

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Кафедра систем штучного інтелекту

Звіт

Розрахункова робота
з дисципліни «Дискретна математика»

Варіант № 19

Виконав:
Студент групи КН-113
Іванюшенко Нестор
Викладач:
Мельникова Н. І.

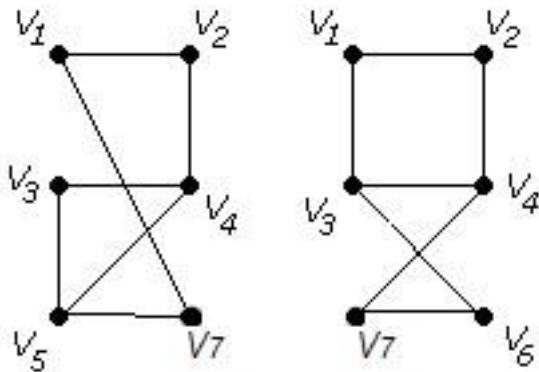
Львів-2019р.

Індивідуальні завдання

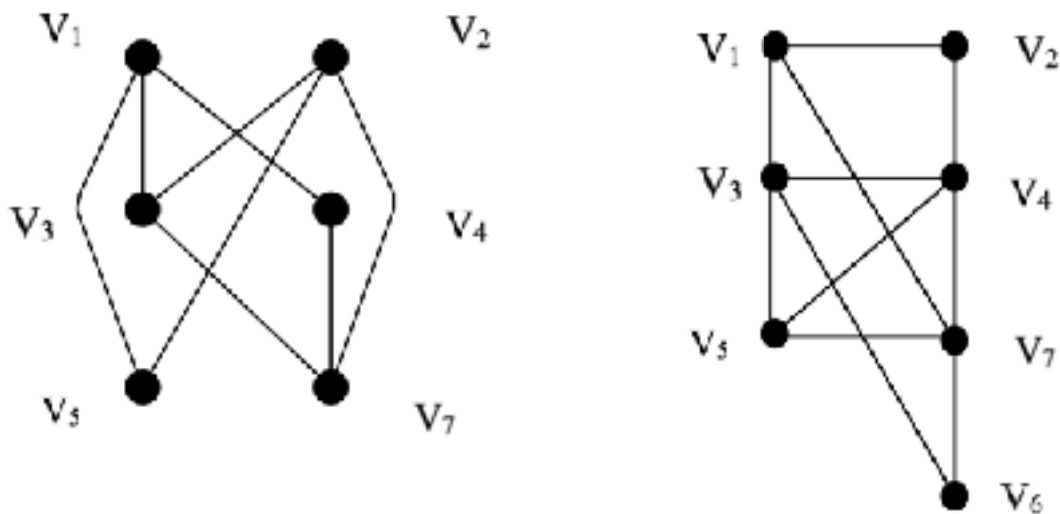
Завдання № 1.

Виконати наступні операції над графами

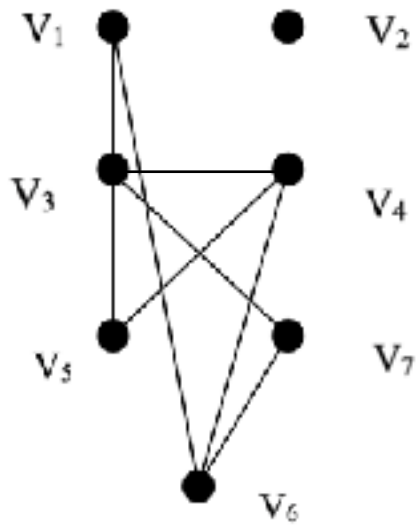
- 1) знайти доповнення до першого графу;
- 2) об'єднання графів;
- 3) кільцеву сумму G_1 та G_2 (G_1+G_2);
- 4) розмножити вершину у другому графі;
- 5) виділити підграф A - що складається з 3-х вершин в G_1 ;
- 6) добуток графів.



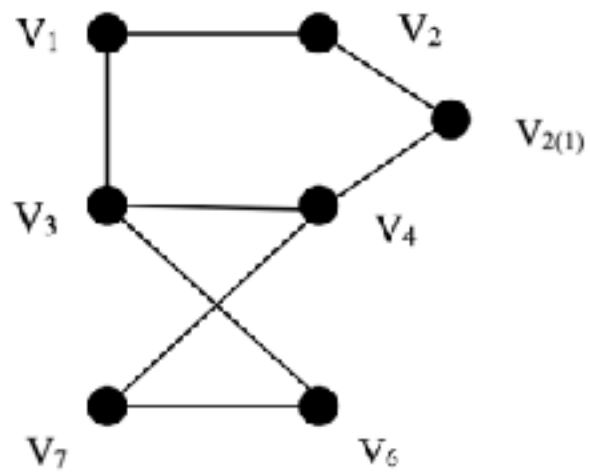
- 1) Доповнення до першого графу: 2) Об'єднання графів:



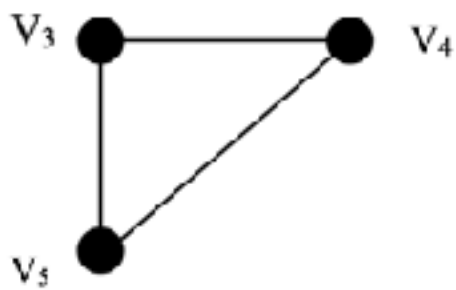
3) Кільцева сума:



4) Розмноження вершини у другому графі:



5) Виділити підграф A, що складається з 3 вершин в першому підграфі:



6) Добуток графів:

Добуток графів матиме 36 вершин, оскільки кожен із графів містить по 6 вершин.

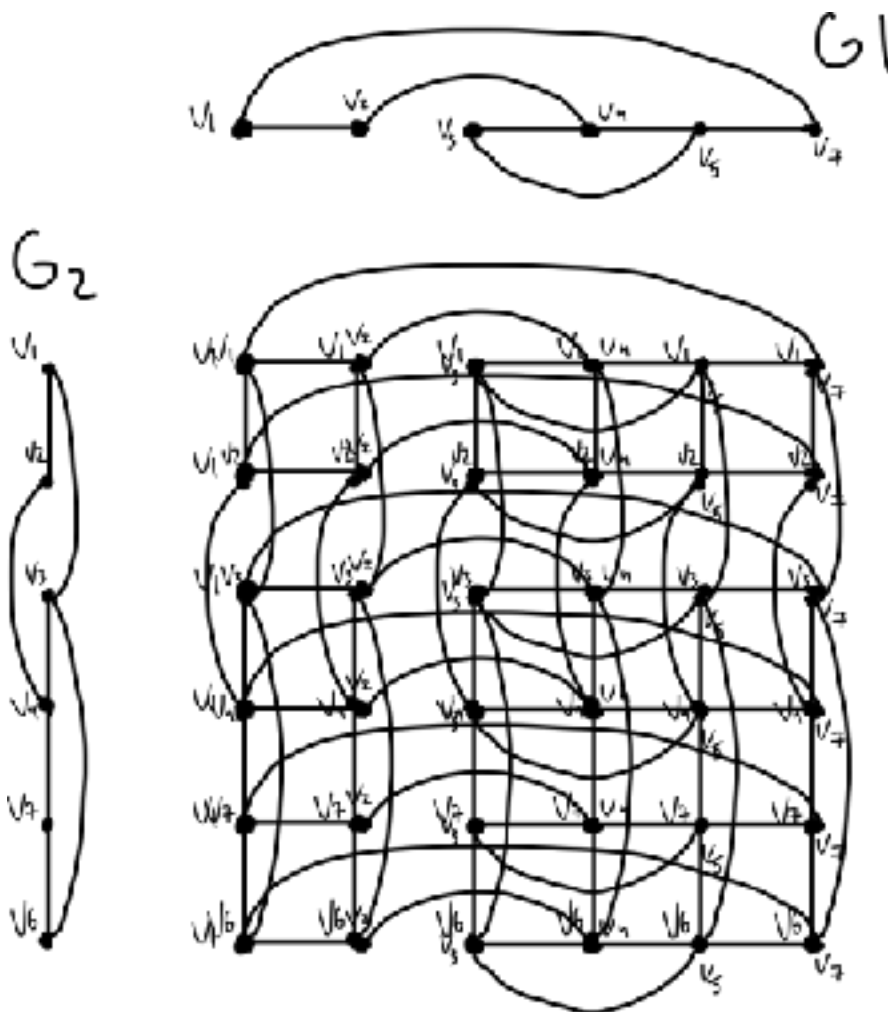
$$G = G_1 \times G_2$$

$$V(G) = V(G_1) \times V(G_2)$$

$$E(G) = E(G_1 \times G_2)$$

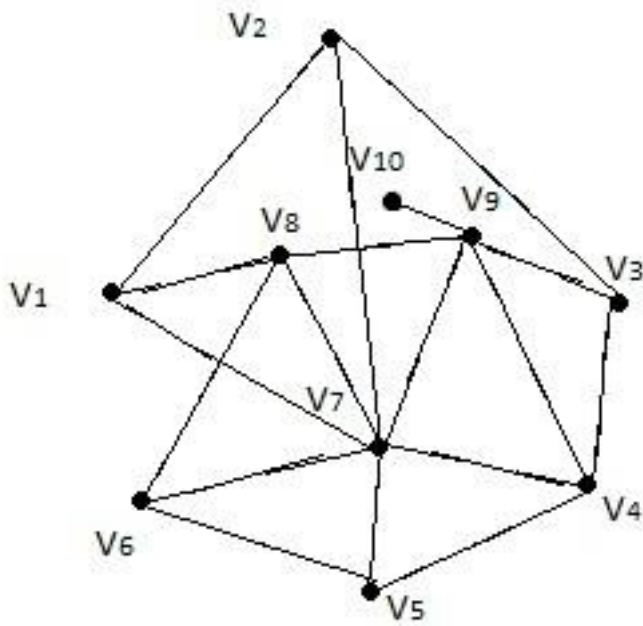
$$V(G) = \{ (1, 1), (1, 2), (1, 3), (1, 4), (1, 7), (1, 6), (2, 1), (2, 2), (2, 3), (2, 4), (2, 7), (2, 6), (3, 1), (3, 2), (3, 3), (3, 4), (3, 7), (3, 6), (4, 1), (4, 2), (4, 3), (4, 4), (4, 7), (4, 6), (5, 1), (5, 2), (5, 3), (5, 4), (5, 7), (5, 6), (6, 1), (6, 2), (6, 3), (6, 4), (6, 7), (6, 6), (7, 1), (7, 2), (7, 3), (7, 4), (7, 7), (7, 6) \}$$

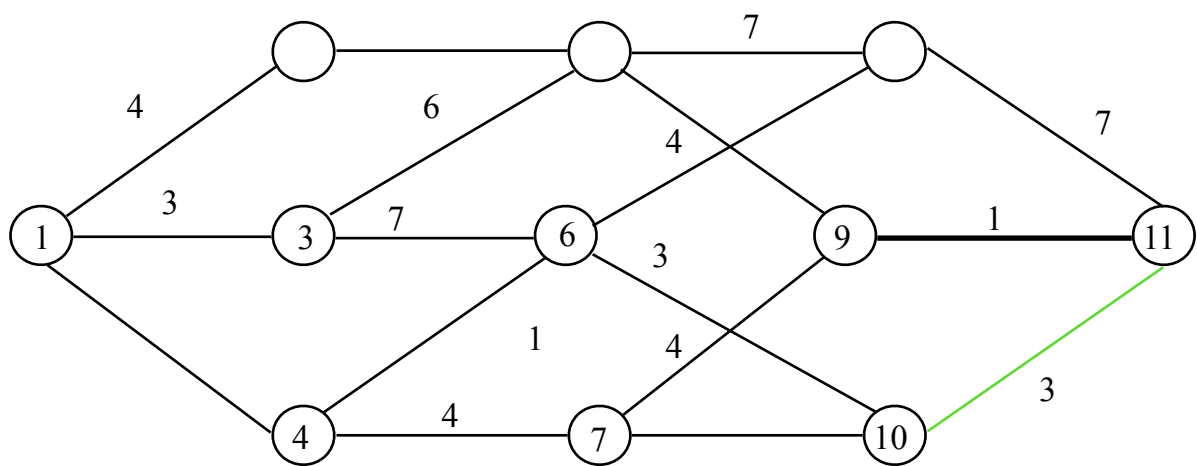
Отже, добуток графів буде виглядати так:



Завдання № 2.

Скласти таблицю суміжності для орграфа.

[illegible]



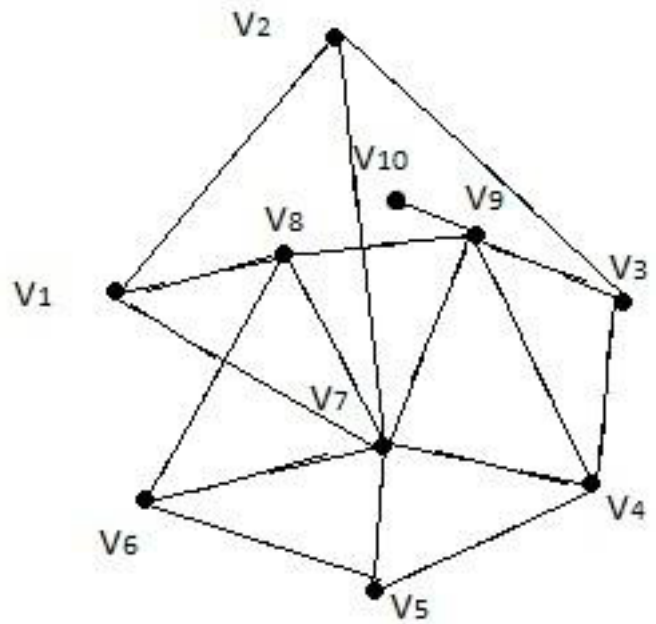
Завдання № 3.

Для графа з другого завдання знайти діаметр.

Діаметр даного графа дорівнює 3, оскільки цьому дорівнює максимальна відстань найкоротшого шлях між двома вершинами (у цьому випадку V_6 і V_{10}).

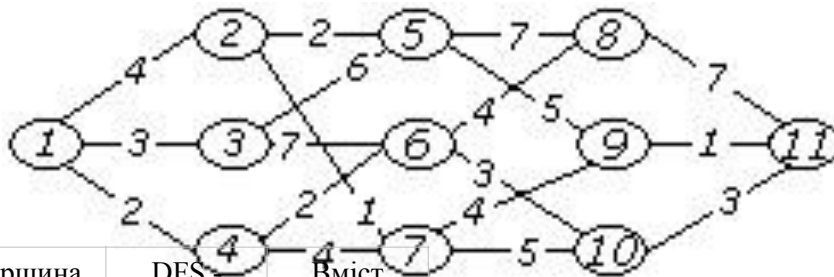
Завдання № 4.

Для графа з другого завдання виконати обхід дерева вглиб.



Завдання № 5.

Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.



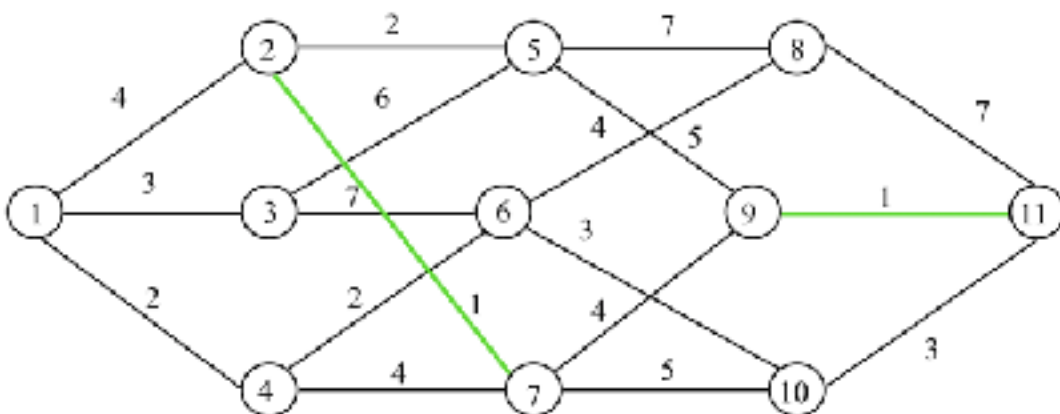
Вершина	DFS номер	Вміст стеку
1	1	1
2	2	12
3	3	123
9	4	1239
10	5	123910
-	-	1239
8	6	12398
7	7	123987
6	8	1239876
5	9	12398765
4	10	123987654
-	-	12398765
-	-	1239876
-	-	123987
-	-	12398
-	-	1239
-	-	123
-	-	12
-	-	1
-	-	∅

- **Метод Краскала:**

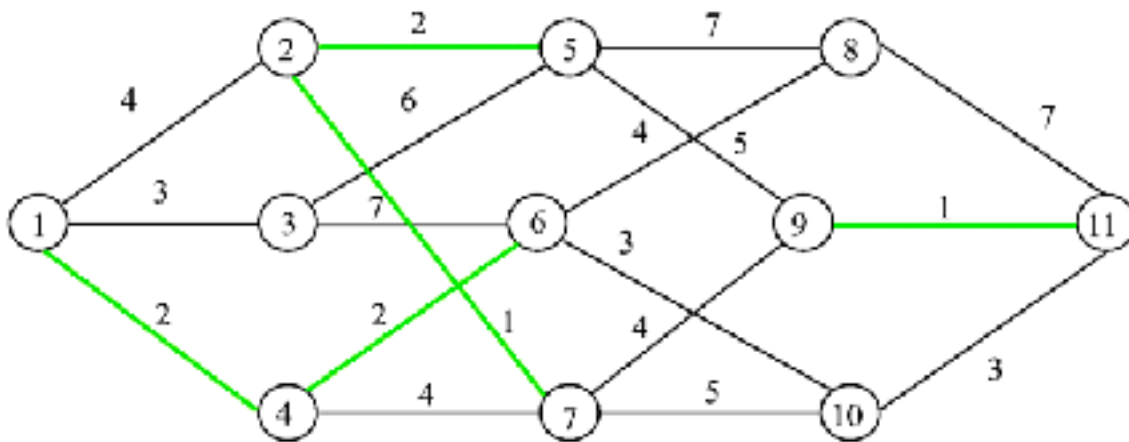
сортуємо для даного графа ребра у порядку неспадання ваг цих ребер.

(2, 7) - 1;
(9, 11) - 1;
(1, 4) - 2;
(2, 5) - 2;
(4, 6) - 2;
(1, 3) - 3;
(6, 10) - 3;
(10, 11) - 3;
(1, 2) - 4;
(4, 7) - 4;
(6, 8) - 4;
(7, 9) - 4;
(5, 9) - 5;
(7, 10) - 5;
(3, 5) - 6;
(5, 8) - 7;
(8, 11) - 7;

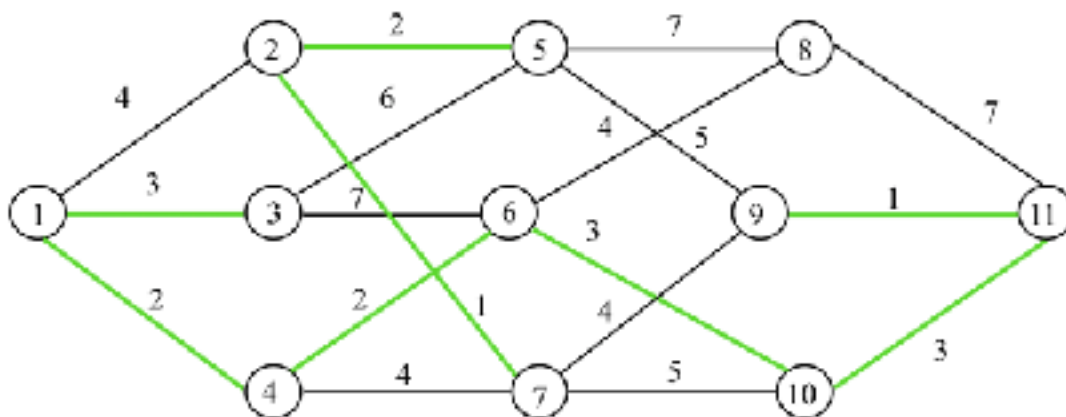
вибираємо ребро графа з найменшою вагою і додаємо до дерева. В даному випадку це ребра (2, 7) і (9, 11) з вагою 1.



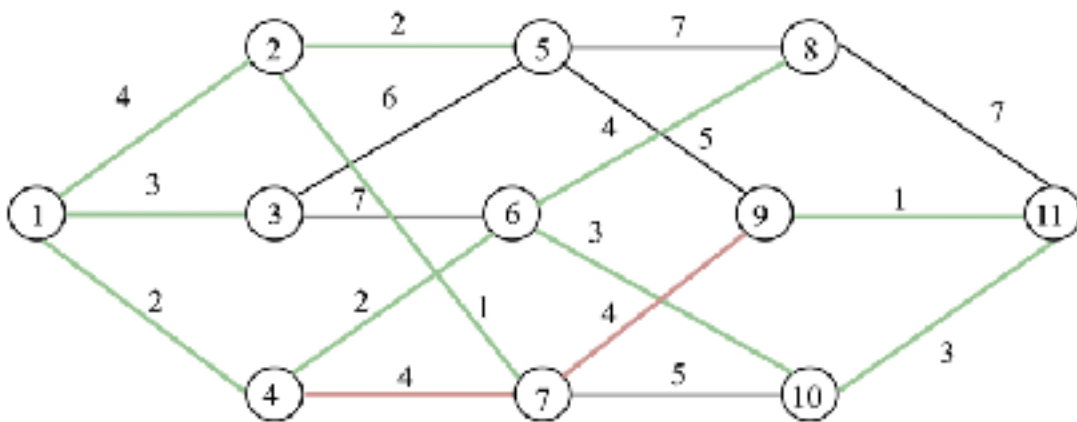
беремо ребро з наступною найменшою вагою. Це ребра (4, 6), (1, 4), (2, 5) з вагою 2.
Додаємо до дерева.



беремо ребра з наступною найменшою вагою - (1, 3), (6, 10), (10, 11) з вагою 3.
Додаємо до кістякового дерева.

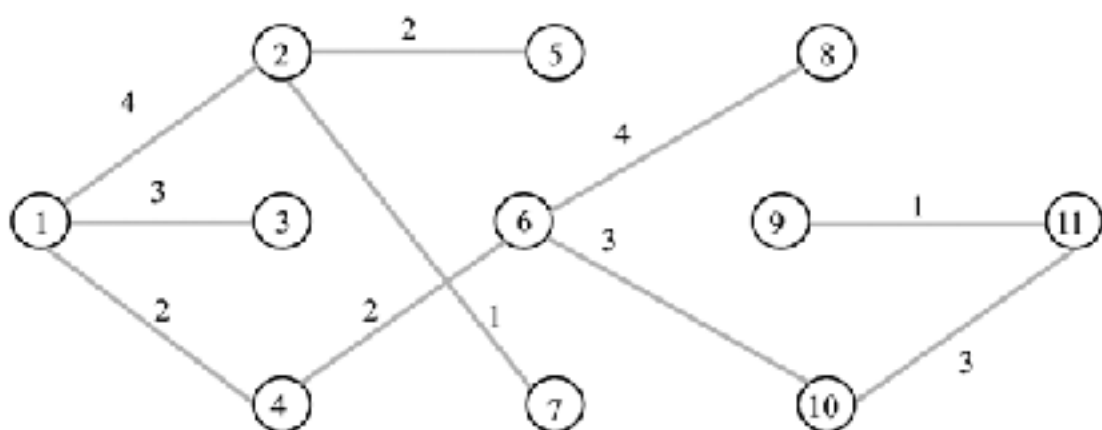


беремо ребра з наступною найменшою вагою - (1, 2), (4, 7), (6, 8), (7, 9) з вагою 4.
Відкидаємо ребра (4, 7) і (7, 9), оскільки вони утворять цикл.

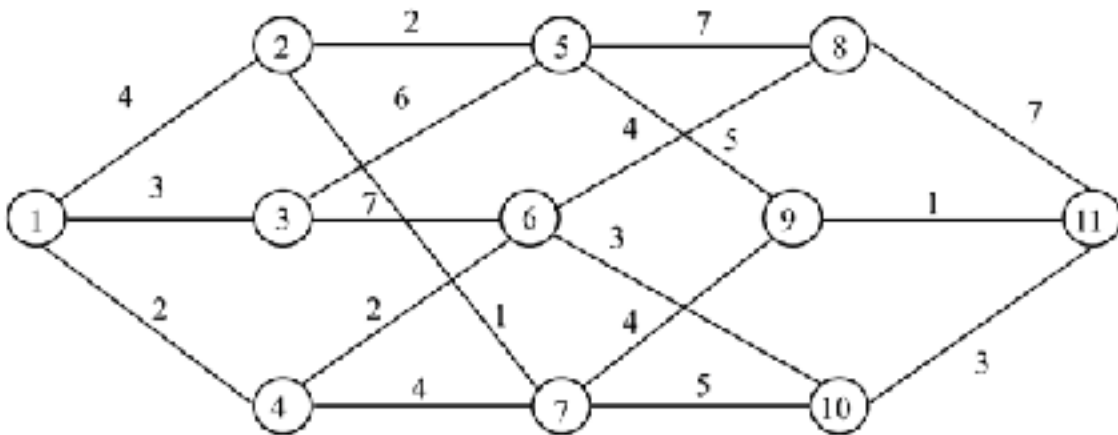


продовжуємо виконувати алгоритм за аналогією і отримуємо дерево після того як перевірили всі ребра із посортованого списку або ж всі вершини стали з'єднаними

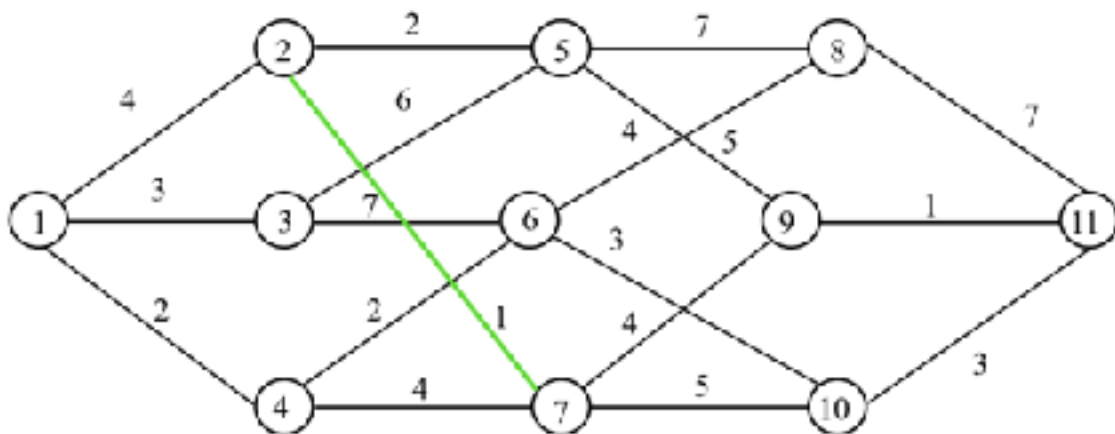
мінімальне остове дерево за алгоритмом Красскала буде виглядати так:



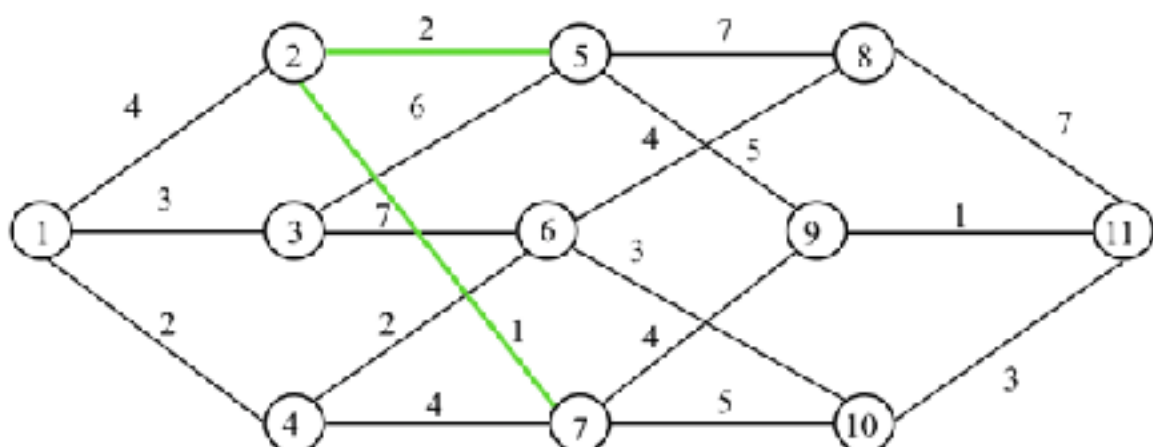
- **Метод Прима:**



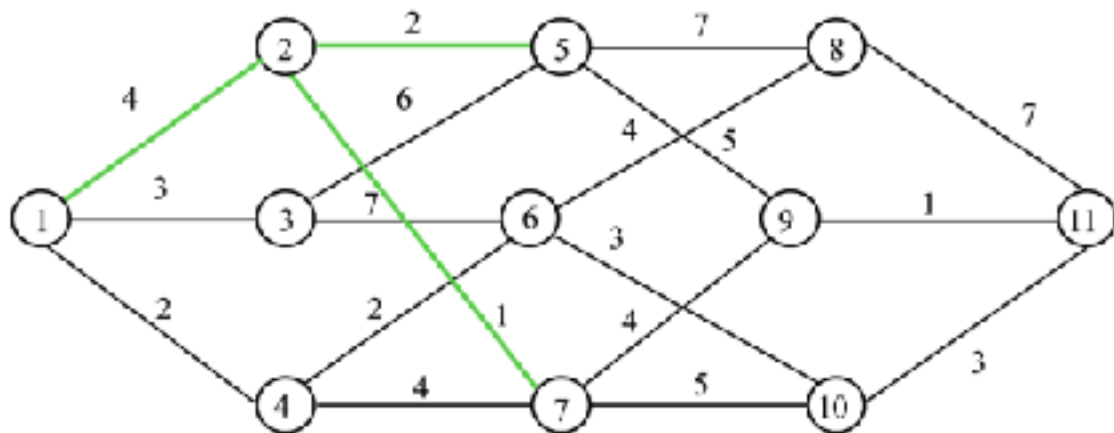
спочатку фіксуємо довільну вершину, з якої розпочнемо пошук мінімального дерева. У даному випадку це вершина 2. Знаходимо ребро, інцидентне цій вершині з найменшою вагою і це (2, 7) з вагою 1. Додаємо до кістякового дерева ребро (2, 7).



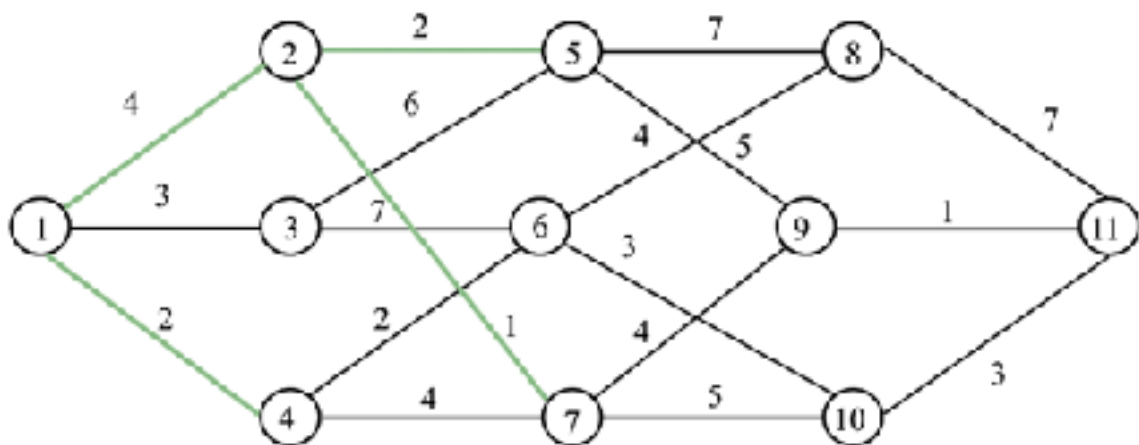
розглядаємо ребра, інцидентні вершинам 2 і 7, і вибираємо те, яке має найменшу вагу. І це (2, 5) з вагою 2. Додаємо його до кістякового дерева.



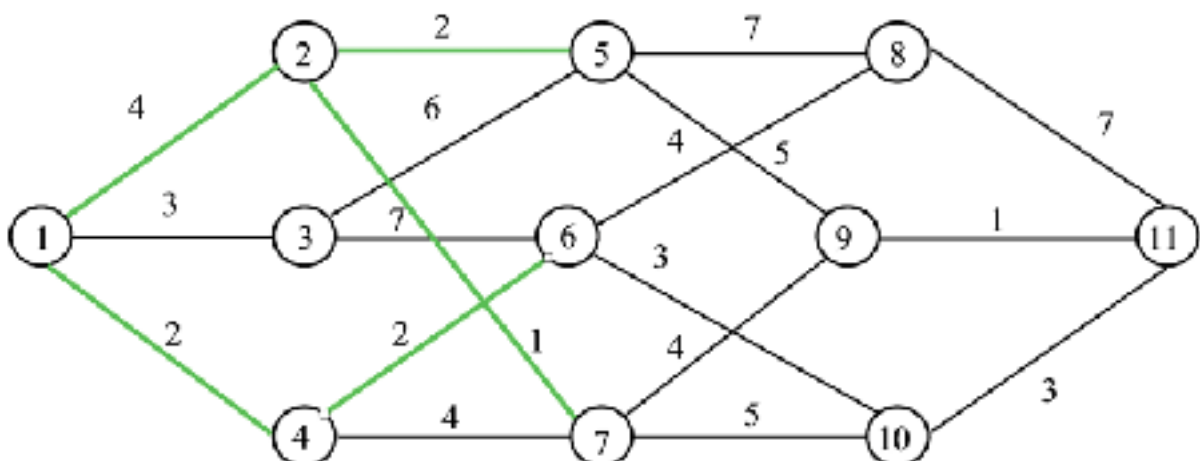
розглядаємо ребра, інцидентні вершинам 2, 7, 5 і вибираємо з найменшою вагою - і це (1, 2) з вагою 4. Додаємо до дерева.



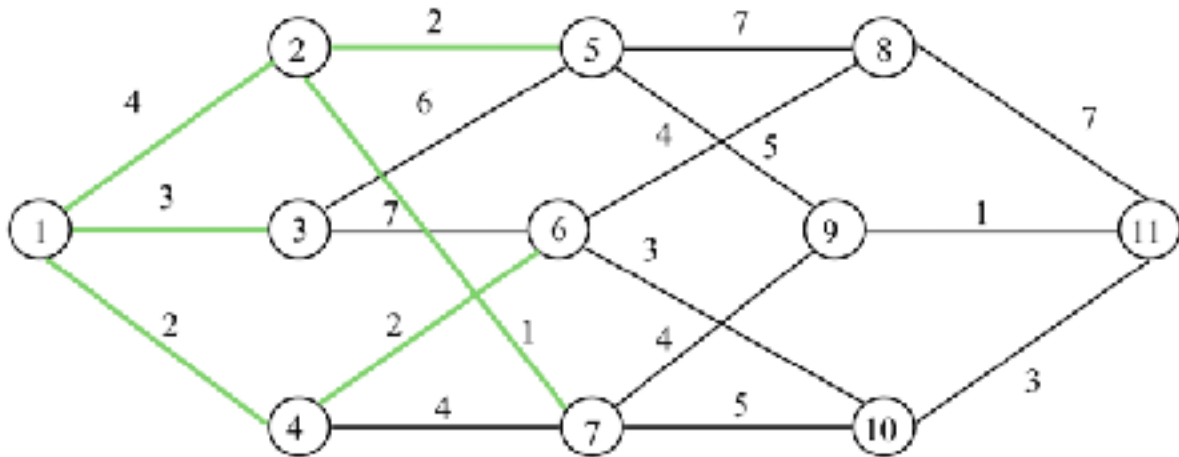
розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1 і вибираємо з найменшою вагою - і це (1, 4) з вагою 2. Додаємо до дерева.



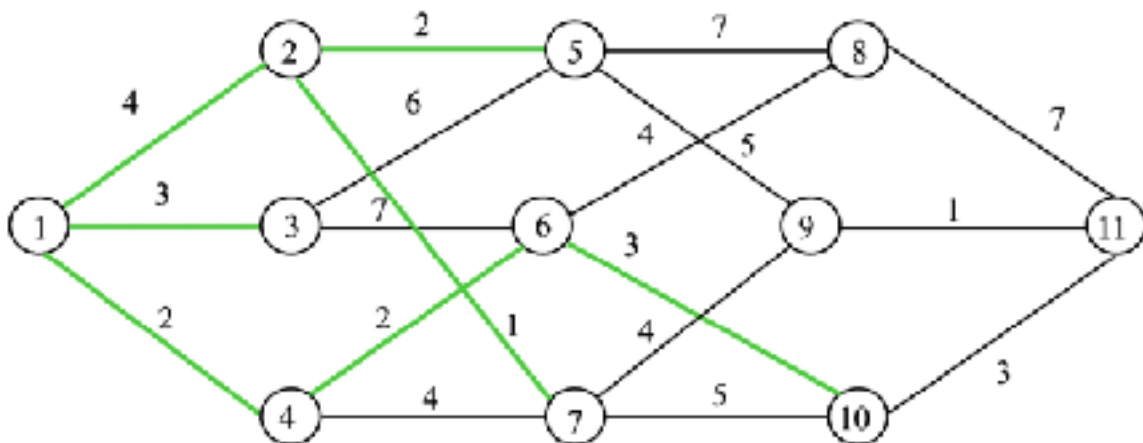
розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1, 4 і вибираємо з найменшою вагою - і це (4, 6) з вагою 2. Додаємо до кістякового дерева.



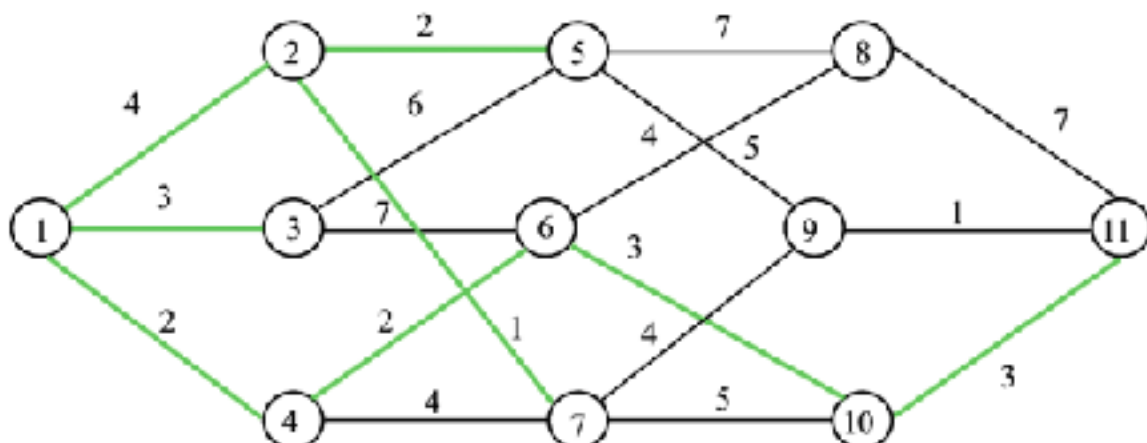
Етап 6: розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1, 4, 6 і вибираємо з найменшою вагою - і це (1, 3) з вагою 3. Додаємо до кістякового дерева.

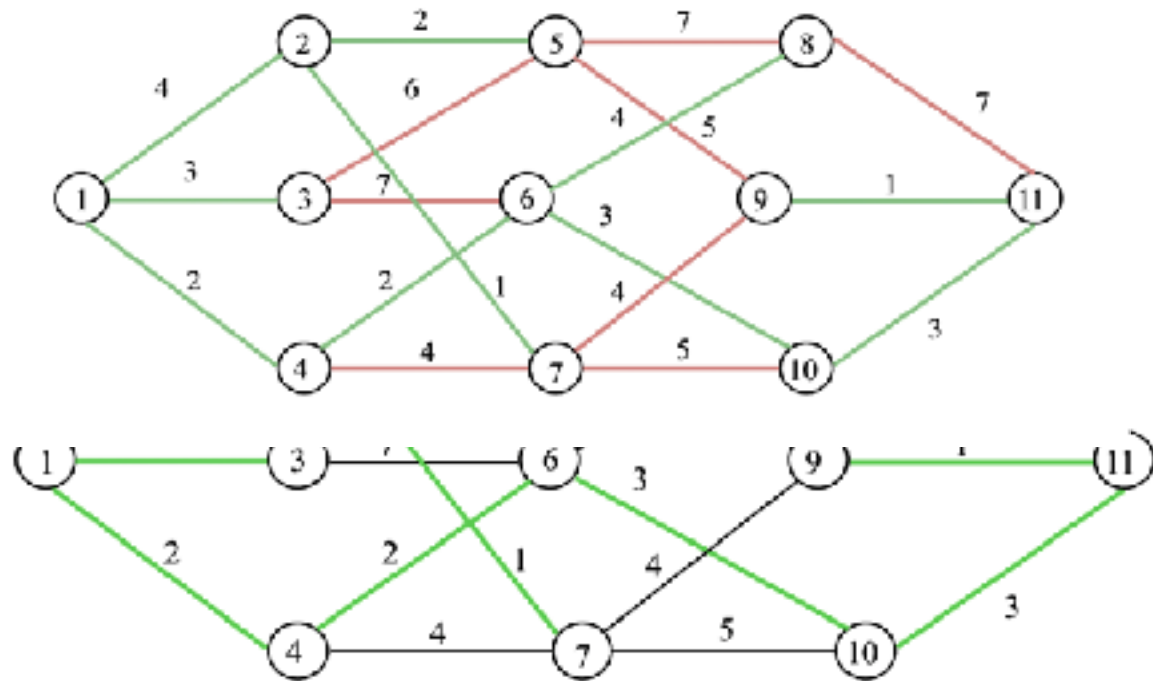


Етап 7: розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1, 4, 6, 3 і вибираємо з найменшою вагою - і це (6, 10) з вагою 3. Додаємо до кістякового дерева.



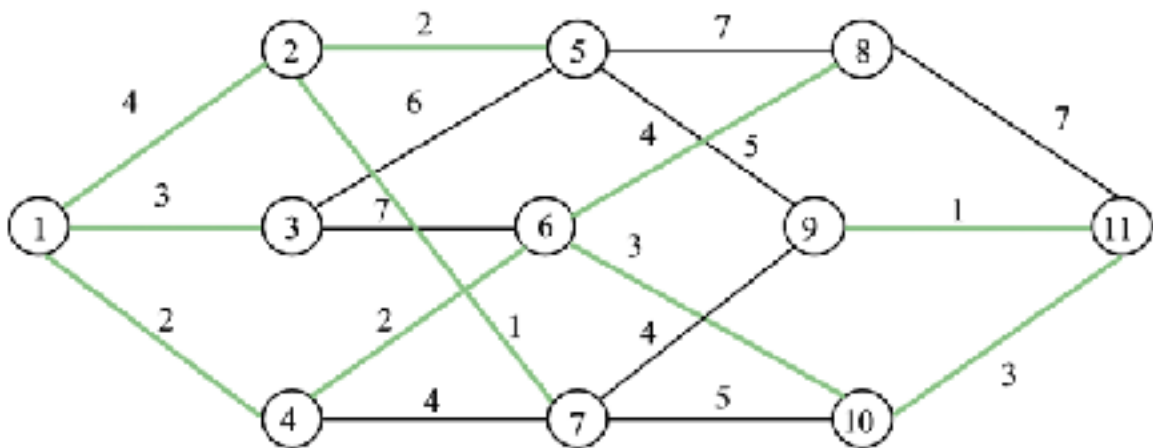
Етап 8: розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1, 4, 6, 3, 10 і вибираємо з найменшою вагою - і це (10, 11) з вагою 3. Додаємо до кістякового дерева.





розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1, 4, 6, 3, 10, 11 і вибираємо з найменшою вагою - і це (9, 11) з вагою 1. Додаємо до дерева.

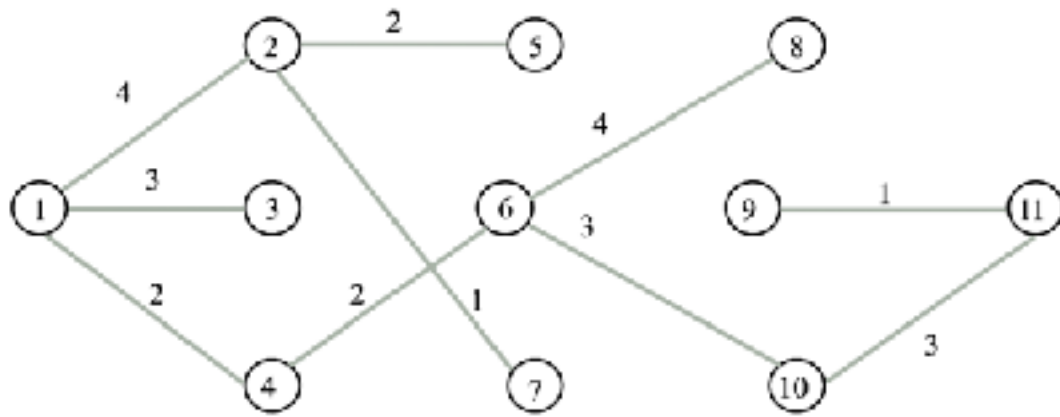
Етап 10: розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1, 4, 6, 3, 10, 11 і вибираємо



з найменшою вагою - і це (6, 8) з вагою 4. Додаємо до дерева.

бачимо що всі вершини з'єднанні і дерево утворене.

Отже, мінімальне остове дерево за алгоритмом Прима виглядає так:



Завдання № 6.

Розв'язати задачу комівояжера для повного 8-ми вершин-ного графа методом «іди у найближчий», матриця вагів якого має вигляд:

19)

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

вибираємо з першого рядка мінімальне значення 2. Викреслюємо 1 рядок, 1 стовпець.

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

вибираємо з восьмого рядка мінімальне значення 1. Викреслюємо 8 рядок, 8 стовпець.

	2	3	4	5	6	7
2	∞	5	1	2	3	2
3	5	∞	6	6	5	1
4	1	6	∞	6	6	6
5	2	6	6	∞	5	1
6	3	5	6	5	∞	2
7	2	1	6	1	2	∞

вибираємо з 6 рядка значення 5. Викреслюємо 6 рядок, 6 стовпець.

	2	3	4	5	7
2	∞	5	1	2	2
3	5	∞	6	6	1
4	1	6	∞	6	6
5	2	6	6	∞	1
7	2	1	6	1	∞

вибираємо з 3 рядка значення 1. Викреслюємо 3 рядок, 3 стовпець.

	2	4	5	7
2	∞	1	2	2
4	1	∞	6	6
5	2	6	∞	1
7	2	6	1	∞

вибираємо з 7 рядка значення 1. Викреслюємо 7 рядок, 7 стовпець.

	2	4	5
2	∞	1	2
4	1	∞	6
5	2	6	∞

вибираємо з 5 рядка значення 2. Викреслюємо 5 рядок, 5 стовпець.

	2	4
2	∞	1
4	1	∞

$1 \rightarrow 8 \rightarrow 6 \rightarrow 3 \rightarrow 7 \rightarrow 5 \rightarrow 2 \rightarrow 4 \rightarrow 1$

$$2 + 1 + 5 + 1 + 1 + 2 + 1 + 2 = 15$$

Це найкоротший маршрут між вершинами та мінімальна вага.

Проходимо через ще один цикл:

з першого рядка вибираємо найменше значення 2. Викреслюємо 1 рядок, 1 стовпець.

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

з 8 рядка вибираємо значення 1. Викреслюємо 8 рядок, 8 стовпець.

	2	3	4	5	6	7
2	∞	5	1	2	3	2
3	5	∞	6	6	5	1
4	1	6	∞	6	6	6
5	2	6	6	∞	5	1
6	3	5	6	5	∞	2
7	2	1	6	1	2	∞

з 6 рядка вибираємо 2. Викреслюємо 6 рядок, 6 стовпець.

	2	3	4	5	7
2	∞	5	1	2	2
3	5	∞	6	6	1
4	1	6	∞	6	6
5	2	6	6	∞	1
7	2	1	6	1	∞

з 7 рядка вибираємо 1. Викреслюємо 7 рядок, 7 стовпець.

	2	3	4	5
2	∞	5	1	2
3	5	∞	6	6
4	1	6	∞	6
5	2	6	6	∞

з 5 рядка вибираємо 2. Викреслюємо 5 рядок, 5 стовпець.

	2	3	4
2	∞	5	1
3	5	∞	6
4	1	6	∞

з 2 рядка вибираємо 1. Викреслюємо 2 рядок, 2 стовпець.

	3	4
3	∞	6
4	6	∞

$1 \rightarrow 8 \rightarrow 6 \rightarrow 7 \rightarrow 5 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$

$$2 + 1 + 2 + 1 + 2 + 1 + 6 + 2 = 17$$

за аналогією знаходимо такі цикли:

$1 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 7 \rightarrow 5 \rightarrow 1$ (18)

$1 \rightarrow 7 \rightarrow 5 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 6 \rightarrow 3 \rightarrow 1$ (20)

$3 \rightarrow 7 \rightarrow 5 \rightarrow 2 \rightarrow 4 \rightarrow 1 \rightarrow 8 \rightarrow 6 \rightarrow 3$ (15)

$1 \rightarrow 5 \rightarrow 7 \rightarrow 3 \rightarrow 6 \rightarrow 8 \rightarrow 2 \rightarrow 4 \rightarrow 1$ (17)

$1 \rightarrow 7 \rightarrow 3 \rightarrow 6 \rightarrow 8 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 1$ (22)

$2 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 7 \rightarrow 5 \rightarrow 6 \rightarrow 8 \rightarrow 2$ (17)

$2 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 7 \rightarrow 5 \rightarrow 8 \rightarrow 6 \rightarrow 2$ (16)

$8 \rightarrow 6 \rightarrow 7 \rightarrow 3 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 8$ (16)

$8 \rightarrow 6 \rightarrow 7 \rightarrow 3 \rightarrow 1 \rightarrow 5 \rightarrow 2 \rightarrow 4 \rightarrow 8$ (17)

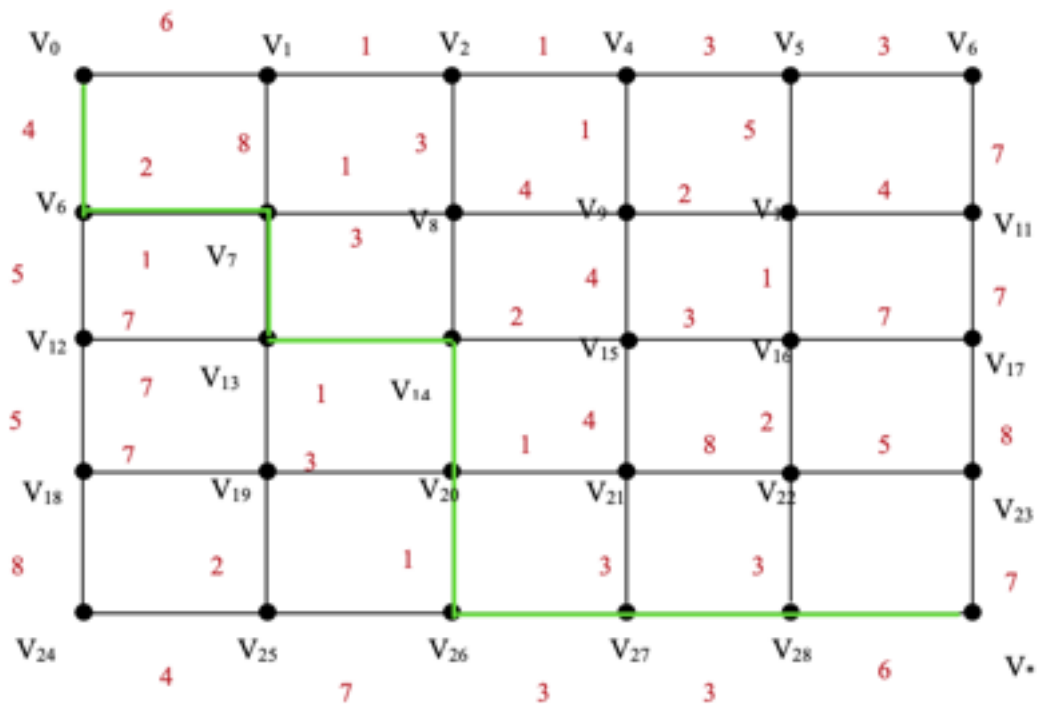
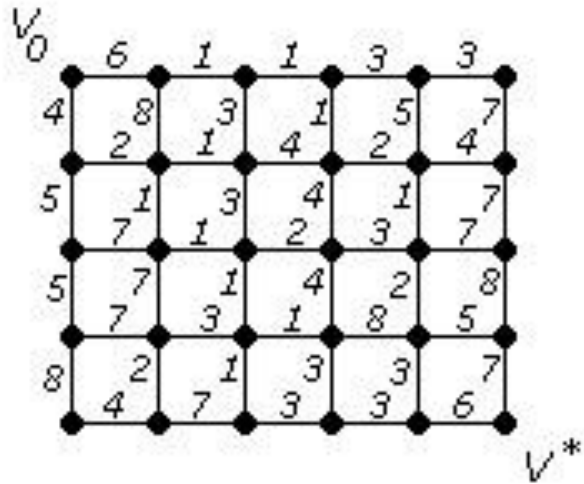
$6 \rightarrow 8 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 3 \rightarrow 6$ (15)

$7 \rightarrow 5 \rightarrow 2 \rightarrow 4 \rightarrow 1 \rightarrow 8 \rightarrow 6 \rightarrow 3 \rightarrow 7$ (15)

Отже, вага найкоротшого маршруту дорівнює 15.

Завдання №7.

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V_0 і V^* .



Виконання:

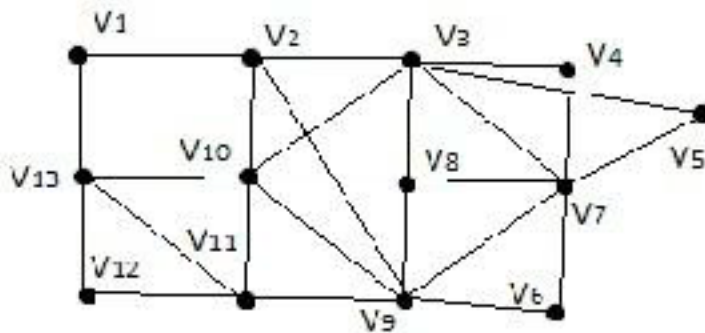
- 1) Розпочинаємо з початкової вершини, яку будемо вважати поточною. В даний момент це вершина V_0 . Переглядаємо її шлях до суміжних вершин - V_1 і V_6 . До першої вершини відстань дорівнює 6, до шостої - 4.
- 2) Проходимо через $V_0 \rightarrow V_1 \rightarrow V_2$, відстань дорівнює 7.
- 3) Проходимо через $V_1 \rightarrow V_2 \rightarrow V_3$, відстань дорівнює 7
- 4) Проходимо через $V_2 \rightarrow V_3 \rightarrow V_4$, відстань дорівнює 8.
- 5) Проходимо через $V_3 \rightarrow V_4 \rightarrow V_5$, відстань дорівнює 11.
- 6) Проходимо через $V_4 \rightarrow V_5 \rightarrow V_{11}$, відстань дорівнює 14.
- 7) Проходимо через $V_1 \rightarrow V_7$, відстань дорівнює 14.
- 8) Проходимо через $V_2 \rightarrow V_8$, відстань дорівнює 10.
- 9) Проходимо через $V_3 \rightarrow V_9$, відстань дорівнює 9.
- 10) Проходимо через $V_4 \rightarrow V_{10}$, відстань дорівнює 16.
- 11) Проходимо через $V_6 \rightarrow V_7$, відстань дорівнює 6, що є меншою за минулу відстань від вершини V_1 , то викреслюємо минулу і робимо отриману відстань поточною.
- 12) Проходимо через $V_7 \rightarrow V_8$, відстань дорівнює 7, що є меншою за минулу відстань від вершини V_2 , то викреслюємо минулу і робимо отриману відстань поточною.
- 13) Проходимо через $V_8 \rightarrow V_9$, відстань дорівнює 11, тому залишаємо минулу відстань без змін.
- 14) Проходимо через $V_9 \rightarrow V_{10}$, відстань дорівнює 11, що є меншою за минулу відстань від вершини V_4 , то викреслюємо минулу і робимо отриману відстань поточною.
- 15) Проходимо через $V_{10} \rightarrow V_{11}$, відстань дорівнює 15, що є меншою за минулу відстань від вершини V_5 , то викреслюємо минулу і робимо отриману відстань поточною.
- 16) Проходимо через $V_{12} \rightarrow V_{13}$, відстань дорівнює 16.
- 17) Проходимо через $V_{13} \rightarrow V_{14}$, відстань дорівнює 17, тому залишаємо минулу відстань без змін.
- 18) Проходимо через $V_{14} \rightarrow V_{15}$, відстань дорівнює 15, тому залишаємо минулу відстань без змін.
- 19) Проходимо через $V_{15} \rightarrow V_{16}$, відстань дорівнює 16, що є меншою за минулу відстань від вершини V_{10} , то викреслюємо минулу і робимо отриману відстань поточною.
- 20) Проходимо через $V_{16} \rightarrow V_{17}$, відстань дорівнює 23, тому залишаємо минулу відстань без змін.
- 21) Проходимо через $V_{18} \rightarrow V_{19}$, відстань дорівнює 21, тому залишаємо минулу відстань без змін.
- 22) Проходимо через $V_{19} \rightarrow V_{20}$, відстань дорівнює 17, що є меншою за минулу відстань від вершини V_{13} , то викреслюємо минулу і робимо отриману відстань поточною.
- 23) Проходимо через $V_{20} \rightarrow V_{21}$, відстань дорівнює 15, що є меншою за минулу відстань від вершини V_{15} , то викреслюємо минулу і робимо отриману відстань поточною.

- 24) Проходимо через $V_{21} \rightarrow V_{22}$, відстань дорівнює 23, тому залишаємо минулу відстань без змін.
- 25) Проходимо через $V_{22} \rightarrow V_{23}$, відстань дорівнює 24, тому залишаємо минулу відстань без змін.
- 26) Проходимо через $V_{24} \rightarrow V_{25}$, відстань дорівнює 26, тому залишаємо минулу відстань без змін.
- 27) Проходимо через $V_{25} \rightarrow V_{26}$, відстань дорівнює 32, тому залишаємо минулу відстань без змін.
- 28) Проходимо через $V_{26} \rightarrow V_{27}$ відстань дорівнює 18, що є меншою за минулу відстань від вершини V_{21} , то викреслюємо минулу і робимо отриману відстань поточною.
- 29) Проходимо через $V_{27} \rightarrow V_{28}$ відстань дорівнює 21, що є меншою за минулу відстань від вершини V_{22} , то викреслюємо минулу і робимо отриману відстань поточною.
- 30) Проходимо через $V_{28} \rightarrow V^*$ відстань дорівнює 22.

Отже, у даному графі за алгоритмом Дейкстри існує один мінімальний шлях від вершини V_0 до V^* , який дорівнює 22.

Завдання №8.

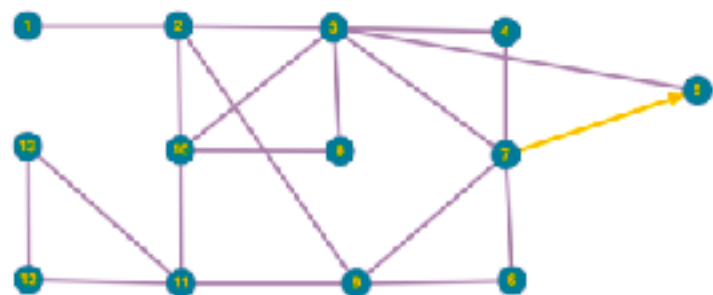
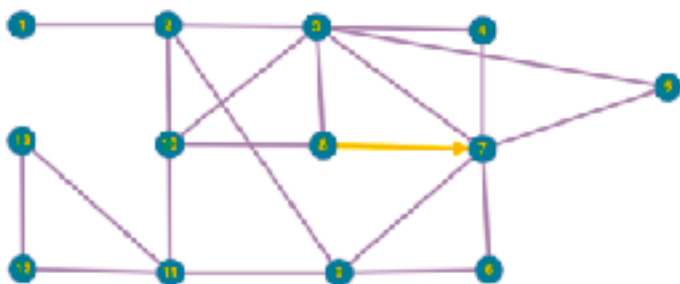
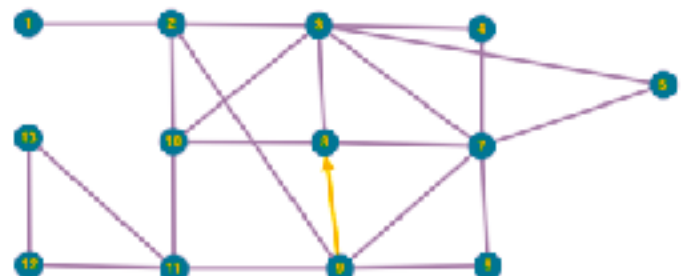
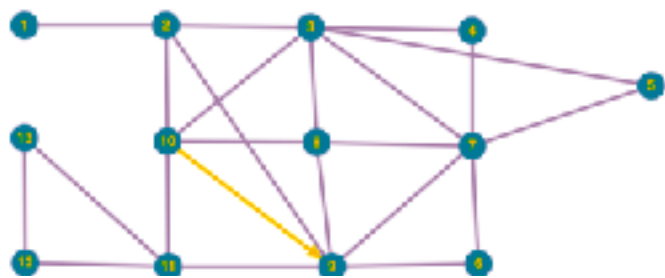
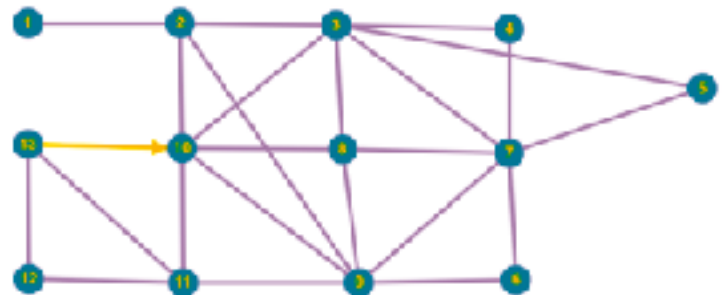
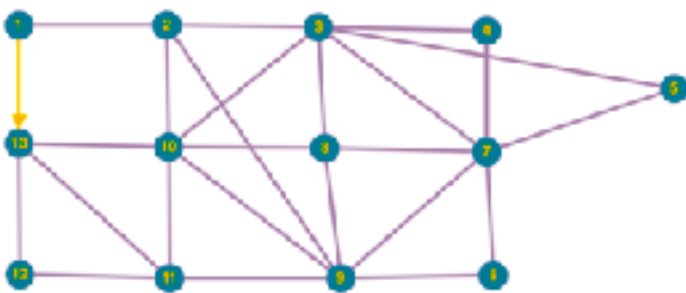
Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.

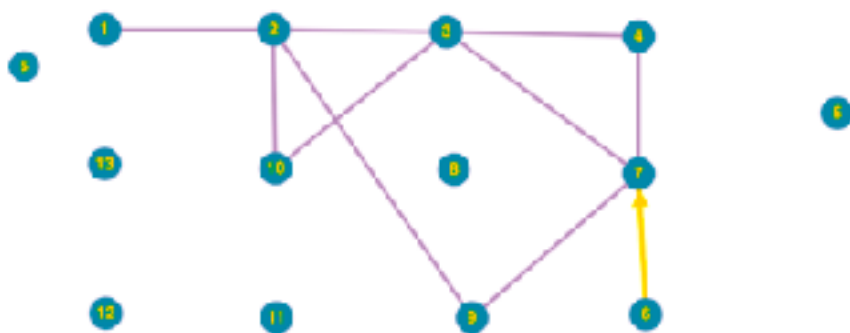
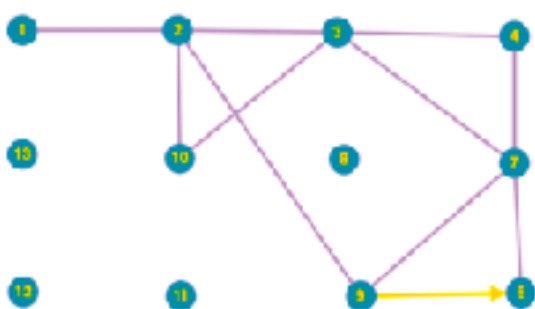
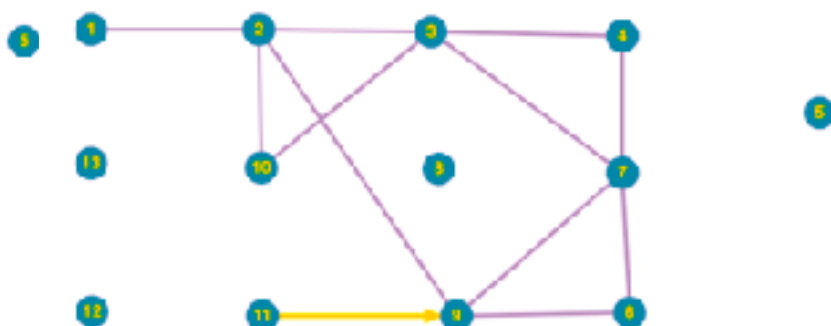
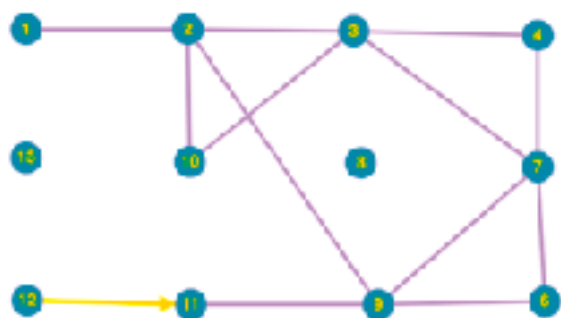
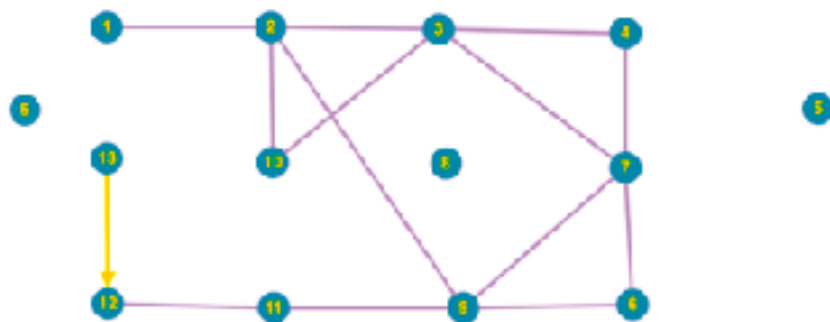
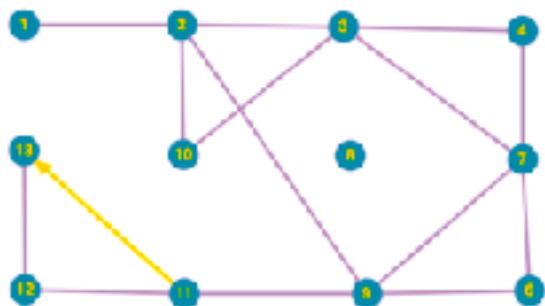
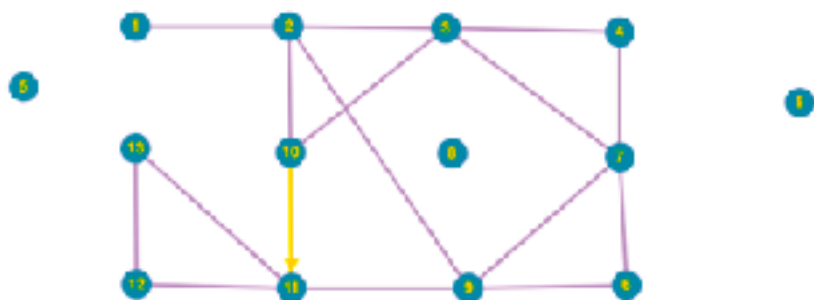
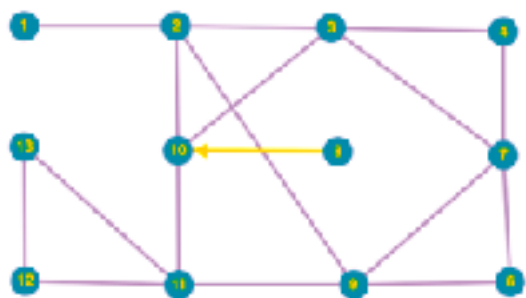


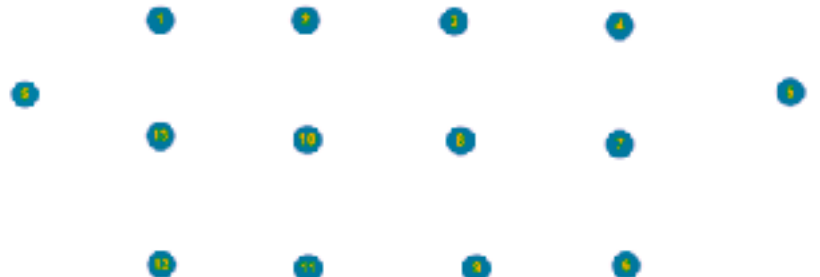
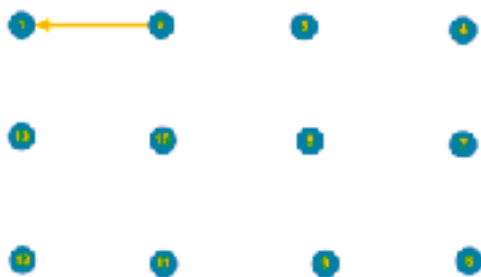
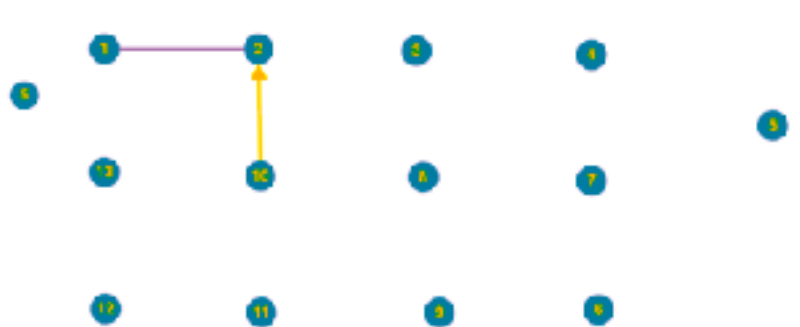
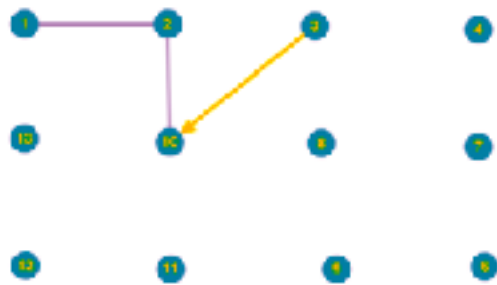
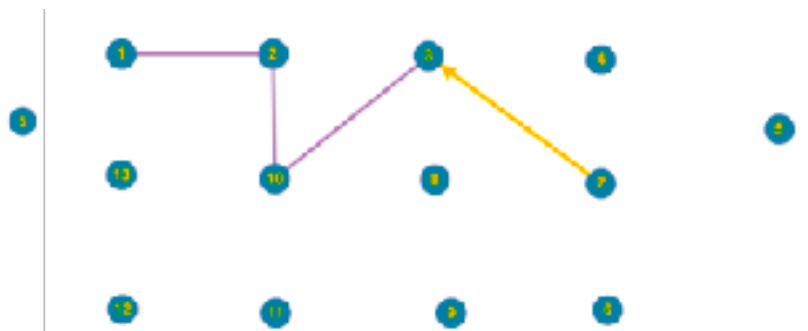
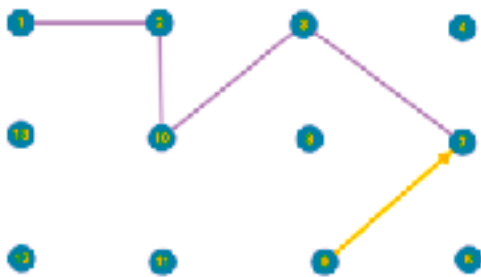
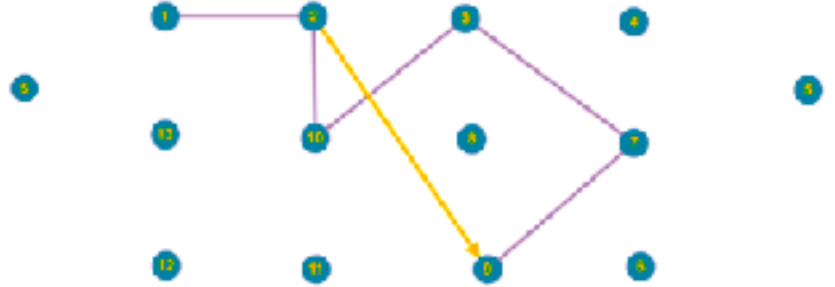
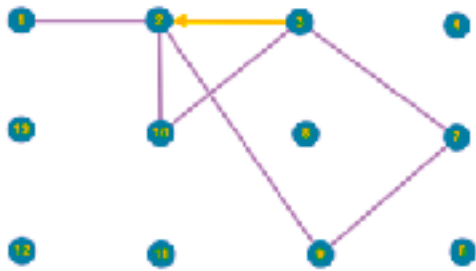
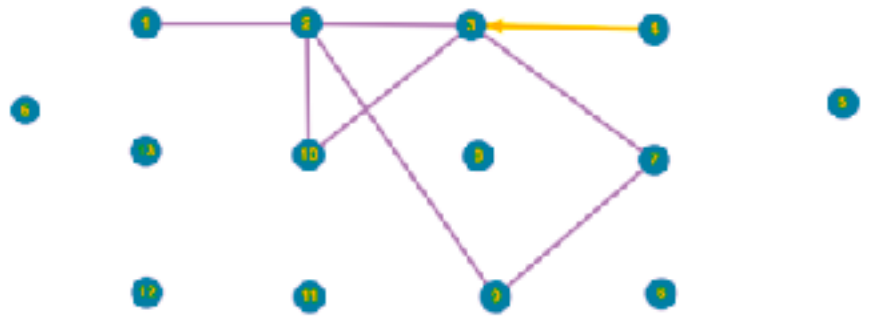
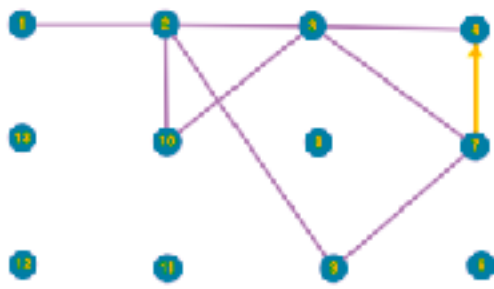
а) метод Флері:

Алгоритм Флері полягає у тому, щоб починаючи з деякої вершини, викреслювати кожне пройдене

ребро. Не проходимо по ребру, якщо видалення цього ребра призводить до розбиття графа на дві зв'язні компоненти, тобто, необхідно перевіряти, чи є ребро мостом чи ні.







Отже, ейлеровий цикл даного графа: $1 \rightarrow 13 \rightarrow 10 \rightarrow 9 \rightarrow 8 \rightarrow 7 \rightarrow 5 \rightarrow 3 \rightarrow 8 \rightarrow 10 \rightarrow 11 \rightarrow 13 \rightarrow 12 \rightarrow 11 \rightarrow 9 \rightarrow 6 \rightarrow 7 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 9 \rightarrow 7 \rightarrow 2 \rightarrow 10 \rightarrow 2 \rightarrow 1$.

б) метод елементарних циклів:

Виділимо кольоровим елементарні цикли та об'єднаємо їх у один:

$V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_7 \rightarrow V_6 \rightarrow V_9 \rightarrow V_{11} \rightarrow V_{12} \rightarrow V_{13}$

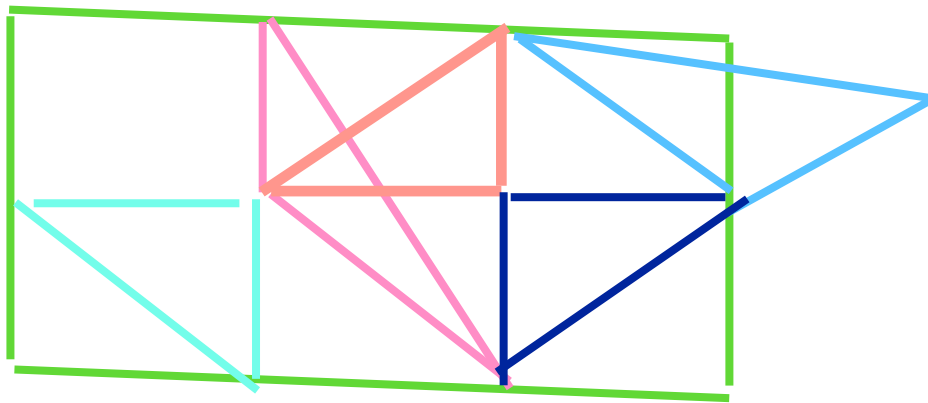
$V_2 \rightarrow V_9 \rightarrow V_{10}$

$V_3 \rightarrow V_5 \rightarrow V_7$

$V_3 \rightarrow V_8 \rightarrow V_{10}$

$V_7 \rightarrow V_8 \rightarrow V_9$

$V_{13} \rightarrow V_{11} \rightarrow V_{10}$



Завдання №9.

Спростити формули (привести їх до скороченої ДНФ).

$$\overline{xy(x\bar{y}z \vee \bar{x}y)}$$

спочатку запишем таблицю істинності для даної формули:

x	y	z	$\neg(\neg(x \wedge y) \wedge (x \wedge \neg y \wedge z \vee \neg x \wedge y))$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	1	1

запишемо кон'юнкцію змінних за яких формула дає 1:

$$\neg x \neg y \neg z$$

$$\neg x \neg y z$$

$$x \neg y \neg z$$

$$xy \neg z$$

$$xyz$$

об'єднаємо диз'юнкцією і отримаємо ДДНФ:

$$\neg x \neg y \neg z \vee \neg x \neg y z \vee x \neg y \neg z \vee xy \neg z \vee xyz$$

після перетворень отримаємо скорочену ДНФ:

$$\neg x \neg y \vee x \neg z \vee xy$$

ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ

1. Обхід вглиб і вшир

```
1 struct Vertex {
2     struct Edge {
3         int to;
4         int w;
5     };
6     int deg;
7     int first;
8     int last;
9     int head;
10    int next;
11    int prev;
12    int color;
13    int dist;
14    int parent;
15    int child;
16    int level;
17    int mark;
18    int type;
19    int val;
20    int x;
21    int y;
22    int z;
23    int t;
24    int u;
25    int v;
26    int w;
27    int x;
28    int y;
29    int z;
30    int t;
31    int u;
32    int v;
33    int w;
34    int x;
35    int y;
36    int z;
37    int t;
38    int u;
39    int v;
40    int w;
41    int x;
42    int y;
43    int z;
44    int t;
45    int u;
46    int v;
47    int w;
48    int x;
49    int y;
50    int z;
51    int t;
52    int u;
53    int v;
54    int w;
55    int x;
56    int y;
57    int z;
58    int t;
59    int u;
60    int v;
61    int w;
62    int x;
63    int y;
64    int z;
65    int t;
66    int u;
67    int v;
68    int w;
69    int x;
70    int y;
71    int z;
72    int t;
73    int u;
74    int v;
75    int w;
76    int x;
77    int y;
78    int z;
79    int t;
80    int u;
81    int v;
82    int w;
83    int x;
84    int y;
85    int z;
86    int t;
87    int u;
88    int v;
89    int w;
90    int x;
91    int y;
92    int z;
93    int t;
94    int u;
95    int v;
96    int w;
97    int x;
98    int y;
99    int z;
100   int t;
101   int u;
102   int v;
103   int w;
104   int x;
105   int y;
106   int z;
107   int t;
108   int u;
109   int v;
110   int w;
111   int x;
112   int y;
113   int z;
114   int t;
115   int u;
116   int v;
117   int w;
118   int x;
119   int y;
120   int z;
121   int t;
122   int u;
123   int v;
124   int w;
125   int x;
126   int y;
127   int z;
128   int t;
129   int u;
130   int v;
131   int w;
132   int x;
133   int y;
134   int z;
135   int t;
136   int u;
137   int v;
138   int w;
139   int x;
140   int y;
141   int z;
142   int t;
143   int u;
144   int v;
145   int w;
146   int x;
147   int y;
148   int z;
149   int t;
150   int u;
151   int v;
152   int w;
153   int x;
154   int y;
155   int z;
156   int t;
157   int u;
158   int v;
159   int w;
160   int x;
161   int y;
162   int z;
163   int t;
164   int u;
165   int v;
166   int w;
167   int x;
168   int y;
169   int z;
170   int t;
171   int u;
172   int v;
173   int w;
174   int x;
175   int y;
176   int z;
177   int t;
178   int u;
179   int v;
180   int w;
181   int x;
182   int y;
183   int z;
184   int t;
185   int u;
186   int v;
187   int w;
188   int x;
189   int y;
190   int z;
191   int t;
192   int u;
193   int v;
194   int w;
195   int x;
196   int y;
197   int z;
198   int t;
199   int u;
200   int v;
201   int w;
202   int x;
203   int y;
204   int z;
205   int t;
206   int u;
207   int v;
208   int w;
209   int x;
210   int y;
211   int z;
212   int t;
213   int u;
214   int v;
215   int w;
216   int x;
217   int y;
218   int z;
219   int t;
220   int u;
221   int v;
222   int w;
223   int x;
224   int y;
225   int z;
226   int t;
227   int u;
228   int v;
229   int w;
230   int x;
231   int y;
232   int z;
233   int t;
234   int u;
235   int v;
236   int w;
237   int x;
238   int y;
239   int z;
240   int t;
241   int u;
242   int v;
243   int w;
244   int x;
245   int y;
246   int z;
247   int t;
248   int u;
249   int v;
250   int w;
251   int x;
252   int y;
253   int z;
254   int t;
255   int u;
256   int v;
257   int w;
258   int x;
259   int y;
260   int z;
261   int t;
262   int u;
263   int v;
264   int w;
265   int x;
266   int y;
267   int z;
268   int t;
269   int u;
270   int v;
271   int w;
272   int x;
273   int y;
274   int z;
275   int t;
276   int u;
277   int v;
278   int w;
279   int x;
280   int y;
281   int z;
282   int t;
283   int u;
284   int v;
285   int w;
286   int x;
287   int y;
288   int z;
289   int t;
290   int u;
291   int v;
292   int w;
293   int x;
294   int y;
295   int z;
296   int t;
297   int u;
298   int v;
299   int w;
300   int x;
301   int y;
302   int z;
303   int t;
304   int u;
305   int v;
306   int w;
307   int x;
308   int y;
309   int z;
310   int t;
311   int u;
312   int v;
313   int w;
314   int x;
315   int y;
316   int z;
317   int t;
318   int u;
319   int v;
320   int w;
321   int x;
322   int y;
323   int z;
324   int t;
325   int u;
326   int v;
327   int w;
328   int x;
329   int y;
330   int z;
331   int t;
332   int u;
333   int v;
334   int w;
335   int x;
336   int y;
337   int z;
338   int t;
339   int u;
340   int v;
341   int w;
342   int x;
343   int y;
344   int z;
345   int t;
346   int u;
347   int v;
348   int w;
349   int x;
350   int y;
351   int z;
352   int t;
353   int u;
354   int v;
355   int w;
356   int x;
357   int y;
358   int z;
359   int t;
360   int u;
361   int v;
362   int w;
363   int x;
364   int y;
365   int z;
366   int t;
367   int u;
368   int v;
369   int w;
370   int x;
371   int y;
372   int z;
373   int t;
374   int u;
375   int v;
376   int w;
377   int x;
378   int y;
379   int z;
380   int t;
381   int u;
382   int v;
383   int w;
384   int x;
385   int y;
386   int z;
387   int t;
388   int u;
389   int v;
390   int w;
391   int x;
392   int y;
393   int z;
394   int t;
395   int u;
396   int v;
397   int w;
398   int x;
399   int y;
400   int z;
401   int t;
402   int u;
403   int v;
404   int w;
405   int x;
406   int y;
407   int z;
408   int t;
409   int u;
410   int v;
411   int w;
412   int x;
413   int y;
414   int z;
415   int t;
416   int u;
417   int v;
418   int w;
419   int x;
420   int y;
421   int z;
422   int t;
423   int u;
424   int v;
425   int w;
426   int x;
427   int y;
428   int z;
429   int t;
430   int u;
431   int v;
432   int w;
433   int x;
434   int y;
435   int z;
436   int t;
437   int u;
438   int v;
439   int w;
440   int x;
441   int y;
442   int z;
443   int t;
444   int u;
445   int v;
446   int w;
447   int x;
448   int y;
449   int z;
450   int t;
451   int u;
452   int v;
453   int w;
454   int x;
455   int y;
456   int z;
457   int t;
458   int u;
459   int v;
460   int w;
461   int x;
462   int y;
463   int z;
464   int t;
465   int u;
466   int v;
467   int w;
468   int x;
469   int y;
470   int z;
471   int t;
472   int u;
473   int v;
474   int w;
475   int x;
476   int y;
477   int z;
478   int t;
479   int u;
480   int v;
481   int w;
482   int x;
483   int y;
484   int z;
485   int t;
486   int u;
487   int v;
488   int w;
489   int x;
490   int y;
491   int z;
492   int t;
493   int u;
494   int v;
495   int w;
496   int x;
497   int y;
498   int z;
499   int t;
500   int u;
501   int v;
502   int w;
503   int x;
504   int y;
505   int z;
506   int t;
507   int u;
508   int v;
509   int w;
510   int x;
511   int y;
512   int z;
513   int t;
514   int u;
515   int v;
516   int w;
517   int x;
518   int y;
519   int z;
520   int t;
521   int u;
522   int v;
523   int w;
524   int x;
525   int y;
526   int z;
527   int t;
528   int u;
529   int v;
530   int w;
531   int x;
532   int y;
533   int z;
534   int t;
535   int u;
536   int v;
537   int w;
538   int x;
539   int y;
540   int z;
541   int t;
542   int u;
543   int v;
544   int w;
545   int x;
546   int y;
547   int z;
548   int t;
549   int u;
550   int v;
551   int w;
552   int x;
553   int y;
554   int z;
555   int t;
556   int u;
557   int v;
558   int w;
559   int x;
560   int y;
561   int z;
562   int t;
563   int u;
564   int v;
565   int w;
566   int x;
567   int y;
568   int z;
569   int t;
570   int u;
571   int v;
572   int w;
573   int x;
574   int y;
575   int z;
576   int t;
577   int u;
578   int v;
579   int w;
580   int x;
581   int y;
582   int z;
583   int t;
584   int u;
585   int v;
586   int w;
587   int x;
588   int y;
589   int z;
590   int t;
591   int u;
592   int v;
593   int w;
594   int x;
595   int y;
596   int z;
597   int t;
598   int u;
599   int v;
600   int w;
601   int x;
602   int y;
603   int z;
604   int t;
605   int u;
606   int v;
607   int w;
608   int x;
609   int y;
610   int z;
611   int t;
612   int u;
613   int v;
614   int w;
615   int x;
616   int y;
617   int z;
618   int t;
619   int u;
620   int v;
621   int w;
622   int x;
623   int y;
624   int z;
625   int t;
626   int u;
627   int v;
628   int w;
629   int x;
630   int y;
631   int z;
632   int t;
633   int u;
634   int v;
635   int w;
636   int x;
637   int y;
638   int z;
639   int t;
640   int u;
641   int v;
642   int w;
643   int x;
644   int y;
645   int z;
646   int t;
647   int u;
648   int v;
649   int w;
650   int x;
651   int y;
652   int z;
653   int t;
654   int u;
655   int v;
656   int w;
657   int x;
658   int y;
659   int z;
660   int t;
661   int u;
662   int v;
663   int w;
664   int x;
665   int y;
666   int z;
667   int t;
668   int u;
669   int v;
670   int w;
671   int x;
672   int y;
673   int z;
674   int t;
675   int u;
676   int v;
677   int w;
678   int x;
679   int y;
680   int z;
681   int t;
682   int u;
683   int v;
684   int w;
685   int x;
686   int y;
687   int z;
688   int t;
689   int u;
690   int v;
691   int w;
692   int x;
693   int y;
694   int z;
695   int t;
696   int u;
697   int v;
698   int w;
699   int x;
700   int y;
701   int z;
702   int t;
703   int u;
704   int v;
705   int w;
706   int x;
707   int y;
708   int z;
709   int t;
710   int u;
711   int v;
712   int w;
713   int x;
714   int y;
715   int z;
716   int t;
717   int u;
718   int v;
719   int w;
720   int x;
721   int y;
722   int z;
723   int t;
724   int u;
725   int v;
726   int w;
727   int x;
728   int y;
729   int z;
730   int t;
731   int u;
732   int v;
733   int w;
734   int x;
735   int y;
736   int z;
737   int t;
738   int u;
739   int v;
740   int w;
741   int x;
742   int y;
743   int z;
744   int t;
745   int u;
746   int v;
747   int w;
748   int x;
749   int y;
750   int z;
751   int t;
752   int u;
753   int v;
754   int w;
755   int x;
756   int y;
757   int z;
758   int t;
759   int u;
760   int v;
761   int w;
762   int x;
763   int y;
764   int z;
765   int t;
766   int u;
767   int v;
768   int w;
769   int x;
770   int y;
771   int z;
772   int t;
773   int u;
774   int v;
775   int w;
776   int x;
777   int y;
778   int z;
779   int t;
780   int u;
781   int v;
782   int w;
783   int x;
784   int y;
785   int z;
786   int t;
787   int u;
788   int v;
789   int w;
790   int x;
791   int y;
792   int z;
793   int t;
794   int u;
795   int v;
796   int w;
797   int x;
798   int y;
799   int z;
800   int t;
801   int u;
802   int v;
803   int w;
804   int x;
805   int y;
806   int z;
807   int t;
808   int u;
809   int v;
810   int w;
811   int x;
812   int y;
813   int z;
814   int t;
815   int u;
816   int v;
817   int w;
818   int x;
819   int y;
820   int z;
821   int t;
822   int u;
823   int v;
824   int w;
825   int x;
826   int y;
827   int z;
828   int t;
829   int u;
830   int v;
831   int w;
832   int x;
833   int y;
834   int z;
835   int t;
836   int u;
837   int v;
838   int w;
839   int x;
840   int y;
841   int z;
842   int t;
843   int u;
844   int v;
845   int w;
846   int x;
847   int y;
848   int z;
849   int t;
850   int u;
851   int v;
852   int w;
853   int x;
854   int y;
855   int z;
856   int t;
857   int u;
858   int v;
859   int w;
860   int x;
861   int y;
862   int z;
863   int t;
864   int u;
865   int v;
866   int w;
867   int x;
868   int y;
869   int z;
870   int t;
871   int u;
872   int v;
873   int w;
874   int x;
875   int y;
876   int z;
877   int t;
878   int u;
879   int v;
880   int w;
881   int x;
882   int y;
883   int z;
884   int t;
885   int u;
886   int v;
887   int w;
888   int x;
889   int y;
890   int z;
891   int t;
892   int u;
893   int v;
894   int w;
895   int x;
896   int y;
897   int z;
898   int t;
899   int u;
900   int v;
901   int w;
902   int x;
903   int y;
904   int z;
905   int t;
906   int u;
907   int v;
908   int w;
909   int x;
910   int y;
911   int z;
912   int t;
913   int u;
914   int v;
915   int w;
916   int x;
917   int y;
918   int z;
919   int t;
920   int u;
921   int v;
922   int w;
923   int x;
924   int y;
925   int z;
926   int t;
927   int u;
928   int v;
929   int w;
930   int x;
931   int y;
932   int z;
933   int t;
934   int u;
935   int v;
936   int w;
937   int x;
938   int y;
939   int z;
940   int t;
941   int u;
942   int v;
943   int w;
944   int x;
945   int y;
946   int z;
947   int t;
948   int u;
949   int v;
950   int w;
951   int x;
952   int y;
953   int z;
954   int t;
955   int u;
956   int v;
957   int w;
958   int x;
959   int y;
960   int z;
961   int t;
962   int u;
963   int v;
964   int w;
965   int x;
966   int y;
967   int z;
968   int t;
969   int u;
970   int v;
971   int w;
972   int x;
973   int y;
974   int z;
975   int t;
976   int u;
977   int v;
978   int w;
979   int x;
980   int y;
981   int z;
982   int t;
983   int u;
984   int v;
985   int w;
986   int x;
987   int y;
988   int z;
989   int t;
990   int u;
991   int v;
992   int w;
993   int x;
994   int y;
995   int z;
996   int t;
997   int u;
998   int v;
999   int w;
1000  int x;
1001  int y;
1002  int z;
1003  int t;
1004  int u;
1005  int v;
1006  int w;
1007  int x;
1008  int y;
1009  int z;
1010  int t;
1011  int u;
1012  int v;
1013  int w;
1014  int x;
1015  int y;
1016  int z;
1017  int t;
1018  int u;
1019  int v;
1020  int w;
1021  int x;
1022  int y;
1023  int z;
1024  int t;
1025  int u;
1026  int v;
1027  int w;
1028  int x;
1029  int y;
1030  int z;
1031  int t;
1032  int u;
1033  int v;
1034  int w;
1035  int x;
1036  int y;
1037  int z;
1038  int t;
1039  int u;
1040  int v;
1041  int w;
1042  int x;
1043  int y;
1044  int z;
1045  int t;
1046  int u;
1047  int v;
1048  int w;
1049  int x;
1050  int y;
1051  int z;
1052  int t;
1053  int u;
1054  int v;
1055  int w;
1056  int x;
1057  int y;
1058  int z;
1059  int t;
1060  int u;
1061  int v;
1062  int w;
1063  int x;
1064  int y;
1065  int z;
1066  int t;
1067  int u;
1068  int v;
1069  int w;
1070  int x;
1071  int y;
1072  int z;
1073  int t;
1074  int u;
1075  int v;
1076  int w;
1077  int x;
1078  int y;
1079  int z;
1080  int t;
1081  int u;
1082  int v;
1083  int w;
1084  int x;
1085  int y;
1086  int z;
1087  int t;
1088  int u;
1089  int v;
1090  int w;
1091  int x;
1092  int y;
1093  int z;
1094  int t;
1095  int u;
1096  int v;
1097  int w;
1098  int x;
1099  int y;
1100  int z;
1101  int t;
1102  int u;
1103  int v;
1104  int w;
1105  int x;
1106  int y;
1107  int z;
1108  int t;
1109  int u;
1110  int v;
1111  int w;
1112  int x;
1113  int y;
1114  int z;
1115  int t;
1116  int u;
1117  int v;
1118  int w;
1119  int x;
1120  int y;
1121  int z;
1122  int t;
1123  int u;
1124  int v;
1125  int w;
1126  int x;
1127  int y;
1128  int z;
1129  int t;
1130  int u;
1131  int v;
1132  int w;
1133  int x;
1134  int y;
1135  int z;
1136  int t;
1137  int u;
1138  int v;
1139  int w;
1140  int x;
1141  int y;
1142  int z;
1143  int t;
1144  int u;
1145  int v;
1146  int w;
1147  int x;
1148  int y;
1149  int z;
1150  int t;
1151  int u;
1152  int v;
1153  int w;
1154  int x;
1155  int y;
1156  int z;
1157  int t;
1158  int u;
1159  int v;
1160  int w;
1161  int x;
1162  int y;
1163  int z;
1164  int t;
1165  int u;
1166  int v;
1167  int w;
1168  int x;
1169  int y;
1170  int z;
1171  int t;
1172  int u;
1173  int v;
1174  int w;
1175  int x;
1176  int y;
1177  int z;
1178  int t;
1179  int u;
1180  int v;
1181  int w;
1182  int x;
1183  int y;
1184  int z;
1185  int t;
1186  int u;
1187  int v;
1188  int w;
1189  int x;
1190  int y;
1191  int z;
1192  int t;
1193  int u;
1194  int v;
1195  int w;
1196  int x;
1197  int y;
1198  int z;
1199  int t;
1200  int u;
1201  int v;
1202  int w;
1203  int x;
1204  int y;
1205  int z;
1206  int t;
1207  int u;
1208  int v;
1209  int w;
1210  int x;
1211  int y;
1212  int z;
1213  int t;
1214  int u;
1215  int v;
1216  int w;
1217  int x;
1218  int y;
1219  int z;
1220  int t;
1221  int u;
1222  int v;
1223  int w;
1224  int x;
1225  int y;
1226  int z;
1227  int t;
1228  int u;
1229  int v;
1230  int w;
1231  int x;
1232  int y;
1233  int z;
1234  int t;
1235  int u;
1236  int v;
1237  int w;
1238  int x;
1239  int y;
1240  int z;
1241  int t;
1242  int u;
1243  int v;
1244  int w;
1245  int x;
1246  int y;
1247  int z;
1248  int t;
1249  int u;
1250  int v;
1251  int w;
1252  int x;
1253  int y;
1254  int z;
1255  int t;
1256  int u;
1257  int v;
1258  int w;
1259  int x;
1260  int y;
1261  int z;
1262  int t;
1263  int u;
1264  int v;
1265  int w;
1266  int x;
1267  int y;
1268  int z;
1269  int t;
1270  int u;
1271  int v;
1272  int w;
1273  int x;
1274  int y;
1275  int z;
1276  int t;
1277  int u;
1278  int v;
1279  int w;
1280  int x;
1281  int y;
1282  int z;
1283  int t;
1284  int u;
1285  int v;
1286  int w;
1287  int x;
1288  int y;
1289  int z;
1290  int t;
1291  int u;
1292  int v;
1293  int w;
1294  int x;
1295  int y;
1296  int z;
1297  int t;
1298  int u;
1299  int v;
1300  int w;
1301  int x;
1302  int y;
1303  int z;
1304  int t;
1305  int u;
1306  int v;
1307  int w;
1308  int x;
1309  int y;
1310  int z;
1311  int t;
1312  int u;
1313  int v;
1314  int w;
1315  int x;
1316  int y;
1317  int z;
1318  int t;
1319  int u;
1320  int v;
1321  int w;
1322  int x;
1323  int y;
1324  int z;
1325  int t;
1326  int u;
1327  int v;
1328  int w;
1329  int x;
1330  int y;
1331  int z;
1332  int t;
1333  int u;
1334  int v;
1335  int w;
1336  int x;
1337  int y;
1338  int z;
1339  int t;
1340  int u;
1341  int v;
1342  int w;
1343  int x;
1344  int y;
1345  int z;
1346  int t;
1347  int u;
1348  int v;
1349  int w;
1350  int x;
1351  int y;
1352  int z;
1353  int t;
1354  int u;
1355  int v;
1356  int w;
1357  int x;
1358  int y;
1359  int z;
1360  int t;
1361  int u;
1362  int v;
1363  int w;
1364  int x;
1365  int y;
1366  int z;
1367  int t;
1368  int u;
1369  int v;
1370  int w;
1371  int x;
1372  int y;
1373  int z;
1374  int t;
1375  int u;
1376  int v;
1377  int w;
1378  int x;
1379  int y;
1380  int z;
1381  int t;
1382  int u;
1383  int v;
1384  int w;
1385  int x;
1386  int y;
1387  int z;
1388  int t;
1389  int u;
1390  int v;
1391  int w;
1392  int x;
1393  int y;
1394  int z;
1395  int t;
1396  int u;
1397  int v;
1398  int w;
1399  int x;
1400  int y;
1401  int z;
1402  int t;
1403  int u;
1404  int v;
1405  int w;
1406  int x;
1407  int y;
1408  int z;
1409  int t;
1410  int u;
1411  int v;
1412  int w;
1413  int x;
1414  int y;
1415  int z;
1416  int t;
1417  int u;
1418  int v;
1419  int w;
1420  int x;
1421  int y;
1422  int z;
1423  int t;
1424  int u;
1425  int v;
1426  int w;
1427  int x;
1428  int y;
1429  int z;
1430  int t;
1431  int u;
1432  int v;
1433  int w;
1434  int x;
1435  int y;
1436  int z;
1437  int t;
1438  int u;
1439  int v;
1440  int w;
1441  int x;
1442  int y;
1443  int z;
1444  int t;
1445  int u;
1446  int v;
1447  int w;
1448  int x;
1449  int y;
1450  int z;
1451  int t;
1452  int u;
1453  int v;
1454  int w;
1455  int x;
1456  int y;
1457  int z;
1458  int t;
1459  int u;
1460  int v;
1461  int w;
1462  int x;
1463  int y;
1464  int z;
1465  int t;
1466  int u;
1467  int v;
1468  int w;
1469  int x;
1470  int y;
1471  int z;
1472  int t;
1473  int u;
1474  int v;
1475  int w;
1476  int x;
1477  int y;
1478  int z;
1479  int t;
1480  int u;
1481  int v;
1482  int w;
1483  int x;
1484  int y;
1485  int z;

```

Number of edges in a graph: 20
Number of vertices in a graph: 10

First verticle -> to edge M1: 1
Second verticle -> to edge M1: 2
First verticle -> to edge M2: 1
Second verticle -> to edge M2: 8
First verticle -> to edge M3: 1
Second verticle -> to edge M3: 7
First verticle -> to edge M4: 2
Second verticle -> to edge M4: 7
First verticle -> to edge M5: 2
Second verticle -> to edge M5: 8
First verticle -> to edge M6: 2
Second verticle -> to edge M6: 8
First verticle -> to edge M7: 2
Second verticle -> to edge M7: 9
First verticle -> to edge M8: 4
Second verticle -> to edge M8: 8
First verticle -> to edge M9: 4
Second verticle -> to edge M9: 7
First verticle -> to edge M10: 4
Second verticle -> to edge M10: 5
First verticle -> to edge M11: 8
Second verticle -> to edge M11: 7
First verticle -> to edge M12: 8
Second verticle -> to edge M12: 8
First verticle -> to edge M13: 8
Second verticle -> to edge M13: 7
First verticle -> to edge M14: 8
Second verticle -> to edge M14: 8
First verticle -> to edge M15: 8
Second verticle -> to edge M15: 9
First verticle -> to edge M16: 8
Second verticle -> to edge M16: 10
First verticle -> to edge M17: 7

Second verticle -> to edge M16: 10
First verticle -> to edge M17: 7
Second verticle -> to edge M17: 8
First verticle -> to edge M18: 7
Second verticle -> to edge M18: 8
From which verticle do you want to start? 1
Choose what do you want to do:

- 1 - DFS
- 2 - BFS

1
1 2
1 2 7
1 2 7 4
1 2 7 4 3
1 2 7 4 3 9
1 2 7 4 3 9 8
1 2 7 4 3 9 8 6
1 2 7 4 3 9 8 6 5
1 2 7 4 3 9 8 6
1 2 7 4 3 9 8
1 2 7 4 3 9
1 2 7 4 3 9 10
1 2 7 4 3 9
1 2 7 4 3
1 2 7 4
1 2 7
1 2
1
Stack is empty

Choose what do you want to do:

- 1 - DFS
- 2 - BFS

2
1 2
1 2 8
1 2 8 7
2 8 7
2 8 7 3
8 7 3
8 7 3 6
8 7 3 6 9
7 3 6 9
7 3 6 9 4
7 3 6 9 4 5
3 6 9 4 5
6 9 4 5
9 4 5
9 4 5 10
4 5 10
5 10
10

Queue is empty

2. Метод Прима

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int x, y;
8     int u, v;
9     int number;
10    int l, j;
11    int m = 5;
12    int size[10] = {0};
13    int minimal, minimal1 = 0;
14    int cost[10][10];
15    int path[100] = {0};
16    int path_index = 0;
17
18    cout << "Введите количество вершин: ";
19    cin >> number;
20    cout << "Введите матрицу смежности:\n";
21
22    for(i = 1; i <= number; i++)
23        for(j = 1; j <= number; j++)
24        {
25            cout << "cost[" << i << "][" << j << "] = ";
26            if(cost[i][j] == 0)
27                cost[i][j] = 100;
28        }
29
30    size[1] = 1;
31    cout << endl;
32
33    while(size < number)
34    {
35        for(i = 1; minimal = 100; i <= number; i++)
36            for(j = 1; j <= number; j++)
37                if(cost[i][j] < minimal)
38                    if(size[i] != 0)
39                    {
40                        minimal = cost[i][j];
41                        x = u + i;
42                        y = v + j;
43                    }
44    }
```

```
Edge *edg = new Edge[EdgeNumber];
array<int,3>;

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main()
{
    int i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, A1, B1, C1, D1, E1, F1, G1, H1, I1, J1, K1, L1, M1, N1, O1, P1, Q1, R1, S1, T1, U1, V1, W1, X1, Y1, Z1, A2, B2, C2, D2, E2, F2, G2, H2, I2, J2, K2, L2, M2, N2, O2, P2, Q2, R2, S2, T2, U2, V2, W2, X2, Y2, Z2, A3, B3, C3, D3, E3, F3, G3, H3, I3, J3, K3, L3, M3, N3, O3, P3, Q3, R3, S3, T3, U3, V3, W3, X3, Y3, Z3, A4, B4, C4, D4, E4, F4, G4, H4, I4, J4, K4, L4, M4, N4, O4, P4, Q4, R4, S4, T4, U4, V4, W4, X4, Y4, Z4, A5, B5, C5, D5, E5, F5, G5, H5, I5, J5, K5, L5, M5, N5, O5, P5, Q5, R5, S5, T5, U5, V5, W5, X5, Y5, Z5, A6, B6, C6, D6, E6, F6, G6, H6, I6, J6, K6, L6, M6, N6, O6, P6, Q6, R6, S6, T6, U6, V6, W6, X6, Y6, Z6, A7, B7, C7, D7, E7, F7, G7, H7, I7, J7, K7, L7, M7, N7, O7, P7, Q7, R7, S7, T7, U7, V7, W7, X7, Y7, Z7, A8, B8, C8, D8, E8, F8, G8, H8, I8, J8, K8, L8, M8, N8, O8, P8, Q8, R8, S8, T8, U8, V8, W8, X8, Y8, Z8, A9, B9, C9, D9, E9, F9, G9, H9, I9, J9, K9, L9, M9, N9, O9, P9, Q9, R9, S9, T9, U9, V9, W9, X9, Y9, Z9, A10, B10, C10, D10, E10, F10, G10, H10, I10, J10, K10, L10, M10, N10, O10, P10, Q10, R10, S10, T10, U10, V10, W10, X10, Y10, Z10, A11, B11, C11, D11, E11, F11, G11, H11, I11, J11, K11, L11, M11, N11, O11, P11, Q11, R11, S11, T11, U11, V11, W11, X11, Y11, Z11, A12, B12, C12, D12, E12, F12, G12, H12, I12, J12, K12, L12, M12, N12, O12, P12, Q12, R12, S12, T12, U12, V12, W12, X12, Y12, Z12, A13, B13, C13, D13, E13, F13, G13, H13, I13, J13, K13, L13, M13, N13, O13, P13, Q13, R13, S13, T13, U13, V13, W13, X13, Y13, Z13, A14, B14, C14, D14, E14, F14, G14, H14, I14, J14, K14, L14, M14, N14, O14, P14, Q14, R14, S14, T14, U14, V14, W14, X14, Y14, Z14, A15, B15, C15, D15, E15, F15, G15, H15, I15, J15, K15, L15, M15, N15, O15, P15, Q15, R15, S15, T15, U15, V15, W15, X15, Y15, Z15, A16, B16, C16, D16, E16, F16, G16, H16, I16, J16, K16, L16, M16, N16, O16, P16, Q16, R16, S16, T16, U16, V16, W16, X16, Y16, Z16, A17, B17, C17, D17, E17, F17, G17, H17, I17, J17, K17, L17, M17, N17, O17, P17, Q17, R17, S17, T17, U17, V17, W17, X17, Y17, Z17, A18, B18, C18, D18, E18, F18, G18, H18, I18, J18, K18, L18, M18, N18, O18, P18, Q18, R18, S18, T18, U18, V18, W18, X18, Y18, Z18, A19, B19, C19, D19, E19, F19, G19, H19, I19, J19, K19, L19, M19, N19, O19, P19, Q19, R19, S19, T19, U19, V19, W19, X19, Y19, Z19, A20, B20, C20, D20, E20, F20, G20, H20, I20, J20, K20, L20, M20, N20, O20, P20, Q20, R20, S20, T20, U20, V20, W20, X20, Y20, Z20, A21, B21, C21, D21, E21, F21, G21, H21, I21, J21, K21, L21, M21, N21, O21, P21, Q21, R21, S21, T21, U21, V21, W21, X21, Y21, Z21, A22, B22, C22, D22, E22, F22, G22, H22, I22, J22, K22, L22, M22, N22, O22, P22, Q22, R22, S22, T22, U22, V22, W22, X22, Y22, Z22, A23, B23, C23, D23, E23, F23, G23, H23, I23, J23, K23, L23, M23, N23, O23, P23, Q23, R23, S23, T23, U23, V23, W23, X23, Y23, Z23, A24, B24, C24, D24, E24, F24, G24, H24, I24, J24, K24, L24, M24, N24, O24, P24, Q24, R24, S24, T24, U24, V24, W24, X24, Y24, Z24, A25, B25, C25, D25, E25, F25, G25, H25, I25, J25, K25, L25, M25, N25, O25, P25, Q25, R25, S25, T25, U25, V25, W25, X25, Y25, Z25, A26, B26, C26, D26, E26, F26, G26, H26, I26, J26, K26, L26, M26, N26, O26, P26, Q26, R26, S26, T26, U26, V26, W26, X26, Y26, Z26, A27, B27, C27, D27, E27, F27, G27, H27, I27, J27, K27, L27, M27, N27, O27, P27, Q27, R27, S27, T27, U27, V27, W27, X27, Y27, Z27, A28, B28, C28, D28, E28, F28, G28, H28, I28, J28, K28, L28, M28, N28, O28, P28, Q28, R28, S28, T28, U28, V28, W28, X28, Y28, Z28, A29, B29, C29, D29, E29, F29, G29, H29, I29, J29, K29, L29, M29, N29, O29, P29, Q29, R29, S29, T29, U29, V29, W29, X29, Y29, Z29, A30, B30, C30, D30, E30, F30, G30, H30, I30, J30, K30, L30, M30, N30, O30, P30, Q30, R30, S30, T30, U30, V30, W30, X30, Y30, Z30, A31, B31, C31, D31, E31, F31, G31, H31, I31, J31, K31, L31, M31, N31, O31, P31, Q31, R31, S31, T31, U31, V31, W31, X31, Y31, Z31, A32, B32, C32, D32, E32, F32, G32, H32, I32, J32, K32, L32, M32, N32, O32, P32, Q32, R32, S32, T32, U32, V32, W32, X32, Y32, Z32, A33, B33, C33, D33, E33, F33, G33, H33, I33, J33, K33, L33, M33, N33, O33, P33, Q33, R33, S33, T33, U33, V33, W33, X33, Y33, Z33, A34, B34, C34, D34, E34, F34, G34, H34, I34, J34, K34, L34, M34, N34, O34, P34, Q34, R34, S34, T34, U34, V34, W34, X34, Y34, Z34, A35, B35, C35, D35, E35, F35, G35, H35, I35, J35, K35, L35, M35, N35, O35, P35, Q35, R35, S35, T35, U35, V35, W35, X35, Y35, Z35, A36, B36, C36, D36, E36, F36, G36, H36, I36, J36, K36, L36, M36, N36, O36, P36, Q36, R36, S36, T36, U36, V36, W36, X36, Y36, Z36, A37, B37, C37, D37, E37, F37, G37, H37, I37, J37, K37, L37, M37, N37, O37, P37, Q37, R37, S37, T37, U37, V37, W37, X37, Y37, Z37, A38, B38, C38, D38, E38, F38, G38, H38, I38, J38, K38, L38, M38, N38, O38, P38, Q38, R38, S38, T38, U38, V38, W38, X38, Y38, Z38, A39, B39, C39, D39, E39, F39, G39, H39, I39, J39, K39, L39, M39, N39, O39, P39, Q39, R39, S39, T39, U39, V39, W39, X39, Y39, Z39, A40, B40, C40, D40, E40, F40, G40, H40, I40, J40, K40, L40, M40, N40, O40, P40, Q40, R40, S40, T40, U40, V40, W40, X40, Y40, Z40, A41, B41, C41, D41, E41, F41, G41, H41, I41, J41, K41, L41, M41, N41, O41, P41, Q41, R41, S41, T41, U41, V41, W41, X41, Y41, Z41, A42, B42, C42, D42, E42, F42, G42, H42, I42, J42, K42, L42, M42, N42, O42, P42, Q42, R42, S42, T42, U42, V42, W42, X42, Y42, Z42, A43, B43, C43, D43, E43, F43, G43, H43, I43, J43, K43, L43, M43, N43, O43, P43, Q43, R43, S43, T43, U43, V43, W43, X43, Y43, Z43, A44, B44, C44, D44, E44, F44, G44, H44, I44, J44, K44, L44, M44, N44, O44, P44, Q44, R44, S44, T44, U44, V44, W44, X44, Y44, Z44, A45, B45, C45, D45, E45, F45, G45, H45, I45, J45, K45, L45, M45, N45, O45, P45, Q45, R45, S45, T45, U45, V45, W45, X45, Y45, Z45, A46, B46, C46, D46, E46, F46, G46, H46, I46, J46, K46, L46, M46, N46, O46, P46, Q46, R46, S46, T46, U46, V46, W46, X46, Y46, Z46, A47, B47, C47, D47, E47, F47, G47, H47, I47, J47, K47, L47, M47, N47, O47, P47, Q47, R47, S47, T47, U47, V47, W47, X47, Y47, Z47, A48, B48, C48, D48, E48, F48, G48, H48, I48, J48, K48, L48, M48, N48, O48, P48, Q48, R48, S48, T48, U48, V48, W48, X48, Y48, Z48, A49, B49, C49, D49, E49, F49, G49, H49, I49, J49, K49, L49, M49, N49, O49, P49, Q49, R49, S49, T49, U49, V49, W49, X49, Y49, Z49, A50, B50, C50, D50, E50, F50, G50, H50, I50, J50, K50, L50, M50, N50, O50, P50, Q50, R50, S50, T50, U50, V50, W50, X50, Y50, Z50, A51, B51, C51, D51, E51, F51, G51, H51, I51, J51, K51, L51, M51, N51, O51, P51, Q51, R51, S51, T51, U51, V51, W51, X51, Y51, Z51, A52, B52, C52, D52, E52, F52, G52, H52, I52, J52, K52, L52, M52, N52, O52, P52, Q52, R52, S52, T52, U52, V52, W52, X52, Y52, Z52, A53, B53, C53, D53, E53, F53, G53, H53, I53, J53, K53, L53, M53, N53, O53, P53, Q53, R53, S53, T53, U53, V53, W53, X53, Y53, Z53, A54, B54, C54, D54, E54, F54, G54, H54, I54, J54, K54, L54, M54, N54, O54, P54, Q54, R54, S54, T54, U54, V54, W54, X54, Y54, Z54, A55, B55, C55, D55, E55, F55, G55, H55, I55, J55, K55, L55, M55, N55, O55, P55, Q55, R55, S55, T55, U55, V55, W55, X55, Y55, Z55, A56, B56, C56, D56, E56, F56, G56, H56, I56, J56, K56, L56, M56, N56, O56, P56, Q56, R56, S56, T56, U56, V56, W56, X56, Y56, Z56, A57, B57, C57, D57, E57, F57, G57, H57, I57, J57, K57, L57, M57, N57, O57, P57, Q57, R57, S57, T57, U57, V57, W57, X57, Y57, Z57, A58, B58, C58, D58, E58, F58, G58, H58, I58, J58, K58, L58, M58, N58, O58, P58, Q58, R58, S58, T58, U58, V58, W58, X58, Y58, Z58, A59, B59, C59, D59, E59, F59, G59, H59, I59, J59, K59, L59, M59, N59, O59, P59, Q59, R59, S59, T59, U59, V59, W59, X59, Y59, Z59, A60, B60, C60, D60, E60, F60, G60, H60, I60, J60, K60, L60, M60, N60, O60, P60, Q60, R60, S60, T60, U60, V60, W60, X60, Y60, Z60, A61, B61, C61, D61, E61, F61, G61, H61, I61, J61, K61, L61, M61, N61, O61, P61, Q61, R61, S61, T61, U61, V61, W61, X61, Y61, Z61, A62, B62, C62, D62, E62, F62, G62, H62, I62, J62, K62, L62, M62, N62, O62, P62, Q62, R62, S62, T62, U62, V62, W62, X62, Y62, Z62, A63, B63, C63, D63, E63, F63, G63, H63, I63, J63, K63, L63, M63, N63, O63, P63, Q63, R63, S63, T63, U63, V63, W63, X63, Y63, Z63, A64, B64, C64, D64, E64, F64, G64, H64, I64, J64, K64, L64, M64, N64, O64, P64, Q64, R64, S64, T64, U64, V64, W64, X64, Y64, Z64, A65, B65, C65, D65, E65, F65, G65, H65, I65, J65, K65, L65, M65, N65, O65, P65, Q65, R65, S65, T65, U65, V65, W65, X65, Y65, Z65, A66, B66, C66, D66, E66, F66, G66, H66, I66, J66, K66, L66, M66, N66, O66, P66, Q66, R66, S66, T66, U66, V66, W66, X66, Y66, Z66, A67, B67, C67, D67, E67, F67, G67, H67, I67, J67, K67, L67, M67, N67, O67, P67, Q67, R67, S67, T67, U67, V67, W67, X67, Y67, Z67, A68, B68, C68, D68, E68, F68, G68, H68, I68, J68, K68, L68, M68, N68, O68, P68, Q68, R68, S68, T68, U68, V68, W68, X68, Y68, Z68, A69, B69, C69, D69, E69, F69, G69, H69, I69, J69, K69, L69, M69, N69, O69, P69, Q69, R69, S69, T69, U69, V69, W69, X69, Y69, Z69, A70, B70, C70, D70, E70, F70, G70, H70, I70, J70, K70, L70, M70, N70, O70, P70, Q70, R70, S70, T70, U70, V70, W70, X70, Y70, Z70, A71, B71, C71, D71, E71, F71, G71, H71, I71, J71, K71, L71, M71, N71, O71, P71, Q71, R71, S71, T71, U71, V71, W71, X71, Y71, Z71, A72, B72, C72, D72, E72, F72, G72, H72, I72, J72, K72, L72, M72, N72, O72, P72, Q72, R72, S72, T72, U72, V72, W72, X72, Y72, Z72, A73, B73, C73, D73, E73, F73, G73, H73, I73, J73, K73, L73, M73, N73, O73, P73, Q73, R73, S73, T73, U73, V73, W73, X73, Y73, Z73, A74, B74, C74, D74, E74, F74, G74, H74, I74, J74, K74, L74, M74, N74, O74, P74, Q74, R74, S74, T74, U74, V74, W74, X74, Y74, Z74, A75, B75, C75, D75, E75, F75, G75, H75, I75, J75, K75, L75, M75, N75, O75, P75, Q75, R75, S75, T75, U75, V75, W75, X75, Y75, Z75, A76, B76, C76, D76, E76, F76, G76, H76, I76, J76, K76, L76, M76, N76, O76, P76, Q76, R76, S76, T76, U76, V76, W76, X76, Y76, Z76, A77, B77, C77, D77, E77, F77, G77, H77, I77, J77, K77, L77, M77, N77, O77, P77, Q77, R77, S77, T77, U77, V77, W77, X77, Y77, Z77, A78, B78, C78, D78, E78, F78, G78, H78, I78, J78, K78, L78, M78, N78, O78, P78, Q78, R78, S78, T78, U78, V78, W78, X78, Y78, Z78, A79, B79, C79, D79, E79, F79, G79, H79, I79, J79, K79, L79, M79, N79, O79, P79, Q79, R79, S79, T79, U79, V79, W79, X79, Y79, Z79, A80, B80, C80, D80, E80, F80, G80, H80, I80, J80, K80, L80, M80, N80, O80, P80, Q80, R80, S80, T80, U80, V80, W80, X80, Y80, Z80, A81, B81, C81, D81, E81, F81, G81, H81, I81, J81, K81, L81, M81, N81, O81, P81, Q81, R81, S81, T81, U81, V81, W81, X81, Y81, Z81, A82, B82, C82, D82, E82, F82, G82, H82, I82, J82, K82, L82, M82, N82, O82, P82, Q82, R82, S82, T82, U82, V82, W82, X82, Y82, Z82, A83, B83, C83, D83, E83, F83, G83, H83, I83, J83, K83, L83, M83, N83, O83, P83, Q83, R83, S83, T83, U83, V83, W83, X83, Y83, Z83, A84, B84, C84, D84, E84, F84, G84, H84, I84, J84, K84, L84, M84, N84, O84, P84, Q84, R84, S84, T84, U84, V84, W84, X84, Y84, Z84, A85, B85, C85, D85, E85, F85, G85, H85, I85, J85, K85, L85, M85, N85, O85, P85, Q85, R85, S85, T85, U85, V85, W85, X85, Y85, Z85, A86, B86, C86, D86, E86, F86, G86, H86, I86, J86, K86, L86, M86, N86, O86, P86, Q86, R86, S86, T86, U86, V86, W86, X86, Y86, Z86, A87, B87, C87, D87, E87, F87, G87, H87, I87, J87, K87, L87, M87, N87, O87, P87, Q87, R87, S87, T87, U87, V87, W87, X87, Y87, Z87, A88, B88, C88, D88, E88, F88, G88, H88, I88, J88, K88, L88, M88, N88, O88, P88, Q88, R88, S88, T88, U88, V88, W88, X88, Y88, Z88, A89, B89, C89, D89, E89, F89, G89, H89, I89, J89, K89, L89, M89, N89, O89, P89, Q89, R89, S89, T89, U89, V89, W89, X89, Y89, Z89, A90, B90, C90, D90, E90, F90, G90, H90, I90, J90, K90, L90, M90, N90, O90, P90, Q90, R90, S90, T90, U90, V90, W90, X90, Y90, Z90, A91, B91, C91, D91, E91, F91, G91, H91, I91, J91, K91, L91, M91, N91, O91, P91, Q91, R91, S91, T91, U91, V91, W91, X91, Y91, Z91, A92, B92, C92, D92, E92, F92, G92, H92, I92, J92, K92, L92, M92, N92, O92, P92, Q92, R92, S92, T92, U92, V92, W92, X92, Y92, Z92, A93, B93, C93, D93, E93, F93, G93, H93, I93, J93, K93, L93, M93, N93, O93, P93, Q93, R93, S93, T93, U93, V93, W93, X93, Y93, Z93, A94, B94, C94, D94, E94, F94, G94, H94, I94, J94, K94, L94, M94, N94, O94, P94, Q94, R94, S94, T94, U94, V94, W94, X94, Y94, Z94, A95, B95, C95, D95, E95, F95, G95, H95, I95, J95, K95, L95, M95, N95, O95, P95, Q95, R95, S95, T95, U95, V95, W95, X95, Y95, Z95, A96, B96, C96, D96, E96, F96, G96, H96, I96, J96, K96, L96, M96, N96, O96, P96, Q96, R96, S96, T96, U96, V96, W96, X96, Y96, Z96, A97, B97, C97, D97, E97, F97, G97, H97, I97, J97, K97, L97, M97, N97, O97, P97, Q97, R97, S97, T97, U97, V97, W97, X97, Y97, Z97, A98, B98, C98, D98, E98, F98, G98, H98, I98, J98, K98, L98, M98, N98, O98, P98, Q98, R98, S98, T98, U98, V98, W98, X98, Y98, Z98, A99, B99, C99, D99, E99, F99, G99, H99, I99, J99, K99, L99, M99, N99, O99, P99, Q99, R99, S99, T99, U99, V99, W99, X99, Y99, Z99, A100, B100, C100, D100, E100, F100, G100, H100, I100, J100, K100, L100, M100, N100, O100, P100, Q100, R100, S100, T100, U100, V100, W100, X100, Y100, Z100, A101, B101, C101, D101, E101, F101, G101, H101, I101, J101, K101, L101, M101, N101, O101, P101, Q101, R101, S101, T101, U101, V101, W101, X101, Y101, Z101, A102, B102, C102, D102, E102, F102, G102, H102, I102, J102, K102, L102, M102, N102, O102, P102, Q102, R102, S102, T102, U102, V102, W102, X102, Y102, Z102, A103, B103, C103, D103, E103, F103, G103, H103, I103, J103, K103, L103, M103, N103, O103, P103, Q103, R103, S103, T103, U103, V103, W103, X103, Y103, Z103, A104, B104, C104, D104, E104, F104, G104, H104, I104, J104, K104, L104, M104, N104, O104, P104, Q104, R104, S104, T104, U104, V104, W104, X104, Y104, Z104, A105, B105, C105, D105, E105, F105, G105, H105, I105, J105, K105, L105, M105, N105, O105, P105, Q105, R105, S105, T105, U105, V105, W105, X105, Y105, Z105, A106, B106, C106, D106, E106, F106, G106, H106, I106, J106, K106, L106, M106, N106, O106, P106, Q106, R106, S106, T106, U106, V106, W106, X106, Y106, Z106, A107, B107, C107, D107, E107, F107, G107, H107, I107, J107, K107, L107, M107, N107, O107, P107, Q107, R107, S107, T107, U107, V107, W107, X107, Y107, Z107, A108, B108, C108, D108, E108, F108, G108, H108, I108, J108, K108, L108, M108, N108, O108, P108, Q108, R108, S108, T108, U108, V108, W108, X108, Y108, Z108, A109, B109, C109, D109, E109, F109, G109, H109, I109, J109, K109, L109, M109, N109, O109, P109, Q109, R109, S109, T109, U109, V109, W109, X109, Y109, Z109, A110, B110, C110, D110, E110, F110, G110, H110, I110, J110, K110, L110, M110, N110, O110, P110, Q110, R110, S110, T110, U110, V110, W110, X110, Y110, Z110, A111, B111, C111, D111, E111, F111, G111, H111, I111, J111, K111, L111, M111, N111, O111, P111, Q111, R111, S111, T111, U111, V111, W111, X111, Y111, Z111, A112, B112, C112, D112, E112, F112, G112, H1
```

3. Метод Краскала

```
#include <iostream>
#include <vector>

using namespace std;

struct Array {
    int arrSize;
    int *arr = 0;
};

struct Edge {
    int iLength;
    int s;
    int e;
    bool dir = false;
};

void initEdge(edgeList, int n) {
    for (int i = 0; i < n; i++) {
        cout << "Length of " << i + 1 << " is " << edges[i].iLength << endl;
        cin >> edges[i].iLength;
        cout << "First vertex connected to edge " << i + 1 << " is " << endl;
        cin >> edges[i].s;
        cout << "Second vertex connected to edge " << i + 1 << " is " << endl;
        cin >> edges[i].e;
        cout << endl;
    }
    return 0;
}

// print main 3 4
```

```
int iEdgeNumber = 0, temp = 100, total = 100;

cout << "Number of edges in a graph: ";
cin >> iEdgeNumber;
int iVertexNumber;
```

```
cout << "Number of vertices in a graph: "
```

```

14 if (temp1 == temp2) {
15     if (temp1 == temp2) {
16         for (let i = 0; i < arr[temp2].edges; i++) {
17             arr[temp1].arr[arr[temp2].edges + i] = arr[temp2].arr[i];
18             arr[temp1].arr[i] = 0;
19         }
20         arr[temp1].edges = arr[temp2].edges;
21         arr[temp2].edges = 0;
22     }
23
24     if (temp2 == temp3) {
25         for (let i = 0; i < arr[temp3].edges; i++) {
26             arr[temp2].arr[arr[temp3].edges + i] = arr[temp3].arr[i];
27             arr[temp2].arr[i] = 0;
28         }
29         arr[temp2].edges = arr[temp3].edges;
30         arr[temp3].edges = 0;
31     }
32 }
33
34 if (temp0 == 100 && temp2 == 100) {
35     arr[0].arrified[0].edges = res[0].edges;
36     arr[0].arrified[0].edges = 0;
37     res[0].is = true;
38 }
39
40 if (temp0 == temp2 && temp0 != 100) {
41     res[0].is = false;
42 }
43
44 temp0 = 100; temp2 = 100;
45
46 let temp = 0;
47
48 for (let i = 0; i < arr.length; i++) {
49     if (res[i].is == true) {
50         cout << "Edge" << " connecting vertices " << res[i].x << " << res[i].y << endl;
51         temp += res[i].length;
52     }
53 }
54
55 cout << "Maximum spanning tree" << temp;
56
57 return 0;
58 }

```

```

1 for (i = 1; minimal = 999; i == number; ++i)
2     for (j = 1; j <= number; j++)
3         if (cost[i][j] < minimal)
4             if (new - cost[i] < 0)
5                 {
6                     minimal = cost[i][j];
7                     x = i + 1;
8                     y = j + 1;
9                 }
10 if (scan[i] == 0 || scan[j] == 1)
11 {
12     path[path_index] = i;
13     path_index++;
14     cost += ans[i];
15     cost += new - cost[i] + cost[j] - cost[i] + minimal;
16
17     minimal += minimal;
18     scan[j] = 1;
19 }
20 cost[i][j] = cost[j][i] = 999;
21
22 cost += ans[i];
23 cost += 1 + " + " + "
24
25 for (list k = 0; k + number - 1; list)
26 {
27     cost += path[i];
28     if (k + number - 1) {
29         cost += " + " + "
30     }
31 }
32 cost += ans[i];
33 cost += "Minimal cost: " + endL;
34 cost += minimal;

```

```

Введіть матрицю ємностей:
0 4 3 2 0 0 0 0 0 0 0
0 0 0 0 2 0 2 0 0 0 0
3 0 0 0 0 2 0 0 0 0 0
2 0 0 0 0 2 4 0 0 0 0
0 2 0 0 0 0 0 2 2 0 0
0 0 7 3 0 0 0 0 0 0 0
0 1 0 4 0 0 0 0 4 3 0
0 0 0 0 7 4 0 0 0 0 0
0 0 0 0 5 0 0 0 0 0 0
0 0 0 0 0 3 2 0 0 0 0
0 0 0 0 0 0 0 7 2 3 0

1 1 42
2 4 82
3 1 33
4 6 183
5 10 113
6 11 91
7 1 24
8 2 71
9 2 52
10 6 84
1 -> 4 -> 6 -> 3 -> 10 -> 11 -> 9 -> 2 -> 7 -> 5 -> 8
Minimal cost:
25

```

```

13 if (temp1 < temp2 && temp1 < 100 && temp2 < 100) {
14     if (temp1 < temp2) {
15         for (int i = 0; i < int(temp2).size(); i++)
16         {
17             int temp3 = int(temp2[i].size() + 1) + int(temp1[i]);
18             int temp2[i].size() = 0;
19         }
20         int temp1.size() = int(temp2.size());
21         int temp2.size() = 0;
22     }
23     if (temp2 < temp1) {
24         for (int i = 0; i < int(temp1).size(); i++)
25         {
26             int temp2[i].size() = int(temp1[i].size() + 1) + int(temp2[i]);
27             int temp1[i].size() = 0;
28         }
29         int temp2.size() = int(temp1.size());
30         int temp1.size() = 0;
31     }
32 }
33 if (temp1 < 100 && temp2 < 100) { temp1[i].size() = temp2[i].size() + 1; temp2[i].size() = 0; }
34 if (temp1 < 100 && temp2 < 100) { temp2[i].size() = temp1[i].size() + 1; temp1[i].size() = 0; }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

```

Length of 12 edge: 1
First verticle connected to edges: 12 2
Second verticle connected to edges: 12 2

Length of 13 edge: 5
First verticle connected to edges: 13 5
Second verticle connected to edges: 13 2

Length of 14 edge: 3
First verticle connected to edges: 14 6
Second verticle connected to edges: 14 10

Length of 15 edge: 6
First verticle connected to edges: 15 2
Second verticle connected to edges: 15 5

Length of 16 edge: 4
First verticle connected to edges: 16 6
Second verticle connected to edges: 16 3

Length of 17 edge: 2
First verticle connected to edges: 17 4
Second verticle connected to edges: 17 5

Length of 18 edge: 4
First verticle connected to edges: 18 2
Second verticle connected to edges: 18 3

Edge connecting verticles M9 11
Edge connecting verticles M2 7
Edge connecting verticles M1 4
Edge connecting verticles M2 5
Edge connecting verticles M4 6
Edge connecting verticles M1 3
Edge connecting verticles M10 11
Edge connecting verticles M6 10
Edge connecting verticles M1 2
Edge connecting verticles M6 8
Minimal spinning tree: 25

```

```

Edge connecting verticles M9 11
Edge connecting verticles M2 7
Edge connecting verticles M1 4
Edge connecting verticles M2 5
Edge connecting verticles M4 6
Edge connecting verticles M1 3
Edge connecting verticles M10 11
Edge connecting verticles M6 10
Edge connecting verticles M1 2
Edge connecting verticles M6 8
Minimal spinning tree: 25

```

4. Алгоритм Дейкстры

```

1 #include <iostream>
2
3 using namespace std;
4
5 int N;
6 int graph[40][40];
7 int path[40];
8 bool check[40];
9 int formation[40];
10
11 void makegrao()
12 {
13     int i, j, min, w, v, u;
14     int col, row;
15
16     cout << "Enter the number of nodes: ";
17     cin >> N;
18     cout << "Enter the number of edges: ";
19     int m;
20     cin >> m;
21     cout << "Enter the number of rows: ";
22     int r;
23     cin >> r;
24
25     for (i = 0; i < N; i++) {
26         for (j = 1; j < N; j++) {
27             if (i == j + 1 || j == i + 1) {
28                 cout << "Vertex: " << i << " to " << j << " : ";
29                 cout << "weight: " << j + i << " ";
30                 cin >> graph[i][j];
31             } else {
32                 graph[i][j] = 0;
33             }
34         }
35     }
36 }
37
38 int pathmin()
39 {
40     int min = 10000, minDist;
41     for (int v = 0; v < N; v++) {
42         if (check[v] == false && path[v] == min)
43             min = path[v];
44             minDist = v;
45     }
46     return minDist;
47 }
48
49 void printgraph(int i)
50 {
51     if (formation[i] == -1)
52         return;
53     printgraph(formation[i]);
54     cout << "Vertex: " << i << " to " << formation[i] << " : ";
55 }
56
57 void pathformation() {
58     int src;
59     cout << "From which node do you want to start search? ";
60     int arc;
61     cin >> arc;
62     cout << "Your choice is accepted..." << endl;
63
64     for (int i = 0; i < N; i++) {
65         formation[i] = -1;
66         path[i] = 10000;
67         check[i] = false;
68     }
69     path[src] = 0;
70     for (int count = 0; count < N - 1; count++) {
71         int j = pathmin();
72         check[j] = true;
73         for (int v = 0; v < N; v++) {
74             if ((!check[v] && graph[j][v]) &&
75                 path[j] + graph[j][v] < path[v]) {
76                 formation[v] = j;
77                 path[v] = path[j] + graph[j][v];
78             }
79         }
80     }
81
82     cout << "Finding the least way..." << endl;
83     cout << "Counting the weight of the least way..." << endl;
84     cout << "Eureka! " << endl;
85     cout << "Here is the most optimal way: " << endl;
86     cout << "Vertex: " << src << " to " << endl;
87     printgraph(formation[src]);
88     cout << endl;
89     cout << "The weight of way: " << endl;
90     cout << path[src] << endl;
91 }
92
93 int main()
94 {
95     makegrao();
96     pathformation();
97     return 0;
98 }

```

```

99     cout << "Finding the least way..." << endl;
100     cout << "Counting the weight of the least way..." << endl;
101     cout << "Eureka! " << endl;
102     cout << "Here is the most optimal way: " << endl;
103     cout << "Vertex: " << src << " to " << endl;
104     printgraph(formation[src]);
105     cout << endl;
106     cout << "The weight of way: " << endl;
107     cout << path[src] << endl;
108 }
109
110 int main()
111 {
112     makegrao();
113     pathformation();
114     return 0;
115 }

```

```

Enter the number of nodes: 30
Enter the number of columns: 3
Enter the number of rows: 5
Verticle # 1 - - verticle # 2: 5
Verticle # 1 - - verticle # 7: 4
Verticle # 2 - - verticle # 3: 7
Verticle # 2 - - verticle # 8: 3
Verticle # 3 - - verticle # 4: 7
Verticle # 3 - - verticle # 9: 3
Verticle # 4 - - verticle # 5: 3
Verticle # 4 - - verticle # 10: 7
Verticle # 5 - - verticle # 6: 3
Verticle # 5 - - verticle # 11: 3
Verticle # 6 - - verticle # 7: 8
Verticle # 6 - - verticle # 12: 7
Verticle # 7 - - verticle # 8: 2
Verticle # 7 - - verticle # 13: 3
Verticle # 8 - - verticle # 9: 7
Verticle # 8 - - verticle # 14: 7
Verticle # 9 - - verticle # 10: 4
Verticle # 9 - - verticle # 15: 3
Verticle # 10 - - verticle # 11: 2
Verticle # 10 - - verticle # 16: 4
Verticle # 11 - - verticle # 12: 4
Verticle # 11 - - verticle # 17: 7
Verticle # 12 - - verticle # 13: 8
Verticle # 12 - - verticle # 18: 7
Verticle # 13 - - verticle # 14: 7
Verticle # 13 - - verticle # 19: 5
Verticle # 14 - - verticle # 15: 7
Verticle # 14 - - verticle # 20: 7
Verticle # 15 - - verticle # 16: 2
Verticle # 15 - - verticle # 21: 7
Verticle # 16 - - verticle # 17: 3

```

```

Verticle # 10 - - verticle # 16: 7
Verticle # 11 - - verticle # 12: 4
Verticle # 11 - - verticle # 17: 7
Verticle # 12 - - verticle # 13: 8
Verticle # 12 - - verticle # 18: 7
Verticle # 13 - - verticle # 14: 7
Verticle # 13 - - verticle # 19: 5
Verticle # 14 - - verticle # 15: 7
Verticle # 14 - - verticle # 20: 7
Verticle # 15 - - verticle # 16: 2
Verticle # 15 - - verticle # 21: 7
Verticle # 16 - - verticle # 17: 3
Verticle # 16 - - verticle # 22: 7
Verticle # 17 - - verticle # 18: 7
Verticle # 17 - - verticle # 23: 3
Verticle # 18 - - verticle # 19: 8
Verticle # 18 - - verticle # 24: 8
Verticle # 19 - - verticle # 20: 7
Verticle # 19 - - verticle # 25: 8
Verticle # 20 - - verticle # 21: 3
Verticle # 20 - - verticle # 26: 2
Verticle # 21 - - verticle # 22: 7
Verticle # 21 - - verticle # 27: 7
Verticle # 22 - - verticle # 23: 3
Verticle # 22 - - verticle # 28: 3
Verticle # 23 - - verticle # 24: 5
Verticle # 23 - - verticle # 29: 3
Verticle # 24 - - verticle # 25: 8
Verticle # 24 - - verticle # 30: 7
Verticle # 25 - - verticle # 26: 4
Verticle # 26 - - verticle # 27: 7
Verticle # 27 - - verticle # 28: 7
Verticle # 28 - - verticle # 29: 3
Verticle # 29 - - verticle # 30: 8

```

```

Verticle # 19 - - verticle # 25: 8
Verticle # 20 - - verticle # 21: 3
Verticle # 20 - - verticle # 26: 2
Verticle # 21 - - verticle # 22: 7
Verticle # 21 - - verticle # 27: 7
Verticle # 22 - - verticle # 23: 3
Verticle # 22 - - verticle # 28: 3
Verticle # 23 - - verticle # 24: 5
Verticle # 23 - - verticle # 29: 3
Verticle # 24 - - verticle # 25: 8
Verticle # 24 - - verticle # 30: 7
Verticle # 25 - - verticle # 26: 4
Verticle # 26 - - verticle # 27: 7
Verticle # 27 - - verticle # 28: 7
Verticle # 28 - - verticle # 29: 3
Verticle # 29 - - verticle # 30: 8
From which node do you want to start search? 1
Your choice is accepted...
Finding the least way...
Counting the weight of the least way...
Eureka!
Here is the most optimal way
Verticle #1 -> Verticle #7 -> Verticle #8 -> Verticle #14 -> Verticle #15 -> Verticle #21 -> Verticle #23 -> Verticle #29 -> Verticle #29 -> Verticle #30 ->
The weight of way: 22

```

5. «Іди в найближчий» для розв'язання задачі комівояжера

```
1 #include <iostream>
2 #include <string>
3 #include <string>
4 #include <fstream>
5
6 using namespace std;
7 ifstream f1;
8 string path = "MyFile.txt";
9
10 struct Array
11 {
12     int size(100);
13     int *arr;
14
15     void Read() {
16         int check = 0;
17         string str;
18         str = "";
19
20         f1.open(path);
21         int n = 0;
22         arr = new int[check];
23         for (int i = 0; i < check; i++)
24             arr[i] = new int[check];
25
26         for (int i = 0; i < check; i++)
27         {
28             for (int j = 0; j < check; j++)
29                 arr[i][j] = 0;
30
31             for (int j = i + 1; j < check; j++)
32             {
33                 getline(f1, str, '\n');
34                 arr[i][j] = atoi(str.c_str());
35                 arr[j][i] = atoi(str.c_str());
36             }
37         }
38         f1.close();
39
40         return arr;
41     }
42 }
```

```
43
44 bool WhichWay(int *arr, int count)
45 {
46     int mas = new int[count];
47
48     for (int i = 0; i < count; i++)
49     {
50         mas[i] = arr[i][i];
51     }
52
53     for (int i = 0; i < count; i++)
54     {
55         if (mas[i] != arr[i][i])
56         {
57             return true;
58         }
59         else
60         {
61             continue;
62         }
63     }
64     return false;
65 }
66
67 bool CycleMatrix(int *mas, int size)
68 {
69     bool k = true;
70
71     for (int i = 0; i < size; i++)
72     {
73         for (int j = 0; j < size; j++)
74         {
75             if (mas[i] == mas[j] && i != j)
76             {
77                 return false;
78             }
79         }
80     }
81
82     return true;
83 }
84
85 int waylines(int *arr, int count)
```

```
86 {
87     int count = 0;
88
89     for (int i = 0; i < count; i++)
90     {
91         count += arr[i][i] - 1;
92     }
93
94     count += arr[count][count] - 1;
95     return count;
96 }
97
98 int main() {
99     int count = 0;
100     int *arr;
101     arr = Read();
102     int var = count + 1;
103     bool k = true;
104     int mas = new int[count];
105
106     int min = 1000;
107     int long = 0;
108
109     int n = 0;
110
111     for (int i = 0; i < count; i++)
112     {
113         mas[i] = 1;
114         minmas[i] = 1;
115     }
116
117     while (WhichWay(mas, count))
118     {
119         while (mas[var] != count)
120         {
121             mas[var]++;
122
123             if (CycleMatrix(mas, count))
124             {
125                 long = way(arr, mas);
126             }
127 }
```

```
128
129     if (CycleMatrix(mas, count))
130     {
131         long = way(arr, mas);
132
133         for (int i = 0; i < count; i++)
134         {
135             count += mas[i] - 1;
136         }
137         count += mas[count] - 1;
138         count += endl;
139
140         if (long < min)
141         {
142             min = long;
143             n = i;
144         }
145         if (long == min)
146         {
147             n++;
148         }
149     }
150
151     while (mas[var] == count)
152     {
153         mas[var] = 1;
154         var++;
155     }
156     mas[var] = 1;
157
158     if (CycleMatrix(mas, count))
159     {
160         for (int i = 0; i < count; i++)
161         {
162             count += mas[i] - 1;
163         }
164         count += mas[count] - 1;
165         count += endl;
166
167         long = way(arr, mas);
168     }
```

```

165         mas = zero;
166         m = 1;
167     }
168     if (long == min)
169     {
170         m++;
171     }
172 }
173 var = count - 1;
174 }
175
176 for (int i = 0; i < count; i++)
177 {
178     mas[i] = 1;
179     minmas[i] = 1;
180 }
181 Array *rez = new Array[a];
182 int iter = 0;
183
184 while (whichway(mas, count))
185 {
186     while (mas[var] != count)
187     {
188         mas[var]++;
189
190         if (CycleMatrix(mas, count))
191         {
192             long = way(arr, mas);
193
194             if (long == min)
195             {
196                 for (int i = 0; i < count; i++)
197                 {
198                     rez[iter].papu[i] = mas[i];
199                 }
200                 rez[iter].papu[count] = mas[0];
201                 iter++;
202             }
203         }
204     }
205     while (mas[var] == count)
206     {
207         mas[var] = 1;

```

```

1         {
2             for (int i = 0; i < count; i++)
3             {
4                 rez[iter].papu[i] = mas[i];
5             }
6             rez[iter].papu[count] = mas[0];
7             iter++;
8         }
9     }
10    var = count - 1;
11 }
12
13 cout << "Вихід: " << endl;
14
15 for (int i = 0; i < iter - 1; i++)
16 {
17     for (int j = 0; j < count; j++)
18     {
19         if (j != 0)
20         {
21             cout << "→ ";
22         }
23         cout << rez[i].papu[j] << " ";
24     }
25     cout << endl;
26     if (i == 0)
27     {
28         for (int j = 0; j < count; j++)
29         {
30             if (j != 0)
31             {
32                 cout << "→ ";
33             }
34             cout << rez[i].papu[count - j] << " ";
35         }
36
37         cout << endl;
38     }
39 }
40 cout << "Мінімальна вага: " << min;

```

Шляхи:

```

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

```

Мінімальна вага: 15

```

8→ 7→ 6→ 5→ 1→ 2→ 4→ 3→ 8 (28)
8→ 7→ 6→ 5→ 1→ 3→ 2→ 4→ 8 (28)
8→ 7→ 6→ 5→ 1→ 3→ 4→ 2→ 8 (27)
8→ 7→ 6→ 5→ 1→ 4→ 2→ 3→ 8 (27)
8→ 7→ 6→ 5→ 1→ 4→ 3→ 2→ 8 (31)
8→ 7→ 6→ 5→ 2→ 1→ 3→ 4→ 8 (30)
8→ 7→ 6→ 5→ 2→ 1→ 4→ 3→ 8 (29)
8→ 7→ 6→ 5→ 2→ 3→ 1→ 4→ 8 (29)
8→ 7→ 6→ 5→ 2→ 3→ 4→ 1→ 8 (29)
8→ 7→ 6→ 5→ 2→ 4→ 1→ 3→ 8 (24)
8→ 7→ 6→ 5→ 2→ 4→ 3→ 1→ 8 (24)
8→ 7→ 6→ 5→ 3→ 1→ 2→ 4→ 8 (29)
8→ 7→ 6→ 5→ 3→ 1→ 4→ 2→ 8 (27)
8→ 7→ 6→ 5→ 3→ 2→ 1→ 4→ 8 (33)
8→ 7→ 6→ 5→ 3→ 2→ 4→ 1→ 8 (33)
8→ 7→ 6→ 5→ 3→ 4→ 1→ 2→ 8 (32)
8→ 7→ 6→ 5→ 3→ 4→ 2→ 1→ 8 (32)
8→ 7→ 6→ 5→ 4→ 1→ 2→ 3→ 8 (32)
8→ 7→ 6→ 5→ 4→ 1→ 3→ 2→ 8 (31)
8→ 7→ 6→ 5→ 4→ 2→ 1→ 3→ 8 (28)
8→ 7→ 6→ 5→ 4→ 2→ 3→ 1→ 8 (28)
8→ 7→ 6→ 5→ 4→ 3→ 1→ 2→ 8 (32)
8→ 7→ 6→ 5→ 4→ 3→ 2→ 1→ 8 (32)

```

```

8-> 7-> 6-> 1-> 3-> 4-> 5-> 2-> 8 (30)
8-> 7-> 6-> 1-> 3-> 5-> 2-> 4-> 8 (27)
8-> 7-> 6-> 1-> 3-> 5-> 4-> 2-> 8 (29)
8-> 7-> 6-> 1-> 4-> 2-> 3-> 5-> 8 (20)
8-> 7-> 6-> 1-> 4-> 2-> 5-> 3-> 8 (26)
8-> 7-> 6-> 1-> 4-> 3-> 2-> 5-> 8 (30)
8-> 7-> 6-> 1-> 4-> 3-> 5-> 2-> 8 (30)
8-> 7-> 6-> 1-> 4-> 5-> 2-> 3-> 8 (30)
8-> 7-> 6-> 1-> 4-> 5-> 3-> 2-> 8 (33)
8-> 7-> 6-> 1-> 5-> 2-> 3-> 4-> 8 (31)
8-> 7-> 6-> 1-> 5-> 2-> 4-> 3-> 8 (26)
8-> 7-> 6-> 1-> 5-> 3-> 2-> 4-> 8 (30)
8-> 7-> 6-> 1-> 5-> 3-> 4-> 2-> 8 (29)
8-> 7-> 6-> 1-> 5-> 4-> 2-> 3-> 8 (29)
8-> 7-> 6-> 1-> 5-> 4-> 3-> 2-> 8 (33)
8-> 7-> 6-> 2-> 1-> 3-> 4-> 5-> 8 (31)
8-> 7-> 6-> 2-> 1-> 3-> 5-> 4-> 8 (32)
8-> 7-> 6-> 2-> 1-> 4-> 3-> 5-> 8 (31)
8-> 7-> 6-> 2-> 1-> 4-> 5-> 3-> 8 (31)
8-> 7-> 6-> 2-> 1-> 5-> 3-> 4-> 8 (32)
8-> 7-> 6-> 2-> 1-> 5-> 4-> 3-> 8 (31)
8-> 7-> 6-> 2-> 3-> 1-> 4-> 5-> 8 (30)
8-> 7-> 6-> 2-> 3-> 4-> 1-> 5-> 8 (30)
8-> 7-> 6-> 2-> 3-> 5-> 1-> 4-> 8 (31)
8-> 7-> 6-> 2-> 3-> 5-> 4-> 1-> 8 (31)
8-> 7-> 6-> 2-> 4-> 1-> 3-> 5-> 8 (26)
8-> 7-> 6-> 2-> 4-> 1-> 5-> 3-> 8 (26)
8-> 7-> 6-> 2-> 4-> 3-> 1-> 5-> 8 (26)

```

```

7-> 4-> 5-> 3-> 2-> 8-> 6-> 1-> 7 (34)
7-> 4-> 5-> 3-> 6-> 1-> 2-> 8-> 7 (37)
7-> 4-> 5-> 3-> 6-> 1-> 8-> 2-> 7 (34)
7-> 4-> 5-> 3-> 6-> 2-> 1-> 8-> 7 (35)
7-> 4-> 5-> 3-> 6-> 2-> 8-> 1-> 7 (35)
7-> 4-> 5-> 3-> 6-> 8-> 1-> 2-> 7 (30)
7-> 4-> 5-> 3-> 6-> 8-> 2-> 1-> 7 (30)
7-> 4-> 5-> 3-> 8-> 1-> 2-> 6-> 7 (32)
7-> 4-> 5-> 3-> 8-> 1-> 6-> 2-> 7 (33)
7-> 4-> 5-> 3-> 8-> 2-> 1-> 6-> 7 (34)
7-> 4-> 5-> 3-> 8-> 2-> 6-> 1-> 7 (34)
7-> 4-> 5-> 3-> 8-> 6-> 1-> 2-> 7 (31)
7-> 4-> 5-> 3-> 8-> 6-> 2-> 1-> 7 (31)
7-> 4-> 5-> 6-> 1-> 2-> 3-> 8-> 7 (37)
7-> 4-> 5-> 6-> 1-> 2-> 8-> 3-> 7 (32)
7-> 4-> 5-> 6-> 1-> 3-> 2-> 8-> 7 (36)
7-> 4-> 5-> 6-> 1-> 3-> 8-> 2-> 7 (33)
7-> 4-> 5-> 6-> 1-> 8-> 2-> 3-> 7 (32)
7-> 4-> 5-> 6-> 1-> 8-> 3-> 2-> 7 (34)
7-> 4-> 5-> 6-> 2-> 1-> 3-> 8-> 7 (34)
7-> 4-> 5-> 6-> 2-> 1-> 8-> 3-> 7 (30)
7-> 4-> 5-> 6-> 2-> 3-> 1-> 8-> 7 (34)
7-> 4-> 5-> 6-> 2-> 3-> 8-> 1-> 7 (34)

```

6. Алгоритм Флері та елементарних циклів знаходження ейлерового ланцюга в ейлеровому графі.

6. Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.

```
#include <iostream>
#include <string.h>
#include <vector>
#include <list>
using namespace std;

// A class that represents an undirected graph
class Graph
{
    int V; // No. of vertices
    list<int> *adj; // A dynamic array of adjacency lists
public:
    // Constructor and destructor
    Graph(int V) { this->V = V; adj = new list<int>[V]; }
    ~Graph() { delete [] adj; }

    // Functions to add and remove edge
    void addEdge(int u, int v) { adj[u].push_back(v); adj[v].push_back(u); }
    void removeEdge(int u, int v);

    // Methods to print Eulerian tour
    void printEulerTour();
    void printEulerUtil(int v);

    // This function returns count of vertices reachable from v. It does DFS
    int DFSCount(int v, bool visited[]);

    // Utility function to check if edge u-v is a valid next edge in
    // Eulerian trail or not
    bool isValidNextEdge(int u, int v);
};

// The main function that print Eulerian Trail. It first finds an odd
// degree vertex (if there is any) and then calls printEulerUtil()
// to print the path v
void Graph::printEulerTour()
{
    // Find a vertex with odd degree
    int u = 0;
    for (int i = 0; i < V; i++)
        if (adj[i].size() % 2 != 0)
            u = i; break;

    // Print tour starting from edge
    printEulerUtil(u);
    cout << endl;
}
```

```
// 3.4) If count1 is greater, then edge (u, v) is a bridge
return (count1 > count2) ? false : true;

// This function removes edge u-v from graph. It removes the edge by
// replacing adjacent vertex value with -1.
void Graph::removeEdge(int u, int v)
{
    // Find u in adjacency list of v and replace it with -1
    list<int>::iterator it = find(adj[v].begin(), adj[v].end(), u);
    *it = -1;

    // Find v in adjacency list of u and replace it with -1
    list<int>::iterator it = find(adj[u].begin(), adj[u].end(), v);
    *it = -1;
}

// A DFS based function to count reachable vertices from v
int Graph::DFSCount(int v, bool visited[])
{
    // mark the current node as visited
    visited[v] = true;
    int count = 1;

    // recur for all vertices adjacent to this vertex
    list<int>::iterator it;
    for (it = adj[v].begin(); it != adj[v].end(); ++it)
        if (*it != -1 && !visited[*it])
            count += DFSCount(*it, visited);

    return count;
}
```

```
// Print Euler tour starting from vertex u
void Graph::printEulerUtil(int u)
{
    // recur for all the vertices adjacent to this vertex
    list<int>::iterator it;
    for (it = adj[u].begin(); it != adj[u].end(); ++it)
    {
        int v = *it;

        // If edge u-v is not removed and it's a valid next edge
        if (*it != -1 && isValidNextEdge(u, v))
        {
            cout << u << " " << v << " ";
            removeEdge(u, v);
            printEulerUtil(v);
        }
    }

    // The function to check if edge u-v can be considered as next edge in
    // Euler trail
    bool Graph::isValidNextEdge(int u, int v)
    {
        // the edge u-v is valid in one of the following two cases:

        // 1) If v is the only adjacent vertex of u
        int count = 0; // to store count of adjacent vertices
        list<int>::iterator it;
        for (it = adj[u].begin(); it != adj[u].end(); ++it)
            if (*it != -1)
                count++;
        if (count == 1)
            return true;

        // 2) If there are multiple adjacents, then u-v is not a bridge
        // in following steps to check if u-v is a bridge

        // 2.a) count of vertices reachable from u
        bool visited[] = {false, ...}; // for V
        int count1 = DFSCount(u, visited);

        // 2.b) Remove edge (u, v) and after removing the edge, count
        // of vertices reachable from u
        removeEdge(u, v);
        bool visited[] = {false, ...}; // for V
        int count2 = DFSCount(u, visited);

        // 2.c) Add the edge back to the graph
        addEdge(u, v);
    }
}
```

```
int main()
{
    Graph g(15);
    g.addEdge(0, 1); g.addEdge(0, 2);
    g.addEdge(0, 3); g.addEdge(0, 4);
    g.addEdge(0, 5); g.addEdge(0, 6);
    g.addEdge(0, 7); g.addEdge(0, 8);
    g.addEdge(0, 9); g.addEdge(0, 10);
    g.addEdge(0, 11); g.addEdge(0, 12);
    g.addEdge(0, 13); g.addEdge(0, 14);
    g.addEdge(1, 2); g.addEdge(1, 3);
    g.addEdge(1, 4); g.addEdge(1, 5);
    g.addEdge(1, 6); g.addEdge(1, 7);
    g.addEdge(1, 8); g.addEdge(1, 9);
    g.addEdge(1, 10); g.addEdge(1, 11);
    g.addEdge(1, 12); g.addEdge(1, 13);
    g.addEdge(1, 14); g.addEdge(1, 15);
    g.printEulerTour();
    return 0;
}
```

2-1 1-11 13-10 10-1 3-2 2-9 9-6 6-7 7-3 3-4 4-7 7-5 5-3 3-8 8-7 7-9 9-8 8-10 10-6 9-11 11-11 12-11 13-11 11-10 10-3
Process finished with exit code 0