Львівський державний університет безпеки життєдіяльності ДСНС України

Кафедра прикладної математики і механіки

ТЕМА 2. ЕЛЕМЕНТИ ТЕОРІЇ ГРАФІВ.
ДЛЯ ПРОВЕДЕННЯ ЛЕКЦІЙНОГО ЗАНЯТТЯ
З КУРСАНТАМИ ТА СТУДЕНТАМИ 2 КУРСУ
З ДИСЦИПЛІНИ СИСТЕМНИЙ АНАЛІЗ ТА ТЕОРІЯ ПРИЙНЯТТЯ РІШЕНЬ

Мета лекції:

Навчальна: вивчити основні поняття теорії графів: орієнтований та неорієнтований граф, матриця суміжності, шлях, цикл, Ейлеровий шлях (цикл), екстремальні шляхи на графах, алгоритм Дейкстри, дерева, алгоритм Краскала.

Виховна: виховання свідомого ставлення до вивчення предмету, самостійності, відповідальності та організованості при підготовці до занять.

Розвиткова: розвиток логічного та абстрактного мислення, розвиток просторової уяви.

План

Література

1. *Аршинова О.І., Шевченко А.В.* Системний аналіз. Навч. посібник. – К.: НАУ, 2008. – 128 с.
2. *Роїк О.М., Шиян А.А., Нікіфорова Л.О.* Системний аналіз. Навч. посібник. – Вінниця: ВНТУ, 2015. – 83 с.
3. *Кузик А.Д.* Основи системного аналізу. – Львів: ЛДУ БЖД, 2005. – 100 с.
4. *Кунда Н.Т.* Дослідження операцій у транспортних системах. – К.: Видавничий Дім "Слово", 2008. – 400 с.
5. *Карагодова О.О., Кігель В.Р., Рожок В.Д.* Дослідження операцій. – К.: Центр учбової літератури, 2007.– 256 с.

Час проведення: 2 години.
Місце проведення: лекційний зал.
Забезпечення заняття: мультимедіа.

1. Основні поняття.

Теорія графів, як теоретична дисципліна, є розділом дискретної математики. Як прикладна дисципліна, теорія графів дозволяє описувати й досліджувати багато технічних, економічних, біологічних і соціальних систем. Спочатку теорія графів здавалась досить незначним розділом математики, тому, що вона мала справу в основному з математичними розвагами і головоломками.

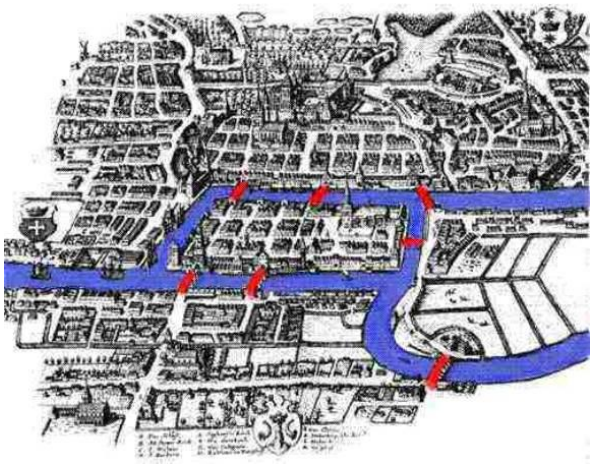


Рис. 2.1.

Початок теорії графів датують 1736 роком, коли Л. Ейлер розв'язав популярну в той час "задачу про кенігсбергські мости" (рис. 2.1.). А саме: до XVIII століття через ріку, на якій стояло місто Кенігсберг (нині Калінінград), було побудовано 7 мостів, які сполучали з берегами й один з одним два острови, розташовані в межах міста. Завдання полягає в наступному: потрібно пройти (якщо це можливо) по всіх мостах так, щоб на кожному з них побувати лише по одному разу й повернутися до того місця, звідки почав маршрут. Ейлер довів, що це неможливо. Відповідну теорему, яка є наслідком даної головоломки ми розглянемо пізніше.

Термін "граф" уперше був введений через 200 років (в 1936 р.) Д. Кенигом. Подальший розвиток математики дав сильний поштовх до розвитку теорії графів. Вже в XIX столітті графи використовувалися при побудові схем електричних ланцюгів і молекулярних схем. З іншого боку, ця теорія широко застосовується в різноманітних практичних питаннях: при встановленні різного виду відповідностей, при вирішенні транспортних задач, задач про потоки в мережі нафтопроводів тощо. Теорія графів тепер застосовується і в таких областях, як економіка, психологія і біологія. Дамо визначення основних понять теорії графів.

Граф – це сукупність об'єктів та зв'язків між ними. Геометрично граф – це фігура, що складається з точок (вершин) і ліній (ребер), що сполучають деякі з цих вершин. Ребра можуть бути неорієнтованими та орієнтованими (дуги).

Якщо дві вершини з'єднує кілька ребер (дуг), то такі ребра (дуги) називають **кратними**. Ребро (дугу), що з'єднує вершину саму с собою називають **петлею**. Це геометричний спосіб задання графа (рис.2.2.).

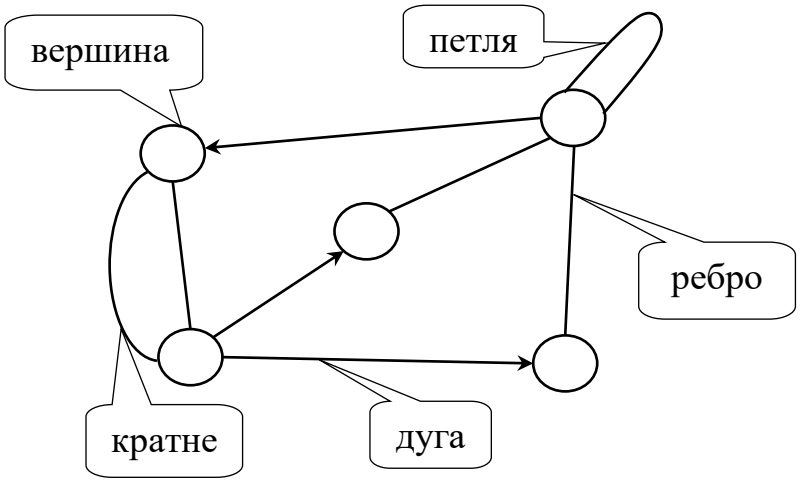


Рис. 2.2.

Граф, у якому заданий напрямок ребер (всі ребра є дугами) називають **орієнтованим**, інакше – граф **неорієнтований**.

В подальшому будемо вважати, що граф не має петель та кратних ребер.

Мова графів є зручною для опису багатьох фізичних, технічних, економічних, біологічних, соціальних та інших систем. Приведемо ряд прикладів застосування теорії графів.

Транспортні задачі, у яких вершинами графа є пункти, а ребрами – дороги (автомобільні, залізні й ін.) і/або інші транспортні (наприклад, авіаційні) маршрути. Інший приклад – мережі постачання (енергопостачання, газопостачання, постачання товарами й т. д.), у яких вершинами є пункти виробництва й споживання, а ребрами – можливі маршрути переміщення (лінії електропередач, газопроводи, дороги й т. д.). Підкласом є задачі про вантажоперевезення.

Технологічні задачі, у яких вершини відображають виробничі елементи (заводи, цехи, верстати й т. д.), а дуги – потоки сировини, матеріалів і продукції, які між ними пролягають. Такі задачі полягають у визначенні оптимального завантаження виробничих елементів та визначенні потоків, що забезпечують оптимальне завантаження.

Обмінні схеми, які є моделями таких явищ як бартер, взаємозаліки й т. д. Вершини графа при цьому описують учасників обмінної схеми (ланцюжка), а дуги – потоки матеріальних і фінансових ресурсів між ними. Задача полягає у визначенні ланцюга обмінів, оптимальних з погляду, наприклад, організатора обміну й погодженої з інтересами учасників ланцюга й існуючих обмежень.

Управління проектами. З погляду теорії графів проект – сукупність операцій і залежностей між ними (сітьовий графік). Хрестоматійним прикладом є проект будівництва деякого об’єкта. Сукупність моделей і методів, що використовують мову й результати теорії графів й орієнтованих на рішення завдань керування проектами, одержала назву календарно-сітьового планування й управління (КСПУ). У рамках КСПУ вирішуються завдання визначення послідовності виконання операцій і розподілу ресурсів між ними, оптимальних з погляду тих або інших критеріїв (часу виконання проекту, витрат, ризику й ін.).

Повернемося до введення основних понять.

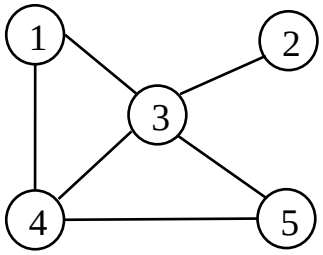
Дві вершини називають **суміжними**, якщо вони з'єднані ребром (дугою). Суміжні вершини називають **граничними вершинами** відповідного ребра (дуги), а це ребро (дуга) – **інцидентним** відповідним вершинам.

У неорієнтованому графі **степеню вершини** називають кількість d_i (i – номер вершини) інцидентних їй ребер. Граф степені всіх вершин якого рівні $n-1$ (n – кількість вершин), називають **повним**.

Вершину, для якої не існує інцидентних їй ребер ($d_i = 0$) називають **ізольованою**. Вершину, для якої існує тільки одне інцидентне їй ребро ($d_i = 1$) називають **висячою.** Наприклад, 2 – висяча вершина (рис. 2.3 *а)*).

Аналітично, графи можна задавати ***матрицею суміжності,*** елементи якої a_{ij} дорівнюють одиниці, якщо вершина i суміжна вершині j, та дорівнюють нулю, якщо вершина i несуміжна вершині j.

Наприклад, графу на рис.2.3. *а)* відповідає матриця суміжності на рис. 2.3. б*)*.



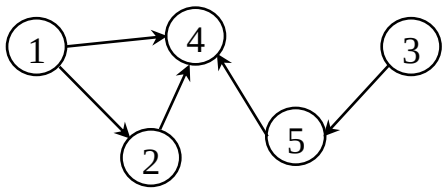
$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

а) *б*)

Рис. 2.3.

Зауважимо, що для неорієнтовного графа матриця суміжності завжди буде симетричною. Якщо в графі нема петель, то діагональні елементи матриці суміжності дорівнюють нулю, тобто $a_{ii} = 0$.

У випадку орієнтованого графа, матриця суміжності буде несиметричною (рис.2.4.).



$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Рис. 2.4.

Твердження. Сума степенів вершин графа число парне і рівне подвоєному числу ребер графа

$$\sum_{i=1}^{n} d_i = 2m,$$

де m – кількість ребер.

Наслідок. Кількість вершин непарного степеня число парне.

Приклад. Чи існує граф з вершинами таких степенів? Якщо так, то скільки ребер він має?

а) 3 3 3 3 2; б) 3 4 3 4 3.

Розв'язання.

а) Існує, оскільки кількість вершин непарного степеня число парне. Сума степенів вершин дорівнює $\sum_{i=1}^{n} d_i = 2m = 14$. А отже кількість ребер $m = 7$.

б) Не існує, оскільки кількість вершин непарного степеня число непарне.

Шляхом називають послідовність ребер (дуг) графа, у якій кінцева вершина одного ребра (дуги) є початком іншого ребра (дуги).

Якщо будь-які дві вершини графа можна з'єднати шляхом, то граф називають **зв'язним.**

Замкнутий шлях називають **циклом**.

Простим шляхом (циклом) називають шлях (цикл), в якому ребра (дуги) не повторюються.

Наведені означення не є загальноприйнятими. Назви описаних об'єктів можуть бути іншими.

Ейлеровим шляхом (циклом) називають простий шлях (цикл) графа, що містить всі його ребра. Граф, що має Ейлерів цикл називають **Ейлеревим графом.**

Іншими словами, якщо граф володіє ейлеревим циклом, то його можна накреслити не відриваючи олівець від паперу, проводячи по кожному ребру тільки один раз. Рух можна почати з будь-якої вершини й закінчити в тій самій вершині.

Теорема Ейлера. Зв'язний граф є Ейлеревим тоді й тільки тоді, коли степені всіх його вершин парні.

Теорема. Неорієнтований граф має Ейлеревий шлях тоді й тільки тоді, коли він зв'язний і число вершин непарного степеня дорівнює 0 або 2.

Якщо граф має дві вершини непарного степеня, то його можна накреслити не відриваючи олівець від паперу, при цьому рух треба починати з однієї з цих вершин непарного степеня і закінчити в другій з них.

Граф, який відповідає задачі про кенігсбергські мости наведено на рис. 2.5. Оскільки він має чотири вершини непарного степеня, то він не є Ейлеровим, тобто не можливо пройти по всіх мостах (рис.2.1.) так, щоб на кожному з них побувати лише по одному разу й повернутися до того місця, звідки почато маршрут.

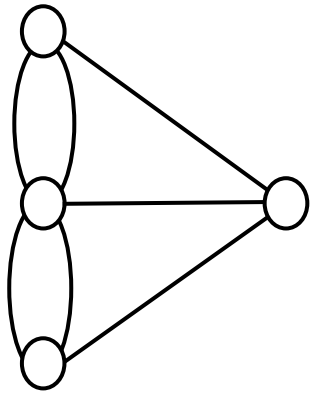


Рис.2.5.

Одним із алгоритмів знаходження Ейлеревого циклу у графі є алгоритм Фльорі, який ґрунтується на двох правилах:

1. Починаємо з довільної вершини, проходимо по ребрах графа довільним чином. При проходженні кожного ребра йому присвоюється черговий номер в циклі і це ребро викреслюється, тобто вилучається з графа (вершини при цьому лишаються);

2. Якщо є декілька можливостей продовжити рух по ребрах, то рухаємось не по ребру, при вилученні якого граф перестає бути зв'язним.

2. Екстремальні шляхи на графах.

Задача пошуку найкоротших і найдовших шляхів на графах виникає у різних областях керування. Розглянемо задачу про найкоротший шлях.

2.1. Задача про найкоротший шлях між двома парами вершин. Алгоритм Дейкстри.

Нехай заданий орієнтований граф з $n+1$ вершинами, у якому виділені дві вершини – вхід (нульова вершина) і вихід (вершина з номером n). Такі графи називають **сітьовими графіками або мережами**. Нехай для кожної дуги задане число l_{ij}, яке будемо називати **довжиною або вагою дуги**. Граф, для якого задано довжини дуг називають **зваженим**.

Довжиною шляху називають суму довжин вхідних у нього дуг. Якщо довжини дуг не задані, то довжина шляху визначається як число вхідних у нього дуг. Завдання полягає в пошуку найкоротшого шляху (шляху мінімальної довжини) від початкової до кінцевої вершини.

Припустимо, що в сітьовому графіку можна пронумерувати вершини таким чином, що для будь-якої дуги (i, j) має місце $j > i$. Така нумерація називається **правильною**.

Найкоротший шлях у сітьовому графіку, що має правильну нумерацію, визначається наступним **алгоритмом Дейкстри**.

Алгоритм Дейкстри.

Крок 0*:* Позначаємо нульову вершину індексом $\lambda_0 = 0$;

Крок k: позначаємо вершину *k* індексом $\lambda_k = \min_{i<k}(\lambda_i + l_{ik})$.

Індекс виходу λ_n буде дорівнювати довжині найкоротшого шляху.

Коли індекси установляться, найкоротший шлях визначається методом зворотного ходу від виходу до входу.

Описаний алгоритм дозволяє знайти найкоротший шлях від нульової вершини до всіх інших. Цей алгоритм можна застосовувати лише для дуг невід'ємної довжини.

На рис.2.6. наведений приклад застосування алгоритму Дейкстри для визначення найкоротшого шляху (числа біля дуг дорівнюють довжинам дуг, індекси вершин поміщені у квадратні дужки, найкоротший шлях виділений жирними лініями).

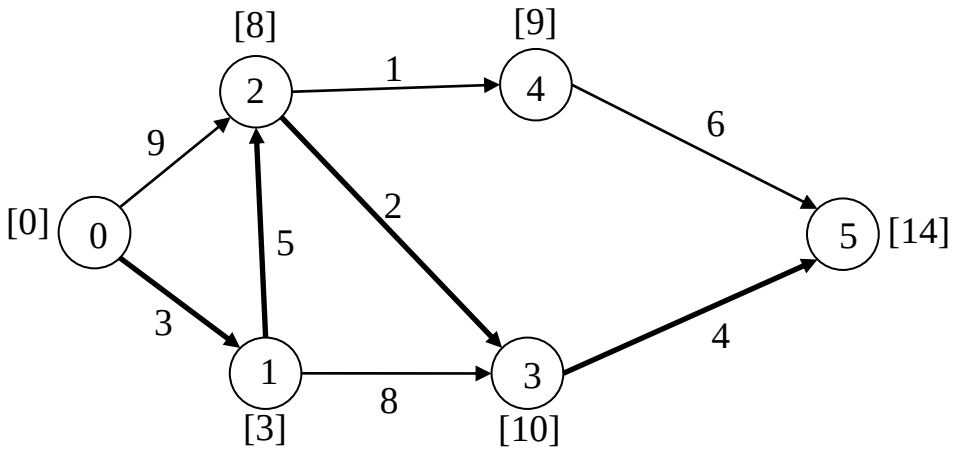


Рис. 2.6.

2.2. Знаходження найкоротших шляхів між всіма парами вершин. Алгоритм Флойда

Для знаходження найкоротших шляхів між всіма парами вершин використовують алгоритм Флойда. Умовою застосування є відсутність в графі циклів з сумарною від'ємною вагою. Але алгоритм дозволяє виявити такі контури.

Розглянемо орієнтований граф $G = (V, A)$, де V – вершини графа, A – ребра графа. Нехай $W^0 = \| w_{ij} \|$ – матриця довжин дуг цього графа. Покладемо $w_{ij} = \infty$, якщо дуга (i, j) відсутня і $w_{ii} = 0$ для всіх $i \in V$ (вважаємо, що петель немає). Алгоритм Флойда будує послідовність матриць W^0, W^1, …, W^n таку, що елемент w_{ij}^n матриці W^n дорівнює довжині найкоротшого шляху від вершини *i* до вершини *j* в графі *G*. Матриця W^k визначається за матрицею W^{k-1} наступним чином:

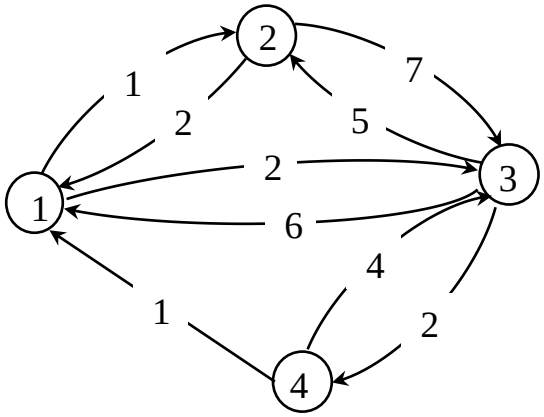
$$w_{ij}^k = \min\{w_{ij}^{k-1}, w_{ik}^{k-1} + w_{kj}^{k-1}\}.$$

В алгоритмі Флойда одночасно з довжинами найкоротших шляхів шукаємо самі шляхи за допомогою матриць Z^0, Z^1, …, Z^n, де елемент z_{ij}^k матриці Z^k вказує на вершину, яка передує на шляху від вершини *i* до *j* передує вершині *j*. Спочатку для всіх *i* та *j* покладаємо $z_{ij}^0 = i$, якщо існує дуга (i, j) (тобто відповідний елемент матриці W^0 не дорівнює ∞) та $z_{ij}^0 = 0$ в іншому випадку. Далі на кожній ітерації матриця Z^k перераховується разом з матрицею W^k за наступним правилом:

$$z_{ij}^k = \begin{cases} z_{kj}^{k-1}, & \textit{якщо } w_{ij}^{k-1} > w_{ik}^{k-1} + w_{kj}^{k-1}; \\ z_{ij}^{k-1}, & \textit{якщо } w_{ij}^{k-1} \le w_{ik}^{k-1} + w_{kj}^{k-1}, \end{cases}$$

тобто, якщо на *k*-ій ітерації елемент w_{ij}^k міняється, то елемент z_{ij}^k отримує значення, що дорівнює елементу, що знаходиться в тому ж стовпці у *k*-му рядку. Розглянемо алгоритм Флойда на наступному прикладі.

Приклад 2.1. Задано орієнтований граф



Знайти найкоротший шлях між всіма парами вершин та їх довжини.

Розв'язання.

Побудуємо матрицю довжин дуг

$$W^0 = \begin{vmatrix} \boxed{0} & 1 & 2 & \infty \\ 2 & 0 & \underline{7} & \infty \\ 6 & \underline{5} & 0 & 2 \\ 1 & \infty & 4 & 0 \end{vmatrix}.$$

Крок 1. Виділяємо перший стовпець та перший рядок матриці W^0. Виділені елементи перераховувати не треба, оскільки 4-ий елемент виділеного рядка дорівнює ∞, то відповідний стовпчик (4-ий) ми не перераховуємо. Аналогічне твердження вірне й для виділеного стовпця, але на цій ітерації виділений стовпець немає елементів, що дорівнюють ∞. Оскільки довжини всіх дуг невід'ємні, то діагональні елементи матриці сталі й дорівнюють нулю. Таким чином, потрібно перерахувати лише елементи w_{23}^0, w_{32}^0, w_{42}^0, w_{43}^0 (підкресленні в таблиці). Отримаємо

$w_{23}^1 = \min(7; 2+2) = 4;$

$w_{32}^1 = \min(5; 6+1) = 5;$

$w_{42}^1 = \min(\infty; 1+1) = 2;$

$w_{43}^1 = \min(4; 1+2) = 3.$

Отримаємо таку матрицю довжин

$$W^1 = \begin{vmatrix} 0 & \boxed{1} & 2 & \infty \\ \boxed{2} & \boxed{0} & 4 & \infty \\ \underline{6} & 5 & 0 & 2 \\ \underline{1} & \boxed{2} & \underline{3} & 0 \end{vmatrix}.$$

Матриця W^0 відрізняється від матриці W^1 елементами w_{23}^0, w_{42}^0, w_{43}^0.

Крок 2. В матриці W^1 виділяємо другий рядочок та другий стовпчик. Аналогічно до попереднього кроку отримаємо наступну послідовність матриць

$$W^2=\left\|\begin{array}{cccc}0 & \underline{1} & 2 & \boxed{\infty}\\ \underline{2} & 0 & 4 & \boxed{\infty}\\ \boxed{6} & \boxed{5} & \boxed{0} & \boxed{2}\\ \underline{1} & \underline{2} & 3 & \boxed{0}\end{array}\right\|;\ W^3=\left\|\begin{array}{cccc}0 & \underline{1} & \underline{2} & \boxed{4}\\ 2 & 0 & \underline{4} & \boxed{6}\\ \underline{6} & \underline{5} & 0 & \boxed{2}\\ \boxed{1} & \boxed{2} & \boxed{3} & \boxed{0}\end{array}\right\|;\ W^4=\left|\begin{array}{cccc}0 & 1 & 2 & 4\\ 2 & 0 & 4 & 6\\ 3 & 4 & 0 & 2\\ 1 & 2 & 3 & 0\end{array}\right|.$$

Таким чином, отримали найкоротші відстані між будь-якою парою вершин. Наприклад, найменша відстань між вершинами 3 та 1 дорівнює $w_{31}^4=3$.

Знайдемо відповідні шляхи. Матриця Z^0 матиме вигляд

$$Z^0=\left|\begin{array}{cccc}0 & 1 & 1 & 0\\ 2 & 0 & 2 & 0\\ 3 & 3 & 0 & 3\\ 4 & 0 & 4 & 0\end{array}\right|.$$

На першій ітерації (крок 1) міняються елементи w_{23}^0, w_{42}^0, w_{43}^0 (виділені зеленим кольором в матриці W^0). Відповідні елементи матриці Z^1 будуть дорівнювати елементам першого рядка матриці Z^0, що знаходяться в тому ж стовпці, тобто

$$z_{23}^1=z_{13}^0=1;$$
$$z_{42}^1=z_{12}^0=1;$$
$$z_{43}^1=z_{13}^0=1,$$

$$Z^1=\left|\begin{array}{cccc}0 & 1 & 1 & 0\\ 2 & 0 & 1 & 0\\ 3 & 3 & 0 & 3\\ 4 & 1 & 1 & 0\end{array}\right|.$$

На другому кроці матриця $W^2=W^1$ не помінялась, а отже, не міняється й матриця $Z^1=Z^2$.

Далі

$$Z^3=\left|\begin{array}{cccc}0 & 1 & 1 & 3\\ 2 & 0 & 1 & 3\\ 3 & 3 & 0 & 3\\ 4 & 1 & 1 & 0\end{array}\right|,\qquad Z^4=\left|\begin{array}{cccc}0 & 1 & 1 & 3\\ 2 & 0 & 1 & 3\\ 4 & 1 & 0 & 3\\ 4 & 1 & 1 & 0\end{array}\right|.$$

Остання матриця Z^4 дозволяє знайти найкоротший шлях для будь-якої пари вершин, наприклад, з 3 в 2. Елемент $z_{32}^4=1$, тобто вершині 2 передує вершина 1. В свою чергу їй передує вершина 4, бо $z_{31}^4=4$, вершині 4 передує 3, бо $z_{34}^4=3$, тобто початкова вершина. Таким чином, маємо найкоротший шлях між вершинами 3 та 2: $3\to4\to1\to2$, його довжина $w_{32}^4=4$.

3. Дерева.

Зв'язний граф, який не має циклів називають **деревом**. Зв'язний граф *T*, що містить всі вершини графа *G* і не містить циклів називають **кістяковим деревом графа *G***.

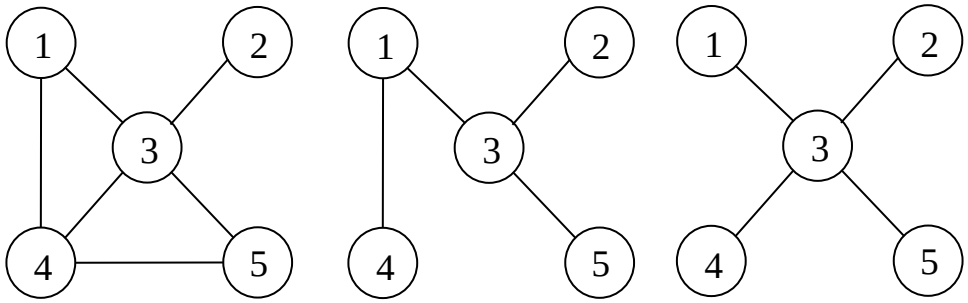


Рис. 2.7.

На рисунку 2.7. зображено граф та два його кістякових дерева.

Однією з важливих задач теорії графів є пошук кістякового дерева мінімальної довжини для зваженого графа. Така задача має широке практичне застосування при проектуванні доріг, електричних мереж, трубопроводів, тобто в ситуаціях, де необхідно з'єднати задану множину об'єктів комунікаційними лініями так, щоб сумарна їх вартість (або довжина) була мінімальною. Одним з алгоритмів знаходження мінімального кістякового дерева є **алгоритм Краскала.**

Алгоритм Краскала. Нехай задано зважений неорієнтований граф *G* з *n* вершинами. Позначимо *V* множину всіх вершин графа *G*, *E* – множину ребер мінімального кістякового дерева *T* графа *G*. За означенням, множина вершин дерева *T* дорівнює множині вершин графа *G*. Множина ребер E_T дерева *T* формується поступово за наступним алгоритмом.

Крок 0. Покладаємо $E_{T,0} = \varnothing$.

Крок *i*. ($i = \overline{1, n-2}$). Покладаємо $E_{T,i+1} = E_{T,i} \cup e$, де e – ребро графа *G*, яке має мінімальну вагу, не належить $E_{T,i}$ та не утворює циклу з ребрами множини $E_{T,i}$.

Приклад 2.2. Телевізійна компанія планує підключення до своєї кабельної мережі п'ять нових районів (рис.2.8.). На графі показана структура мережі, що планується й відстані (в км) між районами (вершини 2,3,4,5,6) й телецентром (вершина 1). Необхідно спланувати найбільш економну кабельну мережу.

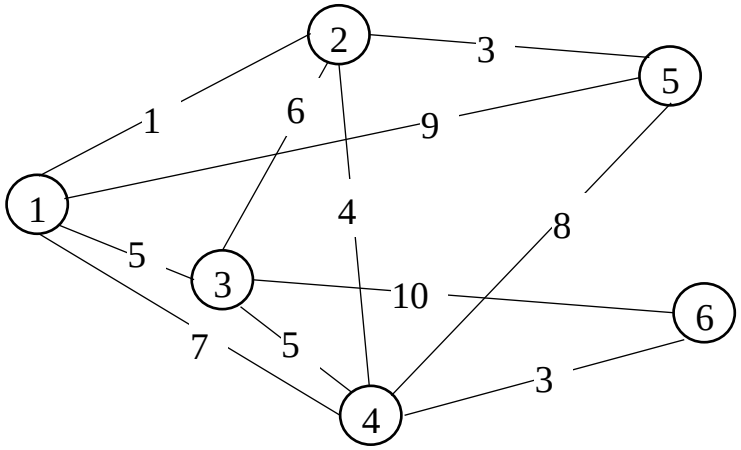


Рис. 2.8.

Розв’язання.

Побудову мінімального кістякового дерева починаємо з відбору всіх вершин графа, заданого на рис. 2.8. та вибору ребра з найменшою довжиною – ребро (1, 2). Наступні ребра, які увійду до шуканого дерева (2,5) та (4,6), жодне з них не утворює циклу з побудованими ребрами кістякового дерева. Далі (2,4), (1,3) або (3,4). Мінімальне кістякове дерево зображено на рис.2.9.

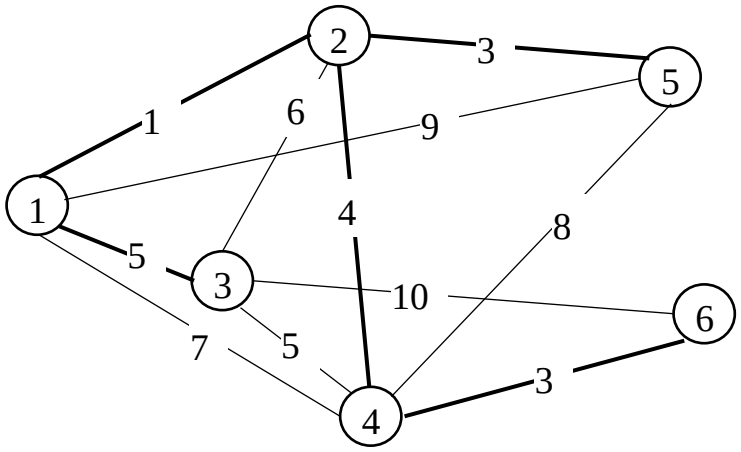


Рис. 2.9.

4. Приклади задач.

Задача про заміну обладнання. Компанія з прокату автомобілів розробляє план заміни парку транспортних засобів на наступні 5 років (2017-2021 рр.). Кожен автомобіль повинен відпрацювати не менше одного і не більше трьох років. В табл. 2.1. наведено вартість заміни автомобіля залежно від року купівлі на тривалості експлуатації (включає вартість нового автомобіля, експлуатаційні витрати та доходи від продажу старого автомобіля). Знайти план заміни транспортних засобів та його вартість.

Таблиця 2.1.

Рік купівлі	Вартість заміни залежно від тривалості експлуатації, тис. грн.		
	1	2	3
2017	40	54	98
2018	43	62	87
2019	48	71	–
2020	49	–	–

Задачу можна зобразити у вигляді графа (мережі) з п’ятьма вершинами, що відповідають 2017-2021 рокам (рис.2.10.). З вершини 1 (2017 рік) дуги йдуть лише до вершин 2, 3 та 4, оскільки автомобіль може експлуатуватись не менше одного і не більше трьох років. Дуги з інших вершин інтерпретуються аналогічно. Довжини дуг є вартістю заміни автомобілів. Розв’язування задачі є еквівалентним пошуку найкоротшого шляху між вершинами 1 та 5.

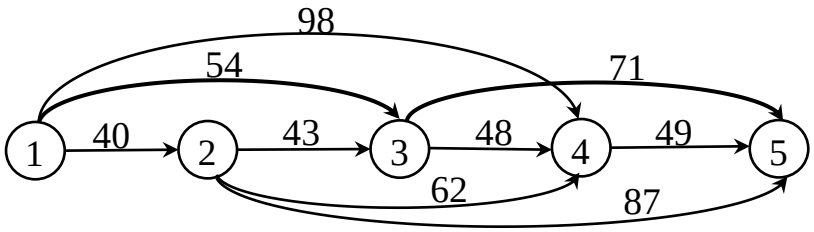


Рис. 2.10.

Найкоротшим шляхом є шлях 1−3−5. Цей розв'язок означає, що автомобілі, придбані в 2017 році (вершина 1), будуть експлуатуватись 2 роки до 2019 року (вершина 3), потім вони замінюються новими, які будуть експлуатуватись до кінця 2021 року. Загальна вартість заміни одного автомобіля становить: 54+71=125 тис. грн.

Найбільш надійний маршрут. На рис. 2.11. наведено схему мережі вулиць, якими певний водій щоденно їздить автомобілем з дому на роботу. Дана ділянка вулично - дорожньої мережі посилено контролюється нарядами поліції, і автомобіль цього водія часто зупиняють за перевищення швидкості. Тому цей водій вирішує розробити маршрут, на якому він би мав найбільшу ймовірність не бути зупиненим інспекторами поліції. Імовірність проїзду без зупинки для кожної вулиці наведено на схемі (рис. 2.11). Знайти найкоротший шлях та ймовірність того, що цей водій не буде зупиненим.

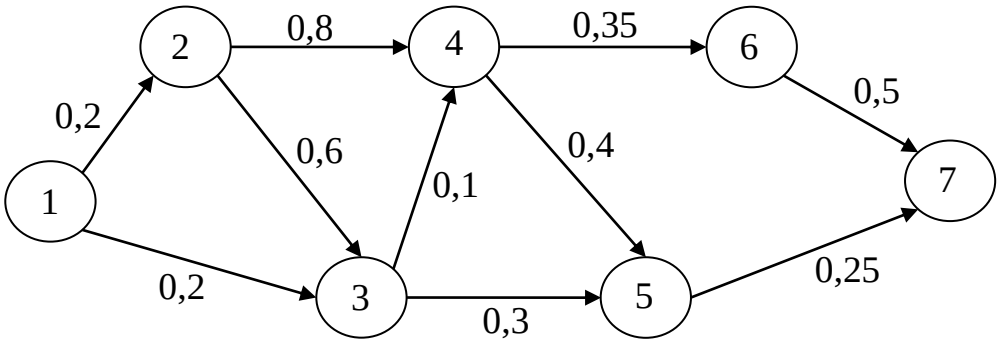


Рис. 2.11.

Імовірність не бути зупиненим на всій ділянці маршруту P дорівнює добутку ймовірностей не бути зупиненим кожної ділянки шляху p_i, тобто $P = p_1 \cdot p_2 \cdot ... \cdot p_k$. Цю задачу можна сформулювати як задачу пошуку найкоротшого шляху, якщо замість ймовірностей використовувати логарифми ймовірностей. Тоді добуток ймовірностей перетвориться в суму логарифмів ймовірностей

$$\lg P = \lg p_1 + \lg p_2 + ... + \lg p_k .$$

Максимізація ймовірностей p_i еквівалентна максимізації величини $\lg p_i$. Оскільки $p_i \le 1$, то $\lg p_i \le 0$. Замінивши ймовірності p_i на величини $(-\lg p_i)$, отримуємо мережу, для якої потрібно знайти найкоротший шлях (рис. 2.12.).

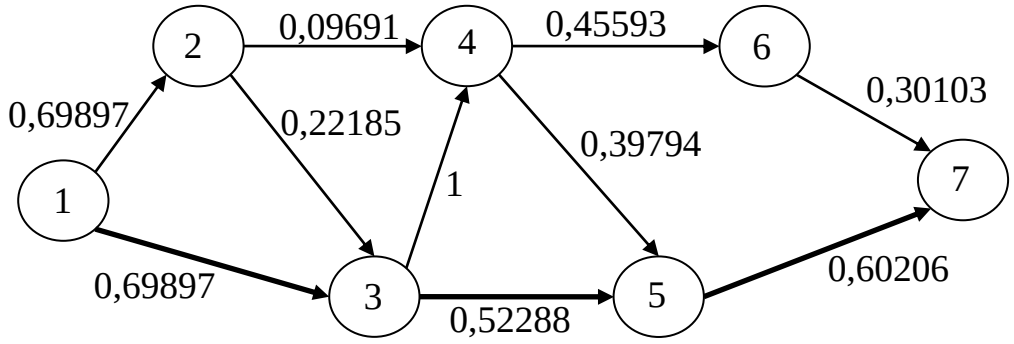


Рис. 2.12.

Найкоротшим шляхом є шлях $1\to3\to5\to7$ з «довжиною»: $(-\lg P)=1{,}82391$. Тоді максимальна ймовірність не бути зупиненим становить $P=0{,}0149$.

5. Розв'язання задач теорії графів в Maple.

Для розв'язання задач теорії графів в ***Maple*** потрібно підключити пакет ***GraphTheory***:

>with(GraphTheory):

Пакет містить понад 150 функцій, детальніше з якими можна познайомитись в довідковій системі ***Maple***. Зокрема, зважений граф (орієнтований та неорієнтований) можна задати функцією ***Graph (A)***, де ***A*** – матриця довжин дуг (ребер) графа. Якщо дуга (ребро) (i, j) відсутня у графі, то елемент матриці $a_{ij}=0$. Зобразити граф можна за допомогою функції

***DrawGraph*(*G*)**

де ***G*** – заданий граф (див. рис. 2.13.).

```
> restart :
> with(GraphTheory) :
> A := Matrix([[0, 1, 2, 0], [2, 0, 7, 0], [6, 5, 0, 2], [1, 0, 4, 0]]);
```

$$A := \begin{bmatrix} 0 & 1 & 2 & 0 \\ 2 & 0 & 7 & 0 \\ 6 & 5 & 0 & 2 \\ 1 & 0 & 4 & 0 \end{bmatrix} \quad (1)$$

```
> G := Graph(A);
```

G := Graph 1: a directed weighted graph with 4 vertices and 9 arc(s) (2)

```
> DrawGraph(G);
```

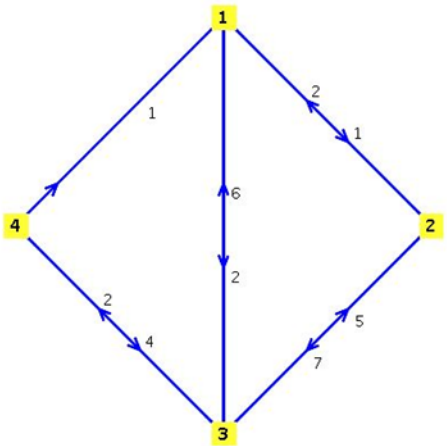


Рис. 2.13.

Для знаходження відстаней між будь-якою парою вершин (алгоритм Флойда) призначена функція

***AllPairsDistance*(*G*)**

Відстань та шлях між парою вершин ***s*** та ***t*** вершин знаходимо за допомогою функції

***DijkstrasAlgorithm*(*G*, *s*, *t*)**

Приклад 2.3. Для графа *G*, заданого на рис.2.13. знайти найменші відстані між будь-якою парою вершин. Знайти найкоротший шлях між вершиною 3 та 2.

```
> Матриця найменших відстаней між будь-якою парою вершин графа G
> AllPairsDistance(G);
```

$$\begin{bmatrix} 0 & 1 & 2 & 4 \\ 2 & 0 & 4 & 6 \\ 3 & 4 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{bmatrix} \quad (3)$$

```
> Шлях та відстань між вершинами 3 та 2 графа G
> DijkstrasAlgorithm(G, 3, 2);
```

$$[[3, 4, 1, 2], 4] \quad (4)$$

Тобто найкоротший шлях між вершиною 3 та 2: 3→4→1→2, довжина його 4.

Задачу комівояжера, як задачу пошуку у неорієнтованому зваженому графі циклу, що проходить через всі вершини графа, крім першої, рівно по одному разу (такий цикл називають Гамільтоновим) розв'язуємо за допомогою функції

***TravelingSalesman*(*G*)**

```
> restart;
> with(GraphTheory) :
Задаємо матрицю відстаней
```

$$> A := \begin{bmatrix} 0 & 7 & 16 & 21 & 2 \\ 13 & 0 & 21 & 15 & 43 \\ 25 & 3 & 0 & 31 & 17 \\ 13 & 10 & 27 & 0 & 33 \\ 9 & 2 & 19 & 14 & 0 \end{bmatrix};$$

$$A = \begin{bmatrix} 0 & 7 & 16 & 21 & 2 \\ 13 & 0 & 21 & 15 & 43 \\ 25 & 3 & 0 & 31 & 17 \\ 13 & 10 & 27 & 0 & 33 \\ 9 & 2 & 19 & 14 & 0 \end{bmatrix} \quad (1)$$

```
> G := Graph(A) : TravelingSalesman(G);
```

$$52, [1, 5, 3, 2, 4, 1] \quad (2)$$

Контрольні запитання.

1. Дайте визначення основних понять теорії графів.
2. Який граф називають Ейлеровим?
3. Сформулюйте умову існування Ейлерового циклу (шляху) в графі.
4. Як обчислити довжину шляху у незваженому графі?
5. За яким алгоритмом можна знайти найкоротший шлях між парою вершин у графі з правильною нумерацією? Опишіть кроки цього алгоритму.
6. За яким алгоритмом можна знайти найкоротший шлях між будь-якою парою вершин у графі?

Завдання на самопідготовку.

1. Віртуальний університет ЛДУ БЖД [Електронний ресурс]. — Режим доступу: http://virt.ldubgd.edu.ua/
2. Системний аналіз та теорія прийняття рішень [Електронний ресурс] / Чмир Оксана Юріївна. — Режим доступу: http://virt.ldubgd.edu.ua/course/view.php?id=1765

Доцент
кафедри прикладної математики і механіки Оксана Чмир

Лекція обговорена на засіданні
кафедри прикладної математики і механіки
Протокол № від “___” _____ 20__ р.