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.

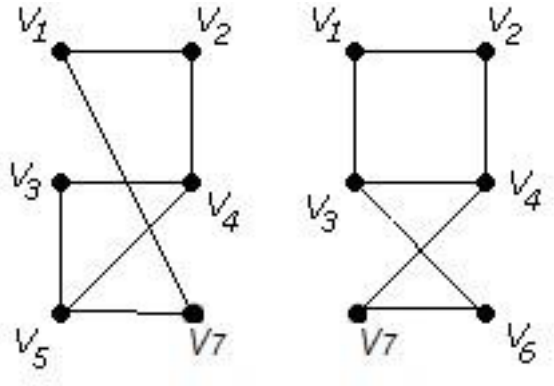
Львів-2019р.

Індивідуальні завдання

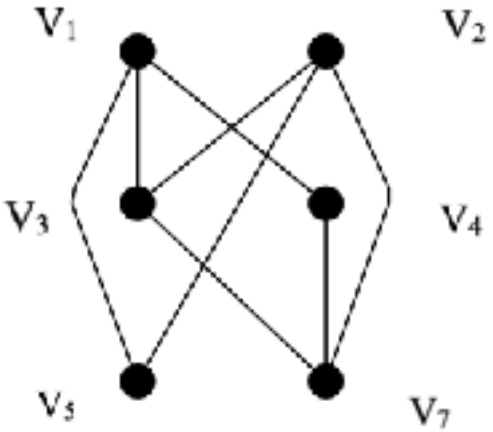
Завдання № 1.

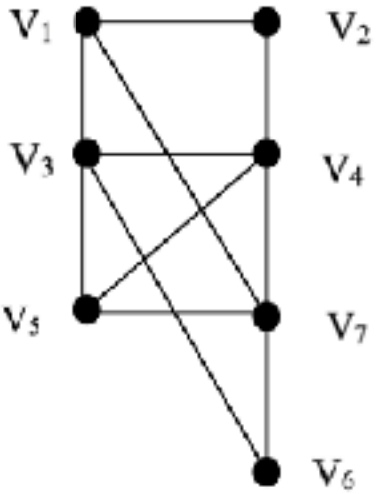
Виконати наступні операції над графами

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 А - що скадається з 3-х вершин в G_1;
6) добуток графів.

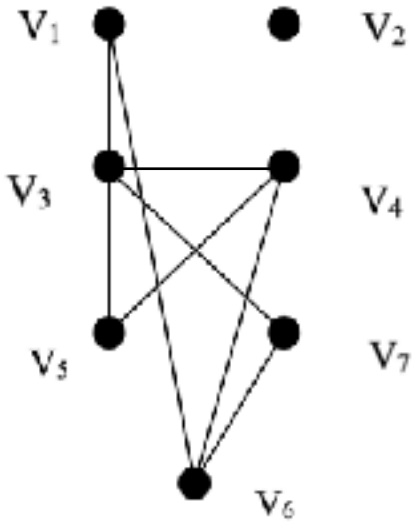


1) Доповнення до першого графу: 2) Об’єднання графів:

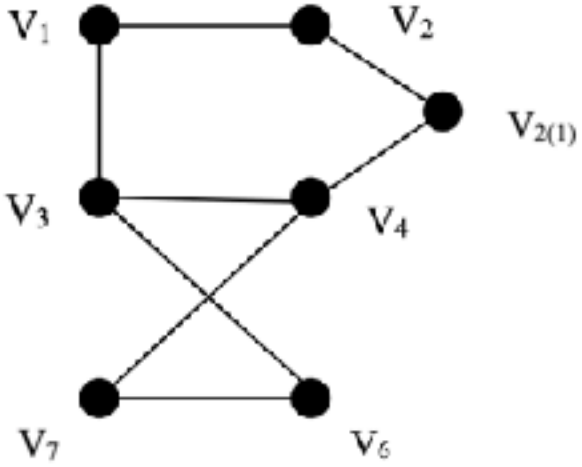




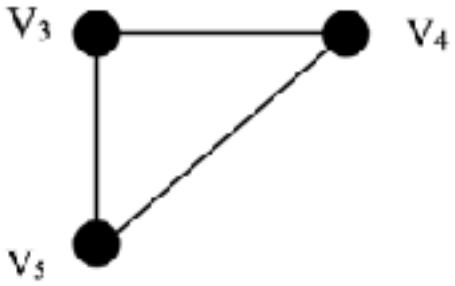
3) Кільцева сума:



4) Розмноження вершини у другому графі:



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



6) Добуток графів:

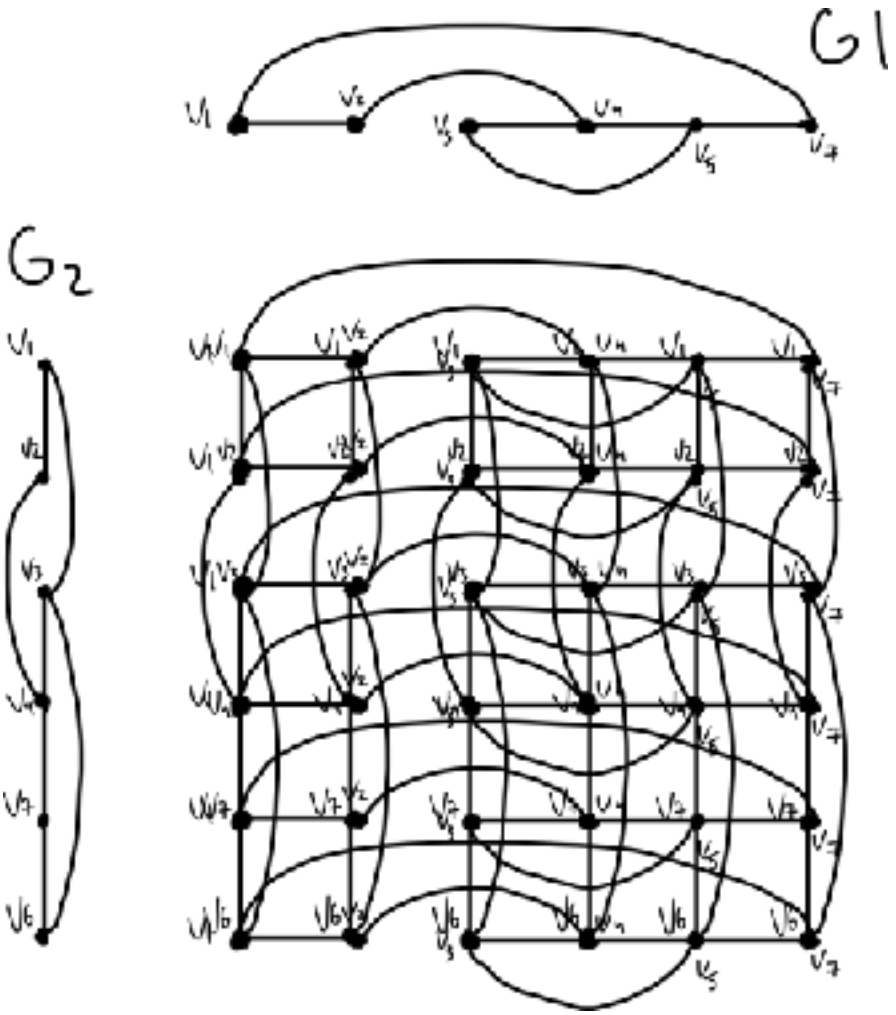
Добуток графів матиме 36 вершин, оскільки кожен із графів містить по 6 вершин.

$G = G_1 \times G_2$

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

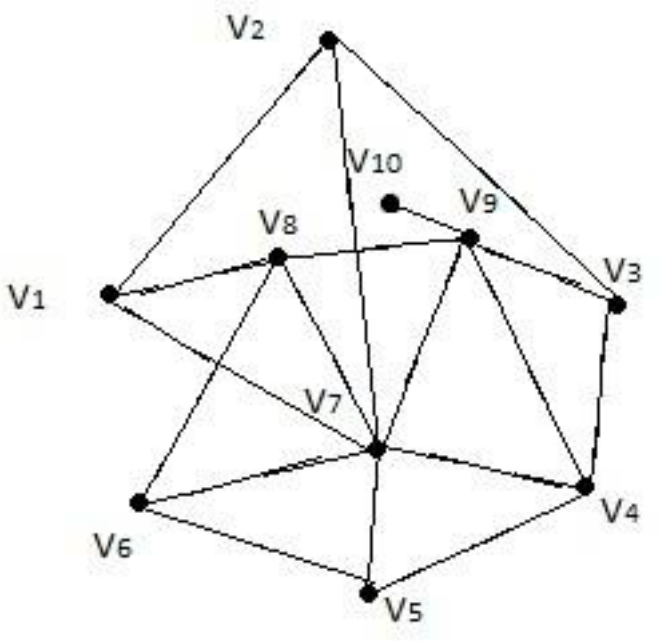
V(G) = { (1, 1), (1, 2), (1, 3), (1, 4), (1, 7), (1, 6), (2, 1), (2, 2), (2, 3), (2, 4), (2, 7), (2, 6), (3, 1), (3, 2), (3, 3), (3, 4), (3, 7), (3, 6), (4, 1), (4, 2), (4, 3), (4, 4), (4, 7), (4, 6), (5, 1), (5, 2), (5, 3), (5, 4), (5, 7), (5, 6), (6, 1), (6, 2), (6, 3), (6, 4), (6, 7), (6, 6), (7,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

Отже, добуток графів буде виглядати так:

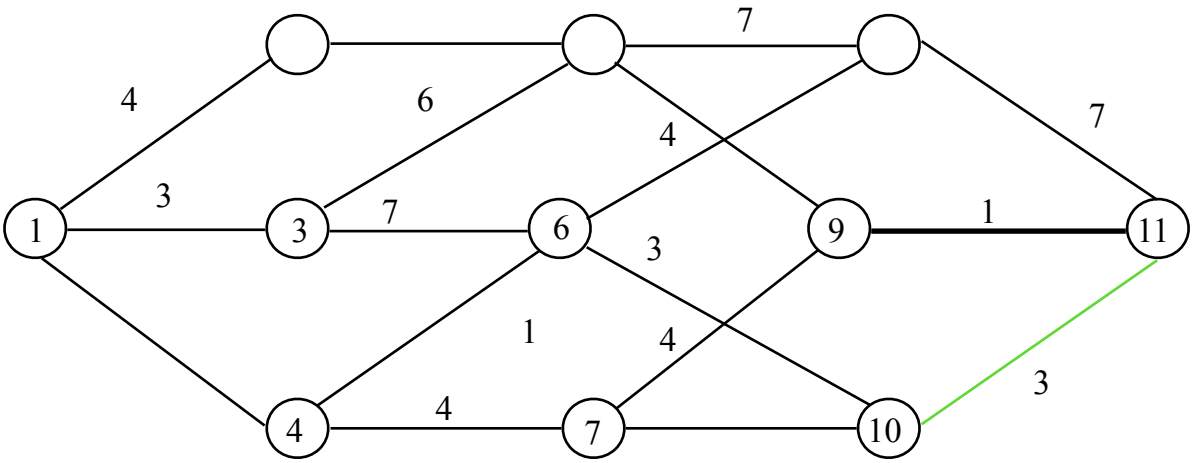


Завдання № 2.

Скласти таблицю суміжності для орграфа.



	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}
V_1	0	1	0	0	0	0	1	1	0	0
V_2	1	0	1	0	0	0	1	0	0	0
V_3	0	1	0	1	0	0	0	0	1	0
V_4	0	0	1	0	1	0	1	0	1	0
V_5	0	0	0	1	0	1	1	0	0	0
V_6	0	0	0	0	1	0	1	1	0	0
V_7	1	1	0	1	1	1	0	1	0	0
V_8	1	0	0	0	0	1	1	0	1	0
V_9	0	0	1	1	0	0	1	1	0	1
V_{10}	0	0	0	0	0	0	0	0	1	0


7
4
6
4
7
3
7
1
1
3
4
4
4
7
10
3
1
3
6
9
11

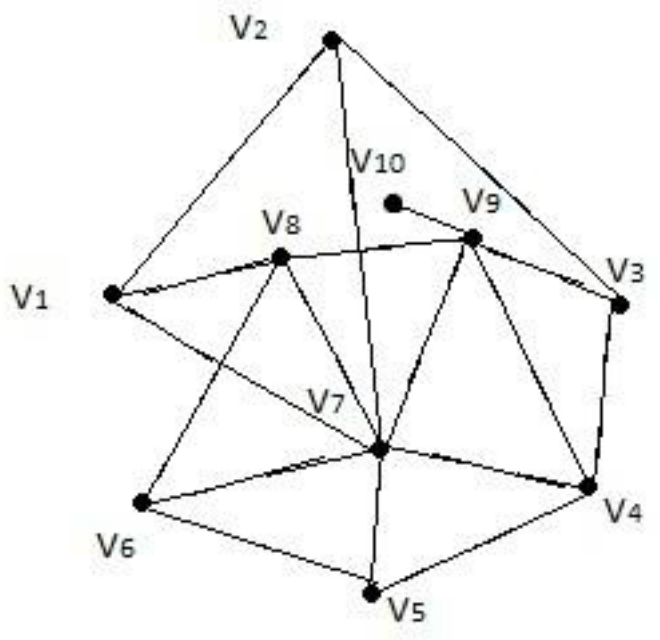
Завдання № 3.

Для графа з другого завданння 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}).

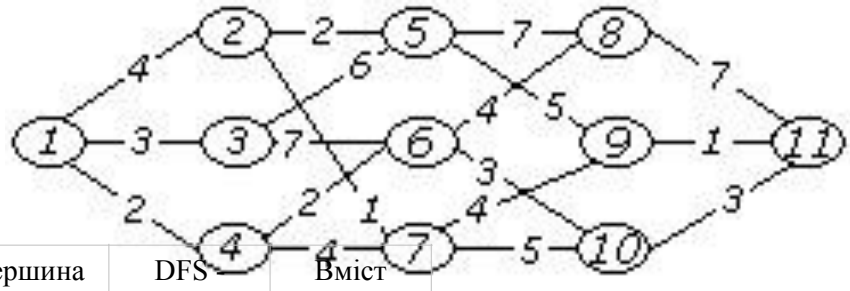
Завдання № 4.

Для графа з другого завдання виконати обхід дерева вглиб.



Завдання № 5.

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

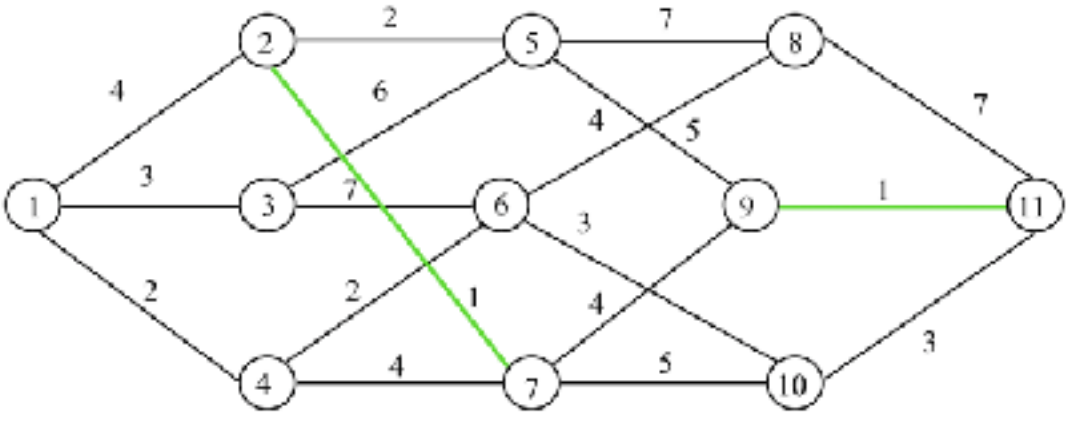


Вершина	DFS номер	Вміст стеку
1	1	1
2	2	12
3	3	123
9	4	1239
10	5	123910
-	-	1239
8	6	12398
7	7	123987
6	8	1239876
5	9	12398765
4	10	123987654
-	-	12398765
-	-	1239876
-	-	123987
-	-	12398
-	-	1239
-	-	123
-	-	12
-	-	1
-	-	∅

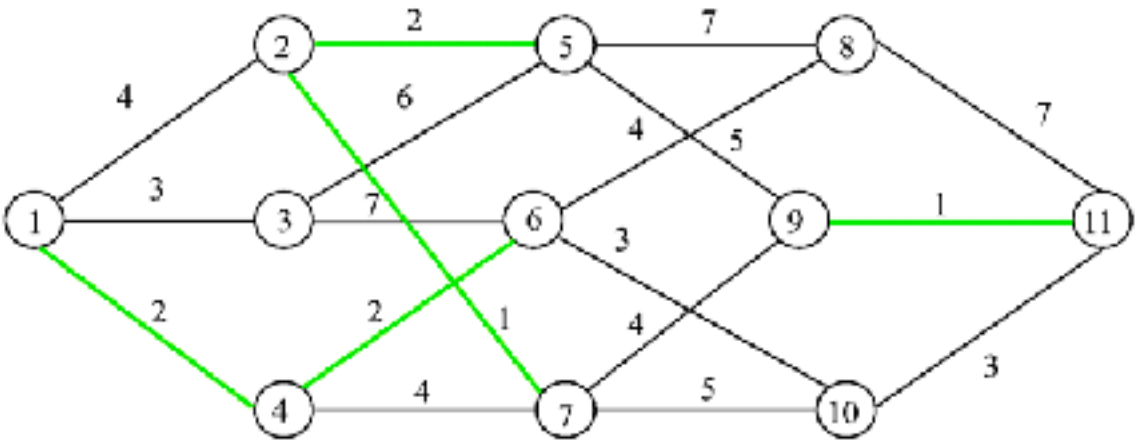
- **Метод Краскала**:

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;

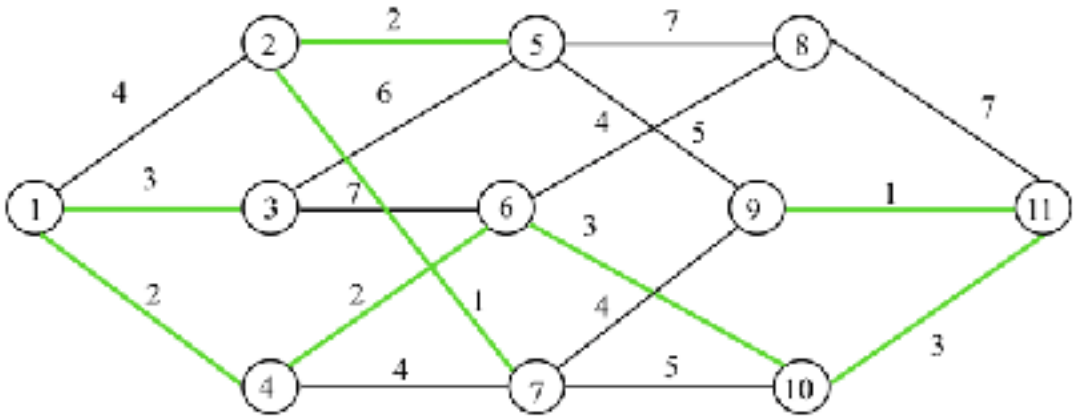
вибираємо ребро графа з найменшою вагою і додаємо до дерева. В даному випадку це ребра (2, 7) і (9, 11) з вагою 1.



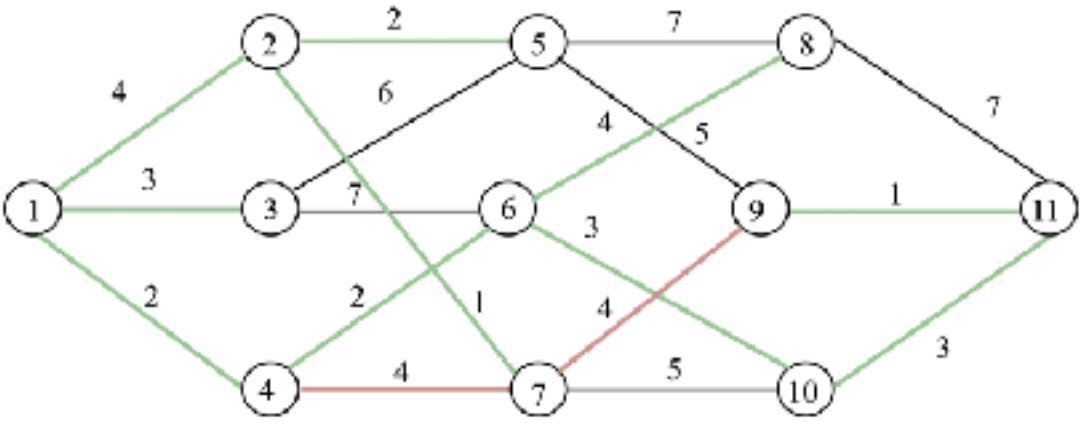
беремо ребро з наступною найменшою вагою. Це ребра (4, 6), (1, 4), (2, 5) з вагою 2. Додаємо до дерева.



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

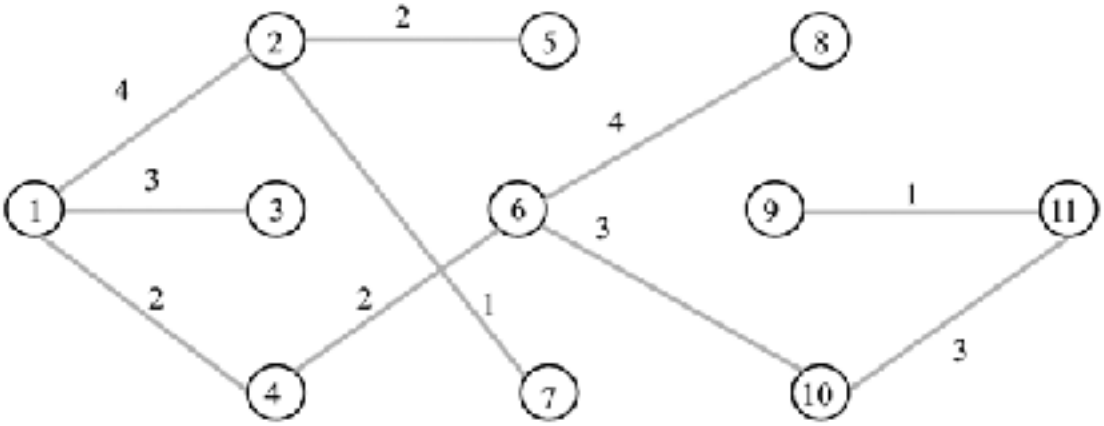


беремо ребра з наступною найменшою вагою - (1, 2), (4, 7), (6, 8), (7, 9) з вагою 4. Відкидаємо ребра (4, 7) і (7, 9), оскільки вони утворять цикл.

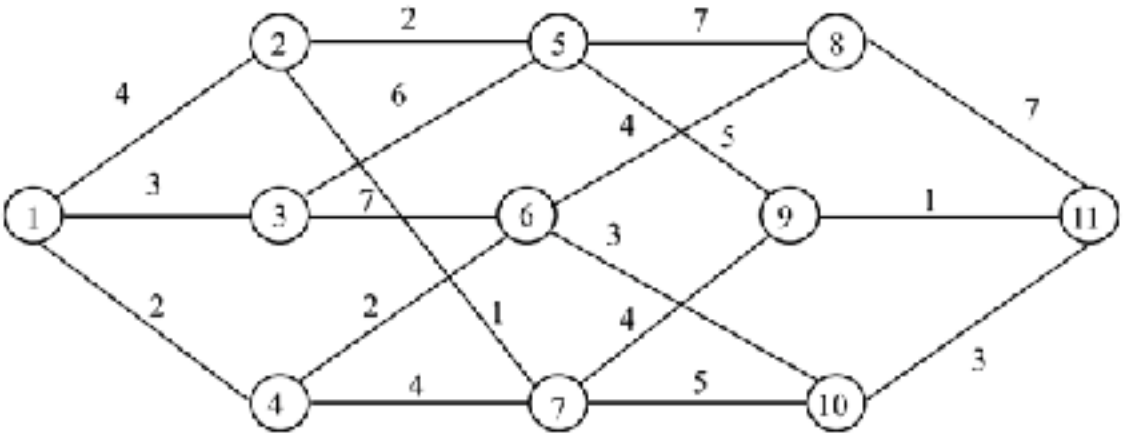


продовжуємо виконувати алгоритм за аналогією і отримуємо дерево після того як перевірили всі ребра із посоротованого списка або ж всі вершини стали з’єднаними

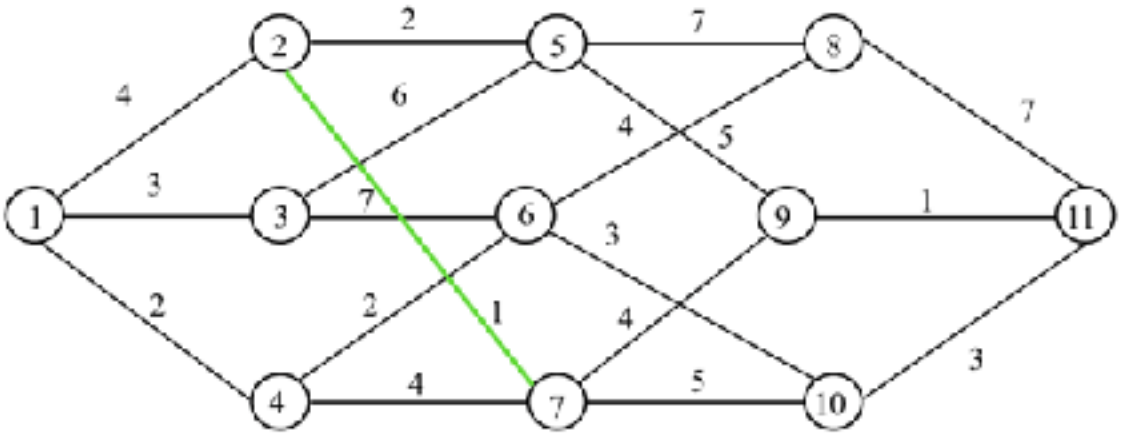
мінімальне остове дерево за алгоритмом Краскала буде виглядати так:



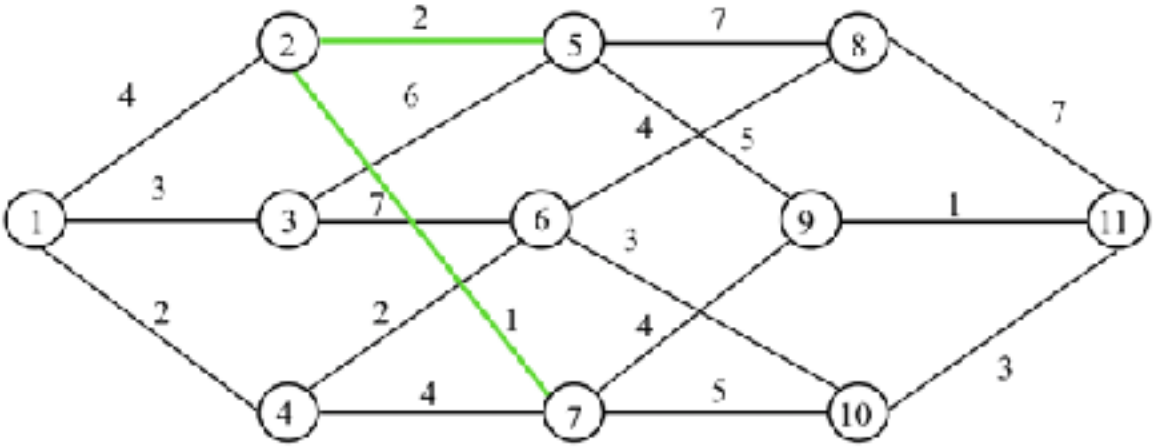
- **Метод Прима:**



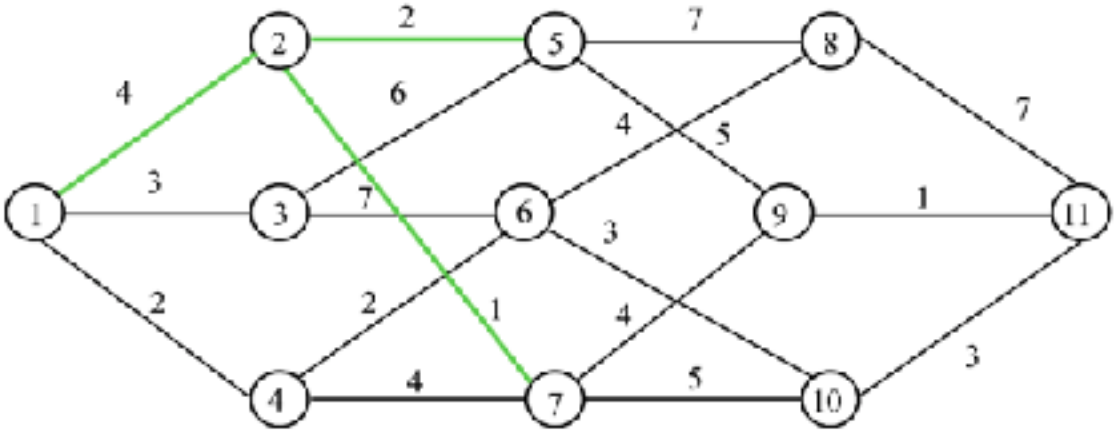
спочатку фіксуємо довільну вершину, з якої розпочнемо пошук мінімального дерева. У даному випадку це вершина 2. Знаходимо ребро, інцидентне цій вершині з найменшою вагою і це (2, 7) з вагою 1. Додаємо до кістякового дерева ребро (2, 7).



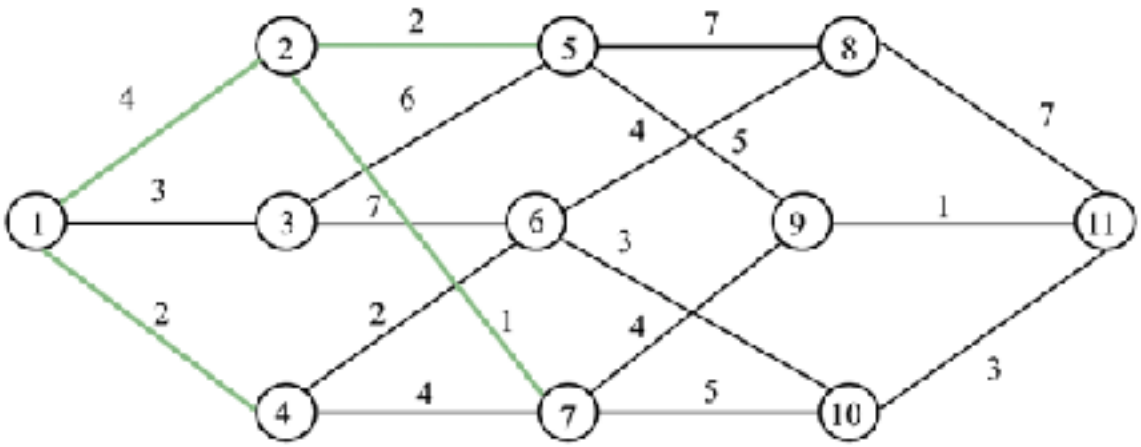
розглядаємо ребра, інцидентні вершинам 2 і 7, і вибираємо те, яке має найменшу вагу. І це (2, 5) з вагою 2. Додаємо його до кістякового дерева.



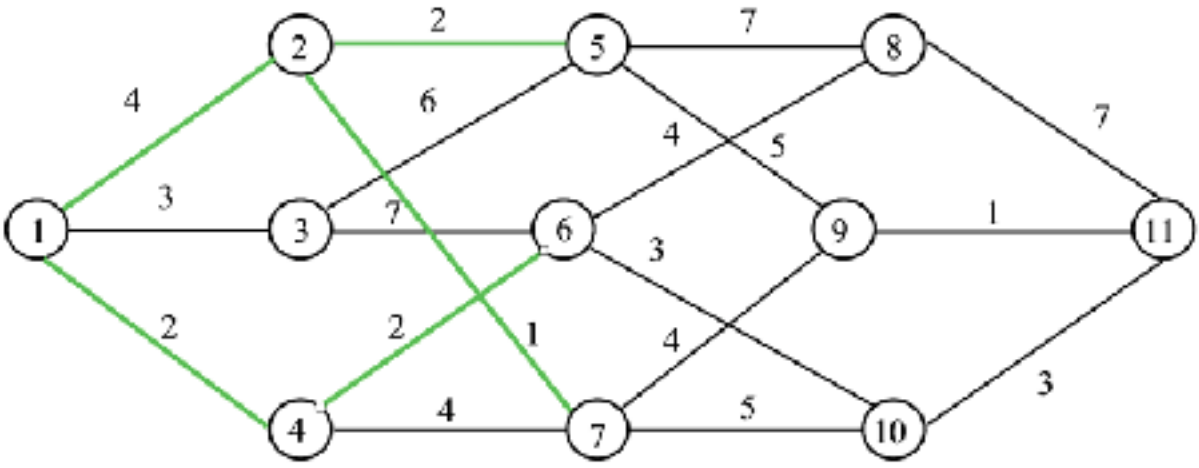
розглядаємо ребра, інцидентні вершинам 2, 7, 5 і вибираємо з найменшою вагою - і це (1, 2) з вагою 4. Додаємо до дерева.



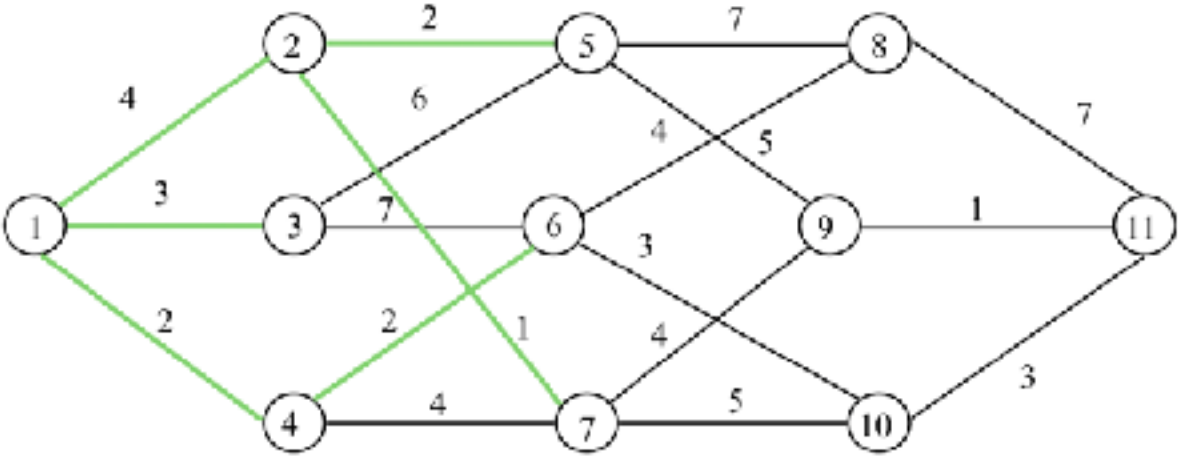
розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1 і вибираємо з найменшою вагою - і це (1, 4) з вагою 2. Додаємо до дерева.



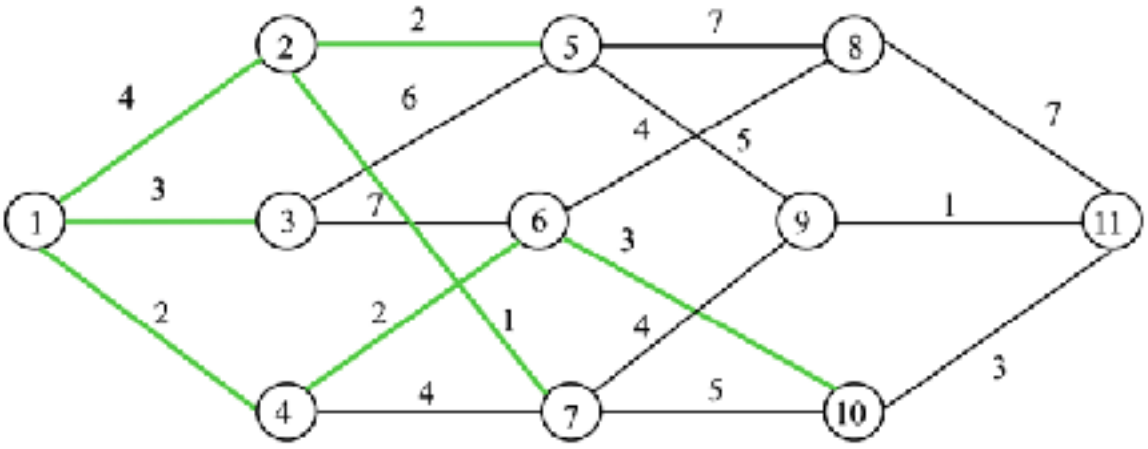
розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1, 4 і вибираємо з найменшою вагою - і це (4, 6) з вагою 2. Додаємо до кістякового дерева.



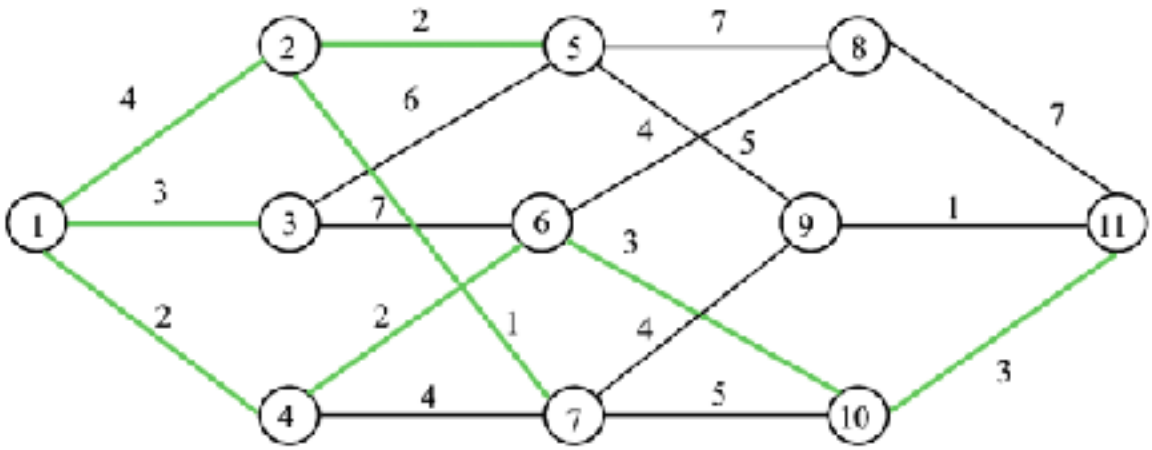
Етап 6: розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1, 4, 6 і вибираємо з найменшою вагою - і це (1, 3) з вагою 3. Додаємо до кістякового дерева.

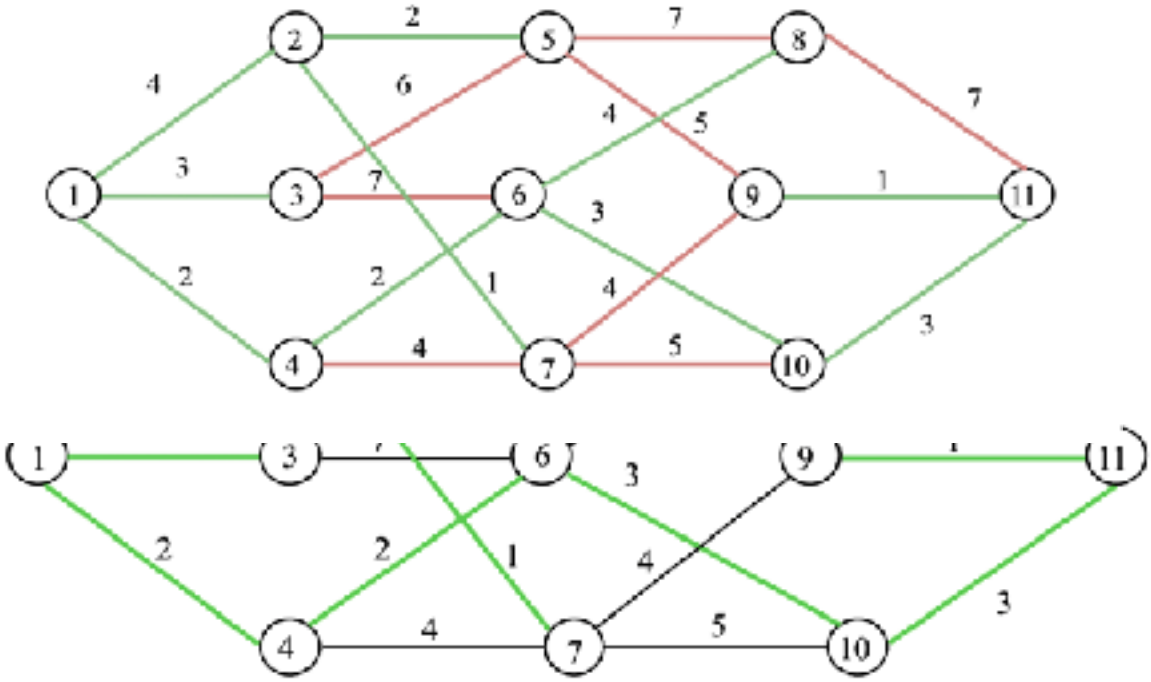


Етап 7: розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1, 4, 6, 3 і вибираємо з найменшою вагою - і це (6, 10) з вагою 3. Додаємо до кістякового дерева.



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.

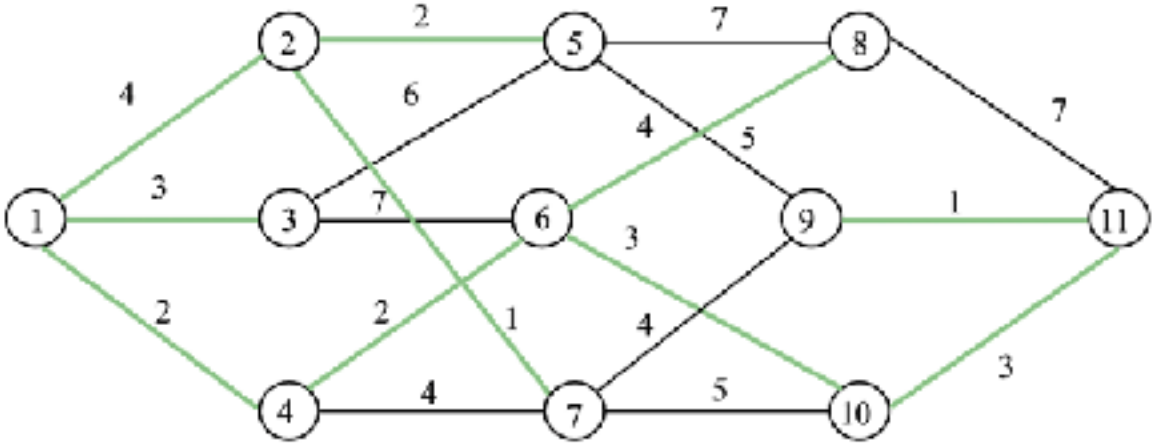




Етап 9:

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.

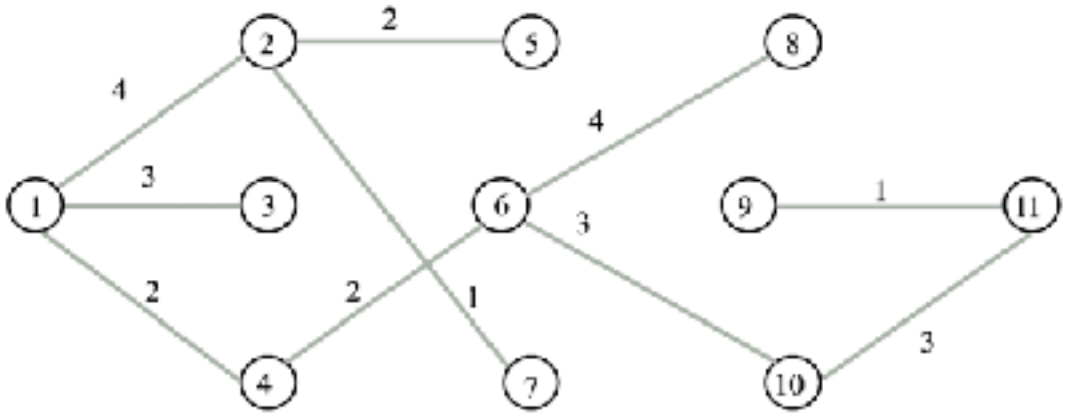
Етап 10: розглядаємо ребра, інцидентні вершинам 2, 7, 5, 1, 4, 6, 3, 10, 11 і вибираємо



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	∞

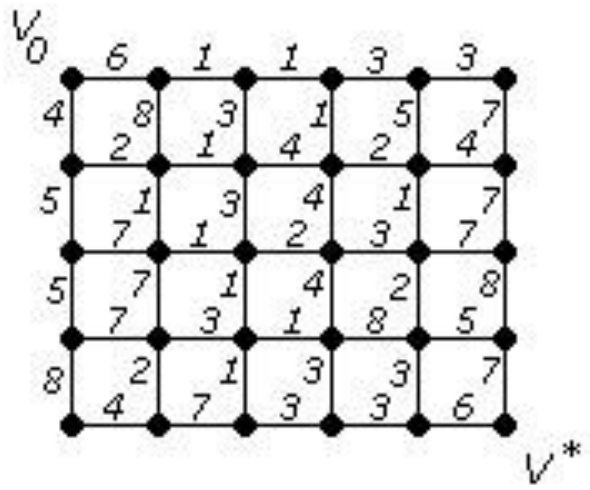
$1 \to 8 \to 6 \to 7 \to 5 \to 2 \to 4 \to 3 \to 1$
$2 + 1 + 2 + 1 + 2 + 1 + 6 + 2 = 17$

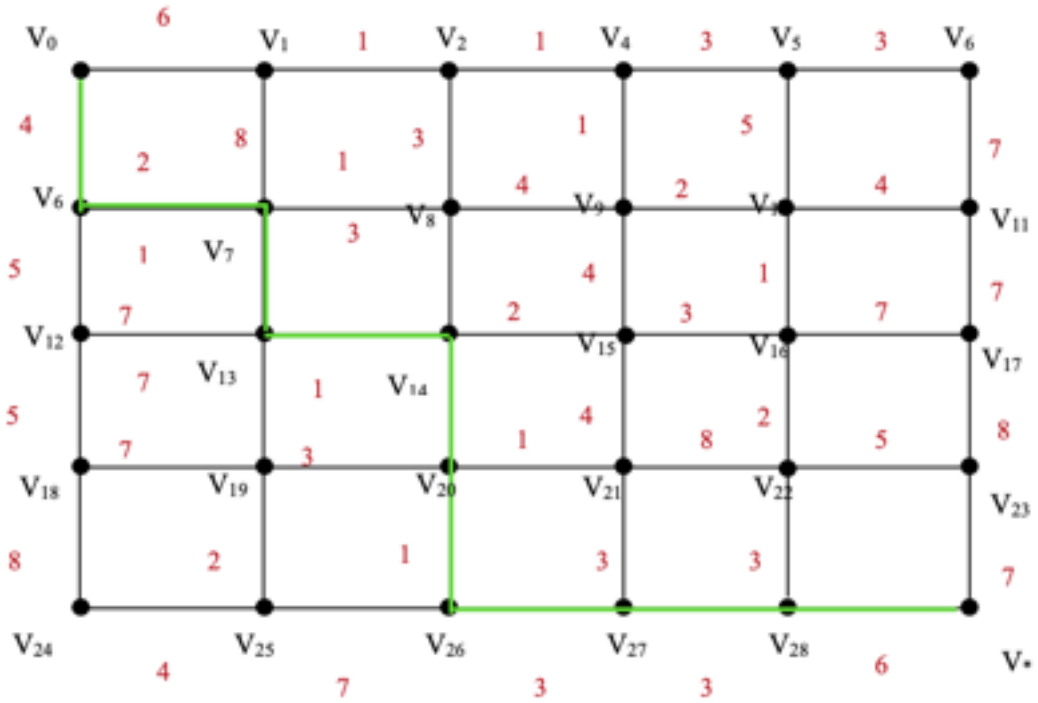
за аналогією знаходимо такі цикли:
$1 \to 2 \to 4 \to 6 \to 8 \to 7 \to 5 \to 1$ (18)
$1 \to 7 \to 5 \to 2 \to 4 \to 8 \to 6 \to 3 \to 1$ (20)
$3 \to 7 \to 5 \to 2 \to 4 \to 1 \to 8 \to 6 \to 3$ (15)
$1 \to 5 \to 7 \to 3 \to 6 \to 8 \to 2 \to 4 \to 1$ (17)
$1 \to 7 \to 3 \to 6 \to 8 \to 2 \to 4 \to 5 \to 1$ (22)
$2 \to 4 \to 1 \to 3 \to 7 \to 5 \to 6 \to 8 \to 2$ (17)
$2 \to 4 \to 1 \to 3 \to 7 \to 5 \to 8 \to 6 \to 2$ (16)
$8 \to 6 \to 7 \to 3 \to 1 \to 4 \to 2 \to 5 \to 8$ (16)
$8 \to 6 \to 7 \to 3 \to 1 \to 5 \to 2 \to 4 \to 8$ (17)
$6 \to 8 \to 1 \to 4 \to 2 \to 5 \to 7 \to 3 \to 6$ (15)
$7 \to 5 \to 2 \to 4 \to 1 \to 8 \to 6 \to 3 \to 7$ (15)

Отже, вага найкоротшого маршруту дорівнює 15.

Завдання №7.

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V0 і V* .





Виконання:

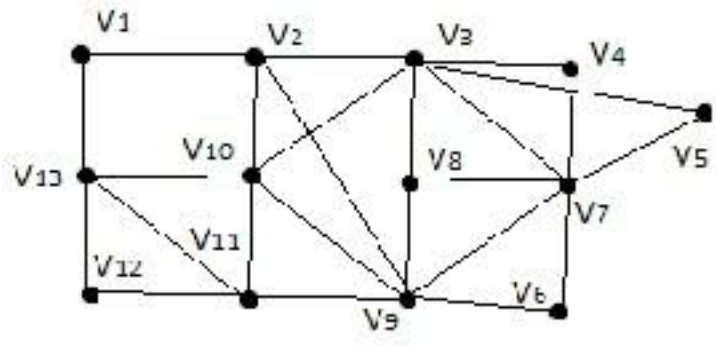
1) Розпочинаємо з початкової вершини, яку будемо вважати поточною. В даний момент це вершина V_0 . Переглядаємо її шлях до суміжних вершин - V_1 і V_6 . До першої вершини відстань дорівнює 6, до шостої - 4.
2) Проходимо через $V_0 \rightarrow V_1 \rightarrow V_2$, відстань дорівнює 7.
3) Проходимо через $V_1 \rightarrow V_2 \rightarrow V_3$, 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 викреслюєм 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 викреслюєм 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 викреслюєм 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 викреслюєм 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 викреслюєм 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 викреслюєм 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 викреслюєм 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 викреслюєм 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 викреслюєм 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.

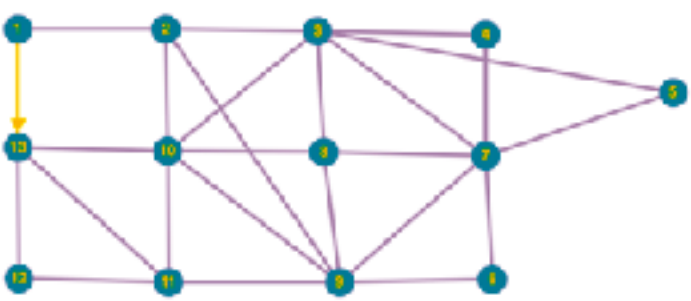
Завдання №8.

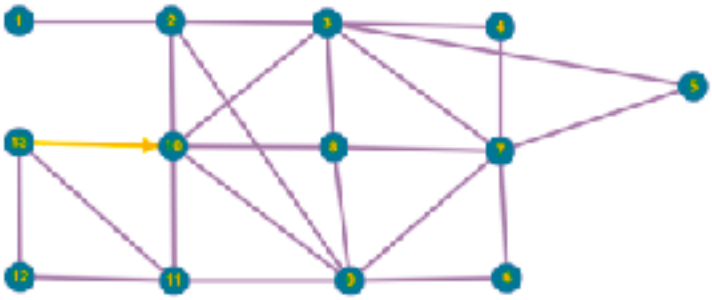
Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.

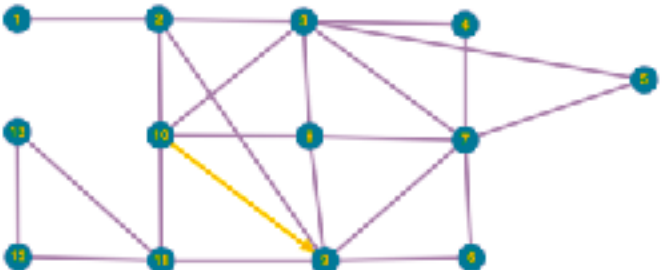


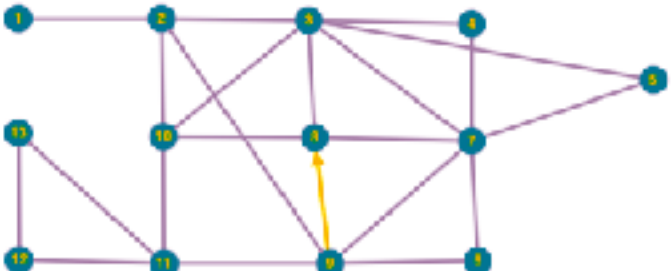
а) метод Флері:

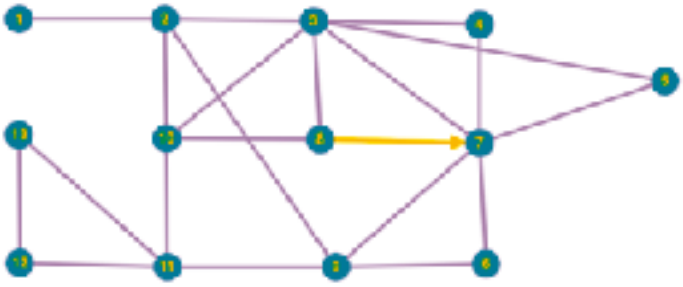
Алгоритм Флері полягає у тому, щоб починаючи з деякої вершини, викреслювати кожне пройдене ребро. Не проходимо по ребру, якщо видалення цього ребра призводить до розбиття графа на дві зв'язні компоненти, тобто, необхідно перевіряти, чи є ребро мостом чи ні.

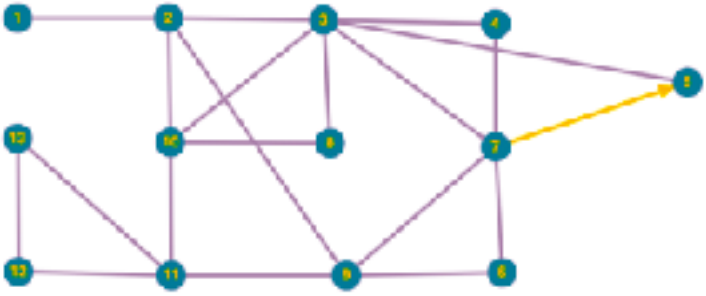


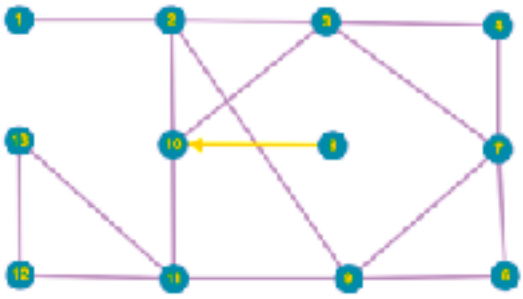


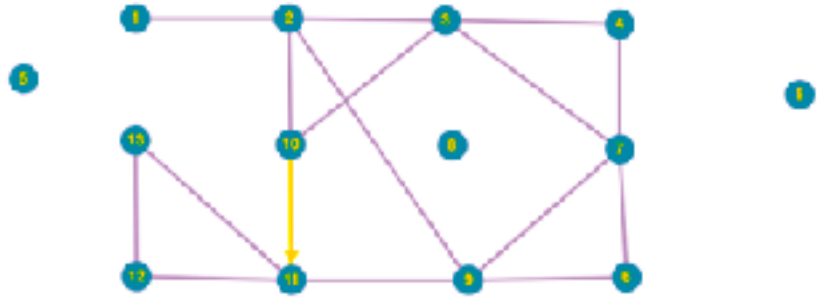


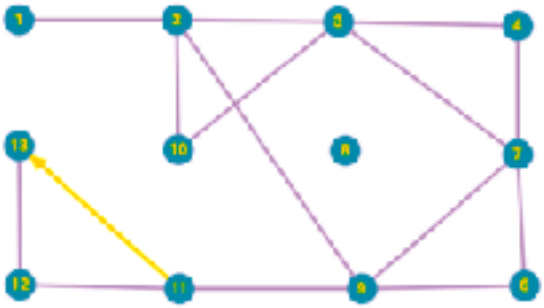


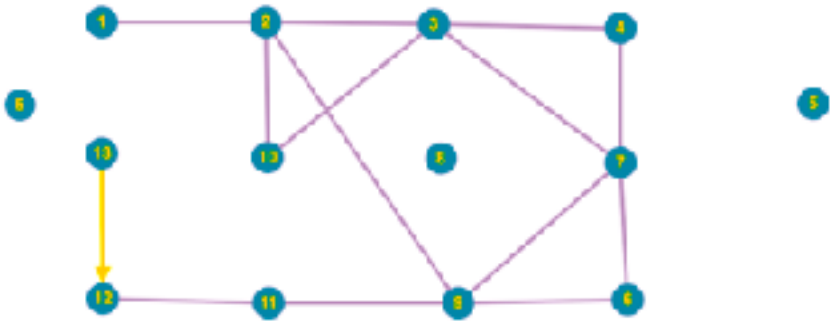


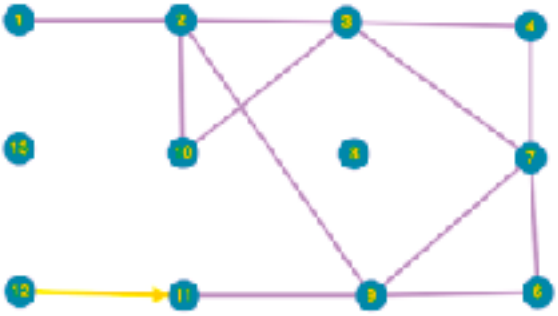


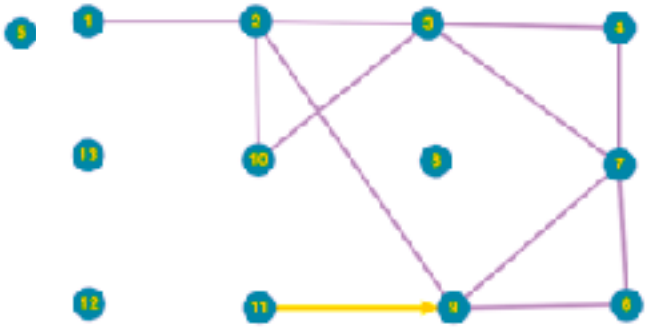


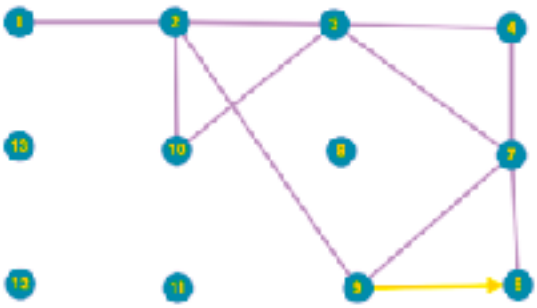


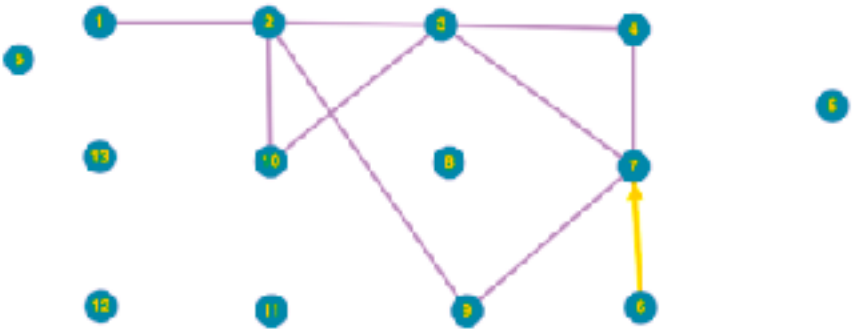


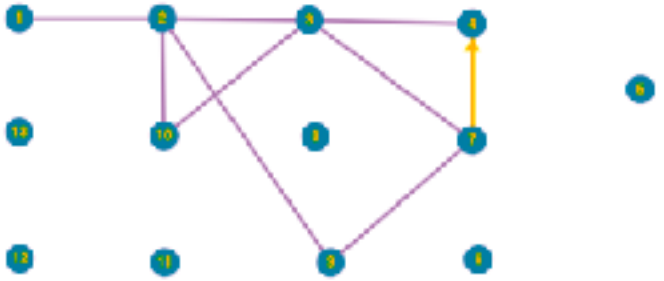


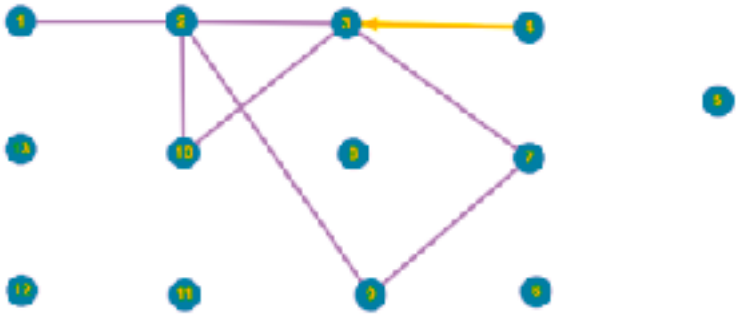


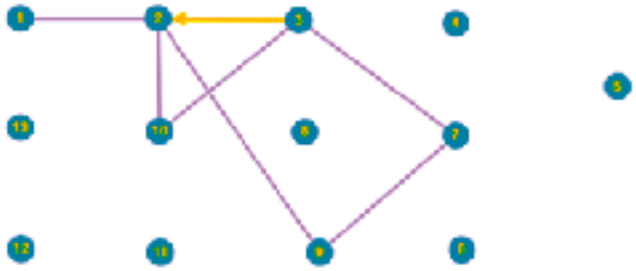








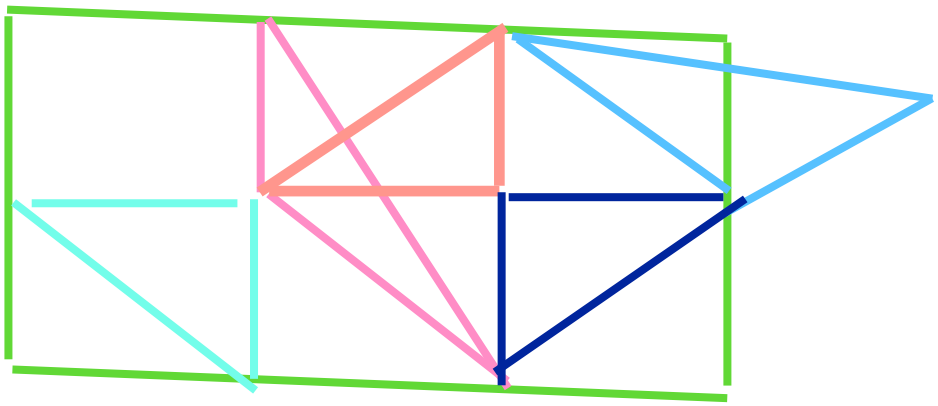




Отже, ейлеровий цикл даного графа: 1 → 13 → 10 → 9 → 8 → 7 → 5 → 3 → 8 → 10 → 11 → 13 → 12 → 11 → 9 → 6 → 7 → 4 → 3 → 2 → 9 → 7 → 2 → 10 → 2 → 1.

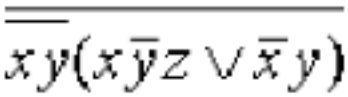
б) метод елементарних циклів:
Виділимо кольоровим елментарні цикли та об’єднаємо їх у один:

$V_1 \to V_2 \to V_3 \to V_4 \to V_7 \to V_6 \to V_9 \to V_{11} \to V_{12} \to V_{13}$
$V_2 \to V_9 \to V_{10}$
$V_3 \to V_5 \to V_7$
$V_3 \to V_8 \to V_{10}$
$V_7 \to V_8 \to V_9$
$V_{13} \to V_{11} \to V_{10}$



Завдання №9.

Спростити формули (привести їх до скороченої ДНФ).



спочатку запишем таблицю істинності для даної формули:

x	y	z	¬(¬(x∧y) ∧ (x∧¬y∧z ∨ ¬x∧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:
¬x¬y¬z
¬x¬yz
x¬y¬z
xy¬z
xyz

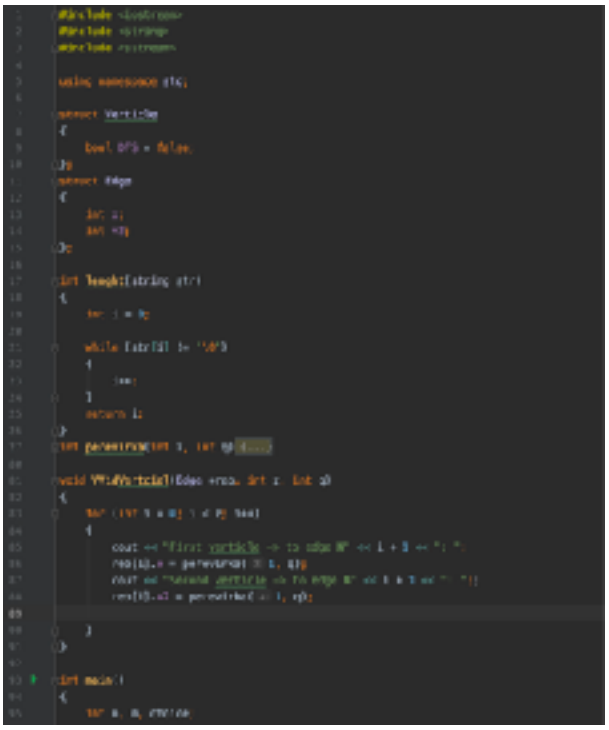
об’єднаємо диз’юнкцією і отримаємо ДДНФ:
¬x¬y¬z ∨ ¬x¬yz ∨ x¬y¬z ∨ xy¬z ∨ xyz

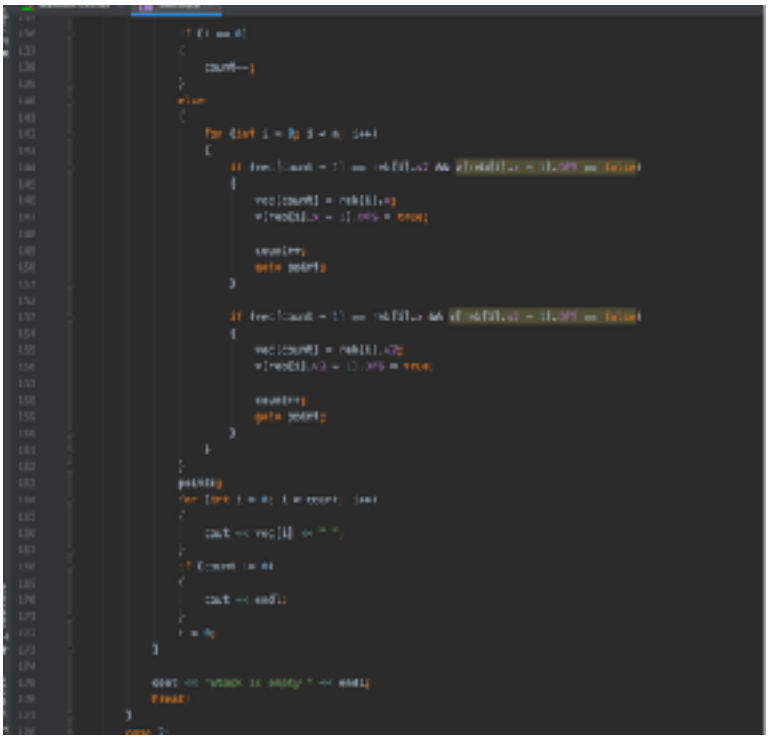
після перетворень отримаємо скорочену ДНФ:

¬x¬y ∨ x¬z ∨ xy

ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ

1. Обхід вглиб і вшир



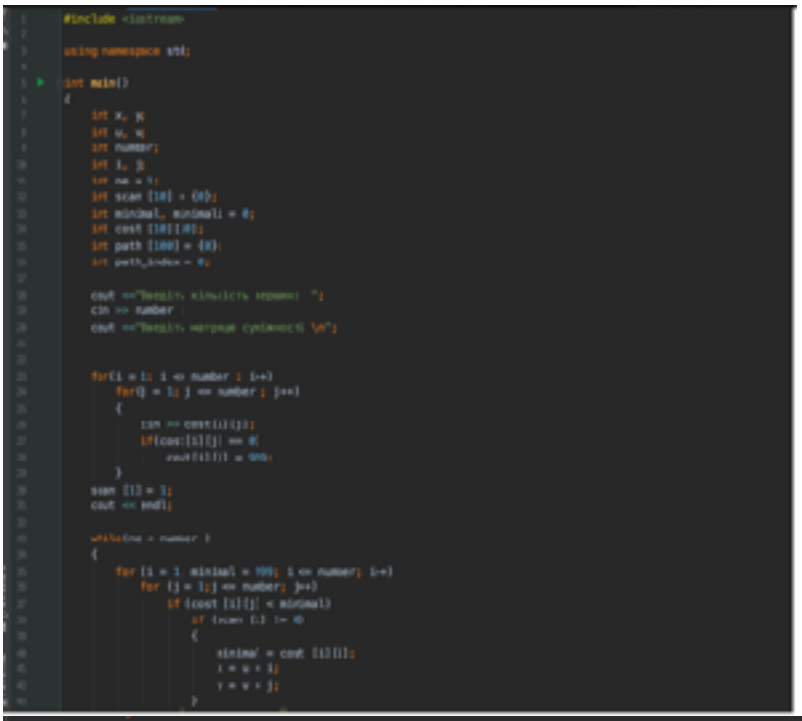


```
Number of edges in a graph: 10
Number of verticles in a graph: 10

First verticle -> to edge №1: 1
Second verticle -> to edge №1: 2
First verticle -> to edge №2: 1
Second verticle -> to edge №2: 8
First verticle -> to edge №3: 1
Second verticle -> to edge №3: 7
First verticle -> to edge №4: 2
Second verticle -> to edge №4: 7
First verticle -> to edge №5: 2
Second verticle -> to edge №5: 3
First verticle -> to edge №6: 3
Second verticle -> to edge №6: 4
First verticle -> to edge №7: 4
Second verticle -> to edge №7: 9
First verticle -> to edge №8: 4
Second verticle -> to edge №8: 8
First verticle -> to edge №9: 4
Second verticle -> to edge №9: 7
First verticle -> to edge №10: 4
Second verticle -> to edge №10: 5
First verticle -> to edge №11: 5
Second verticle -> to edge №11: 7
First verticle -> to edge №12: 5
Second verticle -> to edge №12: 6
First verticle -> to edge №13: 6
Second verticle -> to edge №13: 7
First verticle -> to edge №14: 6
Second verticle -> to edge №14: 8
First verticle -> to edge №15: 8
Second verticle -> to edge №15: 9
First verticle -> to edge №16: 9
Second verticle -> to edge №16: 10
```

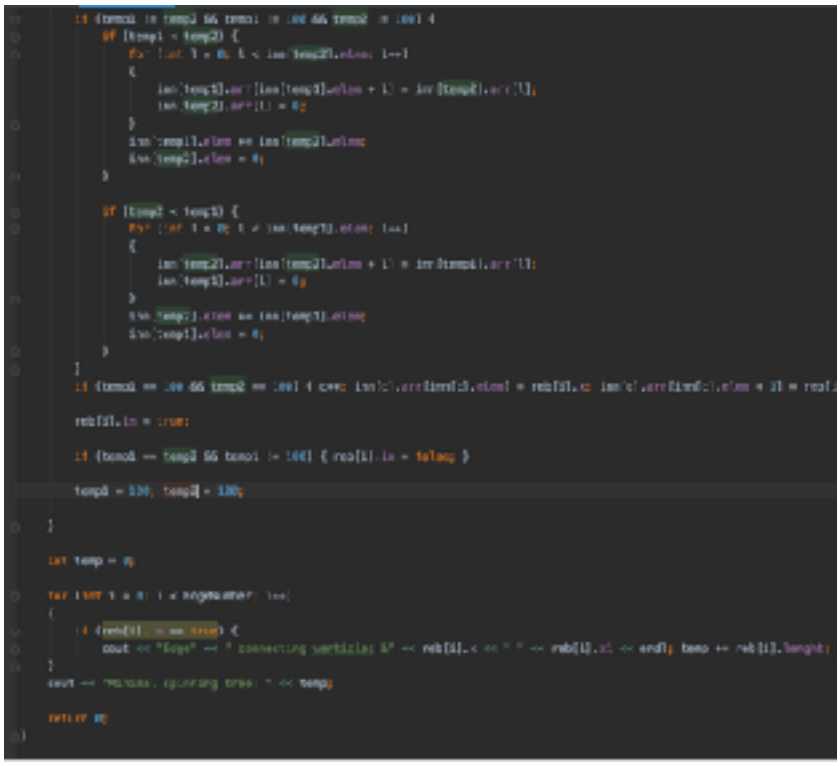
```
Second verticle -> to edge №16: 10
First verticle -> to edge №17: 7
Second verticle -> to edge №17: 8
First verticle -> to edge №18: 7
Second verticle -> to edge №18: 9
From which verticle do you want to start? 1
Choose what do you want to do:
 1 - DFS
 2 - BFS
1
1 2
1 2 7
1 2 7 4
1 2 7 4 3
1 2 7 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
```

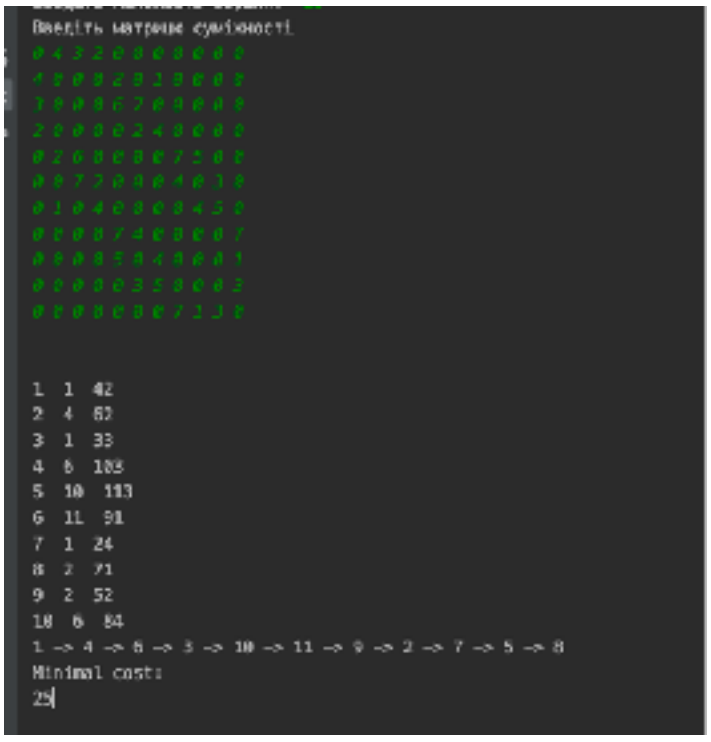
```
Choose what do you went to do:
 1 - DFS
 2 - BFS
2
1 2
1 2 8
1 2 8 7
2 8 7
2 8 7 3
8 7 3
8 7 3 6
8 7 3 6 9
7 3 6 9
7 3 6 9 4
7 3 6 9 4 5
3 6 9 4 5
6 9 4 5
9 4 5
9 4 5 10
4 5 10
5 10
10
Queue is empty
```

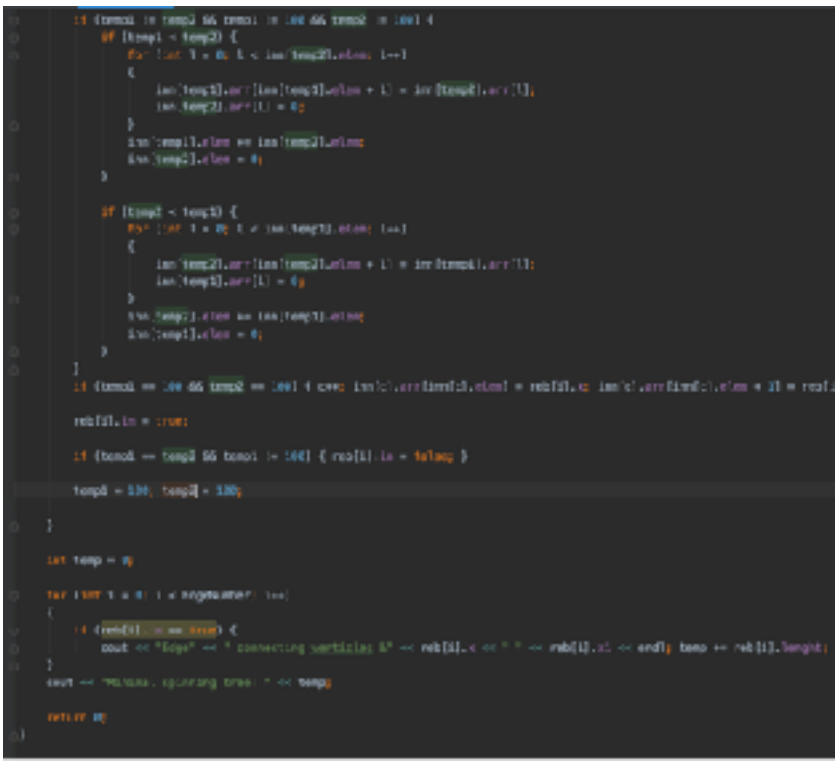
2. Метод Прима



3. Метод Краскала



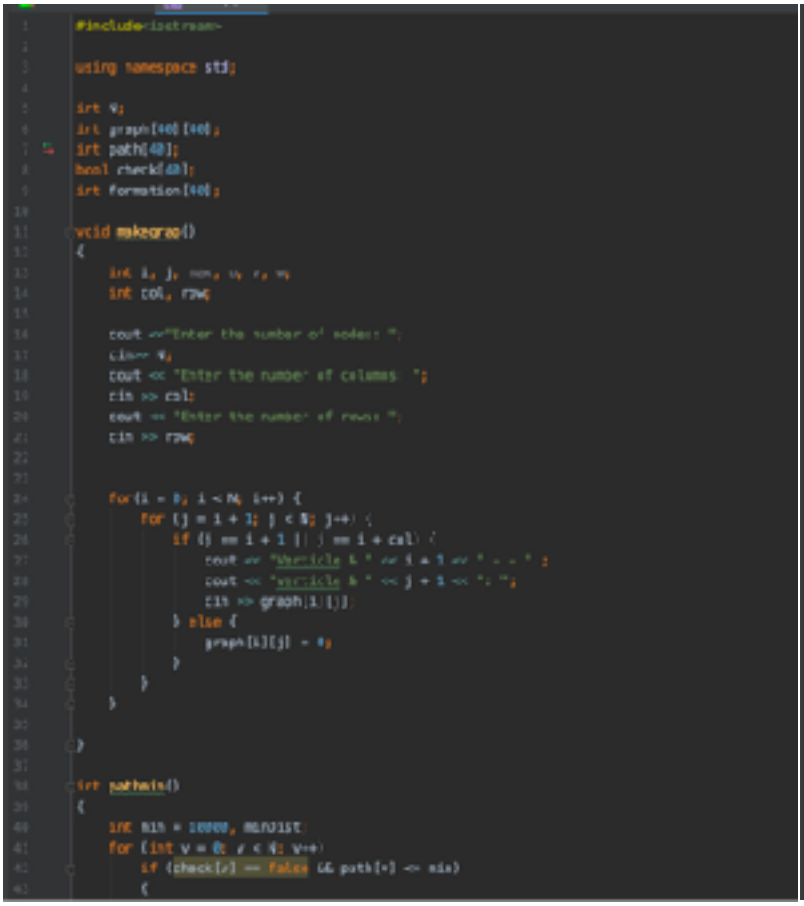


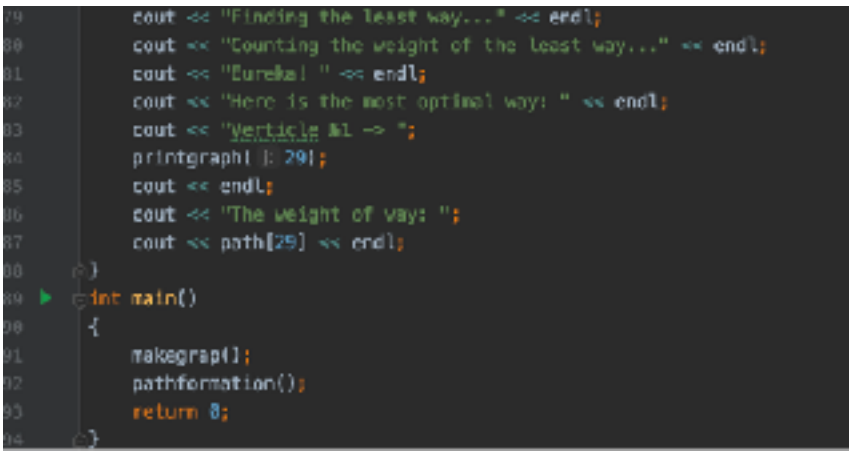


```
Length of 12 edge: 1
First verticle connected to  edge: 12  2
Second verticle connected to  edge: 12  7

Length of 13 edge: 5
First verticle connected to  edge: 13  5
Second verticle connected to  edge: 13  9

Length of 14 edge: 3
First verticle connected to  edge: 14  8
Second verticle connected to  edge: 14  10

Length of 15 edge: 6
First verticle connected to  edge: 15  3
Second verticle connected to  edge: 15  5

Length of 16 edge: 4
First verticle connected to  edge: 16  6
Second verticle connected to  edge: 16  8

Length of 17 edge: 2
First verticle connected to  edge: 17  4
Second verticle connected to  edge: 17  6

Length of 18 edge: 4
First verticle connected to  edge: 18  7
Second verticle connected to  edge: 18  9

Edge connecting verticles №9 11
Edge connecting verticles №2 7
Edge connecting verticles №1 4
Edge connecting verticles №2 5
Edge connecting verticles №4 6
Edge connecting verticles №1 3
Edge connecting verticles №10 11
Edge connecting verticles №6 10
Edge connecting verticles №1 2
Edge connecting verticles №6 8
Minimal spinning tree: 25
```


```
Edge connecting verticles №9 11
Edge connecting verticles №2 7
Edge connecting verticles №1 4
Edge connecting verticles №2 5
Edge connecting verticles №4 6
Edge connecting verticles №1 3
Edge connecting verticles №10 11
Edge connecting verticles №6 10
Edge connecting verticles №1 2
Edge connecting verticles №6 8
Minimal spinning tree: 25
```

4. Алгоритм Дейкстри





```
Enter the number of nodes: 30
Enter the number of columns: 6
Enter the number of rows: 5
Verticle № 1 - - verticle № 2: 6
Verticle № 1 - - verticle № 7: 4
Verticle № 2 - - verticle № 3: 1
Verticle № 2 - - verticle № 8: 8
Verticle № 3 - - verticle № 4: 1
Verticle № 3 - - verticle № 9: 3
Verticle № 4 - - verticle № 5: 3
Verticle № 4 - - verticle № 10: 1
Verticle № 5 - - verticle № 6: 3
Verticle № 5 - - verticle № 11: 5
Verticle № 6 - - verticle № 7: 8
Verticle № 6 - - verticle № 12: 7
Verticle № 7 - - verticle № 8: 2
Verticle № 7 - - verticle № 13: 5
Verticle № 8 - - verticle № 9: 1
Verticle № 8 - - verticle № 14: 1
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: 1
Verticle № 12 - - verticle № 13: 8
Verticle № 12 - - verticle № 18: 7
Verticle № 13 - - verticle № 14: 7
Verticle № 13 - - verticle № 19: 5
Verticle № 14 - - verticle № 15: 1
Verticle № 14 - - verticle № 20: 7
Verticle № 15 - - verticle № 16: 2
Verticle № 15 - - verticle № 21: 1
Verticle № 16 - - verticle № 17: 3
```

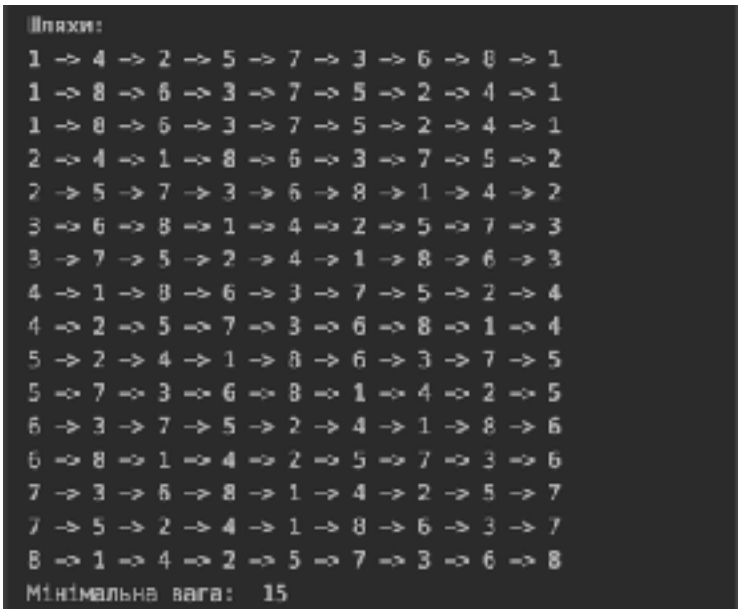
```
Verticle № 10 - - verticle № 16: 4
Verticle № 11 - - verticle № 12: 4
Verticle № 11 - - verticle № 17: 1
Verticle № 12 - - verticle № 13: 8
Verticle № 12 - - verticle № 18: 7
Verticle № 13 - - verticle № 14: 7
Verticle № 13 - - verticle № 19: 5
Verticle № 14 - - verticle № 15: 1
Verticle № 14 - - verticle № 20: 7
Verticle № 15 - - verticle № 16: 2
Verticle № 15 - - verticle № 21: 1
Verticle № 16 - - verticle № 17: 3
Verticle № 16 - - verticle № 22: 1
Verticle № 17 - - verticle № 18: 7
Verticle № 17 - - verticle № 23: 2
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: 1
Verticle № 21 - - verticle № 27: 1
Verticle № 22 - - verticle № 23: 8
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: 3
Verticle № 28 - - verticle № 29: 3
Verticle № 29 - - verticle № 30: 6
```

```
Verticle № 19 - - verticle № 26: 8
Verticle № 20 - - verticle № 21: 3
Verticle № 20 - - verticle № 26: 2
Verticle № 21 - - verticle № 22: 1
Verticle № 21 - - verticle № 27: 1
Verticle № 22 - - verticle № 23: 8
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: 3
Verticle № 28 - - verticle № 29: 3
Verticle № 29 - - verticle № 30: 6
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 №22 -> Verticle №28 -> Verticle №29 -> Verticle №30 ->
The weight of way: 21
```

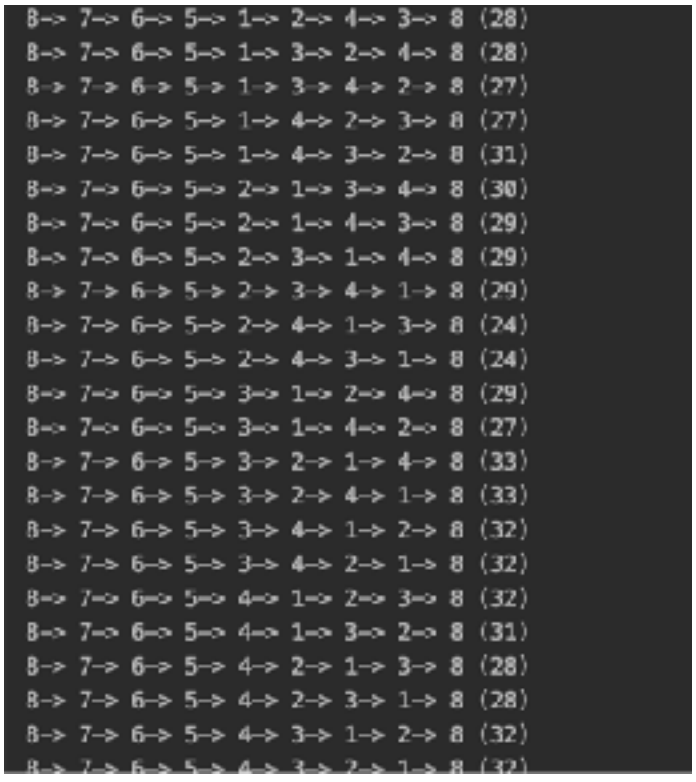
5. «Іди в найближчий» для розв’язання задачі комівояжера

```
            min = long;
            m = 1;
        }
        if (long == min)
        {
            m++;
        }

    }
    var = count - 1;
}

for (int i = 0; i < count; i++)
{
    mas[i] = i;
    minmas[i] = i;
}
Array *rez = new Array[n];
int iter = 0;

while (whichway(mas, count))
{
    while (mas[var] != count)
    {
        mas[var]++;

        if (CycleMatrix(mas, count))
        {
            long = way(arr, mas);

            if (long == min)
            {
                for (int i = 0; i < count; i++)
                {
                    rez[iter].papu[i] = mas[i];
                }
                rez[iter].papu[count] = mas[0];
                iter++;
            }
        }
    }
    while (mas[var] == count)
    {
        mas[var] = 1;
```

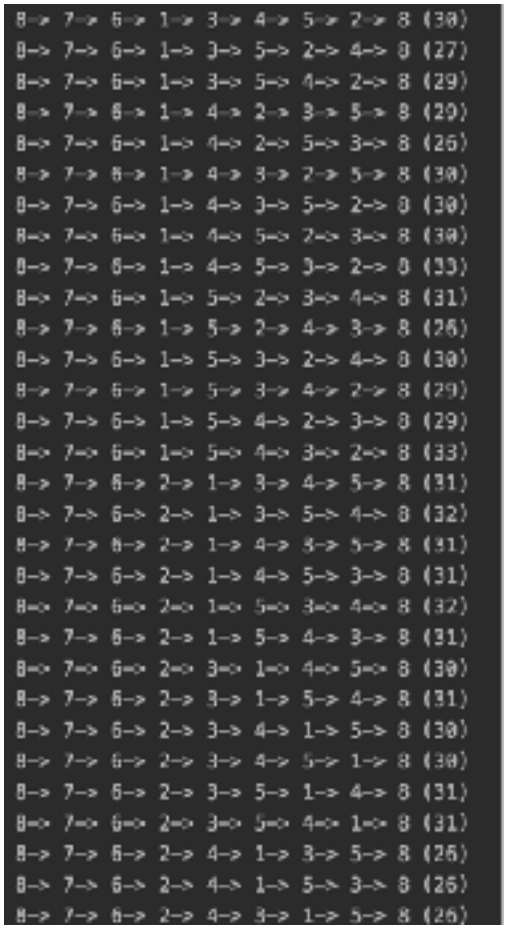
```
            {
                for (int i = 0; i < count; i++)
                {
                    rez[iter].papu[i] = mas[i];
                }
                rez[iter].papu[count] = mas[0];
                iter++;
            }

        }
        var = count - 1;
    }

    cout << "Шляхи: " << endl;

    for (int i = 0; i < iter - 1; i++)
    {
        for (int j = 0; j <= count; j++)
        {
            if (j != 0)
            {
                cout << "-> ";
            }
            cout << rez[i].papu[j] << " ";
        }
        cout << endl;
        if (i == 0)
        {
            for (int j = 0; j <= count; j++)
            {
                if (j != 0)
                {
                    cout << "-> ";
                }
                cout << rez[i].papu[count - j] << " ";
            }

            cout << endl;
        }
    }
    cout << "Мінімальна вага: " << min;
```

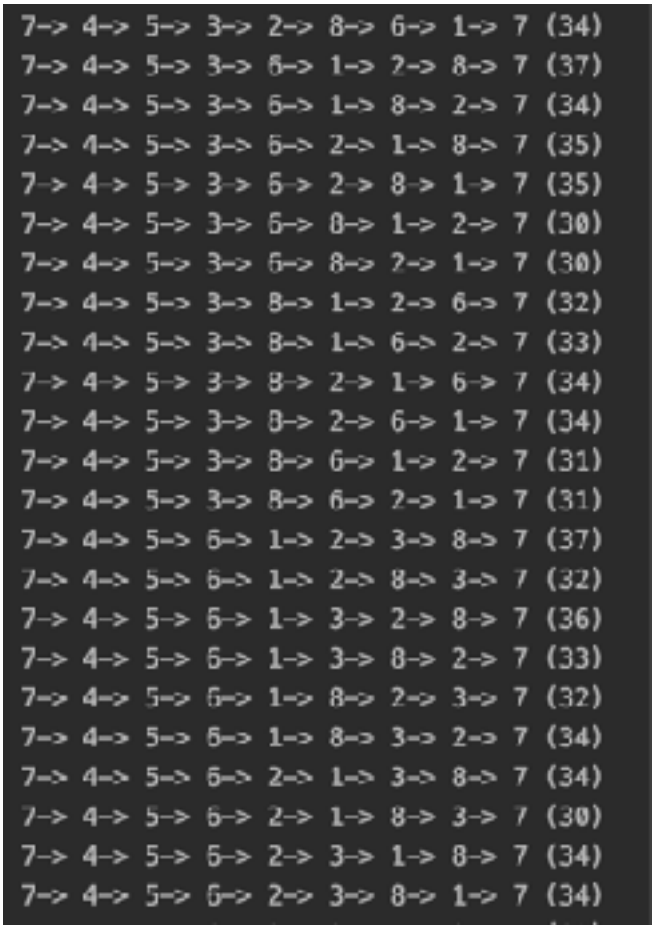


```
Шляхи:
1 -> 4 -> 2 -> 5 -> 7 -> 3 -> 6 -> 8 -> 1
1 -> 8 -> 6 -> 3 -> 7 -> 5 -> 2 -> 4 -> 1
1 -> 8 -> 6 -> 3 -> 7 -> 5 -> 2 -> 4 -> 1
2 -> 4 -> 1 -> 8 -> 6 -> 3 -> 7 -> 5 -> 2
2 -> 5 -> 7 -> 3 -> 6 -> 8 -> 1 -> 4 -> 2
3 -> 6 -> 8 -> 1 -> 4 -> 2 -> 5 -> 7 -> 3
3 -> 7 -> 5 -> 2 -> 4 -> 1 -> 8 -> 6 -> 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)
```





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.

```
2-1  1-11  13-10  10-1  3-2  2-9  9-6  6-7  7-3  3-4  4-7  7-5  5-1  3-8  8-7  7-9  9-8  8-10  10-6  9-11  11-12  12-13  13-1  11-10  10-3

Process finished with exit code 0
```