7. НЕОРІЄНТОВАНІ ГРАФИ

Графи виникли у 18-му столітті, коли відомий швейцарський математик Леонард Ейлер пробував розв'язати тепер вже класичну задачі про Кенігбергські мости. В той час в місті Кенігсберзі було два острови, з'єднані 7-ма мостами з берегами річки Преголь і один з одним, як показано на рис. 7.1.

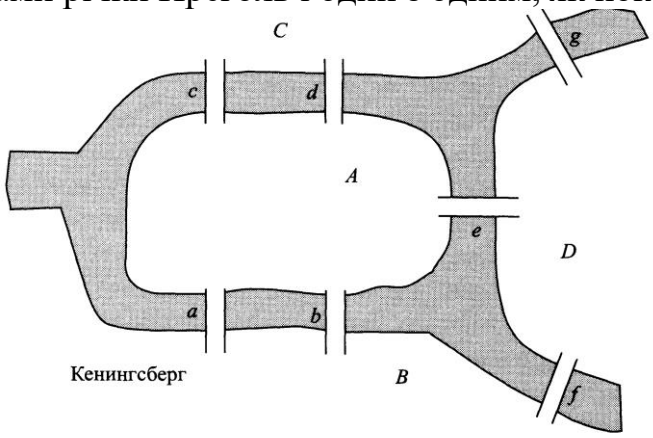


Рис. 7.1. Схема старого Кенігсберга

Задача формулювалась так: здійснити прогулянку містом так, щоб, пройшовши лише раз по одному мосту, повернутись в те місце, звідки починалась прогулянка. Розв'язуючи цю задачу, Ейлер зобразив Кенігсберг у вигляді графа, ототожнивши його вершини з частинами міста, а ребра з мостами, які зв'язували ці частини. Як ми покажемо нижче, йому вдалося довести, що шуканого маршруту в місті не існує.

У цій темі ми введемо стандартну термінологію, яку використовують в теорії графів, і розглянемо алгоритми обходу графів та декілька конкретних задач, які розв'язують за допомогою графів.

7.1. Термінологія

Перед розв'язуванням задачі, поставленої Ейлеру, розглянемо термінологію, яку застосовують в теорії графів.

Означення 1.1. Простий граф визначають як пару $G=(V, E)$, де V, E – скінченні множини вершин і ребер відповідно, причому G *не може* містити *петель* (ребер, які починаються і закінчуються в одній вершині) і *кратних ребер* (кратними називаються ребра, які з'єднують одну і ту саму пару вершин) (рис. 7.2, *а*).

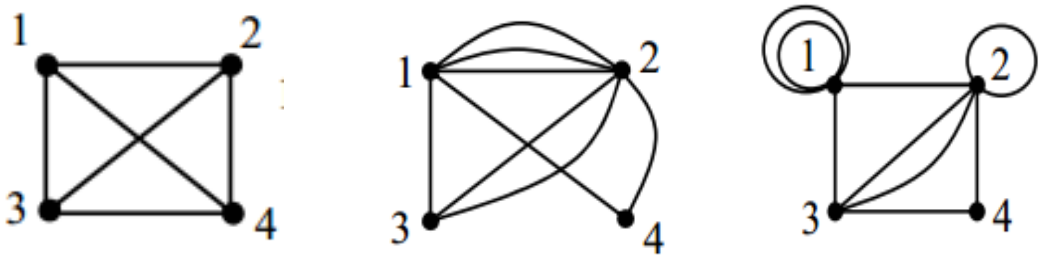
Означення 1.2. Сім'я – це невпорядкована сукупність елементів, у якій вони можуть повторюватись, зокрема, сім'я ребер означає, що в графі є кратні ребра.

Означення 1.3. Неорієнтований мультиграф – це пара $G=(V, E)$, де V – непорожня скінченна множина вершин, E – сім'я невпорядкованих пар *різних* елементів із V, тобто це граф, що містить *кратні ребра* (рис. 7.2, *б*).

Означення 1.4. Псевдограф – це пара $G=(V, E)$, де V – непорожня скінченна множина вершин, E – сім'я невпорядкованих пар *не обов'язково різних* елементів із V, тобто це граф, що містить *кратні петлі* та *кратні ребра* (рис. 7.2, *в*).

За допомогою графів зображують структурні залежності між елементами. Наприклад, в електричній схемі, у молекулі, складеній з атомів або у схемі

міського транспорту. Для побудови графа, що відповідає транспортній схемі, вершинами можна подати зупинки, а ребрами – ділянки маршрутів між ними. Схему доріг міста можна зобразити як мультиграф, вершини якого відповідають перехрестям, а ребра – вулицям з двостороннім рухом.



а) б) в)

Рис. 7.2. Приклади неорієнтованих графів:
а) простий граф, *б*) мультиграф, *в*) псевдограф

Означення 1.5. Підграфом графа $G=(V_G, E_G)$ називають граф $H=(V_H, E_H)$, в якому $V_H \subset V_G$ і $E_H \subset E_G$; позначають це так: $H \subset G$. Підграф H графа G називають **кістяковим (остовним)** підграфом, якщо $V_H = V_G$.

Означення 1.6. Граф $H=(A, E_H)$ називають **вершинно породженим підграфом** графа G (на множині вершин A), якщо $A \subset V_G$ і множина E_H складається з усіх ребер графа G, які з'єднують вершини з множини A. Граф $H=(V_H, B)$ називають **реберно породженим підграфом** графа G (на множині ребер B), якщо $B \subset E_G$ і множина V_H складається з усіх вершин, інцидентних ребрам з B і лише з них.

Очевидно, що реберно породжений граф не містить ізольованих точок.

У наступних означеннях простіше вважати, що графи $G_1=\{V_1, E_1\}$ та $G_2=\{V_2, E_2\}$ є підграфами деякого «більшого» графа.

Означення 1.7. Об'єднанням графів $G_1=\{V_1, E_1\}$ та $G_2=\{V_2, E_2\}$ називають граф $G = G_1 \cup G_2=\{V, E\}$, де $V = V_1 \cup V_2$, $E = E_1 \cup E_2$. **Перетином графів** $G_1=\{V_1, E_1\}$ та $G_2=\{V_2, E_2\}$ називають граф $G=G_1 \cap G_2=\{V, E\}$, де $V = V_1 \cap V_2$; $E=E_1 \cap E_2$.

На рис. 7.3-7.5 показано, як об'єднувати та як знаходити перетин графів:

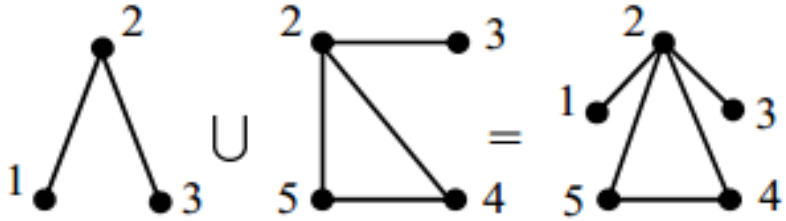


Рис. 7.3. Об'єднання графів

Якщо $V_1=V_2$ та $E_1 \subset E_2$, тоді $G = G_1 \cup G_2=G_2$ (рис. 7.4).

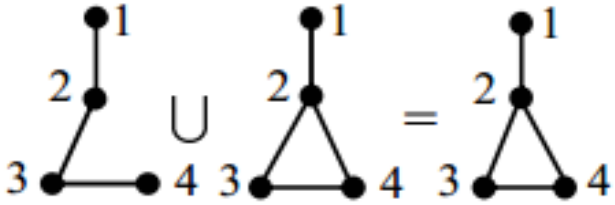


Рис. 7.4. Об’єднання графів зі спільними вершинами

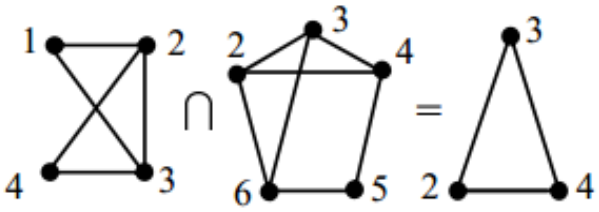


Рис. 7.5. Перетин графів

З означення 1.7 випливає: якщо $V_1 \cap V_2 = \varnothing$ (тобто два графи не мають однаково позначених вершин), то перетин $G = G_1 \cap G_2 = \varnothing$ (*порожній граф*, рис. 7.6).

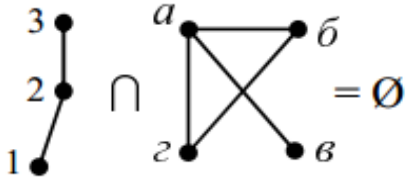


Рис. 7.6. Порожній граф як результат перетину графів

Якщо $V_1 \cap V_2 \neq \varnothing$, а $E_1 \cap E_2 = \varnothing$, то $G = G_1 \cap G_2$ називають **цілком незв’язним** графом (в нього нема жодного ребра) і позначають N_n, множина його вершин дорівнює $V_1 \cap V_2$ (рис. 7.7).

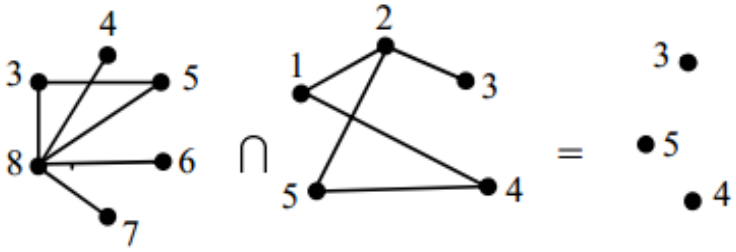


Рис. 7.7. Нуль-граф як результат перетину графів

Нехай граф $H = (V_H, E_H)$ є реберно породженим підграфом на множині ребер E_H.

Означення 1.8. Різницею $G \backslash H$ **графів** G і H називають граф з множиною ребер $E_G \backslash E_H$, множина вершин різниці складається з вершин $v \in V_G \setminus V_H$ і з тих вершин $v \in V_H$, які інцидентні деякому ребру з множини $E_G \backslash E_H$. **Симетричною різницею** $G \Delta H$ **графів** G і H називають граф, реберно породжений на множині ребер $E_G \Delta E_H$. **З’єднанням** $G+H$ **графів** G і H називають граф, який утворюється з об’єднання $G \cup H$, якщо кожну вершину графа G з’єднати ребром з кожною вершиною графа H при умові $V_G \cap V_H = \varnothing$.

Означення 1.9. Граф називають **зв’язним**, якщо його не можна зобразити як об’єднання двох непорожніх графів з множинами вершин, які не перетинаються. Граф називають **незв’язним**, якщо він не є зв’язним.

Довільний незв’язний граф G можна зобразити у вигляді об’єднання скінченної кількості зв’язних графів, кожен з них називають **компонентою зв’язності** G. Мінімальну кількість зв’язних компонент (підграфів) графа називають **зв’язністю графа** і позначають $c(G)$. Будь-який граф можна розбити на зв’язні підграфи (компоненти зв’язності).

Питання зв’язності мають важливе значення при застосуванні теорії графів до комп’ютерних мереж.

Алгоритм визначення зв’язності графа $G=(V, E)$

begin
$V'{:=}V$;
$c{:=}0$;
while $V'=\varnothing$ **do**
begin
вибрати $y \in V'$;
знайти всі вершини, з’єднані маршрутом з y;
забрати вершину y з V' і відповідні ребра з E;
$c{:=}c+1$;
end
end

Приклад 1.1. Прослідкуємо за роботою алгоритму зв’язності на графі, зображеному на рис. 7.8.

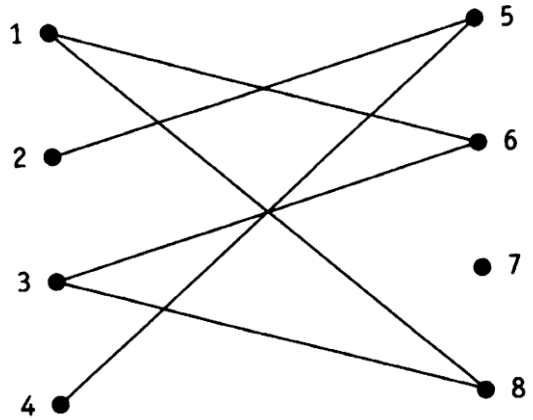


Рис. 7.8. Граф

Розв’язання.

Будуємо таблицю (табл. 1.1)

Отже, $c(G)=3$. Відповідні компоненти зв’язності подані на рис. 7.9.

Таблиця 7.1

	V'	c
Вихідні значення	{1, 2, 3, 4, 5, 6, 7, 8}	0
Вибір y=1	{2, 4, 5, 7}	1
Вибір y=2	{7}	2
Вибір y=7	$\varnothing$	3

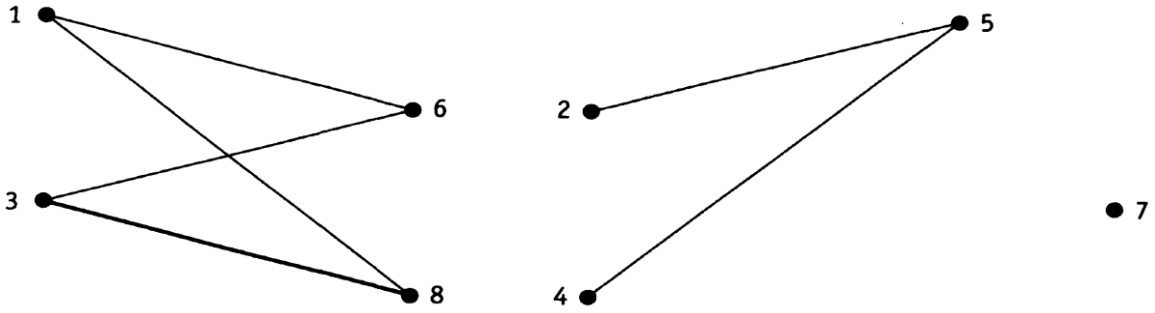


Рис. 7.9. Зв’язні підграфи графа з рис. 7.8 ▲

Означення 1.10. Доповненням до графу G називають граф $\overline{G}$, що містить ту ж саму кількість вершин, які з'єднані в ньому ребрами лише тоді, коли вони не з'єднані в графі G (рис. 7.10).

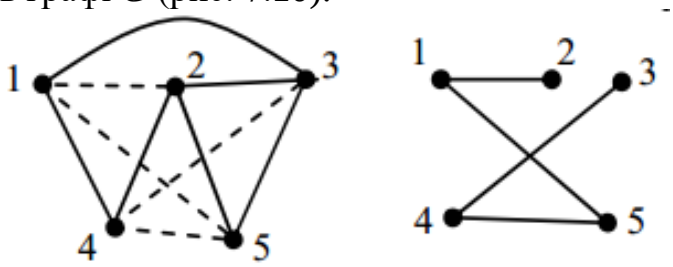


Рис. 7.10. Доповнення графу

Означення 1.11. Дві **вершини** u і v в простому неорієнтованому графі називають **суміжними**, якщо вони з'єднані якимось ребром e, про яке кажуть, що воно **інцидентне** вершині u (і v). Єдине ребро простого графа, яке з'єднує пару вершин u і v, позначають uv (або vu). Два **ребра** називаються **суміжними**, якщо вони мають спільну вершину.

Означення 1.12. Степенем (або **валентністю)** $\rho(v)$ **вершини** v називають кількість інцидентних їй ребер. Зауважимо, що петлю враховують *двічі*. Послідовність S: $d_1, d_2,\ldots, d_n$ називають **послідовністю степенів** простого зв'язного графа, якщо його вершини можна позначити через $v_1, v_2,\ldots, v_n$ так, що $\rho(v_i) = d_i$ для всіх $i \in \{1,...,n\}$.

Через $\delta(G)$ і $\Delta(G)$ позначимо відповідно найменший і найбільший степені вершин графа G.

Означення 1.13. Вершину з нульовим степенем називають **ізольованою**. Вершину зі степенем, рівним 1, називають **висячою**.

На рис. 7.11 вершина 1 є висячою, вершина 7 – ізольованою, степінь вершини 3 дорівнює двом, бо вона інцидентна ребрам $\{3,4\}$ та $\{3,6\}$.

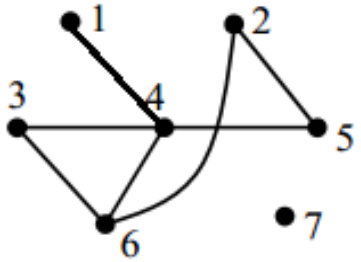


Рис. 7.11. Граф з ізольованою і висячою вершинами

Лема про естафету (*лема про рукостискання*) твердить, що сума степенів вершин *довільного* графа дорівнює подвоєному числу його ребер.

Якщо декілька людей обмінюються рукостисканнями, то кількість потиснутих рук є парною (кожна рука рахується стільки разів, у скількох рукостисканнях вона брала участь).

Теорема 7.1. Неорієнтований простий граф має парну кількість вершин непарного степеня.

Ми можемо уявляти собі множину E ребер простого графа як множину пар суміжних вершин, визначаючи тим самим нерефлексивне, симетричне відношення на множині V. Відсутність рефлексивності пов'язана з тим, що в простому графі немає петель. Симетричність відношення випливає з того

факту, що ребро *e*, яке з'єднує вершину *u* з *v*, з'єднує і *v* з *u* (інакше кажучи, ребра *неорієнтовані*, тобто не мають напряму).

7.2. Спеціальні типи простих графів

У теорії графів виділяють деякі спеціальні типи графів, які застосовують під час розв'язування багатьох задач.

Означення 2.1. Граф називають **однорідним** (**регулярним степеня *k***), якщо степені всіх його вершин рівні *k* (рис. 7.12). Регулярний граф степеня 3 називають кубічним, відомий приклад такого графа – це граф Петерсона (рис. 7.13).

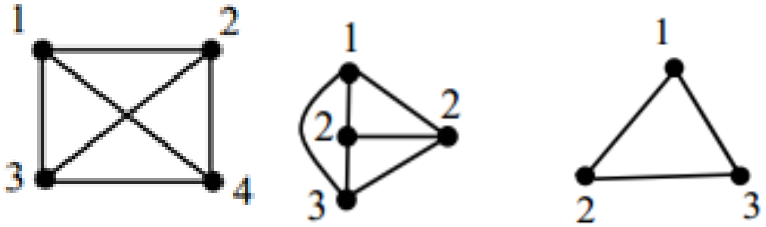


Рис. 7.12. Однорідні графи

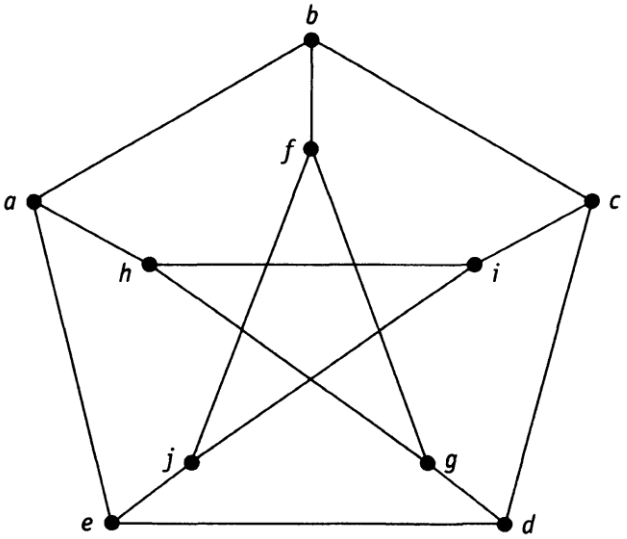
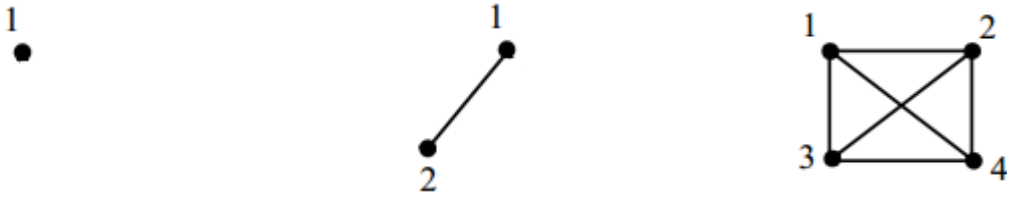


Рис. 7.13. Граф Петерсона

Сума степенів усіх вершин однорідного графу дорівнює $\rho(v)\cdot n$, де n – кількість вершин графа, а кількість ребер рівна $m=\frac{\rho(v)\cdot n}{2}$.

Означення 2.2. Простий граф називають **повним**, якщо всі його вершини з'єднані між собою лише одним ребром. Його позначають K_n, де n – кількість вершин (рис 7.14).



а) *б)* *в)*

Рис. 7.14. Повні графи: *а*) K_1, *б*) K_2, *в*) K_4

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

Повний граф K_n є регулярним степеня (n-1), а цілком незв'язний N_n – регулярний нульового степеня. Регулярними є також **платонові графи** – графи, утворені вершинами і ребрами п'яти платонових тіл: тетраеда (рис. 7.16, *а*), куба, октаедра, додекаедра (рис. 7.29) та ікосаедра (табл. 1.2).

Таблиця 7.2

	Символ Шлефлі	Кількість вершин	Кількість ребер	Кількість граней
Тетраедр	(3,3)	4	6	4
Куб (гектаедр)	(4,3)	8	12	6
Октаедр	(3,4)	6	12	8
Додекаедр	(5,3)	20	30	12
Ікосаедр	(3,5)	12	30	20

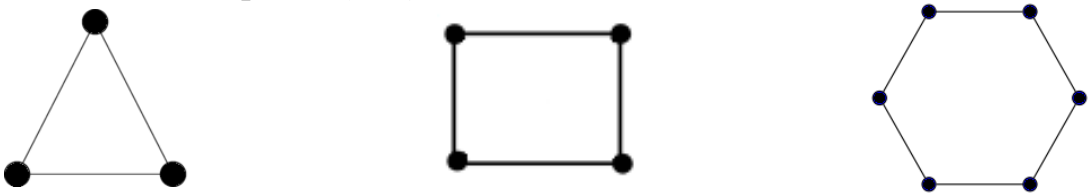
Символ Шлефлі (p, q) означає, що многогранник складається з правильних p-кутників, які утворюють при вершині q-гранний кут.

Використовуючи поняття повного графу можна інакше сформулювати означення 1.10 про доповнення до графу.

Означення 2.3. Граф $\overline{G}$ є **доповненням** до графу G, якщо їхнє об'єднання утворює повний граф, тобто $K_n = G \cup \overline{G}$, а перетин множин ребер графів G та $\overline{G}$ утворює порожню множину.

Кількість ребер у доповнювальному графі $\overline{G}$ рівна $m = C_n^2 - |E|$, де n – кількість вершин графа G, а $|E|$ – кількість ребер графа G. Кількість вершин доповнювального графа $\overline{G}$ дорівнює кількості вершин графа G.

Означення 2.4. Зв'язний простий граф з n вершинами називають **простим ланцюгом**, якщо рівно дві його вершини мають степінь 1, а всі інші – степінь 2, позначають P_n. Зв'язний простий регулярний граф степеня 2 називають **циклічним графом** або **графом-циклом (**його перша і остання вершини співпадають, тобто з'єднані у замкнуте кільце, рис. 7.15) і позначають C_n, де n – кількість вершин ($n \geq 3$).



а) *б)* *в)*

Рис. 7.15. Графи-цикли: *а*) C_3; *б*) C_4; *в*) C_6

Кількість ребер m у графі C_n рівна n.

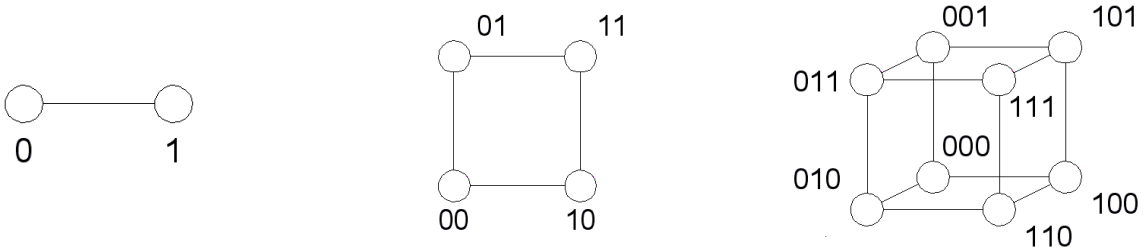
Означення 2.5. З'єднання графів N_1 та C_n називають **графом-колесом** (він отриманий з'єднанням однієї єдиної вершини з усіма вершинами графу C_n, рис. 7.16) і позначають W_n, де n – кількість вершин у графі циклі C_n ($n \geq 3$).

а) *б)* *в)*

Рис. 7.16. Графи-колеса: *а*) W_3; *б*) W_4; *в*) W_6

Кількість ребер m у графі W_n рівна $2n$, а кількість вершин дорівнює n+1.

Означення 2.6. Простий граф називаються ***n*-мірним кубом**, якщо він зображає усі бітові рядки довжиною 2^n (рис. 7.17). Його позначають Q_n. Зауважимо, що дві вершини в графі Q_n з'єднані ребром тоді, коли бітові рядки, які вони зображають, не відрізняються більше, ніж на біт.



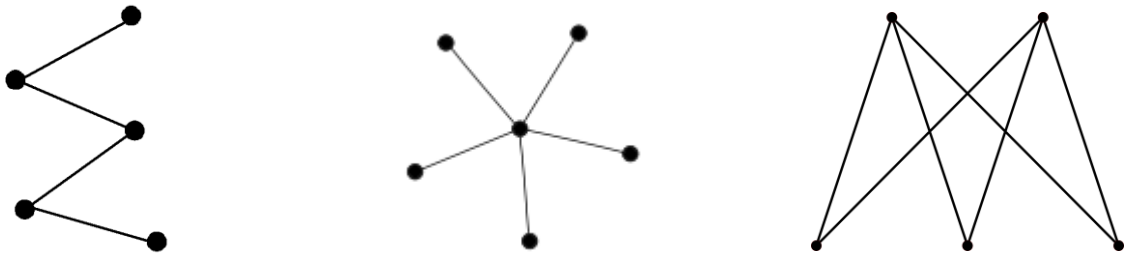
а) *б)* *в)*

Рис. 7.17. n-мірні куби: *а*) Q_1; *б*) Q_2; *в*) Q_3

Означення 2.7. Простий граф називають **двочастковим**, якщо множину його вершин V можна розбити на дві підмножини ($V = V_1 \cup V_2, V_1 \cap V_2 = \varnothing$), що не перетинаються, так, що кожне ребро з'єднує вершину з множини V_1 із вершиною з множини V_2. **Двочастковий** граф називають **повним**, якщо кожна вершина з V_1 з'єднана з усіма вершинами V_2. Його позначають $K_{k,p}$, де $k = |V_1|; p = |V_2|$. Граф $K_{1,p}$ називають **зіркою**.

Приклад двочасткових графів наведено на рис 7.18.

Кількість ребер у графі $K_{k,p}$ рівна $m = k \cdot p$, а кількість вершин $n = k+p$.



а) *б)* *в)*

Рис. 7.18. Двочасткові графи: *а*) неповний двочастковий граф; *б*) зірка $K_{1,5}$; *в*) повний двочастковий граф $K_{2,3}$

Означення 2.8. Маршрутом довжини k в графі G називають таку послідовність вершин v_1, v_2,…, v_{k+1}, що для кожного $i \in \{1,\ldots,k\}$ пара $v_i v_{i+1}$ утворює ребро графа. Позначають такий маршрут $v_1 \to v_2 \ldots \to v_{k+1}$. **Цикл** – це маршрут, який з'єднує вершину саму з собою. Маршрут або цикл називають **простим**, якщо він не містить ребер, що повторюються. Граф, в якому нема циклів, називають **ациклічним**.

Структури *дерев*, які виникають в обчисленнях, є частковим випадком ациклічних графів. Ними ми займемося в дев'ятій темі.

Приклад 2.1. Знайдемо маршрут довжиною 4 та цикли в графі (рис. 7.19).

Розв'язання.

Наприклад, 1-4-3-2-5 – це маршрут довжиною 4.

У цьому графі є 2 різних цикли довжиною 5:

1-3-2-5-4-1 і 1-2-5-4-3-1.

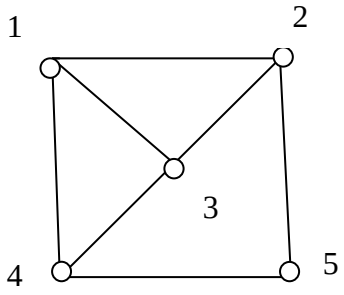


Рис. 7.19. Граф

Ми можемо пройти ці цикли як в одному напрямі, так і в іншому, починаючи з довільної вершини циклу. Крім того, в графі є 3 різні цикли довжиною 4:

1-2-5-4-1, 1-2-3-4-1 і 2-5-4-3-2,

і 2 цикли довжиною 3:

1-2-3-1 і 1-3-4-1. ▲

Теорема 7.2 (критерій двочастковості простого графа, теорема Кеніга) Для того, щоб граф був двочастковим, необхідно і достатньо, щоб він не містив простих циклів із непарною довжиною.

Ця теорема дає простий спосіб розпізнавання двочастковості графа, який називають *пошуком вшир*.

Означення 2.9. Множину вершин, суміжних із вершиною *u*, називають її **оточенням**.

Алгоритм працює так. Починаючи з довільної вершини, приписують їй номер 0. Кожній вершині з оточення вершини 0 приписують номер 1. Далі розглядають почергово оточення всіх вершин з номером 1, і всім вершинам, що їм належать і ще не мають номера, приписують номер 2 (рис. 7.20). Процес присвоювання номерів продовжують, доки це можливо. Якщо граф зв'язний, то пошук ушир занумерує всі його вершини. Далі розіб'ємо множину вершин на дві підмножини, до однієї долучимо всі вершини з парними номерами та 0, до іншої – з непарними. Розглянемо породжені два підграфи. Якщо обидва вони порожні (достатньо перевірити, що всі пари вершин з однієї підмножини не суміжні), то початковий граф є двочастковим, в протилежному випадку – не двочастковний. Зазначимо, що існує й *інший*, більш розповсюджений *варіант пошуку вшир*, коли всі вершини отримують різні номери (його розглянемо в цій темі пізніше).

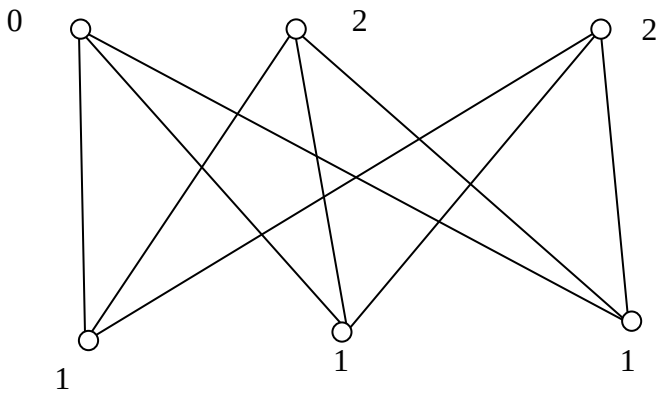


Рис. 7.20. Розпізнавання двочастковості графа

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

Теорема 7.3. Якщо простий граф має n вершин і k компонент, то кількість його ребер m задовольняє нерівність

$$n-k \le m \le (n-k)(n-k+1)/2.$$

7.3. Способи подання графів

Найзрозумілішим і найпростішим для людини є *графічний спосіб* подання графу, коли його зображають на рисунку як сукупність точок, з'єднаних між собою лініями. Але цей спосіб не придатний для опрацювання графів на комп'ютері. Тому розглянемо інші способи подання неорієнтованого графа.

Означення 3.1. Матриця інцидентності (МІ) – матриця, у якій для кожного ребра вказані інцидентні йому вершини (рядкам матриці відповідають номери вершин, стовпцям – ребра графа), тобто МІ – це прямокутна матриця розміром $n\times m$, де n – кількість його вершин, m – кількість ребер.

Для неорієнтованого простого графа елементи МІ знаходять так:

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

Зауважимо, що МІ простого графа в кожному стовпці містить точно дві одиниці і не має однакових стовпців, МІ мультиграфа має однакові стовпці, які відповідають кратним ребрам, МІ псевдографа не булева, бо у відповідних позиціях матриці ставимо 2 (в ньому є петлі).

Приклад 3.1. Побудуємо МІ для псевдографа, зображеного на рис. 7.21.

Розв'язання.

Граф має 5 вершин і 10 ребер (для зручності кожне з них позначене малою латинською літерою), отже, МІ має п'ять рядків та 10 стовпців (табл. 3.1). Оскільки ребро a з'єднує вершини 1 та 2, то в першому стовпці записуємо елементи m_{11}=1 та m_{21}=1, вершина 3 має 2 петлі, тому m_{34}=2, m_{35}=2 і т. д.

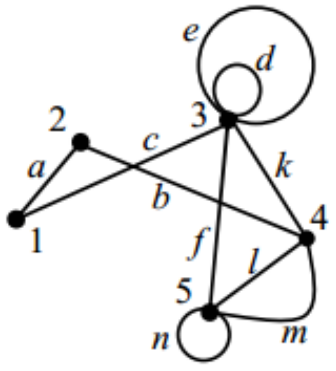


Рис. 7.21. Псевдограф

Таблиця 7.3

	a	*b*	*c*	*D*	*e*	*f*	*k*	*l*	*m*	*n*
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	1	2	2	1	1	0	0	0
4	0	1	0	0	0	0	1	1	1	0
5	0	0	0	0	0	1	0	1	1	2

▲

Для комп'ютера подання графа матрицею інцидентності є незручним, бо, по-перше, для неї потрібно $n \times m$ комірок пам'яті, більшість із яких зайняті нулями, по-друге, знаходження інформації є тривалим, зокрема, щоб отримати відповідь на питання, чи існує ребро $v_i v_j$, у найгіршому випадку потрібно перебрати всі стовпці матриці, тобто виконати m кроків.

Означення 3.2. Матрицю відношення на множині вершин графа, яке задається його ребрами, називають **матрицею суміжності (МС)**, тобто МС – це квадратна матриця розміром $n \times n$, де n – кількість його вершин. Рядки і стовпці матриці відповідають вершинам графа, а на їх перетинах записують числа, які показують, скільки ребер з'єднують відповідні вершини.

Симетричність відношення в термінах матриці суміжності M означає, що вона симетрична відносно головної діагоналі.

Для неорієнтованого графа елементи МС знаходять так:

$$m_{ij} = \begin{cases} k_{ij}, \ v_i v_j \in E, \\ 0, \text{ інакше}, \end{cases},$$

де $v_i v_j$ – ребро графа, k_{ij} – кількість ребер, що виходять з вершини v_i та входять у вершину v_j.

Зауважимо, що МС простого графу є булевою, а внаслідок нерефлексивності цього відношення на її головній діагоналі знаходиться цифра 0. Матриці суміжності мультиграфа та псевдографа не будуть булевими: елемент a_{ij} дорівнюватиме кратності ребер, що з'єднують відповідні вершини v_i та v_j, при цьому МС мультиграфа матиме всі нулі на головній діагоналі, а псевдографа – ні.

Приклад 3.2. Побудуємо матрицю суміжності для псевдографа, наведеного на рис. 7.22.

Розв'язання.

У графі шість вершин, отже, МС має 6 рядків і 6 стовпців (табл. 3.2).

Оскільки вершина 1 не має петлі, то елемент m_{11}=0, вершини 1 і 2 з'єднані трьома кратними ребрами, тому m_{12}=3 і т. д.

Таблиця 7.4

	1	2	3	4	5	6
1	0	3	1	0	0	2
2	3	0	1	1	0	0
3	1	1	0	0	1	0
4	0	1	0	2	2	1
5	0	0	1	2	1	1
6	2	0	0	1	1	1

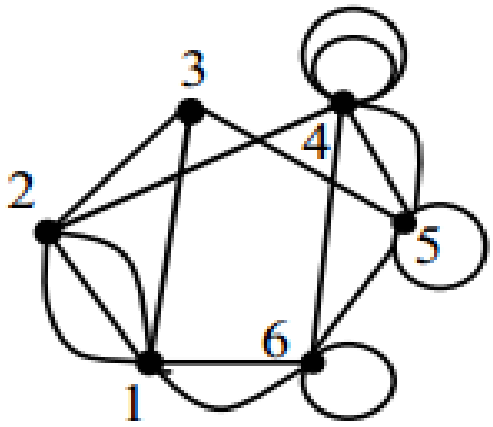


Рис. 7.22. Псевдограф

▲

За допомогою МС легко можна обчислити степінь будь-якої вершини. Для цього достатньо додати всі числа відповідного рядка (або стовпця) і до

результату додати число, що знаходиться на перетині даного рядка (або стовпця) з головною діагоналлю. Наприклад, степінь вершини 4 дорівнює (1+2+2+1)+2. Велика перевага МС як способу подання графу – швидкий доступ до інформації: за один крок можна одержати відповідь на питання, чи існує ребро з v_i у v_j. Окрім того, перевагою МС є легка *перевірка на тип* графа – простий, мультиграф чи псевдограф. Якщо матриця містить лише 0 та 1, то маємо простий граф. Якщо серед елементів є числа, відмінні від одиниць та нулів, і всі діагональні елементи є нулями – це мультиграф. Якщо в матриці хоч один діагональний елемент не рівний нулю – це псевдограф. Недолік такого способу подання полягає в тому, що незалежно від кількості ребер обсяг пам'яті становить n^2 комірок.

Приклад 3.3. Нарисуємо граф $G=(V, E)$ з множиною вершин $V=\{a, b, c, d, e\}$ і множиною ребер $E=\{ab, ae, bc, bd, ce, de\}$. Випишемо його МС.

Розв'язання.

Граф зображений на рис. 7.23.

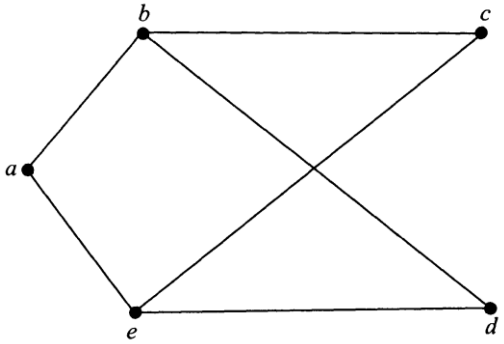


Рис. 7.23. Простий граф

Його матриця суміжності має вигляд:

	a	*b*	*c*	*d*	*e*
a	0	1	0	0	1
b	1	0	1	1	0
c	0	1	0	0	1
d	0	1	0	0	1
e	1	0	1	1	0

Для відновлення графа нам досить лише тих елементів матриці суміжності, які розміщені над головною діагоналлю. ▲

Означення 3.3. Список пар (список ребер) – це спосіб подання графа, при якому пара $[v_i, v_j]$ відповідає ребру $v_i v_j$.

Він є економнішим щодо пам'яті, особливо коли кількість ребер m є значно меншою, ніж кількість вершин n, адже обсяг пам'яті дорівнює $2m$. Недолік – велика (порядку m) кількість кроків для знаходження множини вершин, до яких ідуть ребра з заданої. Ситуацію можна поліпшити, упорядкувавши множину пар лексикографічно та застосувавши двійковий пошук.

Приклад 3.4. Побудуємо список пар для графа, наведеного на рис. 7.24.

Розв'язання.

У список записуємо в лексикографічному порядку усі існуючі ребра (табл. 3.3). Кількість пар дорівнює кількості ребер графу.

Таблиця 7.5

Вершина виходу	*Вершина входу*
v_1	v_1
v_1	v_2
v_1	v_3
v_2	v_2
v_2	v_4
v_3	v_4
v_3	v_4
v_3	v_4

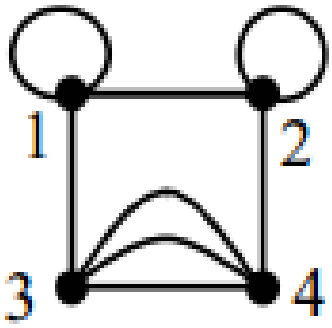


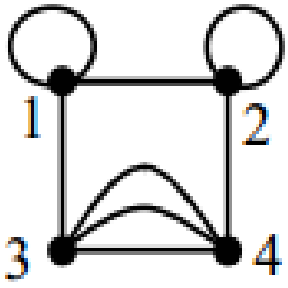
Рис. 7.24. Псевдограф ▲

Означення 3.4. Список суміжності – це спосіб подання графа, при якому для кожної вершини вказують список вершин, у які вона входить.

У табл. 7.4 подано список суміжності неорієнтованого графа, зображеного на рис. 7.25. Цей спосіб подання графа використовують тоді, коли кількість ребер значно менша, ніж кількість вершин в степені 2 ($m<<n^2$), а також для випадків *динамічних* графів, коли у графі постійно додаються нові вершини та ребра.

Головний недолік такого способу подання графу – це неможливість швидкої перевірки наявності ребра v_iv_j.

Таблиця 7.6



Вершина виходу	*Вершина входу*
v_1	v_1, v_2, v_3
v_2	v_1, v_2, v_4
v_3	v_1, v_4, v_4, v_4
v_4	v_2, v_3, v_3, v_3

Рис. 7.25. Неорієнтований граф

7.4. Вершинна та реберна зв'язність

Означення 4.1. Числом вершинної зв'язності (числом зв'язності) $\kappa(G)$ простого графа G називають найменшу кількість вершин, вилучення яких дає незв'язний або одновершинний граф, при цьому вершину вилучають з інцидентними їй ребрами.

Граф G, зображений на рис. 7.26, зв'язний, але його зв'язність можна порушити вилученням вершини u, тому, $\kappa(G)=1$. Якщо ж спробувати порушити зв'язність цього графа вилученням ребер (а не вершин), то треба вилучити не менше трьох ребер.

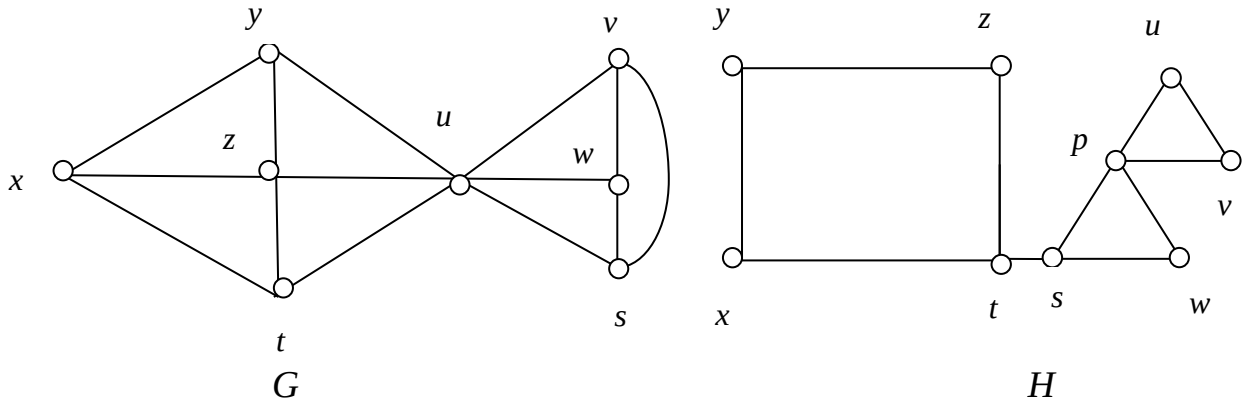


Рис. 7.26. Зв'язний граф

Означення 4.2. Числом реберної зв'язності $\lambda(G)$ простого графа G з $n>1$ вершинами називають найменшу кількість ребер, вилучення яких дає незв'язний граф (але вершини, які з'єднує це ребро, залишаються). Число реберної зв'язності одновершинного графа вважають рівним 0.

Для графа G з рис. 7.26 $\lambda(G)=3$.

Означення 4.3. Вершину простого графа називають **точкою з'єднання**, якщо в разі її вилучення новий граф матиме більше компонент, ніж даний. Ребро графа називають **мостом**, якщо його вилучення збільшує кількість компонент.

Отже, точки з'єднання і мости – це свого роду «вузькі місця» простого графа. Граф H з рис. 7.26 має три точки з'єднання *t, s, p* та один міст *ts*.

Означення 4.4. Простий граф називають **t-зв'язним (вершинно-t-зв'язним)**, якщо $\kappa(G)\geq t$ і **реберно-t-зв'язним**, якщо $\lambda(G)\geq t$.

Граф G, зображений на рис. 7.26, однозв'язний і реберно-3-зв'язний.

Неважко побачити, що зв'язність (вершинна і реберна) незв'язного графа дорівнює нулю, $\kappa(K_n)=\lambda(K_n)=n-1$, $\kappa(C_n)=2$.

Теорема 7.4. Для кожного простого зв'язного графа G маємо

$$\kappa(G)\leq\lambda(G)\leq\delta(G),$$

де $\delta(G)$ – найменший степінь вершин графа G.

Нерівності цієї теореми покращити не можна – має місце теорема.

Теорема 7.5. Для довільних цілих чисел a, b, c $(0<a\leq b\leq c)$ ***існує граф G, для якого*** $\kappa(G)=a$, $\lambda(G)=b$, $\delta(G)=c$.

Якщо зв'язати ці величини з кількістю вершин графа, то отримаємо таку умову.

Теорема 7.6. Якщо граф має n вершин і $\delta(G)\geq[n/2]$, ***то*** $\lambda(G)=\delta(G)$.

Наведемо достатні умови того, що простий граф вершинно-t-зв'язним.

Теорема 7.7 (теорема Бонді). Нехай G – простий граф з n вершинами, які позначені так, що $\rho(v_1)\leq\rho(v_2)\leq\ldots\leq\rho(v_n)$. ***Тоді граф G є вершинно-t-зв'язним, якщо*** $\rho(v_r)\geq r+k-1$ для $1\leq r\leq n-1-\rho(v_{n-k+1})$.

7.5. Ейлерові графи

Модель поставленої Ейлеру задачі – це мультиграф, складений з множини вершин і множини ребер, що з'єднують ці вершини. Вершини *A, B, C* і *D* символізують береги ріки і острови, а ребра *a, b, c, d, e, f* і *g* позначають 7

мостів (рис. 7.27). Шуканий маршрут (якщо він існує) відповідає обходу ребер мультиграфа так, що кожне з них проходять лише один раз. Прохід ребра, очевидно, відповідає переходу ріки по мосту. Отже, задачу про кенігсбергські мости мовою теорії графів можна сформулювати так: чи існує в мультиграфі простий цикл, який містить усі його ребра?

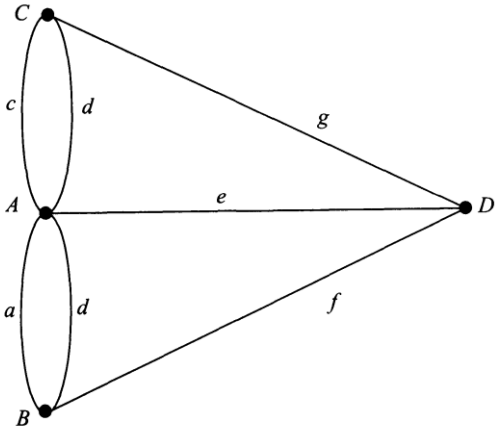


Рис. 7.27. Модель задачі про мости Кенігберга

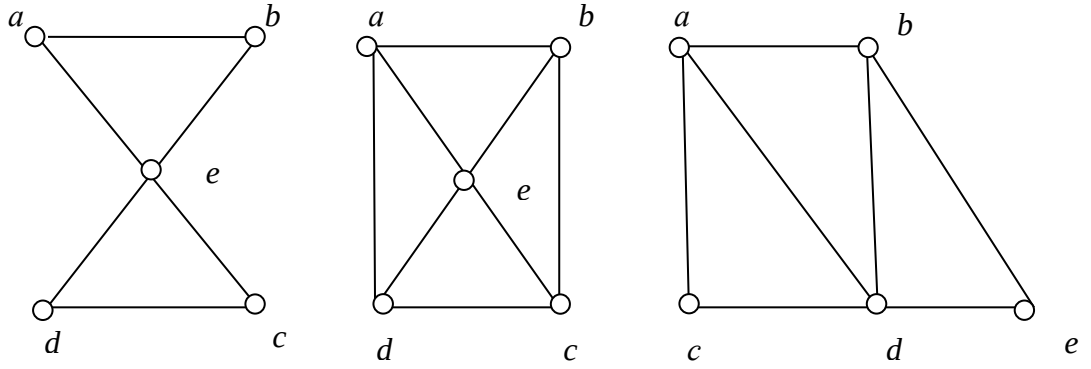
Означення 5.1. Ейлеровим циклом (ЕЦ) у зв'язному мультиграфі називають простий цикл, що містить усі ребра графа. **Ейлеровим маршрутом (ЕМ)** у зв'язному мультиграфі називають простий маршрут, що містить усі ребра графа.

Означення 5.2. Ейлеровим графом (ЕГ) називають граф, який містить ейлеровий цикл, тобто у ньому знайдеться маршрут, що починається і закінчується в тій самій вершині і проходить через *усі ребра* графа лише один раз.

Приклад 5.1. Визначимо, які з графів, зображених на рисунку 7.28, мають ЕЦ і ЕМ.

Розв'язання.

Граф G_1 має ЕЦ, наприклад, *a-e-c-d-e-b-a*; граф G_2 не має ні ЕЦ, ні ЕМ; граф G_3 не має ЕЦ, але має ЕМ: *a-c-d-e-b-d-a-b*.



G_1 G_2 G_3

Рис. 7.28. Ейлерові цикли і ейлерові маршрути ▲

Ейлер зауважив, що якщо в графі є ейлеровий цикл, то для кожного ребра, що входить у якусь вершину, має існувати інше ребро, що з неї виходить (бо з умови задачі, зайшовши у вершину, ми не можемо вийти по тому ж ребру) і отримав з цього простого спостереження такий висновок: *якщо в графі існує ейлеровий цикл, то до кожної вершини має підходити парна кількість ребер*.

Крім того, Ейлеру вдалося довести й протилежне твердження і таким чином сформулювати теорему.

Теорема 7.8 (теорема Ейлера). Граф, в якому будь-яка пара вершин зв'язана деякою послідовністю ребер, є ейлеровим тоді і лише тоді, коли всі його вершини мають парний степінь.

Тепер цілком очевидно, що в графі, яким змодельована задача про мости Кенігсберга, неможливо знайти ейлерового циклу. Дійсно, степені всіх його вершин непарні: $\rho(B)=\rho(C)=\rho(D)=3$ і $\rho(A)=5$.

Алгоритм Флері побудови ейлерового циклу

У процесі побудови ЕЦ здійснено нумерацію його ребер.

Крок 1. Починаємо з довільної вершини *u* та присвоюємо довільному ребру *uv* номер *k*:=1. Викреслюємо це ребро і переходимо у вершину *v*.

Крок 2. Вибираємо довільне ребро, інцидентне вершині *v*, причому міст вибираємо лише тоді, коли немає інших можливостей. Присвоюємо йому номер *k*:=*k*+1 і викреслюємо його.

Крок 3. Якщо всі ребра графа викреслено та пронумеровано, то **кінець** (вказані номери задають послідовність ребер в ЕЦ), інакше переходимо на крок 2.

Теорема 7.9. Зв'язний мультиграф має ЕМ, але не має ЕЦ, тоді і лише тоді, коли він має точно дві вершини непарного степеня.

Доведення. Необхідність. Припустимо, що зв'язний мультиграф має ЕМ від *u* до *v*, але не має ЕЦ. Перше ребро циклу додає 1 до степеня вершини *u*. Щоразу, коли маршрут проходитиме через вершину *u*, він додаватиме до її степеня 2. Останнє ребро шляху додасть 1 до степеня вершини *v*, а кожне проходження маршруту через вершину *v* додаватиме до її степеня 2. Отже, вершини *u* та *v* мають непарний степінь. Усі інші вершини – парного степеня, бо маршрут додає 2 до степеня вершини, коли проходить через неї.

Достатність. Припустимо, що граф *G* має точно 2 вершини, нехай *u* та *v*, з непарним степенем. Розглянемо граф *G*', одержаний з графа *G* додаванням нового ребра *uv*. Кожна вершина графа *G*' має парний степінь, отже, *G*' має ЕЦ. Тепер вилучимо нове ребро і одержимо ЕМ у графі *G*. Теорему доведено.

Зазначимо, що будь-який ЕМ починається в одній з цих двох вершин непарного степеня, а закінчується в іншій.

Оскільки степені всіх вершин графа, яким змодельована задача про мости Кенігсберга, непарні, то він не має навіть ЕМ, тобто неможливо пройти кожний міст по одному разу, навіть якщо не потрібно повертатись у початкову точку.

7.6. Гамільтонові графи

Ми вивчили ейлерові графи, які володіють замкнутим маршрутом, що проходить через всі ребра графа лише один раз.

Означення 6.1. Маршрут, який проходить через кожну *вершину* графа лише один раз, називають **гамільтоновим маршрутом.** Цикл, який проходить через кожну *вершину* графа (окрім першої і останньої) лише один раз, називають **гамільтоновим циклом (ГЦ)**, а відповідний граф – **гамільтоновим графом (ГГ).**

Термін «гамільтоновий» походить від імені відомого ірландського математика В. Гамільтона, який 1857 р. запропонував гру «Навколосвітня

подорож». Кожній з 20-ти вершин додекаедра (правильного 12-тигранника, грані якого є п’ятикутниками) приписано назву одного з великих міст світу. Потрібно, розпочавши з довільного міста, відвідати решту 19 міст лише один раз і повернутись у початкове; перехід дозволено ребрами додекаедра.

Цю задачу зображено на площині (рис. 7.29). Вона зводиться до відшукання в графі ГЦ. Один з можливих розв’язків показано потовщеними лініями.

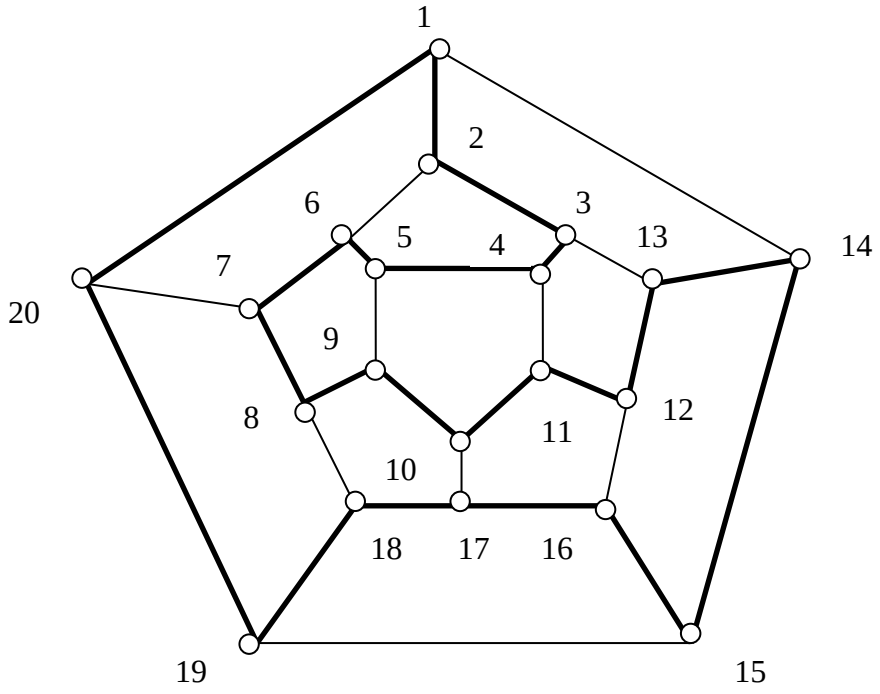


Рис. 7.29. Гамільтоновий цикл

Не всі зв’язні графи мають ГЦ хоча б тому, що такі графи мають бути *двозв’язними* (необхідна умова), але граф, що має дві точки з’єднання, може не мати ГЦ. Граф, зображений на рис. 7.30, є двозв’язним (можна, наприклад, вилучити точки 1 і 6), але цього недостатньо для наявності ГЦ.

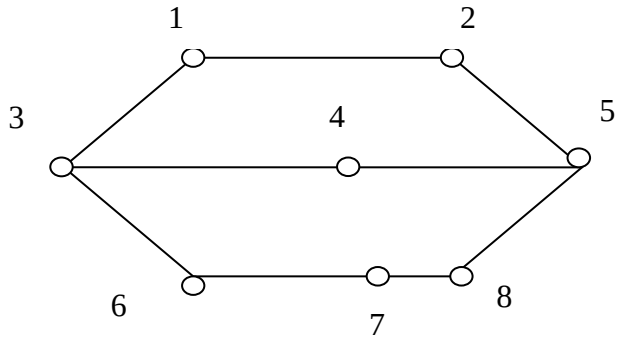


Рис. 7.30. Двозв’язний граф без гамільтонового циклу

Гамільтонові графи служать моделлю при складанні розкладів руху поїздів, для телекомунікаційних мереж і т.д. На відміну від задачі Ейлера, *простого критерію*, чи граф є гамільтоновий, поки що *немає*. Пошук хорошого критерію залишається однією з головних нерозв’язаних задач теорії графів.

Тим не менше, багато графів є гамільтоновими, зокрема, у будь-якому *повному графі* можна знайти ГЦ. Розглянемо повний граф K_5, зображений на рис. 7.31. Його цикл *a-b-c-d-e-a* є гамільтоновим, в ньому є й інші ГЦ. Оскільки кожна вершина суміжна з іншими, то починаючи з вершини *a*, як другу вершину циклу ми можемо вибрати будь-яку з 4-х інших. Далі у нас буде 3

варіанти вибору для 3-ї вершини і 2 для 4-ї, після чого ми вернемося у вершину *a*. Тому ми маємо 4x3x2=24 цикли. Але оскільки кожен цикл може проходити як в одному напрямі, так і в іншому, то реально в графі K_5 є лише 12 різних ГЦ (дві різні послідовності вершин *a-b-c-d-e-a* і *a-e-d-c-b-a* задають, зрозуміло, той самий цикл).

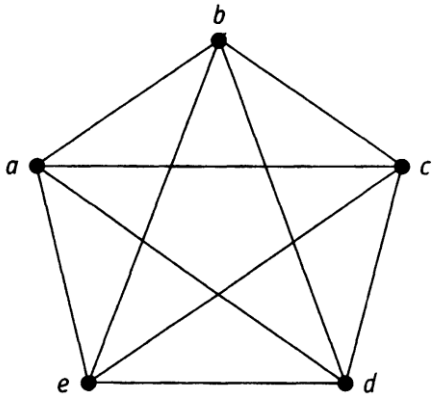


Рис. 7.31. Повний граф K_5

Пошук ГЦ (якщо він існує) в довільному зв'язному графі – задача дуже непроста і трудоємна.

Приклад 6.1. Покажіть, що граф, зображений на рис. 7.32, не є гамільтоновим.

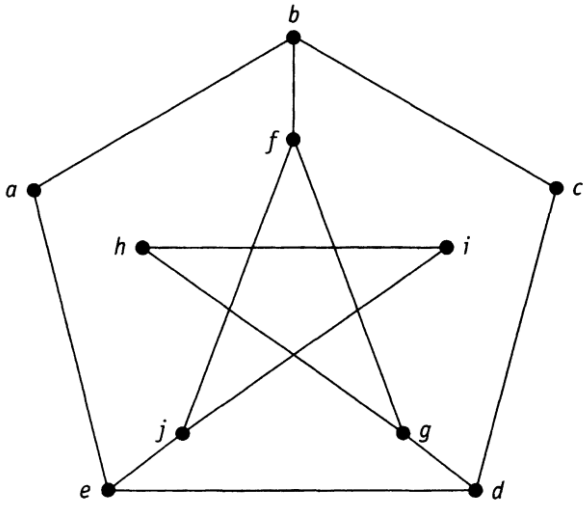


Рис. 7.32. Негамільтоновий граф

Розв'язання.

Допустимо, що в зв'язному графі знайдеться ГЦ. Кожну вершину *v* включають в ГЦ *C* вибором двох інцидентних з нею ребер, тому степінь кожної вершини в ГЦ (після забирання зайвих ребер) дорівнює 2. Степені вершин даного графа – 2 або 3. Вершини степеня 2 входять в цикл разом з обома інцидентними з ними ребрами. Отже, ребра *ab, ae, cd, cb, hi, hg* і *ij* в тому чи іншому порядку входять в ГЦ *C* (рис. 7.33).

Ребро *bf* не може бути частиною циклу *C*, бо кожна вершина такого циклу має мати степінь 2, тому ребра *fj* і *fg* повинні входити в цикл *C*, щоб включити в нього вершину *f*. Але тоді ребра *je* і *gd* ніяк не можуть належати циклу *C*, бо інакше в ньому з'являться вершини степеня 3. Це змушує нас включити в цикл ребро *ed*, що приводить нас до суперечності: ребра, які ми змушені вибрати, утворюють два незв'язних цикли, а не один, існування якого ми допускали. Висновок: граф, зображений на рис. 7.32, не є гамільтоновим.

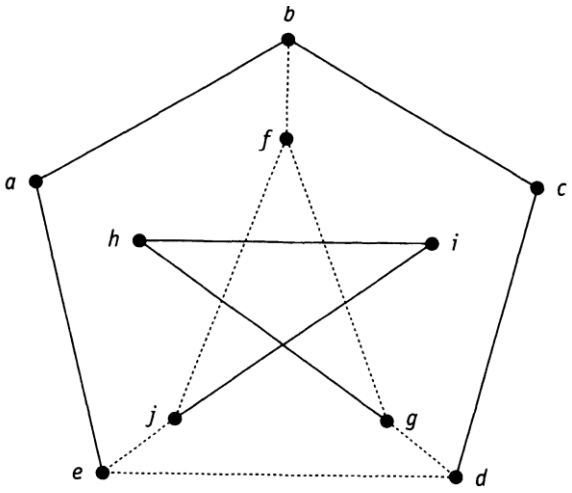


Рис. 7.33. Ребра, які входять у гамільтоновий цикл C ▲

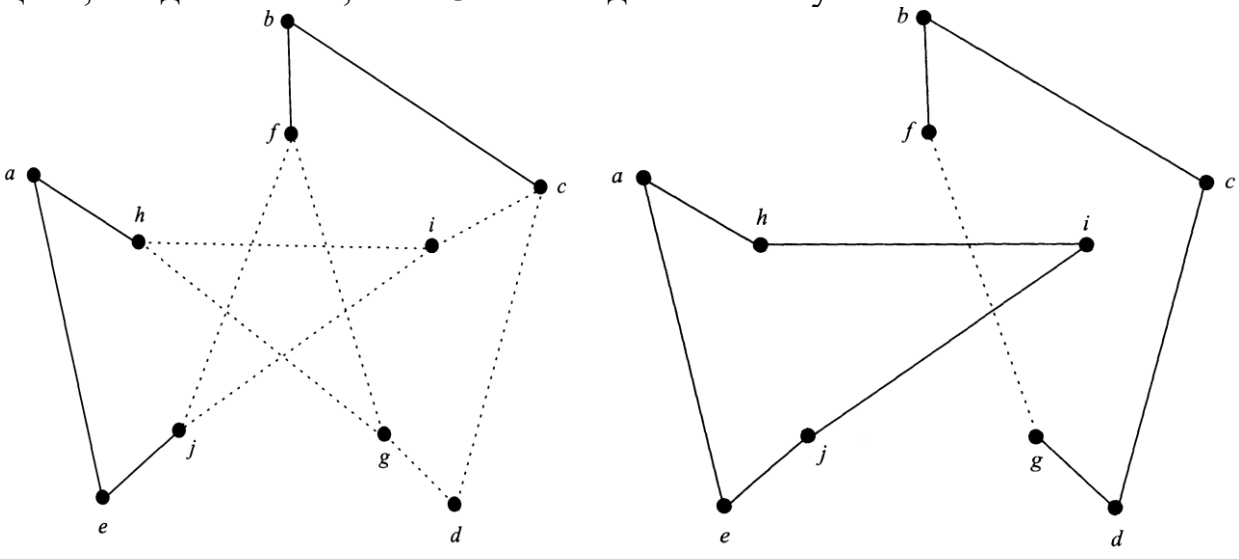
Приклад 6.2. Знайдіть в графі Петерсона P (рис. 7.13) цикл довжиною 9. Покажіть, що P не є гамільтоновим.

Розв'язання.

Цикл довжиною 9: *a-b-f-j-i-c-d-g-h-a*.

Позначимо гіпотетичний ГЦ через C. Оскільки ГЦ має пройти через вершини зовнішнього п'ятикутника і через вершини зірки, він має містити одне з ребер: *ah, bf, ci, dg* або *ej*. Без обмеження загальності можна допустити, що цикл C включає в себе ребро *ah*. Тепер, щоб включити в нього вершину *a*, треба використати ребро *ab* або *ae*. Внаслідок симетричності графа, не порушуючи загальності міркувань, включимо ребро *ae*. Пригадавши, що в ГЦ кожна вершина має мати степінь 2 (одне ребро підходить до неї, інше виходить), розуміємо, що цикл C не може містити ребра *ab*. З іншого боку, ми маємо пройти через ребро *b* (ввійти в нього і вийти), тому він містить два ребра: *bf* і *bc*.

Вершина *e* теж має входити в ГЦ зі степенем 2, тому цикл C має містити ще *одне* з ребер *ej* або *de*. Розглянемо ці два випадки окремо. Допустимо, що в цикл *входить ребро ej*. Тоді ребро *ed* не можна розглядати. Зобразимо граф Петерсена (рис. 7.34, *а*), виділивши на ньому ребра, які ми вже включили в цикл, і видаливши ті, які в C вже входити не можуть.

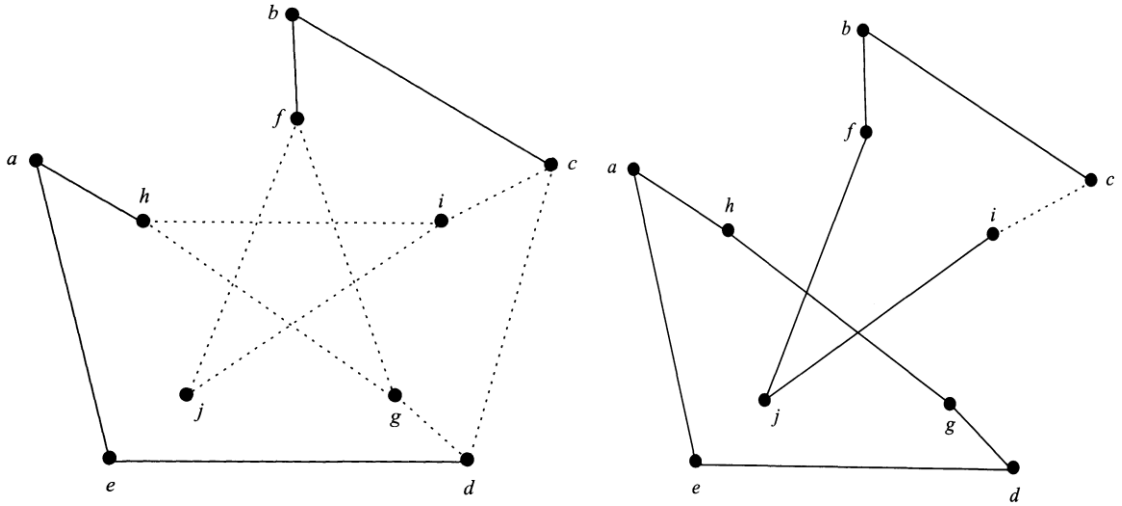


а *б*

Рис. 7.34. Окремі ребра гіпотетичного ГЦ C з ребром *ej*

З рис. 7.34, *а*, видно, що через вершину *d* можна пройти лише по ребрах *dg* і *dc*, тому їх неодхідно включити в цикл. Після цього ребро *ci* виявляється зайвим і його треба видалити. У графі залишилось лише 2 ребра *hi* та *ji*, інцидентних вершині *i*, додавши вершину *i* з вказаними ребрами, видалимо при цьому *hg* і *jf* (рис. 7.34, *б*). Тепер ми вимушені включити в цикл ребро *fg*, щоб мати можливість пройти через вершини *f* і *g*. У результаті *C* буде складатися з двох незв'язних частин, тобто взагалі не буде циклом, тому вибір ребра *ej* веде до невдачі у побудові ГЦ.

Допустимо тепер, що ми *включили в цикл ребро ed*, а ребро *ej*, навпаки, видалили (рис. 7.35, *а*). Для приєднання вершини *j* до циклу нам треба додати ребра *fj* та *ji*, але з включенням *fj* ребро *fg* виявиться зайвим, тому його видалимо. Тепер у вершини *g* залишилось лише два ребра *gd* і *gh*, тому приєднуємо їх, а ребра *hi* та *cd* відкидаємо як непотрібні (рис. 7.35, *б*). З останнього рисунку бачимо, що нам треба приєднати до циклу ребро *ic*, що знову приводить до двох незв'язних компонент.



а *б*

Рис. 7.35. Окремі ребра гіпотетичного ГЦ *C* з ребром *ed*

Одержана суперечність доводить, що в графі *P* нема ГЦ, тобто, що він не є гамільтоновим. ▲

ГГ застосовують для моделювання багатьох практичних задач. Основою всіх таких задач є класична *задача комівояжера*:

Комівояжер повинен здійснити поїздку містами і повернутися, побувавши в кожному місті лише один раз з мінімальними витратами на пересування.

Означення 6.2. Граф, кожному ребру якого поставлено у відповідність число (*вагу*), називають **зваженим**.

Графічна модель задачі комівояжера складається з ГГ, вершини якого відповідають містам, а ребра – дорогам, які їх зв'язують. Вага ребра позначає транспортні витрати, необхідні для мандрівки відповідною дорогою, такі, як, наприклад, відстань між містами чи час руху по дорозі. На жаль, ефективний алгоритм розв'язування даної задачі поки що невідомий. Для складних мереж кількість ГЦ, які необхідно переглянути для виділення мінімального, надзвичайно велика. Однак існують алгоритми пошуку *субоптимального розв'язку*. Субоптимальний розв'язок необов'язково дасть цикл мінімальної

загальної ваги, але знайдений цикл буде мати, як правило, значно меншу вагу, ніж більшість довільних ГЦ. Розглянемо один з таких алгоритмів.

Алгоритм найближчого сусіда

Він знайде субоптимальний розв'язок задачі комівояжера, генеруючи гамільтонові цикли у зваженому графі з множиною вершин *V*. Цикл, отриманий в результаті роботи алгоритму, співпадатиме з кінцевим значенням змінної *маршрут*, а його загальна довжина – кінцеве значення змінної *w*.

begin
вибрати $v \in V$;
маршрут:=*v*;
w:=0;
$v' = v$;
відмітити *v*';
while залишаються невідмічені вершини **do**
begin
вибрати непозначену вершину *u*, найближчу до *v*';
маршрут:=*маршрут* - *u*;
w:=*w* + вага ребра *v*'*u*;
v':=*u*;
відмітити *v*';
end
маршрут:=*маршрут* - *v*;
w:=*w* + вага ребра *v*'*v*;
end

Приклад 6.2. Застосуємо алгоритм найближчого сусіда до графа, зображеного на рис. 7.36. За вихідну вершину візьмемо вершину *D*.

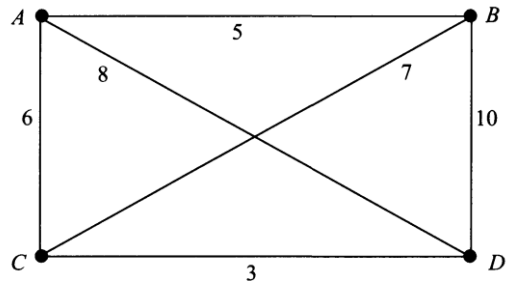


Рис. 7.36. Зважений граф для пошуку ГЦ

Розв'язання.

Запишемо таблицю (табл. 7.7).

Таблиця 7.7

	u	*маршрут*	*w*	*v*'
Вихідні значення	_	*D*	0	*D*
	C	*D-C*	3	*C*
	A	*D-C-A*	9	*A*
	B	*D-C-A-B*	14	*B*
Останній прохід	*B*	*D-C-A-B-D*	24	*B*

У результаті роботи алгоритму знайдено ГЦ *D-C-A-B-D* загальною вагою 24. Роблячи повний перебір всіх циклів в цьому малому графі, можна знайти ще 2 інших ГЦ: *A-B-C-D-A* загальною вагою 23 і *A-C-B-D-A* загальною вагою 31. ▲

У повному графі з 20 вершинами існує приблизно $6{,}1\cdot 10^{16}$ ГЦ, перебір яких вимагає надзвичайно багато машинної пам'яті і часу. Вивчення достатніх умов наявності в графі ГЦ – один з важливих напрямів у теорії графів. Інтуїтивно зрозуміло, що граф з багатьма ребрами, достатньо рівномірно розподіленими, має великий шанс містити ГЦ.

Ми вже зауважували, що такого гарного критерію гамільтоновості графа, як теорема 7.8 для ейлерових графів, нема. Достатні умови гамільтоновості дає, наприклад, така теорема.

Теорема 7.10. Простий граф з послідовністю степенів S: $d_1 \le d_2 \le \ldots \le d_n$, $n \ge 3$***, вершинами є гамільтоновим, якщо виконується одна з таких умов:***

1) $d_j \ge n/2$ для всіх $j = \overline{1,n}$;

2) для кожної пари несуміжних вершин v і w маємо $\rho(v) + \rho(w) \ge n$;

3) для всіх k: $1 \le k \le n/2$ виконується $d_k \ge k$;

4) для всіх j,k, де $j<k$, з того, що $d_j \le j \wedge d_k \le k-1$ випливає, що $d_j + d_k \ge n$;

5) з того, що $d_k \le k < n/2$ випливає, що $d_{n-k} \ge n-k$.

Причому маємо: 1) $\Rightarrow$2) $\Rightarrow$3) $\Rightarrow$4) $\Rightarrow$5).

Доведення.

1) $\Rightarrow$ 2) Запишемо всі вершини графа у вигляді формального циклу:

$$v_1 \to v_2 \to \ldots \to v_n \to v_1.$$

Нехай k пар сусідніх вершин не є суміжні, для визначеності, нехай $v_1=u$ і $v_n=v$ не є суміжні. Тоді існують вершини v_i і v_{i+1}, $i = \overline{2,n-2}$, такі, що v і v_i та u і v_{i+1} попарно суміжні. Покажемо це. Нехай це не так. Це означає, що після кожної вершини v_i, яка суміжна з вершиною v, з'являється вершина v_{i+1}, яка не є суміжна з u. Позначимо через $\bar{\rho}(u)$ кількість вершин графа, які не суміжні з вершиною u. Тоді $\rho(v) \le \bar{\rho}(u)$. Оскільки $\rho(u) + \rho(v) \ge n$, то і $\rho(u) + \bar{\rho}(u) \ge n$. Але це суперечить тому, що $\rho(u) + \bar{\rho}(u) + 1 = n$. Тому завжди за таких умов існує гамільтоновий цикл:

$$u \to v_1 \to v_2 \to \ldots \to v_i \to v \to v_{n-2} \to v_{n-3} \to \ldots \to v_{i+1} \to u.$$

Розглянемо формальний цикл

$$v_1 \to v_i \to v_{i-1} \to \ldots v_2 \to v_{i+1} \to v_n \to v_1.$$

Кількість несуміжних сусідніх вершин у ньому вже не більша за k-1. Продовжуючи цю дію, отримаємо гамільтоновий цикл.

2) $\Rightarrow$ 3) Нехай $v_1, v_2, \ldots, v_n$ – нумерація вершин, узгоджена з послідовністю степенів S. Доводимо від супротивного. Припустимо, що умова 2) виконується, а 3) – ні. Це означає, що існує таке k: $1 \le k \le n/2$, що $d_k < k$. Степені всіх вершин $v_1, v_2, \ldots, v_k$ не більші числа k, і всі вони суміжні між собою (інакше ми мали б $\rho(v_i) + \rho(v_j) \le 2k < n$, що суперечить умові 2)). Тому

кожна з цих вершин суміжна щонайбільше з однією з вершин $v_{k+1},...,v_n$. Отже, існує вершина v_s, $k+1 \le s \le n$, для якої $\rho(v_s) \le n-k-1$ і яка несуміжна з жодною з вершин $v_1, v_2, ..., v_k$. Тому за умовою 2) маємо $\rho(v_s) \ge n - d_k$. Отримали протиріччя: $n-k-1 \ge n-d_k$.

3) $\Rightarrow$ 4) Умова $d_k > k$ рівносильна умові $d_k \ge k+1$. Тому з умови 3) безпосередньо випливає, що для $k \ge n/2$ маємо $d_k > n/2$. Якщо $d_j \le j \wedge d_k \le k-1$, то $j, k \ge n/2$, тому $d_j + d_k \ge n$.

4) $\Rightarrow$ 5) Припустимо, що умова 5) не виконується. Тоді існує j: $d_j \le j < n/2$, для якого $d_{n-j} < n-j$ або $d_{n-j} \le n-j-1$. Приймемо $k=n-j$, тоді $d_j + d_k \le j + (n-j-1) = n-1$. Протиріччя.

5) Доводимо від протилежного. Припустимо, що існує простий негамільтоновий граф G, послідовність степенів якого володіє властивістю 5). Не зменшуючи загальності, можемо вважати, G – максимальний граф з такою властивістю, тобто долучення до нього будь-якого ребра перетворить граф G у гамільтоновий.

Нехай u і v – несуміжні вершини, такі, що сума $\rho(u)+\rho(v)$ є максимальною (для визначеності, нехай $\rho(u) \le \rho(v)$). Долучивши до графа ребро uv, отримаємо гамільтоновий цикл, який це ребро містить. Це означає, що граф містить гамільтоновий маршрут, який з'єднує вершини u і v:

$$u = u_1 \to u_2 \to ... \to u_n = v.$$

Для жодного $i \in \{1,...,n\}$ пари вершин u і u_{i+1} та v і u_i не можуть бути одночасно суміжні, то тоді у графі існував би гамільтоновий цикл:

$$u \to u_{i+1} \to u_{i+2} \to ...v \to u_i \to u_{i-1} \to ... \to u.$$

Тому кількість $\bar{\rho}(v)$ вершин графа, які не суміжні з вершиною v, не менша від кількості вершин, суміжних з u: $\bar{\rho}(v) \ge \rho(u)$. Але $\bar{\rho}(v) = n - \rho(v) - 1$, тому $\bar{\rho}(v) + \rho(v) \le n-1$, звідки отримуємо $\rho(u) < n/2$.

Для всіх вершин w, які не суміжні з вершиною v, $\rho(w) \le \rho(u)$, оскільки сума $\rho(u)+\rho(v)$ є максимальною. Оскільки таких вершин не менше, ніж $\rho(u)$, то приймаємо $k=\rho(u)$. Будемо мати $d_k \le k < n/2$, тому за умовою теореми отримаємо, що $d_{n-k} \ge n-k$. Це означає, що існує не менше, ніж $k+1$ вершина, для якої $\rho(w) \ge n-k$. Але оскільки $\rho(u)=k$, то принаймні з одною такою вершиною v' вершина u не суміжна. Маємо $\rho(u)+\rho(v') \ge n > \rho(u)+\rho(v)$. Оскільки остання сума була вибрана максимальною, то ми отримали протиріччя.

Доведення завершено.

Зауважимо, що жодна з умов 1)-5) може не виконуватися, але граф таки буде гамільтоновим. Найпростіший приклад – циклічний граф $C_n, n \ge 6$. Його послідовність степенів S: 2, 2,..., 2 не задовольняє жодну з цих умов.

Приклад 6.3. Показати, що довільний граф з n вершинами, який має не менше, ніж $C_{n-1}^2 + 2$ ребра, має гамільтоновий цикл.

Розв'язання.

Покажемо, що виконується умова 2) теореми 7.5. Припустимо, що існують дві вершини v_1 і v_2, такі, що $\rho(v_1)+\rho(v_2)<n$. Тоді граф матиме щонайбільше C_{n-2}^2+n-1=(n^2-3n+4)/2 ребер, що менше за C_{n-1}^2+ 2 =(n^2-3n+6)/2.

Як знайти ГЦ або переконатись, що його немає? Очевидний алгоритм повного перебору всіх можливостей, тобто $n!$ перестановок усіх вершин графа і перевірок практичного застосування не має. Ми розглянемо практично застосовний *алгоритм бектрекінг* у дев'ятій темі.

Задача знаходження ГЦ *NP*-повна, тобто належить до класу задач, для яких невідомий (і є вагомі підстави вважати, що його не існує) ефективний (тобто з поліноміальною складністю) алгоритм їхнього розв'язання.

7.7. Обхід графів

Існує багато алгоритмів на графах, які грунтуються на систематизованому переборі їхніх вершин або обході вершин, під час якого кожна вершина одержує унікальний порядковий номер. Алгоритми обходу вершин графа називають *методами пошуку*.

7.7.1. Пошук углиб у простому зв'язному графі (DFS – Depth First Search)

Нехай $G=(V, E)$ – простий зв'язний граф, усі вершини якого позначені різними символами. У процесі пошуку вглиб вершинам графа надають номери та певним чином позначають ребра. У ході роботи алгоритму використовують структуру даних для збереження множин, яку називають **стеком**. Зі стеку можна вилучити лише той елемент, який було додано до нього останнім, тобто він працює за принципом «останній прийшов – перший вийшов» (LIFO – last in, first out). Додавання і вилучення елементів у стеку відбувається з одного кінця, який називають *верхівкою стеку*, номер вершини x позначають DFS(x).

Алгоритм пошуку углиб у простому зв'язному графі

Крок 1. Починаємо з довільної вершини v_s та присвоюємо DFS(v_s):=1. Включаємо цю вершину в стек.

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

Крок 3. Нехай xy – непозначене ребро. Якщо DFS(y) вже визначено, то позначаємо ребро xy штриховою лінією (або тонкою суцільною) та переходимо до кроку 2. Якщо DFS(y) не визначено, то позначаємо ребро xy потовщеною суцільною лінією, визначаємо DFS(y) як черговий DFS-номер, включаємо цю вершину в стек і переходимо до кроку 2.

Крок 4. Виключаємо вершину x зі стеку. Якщо стек порожній, то **кінець**, інакше – перейдемо до кроку 2.

Щоб вибір був однозначним, доцільно домовитись, що вершини, суміжні з тією, яка вже отримала DFS-номер, аналізують *за зростанням їх нумерації* (або *в алфавітному порядку*). Динаміку роботи алгоритму зручно відображати

протоколом обходу графа пошуком углиб – таблицею з трьома стовпцями: вершина, DFS-номер, вміст стеку.

Приклад 7.1. Виконаємо обхід графа, зображеного на рис. 7.37, пошуком углиб, починаючи з вершини *b*.

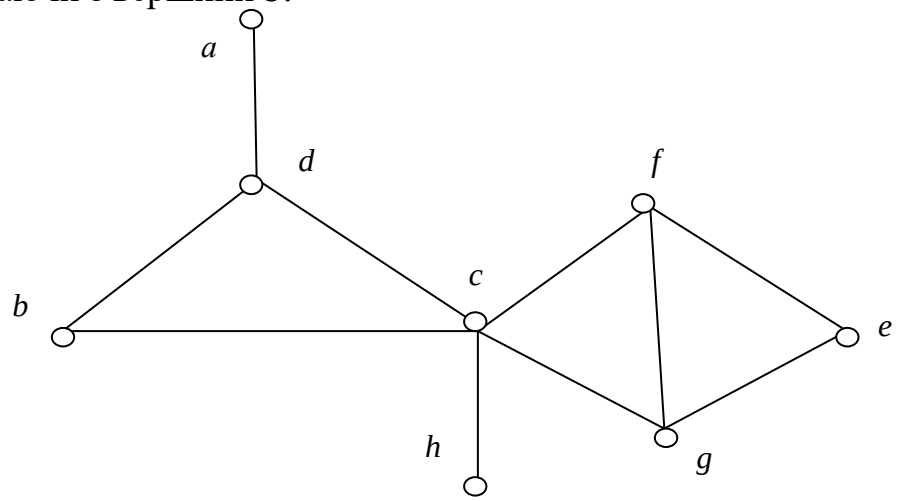


Рис. 7.37. Граф

Розв’язання.

Розв’язок подано на рис. 7.38, протокол пошуку вглиб – в табл. 7.8 (у ній в третьому стовпці вважаємо, що верхівка стеку праворуч).

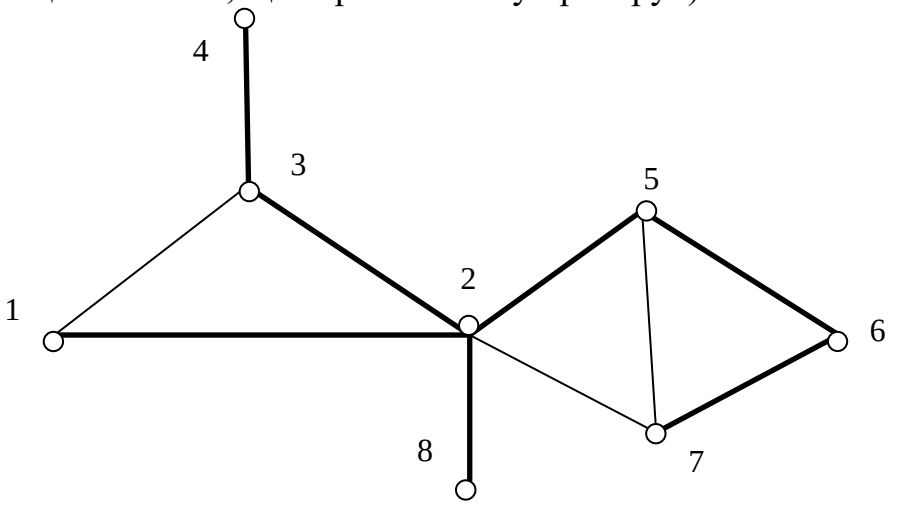


Рис. 7.38. Пошук углиб у графі

Таблиця 7.8

Вершина	DFS-номер	Вміст стеку
b	1	*b*
c	2	*b-c*
d	3	*b-c-d*
a	4	*b-c-d-a*
-	-	*b-c-d*
-	-	*b-c*
f	5	*b-c-f*
e	6	*b-c-f-e*
g	7	*b-c-f-e-g*
-	-	*b-c-f-e*
-	-	*b-c-f*

-	-	*b-c*
h	8	*b-c-h*
-	-	*b-c*
-	-	*b*
-	-	∅

Ребра, які позначено потовщеною суцільною лінією, називають *прямими*, а звичайною – *зворотними*. Їх використовують у різних алгоритмах, заснованих на пошуку углиб.▲

Приклад 7.2. Здійснити пошук углиб графу, зображеного на рис. 7.39, починаючи з вершини *a*.

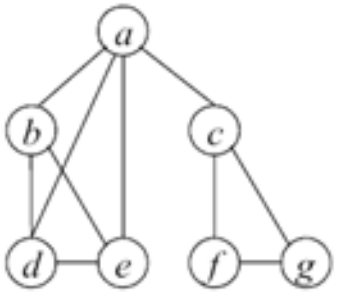


Рис. 7.39. Граф

Розв’язання.

Розв’язок подано на рис. 7.40, протокол пошуку вглиб – в табл. 7.9

Таблиця 7.9

Вершини	DFS-Номер	Вміст стеку
A	1	*a*
B	2	*ab*
D	3	*abd*
E	4	*abde*
-	-	*abd*
-	-	*ab*
-	-	*a*
C	5	*ac*
F	6	*acf*
G	7	*acfg*
-	-	*acf*
-	-	*ac*
-	-	*a*
-	-	-

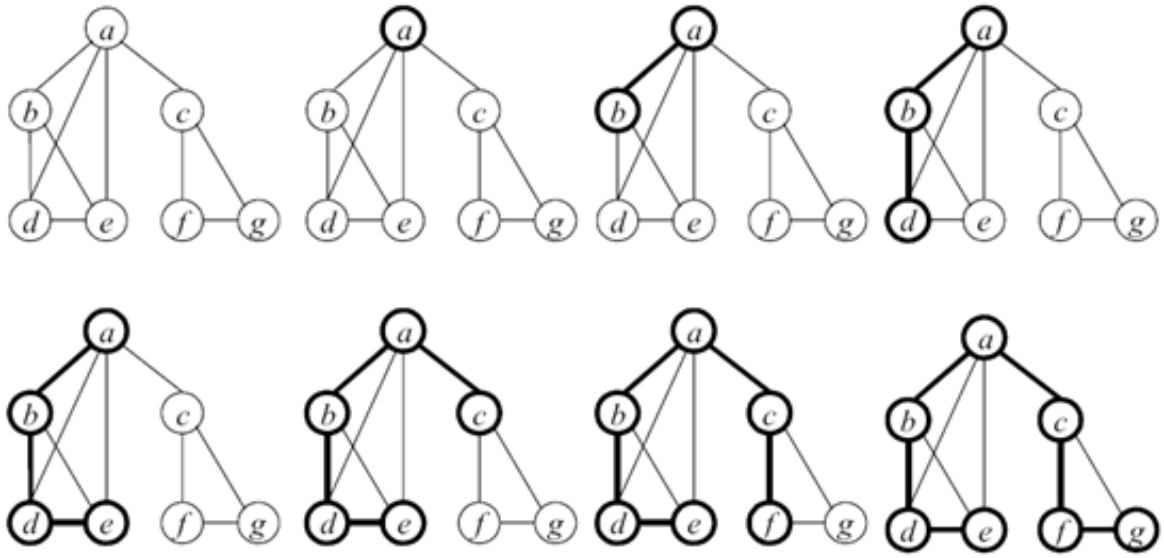


Рис. 7.40. Демонстрація методу пошуку вглиб ▲

7.7.2. Пошук вшир у простому зв'язному графі (BFS – Breadth First Search)

Під час пошуку рухаються вшир: спочатку проглядають всі сусідні вершини, їм надають BFS-номери, потім сусіди сусідів і т.д. У ході реалізації алгоритму використовують структуру даних для збереження множин, яку називають **чергою**. З черги можна вилучити лише той елемент, який перебував у ній найдовше: працює принцип «першим прийшов – першим вийшов» (FIFO – first in, first out). Елемент включають в хвіст черги (праворуч), а виключають з її голови (ліворуч). Використання вершини полягає у перегляді всіх ще не відвіданих її сусідів, номер вершини x позначають BFS(x).

Алгоритм пошуку вшир у простому зв'язному графі

Крок 1. Починаємо з довільної вершини v_s та присвоюємо BFS(v_s):=1. Включаємо цю вершину в чергу.

Крок 2. Розглядаємо вершину, яка знаходиться на початку черги: нехай це буде вершина x. Якщо для всіх вершин, суміжних з вершиною x, уже визначені BFS-номери, то переходимо до кроку 4, інакше – до кроку 3.

Крок 3. Нехай xy – ребро, в якому номер BFS(y) не визначено, тоді позначимо це ребро xy потовщеною суцільною лінією, визначимо BFS(y) як черговий BFS-номер, включаємо вершину y у чергу та переходимо до кроку 2.

Крок 4. Виключаємо вершину x з черги. Якщо черга порожня, то **кінець**, інакше – перейдемо до кроку 2.

Щоб вибір був однозначним, вершини, суміжні з вершиною x, аналізують *за зростанням їх нумерації* (або *в алфавітному порядку*). Динаміку роботи алгоритму зручно відображати *протоколом обходу*, аналогічним попередньому за винятком третього стовпця (тепер це вміст черги).

Приклад 7.3. Виконаємо обхід графа, зображеного на рис. 7.37, пошуком вшир, починаючи з вершини b.

Розв'язання.

Розв'язок подано на рис. 7.41, протокол пошуку вшир – в табл. 7.10.

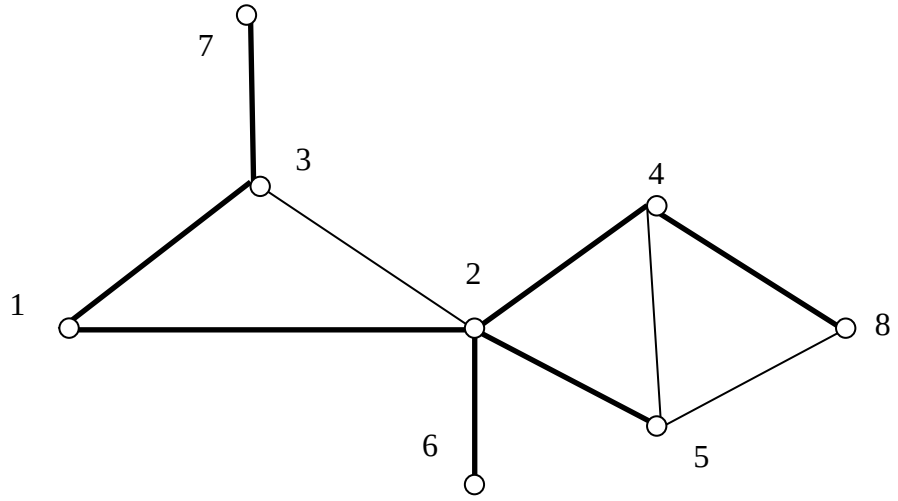
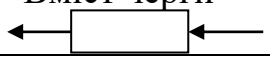


Рис. 7.41. Пошук вшир у графі ▲

Обчислювальна складність обох алгоритмів обходу однакова й у разі подання графа *списками суміжності* (цей спосіб подання графів розглянемо в наступній темі) становить $O(m+n)$, де m – кількість ребер, n – кількість вершин графа, отже, ці алгоритми мають лінійну складність, тобто їх ефективність досить висока.

Таблиця 7.10

Вершина	BFS-номер	Вміст черги 
b	1	*b*
c	2	*b, c*
d	3	*b, c, d*
-	-	*c, d*
f	4	*c, d, f*
g	5	*c, d, f, g*
h	6	*c, d, f, g, h*
-	-	*d, f, g, h*
a	7	*d, f, g, h, a*
-	-	*f, g, h, a*
e	8	*f, g, h, a, e*
-	-	*g, h, a, e*
-	-	*h, a, e*
-	-	*a, e*
-	-	*e*
-	-	∅

Приклад 7.4. Здійснити пошук вшир у графі, зображеного на рис. 7.39, починаючи з вершини *a*.

Розв’язання.

Розв’язок подано на рис. 7.42, протокол пошуку вшир – в табл. 7.11.

Таблиця 7.11

Вершини	BFS-Номер	Вміст черги
a	1	*a*
b	2	*ab*

c	3	*abc*
e	4	*abce*
d	5	*abcde*
-	-	*bcde*
-	-	*cde*
f	6	*cdef*
g	7	*cdefg*
-	-	*defg*
-	-	*efg*
-	-	*fg*
-	-	*g*
-	-	-

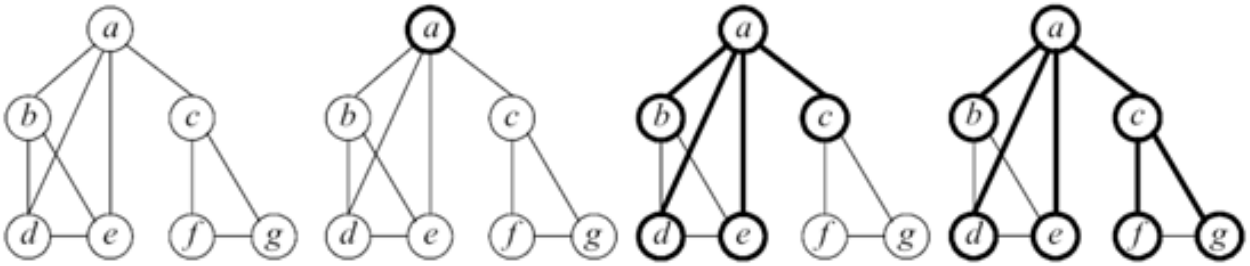


Рис. 7.42. Демонстрація методу пошуку вшир ▲

7.8. Ізоморфізм графів

Буквальний переклад слова “ізоморфізм” означає “однаковість форми”. Форма графа – це його структура. Таким чином, ізоморфізм графів означає однаковість їх структури.

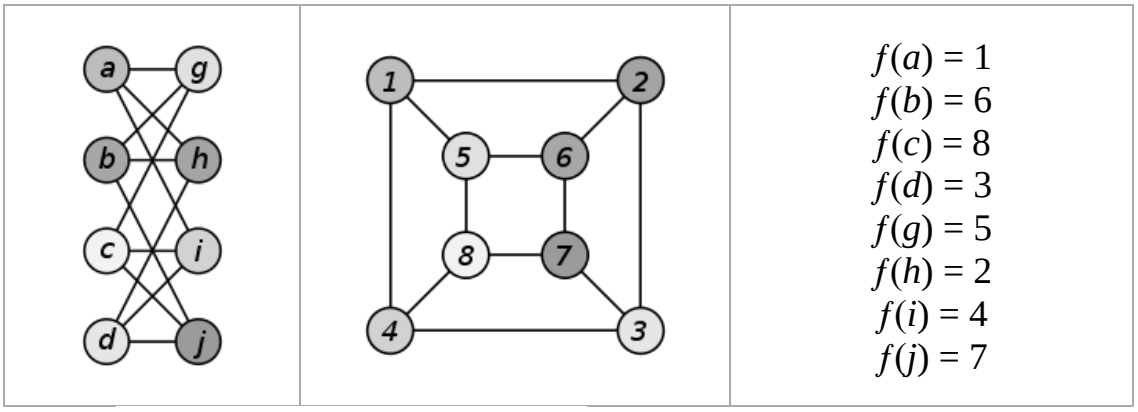
Означення 8.1. Два графи $G_1 = (V_1, E_1)$ і $G_2 = (V_2, E_2)$ називають **ізоморфними** (позначають $G_1 \cong G_2$), якщо між множинами їх вершин існує взаємно однозначне відображення $f : V_1 \to V_2$, яке зберігає суміжність, тобто для довільних вершин v і w ребро $vw \in E_1$ тоді й лише тоді, коли $f(v)f(w) \in E_2$. При цьому f називають **ізоморфним відображенням** або **ізоморфізмом** графа G_1 на граф G_2.

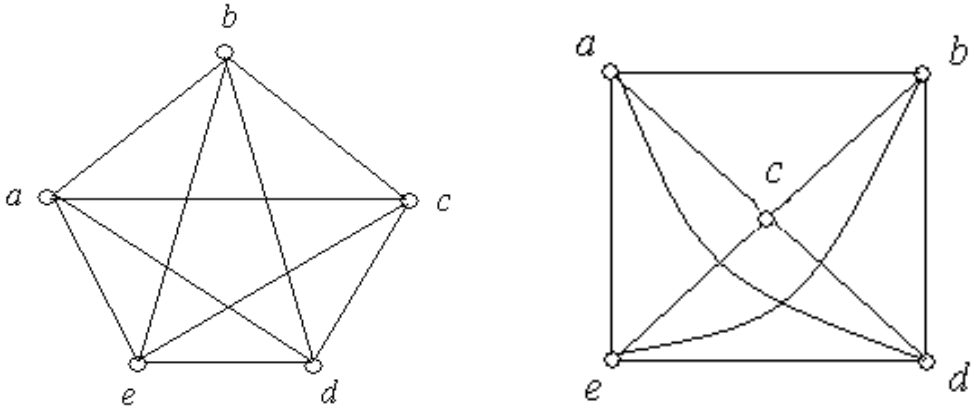
Простіше кажучи, граф, ізоморфний до заданого, – це той самий граф з точністю до позначення вершин і графічного зображення. Але побачити це часом не так просто.

Два графи, показані в табл. 7.12, ізоморфні, незважаючи на свою зовнішню відмінність. Графи, зображені на рис. 7.43 *а, б*, теж ізоморфні.

Таблиця 7.12

Граф G	*Граф H*	*Ізоморфізм між G і H*





а) *б*)

Рис. 7.43. Приклади ізоморфних графів.

Алгоритм перевірки графів на ізоморфізм (*необхідні умови*).

Крок 1. Кількості вершин графів G_1 та G_2 мають співпадати.

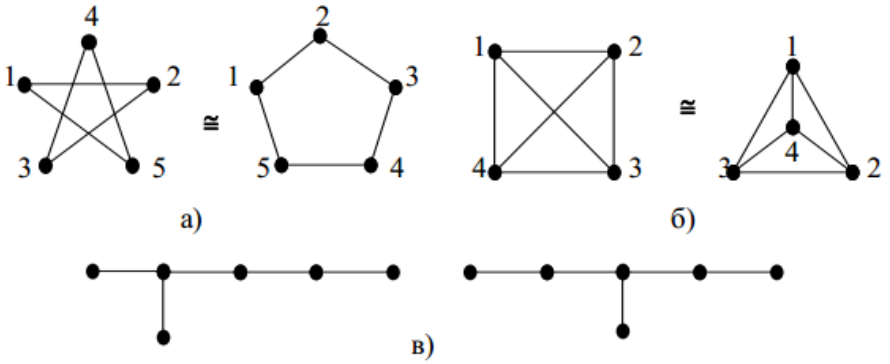
Крок 2. Кількість ребер графів G_1 та G_2 мають співпадати.

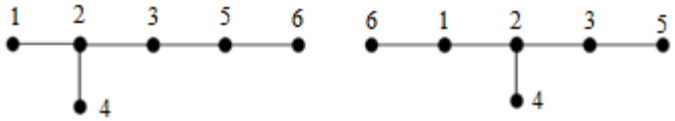
Крок 3. Степені вершин графів G_1 та G_2 мають співпадати.

Крок 4. Якщо у графі G_1 існує маршрут $v_1 \to v_2 \to \ldots \to v_n$ і $\rho(v_1)=k_1$, $\rho(v_2)=k_2$, … $\rho(v_n)=k_n$, то і у графі G_2 повинен існувати маршрут через вершини зі степенями k_1, k_2, … k_n.

Крок 5. Якщо хоча б одна з цих умов не виконується, то графи не ізоморфні. Але якщо вони всі виконуються, то це не означає, що графи ізоморфні. Тоді потрібно шукати нумерацію вершин таким чином, щоби виконувалось означення 8.1. У загальному випадку таких способів є $n!$. Отже, кроки 1-4 пришвидшують відповідь на запитання, чи графи ізоморфні.

Приклад 8.1. Показати, що пари графів, зображених на рис. 7.44 *а, б*, є ізоморфними, а пара графів на рис. 7.44 *в* не є ізоморфною.





г) *д)*

Рис. 7.44. Приклади ізоморфних та неізоморфного графів

Розв'язання.

Розглянемо пару графів (рис. 7.44, *а*). Перевіримо необхідні умови ізоморфізму.

Крок 1: кількість вершин – 5=5.

Крок 2: кількість ребер – 5=5.

Крок 3: сума степенів – 10=10.

Крок 4: сума степенів сусідніх вершин співпадає, бо кожна вершина має однаковий степінь (2).

Крок 5: Пронумерувавши кожну вершину графа, ми бачимо, що існує однозначне відображення кожної вершини обох графів.

Отже, графи на рис. 7.44, *а* ізоморфні.

Розглянемо пару графів (рис. 7.44, *б*). Перевіримо необхідні умови ізоморфізму.

Крок 1: кількість вершин – 4=4.

Крок 2: кількість ребер – 6=6.

Крок 3: сума степенів – 12=12.

Крок 4: сума степенів сусідніх вершин співпадає, бо кожна вершина має однаковий степінь (3).

Крок 5: Пронумерувавши кожну вершину графа, ми бачимо, що існує однозначне відображення кожної вершини обох графів.

Отже, графи на рис. 7.44, *б* ізоморфні.

Розглянемо пару графів (рис. 7.44, *в*). Перевіримо необхідні умови ізоморфізму.

Крок 1: кількість вершин – 6=6.

Крок 2: кількість ребер – 5=5.

Крок 3: сума степенів – 3+2x2+3=10, 10=10.

Кроки 4-5: пронумеруємо вершини обох графів (рис. 7.44, *г, д*):

Після нумерації вершин обох графів у графі 7.44, *г* є вершина 2 зі степенем $\rho(2)=3$, сусідами якої є вершини 1, 3 та 4 зі степенями $\rho(1)=1$, $\rho(3)=2$, $\rho(4)=1$ відповідно. У графі 7.44, *д* теж є вершина 2 зі степенем $\rho(2)=3$, сусідами якої є вершини 1, 3 та 4 зі степенями $\rho(1)=2$, $\rho(3)=2$, $\rho(4)=1$ відповідно. Отже, не існує однозначної відповідності вершин, тому графи не є ізоморфними. ▲

Для того, щоб граф G_1 був ізоморфним графу G_2, необхідно і достатньо, щоб існувала така підстановка, яка встановлювала б взаємно однозначну відповідність між вершинами графа, а також між їх ребрами.

Далі дамо відповідь на запитання: чи можливо встановити ізоморфізм графів за їхніми матрицями інцидентності та суміжності або за списком ребер?

Для перевірки ізоморфності графів G_1 і G_2 за матрицями суміжності необхідно визначити, чи існує така перестановка рядків і стовпців у матриці

суміжності G_1, щоб у результаті вийшла матриця G_2. З цією метою треба зробити всі можливі перестановки рядків і стовпців (а їхня максимальна кількість дорівнює $n!\cdot n!$). Якщо після однієї із цих перестановок матриці суміжності тотожньо збігаються, то графи ізоморфні.

Для перевірки ізоморфності графів G_1 і G_2 за матрицями інцидентності необхідно визначити, чи існує така перестановка рядків і стовпців у матриці інцидентності G_1, щоб у результаті вийшла матриця G_2. З цією метою треба зробити всі можливі пари перестановок рядків і стовпців (а їхня максимальна кількість дорівнює $n!\cdot m!$). Якщо після однієї із цих перестановок матриці інцидентності тотожно збігаються, то графи ізоморфні. Аналогічно, переставляючи рядки у списку ребер одного графа, можна визначити, чи він є ізоморфним іншому.

І в першому, і в другому випадку це досить трудомісткі операції, і розв'язування задачі "вручну" не завжди виправдано. Часто ізоморфність графів простіше встановити з їх графічних подань.

Наприклад, на рис. 7.45 зображено графи *G* і *H* з однаковою кількістю вершин і ребер.

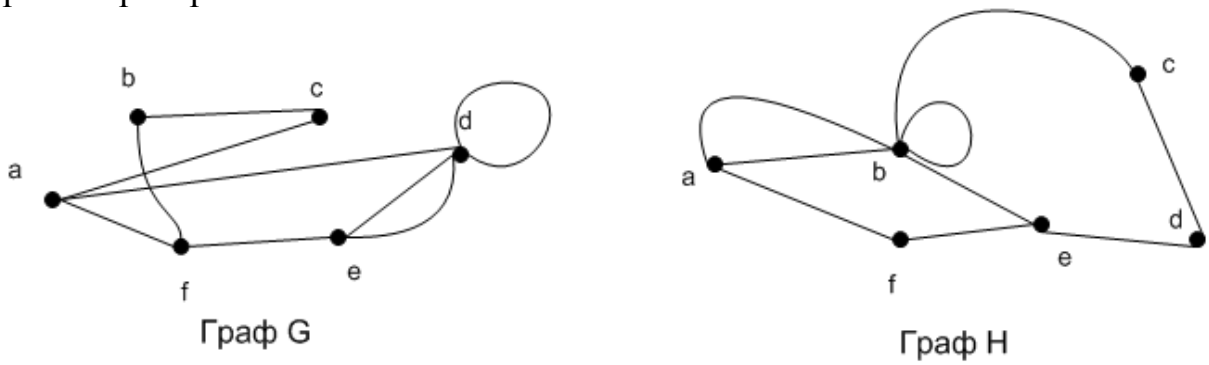


Рис. 7.45. Пара графів для перевірки на ізоморфізм

Степінь кожної вершини для цих графів запишемо в табл. 7.13:

Таблиця 7.13

Степінь у графі H	*Назва вершини*	*Степінь у графі G*
3	a	3
6	b	2
2	c	2
2	d	5
3	e	3
2	f	3

Як бачимо з табл. 7.13, степені вершин b, d та f не співпадають, тому графи не є ізоморфними.

Приклад 8.2. Знайдемо серед графів *H, K* і *L*, зображених на рис. 7.46, підграфи графа *G*.

Розв'язання.

Позначимо вершини графів *H, K* і *G* як показано на рис. 7.47. Графи *H* і *K* – підграфи в *G*, а граф *L* не є підграфом в *G*, бо у нього є вершина степеня 4, якої нема в *G*.

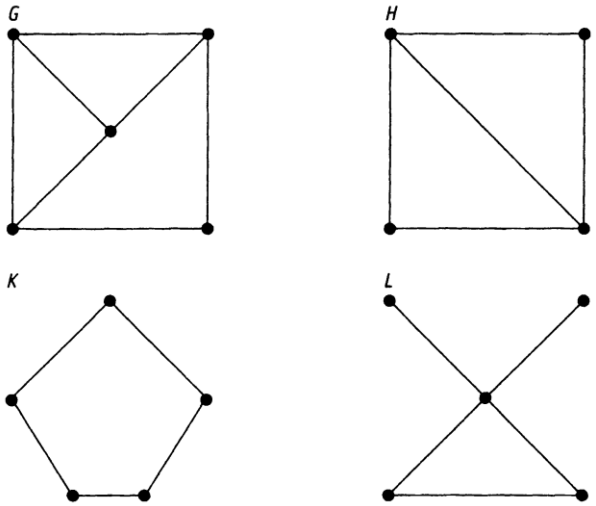


Рис. 7.46. Графи для пошуку підграфів

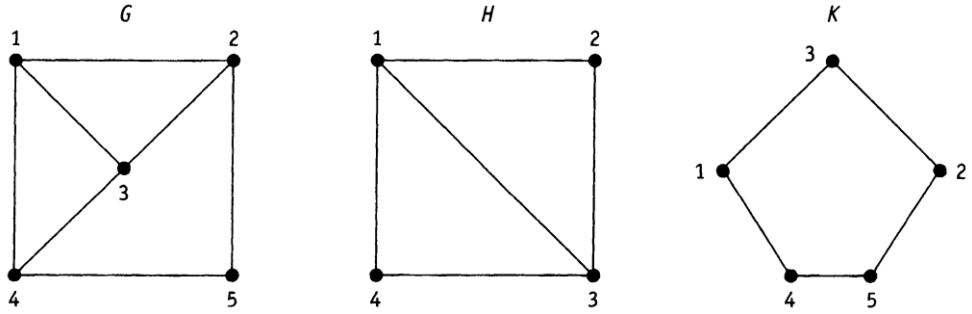


Рис. 7.47. Граф G і підграфи ▲

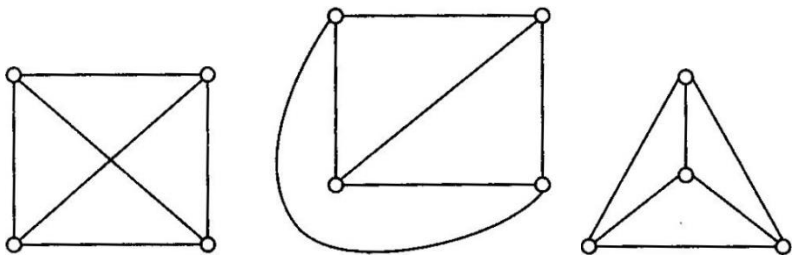
7.9. Планарні та плоскі графи

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.

Означення 9.1. Плоским називають граф, зображений на площині так, що жодні два його ребра геометрично не перетинаються ніде, окрім інцидентних їм вершин. Граф, ізоморфний до плоского, називають **планарним**.

Плоский граф розбиває площину на області (*грані)*, одна з яких необмежена (*зовнішня грань),* інші називають внутрішніми.

Наприклад, усі три графи на рис. 7.48 планарні, але лише другий і третій із них плоскі. На рис. 7.49 зображено плоский граф. Він має чотири грані: r_1, r_2, r_3, r_4, з них грань r_4 – зовнішня.



а) *б)* *в)*

Рис. 7.48. Планарні (*а-в*) та плоскі (*б, в*) графи

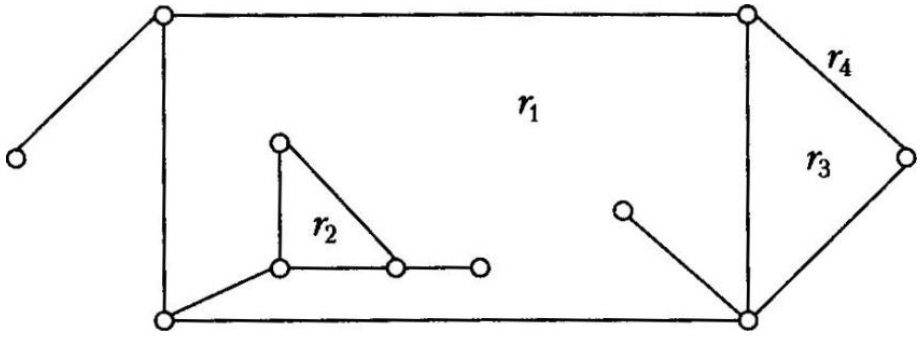


Рис. 7.49. Плоский граф з гранями

Теорема 7.11 (теорема Ейлера про плоскі графи). *Нехай зв'язний плоский граф G має n вершин, m ребер і r граней. Тоді* $n + r - m = 2$.

Доведення.

Здійснимо індукцію 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*, тоді 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 трьох випадків. Оскільки інші випадки неможливі, то індукцію завершено й теорему доведено.

Легко побачити, що формула Ейлера правильна і для кількості вершин, ребер та граней довільних многогранників. Неважко перенести цю формулу і на випадок незв'язних графів.

Теорема 7.12. *Нехай плоский граф G має n вершин, m ребер, r граней і k компонент зв'язності. Тоді* $n + r - m = k + 1$.

Доведення.

Застосуємо формулу Ейлера до кожної компоненти G_i окремо: $n_i + r_i - m_i = 2$ і просумуємо $\sum_{i=1}^{k}(n_i + r_i - m_i) = 2k$. Враховуючи, що необмежену компоненту потрібно порахувати лише раз, отримаємо $\sum_{i=1}^{k} n_i = n$, $\sum_{i=1}^{k} m_i = m$, $\sum_{i=1}^{k} r_i - (k-1) = r$, $n-m+r=2k-(k-1)$.

Теорема 7.13. *Графи* K_5 *і* $K_{3,3}$ *не планарні.*

Доведення.

Граф K_5 (див. рис. 7.31) має 5 вершин і 10 ребер. Припустимо, що він планарний; тоді існує ізоморфний до нього плоский граф. За теоремою Ейлера

$$r = m + 2 - n = 10 + 2 - 5 = 7.$$

Зазначимо, що будь-яке ребро плоского графа або розділяє дві різні грані, або є мостом. Позаяк граф K_5 не має петель і кратних ребер, то кожна грань обмежена принаймні трьома ребрами. Тому число $3r$ – оцінка знизу подвоєної кількості ребер графа, тобто $3r \leq 2m$, звідки випливає, що $21 \leq 20$. Отримали суперечність.

У графі $K_{3,3}$ (див. рис. 7.13) кількість вершин $n=3+3=6$, а кількість ребер $m=3x3=9$. Припустимо, що він планарний. Тоді в ізоморфному до нього плоскому графі за теоремою Ейлера кількість граней $r = m + 2 - n = 9 + 2 - 6 = 5$. Будь-яка грань двочасткового графа має бути обмежена принаймні чотирма ребрами (за теор. 7.2 двочастковий граф має лише прості цикли з парною довжиною). Отже, $4r \leq 2m$ ($2r \leq m$), звідки випливає, що $10 \leq 9$. Знову отримали суперечність. Доведення завершено.

Означення 9.3. Граф G_1 називають **гомеоморфним** графу G, якщо його можна отримати з графа G додаванням до його ребер нових вершин степеня 2 (рис. 7.50).

Іншими словами, якщо графи гомеоморфні, то це той самий граф з точністю до вершин степеня 2.

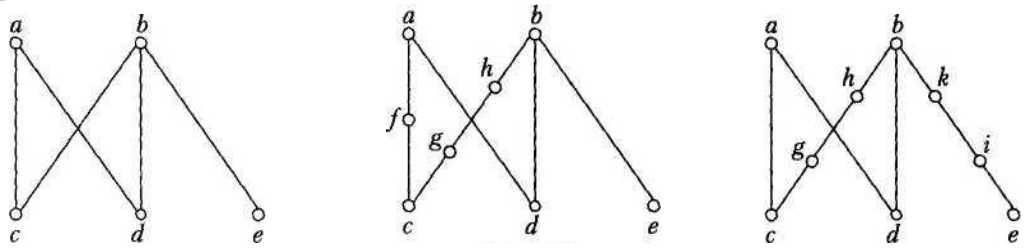


Рис. 7.50. Гомеоморфні графи

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

Теорема 7.14. (теорема Куратовського). *Граф планарний тоді й лише тоді, коли він не містить підграфів, гомеоморфних графам K_5 або $K_{3,3}$.*

Необхідність умов теореми вже доведено, бо доведено непланарність графів K_5 і $K_{3,3}$ (теорема 7.13), а доведення достатності складне, і ми його не наводимо.

На рис 7.50 зображено підграф графа Петерсона (див. рис. 7.13), гомеоморфний $K_{3,3}$. Тому за теоремою Куратовського граф Петерсона непланарний.

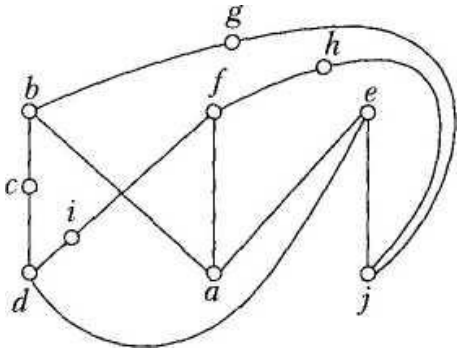


Рис. 7.51. Підграф графа Петерсена, гомеоморфний $K_{3,3}$

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).