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

Кузьменко І.М.

Теорія графів

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для здобувачів ступеня бакалавра за спеціальністю 122 «Комп'ютерні науки»

Київ
КПІ ім. Ігоря Сікорського 2020

Теорія графів. [Електронний ресурс]: навч. посіб. для здобувачів ступеня бакалавра за освітньою програмою «Комп'ютерний моніторинг та геометричне моделювання процесів і систем» спеціальності 122 «Комп'ютерні науки»/ І.М. Кузьменко; КПІ ім. Ігоря Сікорського. — Електронні текстові дані (1 файл: 1,7 Мбайт). — Київ: КПІ ім. Ігоря Сікорського, 2020. — 71 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 10 від 18.06.2020 р.)
за поданням Вченої ради ТЕФ (протокол № 10 від 25.05.2020 р.)

Електронне мережне навчальне видання

Кузьменко Ігор Миколайович, к.т.н., доц.

ТЕОРІЯ ГРАФІВ

Навчальний посібник для здобувачів ступеня бакалавра

за спеціальністю 122 "Комп'ютерні науки"

Відповідальний редактор *Гагарін О.О.,* к. т. н., доц. каф. АПЕПС

Рецензент: *Голінко І.М.,* к. т. н., доц. каф. АТЕП

Вирішення прикладних задач в області інформаційних технологій потребує адаптації інформації із загального опису на математичну основу, потім на мову алгоритмів, і далі - на конкретну мову програмування, або навпаки, в зворотному порядку. Знання способів побудови логічних математичних конструкцій на базі теорії графів, знадобляться всім, хто буде ефективно аналізувати інформацію та створювати ефективні програмні продукти.

Поряд з розглядом теоретичних питань, підручник містить приклади і задачі, які допоможуть зрозуміти основні поняття та практичні принципи їх використання.

Посібник призначений для студентів та для спеціалістів в області інформаційних технологій, які вивчають теорію графів самостійно.

ЗМІСТ

ВСТУП

Теорія графів у математиці займається вивченням особливого виду математичних структур - графів, що використовуються для моделювання парних відношень між об'єктами. Графи у цьому контексті складаються з вершин (точок), які з'єднані ребрами (лініями).

Враховуючи, що набір вершин можна використовувати для абстрагування будь-якого типу комп'ютерних даних, теорія графів глибоко вивчає взаємозв'язок між ними та дає відповіді на ряд питань розташування, налаштування мережі, оптимізації, узгодження та ін.

В такому ключі структури даних і їх використання вперше розглянув відомий математик Леонард Ейлер, який сформулював основні поняття теорії графів як розділу математики.

Теорія графів відіграє критичну роль у багатьох проблемах інформатики. Зокрема, теорія графів використовується для моделювання парних взаємозв'язків між об'єктами певної множини. Ряд комп'ютерних підходів було розроблено для полегшення використання графів, наприклад як SPANTREE або GTP для передачі даних.

Тобто, графи - це метод візуальної ілюстрації даних та відношень між ними. Мета графів - представити занадто численні, або складні дані, для їх адекватного опису в тексті, або алгоритмі. Забезпечення чіткості та коректність опису даних у графах забезпечують ефективність використання графів.

1. ОСНОВНІ ПОНЯТТЯ

Зображення графа містить точки та ребра. Однак, точки та ребра не є графом, це лише елементи зображення графа, як наприклад показано на рис.1.

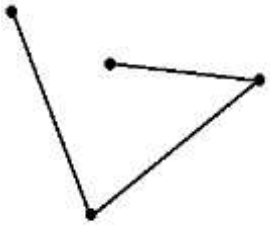


Рисунок 1 – Загальний вигляд графа

Щоб перейти до визначення графа розглянемо спочатку кілька прикладів.

Приклад 1. Розглянемо множину V студентів в аудиторії. Потужність множини |V| = 21. Якщо студенти сидять по двоє за партою і студентів позначимо точками, то студентів за однією партою можна позначити ребром. Маємо зображення, що показано на рис. 2. Із зображення зрозуміло , що парти стоять в два ряди, чотирнадцять студентів сидять по двоє за партою, п’ять студентів сидять по одному.

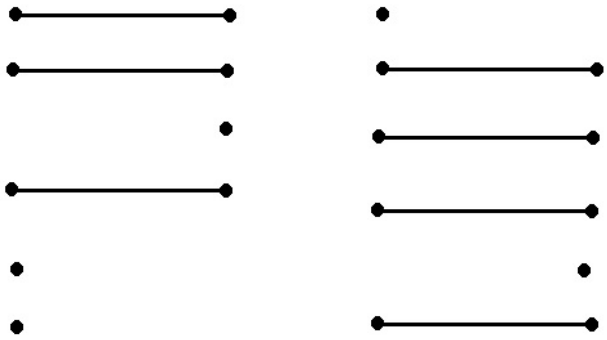


Рисунок 2 – Розміщення студентів за партами

На рисунку 3 показано розміщення студентів за партами, відповідно до прикладу 1. Із рис.3 видно, що в аудиторії залишилося 7 студентів, один студент сидить сам за партою, а ще шість студентів сидять попарно, проте парти перекомбіновані, але це не значить що вони розміщені одна на одній.

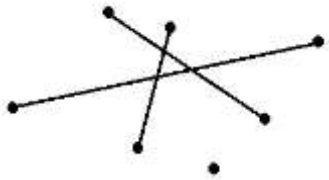


Рисунок 3 – Розміщення 7 студентів за партами

В обох випадках, описаних в прикладі 1 і показаних на рис 2-3 маємо справу з графом. Тобто, використання графа дозволяє підвищити інформативність передачі матеріалу і його засвоєння.

Можна сказати, що граф це не просто зображення на площині.

Граф – це абстрактний комбінаторний об’єкт, що складається з двох множин V (vertex) – множина вершин і E (edge) – множина ребер.

Саме тому в прикладі 1 кожне ребро має дві вершини – двох студентів, що сидять за однією партою.

Приклад 2 – Розглянемо ту ж множину студентів в аудиторії V, що являють собою вершини графа. А в якості відношення між вершинами візьмемо співпадіння днів народження – це буде множина ребер E. Тобто, V – множина студентів в аудиторії, E – множина людей зі співпадаючим днем народження.

Розглянемо приклад 3, що включає складнішу конструкцію – з орієнтованими ребрами.

Приклад 3. Повернемося до множини студентів в аудиторії V, а в якості відношення, що характеризує множину ребер, розглядатимемо симпатію між ними. В цьому разі симпатія є направленою. І може бути як односторонньою, так і взаємонаправленою, як показано на рис.4.

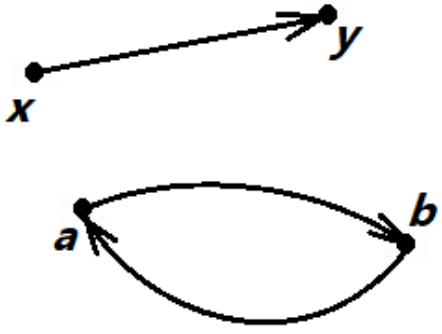


Рисунок 4 – Орієнтовані графи

Як видно з верхнього графа на рис. 4, E = (x, y), тобто x симпатизує y. Проте, зворотне не вірно (x, y) ≠ (y, x). В цьому разі ребра зображають зі стрілкою і називають дугами, а сам граф називають орієнтований граф, або скорочено орграф.

Із нижнього графа на рис.4 можна зробити висновок, що (a, b) = (b,a), тобто між a і b існує взаємна симпатія.

Приклад 4. Задача про Кенігсбергські мости. Ця задача сформульована в XVIII ст Леонардо Ейлером поклала початок теорії графів. Суть задачі в наступному. Ейлер, живучи в Кенігсберзі любив прогулюватися понад річкою і через мости. І от в нього виникало запитання чи існує такий шлях, щоб вийти з дому, обійти всі мости і повернутися додому але обійти мости так, щоб кожен міст пройти рівно один раз. Ейлер сам цю задачу сформулював і сам її розв'язав тим самим поклавши початок використанню графів у математиці. Більш того у цьому разі суша – це вершини графа, мости – ребра. Тоді схема мостів у задачі Ейлера приводить нас до графа, що показаний на рис. 6. Як видно з рис.6, з вершини до вершини є кілька ребер. Такі ребра називаються кратними, а сам граф носить назву – Мультиграфа.

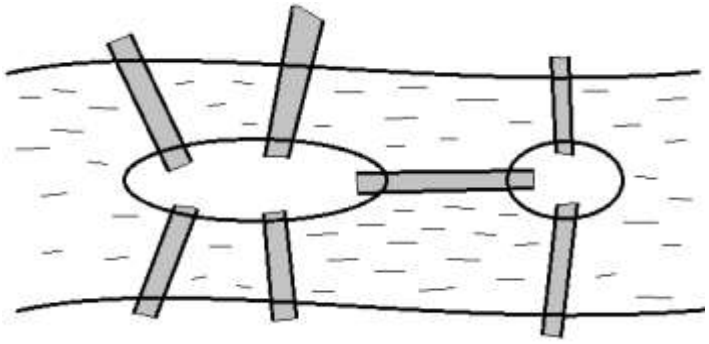


Рисунок 5 – Схема мостів у задачі Ейлера

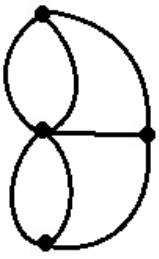


Рисунок 6 – Мультиграф до задачі Ейлера

В цілому ми розглянули три види графів: простий, орієнтований і мультиграф.

Приклад 4. Розглянемо структуру певного Web-сайту де вершинами будуть сайти, а ребрами – гіперлінки. У цьому разі зображення графа може мати вигляд, наведений на рис. 7. Зліва на рис. 7 показано гіперлінк сторінки на саму себе. Ребро такого типу називається петля. Як видно з рис. 7, петлі можуть бути кратні. Граф, що містить лише петлі (x, x) називається псевдографом.

Рисунок 7 – Граф Web-сайту

В цілому, граф на рис. 7 є ор-, мульти-, псевдографом.

Таким чином, підходимо до визначення. Граф G = <V, E> – це абстрактний комбінаторний об’єкт, що складається з двох множин. V – довільна множина вершин, або об’єктів. Множина може бути нескінченною. E- множина ребер, або пар об’єктів з V.

При цьому

1. Якщо немає кратних ребер, то кожна пара об’єктів зустрічається в множині V не більше одного разу.

2. Якщо це не орграф, то пари в E – не впорядковані.

3. Якщо немає петель, то (x, x) $\notin$ E.

Якщо пункт 1 не виконується – маємо мультиграф. Якщо пункт 2 не виконується – маємо орграф. Якщо пункт 3 не виконується маємо псевдограф. Можлива також комбінація цих пунктів, наприклад псевдо- мультиграф і т.п.

Тобто, графом можна оперувати не використовуючи його зображення на площині.

1.1. Маршрути в графах

Маршрут в графі – це спосіб пройти в графові по ребрах – від вершини, до вершини. Тобто, маршрут – це послідовність $v_1e_1v_2e_2\ldots v_ne_nv_{n+1}$ з вершини v_1 до вершини v_{n+1}. Вершини v_i до вершини v_{i+1} це кінці ребра e_i. Можна записати $e_i=(v_i, v_{i+1})$. Однак це не означає, що в маршруті всі ребра, або всі вершини, різні. Наприклад, для графа на рис. 8 маршрутом буде $v_1e_1v_2e_1v_1$ (як маршрут руху патрульного авто). Тобто, в маршруті вершини, або ребра можуть повторюватися. Якщо маршрут замкнений то перша і остання вершини – співпадають: $v_1 = v_{n+1}$. Довжина маршруту вимірюється кількістю ребер, що входять до його складу. Якщо в маршруті всі ребра – різні і при цьому маршрут замкнений (вершини можуть повторюватися), то маршрут носить назву – цикл. Наприклад, на рис. 8 $v_2e_2v_3e_3v_4e_4v_2$– цикл.

Якщо в маршруті всі ребра – різні і при цьому маршрут не замкнений, то маршрут носить назву – ланцюг. Наприклад, на рис. 8 $v_1e_1v_2e_2v_3e_3v_4e_4v_2$ – ланцюг. Якщо в ланцюгу, або в циклі вершини не повторюються (крім першої, яка є останньою в циклі) то такий ланцюг, або цикл називається простим. На рис. 9 показано простий ланцюг і простий цикл.

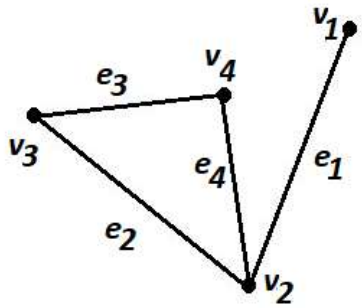


Рисунок 8 – Маршрут у простому графі

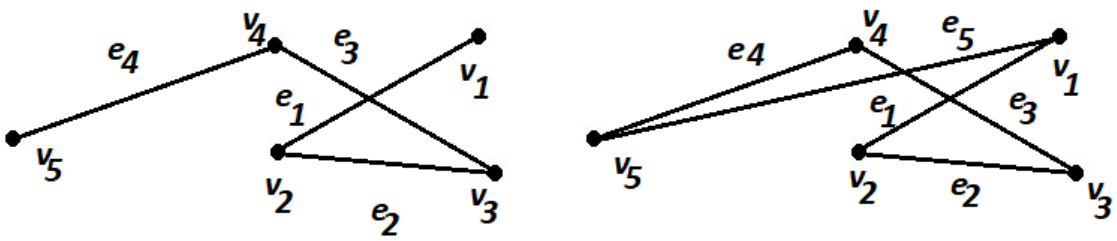


Рисунок 9 – Простий ланцюг (зліва) і простий цикл

На основі маршруту дається визначення зв’язності графа. Граф називається зв’язним, якщо між будь-якими його вершинами існує маршрут. Відповідно у не зв’язного графа не існує маршруту між будь-якими його вершинами. Не зв’язний граф, показаний на рис.10, розглядається за компонентами зв’язності графа. Кожна з компонент зв’язності не зв’язного графа має маршрут між будь-якими вершинами однієї компоненти. Тобто, компонента зв’язності графа – це така зв’язна частина графа до якої не можна додати жодної вершини графа без порушення зв’язності.

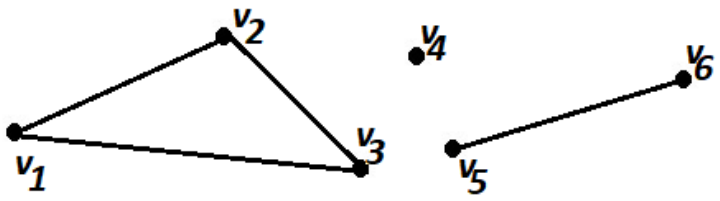


Рисунок 10 – Три компоненти зв’язності не зв’язного графа G

Ребро, вилучення якого з графа збільшує кількість компонент зв’язності, називають міст. Наприклад, у простого ланцюга будь-яке ребро буде мостом. Якщо вилучення вершини (разом з інцидентними їй ребрами) збільшує кількість компонент зв’язності графа, то така вершина називається точка з’єднання, або шарнір. Знову таки, якщо розглядати простий ланцюг, то будь-яка вершина, крім першої та останньої буде точкою з’єднання.

Відстань d(u,v) між будь-якими двома вершинами u і v зв’язного графа – це найкоротший маршрут, який дорівнює кількості ребер у маршруті. Для простого графу відстань від вершини u до вершини v і навпаки від вершини v до вершини є тотожними.

Ексцентриситет e(u) вершини u зв’язного графа це відстань від даної вершини до найвіддаленішої вершини. Можна сказати, що ексцентриситет – це максимальний маршрут для даної вершини у зв’язному графі, або найдовший серед найкоротших маршрутів від даної вершини до всіх інших.

Діаметр d(G) зв’язного графа G визначається як найбільший ексцентриситет серед ексцентриситетів всіх вершин даного графа G. Відповідно, радіус r(G)

зв’язного графа G визначається як найменший ексцентриситет серед ексцентриситетів всіх вершин даного графа G.

Якщо ексцентриситет вершини u графа G тотожний r(G), тобто e(u) =r(G) - така вершина називається центральною. Якщо ексцентриситет вершини u графа G тотожний d(G), тобто e(u) = d(G) - така вершина називається периферійною.

Центр графа – множина центральних вершин, тобто вершин з мінімальним ексцентриситетом. Наприклад, для графа G, зображеного на рис. 10 б) центром буде множина вершин { b, c}.

Рисунок 10 б) – До визначення центру графа

Приклад 5. Для графа G на рис. 11 визначити мости, розділювальні вершини, відстані d(1, 7) та d(10,2), ексцентриситети всіх вершин графа, діаметр та радіус графа, центральні та периферійні вершини.

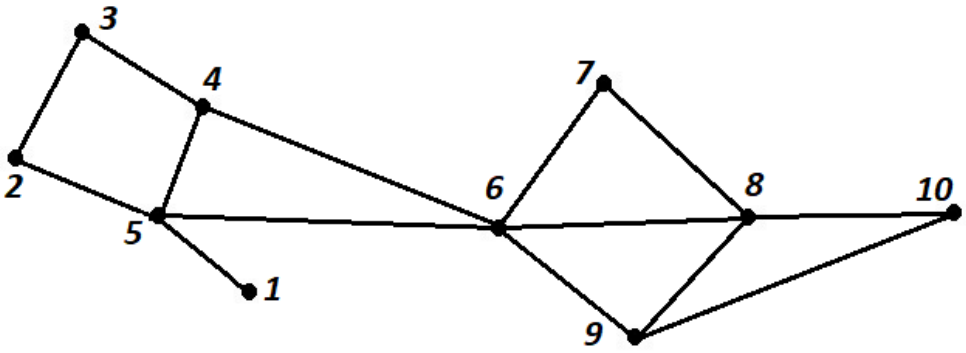


Рисунок 11 – Граф G до прикладу 5

Відповідь: для графа G мостом є лише ребро (1, 5), оскільки у разі його видалення вершина 1 залишається окремою компонентою зв’язності графа G. Аналогічно, розділювальними вершинами є 5та 6 для даного графу G, оскільки у разі видалення будь-якої з вершин 5 або 6 (разом з інцидентними їй ребрами) у графа з’являється дві компоненти зв’язності. Відстань d(1, 7) =3 оскільки найкоротший маршрут складає три ребра. Відповідно, d(10,2) = 4. Ексцентриситети кожної з вершин графа G, відповідно до визначення: e(1) = d (1, 10)=4, e(2) = 4, e(3)

= 4, e(4) = 3, e(5) = 3, e(6) = 2, e(7) = 3, e(8) = 3, e(9)= 3, e(10) = 4. Діаметр графа d(G) = 4, радіус графа r(G) = 2. Центральна вершина для графа 6, що має мінімальний ексцентриситет а периферійних вершин кілька – 1, 2, 3, 10, що мають максимальний ексцентриситет.

Від компонент зв'язності графа варто відрізняти частину графа, або підграф – як її ще називають. Для графа G = <V, E>, що включає вершини v_1, v_2, v_3, v_4, v_5, v_6 на рис.10 довільним підграфом G' є частина графу, яка включає вершини v_1, v_2, v_3. В загальному випадку, підграф G' = <V', E'> такий, що V' є підмножиною множини V, де $V' \subseteq V$ і при цьому $E' \subseteq \{(x,y) \in E \wedge x, y \in V'\}$ тобто частина ребер підграфа може бути відсутня, хоча вершини цих ребер входять в підграф. Наприклад, для графа G, показаного на рис. 10, підграфом також може бути множина вершин (v_5, v_6) без ребра, що їх з'єднує.

Якщо $E' \subseteq \{(x,y) \in E \wedge (x, y) \in V'\}$ - тобто всі ребра підграфу присутні для вершин, що входять в підграф – то такий підграф називається породженим підграфом. Породжений підграф – це такий підграф, у якого всі ребра заданого графу на даній множині вершин – збережено.

Якщо повернутися до рис. 10, то варто відмітити, що вершини, які з'єднані одним ребром називають суміжними. Наприклад, вершини v_1, v_2 – суміжні, оскільки їх з'єднує ребро (v_1, v_2). Аналогічно, ребра, які з'єднані однією вершиною теж називають суміжними. Наприклад ребра (v_1, v_2) і (v_2, v_3) – суміжні, оскільки їх з'єднує вершина v_2.

Вершина, що належить ребру, або ребро, що має цю вершину одним зі своїх кінців – називаються інцидентними. Наприклад, на рис.10 ребро e_1 = (v_1, v_2) від вершини v_1 до v_2 є інцидентним вершині v_1. Або вершина v_2 є інцидентною ребру e_1.

Якщо взяти певну вершину v_i та визначити кількість інцидентних їй ребер графа G, то це значення носить назву – степінь вершини і позначається $\deg(v_i)$. Наприклад, для графа G, поданого на рис. 10 маємо $\deg(v_2)=2$, оскільки вершині v_2 є інцидентними два ребра (v_1, v_2) та (v_2, v_3). Аналогічно, $\deg(v_4) = 0$, оскільки

вершина v_4 не має інцидентних ребер. Вершина, для якої deg(v) = 0 називається ізольованою, тому v_4 на рис. 10 – ізольована вершина.

Оскільки у будь-якого графа G кожне його ребро має дві вершини, то подвоєна кількість ребер графа G тотожна сумі інцидентних ребер всіх вершин, або сумі степеней всіх вершин графа G:

$$\sum_{v \le V} \deg(v) = 2|E| \qquad (1)$$

1.2. Способи представлення графів

Використовується цілий ряд способів представлення графів, що обумовлено областю їх використання. Наприклад, матричне представлення графів використовується в інформаційних технологіях.

1. Явний спосіб. Граф розглядається як дві множини G=<V, E> де V – множина вершин V={a, b, c, d} і E – множина ребер E={ (a, b), (a, c), (b,d), (b, c)}.

2. Графічний спосіб. Подання графа у вигляді зображення є поширеним, оскільки це зручний і наочний спосіб. Для графа G, описаного в способі 1, маємо зображення на рис. 12 зліва (1), або справа (2). Будь-яке з цих двох зображень графа G буде вірним і, навіть, можна сказати, що існує нескінченна множина таких зображень одного й того ж графа.

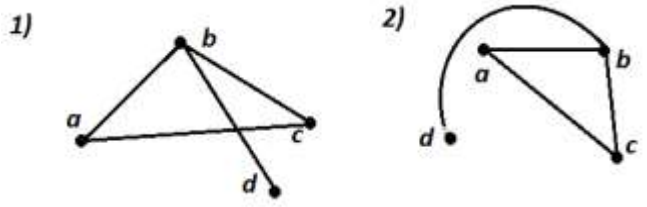


Рисунок 12 – Два зображення графа G

3. Матриця суміжності. Граф подається у вигляді матриці суміжності A_G. Спосіб використовується при роботі комп'ютера з графами. Припустимо граф G має певну кількість вершин n, тобто - потужність множини вершин графа |V| = n. Тоді можна записати матрицю A_G, розміром nxn, яка містить множину елементів $A_G = (a_{i,j})_{nxn}$. Тут індекси і та j змінюються від 1 до n: $i, j = \overline{1,n}$, а значення $a_{i,j}$ приймає

значення 1, коли між відповідними вершинами v_i та v_j існує ребро, або приймає значення нуль, якщо такого ребра не існує. Тобто,

$$a_{i,j} = \left[\left(v_i, v_j \right) \in E \right] = \begin{cases} 1, \left(v_i, v_j \right) \in E \\ 0, \left(v_i, v_j \right) \notin E \end{cases} \quad (2)$$

Як приклад, запишемо матрицю суміжності для графа G, показаного на рис. 10. Граф має шість вершин, тому n=6 і квадратна матриця має розмір 6x6.

A_G =

	v1	v2	v3	v4	v5	v6
v1	0	1	1	0	0	0
v2	1	0	1	0	0	0
v3	1	1	0	0	0	0
v4	0	0	0	0	0	0
v5	0	0	0	0	0	1
v6	0	0	0	0	1	0

З означення матриці суміжності (2) витікає наступне: для не орієнтованого графа матриця суміжності завжди симетрична відносно головної діагоналі (відношення між вершинами є симетричними, користуючись апаратом теорії відношень); для не орієнтованого графа матриця відображає відношення сусідства вершин графа.

4. Матриця інцидентності. Матриця B_G зв'язує вершини та ребра в деякому графі G. Як і попередній, спосіб використовується при роботі з графами на комп'ютері $B_G = (b_{i,j})_{nx|E|}$ де $i, j = \overline{1,n}$. $b_{i,j}$ приймає значення 1, якщо вершина v_i є кінцем ребра e_j та навпаки – приймає значення нуль, якщо умова не виконується.

У якості прикладу представимо граф G, поданий на рис. 10, матрицею інцидентності. Позначимо ребра графа наступним чином $e_1=(v_1, v_2)$, $e_2=(v_2, v_3)$, $e_3=(v_1, v_3)$, $e_4=(v_5, v_6)$. Маємо наступну матрицю інцидентності

B_G =

	e1	e2	e3	e4
v1	1	0	1	0
v2	1	1	0	0
v3	0	1	1	0
v4	0	0	0	0
v5	0	0	0	1
v6	0	0	0	1

Для простого графа маємо дві важливі властивості матриці інцидентності. Перша полягає в наступному - кожен стовпчик матриці інцидентності містить рівно дві одиниці. Друга властивість матриці інцидентності – сума елементів кожної зі строк матриці дає значення степені вершини.

Зокрема, для вищенаведеної матриці, строка v_4 дає суму елементів строки рівну нулеві, що відповідає $\deg(v_4) = 0$ для графа G, відповідно до рис. 10.

5. Списки суміжності. Спосіб включає перерахунок всіх вершин, суміжних даній. Тобто:

$\forall v \in V$ виконується $\Gamma(v) = \{v = V \mid (v_i, v_j) \in E\}$.

Наприклад, для графа G, показаного на рис. 10, маємо наступні списки суміжності

Γ(v1) = {v2, v3},

Γ(v2) = {v1, v3},

Γ(v3) = {v1, v2},

Γ(v4) = ∅,

Γ(v5) = {v6},

Γ(v6) = {v5}.

Можна також відзначити, що в літературі також вказують на інші способи представлення графів. Однак, можна вважати, що вищенаведені п’ять способів є основними.

Задачі для самостійного розв’язання

Дано простий граф що має 8 вершин. Чи може бути наступна послідовність чисел степенями вершин цього графа?

1) 5,4,3,2,2,2,2,1.

Відповідь – ні. Оскільки сума степеней вершин є непарним числом, що протирічить (1).

2) 7,6,4,1,1,1,1,1

Відповідь – ні. Оскільки сума степеней двох перших вершин 7+6=13 свідчить, що їм є інцидентними 13 ребер. Разом з цим, сума степеней всіх інших вершин 4+5*1=9 свідчить, що їм є інцидентними 9 ребер. Проте 13-9=5 – п’ять ребер не вистачає.

3) 7,6,5,4,3,2,1,0

Відповідь – ні. Оскільки першій вершині є інцидентними 7 ребер, а вершин з яких виходять ребра лише шість. З останньої вершини жодне ребро не виходить, оскільки степінь її вершини нуль.

4) 6,6,5,4,3,2,2,2. – так. Варіант зображення графа наведено на рис. 13.

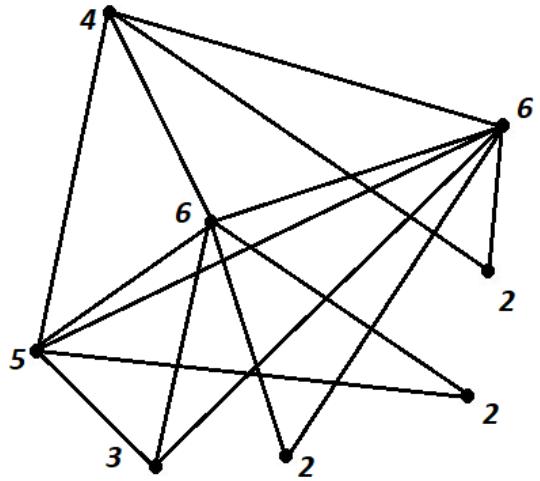


Рисунок 13 – Варіант зображення графа

1.3. Дерева в теорії графів

Найпростіші зв’язні графи на яких базується цілий клас в теорії алгоритмів задач носить назву – Дерева. Дерево G= <V, E> це зв’язний ациклічний (без циклів) граф. З того, що граф G – зв’язний, витікає, що між будь-якими його вершинами є маршрут типу $v_1e_1v_2e_2….e_nv_n$. Однак, оскільки граф G – ациклічний, то він завжди простий. Такий граф схожий на дерева всі гілки якого зв’язані, але не мають циклів, див. рис. 14.

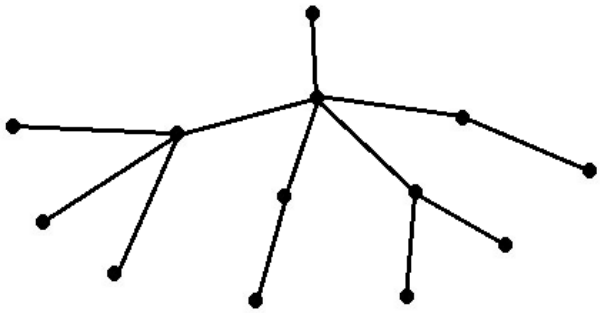


Рисунок 14 – Дерево G

Оскільки на основі зображення дерева не можливо встановити властивості, тому сформулюємо і доведемо теорему про характерні властивості дерева в теорії графів.

Теорема: наступні визначення справедливі і рівносильні:

1) G – дерево – тобто, зв’язний і ациклічний граф.

2) Між будь-якими двома вершинами графа G існує єдиний і простий ланцюг. Тобто, вершина і ребра не повторюються.

3) G – зв’язний і кількість ребер на одне менше кількості вершин.

4) G –ациклічний граф і кількість ребер на одне менше кількості вершин.

Тобто, з визначення 3 витікає ациклічність, як написано в 1. А з визначення 4 витікає зв’язність, як написано в 1.

Доведення. Побудуємо доведення наступним чином: якщо виконується визначення 1, то виконується визначення 2, якщо виконується визначення 2, то виконується визначення 3, якщо виконується визначення 3, то виконується визначення 4, якщо виконується визначення 4, то виконується визначення 1. Таким чином, прийдемо до висновку, що всі визначення є істині та рівносильні.

1) Доведемо що з визначення 1, витікає визначення 2. Тобто, маємо G – дерево яке є зв’язний і ациклічний граф. Довести, що існує єдиний і простий ланцюг. Розглянемо доведення контрапозицією. Припустимо існує не єдиний ланцюг, або не простий і не ланцюг рис. 15. Тобто, якби не проходив другий ланцюг – цикл буде утворено. Проте, за умовою G – ациклічний. Тому припущення – не вірно. Тобто, із визначення 1, витікає визначення 2 – між будь-якими двома вершинами графа G, який є деревом, існує єдиний і простий ланцюг.

Рисунок 15 – До 1 частини теореми про властивості дерева

2) Доведемо що з визначення 2, витікає визначення 3. Тобто, існує єдиний і простий ланцюг. Довести, що G – зв'язний і кількість ребер на одне менше кількості вершин.

Доведення. Якщо в G існує єдиний і простий ланцюг між будь-якими двома вершинами, то існує маршрут між будь-якими двома вершинами і граф G – зв'язний. Першу частини доведено.

Далі за індукцією доведемо, що кількість ребер на одиницю менша, ніж кількість вершин n графа G.

База індукції: n=1 оскільки мова йде про простий ланцюг, не псевдограф, то кількість ребер рівна нулеві і умова виконується.

Візьмемо n=2 – в цьому разі граф може складатися з двох вершин G_1, або з двох вершин та ребра між ними G_2, як показано на рис. 16.

Рисунок 16 – До 2 частини теореми про властивості дерева

Однак, оскільки за умовою існує ланцюг, то маємо G_2 (рис. 16). Відповідно, кількість ребер на одиницю менша, ніж вершин і умова виконується. Далі розширимо твердження за індукцією для всіх графів, де кількість ребер k > n+1. Розглянемо граф G, що має (n+1) вершин і існує єдиний і простий ланцюг. Тоді існує маршрут з вершини x до вершини y і він єдиний. Тобто, існують ребра від x до y. Припустимо від x до y одне ребро e=(x,y) і ми його видаляємо. Тоді з графа G отримаємо граф G'(показаний на рис. 17), що має дві компоненти зв'язності, а вершини x і y будуть у різних компонентах зв'язності графа G'(якщо це було б не так, то було б ще одне ребро, принаймні). Зокрема, вершина x належить компоненті зв'язності H1, а вершина y – компоненті зв'язності H2.

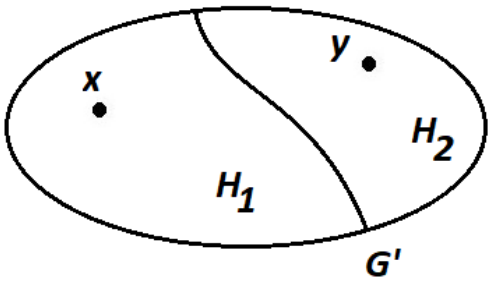


Рисунок 17 – Граф G’

Оскільки граф G має (n+1) вершин, компоненти зв’язності H_1 та H_2 мають відповідно n_1 та n_2 вершин. До того ж n1<n+1, оскільки вершина y ∉ H1 окрім цього між будь-якими вершинами H1 існує єдиний і простий ланцюг. З іншого боку, відповідно до визначення 3, H1 має (n1-1) ребер. Для компоненти зв’язності H2 маємо кількість вершин n2<n+1, і H2 має (n2-1) ребер. В загальному, кількість вершин в обох компонентах графа відповідає кількості вершин в графі в цілому $n_1+n_2 = (n+1)$. (2)

Кількість ребер у графі G відповідає кількості ребер у H1 та кількість ребер у H2 (n1-1) та (n2-1) а також ребро e= (x, y). Тобто, (n1-1)+(n2-1)+1= n1+n2-1 = n з урахуванням (2). Тобто, у графа G є (n+1) вершин та n ребер. Що і потрібно було довести.

3) Доведемо, що з визначення 3 витікає визначення 4. Тобто, дано граф G - зв’язний, що має n вершин і (n-1) ребер. Довести, що граф G – ациклічний (без циклів).

Доведення побудуємо методом контрапозиції. Для цього припустимо, що в графі G є цикл який проходить через k вершин графа. Відповідно k≤n, де n - кількість вершин графа G. Припустимо цикл в графі G – простий. Тоді він складається з k-ребер, як показано на рис. 18.

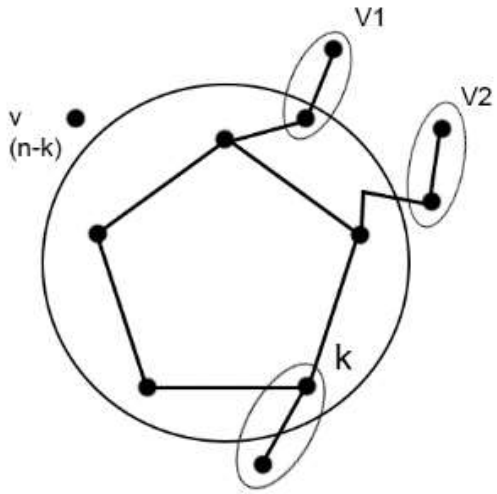


Рисунок 18 – Кількість вершин і ребер у циклі

На рис. 18 показано граф G в якому n вершин k вершин, що входять до циклу та (n-k) – вершин, що не входять до циклу графа G.

Оскільки граф G - зв’язний, то для будь-якої вершини v_i графа G справедливо, що то кожна вершина графа з’єднана з циклом у графі G як мінімум, одним ребром, або маршрутом з кількох ребер.

1) Для вершини vi оберемо серед всіх маршрутів, які з’єднують її циклом у графі G, найкоротший за кількістю ребер. Для кожного з найкоротших маршрутів розглянемо перше ребро в маршруті.

Ці ребра – всі відмінні, оскільки вони формують найкоротший маршрут. Якщо два маршрути почнуться з одного ребра, то вони не будуть найкоротші. Як видно з рис. 19, маршрут від vi до циклу в графі G, що проходить через vi+1 буде складатися з двох ребер і тому не буде найкоротшим. Тому, кількості вершин (n-k) поза циклом графа G відповідає (n-k) відмінних ребер.

В цілому ребер маємо (n-k) +k = n. Однак, за умовою маємо лише (n-1) ребер. Тому припущення про існування циклу – не вірне, а граф G – ациклічний. Що і потрібно було довести.

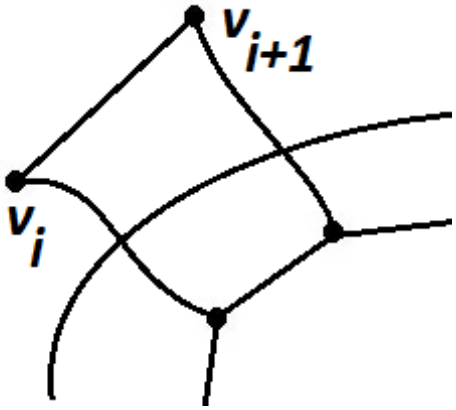


Рисунок 19 – До визначення маршрутів у графі G

4) Доведемо, що з визначення 4 витікає визначення 1. Тобто, дано граф G – ациклічний, що має n вершин і (n-1) ребер. Довести, що граф G – дерево і зв'язний граф.

Доведення. Для прямого доведення позначимо k – кількість зв'язних компонент графа G і доведемо, що k=1. Припустимо що ni – кількість вершин графа G в і-тій компоненті зв'язності графа. Тоді n1, n2,…., nk – кількість вершин в відповідних компонентах зв'язності 1,2,…,k. В цілому маємо

n1+n2+…+nk = n, де n – кількість вершин у графі G.

Оскільки G має n вершин і (n-1) ребер, то можна записати що кількість ребер

(n1-1)+(n2-1)+…+(nk-1) = (n1+n2+…+nk) - k

З іншого боку, для графа G маємо кількість ребер (n-1), за умовою. Тобто

(n1+n2+…+nk) – k = n -1.

Звідси k=1, кількість компонент зв'язності графа – одна. Що і потрібно було довести.

Тобто, доведено окремо кожне визначення і тому доведено всю теорему в цілому.

В подальшому розглянемо задачі, пов'язані з зображенням графів на площині.

Питання для самоконтролю

1. Чи може бути мультиграф псевдографом ?

2. Чи може бути граф, що складається лише з вершин?
3. Чи може бути граф, що складається лише з ребер?
4. Чи може бути ексцентриситет вершини графа більший за його діаметр?
5. Чи може бути ексцентриситет вершини графа більший за його радіус?
6. Чи може ребро графа не мати інцидент них вершин?
7. Які способи представлення графів існують?
8. В чому різниця між матрицею суміжності і матрицею інцидентності?
9. Що коректне: будь-яке дерево є графом, або навпаки - будь який граф є деревом?
10. Скільки циклів має дерево?

2. ЗОБРАЖЕННЯ ГРАФА НА ПЛОЩИНІ

Граф G= <V, G> називається планарним, якщо можна так зобразити його на площині, що перетини його ребер на зображенні будуть лише у вершинах. Або по іншому: планарний граф ізоморфний деякому плоскому графу, а плоский граф – це граф у якого ребра на зображенні не перетинаються.

Ізоморфний граф – це граф, отриманий з даного шляхом зміни його зображення на площині.

На рис. 20 зображено планарний граф, що має чотири вершини і його ребра перетинаються лише в цих вершинах.

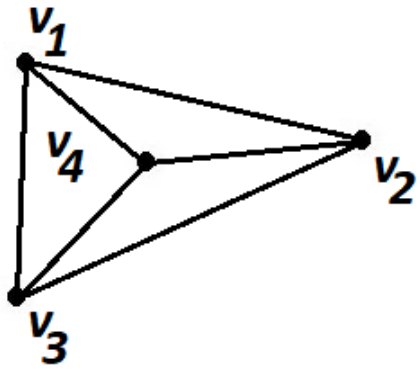


Рисунок 20 – Планарний граф

На наступному рис. 21 зліва показано граф на вершинах {a,b,c,d}, який є планарним, оскільки граф зліва ізоморфний плоскому графу на вершинах {v10,v20,v30,v40}, зображеному справа

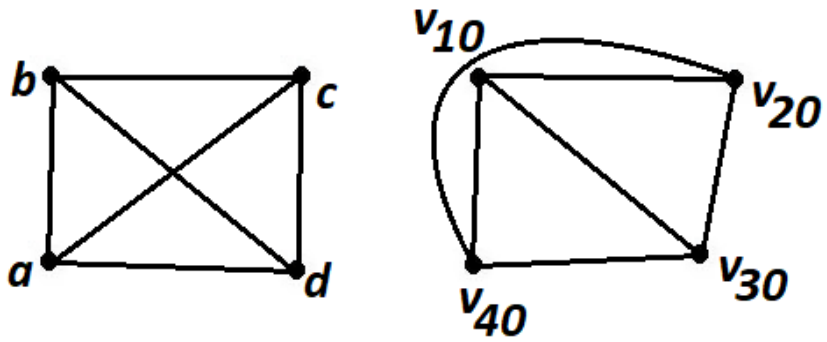


Рисунок 21. Планарний граф та ізоморфний йому плоский граф (справа)

Можливо, всі графи планарні? Ні.

Приклад повного не планарного графа K_5 на п’яти вершинах показано на рис. 22. Як не змінюй його зображення на площині, а перетини ребер, що не є вершинами будуть присутні. Також вище вказано К5 – позначення повного графа на п’яти вершинах.

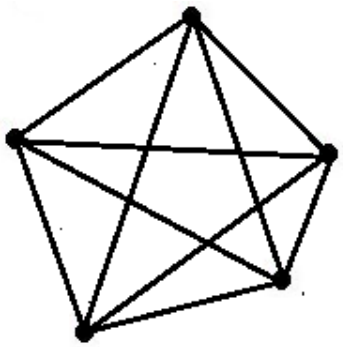


Рисунок 22 – Не планарний повний граф K_5

Другий приклад не планарного графа К3,3, що показаний на рис. 23, пов'язаний з задачею про три будинки і три колодязі. Суть задачі в наступному: від трьох будинків йде по три стежинки до трьох колодязів. І як би не прокладалися ці стежинки, або як би не переносилися будинки і колодязі - стежини обов’язково будуть перетинатися. Вище вказано позначення К3,3 – повного дводольного графа на трьох парах вершин.

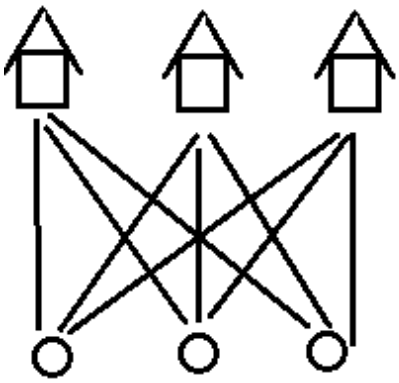


Рисунок 23 – Граф К3,3

В загальному випадку Кn,m – повний дводольний граф із двома підмножинами, що містять n та m вершин кожна і ребрами, які з’єднують вершини із цих підмножин.

Оскільки К3,3 дводольний (рис.24), то множина вершин V цього графа розбивається на дві підмножини так, що будь-яке ребро графа має вершину vi, що належить множині непарних чисел (або першій множині) та вершину vi+1, що належить множині парних чисел (або другій множині).

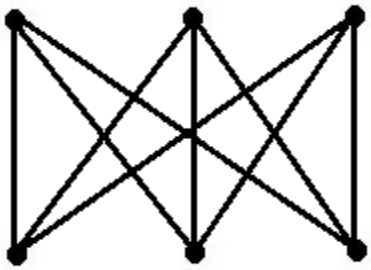


Рисунок 24 – Абстрактне зображення повного дводольного графа К3,3

2.1. Гомеоморфізм графів

З графа К5 можна зробити ще один граф, який буде відрізнятися від К5 більшою кількістю вершин, або ребер, і основне, що новий граф теж буде не планарний. Наприклад, графи на рис. 25 не є графом К5, не включають граф К5, як підграф, і не є ізоморфними до графа К5 (оскільки мають іншу кількість вершин та ребер). Хоча графи G1 та G2 на рис. 25 є повними та не планарними, кожен з них.

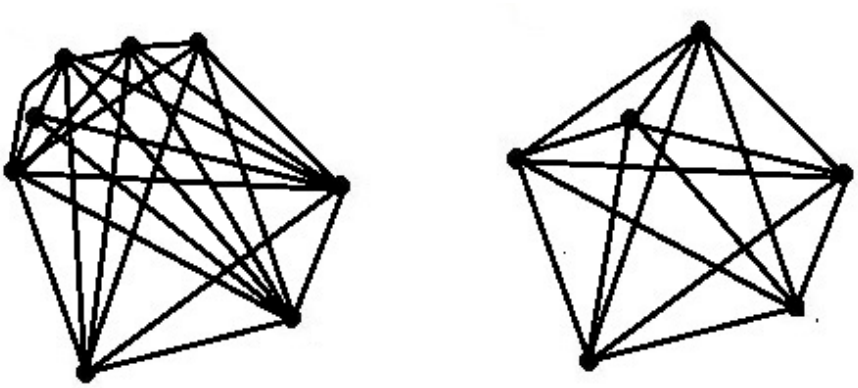


Рисунок 25 – Графи G1 та G2, отримані з графа K5

На рис. 26 показано не планарний граф, не повний (тобто, не всі вершини з’єднані між собою) граф, отриманий з повного дводольного графа K3,3. Граф на рис. 26 не є графом K3,3, не включає граф K3,3, як підграф, і не є ізоморфним до графа K3,3 (оскільки має іншу кількість вершин та ребер).

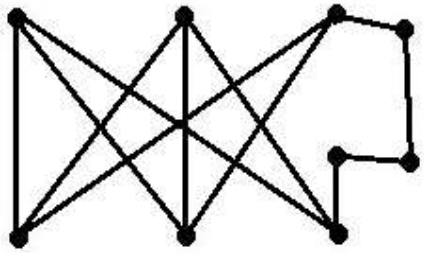


Рисунок 26 – Граф, отриманий з графа K3,3

Графи, показані на рис. 25 і 26 отримані з графів K5 та K3,3 шляхом додавання вершин на деякі ребра. Графи, що отримані додаванням вершин на деякі ребра називаються гомеоморфними.

Тобто, маємо - всі гомеоморфні графи, отримані з K5 та K3,3 будуть не планарними.

Тоді, можливо існує багато не планарних графів? Відповідь на це питання дав Казимир Куратовський в 1930 р. Наведемо без доведення теорему Куратовського.

Граф є планарним тоді і лише тоді, коли у ньому немає підграфів, гомеоморфних графам K5, або K3,3.

Тобто, інших не планарних графів, крім графів типу К5, або К3,3, або схожих на них – не існує. І якщо граф не схожий на граф К5, або граф К3,3, то він планарний.

Для плоского графа сформулюємо наступні характеристики: v – вершини, e – ребра, f – грані плоского графа. Грань – це частина, на яку граф ділить площину. Оскільки граф однозначно задає розбиття площини на певну кількість областей.

Наприклад, на рис. 27 зображено граф К4, що має 4 вершини, 6 ребер і 4 грані (маємо vi i=1…4, ei i=1…6, fi i=1…4). Із чотирьох граней f1, f2,f3 обмежені циклом, довжини три, що утворює три внутрішні області і четверта f4– зовнішня грань відповідає зовнішній області, яка є необмежена.

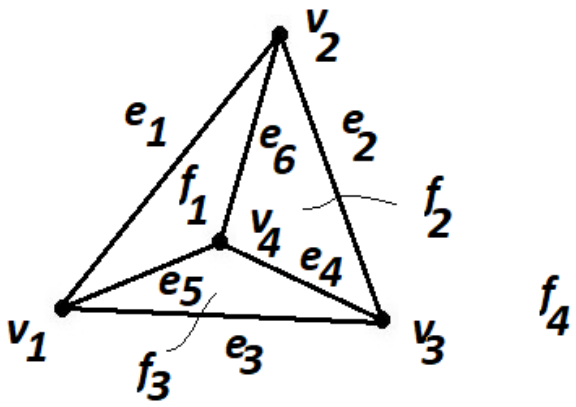


Рисунок 27 – Плоский граф К4

На рис. 28 зображено планарний плоский граф, тобто граф ребра якого не перетинаються. Виходячи з зображення, може бути не зрозуміло де і скільки граней у графа, де внутрішня/зовнішня області. На ці питання дає відповідь теорема Жордана.

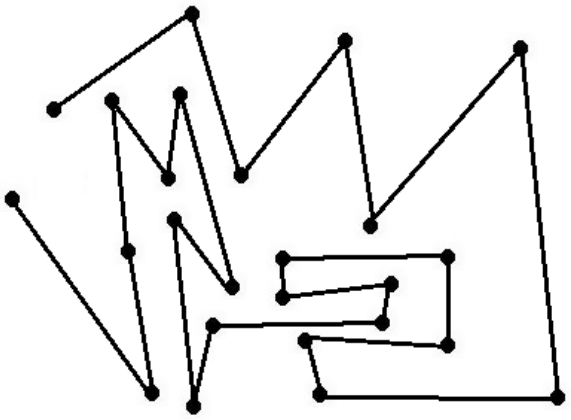


Рисунок 28 – Планарний плоский граф

Теорема Жордана: якщо крива неперервна, замкнута і без самоперетинів, то як би вона не виглядала, вона буде розмежовувати рівно дві області. Одна з яких внутрішня, відносно кривої, а інша – зовнішня.

Наприклад, на рис. 29 показано граф, ребра якого на площині є кривою, яка неперервна, замкнута і без самоперетинів. Тому ця крива розмежовує дві області, які утворюють внутрішнє f1 та зовнішнє f2 ребро графа.

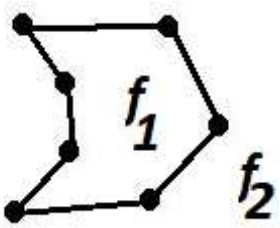
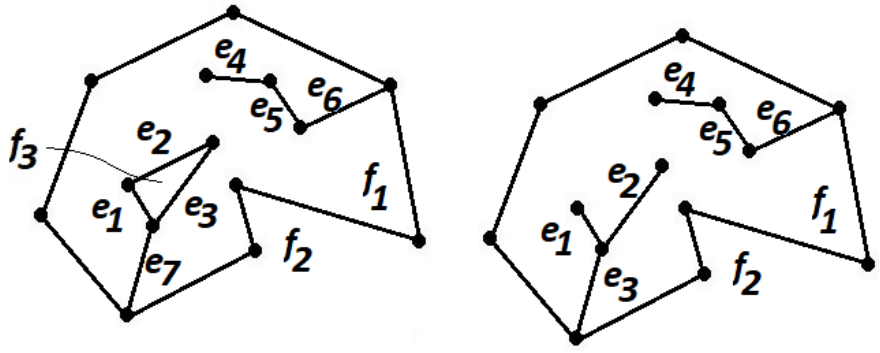


Рисунок 29 – Граф до теореми Жордана

У планарного графа грань може мати вигляд, наприклад, річки на мапі. У цьому разі, якщо ребра e1, e2, e3 утворюють циклічну границю, то утворюється нова грань f3, як показано на рис. 30 а). Якщо ребра не формують циклічну границю для грані, то вони є внутрішніми ребрами цієї грані. Наприклад, на рис. 30 б) ребра e1,e2,…,e6 є внутрішніми ребрами для грані f1.



а) б)

Рисунок 30 – Графи з внутрішніми ребрами

Якщо повернутися до дерев, то очевидно, що будь-яке дерево є плоским графом без циклів (як зображено на рис. 31). Для довільного дерева маємо кількість

граней f=1 і ця грань – зовнішня. Відповідно, всі ребра дерева будуть внутрішніми ребрами по відношенню до цієї грані.

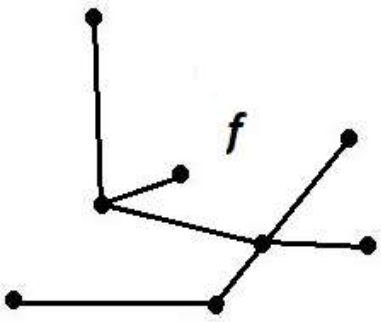


Рисунок 31 – Грань f дерева

Кількість ребер, граней та вершин зв’язана між собою формулою Ейлера.

Формула Ейлера: якщо плоский граф є зв’язним, то кількість його вершин v і граней f на два більше, ніж кількість його ребер e.

$|v|+|f| = |e|+2.$ (3)

Наприклад, для графа на рис.31 маємо $|v| =8$, $|f| = 1$, $|e| =7$ і $|v|+|f| = |e|+2$ – формула Ейлера виконується.

Якщо граф не є зв’язний, то формула змінюється

$|v|+|f| = |e|+1+k,$

де k – кількість компонент зв’язності графа.

Наприклад, для графа на рис. 32, маємо $|v| = 8$, $|f| = 4$, $|e| = 9$, $k = 2$ і $|v|+|f| = |e|+1+k$ – змінена формула Ейлера виконується.

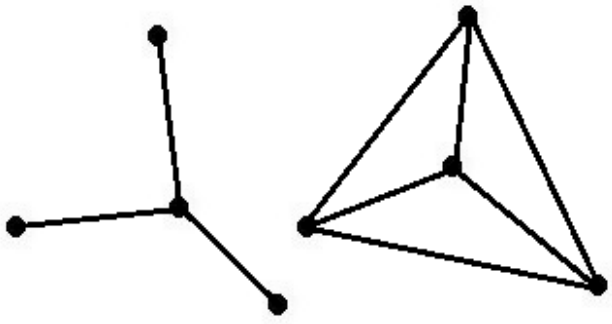


Рисунок 32 – Зв’язний плоский граф

Доведемо за індукцією формулу Ейлера (3).

Доведення: якщо граф – звязний, то $e \geq v-1$, оскільки в дерева, як мінімально можливого зв’язного графа $e = v-1$ і за умови зростання кількості ребер маємо $e \geq$

v-1. Якщо граф – дерево, то |f| =1 і e = v-1. Тоді |v|+|f| = |e|+2 = v-1+2 = v+1. База індукції доведена для всіх дерев, які мають |e| = n ребер і |v| = (n+1) вершин.

Розширимо базу індукції для дерев, які мають |e| = n+1 ребер і |v| = (n+1) вершин. У цьому разі новий граф не буде деревом, оскільки нове (n+1) ребро утворить цикл. Утворення циклу призведе до утворення нової грані у графі. Тобто, в дерево G додали ребро і отримали нову грань в новому плоскому графі G'. До того ж граф G' - циклічний:

G: v, e, f $\rightarrow$ G: v, e+1, f+1. (*)

Якщо до дерева G додали k ребер, то це означає, що процедуру (*) повторили k разів і отримали k нових граней в новому плоскому графі G'. До того ж граф G' - циклічний

Тобто, формула Ейлера (3) залишається справедливою для всіх плоских графів.

Що і потрібно було довести.

Із формули Ейлера (3) для плоского зв'язного графа маємо наступну теорему.

Теорема: якщо в плоского зв'язного графа вершин більше, ніж дві v ≥ 3, то кількість ребер не перевищує e ≤ 3v-6.

Зрозуміло, якщо у зв'язного графа v=2, то e=1 і теорема не виконується.

Доведення: визначимо розмір грані – це кількість ребер на її границі плюс подвоєна кількість внутрішніх ребер. Наприклад, граф на рис. 33 має розмір грані f1=17 і f2 = 7.

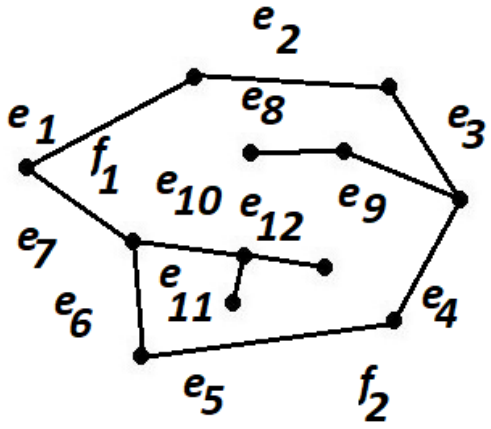


Рисунок 33 – До визначення розміру грані

Розглянемо довільну грань. Якщо її розмір не перевищує два, то цикл – відсутній. Якщо внутрішні ребра – відсутні, то цикл (грань графа) має принаймні три ребра і розмір кожної грані – не менше ніж 3. Як видно з рис. 33, всі ребра, крім внутрішніх, входять в грані f1 та f2. Тобто, кожне ребро, крім внутрішніх, двічі входить в розмір грані: 2|e| = Σ сума розмірів граней.

З іншого боку, сума розмірів граней ділена на три дає грань Σ/ 3 = |f|, оскільки три ребра утворюють грань. Маємо

2|e| = Σ = 3|f|, або |f| = 2e/3.

Підставимо це значення у формулу Ейлера (3) |v|+2|e|/3= |e|+2. Отримаємо |e|=3(|v|-2)

Якщо грань утворюють більше, ніж три ребра, то маємо

2|e| = Σ > 3|f|, або |f| < 2e/3.

І відповідно 2-|v|+|e|=f< 2|e|/3. Звідси 2-|v|+|e|< 2|e|/3. Отримаємо |e|<3(|v|-2).

Тобто, в загальному випадку e ≤ 3v-6. Що і потрібно було довести.

Із вищенаведеної теореми слідує, що граф К5, зображений на рис. 22 – не планарний. Оскільки для К5 v = 5, e = 10 і теорема не виконується 10> 3*5-6. Тому К5 – не плоский і не може бути планарним.

Також із вищенаведеної теореми, що слідує з формули Ейлера, витікає розв’язок «проблеми чотирьох фарб», що пов’язана з хроматичним числом графа.

2.2. Хроматичне число графа

Хроматичне число графа витікає із задачі сформованої в 1852 р. Френсісом Гасрі – яка мінімальна кількість кольорів потрібна, щоб розформувати карту. До того ж карту фарбують так, щоб країни (без анклавів), які мають спільні кордони, мали різні кольори і розрізнялися на карті. На даний час відомо, що для розфарбування карти вистачить 4 кольорів, але сама задача зводиться до теорії графів. Припустимо, столиці країн є вершинами графа і мають колір, що відповідає кольору країни, як показано на рис. 34. У цьому разі маємо граф, суміжні вершини якого будуть мати відмінний колір. До того ж кількість кольорів вершин має бути мінімальною, а граф отриманий на основі карти, буде планарним. Виходить, що

будь-який планарний граф має хроматичне число, що не перевищує чотири. Чотири фарби буде достатньо, щоб розфарбувати вершини планарного графа у різні кольори.

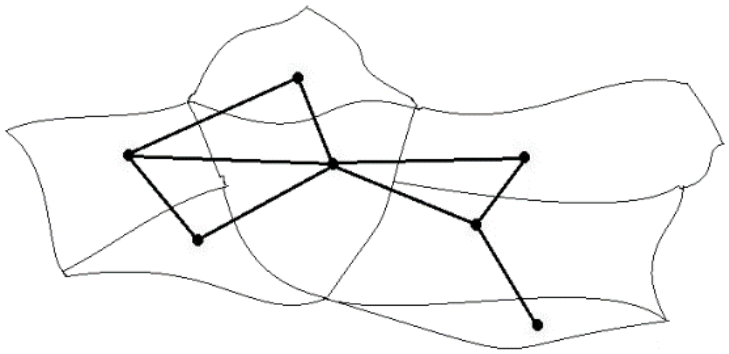


Рисунок 34 – Граф до задачі про мінімальну кількість кольорів суміжних вершин

Таким чином, в термінах теорії графів дана задача зводиться до поняття – хроматичне число графа $\chi(G)$, або ще називають кольорове число графа.

Хроматичне число графа $\chi(G)$ – найменша кількість кольорів у які можна розфарбувати його вершини, щоб суміжні вершини, з'єднані одним ребром, були різного кольору.

Тобто, вершини кластеризуються за кольором

$\chi(G) = \min\{x: V=V1 \coprod \ldots \coprod \rightarrow V\chi \ \forall i \forall x,y \in Vi \ (x,y) \notin E\}$

Формально хроматичне число плоского графа визначається теоремою.

Теорема: хроматичне число будь-якого плоского графа $\chi(G) \leq 4$.

Доведення теореми виконано методом грубої сили. Тобто, спочатку вручну, а потім з допомогою комп'ютерів, перебрали всі відомі варіанти карт і встановили істинність теореми.

Інших способів доведення вищевказаної теореми – не існує на даний час.

Однак існує, хоча і досить об'ємне, доведення наступної теореми: хроматичне число будь-якого плоского графа $\chi(G) \leq 5$.

А доведення теореми: : хроматичне число будь-якого плоского графа $\chi(G) \leq 6$ рекомендується для самостійного розбору, оскільки це доведення займає менше сторінки.

Задача: яка мінімальна кількість ребер у графа на 12 вершинах і з трьома компонентами зв'язності?

Розв’язок: Оскільки мінімальну кількість ребер серед графів має дерево, то кількість ребер у першій компоненті зв’язності графа (n1-1), у другій компоненті зв’язності графа (n2-1), у третій компоненті зв’язності графа (n3-1). Тут n1, n2, n3 – кількість вершин у відповідній компоненті зв’язності n1+ n2+ n3 = n = 12

Маємо

e = (n1-1)+ (n2-1)+ (n3-1)=n-k =9

Тобто, мінімальна кількість ребер у графа на 12 вершинах і з трьома компонентами зв’язності складає 9, як показано на рис. 35.

Рисунок 35 – Граф, що має 3 компоненти зв’язності

2.3. Формула Келі для дерев

В цілому формула дозволяє обрахувати кількість різних дерев на n помічених вершинах. Вершини дерева V = { 1, 2, …, n} називаються поміченими, якщо вони розрізняються за номерами, або кольором.

Варто також відрізняти поняття «різні» дерева від поняття «ізоморфні» дерева. Почнемо з останнього.

Дерева G1 =<V1, E1>, G2 =<V2, E2> (як і графи) називаються ізоморфні (з грец. ізо – рівний; морф - форма), $G_1(V_1, E_1) \approx G_2(V_2, E_2)$ якщо вони мають рівну кількість вершин та ребер $|V_1| = |V_2|$, $|E_1| = |E2|$ та їх можна привести до однакової форми. Приклад ізоморфних дерев показано на рис. 36.

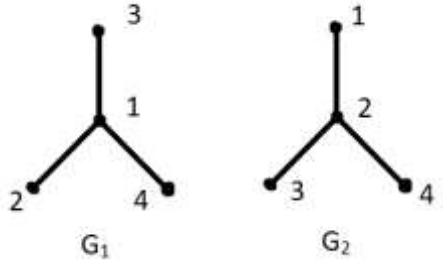


Рисунок 36 – Ізоморфні дерева

Дерева G1 =<V1, E1>, G2 =<V2, E2> (як і графи) називаються різними $G_1 \neq G_2$, якщо існує як мінімум одне ребро, яке належить одному дереву і не належить іншому. Наприклад, відповідно до рис. 36, ребро $e=(1,4) \in G_1$ і $e=(1,4) \notin G_2$ є. Тобто, ребра (1,4) в дереві G_2 немає. З прикладної точки зору, наприклад якщо дерево відображає кристалічну решітку хімічного елемента, то різні дерева відповідатимуть різним хімічним елементам.

Тому хоча дерева, зображені на рис. 36 будуть $G_1 \approx G_2$ (ізоморфні) - форми у них однакові, але з точки зору наявності ребер, G_1 і G_2 - різні дерева.

Саме формула Келі визначає скільки різних дерев можна отримати на n помічених вершинах. Позначимо t_n – кількість дерев на n вершинах і знайдемо t_1, t_2, t_3…

Для дерева з однією вершиною, кількість різних дерев – рівно одне $t_1 = 1$. Для дерева з двома вершинами (рис. 37), кількість різних дерев теж одне $t_2 = 1$. До того ж в цьому разі (та в подальшому) не важливо дерево розглядається справа-наліво, чи зліва-направо.

Рисунок 37 – Дерево з двома поміченими вершинами

Кількість різних дерев з трьома поміченими вершинами (рис. 38), складає $t_3 = 3$. Оскільки $t_3 = P(3)/2$ визначається як перестановки всіх трьох вершин зменшені вдвічі, оскільки розглядати справа наліво, чи зліва направо - немає різниці. Тут і далі P(n) – перестановки n помічених вершин між собою.

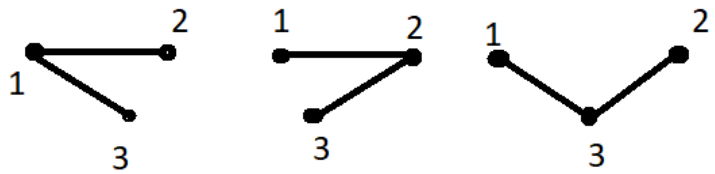
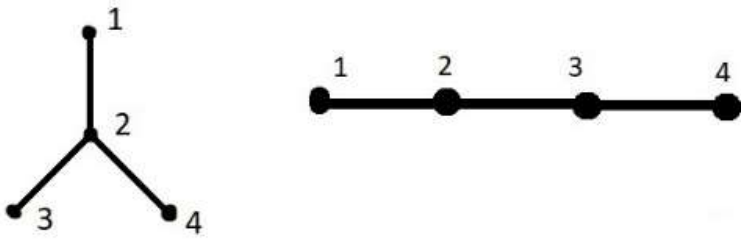


Рисунок 38 – Різні дерева з трьома поміченими вершинами

Кількість різних дерев з чотирма поміченими вершинами (рис. 39) складає $t_4 = 16$. Оскільки $t_4 = 4+P(4)/2$, де перший доданок відповідає дереву на рис. 39 а) з 4

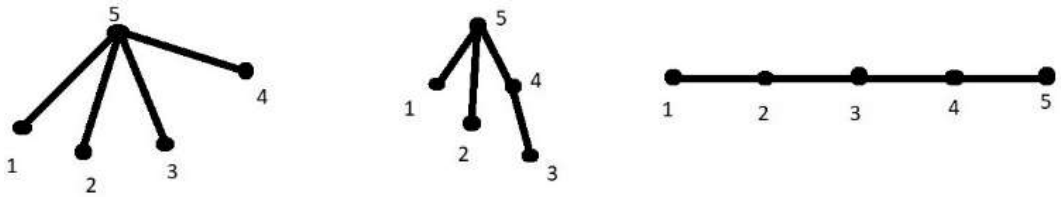
помічених вершин. Кількість таких дерев, що дорівнює 4 визначається значенням вершини в центрі, незалежно від розміщення висячих вершин. Другий доданок відповідає дереву на рис. 39 б) і визначається як перестановки всіх 4 вершин зменшені вдвічі, оскільки розглядати справа наліво, чи зліва направо - немає різниці.



а) б)

Рисунок 39 – Різні дерева з чотирма поміченими вершинами

Кількість різних дерев з п’ятьма поміченими вершинами (рис. 40), складає t_5 = 125. Оскільки t_5 = 5+P(5)/2+P(5)/2, де перший доданок відповідає дереву на рис. 40 а) з 5 помічених вершин. Кількість різних дерев, що дорівнює 5 визначається значенням вершини в центрі, незалежно від розміщення висячих вершин. Другий і третій доданки відповідають різним деревам на рис. 40 б) і в) визначаються як перестановки всіх 5 вершин зменшені вдвічі, оскільки розглядати справа наліво, чи зліва направо - немає різниці.



а) б) в)

Рисунок 40 – Різні дерева з п’ятьма поміченими вершинами

Тобто, маємо наступну кількість різних дерев з поміченими вершинами $t_1 = 1^{-1}$, $t_2 = 2^0$, $t_3 = 3^1$, $t_4 = 4^2$, $t_5 = 5^3$. Або, в загальному випадку маємо формулу Келі

$t_n = n^{n-2}$.

Істинність формули Келі за 130 років має різні доведення. Розглянемо доведення Прюфера.

Для доведення встановимо відповідність між множиною всіх послідовностей, довжиною (n-2) із n натуральних чисел та множиною різних дерев на n помічених вершинах.

Розглянемо множину всіх послідовностей, довжиною (n-2), які складаються з n чисел, де $n \in \{1, 2, \ldots, n\}$. Потужність множини всіх послідовностей (кількість таких послідовностей) визначається як кількість перестановок n елементів в множині з (n-2) елементів, як показано на рис. 41.

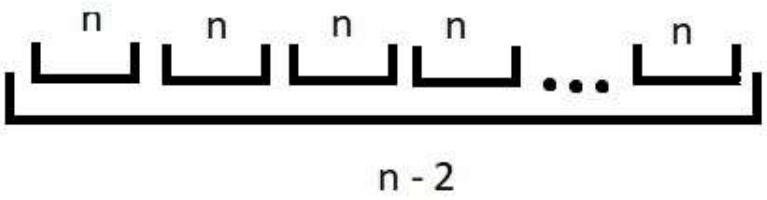


Рисунок 41 – До визначення кількості перестановок n елементів в множині з (n-2) елементів

Наприклад, за аналогією, кількість тризначних бінарних чисел визначається як кількість перестановок двох елементів (нуль і один) у множині з трьох елементів $2^3=8$.

З іншого боку покажемо, що множина різних дерев з (n-2) елементів на n помічених вершинах, знаходиться у взаємооднозначній відповідності з множиною таких послідовностей.

Тобто, встановивши таку відповідність, формула $t_n = n^{n-2}$ буде доведена.

Припустимо, така відповідність існує, тобто кожному дереву відповідає певна послідовність чисел, код. Тому послідовність Прюфера називають також код Прюфера.

Наприклад, встановимо_відповідність між множиною всіх послідовностей, довжиною (n-2) із n натуральних чисел та множиною різних дерев із n помічених вершинах для дерева із 11 вершин, показаного на рис. 42. Збудуємо код Прюфера.

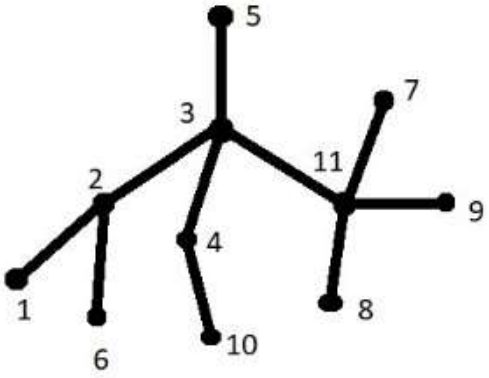


Рисунок 42 – Дерево на 11 вершинах

Як пам’ятаємо, висяча вершина завжди має степінь вершини рівно один $deg(v_i) = 1$. Тому на рис. 42 вершини 1, 6, 10, 8, 9, 7, 5 – висячі. У вищевказаного дерева з $n \neq 1$ поміченими вершинами знайдемо висячу вершину з мінімальним номером.Це буде вершина 1, яка суміжна ребру (1, 2).

Далі в код Прюфера вносимо другу вершину, що суміжна ребру – 2, а саме ребро - видаляємо. Дерево матиме вигляд, показаний на рис. 43, а в код Прюфера заносимо 2.

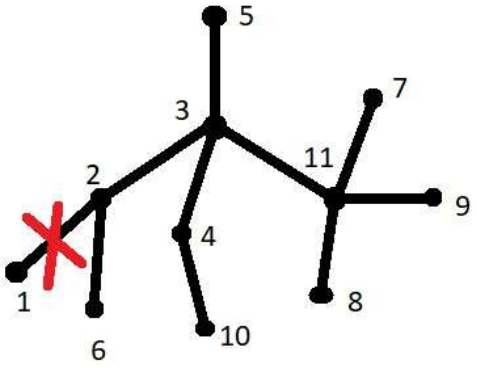


Рисунок 43 – Дерево без ребра (1, 2)

Граф, що залишився, показаний на рис. 43, і це теж дерево. Тому знову серед помічених вершин, які залишилися, знайдемо висячу вершину з мінімальним номером. Це буде вершина 5, яка суміжна ребру (5, 3).

Далі в код Прюфера вносимо другу вершину, що суміжна ребру – 3, а саме ребро (5,3) - видаляємо. Дерево матиме вигляд, показаний на рис. 44, а код Прюфера матиме вигляд 2 3.

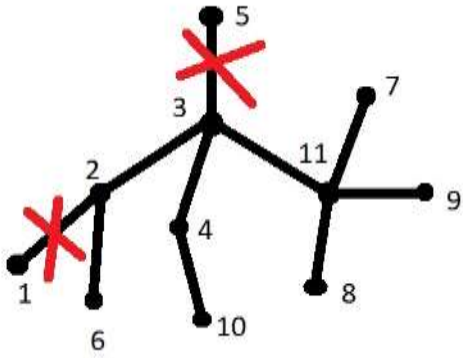


Рисунок 44 – Дерево без ребер (1, 2) і (5, 3)

Знову серед помічених вершин, які залишилися, знайдемо висячу вершину з мінімальним номером. Це буде вершина 6, яка суміжна ребру (6, 2).Далі в код Прюфера вносимо другу вершину, що суміжна ребру – 2, а саме ребро (6, 2) - видаляємо. Дерево матиме вигляд, показаний на рис. 45, а код Прюфера матиме вигляд 2 3 2.

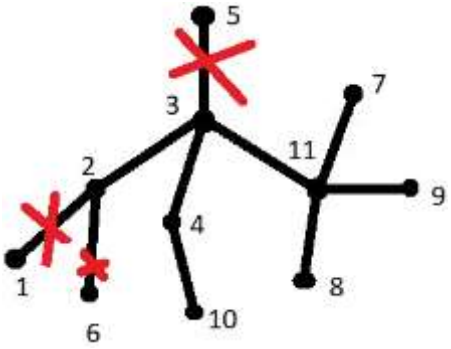


Рисунок 44 – Дерево без ребер (1, 2), (5, 3) і (6,2)

Надалі мінімальною вершиною буде 2 – видаляємо ребро (2,3), заносимо в код 3; далі - 7, видаляємо ребро (7, 11), заносимо в код 11; далі 8, видаляємо ребро (8,11), заносимо в код 11; далі - 9, видаляємо ребро (9, 11), заносимо в код 11; далі – 10, видаляємо ребро (10,4), заносимо в код 4; далі -4, видаляємо ребро (4, 3), заносимо в код 3. Повторяючи процедуру видалення ребер, отримаємо останнє ребро (3,11), що залишиться і код Прюфера у вигляді: 2 3 2 3 11 11 11 4 3.

Як видно, отримана множина містить 9 чисел із множини в 11 чисел. Очевидно, що для різних дерев коди Прюфера будуть різнитися.

Надалі покажемо, що отриманий код Прюфера відповідає дереву на рис. 42. Для цього опишемо процедуру декодування.

Код Прюфера: 2, 3, 2, 3, 11, 11, 11, 4, 3 (**)

Його нумерація: 1 2 3 4 5 6 7 8 9 10 11 (*)

Його нумерація (*) – множина натуральних чисел {1, 2,..., 11}, n =11 має числа, яких немає в (**) коді Прюфера. Тому:

1) З (*) виберемо число з мінімальним номером, якого немає в (**) – 1
2) З (**) видаляємо число під номером1. Це буде двійка.
3) Запишемо (1,2) як перше висяче ребро. Одиницю із (*) викреслюємо.

Повторяємо пп.1-3:

1) З (*) виберемо число з мінімальним номером, якого немає в (**) – 5
2) З (**) видаляємо число під номером 2. Це буде трійка.
3) Запишемо (5, 3) як висяче ребро. П'ятірку із (*) викреслюємо.

Далі (6,2) – третє ребро з висячою вершиною 6. Далі (2, 3), (7,11), (8,11), (9, 11), (10, 4), (4, 3). На цей момент із коду (**) буде викреслено всі числа, а в номерації (*) – залишаться два 3 і 11, які сформують останнє ребро (3,11).

Залишилося довести, що така відповідність між деревом і кодом Прюфера взаємооднозначна.

Оскільки ми збудували (n-1) ребро на n вершинах, то відповідно до теореми про визначення дерева, отриманий граф за умови його ациклічності – буде дерево. Доведемо, що при декодуванні не буде циклів, тобто граф – ациклічний.

І таким чином буде доведено, що існує відповідність між множиною (n-2) натуральних чисел та множиною n вершин дерева.

Кількість різних дерев на n помічених вершинах $t_n = n^{n-2}$

Однак, якщо зв'язаний граф має лише один цикл, то він називається уніциклічний (рис. 45). В літературі наводиться теорема про U_n - кількість уніциклічних графів на n вершинах, яка вказує, що

$$U_n = \sum_{r=3}^{n} C_n^r \frac{r!}{2} \cdot n^{n-r-1}$$

Наслідок з теореми для n -> ∞ : $U_n \approx \sqrt{\frac{\pi}{8} \cdot n^{n-\frac{1}{2}}}$, або $\lim_{n\to\infty} \frac{U_n}{\sqrt{\frac{\pi}{8} \cdot n^{n-\frac{1}{2}}}} = 1.$

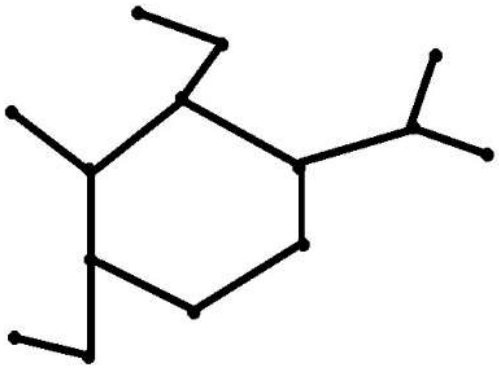


Рисунок 45 – Уніциклічний граф

Тобто, множина уніциклічних графів більша, потужніша, ніж множина дерев на n помічених вершинах. Виходячи з цього, за індукцією можна довести, що різні графи на n помічених вершинах є ациклічні. А отже є деревами.

Таким чином, істинність формули Келі з використанням коду Прюфера – доведена.

В множину уніциклічних графів також входить також ліс, як множина дерев. Якщо з уніциклічного графу видалити ребра, що утворюють цикл (єдиний і простий цикл) – залишиться ліс, який складається із r дерев, які мають n вершин і r – компонент зв'язності.

Ліс – незв'язний неорієнтований граф без циклів, а зв'язані компоненти лісу – є деревами. На рис.46 показано ліс, що містить 6 – компонент зв'язності, тобто - 6 дерев (виділені компоненти належать різним деревам.

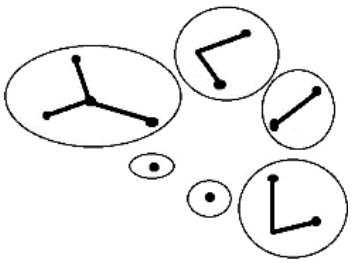


Рисунок 46 – Ліс із 6 компонентами зв'язності

2.4. Центральна вершина дерева

В кожного дерева G існує вершина відстань від якої до інших вершин дерева - найменша. Якщо таких вершин буде кілька, то їх множина називається – центр дерева.

Доведемо істинність наступного твердження.

У разі, якщо довжина найдовшого ланцюга дерева (діаметр дерева) d(G) = 2n, де n ∈N, то центральна вершина v дерева G має ексцентриситет (найкоротший маршрут до найвіддаленішої вершини), що не перевищує n e(v, x) ≤ n, де x ∈ V. Припустимо, що довжина найдовшого ланцюга дерева показана на рис. 47.

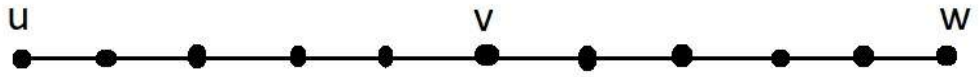


Рисунок 47 – Центральна вершина дерева

v –центральна вершина, u, w – висячі вершини дерева.

Доведення побудуємо методом контрапозиції. Припустимо, що існує така центральна вершина для якої ексцентриситет перевищує n ∃ x ϵ V: e(x, u) ≥ (n+1)

Оскільки розглядається дерево, то і найкоротший маршрут буде простим ланцюгом, оскільки у дерева за визначенням відсутні цикли. Тобто простий ланцюг з'єднує u і v. Також простий ланцюг з'єднує w і v. Однак в останньому випадку e(w, v) < (n+1).

Тобто, припущення, що центральна вершина дерева має ексцентриситет, що перевищує n: e(x, u) ≥ (n+1) – не вірне.

Твердження доведене.

Питання для самоконтролю

1. Як пов'язані питання планарності та ізоморфність графів?
2. Які приклади дводольних графів знаєте? чи може бути дводольний граф не планарним?
3. Що таке гомеоморфність?
4. Як співвідносяться поняття планарності та гомеоморфність у графах?
5. Скільки внутрішніх ребер має дерево?
6. Скільки граней має дерево?
7. Чи входить у формулу Ейлера кількість граней?
8. Як пов'язана кластеризація вершин із хроматичним числом графа?

9. Що описує формула Келі?
10. Чи можна кодом Прюфера закодувати граф?

3. ЦИКЛІЧНІ ГРАФИ

3.1. Ейлерів цикл

Повернемося до задачі Ейлера про сім мостів. Схема, що наведена на рис. 5, зводиться до мультиграфа, показаного на рис. 48.

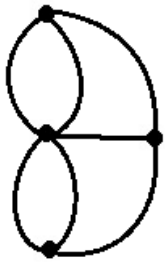


Рисунок 48 – Мультиграф до задачі Ейлера

Задача Ейлера звучить так - чи можна вийти з деякої початкової вершини, пройти всіма ребрами рівно один раз і повернутися в ту ж вершину?

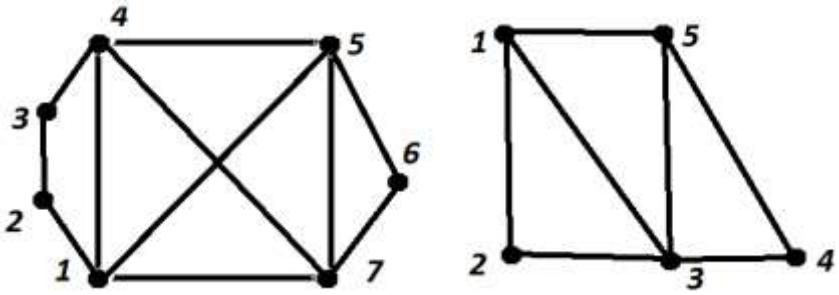
Відповідь дав Леонард Ейлер – не можна. Але Ейлер не був би Ейлером, якби він не пішов далі. Він встановив - щоб це зробити потрібно, щоб у кожної вершини був вхід та вихід. Тобто, щоб степінь кожної вершини графа була парним числом.

Ейлерів цикл в графі G (який називають ейлерів граф) – якщо існує v_i така, що вийшовши з неї та пройшовши через всі ребра рівно один можна повернутися в цю ж вершину v_i.

Тобто ейлерів граф являє собою цикл в тому розумінні, що це – маршрут в якому всі ребра – різні. На рис. 49 а) показано ейлерів цикл, що проходить через вершини 1-2-3-4-5-6-7-1-4-7-5-1.

Якщо маршрут через всі ребра по одному разу не замкнений (тобто, циклу немає), такий граф називають – напівейлерів, а маршрут називають ейлерів. На рис. 49 б) показано напівейлерів граф у якому ейлерів маршрут проходить через вершини 1-2-3-4-5-3-1-5.

Тобто, знаємо, що степінь вершин має бути парною, щоб граф був Ейлеровим.



а) б)

Рисунок 49 – Ейлерів цикл а) та ейлерів маршрут б)

Ейлер довів, що остання умова є необхідна і достатня одночасно.

Теорема Ейлера. Для зв'язного графа G наступні три твердження є еквівалентні:

1. G – ейлерів граф (мультиграф і зв'язний).
2. Степінь кожної з вершин графа G є парною.
3. Множина ребер графа G розпадається в об'єднання циклів, що попарно не перетинаються. Тобто в них немає спільних ребер. Хоча можуть бути спільні вершини.

Доведення опустимо. Скажемо лише, що доводиться наступним чином з пункту 1 витікає доведення пункту 2 і т.д.: 1 => 2=> 3 => 1

3.2. Гамільтонів цикл, Гамільтонів граф

Іншою задачею обходу графа, сформульованої Вільямом Гамільтоном, є задача навколосвітньої подорожі. Подорож, за задумом Гамільтона, мала включати відвідування всіх столиць і це відвідування мало бути рівно один раз. Звісно, після подорожі слід було повернутися в початкову точку.

Якщо столиці розглядати як вершини графа, а шлях між ними – як ребра, то отримаємо граф, маршрут у якому включає всі вершини рівно один раз і цей маршрут є циклом.

Тому, на відміну від Ейлерового циклу, який проходить через всі ребра рівно один раз, Гамільтонів цикл проходить через всі вершини рівно один раз. Тобто, маємо:

Гамільтонів граф – граф, що містить Гамільтонів цикл. А Гамільтонів цикл – простий цикл, який містить всі вершини графа і оскільки простий, то повторення вершин– відсутні.

Задачі, пов’язані з визначенням оптимального маршруту доставки, будуються саме на Гамільтонових графах. Розглянемо кілька прикладів таких графів.

Приклад 1. Приклад задачі, яку розглядав Гамільтон в 1859 р., Земля зображена як додекаедр з 12-и плоских п’ятикутників, а вершини – столиці, які потрібно відвідати рівно один раз, показано на рис. 50. На цьому рисунку зображено граф G=<V,E>, який є проекцією додекаедра на площину і оскільки існує Гамільтонів цикл - простий цикл, який проходить через усі вершини графа (рівно один раз), такий граф є Гамільтоновим.

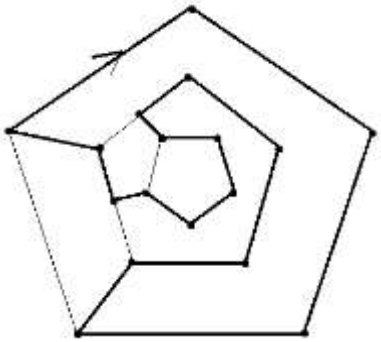


Рисунок 50 – Гамільтонів граф на проекції додекаедра

Приклад 2. На рис. 51 зображені графи G і H, які є Гамільтоновими, оскільки в графах G і H Гамільтоновий цикл проходить через ребра 1-2-3-4-5-6 та 1-2-3-4-5, відповідно.

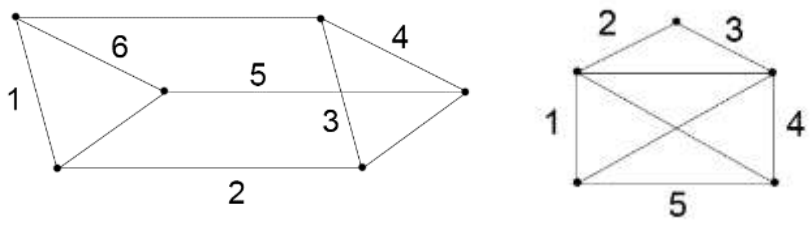


Рисунок 51 – Гамільтонові графи G та H (справа)

Приклад 3. Прикладом не Гамільтонового графа є граф на рис. 52. Яким би чином не будувався цикл через усі вершини цього графа, обов'язково будуть вершини, що потраплять в цикл більше, ніж один раз.

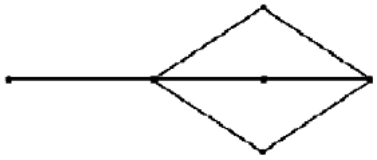
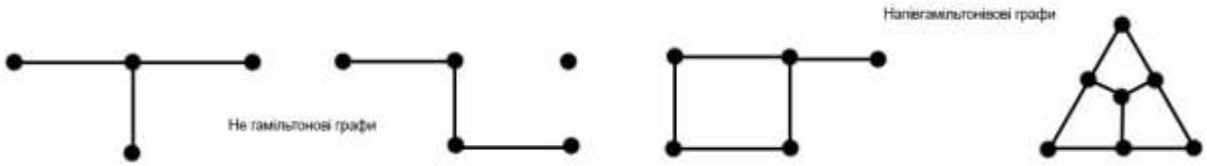


Рисунок 52 – Приклад не Гамільтонового графа

Графи, які не мають Гамільтонових циклів, але мають Гамільтонів ланцюг – напів-Гамільтонові. Гамільтонів ланцюг – маршрут, що містить всі вершини по одному разу. На рис. 53 показано не Гамільтонові та напів-Гамільтонові графи (H1, H2).



G1 G2 H1 H2

Рисунок 53 – Не Гамільтонові та напів-Гамільтонові графи

3.3. Умова існування Гамільтонового графа

На відміну від Ейлерового графа, де існує критерій існування циклу (парність степеней всіх вершин графа), для Гамільтонового графа такого критерія не існує.

Тобто критерія Гамільтонового графа немає, але є умова існування Гамільтонового графа. Умова існування сформульована і доведена Габріелем Діраком, ще одна умова існування сформульована і доведена Ойстайном Оре, наступна інша умова існування сформульована і доведена Джоном Бонді і Яцеком Хваталом та ряд інших.

Розглянемо дві найбільш відомі умови існування – умову Дірака і умову Оре.

Умова Дірака сформульована у вигляді теореми.

Теорема: Якщо будь-яка з вершин графа G=<V,E>, |V| = n має степінь не меншу, ніж половина потужності множини вершин графа ≥ n/2, то граф - Гамільтоновий.

Наприклад, на рис. 54 показано граф, який граф, що задовольняє умові Дірака: |V| = n = 6 і deg(b) = 5 > deg(a) = 3 ≥ 6/2 = 3. Граф, що задовольняє умові Дірака називають граф Дірака.

Для доведення теореми Дірака доведемо твердження 1, що граф у якого степінь кожної із вершин більша, або рівна половині від потужності множини його вершин n/2 а буде зв'язний.

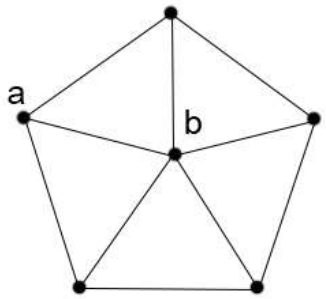


Рисунок 54 – Граф Дірака

Доведення твердження

Розглянемо довільні дві вершини зв'язного графа.

Якщо граф має всього дві суміжні вершини v та w, то deg(v) = deg(w) = n/2 = 2/2 = 1 і граф зв'язний.

Припустимо, що між v та w де v,w ∈ V для G=<V,E> немає ребра і доведемо, що граф G - зв'язний.

За умовою, кожна з вершин v та w має n/2 сусідів, як показано на рис. 55. Тобто, множина вершин, суміжних вершині v має потужність n/2 та множина вершин, суміжних вершині w має потужність n/2. В цілому потужність множини вершин графа G без вершин v та w (n – 2). Тобто n/2 + n/2 ≤ n-2, ця нерівність справедлива лише за умови, що вершини v і w мають спільні суміжні вершини. Або

{Множина сусідів v} ⋂ {Множина сусідів w} = ∅.

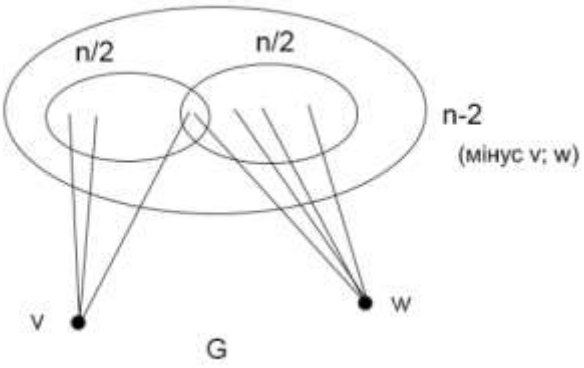


Рисунок 55 – Доведення зв’язності Гамільтонового графа

Тобто існує вершина, яка належить обом множинам сусідів v та w. Звідси, граф G – зв’язний. Що і потрібно було довести.

Повернемося до доведення теореми.

Візьмемо граф G(V,E), де |V|=n і розглянемо найдовший простий ланцюг, що проходить через всі вершини графа рівно один раз і доведемо, що він буде циклом. Припустимо, множина вершин цього циклу V = {x1, x2, … , xm} у графі G. Зрозуміло, що, оскільки це простий ланцюг, то m ≤ n, де n – загальна кількість вершин у графі G.

Припустимо, що серед множини вершин ланцюга існують вершини xi та xi+1 інцидентні одному ребру (xi, xi+1). Доведемо, що серед вершин ланцюга {x1, x2, … , xm} існує деяка вершина xi, що з’єднана ребром із вершиною xm, а суміжна до неї вершина xi+1 з’єднана ребром з x1 і таким чином доведемо, що ланцюг – цикл.

Для цього серед множини вершин ланцюга без вершини xm {x1,x2,…, xi, xi+1,…, xm-1} виберемо підмножину вершин так, щоб вершини, які входять в цю підмножину були з’єднані ребром із останньою вершиною xm. Розглянемо множину ребер, що з’єднують ці вершини з останньою вершиною xm I1 = {i: (xi, xm) ∈ E}. Відповідно до умов теореми Дірака для Гамільтонового графа кількість таких ребер не менша, ніж половина загальної кількості вершин |I1| ≥ n/2

Надалі серед множини вершин циклу виберемо підмножину вершин так, щоб вершини, які входять в цю підмножину були з’єднані ребром із першою вершиною x1. Розглянемо множину ребер, що з’єднують ці вершини з першою вершиною x1

I2 = {i: (x1, xi+1) ∈ E}. Відповідно до умов теореми Дірака для Гамільтонового графа кількість таких ребер не менша, ніж половина загальної кількості вершин |I2| ≥ n/2

Це знов всі ті ж вершини, які належать ланцюгу. Якщо кількість вершин у ланцюзі m:≤n, то m-1 < n.

Оскільки, якби був хоч один сусід поза нашим простим ланцюгом, то ланцюг би подовжився.

Оскільки |I1|+|I2| = m-1 ≥n, то маємо спільні вершини, що належать обом підмножинам: I1 ⋂ I2 ≠ ∅. Припустимо, це вершина i. Тоді маємо i ∈ I1 ⋂ I2 → i ∈ I1 (xi, xm) ∈ E та i ∈ I2 (x1, xi+1) ∈ E. Отримали простий цикл, показаний на рис. 56: xi+1 - xi+2 – xm - - xi - xi-1 - x1 - xi+1 в який входять m вершин графа і якщо m = n – то m – всі вершини графа.

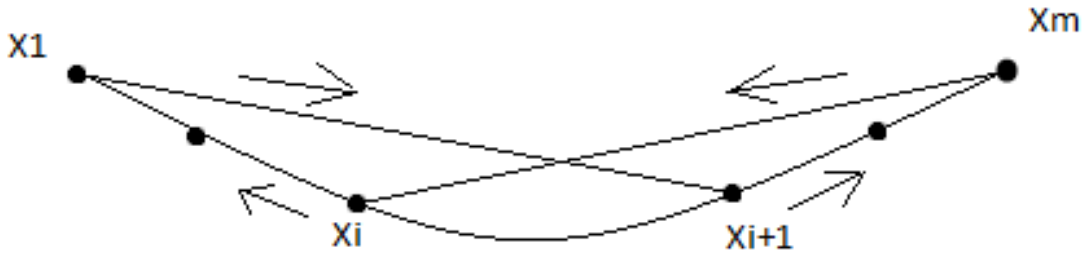


Рисунок 56 – Доведення існування Гамільтонового циклу

А сам граф має простий цикл, що проходить через всі вершини і тому є Гамільтоновим.

Теорему доведено.

Розглянемо наступну умову існування Гамільтонового графа. Умова Оре формулюється як теорема, а Гамільтонові графи, що задовольняють даній умові носять назву графи Оре.

Теорема Оре: Якщо для будь-яких не суміжних вершин u, v u,v ∈ V, графа G = <V,E> виконується deg(n) + deg(v) ≥ n, де n = |V| ≥ 3, то граф G – Гамільтонів.

На рис. 57 зображено приклад графа Оре для якого deg(a)+ deg(b) 5 ≥ n=5.

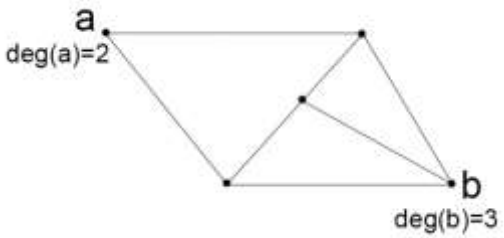


Рисунок 57 – Граф Оре

Доведення. Теорема Оре витікає з теореми Дірака. Якщо u, v де u,v ∈ V Гамільтонового графа G = <V,E>, то deg(u) ≥ n/2, deg(v) ≥ n/2. Якщо n=2, то вершини суміжні та deg(u) +deg(v) ≥ n, що очевидно. Якщо n ≥ 3, то має існувати вершина, що суміжна обом вершинам u, v, відповідно до твердження 1. В цьому разі маємо, що умова Дірака виконується, відповідно, якщо n ≥ 3. Що і потрібно було довести.

Якщо граф не задовольняє вищевказаним умовам, це ніяк не означає, що він не Гамільтоновий. В загальному випадку, задача може бути остаточно вирішена лише перебором варіантів побудови циклу.

Питання для самоконтролю

1. Чи може бути Гамільтонів граф графом Оре? Чому?
2. Чи є даний граф Гамільтоновим ? Напів-Гамільтоновим?

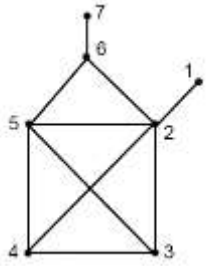


Рисунок 58 – Граф до питання 2

3. Чи може бути Еейлерів граф ациклічним?
4. Чи може бути степінь вершини непарною у Ейлерового графа?

5. Чи може степінь вершини бути непарною у напів- Ейлеровому графі?
6. У чому відмінність між Ейлеровим і Гамільтоновим графами?
7. Які умови існування Гамільтонового графа знаєте?
8. Чи обов'язково Гамільтонів граф має бути зв'язним? Чому?
9. Чи має бути граф Оре Гамільтоновим графом?
10. Чи має бути Гамільтонів граф графом Оре?

4. АЛГОРИТМИ НА ГРАФАХ

Поняття графа дозволяє структурувати інформацію та алгоритмізувати цілий ряд типових задач. Саме тому існує цілий ряд алгоритмів які працюють з інформацією, що представлена у вигляді графів. Або їх ще називають алгоритми на графах. Зокрема, кожен раз коли ви відкриваєте файл на комп'ютері, останній запускає алгоритм пошуку в пам'яті адреси файлу. При цьому пошук виконується з використанням алгоритму на графах –пошук у бінарному дереві. Розглянемо цей та ряд інших алгоритмів детальніше.

4.1. Алгоритм пошуку в бінарному дереві

Бінарне дерево – дерево в якому всі вершини помічені і кожна помічена вершина має не більше двох інцидентних ребер нижнього рівня (дітей). Вершина на найвищому рівні – корінь дерева. Інцидентні ребра розміщаються наступним чином, якщо мітка вершини більша, ніж корінь, то вершина інцидента правому ребру, якщо менша, то лівому.

Наприклад: для даного масиву V = {13, 17, 15, 9, 11, 18, 10, 12} із натуральних чисел збудуємо дерево, показане на рис. 50. Потужність множини |V| = 8. Перше число в множині – корінь і оскільки друге число множини більше, ніж корінь, то розміщуємо його на праве ребро. Для третього числа множини розміщення виконуємо наступним чином, оскільки 15> 13 (корінь), то рухаємося вправо до вершини 17. Оскільки 15< 17, то рухаємося вліво, формуючи ліве ребро.

Аналогічно, оскільки 9< 13, то рухаємося вліво по бінарному дереву, формуючи ліве ребро. І так далі. Загальний вигляд бінарного дерева показано на рис. 59.

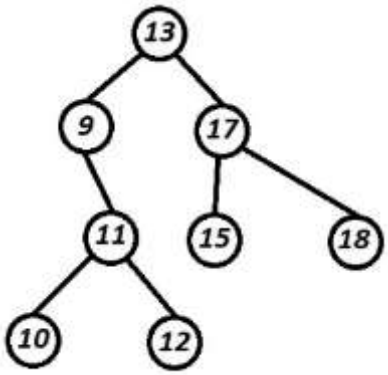


Рисунок 59 – Бінарне дерево

Завдяки побудові бінарного дерева для масиву максимальна довжина шляху пошуку в масиві зменшується. Зокрема, якщо б пошук проводився послідовним перебором по всій довжині масиву, то для бінарного дерева не більше висоти дерева. Наприклад, щоб знайти число 12 в масиві потрібно перевірити 7 попередніх чисел. Однак, якщо ми шукаємо 12 в бінарному дереві, то потрібно виконати лише три порівняння. Зх.

Якщо бінарне дерево – рівномірне, тобто кількість ребер справа не більше, ніж на одне відрізняється від кількості ребер зліва відносно кореня (рис. 60), то максимальний час пошуку пропорційний бінарній степені, тобто ln (n). Тут n – кількість елементів масиву.

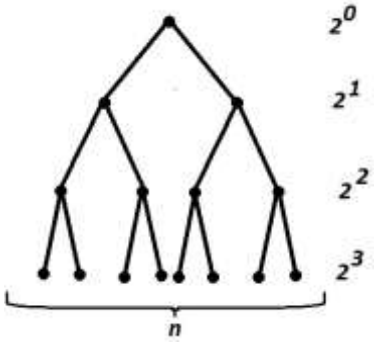


Рисунок 60 – Рівномірне бінарне дерево

4.2. Алгоритм пошуку найкоротшого шляху

Розглянемо простий неорієнтований граф, наприклад показаний на рис. 61, і розв’яжемо для нього задачу знаходження відстані між заданими вершинами.

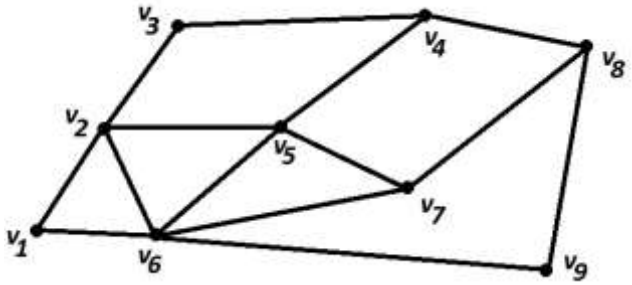


Рисунок 61 – Простий неорієнтований граф

У найпростішому випадку у графа всі ребра будуть рівні і їх довжина, наприклад 1. Знайдемо відстань від v1 до всіх інших вершин графа.

Для цього проведемо ранжування вершин графа. Ранг це відстань від v1 вершини до даної vi, заміряна в ребрах. Тому вершина v1, від якої починаємо отримує ранг 0 (ранги не перераховуються). Наступний ранг (i+1) присвоїмо вершинам, які суміжні з попередніми (i-ми) вершинами, але ще не мають рангу. Обрахунки рангів занесено в Табл.2, а вказані в ній ранги вершин є найкоротшими маршрутами від вершини v1 до v_i.

Табл. 2. Ранги вершин

Ранг	Вершини
0	v_1
1	v_2, v_6
2	v_3, v_5, v_7, v_9
3	v_4, v_8

4.3. Зважений граф, алгоритм Дейкстри

Граф, ребрам якого приписано, тобто - поставлено у відповідність деякі дійсні числа (ваги, або мітки) називаються зважений. Зважений граф, зображений на рис. 62. Як видно з рисунку, поруч з кожним ребром проставлена цифра - його вага, яка у даному разі визначає довжину ребра, виміряну в деяких умовних одиницях.

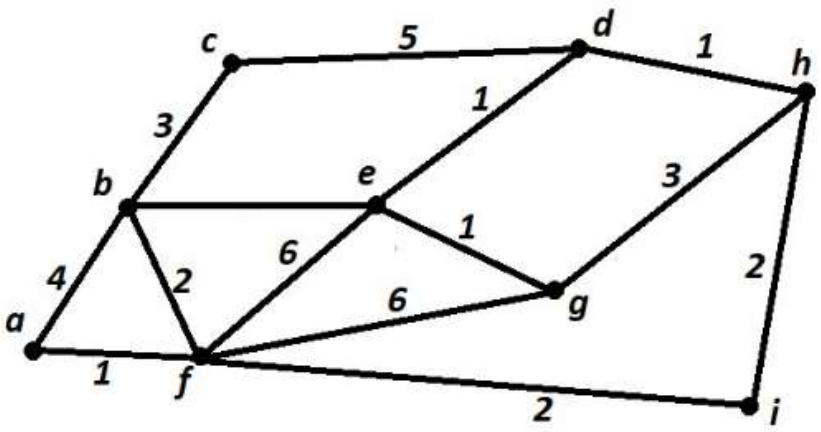


Рисунок 62 – Зважений граф

Якщо зважений граф має ще і дуги (орієнтовані ребра), він називається мережа. Наприклад, потокова мережа.

У цьому разі вищеописаний алгоритм не працюватиме, остільки кожне ребро має свою вагу, яку можна розглядати як довжину ребра, наприклад. Для знаходження найкоротшого маршруту у зваженому графі розглянемо алгоритм відомого нідерландського математика і інформатика Едсгера Дейкстри.

Суть алгоритму Дейкстри полягає в наступному – рухатися завжди через найкоротше ребро, з найменшою вагою. Опишемо сам алгоритм Дейкстри покроково:

1) помічаємо $d(v_1) = 0$; $d(v_i) = \infty$; $y = v_1$, де d – відстань, v1 – початкова вершина.
2) Для сусідніх вершин на кожному кроці знаходимо найкоротший шлях.
3) Для наступного кроку обираємо вершину з ребром, що має найменшу вагу. Якщо потрібно, то перераховуємо відстані. Перерахунок відстані для вершин v_i, суміжних з y, виконується наступним чином $d(v_i) = \min \{ d(v_i), d(y) + l(y, v_i)\}$. Тут l – вага ребра, що інцидентне обом вказаним вершинам і повторюємо, починаючи з п.2.

Як приклад, визначимо найкоротші шляхи на зваженому графі, зображеному на рис. 53, і зведемо їх у таблицю 3.

Таблиця 3. Розрахунок відстаней на зваженому графі за алгоритмом Дейкстри

N	Поточна вершина y	a	b	c	d	e	f	g	h	i
1	a	0a	4a	∞	∞	∞	1a	∞	∞	∞
2	f	0a	3f	∞	∞	7f	1a	7f	∞	3f
3	b	0a	3f	6b	∞	7f	1a	7f	∞	3f
4	i	0a	3f	6b	∞	7f	1a	7f	5i	3f
5	h	0a	3f	6b	6h	7f	1a	7f	5i	3f
6	c	0a	3f	6b	6h	7f	1a	7f	5i	3f

В оголовку таблиці перераховано всі вершини зваженого графа та вказано поточну вершину y. Припустимо, визначатимемо мінімальні відстані від вершини a, тобто v1=a. Розглянемо як працює алгоритм Дейкстри, результати якого наведено вище в Табл. 3. В рядку 1 вказано за поточну – першу вершину a і помічено, що відстань від a до всіх інших невідома і нескінченна. Крім вершин b і f відстані до яких 4a і 1a, відповідно. Для вершини f маємо:

d(f) = min { d(f); d(a) + l (a, f) }

1 =min{ 1, 0 + 1}

Нижній індекс вказує з якої вершини ми потрапили в дану. Значення 0a вказує на те, що відстань від вершини a до самої себе складає нуль, а підкреслення цього значення вказує, що воно є фіксованим, оскільки ця вершина пройдена. Окрім підкресленого значення в 1 рядку найменша відстань буде до вершини f (1a) і тому беремо її за поточну вершину y та заносимо в 2 рядок.

Другий рядок містить відстані від вершини f до вершин, які є суміжними з f, крім вершини a, яка вже пройдена. Як видно з таблиці 3, у 2 рядку для вершини b відстань була замінена на 3f. Тобто, довжина відстані від вершини a до вершини b складає 3 і проходить через вершину f. Оскільки відстань через вершину менша, ніж через вершину a 3f <4a, то в табл. 3 вказана менша відстань. Для вершини b маємо:

d(b) = min { d(b); d(f) + l (f, b) }

3 = min { 4, 1 + 2}

Підкреслення значення 1a вказує, що вершину f пройдено і відстань до неї через вершину a вже зафіксовано і в подальшому не буде розглядатися, або

змінюватися. Далі в 2 рядку встановлюємо дві вершини, що мають мінімальні відстані – b та i. За поточну беремо вершину, яка розміщена лівіше а таблиці – b. Тому 3 рядок записано для поточної вершини b.

Аналогічно заповнюються всі рядки табл.3. Зупинка роботи алгоритму відбувається, якщо подальші кроки не змінюють рядок результатів. Зокрема, в табл. 3 рядок 5 є тотожним рядку 6, що вказує на завершення роботи алгоритму.

На основі розрахунку відстаней на зваженому графі за алгоритмом Дейкстри, що наведений в Табл. 3 можна зробити наступні висновки:

I) отримано найкоротші відстані на графі. Зокрема, від вершини а до b = 3; від вершини а до g = 7; від вершини а до d = 6 і так далі.

II)Через зворотній хід отримано найкоротші маршрути між вершинами зваженого графа. Наприклад, якщо хочемо потрапити з вершини і до початкової вершини а, дивимося, що до вершини і йдемо через f(3f), а до вершини f йдемо через вершину а(1а). Тобто, маємо маршрут: a – f – i

Або до вершини d через 6h, до вершини h через 5i, до вершини і через 3f, до вершини f через 1а. Тобто, маємо найкоротший маршрут для зваженого графа a – f – i – h – d.

4.4. Алгоритм пошуку у ширину

Наступним розглянемо популярний алгоритм пошуку на графі в ширину, або ще називають – обхід графа в ширину. Задачею алгоритму є пошук певної інформації у структурованих даних, поданих у вигляді графа.

Найпростішим прикладом роботи алгоритму пошуку у ширину є колекціонування (марок, значків і т.д.). Алгоритми дослідження графа шар за шаром, починаючи з вершини S, показано на рис. 63. Припустимо, що ми, знаходячись на нульовому рівні у дереві, шукаємо марку зі слоном, якої в нас немає. В першу чергу шукаємо цю марку серед близьких знайомих і друзів, які знаходяться найближче до нас (рівень 1), далі – серед знайомих цих близьких знайомих (рівень 2), далі серед інших знайомих і врешті-решт на деякому n рівні

ми знаходимо шукану марку. Або припиняємо пошук, обійшовши всі n рівнів знайомих.

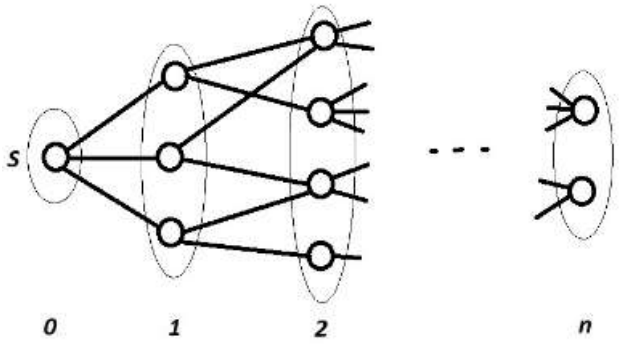


Рисунок 63 – Обхід графа в ширину

За аналогічним алгоритмом відбувається збирання кубика Рубика, збирання пазлів та інші логічні ігри. Алгоритм пошуку в ширину (англ. breadth-first research (BFS) використовують також для:

- Web серфінгу,
- пошуку друзів у соціальних мережах,
- доступу до мережі Internet,
- обрахунку кон’юнкції,
- перевірки моделей в теорії автоматів та ін.

В цілому, алгоритм пошуку у ширину розглядає простий зв’язний граф G = <V, E>. Задачею алгоритму є побудова маршруту від початкової і до всіх вершин графа, обхід графа.

Якщо розглянути граф на рис. 64, то початковою обрана вершина 2, що має нульовий рівень. На 1 рівні розміщено вершини 1, 3, 4, 5, які інцидентні вершині 2 і лежать на відстань ребра від неї. В цілому, побудова деякого і рівня опирається на (i - 1) рівень інцидентних вершин, які лежать на відстані ребра. Однак, варто звернути увагу, що вершини попереднього (i - 1) рівня ігноруються.

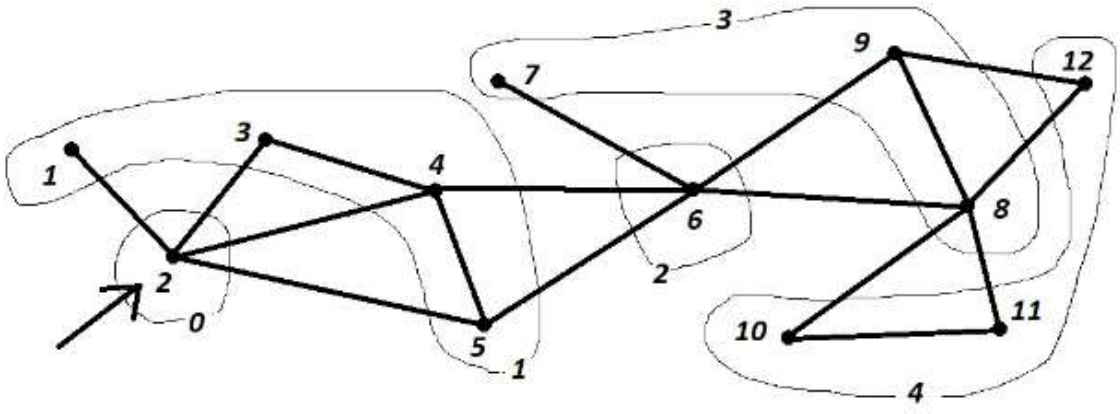


Рисунок 64 – Рівні обходу графа в ширину

Приклад 1. За алгоритмом пошуку в ширину обійдемо граф, зображений на рис. 55. Результати обходу графа подано в табл. 4. Оголовок таблиці описує відкриті ребра, які ми маємо обійти, закрите ребро – ребро, яке пройшли, щоб потрапити в дану вершину і-го рівня та реєстр, де перераховано (наприклад, за годинниковою стрілкою) всі ребра для вершин на (і+1) рівні, які інцидентні даній вершини і-го рівня. В подальшому ребра із реєстру додаються у кінець списку відкритих ребер, а перше з відкритих ребер – закривається.

Як видно, в 1 рядку Табл.4 записано уявно ребро (p,2), що інцидентне початковій вершині 2. У другому рядку перенесено це ребро в закриті і для вершини 2 заносимо в реєстр інцидентні їй чотири ребра. Далі ці ребра перенесемо у відкриті ребра. У третьому рядку в закрите ребро перенесемо перше ребро (2,1) із відкритих ребер і, стоячи в вершині (1), бачимо що список наступних вершин відсутній – реєстр буде порожнім. Аналогічно і для 4 рядка табл. 4. Далі, в рядку 5 перенесено ребро (2,4) в закриті ребра, а 4 - в вершини і в реєстр заносимо ребро (4,6). При цьому ребро (4,5) не заноситься в реєстр, оскільки вершина 5 вже потрапила з ребром (2,5) у список відкритих ребер. Аналогічно обходяться інші вершини. Робота алгоритму закінчується коли у стовпчику вершин будуть всі вершини графа.

Табл. 4. Результат обходу графа в ширину (BFS)

N п/п	Відкриті ребра	Закрите ребро	Вершина	Реєстр
1	(p, 2)			
2	(2,1), (2,3), (2,4), (2,5)	(p, 2)	2	(2,1), (2,3), (2,4), (2,5)
3	(2,3), (2,4), (2,5)	(2,1)	1	
4	(2,4), (2,5)	(2,3)	3	
5	(2,5), (4,6)	(2,4)	4	(4,6)
6	(4,6)	(2,5)	5	
7	(6,7), (6,8), (6,9)	(4,6)	6	(6,7), (6,8), (6,9)
8	(6,8), (6,9)	(6,7),	7	
9	(6,9), (8,12), (8,10), (8,11)	(6,8)	8	(8,12), (8,10), (8,11)
10	(8,12), (8,10), (8,11)	(6,9)	9	
11	(8,10), (8,11)	(8, 12)	12	
12	(8; 11)	(8, 10)	10	
13		(8, 11)	11	

4.5. Алгоритм пошуку в глибину

Наступним розглянемо популярний алгоритм пошуку на графі в глибину, або ще називають – обхід графа в глибину (англ. Depth-First Search (DFS). Задачею алгоритму є пошук певної інформації у структурованих даних, поданих у вигляді графа.

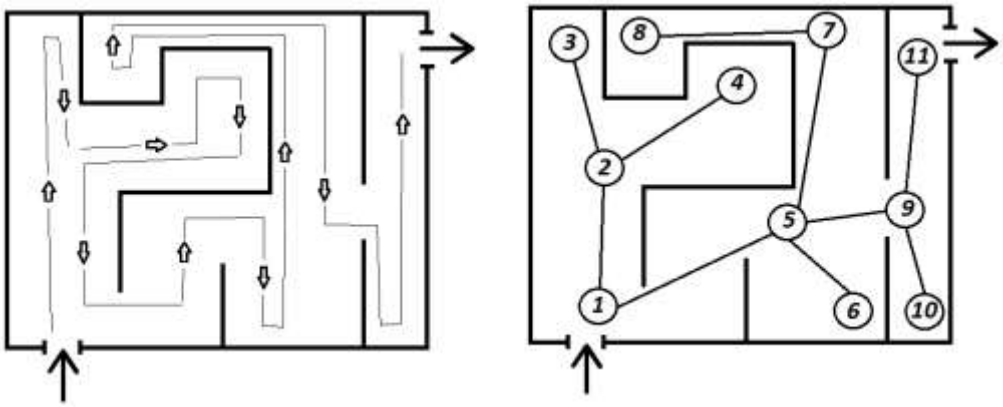
В цілому, алгоритм пошуку у ширину розглядає простий зв’язний граф G = <V, E>. Задачею алгоритму є побудова маршруту під початкової і до всіх вершин графа, обхід графа.

Найпростішим прикладом роботи алгоритму є пошук виходу з лабіринту – якщо ми можемо рухатися вперед, ми рухаємося вперед, шукаючи вихід. В іншому разі повертаємося на перехрестя і рухаємося за іншим маршрутом. Однак, на відміну від попереднього алгоритму, пошук йде не за рівнями, а за маршрутом.

Приклад 1. Розглянемо лабіринт та його подання у вигляді дерева як показано на рис. 56. Для знаходження виходу з лабіринту в найгіршому випадку доведеться пройти за всіма шляхами, які вказано стрілками в лабіринті (рис. 65 а). Для алгоритмізації цієї задачі, лабіринт зобразимо у вигляді графа, в даному випадку -

дерева (рис. 65 б). При цьому вершини дерева ставляться у точках зміни шляху, а ребрами є шлях в лабіринті – даний підхід використовується для будь-якого лабіринту. Алгоритми пошуку в глибину працює наступним чином: починаючи з входу, рухаємося – наприклад – завжди вліво. Тоді маршрут пошуку виходу в лабіринті на рис. 65 б) має наступний вигляд: 1-2-3-2-4-2-1-5-7-8-7-5-9-11. До речі, для графа лабіринту на рис. 65 б) у випадку вибору руху завжди вправо, маршрут виходу буде меншим і має наступний вигляд: 1-5-6-9-10-9-11.

В цілому, алгоритм пошуку у глибину розглядає простий зв'язний граф G = <V, E>. Задачею алгоритму є побудова маршруту від початкової і до всіх вершин графа, обхід графа. У вищенаведеному прикладі 1 маршрут виходу з лабіринту є частиною маршруту обходу графа в глибину, який включає всі вершини графа.



а) б)

Рисунок 65 – Лабіринт а) та граф шляхів у ньому б)

Приклад 2. Даний алгоритм обходу графа в глибину застосуємо до графа на рис. 66, який інцидентний графу на рис. 64, однак алгоритм обходу графа в глибину не потребує визначення рівнів вершин.

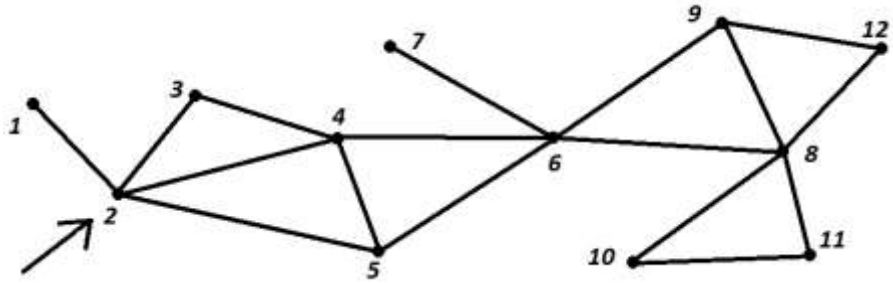


Рисунок 66 – До прикладу 2 алгоритму обходу графа в глибину

Головна відмінність даного алгоритму від алгоритму пошуку в ширину легко зрозуміла з Табл. 5, і полягає в тому, що ребра з реєстру записуються перед іншими ребрами у відкриті ребра. Цей принцип носить назву останнім прийшов – першим пішов. Наприклад це схоже на стос паперів на столі - той листок, який ви покладете останнім у цей стос за потреби буде забрано першим. В інформатиці це аналог інформації у стеку, яка обробляється за принципом LIFO (англ. Last In, First Out). Саме тому, останній стовпчик таблиці – стек.

В цілому Оголовок Табл.5 описує відкриті ребра, які ми маємо обійти, закрите ребро – ребро, яке пройшли, щоб потрапити в дану вершину, стек, де перераховано (наприклад, за годинниковою стрілкою) всі ребра, що інцидентні даній вершині. В подальшому ребра зі стеку додаються у початок відкритих ребер (LIFO), а перше з відкритих ребер – закривається.

В 1 рядку Табл. 5 записано уявно ребро (p,2), що інцидентне початковій вершині 2. У другому рядку це ребро перенесено в закриті і для вершини 2 заносимо в реєстр інцидентні їй 4 ребра. Далі ці ребра перенесемо у відкриті ребра. У третьому рядку в закрите ребро перенесемо перше ребро (2,1) із відкритих ребер і, стоячи в вершині (1), бачимо що список наступних вершин відсутній – стек залишаємо не заповненим. Рядок 4 табл. 5 свідчить, що ребро (2,3) – пройдене, оскільки потрапляє в список закритих ребер і ми знаходимося в вершині 3. Вершині 3 інцидентні два ребра (2,3), що вже пройдене і (3,4), яке і заноситься в стек. Крім стеку, ребро (3,4) додаються у початок відкритих ребер (LIFO). Далі, в рядку 5 ребро (3,4) перенесено в закриті ребра, а 4 - в вершини і в стек заносимо ребра (4,5) і (4,6). При цьому ребро (2,4) не заноситься в стек, оскільки вершина 2 вже пройдена. Далі ребра (4,5) і (4,6) додаються у початок відкритих ребер. Аналогічно обходяться інші вершини. Робота алгоритму закінчується коли у стовпчику вершин будуть всі вершини графа.

Табл. 5. Результат обходу графа в глибину (DFS)

N п/п	Відкриті ребра	Закрите ребро	Вершина	Стек
1	(p, 2)			
2	(2,1), (2,3), (2,4), (2,5)	(p, 2)	2	(2,1), (2,3), (2,4), (2,5)
3	(2,3), (2,4), (2,5)	(2,1)	1	
4	(3,4), (2,4), (2,5)	(2,3)	3	(3,4)
5	(4,5), (4,6), (2,4), (2,5)	(3,4)	4	(4,5), (4,6)
6	(5,6) (4,6), (2,4), (2,5)	(4,5)	5	(5,6)
7	(6,7), (6,9), (6,8), (4,6), (2,4), (2,5)	(5,6)	6	(6,7), (6,9), (6,8)
8	(6,9), (6,8), (4,6), (2,4), (2,5)	(6,7),	7	
9	(9,8), (9, 12), (6,8), (4,6), (2,4), (2,5)	(6,9)	9	(9,8), (9, 12)
10	(8,12), (8,11), (8,10), (9, 12), (6,8), (4,6), (2,4), (2,5)	(9,8)	8	(8,12), (8,11), (8,10)
11	(8,11), (8,10), (9, 12), (6,8), (4,6), (2,4), (2,5)	(8, 12)	12	
12	(8,10), (9, 12), (6,8), (4,6), (2,4), (2,5)	(8, 11)	11	(11, 10)
13	(8,10), (9, 12), (6,8), (4,6), (2,4), (2,5)	(11, 10)	10	

Питання для самоконтролю

1. Який принцип побудови бінарного дерева ?
2. Який максимальний час пошуку на бінарному дереві? Чому?
3. Що таке рівномірне бінарне дерево?
4. Що таке ранжування вершин графа?
5. Яка мета алгоритму Дейкстри?
6. Які приклади зважених графів знаєте?
7. Яка послідовність кроків в алгоритму пошуку в ширину? Наведіть приклади використання алгоритму пошуку.
8. Яка послідовність кроків в алгоритму пошуку в глибину? Наведіть приклади використання алгоритму пошуку в глибину.

5. ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Відповідно до варіанту:

1) записати матрицю суміжності графа,

2) записати матрицю інцидентності,

3) представити граф явним способом,

4) представити граф списками суміжності,

5) визначити мости і точки з’єднання,

6) визначити ексцентриситети всіх вершин графа,

7) визначити діаметр, радіус і центр графа,

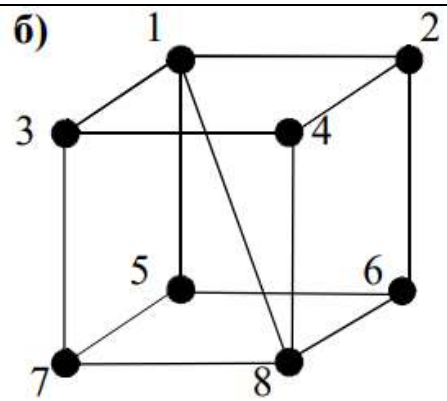
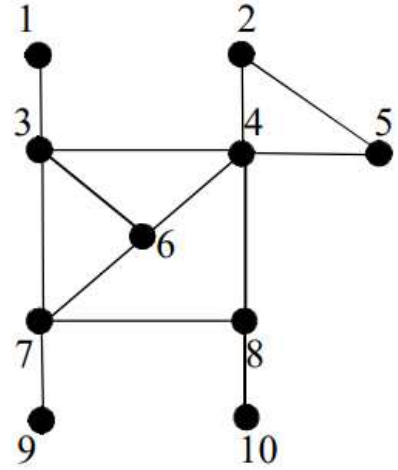
8) перевірити граф на існування Ейлерового циклу і, якщо є, записати

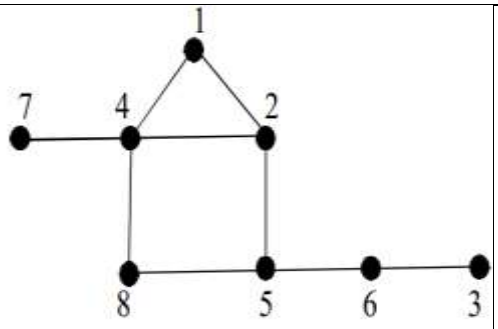
9) обійти граф за алгоритмом пошуку в ширину,

10) обійти граф за алгоритмом пошуку в глибину,

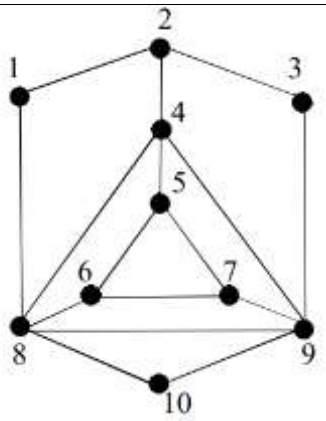
11) записати код Прюфера для дерева на основі заданого графа.

5.1. Варіанти завдань

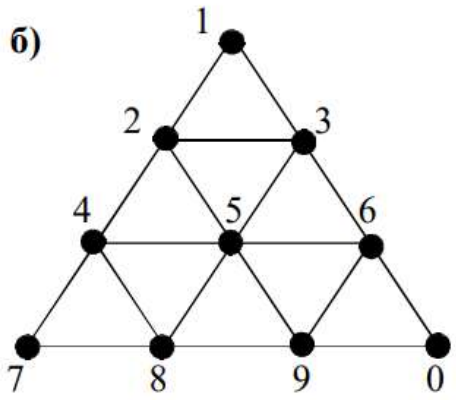
	
Варіант 1	Варіант 2



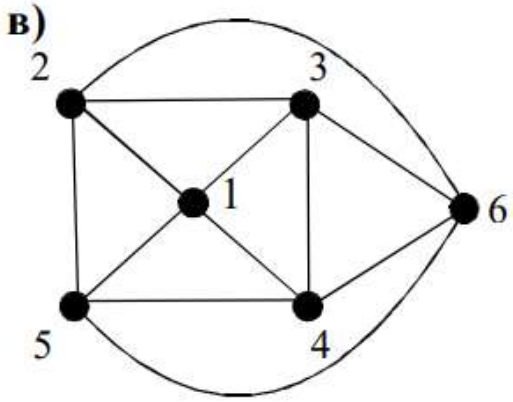
Варіант 3



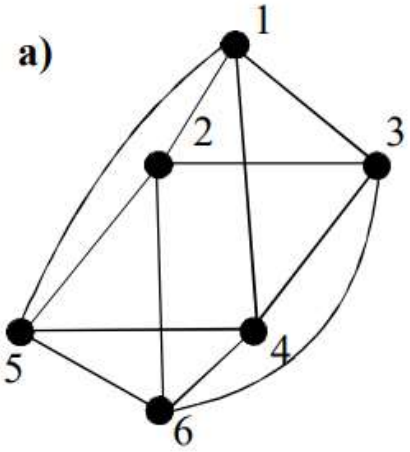
Варіант 4



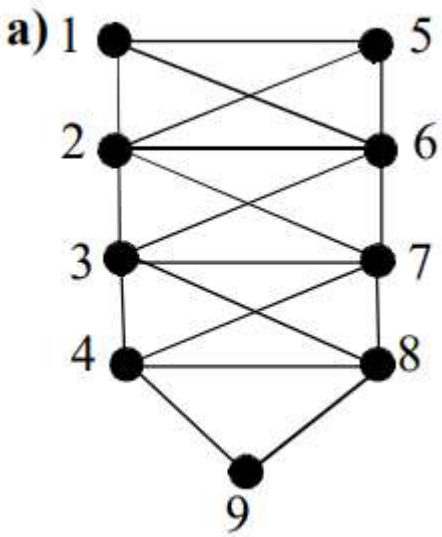
Варіант 5



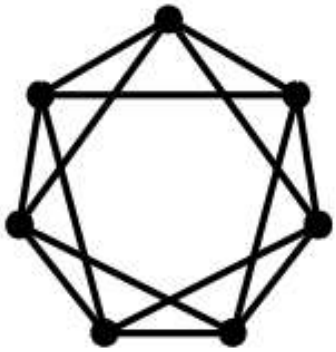
Варіант 6



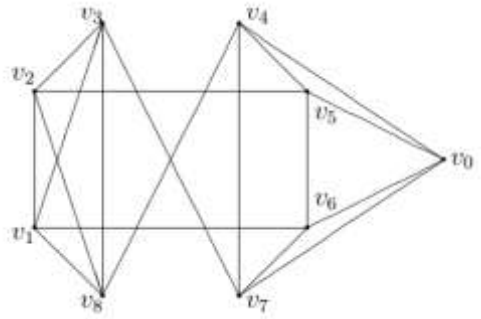
Варіант 7



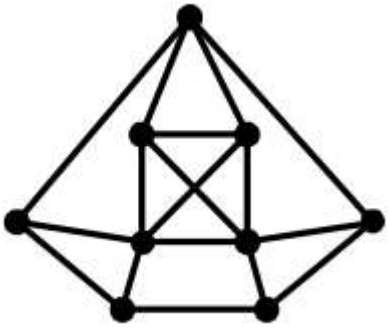
Варіант 8



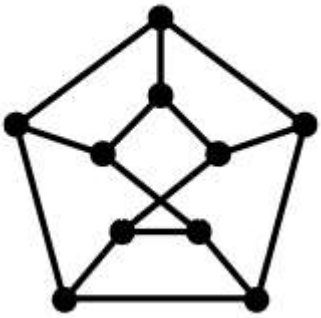
Варіант 9



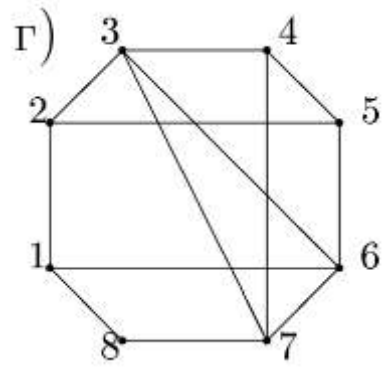
Варіант 10



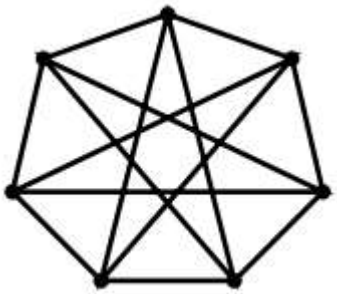
Варіант 11



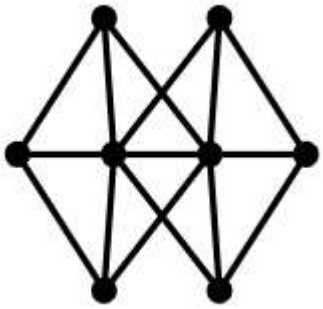
Варіант 12



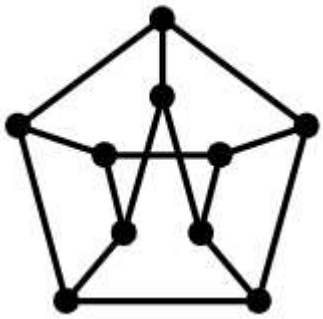
Варіант 13



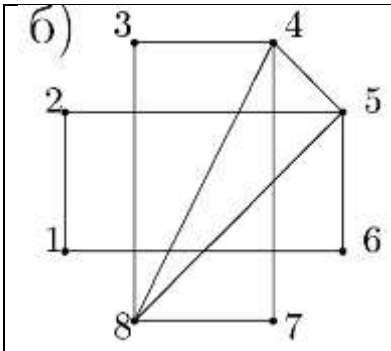
Варіант 14



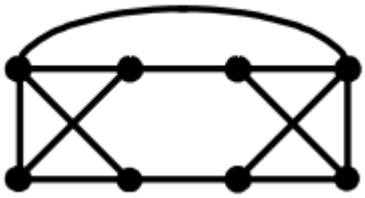
Варіант 15



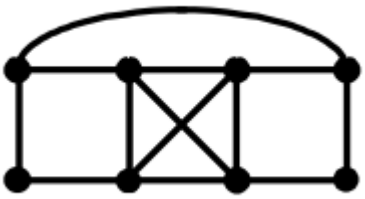
Варіант 16



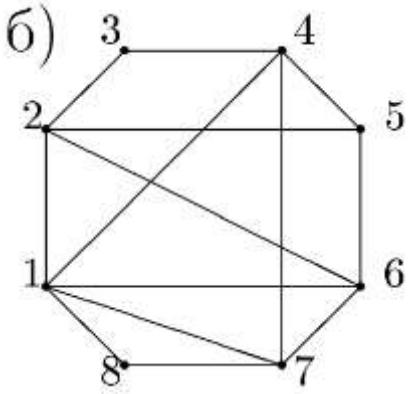
Варіант 17



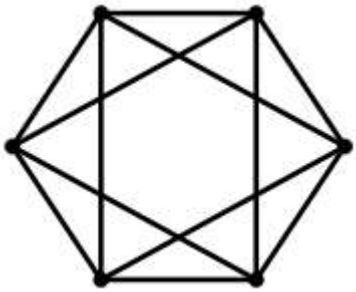
Варіант 18



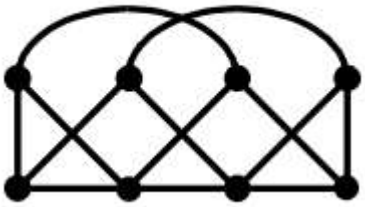
Варіант 19



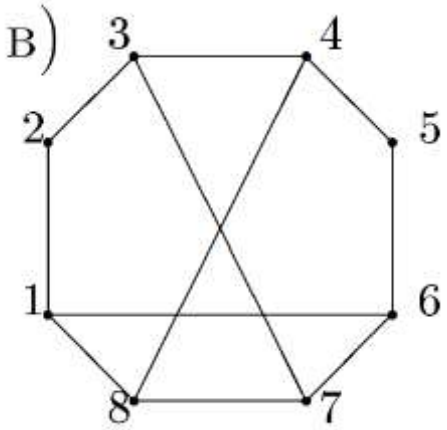
Варіант 20



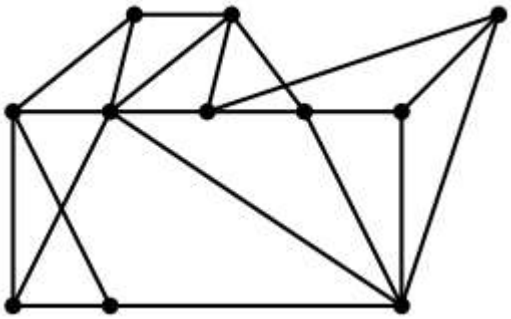
Варіант 21



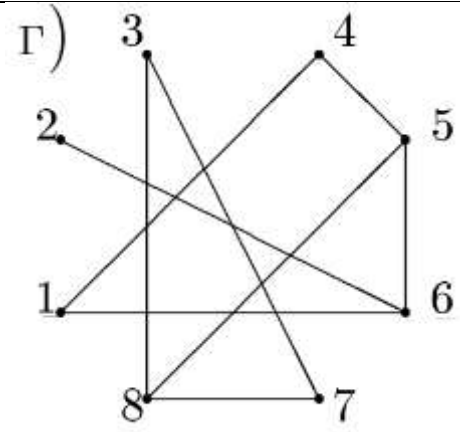
Варіант 22



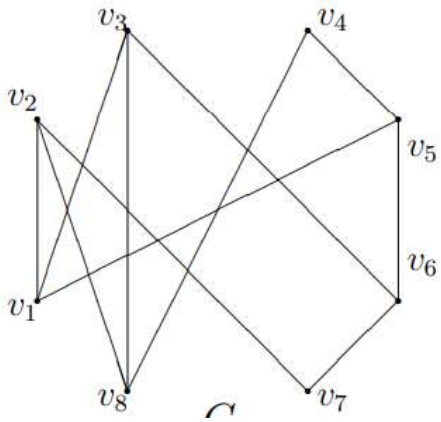
Варіант 23



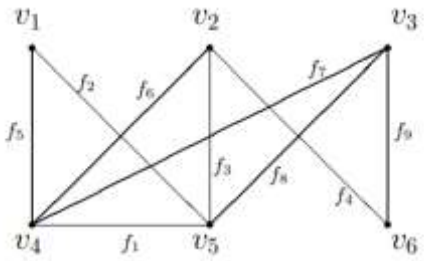
Варіант 24



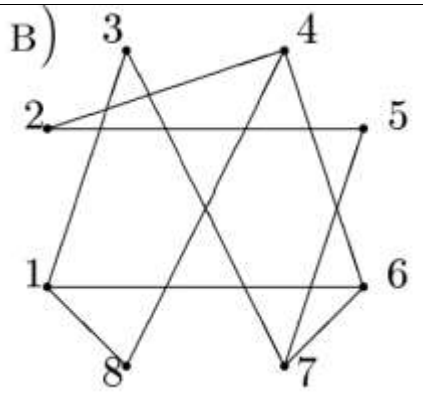
Варіант 25



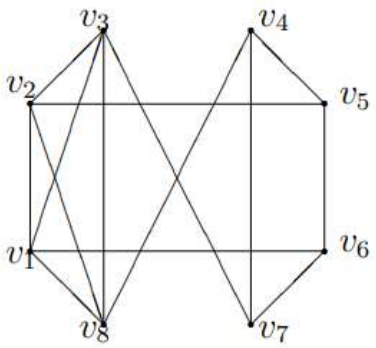
Варіант 26



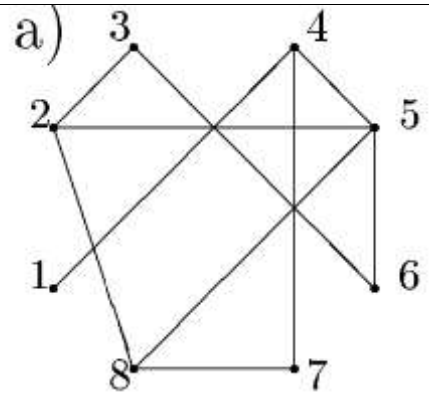
Варіант 27



Варіант 28

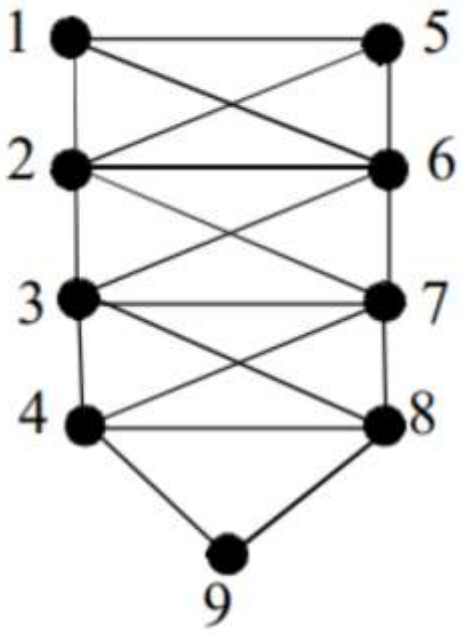


Варіант 29



Варіант 30

5.2. Приклад розв’язання завдання



1. Матриця суміжності графа

№	1	2	3	4	5	6	7	8	9
1	0	1	0	0	1	1	0	0	0
2	1	0	1	0	1	1	1	0	0
3	0	1	0	1	0	1	1	1	0
4	0	0	1	0	0	0	1	1	1
5	1	1	0	0	0	1	0	0	0
6	1	1	1	0	1	0	1	0	0
7	0	1	1	1	0	1	0	1	0
8	0	0	1	1	0	0	1	0	1
9	0	0	0	1	0	0	0	1	0

2. Матриця інцидентності графа

№	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0	1	0	1	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	0
5	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

3. Представити граф явним способом

G = (V, E)

V = {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r}

E = {1, 2, 3, 4, 5, 6, 7, 8, 9}

4. Представити граф списками суміжності

1	суміжно до	2, 5, 6
2	суміжно до	1, 3, 5, 6, 7
3	суміжно до	2, 4, 6, 7, 8
4	суміжно до	3, 7, 8, 9
5	суміжно до	1, 2, 6
6	суміжно до	1, 2, 3, 5, 7
7	суміжно до	2, 3, 4, 6, 8
8	суміжно до	3, 4, 7, 9
9	суміжно до	4, 8

5. Визначити мости і точки з’єднання

Граф не має мостів і тому не має точок з’єднання.

6. Визначити ексцентриситети всіх вершин графа

1	4
2	3
3	2
4	3
5	4
6	3
7	2
8	3
9	4

7. Визначити діаметр, радіус і центр графа

d = max(e(v)) = 4

r = min(e(v)) = 2

Центральні вершини: 3, 7

8. Перевірити граф на існування Ейлерового циклу і, якщо є, записати

Граф не містить Ейлерів цикл, оскільки вершини 1, 2, 3, 5, 6, 7 мають непарний степінь.

9. Обійти граф за алгоритмом пошуку в ширину

Почнемо обхід з вершини (1)

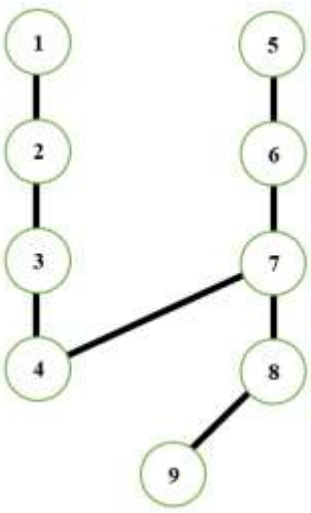
Відповідь: 1 5 2 6 7 3 8 4 9

10. Обійти граф за алгоритмом пошуку в глибину

Почнемо обхід з вершини (1)

Відповідь: 1 5 6 7 8 9 4 3 2

11. Записати код Прюфера для дерева на основі заданого графа



Код Прюфера: 2 3 4 7 6 7 8 та ребро (8, 9)

СПИСОК ЛІТЕРАТУРИ

1. Андерсон Д.А. Дискретная математика и комбинаторика. М.: Вильямс, 2003. - 960 с
2. Бондаренко М.Ф., Білоус Н.В., Руткас А.Г. Комп’ютерна дискретна математика: Підручник. - Харків, 2004. - 480 с.
3. Kenneth H. Rosen Discrete Mathematics and Its Applications 2002 by McGraw-Hill Science, 928 p.
4. О. Оре. Теория графов. М.: Наука, 1980, 336 с.
5. Кристофидес Н. Теория графов. Алгоритмический подход. М.: Мир, 1978, - 432 с.