ЗМІСТ

Вступ……4
1 Матричні характеристики графів……..4
2 Ейлерів цикл та ланцюг……...7
3 Мінімальне остовне коло……9
Алгоритм Краскала……10
Алгоритм найближчого сусіда (Прима)……..12
4 Паросполучення……15
Алгоритм Куна……..16
Варіанти завдань……...25
Завдання 1……...25
Завдання 2……...31
Завдання 3……...32
Завдання 4……...36
Завдання 5……...39
Список літератури……42

ВСТУП

Теорія графів – один з розділів сучасної математики, який має велике прикладне значення для візуального моделювання об’єктів, в яких ключову роль відіграють зв’язки між елементами об’єкта, а також для формалізації об’єктів та їх внутрішніх зв’язків.

Графи широко застосовуються у програмуванні. Наприклад, одна з основних структур, що використовується при розробленні компіляторів і взагалі для подання комп'ютерних програм, — граф потоків даних.

Для початкового оволодіння теорією графів студент повинен мати знання математики в обсязі середньої школи і деякі основні поняття з розділів вищої математики, зокрема, теорії матриць. Для більш досконалого вивчення пропонуємо [1, 3-8].

Розглянемо наступні теми з теорії графів:

1 Матричні характеристики графів

Теоретичні відомості та завдання з розгорнутими поясненнями містяться, наприклад, у конспекті лекцій [1, с. 17-22].

Приклад 1.1

Знайти матриці суміжності S та інцидентності T для неорієнтованого графа G (рисунок 1.1).

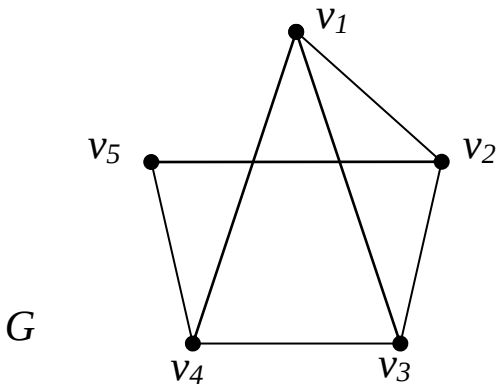


Рисунок 1.1 – Неорієнтований граф G

Матриця суміжності S має розмір 5 × 5, де 5 – кількість вершин. Елементи матриці дорівнюють кількості ребер, що поєднують відповідні вершини. Для перевірки правильності підрахуємо степені відповідних вершин (deg v_i) на рисунку 1.1.

$$S = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array} \begin{array}{l} \\ \deg v_1 = 3 \\ \deg v_2 = 3 \\ \deg v_3 = 3 \\ \deg v_4 = 3 \\ \deg v_5 = 2 \end{array}$$

У випадку орієнтованого графа враховується кількість ребер, які *виходять* з *i-ї* вершини графа в *j-ту* вершину.

Для складання матриці інцидентності T пронумеруємо ребра графа, наприклад, таким чином (рисунок 1.2).

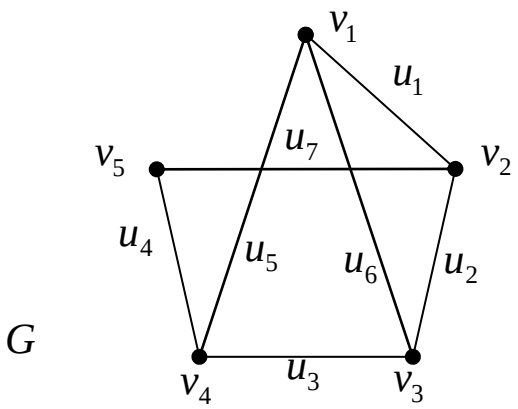


Рисунок 1.2 – Нумерація ребер неорієнтованого графа з рисунка 1.1

Матриця інцидентності T має розмір 5 × 7, де 5 – кількість вершин, 7 – кількість ребер. Одиниці стоять навпроти тих вершин, які поєднуються відповідним ребром u_i, i=1,…,7.

$$T = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Випадок орієнтованого графа відрізняється тим, що будуть стояти -1, якщо *i-а* вершина графа є кінцем *j-го* ребра та +1, якщо *i-а* вершина є початком.

Приклад 1.2

За заданою матрицею суміжності *S* побудувати граф.

$$S = \begin{pmatrix} 2 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

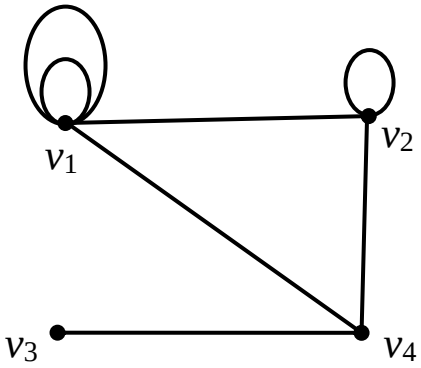
Матриця симетрична відносно головної діагоналі, граф неорієнтований.

Граф має чотири вершини – кількість рядків та стовпців матриці *S*.

$$S = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 2 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Кількість ребер графа, які поєднують дві вершини v_i, v_j, дорівнює числу, яке стоїть на перетині рядка v_i та стовпця v_j.

Наприклад, на головній діагоналі – це кількість петель у відповідних вершинах.



2 Ейлерів цикл та ланцюг

У теорії графів *ейлеровим ланцюгом* називається ланцюг у графі, що проходить кожне ребро рівно один раз.

Ейлерів цикл — це ейлерів ланцюг, що починається і завершується в одній вершині.

Ці поняття вперше було розглянуто Леонардом Ейлером під час розв'язання відомої задачі про кенігсберзькі мости у 1736 р. [1, с. 4-6].

Граф, у якому є ейлерів цикл, називається *ейлеровим*, а граф, що містить ейлерів ланцюг, *напівейлеревим*.

Приклад 2.1

Граф G_1 (рисунок 2.1) є ейлеровим, тому що в ньому є цикл, який містить усі ребра, наприклад, $v_1, v_2, v_3, v_4, v_1, v_3, v_5, v_2, v_4, v_6, v_1$. У графі G_2 такий цикл відсутній, але є ейлерів ланцюг $v_1, v_2, v_3, v_4, v_1, v_3, v_5, v_2, v_4$, тому граф G_2 напівейлерів. Граф G_3 не є ні ейлеровим, ні напівейлеровим, у ньому немає ані відповідного циклу, ані ланцюга.

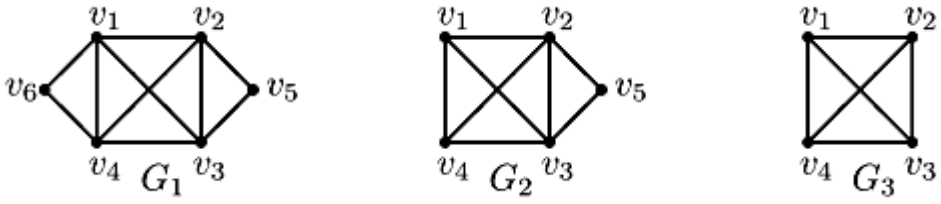


Рисунок 2.1 – Ейлерів граф та ланцюг

Доведення цього ґрунтується на такій теоремі:

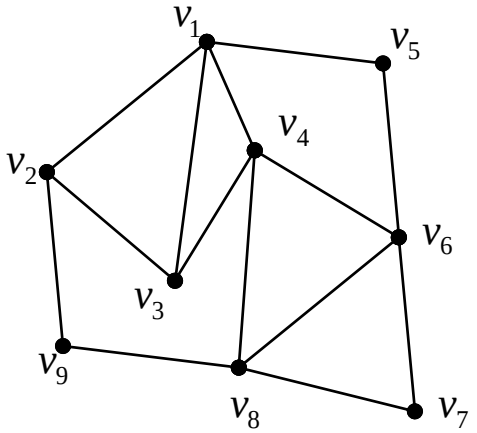
Теорема 1. *Про існування ейлерова циклу та ейлерова ланцюга для неорієнтованого графа.*

Зв'язний неорієнтований граф з *n* вершинами містить *ейлерів цикл* тоді і тільки тоді, коли степені будь-якої його вершини *парні*.

Зв'язний неорієнтований граф з *n* вершинами містить *ейлерів ланцюг*, якщо *не більше ніж дві вершини у графі мають непарний степінь*.

Приклад 2.2

Перевірити, чи містить заданий граф ейлерів цикл або ейлерів ланцюг.



Знайдемо степені вершин графа:

deg v_1=4, deg v_2=3, deg v_3=3, deg v_4=4, deg v_5=2, deg v_6=4, deg v_7=2, deg v_8=4, deg v_9=2.

Дві вершини непарного степеня deg v_2=3, deg v_3=3, граф містить ейлерів ланцюг. Він починається та закінчується в вершинах v_2 і v_3, наприклад, $v_2, v_1, v_5, v_6, v_7, v_8, v_9, v_2, v_3, v_1, v_4, v_6, v_8, v_4, v_3$.

Теорема 2. *Про існування ейлерова циклу та ейлерова ланцюга для орієнтованого графа.*

Зв'язний *орієнтований* граф містить *ейлерів цикл* тоді і тільки тоді, коли відповідний неорієнтований граф зв'язний, і всі вершини мають рівні вхідні і вихідні степені, тобто

$$\deg {}^{+}v = \deg {}^{-}v \text{ для всіх вершин.}$$

Зв'язний *орієнтований* граф містить *ейлерів ланцюг* тоді і тільки тоді, коли відповідний неорієнтований граф зв'язний, не більш ніж одна вершина має вихідний степінь – вхідний степінь 1, не більше ніж одна вершина має вхідний степінь – вихідний степінь 1, а всі інші вершини мають рівні вхідні і вихідні степені, тобто

$$\deg^+ v = \deg^- v, \text{ крім двох вершин}$$

u – початок, w – кінець ейлерова ланцюга;

$$\deg^+ u = \deg^- u + 1, \quad \deg^- w = \deg^+ w + 1.$$

3 Мінімальне остовне дерево

Основні визначення і теореми з цієї теми містяться у [1, с. 6-17].

Відмінною особливістю *дерева* є те, що між будь-якими двома його вершинами існує єдиний шлях. Дерево не містить циклів і петель.

Остов – це підграф даного графа, що складається з усіх вершин і ребер, які не утворюють цикл. Тому кількість ребер має бути n-k, де n – порядок графа, тобто кількість його вершин, а k – кількість компонент зв'язності.

Мінімальне остовне дерево у зв'язному зваженому неорієнтованому графі – це остов цього графа, що має мінімальну можливу вагу, де під вагою дерева розуміється сума ваг ребер, які входять до нього.

Завдання про знаходження мінімального остовного дерева можна задати у такій постановці: припустимо, є n міст, які необхідно з'єднати залізничними коліями так, щоб можна було дістатися з будь-якого міста в будь-яке інше (безпосередньо або через інші міста). Дозволяється будувати колії між заданими парами міст і відома вартість будівництва кожної такої колії. Необхідно вирішити, які саме колії потрібно будувати, щоб мінімізувати загальну вартість будівництва.

Знаходження мінімального остовного дерева за алгоритмом *Краскала* (або Крускала) і *найближчого сусіда (Прима)* розглянемо на прикладі.

Приклад 3.1

Заданий зважений граф (рисунок 3.1). Знайти остов графа мінімальної ваги за алгоритмами Краскала і найближчого сусіда.

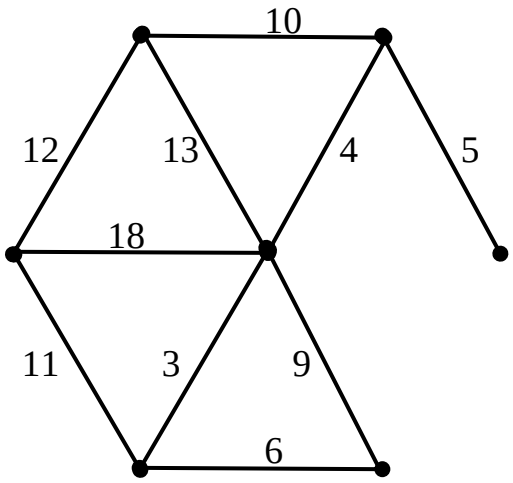


Рисунок 3.1 – Зважений граф

Остов графа складається з усіх вершин n=7 та ребер, які не містять циклу. Ребер в остові має бути n-k = 7-1 = 6, k – кількість компонент зв’язності.

Алгоритм Краскала

Знаходимо ребро з найменшою вагою, це ребро з вагою 3, наводимо його (рисунок 3.2). Якщо таких ребер декілька, обираємо довільне.

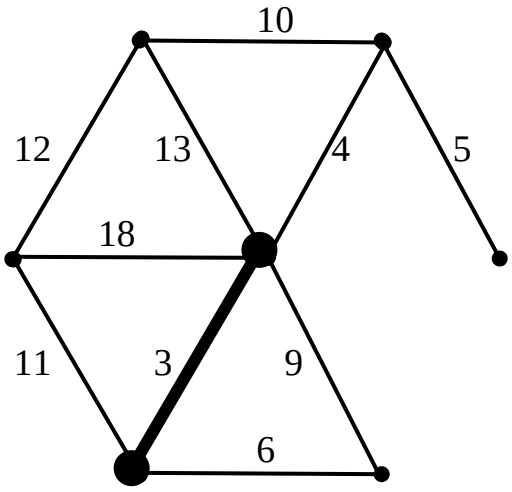


Рисунок 3.2

Наступне наводимо ребро з вагою 4 (рисунок 3.3). Два ребра не утворюють цикл, переходимо до наступного.

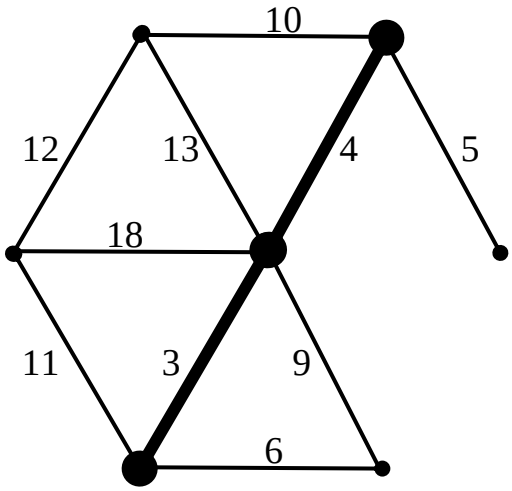


Рисунок 3.3

Далі знаходимо по черзі ребра з найменшою вагою, які не утворюють цикл (рисунок 3.4, 3.5).

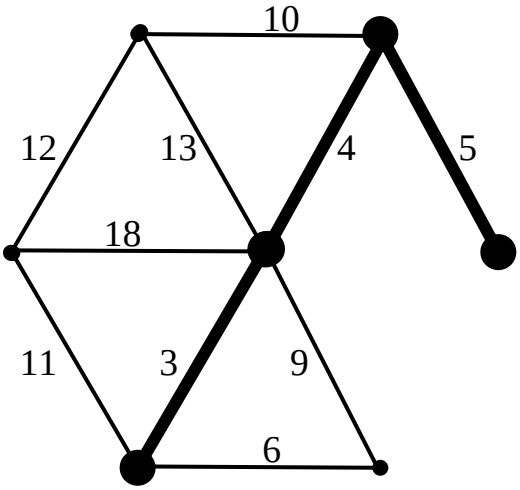


Рисунок 3.4

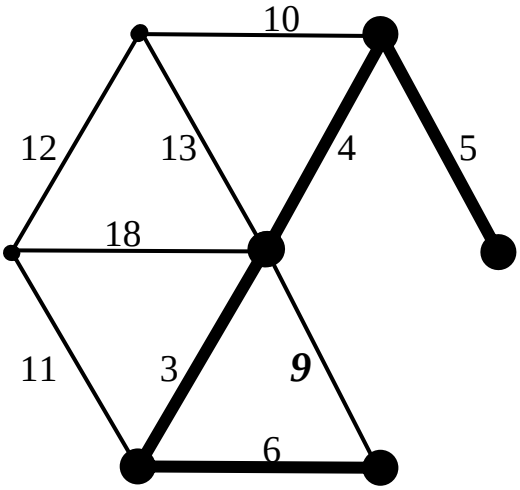


Рисунок 3.5

Ребро з вагою ***9*** не можна обрати, утвориться цикл (див. рисунок 3.5), наступне наводимо з вагою 10 (рисунок 3.6) і 11 (рисунок 3.7).

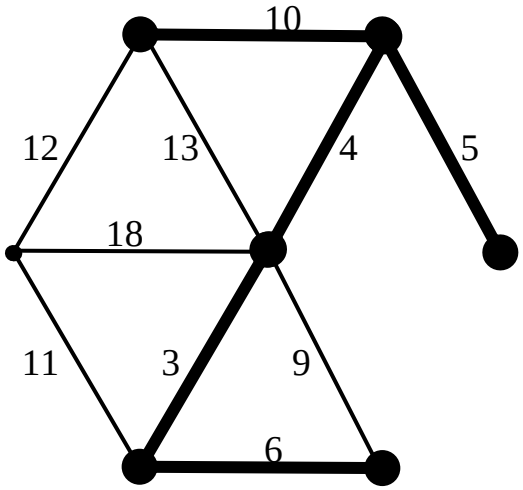


Рисунок 3.6

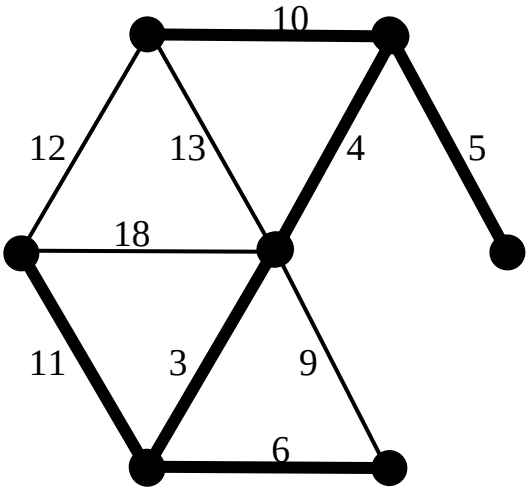


Рисунок 3.7

Ізольованих вершин більше немає (рисунок 3.7), всього 6 ребер, це – мінімальне остовне дерево (рисунок 3.8).

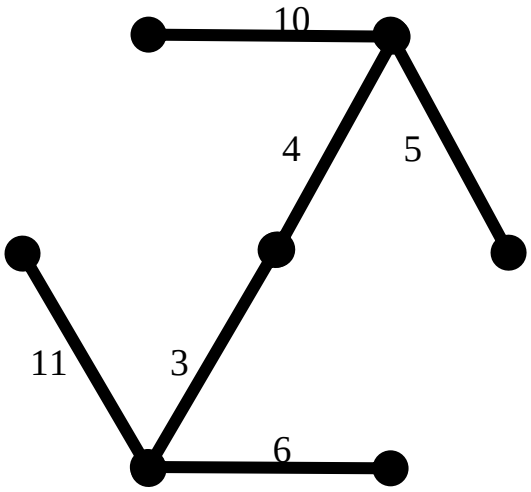


Рисунок 3.8 – Мінімальне остовне дерево, знайдене за алгоритмом Краскала

Мінімальна вага остовного дерева дорівнює сумі ваг жирних ребер 3+4+5+6+10+11=39.

Алгоритм найближчого сусіда (Прима)

Знаходимо будь-яку вершину графа з рисунка 3.1. Розглядаються ребра графа, один кінець яких – вершина, яка вже належить дереву, а інша – ні; з цих ребер вибирається ребро

найменшої ваги. Наприклад, оберемо верхню ліву вершину. Найменша вага ребра, що виходить з цієї вершини, дорівнює 10, наводимо це ребро (рисунок 3.9). До дерева тепер належить друга вершина – кінець обраного ребра.

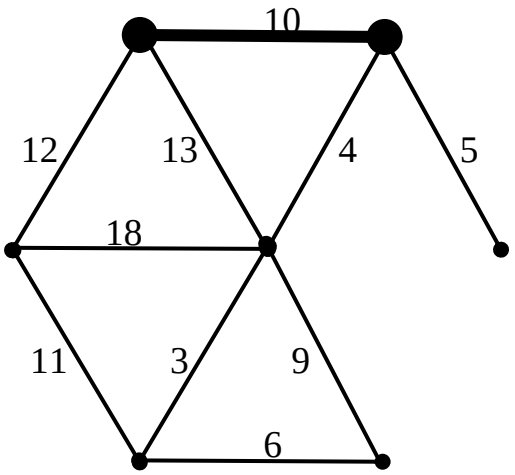


Рисунок 3.9

Тепер знаходимо ребро, яке інцидентне (входить або виходить) будь-якій з цих двох вершин та має найменшу вагу. Це ребро з вагою 4, наводимо його (рисунок 3.10). На кожному кроці ребро, яке обирається, приєднується до дерева. Зростання дерева відбувається, доки не будуть вичерпані всі вершини вихідного графа.

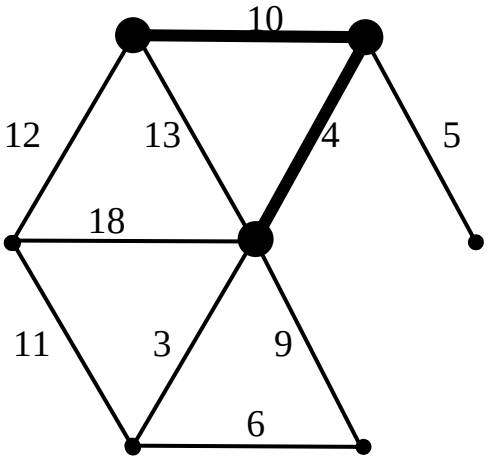


Рисунок 3.10

Наступне ребро з вагою 5, інцидентне одній з обраних вершин, має найменшу вагу, не утворюється цикл (рисунок 3.11):

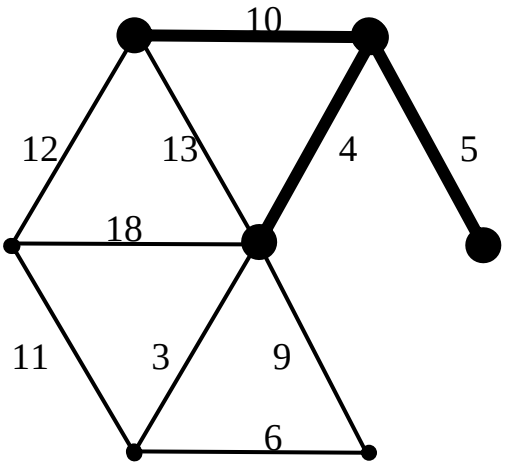


Рисунок 3.11

Далі аналогічно, на кожному кроці перевіряємо відсутність циклу. Додаємо ребра, доки всі вершини не будуть вичерпані (рисунки 3.12–3.14).

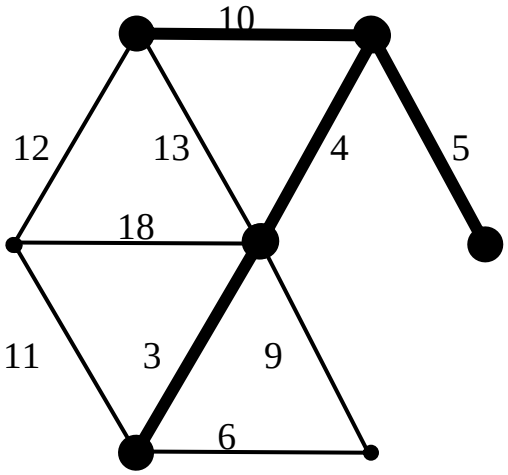


Рисунок 3.12

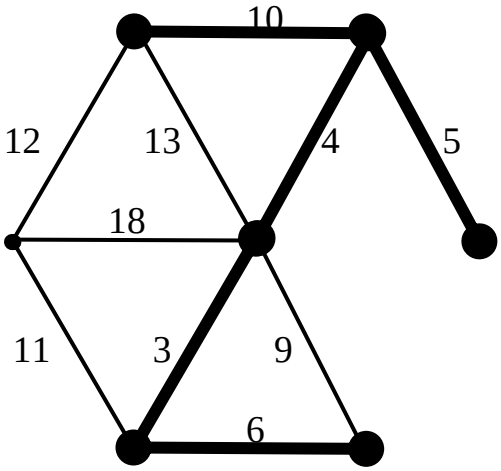


Рисунок 3.13

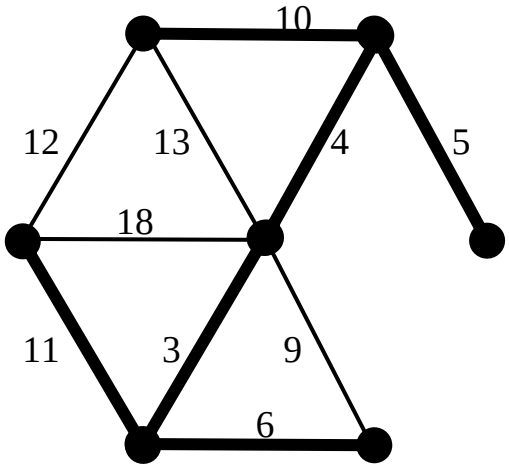


Рисунок 3.14 – Мінімальне остовне дерево, знайдене за алгоритмом найближчого сусіда

Цей розв’язок збігається із найденим за алгоритмом Краскала (див. рисунок 3.8), мінімальна вага остовного дерева дорівнює 39.

4 Паросполучення

У застосуванні теорії графів широкої популярності набула задача про паросполучення. Вона полягає у знаходженні в заданому графі паросполучення з найбільшою кількістю ребер – максимального паросполучення.

Підмножина ребер графа *G*, що не мають спільних вершин, називається *паросполученням M*.

Паросполучення графа *G* називають *максимальним*, якщо воно не міститься в жодному паросполученні з більшою кількістю ребер, і *найбільшим*, якщо кількість ребер у ньому найбільша серед усіх паросполучень графа *G*.

Досконалим паросполученням називається таке максимальне паросполучення, у якого сума ваг ребер буде мінімальною.

Приклад 4.1

Надана невід'ємна матриця *A* розміром 5 × 5, де елемент в *i*-му рядку і *j*-му стовпці відповідає вартості виконання *j*-го виду робіт *i*-м працівником. Потрібно знайти таку відповідність робіт працівникам, щоб витрати на оплату праці були найменшими. Побудувати дводольний граф, знайти максимальне паросполучення, найбільше паросполучення і досконале паросполучення.

$$A = \begin{pmatrix} 11 & 12 & 13 & 14 & 15 \\ 2 & 2 & 4 & 2 & 12 \\ 3 & 9 & 9 & 6 & 13 \\ 3 & 3 & 3 & 8 & 10 \\ 5 & 7 & 8 & 9 & 15 \end{pmatrix}$$

Запишемо *математичну модель* для нашої задачі.

Змінні x_{ij} набувають значення 1, якщо *i*-й кандидат займає *j*-ту вакансію. Якщо ця умова не виконується, то $x_{ij} = 0$.

Обмеження щодо кандидатів:

$$x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = 1$$
$$x_{21} + x_{22} + x_{23} + x_{24} + x_{25} = 1$$
$$x_{31} + x_{32} + x_{33} + x_{34} + x_{35} = 1$$
$$x_{41} + x_{42} + x_{43} + x_{44} + x_{45} = 1$$
$$x_{51} + x_{52} + x_{53} + x_{54} + x_{55} = 1$$

Обмеження щодо вакансій:

$$x_{11} + x_{21} + x_{31} + x_{41} + x_{51} = 1$$
$$x_{12} + x_{22} + x_{32} + x_{42} + x_{52} = 1$$
$$x_{13} + x_{23} + x_{33} + x_{43} + x_{53} = 1$$
$$x_{14} + x_{24} + x_{34} + x_{44} + x_{54} = 1$$
$$x_{15} + x_{25} + x_{35} + x_{45} + x_{55} = 1$$

Цільова функція:

$$F=11x_{11} + 12x_{12} + 13x_{13} + 14x_{14} + 15x_{15} + 2x_{21} + 2x_{22} + 4x_{23} + 2x_{24} + 12x_{25} + 3x_{31} + 9x_{32} + 9x_{33}+ 6x_{34} + 13x_{35} + 3x_{41} + 3x_{42} + 3x_{43} + 8x_{44} + 10x_{45} + 5x_{51} + 7x_{52} + 8x_{53} + 9x_{54} + 15x_{55} \rightarrow min$$

Цю задачу можна розв’язувати, наприклад, симплекс-методом [2]. Але існує спеціальний, більш легкий алгоритм саме для розв’язання таких задач – задач про призначення. Він називається *алгоритмом Куна або угорським алгоритмом*.

Алгоритм Куна

1 Перетворимо матрицю *A*. Віднімемо від кожного рядка мінімальний елемент цього рядка. У кожному рядку з'явиться по одному нулю. Запишемо матрицю у вигляді таблиці 4.1.

Таблиця 4.1

11	12	13	14	15	віднімаємо 11
2	2	4	2	12	віднімаємо 2
3	9	9	6	13	віднімаємо 3
3	3	3	8	10	віднімаємо 3
5	7	8	9	15	віднімаємо 5

Отримали таблицю 4.2.

Таблиця 4.2

0	1	2	3	4
0	0	2	0	10
0	6	6	3	10
0	0	0	5	7
0	2	3	4	10

Якщо в будь-якому стовпці не виявилося жодного нуля, з ним треба виконати таку саму процедуру, як і з рядками: знайти мінімальний елемент і відняти його з цього стовпця. Віднімаємо 4 від останнього стовпця (таблиця 4.3).

Таблиця 4.3

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

2 Будуємо дводольний граф *G*. *Дводольним графом* називається граф, множина вершин якого може бути розбита на дві підмножини так, що кожне ребро графа має одну вершину з першої підмножини і одну – з другої.

Перша підмножина X – множина рядків матриці A.

Друга підмножина Y – множина стовпців матриці A.

$X=\{ x_1, x_2, x_3, x_4, x_5\}$

$Y=\{ y_1, y_2, y_3, y_4, y_5\}$

Перенумеруємо відповідним чином отримані рядки та стовпці таблиці 4.3 (таблиця 4.4).

Таблиця 4.4

X \ Y	y_1	y_2	y_3	y_4	y_5
x_1	**0**	1	2	3	**0**
x_2	**0**	**0**	2	**0**	6
x_3	**0**	6	6	3	6
x_4	**0**	**0**	**0**	5	3
x_5	**0**	2	3	4	6

Ребра графа поєднують вершини, яким відповідають елементи таблиці 4.4 з нульовими елементами. Перший нуль міститься у 1-му рядку і в 1-му стовпці, будуємо ребро з вершини x_1 множини X у вершину y_1 множини Y. З іншими нулями робимо так само. Всього 10 ребер (рисунок 4.1).

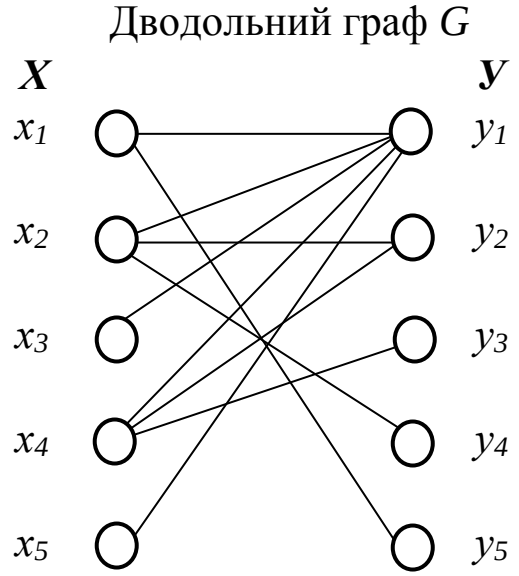


Рисунок 4.1 – Дводольний граф G

3 Будуємо *максимальне паросполучення*.

Починаємо з першої вершини множини X, наводимо жирною лінією будь-яке ребро з першої множини до другої. Всі інші ребра, поєднані з цими вершинами, будуть зображуватися тонкими лініями. Потім переходимо до другої вершини множини X і т. д. *Максимальним* буде паросполучення, коли більше жодного жирного ребра не можна додати (рисунок 4.2). Максимальних паросполучень може бути кілька, наприклад, перше жирне ребро можна було б провести не з x_1 до y_1, а до y_5.

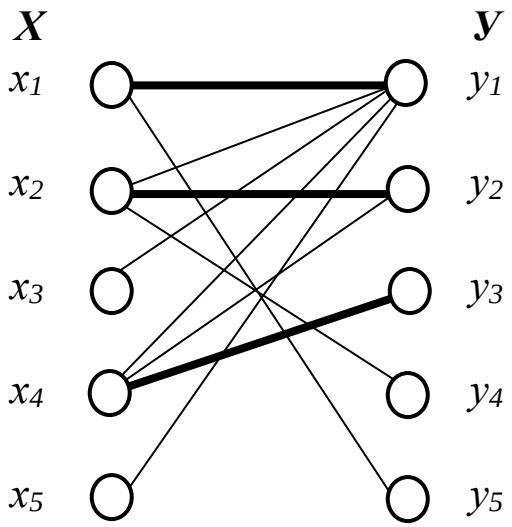


Рисунок 4.2 – Максимальне паросполучення *M* графа *G*

Наше максимальне паросполучення *M* містить 3 жирних ребра.

4 Знаходимо *найбільше паросполучення* отриманого дводольного графа *G*. Для цього можна скористатися алгоритмом *Форда–Фалкерсона*. Зображуємо граф і одне з його максимальних паросполучень (див. рисунок 4.2).

Перевіримо, чи є воно найбільшим. Якщо максимальне паросполучення не є найбільшим паросполученням, то можна обрати *ланцюг, що чергується*.

Вершину графа назвемо *насиченою*, якщо вона інцидентна (ребро входить або виходить з неї) жирному ребру, у протилежному випадку – *ненасиченою*.

Ланцюг, що виходить з ненасиченої вершини множини *X* та входить у ненасичену вершину множини *Y*, складається з непарної кількості тонких і жирних ребер, що йдуть по черзі, назвемо *ланцюгом, що чергується*.

Якщо такий ланцюг, що чергується, існує, то ми маємо його *обернути* – замінити в ньому жирні ребра на тонкі і, навпаки, тонкі на жирні.

Знайдемо ненасичені вершини максимального паросполучення. Це вершини x_3, x_5, y_4, y_5, з них виходять лише тонкі ребра (рисунок 4.3).

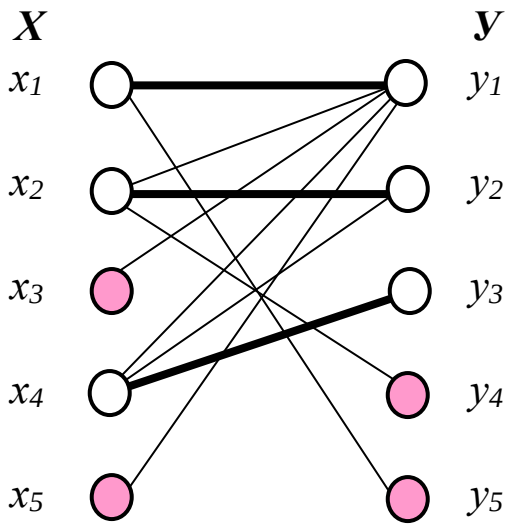


Рисунок 4.3 – Ненасичені вершини паросполучення x_3, x_5, y_4, y_5

Існує ланцюг, що чергується, з ненасиченої вершини x_3 до y_1 по тонкому ребру, з y_1 до x_1 по жирному, з x_1 до ненасиченої вершини y_5.

Замінюємо тонкі ребра на жирні за цим ланцюгом, а жирні на тонкі (рисунок 4.4).

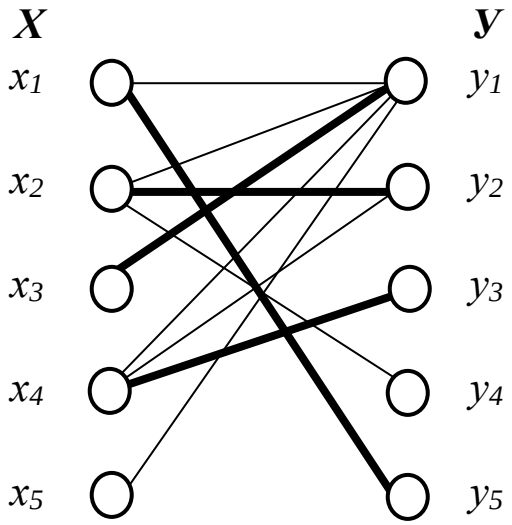


Рисунок 4.4 – Найбільше паросполучення

Більше ланцюгів, що чергуються, немає. Найбільше паросполучення в цьому випадку містить 4 жирних ребра (див. рисунок 4.4).

5 Знайдемо досконале паросполучення.

У *досконалого паросполучення* кількість жирних ребер має дорівнювати розмірності матриці A, тобто в нашому випадку має бути 5 жирних ребер, у найбільшому паросполученні їх у даному випадку 4 (див. рисунок 4.4).

Утворимо множини ненасичених найбільшим паросполученням вершин.

Розглянемо множини X_m і Y_m. До них входять ті вершини, які не охоплюються найбільшим паросполученням. Якщо ці множини порожні, то задачу розв'язано, знайдено оптимальний розв'язок.

У нашому прикладі це 5-та вершина множини X та 4-та вершина множини Y (див. рисунок 4.4).

$$X_m=\{x_5\},$$
$$Y_m=\{y_4\}$$

Знайдемо також допоміжні множини X' та Y'. Множина X' містить елементи, які обираються за правилом: виходимо з X_m, потім повертаємося до X, вперед по тонких ребрах, назад – по жирних. Ті елементи, які проходимо, записуємо у ці множини X' та Y'. Тобто множина X' містить перший елемент X_m, елемент x_5 , елемент, з яким поєднується x_5 у множині Y (це елемент y_1), належить Y', та елемент множини X, з яким поєднується y_5 (це елемент x_1), теж належить X'. Ребра мають чергуватися – тонкі та жирні (по тонких вперед, по жирних назад).

$X'=\{x_5, x_3\}$, позначимо «+» на рисунку 4.5.
$Y'=\{y_1\}$

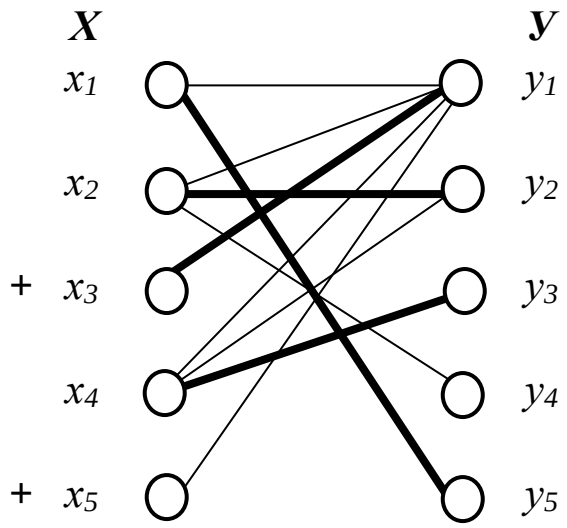


Рисунок 4.5 – Множина X'

За допомогою множин $X'=\{x_5, x_3\}$, $Y'=\{y_1\}$ будуємо α*-перетворення.*

Беремо в таблиці з п. 1 (див. таблицю 4.4) 3-й і 5-й рядки і **не** 1 стовпець. Це відповідає обраним елементам множин X' та Y', по рядках x_5, x_3 і **не** y_1 по стовпцях.

Таблиця 4.5

Y / X	y_1	y_2	y_3	y_4	y_5
x_1	0	1	2	3	0
x_2	0	0	2	0	6
x_3	0	**6**	**6**	**3**	**6**
x_4	0	0	0	5	3
x_5	0	**2**	**3**	**4**	**6**

Знаходимо серед обраних елементів (жирно наведені елементи таблиці 4.5) мінімальне число. Це число 2.

Зауважимо, що коли серед обраних елементів є нульові, то ви помилились при обчисленні.

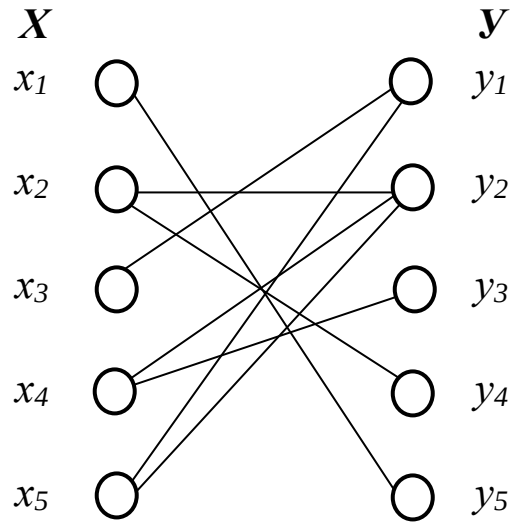
З рядків x_3, x_5 треба відняти 2, а до стовпця y_1 треба додати 2. (таблиця 4.6) Числа на перетині цих рядків та стовпців у таблиці 4.6 (-2+2=0) не зміняться.

Таблиця 4.6

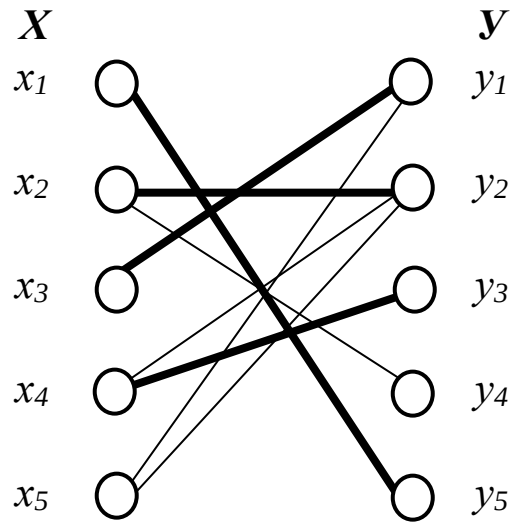
Y / X	y_1	y_2	y_3	y_4	y_5
x_1	2	1	2	3	**0**
x_2	2	**0**	2	**0**	6
x_3	**0**	4	4	1	4
x_4	2	**0**	**0**	5	3
x_5	**0**	**0**	1	2	4

Далі треба повернутися до пункту **2** та повторити алгоритм. З цієї матриці будуємо дводольний граф.

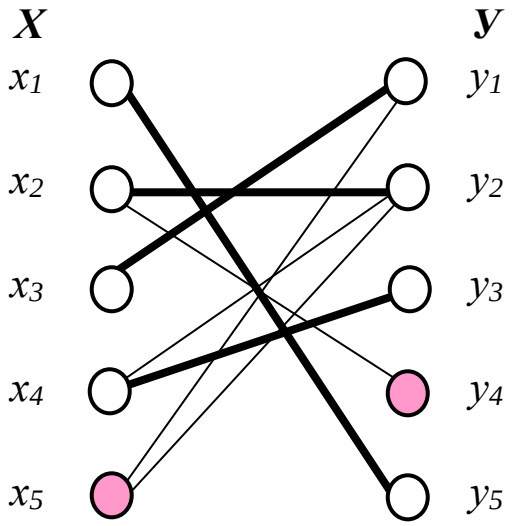
Дводольний граф G



Максимальне паросполучення (див. пункт **3**).



Існують ненасичені вершини в максимальному парослученні x_5, y_4.



Існує ланцюг, що чергується. Він поєднує вершини x_5, y_2, x_2, y_4 вперед по тонких, назад – по жирних ребрах. Обертаючи їх, отримаємо найбільше парослучення. Це отримане найбільше парослучення є також досконалим парослученням, оскільки жирних ребер рівно 5 (рисунок 4.6.).

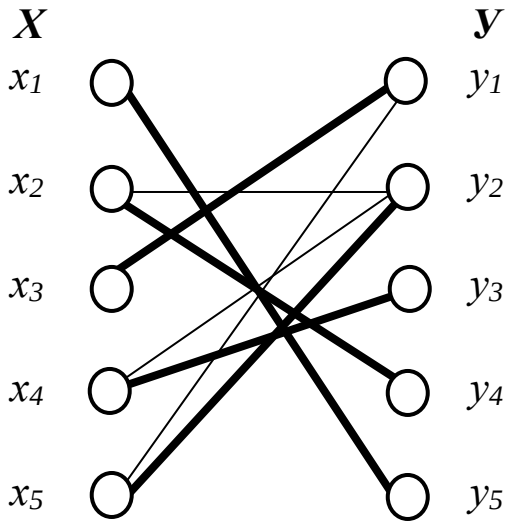


Рисунок 4.6 – Досконале парослучення

Переносимо цей розподіл на матрицю A. Відповідні жирні ребра (див. рисунок 4.6) – клітинки в таблиці 4.7. Це покаже, який виконавець, яку роботу має виконати, а сума виділених елементів – оплату праці.

Таблиця 4.7

X \ Y	y_1	y_2	y_3	y_4	y_5
x_1	11	12	13	14	**15**
x_2	2	2	4	**2**	12
x_3	**3**	9	9	6	13
x_4	3	3	**3**	8	10
x_5	5	**7**	8	9	15

Мінімальна вартість дорівнює сумі вартостей у виділених клітинках.

$$F_{min} = 15+2+3+3+7 = 30$$

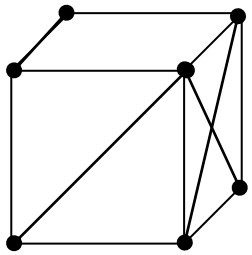
ВАРІАНТИ ЗАВДАНЬ

Завдання 1

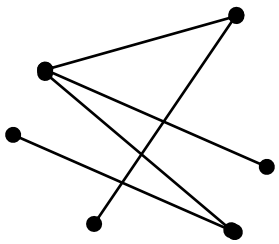
Перенумерувати вершини та ребра заданого графа. Скласти матриці суміжності S та інцидентності T (див. приклад 1.1).

а) для неорієнтованих графів

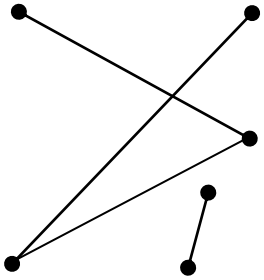
1



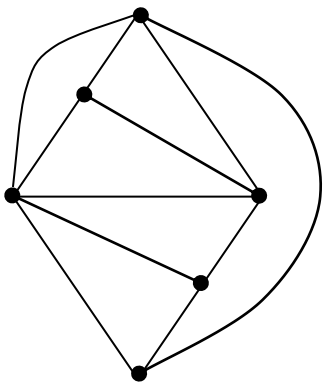
2



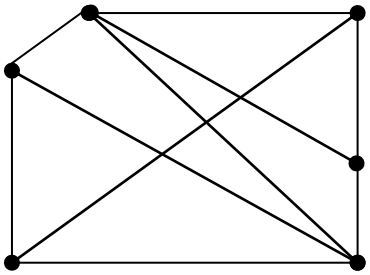
3



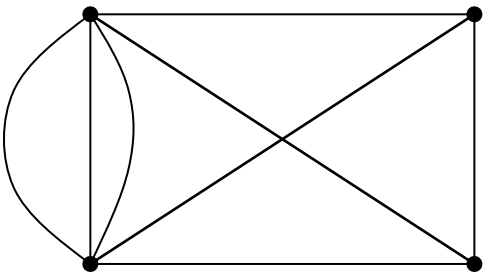
4



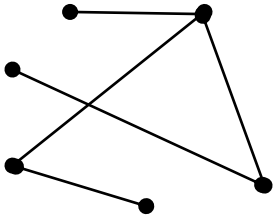
5



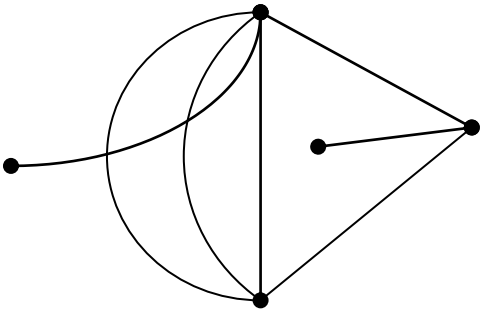
6



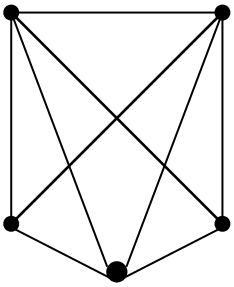
7



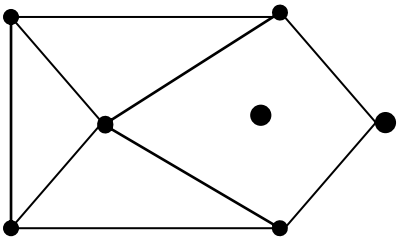
8



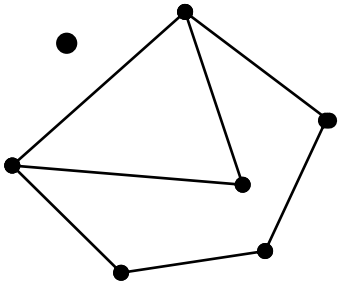
9



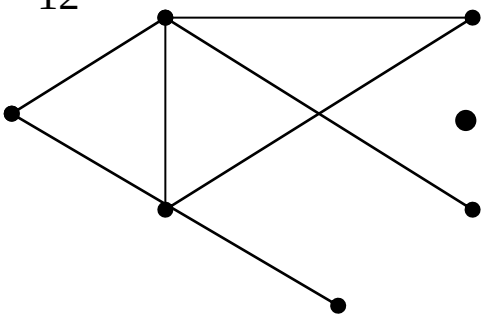
10



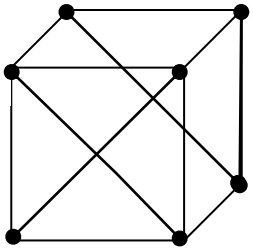
11



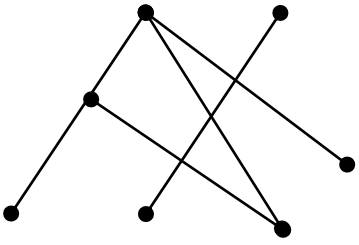
12



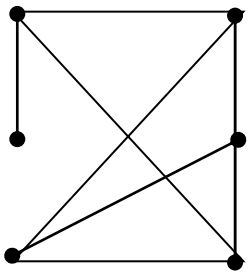
13



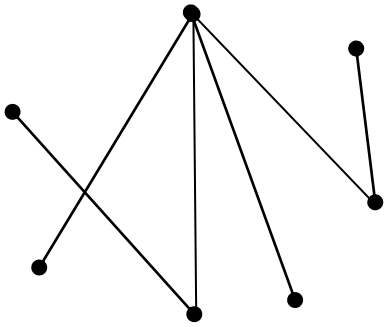
14



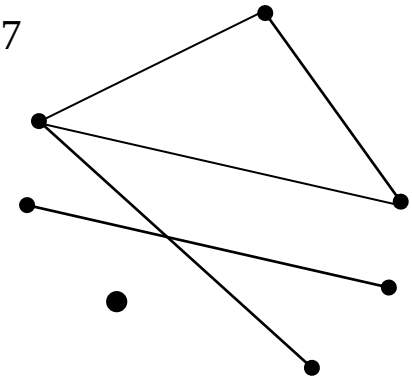
15



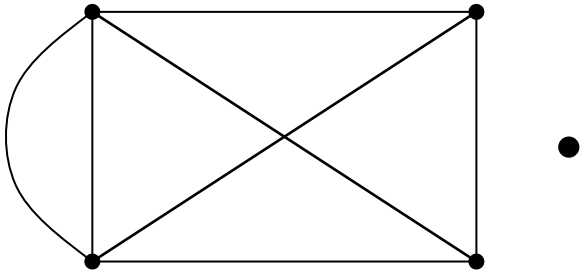
16



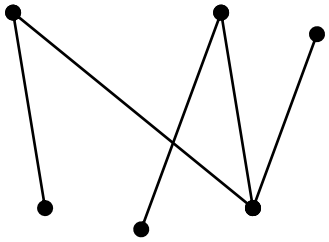
17



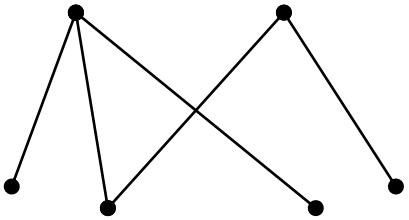
18



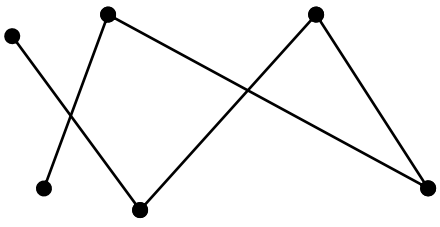
19



20

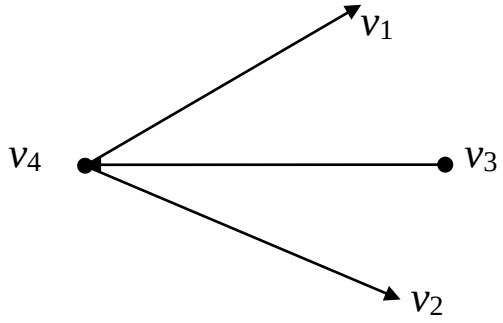


21

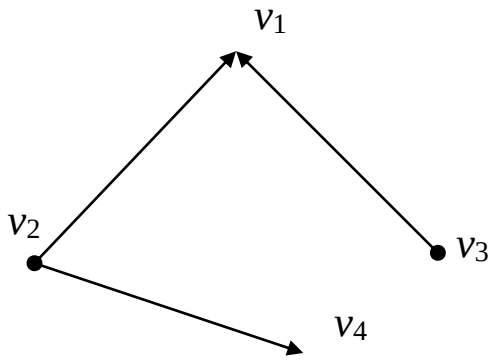


б) для орієнтованих графів

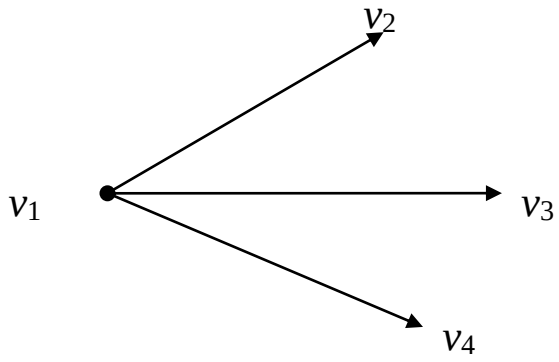
1



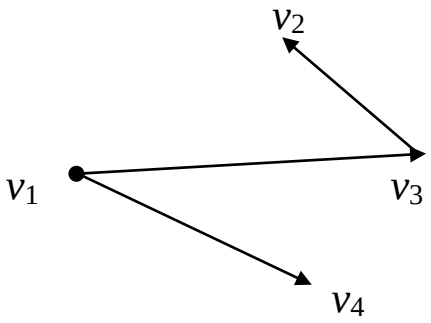
4



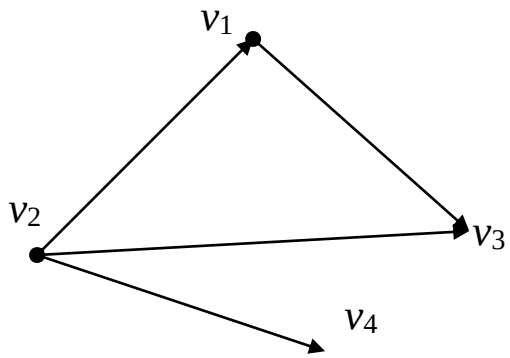
2



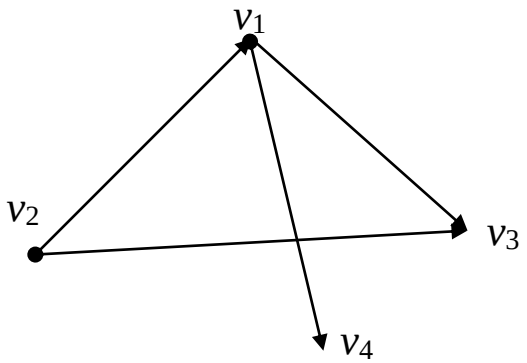
5



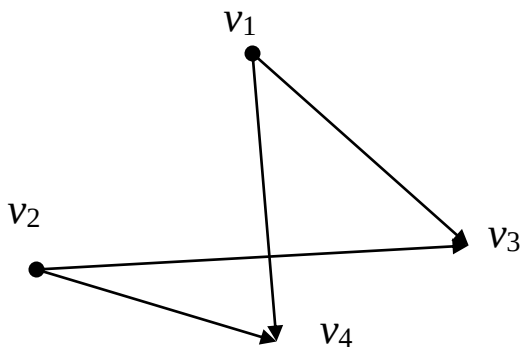
3



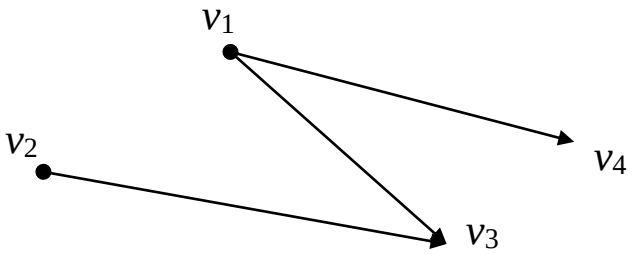
6



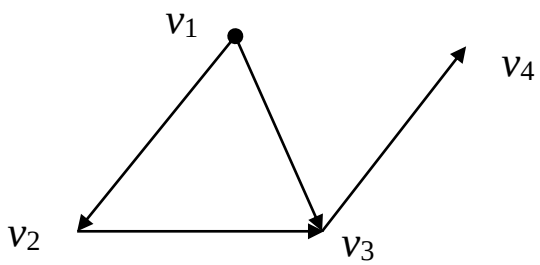
7



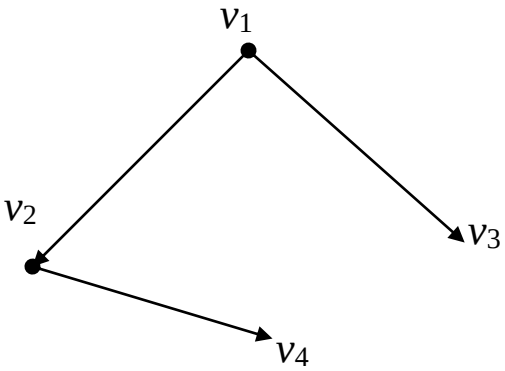
11



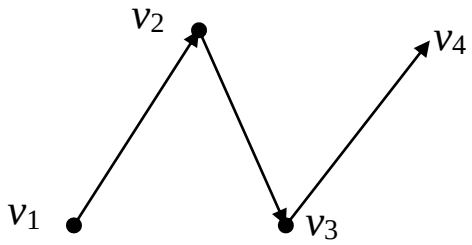
8



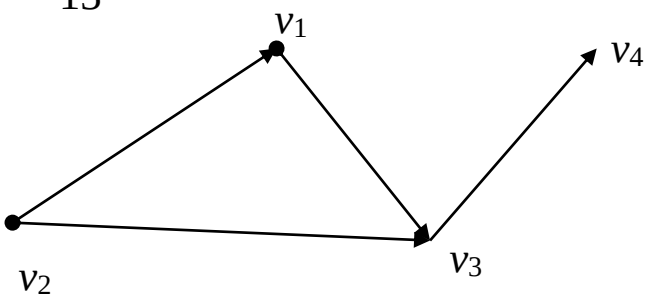
12



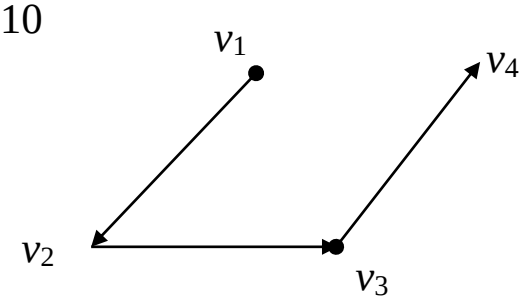
9



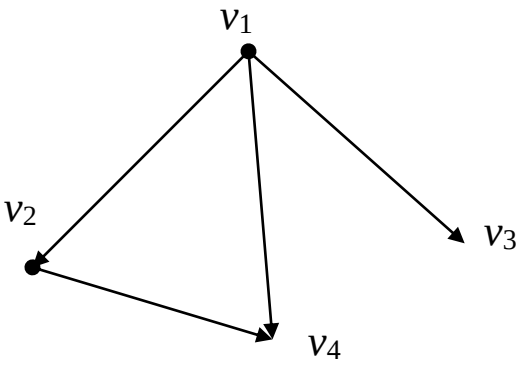
13



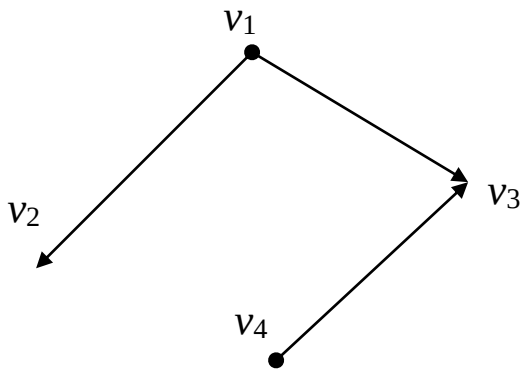
10



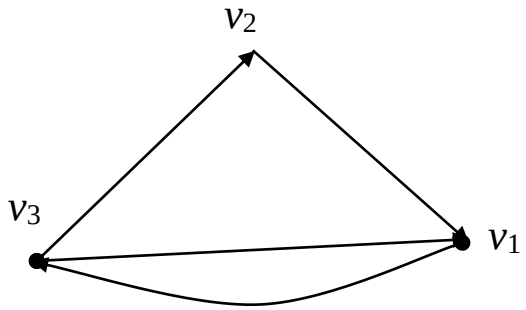
14



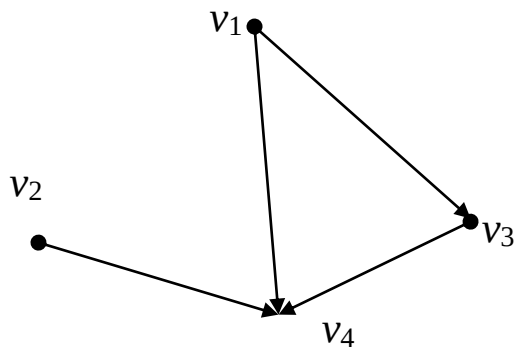
15



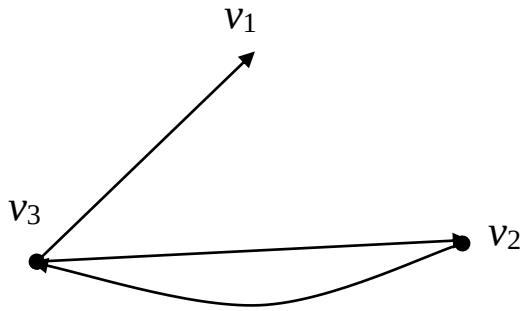
19



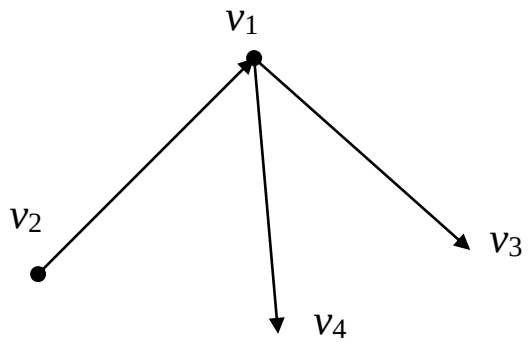
16



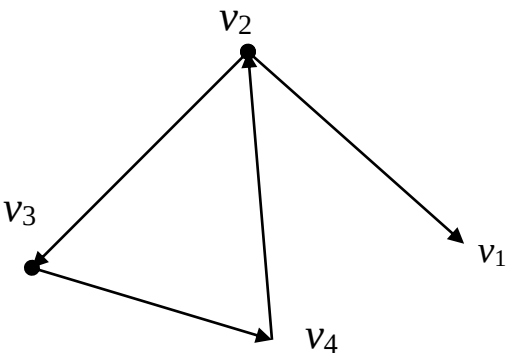
20



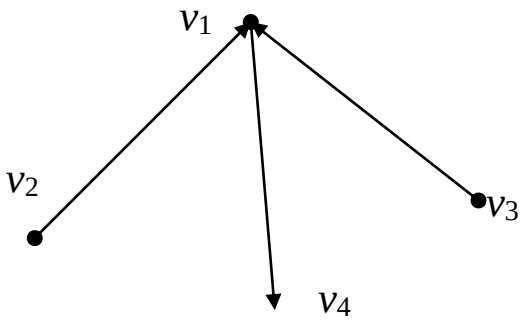
17



21



18



Завдання 2

За заданою матрицею суміжності S побудувати граф, а за отриманим графом скласти матрицю інцидентності T (див. приклад 1.2. та приклад 1.1.)

1 $S=\begin{pmatrix}1&0&0&2\\0&1&2&0\\0&2&1&0\\2&0&0&1\end{pmatrix}$

2 $S=\begin{pmatrix}1&2&0&2\\2&0&0&0\\0&0&0&1\\2&0&1&1\end{pmatrix}$

3 $S=\begin{pmatrix}0&3&0&2\\3&2&1&0\\0&1&0&1\\2&0&1&2\end{pmatrix}$

4 $S=\begin{pmatrix}1&1&0&1\\1&1&1&0\\0&1&0&1\\1&0&1&1\end{pmatrix}$

5 $S=\begin{pmatrix}3&2&0&1\\2&0&0&0\\0&0&0&1\\1&0&1&1\end{pmatrix}$

6 $S=\begin{pmatrix}2&1&0&1\\1&2&1&0\\0&1&0&3\\1&0&3&3\end{pmatrix}$

7 $S=\begin{pmatrix}2&1&1&1\\1&2&2&1\\1&2&1&1\\1&1&1&1\end{pmatrix}$

8 $S=\begin{pmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{pmatrix}$

9 $S=\begin{pmatrix}0&1&1&1\\1&0&1&1\\1&1&0&1\\1&1&1&0\end{pmatrix}$

10 $S=\begin{pmatrix}0&1&1&2\\1&1&2&1\\1&2&0&1\\2&1&1&0\end{pmatrix}$

11 $S=\begin{pmatrix}0&1&1&0\\1&0&0&1\\1&0&0&1\\0&1&1&2\end{pmatrix}$

12 $S=\begin{pmatrix}0&1&1&0\\1&2&2&1\\1&2&2&1\\0&1&1&2\end{pmatrix}$

13 $S=\begin{pmatrix}3&1&1&1\\1&1&2&1\\1&2&2&1\\1&1&1&0\end{pmatrix}$

14 $S=\begin{pmatrix}2&1&1&1\\1&2&1&1\\1&1&2&1\\1&1&1&0\end{pmatrix}$

15 $S=\begin{pmatrix}2&0&1&1\\0&0&1&2\\1&1&0&1\\1&2&1&0\end{pmatrix}$

16 $S=\begin{pmatrix}1&0&1&1\\0&1&1&1\\1&1&0&1\\1&1&1&4\end{pmatrix}$

17 $S=\begin{pmatrix}1&3&1&1\\3&2&1&0\\1&1&1&1\\1&0&1&0\end{pmatrix}$

18 $S=\begin{pmatrix}3&3&0&0\\3&3&0&0\\0&0&3&0\\0&0&0&3\end{pmatrix}$

19 $S=\begin{pmatrix}1&2&0&1\\2&1&2&0\\0&2&1&0\\1&0&0&1\end{pmatrix}$

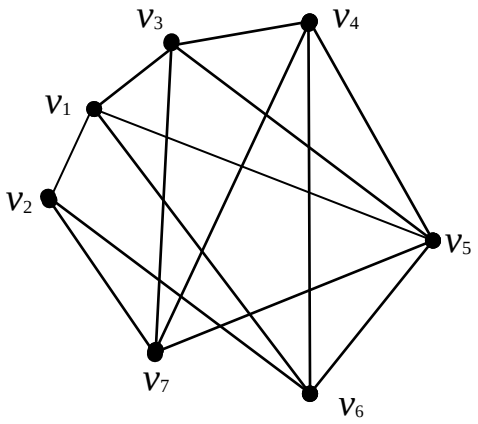
20 $S=\begin{pmatrix}1&1&1&1\\1&2&2&0\\1&2&1&0\\1&0&0&1\end{pmatrix}$

21 $S=\begin{pmatrix}1&1&1&1\\1&0&2&1\\1&2&1&1\\1&1&1&1\end{pmatrix}$

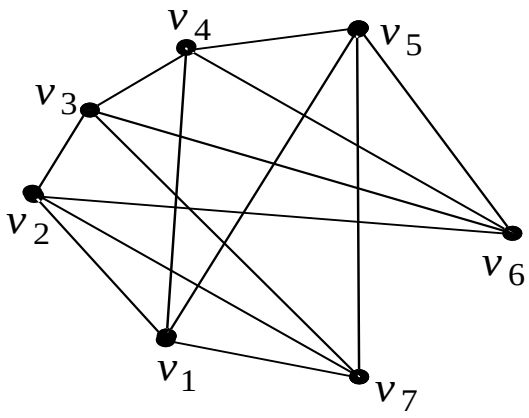
Завдання 3

Перевірити, чи має заданий граф ейлерів цикл або ейлерів ланцюг (див. приклад 2.2).

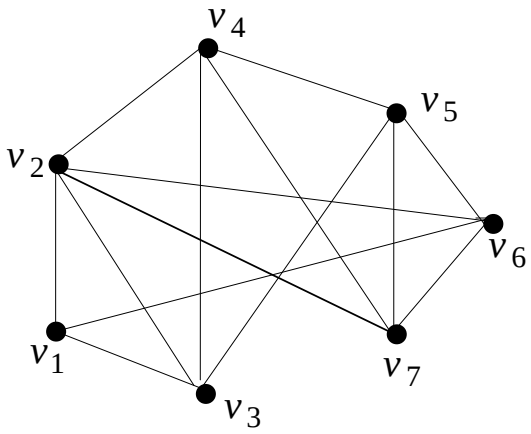
Варіант 1



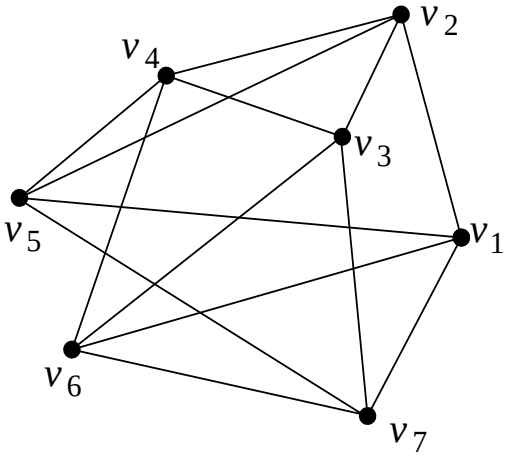
Варіант 2



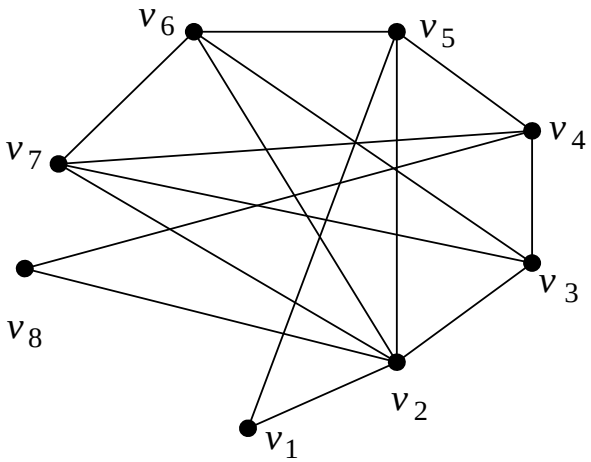
Варіант 3



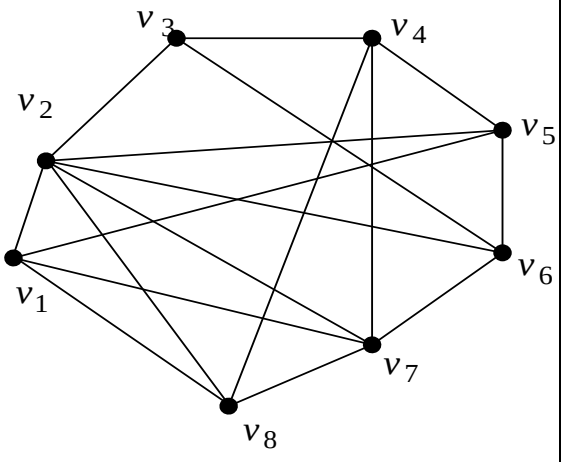
Варіант 4



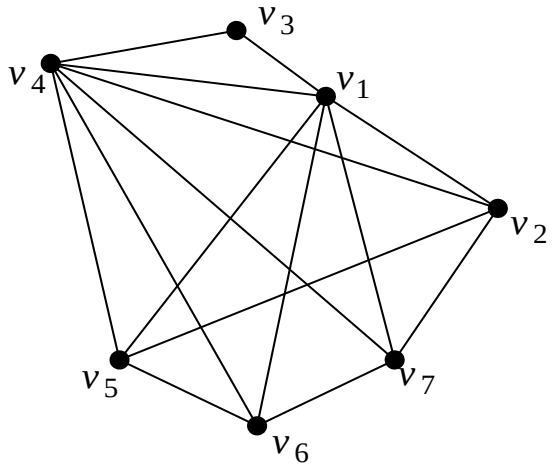
Варіант 5



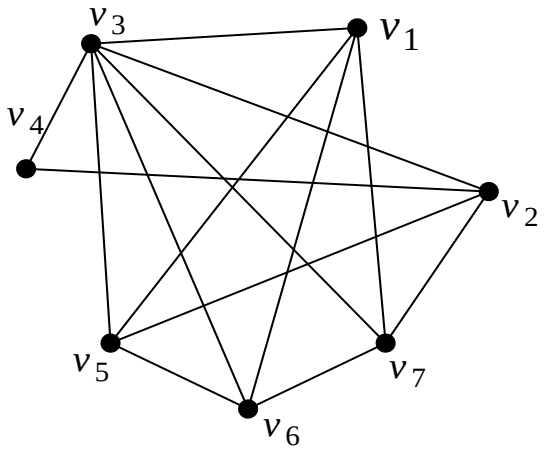
Варіант 6



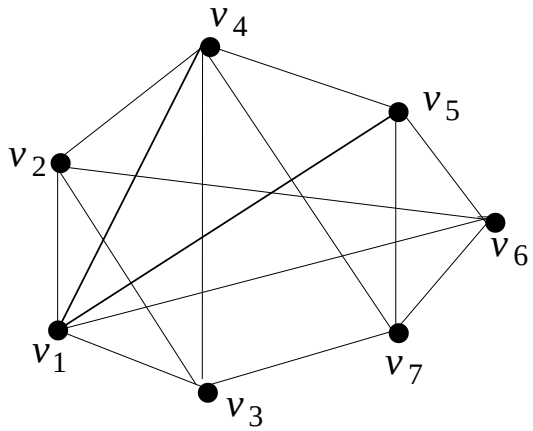
Варіант 7



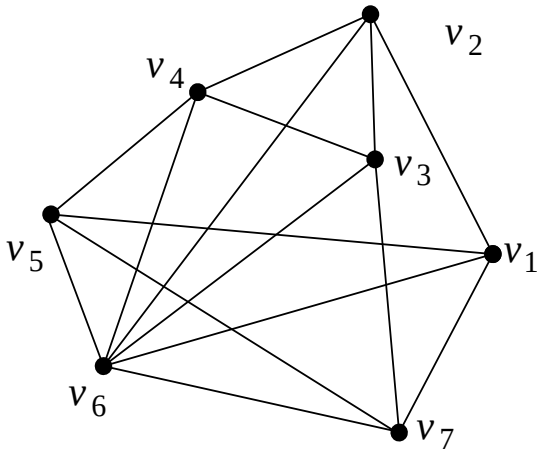
Варіант 8



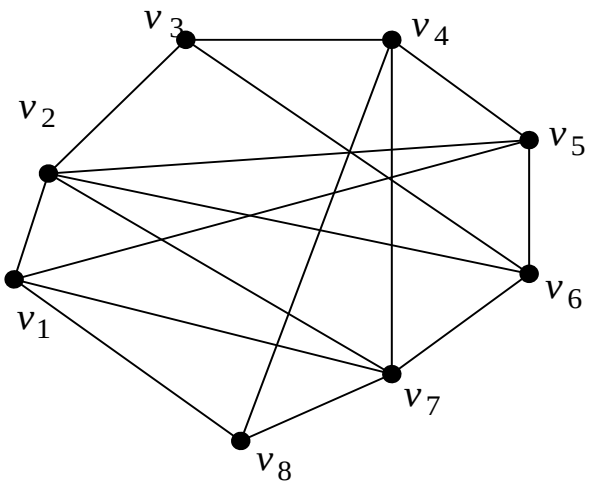
Варіант 9



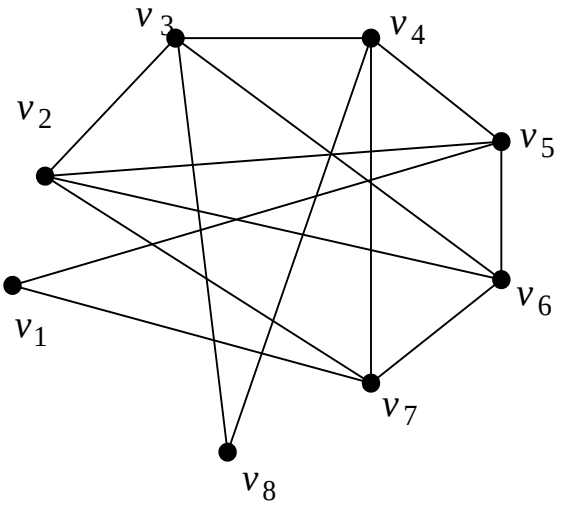
Варіант 10



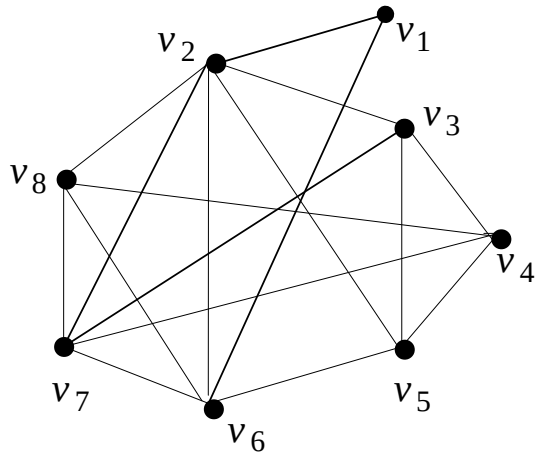
Варіант 11



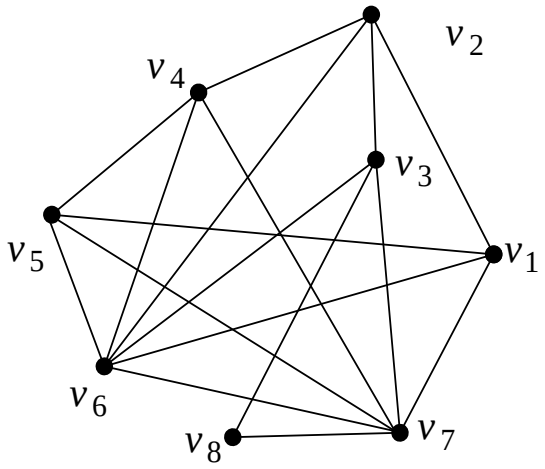
Варіант 12



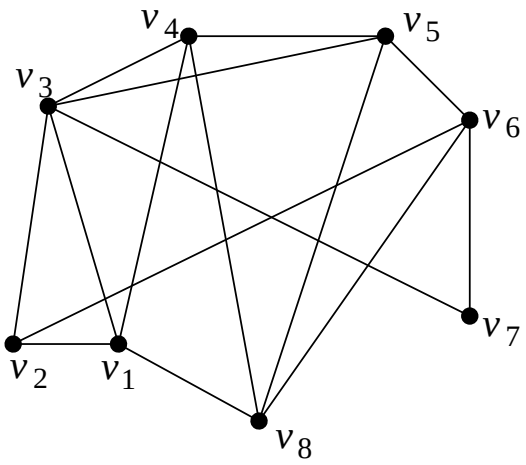
Варіант 13



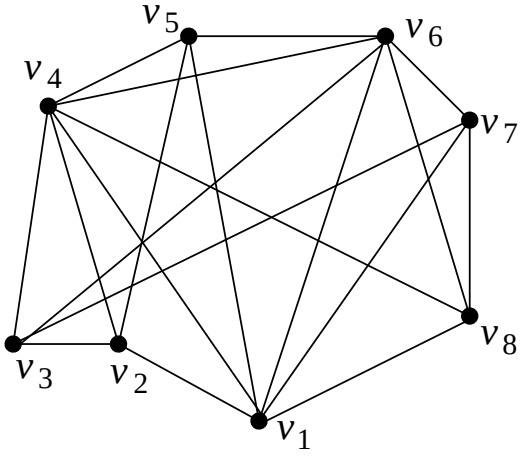
Варіант 14



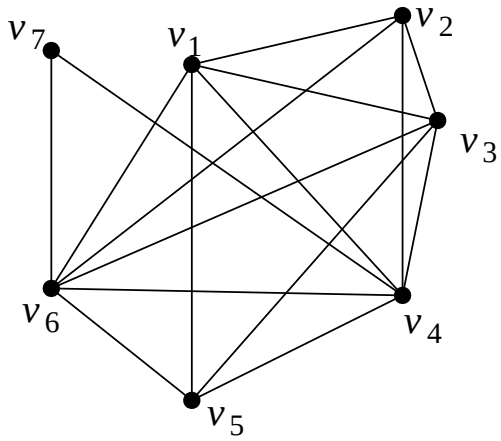
Варіант 15



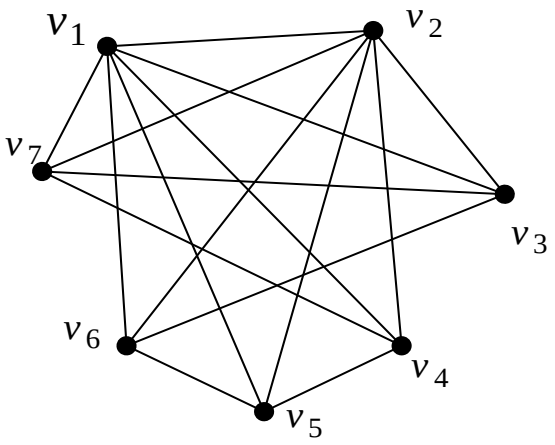
Варіант 16



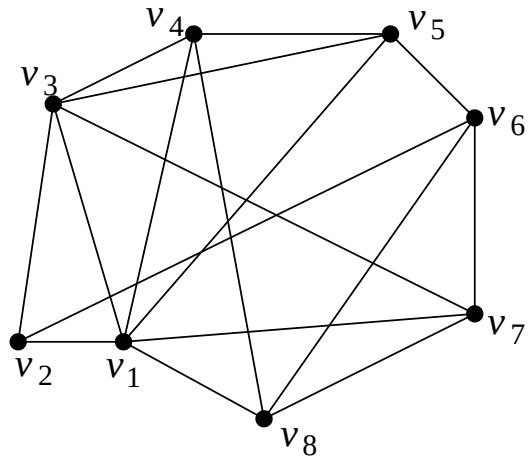
Варіант 17



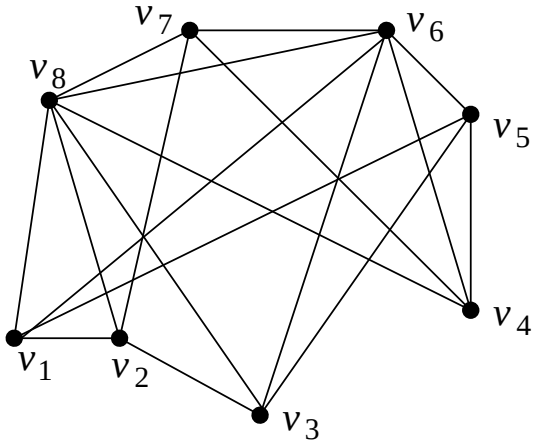
Варіант 18



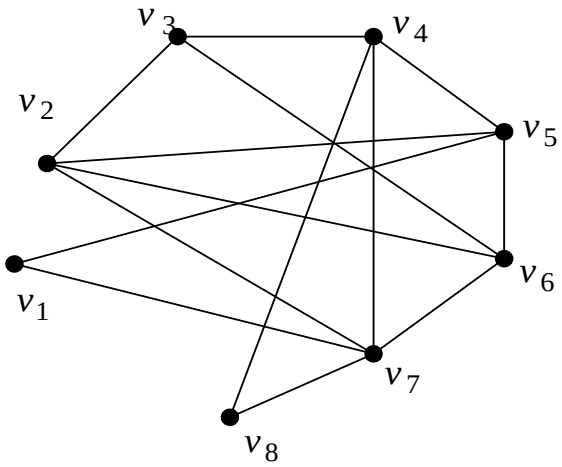
Варіант 19



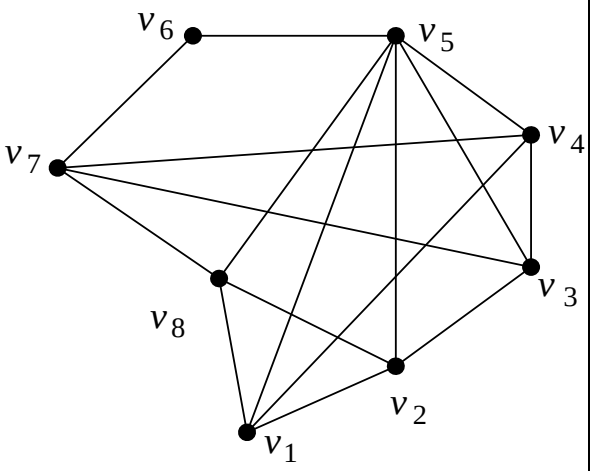
Варіант 20



Варіант 21



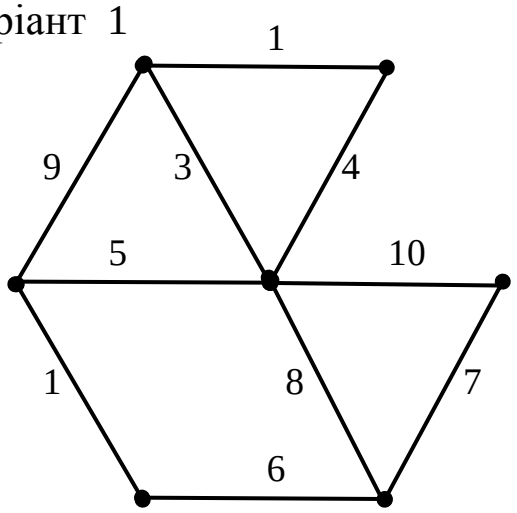
Варіант 22



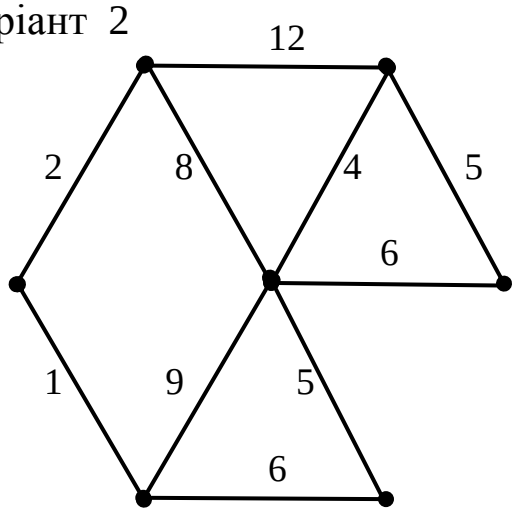
Завдання 4

Задано зважений граф. Знайти остов графа мінімальної ваги за алгоритмами Краскала і найближчого сусіда (див. приклад 3.1).

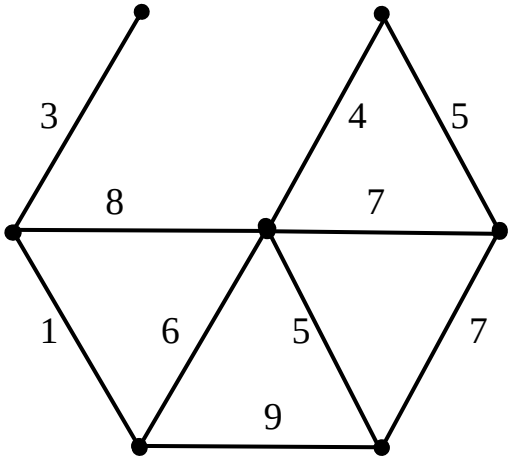
Варіант 1



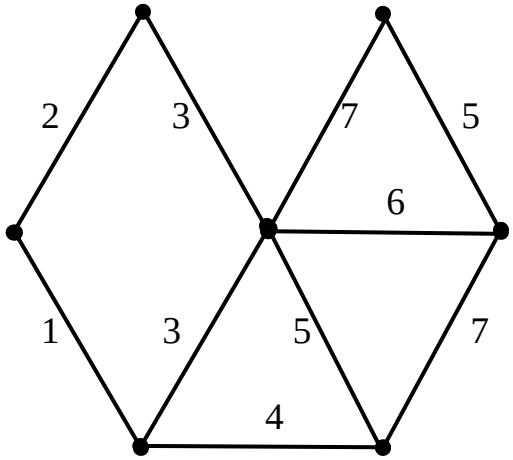
Варіант 2



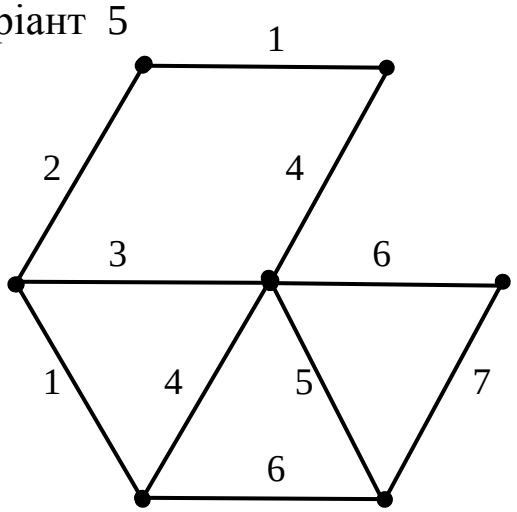
Варіант 3



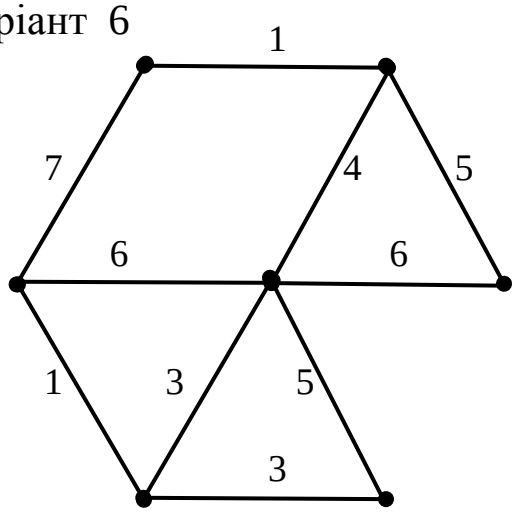
Варіант 4



Варіант 5

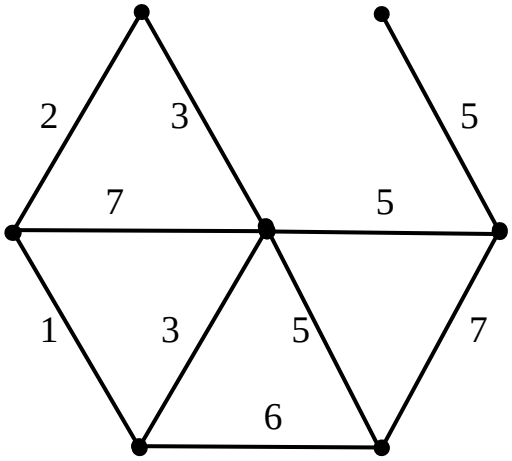


Варіант 6



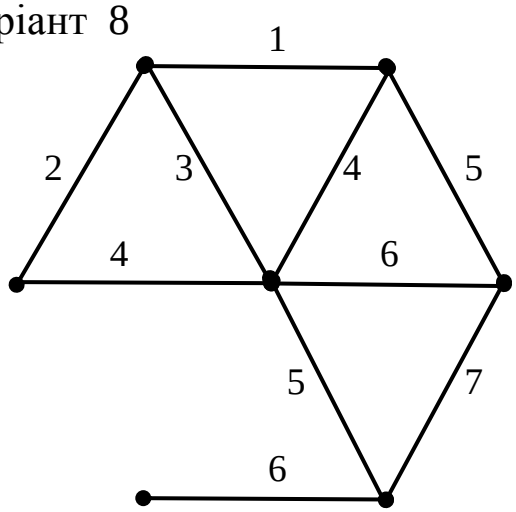
Варіант 7

.



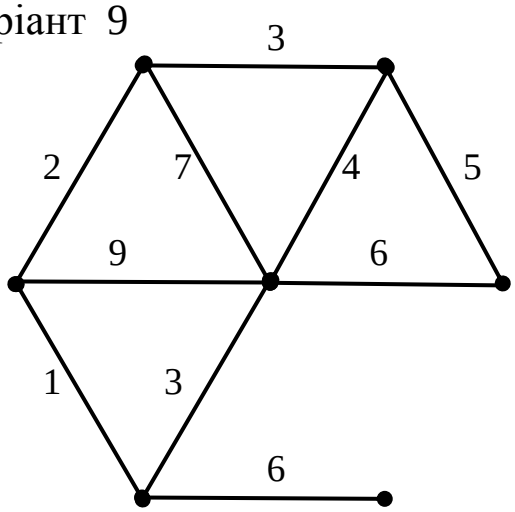
Варіант 8

.



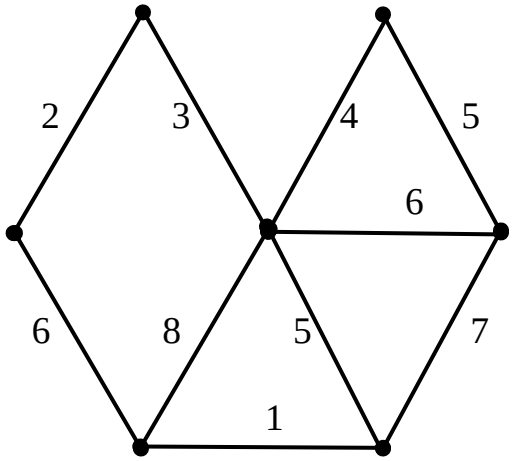
Варіант 9

.



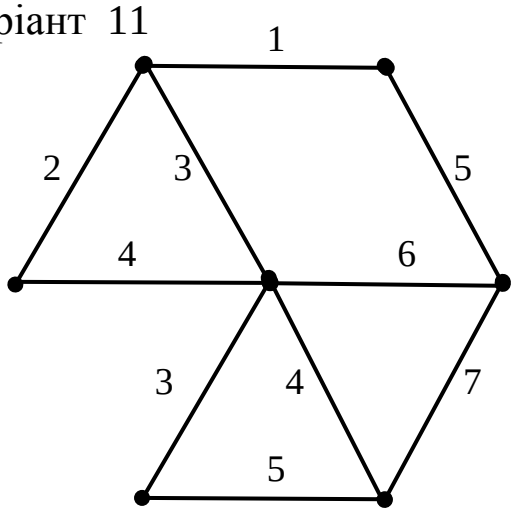
Варіант 10

.



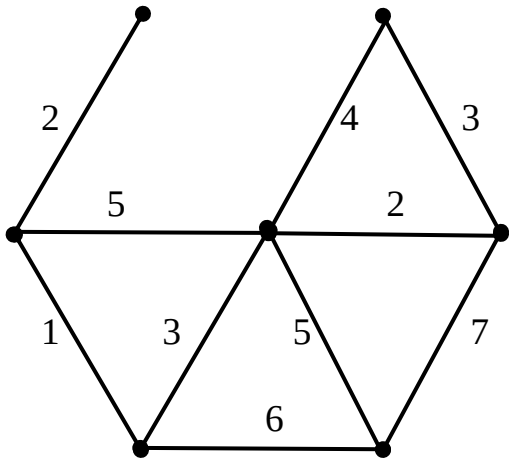
Варіант 11

.

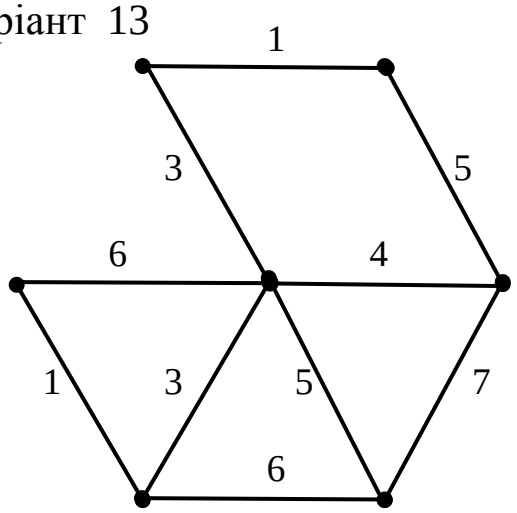


Варіант 12

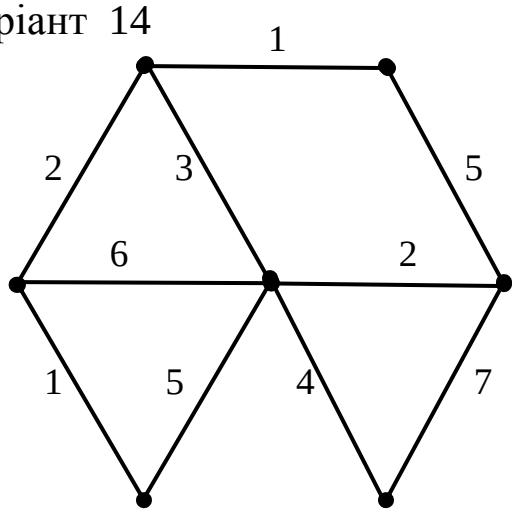
.



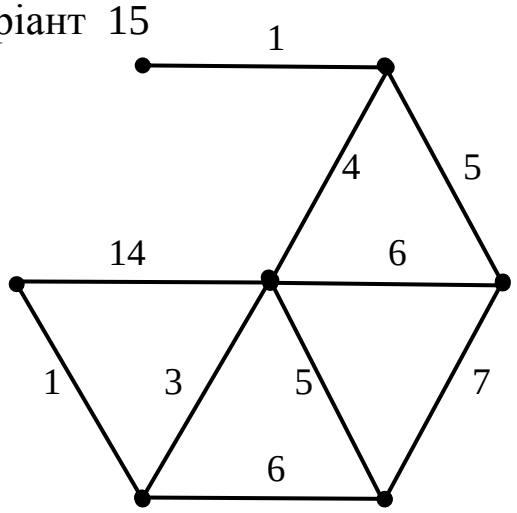
Варіант 13
.



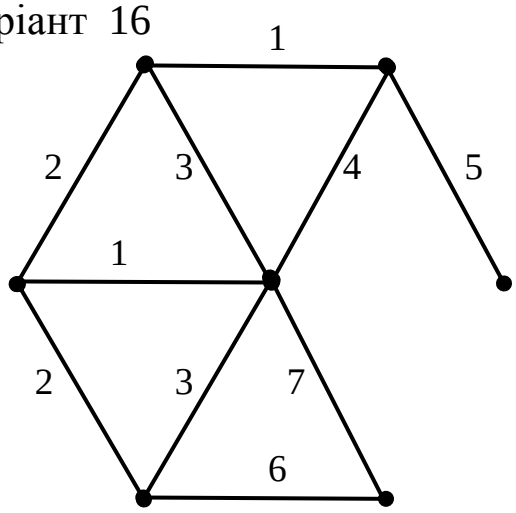
Варіант 14
.



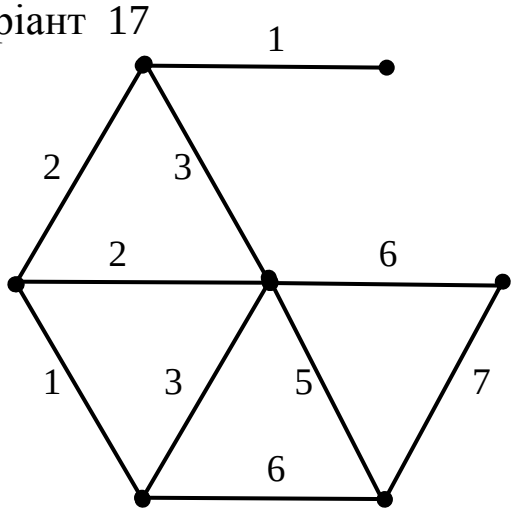
Варіант 15
.



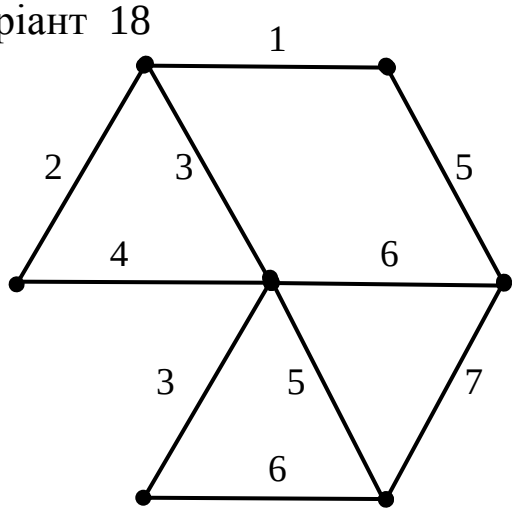
Варіант 16
.



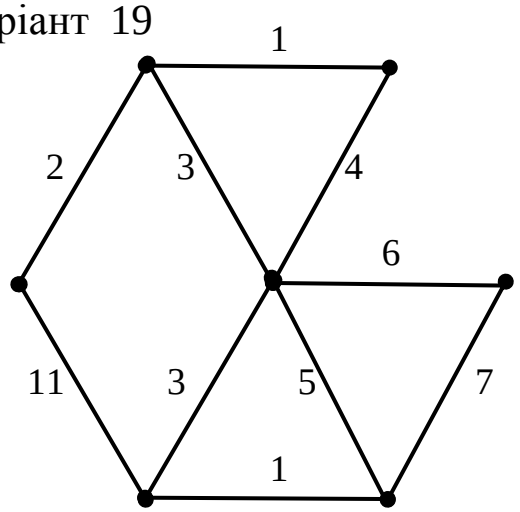
Варіант 17
.



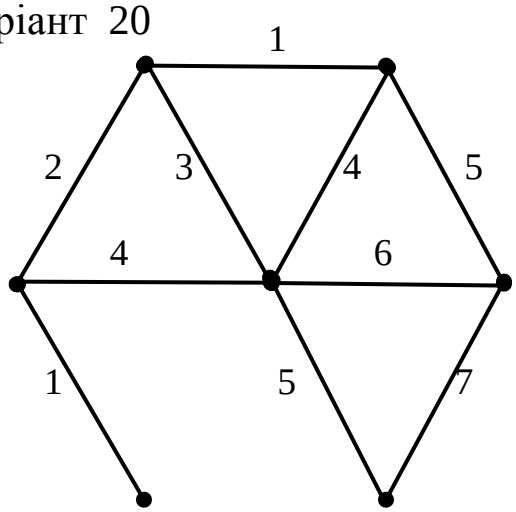
Варіант 18
.



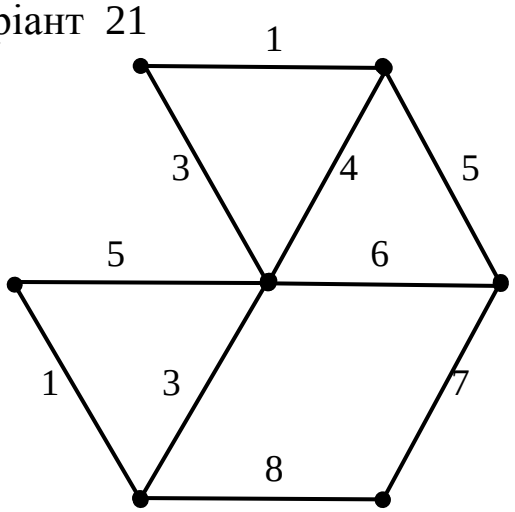
Варіант 19
.



Варіант 20
.



Варіант 21
.



Завдання 5

Надана невід'ємна матриця *A* розміром 5×5, де елемент в *i*-му рядку і *j*-му стовпці відповідає вартості виконання *i*-м працівником *j*-го виду робіт. Потрібно знайти таку відповідність робіт працівникам, щоб витрати на оплату праці були найменшими. Побудувати дводольний граф, знайти максимальне паросполучення, найбільше паросполучення і досконале паросполучення (див. приклад 4.1).

Варіант 1

$$A=\begin{pmatrix}13 & 9 & 8 & 14 & 12\\ 9 & 10 & 9 & 7 & 13\\ 15 & 13 & 10 & 7 & 9\\ 11 & 8 & 11 & 5 & 10\\ 13 & 14 & 14 & 7 & 7\end{pmatrix}$$

Варіант 2

$$A=\begin{pmatrix}5 & 7 & 12 & 13 & 9\\ 10 & 8 & 13 & 6 & 13\\ 13 & 15 & 11 & 10 & 14\\ 12 & 11 & 8 & 10 & 8\\ 5 & 7 & 13 & 12 & 13\end{pmatrix}$$

Варіант 3

$$A=\begin{pmatrix}13 & 5 & 9 & 14 & 14\\ 10 & 7 & 10 & 6 & 5\\ 7 & 11 & 12 & 12 & 8\\ 8 & 10 & 14 & 10 & 7\\ 9 & 10 & 9 & 6 & 11\end{pmatrix}$$

Варіант 4

$$A=\begin{pmatrix}10 & 6 & 5 & 14 & 14\\ 6 & 12 & 8 & 10 & 10\\ 14 & 8 & 14 & 5 & 6\\ 9 & 12 & 13 & 6 & 10\\ 7 & 12 & 8 & 11 & 10\end{pmatrix}$$

Варіант 5

$$A=\begin{pmatrix}5 & 9 & 5 & 8 & 13\\ 14 & 6 & 8 & 6 & 10\\ 6 & 13 & 12 & 5 & 14\\ 11 & 14 & 12 & 13 & 13\\ 11 & 5 & 10 & 10 & 8\end{pmatrix}$$

Варіант 6

$$A=\begin{pmatrix}11 & 9 & 14 & 13 & 14\\ 6 & 7 & 10 & 12 & 8\\ 7 & 10 & 9 & 6 & 7\\ 7 & 14 & 9 & 11 & 10\\ 10 & 11 & 9 & 9 & 8\end{pmatrix}$$

Варіант 7

$$A=\begin{pmatrix}9 & 8 & 9 & 8 & 8\\ 11 & 14 & 6 & 9 & 10\\ 15 & 11 & 15 & 15 & 14\\ 14 & 9 & 7 & 6 & 7\\ 14 & 12 & 14 & 7 & 13\end{pmatrix}$$

Варіант 8

$$A=\begin{pmatrix}8 & 9 & 8 & 13 & 14\\ 12 & 7 & 10 & 5 & 11\\ 13 & 9 & 9 & 12 & 8\\ 10 & 6 & 14 & 14 & 11\\ 8 & 6 & 8 & 6 & 12\end{pmatrix}$$

Варіант 9

$$A=\begin{pmatrix}14 & 14 & 10 & 6 & 7\\ 8 & 10 & 11 & 10 & 12\\ 13 & 14 & 10 & 8 & 13\\ 10 & 10 & 10 & 11 & 10\\ 13 & 14 & 5 & 9 & 6\end{pmatrix}$$

Варіант 10

$$A=\begin{pmatrix}6 & 14 & 12 & 12 & 7\\ 9 & 8 & 10 & 6 & 14\\ 12 & 8 & 5 & 10 & 10\\ 7 & 6 & 11 & 7 & 9\\ 8 & 12 & 7 & 8 & 15\end{pmatrix}$$

Варіант 11

$$A=\begin{pmatrix}12 & 9 & 5 & 7 & 14\\ 13 & 7 & 13 & 13 & 11\\ 10 & 10 & 13 & 9 & 9\\ 13 & 9 & 6 & 7 & 10\\ 7 & 9 & 7 & 9 & 12\end{pmatrix}$$

Варіант 12

$$A=\begin{pmatrix}12 & 11 & 7 & 13 & 15\\ 14 & 12 & 12 & 12 & 8\\ 7 & 5 & 6 & 8 & 14\\ 6 & 9 & 12 & 13 & 10\\ 10 & 14 & 12 & 11 & 6\end{pmatrix}$$

Варіант 13

$$A=\begin{pmatrix}14 & 12 & 7 & 8 & 8\\ 11 & 10 & 8 & 5 & 6\\ 6 & 9 & 10 & 13 & 9\\ 8 & 8 & 11 & 12 & 6\\ 6 & 10 & 14 & 12 & 11\end{pmatrix}$$

Варіант 14

$$A=\begin{pmatrix}5 & 5 & 13 & 14 & 11\\ 7 & 6 & 13 & 9 & 11\\ 9 & 12 & 15 & 10 & 9\\ 5 & 15 & 7 & 6 & 15\\ 15 & 8 & 8 & 7 & 7\end{pmatrix}$$

Варіант 15

$$A=\begin{pmatrix}7 & 9 & 13 & 11 & 11\\ 6 & 10 & 6 & 6 & 6\\ 7 & 8 & 7 & 14 & 9\\ 9 & 10 & 11 & 10 & 12\\ 11 & 10 & 12 & 11 & 13\end{pmatrix}$$

Варіант 16

$$A=\begin{pmatrix}13 & 11 & 10 & 7 & 15\\ 7 & 13 & 6 & 9 & 11\\ 11 & 7 & 10 & 13 & 13\\ 15 & 14 & 8 & 14 & 10\\ 15 & 10 & 13 & 6 & 13\end{pmatrix}$$

Варіант 17

$$A=\begin{pmatrix}5 & 15 & 14 & 9 & 7\\ 9 & 14 & 12 & 5 & 13\\ 11 & 6 & 13 & 13 & 11\\ 9 & 13 & 8 & 9 & 13\\ 11 & 7 & 6 & 9 & 14\end{pmatrix}$$

Варіант 18

$$A=\begin{pmatrix}6 & 11 & 6 & 13 & 8\\ 13 & 12 & 8 & 6 & 13\\ 9 & 11 & 7 & 6 & 9\\ 5 & 6 & 11 & 11 & 13\\ 7 & 11 & 11 & 9 & 5\end{pmatrix}$$

Варіант 19

$$A=\begin{pmatrix}11 & 11 & 13 & 13 & 9\\ 7 & 15 & 8 & 13 & 8\\ 13 & 10 & 7 & 11 & 7\\ 6 & 11 & 12 & 8 & 14\\ 15 & 12 & 13 & 7 & 11\end{pmatrix}$$

Варіант 20

$$A=\begin{pmatrix}9 & 13 & 15 & 9 & 12\\ 8 & 6 & 11 & 11 & 14\\ 8 & 13 & 14 & 13 & 13\\ 13 & 5 & 12 & 9 & 14\\ 13 & 9 & 13 & 9 & 12\end{pmatrix}$$

Варіант 21

$$A=\begin{pmatrix} 7 & 7 & 8 & 6 & 10 \\ 9 & 12 & 8 & 13 & 14 \\ 8 & 7 & 14 & 8 & 8 \\ 6 & 14 & 9 & 6 & 9 \\ 8 & 10 & 13 & 15 & 10 \end{pmatrix}$$

СПИСОК ЛІТЕРАТУРИ

1 Удодова, О. І. Дискретна математика. Ч. 2. Елементи теорії графів. Елементи комбінаторного аналізу [Текст] : конспект лекцій / О. І. Удодова, Ю. С. Шувалова, Н. С. Юрчак. – Харків : УкрДУЗТ, 2015. – 50 с.

2 Лінійне програмування [Текст] : конспект лекцій з дисципліни «Оптимізаційні методи і моделі» / Ю. О. Акімова, О. О. Думіна, О. І. Удодова, Ю. С. Шувалова. – Харків : УкрДУЗТ, 2013. – 44 с.

3 Теорія графів у задачах розподілу ресурсів. Кн. 1: Алгоритми та методи обчислень [Текст] : підручн. для студ. техн. спец. вищих навчальних закладів / Ф. О. Демченко, С. В. Лістровий, М. І. Луханін, Р. В. Семчук. – Харків : Нове слово, 2008. – 120 с.

4 Теорія графів у задачах розподілу ресурсів. Кн. 2. Диференціально-ігровий підхід до моделювання систем [Текст] : підруч. для студ. техн. спец. вищих навчальних закладів / С. В. Лістровий, М. І. Луханін, О.П. Мартинова, Р.В. Семчук. – Харків : Нове слово, 2007. – 144 с.

5 Гончарова, Г. А. Элементы дискретной математики [Текст] : учеб. пособие / Г. А. Гончарова, А. А. Мочалин. – М. : Форум: Инфра-М, 2007. – 128 с.

6 Основи дискретної математики [Текст] : підручник / Ю. В. Капітонова, С. Л. Кривий, О. А. Летичевський, Г. М. Луцький, М. К. Печорін. – К. : Наукова думка, 2002. – 579 с.

7 Кирсанов, М. Н. Графы в Maple [Текст] / М. Н. Кирсанов. – М. :ФИЗМАТЛИТ, 2007 – 168 с.

8 Новиков, Ф. А. Дискретная математика для программистов [Текст] : учеб. для вузов / Ф. А. Новиков. – 2-е изд. – СПб. : Питер, 2006. – 364 с.