

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Жученко О.А. Дунаєва Т.А.

**ГРАФИ, ЯК ІНСТРУМЕНТ МОДЕЛЮВАННЯ
СКЛАДНИХ ОБ'ЄКТІВ ТА СИСТЕМ**

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для студентів, які навчаються
за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології»*

Київ
КПІ ім. Ігоря Сікорського
2020

Графи. як інструмент моделювання складних об'єктів та систем: [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського; уклад.: О.А. Жученко, Т.А. Дунаєва, – Електронні текстові дані (1 файл: 2,54 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 68 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 10 від 18.06.2020 р.) за поданням Вченої ради Інженерно-хімічного факультету (протокол № від р.)

Електронне мережне навчальне видання

ГРАФИ, ЯК ІНСТРУМЕНТ МОДЕЛЮВАННЯ СКЛАДНИХ ОБ'ЄКТІВ ТА СИСТЕМ

Укладачі	<i>Жученко Олексій Анатолійович, доктор технічних наук, доцент Дунаєва Тамара Альбінівна, кандидат фіз-мат - наук, доцент</i>
Відповідальний редактор	<i>Жученко А. І., завідувач кафедри «Автоматизація хімічних виробництв», доктор технічних наук, професор</i>
Рецензент	<i>Степанюк Андрій Романович, к.т.н., доцент кафедри «Машини і апарати хімічних та нафтопереробних виробництв» інженерно-хімічного факультету КПІ ім. Ігоря Сікорського</i>

В запропонованому навчальному посібнику викладено основи теорії графів, що мають практичне застосування в сучасних задачах керування технологічними об'єктами. Визначено фундаментальні властивості загальної структури графів Представлено опис основних алгоритмів на графах та досліджено поведінку складних систем з використанням алгоритмів теорії графів. При виконанні практичних робіт передбачено використання математичних процесорів MS Exel та MATLAB.

Призначений для студентів спеціальності «Автоматизація та комп'ютерно-інтегровані технології» усіх форм навчання

© КПІ ім. Ігоря Сікорського, 2020

ВСТУП

В основу посібника покладено курс лекцій, з дисципліни «Спеціальні розділи математики» для студентів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології».

Створення ефективних математичних моделей, комп'ютерно-автоматизованих систем та їх функціонування не можливе без дискретного моделювання.

Під дискретним моделюванням ми розуміємо моделювання системи стан якої змінюється в дискретні моменти часу. Реальний світ є дискретним по своїй суті. Практично всі обчислювальні моделі є дискретними скінченими структурами. На сьогоднішній день виникла необхідність у вивченні методів аналізу та синтезу дискретних систем. Тому прості для сприйняття графові моделі, є універсальним інструментом для різних наукових напрямків.

Проблеми оптимізації теплових, газових та електричних мереж, питання удосконалення алгоритмів і створення нових хімічних сполук пов'язані з фундаментальними властивостями таких абстрактних математичних об'єктів як графи.

Невід'ємною частиною моделювання сучасної цифрової економіки є інформаційно-аналітичні системи, які враховують не тільки кількісні оцінки економічного життя, але й суб'єктивні данні, наприклад особисті мотиви поведінки людей, які приймають рішення. На сьогоднішній день уже існують математичні моделі конкуренції, розподілення влади і моделі корупції.

Отримали розвиток такі наукові напрямки як *Data science* - наука про дані, *Social mining* - соціальна аналітика, складовою частиною яких є, *Network visualization*-візуалізація графів та мереж.

Теорія графів тісно пов'язана з багатьма розділами математики серед яких – теорія груп, теорія матриць, чисельний аналіз.

Наочність теоретико–графових структур і дохідливість мови теорії графів дозволяють зробити для студентів більш доступними і формулювання досить складних прикладних завдань, і дуже тонкі методи їх розв'язання завдяки їх представленню у вигляді діаграм.

Рукопис складається з двох розділів. У першому розділі описуються основні визначення та властивості теорії графів, обходи графів, операції над графами, дерева.

Другий розділ присвячений прикладній теорії алгоритмів на графах.

В кінці посібника для закріплення введених понять приведені завдання для практичних робіт, розглянутих алгоритмів і моделей.

РОЗДІЛ 1. ТЕОРІЯ ГРАФІВ

Останнім часом теорія графів привертає усе більш пильну увагу фахівців різних областей знання. Поряд з традиційними використаннями її в таких науках, як фізика, електротехніка, хімія, вона проникла і в науки, що вважалися раніше далекими від неї – економіку, соціологію, лінгвістику і ін. Особливо важливий зв'язок існує між теорією графів і теоретичною кібернетикою (особливо теорією автоматів, дослідженням операцій, теорією кодування, теорією ігор).

Велике значення має теорія графів для вирішення багатьох важливих питань практики, з числа яких тут слід згадати, наприклад, про так звані «транспортні задачі (задачі про планування найбільш раціональної системи перевезень вантажів і транспортної мережі) або задачі, пов'язані з електричними мережами, задачі про потоки в мережі нафтопроводів і взагалі про так зване «математичне програмування».

Народження теорії графів в 18 столітті було пов'язано з математичними головоломками, і досить довго на вчення про графи дивилися як на «несерйозну» тему, прикладне значення якої цілком пов'язано з іграми та розвагами. В цьому відношенні долю теорії графів можна порівняти з долею теорії ймовірностей, яку також спочатку розглядали лише у зв'язку з її застосуваннями до азартних ігор.

Теорія графів є одною з небагатьох областей математики, дата народження якої може бути вказана. Батьком теорії графів (так само як і топології) є швейцарський математик Леонард Эйлер (1707-1783). Його перша робота з теорії графів, з'явилася в 1736 р. в публікаціях Петербурзької Академії наук, в якій він розв'язав тепер уже класичну задачу, яка називалася «задача про Кенігсбергські мости».

Цей граф зображено на рис.1.1. Ейлер показав, з якої б вершини ми не почали обхід, ми не зможемо обійти весь граф і повернутися назад, не проходячи ніякого ребра двічі. Для того, щоб такий обхід існував, потрібно, щоб у кожній вершині графа було б стільки ребер, що входять до неї, скільки і виходять з неї, тобто в кожній вершині графа має бути парне число ребер.

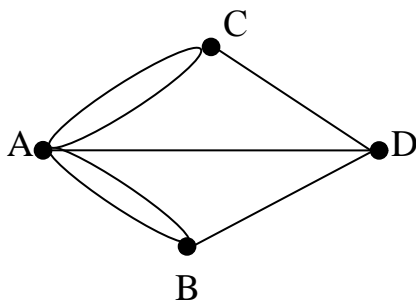


Рис. 1.1. Графічне представлення задачі про Кенігсбергські мости

1.1. Основні визначення та властивості графів

Припустимо, що футбольна команда вашого факультету бере участь у змаганнях і грає з командами інших факультетів. Загальна кількість команд дорівнює шести. Вашу команду позначимо буквою A , а інші команди – B, C, D, E і F . Через кілька тижнів після початку змагань виявилося, що деякі з команд вже зіграли одна з одною, наприклад:

A з C, D, F , E з B, D, F , B з C, E, F , F з A, B, D, E .

C з A, B , D з A, E, F

Це можна зобразити за допомогою геометричної схеми. Кожну команду представимо точкою або маленьким кружечком і з'єднаємо відрізком ті пари точок, які відповідають командам, які вже зіграли одна з одною. Тоді для даного списку проведених ігор ми отримаємо схему, зображену на рис.1.2.

Схема такого виду називається *графом*. Вона складається з декількох точок A, B, C, D, E, F , званих *вершинами*, і декількох, що з'єднують ці вершини, відрізків, таких, як AB або EB , званих *ребрами* графа.

Кожну сукупність ігор будь-якого турніру можна уявити відповідним графом. Навпаки, якщо задано деякий граф, тобто фігура, що складається з точок - вершин, сполучених прямолінійними відрізками – ребрами, то його можна розглядати як схему такого змагання.

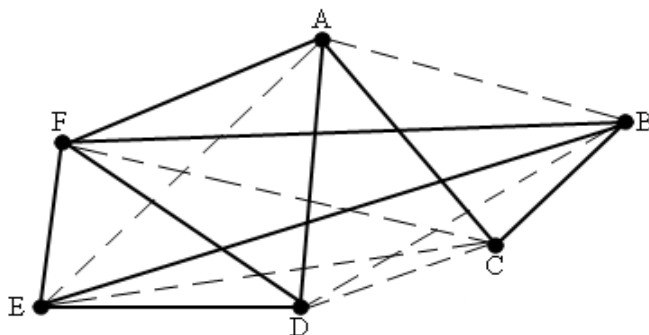


Рис.1.2. Граф, що описує ігри, штрихом позначено ще не зіграні ігри

Як приклад можна розглянути граф, зображений на рис. 1.3.

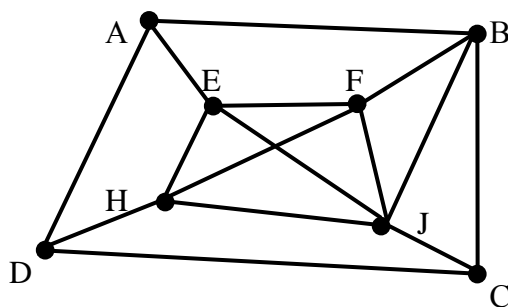


Рис.1.3. Граф змагань, ребрами позначено зіграні ігри

Однією з основних причин пізнього народження теорії графів як самостійної теорії в межах математики, є те, що досі не досягнуто однаковості в термінології. Не тільки окремі елементи, але і саме поняття "графа" визначається по різному.

Визначення. Простий нерієнтований граф $G = (V, X)$ складається зі скінченної не пустої множини вершин V , яка містить v вершин, і множини ребер X , яка містить x неупорядкованих пар різних вершин з V .

$$G = (V, X)$$

Визначення. Графом називають множину вершин V , зв'язок між якими визначений множиною ребер X .

Кожну пару $x = \{u, v\}$ вершин в X називають ребром графа G і кажуть, що x з'єднує u і v . Ми будемо писати $x = uv$ і говорити, що u і v **суміжні вершини** (рис. 1.4). Іноді це позначається **$u \text{ adj } v$** .

Ми можемо представити собі множину X ребер як множину пар суміжних вершин, визначаючи тим самим не рефлексивне, симетричне відношення на множині V . Відсутність рефлексивності пов'язана з тим, що в простому графі нема петель, тобто ребер, два кінця яких знаходяться в одній вершині. Симетричність відношення витікає з того факту, що ребро, яке з'єднує вершину u з v з'єднує v з u (інакше кажучи ребра не орієнтовані, тобто не мають напрямку).

Вершина u і ребро x **інцидентні**, також як v і x .

Інцидентність - геометричний термін, який застосовується для позначення відношення приналежності (\in) (зв'язку, з'єднання) між основними об'єктами геометрії: точками, прямими площинами.

Якщо вершина u є кінцем ребра x , то говорять, що вони **інцидентні**. Вершина u інцидентна ребру x і ребро x інцидентно вершині u (рис.2.4). В той час, як суміжність це відношення між однорідними об'єктами (вершинами), інцидентність, – відношення між різними об'єктами (вершинами і ребрами).

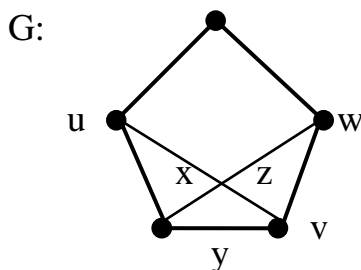


Рис. 1.4. Суміжність вершин і ребер

Якщо два різних ребра x та y інцидентні одній і тій самій вершині, то вони називаються **суміжними**. Таким чином на рис.1.4. вершини u і v суміжні, а вершини u і w ні; ребра x та y суміжні, а x і z ні.

Граф з p вершинами і q ребрами називається (p, q) - графом.

Таким чином на рис.1.4. вершини u і v суміжні, а вершини u і w ні; ребра x та y суміжні, а x і z ні.

З визначення графа випливає, що у графа не може бути петель (тобто ребер, що з'єднують вершини самі з собою).

Нуль - граф і повний граф. Існують деякі спеціальні графи, що зустрічаються в багатьох додатках теорії графів. Будемо знову розглядати граф або наочну схему, що ілюструє перебіг спортивних змагань. До початку сезону, поки ще ніякі ігри не проводилися, на графі нема ніяких ребер. Такий граф складається з одних ізольованих вершин, тобто з вершин, не сполучених ніякими ребрами. Граф такого виду називається *нуль – графом* або *тривіальним*. Нуль-граф зазвичай позначається символами O_1, O_2, O_3 і т.д., так, що O_n – це нуль - граф з n вершинами, який не має ребер.

Розглянемо інший крайній випадок. Припустимо, що по закінченні сезону кожна команда зіграла по одному разу з кожною з інших команд. Тоді на відповідному графі кожна пара вершин буде сполучена ребром. Такий граф називається *повним* графом. На рис. 1.5. зображені повні графи з числом вершин $n = 1, 2, 3, 4, 5$.

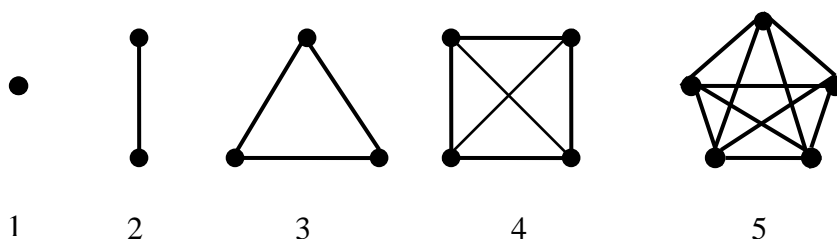


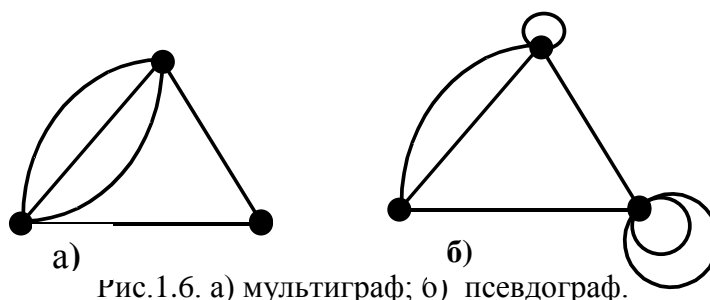
Рис. 1.5. Приклади повних графів

Ми позначимо ці повні графи відповідно через U_1, U_2, U_3, U_4, U_5 , так, що граф U_n складається з n вершин і ребер, які з'єднують різні пари цих вершин. Цей граф можна уявити собі як n -кутник, в якому проведено всі діагоналі.

Маючи деякий граф, наприклад граф G , зображений на рис. 1.3, ми завжди можемо перетворити його в повний граф з тими ж самими вершинами, додавши відсутні ребра (тобто ребра, які відповідають іграм, що тільки будуть зіграні, ребра зображені штрих–пунктиром).

Граф, у якого петлі не допускаються, але пари вершин можуть з'єднуватися більш ніж одним ребром, називається **мультиграфом**. Ці ребра називаються **кратними** (рис.1.6 а)).

Псевдографом називається неорієнтований граф, у якого допускаються петлі й кратні ребра (рис.1.6.б)).



Граф у задачі про «Кенігсбергські мости» є насправді мультиграфом.

Орієнтовані графи. Часто зв'язки між об'єктами характеризуються цілком певною орієнтацією. Наприклад, на деяких вулицях допускається односторонній рух, відношення між людьми можуть визначатися підпорядкованістю або старшинством.

Для вказівки напрямку зв'язку між вершинами графа відповідне ребро позначається стрілкою.

Визначення. Орієнтований граф, або орграф $G=(V, X)$, складається з скінченної не пустої множини V вершин і заданого набору X впорядкованих пар різних вершин. Елементи з X називаються орієнтованими ребрами або *дугами*, елементи з множини V – *вузлами*. За визначенням у орграфі нема петель і кратних дуг.

Множина V може бути нескінченною. Ми обмежимося скінченими орієнтованими графами, і сам термін орграф завжди будемо відносити до скінченного графа.

Взагалі теорія графів розділяється на дві частини: теорію орієнтованих графів та теорію неорієнтованих графів. Обидві теорії існують самостійно, і кожна з них має свій клас об'єктів, в кожній використовуються свої правила.

Відповідно до визначення поняття *орієнтований граф* є еквівалентним поняттю відношення разом з тією множиною, де відношення визначено. Таким чином, теорію орієнтованих графів можна розглядати як продовження теорії відношень.

Коли дуга позначається впорядкованою парою, що складається з початкової та кінцевої вершин (тобто двома кінцевими вершинами дуги), її напрямок вважається заданим від першої вершини до другої. Так, наприклад, на рис.2.8. а) позначення (v_1, v_2) відноситься до дуги x_1 , а (v_2, v_1) до дуги x_2 .

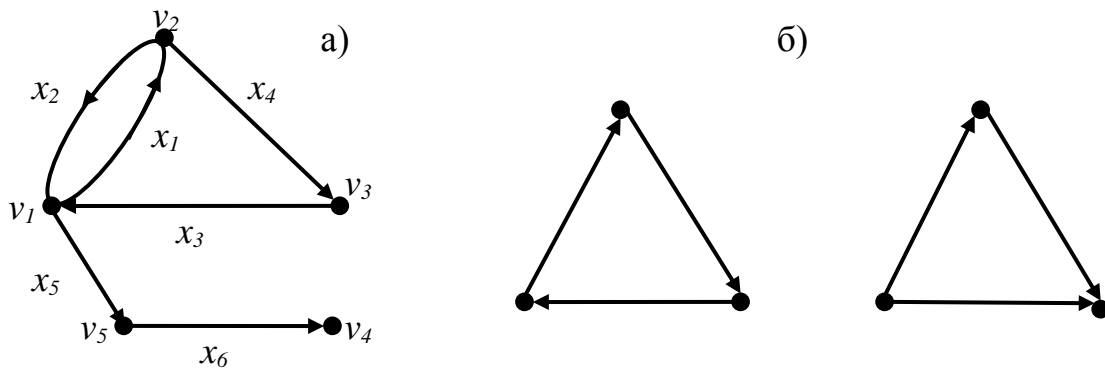


Рис.1.7. а) Орієнтований граф з 5 вершинами і 6 дугами, б) 2 останні графи спрямовані

Спрямований граф – це орграф, що не має симетричних пар (т.б. (u,v) і (v,u)) орієнтованих ребер.

1.2. Матриці графів

Неорієнтований граф повністю визначається або його суміжностями, або його інцидентіями. Зазначену інформацію про граф зручно представляти в матричній формі. З неорієнтованим графом зв'язано кілька матриць, це матриця *суміжностей*, матриця *інцидентій*, матриця *циклів* і матриця *коциклів*. Часто ці матриці використовуються при виявленні певних властивостей графів.

Матриця суміжностей. Матрицею суміжностей $A = \|a_{ij}\|$ неорієнтованого графа G з v вершинами називається $(v \times v)$ – матриця, в якій $a_{ij} = 1$, якщо вершина v_i суміжна v_j , і $a_{ij} = 0$ в іншому випадку. Таким чином, існує взаємно-однозначна відповідність між неорієнтованими графами з v вершинами і симетричними бінарними $(v \times v)$ – матрицями з нулями по діагоналі (рис. 1.8).

Симетричність відношення в термінах матриці суміжності означає, що матриця A симетрична відносно головної діагоналі, а нереклексивність цього відношення дає на головній діагоналі нулі.

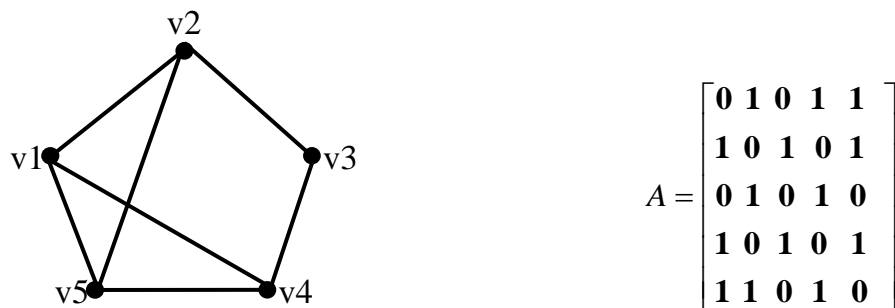


Рис.1.8. Граф і його матриця суміжностей.

Рядки та стовпці матриці відповідають вершинам графа, а її (ij) - елемент дорівнює числу кратних ребер, що зв'язують вершини v_i і v_j .

Матриця суміжності неорієнтованого графа завжди *симетрична*, а орграфа – у загальному випадку *несиметрична*. Неорієнтованим ребрам відповідають пари ненульових елементів, симетричних щодо головної діагоналі матриці, дугам – ненульові елементи матриці, а петлям – ненульові елементи головної діагоналі.

Матриця інцидентності графа. Матрицею інцидентності $A = \|a_{ij}\|$ графа G з v вершинами і x ребрами називається $(v \times x)$ - матриця, в якій $a_{ij} = 1$, якщо вершина v_i інцидентна ребру x_j , і $a_{ij} = 0$ в іншому випадку (рис.2.10).

Що треба відмітити для неорієнтованих графів:

1) Рядки матриці інциденцій відповідають вершинам, а стовпці - ребрам.

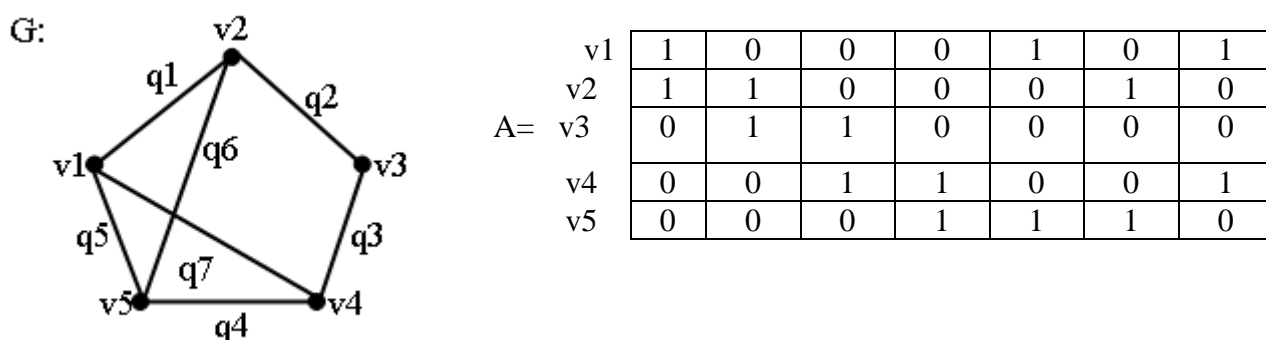


Рис.1.9. Матриця інциденцій графа G

2) Кожен стовпчик матриці містить обов'язково 2 одиничних елемента (для орграфа ці елементи завжди мають різні знаки і дорівнюють відповідно 1 і -1).

3) Кількість одиниць у рядку дорівнює степені відповідної вершини (для орграфа кількість позитивних одиниць визначає позитивну степінь, а кількість негативних одиниць - негативну степінь).

4) Нульовий рядок відповідає ізольованій вершині, а нульовий стовпчик – петлі.

Слід зазначити, що нульовий стовпчик матриці лише вказує на наявність петлі, але не містить відомостей про те, з якою вершиною ця петля пов'язана.

1.3. Матриці орграфів

При розгляді орграфів (рис.1.10) розрізняють *позитивну* інцидентність (дуга виходить з вершини) і *негативну* інцидентність (дуга входить у вершину). У матриці інциденцій орграфа ненульовий ij – елемент дорівнює 1, якщо v_i – початкова вершина дуги x_j , і дорівнює -1, якщо v_i – кінцева вершина дуги x_j .

Крім матриці суміжності та матриці інциденцій з орграфом пов'язані ще три матриці – матриця *відстаней*, матриця *обходів* та матриця *досяжності*.

В матриці *досяжності* R елемент r_{ij} дорівнює 1, якщо вершина v_i досяжна з v_j , і дорівнює 0 в протилежному випадку.

В матриці *відстаней* (i,j) -й елемент дорівнює відстані від вершини v_i в вершину v_j , якщо ж з v_i в v_j нема шляхів, то відповідний елемент вважаємо рівним нескінченості.

В матриці *обходів* (i,j) -й елемент дорівнює довжині самого довгого шляху від v_i в v_j , а якщо таких шляхів немає, то такий елемент вважаємо рівним нескінченості.

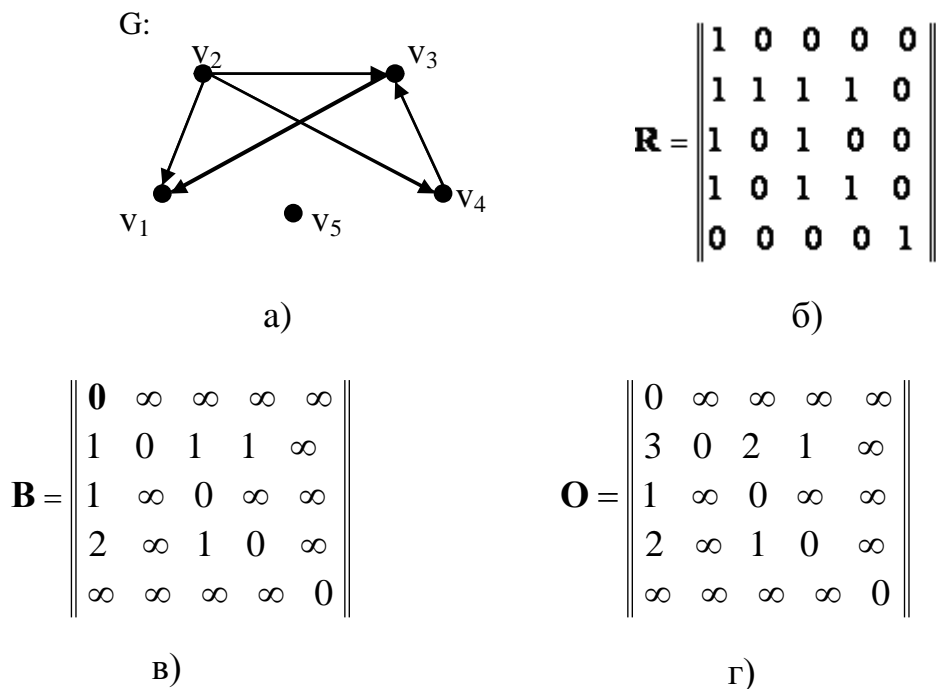


Рис.1.10. а) Приклад орграфа, б) матриця досяжності, в) матриця відстаней, г) матриця обходів графа G

Ефективних методів знаходження елементів матриці обходів не існує. Ця проблема пов'язана з деякими іншими задачами теорії графів, а саме – знаходження остовних циклів, розв'язок задачі про комівояжера.

1.4. Ізоморфізм графів

В перекладі з грецької **мови «ізіс»** означає рівний, а **«морфі»**– вид, форма.

Для багатьох практичних задач важливим є розпізнавання ізоморфізму та ізоморфного вкладення складних структур, які задаються у вигляді графів. З змістовних міркувань ізоморфізм графів структур означає тотожне функціонування самих структур, що дає можливість в деяких випадках замінити одну структуру іншою, їй ізоморфною.

На рис.1.11. зображені 3 графа, які з геометричної точки зору абсолютно різні.

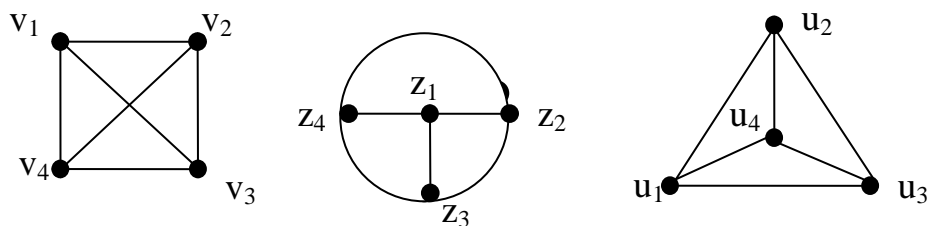


Рис.1.11. Ізоморфні графи

Але по суті вони розрізняються лише накресленням, а відношення інцидентності (при відповідному позначенні вершин і ребер) для них однакові. Так, скажімо вершини v_1 і v_4 з'єднані ребром, відповідні їм вершини z_1 і z_4 та u_1 і u_4 , також з'єднані ребром і навпаки.

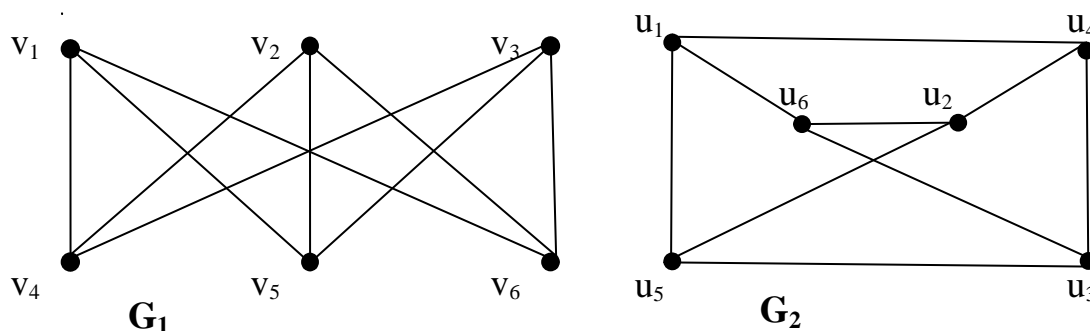
Можна сказати, що ізоморфні графи, це один і той же граф, у якому вершини названі по іншому.

Визначення. Два графа G і H ізоморфні ($G \sim H$) або ($G = H$), якщо між множинами вершин існує взаємнооднозначна відповідність, що зберігає суміжність.

Визначення. Два графа називаються ізоморфними, якщо існує ізоморфне відображення одного з цих графів на інший.

Визначення. Ізоморфним відображенням одного неорієнтованого графа на інший називається взаємнооднозначне відображення вершин і ребер одного графа відповідно на вершини і ребра іншого графа, при якому зберігається відношення інцидентності.

Наприклад: G_1 і G_2 (рис. 1.12.а) ізоморфні при відповідності $v_i \leftrightarrow u_i$ і G_3 (рис.1.13) ізоморфний кожному з них. Очевидно, що ізоморфізм є відношення еквівалентності на графах.



Р

ис.1.12. а) Ізоморфні графи G_1 та G_2

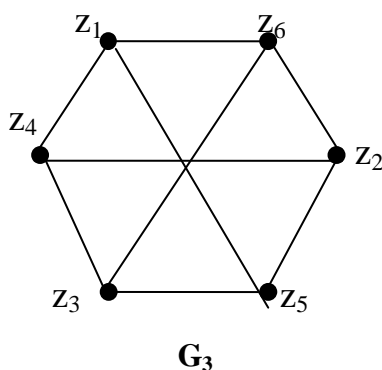


Рис1.13. Граф G_3 ізоморфний двом попереднім

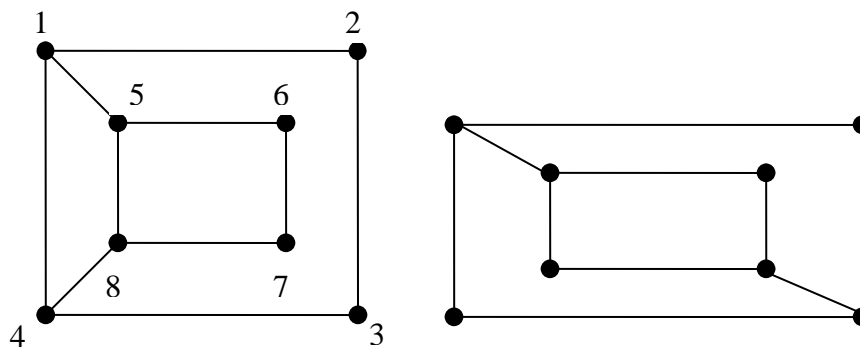


Рис.1.14. Неізоморфні графи

На рис.1.14. зображені не ізоморфні графи. На першому графі є послідовність з 8 суміжних ребер, що повертаються до вихідної вершини, у той час як на другому графі (рис. 1.14) такої послідовності немає, як би ми не позначали вершини 2-го графа, ми не зможемо для кожної пари сполучених ребром вершин одного графа вказати у другому відповідну пару вершин, також сполучену ребром.

Числова характеристика, однакова для всіх ізоморфних графів, називається *інваріантом* графа. Так для графів з числом вершин менших чотирьом, кількість вершин $V(G)$ та кількість ребер $X(G)$ вважаються повним інваріантом графа G .

Для графів, зображених на рис.1.14, інваріантами є а) кількість вершин, б) кількість ребер, в) кількість вершин конкретної степені.

Але на сьогоднішній день невідомо ніякої повної системи інваріантів, які б визначали граф з точністю до ізоморфізму.

1.5. Елементи графів

Після визначення графів як цілісних об'єктів, дамо визначення різним складовим елементам графів.

Нехай задано граф $G=(V,X)$.

Визначення. Підграфом графа G називається граф, у якого всі вершини і ребра належать G .

Визначення. Остовним підграфом G_p графа $G = (V, X)$ називається граф (V, X_p) , для якого $X_p \subseteq X$.

Таким чином, остовний підграф має таку ж кількість вершин, що і граф G , але множина ребер підграфа G_p є підмножиною множини ребер вихідного графа G , тобто це підграф графа G , що містить всі його вершини.

Граф на рис.1.15 б) – остовний підграф G_p графа G , зображеного на рис.1.15 а).

Якщо представити граф, вершинами якого є співробітники деякої організації, а ребрами – лінії зв'язку між ними, тоді граф, який представляє тільки найбільш важливі канали зв'язку

даної організації є остовним підграфом; граф, який детально представляє лінії зв'язку тільки деякої частини цієї організації (наприклад, відділу), є породженим підграфом, а граф, який представляє тільки важливі зв'язки в межах відділу, є підграфом.

На рис.1.15.в) показано *породжений* підграф графа, зображеного на рис.1.15.а), який має тільки вершини v_1, v_2, v_3, v_4 і ребра, які їх зв'язують.

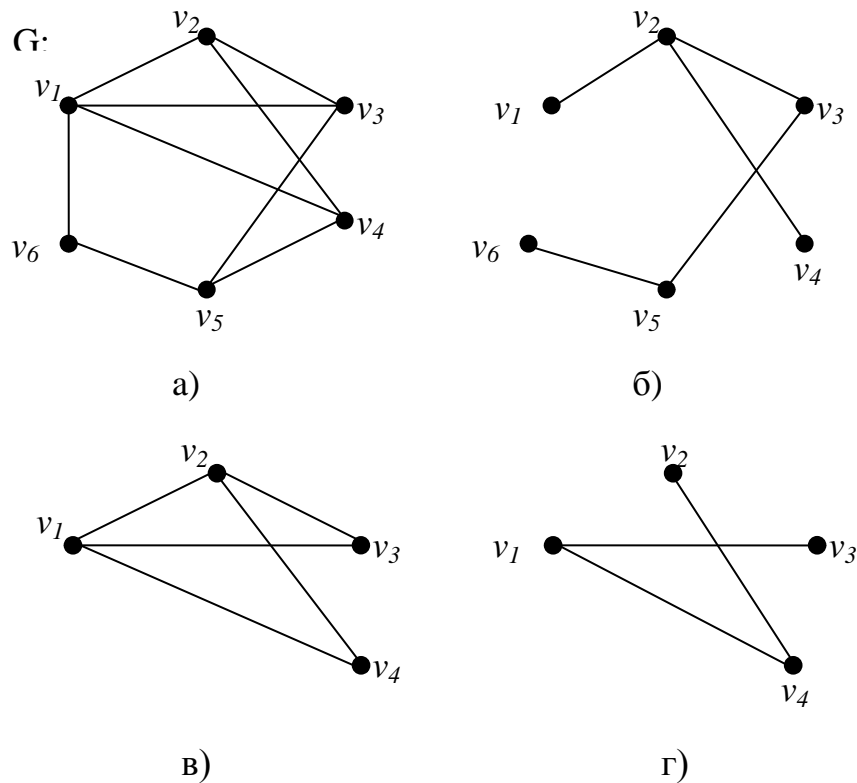


Рис. 1.15. а) граф G, б) остовний підграф, в) породжений підграф, г) підграф

1.6. Маршрути та зв'язність

У даному параграфі ми розглянемо основні структурні властивості графів. Однією з основних властивостей графа є властивість бути *зв'язним*. Але спочатку введемо певні позначення.

Визначення. *Маршрутом* у графі G називається послідовність вершин і ребер, що чергуються: $v_0 x_1 v_1 x_2 v_2 x_3 v_3 x_4 \dots v_{n-1} x_n v_n$.

Ця послідовність починається і закінчується вершиною і кожне ребро послідовності *інцидентне* двом вершинам. Зазначений маршрут поєднує вершини v_0 і v_n , його можна позначити як $v_0 v_1 v_2 v_4 \dots v_{n-1} v_n$.

Маршрут *замкнутий*, якщо $v_0 = v_n$, і *відкритий* в протилежному випадку.

Маршрут називається *ланцюгом* (trail), якщо всі його ребра різні.

Маршрут називається *простим* (елементарним) ланцюгом, якщо всі вершини (а значить, і всі ребра) різні.

Замкнутий ланцюг називається **циклом**.

Замкнутий маршрут називається **простим циклом**, якщо всі його n вершин різні і $n \geq 3$.
Для ілюстрації маршрутів розглянемо граф, зображений на рис.1.16

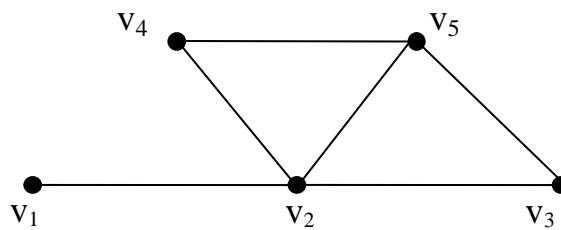


Рис.1.16. Граф для ілюстрації маршрутів [4]

В даному графі

- $v_1 v_2 v_5 v_2 v_3$ - маршрут, який не є ланцюгом,
- $v_1 v_2 v_5 v_4 v_2 v_3$ - ланцюг, але не простий ланцюг,
- $v_1 v_2 v_5 v_4$ - простий ланцюг,
- $v_2 v_5 v_4 v_2$ - простий цикл.

Позначимо через C_n граф, що складається з одного простого циклу з n вершинами і через P_n простий ланцюг з n вершинами; C_3 часто називають трикутником.

Граф називається **зв'язним**, якщо будь-яка пара його вершин з'єднана простим ланцюгом.

Орієнтований граф називається

слабо зв'язним, якщо відповідний йому неорієнтований граф є зв'язним.

сильно зв'язним, будь-яка вершина графа досяжна з будь-якої іншої вершини.

Теорема: (Уїтні, 1932) Простий граф G є k -зв'язним тоді і тільки тоді, коли між двома його вершинами u та w існує k – шляхів, які не мають спільних ребер.

Відношення зв'язності вершин є еквівалентністю. Класи еквівалентності по відношенню до зв'язності називаються **компонентами зв'язності** графа. Число компонент зв'язності графа позначається як $k(G)$.

Визначення: Компонентами зв'язності називається множина вершин графа G така, що для будь-яких двох вершин з цієї множини існує шлях із однієї вершини до другої, і не існує шляху з вершин цієї множини в вершину з іншої множини.

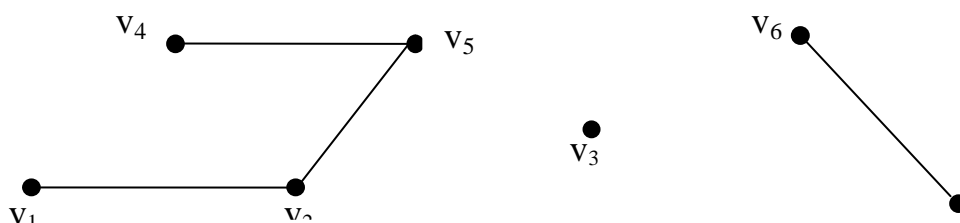


Рис 1 17 Незв'язний граф і його 3 компоненти

Граф G є зв'язним тоді і тільки тоді, коли $k(G) = 1$.

Не зв'язний граф має принаймні дві компоненти. Граф на рис.1.17. має 3 компоненти.

Довжина маршруту дорівнює кількості ребер у графі, причому кожне ребро вважається стільки разів, скільки воно зустрічається у даному маршруті.

Обхват графа G - це довжина найкоротшого простого циклу графа G і позначається $g(G)$.

Оточення графа G - довжина найдовшого простого циклу графа G позначається $c(G)$.

Відстанню між двома вершинами u і v графа G називається довжина найкоротшого простого ланцюга, що сполучає їх і позначається $d(u, v)$.

Геодезичним ланцюгом називається найкоротший простий ($u - v$) - ланцюг, довжина якого дорівнює $d(u, v)$.

Діаметром графа називається довжина найдовшого геодезичного ланцюга, яка позначається $d(G)$.

Граф на рис.1.16. має обхват $g = 3$, оточення $c = 4$ і діаметр $d = 2$.

1.7. Степінь графа

Визначення. Степеню вершини v_i в неорієнтованому графі G називається число ребер, інцидентних v_i і позначається d_i або $\deg v_i$.

Оскільки кожне ребро інцидентне двом вершинам, то в сумі степені вершин графа кожне ребро рахується двічі. Таким чином ми приходимо до теореми, що є історично першої теоремою теорії графів і була сформульована Ейлером.

Т е р е м а . Сума степенів вершин неорієнтованого графа G дорівнює подвоєному числу його ребер:

$$\sum \deg v_i = 2q$$

Н а с л і д о к 1. У будь-якому графі кількість вершин з непарними степенями парна.

Для орієнтованого графа число дуг, які мають вершину v_i як початкову, називається *напівстепенем виходу* вершини v_i , аналогічно, число дуг, у яких v_i є кінцевою вершиною, називається *напівстепенем заходу* вершини v_i .

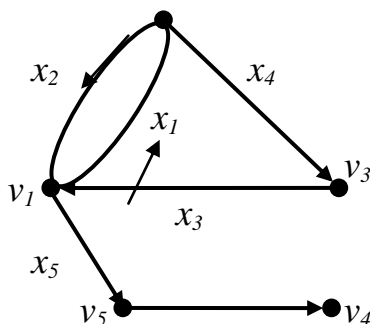


Рис.1.18. Орієнтований граф G

Таким чином, на рис.1.18 напівстепінь виходу вершини v_l яку позначимо як $d_0(v_l)$, дорівнює $|\Gamma(v_l)| = 2$, і напівстепінь заходу вершини v_l

позначимо $d_l(v_l)$, дорівнює $|\Gamma^{-1}(v_l)| = 2$.

Очевидно, що сума напівстепенів заходу для всіх вершин графа, а також сума напівстепенів виходу всіх вершин дорівнює загальному числу дуг графа G , тобто

$$\sum_{i=1}^n d_0(v_i) = \sum_{i=1}^n d_l(v_i) = m,$$

де n – число вершин, m – число дуг орієнтованого графа.

Мінімальна степінь вершин позначається через $\mindeg G$ або $\delta(G)$, максимальна степінь – через $\maxdeg G = \Delta(G)$. Якщо $\delta(G) = \Delta(G) = r$, то всі вершини мають однакову степінь і такий граф називається регулярним (або однорідним, Ортега) степені r . У цьому випадку говорять про степінь графа і пишуть $\deg G = r$.

Якщо G – регулярний граф степені 1, то кожна його компонента зв'язності містить точно одне ребро. У регулярному графі степені 2 кожна компонента зв'язності – цикл. Перші цікаві по своїй структурі графи мають степінь 3 і називаються кубічними. З теореми Ейлера випливає наступне твердження.

Н а с л і д о к : 2. Кубічний граф має парне число вершин.

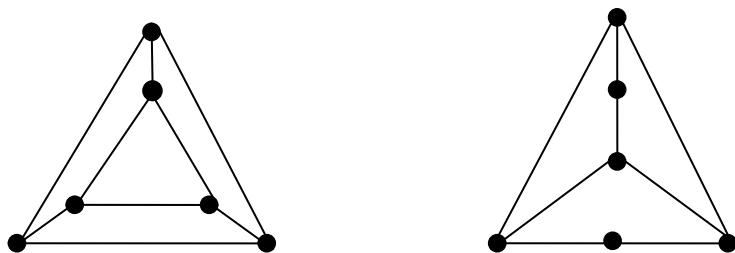


Рис.1.19. Кубічні графи з 6 вершинами

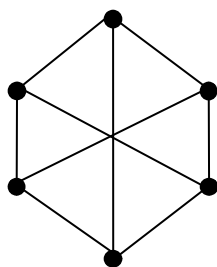
Серед цікавих задач – головоломок, існує, так звана, задача Рамсея: довести, що серед будь-яких шести осіб знайдуться або троє попарно знайомих, або троє попарно незнайомих.

Зазначену ситуацію можна описати графом з 6 вершинами, що представляють людей; суміжність двох вершин відповідає знайомству. Потрібно довести, що у графі знайдуться або три попарно суміжні, або три попарно не суміжні вершини. Доповнення графа G має в якості множини вершин множину $V(G)$, дві вершини в \bar{G} суміжні тоді і тільки тоді, коли вони не суміжні в G .

На рис. 1.20 у графі G немає трикутників, а в графі \bar{G} їх рівно два.

Граф \bar{G} , який є об'єднанням 2-х трикутників, називається графом Давида.

G:



\bar{G} :

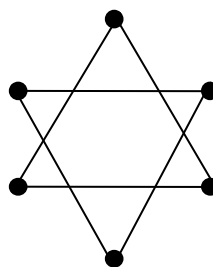


Рис.1.20. Граф G: та його доповнення

Т е р е м а. Якщо G - граф з шістьма вершинами, то або G або \bar{G} має трикутник.

В и з н а ч е н н я . Дводольний граф (або біграф) G – це граф, множину вершин якого можна розбити на дві підмножини V_1 і V_2 таким чином, що кожне ребро графа G з'єднує вершини з різних множин.

Якщо дводольний граф містить всі ребра, які з'єднують множини V_1 і V_2 , то він називається *повним* дводольним графом [4].

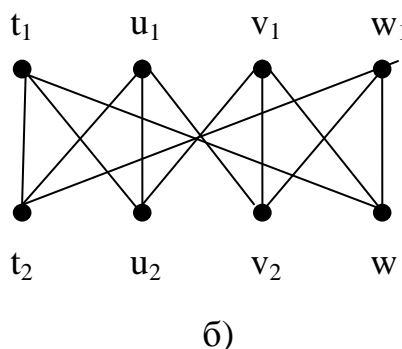
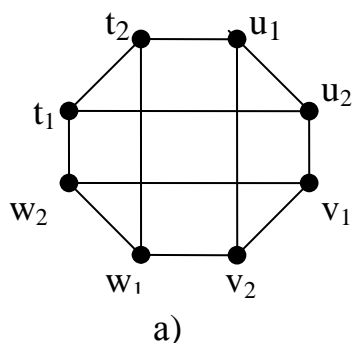


Рис.1.21.б) Дводольний граф

Повний дводольний граф у якого $|V_1| = m$ і $|V_2| = n$ позначається як $K(m, n) = K_{m,n}$. У графі $K_{m,n}$ є $m \times n$ ребер (рис.1.21б)).

1.8. Операції над графами

Нехай графи G_1 і G_2 мають непересічні множини вершин V_1 і V_2 і множини ребер X_1 і X_2 , які не перетинаються .

Видалення вершини v_i з графа G призводить до підграфа $G - v_i$, що містить всі вершини графа G , за винятком v_i , і всі ребра графа G , не інцидентні v_i . Іншими словами, $G - v_i$ є максимальний підграф графа G , що не містить v_i (рис.1.22).

Видалення ребра x_j з G , призводить до остовного підграфа, що містить всі ребра графа G , за винятком x_j . Тобто $G - x_j$ є максимальний підграф графа G , що не містить x_j (рис.1.22).

З іншого боку . якщо v_i і v_j не суміжні в G , то додавання ребра $v_i v_j$ утворює *найменший підграф* графа G , що містить ребро $v_i v_j$ (рис.1.22). Нехай графи G_1 і G_2 мають непересічні

множини вершин V_1 і V_2 і множини ребер X_1 і X_2 , які не перетинаються. З'єднання графів, яке позначається $G_1 + G_2$ - складається з $G_1 \cup G_2$ і всіх ребер, що з'єднують V_1 і V_2 . Ця операція показана на рис.1.23 [4].

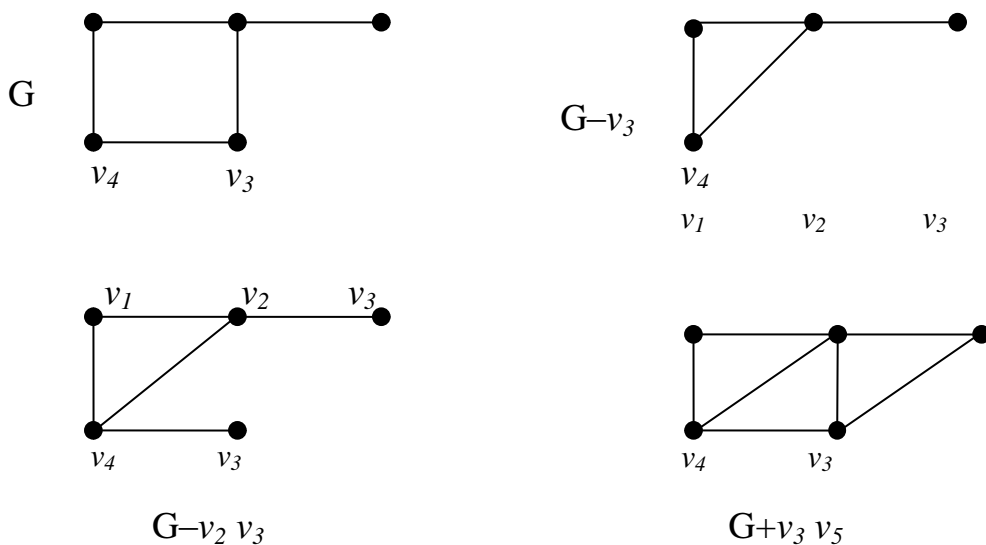


Рис. 1.22. Графи з видаленою вершиною або видаленим ребром.

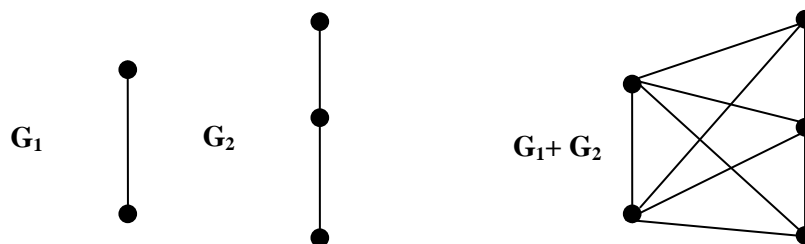
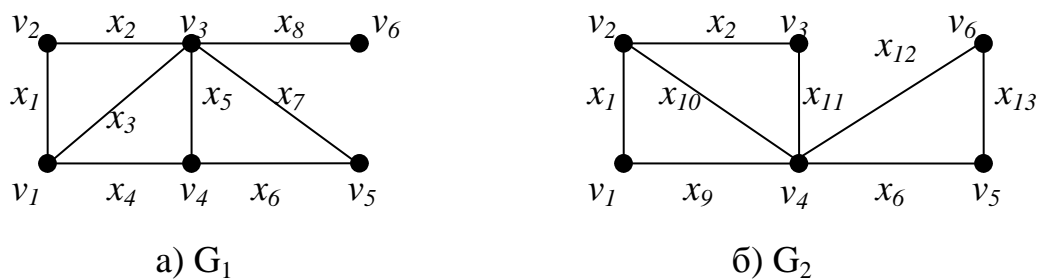
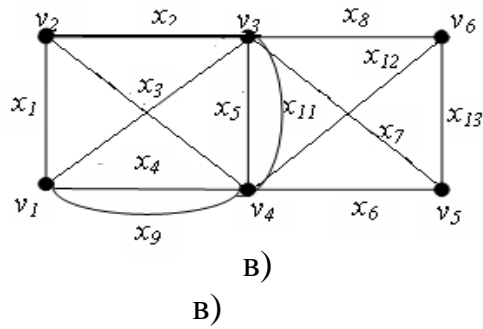


Рис.1.23. З'єднання двох графів

Визначення. Об'єднанням графів G_1 та G_2 , позначається як $G_1 \cup G_2$, називається такий граф $G_3=(V_1 \cup V_2, X_1 \cup X_2)$ множиною вершин якого є об'єднання вершин (Рис. 1.24. в)).



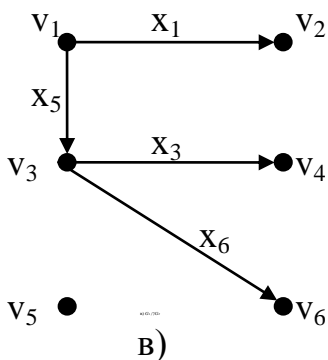
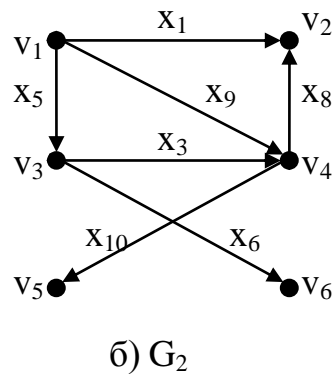
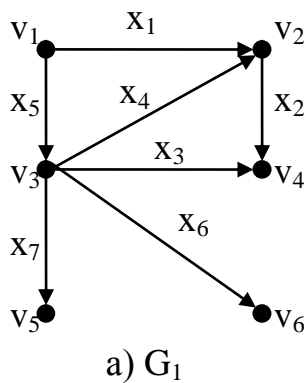


$A =$

		1	1	1		
1			1	1		
1	1			1	1	1
1	1	1			1	1
			1	1		1
			1	1	1	

Рис.1.24. а) граф G_1 , б) граф G_2 в) об'єднання графів, г) матриця суміжності об'єднаних графів

Визначення. Перетином двох графів G_1 і G_2 , позначається як $G_1 \cap G_2$, називається граф $G_4 = (V_1 \cap V_2, X_1 \cap X_2)$, множина вершин якого складається лише з тих вершин, які є одночасно в G_1 і G_2 , а множина ребер складається з тих ребер, які одночасно є і в G_1 і в G_2 . На рисунку 1.25.в) показано перетин двох орієнтованих графів.



$A =$

		1	1			
				1		1

Рис. 1.25. Перетин двох графів: а) граф G_1 , б) граф G_2 , в) граф $G_1 \cap G_2$, г) матриця суміжності графа $G_1 \cap G_2$.

Щоб визначити добуток $G_1 \times G_2$ двох графів розглянемо дві вершини $u = (u_1, u_2)$ і $v = (v_1, v_2)$ з $V = V_1 \times V_2$. Вершини u і v суміжні в $G_1 \times G_2$ тоді і тільки тоді, коли $[u_1 = v_1, \text{ і } u_2 \text{ adj } v_2]$ або $[u_2 = v_2 \text{ і } u_1 \text{ adj } v_1]$. Добуток графів $G_1 = P_2$ і $G_2 = P_3$ показано на рис. 1.26.

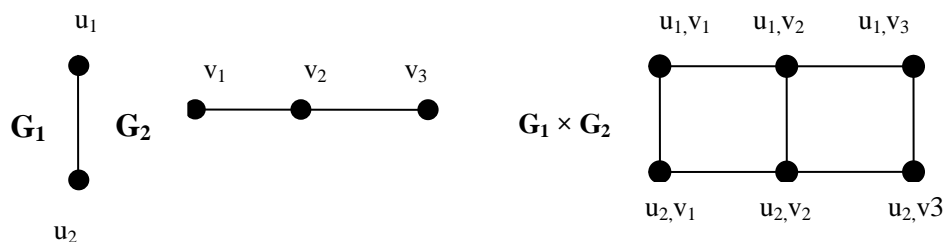


Рис1.26 Добуток двох графів

Nota Bene! Для побудови добутку двох графів беремо стільки копій графа G_2 , скільки існує вершин у графі G_1 .

Композиція $G = G_1[G_2]$ також має $V = V_1 \times V_2$ вершин і вершина $u = (u_1, u_2)$ і суміжна з $v = (v_1, v_2)$ з $V = V_1 \times V_2$ тоді і тільки тоді, коли $[u_1 \text{ adj } v_1]$ або $[$

$u_1 = v_1, \text{ і } u_2 \text{ adj } v_2]$. Для графів G_1, G_2 , (рис. 2.27), дві композиції $G_1[G_2]$ і $G_2[G_1]$, які, очевидно, не ізоморфні, показані на рис.2.28.

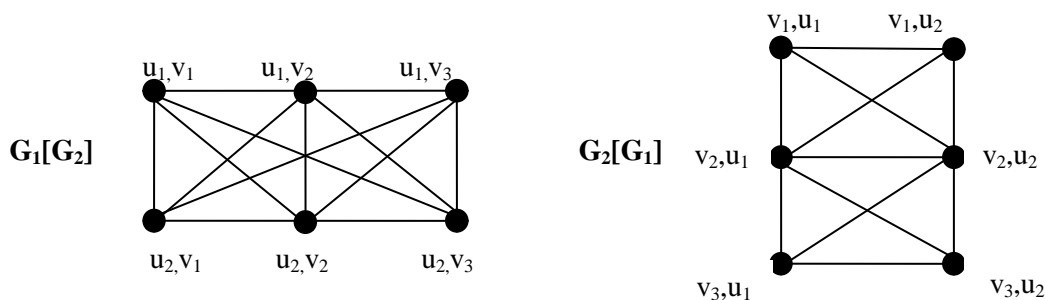


Рис. 1.27. Дві композиції графів

Якщо G_1, G_2 - це (v_1, x_1) і (v_2, x_2) - графи відповідно, то для кожної з визначених вище операцій можна знайти кількість вершин і число ребер у вихідному графі.

За відповідними формулами можна порахувати кількість вершин та ребер для операцій об'єднання, з'єднання та добутку.

Операція	Кількість вершин	Число ребер
Об'єднання $G_1 \cup G_2$	$v_1 + v_2$	$x_1 + x_2$
З'єднання $G_1 + G_2$	$v_1 + v_2$	$x_1 + x_2 + v_1 v_2$
Добуток $G_1 \times G_2$	$v_1 v_2$	$v_1 x_2 + v_2 x_1$

1.9. Блоки

В деяких ситуаціях дуже важливо, щоб граф залишався зв'язним, якщо видалити з нього будь-яку вершину або ребро. Тобто іноді треба мати більш ніж один шлях між кожною парою вершин графа для того, щоб підстрахуватись від можливих відмов. Наприклад, з Києва можна долетіти до Варшави навіть, якщо аеропорт у Львові закрито через Івано-Франківськ. Або треба прокласти такий залізничний шлях, щоб у час воєнних дій противник зруйнував не менш ніж дві станції. Аналогічно, треба прокласти з'єднання в інтегральній схемі або в мережах зв'язку таким чином, щоб при обриві будь-якого дроту або відмови з'єднання решта схеми продовжувала працювати. Всі ці приклади демонструють дві різні концепції: у випадку інтегральної схеми і мереж ми зацікавлені у збереженні зв'язності при видаленні ребра, а у випадку залізничного сполучення треба зберегти зв'язність при видаленні вершини.

Деякі зв'язні графи можна зробити незв'язними, видаливши одну вершину, яка називається точкою з'єднання. Виділення таких вершин сильно допомагає у вивченні структури зв'язного графа. Ребра з аналогічною властивістю називаються *мостами*. Частини графа разом з його точками з'єднання – це його блоки. Після визначення цих трьох понять введемо і вивчимо два нових графа – *граф блоків* і *граф його точок з'єднання*.

1.9.1. Точки з'єднання, мости та блоки

Визначення. Точкою з'єднання неорієнтованого графа називається вершина, видалення якої збільшує число компонентів зв'язності.

Визначення. Мостом неорієнтованого графа називається ребро, видалення якого збільшує число компонентів зв'язності графа. (separation edge)

Таким чином, якщо v – точка з'єднання зв'язного графа G , то граф $G - v$ не зв'язний.

Визначення. Неподільним графом називається зв'язний, непустий неорієнтований граф, що не має точок з'єднання.

Визначення. Блок графа – це його максимальний неподільний підграф. Якщо G – неподільний граф, то часто він сам називається блоком.

На рисунку 2.29. v , u – точки з'єднання, а w – ні, x – міст, а y – ні; окремо наведено чотири блоки графа G . Кожне ребро графа належить тільки одному з його блоків, також як і кожна вершина, яка не є ні ізольованою, ні точкою з'єднання. Ребра будь-якого простого циклу графа також належать тільки одному блоку.

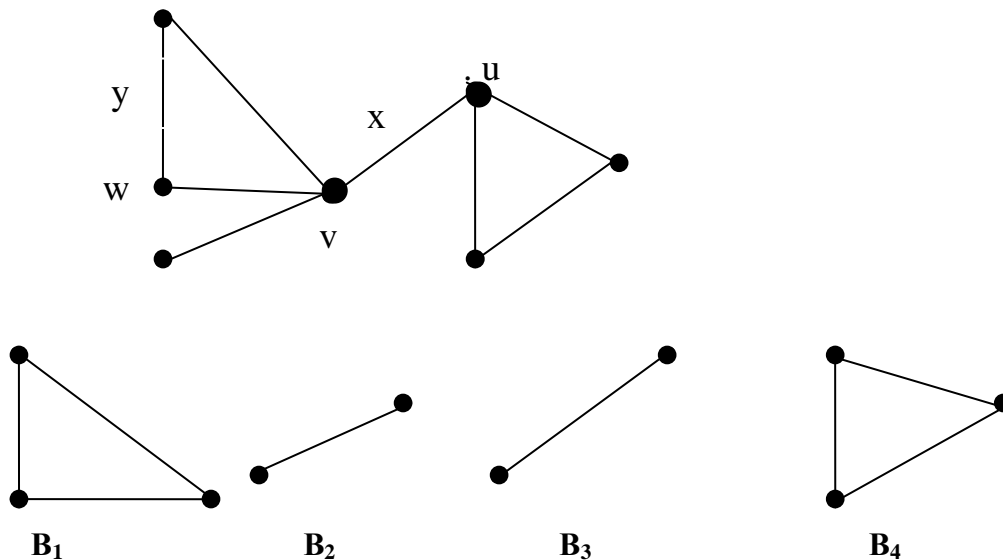


Рис.1.28. Граф та його блоки

У наведених нижче теоремах встановлюється декілька еквівалентних умов, що забезпечують існування у графа точки з'єднання і мосту та неподільності графа.

Т е о р е м а 1. Якщо v – точка з'єднання графа G , то

а) існують такі вершини u і w , відмінні від v , що v належить будь-якому $(u - w)$ ланцюгу;

б) існує розподіл множини вершин $V - \{v\}$ на такі дві підмножини U і W , що для будь-яких вершин $u \in U$ і $w \in W$ вершина v належить будь-якому простому $(u - w)$ ланцюгу.

Т е о р е м а 2: Якщо x – міст графа G , то

а) x – не належить жодному простому циклу графа G ;

б) у графа G існують такі вершини u і w , що ребро x належить будь-якому простому ланцюгу, що сполучає u і w ;

в) існує розподіл множини вершин V на такі дві підмножини U і W , що для будь-яких вершин $u \in U$ і $w \in W$ ребро x належить будь-якому простому $(u - w)$ ланцюгу.

1.9.2. Графи блоків і графи точок з'єднання

Відомі кілька графів перетинів, які можна одержати з графа G , і які представляють його структуру. *Граф перетинів* називається графом блоків

графа G і позначається через $B(G)$.

Блоки графа G відповідають вершинам графа $B(G)$ і дві вершини графа $B(G)$ суміжні тоді і тільки тоді, коли відповідні їм блоки графа G мають спільну точку з'єднання.

Граф, вершини якого відповідають точкам з'єднання графа G називається *графом точок з'єднання*.

Граф точок з'єднання позначається $C(G)$. Дві вершини графа $C(G)$ суміжні тоді і тільки тоді, коли відповідні їм точки з'єднання графа G

належать одному блоку.

На рис.1.29 а) зображено граф G , б) граф блоків $B(G)$, в) граф точок з'єднання.

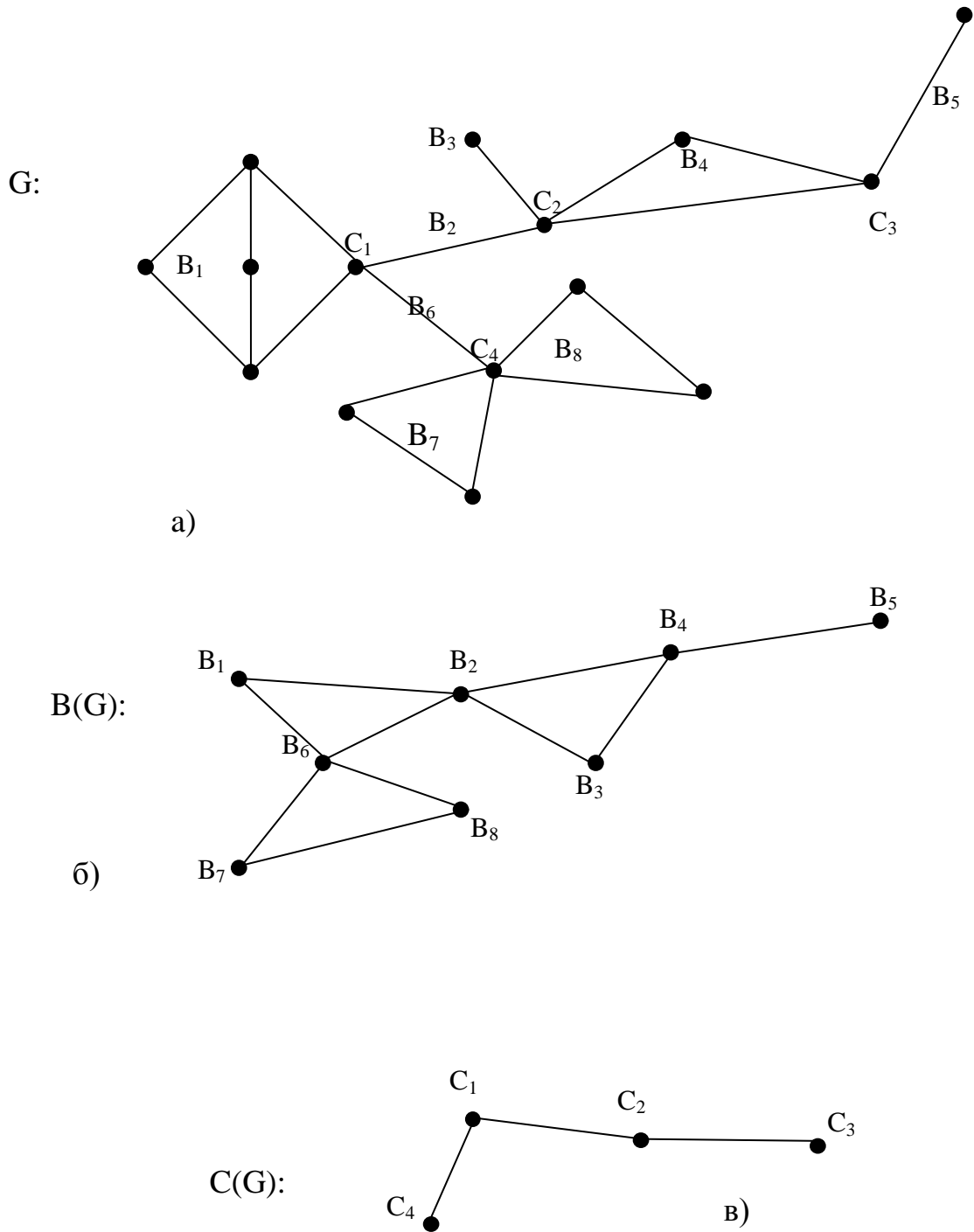


Рис.1.29. а) Граф G , б) граф блоків $B(G)$ і в) граф точок з'єднання

На рис. 1.30. показано граф блоків та точок з'єднання. В цьому графі вершинами є блоки та точки з'єднання, і кожен блок з'єдується з іншим через точку з'єднання.

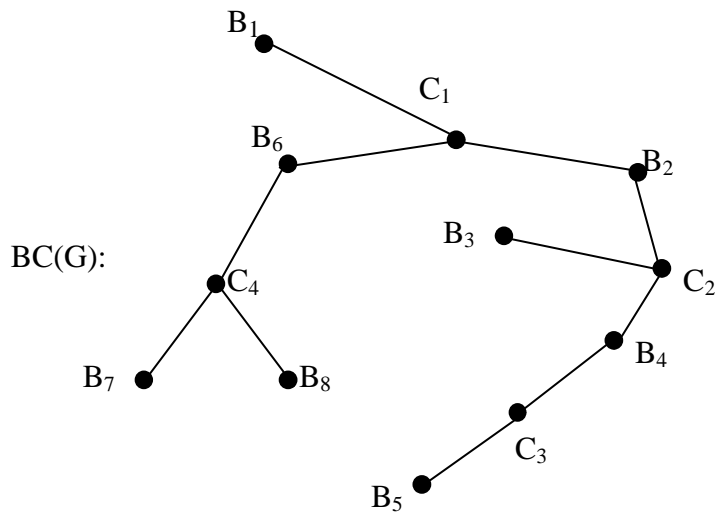


Рис. 1.30. Граф блоків і точок з'єднання

1.10. Обходи графів

1.10.1. Графи Ейлера

Як ми вже згадували в першому розділі, негативний розв'язок Ейлером задачі про кенігсберські мости в 1736 р. привів до першої публікації з теорії графів [10]. Задачу про обходи мостів Ейлер узагальнив і ми отримали наступну задачу теорії графів.

На яких графах можна знайти цикл, що містить всі ребра графа, причому кожне ребро в точності по одному разу?

Очевидно, не всі графи мають ейлерові цикли, але якщо ейлерів цикл існує, то це означає, що, слідую уздовж цього циклу, можна накреслити граф на папері, не відриваючи від нього олівця.

Граф на рис. 1.31. має наступний ейлерів цикл (починаємо з вершини v_1). Напрямок руху по кожному ребру показано на рисунку стрілками.

Граф, у якому це можливо, називається *ейлеровим*. Граф Ейлера має ейлерів цикл – замкнутий ланцюг, що містить всі вершини і всі ребра. Ейлерів граф повинен бути зв'язним.

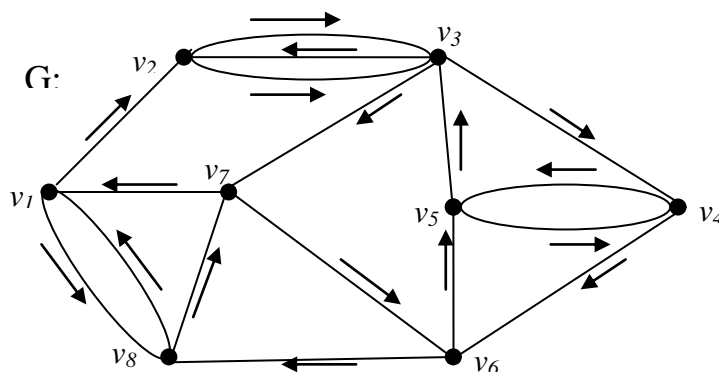


Рис.1.31. Эйлерів цикл графа з вершини v_1 .

Як і в задачі про кенігсберські мости, ясно, що кожна ейлерова лінія повинна входити в кожную вершину і виходити з неї одне і те ж число разів, тобто степені вершин графа повинні бути парними.

Основна теорема існування ейлерового циклу формулюється таким чином.

Т е о р е м а (а). Зв'язний неорієнтований граф містить ейлерів цикл тоді і тільки тоді, коли степені всіх його вершин парні.

Для орієнтованого графа справедлива теорема, повністю аналогічна попередній, а саме

Т е о р е м а (б). Зв'язний орієнтований граф містить ейлерів цикл (ейлерів ланцюг) тоді і тільки тоді, коли всі напівстепені заходу $d_i(v_i)$ і напівстепені виходу $d_o(v_i)$ задовольняють умовам:

Флері [3] дав дуже простий алгоритм побудови ейлерового циклу в неорієнтованому графі (якщо такий цикл існує). Цей алгоритм легко можна поширити на орієнтовані графи і полягає він у наступному: починаємо з деякої вершини v і кожний раз викреслюємо пройдене ребро. Не проходимо по ребру, якщо видалення цього ребра призводить до поділу графа на дві зв'язні компоненти (не рахуючи ізольованих вершин).

Досі ми говорили про цикл, що проходить по всім ребрам в точності по одному разу; але існує ланцюг, що має ті ж властивості.

Т е о р е м а. Для того, щоб зв'язний неорієнтований граф мав відкритий ланцюг $Z(A,B)$, що містить всі його ребра по одному разу, необхідно і достатньо, щоб A і B були єдиними непарними вершинами в цьому графі.

На рис. 2.33. зображено ланцюг $FCDBAEC$ графа G . Цей граф має дві непарні вершини F і C . Виникає питання, а яке найменше число ланцюгів таких, що ніякі два з них не мають спільних ребер і всі вони разом покривають всі ребра графа, існує в довільному графі?

Як ми вже знаємо з теореми Ейлера, число непарних вершин у графі – парне, нехай – $2k$.

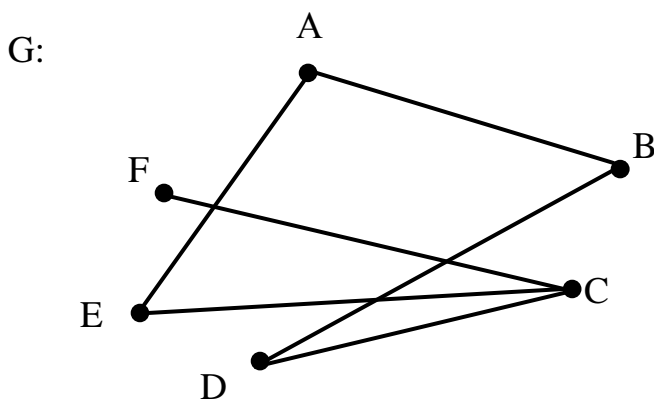


Рис. 1.32. Граф G має відкритий ейлерів ланцюг

Т е о р е м а. В будь-якому зв'язному графі з $2k$ непарними вершинами є сімейство з k ланцюгів, які в сукупності містять всі ребра графа в точності по одному разу.

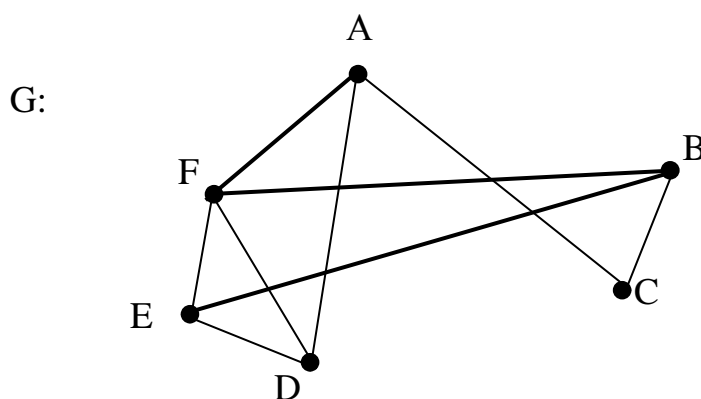


Рис. 1. 34. Граф G має 2 ланцюга Ейлера

Для прикладу візьмемо граф G , зображений на рис. 1.34. Він має 4 непарні вершини A, B, D, E і покривається двома ланцюгами $EBFA, BCADFED$.

Нехай ребрам графа допишемо позитивні ваги. Потрібно знайти цикл, що проходить через кожне ребро графа принаймні один раз і такий, що для нього загальна вага (а саме сума величин $n_j c(x_j)$), де число n_j показує скільки разів проходило ребро x_j , а $c(x_j)$ – вага ребра) мінімальна. Очевидно, що якщо граф має ейлерів цикл, то будь-який такий цикл буде оптимальним, оскільки кожне ребро проходиться тільки один раз і вага цього циклу дорівнює тоді $\sum_{j=1}^m c(x_j)$.

Сформульована задача називається задачею *китайського листоніші*, і її розв'язок має багато застосувань, як наприклад:

Доставка молока або пошти (знаходження маршруту, який мінімізує загальний кілометраж, або вартість, або час);

Перевірка електричних, телефонних або залізничних ліній.

Інші застосування можуть бути пов'язані з розкиданням суміші піску і солі на головних дорогах взимку для запобігання обмерзання, з найкращим методом роботи автоматичних вентиляційних пристроїв, з прибиранням приміщень і коридорів у великих установах і навіть з найкращим маршрутом огляду музею!

1.10.2. Гамільтонови графи

У 1859 р. відомий ірландський математик сер Вільям Роуен Гамільтон випустив в продаж своєрідну головоломку. Її основною частиною був правильний додекаедр, зроблений

з дерева. Це один з так званих правильних багатогранників: його гранями служать 12 правильних п'ятикутників, причому в кожній з 20 його вершин сходиться по три ребра.

Кожна вершина гамільтонова додекаедра була позначена назвою одного з великих міст - Брюссель, Кантон, Делі, Франкфурт і т.д. Завдання полягало в знаходженні шляху уздовж ребер додекаедра, що проходить через кожне місто в точності по одному разу; щоб зробити задачу ще більш цікавою, порядок проходження декількох перших міст встановлювався заздалегідь.

Для того, щоб легше було запам'ятовувати шлях, в кожну вершину був вколотий цвях з великою голівкою, так що навколо цих цвяхів могла витися мотузка, що вказує пройдений шлях. Однак додекаедр був занадто громіздким, і Гамільтон запропонував інший варіант своєї гри, де багатогранник замінювався плоским графом, ізоморфним графу, утвореного ребрами додекаедра (рис. 1.35).

Головоломку давно забули, а ось **гамільтоновим циклом** і зараз називають *цикл, що проходить через кожну вершину графа точно по одному разу*

Гамільтонові графи слугують моделлю при складанні розкладу руху поїздів, для телекомунікаційних мереж і так далі.

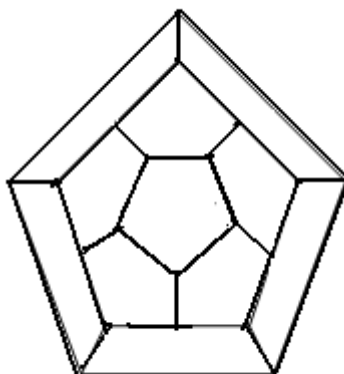


Рис.1.35. Гамільтоновий цикл для додекаедра

Гамільтонів граф не покриває точно всіх ребер, адже у кожній вершині він проходить по двом ребрам. Цикл, зображений на рис.1.35. називається гамільтоновим циклом для додекаедра.

Таким чином, є відома аналогія між ейлеровим і гамільтоновим циклом. Перший проходить один раз по *кожному ребру*, а другий - через *кожну вершину*.

Але не зважаючи на таку схожість, ці задачі абсолютно різної складності. Для ейлерового графа досить перевірити, чи є всі його вершини парними. На відміну від задачі Ейлера, простого критерію гамільтоновості графа поки що невідомо.

Існують, однак, достатні умови існування гамільтонових циклів.

Теорема (Дірака): Нехай G неорієнтований граф; $\deg v_i = \delta$ – мінімальна степінь його вершин. Якщо $n \geq 3$ і $\delta \geq n/2$, то граф G – *гамільтонів граф*.

Теорема (Оре[8]): Якщо $n \geq 3$ і $\deg v + \deg u \geq n$ для будь-яких несуміжних вершин u та v , то неорієнтований граф G – *гамільтонів*.

Теорема (Гуйя-Урі): Оргзв'язний граф має гамільтоновий цикл, якщо для будь-якої його вершини v $\deg^{in} v \geq n/2$, $\deg^{out} v \geq n/2$ для $n \geq [2]$.

Орграф називається *напівгамільтоновим*, якщо він має гамільтоновий ланцюг.

Для гамільтонового графа досі ще не знайдено такого загального критерію або алгебраїчного методу, що дає відповідь на питання, існує чи ні в довільному графі G гамільтоновий цикл.

З алгебраїчних методів визначення гамільтонового циклу найбільш прийнятними є спосіб Робертса і Флореса [13], який не висуває особливі вимоги до пам'яті комп'ютера, але час роботи цього алгоритму залежить експоненціально від числа вершин у графі і неявного методу перебору, який може бути використаний для знаходження гамільтонового циклу в дуже великих графах.

Кожен гамільтоновий граф двозв'язний.

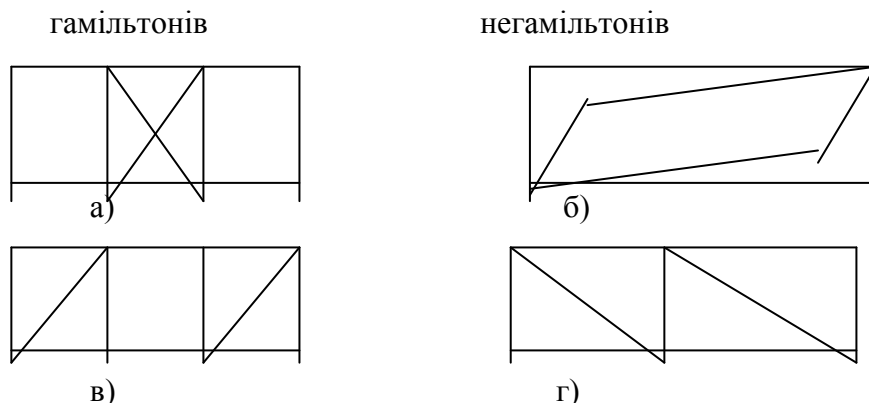


Рис.1.36. а) гамільтонів і ейлерів, б) негамільтонів, але ейлерів, в) гамільтонів не ейлерів, г) негамільтонів і не ейлерів [4].

1.10.3. Зв'язок між ейлеровими і гамільтоновими циклами

Гамільтонів цикл у графі був визначений як простий цикл, що проходить (один і тільки один раз) через кожну вершину графа. Не дивно, що подвійність між ейлеровими і гамільтоновими циклами (заміна вершини на ребро і навпаки) призводить до тісного зв'язку між цими двома поняттями в застосуванні до неорієнтованого графа і відповідного йому *реберного* графа.

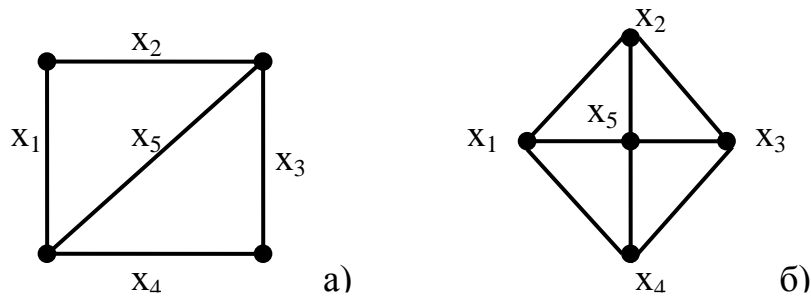


Рис.1.37. а) граф G , б) реберний граф $L(G)$

Визначення. Реберний граф $L(G)$ графа G має стільки ж вершин, скільки ребер у графа G . Ребро між двома вершинами графа $L(G)$ існує тоді і тільки тоді, коли ребра графа G , які відповідають цим двом вершинам, суміжні (тобто інцидентні одній і тій ж вершині графа G).

Наприклад, для графа на рис. 1.37.а), граф $L(G)$ зображено на рис. 1.37.б).

Легко можна показати, що вірні два наступні твердження про взаємовідносини між ейлеровими і гамільтоновими циклами.

1) Якщо граф G має ейлерів цикл, то його реберний граф $L(G)$ має як ейлерів так і гамільтонів цикли.

2) Якщо граф G має гамільтонів цикл, то його реберний граф $L(G)$ також має гамільтонів цикл.

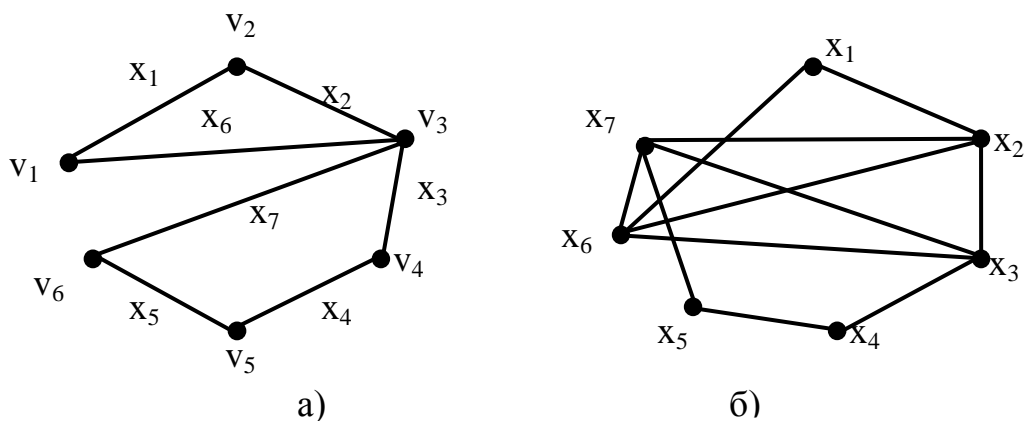


Рис. 1.38. а) Граф G , б) реберний граф $L(G)$

Зворотнє твердження невірне. Для графа, зображеного на рис. 1.38.а), степені всіх його вершин парні і тому існує ейлерів цикл. Таким чином, граф $L(G)$ також має ейлерів цикл (оскільки степені всіх його вершин знову парні) і, крім того, має гамільтонів цикл, що задається послідовністю вершин $x_1, x_2, x_3, x_4, x_5, x_7, x_6, x_1$.

Задача з'ясування чи існує в даному неорієнтованому графі G гамільтонів цикл, є важливою задачею теорії графів як з практичної так і з теоретичної точки зору. Найбільш відома задача в теорії графів – це задача комівояжера.

Нехай є n міст, відстані між ними відомі. Треба відвідати всі ці міста по одному разу і повернутися в те, звідки почали рух. Крім того, треба вибрати такий маршрут руху, щоб пройдений шлях був мінімальним.

Якщо представити міста у вигляді вершин графа, а ребра графа – це відстані між містами, то пошук мінімального шляху означає пошук гамільтонового циклу. Алгоритми розв'язку задачі комівояжера та її варіантів мають багато практичних застосувань в різних областях людської діяльності. Але досі ще немає алгоритму для швидкого розв'язання цієї задачі.

Наприклад, грузовик виїжджає з бази та виконує доставку товарів даному числу споживачів і повертається назад на базу. Вартість перевезень пропорційна пройдений відстані грузовика. При заданій матриці відстаней між споживачами маршрут з найменшими транспортними затратами можна отримати як розв'язок відповідної задачі комівояжера.

1.11. Древа

Існує один простий і важливий вид графів, якому різні автори дали однакову назву, це – *древа*. Древа важливі не тільки тому, що вони знаходять застосування в різних областях знання (наприклад задачі сортування), а в силу особливого становища їх у самій теорії графів. Останнє викликане простотою побудови дерев.

Поняття дерева як математичного об'єкта було вперше запропоновано Кірхгофом у зв'язку з визначенням фундаментальних циклів, що застосовуються при аналізі електричних ланцюгів. Через деякий час, незалежно від Кірхгофа, Келі [15] знову ввів поняття дерева і отримав більшу частину перших результатів в області дослідження властивостей дерев.

Область застосування дерев обширна. Це, наприклад, і інформатика, і біологія, і менеджмент.

Опис дерев

Граф називається *ациклічним*, якщо в ньому нема циклів. Приведемо декілька визначень дерев.

Можна сформулювати декілька необхідних та достатніх умов при яких G є деревом:

- Будь-яка пара вершин в G з'єднана єдиним ланцюгом;
- G зв'язний і $m = n - 1$;
- G зв'язний, а видалення хоча б одного його ребра порушує зв'язність графа;
- G ациклічний, але, якщо додати хоча б одне ребро, то в G з'явиться цикл.

Отже можна дати таке визначення дерева:

Визначення. Неорієнтованим деревом називається

- а) зв'язний ациклічний граф, або
- б) зв'язний граф, що містить n вершин і $n-1$ ребро, або

в) граф, у якому кожна пара вершин з'єднана одним і тільки одним простим ланцюгом.

Т е о р е м а . Якщо граф G дерево, то наступні твердження еквівалентні:

- 1) G – ациклічний граф, і якщо любую пару несуміжних вершин з'єднати ребром x , то в графі $G + x$ буде точно один простий цикл;
- 2) G – зв'язний граф, відмінний від K_v для $v \geq 3$, і якщо будь-яку пару несуміжних вершин з'єднати ребром x , то в графі $G + x$ буде точно один простий цикл.

Якщо $G = (V, X)$ неорієнтований граф з n вершинами, то *остовним деревом* (або *остовом*) графа G називається всякий остовний підграф графа G , який є деревом. Наприклад, якщо G – граф (рис. 2.39.а)), то граф на рис. 2.39.б) є остовним деревом графа G .

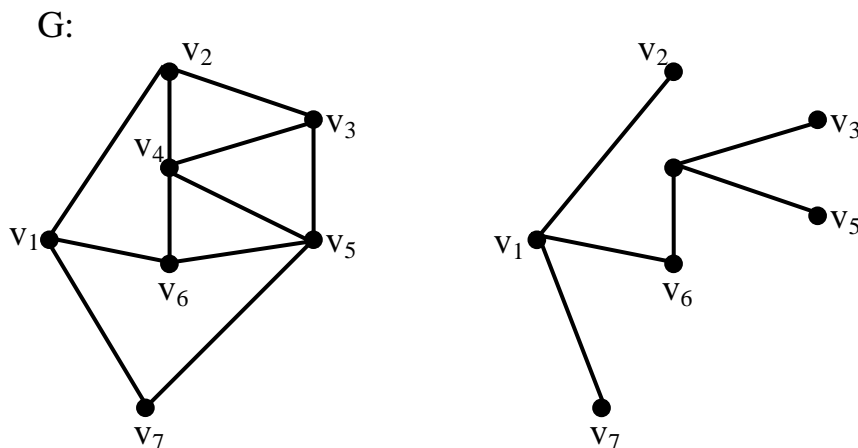


Рис. 1.39.а) Граф G , б) остов графа G

Орієнтоване дерево визначається аналогічно.

В и з н а ч е н н я . Орієнтоване дерево – це орієнтований граф без циклів, у якому напівстепінь заходу кожної вершини, за винятком однієї (наприклад v_1), дорівнює одиниці, а напівстепінь заходу вершини v_1 , яку будемо називати коренем цього дерева) дорівнює нулю.

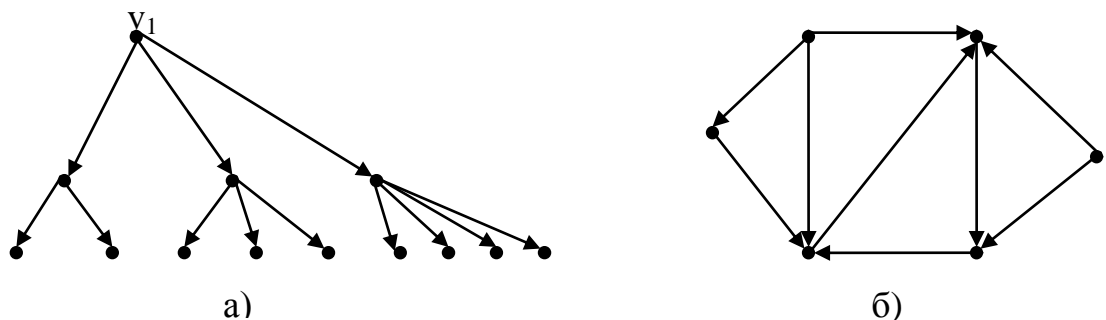


Рис.1.40.а) орієнтоване дерево з коренем в вершині v_1 ,
б) граф, що не має остовного орієнтованого дерева

На рис.1.40.а) показано граф, який є орієнтованим деревом з коренем у вершині v_1 . Очевидно, що не всякий орієнтований граф має остовне орієнтоване дерево. На рис. 1.40.б) зображено такий граф.

Приклад 1.1. Доведіть, використовуючи індукцію по числу вершин, що для дерева T з n вершинами та m ребрами виконується співвідношення $m = n - 1$.

Розв'язок: Оскільки дерево з єдиною вершиною взагалі не має ребер, то наша твердження справедливо для $n = 1$. Розглянемо тепер дерево T з n вершинами ($n > 1$) (і m ребрами). Припустимо, що будь-яке дерево з $k < n$ вершинами має $k - 1$ ребро. Видалимо ребро з дерева. З третьої властивості випливає, що після цієї процедури дерево T перетвориться в незв'язний граф. Одержим дві компоненти зв'язності, жодна з яких не має циклів. В протилежному випадку вихідний граф теж містив би цикли і не міг би бути деревом. Таким чином, отримані компоненти зв'язності – теж дерева.

Позначимо дерева як T_1 і T_2 . Нехай n_1 – кількість вершин дерева T_1 , а n_2 – дерева T_2 . Оскільки $n_1 + n_2 = n$ то $n_1 < n$ і $n_2 < n$. Оскільки, по визначенню індукції, дерево T_1 має $n_1 - 1$ ребро, а дерево T_2 – $n_2 - 1$ ребро, то вихідне дерево T має $(n_1 - 1) + (n_2 - 1) + 1 = n - 1$ ребро, що і треба було довести.

1.11.1.Дерева на множині вершин

Нехай множина V містить n вершин, які пронумеровані порядковими номерами від 1 до n , тобто $V = \{1, 2, \dots, n\}$. Зв'язавши ці вершини $n-1$ ребрами так, щоб були відсутні цикли, отримаємо деяке дерево, що покриває дану множину n вершин. При $n=2$ таке дерево єдине і воно складається з однієї гілки. Зі збільшенням n число різних дерев t_n швидко зростає і виражається співвідношенням $t_n = n^{n-2}$

Будь-яке дерево можна розглядати як кореневе дерево, в якому деяка обрана вершина v_0 називається коренем. Вершини, суміжні з коренем дерева, можна розглядати як коріння субдерев, які "виростають" з цих вершин і є частинами вихідного дерева, так званими *факторами*. Взагалі кажучи, кожна вершина графа грає роль кореня деякого фактора, причому кінцеві вершини – це коріння тривіальних дерев, що складаються з єдиної вершини і не містять ребер.

Визначення. Вагою вершини називається загальна кількість вершин фактора.

На кожному рівні фактори розташовуються в порядку зростання ваг їх коріння. На рис. 1.41.а) кореневе дерево зображено у стандартному вигляді і вказані ваги всіх його вершин.

Вага кореня дерева дорівнює числу всіх його вершин.

Кореневе дерево можна визначити впорядкованою послідовністю $\beta(t)$ ваг його вершин. На першому місці стоїть вага кореня дерева, а потім відповідна послідовність для факторів у порядку зростання ваг їх коріння.

Ваги кінцевих вершин рівні одиниці.

У свою чергу, кожна така послідовність будується за принципом: на першому місці стоїть вага кореня фактора, а потім слідує послідовності для факторів даного фактору і т.д. Так, для дерева (Рис.1.41а) з позначенням ваг вершин маємо:

$$\beta(t) = (17, 1, 4, 1, 1, 1, 11, 4, 1, 2, 1, 6, 1, 1, 3, 1, 1).$$

Кількість членів послідовності $\beta(t)$ дорівнює числу вершин дерева. Перенесення кореня в іншу вершину приводить до іншого кореневого дерева, а значить і до іншої послідовності. Різним кореневим деревам відповідають і різні послідовності $\beta(t)$. Достатнім критерієм ідентичності кореневих дерев є співпадання відповідних їм послідовностей.

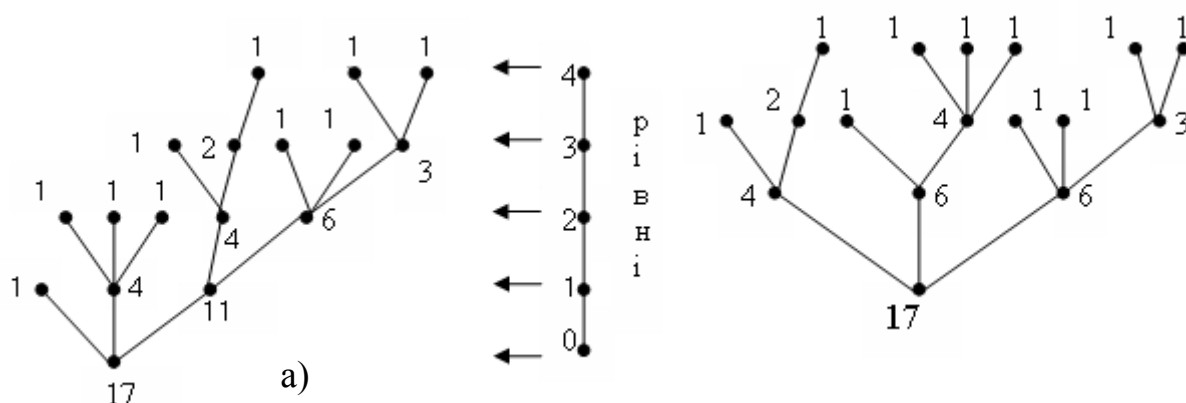


Рис. 1.41. а) стандартне представлення кореневого дерева, б) коренева форма

З іншого боку можна побудувати дерево за його послідовністю $\beta(t)$. Перший член відповідає кореню дерева. Потім $\beta(t)$ розбивається на послідовності факторів так, що кожна з них починається членом, не меншим, ніж попередня, тобто.

$$\beta(t_1) = (1); \quad \beta(t_2) = (4, 1, 1, 1); \quad \beta(t_3) = (11, 4, 1, 2, 1, 6, 1, 1, 3, 1, 1).$$

З кожної такої послідовності видаляємо перші члени і сполучаємо відповідні їм вершини з коренем. Потім робимо аналогічно до тих пір, поки не будуть вичерпані всі члени. Так, після побудови коренів факторів першого рівня послідовність $\beta(t_1)$ вичерпується, а $\beta(t_2)$ і $\beta(t_3)$ розбиваються на послідовності:

$$\begin{aligned} \beta(t_{21}) &= (1); \quad \beta(t_{22}) = (1); \quad \beta(t_{23}) = (1); \\ \beta(t_{31}) &= (4, 1, 2, 1); \quad \beta(t_{32}) = (6, 1, 1, 3, 1, 1); \end{aligned}$$

Перші члени цих послідовностей відповідають корінням факторів другого рівня, які з'єднуються ребрами з тими коренями, з яких вони виростають і т.д., і т.д.

Як ми вже згадували, область застосування дерев з коренем обширна. Це, наприклад і інформатика, і біологія, і менеджмент. В інформатиці найбільш важливими є двійкові або бінарні дерева з коренем. Двійкове дерево відрізняється від решти тим, що його кожна

вершина має не більш ніж 2 – х *синів*. (вершина, що лежить під даною вершиною). З другої боку, вершина, що стоїть перед сином, називається *батьком*.

В двійковому дереві з коренем вниз від кожної вершини йде не більш ніж два ребра.

Таким чином, можна сказати, що кожній вершині двійкового дерева відповідає не більш ніж два піддерева, які називають лівим та правим піддеревами цієї вершини. Якщо у деякої вершини дерева відсутній спадкоємець зліва, то її ліве піддерево називають нульовим деревом. Аналогічно з правим під деревом.

1.11.2. Ідентифікація дерев

У багатьох випадках важливо розрізняти тільки неізоморфні дерева.

Ізоморфізм – це відношення еквівалентності на множині різних дерев, яке розбиває цю множину на непересічні класи не ізоморфних дерев. Будь-яке дерево з даного класу може виступати його представником. Але при різних способах задання і накреслення дерев встановити їх ізоморфізм безпосереднім порівнянням не так просто.

При ідентифікації дерев зазвичай використовують якусь канонічну форму, в якій ізоморфні дерева не розрізнити, а не ізоморфні отримують різні уявлення. Зручною для цих цілей є коренева форма зображення дерева.

Одна з стандартних процедур вибору кореня полягає в наступному: з дерева видаляються усі кінцеві вершини і ребра, потім в отриманому дереві знову видаляються усі кінцеві вершини і ребра, і так до тих пір, поки вихідне дерево не скоротиться до єдиної вершини або ребра.

У першому випадку вершина, яка залишилася, називається *центром* і вибирається як корінь. У другому – дві вершини і ребро, що їх зв'язує, утворюють *біцентр*. При цьому за корінь приймається та вершина, з якої виростає дерево з меншим числом вершин. Дерево, зображене на рис.1.42.а) має центр і його кореневій формі відповідає послідовність:

$$\beta(t) = (17, 4, 1, 2, 1, 6, 1, 4, 1, 1, 1, 6, 1, 1, 3, 1, 1).$$

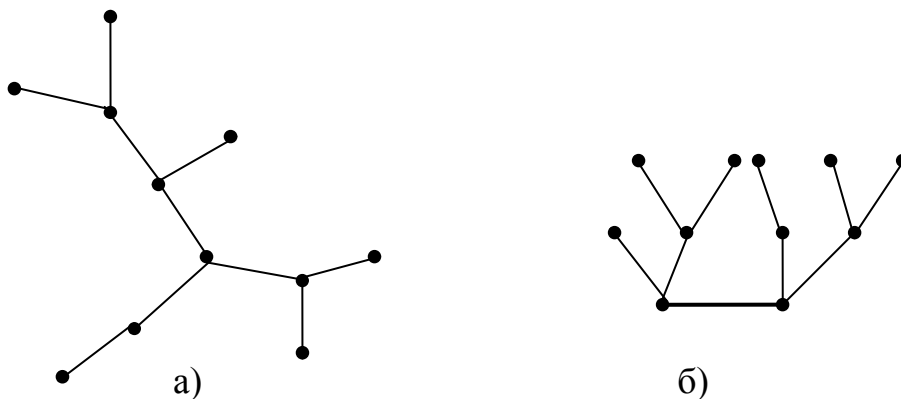


Рис.1.42. а) Дерево, б) його біцентр

На рис.1.42.а) зображено дерево, яке має біцентр, а його коренева форма характеризується послідовністю

$$\beta(t) = (11, 1, 3, 1, 1, 5, 2, 1, 3, 1, 1).$$

Інший спосіб вибору кореня дерева для його кореневої форми заснований на понятті висоти вершини.

З кожною вершиною дерева зв'язані розгалуження, що представляють собою частини дерева, причому число ребер в розгалуженні характеризує його довжину.

Визначення. Висота вершини – це число, рівне найбільшій довжині пов'язаних з нею розгалужень.

Визначення. Центроїдом називається вершина з найменшою висотою, яка вибирається в якості кореня.

Якщо є дві таких вершини, то вони разом з ребром, що їх з'єднує, утворюють біцентроїд. При цьому за корінь приймається та вершина, з якої виростає дерево з меншим числом вершин.

Кінцеві вершини мають по одному розгалуженню, яке містить всі ребра дерева, і, отже, в дереві з p вершинами висоти кінцевих точок дорівнюють $p-1$. На рис. 1.44. вказані два неізоморфні дерева з десятима вершинами і вказані висоти вершин. Перше з них має біцентр і центроїд, які не збігаються, друге має центр і біцентроїд.

Визначення. Два дерева є ізоморфними, якщо послідовності для їх корневих форм збігаються (зрозуміло, обидві кореневі форми повинні бути утворені з коренем в центрі або в центроїді).

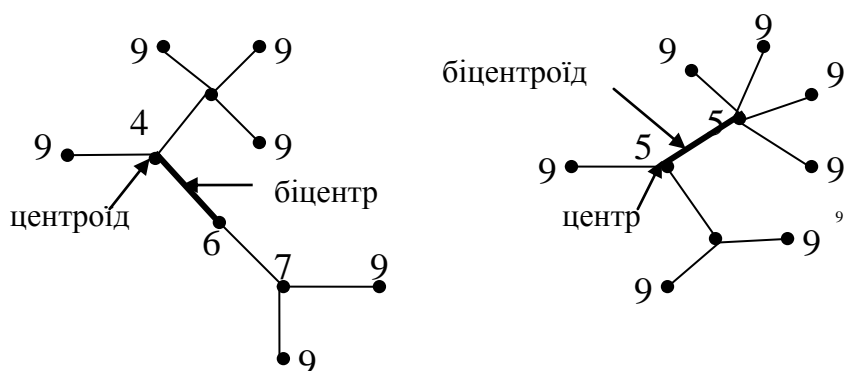


Рис.1.43. Не ізоморфні дерева (цифри означають висоти вершин)

Таким чином ідентифікація дерев зводиться до порівняння відповідних їм послідовностей для корневих форм.

1.11.3. Дерева графа

В деяких ситуаціях виникає необхідність в визначенні всіх остовних дерев неорієнтованого графа G .

Будемо називати каркасом або остовом зв'язного графа будь-яке дерево, що зв'язує всі його вершини і яке утворено із ребер цього графа.

Два дерева називаються *різними*, якщо вони відрізняються хоча б одним ребром.

Існує простий спосіб визначення кількості різних каркасів графа без петель (або мультиграфа) з p вершинами. Для цього необхідно записати квадратну матрицю p -того порядку, на головній діагоналі якої розташовані степені вершин, а ij - і ji - елементи рівні взятому зі знаком мінус числу ребер, що зв'язують вершини i та j . Якщо обчислимо будь-який з головних мінорів цієї матриці, отримаємо шукане число дерев (каркасів) графа. Наприклад, для графа (Рис.1.45.) маємо:

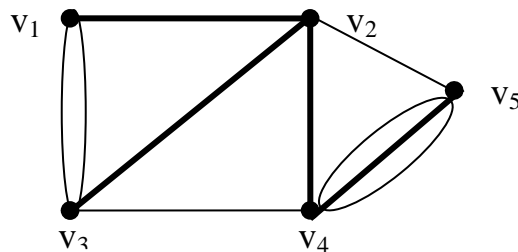


Рис. 1.44. Зв'язний граф і одне з його дерев

Одне з 76 дерев графа зображено на рис. 2.45 наведеними лініями. Приведений спосіб визначення числа дерев графа відомий як теорема Трента.

$$A = \begin{vmatrix} 3 & -1 & -2 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -2 & -1 & 4 & -1 & 0 \\ 0 & -1 & -1 & 5 & -3 \\ 0 & -1 & 0 & -3 & 4 \end{vmatrix} \quad \Delta_{22} = \begin{vmatrix} 3 & -2 & 0 & 0 \\ -2 & 4 & -1 & 0 \\ 0 & -1 & 5 & -3 \\ 0 & 0 & -3 & 4 \end{vmatrix} = 76$$

Запитання для самоконтролю

1. Який спосіб не є способом задання графа?
2. Чи можна показати, що будь-який повний симетричний граф містить гамільтонів цикл?
3. Довести, що неорієнтований зв'язний граф залишається зв'язним після видалення деякого ребра тоді і тільки тоді, коли це ребро належить будь-якому циклу даного графа.
4. Довести для неорієнтованого графа, що число вершин з непарною степенню парне.
5. Нащо вказує число інцидентних ребер графа?
6. Як називається граф G , у якому існує маршрут між будь-якою парою вершин?

7. Клас графів утворюють дерева з однією позначеною вершиною, яка називається ...?

Комп'ютерний практикум №1

1.1. Поясніть, чому сума степенів всіх вершин простого графа G співпадає з подвоєнною кількістю ребер. Цей факт називають *лемою* про естафету.

Використовуючи цю лему, покажіть, що будь-якому повному графі K_n з n вершинами є рівно $n(n-1)/2$ ребер.

Для яких значень n граф K_n буде ейлеровим.

1.2. Опираючись на принцип Дирихле, докажіть, що якщо простий граф має більш ніж одну вершину, то у нього знайдеться по крайній мірі, дві вершини однакової степені.

1.3. Нарисуйте граф, чия матриця суміжності має вигляд

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

Опишіть матрицю суміжності повного графа K_n .

1.4. Треба підібрати підходящі позначення вершин, для кожного з графів на рис. 2.45. і побудувати відповідну матрицю суміжності. із перелічених нижче.

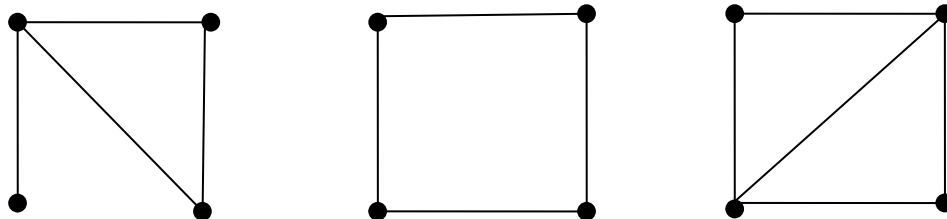


Рис.1.45

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

а)

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

б)

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

в)

1.5. Які з графів на рис.1.47 можуть бути підграфами графа з вправи 1.3?

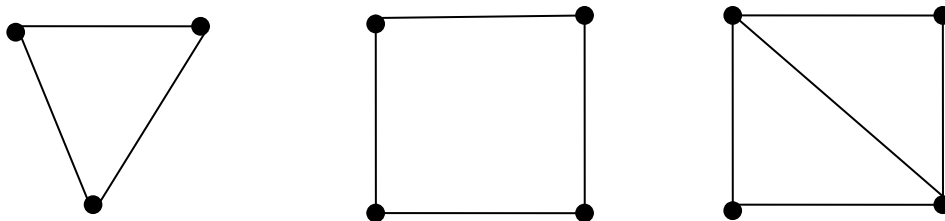


Рис.1.46. Кандидати в підграфи

1.6. Найдіть гамільтонові цикли в графі на рис.1.48.

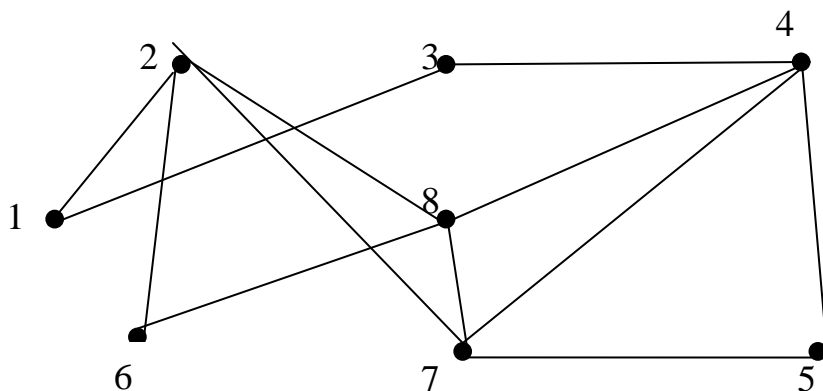


Рис.1.47

Найдіть в ньому цикли довжиною 3, 4, 5, 6 і 7.

1.7. На рис.1.49 зображено граф Петерсона P .

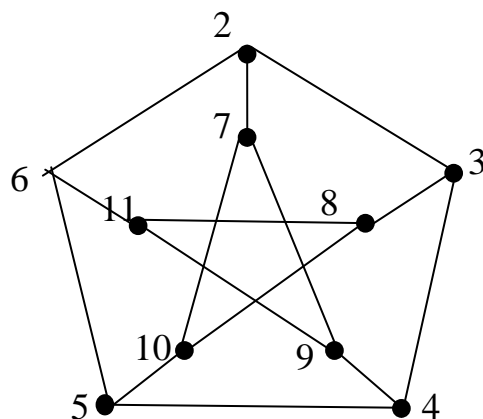


Рис.1.48. Граф Петерсона

Найдіть в ньому цикл довжиною 9. Покажіть, що P не є гамільтоновим графом.

1.8. Чи будуть графи, які задаються наступними матрицями суміжності, деревами?

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

а)

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

б)

1.9. Відомо, Що дерево має три вершини степені 3 і чотири вершини степені 2. Решта вершин дерева мають степінь 1. Скільки вершин степені 1 є у дерева T?(Вказівка. Позначте число вершин дерева T через і застосуйте лему про естафету).

1.10. Найдіть мінімальне остовне дерево графа, зображеного на рис.1.50.

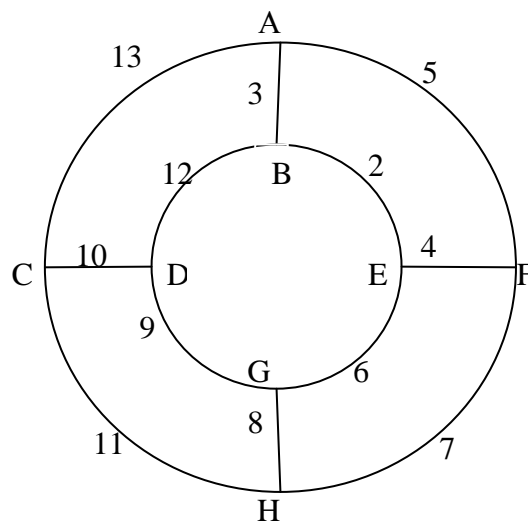


Рис. 1.49.

1.11. Глибина вершини v дерева з корнем T визначається як довжина єдиного шляху від неї до кореня дерева. Глибина графа T – це максимальна глибина його вершин.

Накресліть наступні дерева:

- (i) дерево з корнем глибини з шістьма вершинами;
- (ii) повне двійкове дерево з корнем глибини 2;
- (iii) дерево з корнем глибини 3, кожна вершина глибини i ($i \geq 0$) якого має $(i + 1)$ сина.

РОЗДІЛ 2. АЛГОРИТМИ НА ГРАФАХ

Впровадження комп'ютерних технологій поставило і перед всією математикою, і перед теорією графів проблему знаходження не довільних алгоритмів, які б дозволяли вирішувати ті, чи інші класи задач, а таких алгоритмів, які допускали б практичну реалізацію з використанням сучасних комп'ютерних технологій. Таким чином виникла проблема практичної вирішуваності задач: знайти ефективний або хоча б достатньо простий в практично важливих випадках алгоритм розв'язку задачі.

В цьому розділі ми розглянемо декілька основних алгоритмів обробки графів.

Почнемо з розгляду задачі побудови остовного дерева мінімальної вартості або SST (shortest spanning tree, або MST - minimal spanning tree).

Багато дослідників працювали над розробленням алгоритму даної проблеми. Ми розглянемо найбільш відомі з них: алгоритм Р. Пріма та Й. Крускала.

Перший і самий простий алгоритм даної проблеми зв'язаний з ім'ям Боруевки, який у 1926 р. представив розв'язок даної задачі.

Другий відомий алгоритм побудови мінімального остовного дерева розробив *Войтех Ярнік* у 1930р. Незалежно від Ярніка в 1956 р. *Роберт Прім* розробив і представив більш детальний опис алгоритму, тож назва алгоритму залишилася за Прімом.

Найбільш відомий підхід носить назву *Йозефа Крускала* (1956). Основна ідея цього алгоритму заключається в тому, що ребра впорядковуються по вазі; на кожному кроці до остову, який ми будуємо, додається саме легке ребро, яке з'єднує вершини з різних компонент.

2.1. Остовні дерева мінімальної вартості

Нехай $G = (V, X)$ – зв'язний граф, у якому кожне ребро (v, u) відмічено числом $c(v, u)$, яке називається вартістю ребра. *Остовним* деревом графа G називається дерево, що містить всі вершини графа G . *Вартість* остовного дерева вираховується як сума вартості всіх ребер, що входять в це дерево.

Обидва алгоритма (Крускала і Пріма) слідуєть «жадібній» стратегії: на кожному кроці вибирається «локально найкращий» варіант. Не для всіх задач такий вибір приводить до оптимального рішення, але для задачі про побудову остовного дерева мінімальної вартості це саме так.

Визначення. «Жадібним» алгоритмом називається алгоритм, який на кожному кроці робить локально оптимальний вибір, – в надії, що підсумкове рішення теж виявиться оптимальним.

Загальна схема алгоритмів така. Шуканий остов будується поступово: до пустої множини A на кожному кроці додається одне ребро. Множина A завжди є підмножиною деякого мінімального остова. Ребро (v, u) , що додається на черговому кроці, вибирається таким чином, щоб не порушити цю властивість. Об'єднання $A \cup \{(v, u)\}$ також повинно бути підмножиною мінімального остова. Таке ребро називається «безпечним» ребром для A .

Обидва алгоритми слідуєть описаній схемі, але по різному вибирають безпечне ребро. В алгоритмі Крускала множина ребер A представляє собою ліс, який складається з декількох зв'язних компонент (дерев). Додається ребро мінімальної вартості серед всіх ребер, кінці яких лежать в різних компонентах.

В алгоритмі Пріма множина A представляє собою одне дерево. «Безпечне» ребро, яке додається до A , вибирається як ребро найменшої вартості, що з'єднує це вже побудоване дерево з деякою новою вершиною.

Остовні дерева мінімальної вартості (ОДМВ) можна застосовувати, наприклад, при розробці комунікаційних мереж. Тут вершинами графа можна представити міста, а ребрами – можливі комунікаційні лінії між містами, а вартість ребер відповідає вартості комунікаційних ліній. В цьому випадку остовне дерево мінімальної вартості представляє комунікаційну мережу, яка об'єднує всі міста комунікаційними лініями мінімальної вартості.

2.2. Алгоритм Крускала

Побудова остовного дерева мінімальної вартості алгоритмом Крускала починається з графа $T = (V, \emptyset)$, який складається з тільки з n вершин графа G і який не має ребер. Таким чином кожна вершина є зв'язною (сама з собою) компонентою. Поступово об'єднуючи ці компоненти, формуємо ОДМВ. В алгоритмі Крускала всі ребра перебираються по зростанню вартості. Для чергового ребра перевіряється, чи не лежать кінці ребра в різних компонентах зв'язності, а якщо так, ребро додається, і компоненти з'єднуються.

Вибираючи ребра, ми повинні пам'ятати, що дерево мінімальної вартості 1) повинно бути зв'язним і містити всі вершини; 2) не повинно містити циклів; 3) повинно мати мінімальну вартість.

На рисунках 2.1.а)–2.1.і) зображено роботу алгоритму.

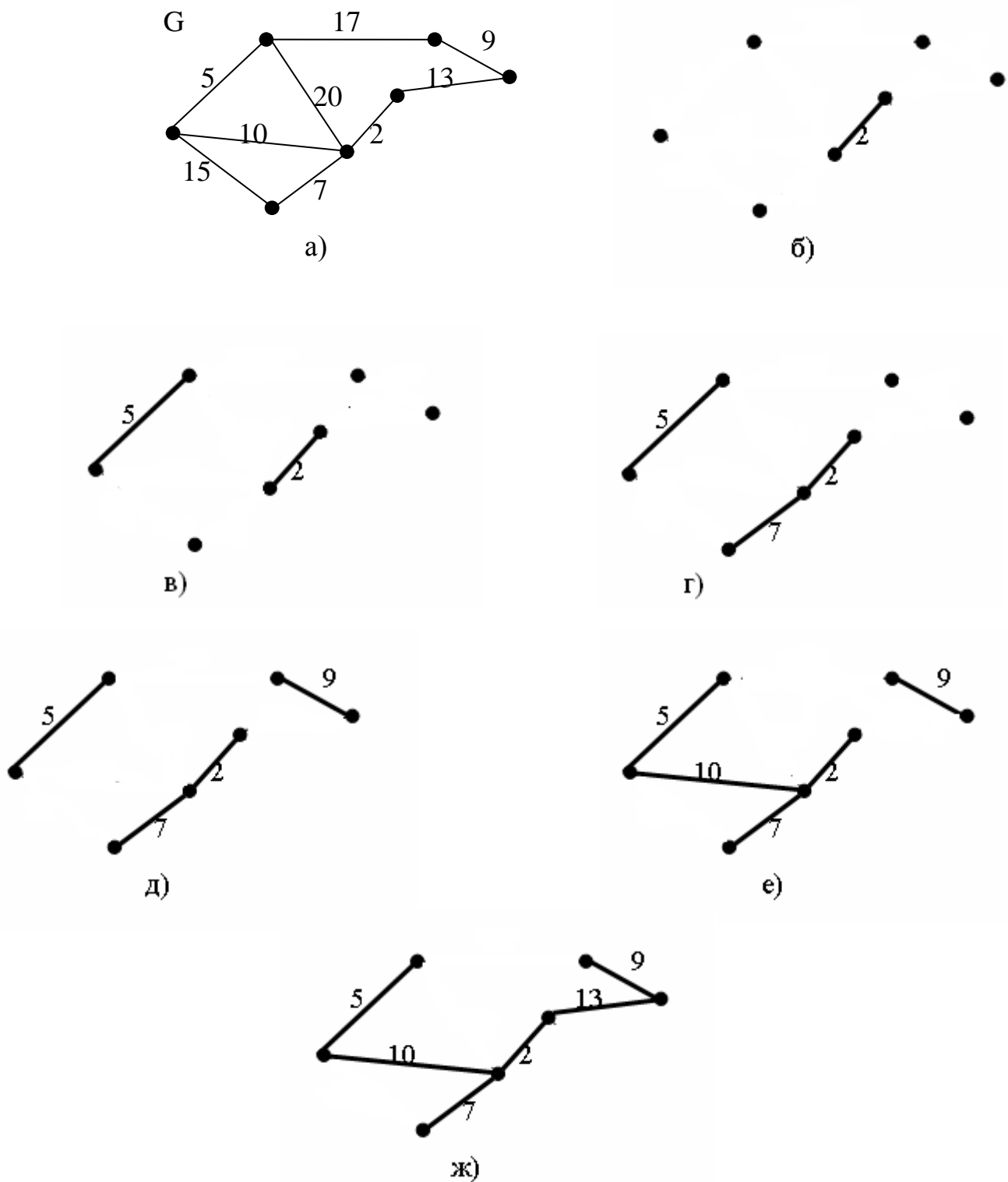


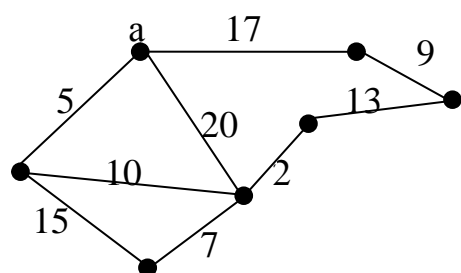
Рис. 2.1. Схема алгоритму Крускала

На рисунку 2.1.а) показано граф $G = (V, X)$. Маємо множину вершин V і підмножину деякого мінімального остова A , яка пуста. З множини ребер графа X вибираємо ребро з найменшою вагою. Це ребро з вагою 2, додаємо його до множини A (рис. 2.1.б). Наступне безпечне ребро з вагою 5 (рис. 2.1.в), додаємо його до дерева A . На третьому кроці додаємо ребро з вагою 7 (рис. 2.1.г) і т.д. Оскільки кінці ребер з вагою 17, 20, 15 лежать в одній

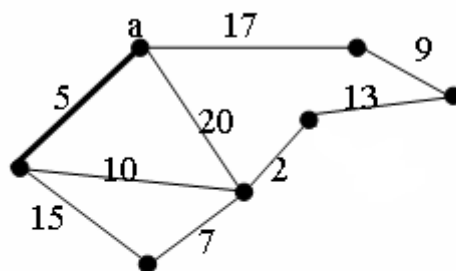
компоненті зв'язності, то вони не додаються. Таким чином мінімальне остовне дерево побудоване.

2.3. Алгоритм Пріма

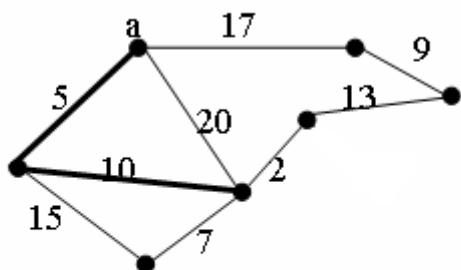
Як і в алгоритмі Крускала в алгоритмі Пріма на кожному кроці додається до остова, яке ми будуємо, безпечне ребро. Алгоритм Пріма відноситься до групи алгоритмів нарощування мінімального остова: на кожному кроці існує не більш однієї нетривіальної компоненти зв'язності, і кожний раз до неї додається ребро найменшої ваги, що з'єднує вершини компоненти з рештою вершин. Процес продовжується до тих пір, поки число ребер не стане рівним $n-1$ (n – кількість вершин).



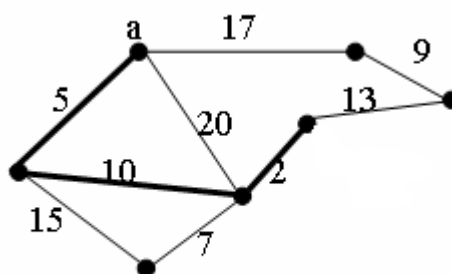
а) Початкова фаза. Корінь а



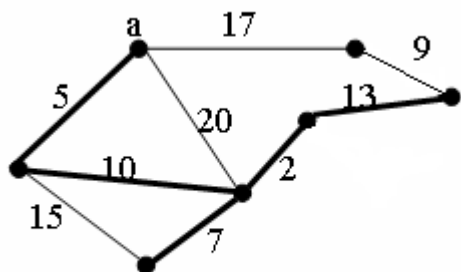
б) ребро з вагою 5 є мінімальним ребром, що з'єднує корінь а з вершинами



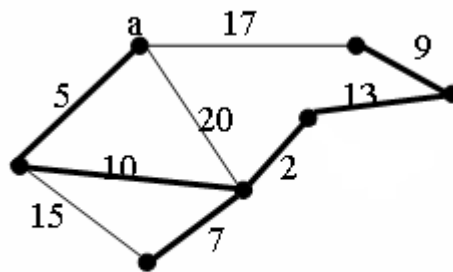
в) Наступне безпечне ребро з вагою 10



г)



д)



ж)

Рис. 2.2. Схема роботи алгоритма з корнем а

На рисунку 2.2. показано граф G і його корінь a , з якого починається побудова остовного дерева мінімальної вартості. Від кореня відходять три ребра з вагою 5, 20 і 17. оскільки вага 5 найменша, вибираємо ребро з цією вагою (рис.2.2.б). В наступній вершині маємо два ребра з вагою 10 і 15. Вибираємо менше і з'єднуємо з третьою вершиною. Процес продовжується до тих пір, поки всі вершини не попадуть до дерева мінімальної вартості. Ребра 17, 20, 15 утворюють цикл, тому ми їх не додаємо (рис.2.2.ж).

2.4. Алгоритм Дейкстри

Нехай задана карта автомобільних доріг, які з'єднують міста між собою. Треба знайти найкращий маршрут, наприклад, між двома містами. Така задача виглядає досить простою, але що значить «найкращий» маршрут? Це може бути відстань, час проходження маршруту з урахуванням обмежень в швидкості, затримки, які визвані проїздом через міста або об'їздом міст і т.д.

Будемо розв'язувати задачу пошуку найкоротшого маршрута з точки зору найкоротшої відстані. Цю задачу легко можна змодельовати за допомогою графів. Отже, нехай задано зв'язний граф G , в кому кожне ребро має додатню вагу, що дорівнює довжині ребра. Довжина шляху в такому графі дорівнює сумі довжин ребер, що складають цей шлях. В термінах графів задача зводиться до пошуку найкоротшої відстані між двома заданими вершинами.

Задачі про найкоротші шляхи відносяться до фундаментальних задач комбінаторної оптимізації, бо більшість таких задач можна звести до пошуку найкоротшого шляху на графах. Існують різні типи задач про найкоротші шляхи: між двома заданими вершинами, між заданою вершиною та рештою вершин, між кожною парою вершин на графі, між двома заданими вершинами для шляхів, які проходять через одну або декілька заданих вершин.

Серед багатьох алгоритмів пошуку найкоротшого шляху один з найкращих належить нідерландському вченому Е. Дейкстрі і по суті співпадає з розглянутим вище алгоритмом Пріма.

Розглянемо на прикладі алгоритм Дейкстри. Представимо карту автомобільних доріг у вигляді графа, в якому нам треба знайти найкоротшу відстань між містами Луцьк та Чернівці (рис. 2.3.). Для наглядності вибрані умовні відстані.

Алгоритм Дейкстри складається з наступних кроків:

1. На першому кроці визначаємо відстані (довжиною в одне ребро) від заданої вершини (Луцьк) до всіх інших (рис. 2.3.а).
2. Далі відбираємо найменшу з відстаней (вибираємо виділене ребро між (Луцьк) та (Рівне)).

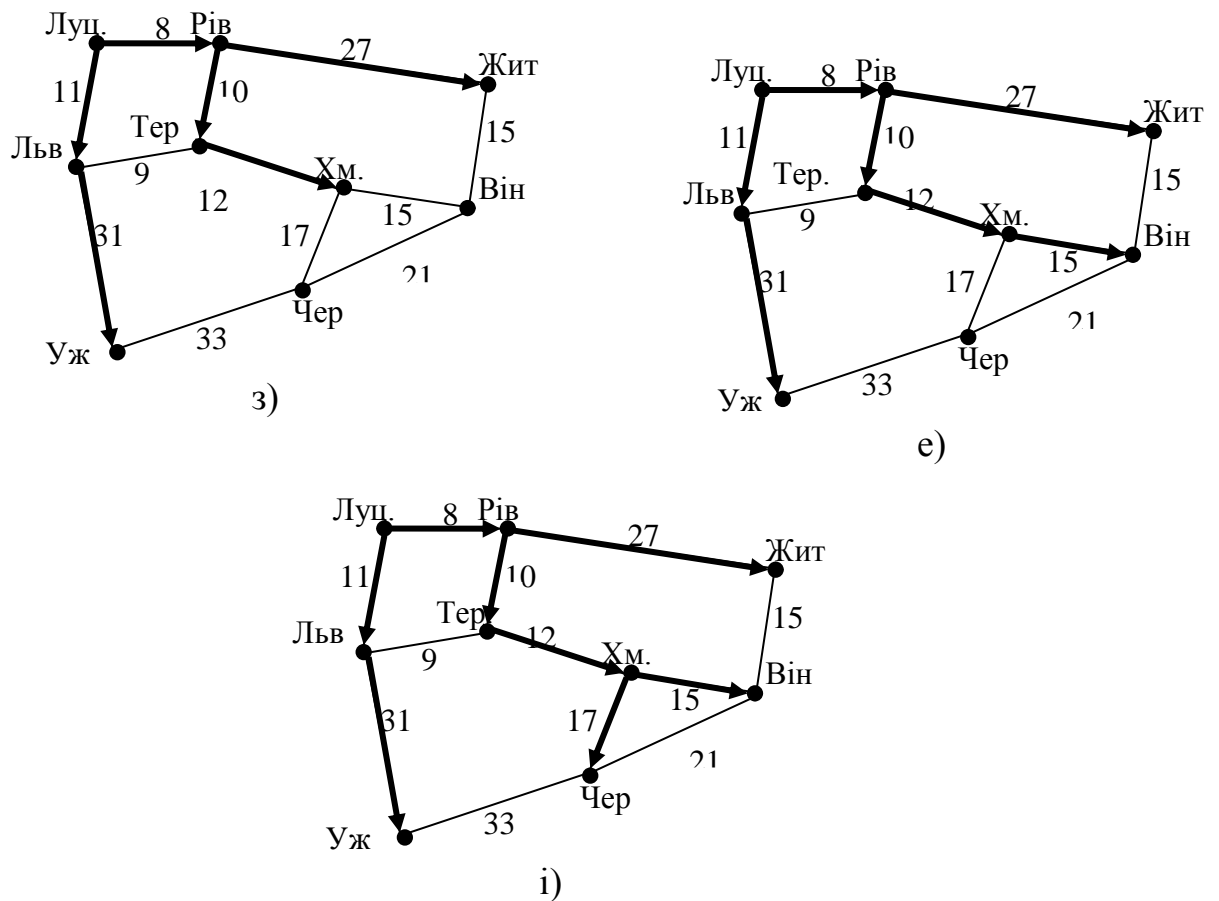


Рис. 2.3. Ілюстрація алгоритму Дейкстри для знаходження найкоротшого шляху від початкової вершини (Луцьк) до кінцевої (Чернівці)

Таким чином весь цей процес повторюється, додається нова найкоротша відстань до списку і т.д., аж поки кінцева вершина (Чернівці) не буде сполучена з вершиною (Луцьк) шляхом з виділених ребер.

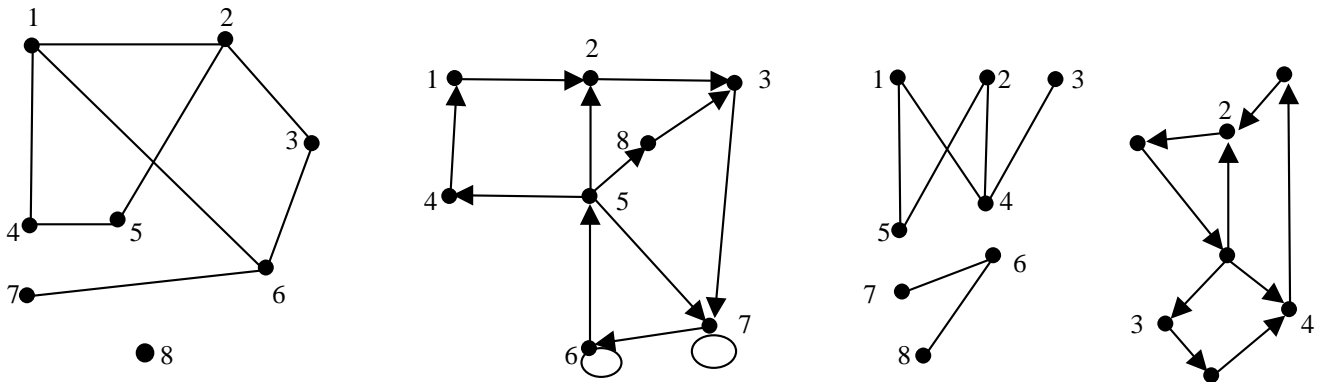
На другій ітерації відстань (рис. 2.3.б) до (Львів) (11) мінімальна. Це значення використовуємо для оновлення відстані до (Ужгород) (стає 42 замість нескінченності), а потім (Львів) видаляється зі списку. На третій ітерації (рис.2.3.в), відстань до (Тернопіль) мінімальна, оновлюємо відстань до Хмельницька(замість нескінченності робимо 30). Множину найкоротших відстаней ми отримали у вигляді дерева найкоротших шляхів (рис. 2.3. і).

Таким чином, для того, щоб обчислити найкоротшу відстань від (Луцьк) до (Чернівці), треба обчислити найкоротші шляхи від (Луцьк) до всіх вершин.

Отже можна зробити висновок, що в самому найгіршому випадку задача пошуку найкоротшої відстані від початкової вершини до кінцевої має таку ж складність, що і задача про пошук найкоротших шляхів від початкової вершини до всіх інших вершин.

Комп'ютерний практикум №2

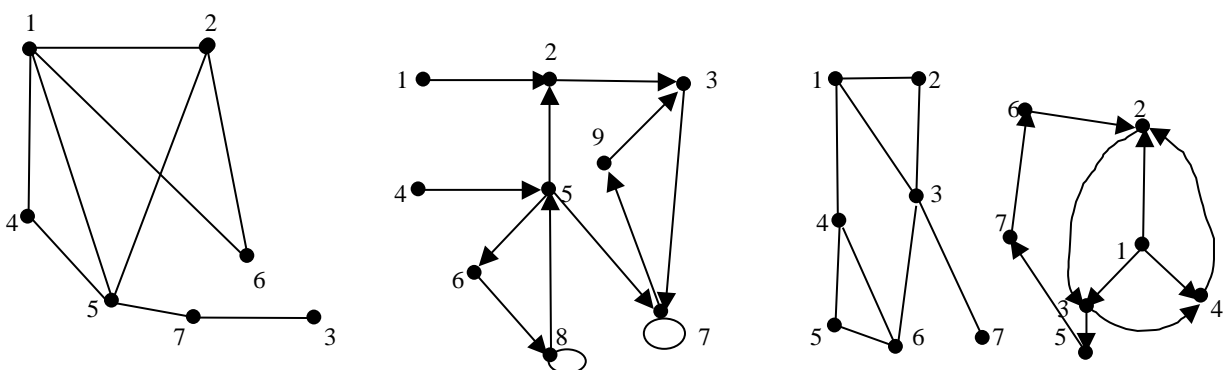
1. Задано граф. Записати всі матричні представлення у вигляді всіх можливих матриць
Варіанти: 1.1 - 1.4



2. По заданій матриці суміжності графа відновити граф. У варіантах 2.1, 2.3- граф Бержа, у варіантах 2.2, 2.4 - звичайний граф

$$A(G) = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad A(G) = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \quad A(G) = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \quad A(G) = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

1. Задано граф. Записати всі матричні представлення у вигляді всіх можливих матриць
Варіанти: 1.5 - 1.8



2. По заданій матриці суміжності графа відновити граф.

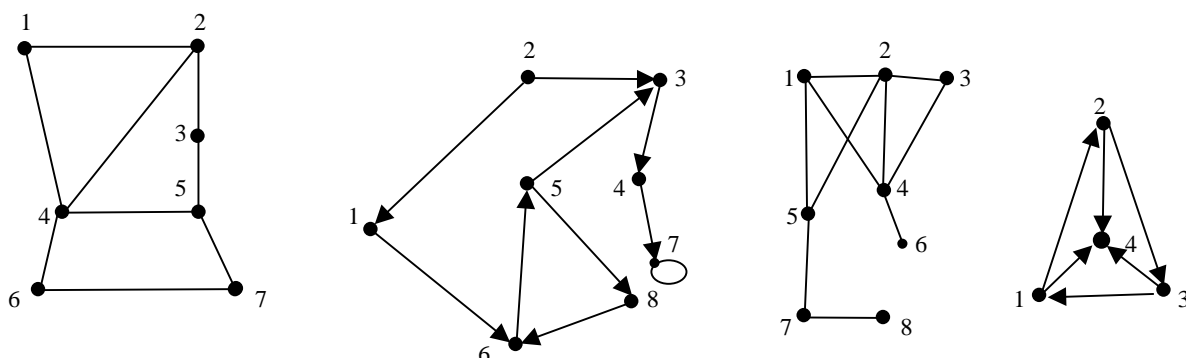
$$A(G) = \begin{vmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{vmatrix}$$

$$A(G) = \begin{vmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{vmatrix}$$

$$A(G) = \begin{vmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{vmatrix}$$

$$A(G) = \begin{vmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{vmatrix}$$

1. Задано граф. Записати всі матричні представлення у вигляді всіх можливих матриць
Варіанти: 1.9 - 1.12



2. По заданій матриці суміжності графа відновити граф.

$$A(G) = \begin{vmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{vmatrix}$$

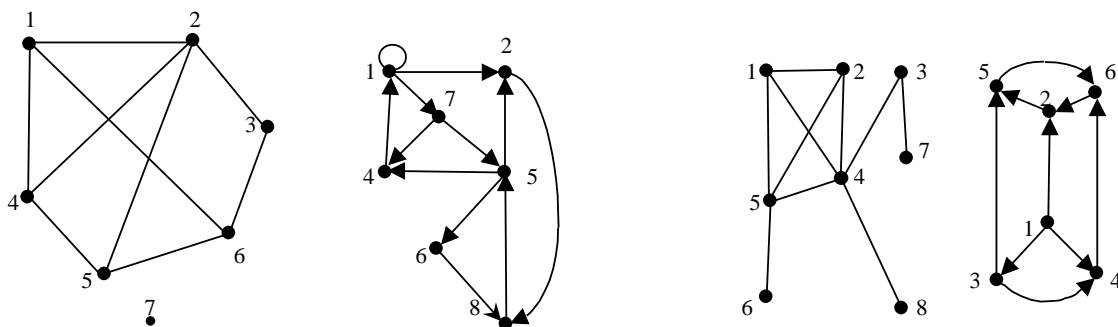
$$A(G) = \begin{vmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{vmatrix}$$

$$A(G) = \begin{vmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

$$A(G) = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{vmatrix}$$

1. Задано граф. Записати всі матричні представлення у вигляді всіх можливих матриць
Варіанти: 1.13 - 1.16

2. По заданій матриці суміжності графа відновити граф



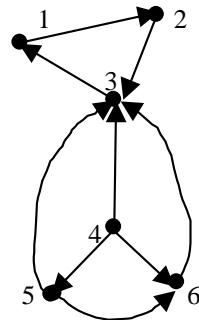
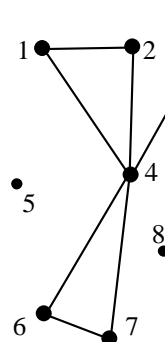
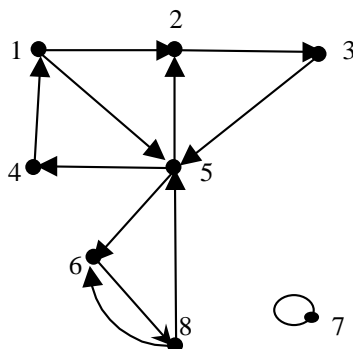
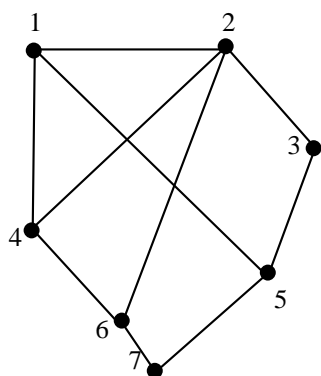
$$A(G) = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$A(G) = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$A(G) = \begin{vmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{vmatrix}$$

$$A(G) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

1.Задано граф. Записати всі матричні представлення у вигляді всіх можливих матриць
Варіанти: **1.17 1.20**



2. По заданій матриці суміжності графа відновити граф

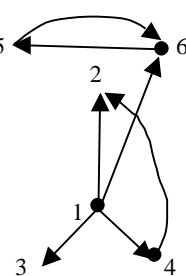
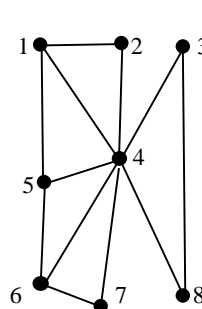
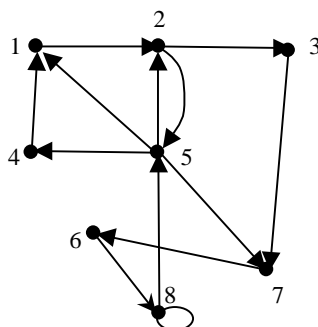
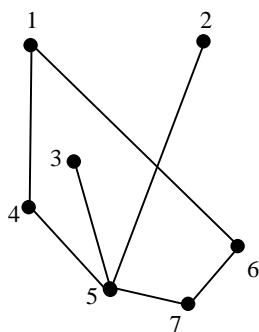
$$A(G) = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$A(G) = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$A(G) = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$A(G) = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

1.Задано граф. Записати всі матричні представлення у вигляді всіх можливих матриць.
Варіанти: **1.21** **1.24**



2. По заданій матриці суміжності графа відновити граф

$$A(G) = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

$$A(G) = \begin{vmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{vmatrix}$$

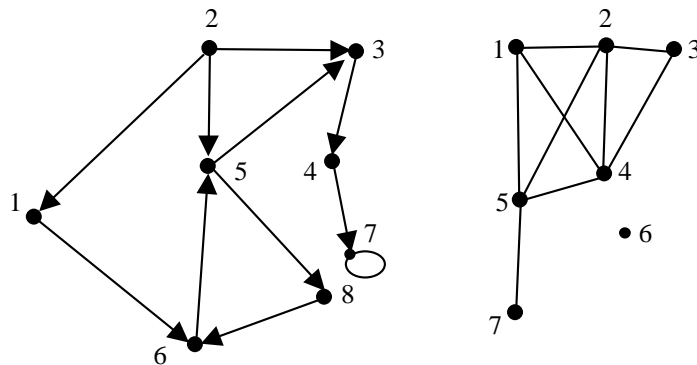
$$A(G) = \begin{vmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{vmatrix}$$

$$A(G) = \begin{vmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{vmatrix}$$

1. Задано граф. Записати всі матричні представлення у вигляді всіх можливих матриць.

Варіанти: **1.25**

1.26



2. По заданій матриці суміжності графа відновити граф

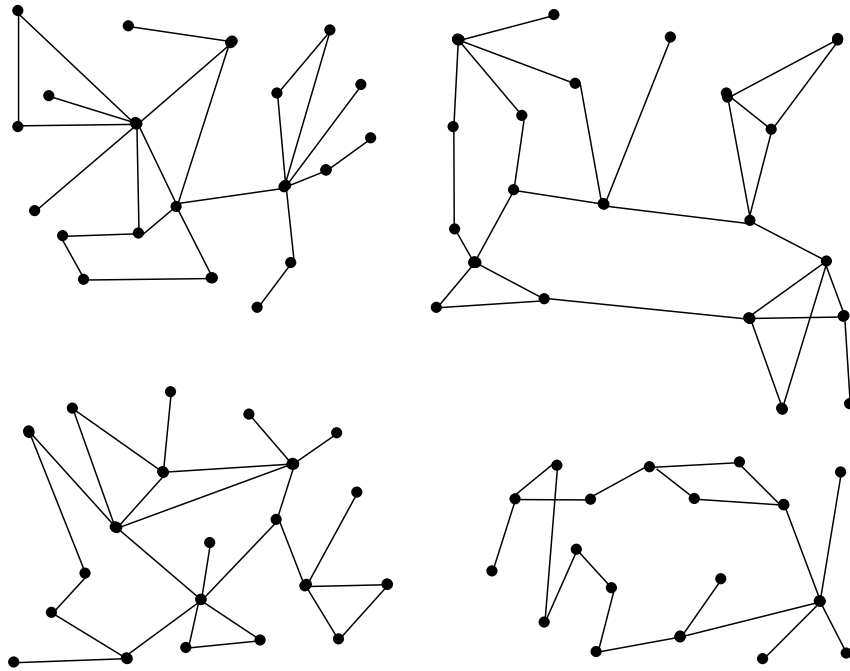
$$A(G) = \begin{vmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

$$A(G) = \begin{vmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{vmatrix}$$

Комп'ютерний практикум №3

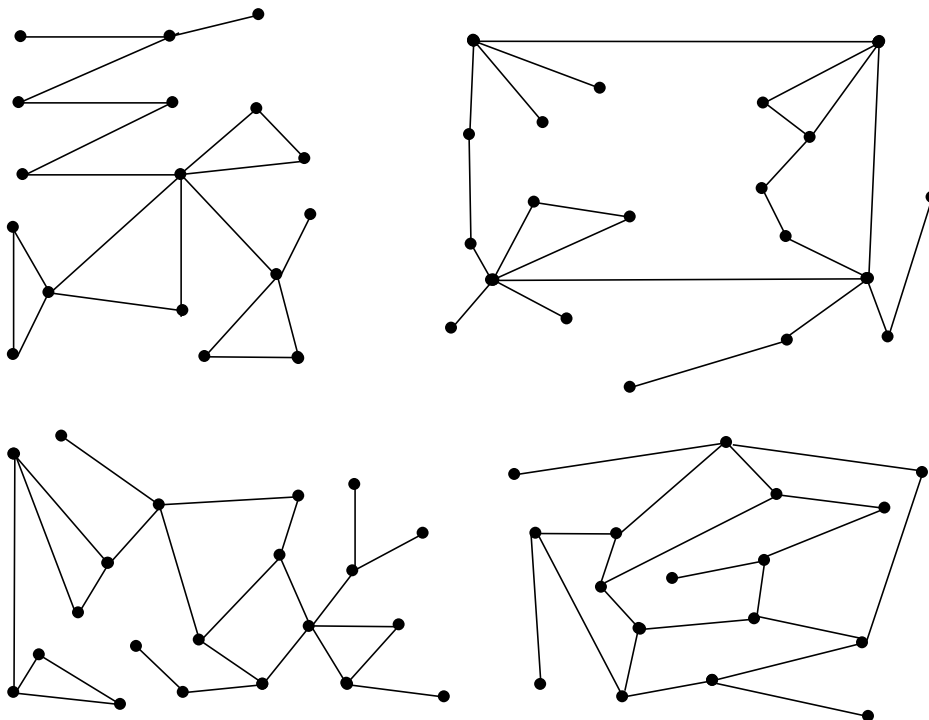
Варіант 1-4

Задано граф G . Побудувати $B(G), C(G), bc(G)$. Для графа $bc(G)$ показати центр і центроїд.

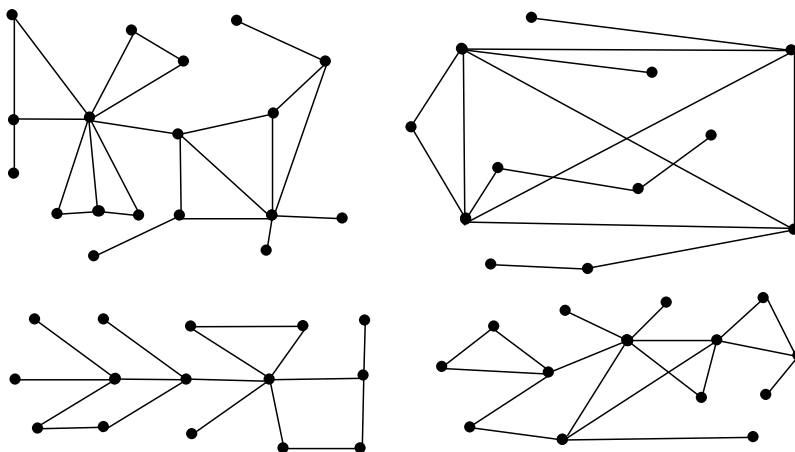


Варіант 5-8

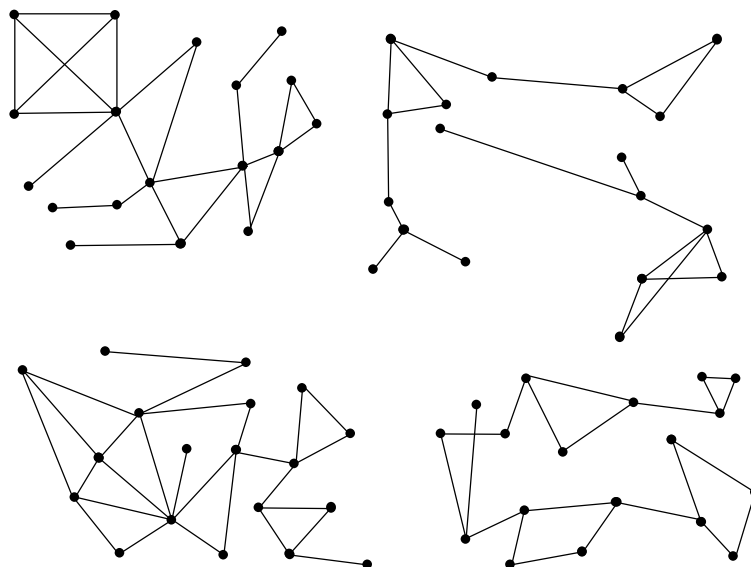
Задано граф G . Побудувати $B(G), C(G), bc(G)$. Для графа $bc(G)$ показати центр і центроїд.



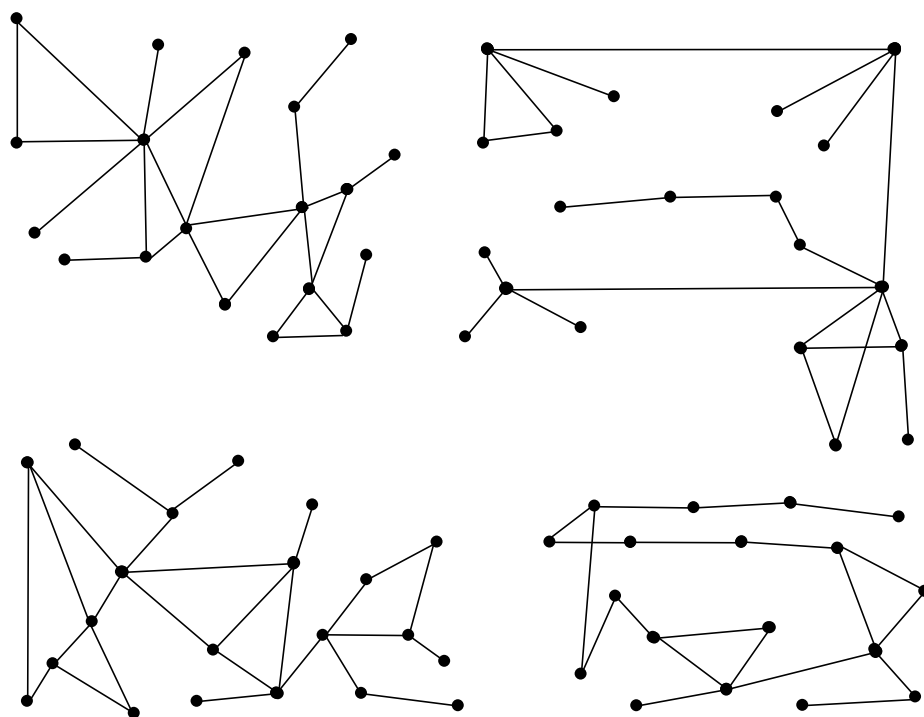
Варіант 9-12. Задано граф G . Побудувати $B(G), C(G), bc(G)$. Для графа $bc(G)$ показати центр і центроїд.



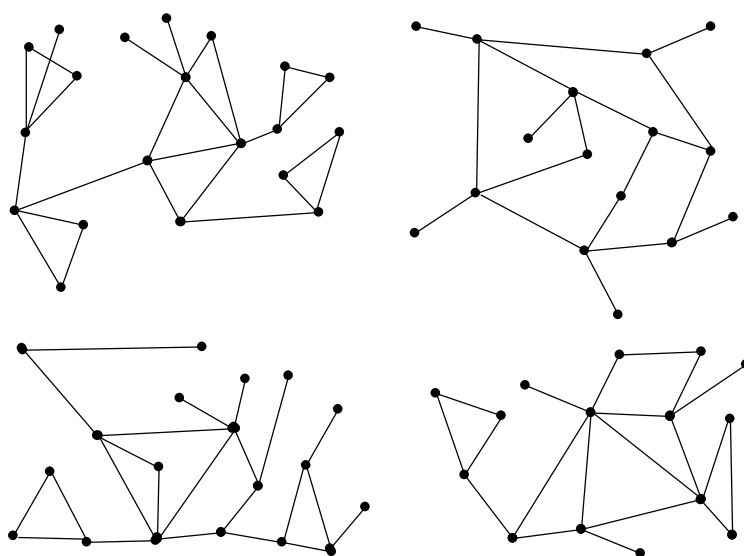
Варіант 13-16. Задано граф G . Побудувати $B(G), C(G), bc(G)$. Для графа $bc(G)$ показати центр і центроїд.



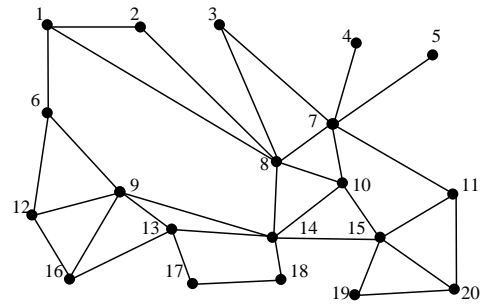
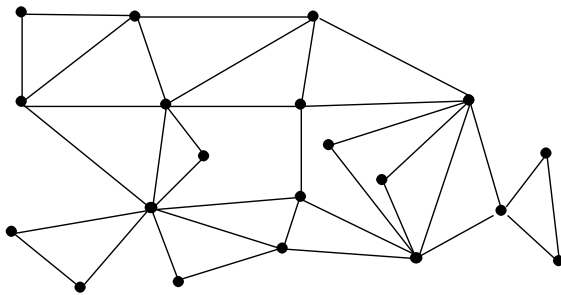
Варіант 17-20. Задано граф G . Побудувати $B(G), C(G), bc(G)$. Для графа $bc(G)$ показати центр і центроїд.



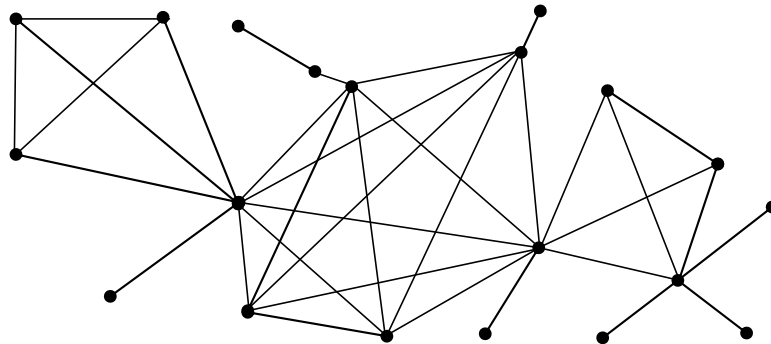
Варіант 21-24. Задано граф G . Побудувати $B(G), C(G), bc(G)$. Для графа $bc(G)$ показати центр і центроїд.



Варіант 25, 26. Задано граф G . Побудувати $V(G), C(G), bc(G)$. Для графа $bc(G)$ показати центр і центроїд.



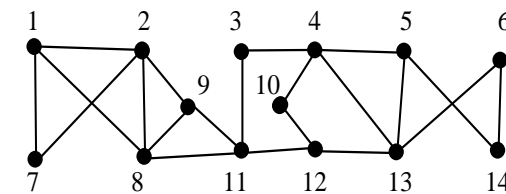
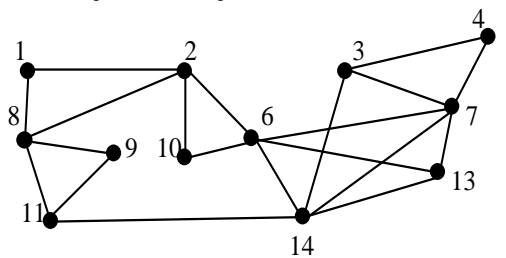
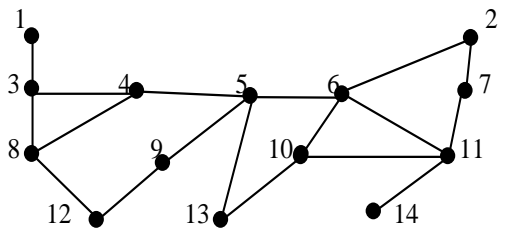
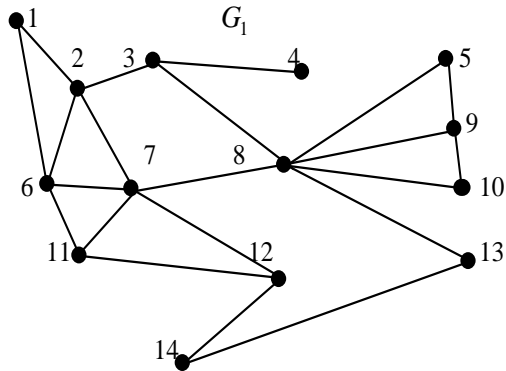
Варіант 27. Задано граф G . Побудувати $V(G), C(G), bc(G)$. Для графа $bc(G)$ показати центр і центроїд.



Комп'ютерний практикум №4

Показати дію алгоритма Краскала на даному графі. Варіант 1-4

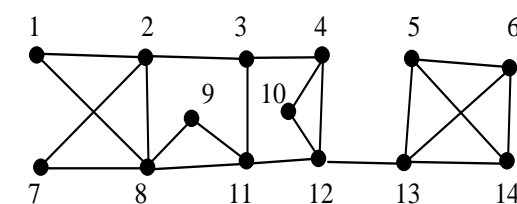
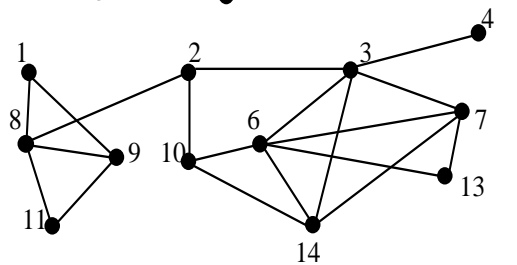
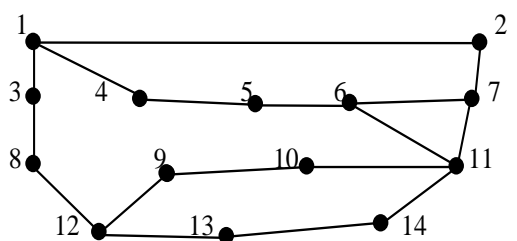
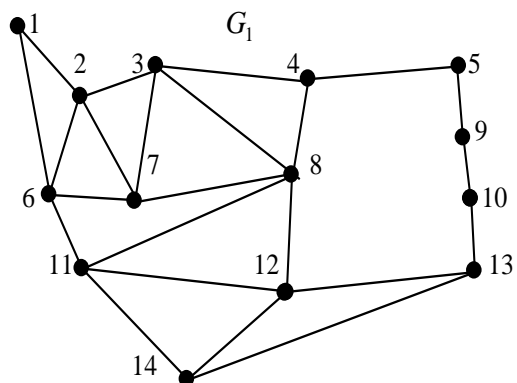
Функція вартості для графів:



F(G_1)					
r_i	c	r_i	c	r_i	c
(1,2)	2	(6,7)	67	(5,9)	6
(1,6)	5	(7,8)	45	(9,10)	8
(2,6)	7	(6,11)	22	(8,9)	13
(2,3)	9	(7,11)	11	(8,10)	4
(3,4)	12	(13,14)	8	(8,13)	8
(3,8)	4	(7,12)	9	(11,12)	9
(2,7)	7	(5,8)	1	(12,14)	1
F(G_2)					
r_i	c	r_i	c	r_i	c
(1,3)	7	(9,12)	1	(7,11)	1
(3,8)	5	(5,6)	3	(6,11)	24
(3,4)	3	(5,13)	90	(10,11)	6
(4,8)	45	(10,13)	456	(11,14)	2500
(4,5)	67	(6,10)	4		
(8,12)	11	(2,6)	7		
(5,9)	24	(2,7)	4		
F(G_3)					
r_i	c	r_i	c	r_i	c
(1,2)	13	(2,6)	9	(3,14)	3
(1,8)	5	(2,10)	1	(3,7)	7
(2,8)	1	(6,10)	2	(7,13)	9
(8,9)	78	(11,14)	77	(13,14)	13
(8,11)	24	(6,14)	56	(3,4)	4
(9,11)	56	(6,13)	20	(4,7)	1
(2,10)	77	(6,7)	10		
F(G_4)					
r_i	c	r_i	c	r_i	c
(1,2)	22	(9,11)	12	(4,5)	7
(1,7)	11	(3,11)	3	(5,13)	3
(2,7)	34	(3,4)	7	(5,14)	2
(2,8)	45	(4,10)	87	(6,13)	1
(1,8)	1	(10,12)	15	(6,14)	6
(2,9)	8	(11,12)	7		
(8,9)	6	(12,13)	4		
(8,11)	89	(4,13)	6		

Показати дію алгоритма Краскала на даному графі. Варіант 5-8

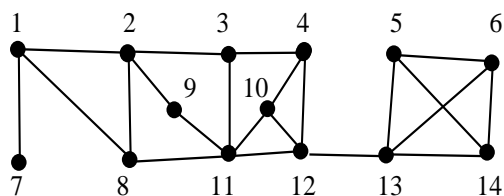
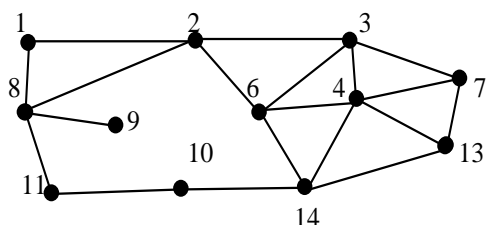
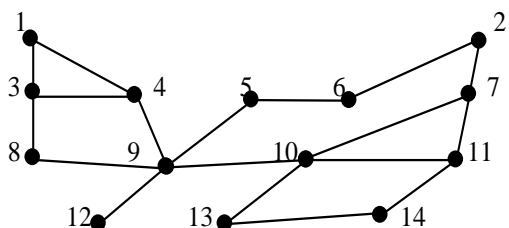
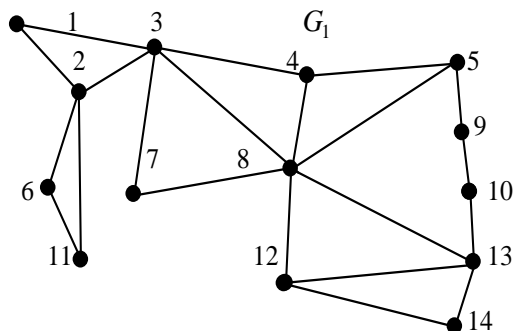
Функція вартості для графів



$F(G_1)$					
r_i	c	r_i	c	r_i	c
(1,2)	4	(6,11)	14	(11,12)	0.1
(2,6)	78	(7,8)	5	(11,14)	9
(1,6)	3	(3,8)	6	(12,14)	6
(2,3)	1	(8,11)	8	(12,13)	17
(2,7)	2	(3,4)	3	(13,14)	6
(3,7)	7	(4,8)	3	(4,5)	201
(6,7)	89	(8,12)	3	(5,9)	3
(9,10)	4	(10,13)	5		
$F(G_2)$					
r_i	c	r_i	c	r_i	c
(1,2)	2	(4,5)	1	(10,11)	5
(1,3)	4	(5,6)	5	(13,14)	9
(1,4)	16	(6,7)	8	(11,14)	11
(3,8)	12	(6,11)	9		
(8,12)	8	(7,11)	7		
(9,12)	9	(2,7)	4		
(12,13)	10	(9,10)	3		
$F(G_3)$					
r_i	c	r_i	c	r_i	c
(1,9)	78	(2,10)	22	(7,13)	8
(1,8)	45	(6,10)	76	(6,13)	9
(8,9)	37	(3,6)	55	(6,7)	51
(8,11)	24	(6,14)	54	(3,4)	23
(9,11)	89	(10,14)	37		
(2,8)	101	(3,14)	47		
(2,3)	13	(3,7)	11		
$F(G_4)$					
r_i	c	r_i	c	r_i	c
(1,2)	22	(13,14)	14	(10,12)	1
(2,3)	22	(1,8)	17	(4,12)	1
(3,4)	23	(2,7)	37	(5,13)	3
(5,6)	22	(2,8)	25	(5,14)	5
(7,8)	45	(8,9)	16	(6,13)	8
(8,11)	6	(9,11)	54	(6,14)	13
(11,12)	5	(3,11)	19		
(12,13)	19	(4,10)	10		

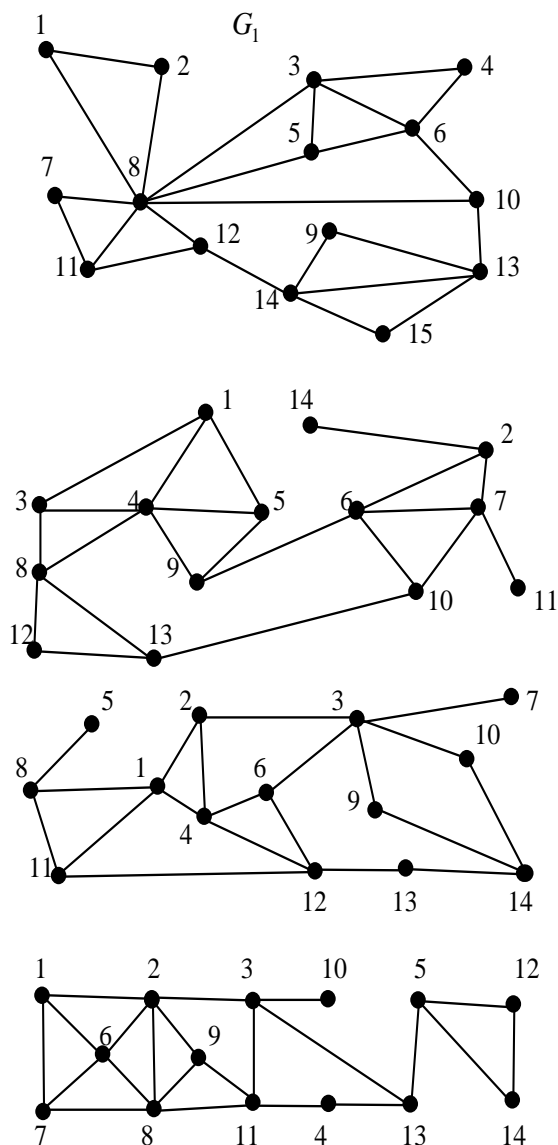
Показати дію алгоритма Краскала на даному графі. Варіант 9-12

Функція вартості для графів



$F(G_1)$					
r_i	c	r_i	c	r_i	c
(1,2)	5	(7,8)	1	(8,13)	1
(1,3)	6	(3,8)	65	(12,13)	1
(2,3)	9	(3,4)	43	(13,14)	9
(2,6)	14	(4,8)	78	(12,14)	7
(6,11)	5	(4,5)	25	(5,9)	6
(2,11)	3	(5,8)	23	(9,10)	5
(3,7)	2	(8,12)	11	(10,13)	4
$F(G_2)$					
r_i	c	r_i	c	r_i	c
(1,3)	5	(5,9)	23	(10,11)	1
(3,4)	3	(9,10)	11	(10,13)	1
(1,4)	2	(5,6)	89	(13,14)	2
(3,8)	8	(2,6)	76	(11,14)	3
(8,9)	14	(2,7)	45		
(4,9)	19	(7,10)	37		
(9,12)	45	(7,11)	13		
$F(G_3)$					
r_i	c	r_i	c	r_i	c
(1,2)	22	(2,3)	2	(3,7)	6
(1,8)	22	(2,6)	3	(4,7)	7
(2,8)	22	(3,6)	7	(4,13)	0.8
(8,9)	22	(3,4)	40	(7,13)	56
(8,11)	67	(4,6)	21	(13,14)	12
(11,10)	67	(6,14)	29		
(10,14)	54	(4,14)	1		
$F(G_4)$					
r_i	c	r_i	c	r_i	c
(1,2)	20	(3,11)	8	(5,6)	6
(1,7)	1	(3,4)	7	(6,14)	8
(1,8)	3	(4,10)	65	(13,14)	11
(2,8)	6	(10,12)	40	(5,13)	29
(2,9)	9	(4,12)	11	(6,13)	34
(9,11)	7	(11,12)	12	(5,14)	15
(8,11)	13	(12,13)	56		
(2,3)	5	(10,11)	75		

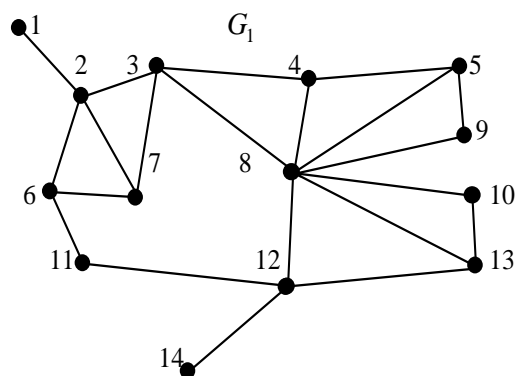
Показати дію алгоритма Краскала на даному графі. Варіант 13-16
Функція вартості для графів



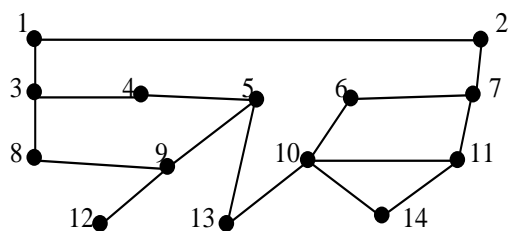
$F(G_1)$					
r_i	c	r_i	c	r_i	c
(1,2)	5	(11,12)	1	(5,6)	5
(2,8)	17	(8,3)	15	(3,4)	7
(1,8)	3	(8,5)	4	(4,6)	19
(7,8)	1	(8,10)	11	(6,10)	14
(7,11)	8	(12,14)	18	(10,13)	12
(8,11)	35	(3,5)	239	(9,14)	7
(14,15)	7	(13,15)	16	(9,13)	6
(8,12)	16	(3,6)	5	(14,13)	15
$F(G_2)$					
r_i	c	r_i	c	r_i	c
(3,4)	2	(3,8)	32	(6,10)	25
(1,3)	13	(4,8)	5	(6,7)	7
(1,4)	7	(8,12)	7	(7,10)	10
(4,5)	6	(8,13)	6	(7,11)	8
(1,5)	11	(12,13)	2	(2,6)	13
(4,9)	9	(13,10)	21	(2,7)	15
(5,9)	1	(6,9)	7	(2,14)	9
$F(G_3)$					
r_i	c	r_i	c	r_i	c
(5,8)	1	(2,3)	71	(10,14)	21
(8,1)	6	(4,6)	16	(4,12)	17
(8,11)	9	(3,6)	32	(6,12)	2
(1,11)	3	(3,7)	91	(11,12)	3
(1,2)	5	(3,9)	31	(12,13)	12
(1,4)	7	(3,10)	66	(13,14)	3
(2,4)	11	(9,14)	97		
$F(G_4)$					
r_i	c	r_i	c	r_i	c
(1,2)	6	(2,3)	8	(4,13)	1
(1,6)	3	(2,9)	6	(3,13)	8
(1,7)	8	(9,8)	5	(5,13)	6
(6,7)	2	(9,11)	3	(5,14)	4
(6,2)	1	(8,11)	5	(5,12)	3
(6,8)	9	(3,11)	2	(12,14)	2
(2,8)	15	(3,10)	2		
(7,8)	14	(4,11)	2		

Показати дію алгоритма Краскала на даному графі. Варіант 17-20

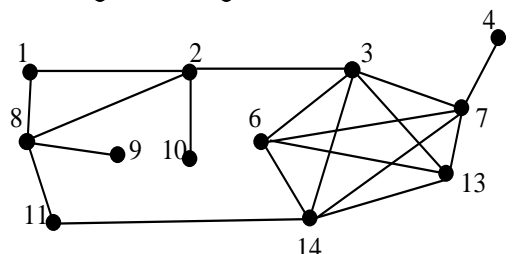
Функція вартості для графів



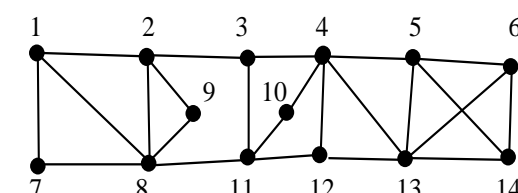
$F(G_1)$					
r_i	c	r_i	c	r_i	c
(1,2)	13	(6,11)	2	(5,9)	7
(2,3)	4	(3,4)	5	(8,9)	14
(2,6)	6	(3,8)	2	(8,10)	20
(2,7)	7	(8,12)	5	(8,13)	9
(3,7)	4	(4,8)	17	(10,13)	20
(6,7)	8	(4,5)	3	(5,8)	18
(12,14)	6	(12,13)	2	(11,12)	5



$F(G_2)$					
r_i	c	r_i	c	r_i	c
(1,2)	11	(5,9)	10	(10,11)	4
(1,3)	1	(6,7)	3	(11,14)	3
(2,7)	0.5	(8,9)	2	(10,14)	15
(3,4)	3	(7,11)	4		
(3,8)	15	(9,12)	51		
(4,5)	2	(6,10)	19		
(5,13)	1	(10,13)	8		

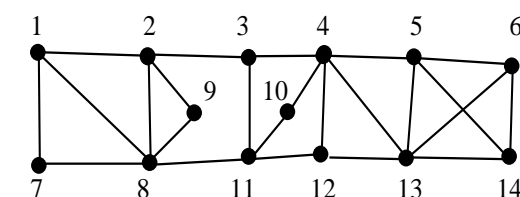
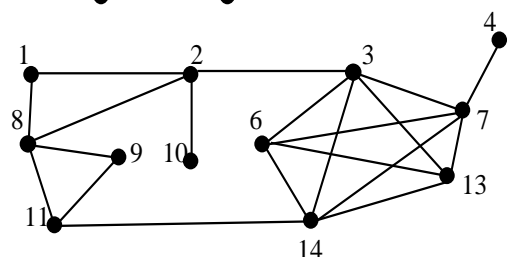
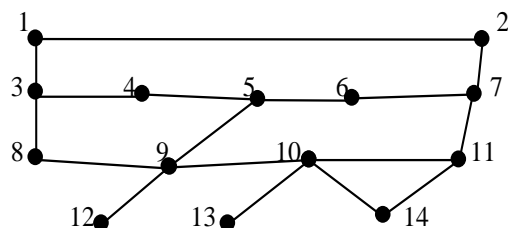
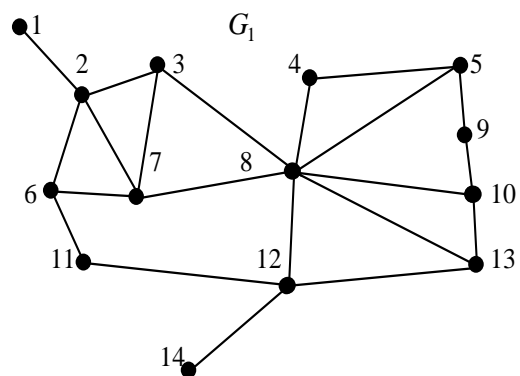


$F(G_3)$					
r_i	c	r_i	c	r_i	c
(1,2)	2	(2,3)	9	(7,13)	8
(1,8)	3	(3,7)	1	(6,7)	4
(2,8)	5	(4,7)	2	(6,14)	2
(8,9)	15	(3,6)	3	(13,14)	6
(8,11)	4	(3,14)	4	(6,13)	7
(11,14)	3	(3,13)	5		
(2,10)	4	(7,14)	17		



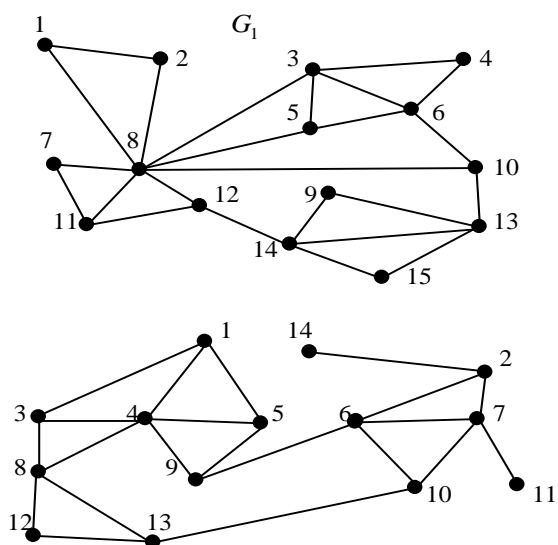
$F(G_4)$					
r_i	c	r_i	c	r_i	c
(1,2)	2	(8,11)	3	(2,9)	10
(2,3)	3	(11,12)	4	(8,9)	1
(3,4)	72	(12,13)	27	(3,11)	6
(4,5)	11	(13,14)	9	(10,11)	4
(5,6)	1	(6,14)	5	(4,10)	7
(1,7)	5	(1,8)	4	(4,12)	8
(7,8)	9	(2,8)	13	(4,13)	2
(5,13)	2	(5,14)	7	(6,13)	17

Показати дію алгоритма Краскала на даному графі. Варіант 20-23
Функція вартості для графів



$F(G_1)$					
r_i	c	r_i	c	r_i	c
(1,2)	2	(6,11)	11	(5,9)	77
(2,3)	4	(7,8)	20	(9,10)	4
(2,6)	1	(3,8)	6	(8,10)	2
(2,7)	7	(8,12)	15	(8,13)	11
(3,7)	19	(4,8)	7	(10,13)	9
(6,7)	3	(4,5)	13	(5,8)	81
(12,14)	27	(12,13)	21	(11,12)	3
$F(G_2)$					
r_i	c	r_i	c	r_i	c
(1,2)	1	(5,9)	0.5	(10,11)	14
(1,3)	11	(6,7)	3	(11,14)	3
(2,7)	15	(8,9)	2	(10,14)	15
(3,4)	3	(7,11)	4		
(3,8)	12	(9,12)	5		
(4,5)	3	(9,10)	9		
(5,6)	4	(10,13)	8		
$F(G_3)$					
r_i	c	r_i	c	r_i	c
(1,2)	4	(2,3)	39	(7,13)	8
(1,8)	1	(3,7)	117	(6,7)	14
(2,8)	5	(4,7)	8	(6,14)	2
(8,9)	16	(3,6)	3	(13,14)	26
(8,11)	14	(3,14)	24	(6,13)	7
(11,14)	31	(3,13)	4	(9,11)	18
(2,10)	18	(7,14)	7		
$F(G_4)$					
r_i	c	r_i	c	r_i	c
(1,2)	5	(8,11)	3	(2,9)	12
(2,3)	11	(11,12)	14	(8,9)	1
(3,4)	121	(12,13)	7	(3,11)	6
(4,5)	12	(13,14)	9	(10,11)	19
(5,6)	2	(6,14)	5	(4,10)	7
(1,7)	4	(1,8)	9	(4,12)	18
(7,8)	7	(2,8)	18	(4,13)	2
(5,13)	32	(5,14)	75	(6,13)	17

Показати дію алгоритма Краскала на даному графі. Варіант **24-25**
Функція вартості для графів

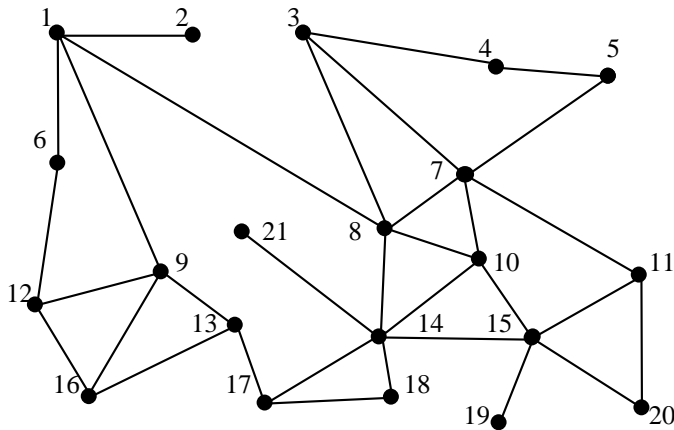


$F(G_1)$					
r_i	c	r_i	c	r_i	c
(1,2)	7	(11,12)	10	(5,6)	15
(2,8)	27	(8,3)	1	(3,4)	74
(1,8)	30	(8,5)	3	(4,6)	99
(7,8)	11	(8,10)	10	(6,10)	13
(7,11)	8	(12,14)	12	(10,13)	12
(8,11)	31	(3,5)	39	(9,14)	7
(14,15)	8	(13,15)	26	(9,13)	2
(8,12)	14	(3,6)	5	(14,13)	15
$F(G_2)$					
r_i	c	r_i	c	r_i	c
(3,4)	21	(3,8)	1	(6,10)	5
(1,3)	32	(4,8)	2	(6,7)	37
(1,4)	26	(8,12)	7	(7,10)	20
(4,5)	24	(8,13)	26	(7,11)	18
(1,5)	15	(12,13)	12	(2,6)	23
(4,9)	17	(13,10)	1	(2,7)	25
(5,9)	12	(6,9)	17	(2,14)	91

Комп'ютерний практикум №5

Продемонструвати дію алгоритма Дейкстри на даному графі. Варіант 4.1-4.4

Використати цей алгоритм для пошуку найкоротшого шляху між вершинами: Довести, що обраний вами шлях дійсно єнайкоротший.



4.1. Між вершинами v_1 и v_{19}

4.2. Між вершинами v_2 и v_{18}

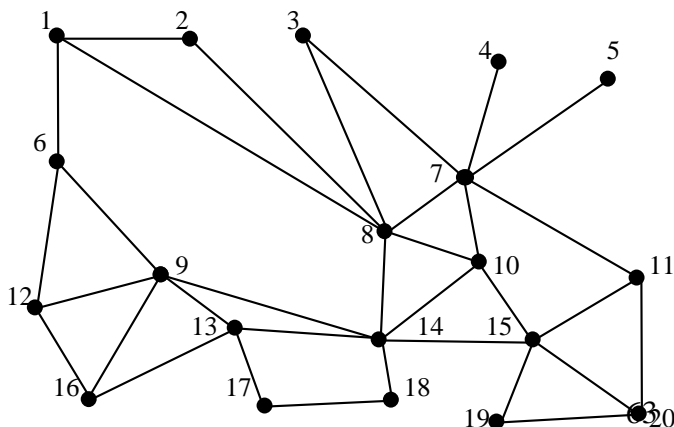
4.3. Між вершинами v_3 и v_{16}

4.4. Між вершинами v_4 и v_9

$F(G_1)$					
r_i	c	r_i	c	r_i	c
(1,2)	2	(6,1)	5	(9,13)	6
(1,6)	5	(7,8)	20	(9,12)	8
(1,9)	7	(6,12)	22	(8,7)	20
(1,8)	9	(7,11)	11	(8,10)	4
(3,4)	12	(15,14)	8	(8,14)	15
(3,8)	4	(5,4)	9	(11,15)	9
(3,7)	7	(5,7)	1	(12,16)	1
		(15,20)	25	(10,15)	5
(14,21)	40	(15,19)	15	(10,14)	10
(10,7)	7	(14,18)	4	(14,17)	15

Продемонструвати дію алгоритма Дейкстри на даному графі. Варіант 4.5-4.8

Використати цей алгоритм для пошуку найкоротшого шляху між вершинами: Довести, що обраний вами шлях дійсно єнайкоротший.



4.5. Між вершинами v_6 и v_{19}

4.6. Між вершинами v_2 и v_{18}

4.7. Між вершинами v_3 и v_{10}

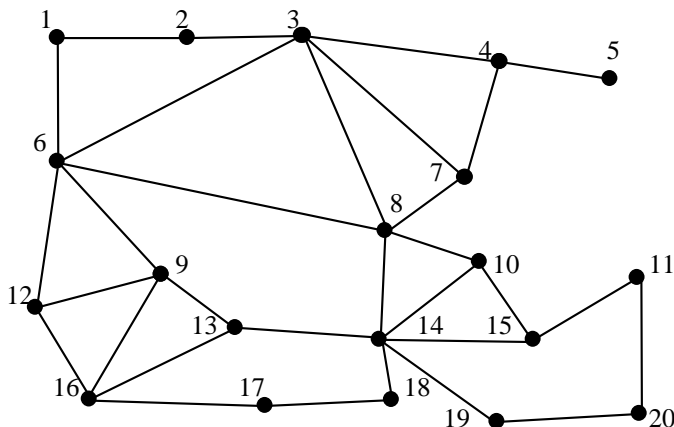
4.8. Між вершинами v_4 и v_5

$F(G_2)$					
r_i	c	r_i	c	r_i	c
(1,2)	7	(9,12)	1	(7,11)	1
(3,8)	20	(9,16))	3	(10,8)	11
(3,7)	3	(19,15)	90	(10,14)	6
(4,7)	45	(10,13)	456	(11,15)	250
(8,14)	67	(6,12)	4	(8,7)	7
(8,10)	11	(1,6)	7	(8,3)	20
(5,7)	24			(10,7)	5
(8,1)	25	(9,13)	3	(11,7)	30
(8,2)	30	(19,20)	10	(11,20)	15
(13,16)	8	(9,14)	15	(10,15)	11
(13,17)	4	(13,14)	10	(14,18)	25

Продемонструвати дію алгоритма Дейкстри на даному графі.

Варіант 4.9-4.12

Використати цей алгоритм для пошуку найкоротшого шляху між вершинами: Довести, що обраний вами шлях дійсно є найкоротший.



4.9. Між вершинами v_5 и v_{19}

4.10. Між вершинами v_1 и v_{18}

4.11. Між вершинами v_3 и v_{16}

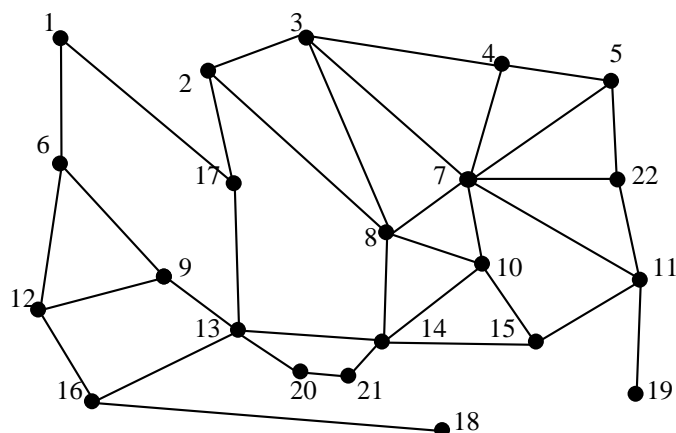
4.12. Між вершинами v_6 и v_{10}

$F(G_3)$					
r_i	c	r_i	c	r_i	c
(1,2)	13			(3,4)	3
(1,6)	5	(14,10)	1	(3,7)	7
(2,3)	1	(6,12)	2	(7,4)	9
(8,6)	78	(11,15)	77	(13,14)	13
(8,10)	24	(6,9)	56	(11,20))	14
(9,12)	56	(16,13)	20		
(15,10)	77			(3,8)	15
(16,17)	20	(17,18)	9	(7,8)	8
(14,18)	3	(14,15)	30	(14,19)	18

Продемонструвати дію алгоритма Дейкстри на даному графі.

Варіант 4.13-4.16

Використати цей алгоритм для пошуку найкоротшого шляху між вершинами. Довести, що обраний вами шлях дійсно є найкоротший.



4.13. Між вершинами v_1 и v_7

4.14. Між вершинами v_2 и v_{18}

4.15. Між вершинами v_{15} и v_{19}

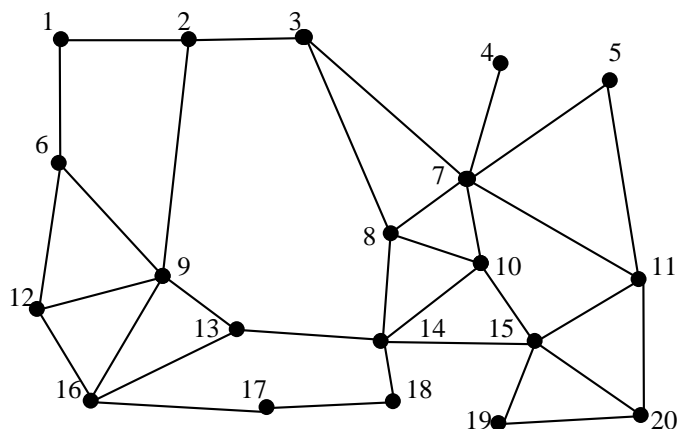
4.16. Між вершинами v_1 и v_2

$F(G_4)$					
r_i	c	r_i	c	r_i	c
(1,17)	22	(22,11)	12	(20,21)	7
(1,6)	11	(19,11)	3	(5,22)	3
(2,17)	34	(5,4)	7	(15,14)	2
(2,3)	45	(4,7)	87	(6,12)	1
(2,8)	1	(10,14)	15	(6,9)	6
(4,3)	8	(11,7)	7	(13,16)	17
(8,3)	6	(12,16)	4	(21,14)	3
(7,3)	89	(20,13)	6	(8,10)	10

Продемонструвати дію алгоритма Дейкстри на даному графі.

Варіант 4.17-4.20

Використати цей алгоритм для пошуку найкоротшого шляху між вершинами



4.17. Між вершинами v_1 и v_{20}

4.18. Між вершинами v_3 и v_{18}

4.19. Між вершинами v_3 и v_{17}

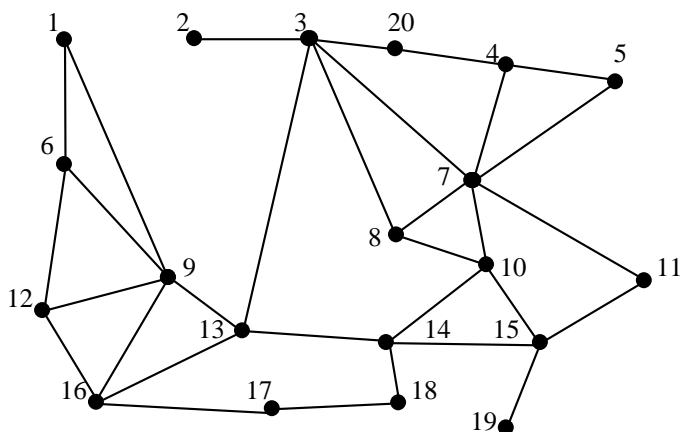
4.20. Між вершинами v_4 и v_5

$F(G_1)$					
r_i	c	r_i	c	r_i	c
(1,2)	4	(5,11)	14	(11,20)	0.1
(2,3)	78	(7,8)	5	(11,7)	9
(1,6)	3	(3,8)	6	(11,5)	6
(6,9)	5	(8,10)	8	(12,16)	17
(2,9)	10	(7,4)	3	(13,14)	6
(9,12)	15	(5,7)	10	(7,3)	25
(3,7)	7	(4,8)	3	(4,5)	201
(6,12)	89	(8,14)	5	(13,16)	3
(9,16)	4	(10,14)	5	(9,13)	5
(15,14)	7	(15,19)	6	(15,11)	10
(15,20)	9	(20,19)	8	(18,17)	16
(14,18)	3	(15,10)	4		

Продемонструвати дію алгоритма Дейкстри на даному графі.

Варіант 4.21-4.24

Використати цей алгоритм для пошуку найкоротшого шляху між вершинами



4.21. Між вершинами v_{20} и v_{19}

4.22. Між вершинами v_1 и v_{10}

4.23. Між вершинами v_3 и v_{17}

4.24. Між вершинами v_9 и v_{11}

$F(G_2)$					
r_i	c	r_i	c	r_i	c
(1,6)	2	(4,5)	1	(10,15)	5
(1,9)	4	(5,7)	5	(13,14)	9
(2,3)	26	(6,7)	8	(11,15)	11
(3,8)	12	(6,12)	9	(3,13)	30
(8,10)	8	(7,11)	7	(3,7)	13
(9,12)	9	(2,7)	4	(3,20)	6
(12,13)	10	(9,16)	3	(4,20)	4
(8,7)	4	(9,13)	5	(9,6)	15
(4,7)	7	(15,14)	11	(15,19)	40
		(10,7)	6	(10,14)	100
(13,16)	10	(12,16)	10	(15,19)	35

СПИСОК ЛІТЕРАТУРИ

1. J.A. Bondy and U. S. R. Murty. Graph Theory. graduated text in mathematics series. ISSN 0072-5285. 2008. – 648p.
2. .A. Bondy and U. S. R. Murty. Graph Theory With Applications. North-Holland New York ISBN 0-444-13451-7. 1982. –264p.
3. Кристофидес Н. Теория графов. Алгоритмический подход: пер. с англ. / Н. Кристофидес – М.: Мир, 1978. – 358 с.
4. Харари Ф. Теория графов: навч. посіб. / Ф. Харари. – М.: изд. УРСС, 2003. – 300 с.
5. Гудман С. Введение в разработку и анализ алгоритмов: пер. с англ. / С. Гудман, С. Хидетниemi. – М.: Мир, 1981. – 366 с.
6. Кристофидес Н. Теория Графов. Алгоритмический подход: пер.с англ. / Н. Кристофидес. – М.: Мир, 1978. – 487 с.
7. Бертисс А. Т. Структуры данных: учебник. / А.Т. Бертисс. – М.: Статистика, 1974. – 408 с.
8. Оре О. Графы и их применение: пер. с англ. / О. Оре. – М.: Мир, 1965. – 174 с.
9. Берж К. Теория графов и ее применения: пер.с англ. / К. Берж. – М.: Мир, 1984. – 307 с.
10. Альфред В. Ахо. Структуры данных и алгоритмы: пер.с англ. / Альфред В. Ахо, Джон Э. Хопкрофт, Джефри Д. Ульман. –Издательский дом «Вильямс» М., СПб., К.: 2000. – 382 с.
11. Euler The Konigsberg bridges, Sci. Amer., 18 (1953), p. 66 – 70
12. Nash-Willians C, St J.A. (1966), On Hamiltonian circuits in finite graphs, Pro. American Mathematical Soc. 17. p. 466.
13. Ore O. Theory of Graphs, American Mathematical Soc., New York. 1962.

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1. ТЕОРІЯ ГРАФІВ	4
1.1. Основні визначення та властивості графів.....	5
1.2. Матриці графів	9
1.3. Матриці оргграфів.....	10
1.4. Ізоморфізм графів	11
1.5. Елементи графів.....	13
1.6. Маршрути та зв'язність.....	14
1.7. Степінь графа.....	16
1.8. Операції над графами.....	18
1.9. Блоки.....	22
1.9.1 Точки з'єднання, мости та блоки.....	22
1.9.2. . . Графи блоків і графи точок з'єднання	23
1.10. Обходи графів.....	25
1.10.1. Графи Ейлера.....	25
1.10.2. Гамільтонові графи.....	27
1.10.3.. Зв'язок між ейлеровими і гамільтоновими циклами.....	29
1.11. Древа. Опис дерев.....	31
1.11.1 Древа на множині вершин.....	33
1.11.2. Ідентифікація дерев.....	35
1.11.3. Древа графа.....	37
Комп'ютерний практикум №1.....	39
РОЗДІЛ 2. АЛГОРИТМИ НА ГРАФАХ.....	41
2.1. Основні древа мінімальної вартості.....	41
2.2. Алгоритм Крускала.....	42
2.3. Алгоритм Пріма.....	44
2.4. Алгоритм Дейкстри.....	45
Комп'ютерний практикум №2.....	48
Комп'ютерний практикум №3.....	52
Комп'ютерний практикум №4.....	56
Комп'ютерний практикум №5.....	63
Список літератури.....	67