Розділ 3

Теорія графів

- Основні означення та властивості
- Деякі спеціальні класи простих графів
- Способи подання графів
- Шляхи та цикли. Зв'язність
- Ізоморфізм графів
- Ейлерів цикл у графі
- Гамільтонів цикл у графі
- Зважені графи й алгоритми пошуку найкоротших шляхів
- Обхід графів
- Планарні графи
- Розфарбовування графів
- Незалежні множини вершин. Кліки
- Паросполучення в графах. Теорема Холла
- Найбільше паросполучення у дводольних графах

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.

3.1. 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, називаних *вершинами*, E — множина невпорядкованих пар різних елементів із V. Елементи множини E (невпорядковані пари різних вершин) називають *ребрами*.

Приклад 3.1. На рис. 3.1 зображено простий граф G з множиною вершин $V=\{v_1, v_2, v_3, v_4\}$ і множиною ребер $E=\{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_3, v_4\}\}$.

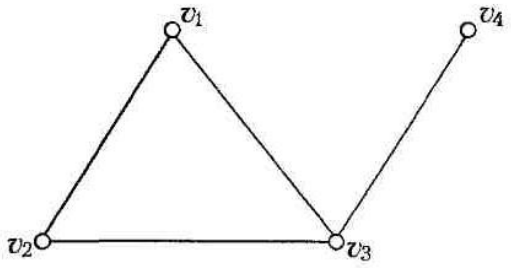


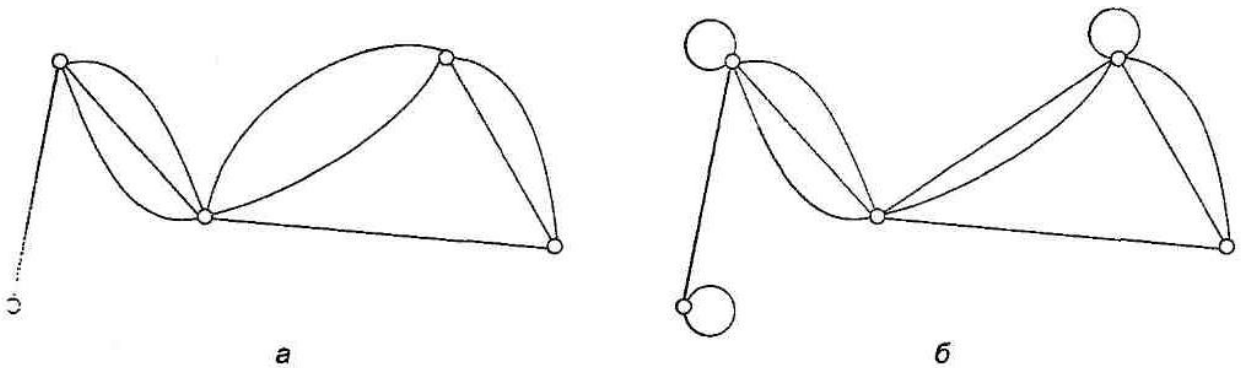
Рис. 3.1

Говорять, що ребро $\{u, v\}$ *з'єднує* вершини u та v. Оскільки E — множина, то в 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* невпорядкованих 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 невпорядкованих пар не обов'язково різних вершин.

Приклад 3.2. На рис. 3.2 зображено мультиграф (рис. 3.2, *а*) і псевдограф (рис. 3.2, *б*).



а *б*

Рис. 3.2

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 3.3. На рис. 3.3 зображено орієнтований граф із множиною вершин $V=\{v_1, v_2, v_3, v_4, v_5\}$ і множиною дуг $E=\{(v_2, v_1), (v_2, v_3), (v_3, v_2), (v_3, v_4), (v_4, v_3), (v_4, v_5), (v_5, v_5)\}$.

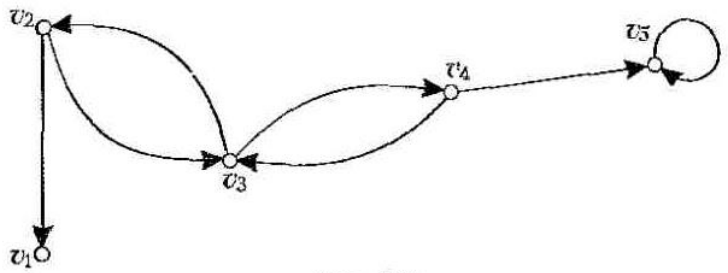


Рис. 3.3

Зазначимо, що дуга — це впорядкована пара вершин (записують у круглих дужках), тому в графі на рис. 3.3 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 3.4. На рис. 3.4 наведено приклад орієнтованого мультиграфа. Дуги e_2 та e_3 — кратні, а дуги e_5, e_6 — ні.

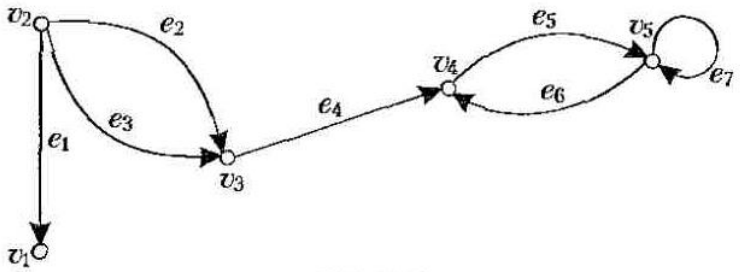


Рис. 3.4

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 [52]. Означення різних типів графів зведено в табл. 3.1.

Таблиця 3.1

Тип графа	Ребра	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 *ізольованою*; якщо $\deg(v) = 1$ — *висячою*, або *кінцевою*.

Приклад 3.5. У неорієнтованому графі на рис. 3.5 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 — ізольована, а v_4 — висяча.

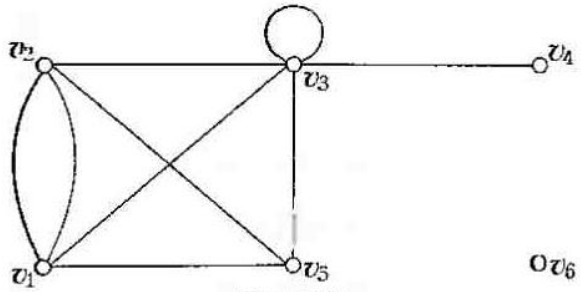


Рис. 3.5

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

ТЕОРЕМА 3.1. 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 3.2. 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 3.6. Для графа, зображеного на рис. 3.6, напівстепені вершин такі:

$$\deg^-(v_1)=0,\ \deg^+(v_1)=1,\ \deg^-(v_2)=2,\ \deg^+(v_2)=0,\ \deg^-(v_3)=1,\ \deg^+(v_3)=2.$$

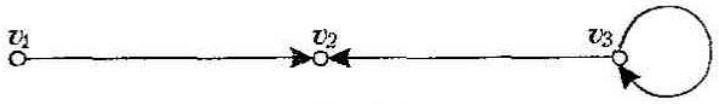


Рис. 3.6

ТЕОРЕМА 3.3. 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 *породженим* (або *індукованим*) *множиною* W і позначають як $G(W)$.

Приклад 3.7. На рис. 3.7 зображено граф G та три його підграфа H_1, H_2, H_3, серед яких H_2 породжений, а H_3 — каркасний.

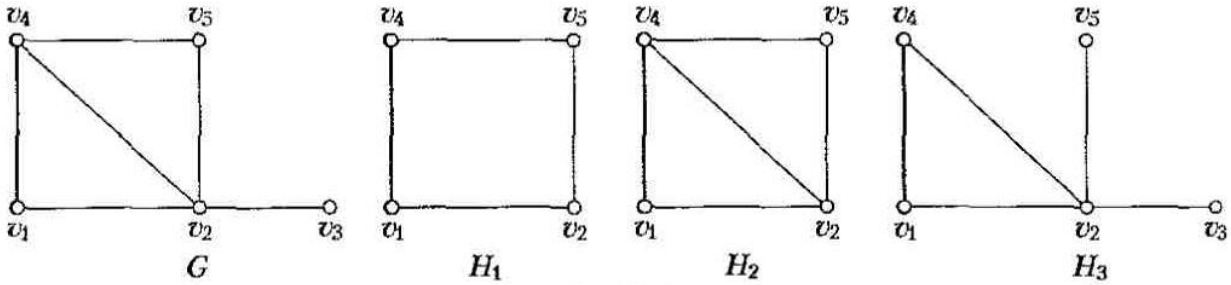


Рис. 3.7

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 3.8. На рис. 3.8 зображено приклад об'єднання двох простих графів. Знаки „$\cup$" та „$=$" тут мають символічний зміст.

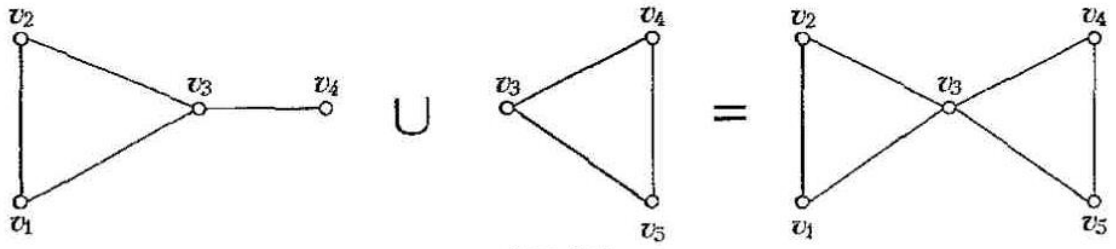


Рис. 3.8

Надалі значну увагу буде приділено алгоритмам на графах, тому дамо деякі означення, що стосуються алгоритмів. Під час аналізу алгоритму нас буде цікавити передусім його складність, під якою розуміємо час виконання відповідної програми на комп'ютері. Зрозуміло, що цей показник суттєво залежить від типу комп'ютера. Щоб висновки про складність були достатньо універсальними, будемо вважати, що всі обчислення виконуються на якомусь ідеалізованому комп'ютері. Цим питанням повністю присвячено розділ 10, а тут лише коротко окреслимо суть справи.

Означимо *складність* алгоритму розв'язування задачі як функцію f, яка кожному невід'ємному цілому числу n ставить у відповідність час роботи $f(n)$ алгоритму в найгіршому випадку на входах довжиною n. Час роботи алгоритму вимірюють у кроках (операціях), виконуваних на ідеалізованому комп'ютері.

Наприклад, якщо алгоритм приймає як вхідні дані довільний граф $G=(V, E)$, то під довжиною входу можна розуміти $|V|$ чи $\max\{|V|, |E|\}$.

Аналіз ефективності алгоритмів полягає в з'ясуванні того, як швидко зростає функція $f(n)$ зі збільшенням n. Для порівняння швидкості зростання двох функцій $f(n)$ і $g(n)$ (з невід'ємними значеннями) використовують такі позначення:

- $f(n)=O(g(n))$ означає, що існують додатна стала c та натуральне число n_0 такі, що $f(n) \leqslant cg(n)$ для всіх $n \geqslant n_0$;
- $f(n)=\Omega(g(n))$ означає, що існують додатна стала c та натуральне число n_0 такі, що $f(n) \geqslant cg(n)$ для всіх $n \geqslant n_0$.

Для аналізу ефективності алгоритмів використовують O-символіку. Вираз „складність алгоритму дорівнює (становить) $O(g(n))$" має саме такий зміст, як у наведеному вище означенні. Зокрема, складність $O(1)$ означає, що час роботи відповідного алгоритму не залежить від довжини входу.

Алгоритм зі складністю $O(n)$, де n – довжина входу, називають *лінійним*. Такий алгоритм для переважної більшості задач найліпший (за порядком) щодо складності.

Алгоритм, складність якого дорівнює $O(p(n))$, де $p(n)$ – поліном, називають *поліноміальним*. Часто замість $O(p(n))$ пишуть $O(n^a)$, де a – константа. Особливу роль поліноміальних алгоритмів описано в розділі 10. Тут лише зазначимо, що всі задачі дискретної математики, які вважають важкими для алгоритмічного розв'язування, нині не мають поліноміальних алгоритмів. Крім того, поняття „поліноміальний алгоритм" – це найпоширеніша формалізація поняття „ефективний алгоритм".

Алгоритми, часова складність яких не піддається подібній оцінці, називають *експоненціальними*. Більшість експоненціальних алгоритмів – це просто варіанти повного перебору, а поліноміальні алгоритми здебільшого можна побудувати лише тоді, коли вдається заглибитись у суть задачі. Задачу називають *важкорозв'язною*, якщо для її розв'язання не існує поліноміального алгоритму.

3.2. Деякі спеціальні класи простих графів

Розглянемо деякі спеціальні класи простих графів. часто використовувані як приклади й широко застосовувані [15].

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 3.9. На рис. 3.9 зображено графи K_n для $n = 1, 2, 3, 4, 5$.

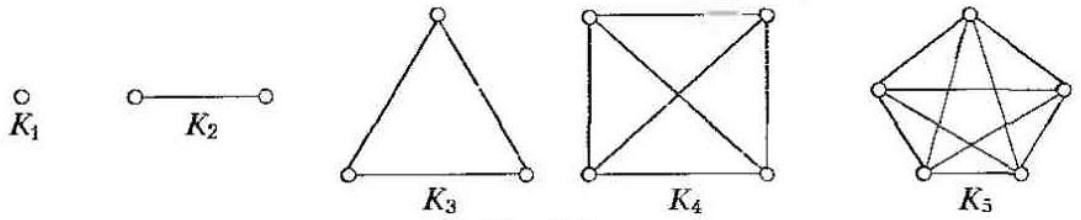


Рис. 3.9

Граф називають *порожнім*, якщо $E = \varnothing$, 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 ($V_1 \cup V_2 = V$, $V_1 \cap V_2 = \varnothing$), 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 $K_{m,n}$ має $n + m$ вершин та mn ребер.

Приклад 3.10. На рис. 3.10 наведено повні дводольні графи $K_{1,5}$, $K_{3,2}$ та $K_{3,3}$.

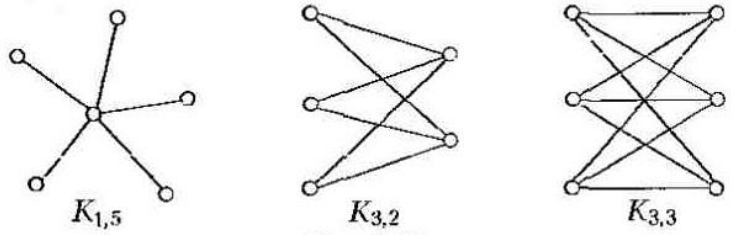


Рис. 3.10

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

Приклад 3.11. На рис. 3.11 зображено цикли C_3, C_4, C_5 і C_6.

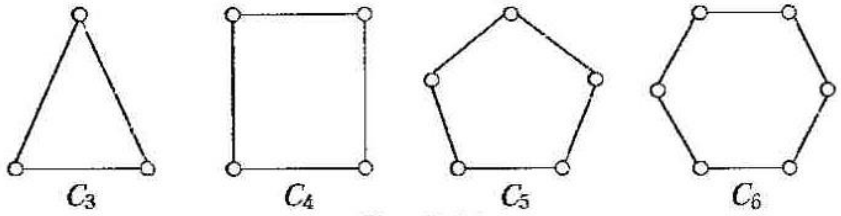


Рис. 3.11

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

Приклад **3.12.** На рис. 3.12 зображено колеса W_3, W_4, W_5, W_6.

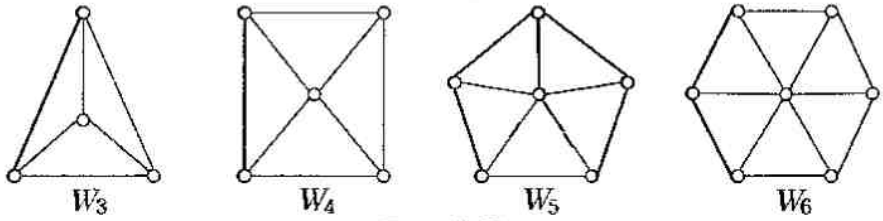


Рис. 3.12

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 і лише тоді, коли бітові рядки, які їх подають, відрізняються точно в одному біті.

Приклад **3.13.** На рис. 3.13 зображено графи Q_1, Q_2 та Q_3.

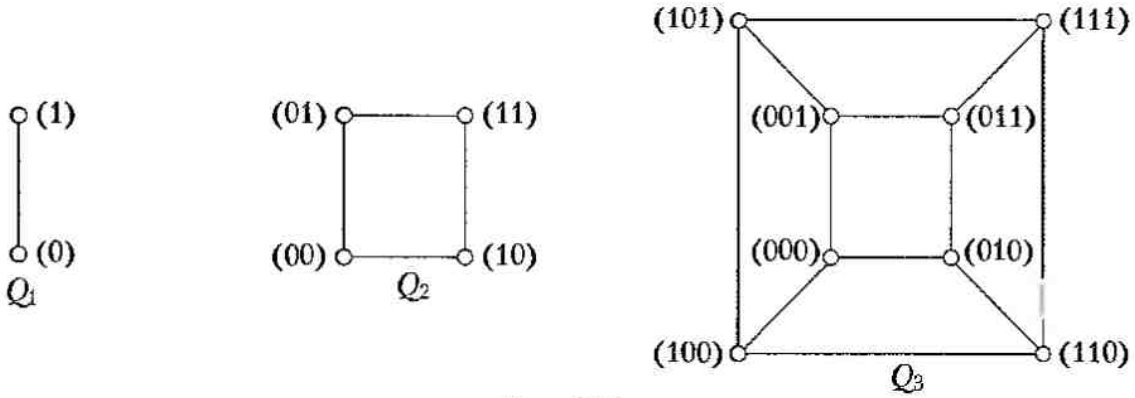


Рис. 3.13

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 зліва дописують 0, другого — 1.

3.3. 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 (рис. 3.14) й орієнтованого (рис. 3.15).

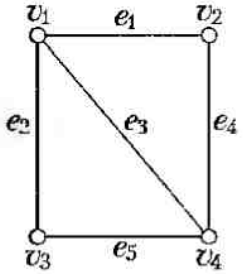


Рис. 3.14

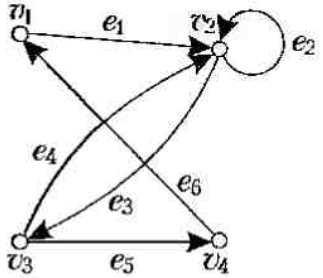


Рис. 3.15

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

3.3.1. Матриця інцидентності

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

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

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

Приклад 3.14. Для графа, зображеного на рис. 3.14, матриця інцидентності має вигляд

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

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 $V=\{v_1, v_2, ..., v_n\}$ і множиною дуг $E=\{e_1, e_2, ..., e_m\}$.

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

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

Приклад 3.15. Для графа, зображеного на рис. 3.15, матриця інцидентності має вигляд

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

З алгоритмічного погляду матриця інцидентності — це, мабуть, найгірший спосіб подання графа, який тільки можна собі уявити [23]. По-перше, для цього

потрібно nn 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 n кроків.

3.3.2. Матриця суміжності

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

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

Приклад 3.16. Матриця суміжності для графа, зображеного на рис. 3.14, має вигляд

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

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 $a_{ii}=0$ $(i=1, \ldots, n)$.

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} $(i, j=1, \ldots, n)$, де

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

Приклад 3.17. Матриця суміжності для графа, зображеного на рис. 3.15, має вигляд

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

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 інший аргумент проти використання матриці суміжності можна навести теорему про кількість кроків алгоритму, який на основі матриці суміжності перевіряє якусь властивість простого графа (див. підрозділ 3.5, теорему 3.9).

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*.

3.3.3. 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 рис. 3.14 та 3.15, списки пар подані відповідно на рис. 3.16, *а* та 3.16, *б*.

v_1	v_2
v_1	v_3
v_1	v_4
v_2	v_4
v_3	v_4

а

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

б

Рис. 3.16

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.

3.3.4. Подання графа списками суміжності

Орієнтований граф G (без кратних дуг, але, можливо, з петлями) можна подати, зазначивши скінченну непорожню множину вершин V та відповідність Г, котра показує, як зв'язані між собою вершини. Відповідність Г — 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 [19, 20].

Приклад 3.18. Для орієнтованого графа, зображеного на рис. 3.15, відповідність Г задано табл. 3.2.

Таблиця 3.2

v	**Г(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 3.19. Для простого графа, зображеного на рис. 3.14 відповідність Г задано табл. 3.3.

Таблиця 3.3

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

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

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 3.20. На рис. 3.17 зображено неорієнтований граф з рис. 3.14 за допомогою списків суміжності. Аналогічне зображення для орієнтованого графа з рис. 3.15 показано на рис. 3.18.

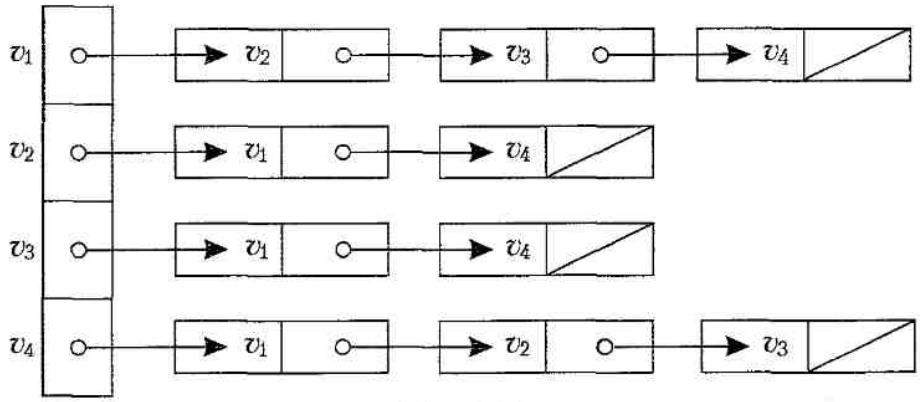


Рис. 3.17

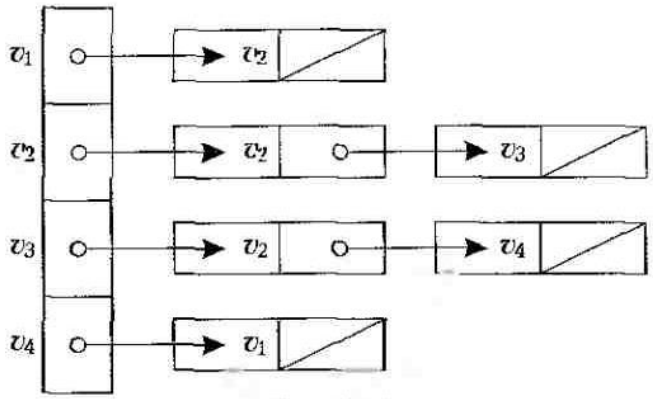


Рис. 3.18

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

3.4. Шляхи та цикли. Зв'язність

Шляхом довжиною r [52] із вершини u в вершину v в неорієнтованому графі називають послідовність ребер $e_1 = \{x_0, x_1\}$, $e_2 = \{x_1, x_2\}$, ..., $e_r = \{x_{r-1}, x_r\}$, 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 собою, тобто $u = v$.

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

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 3.21. На рис. 3.19 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 $\{b, c\}$, $\{c, f\}$, $\{f, e\}$ й $\{e, b\}$ – 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 $\{a, b\}$.

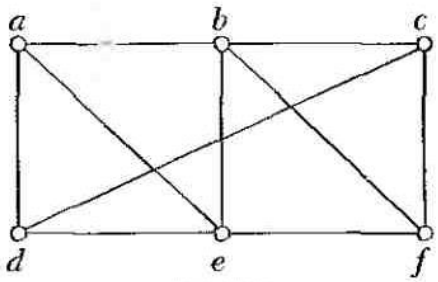


Рис. 3.19

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 3.22. Граф G на рис. 3.20 зв'язний; граф H — незв'язний, оскільки не існує шляху $\langle y, v\rangle$. Граф H має дві компоненти.

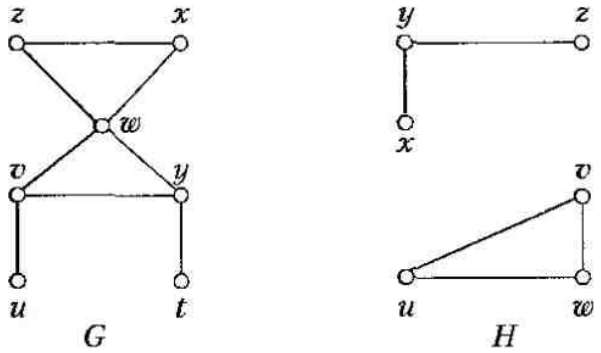


Рис. 3.20

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 3.4. 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 $e_1 = (x_0, x_1)$, $e_2 = (x_1, x_2)$, ..., $e_r = (x_{r-1}, x_r)$, 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 напрямок.

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 слабко зв'язний.

Приклад 3.23. На рис. 3.21 зображено графи G й H. Граф G сильно зв'язний. Граф H слабко зв'язний; він не сильно зв'язний, бо не існує орієнтованого шляху від w_1 до w_2.

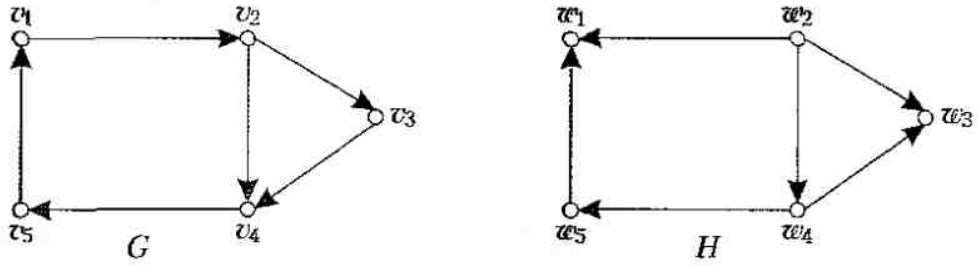


Рис. 3.21

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

ТЕОРЕМА 3.5. Нехай G — граф (орієнтований або неорієнтований; можуть бути також кратні ребра й петлі), A — його матриця суміжності, яка відповідає заданій нумерації вершин $v_1, v_2, ..., v_n$. 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 (див. рис. 3.9). 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*) $\kappa(G)$ простого графа G називають найменшу кількість вершин, вилучення яких дає незв'язний або одновершинний граф. Зазначимо, що вершину вилучають разом із інцидентними їй ребрами. Наприклад, $\kappa(K_1) = 0$, $\kappa(K_n) = n - 1$, $\kappa(C_n) = 2$. Граф G, зображений на рис. 3.22, зв'язний, але його зв'язність можна порушити вилученням вершини u. Отже, $\kappa(G) = 1$. Якщо ж спробувати порушити зв'язність цього графа вилученням ребер (а не вершин), то потрібно вилучити 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, вилучення яких дає незв'язний граф. Число реберної зв'язності одновершинного графа вважають таким, що дорівнює 0. Для графа G, зображеного на рис. 3.22, $\lambda(G) = 3$.

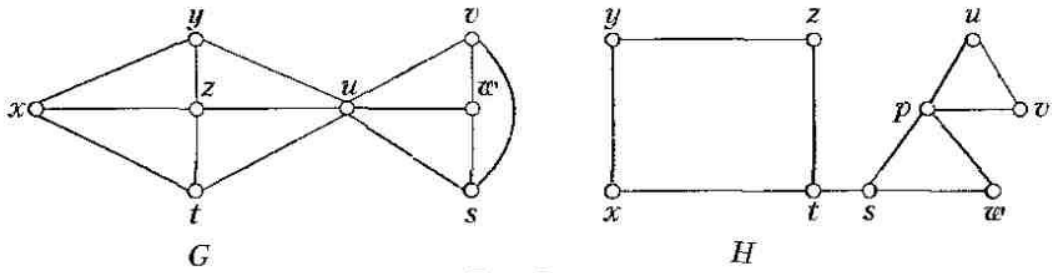


Рис. 3.22

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

Приклад 3.24. Граф H, зображений на рис. 3.22, 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 $\kappa(G) \leqslant \lambda(G) \leqslant \delta(G)$. Простий граф G називають *t-зв'язним*, якщо $\kappa(G) \geqslant t$, і *реберно t-зв'язним*, якщо $\lambda(G) \geqslant t$. 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 3.25. Граф G, зображений на рис. 3.22, 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 3.6. Якщо простий граф G має n вершин і k компонент, то кількість m його ребер задовольняє нерівності

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

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, $p \geqslant q > 1$, 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, $0 \leqslant 0$. 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: $n-(k-1)\leqslant m_{\min}-1$, звідки випливає нерівність $n-k\leqslant m_{\min}$.

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

ТЕОРЕМА 3.7 (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. повернемось 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 $V=A\cup B$ ($A\cap B=\varnothing$) 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 (див. підрозділ 3.1) $G(A)$ 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 (рис. 3.23). 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.

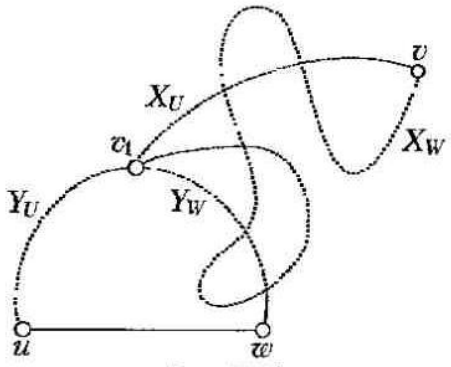


Рис. 3.23

Доведення теореми Кеніга підказує простий спосіб розпізнавання дводольності графа, що ґрунтується на простому алгоритмі, називаному пошуком ушир [15]. 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 почергово оточення всіх вершин із номером 1, і всім вершинам, що нале-

жать 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 (див. підрозділ 3.9).

3.5. Ізоморфізм графів

У теорії графів і її застосуваннях істотно, що існують об'єкти (вершини графа) і зв'язки між ними (ребра). Тому доцільно не розрізняти графи, котрі можна одержати один з одного зміною позначень вершин. Сформулюємо ці міркування у вигляді такого означення.

Нехай $G_1=(V_1, E_1)$ і $G_2=(V_2, E_2)$ — прості графи, а φ: $V_1 \to V_2$ — бієкція. Якщо для будь-яких вершин u та v графа G_1 їх образи $\varphi(u)$ та $\varphi(v)$ суміжні в G_2 тоді й лише тоді, коли u та v суміжні в G_1, то цю бієкцію називають *ізоморфізмом* графа G_1 на граф G_2, а графи G_1 і G_2 — *ізоморфними* (записують $G_1 \cong G_2$).

Отже, прості графи G_1 і G_2 ізоморфні, якщо існує така бієкція $\varphi: V_1 \to V_2$, що

$$\forall\, u, v \in V_1\ (\{u, v\} \in E_1 \text{ тоді й лише тоді, коли } \{\varphi(u), \varphi(v)\} \in E_2).$$

Приклад 3.26. Графи на рис. 3.24 ізоморфні; бієкцію φ можна задати так: $\varphi(x_1)=y_1$; $\varphi(x_2)=y_4$; $\varphi(x_3)=y_3$; $\varphi(x_4)=y_2$.

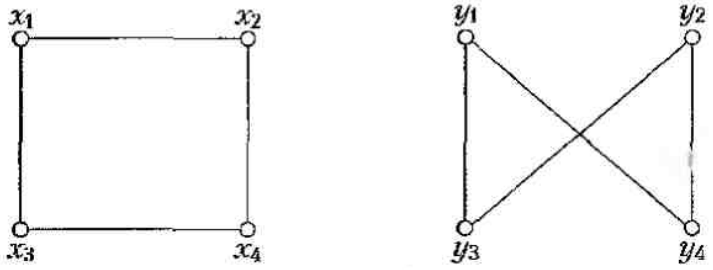


Рис. 3.24

Приклад 3.27. Усі три графи, зображені на рис. 3.25, ізоморфні. Довести це пропонуємо як вправу.

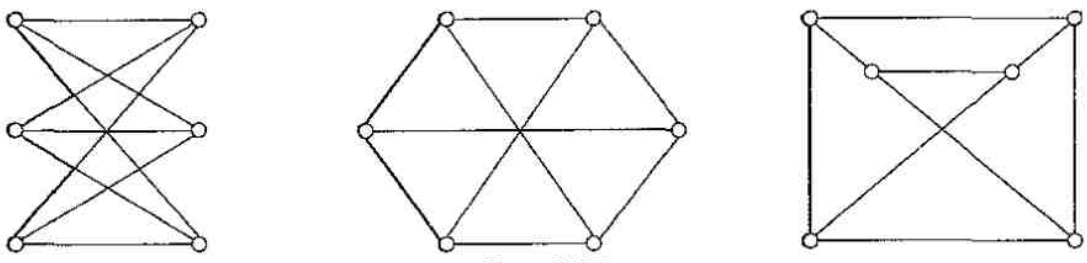


Рис. 3.25

Ізоморфні графи природно ототожнювати (їх можна зобразити одним рисунком). Вони могли б різнитися природою своїх елементів, але це не беруть до уваги, уводячи поняття „граф". Проте інколи все ж доводиться розрізняти ізоморфні графи, і тоді корисне поняття „позначеного графа". Граф з n вершинами називають *позначеним*, якщо його вершинам присвоєно якісь мітки, наприклад, числа 1, 2, ..., n. Ототожнимо кожну з вершин з її номером (і, отже, множину вершин — із множиною чисел $\{1, 2, ..., n\}$) й означимо рівність простих позначених графів G_1 і G_2 з однаковою кількістю вершин n так: $G_1 = G_2$ тоді й лише тоді, коли $E_1 = E_2$.

Приклад 3.28. На рис. 3.26 зображено три різні позначені графи.

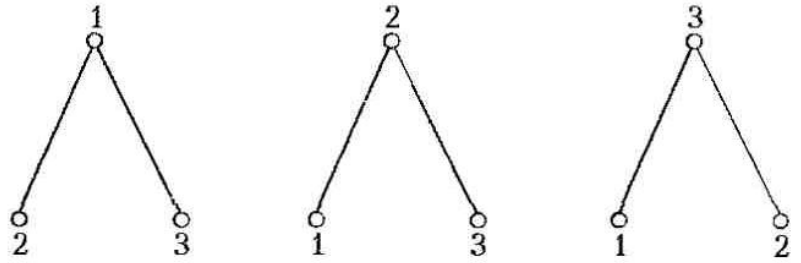


Рис. 3.26

Щоб наголосити, що графи розрізняють лише з точністю до ізоморфізму, говорять про абстрактний граф. Останній має різні матриці суміжності залежно від нумерації вершин. З'ясуємо, як пов'язані між собою ці матриці. Нехай G_1 і G_2 — прості позначені графи з n вершинами, і $G_1 \cong G_2$. Тоді ці графи різняться лише нумерацією вершин, тобто існує підстановка s на множині вершин, яка *зберігає суміжність*: вершини u та v тоді й лише тоді суміжні в графі G_1, коли їх образи $s(u)$ та $s(v)$ суміжні в G_2. Нехай $A = [a_{ij}]$ і $B = [b_{ij}]$ — матриці суміжності відповідно графів G_1 та G_2. Тоді $b_{s(i)s(j)} = a_{ij}$ $(i, j = 1, 2, ..., n)$. Тим самим доведено таку теорему.

ТЕОРЕМА 3.8. Прості графи ізоморфні тоді й лише тоді, коли їх матриці суміжності можна отримати одну з одної однаковими перестановками рядків і стовпців.

Виявити ізоморфізм дуже складно. Теоретично алгоритм перевірки пари простих графів на ізоморфізм існує — його сформульовано в попередній теоремі. Проте він незручний, бо для його виконання може бути потрібно до $n!$ перестановок і перевірок.

Часто неважко довести, що два прості графи не ізоморфні, якщо порушується властивість, інваріантна щодо ізоморфізму, наприклад:

- кількість вершин;
- кількість ребер;
- кількість вершин конкретного степеня (вершині $v \in V_1$, $\deg(v) = d$, має відповідати вершина $u = \varphi(v) \in V_2$, $\deg(u) = d$).

Є й інші інваріанти, але порушення інваріантності — це лише достатня умова неізоморфності графів. Не існує набору інваріантів для виявлення ізоморфізму.

Приклад 3.29. Графи на рис. 3.27 неізоморфні. Вони мають по п'ять вершин і по шість ребер, проте граф G_2 має вершину степеня 1, якої не має граф G_1.

Рис. 3.27

Приклад 3.30. На рис. 3.28 зображено графи G_1 і G_2. Обидва вони мають по вісім вершин і по десять ребер. Вони також мають по чотири вершини степеня 2 і по чотири вершини степеня 3. *Однак ці графи не ізоморфні. Справді, позаяк* $\deg(x_i)=2$ *в* G_1, *то вершині* x_1 має відповідати одна із чотирьох вершин y_2, y_4, y_6, y_8 у графі G_2. Зазначені вершини мають у графі G_2 степінь 2. Проте кожна з цих чотирьох вершин суміжна з іншою вершиною степеня 2, що не виконується для вершини x_i у графі G_1.

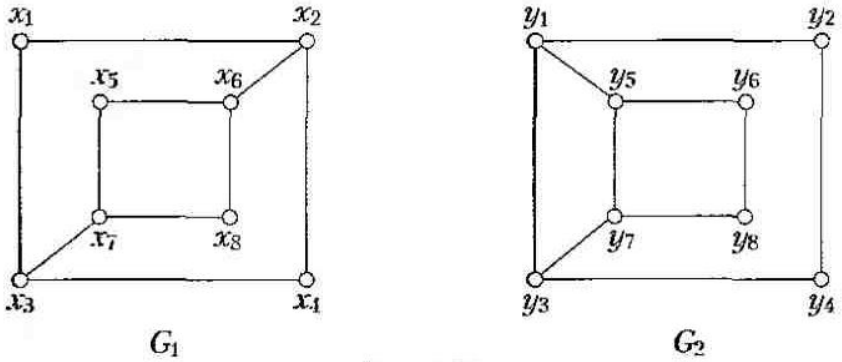


Рис. 3.28

Те, що графи, зображені на рис. 3.28, не ізоморфні, можна довести й інакше. На рис. 3.29 зображено підграфи графів G_1 і G_2, породжені вершинами степеня 3. Якщо графи G_1 і G_2 ізоморфні, то й зазначені підграфи мають бути ізоморфними. Проте підграфи з рис. 3.29 не ізоморфні.

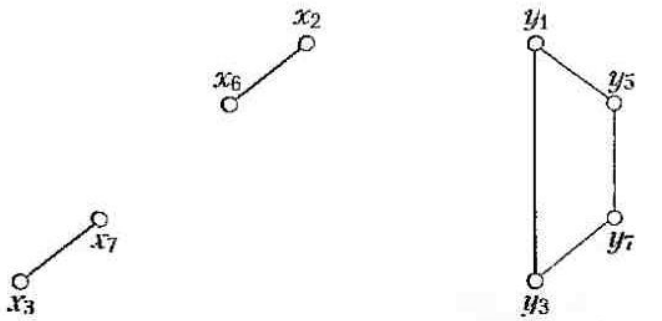


Рис. 3.29

Ми розглянули поняття ізоморфізму для простого графа. Для неорієнтованих мультиграфів і псевдографів, а також орієнтованих графів природно вводять

поняття ізоморфізму як бієкції між множинами вершин, яка зберігає суміжність, кратності ребер, петлі та спрямування дуг. Теорема 3.8 залишається правильною для мультиграфів, псевдографів і орієнтованих графів.

Розглянемо тепер один результат, пов'язаний з оцінкою складності алгоритмів на графах, які задано матрицею суміжності. Нехай P — якась властивість простого графа: $P(G)=1$ або $P(G)=0$ залежно від того, чи має граф G цю властивість. Припустимо, що властивість P задовольняє такі умови.

1. $P(G)=P(G')$, якщо графи G та G' ізоморфні.
2. $P(G)=0$ для довільного порожнього графа й $P(G)=1$ для довільного повного графа з достатньо великою кількістю вершин.
3. Додавання ребра не порушує властивості P, тобто $P(G)\leqslant P(G')$ для довільних простих графів $G=(V, E)$ та $G'=(V, E')$ таких, що $E\subset E'$.

Приклад такої властивості P — існування простого циклу в графі, який має принаймні три вершини.

ТЕОРЕМА 3.9. Якщо P — властивість простого графа G, що задовольняє умови 1–3, то будь-який алгоритм, котрий перевіряє властивість P (тобто обчислює значення $P(G)$ для графа G), виходячи з матриці суміжності, виконує в найгіршому випадку $\Omega(n^2)$ кроків, де n — кількість вершин графа.

3.6. 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 рис. 3.30. 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 рис. 3.31.

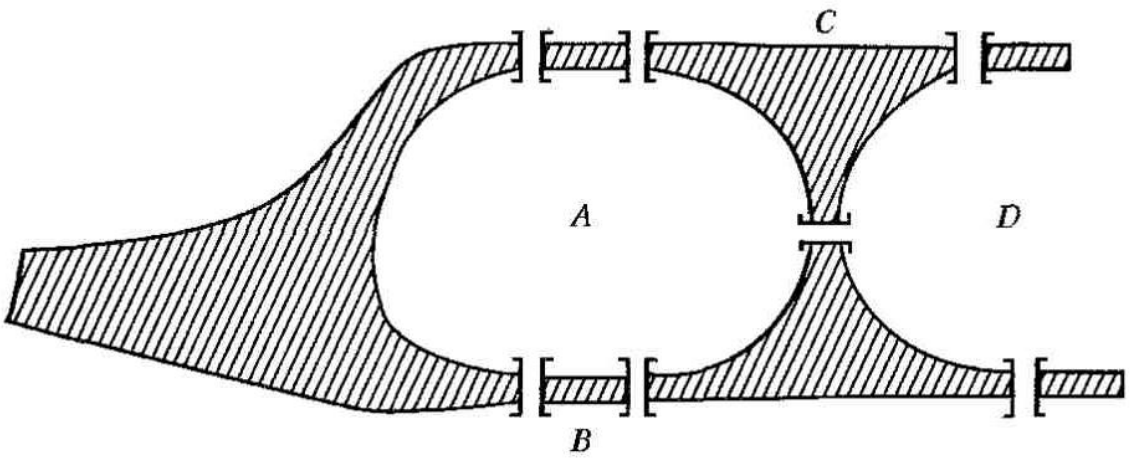


Рис. 3.30

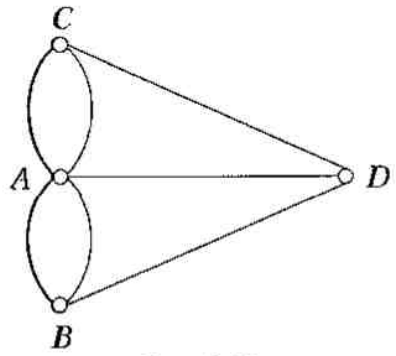


Рис. 3.31

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, що містить усі ребра графа.

Приклад 3.31. На рис. 3.32 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.

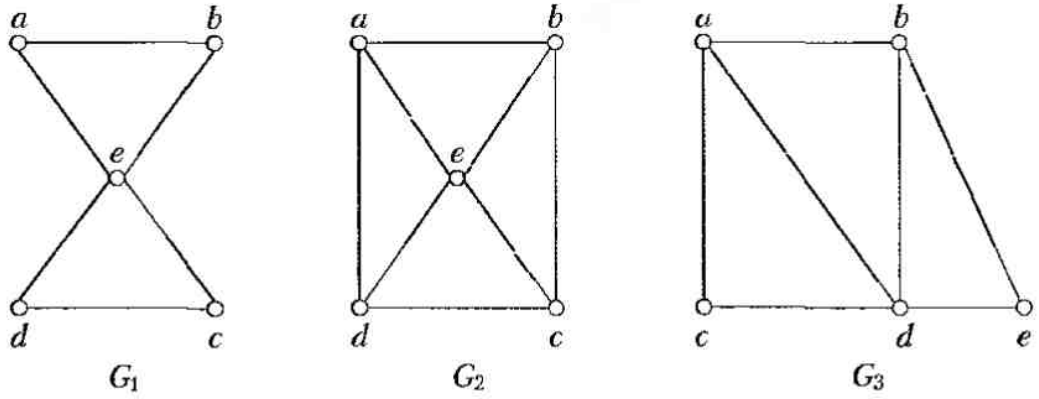


Рис. 3.32

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

ТЕОРЕМА 3.10. 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 рис. 3.33.

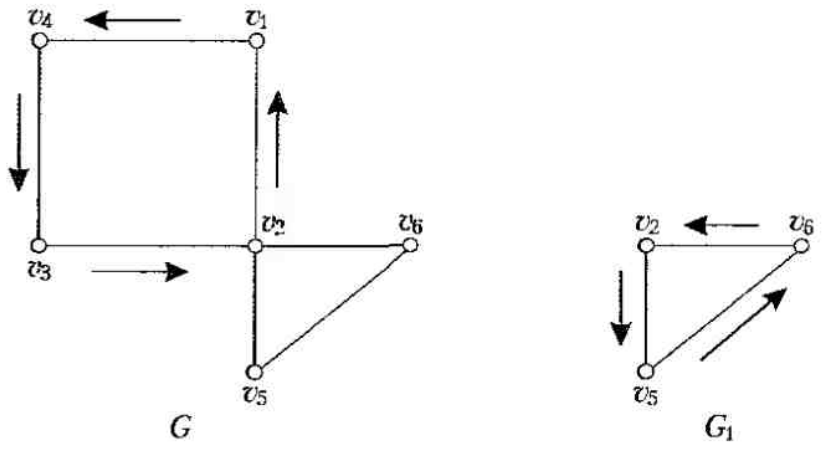


Рис. 3.33

Цикл $P_3 = v_1, v_4, v_3, P_2, v_1 = v_1, v_4, v_3, v_2, v_5, v_6, v_2, v_1$ 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. Вибираємо 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 рис. 3.31, 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 3.11. 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 слабко зв'язному орієнтованому мультиграфі називають орієнтований простий цикл, який містить усі ребра графа.

ТЕОРЕМА 3.12. Орієнтований слабко зв'язний мультиграф має ейлерів цикл тоді й лише тоді, коли напівстепінь входу кожної вершини дорівнює її напівстепеню виходу.

3.7. Гамільтонів цикл у графі

Шлях $x_0, x_1, ..., x_{n-1}, x_n$ у зв'язному графі $G=(V, E)$ називають *гамільтоновим шляхом*, якщо $V=\{x_0, x_1, ..., x_{n-1}, x_n\}$ і $x_i \neq x_j$ для $0 \leqslant i < j \leqslant n$. Цикл $x_0, x_1, ..., x_{n-1}, x_n, x_0$ (тут $n > 1$) у графі $G=(V, E)$ називають *гамільтоновим циклом*, якщо $x_0, x_1, ..., x_{n-1}, x_n$ — 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 [52].

Приклад 3.32. Ту саму задачу можна зобразити й на площині (рис. 3.34). Вона зводить-ся до відшукання в графі гамільтонового циклу. Один із можливих розв'язків показано потовщеними лініями.

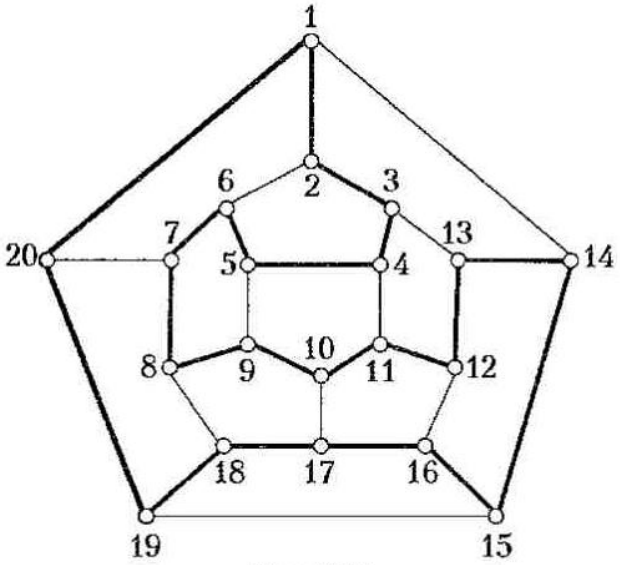


Рис. 3.34

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 рис. 3.35, 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.

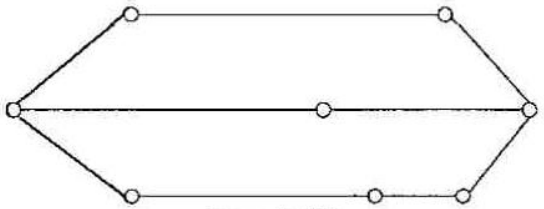


Рис. 3.35

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

Доведення. Додамо до графа G k нових вершин і з'єднаємо ребром кожну з них з 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 $k \geqslant 1$ призводить до суперечності.

Нехай $v, p, w, ..., v$ — 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 $v, p, w, ..., v', w', ..., v$, то ми можемо замінити його на $v, v', ..., w, w', ..., v$, повернувши частину циклу між w та v'. Це знову суперечить мінімальності числа k (рис. 3.36).

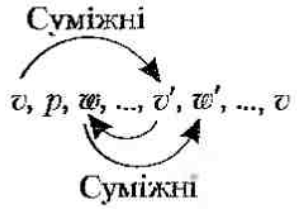


Рис. 3.36

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 водночасно 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. Розгляд реального алгоритму бектрекінг відкладемо до наступного розділу (розділ 4.5, приклад 4.15).

Задача знаходження гамільтонового циклу *NP*-повна (див. розділ 10). 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 *гамільтоновим графом*.

3.8. Зважені графи й алгоритми пошуку найкоротших шляхів

У реальних задачах на графах часто потрібно брати до уваги додаткову інформацію — фактичну відстань 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) уважають 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 замінюємо 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 $v \neq a$ й уважати ці мітки тимчасовими. Виконати $x := a$, $M := \{a\}$.

Крок 2. Оновлення міток. Для кожної вершини $v \in \Gamma(x) \setminus M$ замінити мітки: $l(v) := \min\{l(v); l(x) + w(x, v)\}$, 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-й 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-ї ітерації.

Розглянемо ситуацію перед початком k-ї 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 $w(P^*)\geq w(P_1^*)\geq$ $\geq l(v)=w(P^0)$, 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 $O(n^2)$ [2, 23].

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 і θ.

Приклад 3.33. Знайдемо довжину найкоротшого шляху від початкової вершини a до вершини z у графі на рис. 3.37, *а*. Послідовність дій зображено на рис. 3.37, *б–е*, 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 θ вершин, з яких у дану вершину безпосередньо потрапляє найкоротший шлях. У табл. 3.4 в першому рядку містяться довільно впорядковані вершини графа, у другому — відповідні постійні мітки (компоненти вектора l), а в третьому — компоненти вектора θ. 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 θ). 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.

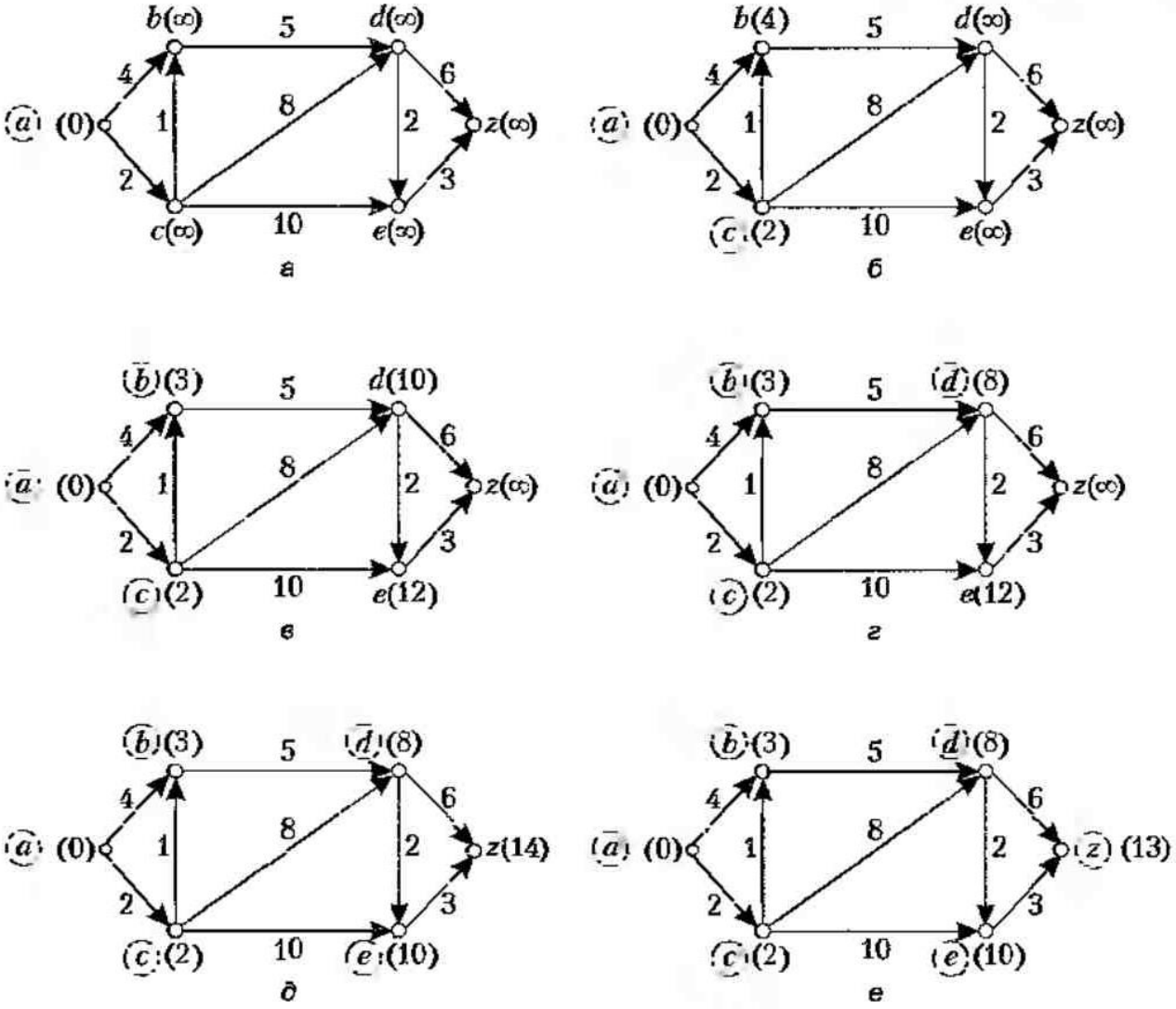


Рис. 3.37

Таблиця 3.4

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

Ми розглянули задачу відшукання 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 $G = (V, \Gamma)$ — орієнтований граф. Нагадаємо, що внутрішні вершини шляху $a, x_1, x_2, ..., x_{m-1}, b$ в графі G — $x_1, x_2, ..., x_{m-1}$. 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,

$$w_{ij}^{(k)} = \min\{w_{ik}^{(k-1)} + w_{kj}^{(k-1)}, w_{ij}^{(k-1)}\}. \qquad (3.1)$$

Зі співвідношення (3.1) 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, ..., n, визначити за елементами матриці $W^{(k-1)}$ елементи матриці $W^{(k)}$, використовуючи рекурентне співвідношення (3.1).

Після закінчення цієї процедури (i, j)-елемент матриці $W^{(n)}$ 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 $W^{(1)}$, $W^{(2)}$, ..., $W^{(n)}$. Окрім того, для всіх $i = 1, 2, ..., n$ 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 формулою (3.1) 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 $i, j = 1, ..., n$, $i \neq j$; $\theta_{ii}^{(0)} := 0$. 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 (i, j)-шляху, тобто найкоротшому $\langle i, j\rangle$-шляху, усі внутрішні вершини якого містяться в множині $\{1, 2, ..., k\}$. Матриця $\Theta^{(n)} = [\theta_{ij}^{(n)}]$ відіграє ту саму роль, що й вектор θ в алгоритмі Дейкстри. За допомогою матриці $\Theta^{(n)}$ вершини, через які проходить найкоротший $\langle i, j\rangle$-шлях, визначають так: $i, ..., j_3, j_2, j_1, j$, де $j_1 = \theta_{ij}^{(n)}$, $j_2 = \theta_{ij_1}^{(n)}$, $j_3 = \theta_{ij_2}^{(n)}$, ...

Приклад 3.34. Знайти найкоротші шляхи між усіма парами вершин орієнтованого графа на рис. 3.38.

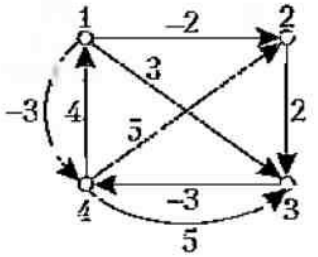


Рис. 3.38

Нижче наведено результати виконання кожної з чотирьох ітерацій алгоритму Флойда:

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

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

$$W^{(2)} = \begin{pmatrix} 0 & -2 & 0 & -3 \\ \infty & 0 & 2 & \infty \\ \infty & \infty & 0 & -3 \\ 4 & 2 & 4 & 0 \end{pmatrix}, \qquad \Theta^{(2)} = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 2 & 2 \\ 3 & 3 & 0 & 3 \\ 4 & 1 & 2 & 0 \end{pmatrix},$$

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

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

Матриця $W^{(4)}$ дає довжини найкоротших шляхів між усіма парами вершин графа на рис. 3.38. Зокрема, $a_{21}^{(4)} = 3$, 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 ураховувати, що неорієнтоване ребро із від'ємною вагою одразу приводить до циклу із від'ємною довжиною, що робить алгоритм Флойда незастосовним.

3.9. 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. Алгоритми обходу вершин графа називають *методами пошуку* [23].

3.9.1. 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 вглиб 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" (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 вглиб 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, додатково 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 *протоколом обходу* графа пошуком вглиб.

Приклад 3.35. Виконаємо обхід графа на рис. 3.39 пошуком вглиб, починаючи з вершини b. Розв'язок подано на рис. 3.40; протокол такого пошуку подано в табл. 3.5. У цій таблиці в третьому стовпці вважаємо, що верхівка стеку праворуч.

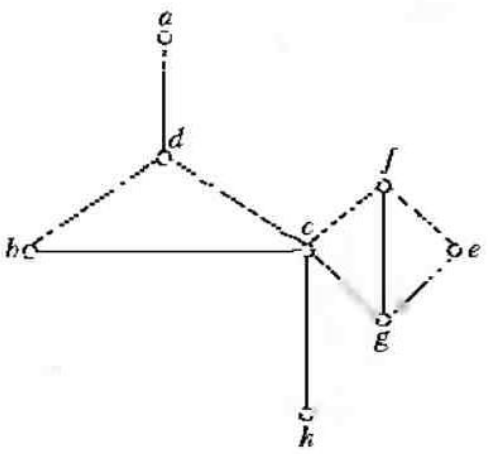


Рис. 3.39

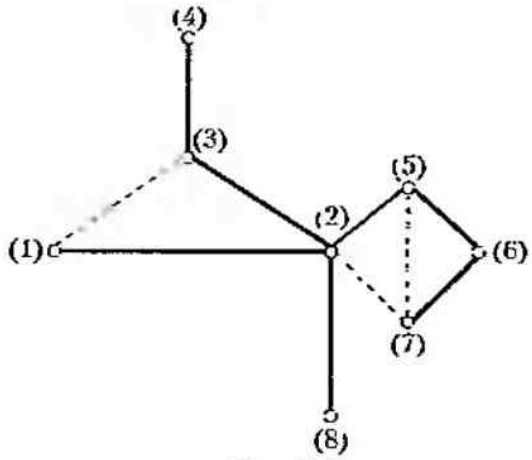


Рис. 3.40

Таблиця 3.5

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*
–	–	*bcfe*
–	–	*bcf*
–	–	*bc*
h	8	*bch*
–	–	*bc*
–	–	*b*
–	–	∅

3.9.2. Пошук ушир у простому зв'язному графі

У процесі *пошуку вшир* вершини графа проглядають 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 сусіди 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" (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 вершину 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 3.36. Виконаємо обхід графа на рис. 3.39 пошуком вшир, починаючи з вершини b. Розв'язок зображено на рис. 3.41, протокол пошуку вшир подано в табл. 3.6.

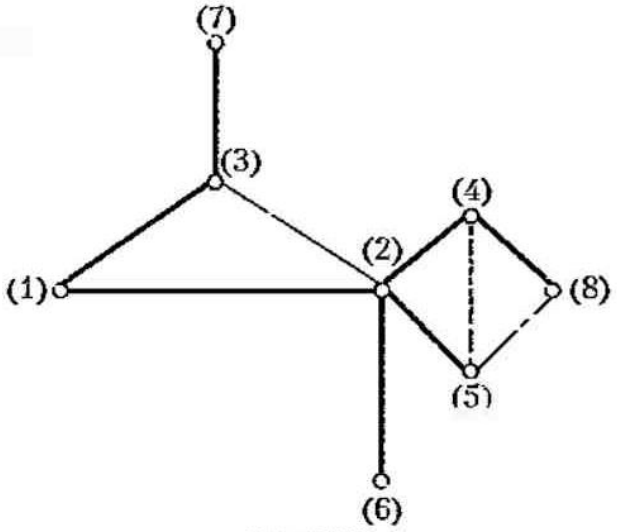


Рис. 3.41

Таблиця 3.6

Вершина	BFS-номер	Вміст черги
b	1	*b*
c	2	*bc*
d	3	*bcd*
-	–	*cd*
f	4	*cdf*
g	5	*cdfg*
h	6	*cdfgh*
–	–	*dfgh*
a	7	*dfgha*
–	–	*fgha*
e	8	*fghae*
–	–	*ghae*
–	–	*hae*
–	–	*ae*
–	–	*e*
–	–	∅

У процесі роботи наведених алгоритмів будується *дерево T пошуку* відповідно *вглиб* і *вшир* — на рис. 3.40 і 3.41 його виділено потовщеними лініями (див. розділ 4).

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.

3.10. 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 3.37. Усі три графи на рис. 3.42 планарні, але лише другий і третій із них плоскі.

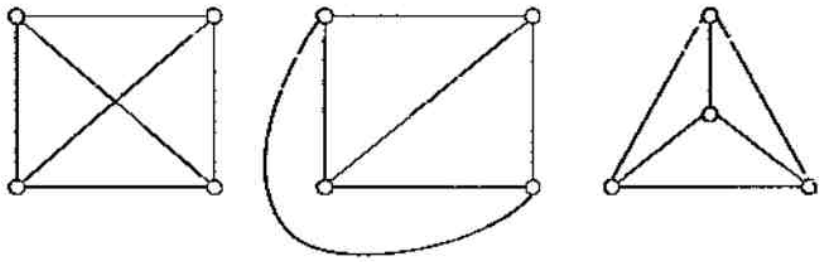


Рис. 3.42

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

Приклад 3.38. На рис. 3.43 зображено плоский граф. Він має чотири грані: r_1, r_2, r_3, r_4; грань r_4 зовнішня.

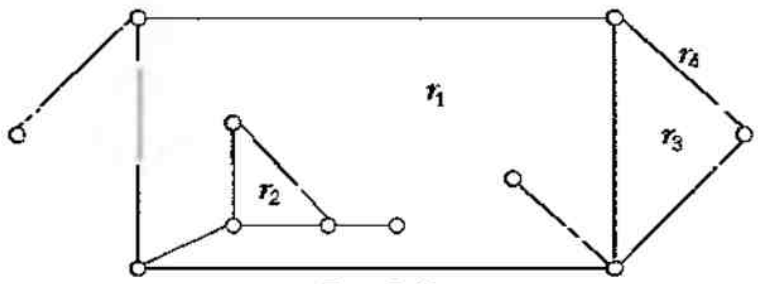


Рис. 3.43

ТЕОРЕМА 3.14 (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 3.15. Графи K_5 і $K_{3,3}$ непланарні.

Доведення. Граф K_5 має п'ять вершин і десять ребер (див. рис. 3.9). 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 $3r \leqslant 2m$, звідки випливає, що $21 \leqslant 20$. Суперечність.

У графі $K_{3,3}$ кількість вершин $n = 6$, а кількість ребер $m = 9$ (див. рис. 3.10). 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, $4r \leqslant 2m$, звідки випливає, що $10 \leqslant 9$. Суперечність.

Два графи називаються *гомеоморфними*, якщо їх можна отримати з одного графа додаванням до його ребер нових вершин степеня 2 (рис. 3.44).

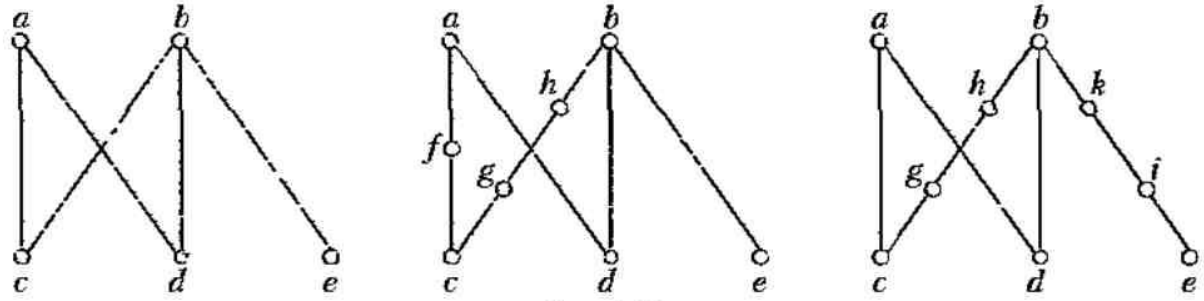


Рис. 3.44

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

ТЕОРЕМА 3.16 (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 3.39. На рисунку 3.45 зображено *граф Петерсена*, а на рис 3.46 — його підграф, гомеоморфний $K_{3,3}$. Отже, за теоремою Куратовського, граф Петерсена непланарний.

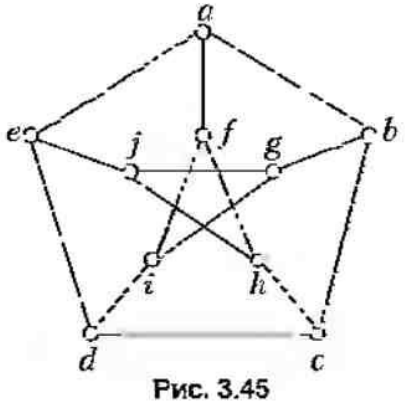


Рис. 3.45

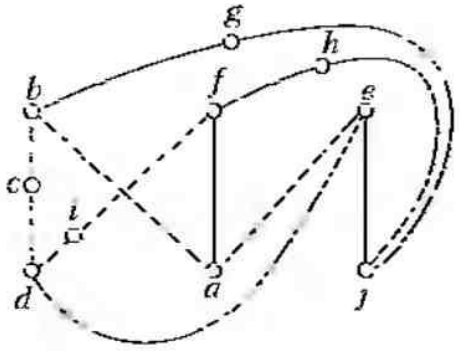


Рис. 3.46

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) [15].

3.11. 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-розфарбування*) для будь-якого k в діапазоні $\chi(G) \le k \le n$, 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 (про незалежні множини вершин див. підрозділ 3.12).

Очевидно, що $\chi(K_n) = n$, і, отже, легко побудувати графи 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 *бінарним*).

ТЕОРЕМА 3.17. Якщо найбільший зі степенів вершин графа дорівнює ρ, 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 ρ. 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 ρ).

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 $\chi(G) \leqslant 4$ для будь-якого планарного графа G.

Перше з помилкових „доведень" належить А. Кемпе (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 3.18 (Р. Хейвуда). Будь-який планарний граф можна розфарбувати в п'ять кольорів, тобто $\chi(G) \leqslant 5$ 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 (див. задачу 65).

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 $n \geqslant 5$. 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. $|W| \leqslant 4$. 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 неплanарний. Граф G', одержаний із графа $G - v_0$ злиттям вершин v_1 і v_2 в одну вершину v (рис. 3.47), 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, приплишемо вершині v_0 колір, не використаний для розфарбовування вершин із W.

Американські математики К. Аппель (K. Appel) і У. Гайкен (W. Haken) 1976 р. довели гіпотезу чотирьох фарб, суттєво використавши комп'ютерні обчислення. Це перший випадок, коли настільки відому математичну проблему було розв'язано за допомогою комп'ютера [21]. 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 [15, 21], 1976 р. 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 [15].

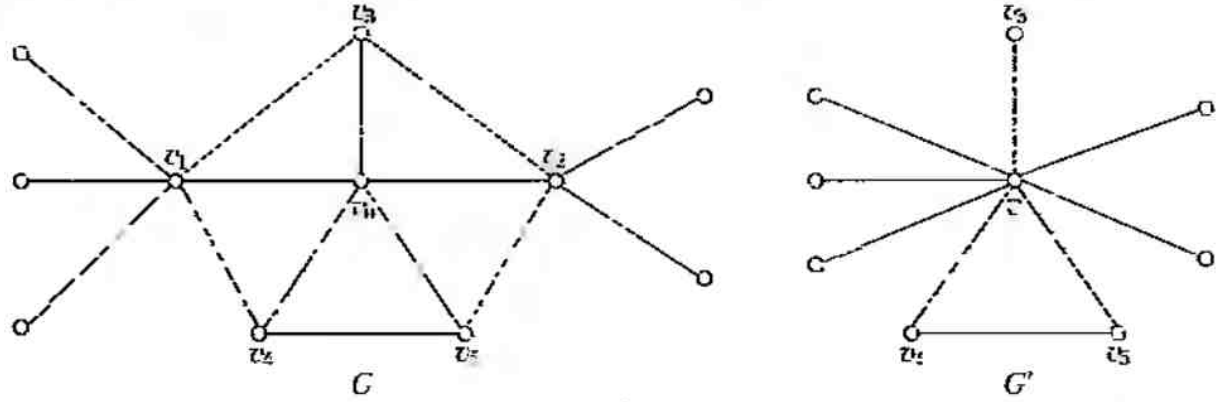


Рис. 3.47

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 $P_G(k) = k(k-1)(k-2)...$ $...(k-n+1)$. 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.

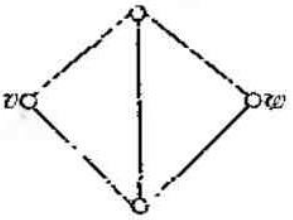
Якщо $k < \chi(G)$, то $P_G(k) = 0$, а для $k \geq \chi(G)$ виконується нерівність $P_G(k) > 0$. Гіпотеза чотирьох фарб еквівалентна такому твердженню: якщо G — простий планарний граф, то $P_G(4) > 0$. Якщо задано довільний простий граф, то в загальному випадку важко отримати його хроматичну функцію 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 3.19. *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.

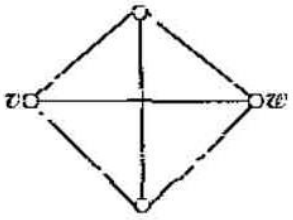
Отже, ця кількість дорівнює $P_{G_2}(k)$. Залишилося застосувати комбінаторне правило суми. Теорему доведено.

Приклад 3.40. На рис. 3.48 зображено граф G, а на рис 3.49 – графи G_1 і G_2.

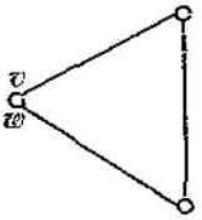


G

Рис. 3.48



G_1



G_2

Рис. 3.49

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* $-m$, де m – кількість ребер графа G; знаки коефіцієнтів чергуються; вільний член хроматичного полінома дорівнює 0.

Хроматичний поліном будують на основі теореми 3.19 у вигляді суми хроматичних поліномів повних графів.

Приклад 3.41. На рис 3.50 зображено процес побудови хроматичного полінома, де знаки „ = " та „ + " мають умовний зміст. Отже,

$$P_G(k) = k(k-1)(k-2)(k-3) + 2k(k-1)(k-2) + k(k-1) = k^4 - 4k^3 + 6k^2 - 3k.$$

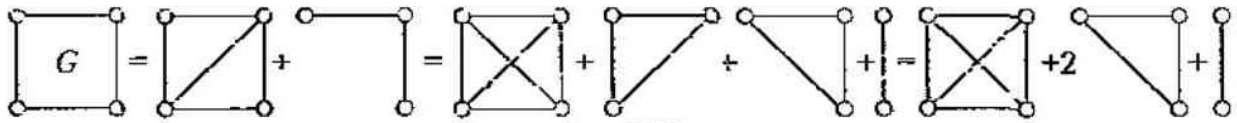


Рис. 3.50

Нарешті, розглянемо деякі практичні задачі, які зводяться до розфарбовування графів.

Задача складання розкладу

Припустимо, що потрібно прочитати декілька лекцій у найкоротший термін. Кожна лекція окремо займає одну годину, але деякі лекції не можна читати одночасно (наприклад, якщо це робить один і той самий лектор). Побудуємо граф G, вершини якого взаємно однозначно відповідають лекціям, і дві вершини суміжні тоді й лише тоді, коли відповідні лекції не можна читати одночасно. Очевидно, що будь-яке розфарбовування цього графа задає можливий розклад: лекції, які відповідають вершинам одноколірного класу, читають одночасно. Навпаки, будь-який можливий розклад задає розфарбовування графа G. Оптимальні розклади відповідають розфарбовуванням мінімальною кількістю кольорів, а час, потрібний для читання всіх лекцій, дорівнює $\chi(G)$.

Задача розподілу обладнання

Задано множини $V = \{v_1, ..., v_n\}$ і $S = \{s_1, ..., s_m\}$ відповідно робіт і механізмів. Для виконання кожної роботи потрібен певний час, однаковий для всіх робіт, і якісь

механізми. Жоден із механізмів не може бути зайнятий на декількох роботах одночасно. Потрібно розподілити механізми так, щоб загальний час виконання всіх робіт був мінімальним. Побудуємо граф G з множиною вершин V, вершини v_i та v_j $(i \neq j)$ якого суміжні тоді й лише тоді, коли для виконання робіт v_i та v_j потрібний хоча б один спільний механізм. Розфарбуємо граф G. Роботи, що відповідають вершинам одного кольору, можна виконувати одночасно, а мінімальний час виконання всіх робіт відповідає розфарбуванню мінімальною кількістю кольорів.

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.

3.12. 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*, *нещільністю*) і позначають $\alpha(G)$.

Приклад 3.42. У графі, зображеному на рис. 3.51, 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, $\alpha(G) = 4$.

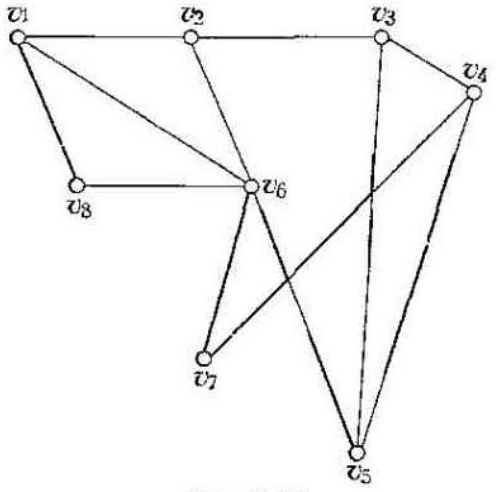


Рис. 3.51

Із поняттям незалежності в графі пов'язане поняття домінування [15]. 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 відшуканні 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 3.43. У графі, зображеному на рис. 3.51, кожна з множин $X_1 = \{v_1, v_3, v_4, v_6\}$, $X_2 = \{v_1, v_3, v_4, v_5, v_6\}$, $X_3 = \{v_2, v_4, v_5, v_6, v_8\}$, $X_4 = \{v_1, v_2, v_3, v_5, v_7, v_8\}$ 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 3.20. 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 $|Y| + |X| = |V|$, то, очевидно, найбільшим Y відповідають найменші X і навпаки.

Протилежне до поняття незалежної множини поняття кліки. Підмножину $V' \subset V$ вершин графа G називають *кліко*ю, 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 $\overline{G}$ 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 3.21. Підмножина вершин графа G являє собою кліку тоді й лише тоді, коли вона незалежна в доповнювальному графі $\overline{G}$. Отже, $\varphi(G) = \alpha(\overline{G})$.

Доведення випливає з означень.

Із теореми 3.21 випливає, що побудову максимальної кліки можна звести до побудови максимальної незалежної множини вершин. Метод побудови максимальних незалежних множин розглянуто в підрозділі 4.5 (приклад 4.19).

3.13. Паросполучення в графах. Теорема Холла

Теорема Холла має багато різних застосувань, три з яких ми розглянемо перед тим, як сформулювати цю теорему.

1. **Задача про весілля.** Розглянемо множину юнаків, кожний з яких знайомий із кількома дівчатами (табл. 3.7). Потрібно визначити умови, за яких кожен з юнаків міг би одружитися зі знайомою йому дівчиною.

Таблиця 3.7

Юнак	Дівчата, з якими знайомий юнак
b_1	g_1, g_4, g_5
b_2	g_1
b_3	g_2, g_3, g_4
b_4	g_2, g_4

2. **Трансверсаль.** Нехай $S = \{S_1, ..., S_m\}$ — сім'я підмножин скінченної множини M; S_i $(i = 1, ..., m)$ необов'язково різні та можуть перетинатись. *Системою різних представників* S (або *трансверсаллю* S) називають підмножину $C = \{c_1, ..., c_m\}$ з m таких елементів множини M, що $c_i \in S_i$. Потрібно визначити умови, за яких існує трансверсаль.

3. **Досконале паросполучення.** *Паросполученням* (або *незалежною множиною ребер*) у простому графі $G = (V, E)$ називають множину ребер, у якій ніякі два ребра не суміжні. Паросполучення графа G називають *максимальним*, якщо воно не міститься в жодному паросполученні з більшою кількістю ребер, і *найбільшим*, якщо кількість ребер у ньому найбільша серед усіх паросполучень графа G.

   Нехай $G = (V, E)$ — дводольний граф ($V = V_1 \cup V_2$, $V_1 \cap V_2 = \varnothing$, кінці ребер належать різним множинам V_1, V_2). *Досконалим паросполученням* із V_1 у V_2 називають паросполучення, яке *покриває* вершини V_1 (тобто всі вершини з V_1 інцидентні ребрам, що утворюють паросполучення). За яких умов існує досконале паросполучення з V_1 у V_2?

Можна довести, що задачі 1–3 — це по суті, одна й та сама задача. Справді, задача 1 зводиться до задачі 3 так. Нехай V_1 — множина юнаків, V_2 — множина

дівчат, ребра — знайомства юнаків із дівчатами (рис. 3.52). У такому разі досконале паросполучення у дводольному графі — це шукана множина весіль (один із можливих розв'язків на рисунку зображено потовщеними лініями).

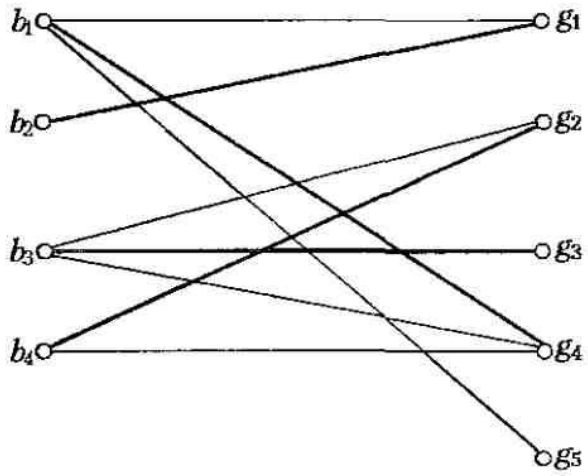


Рис. 3.52

Задача 2 зводиться до задачі 3 так. Нехай $V_1 = S$, $V_2 = M$; ребро $\{S_k, c_i\}$ існує, якщо $c_i \in S_k$. Тоді досконале паросполучення — це шукана трансверсаль.

Отже, задачі 1–3 мають спільну відповідь [35]:

$$\text{тоді й лише тоді, коли будь-які } k \begin{Bmatrix}\text{юнаків}\\ \text{підмножин}\\ \text{вершин із } V_1\end{Bmatrix}$$

$$\text{у сукупності} \begin{Bmatrix}\text{знайомі з}\\ \text{містять}\\ \text{суміжні з}\end{Bmatrix} \text{ не менше ніж } k \begin{Bmatrix}\text{дівчатами}\\ \text{елементів}\\ \text{вершинами з } V_2\end{Bmatrix}$$

$$\text{для всіх } k = 1, 2, \ldots, m.$$

Цю загальну схему обгрунтовує така теорема.

ТЕОРЕМА 3.22 (Ф. Холла). Нехай $G = (V, E)$ — дводольний граф, $V = V_1 \cup V_2$. Досконале паросполучення з V_1 у V_2 існує тоді й лише тоді, коли для кожної множини $A \subset V_1$ виконується умова $|A| \leqslant |\Gamma(A)|$ (означення множини $\Gamma(A)$ наведено в підрозділі 3.3).

Доведення. Необхідність. Нехай існує досконале паросполучення з V_1 у V_2. Тоді множина $\Gamma(A)$ містить $|A|$ вершин із V_2, які відповідають вершинам з A у цьому паросполученні, і, можливо, ще якісь вершини з V_2. Отже, $|A| \leqslant |\Gamma(A)|$.

Достатність. Застосуємо індукцію за кількістю вершин у множині V_1. Нехай $|V_1| = m > 0$. У разі $m = 1$ єдина вершина з V_1 інцидентна принаймні одному ребру. Це ребро і являє собою потрібне паросполучення. Нехай $m > 1$ й теорема справджується для графів, у яких $|V_1| < m$. Окремо розглянемо два можливі випадки.

Випадок 1. Для кожної підмножини $A \subset V_1$, $A \neq V_1$ виконується строга нерівність $|A| < |\Gamma_G(A)|$. Тут і далі в доведенні індекс біля Γ показує, якого графа стосується позначення. Виберемо в графі G довільне ребро $\{x, y\}$. Розглянемо

граф $\tilde{G}$, одержаний із графа G вилученням вершин x та y і ребер, які інцидентні цим вершинам:

$$\tilde{V}_1 = V_1 \setminus \{x\}, \quad \tilde{V}_2 = V_2 \setminus \{y\},$$
$$\tilde{E} = E \setminus \{\{x, u\}, \{v, y\} \mid u \in V_2, v \in V_1\}.$$

Отриманий граф $\tilde{G} = (\tilde{V}_1 \cup \tilde{V}_2, \tilde{E})$ дводольний, причому $|\tilde{V}_1| = m - 1$. Нехай $\tilde{A}$ – довільна підмножина множини $\tilde{V}_1$. Оскільки $|\tilde{A}| < |\Gamma_G(\tilde{A})|$, а з множини V_2 вилучено лише одну вершину, то $|\tilde{A}| \leqslant |\Gamma_{\tilde{G}}(\tilde{A})|$. За індуктивним припущенням у графі $\tilde{G}$ існує паросполучення M, яке покриває $\tilde{V}_1$. Додамо до паросполучення M ребро $\{x, y\}$ і одержимо потрібне паросполучення в графі G.

Випадок 2. У множині V_1 існує така підмножина $A_0 \subset V_1$, $A_0 \neq V_1$, що

$$|A_0| = |\Gamma_G(A_0)|. \tag{3.2}$$

Позначимо $B = A_0 \cup \Gamma_G(A_0)$, і нехай H_1 та H_2 – підграфи графа G, породжені відповідно множинами вершин B та $V \setminus B$.

Розглянемо спочатку підграф H_1. Для будь-якої множини $A \subset A_0$ маємо $\Gamma_G(A) = \Gamma_{H_1}(A)$; отже, $|A| \leqslant |\Gamma_{H_1}(A)|$. Тому за індуктивним припущенням в H_1 існує паросполучення, яке покриває множину A_0.

Тепер розглянемо підграф H_2. Для будь-якої підмножини $A \subset V_1 \setminus A_0$

$$|A_0| + |A| = |A_0 \cup A| \leqslant |\Gamma_G(A_0 \cup A)| = |\Gamma_G(A_0)| + |\Gamma_{H_2}(A)|.$$

З урахуванням рівності (3.2) одержимо, що $|A| \leqslant |\Gamma_{H_2}(A)|$, і за припущенням індукції в графі H_2 існує паросполучення, яке покриває множину $V_1 \setminus A_0$. Об'єднаємо його з паросполученням, яке покриває множину A_0, й отримаємо паросполучення, що покриває всю множину V_1.

3.14. Найбільше паросполучення у дводольних графах

Задача побудови найбільшого паросполучення в графі (див. підрозділ 3.13) широко розповсюджена, і є ефективні алгоритми її розв'язування. Ці алгоритми ґрунтуються на методі почергових шляхів, ідея якого належить Дж. Петерсену [15].

Нехай M – паросполучення в графі G. Простий шлях, ребра якого почергово входять і не входять в M, називають *почерговим шляхом* відносно паросполучення M. Шлях довжиною 1 за означенням також почерговий. Ребра шляху називають *темними* (*світлими*), якщо вони належать (не належать) паросполученню M. Вершини графа G, інцидентні ребрам із паросполучення M, називають *насиченими*, усі інші вершини – *ненасиченими*.

Приклад 3.44. Розглянемо, наприклад, граф, зображений на рис. 3.53. Множина ребер $M = \{e_1, e_7, e_{10}\}$ являє собою паросполучення; $L = 7, 8, 4, 1, 2, 5$ – почерговий щодо M шлях; $e_1 = \{1, 2\}$, $e_{10} = \{4, 8\}$ – темні ребра шляху L; $e_3 = \{1, 4\}$, $e_5 = \{2, 5\}$, $e_{13} = \{7, 8\}$ – світлі; $\{1, 2, 3, 4, 6, 8\}$ і $\{5, 7, 9, 10\}$ – множини відповідно насичених і ненасичених вершин.

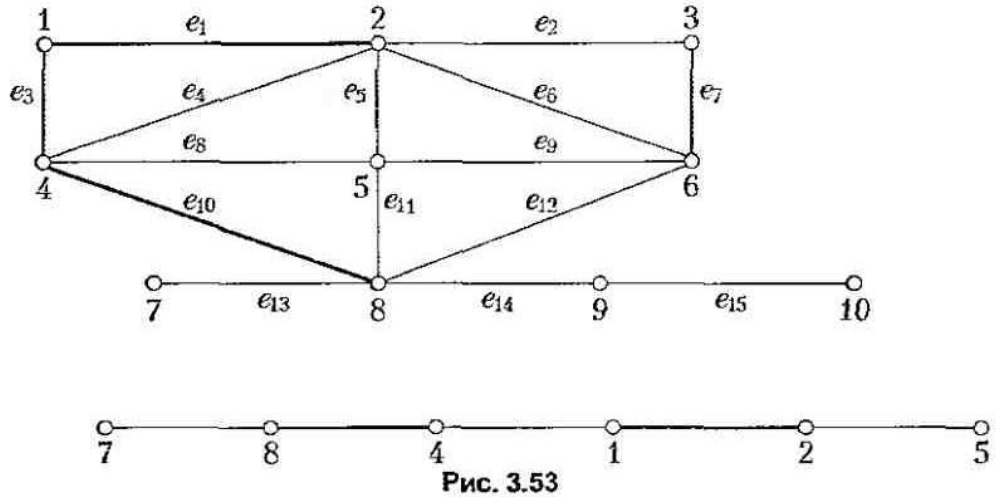


Рис. 3.53

Очевидно, що коли в графі G існує почерговий відносно паросполучення M шлях, який з'єднує дві різні ненасичені вершини, то в цьому графі можна побудувати паросполучення з більшою кількістю ребер, ніж у паросполученні M. Справді, у такому разі кількість темних ребер на одиницю менша, ніж кількість світлих. Вилучивши з M усі темні ребра та приєднавши всі світлі, отримаємо нове паросполучення, у якому на одне ребро більше. Тому почерговий щодо паросполучення M шлях, який з'єднує дві різні ненасичені вершини, називають *збільшувальним* стосовно M шляхом у графі G. Отже, відсутність збільшувальних відносно M шляхів – необхідна умова того, що паросполучення найбільше. Цяж умова виявляється й достатньою.

ТЕОРЕМА 3.23. Паросполучення M у графі найбільше тоді й лише тоді, коли в цьому графі немає збільшувальних щодо M шляхів.

Доведення. Необхідність, як уже було зазначено, очевидна. Достатність доведемо від протилежного. Нехай M_1 – також паросполучення в графі G, $|M_1| > |M|$. Розглянемо граф H, утворений ребрами $(M \cup M_1) \setminus (M \cap M_1)$. Оскільки довільна його вершина v інцидентна не більше ніж одному ребру кожного паросполучення M і M_1, то її степінь не більший ніж 2. Якщо $\deg(v) = 2$, то одне з інцидентних вершині v ребер належить паросполученню M, друге – M_1. Тому будь-яка зв'язна компонента графа H являє собою або цикл, що містить однакову кількість ребер із паросполучень M і M_1, або почерговий відносно M шлях. Але $|M_1| > |M|$, тому серед цих компонент обов'язково є почерговий щодо паросполучення M шлях, крайні ребра якого (перше й останнє) належать M_1. Тоді крайні вершини цього шляху не насичені паросполученням M, що суперечить умові теореми.

Приклад 3.45. Знову розглянемо граф, зображений на рис. 3.53. Почерговий шлях з'єднує ненасичені вершини 5 і 7. Отже, можна побудувати паросполучення M' із більшою кількістю ребер: $M' = (M \setminus \{e_1, e_{10}\}) \cup \{e_3, e_5, e_{13}\} = \{e_3, e_5, e_7, e_{13}\}$. Паросполучення M' також не

найбільше: 9, 10 – збільшувальний щодо M' шлях. Паросполучення $M'' = M' \cup \{e_{15}\} = \{e_3, e_5, e_7, e_{13}, e_{15}\}$ найбільше в графі.

Отже, теорема 3.23 підказує таку стратегію пошуку найбільшого паросполучення: почати з довільного паросполучення M_1 і будувати послідовність M_1, M_2, M_3, ..., у якій паросполучення M_{k+1} отримано з M_k за допомогою щойно розглянутої зміни вздовж якогось збільшувального шляху. Оскільки $|M_{k+1}| = |M_k| + 1$, то для одержання найбільшого паросполучення потрібно не більше ніж $|V|/2$ ітерацій (тобто переходів від M_k до M_{k+1}). Початкове паросполучення M_1 завжди є в нашому розпорядженні: можна взяти довільне ребро графа чи, краще, якесь максимальне паросполучення. Тому розробка ефективного алгоритму, що ґрунтується на зазначеній стратегії, зводиться до побудови процедури, яка швидко знаходить збільшувальний шлях у графі чи виявляє, що його немає. Обмежимося дводольними графами [2], хоча така процедура відома й для довільних графів [25].

Отже, нехай $G = (V, E)$ – дводольний граф і множину його вершин V розбито на дві підмножини V_1 і V_2, $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \varnothing$ (рис. 3.54).

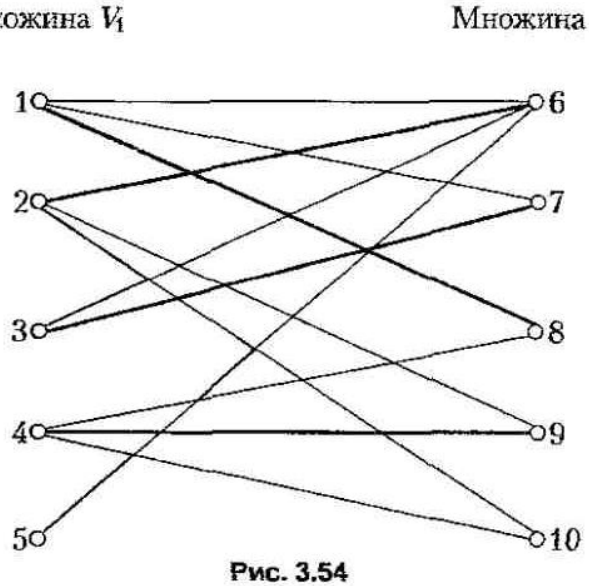


Рис. 3.54

Побудуємо граф збільшувального шляху за рівнями $i = 0, 1, 2, ...$, використовуючи процес, подібний до пошуку вшир. Граф збільшувального шляху рівня 0 містить усі ненасичені вершини з множини V_1. На рівні з непарним номером i додають нові вершини, суміжні з вершинами рівня $i - 1$ і з'єднані світлим ребром (яке не належить паросполученню). Це ребро також додають до будованого графа. На рівні з парним номером i також додають нові вершини, суміжні з вершинами рівня $i - 1$, але з'єднані темним ребром (яке належить паросполученню). Це ребро також додають до графа збільшувального шляху.

Процес побудови продовжують доти, доки до графа почергового шляху можна приєднувати нові вершини. Зазначимо, що ненасичену вершину можна приєднати до цього графа тільки на непарному рівні. У побудованому графі шлях від будь-

якої ненасиченої вершини (яка може бути тільки на непарному рівні) до будь-якої вершини рівня 0 — збільшувальний шлях відносно паросполучення M.

На рис. 3.55 зображено граф збільшувального шляху для дводольного графа, який зображено на рис. 3.54, щодо паросполучення, показаного потовщеними лініями (темні ребра).

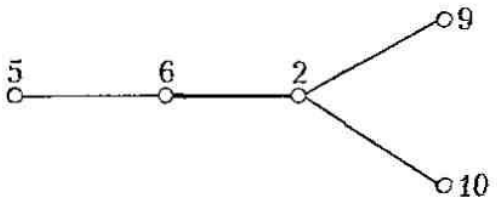


Рис. 3.55

На рівні 0 маємо одну ненасичену вершину 5. На рівні 1 додано світле ребро {5, 6} (яке не належить паросполученню з рис. 3.54). На рівні 2 додано темне ребро {6, 2} (яке належить паросполученню). На рівні 3 додано світлі ребра {2, 9} і {2, 10}, які не належать паросполученню. Оскільки вершина 10 у графі, зображеному на рис. 3.54, ненасичена, то можна зупинити процес побудови графа збільшувального шляху. Шлях 10, 2, 6, 5 збільшувальний відносно паросполучення, показаного на рис. 3.54. Із паросполучення рис. 3.54 вилучимо всі темні ребра, що належать цьому шляху, і додамо всі світлі. Отримаємо нове паросполучення, яке містить на одне ребро більше (рис. 3.56).

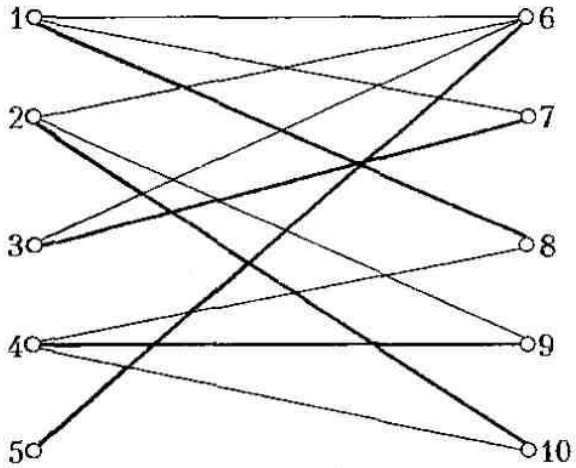


Рис. 3.56

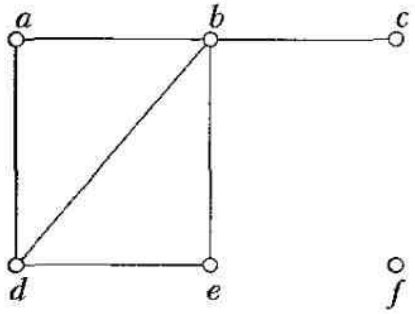
Для паросполучення, зображеного на рис. 3.56, очевидно, не існує збільшувальних шляхів. Отже, це паросполучення найбільше.

Нехай граф $G = (V, E)$ задано списками суміжності. Тоді на побудову графа збільшувального шляху потрібно часу порядку $O(|E|)$. Для знаходження найбільшого паросполучення, як уже було зазначено, потрібно побудувати не більше ніж $|V|/2$ збільшувальних шляхів. Тому найбільше паросполучення дводольного графа можна знайти за час порядку $O(|V| \cdot |E|)$.

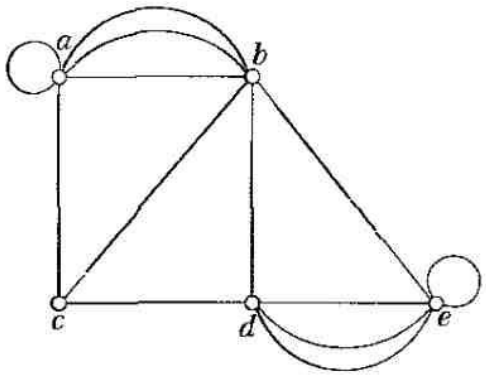
Контрольні запитання та завдання

1. Знайти кількість вершин, ребер і степені кожної вершини несорієнтованих графів:

а)



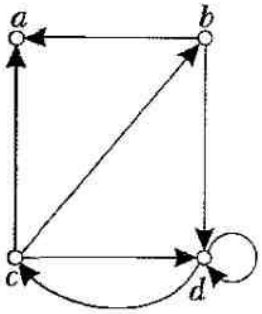
б)



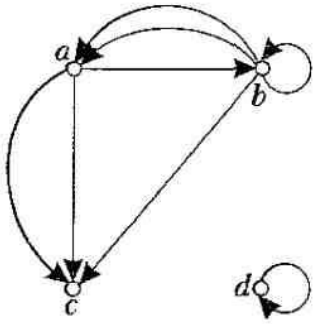
2. Знайти суму степенів вершин кожного з графів задачі 1 та переконатись, що вона вдвічі більша за кількість ребер графа.

3. Визначити кількість вершин та дуг і знайти напівстепені входу й виходу для кожної вершини орієнтованих мультиграфів:

а)



б)



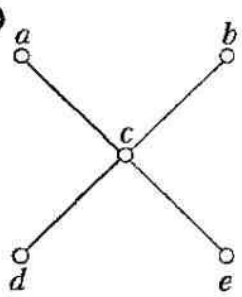
4. Для кожного з графів задачі 3 знайти суму напівстепенів входу та суму напівстепенів виходу вершин. Переконатись, що кожна з них дорівнює кількості дуг графа.

5. Побудувати графи:

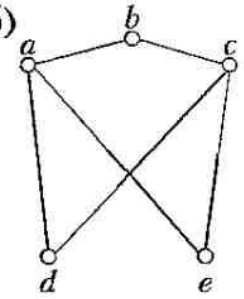
а) K_7; б) $K_{1,8}$; в) $K_{1,4}$; г) C_7; д) W_7; е) Q_4.

6. Які з наведених нижче графів дводольні?

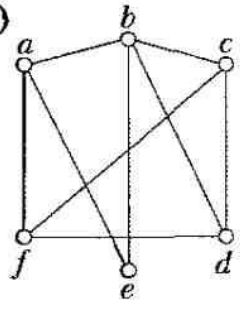
а)



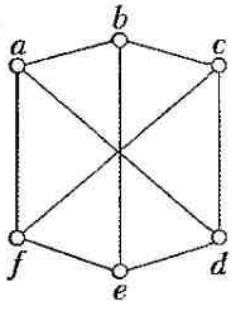
б)



в)



г)



7. Для яких значень n наведені нижче графи дводольні:

а) K_n; б) C_n; в) W_n; г) Q_n?

8. Скільки вершин і ребер мають наведені нижче графи:

   а) K_n; б) C_n; в) W_n; г) Q_n; д) $K_{m,n}$?

9. Скільки ребер має граф, у якого вершини мають такі степені: 4, 3, 3, 2, 2? Зобразити його.

10. Чи існує простий граф із вершинами таких степенів? Якщо так, то зобразити його:

    а) 3, 3, 3, 3, 2; б) 3, 4, 3, 4, 3; в) 1, 2, 3, 4, 5;

    г) 1, 2, 3, 4, 4; д) 0, 1, 2, 2, 3; е) 1, 1, 1, 1, 1.

11. Нехай G — граф з n вершинами та m ребрами. Нехай $d_{\max}$ — максимальний степінь вершини цього графа, а $d_{\min}$ — мінімальний. Довести, що $d_{\min} \leqslant 2m/n \leqslant d_{\max}$.

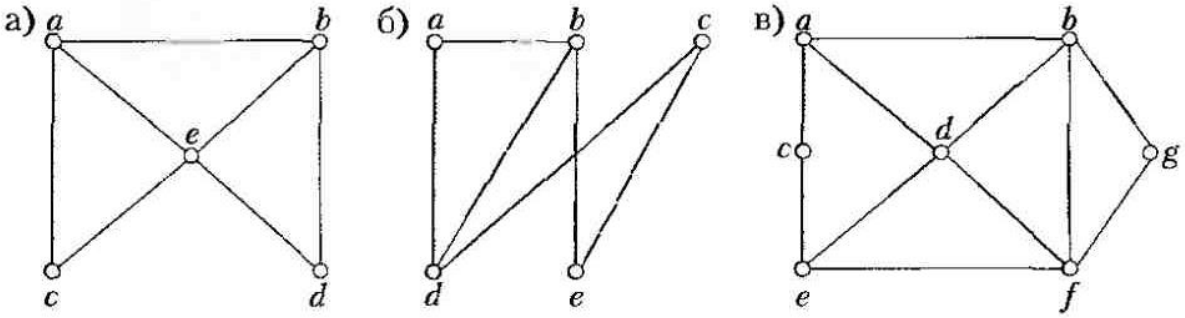
12. Простий граф називають *регулярним*, якщо всі його вершини мають однаковий степінь. Граф називають ρ-регулярним, якщо кожна його вершина має степінь ρ. Для якого n наведені нижче графи регулярні:

    а) K_n; б) C_n; в) W_n; г) Q_n

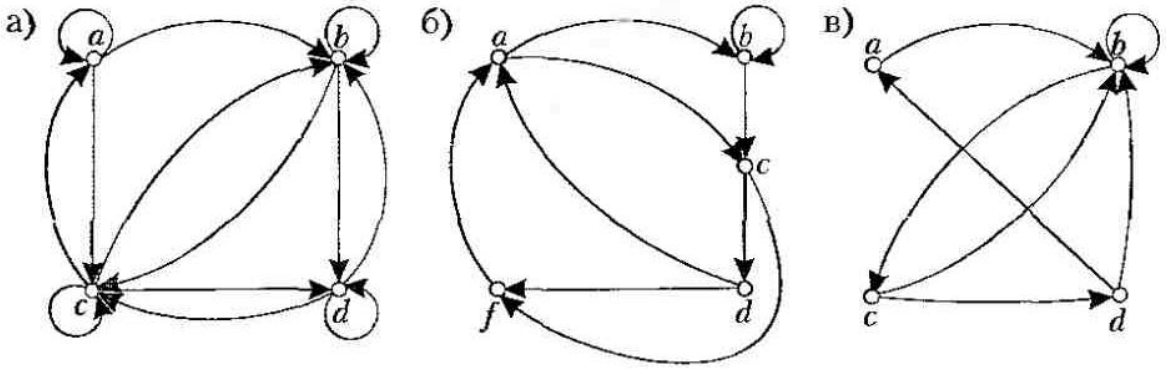
13. Для яких m і n граф $K_{m,n}$ регулярний?

14. Скільки вершин має регулярний граф степеня 4 з 10 ребрами?

15. Задати прості графи за допомогою матриць інцидентності.



16. Задати орієнтовані графи за допомогою матриць інцидентності.



17. Задати графи із задачі 15 за допомогою матриць суміжності.

18. Задати графи із задачі 16 за допомогою матриць суміжності.

19. Задати графи задачі 15 списками пар (списками ребер), записаних у лексикографічному порядку.

20. Задати графи задачі 16 списками пар (списками дуг), записаних у лексикографічному порядку.

21. Задати графи задачі 15 списками суміжності.

22. Задати графи задачі 16 списками суміжності.

23. Зобразити неорієнтовані графи за матрицями суміжності:

а) $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$; б) $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$; в) $\begin{pmatrix} 1 & 2 & 0 & 1 \\ 2 & 0 & 3 & 0 \\ 0 & 3 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$.

24. Чи є будь-яка квадратна симетрична (0,1)-матриця, що містить 0 на головній діагоналі, матрицею суміжності якогось простого графа?

25. Зобразити орієнтовані графи за матрицями суміжності:

а) $\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$; б) $\begin{pmatrix} 0 & 2 & 3 & 0 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 1 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}$.

26. Зобразити неорієнтований граф, заданий множиною вершин і багатозначним відображенням Γ.

v	v_1	v_2	v_3	v_4	v_5
$\Gamma(v)$	v_2, v_3, v_5	v_1	v_1, v_4, v_5	v_3, v_5	v_1, v_3, v_4

27. Зобразити орієнтований граф, заданий множиною вершин і багатозначним відображенням Γ.

v	v_1	v_2	v_3	v_4	v_5
$\Gamma(v)$	v_2, v_3, v_4, v_5	v_2, v_4	v_1, v_3, v_5	–	v_2, v_3, v_4

28. Знайти матриці суміжності для графів:
а) K_n; б) C_n; в) W_n; г) $K_{m,n}$; д) Q_n.

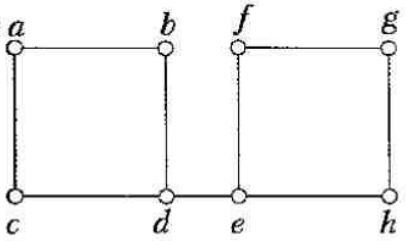
29. Нехай G – простий граф. Нагадаємо, що *доповнювальним* називають такий граф $\overline{G}$, у якого та сама множина вершин і дві вершини в $\overline{G}$ з'єднано ребром тоді й лише тоді, коли їх не з'єднано ребром у графі G. Знайти:
а) $\overline{K}_n$; б) $\overline{K}_{m,n}$; в) $\overline{C}_n$.

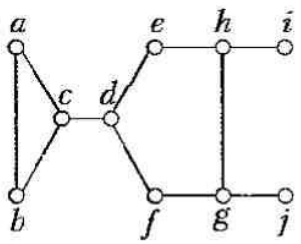
30. Нехай G – простий граф з n вершинами та m ребрами. Довести, що об'єднання G та $\overline{G}$ утворює граф K_n.

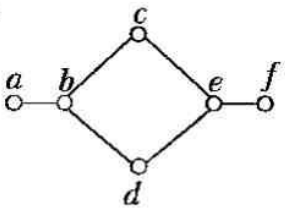
31. Простий граф G має 15 ребер, а граф $\overline{G}$ – 13. Знайти кількість вершин графа G.

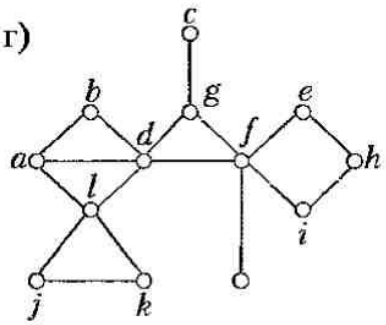
32. Простий граф G має n вершин і m ребер. Знайти кількість ребер графа $\overline{G}$.

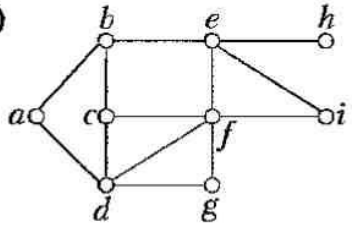
33. Обійти наведені нижче графи пошуком углиб. Уважати, що вершини впорядковано за алфавітом, а початкова — вершина a.

а)


б)


в)


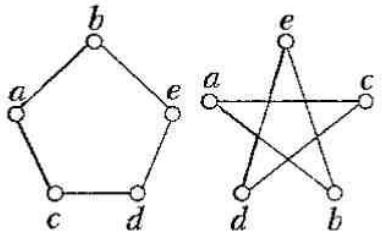
г)


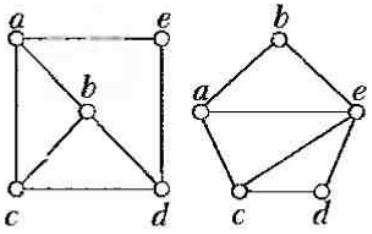
д)


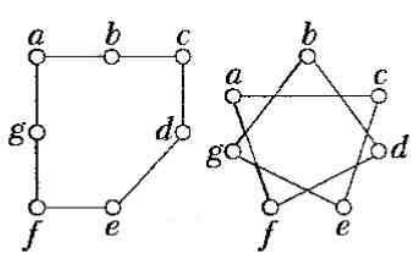
34. Обійти графи із задачі 33 пошуком ушир. Уважати, що вершини впорядковано за алфавітом, а початкова — вершина a.

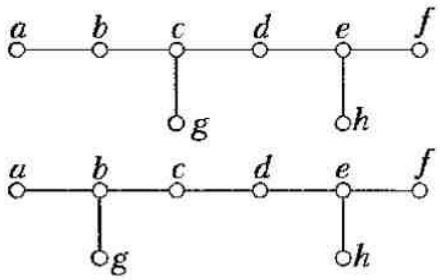
35. Розв'язати задачі 33 і 34 за умови, що початкова вершина обходу графа — d.

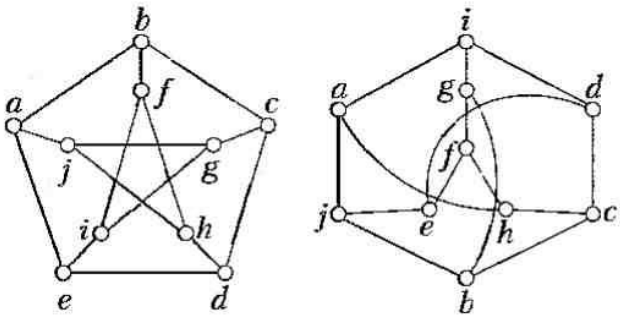
36. Визначити, які пари графів, наведених нижче, ізоморфні.

а)


б)


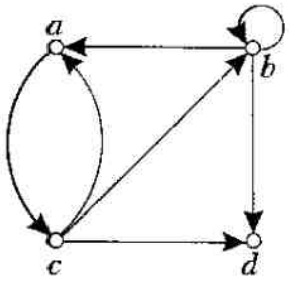
в)


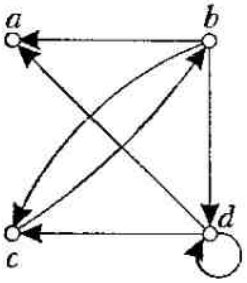
г)


д)


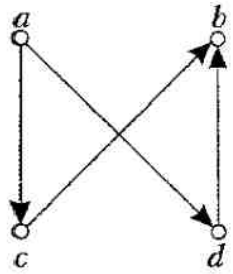
37. Знайти кількість неізоморфних простих графів з n вершинами, якщо n дорівнює:
    а) 2; б) 3; в) 4.
38. Знайти кількість неізоморфних простих графів з п'ятьма вершинами та трьома ребрами.
39. Визначити, які пари орієнтованих графів ізоморфні.

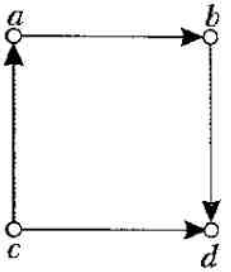
а)



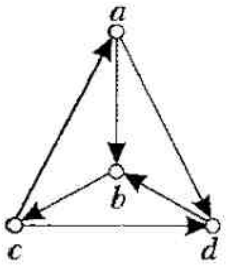


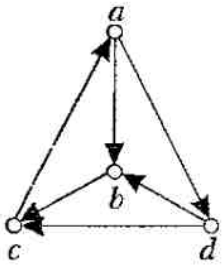
б)





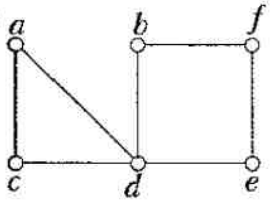
в)



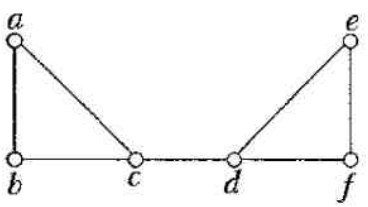


40. Знайти точки з'єднання графів.

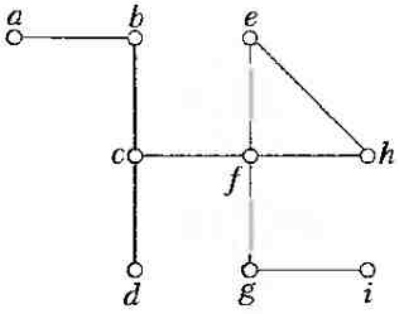
а)



б)



в)



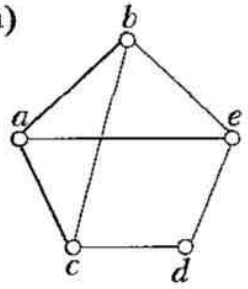
41. Знайти всі мости в графах задачі 40.

42. Знайти кількість шляхів довжиною n між двома різними вершинами графа K_4, якщо n дорівнює 2; 3; 4; 5.

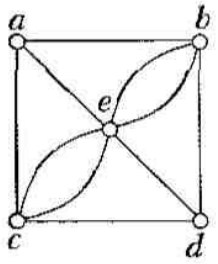
43. Знайти кількість шляхів довжиною n між двома несуміжними вершинами графа $K_{3,3}$, якщо n дорівнює 2, 3, 4, 5. Визначити те саме для суміжних вершин.

44. Визначити, які з наведених нижче графів мають ейлерів цикл. Зобразити його.

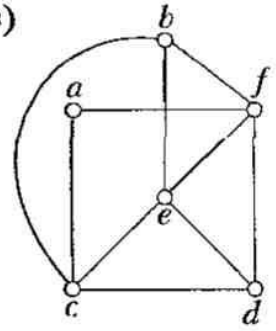
а)



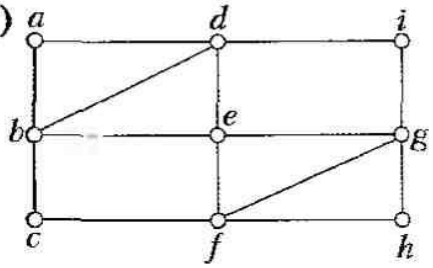
б)



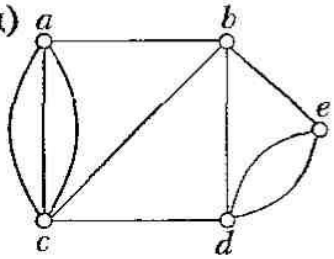
в)



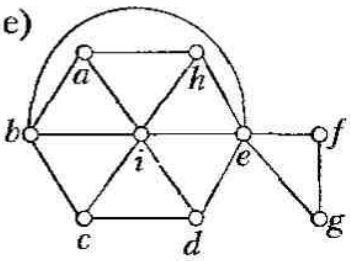
г)



д)



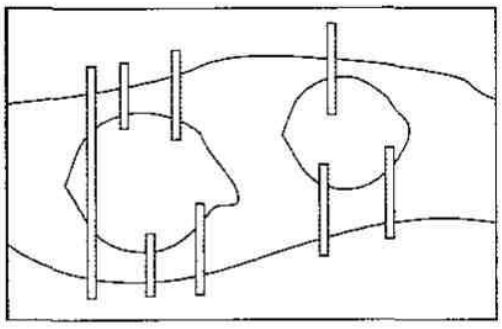
е)



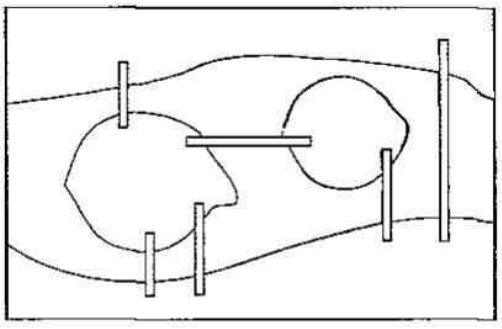
45. Який із графів задачі 44 має ейлерів шлях, але не має ейлеревого циклу?

46. Чи можна утворити ейлерів цикл у разі розміщення мостів?

а)



б)



47. Для яких значень n наведені нижче графи мають ейлерів цикл:

а) C_n; б) K_n; в) W_n; г) Q_n?

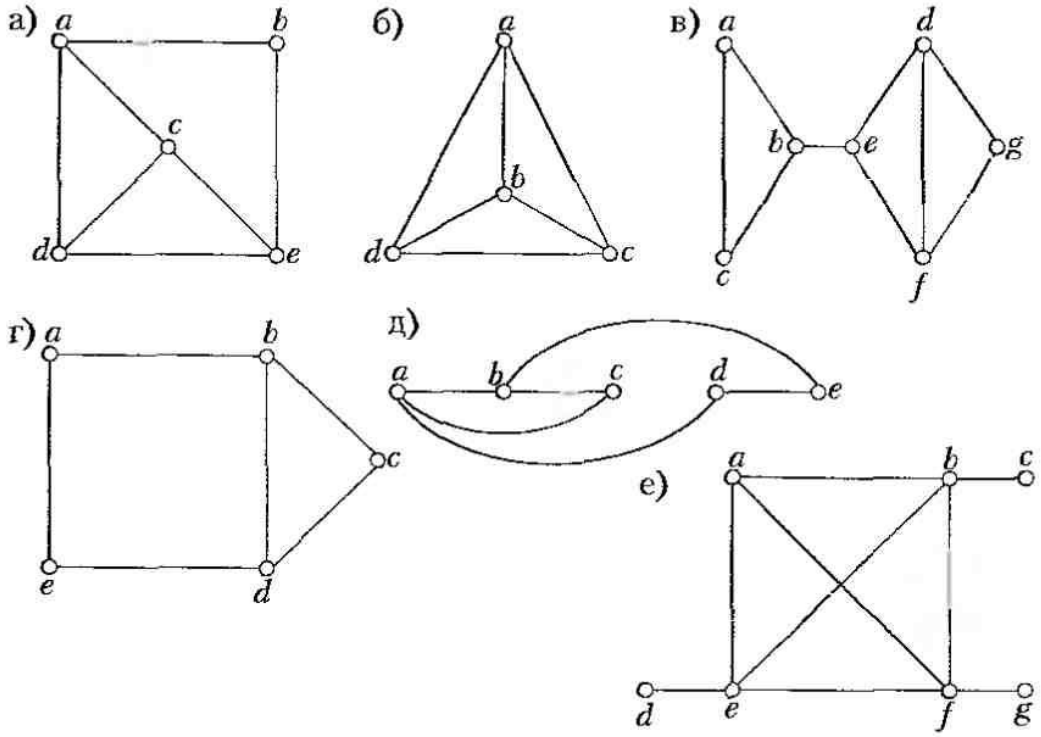
48. Для яких значень n наведені нижче графи мають ейлерів шлях, але не мають ейлерового циклу:

а) C_n; б) K_n; в) W_n; г) Q_n?

49. Визначити довжину найкоротшого циклу в наведених нижче графах, який проходить через кожне ребро принаймні один раз:

а) K_6; б) K_n; в) $K_{2,5}$; г) $K_{m,n}$.

50. Які з наведених нижче графів мають гамільтонів цикл?



51. Які з графів задачі 50, що не мають гамільтонового циклу, мають гамільтонів шлях?

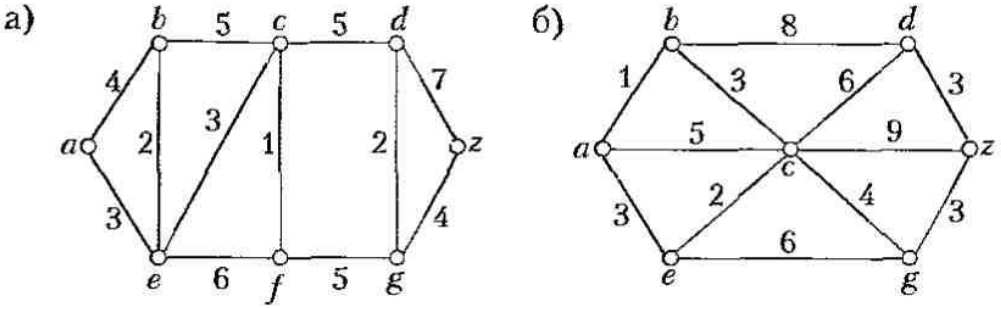
52. Для яких значень n наведені нижче графи мають гамільтонів цикл:
а) C_n; б) K_n; в) W_n; г) Q_n?

53. Для яких значень m і n граф $K_{m,n}$ має гамільтонів цикл?

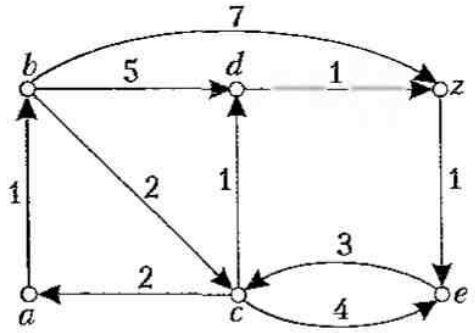
54. Навести приклад графа, який має ейлерів цикл, але не має гамільтонового циклу, а також графа, який має гамільтонів цикл, але не має ейлерового циклу. Як можна охарактеризувати графи, які мають водночас і ейлерів, і гамільтонів цикли?

55. Навести контрприклад, який показує, що у формулюванні теореми Дірака умову $\deg(v) \geqslant \frac{n}{2}$ не можна замінити слабшою умовою $\deg(v) \geqslant \frac{n}{2} - 1$.

56. За допомогою алгоритму Дейкстри знайти найкоротший шлях від a до z у неорієнтованих графах, зображених на рисунках.



57. За допомогою алгоритму Дейкстри знайти найкоротший шлях від a до z в орієнтованому графі, зображеному на рисунку.



58. Довести (навести приклад), що алгоритм Дейкстри може й не визначити найкоротший шлях, якщо в графі є дуги з від'ємними довжинами.

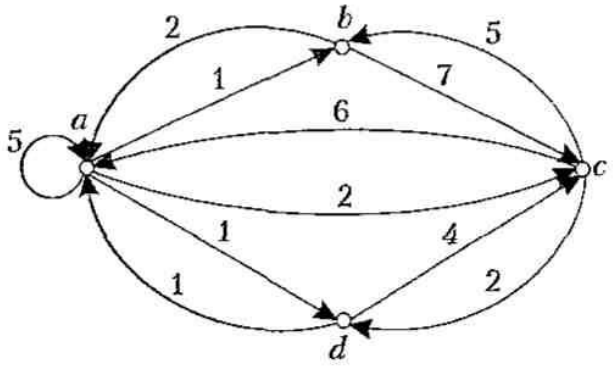
59. Нехай алгоритм Дейкстри застосовано до графа з вершинами $v_1, v_2, ..., v_7$, унаслідок чого одержано вектор довжин (постійних міток) $l = (0, 6, 5, 8, 12, 13, 14)$ і вектор вершин $\theta = (-, 1, 3, 4, 4, 6)$, у якому вершину v_i позначено як i. Виконати такі завдання:

    а) визначити найкоротший шлях від v_1 до v_7;

    б) визначити найкоротший шлях від v_1 до v_5;

    в) зобразити цей граф, якщо це можливо.

60. За допомогою алгоритму Флойда визначити довжини найкоротших шляхів між усіма парами вершин орієнтованого графа. У процесі розв'язування будувати матриці W та Θ. За матрицею Θ визначити найкоротші шляхи від вершини a до вершини d та від вершини b до вершини d.



61. Зв'язний планарний граф має шість вершин степеня 4. Скільки граней має цей граф?

62. Зв'язний планарний граф регулярний і містить 30 ребер. Кількість його граней дорівнює 20. Скільки вершин має цей граф?

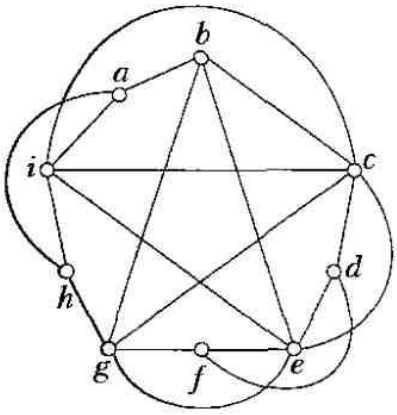
63. Нехай зв'язний простий планарний граф містить $n \geqslant 3$ вершин і m ребер. Довести, що $m \leqslant 3n - 6$.

64. Нехай зв'язний дводольний планарний граф містить $n \geqslant 3$ вершин і m ребер. Довести, що $m \leqslant 2n - 4$.

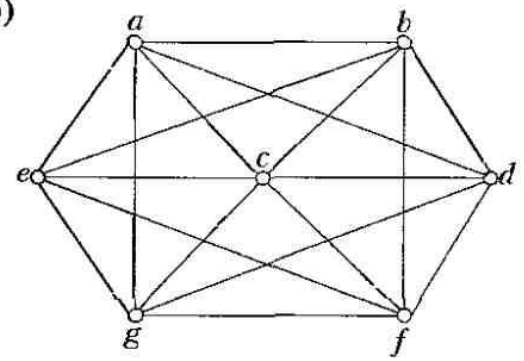
65. Довести, що в будь-якому простому планарному графі є вершина, степінь якої не більший ніж 5.

66. За допомогою теореми Куратовського довести, що графи, наведені нижче, непланарні.

а)

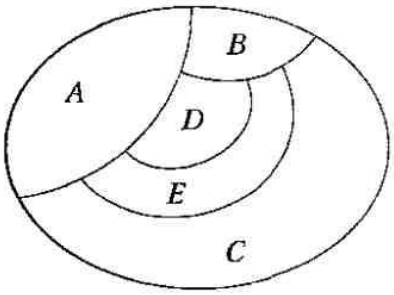


б)

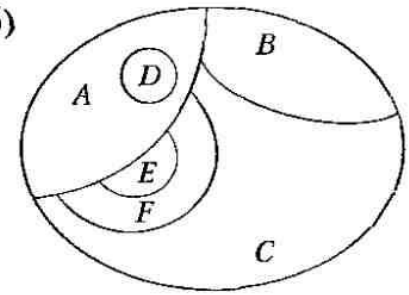


67. Побудувати графи, які відповідають наведеним нижче картам, і визначити найменшу кількість кольорів для їх розфарбовування.

а)



б)



68. Знайти хроматичні числа графів:

а) C_n; б) W_n; в) Q_n.

69. Довести, що простий граф, який містить цикл із непарною кількістю вершин, не можна розфарбувати у два кольори.

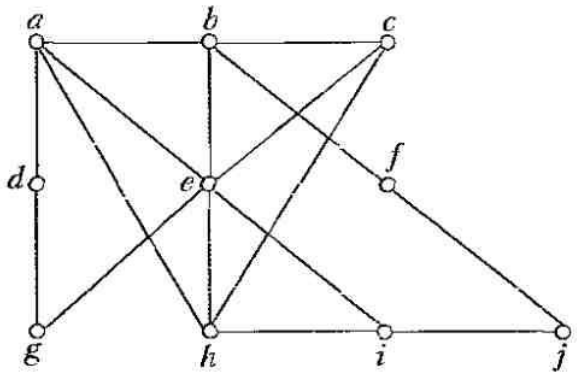
70. Для розфарбовування простого графа G можна застосувати такий алгоритм. Перенумерувати вершини $v_1, v_2, ..., v_n$ за спаданням степенів:

$$\deg(v_1) \geqslant \deg(v_2) \geqslant ... \geqslant \deg(v_n).$$

Зафарбувати кольором 1 вершину v_1 і послідовно кожну вершину зі списку, не суміжну з вершиною кольору 1. Зафарбувати кольором 2 першу серед не зафарбованих іще вершин зі списку та послідовно зафарбувати кольором 2 вершини зі списку, які ще не зафарбовані й не суміжні з вершинами кольору 2. Далі вибрати колір 3 та продовжити процес.

За допомогою цього алгоритму розфарбувати граф, зображений на рисунку. Довести (навести приклад), що цей алгоритм може й не дати оптимального

розфарбування графа G, тобто розфарбування в найменшу можливу кількість кольорів $\chi(G)$.



71. Скільки каналів щонайменше потрібно для шести телевізійних станцій, відстані між якими задано таблицею, якщо дві станції не можуть працювати на одному каналі, коли відстані між ними менша ніж 250 км?

Телевізійна станція	1	2	3	4	5	6
1	–	145	280	300	90	160
2	145	–	200	280	150	270
3	280	200	–	220	350	400
4	300	280	220	–	310	360
5	90	150	350	310	–	120
6	160	270	400	360	120	–

72. На кафедрі працюють шість студентських наукових семінарів один раз на місяць в один і той самий час. Скільки різних днів щонайменше потрібно для проведення цих семінарів, якщо склад їх керівників такий:

1) C1 = {Пасічник, Щербина, Нікольський};

2) C2 = {Щербина, Годич, Катренко};

3) C3 = {Пасічник, Катренко, Нікольський};

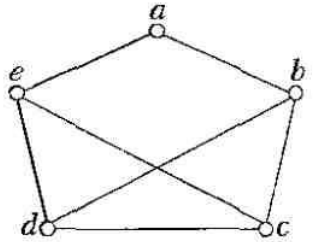
4) C4 = {Годич, Нікольський, Катренко};

5) C5 = {Пасічник. Щербина}:

6) C6 = {Щербина, Катренко, Нікольський}.

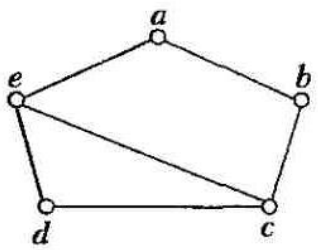
Умову задачі зобразити у вигляді графа.

73. Для графа, зображеного на рисунку, побудувати хроматичний поліном.

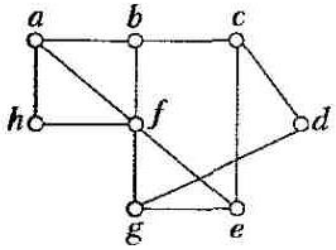


74. Для наведених нижче графів знайти всі максимальні незалежні множини вершин. Зазначити найбільшу незалежну множину та число незалежності кожного з цих графів.

а)



б)



75. Для графів задачі 74 знайти всі максимальні кліки. Визначити клікове число (щільність) цих графів.

76. Будівельній фірмі для виконання певної роботи потрібні муляр, столяр, слюсар і сантехнік. На ці місця є п'ять претендентів: один може працювати муляром, другий — столяром, третій — муляром і сантехніком, ще двоє мають по дві спеціальності — сантехніка та слюсаря. Чи можуть виконати всі роботи четверо з цих робітників? Для обґрунтування відповіді докладно перевірити виконання умов теореми Холла.

Комп'ютерні проекти

Скласти програми із зазначеними вхідними даними та результатами.

1. Задано множину пар вершин, що відповідають ребрам неорієнтованого графа. Знайти степінь кожної вершини.
2. Задано множину впорядкованих пар вершин, що відповідають дугам орієнтованого графа. Знайти напівстепені входу та виходу кожної вершини.
3. Задано множину пар вершин, що відповідають ребрам простого графа. Визначити, чи дводольний цей граф.
4. Задано множину пар вершин, що відповідають ребрам графа. Побудувати матрицю суміжності графа з урахуванням того, що граф може мати петлі та кратні ребра, а також бути орієнтованим.
5. Задано матрицю суміжності неорієнтованого графа. Побудувати множину пар вершин, що відповідають їй, і знайти кратність кожного ребра.
6. Задано множину пар вершин, що відповідають ребрам неорієнтованого графа, а також кратність кожного ребра. Побудувати матрицю інцидентності цього графа.
7. Задано матрицю інцидентності неорієнтованого графа. Побудувати множину пар вершин, що відповідають його ребрам, і визначити кратність кожного ребра.
8. Задано множини пар вершин, що відповідають ребрам двох простих графів не більше ніж із шістьма вершинами. Визначити, чи ізоморфні ці графи.
9. Задано матрицю суміжності графа та натуральне число r. Знайти кількість шляхів довжиною r між двома вершинами (граф може бути як орієнтованим, так і неорієнтованим).

10. Задано множину пар вершин, що відповідають ребрам простого графа. Визначити, чи зв'язний граф. Якщо виявиться, що граф незв'язний, знайти кількість його компонент.
11. Задано матрицю суміжності мультиграфа. Зазначити, чи має він ейлерів цикл, а якщо ні, то ейлерів шлях. Побудувати ейлерів цикл або шлях, якщо вони існують.
12. Задано множину пар вершин, що відповідають ребрам мультиграфа, і кратність кожного ребра. Визначити, чи має цей мультиграф ейлерів цикл, а якщо ні, то ейлерів шлях. Побудувати ейлерів цикл або шлях, якщо вони існують.
13. Задано матрицю суміжності орієнтованого мультиграфа. Визначити, чи має він ейлерів цикл, а якщо ні, то ейлерів шлях. Побудувати ейлерів цикл або шлях, якщо вони існують.
14. Задано множину впорядкованих пар вершин, що відповідають дугам орієнтованого мультиграфа, і кратність кожної дуги. Визначити, чи має він ейлерів цикл, а якщо ні, то ейлерів шлях. Побудувати ейлерів цикл або шлях, якщо вони існують.
15. Задано матрицю суміжності простого графа. Побудувати цикл із найменшою довжиною, який проходить через кожне ребро графа принаймні один раз.
16. Задано зважений неорієнтований граф із двома вершинами непарного степеня. Знайти цикл із найменшою довжиною, який проходить через кожне ребро графа принаймні один раз.
17. Задано множину пар вершин, що відповідають ребрам зваженого зв'язного простого графа, і дві його вершини. За допомогою алгоритму Дейкстри знайти довжину найкоротшого шляху між зазначеними вершинами та побудувати цей шлях.
18. Задано віддалі між парами телевізійних станцій. Призначити розподіл каналів для них, тобто побудувати граф, у якому станції позначені вершинами. Дві вершини з'єднано ребром, якщо віддаль між ними не більша ніж 250 км. Призначенню каналів відповідає таке розфарбування вершин графа, у якому різні кольори відповідають різним каналам. Використати найменшу можливу кількість каналів.
19. Знайти всі точки з'єднання заданого простого графа.
20. Знайти всі мости заданого простого графа за допомогою алгоритму пошуку вглиб.
21. Задано дводольний граф. Знайти найбільше паросполучення.