j$  – кінцевою.

Велика перевага матриці суміжності як способу подання графа – швидкий доступ до інформації: за один крок можна одержати відповідь на питання, чи існує ребро (дуга) з  $v_i$  у  $v_j$ . Вада полягає в тому, що незалежно від кількості ребер обсяг пам'яті становить  $n^2$  комірок. Як ще один аргумент проти використання матриці суміжності можна навести теорему про кількість кроків алгоритму, який на основі матриці суміжності перевіряє якусь властивість простого графа.

Нарешті, розглянемо ще два способи подання графів у пам'яті комп'ютера. Будь-яку скінченну послідовність довільних елементів будемо називати *списком*, а кількість елементів списку – його *довжиною*.

### ***Подання графа списком пар (списком ребер)***

Метод зображення графа списком пар, які відповідають його ребрам (або дугам), значно економніший щодо пам'яті, особливо якщо  $m$  (кількість ребер) значно менша, ніж  $n^2$  ( $n$  – кількість вершин). Пара  $[u, v]$  відповідає ребру  $\{u, v\}$ , якщо граф неорієнтований, і дузі  $(u, v)$ , якщо граф орієнтований. Для графів, зображених на рисунку, списки пар будуть такими:

$v_1$	$v_2$
$v_1$	$v_3$
$v_1$	$v_4$
$v_2$	$v_4$
$v_3$	$v_4$

$v_1$	$v_2$
$v_2$	$v_2$
$v_2$	$v_3$
$v_3$	$v_2$
$v_3$	$v_4$
$v_4$	$v_1$

Очевидно, що обсяг пам'яті в цьому разі дорівнює  $2m$  ( $m$  – кількість ребер або дуг). Це найекономніший щодо пам'яті спосіб. Його вада – велика (порядку  $m$ ) кількість кроків для знаходження множини вершин, до яких ідуть ребра чи дуги із заданої вершини. Ситуацію можна значно поліпшити, упорядкувавши множину пар лексикографічно та застосувавши двійковий пошук.

### ***Подання графа списками суміжності***

Орієнтований граф  $G$  (без кратних дуг, але, можливо, з петлями) можна подати зазначивши скінченну непорожню множину вершин  $V$  та відповідність  $\Gamma$ , котра. показує, як зв'язані між собою вершини. Відповідність  $\Gamma$  – багатозначне відображення множини  $V$  у  $V$ . Граф у такому разі позначають парою  $G = (V, \Gamma)$ . У літературі часто означають (орієнтований) граф саме в таких поняттях.

**Приклад.** Для орієнтованого графа, зображеного на рисунку, відповідність  $\Gamma$  подано таблиці.

Таблиця

$v$	$\Gamma(v)$ (термінальні вершини)
$v_1$	$v_2$
$v_2$	$v_2, v_3$
$v_3$	$v_2, v_4$
$v_4$	$v_1$

Цей спосіб подання можна використовувати й для неорієнтованих простих графів, якщо кожне ребро умовно замінити двома протилежно

спрямованими дугами.

**Приклад.** Для простого графа, зображеного на рисунку відповідність  $\Gamma$  задано таблицею.

Таблиця

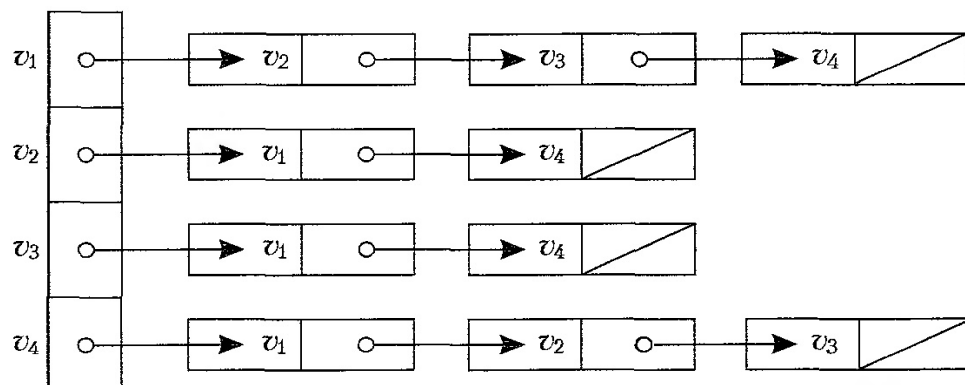
$v$	$\Gamma(v)$ (суміжні з $v$ вершини)
$v_1$	$v_2, v_3, v_4$
$v_2$	$v_1, v_4$
$v_3$	$v_1, v_4$
$v_4$	$v_1, v_2, v_3$

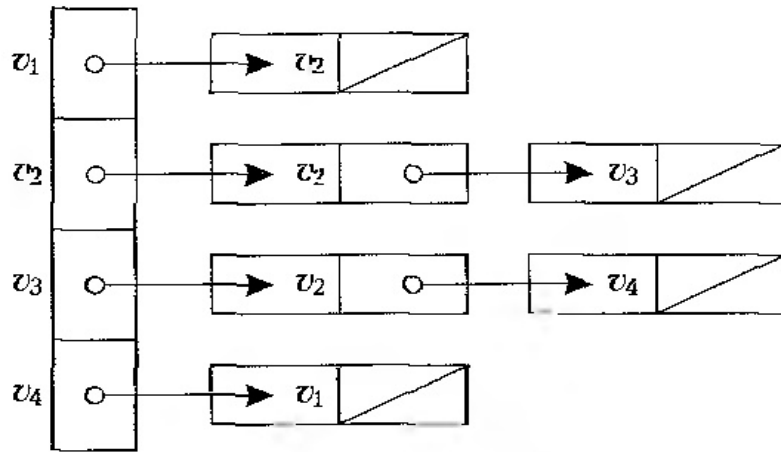
Якщо відображення  $\Gamma$  діє не на одну вершину, а на множину вершин  $A = \{v_{i_1}, v_{i_2}, \dots, v_{i_p}\}$ , то під  $\Gamma(A)$  розуміють об'єднання множин

$$\Gamma(A) = \Gamma(v_{i_1}) \cup \Gamma(v_{i_2}) \cup \dots \cup \Gamma(v_{i_p}).$$

Розглянемо спосіб комп'ютерного подання графа *списками суміжності*. Для цього використовують масив  $Adj$  із  $|V| = n$  списків – по одному на кожну вершину. Для кожної вершини  $u \in V$  список  $Adj[u]$  містить у довільному порядку покажчики на всі вершини множини  $\Gamma(u)$ .

**Приклад.** Далі на рисунку подано списки суміжності для неорієнтованого та орієнтованого графів.





Для орієнтованого графа сума довжин усіх списків суміжних вершин дорівнює загальній кількості дуг: дузі  $(u, v)$  відповідає елемент  $v$  зі списку  $Adj[u]$ . Для неорієнтованого графа ця сума дорівнює подвоєній кількості ребер, бо ребро  $\{u, v\}$  породжує елемент у списку суміжних вершин як для вершини  $u$ , так і для вершини  $v$ .

### Шляхи та цикли. Зв'язність

*Шляхом* довжиною  $r$  із вершини  $u$  в вершину  $v$  в неорієнтованому графі називають послідовність ребер  $e_1 = \{x_0, x_1\}, e_2 = \{x_1, x_2\}, \dots, e_r = \{x_{r-1}, x_r\}$ , де  $x_0 = u$ ,  $x_r = v$ ,  $r$  - натуральне число. Отже, шлях довжиною  $r$  має  $r$  ребер, причому ребро враховують стільки разів, скільки воно міститься в шляху. Вершини  $u$  та  $v$  називають *крайніми*, а решту вершин шляху - *внутрішніми*.

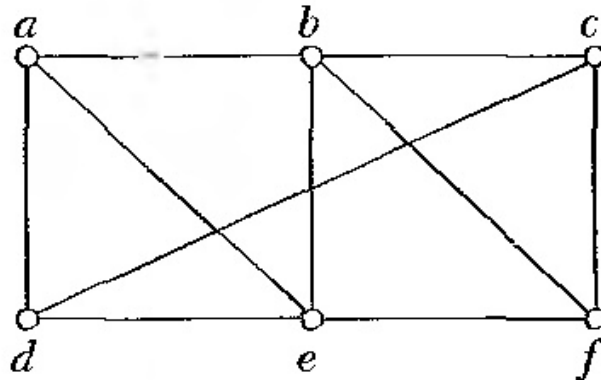
*Циклом* у неорієнтованому графі називають шлях, який з'єднує вершину саму з собою, тобто  $u = v$ .

У простому графі шлях можна задати послідовністю вершин, через які він проходить:  $x_0, x_1, x_2, \dots, x_{r-1}, x_r$ .

Шлях або цикл називають *простим*, якщо він не містить повторюваних ребер.

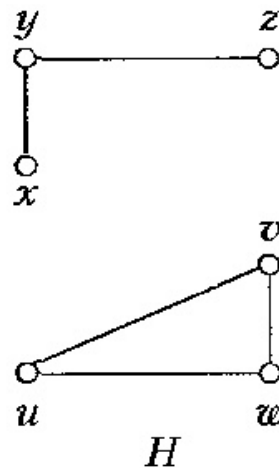
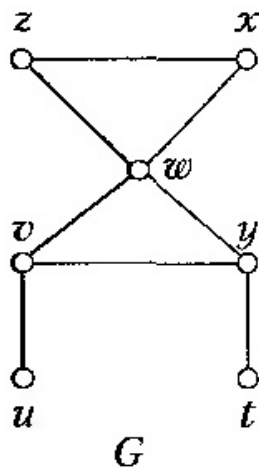
Говорять, що шлях із крайніми вершинами  $u$  та  $v$  з'єднує ці вершини. Шлях, що з'єднує вершини  $u$  та  $v$ , позначають як  $\langle u, v \rangle$  та називають  $\langle u, v \rangle$ -*шляхом*.

**Приклад.** Нижче на рисунку зображено простий граф. У ньому  $a, d, c, f, e$  - простий шлях довжиною 4, оскільки пари  $\{a, d\}, \{d, c\}, \{c, f\}$  і  $\{f, e\}$  - ребра. Однак  $d, e, c, b$  - не шлях, бо пара  $\{e, c\}$  - не ребро. Зазначимо, що  $b, c, f, e, b$  - цикл довжиною 4, позаяк  $\{b, c\}, \{c, f\}, \{f, e\}$  і  $\{e, b\}$  - ребра та цей шлях починається й закінчується в одній і тій самій вершині  $b$ . Шлях  $a, b, e, d, a, b$ , довжина якого дорівнює 5, не простий, тому що він двічі проходить через ребро  $\{a, b\}$ .



Неорієнтований граф називають *зв'язним*, якщо будь-які дві його вершини з'єднані шляхом. Граф називають *незв'язним*, якщо він не є зв'язним. Незв'язний граф складається з двох або більше зв'язних підграфів, кожна пара з яких не має спільних вершин. Ці зв'язні підграфи називають *компонентами зв'язності* чи просто *компонентами* графа.

**Приклад.** Граф  $G$  на рисунку зв'язний; граф  $H$  - незв'язний, оскільки не існує шляху  $\langle u, v \rangle$ . Граф  $H$  має дві компоненти.



Віддаллю  $d(u, v)$  між вершинами  $u$  та  $v$  називають довжину найкоротшого



$\langle u, v \rangle$ -шляху, а сам цей шлях називають *геодезичним*. Легко переконатись, що найкоротший шлях не містить повторюваних вершин (і ребер). Звідси випливає така теорема.

**Теорема.** Між кожною парою різних вершин зв'язного неорієнтованого графа існує простий шлях.

Для орієнтованого графа вводять поняття *орієнтованого шляху* (або просто *шляху*) з вершини  $u$  у вершину  $v$ . Це скінченна послідовність дуг  $e_1 = \{x_0, x_1\}, e_2 = \{x_1, x_2\}, \dots, e_r = \{x_{r-1}, x_r\}$ , де  $x_0 = u$ ,  $x_r = v$ . Вершини  $u$  та  $v$ , як і в неорієнтованому графі, називають *крайніми*, а решту вершин шляху - *внутрішніми*. *Довжиною* шляху називають кількість дуг, з яких він складається. *Орієнтованим циклом* називають орієнтований шлях, який з'єднує вершину саму із собою, тобто  $u = v$ .

Орієнтований шлях або цикл називають *простим*, якщо жодна дуга не міститься в ньому більше одного разу.

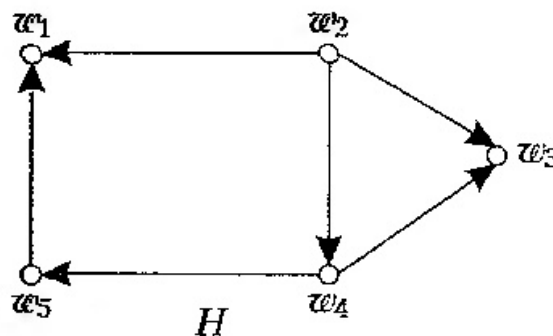
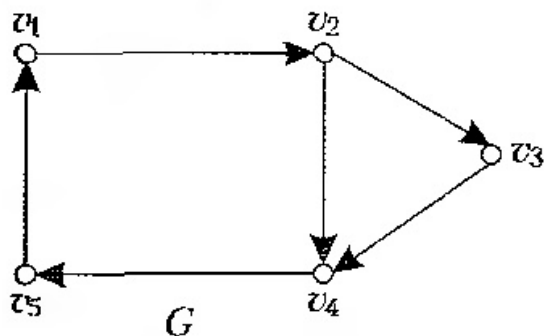
Для орієнтованого графа поняття зв'язності вводять по-різному, залежно від того, чи враховано напрямки.

Орієнтований граф називають *сильно зв'язним*, якщо для будь-яких його різних вершин  $u$  та  $v$  існують орієнтовані шляхи від  $u$  до  $v$  та від  $v$  до  $u$ . Отже, для сильної зв'язності орієнтованого графа має існувати послідовність дуг з урахуванням орієнтації від будь-якої вершини графа до будь-якої іншої.

Орієнтований граф може не бути сильно зв'язним, але може бути, так би мовити, „в одному цілому”. У зв'язку з цим дамо таке означення. Орієнтований граф називають *слабко зв'язним*, якщо існує шлях між будь-якими двома різними вершинами у відповідному йому неорієнтованому графі (тобто без урахування напрямку дуг).

Зрозуміло, що сильно зв'язний граф водночас і слабо зв'язний.

**Приклад.** На рисунку зображено графи  $G$  й  $H$ . Граф  $G$  сильно зв'язний. Граф  $H$  слабо зв'язний; він не сильно зв'язний, бо не існує орієнтованого шляху від  $w_1$  до  $w_2$ .



Кількість різних шляхів між двома довільними вершинами графа можна підрахувати за допомогою матриці суміжності.

**Теорема.** Нехай  $G$  - граф (орієнтований або неорієнтований; можуть бути також кратні ребра й петлі),  $A$  - його матриця суміжності, яка відповідає заданій нумерації вершин  $v_1, v_2, \dots, v_n$ . Тоді кількість різних шляхів довжиною  $r$  ( $r$  - натуральне) з вершини  $v_i$  у вершину  $v_j$  дорівнює  $(i, j)$ -му елементу матриці  $A^r$ .

**Доведення.** (методом математичної індукції за  $r$ ). Для  $r = 1$  твердження теореми очевидне: кількість шляхів від  $v_i$  до  $v_j$  довжиною 1 дорівнює  $(i, j)$ -му елементу матриці  $A$ , бо цей елемент дорівнює кількості ребер (дуг), які з'єднують  $v_i$  та  $v_j$ .

Припустимо, що  $(i, j)$ -й елемент матриці  $A^r$  дорівнює кількості різних шляхів довжиною  $r$  від вершини  $v_i$  до вершини  $v_j$ . Це індуктивна гіпотеза

Оскільки  $A^{r+1} = A^r A$ , то  $(i, j)$ -й елемент матриці  $A^{r+1}$  дорівнює  $\sum_{k=1}^n b_{ik} a_{kj}$ , де

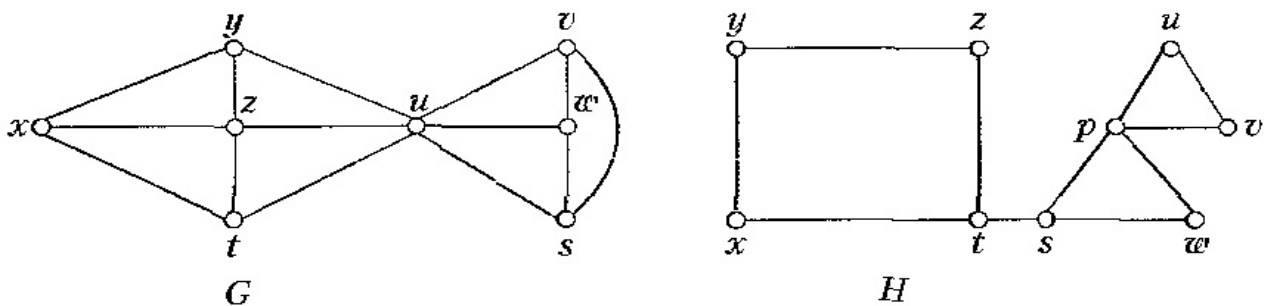
$b_{ik}$  -  $(i, k)$ -й елемент матриці  $A^r$ ,  $a_{kj}$  -  $(k, j)$ -й елемент матриці суміжності  $A$ .

За індуктивною гіпотезою  $b_{ik}$  дорівнює кількості шляхів довжиною  $r$  із вершини  $v_i$  у вершину  $v_k$ . Шлях довжиною  $r + 1$  із  $v_i$  у  $v_j$  складається зі шляху довжиною  $r$  із  $v_i$  до якоїсь проміжної вершини  $v_k$  та ребра (дуги) з  $v_k$  до  $v_j$ . За правилом добутку з комбінаторики кількість таких шляхів дорівнює добутку кількості  $b_{ik}$  шляхів довжиною  $r$  із  $v_i$  до  $v_k$  та кількості  $a_{kj}$  ребер (дуг) із  $v_k$  до  $v_j$  тобто  $b_{ik} a_{kj}$ . Якщо ці добутки підсумувати для всіх можливих проміжних вершин  $v_k$ , то потрібний результат випливає з комбінаторного правила суми.

Розглянемо неорієнтовані графи  $K_n$  і  $C_n$ . Обидва ці графи зв'язні, проте інтуїтивно зрозуміло, що для  $n > 3$  граф  $K_n$  „сильніше зв'язаний”, ніж граф  $C_n$ . Розглянемо два поняття, які характеризують міру зв'язності простого графа.

*Числом вершинної зв'язності* (або просто *числом зв'язності*)  $k(G)$  простого графа  $G$  називають найменшу кількість вершин, видалення яких дає незв'язний або одновіршинний граф. Зазначимо, що вершину видалають разом із інцидентними їй ребрами. Наприклад,  $k(K_1) = 0$ ,  $k(K_n) = n - 1$ ,  $k(C_n) = 2$ . Граф  $G$ , зображений на рисунку, зв'язний, але його зв'язність можна порушити видаленням вершини  $u$ . Отже,  $k(G) = 1$ . Якщо ж спробувати порушити зв'язність цього графа видаленням ребер (а не вершин), то потрібно видалити не менше ніж три ребра.

Нехай  $G$  - простий граф з  $n > 1$  вершинами. *Числом реберної зв'язності*  $\lambda(G)$  графа  $G$  називають найменшу кількість ребер, видалення яких дає незв'язний граф. Число реберної зв'язності одновіршинного графа вважають таким, що дорівнює 0. Для графа  $G$ , зображеного на рисунку,  $\lambda(G) = 3$ .



Вершину  $u$  простого графа  $G$  називають *точкою з'єднання*, якщо граф  $G$  в разі її видалення матиме більше компонент, ніж даний граф  $G$ . Зокрема, зв'язний граф  $G$  без вершини  $u$  стає незв'язним. Нагадаємо, що вершину  $u$  при цьому видалають разом з інцидентними їй ребрами. Ребро графа  $G$  називають *мостом*, якщо його видалення збільшує кількість компонент. Отже, точки з'єднання й мости - це своєрідні „вузькі місця” простого графа.

**Приклад.** Граф  $H$ , зображений на рисунку, має три точки з'єднання  $t$ ,  $s$ ,  $p$  й один міст  $\{t, s\}$ .

Позначимо як  $\delta(G)$  мінімальний степінь вершин графа  $G$ . Можна довести, що  $k(G) \leq \lambda(G) \leq \delta(G)$ . Простий граф  $G$  називають  $t$ -зв'язним, якщо  $k(G) \geq t$ , і реберно  $t$ -зв'язним, якщо  $\lambda(G) \geq t$ . Отже, відмінний від  $K_1$  граф однозв'язний тоді й лише тоді, коли він зв'язний. Двозв'язні графи - це зв'язні графи без точок з'єднання, відмінні від графа  $K_1$ .

**Приклад.** Граф  $G$ , зображений на рисунку, однозв'язний і реберно 3-зв'язний.

Очевидно, що кількість ребер у зв'язному простому графі з  $n$  вершинами не перевищує кількості ребер у графі  $K_n$  тобто  $n(n-1)/2$ . Але скільки може бути ребер у простому графі з  $n$  вершинами й фіксованою кількістю компонент?

**Теорема.** Якщо простий граф  $G$  має  $n$  вершин і  $k$  компонент, то кількість  $m$  його ребер задовольняє нерівності

$$n - k \leq m \leq \frac{1}{2}(n - k)(n - k + 1).$$

**Доведення.** Доведемо спочатку верхню оцінку. Нехай  $G$  - простий граф з  $n$  вершинами,  $k$  компонентами й максимальною для таких графів кількістю ребер  $m_{\max}$ . Очевидно, що кожна компонента графа  $G$  - повний граф. Нехай  $K_p$ ,  $K_q$  - дві компоненти,  $p \geq q > 1$ ,  $v$  - вершина з другої компоненти. Вилучимо з графа всі ребра, інцидентні вершині  $v$ , і з'єднаємо цю вершину ребром із кожною вершиною з першої компоненти. Кількість вершин і компонент при цьому не зміниться, а кількість ребер зросте на  $p - (q - 1) = p - q + 1 > 1$ , що неможливо, бо граф  $G$  має максимально можливу кількість ребер. Отже, лише одна компонента графа  $G$  являє собою повний граф із більшою ніж 1 кількістю вершин  $n - (k - 1) = n - k + 1$ . Отже,  $m_{\max} = (1/2)(n - k)(n - k + 1)$ .

Доведемо нижню оцінку математичною індукцією за кількістю ребер  $m$ . Для  $m = 0$  твердження очевидне, оскільки тоді  $k = n$  і, отже,  $0 \leq 0$ . Нехай тепер  $m > 0$ , і нижня оцінка справджується для графів із меншою кількістю ребер, ніж  $m$ . Припустимо, що граф  $G$  має найменшу можливу кількість ребер  $m_{\min}$  серед

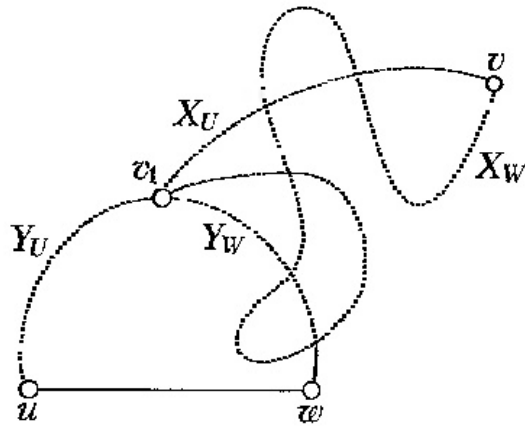
усіх простих графів з  $n$  вершинами й  $k$  компонентами. Вилучивши довільне ребро, отримаємо граф з  $n$  вершинами,  $k + 1$  компонентою й  $m_{\min} - 1$  ребром. Для нього справджується припущення індукції:  $n - (k - 1) \leq m_{\min} - 1$ , звідки випливає нерівність  $n - k \leq m_{\min}$ .

Д. Кеніг сформулював простий критерій дводольності графа в термінах довжин простих циклів.

**Теорема (Кеніга, 1936 р.).** Для того, щоб граф  $G$  був дводольним, необхідно й достатньо, щоб він не містив простих циклів із непарною довжиною.

**Доведення. Необхідність.** Нехай  $G$  - дводольний граф,  $C$  - один із його простих циклів довжиною  $k$ . Пройдемо всі ребра цього циклу, починаючи з вершини  $v$ . Зробивши  $k$  кроків, повернемося у вершину  $v$ . Оскільки кінці кожного ребра містяться в різних підмножинах вершин, то  $k$  - парне число.

**Достатність.** Нехай зв'язний граф  $G = (V, E)$  з  $n > 1$  вершинами не має простих циклів із непарною довжиною та  $v \in V$ . Побудуємо розбиття  $V = A \cup B$  ( $A \cap B = \emptyset$ ) так. Довільну вершину  $x \in V$  долучимо до множини  $A$ , якщо віддаль  $d(x, v)$  парна, а ні, то до множини  $B$ . Залишилося довести, що породжені підграфи  $G(A)$  та  $G(B)$  порожні. Припустимо, що це не так, тобто існують дві суміжні вершини  $u$  та  $w$ , які належать одній множині. Тоді жодна з цих вершин не збігається з  $v$ , бо  $v \in A$ , а всі вершини, суміжні з  $v$ , належать множині  $B$ . Нехай  $U = \langle u, v \rangle$  та  $W = \langle w, v \rangle$  - геодезичні шляхи,  $v_1$  - остання (якщо починати від  $v$ ) зі спільних вершин цих шляхів (див. рисунок). Позначимо як  $X_U$  й  $Y_U$ , відповідно частини шляху  $U$  від  $v$  до  $v_1$  і від  $v_1$  до  $u$ . Аналогічно, як  $X_W$  та  $Y_W$  позначимо відповідно частини шляху  $W$  від  $v$  до  $v_1$  і від  $v_1$  до  $w$ . Очевидно, що довжини шляхів  $X_U$  й  $X_W$  збігаються (тому ці шляхи геодезичні). Отже, довжини шляхів  $Y_U$  й  $Y_W$  мають один тип парності (позаяк вершини  $u$  та  $w$  належать одній множині, то їх віддалі від вершини  $v$  мають один тип парності). Але тоді об'єднання шляхів  $Y_U$  й  $Y_W$  та ребра  $\{u, w\}$  являє собою простий цикл із непарною довжиною. Суперечність.



Доведення теореми Кеніга підказує простий спосіб розпізнавання дводольності графа, що ґрунтується на простому алгоритмі, названому *пошуком ушир*. Множину вершин, суміжних із вершиною  $v$ , називають *оточенням* вершини  $v$ . Пошук ушир нумерує вершини графа. Починають із довільної вершини, надають їй номер 0. Кожній вершині з оточення вершини 0 приписують номер 1. Тепер розглядають по чергово оточення всіх вершин із номером 1, і всім вершинам, що належать цим оточенням і ще не мають номера, надають номер 2. Розглядають оточення всіх вершин із номером 2 та продовжують процес нумерації, доки це можливо.

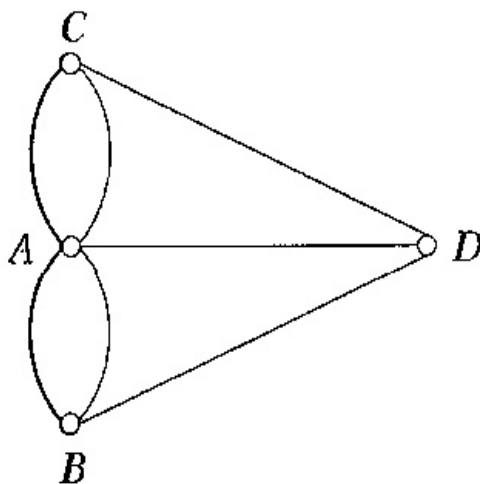
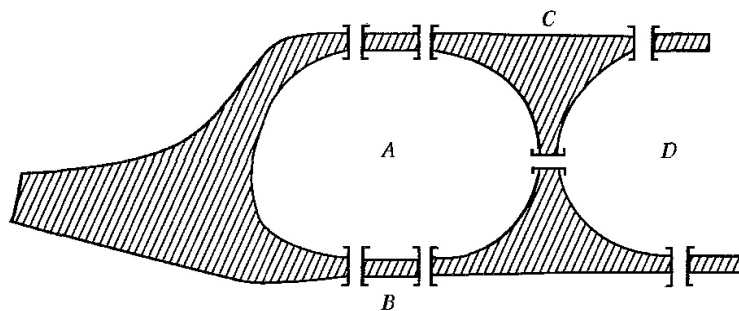
Якщо даний граф  $G = (V, E)$  зв'язний, то пошук ушир занумерує всі його вершини.

Далі розіб'ємо множину вершин  $V$  на дві підмножини -  $A$  та  $B$ . До множини  $A$  долучимо всі вершини з парними номерами (та 0), а до множини  $B$  - з непарними. Розглянемо породжені підграфи  $G(A)$  та  $G(B)$ . Якщо обидва вони порожні (достатньо перевірити, що всі пари вершин з однаковими номерами не суміжні), то  $G$  - дводольний граф, а ні, то - не дводольний.

Існує й інший, більш розповсюджений варіант пошуку вшир. Він відрізняється тим, що всі вершини отримують різні номери.

## Ейлерів цикл у графі

Початок теорії графів як розділу математики пов'язують із задачею про кенігсберзькі мости. Сім мостів міста Кенігсберга (нині – Калінінград у Росії) було розміщено на річці Прегель так, як зображено на рисунку. Чи можна, починаючи з якоїсь точки міста, пройти через усі мости точно по одному разу й повернутись у початкову точку? Швейцарський математик Л. Ейлер розв'язав цю задачу. Його розв'язання, опубліковане 1736 р., було першим явним застосуванням теорії графів. Ейлер поставив у відповідність плану міста мультиграф  $G$ , вершини якого відповідають чотирьом частинам  $A$ ,  $B$ ,  $C$ ,  $D$  міста, а ребра – мостам. Цей мультиграф зображено на рисунку.

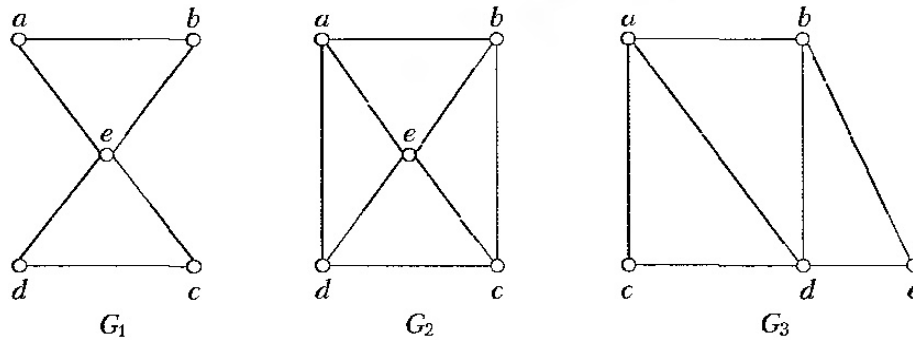


Отже, задачу про кенігсберзькі мости мовою теорії графів можна сформулювати так: чи існує в мультиграфі  $G$  простий цикл, який містить усі ребра цього мультиграфа? Ейлер довів нерозв'язність задачі про кенігсберзькі мости. Нагадаємо, що в простому циклі ребра не повторюються, а вершини можуть повторюватись.

*Ейлеровим циклом у зв'язному мультиграфі  $G$  називають простий цикл,*

який містить усі ребра графа, *ейлеровим шляхом* – простий шлях, то містить усі ребра графа.

**Приклад.** На рисунку проілюстровано концепцію ейлерових циклів і шляхів. Граф  $G_1$  має ейлерів цикл, наприклад,  $a, e, c, d, e, b, a$ ; граф  $G_3$  не має ейлерового циклу, але має ейлерів шлях:  $a, c, d, e, b, d, a, b$ ; граф  $G_2$  не має ні ейлерового циклу, ні ейлерового шляху.



Існує простий критерій (необхідна й достатня умова) для виявлення того, чи має граф ейлерів цикл.

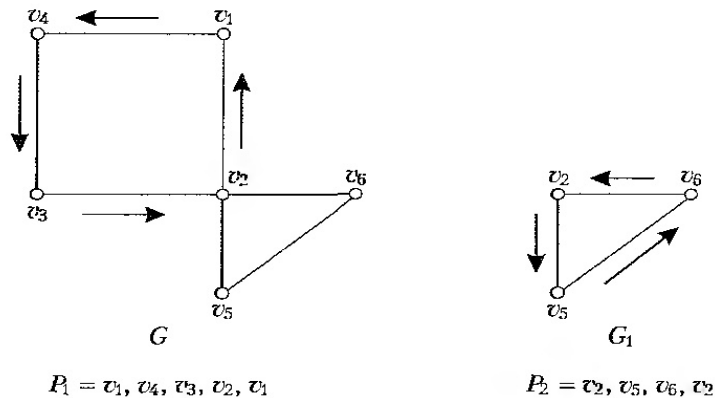
**Теорема.** Зв'язний мультиграф  $G$  має ейлерів цикл тоді й лише тоді, коли степені всіх його вершин парні.

**Доведення. Необхідність.** Нехай у графі  $G$  існує ейлерів цикл. Тоді він проходить через кожну вершину графа та входить до неї по одному ребру, а виходить по іншому. Це означає, що кожна вершина інцидентна парній кількості ребер ейлерового циклу. Оскільки такий цикл містить усі ребра графа  $G$ , то звідси випливає парність степенів усіх його вершин.

**Достатність.** Припустимо тепер, що всі вершини графа  $G$  мають парний степінь. Почнемо шлях  $P_1$  із довільної вершини  $v_1$  і продовжимо його, наскільки це можливо, вибираючи щоразу нове ребро. Позаяк степені всіх вершин парні, то, увійшовши в будь-яку вершину, відмінну від  $v_1$ , ми завжди маємо можливість вийти з неї через іще не пройдене ребро. Тому шлях  $P_1$  можна продовжити, додавши це ребро. Отже, побудова шляху  $P_1$  завершиться у вершині  $v_1$ , тобто  $P_1$  обов'язково виявиться циклом. Якщо з'ясується, що шлях  $P_1$  містить усі ребра графа  $G$ , то це ейлерів цикл. У протилежному випадку вилучимо з  $G$  всі ребра циклу  $P_1$  і всі вершини, інцидентні лише вилученим



ребрам. Отримаємо якийсь зв'язний граф  $G_1$ . Оскільки  $P_1$  та  $G$  мають вершини лише парних степенів, то, очевидно, і граф  $G_1$ , матиме цю властивість. Окрім того, позаяк граф  $G$  зв'язний, то графи  $P_1$  і  $G_1$  мають принаймні одну спільну вершину  $v_2$ . Тепер із вершини  $v_2$  побудуємо цикл  $P_2$  в графі  $G_1$  аналогічно до того, як ми будували цикл  $P_1$  у графі  $G$ . Цикл  $P_2$  вставимо в цикл  $P_1$  на місце вершини  $v_2$ . Одержимо цикл  $P_3$ . Описані побудови показано на рисунку.



Цикл  $P_3 = v_1, v_4, v_3, P_2, v_1 = v_1, v_4, v_3, v_2, v_5, v_6, v_2, v_1$  є ейлерів. Якби він виявився не ейлеровим, то потрібно продовжити аналогічні побудови й отримати ще більший цикл. Цей процес закінчиться побудовою ейлерового циклу. Зазначимо, що доведення достатності має конструктивний характер; подано алгоритм побудови ейлерового циклу.

Існує й інший алгоритм побудови ейлерового циклу, який дає змогу побудувати цей цикл одразу. Це алгоритм Флері (Fleury).

**Алгоритм Флері побудови ейлерового циклу.** Робота алгоритму полягає в нумерації ребер у процесі побудови ейлерового циклу.

Крок 1. *Початок.* Починаємо з довільної вершини  $u$  та присвоюємо довільному ребру  $\{u, v\}$  номер 1. Викреслюємо ребро  $\{u, v\}$  й переходимо у вершину  $v$ .

Крок 2. *Ітерація.* Нехай  $v$  – вершина, у яку ми перейшли на попередньому кроці,  $k$  – останній присвоєний номер. Вибраємо довільне ребро, інцидентне вершині  $v$ , причому міст вибираємо лише тоді, коли немає інших можливостей. Присвоюємо вибраному ребру номер  $(k+1)$  і викреслюємо

його.

Крок 3. *Закінчення.* Цей процес закінчуємо, коли всі ребра графа викреслено та пронумеровано – ці номери задають послідовність ребер в ейлеровому циклі.

Повертаючись до задачі про кенігсберзькі мости, виявляємо, що мультиграф, зображений на рисунку, має всі вершини непарного степеня. Отже, цей мультиграф не має ейлерового циклу, тому неможливо пройти кожний міст по одному разу й повернутись у початкову точку шляху.

**Теорема.** Зв'язний мультиграф має ейлерів шлях, але не має ейлерового циклу тоді й лише тоді, коли він має точно дві вершини непарного степеня.

Зазначимо, що будь-який ейлерів шлях починається в одній із цих двох вершин непарного степеня, а закінчується в іншій. Оскільки мультиграф для кенігсберзьких мостів має чотири вершини з непарними степенями, можна дійти висновку про неможливість пройти кожний міст по одному разу, навіть якщо не потрібно повертатись у початкову точку.

Граф, який має ейлерів цикл, часто називають *ейлеровим*.

*Ейлеровим циклом* у слабо зв'язному орієнтованому мультиграфі називають орієнтований простий цикл, який містить усі ребра графа.

**Теорема.** Орієнтований слабо зв'язний мультиграф має ейлерів цикл тоді й лише тоді, коли напівстепінь входу кожної вершини дорівнює її напівстепеню виходу.

### Гамільтонів цикл у графі

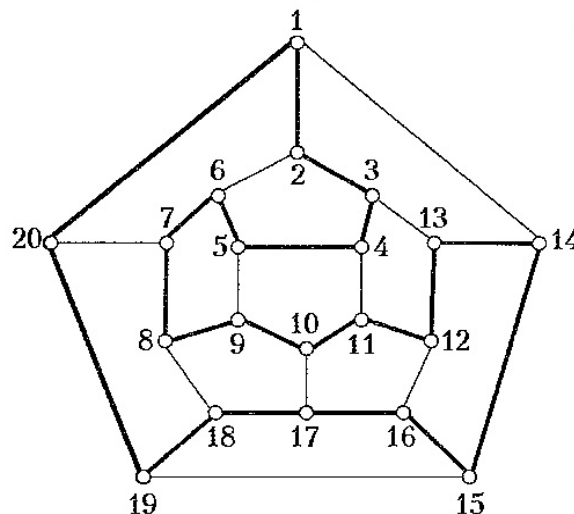
Шлях  $x_0, x_1, \dots, x_{n-1}, x_n$  у зв'язному графі  $G = (V, E)$  називають *гамільтоновим шляхом*, якщо  $V = \{x_0, x_1, \dots, x_{n-1}, x_n\}$  і  $x_i \neq x_j$  для  $0 \leq i < j \leq n$ .

Цикл  $x_0, x_1, \dots, x_{n-1}, x_n, x_0$  (тут  $n > 1$ ) у графі  $G = (V, E)$  називають *гамільтоновим циклом*, якщо  $x_0, x_1, \dots, x_{n-1}, x_n$  - гамільтонів шлях. Інакше кажучи, гамільтонів цикл і гамільтонів шлях проходять через кожну вершину графа точно один раз

(можливо, окрім першої й останньої вершин). Зазначимо, що гамільтонові цикл і шлях, узагалі кажучи, не містять усіх ребер графа.

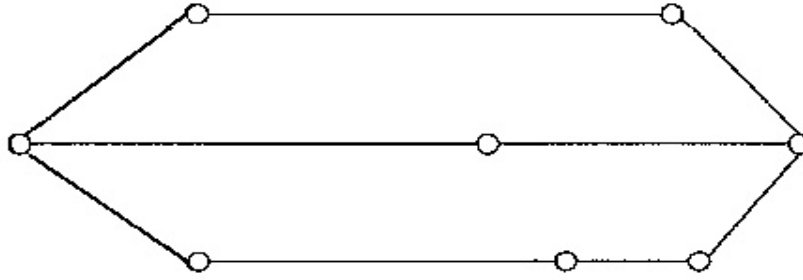
Термін „гамільтонів" у цих означеннях походить від імені відомого ірландського математика У. Гамільтона (W. Hamilton), який 1857 р. запропонував гру „Навколосвітня подорож". Кожній із двадцяти вершин додекаедра (правильного дванадцятигранника, грані якого - п'ятикутники) приписано назву одного з великих міст світу. Потрібно, розпочавши з довільного міста, відвідати решту 19 міст точно один раз і повернутись у початкове місто. Перехід дозволено ребрами додекаедра.

**Приклад.** Ту саму задачу можна зобразити й на площині (рисунок). Вона зводиться до відшукування в графі гамільтонового циклу. Один із можливих розв'язків показано потовщеними лініями.



Не всі зв'язні графи мають гамільтонів цикл хоча б тому, що такий граф має бути двозв'язним (тобто граф, який має точки з'єднання, не може мати гамільтонового циклу). Приклад графа, зображеного на рисунку, свідчить, що двозв'язності недостатньо для наявності гамільтонового циклу.

Незважаючи на зовнішню подібність формулювань задач про існування ейлерового й гамільтонового циклів, ці задачі принципово різні. Використовуючи результати попереднього підрозділа, легко виявити, чи має граф ейлерів цикл, і, якщо має, то побудувати його.



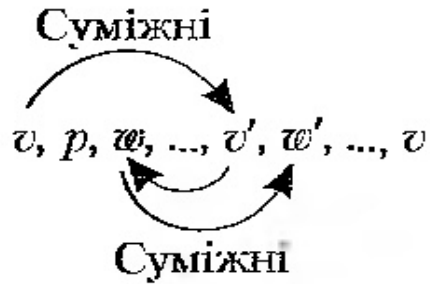
Ситуація для гамільтонового циклу істотно інша. Відповісти на питання, чи має граф гамільтонів цикл, зазвичай, дуже важко. Вивчення достатніх умов наявності в графі гамільтонового циклу - один із важливих напрямів у теорії графів. Інтуїтивно зрозуміло, що граф із багатьма ребрами, достатньо рівномірно розподіленими, з великою ймовірністю має гамільтонів цикл. Доведемо одну з теорем такого типу.

**Теорема (Г. Дірака, 1952 р.).** Якщо для кожної вершини  $v$  зв'язного простого графа з  $n \geq 3$  вершинами виконується нерівність  $\deg(v) \geq n/2$ , то цей граф має гамільтонів цикл.

**Доведення.** Додамо до графа  $G$   $k$  нових вершин і з'єднаємо ребром кожну з них із кожною вершиною з  $G$ . Отриманий граф з  $n + k$  вершинами позначимо як  $G'$ .

Уважатимемо, що  $k$  - найменша кількість вершин, потрібних для того, щоб у графі  $G$  з'явився гамільтонів цикл. Доведемо, що припущення  $k \geq 1$  призводить до суперечності.

Нехай  $v, p, w, \dots, v$  - гамільтонів цикл у графі  $G'$ , де  $v$  та  $w$  - вершини з  $G$ , а  $p$  - одна з нових вершин. Тоді вершина  $w$  несуміжна з  $v$ , інакше ми могли б не використовувати вершину  $p$ , що суперечить мінімальності числа  $k$ . Більше того, вершина  $w'$ , суміжна з вершиною  $w$ , не може в гамільтоновому циклі безпосередньо йти за вершиною  $v'$ , суміжною з  $v$ . Справді, якщо є гамільтонів цикл  $v, p, w, \dots, v', w', \dots, v$ , то ми можемо замінити його на  $v, v', \dots, w, w', \dots, v$  повернувши частину циклу між  $w$  та  $v'$ . Це знову суперечить мінімальності числа  $k$  (див. рисунок).



Отже, кількість вершин графа  $G'$ , не суміжних з  $w$ , не менша від кількості вершин, суміжних з  $v$  (тобто дорівнює принаймні  $n/2 + k$ ). Натомість очевидно, що кількість вершин графа  $G'$ , суміжних з  $w$ , також дорівнює принаймні  $n/2 + k$ . Але жодна з вершин графа  $G'$  не може бути одночасно суміжною й несуміжною з вершиною  $w$ , тому загальна кількість вершин графа  $G'$  дорівнює щонайменше  $n + 2k$ . Але це суперечить тому, що кількість вершин графа  $G'$  дорівнює  $n + k$ .

Як знайти гамільтонів цикл або переконатись, що його немає? Очевидний алгоритм, який можна застосувати, - це повний перебір усіх можливостей, тобто  $n!$  перестановок усіх вершин графа й перевірок. Зрозуміло, що такий спосіб не практичний.

Задача знаходження гамільтонів циклу  $NP$ -повна. Для таких задач невідомий ефективний (тобто з поліноміальною складністю) алгоритм їх розв'язання, і є вагомі підстави вважати, що його не існує.

Граф, який містить гамільтонів цикл, часто називають *гамільтонівим графом*.

## Обхід графів

Існує багато алгоритмів на графах, які ґрунтуються на систематичному переборі їх вершин або обході вершин, під час якого кожна вершина одержує унікальний порядковий номер. Алгоритм обходу вершин графа називають *методами пошуку*.

### Пошук углиб у простому зв'язному графі

Опишемо метод пошуку в простому зв'язному графі, який являє собою одну з основних методик проектування алгоритмів, пов'язаних із графами. Цей метод називають *пошуком углиб*, або DFS-методом (від англ. Depth First Search).

Нехай  $G = (V, E)$  – простий зв'язний граф, усі вершини якого позначено попарно різними символами. У процесі пошуку углиб вершинам графа  $G$  надають номери (DFS-номери) та певним способом позначають ребра. У ході роботи алгоритму використовують структуру даних для збереження множин, яку називають *стеком* (англ. stack – стіг). Зі стеку можна вилучити тільки той елемент, котрий було додано до нього останнім: стек працює за принципом “останнім прийшов – першим вийшов” (англ. Last In, First Out – скорочено LIFO). Інакше кажучи, додавання й вилучення елементів у стеку відбувається з одного кінця, який називають *верхівкою стеку*. DFS-номер вершини  $x$  позначають  $DFS(x)$ .

#### *Алгоритм пошуку углиб у простому зв'язному графі*

Наведемо кроки алгоритму.

Крок 1. Почати з довільної вершини  $v_s$ . Виконати  $DFS(v_s) = 1$ . Включити цю вершину в стек.

Крок 2. Розглянути вершину у верхівці стеку: нехай це вершина  $x$ . Якщо всі ребра, інцидентні вершині  $x$ , позначено, то перейти до кроку 4, інакше – до кроку 3.

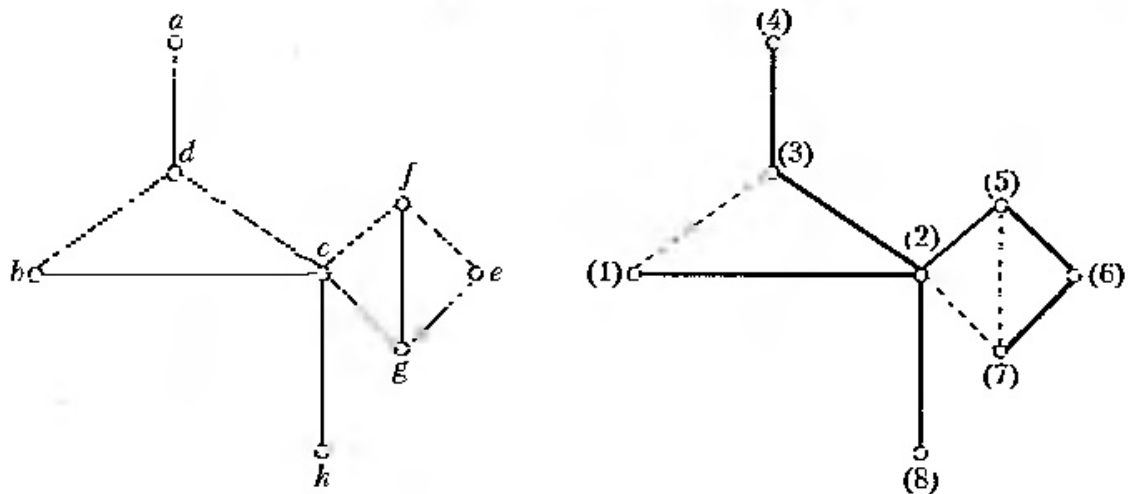
Крок 3. Нехай  $\{x, y\}$  – непозначене ребро. Якщо  $DFS(y)$  уже визначено, то позначити ребро  $\{x, y\}$  штриховою лінією та перейти до кроку 2. Якщо  $DFS(y)$  не визначено, то позначити ребро  $\{x, y\}$  потовщеною суцільною лінією,

визначити  $DFS(y)$  як черговий DFS-номер, включити цю вершину в стек і перейти до кроку 2.

Крок 4. Виключити вершину  $x$  зі стеку. Якщо стек порожній, то зупинитись, інакше – перейти до кроку 2.

Щоб вибір номерів був однозначним, доцільно домовитись, що вершини, суміжні з тією, яка вже отримала DFS-номер, аналізують за зростанням їх порядкових номерів (або в алфавітному порядку). Динаміку роботи алгоритму зручно відображати за допомогою таблиці з трьома стовпцями: вершина, DFS-номер, уміст стеку. Її називають *протоком обходу* графа пошуком вглиб.

**Приклад.** Виконаємо обхід графа на рисунку зліва пошуком углиб, починаючи з вершини  $b$ . Розв’язок подано на рисунку справа; протокол такого пошуку подано в таблиці. У цій таблиці в третьому стовпці вважаємо, що верхівка стеку праворуч.



Таблиця

Вершина	DFS-номер	Вміст стеку
$b$	1	$b$
$c$	2	$bc$
$d$	3	$bcd$
$a$	4	$bcda$
—	—	$bcd$
—	—	$bc$
$f$	5	$bcf$
$e$	6	$bcfe$
$g$	7	$bcfeg$

—	—	<i>bcfe</i>
—	—	<i>bcf</i>
—	—	<i>bc</i>
<i>h</i>	8	<i>bch</i>
—	—	<i>bc</i>
—	—	<i>b</i>
—	—	$\emptyset$

### Пошук ушир у простому зв'язному графі

У процесі пошуку *вшир* вершини графа пропадають в іншій послідовності, ніж у методі пошуку *вглиб*, і їм надають BFS-номери (від англ. Breadth First Search). BFS-номер вершини  $x$  позначають  $BFS(x)$ . Під час пошуку рухаються *вшир*, а не *вглиб*; спочатку проглядають усі сусідці вершини, після цього – сусіди сусідів і так далі.

У ході реалізації алгоритму використовують структуру даних для збереження множин, яку називають *чергою* (англ. queue). Із черги можна вилучити тільки той елемент, який перебував у ній найдовше: працює принцип “першим прийшов – першим вийшов” (англ. First In, First Out – скорочено FIFO). Елемент включається у *хвіст* черги, а виключається з її *голови*. Пошук *ушир*, узагалі кажучи відрізняється від пошуку *вглиб* заміною стеку на чергу. Після такої модифікації що раніше відвідується вершина (включається в чергу), то раніше вона використовується (і виключається з черги). Використання вершини полягає в перегляді одразу всіх іще не відвіданих її сусідів. Усю процедуру подано нижче.

#### ***Алгоритм пошуку вшир у простому зв'язному графі***

Наведемо кроки алгоритму.

Крок 1. Почати з довільної вершини  $v_s$ . Виконати  $BFS(v_s) = 1$  Включити вершину  $v_s$  у чергу.

Крок 2. Розглянути вершину, яка перебуває на початку черги; нехай це буде вершина  $x$ . Якщо для всіх вершин, суміжних із вершиною  $x$ , уже визначено BFS-номери, то перейти до кроку 4, інакше – до кроку 3.

Крок 3. Нехай  $\{x, y\}$  - ребро, у якому номер  $BFS(y)$  не визначено.

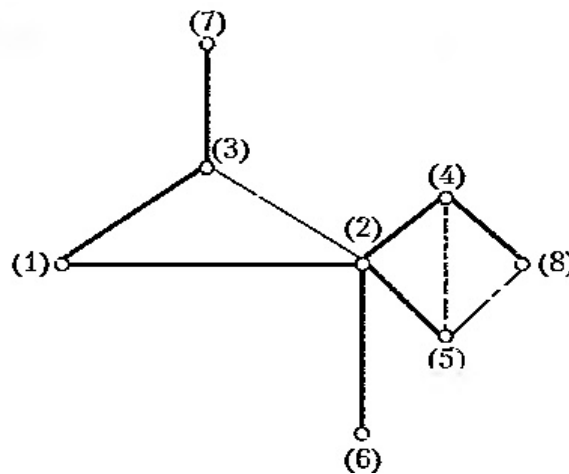


Позначити це ребро потовщеною суцільною лінією, визначити  $BFS(y)$  як черговий BFS-номер, включити вершину  $y$  у чергу й перейти до кроку 2.

Крок 4. Виключити вершині  $x$  із черги. Якщо черга порожня, то зупинитись, інакше – перейти до кроку 2.

Щоб результат виконання алгоритму був однозначним, вершини, які суміжні з вершиною  $x$ , аналізують за зростанням їх порядкових номерів (або в алфавітному порядку). Динаміку роботи алгоритму пошуку вшир також зручно відображати за допомогою протоколу обходу. Він аналогічний попередньому й відрізняється лише третім стовпцем: тепер це – уміст черги (уважаємо, що голова черги ліворуч, а хвіст – праворуч).

**Приклад.** Виконаємо обхід графа на попередньому рисунку пошуком ушир, починаючи з вершини  $b$ . Розв'язок зображено на рисунку, протокол пошуку вшир подано в таблиці.



Таблиця

Вершина	DFS-номер	Вміст стеку
$b$	1	$b$
$c$	2	$bc$
$d$	3	$bcd$
–	–	$cd$
$f$	4	$cdf$
$g$	5	$cdfg$
$h$	6	$cdfgh$
–	–	$dfgh$
$a$	7	$dfgha$
–	–	$fgha$

<i>e</i>	8	<i>fghae</i>
—	—	<i>ghae</i>
—	—	<i>hae</i>
—	—	<i>ae</i>
—	—	<i>e</i>
—	—	$\emptyset$

У процесі роботи наведених алгоритмів будується *дерево  $T$  пошуку* відповідно *вглиб і вишир* – на рисунках його виділено потовщеними лініями.

Обчислювальна складність обох алгоритмів обходу однакова й у разі подання графа списками суміжності становить  $O(m + n)$ , де  $m$  – кількість ребер, а  $n$  – кількість вершин графа.

### **Зважені графи й алгоритми пошуку найкоротших шляхів**

У реальних задачах на графах часто потрібно брати до уваги додаткову інформацію – фактичну віддаль між окремими пунктами, вартість проїзду, час проїзду тощо. Для цього використовують поняття зваженого графа. *Зваженим* називають простий граф, кожному ребру  $e$  якого приписано дійсне число  $w(e)$ . Це число називають *вагою ребра  $e$* . Аналогічно означають *зважений орієнтований граф*: це такий орієнтований граф, кожній дузі  $e$  якого приписано дійсне число  $w(e)$ , називане *вагою дуги*.

Розглянемо три способи зберігання зваженого графа  $G = (V, E)$  в пам'яті комп'ютера. Нехай  $|V| = n, |E| = m$ .

Перший – подання графа матрицею ваг  $W$ , яка являє собою аналог матриці суміжності, її елемент  $w_{ij} = w(v_i, v_j)$ , якщо ребро  $\{v_i, v_j\} \in E$  (у разі орієнтованого графа дуга  $(v_i, v_j) \in E$ ). Якщо ж ребро  $\{v_i, v_j\} \notin E$  (або дуга  $(v_i, v_j) \notin E$ ), то  $w_{ij} = 0$  чи  $w_{ij} = \infty$  залежно від розв'язуваної задачі.

Другий спосіб – подання графа списком ребер. Для зваженого графа під кожний елемент списку  $E$  можна відвести три комірки – дві для ребра й одну для його ваги, тобто всього потрібно  $3m$  комірок.

Третій спосіб – подання графа списками суміжності. Для зваженого графа

кожний список  $Adj[u]$  містить окрім покажчиків на всі вершини  $v$  множини  $\Gamma(u)$  ще й числа  $w(u, v)$ .

*Довжиною шляху в зваженому графі* називають суму ваг ребер (дуг), які утворюють цей шлях. Якщо граф не зважений, то вагу кожного ребра (кожної дуги) вважають рівною 1 й отримують раніше введене поняття довжини шляху як кількості ребер (дуг) у ньому.

*Задача про найкоротший шлях* полягає в знаходженні найкоротшого шляху від заданої початкової вершини  $a$  до заданої вершини  $z$ . Наступні дві задачі – безпосередні узагальнення сформульованої задачі про найкоротший шлях.

1. Для заданої початкової вершини  $a$  знайти найкоротші шляхи від  $a$  до всіх інших вершин.

2. Знайти найкоротші шляхи між усіма парами вершин.

Виявляється, що майже всі методи розв'язання задачі про найкоротший шлях від заданої початкової вершини  $a$  до заданої вершини  $z$  також дають змогу знайти й найкоротші шляхи від вершини  $a$  до всіх інших вершин графа. Отже, за їх допомогою можна розв'язати задачу 1 із невеликими додатковими обчислювальними витратами. З іншого боку, задачу 2 можна розв'язати або  $n$  разів застосувавши алгоритм задачі 1 із різними початковими вершинами, або один раз застосувавши спеціальний алгоритм.

Розглянемо два алгоритми: перший алгоритм призначений для розв'язування задачі 1, другий – для задачі 2.

Найефективніший алгоритм визначення довжини найкоротшого шляху від фіксованої вершини до будь-якої іншої запропонував 1959 р. датський математик Е. Дейкстра (E. Dijkstra). Цей алгоритм застосовний лише тоді, коли вага кожного ребра (дуги) додатна. Опишемо докладно цей алгоритм для орієнтованого графа.

Нехай  $G=(V, E)$  – зважений орієнтований граф,  $w(v_i, v_j)$  – вага дуги  $(v_i, v_j)$ . Почавши з вершини  $a$ , знаходимо віддаль від  $a$  до кожної із суміжних із нею вершин. Вибираємо вершину, віддаль від якої до вершини  $a$  найменша; нехай це буде вершина  $v^*$ . Далі знаходимо віддалі від вершини  $a$  до кожної

вершини суміжної з  $v^*$  вздовж шляху, який проходить через вершину  $v^*$ . Якщо для якоїсь із таких вершин ця віддаль менша від поточної, то заміняємо нею поточну віддаль. Знову вибираємо вершину, найближчу до  $a$  й не вибрану раніше; повторюємо процес.

Описаний процес зручно виконувати за допомогою присвоювання вершинам міток. Є мітки двох типів – тимчасові та постійні. Вершини з постійними мітками групують у множину  $M$ , яку називають *множиною позначених вершин*. Решта вершин має тимчасові мітки, і множину таких вершин позначимо як  $T$ ,  $T = V \setminus M$ . Позначатимемо мітку (тимчасову чи постійну) вершини  $v$  як  $l(v)$ . Значення постійної мітки  $l(v)$  дорівнює довжині найкоротшого шляху від вершини  $a$  до вершини  $v$ , тимчасової – довжині найкоротшого шляху, який проходить лише через вершини з постійними мітками.

Фіксованою початковою вершиною вважаємо вершину  $a$ ; довжину найкоротшого шляху шукаємо до вершини  $z$  (або до всіх вершин графа). Тепер формально опишемо алгоритм Дейкстри.

### Алгоритм Дейкстри

Наведемо кроки алгоритму.

Крок 1. *Присвоювання початкових значень*. Виконати  $l(a) = 0$  та вважати цю мітку постійною. Виконати  $l(v) = \infty$  для всіх  $v \neq a$  уважати ці мітки тимчасовими. Виконати  $x = a$ ,  $M = \{a\}$ .

Крок 2. *Оновлення міток*. Для кожної вершини  $v \in \Gamma(x) \setminus M$  замінити мітку:  $l(v) = \min\{l(u); l(x) + w(x, v)\}$ , тобто оновлювати тимчасові мітки вершин, у які з вершини  $x$  іде дуга.

Крок 3. *Перетворення мітки в постійну*. Серед усіх вершин із тимчасовими мітками знайти вершину з мінімальною міткою, тобто знайти вершину  $v^*$  з умови  $l(v^*) = \min\{l(v)\}$ ,  $v \in T$ , де  $T = V \setminus M$ .

Крок 4. Уважати мітку вершини  $v^*$  постійною й виконати  $M = M \cup \{v^*\}$ ;  $x = v^*$  (вершину  $v^*$  включено в множину  $M$ ).

Крок 5.

а) Для пошуку шляху від  $a$  до  $z$ : якщо  $x = z$ , то  $l(z)$  – довжина найкоротшого шляху від  $a$  до  $z$ , зупинитись; якщо  $x \neq z$ , то перейти до кроку 2.

б) Для пошуку шляхів від  $a$  до всіх інших вершин: якщо всі вершини отримали постійні мітки (включені в множину  $M$ ), то ці мітки дорівнюють довжинам найкоротших шляхів, зупинитись; якщо деякі вершини мають тимчасові мітки, то перейти до кроку 2.

### **Обґрунтування алгоритму Дейкстри**

Мітка вершини  $v$  ( $l(v) \neq \infty$ ) дорівнює довжині  $\langle a, v \rangle$ -шляху, побудованого за алгоритмом до певного моменту. Тому достатньо довести, що постійна мітка  $l(v)$  кожної вершини  $v$  дорівнює довжині найкоротшого  $\langle a, v \rangle$ -шляху. Отже, доведемо, що значення  $l(v)$  – постійної мітки вершини  $v$  – дорівнює довжині найкоротшого шляху від початкової вершини  $a$  до вершини  $v$ . Для доведення цього застосуємо математичну індукцію.

Нехай вершина  $v$  одержала свою постійну мітку на  $k$ -ій ітерації, тобто після  $k$ -го виконання кроку 4 алгоритму Дейкстри. У разі  $k = 1$  твердження очевидне. Припустимо, що воно справджується для вершин, які отримали свої постійні мітки на ітераціях із номерами 2, 3, ...,  $k - 1$ . Позначимо як  $P^0$  шлях від  $a$  до  $v$ , побудований за алгоритмом Дейкстри за  $k$  ітерацій, а як  $P^*$  – найкоротший шлях від  $a$  до  $v$ . Довжину шляху  $P$  позначатимемо як  $w(P)$ . За умовою  $w(P^0) = l(v)$ . Нехай  $M$  і  $T = V \setminus M$  – множини вершин відповідно з постійними та тимчасовими мітками перед початком  $k$ -ої ітерації.

Розглянемо ситуацію перед початком  $k$ -ої ітерації. Можливі два випадки.

*Випадок 1.* Деякі вершини шляху  $P^*$  належать множині  $T$ . Нехай  $\bar{v}$  – перша (починаючи від  $a$ ) з тих вершин шляху  $P^*$ , які належать множині  $T$ , а вершина  $v'$  безпосередньо передує  $\bar{v}$  на шляху  $P^*$ . За вибором  $\bar{v}$  маємо, що  $v' \in M$ . Позначимо як  $P_1^*$  частину шляху  $P^*$  від  $a$  до  $\bar{v}$ . За припущенням індукції  $l(v')$  – довжина найкоротшого шляху від  $a$  до  $v'$ . Тому  $w(P_1^*) = l(v') + w(v', \bar{v}) \geq l(\bar{v})$  (нерівність зумовлена кроком 2). Оскільки  $l(\bar{v})$  –

тимчасова мітка, а постійну мітку  $l(v)$  вершини  $v$  вибрано на  $k$ -й ітерації як найменшу з тимчасових (крок 3), то  $l(\bar{v}) \geq l(v)$ . Об'єднавши дві останні нерівності, одержимо  $w(P^*) \geq w(P_1^*) \geq l(v) = w(P^0)$ , тобто  $w(P^*) \geq w(P^0)$ . Але за означенням  $w(P^*) \leq w(P^0)$ ; отже,  $w(P^*) = w(P^0)$ , тобто  $P^0$  – найкоротший шлях від  $a$  до  $v$ .

*Випадок 2.* Усі вершини шляху  $P^*$  містяться в множині  $M$ . Нехай  $v'$  та  $v''$  – вершини, які безпосередньо передують вершині  $v$  відповідно в шляхах  $P^*$  та  $P^0$ . Позначимо як  $P'$  частину шляху  $P^*$  від початкової вершини  $a$  до вершини  $v'$ . За припущеннями індукції  $w(P') \geq l(v')$ . Якщо  $v' = v''$ , то

$$w(P^*) = w(P') + w(v', v) \geq l(v') + w(v', v) = w(P^0).$$

Припустимо тепер, що  $v' \neq v''$ . Позаяк  $v$  одержує постійну мітку  $l(v)$  з  $v''$ , а не з  $v'$ , то

$$w(P^*) = l(v') + w(v', v) \geq l(v'') + w(v'', v) = w(P^0).$$

Отже, і у випадку 2 справджується нерівність  $w(P^*) \geq w(P^0)$ , тобто  $P^0$  – найкоротший шлях від  $a$  до  $v$ .

Якщо граф подано матрицею суміжності, складність алгоритму Дейкстри становить  $O(n^2)$ . Коли кількість дуг  $m$  значно менша, ніж  $n^2$ , то найкраще подавати орієнтований граф списками суміжності. Тоді алгоритм можна реалізувати зі складністю  $O(m \log n)$ , що в цьому разі істотно менше, ніж  $O(n^2)$ .

Алгоритм Дейкстри дає змогу обчислити довжину найкоротшого шляху від початкової вершини  $a$  до заданої вершини  $z$ . Для знаходження самого шляху потрібно лише нагромаджувати вектор вершин, з яких найкоротший шлях безпосередньо потрапляє в дану вершину. Для цього з кожною вершиною  $v$  графа  $G$ , окрім вершини  $a$ , зв'язують іще одну мітку –  $\theta(v)$ . Крок 2 модифікують так. Для кожної вершини  $v \in \Gamma(x) \setminus M$  якщо  $l(v) > l(x) + w(x, v)$ , то  $l(v) = l(x) + w(x, v)$  та  $\theta(v) := x$ , а ні, то не змінювати  $l(v)$  та  $\theta(v)$ . Коли мітка  $l(v)$  стане постійною, найкоротший  $\langle a, v \rangle$ -шлях буде потрапляти у вершину  $v$  безпосередньо з вершини  $x$ . Із постійних міток  $l(v)$  та  $\theta(v)$  утворюємо вектори

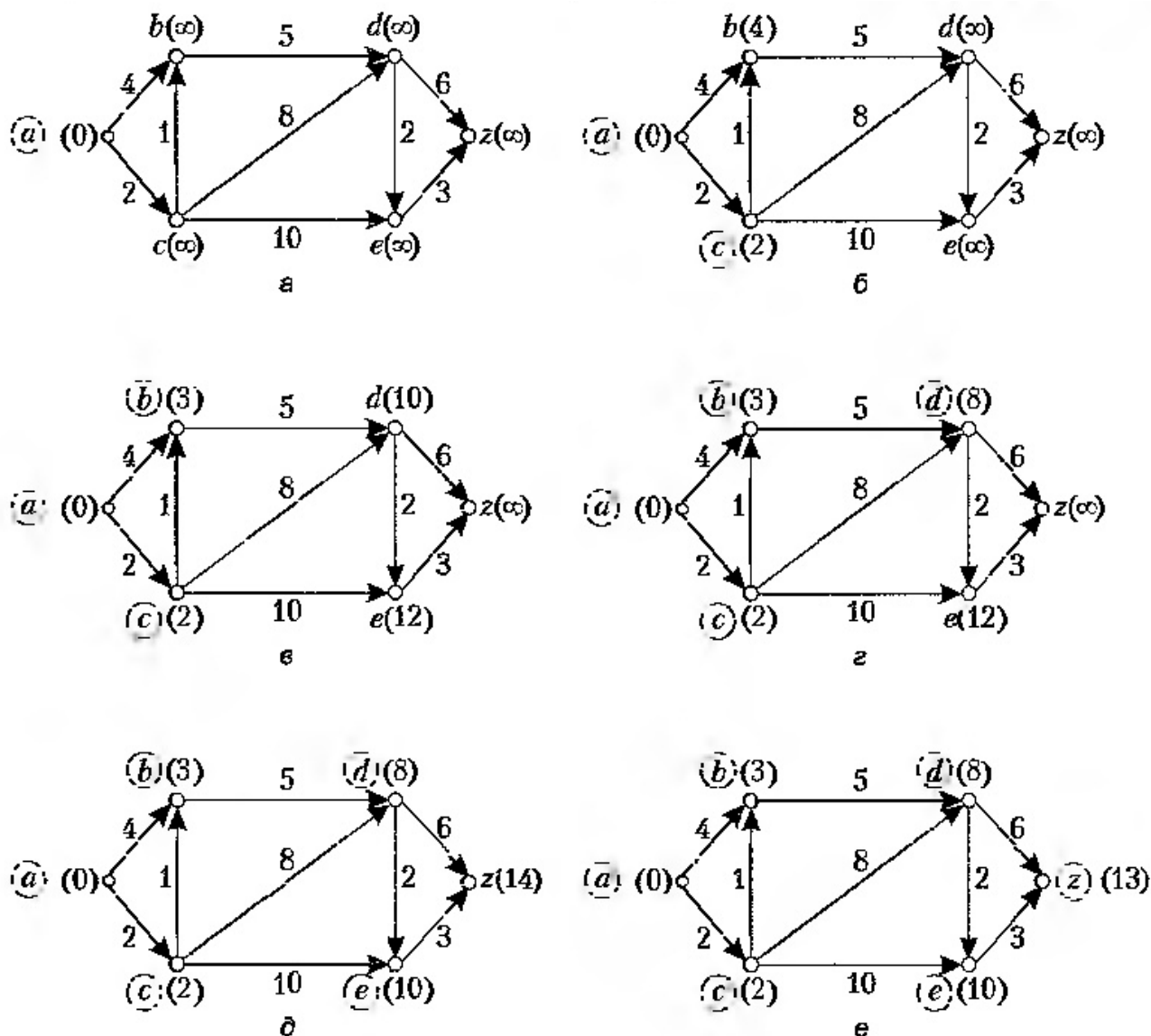
$l$  та  $\theta$ .

**Приклад.** Знайдемо довжину найкоротшого шляху від початкової вершини  $a$  до вершини  $z$  у графі на рисунку. Послідовність дій зображено на цьому ж рисунку, мітки записано в дужках біля вершин. Вершини, які включено в множину  $M$ , обведено кружечками; мітки таких вершин оголошують постійними. У процесі роботи алгоритму будують два вектори: вектор  $l$  постійних міток (довжини найкоротших шляхів від вершини  $a$  до даної вершини) і вектор  $\theta$  вершин, з яких у дану вершину безпосередньо потрапляє найкоротший шлях. У таблиці в першому рядку містяться довільно впорядковані вершини графа, у другому – відповідні постійні мітки (компоненти вектора  $l$ ), а в третьому – компоненти вектора  $\theta$ . Постійна мітка вершини  $z$  дорівнює 13. Отже, довжина найкоротшого шляху від  $a$  до  $z$  дорівнює 13. Сам шлях знаходять за допомогою першого й третього рядків таблиці та будують у зворотному порядку. Кінцева вершина –  $z$ ; у неї потрапляємо з вершини  $e$  (див. вектор  $\theta$ ). У вершину  $e$  потрапляємо з вершини  $d$ , у  $d$  – з  $b$  та продовжуємо цей процес до вершини  $a$ :  $z \leftarrow e \leftarrow d \leftarrow b \leftarrow c \leftarrow a$ . Отже, найкоротший шлях такий:  $a, c, b, d, e, z$ .

Таблиця

Вершина графа (елементи множини $V$ )	$a$	$b$	$c$	$d$	$e$	$z$
Вектор $l$ (постійні мітки вершин)	0	3	2	8	10	13
Вектор $\theta$ (вершини, з яких у дану вершину заходить найкоротший шлях)	–	$c$	$a$	$b$	$d$	$e$

Ми розглянули задачу відшукування в графі найкоротшого шляху від якоїсь виділеної (початкової) вершини до будь-якої іншої. Розглянемо задачу пошуку в графі найкоротшого шляху між кожною парою вершин. Звичайно, цю загальнішу задачу можна розв'язати багатократним застосуванням алгоритму Дейкстри з послідовним вибором кожної вершини графа як початкової. Проте є й прямий спосіб розв'язання цієї задачі за допомогою алгоритму Флойда. У ньому довжини дуг можуть бути від'ємними, проте не може бути циклів із від'ємною довжиною.



Нехай  $G = (V, E)$  – орієнтований граф. Нагадаємо, що внутрішні вершини шляху  $a, x_1, x_2, \dots, x_{m-1}, b$  в графі  $G$  –  $x_1, x_2, \dots, x_{m-1}$ . Наприклад, внутрішні вершини шляху  $a, c, d, a, f, b$  –  $c, d, a, f$ . Пронумеруємо вершини графа цілими числами від 1 до  $n$ . Позначимо як  $w_{ij}^{(k)}$  довжину найкоротшого шляху з вершини  $i$  у вершину  $j$ , у якому як внутрішні можуть бути лише перші  $k$  вершин графа  $G$ . Якщо між вершинами  $i$  та  $j$  не існує жодного такого шляху, то умовно вважатимемо, що  $w_{ij}^{(k)} = \infty$ . Зі сказаного випливає, що  $w_{ij}^{(0)}$  – це вага дуги  $(i, j)$ , а якщо такої дуги немає, то  $w_{ij}^{(0)} = \infty$ . Для довільної вершини  $i$  вважатимемо  $w_{ii}^{(0)} = 0$ . Отже  $w_{ij}^{(n)}$  дорівнює довжині найкоротшого шляху з вершини  $i$  у вершину  $j$ . Позначимо як  $W^{(k)}$   $n \times n$  матрицю,  $(i, j)$ -й елемент якої дорівнює  $w_{ij}^{(k)}$ . Якщо в заданому орієнтованому графі  $G$  відома вага кожної дуги, то, виходячи



з попередніх міркувань, можна сформувати матрицю  $W^{(0)}$ . Тепер потрібно визначити матрицю  $W^{(n)}$ , елементи якої дорівнюють довжинам найкоротших шляхів між усіма парами вершин графа  $G$ .

В алгоритмі Флойда як початкову беруть матрицю  $W^{(0)}$ . Спочатку за нею обчислюють матрицю  $W^{(1)}$ , потім –  $W^{(2)}$ , і процес повторюють доти, доки за матрицею  $W^{(n-1)}$  не буде обчислено матрицю  $W^{(n)}$ . Розглянемо ідею, на якій ґрунтується алгоритм Флойда. Припустимо, що відомі.

1. Найкоротший шлях із вершини  $i$  у вершину  $k$ , у якому як внутрішні використано лише перші  $(k-1)$  вершин.
2. Найкоротший шлях із вершини  $k$  у вершину  $j$ , у якому як внутрішні використано лише перші  $(k-1)$  вершин.
3. Найкоротший шлях із вершини  $i$  у вершину  $j$ , у якому як внутрішні використано лише перші  $(k-1)$  вершин.

Оскільки за припущенням граф  $G$  не містить циклів із від’ємною довжиною, то один із двох шляхів – шлях із п. 3 чи об’єднання шляхів із пп. 1 і 2 – найкоротший шлях із вершини  $i$  у вершину  $j$ , у якому як внутрішні використано лише перші  $(k-1)$  вершин. Отже,

$$w_{ij}^{(k)} = \min \left\{ w_{ik}^{(k-1)} + w_{kj}^{(k-1)}, w_{ij}^{(k-1)} \right\}.$$

З цього співвідношення видно, що для обчислення елементів матриці  $W^{(k)}$  потрібні тільки елементи матриці  $W^{(k-1)}$ . Тепер ми можемо формально описати алгоритм Флойда для знаходження в графі найкоротших шляхів між усіма парами вершин.

### Алгоритм Флойда

Наведемо кроки алгоритму.

Крок 1. *Присвоювання початкових значень.* Пронумерувати вершини графа  $G$  цілими числами від 1 до  $n$ . Побудувати матрицю  $W^{(0)} = W$ , задавши кожний її  $(i, j)$ -елемент такий, що дорівнює вазі дуги, котра з’єднує вершину  $i$  з

вершиною  $j$ . Якщо в графі  $G$  ці вершини не з'єднано дугою, то виконати  $w_{ij}^0 = \infty$ . Крім того, для всіх  $i$  виконати  $w_{ii}^0 = 0$ .

Крок 2. Цикл. Для  $k$ , що послідовно набуває значення  $1, 2, \dots, n$ , визначити за елементами матриці  $W^{(k-1)}$  елементи матриці  $W^{(k)}$ , використовуючи рекурентне співвідношення  $w_{ij}^{(k)} = \min \{ w_{ik}^{(k-1)} + w_{kj}^{(k-1)}, w_{ij}^{(k-1)} \}$ .

Після закінчення цієї процедури  $(i, j)$ -елемент матриці  $W^{(k)}$  дорівнює довжині найкоротшого шляху з вершини  $i$  у вершину  $j$ .

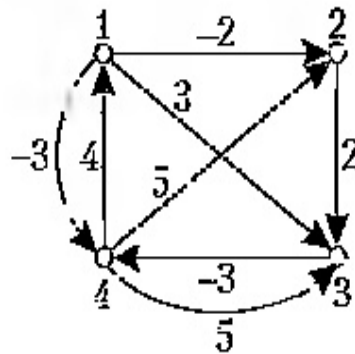
Якщо під час роботи алгоритму для якихось  $k$  й  $i$  виявиться, що  $w_{ii}^{(k)} < 0$ , то в графі  $G$  існує цикл із від'ємною довжиною, який містить вершину  $i$ . Тоді роботу алгоритму потрібно припинити.

Якщо заздалегідь відомо, що в графі  $G$  немає циклів із від'ємною довжиною, то обсяг обчислень можна дещо зменшити. У цьому разі для всіх  $i$  та всіх  $k$  має бути  $w_{ii}^{(k)} = 0$ . Тому не потрібно обчислювати діагональні елементи матриць  $W^{(1)}, W^{(2)}, \dots, W^{(n)}$ . Окрім того, для всіх  $i = 1, 2, \dots, n$  справджуються співвідношення  $w_{ik}^{(k-1)} = w_{ik}^{(k)}$ ,  $w_{ki}^{(k-1)} = w_{ki}^{(k)}$  які випливають із того, що коли немає циклів із від'ємною довжиною, вершина  $k$  не може бути внутрішньою в будь-яких найкоротших шляхах, котрі починаються чи закінчуються в самій вершині  $k$ . Отже, обчислюючи матрицю  $W^{(k)}$ , немає потреби переобчислювати елементи  $k$ -го рядка й  $k$ -го стовпця матриці  $W^{(k-1)}$ . Отже, у матриці  $W^{(k)}$  за рекурентною формулою потрібно обчислювати лише  $n^2 - 3n + 2$  елементи. Очевидно, що складність алгоритму Флойда становить  $O(n^3)$ .

Щоб після закінчення його роботи можна було швидко знайти найкоротший шлях між будь-якою парою вершин, на  $k$ -й ітерації разом із матрицею  $W^{(k)}$  побудуємо матрицю  $\Theta^{(k)} = [\theta_{ij}^{(k)}]$ . Спочатку беремо  $\theta_{ij}^{(0)} = i$  для всіх  $i, j = 1, \dots, n, i \neq j$ ;  $\theta_{ii}^{(0)} = 0$ . Далі на  $k$ -й ітерації візьмемо  $\theta_{ij}^{(k)} = \theta_{ij}^{(k-1)}$ , якщо  $w_{ij}^{(k)} = w_{ij}^{(k-1)}$ , і  $\theta_{ij}^{(k)} = \theta_{kj}^{(k-1)}$ , якщо  $w_{ij}^{(k)} = w_{ik}^{(k-1)} + w_{kj}^{(k-1)}$ . Отже,  $\theta_{ij}^{(k)}$  – номер вершини, яка є перед вершиною  $j$  у поточному  $\langle i, j \rangle$ -шляху, тобто найкоротшому  $\langle i, j \rangle$ -

шляху, усі внутрішні вершини якого містяться в множині  $\{1, 2, \dots, k\}$ . Матриця  $\Theta^{(n)} = [\theta_{ij}^{(n)}]$  відіграє ту саму роль, що й вектор  $\theta$  в алгоритмі Дейкстри. За допомогою матриці  $\Theta^{(n)}$  вершини, через які проходить найкоротший  $\langle i, j \rangle$ -шлях, визначають так:  $i, \dots, j_3, j_2, j_1, j$ , де  $j_1 = \theta_{ij}^{(n)}, j_2 = \theta_{ij_1}^{(n)}, j_3 = \theta_{ij_2}^{(n)}, \dots$ .

**Приклад.** Знайти найкоротші шляхи між усіма парами вершин орієнтованого графа на рисунку.



Нижче наведено результати виконання кожної з чотирьох ітерацій алгоритму Флойда.

$$W^{(0)} = W = \begin{bmatrix} 0 & -2 & 3 & -3 \\ \infty & 0 & 2 & \infty \\ \infty & \infty & 0 & -3 \\ 4 & 5 & 5 & 0 \end{bmatrix};$$

$$\Theta^{(0)} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 2 & 0 & 2 & 2 \\ 3 & 3 & 0 & 3 \\ 4 & 4 & 4 & 0 \end{bmatrix};$$

$$W^{(1)} = \begin{bmatrix} 0 & -2 & 3 & -3 \\ \infty & 0 & 2 & \infty \\ \infty & \infty & 0 & -3 \\ 4 & 2 & 5 & 0 \end{bmatrix};$$

$$\Theta^{(1)} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 2 & 0 & 2 & 2 \\ 3 & 3 & 0 & 3 \\ 4 & 1 & 4 & 0 \end{bmatrix};$$

$$W^{(2)} = \begin{bmatrix} 0 & -2 & 0 & -3 \\ \infty & 0 & 2 & \infty \\ \infty & \infty & 0 & -3 \\ 4 & 2 & 4 & 0 \end{bmatrix};$$

$$\Theta^{(2)} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 2 & 2 \\ 3 & 3 & 0 & 3 \\ 4 & 1 & 2 & 0 \end{bmatrix};$$

$$W^{(3)} = \begin{bmatrix} 0 & -2 & 0 & -3 \\ \infty & 0 & 2 & -1 \\ \infty & \infty & 0 & -3 \\ 4 & 2 & 4 & 0 \end{bmatrix};$$

$$\Theta^{(3)} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 2 & 3 \\ 3 & 3 & 0 & 3 \\ 4 & 1 & 2 & 0 \end{bmatrix};$$

$$W^{(4)} = \begin{bmatrix} 0 & -2 & 0 & -3 \\ 3 & 0 & 2 & -1 \\ 1 & -1 & 0 & -3 \\ 4 & 2 & 4 & 0 \end{bmatrix};$$

$$\Theta^{(4)} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 4 & 0 & 2 & 3 \\ 4 & 1 & 0 & 3 \\ 4 & 1 & 2 & 0 \end{bmatrix}.$$

Матриця  $W^{(4)}$  дає довжини найкоротших шляхів між усіма парами вершин графа, зображеного на рисунку. Зокрема,  $w_{21}^{(4)} = 3$ , тобто довжина найкоротшого шляху від вершини 2 до вершини 1 дорівнює 3. Для знаходження самого шляху скористаємося матрицею  $\Theta^{(4)}$  та запишемо у зворотному порядку вершини, через які він проходить:  $\theta_{21}^{(4)} = 4, \theta_{24}^{(4)} = 3, \theta_{23}^{(4)} = 2$ . Отже, найкоротший шлях із вершини 2 у вершину 1 проходить через вершини 2, 3, 4, 1.

Очевидно, що як алгоритм Дейкстри, так і алгоритм Флойда можна застосовувати без жодних змін і до неорієнтованих графів. Для цього достатньо кожне ребро  $\{u, v\}$ , що має вагу  $w(u, v)$  розглядати як пару дуг  $(u, v)$  та  $(v, u)$  з тією самою вагою. Слід урахувувати, що неорієнтоване ребро із від'ємною вагою одразу приводить до циклу із від'ємною довжиною, що робить алгоритм Флойда незастосовним.

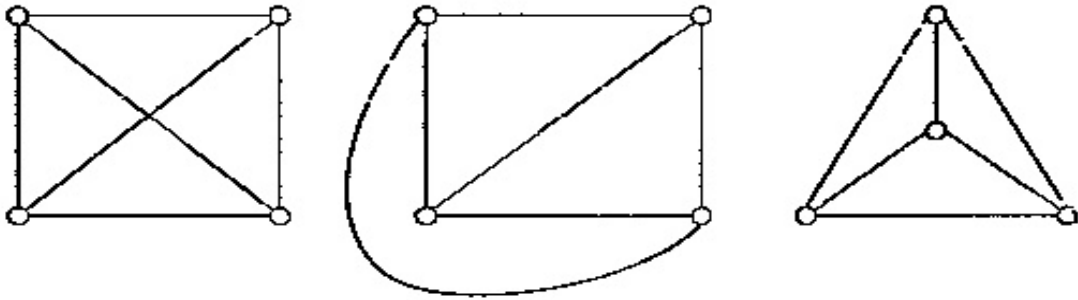
## Планарні графи

Розглянемо неорієнтовані графи. Часто не має значення, як зобразити граф на рисунку, бо ізоморфні графи несуть одну й ту саму інформацію. Проте інколи важливо, чи можна подати граф на площині так, щоб його зображення задовольняло певним вимогам. Наприклад, у радіоелектроніці в процесі виготовлення мікросхем друкованим способом електричні ланцюги наносять на плоску поверхню ізоляційного матеріалу. Оскільки провідники не ізольовані, то вони не мають перетинатись. Аналогічна задача виникає під час проектування залізничних та інших шляхів, де переїзди небажані. Так виникає поняття

плоского графа.

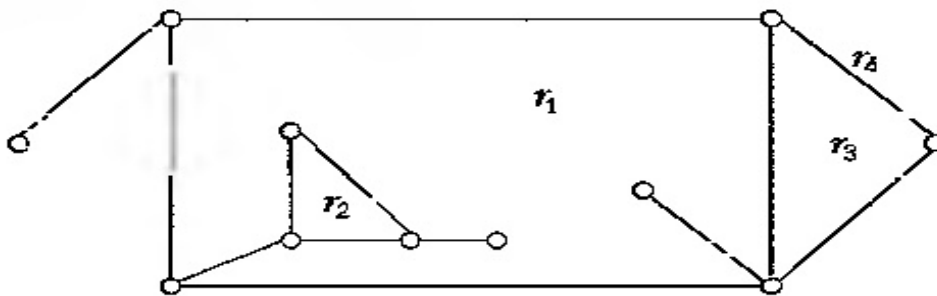
*Плоским* називають граф, зображений на площині так, що ніякі два його ребра геометрично не перетинаються ніде, окрім інцидентних їм вершин. Граф, ізоморфний до плоского графа, називають *планарним*.

**Приклад.** Усі три графи на рисунку планарні, але лише другий і третій із них плоскі.



Плоский граф розбиває площину на області, одна з яких необмежена. Їх називають *гранями* плоского графа; необмежену область називають *зовнішньою гранню*.

**Приклад.** На рисунку зображено плоский граф. Він має чотири грані:  $r_1$ ,  $r_2$ ,  $r_3$ ,  $r_4$ ; грань  $r_4$  зовнішня.



**Теорема (Ейлера про плоскі графи).** Нехай зв'язний плоский граф  $G$  має  $n$  вершин,  $m$  ребер і  $r$  граней. Тоді  $n + r = m + 2$ .

**Доведення** проводимо індукцією за кількістю ребер у графі  $G$ . Якщо  $m = 0$ , то  $n = 1$ ,  $r = 1$ , і теорема справджується. Допустимо, що теорема справджується для довільного плоского графа  $G$ , який має  $m - 1$  ребро, і додамо до  $G$  нове ребро  $e$ .

Можливі три випадки.

1. Ребро  $e$  - петля; тоді виникне нова грань, а кількість вершин залишиться незмінною.

2. Ребро  $e$  з'єднує дві різні вершини графа  $G$ ; у такому разі одна з граней розпадеться на дві, тому кількість граней збільшиться на одну, а кількість вершин не зміниться.

3. Ребро  $e$  інцидентне лише одній вершині в  $G$ ; тоді потрібно додати ще одну вершину; отже, кількість вершин збільшиться на одну, а кількість граней не зміниться.

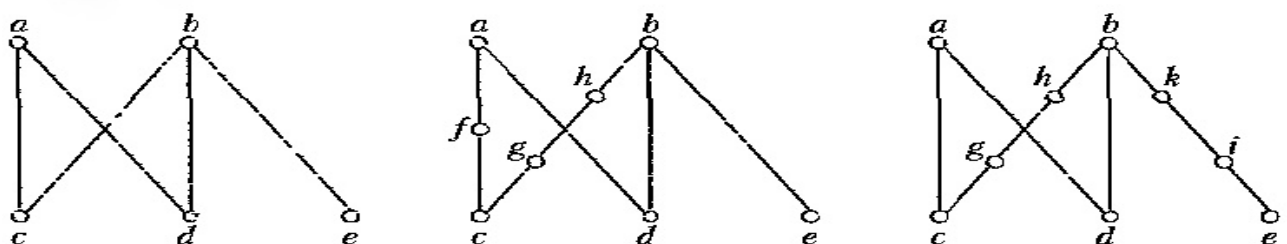
Твердження теореми залишається правильним у кожному з цих випадків. Оскільки інші випадки неможливі, то індукцію завершено й теорему доведено.

**Теорема.** Графи  $K_5$  і  $K_{3,3}$  непланарні.

**Доведення.** Граф  $K_5$  має п'ять вершин і десять ребер. Припустимо, що він планарний; тоді існує ізоморфний до нього плоский граф. За теоремою Ейлера  $r = m + 2 - n = 10 + 2 - 5 = 7$ . Зазначимо, що будь-яке ребро плоского графа або розділяє дві різні грані, або являє собою міст. Позаяк граф  $K_5$  не має петель і кратних ребер, то кожна грань обмежена принаймні трьома ребрами. Тому число  $3r$  - оцінка знизу подвоєної кількості ребер графа, тобто  $3r \leq 2m$ , звідки випливає, що  $21 \leq 20$ . Суперечність.

У графі  $K_{3,3}$  кількість вершин  $n = 6$ , а кількість ребер  $m = 9$ . Припустимо, що він планарний. Тоді в ізоморфному до нього плоскому графі за теоремою Ейлера кількість граней  $r = m + 2 - n = 9 + 2 - 6 = 5$ . Будь-яка грань дводольного графа має бути обмежена принаймні чотирма ребрами. Отже,  $4r \leq 2m$ , звідки випливає, що  $20 \leq 18$ . Суперечність.

Два графи називаються *гомеоморфними*, якщо їх можна отримати з одного графа додаванням до його ребер нових вершин степеня 2 (рисунок).

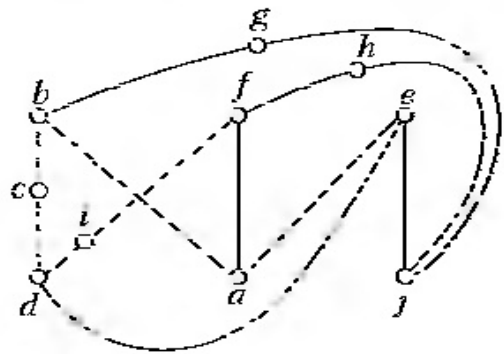
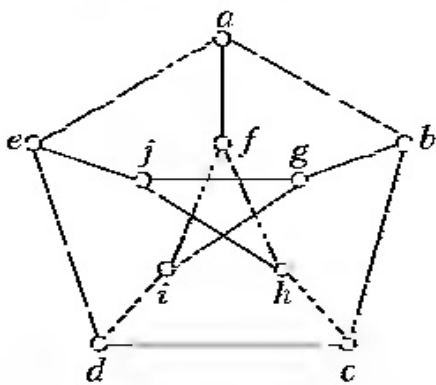


Наступна теорема дає критерій (необхідну й достатню умову) планарності графа.

**Теорема (Куратовського).** Граф планарний тоді й лише тоді, коли він не містить підграфів, гомеоморфних графам  $K_5$  або  $K_{3,3}$ .

Необхідність умов теореми вже доведено, бо доведено непланарність графів  $K_5$  і  $K_{3,3}$ , а доведення достатності складне, і ми його не наводимо.

**Приклад.** На рисунку зліва зображено *граф Петерсена*, а справа - його підграф, гомеоморфний  $K_{3,3}$ . Отже, за теоремою Куратовського, граф Петерсена непланарний.



Окрім теореми Куратовського є й інші критерії планарності графів. Практично перевірити умови, якими характеризуються планарні графи, не завжди просто. Проте розроблено ефективні алгоритми, які дають змогу для будь-якого заданого графа знайти його зображення на площині без перетину ребер або переконатись, що це неможливо (якщо граф непланарний).

## Розфарбовування графів

У цьому підрозділі розглянуто лише прості графи. *Розфарбовуванням простого графа  $G$*  називають таке приписування кольорів (або натуральних чисел) його вершинам, що ніякі дві суміжні вершини не набувають однакового кольору.

Найменшу можливу кількість кольорів у розфарбуванні називають хроматичним числом і позначають  $\chi(G)$ . Очевидно, що існує розфарбування

графа  $G$  в  $k$  кольорів ( $k$ -розфарбовування) для будь-якого  $k$  в діапазоні  $\chi(G) \leq k \leq n$ , де  $n$  - кількість вершин графа. Множину вершин, розфарбованих в один колір, називають *одноколірним класом*. Такі класи утворюють незалежні множини вершин, тобто ніякі дві вершини в одноколірному класі не суміжні.

Очевидно, що  $\chi(K_n) = n$ , і, отже, легко побудувати граfi з як завгодно великим хроматичним числом. З іншого боку,  $\chi(G) = 1$  тоді й лише тоді, коли  $G$  - вироджений граф;  $\chi(G) = 2$  тоді й лише тоді, коли  $G$  - дводольний граф (тому дводольний граф називають *біхроматичним*).

**Теорема.** Якщо найбільший зі степенів вершин графа дорівнює  $\rho$ . то цей граф можна розфарбувати в  $\rho + 1$  колір.

**Доведення.** Застосуємо індукцію за кількістю вершин графа. Нехай граф  $G$  має  $n$  вершин; вилучимо з нього довільну вершину  $v$  разом з усіма інцидентними їй ребрами. Отримаємо граф з  $n - 1$  вершиною; степінь кожної вершини не більший ніж  $\rho$ . За припущенням індукції цей граф можна розфарбувати в  $\rho + 1$  колір. Отже, у  $\rho + 1$  колір можна розфарбувати й граф  $G$ , якщо розфарбувати вершину  $v$  кольором, що відрізняється від тих, якими розфарбовано суміжні з нею вершини (а їх разом не більше ніж  $\rho$ ).

Із середини XIX ст. відкритою залишалася проблема, відома під назвою гіпотези чотирьох фарб. Її формулюють так: будь-який планарний граф можна розфарбувати в чотири кольори, тобто  $\chi(G) \leq 4$  для будь-якого планарного графа  $G$ .

Перше з помилкових „доведень” належить А. Кемке (1879 р.), але помилку було виявлено не відразу. її знайшов Р. Хейвуд (1890 р.) й тоді ж довів, що будь-який планарний граф можна розфарбувати в *п'ять* кольорів.

**Теорема (Р. Хейвуда).** Будь-який планарний граф можна розфарбувати в п'ять кольорів, тобто  $\chi(G) \leq 5$  для будь-якого планарного графа  $G$ .

**Доведення** цієї теореми ґрунтується на тому, що в будь-якому простому планарному графі є вершина, степінь якої не більший ніж п'ять. Цей результат



легко одержати як наслідок із теореми Ейлера про плоскі графи.

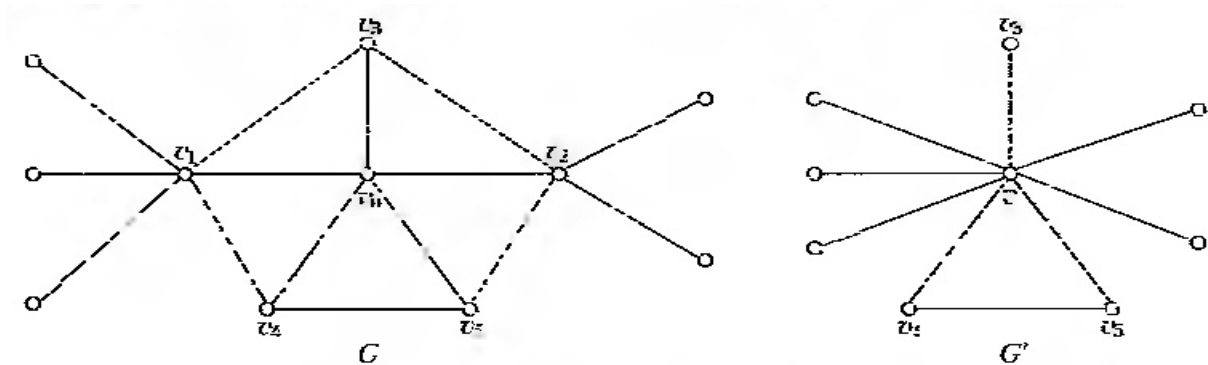
Застосуємо математичну індукцію за кількістю вершин графа. Теорема справджується для графів із не більше ніж п'ятьма вершинами. Припустимо, що вона справджується для графів із не більше ніж  $n$  вершинами, де  $n \geq 5$ . Розглянемо довільний плоский граф  $G$  з  $(n + 1)$  вершиною. Він містить вершину  $v_0$ , степінь якої не більший ніж п'ять. Нехай  $W = \Gamma(v_0)$  - множина вершин, суміжних із вершиною  $v_0$  в графі  $G$ . Окремо розглянемо два випадки.

*Випадок 1.*  $|W| \leq 4$ . Позначимо як  $G - v_0$  граф, отриманий із графа  $G$  вилученням вершини  $v_0$  та всіх інцидентних їй ребер. За індуктивним припущенням граф  $G - v_0$  можна розфарбувати в п'ять кольорів. Зафіксуємо одне з таких розфарбувань і зафарбуємо вершину  $v_0$  в той із п'яти кольорів, який не використано для фарбування вершин із множини  $W$ .

*Випадок 2.*  $|W| = 5$ . У множині  $W$  є дві несуміжні вершини  $v_1$  і  $v_2$ , а ні, то граф  $G(W)$ , породжений множиною вершин  $W$ , - це  $K_5$ , і тоді граф  $G$  непланарний. Граф  $G'$ , одержаний із графа  $G - v_0$  злиттям вершин  $v_1$  і  $v_2$  в одну вершину  $v$ , плоский, і за індуктивним припущенням його можна розфарбувати в п'ять кольорів. Зафіксуємо одне з таких розфарбувань графа  $G'$ . Тепер у графі  $G$  розфарбуємо вершини  $v_1$  і  $v_2$  в колір вершини  $v$ , а решту вершин, відмінних від  $v_0$ , - у ті самі кольори, що й відповідні вершини графа  $G'$ . Нарешті, припишемо вершині  $v_0$  колір, не використаний для розфарбовування вершин із  $W$ .

Американські математики К. Аппель (K. Appel) і У. Гайкен (W. Haken) у 1976 р. довели гіпотезу чотирьох фарб, суттєво використавши комп'ютерні обчислення. Це пертий випадок, коли настільки відому математичну проблему було розв'язано за допомогою комп'ютера. Спочатку проблему було зведено до розгляду скінченної (хоча й великої) кількості випадків. Надалі список „підозрілих” графів було зменшено й за допомогою комп'ютера розфарбовано в чотири кольори планарні графи з цього списку. За різними джерелами, це можна було зробити за 1500-2000 годин роботи потужного комп'ютера. Отже, проблему чотирьох фарб було розв'язано. Хоча сам по собі метод доведення

являє собою видатне досягнення й цілком достатній, щоб припинити пошуки контрприкладу, було б добре, щоб хтось знайшов елегантніше доведення гіпотези. Найцікавіше в цьому доведенні те, що воно розширило наше уявлення про математичне доведення. Проте не всі поділяють цю думку. Опоненти вважають, що важко погодитись із таким доведенням, бо й зведення загального випадку до скінченної множини графів, і розфарбовування останніх дуже важко повторити.



Нехай  $G$  - простий граф, а  $P_G(k)$  дорівнює кількості способів, якими можна розфарбувати вершини графа  $G$  в  $k$  кольорів.  $P_G(k)$  називають *хроматичною функцією* графа  $G$ . Для повного графа  $K_n$  виконується рівність  $P_G(k) = k(k-1)(k-2)\dots(k-n+1)$ . Справді, для першої вершини є  $k$  варіантів кольору. Оскільки друга вершина суміжна з першою, то її можна розфарбувати в один з  $(k-1)$  кольорів. Третя вершина суміжна з першою та другою, тому для неї є  $(k-2)$  варіантів вибору. Продовжуючи ці міркування, доходимо висновку, що для останньої вершини залишиться  $k-(n-1)$  варіантів вибору кольору.

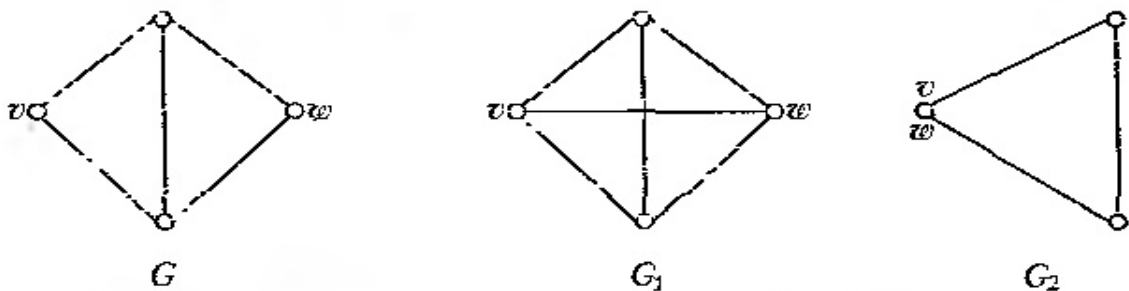
Якщо  $k < \chi(G)$ , то  $P_G(k) = 0$ , а для  $k \geq \chi(G)$  виконується нерівність  $P_G(k) > 0$ . Гіпотеза чотирьох фарб еквівалентна такому твердженню: якщо  $G$  - простий планарний граф, то  $P_G(4) > 0$ . Якщо надано довільний простий граф, то в загальному випадку важко отримати його хроматичну функцію за допомогою простих міркувань. Наступна теорема й наслідок із неї дають систематичний метод одержання хроматичної функції простого графа у вигляді

суми хроматичних функцій повних графів.

**Теорема.** Нехай  $G$  - простий граф, а  $v$  та  $w$  - його несуміжні вершини. Якщо граф  $G_1$ , отримано з  $G$  за допомогою з'єднання вершин  $v$  та  $w$  ребром, а граф  $G_2$  - ототожненням вершин  $v$  та  $w$  (і кратних ребер, якщо їх буде одержано), то  $P_G(k) = P_{G_1}(k) + P_{G_2}(k)$ .

**Доведення.** За будь-якого допустимого розфарбовування вершин графа  $G$  існує альтернатива:  $v$  та  $w$  мають або різні кольори, або той самий. Кількість розфарбовувань, за яких  $v$  та  $w$  мають різні кольори, не зміниться, якщо долучити ребро  $\{v, w\}$ . Отже, ця кількість дорівнює  $P_{G_1}(k)$ . Аналогічно, кількість розфарбовувань, за яких  $v$  та  $w$  мають один колір, не зміниться, якщо ототожнити  $v$  та  $w$ . Отже, ця кількість дорівнює  $P_{G_2}(k)$ . Залишилося застосувати комбінаторне правило суми. Теорему доведено.

**Приклад.** На рисунку зліва зображено граф  $G$ , а справа - графи  $G_1$  і  $G_2$ .



**Наслідок.** Хроматична функція простого графа - поліном.

Тепер ми будемо називати  $P_G(k)$  *хроматичним поліномом*. Зазначимо деякі властивості хроматичного полінома. Якщо граф  $G$  має  $n$  вершин, то степінь полінома  $P_G(k)$  дорівнює  $n$ . Коефіцієнт при  $k^n$  дорівнює 1, а при  $k^{n-1}$  дорівнює  $t$ , де  $t$  - кількість ребер графа  $G$ ; знаки коефіцієнтів чергуються; вільний член хроматичного полінома дорівнює 0.

Хроматичний поліном будують на основі наведеної вище теореми у вигляді суми хроматичних поліномів повних графів.

**Приклад.** На рисунку зображено процес побудови хроматичного полінома, де знаки „=” та „+” мають умовний зміст. Отже,



### **Задача призначення телевізійних каналів.**

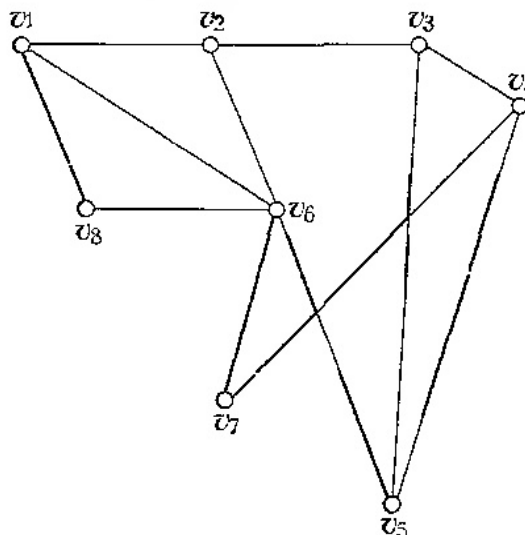
Передавальні станції зображають вершинами графа. Якщо віддаль між будь-якими двома станціями менша за  $l$ , то відповідні вершини графа з'єднують ребром. Граф розфарбовують зіставляючи різним кольорам вершин різні канали. Мінімальна кількість каналів відповідає розфарбуванню графа мінімальною кількістю кольорів.

### **Незалежні множини вершин. Кліки**

Нехай  $G$  - простий граф, Множину його вершин називають *незалежною* (або *внутрішньо стійкою*), якщо ніякі вершини цієї множини не суміжні. Незалежну множину називають *максимальною*, якщо вона не є підмножиною жодної іншої незалежної множини з більшою кількістю вершин.

Найпотужнішу максимальну незалежну множину називають *найбільшою*. Кількість вершин у найбільшій незалежній множині графа  $G$  називають *числом незалежності* (числом внутрішньої стійкості, нецільністю) і позначають  $\alpha(G)$ .

**Приклад.** У графі, зображеному на рисунку, розглянемо множини вершин  $Y_1 = \{v_7, v_8, v_2, v_5\}$ ,  $Y_2 = \{v_7, v_8, v_2\}$ ,  $Y_3 = \{v_1, v_3, v_7\}$ ,  $Y_4 = \{v_4, v_6\}$ . Максимальні незалежні множини –  $Y_1$ ,  $Y_3$  та  $Y_4$ . Множина вершин  $Y_2$  незалежна, але не максимальна. Найбільша незалежна множина –  $Y_1 = \{v_7, v_8, v_2, v_5\}$ . Отже,  $\alpha(G) = 4$ .



Із поняттям незалежності в графі пов'язане поняття домінування. Підмножину  $V'$  вершин графа  $G=(V,E)$  називають *домінантною* (або *зовнішньо стійкою*), якщо кожна вершина з  $V \setminus V'$  суміжна з якоюсь вершиною з  $V'$ . Інакше кажучи, кожна вершина графа віддалена від домінантної множини не більше, ніж на одиницю. Домінантна множина має назву *мінімальної*, якщо жодна з її власних підмножин не домінантна. Домінантну множину з найменшою потужністю називають *найменшою*.

У відшуванні в графі найменшої домінантної множини полягає зміст багатьох практичних задач. Типова задача така. Є множина населених пунктів, зв'язаних дорогами. У деяких із них потрібно розмістити підприємства обслуговування так, щоб віддаль від кожного з населених пунктів до будь-якого підприємства не перевищувала заданої величини. Розміщення слід виконати так, щоб обійтися мінімальною кількістю підприємств. Населеним пунктам поставимо у відповідність вершини графа, у якому дві вершини з'єднано ребром тоді й лише тоді, коли віддаль між відповідними пунктами не перевищує заданої величини. Тоді задача зводиться до побудови в цьому графі найменшої домінантної множини.

Уведемо ще одне поняття, пов'язане з поняттям незалежності. Говорять, що вершина й ребро графа *покривають одне одного*, якщо вони інцидентні. Отже, ребро  $e = \{u, v\}$  покриває вершини  $u$  та  $v$ , а кожна з цих вершин покриває ребро  $e$ .

Підмножину вершин  $V' \subset V$  називають *покриттям* (вершинним покриттям, опорою) графа  $G=(V,E)$ , якщо кожне ребро з  $E$  інцидентне принаймні одній вершині з  $V'$ . Покриття графа  $G$  називають *мінімальним*, якщо воно не містить покриття з меншою кількістю вершин, і *найменшим*, якщо кількість вершин у ньому найменша серед усіх покриттів графа  $G$ . Кількість вершин у найменшому покритті графа  $G$  називають *числом покриття* (або *числом вершинного покриття*) графа  $G$  та позначають як  $\beta(G)$ .

**Приклад.** У графі, зображеному на рисунку, кожна з множин

$X_1 = \{v_1, v_3, v_4, v_6\}, \quad X_2 = \{v_1, v_3, v_4, v_5, v_6\}, \quad X_3 = \{v_2, v_4, v_5, v_6, v_8\},$   
 $X_4 = \{v_1, v_2, v_3, v_5, v_7, v_8\}$  являє собою покриття, причому  $X_1$  - найменше покриття,  
 а  $X_3, X_4$  - мінімальні.

Наступна теорема свідчить про тісний зв'язок між покриттями та незалежними множинами графа.

**Теорема.** Множина  $X$  вершин графа  $G = (V, E)$  являє собою (найменше, мінімальне) покриття тоді й лише тоді, коли  $Y = V \setminus X$  - (найбільша, максимальна) незалежна множина. Отже,  $\alpha(G) + \beta(G) = |V|$ .

**Доведення.** За означенням множина  $Y$  незалежна тоді й лише тоді, коли в графі немає ребра, обидва кінці якого містяться в  $Y$ , тобто хоча б один із кінців кожного ребра належить  $X$ . Останнє означає, що  $X$  - вершинне покриття. Оскільки  $|X| + |Y| = |V|$ , то, очевидно, найбільшим  $Y$  відповідають найменші  $X$  і навпаки.

Протилежне до поняття незалежної множини є поняття кліки. Підмножину  $V' \subset V$  вершин графа  $G$  називають *клікою*, якщо будь-які дві вершини з  $V'$  суміжні, тобто породжений підграф  $G(V')$  повний.

Кліку називають *максимальною*, якщо вона не є підмножиною жодної іншої кліки з більшою кількістю вершин. Найпотужнішу максимальну кліку називають *найбільшою*. Кількість вершин у найбільшій кліці графа називають *щільністю*, або *кліковим числом*, і позначають  $\varphi(G)$ . Доповнювальним до графа  $G$  називають граф  $\bar{G}$  з тією самою множиною вершин, будь-які дві вершини якого з'єднано ребром тоді й лише тоді, коли їх не з'єднано ребром у графі  $G$ .

**Теорема.** Підмножина вершин графа  $G$  являє собою кліку тоді й лише тоді, коли вона незалежна в доповнювальному графі  $\bar{G}$ . Отже,  $\varphi(G) = \alpha(\bar{G})$ .

**Доведення** випливає з означень.

Із теореми випливає, що побудову максимальної кліки можна звести до побудови максимальної незалежної множини вершин.