Лекція 16

Тема 4. Графи та їхні властивості

План лекції

- **Розфарбування графів**
- **Незалежні множини вершин. Кліки**
- **Паросполучення в графах. Теорема Голла**
- **Задача про найкоротший шлях. Алгоритм Дейкстри.**
- **Задача комівояжера**

Розфарбування графів

Хроматичне число

У цьому підрозділі розглядаємо лише прості графи. *Розфарбуванням 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 в k кольорів (використовують термін *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 (незалежні множини вершин розглянемо дещо пізніше).

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 – порожній граф, $\chi(G) = 2$ тоді й лише тоді, коли G – двочастковий граф (зважаючи на це, двочастковий граф називають *біхроматичним*).

Теорема. Якщо найбільший зі степенів вершин графа G 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 і граф G, якщо розфарбувати вершину v кольором, що відрізняється від тих, якими розфарбовані суміжні з нею вершини (а їх разом не більше ніж ρ).

Проводячи міркування акуратніше, можна дещо посилити цю теорему й одержати результат, відомий як теорема Брукса; її доведення ми не наводимо.

Теорема (Брукс, 1941). Якщо G – зв'язний граф, який не є повним, і найбільший зі степенів його вершин дорівнює ρ ($\rho \geq 3$), то цей граф можна розфарбувати в ρ кольорів.

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

Перше з помилкових «доведень» належить А. Кемпе (1879 р.), але помилку було виявлено не відразу. Її знайшов Р. Гейвуд 1890 р. й тоді ж довів, що будь-який планарний граф можна розфарбувати в п'ять кольорів.

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

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

Американські математики Апель (Appel) і Гайкен (Haken) 1976 р. довели гіпотезу чотирьох фарб, великою мірою використавши комп'ютерні обчислення. Це перший випадок, коли настільки відому математичну проблему було розв'язано за допомогою комп'ютера. Спочатку проблему було зведено до розгляду скінченної (хоча й великої) кількості випадків. Надалі список „підозрілих" графів було зменшено й за допомогою комп'ютера розфарбовано в чотири кольори планарні графи з цього списку. Для цього в 1976 р. за різними джерелами було потрібно від 1500 до 2000 годин роботи потужного комп'ютера. Отже, проблему чотирьох фарб було розв'язано.

Практичні задачі, які зводяться до розфарбування графів

Розглянемо деякі практичні задачі, які зводяться до розфарбування графів.

1. 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 розфарбування цього графа задає можливий розклад: лекції, які відповідають вершинам одноколірного класу, читають одночасно. Навпаки, будь-який можливий розклад задає розфарбування графа G. Оптимальні розклади відповідають розфарбуванням мінімальною кількістю кольорів, а час, необхідний для читання всіх лекцій, дорівнює $\chi(G)$.

2. Задача розподілу обладнання. Задано множини $V=\{v_1, \ldots, v_n\}$ та $S=\{s_1, \ldots, s_m\}$ 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.

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

Незалежні множини вершин. Кліки

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

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

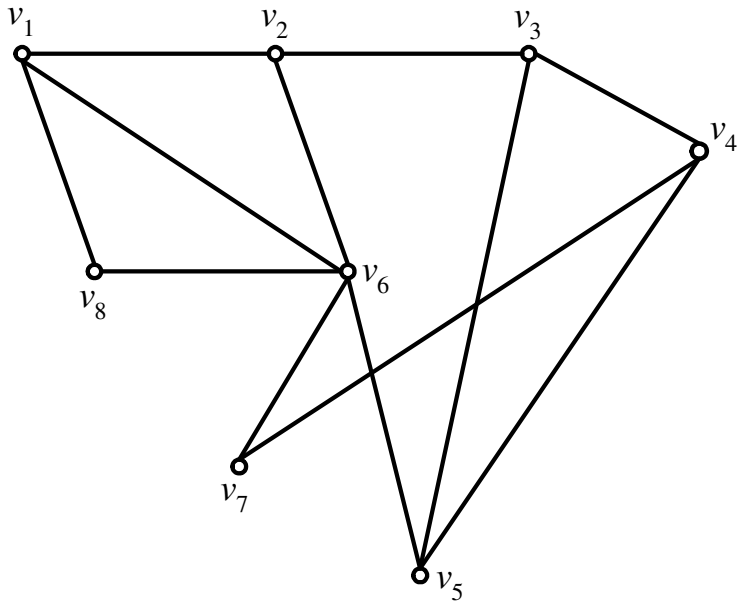


Рис. 1

Приклад. У графі, зображеному на рис. 1 розглянемо множини вершин: $Y_1=\{v_2, v_5, v_7, v_8\}$, $Y_2=\{v_2, v_7, v_8\}$, $Y_3=\{v_1, v_3, v_7\}$, $Y_4=\{v_4, v_6\}$ Максимальні незалежні множини – Y_1, Y_3 та Y_4. Множина вершин Y_2 незалежна, але не максимальна. Найбільша незалежна множина –це $Y_1=\{ v_2, v_5, v_7, v_8\}$. Отже, $\alpha(G)=4$.

Протилежне до поняття незалежної множини поняття кліки. Підмножину $V' \subset V$ вершин графа G називають *клікою*, якщо будь-які дві вершини з V' суміжні, тобто породжений підграф $G(V')$ повний. Кліку називають *максимальною*, якщо вона не є підмножиною жодної іншої кліки з більшою кількістю вершин. Найпотужнішу з максимальних клік називають *найбільшою*. Кількість вершин у найбільшій кліці графа називають *щільністю*, або *кліковим числом*, і позначають як $\varphi(G)$.

Між незалежними множинами вершин і кліками існує тісний взаємозв'язок. Для формулювання відповідної теореми нам потрібно таке поняття. *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. 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 цієї теореми випливає з означень.

З останньої теореми випливає, що побудову максимальної кліки можна звести до побудови максимальної незалежної множини вершин. Метод побудови максимальних незалежних множин розглянемо пізніше.

Паросполучення в графах. Теорема Голла

Теорема, яку сформулював Ф. Голл, має багато різних застосувань, два з яких ми розглянемо.

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

Таблиця знайомств

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. Досконале паросполучення в двочастковому графі. Нехай $G=(V_1 \cup V_2, E)$ – двочастковий граф ($V_1 \cap V_2 = \varnothing$ і кінці кожного ребра належать різним множинам V_1 та V_2). *Досконалим паросполученням* із V_1 у V_2 називають паросполучення, яке *покриває* вершини V_1 (це означає, що всі вершини з V_1 інцидентні ребрам, які утворюють паросполучення). За яких умов існує досконале паросполучення з V_1 у V_2?

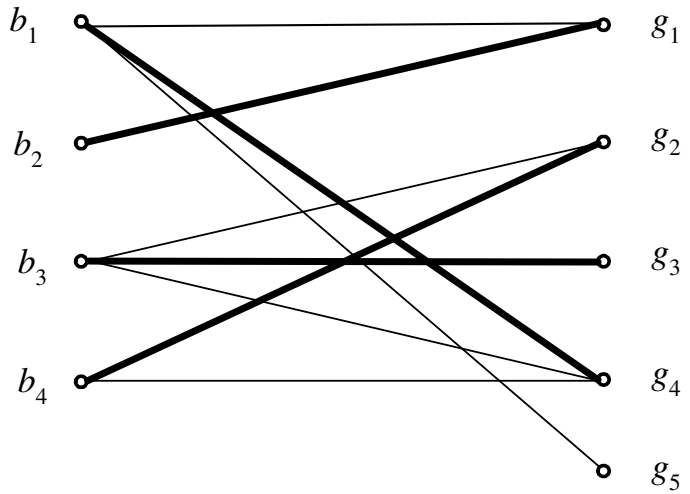


Рис. 2

Можна довести, що задачі 1 та 2 – це по суті, одна й та сама задача. Справді, нехай V_1 – множина юнаків, V_2 – множина дівчат, ребра – знайомства юнаків із дівчатами (рис. 2). У такому разі досконале паросполучення в двочастковому графі – це шукана множина весіль (один із можливих розв'язків на рисунку зображено потовщеними лініями).

Використовуючи термінологію задачі про весілля, можна сформулювати таке очевидне твердження: необхідна умова в задачі про весілля полягає в тому, що будь-які k юнаків із даної множини знайомі (у сукупності) принаймні з k дівчатами (для всіх k=1, 2, …, m, де як m позначено загальну кількість юнаків). Необхідність цієї умови одразу випливає з того, що якщо вона не виконується для якоїсь множини з k юнаків, то ми не зможемо одружити потрібним способом навіть цих k юнаків, не говорячи вже про решту. Цікаво, що ця очевидна необхідна умова виявляється водночас і достатньою. У цьому й полягає теорема Голла про весілля.

Теорема (Голл, 1935 р.). Розв'язок задачі про весілля існує тоді й лише тоді, коли будь-які k юнаків із даної множини знайомі в сукупності принаймні з k дівчатами (для всіх k=1, 2, …, m, де як m позначено загальну кількість юнаків).

Нехай $G=(V, E)$ – простий граф, $V=\{v_1,\ldots,v_n\}$. Через $\Gamma(x)$, $x\in V$, позначають *оточення* вершини x, тобто множину вершин, суміжних з вершиною x. Нехай $A=\{x_1,x_2,\ldots,x_k\}$ і $A\subset V$. Тоді $\Gamma(A)$ означують так: $\Gamma(A)=\Gamma(x_1)\cup\Gamma(x_2)\cup\ldots\cup\Gamma(x_k)$.

У цих термінах теорему Голла можна сформулювати так.

Теорема. Нехай $G = (V_1 \cup V_2, E)$ – двочастковий граф. Досконале паросполучення з V_1 у V_2 існує тоді й лише тоді, коли для будь-якої множини $A \subset V_1$ виконується умова $|A| \le |\Gamma(A)|$.

Різні формулювання задач про найкоротший шлях.

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 знаходять у процесі розв'язування 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.

Алгоритм Дейкстри

Найефективнішим алгоритмом визначення довжини найкоротшого шляху від фіксованої вершини до будь-якої іншої є алгоритм, запропонований 1959 р. датським математиком Е. Дейкстрою (E. Dijkstra). Цей алгоритм застосовний лише у випадку, 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 суміжної з v^* вершини вздовж шляху, який проходить через вершину v^*. Якщо для якоїсь із таких вершин ця віддаль менша від поточної, то заміняємо нею поточну віддаль. Знову вибираємо вершину, найближчу до a, але таку, що не вибиралась 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 опишемо алгоритм Дейкстри формально.

Опис алгоритму Дейкстри

Присвоювання початкових значень

Крок 1. Виконати $l(a):=0$ і вважати цю мітку постійною. Виконати $l(v):=\infty$ для всіх $v \neq a$ і вважати ці мітки тимчасовими. Виконати $x:=a$, $M:=\{a\}$.

Оновлення міток

Крок 2. 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.

Перетворення мітки в постійну

Крок 3. 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).

Закінчення

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.

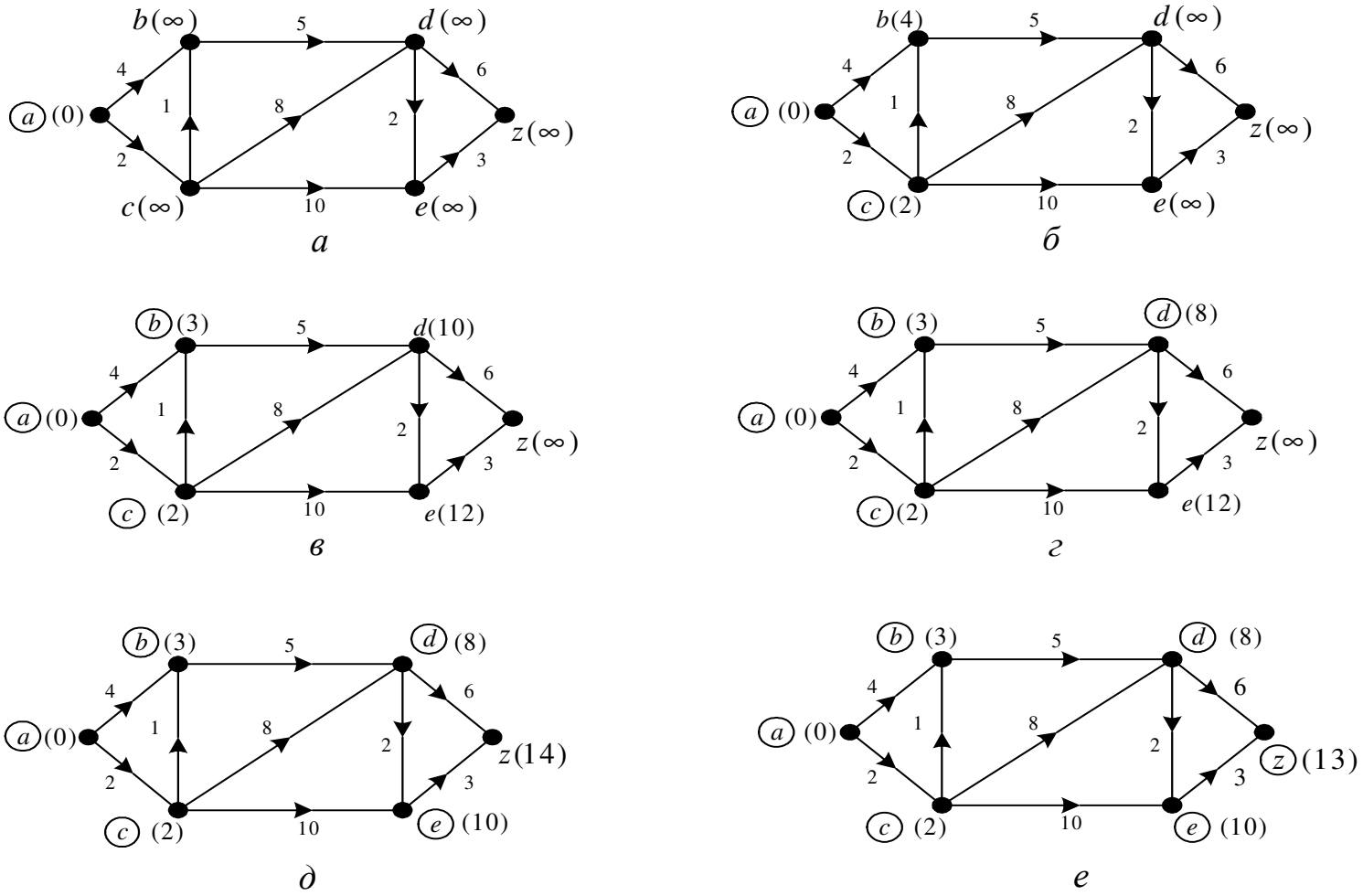


Рис. 3

Табл. 1

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

Якщо граф подано матрицею суміжності, складність алгоритму Дейкстри становить $O(n^2)$.

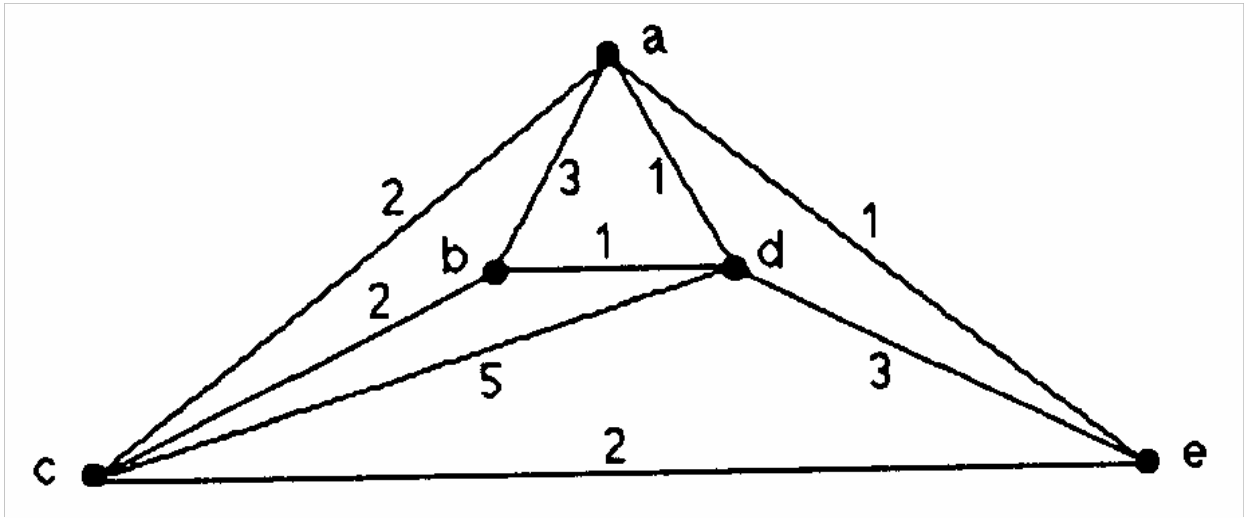
Алгоритм Дейкстри дає змогу обчислити довжину найкоротшого шляху від початкової вершини a до заданої вершини z. Для знаходження самого шляху потрібно лише поступово будувати вектор вершин, з яких найкоротший шлях безпосередньо потрапляє в дану вершину. Для цього з кожною вершиною v графа G, окрім вершини a, зв'язують іще одну мітку – $\theta(v)$. Крок 2 модифікують так. Для кожної вершини $v \in \Gamma(x) \setminus M$ виконати: 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, то такою стане й мітка $\theta(v)$, і найкоротший $\langle a, v \rangle$-шлях буде потрапляти у вершину v безпосередньо з вершини x. З постійних міток $l(v)$ та $\theta(v)$ утворюємо вектори l та θ.

Приклад. Знайдемо довжину найкоротшого шляху від початкової вершини a до вершини z у графі на рис. 3, *а*. Послідовність дій зображено на рис. 3, *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. 1 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 табл. 1). У вершину e потрапляємо з вершини d, у вершину d потрапляємо 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.

Зауваження. Ми розглянули задачу знаходження на графі найкоротшого шляху від якоїсь виділеної (початкової) вершини до будь-якої іншої. Розглянемо задачу пошуку на графі найкоротших шляхів 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'язування цієї задачі – *алгоритм Флойда*. У цьому алгоритмі для довжин дуг допускаються від'ємні значення, проте не допускаються цикли від'ємної довжини. Цей алгоритм докладно розглянуто в підручнику.

Задача комівояжера

Зараз ми обговоримо важливу проблему, яка стосується навантажених графів. Комівояжер (traveling salesman) має відвідати кожне з n міст точно один раз і повернутись у початкове місто. Цю проблему називають задачею комівояжера (Traveling salesman problem). Для її розв'язання потрібно в простому зваженому графі знайти гамільтонів цикл найменшої довжини, бо кожну вершину потрібно відвідати точно один раз. Найпростіший спосіб розв'язання цієї задачі полягає в розгляді всіх гамільтонових циклів і виборі того, який має найменшу довжину. Скільки циклів нам потрібно розглянути, щоб розв'язати задачу комівояжера для графа з n вершинами? Якщо початкову вершину вибрано, то в повному графі існує $(n-1)!$ різних гамільтонових циклів, бо є $n-1$ можливостей для вибору другої вершини, $n-2$ можливостей для вибору третьої вершини і т. д. Оскільки гамільтонів цикл можна пройти у зворотному напрямку, то достатньо розглянути $(n-1)!/2$ циклів для знаходження розв'язку задачі комівояжера.



Permutation	Length	Permutation	Length
a, b, c, d, e, a	14	a, d, e, c, b, a	11
a, b, d, c, e, a	12	a, b, d, e, c, a	11
a, d, c, e, b, a	does not exist	a, b, e, d, c, a	does not exist
a, c, b, d, e, a	9	a, c, b, e, d, a	does not exist
a, c, d, b, e, a	does not exist	a, c, e, b, d, a	does not exist
a, d, b, c, e, a	7	a, d, c, b, e, a	does not exist

Зазначимо, що функція $(n-1)!/2$ зростає дуже швидко. Якщо ми захочемо розв'язати задачу комівояжера для повного графа з 25 вершинами, то потрібно розглянути всього $24!/2 \approx 3.1\times10^{23}$ різних гамільтонових циклів. Якщо припустити, що для розгляду одного такого циклу потрібно одну наносекунду (10^{-9} секунди), то загалом буде потрібно приблизно десять мільйонів років для знаходження гамільтонового циклу найменшої довжини (за умови використання описаного методу).

Оскільки задача комівояжера є надзвичайно важливою як теоретично, так і практично, то великі сили були спрямовані для знаходження ефективного алгоритму для її розв'язання. Проте так і не було знайдено поліноміального алгоритму для розв'язання задачі комівояжера. Більше того, якщо б такий алгоритм удалося б знайти, багато інших важкорозв'язних задач також можна було б розв'язати поліноміальними алгоритмами. Цей факт випливає з теорії *NP*-повноти.

Практично застосовними алгоритмами для розв'язування задачі комівояжера для графа з багатьма вершинами є наближені алгоритми. Вони не гарантують знаходження точного розв'язку, але дають змогу за прийнятний час знайти наближений розв'язок. Це означає що ці алгоритми можуть знайти гамільтонів цикл довжиною W' такою, що $W \le W' \le cW$, де W – точний розв'язок, c – константа. Наприклад, є поліноміальний алгоритм, який працює, коли зважений граф задовольняє нерівність трикутника, і для цього алгоритму $c = 3/2$. На практиці нині розроблені наближені алгоритми, які дають змогу розв'язати задачу комівояжера для графа з 1000 вершинами з 2% відхиленням від точного розв'язку всього за декілька хвилин комп'ютерного часу.