

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА  
ПОЛІТЕХНІКА”**

**Кафедра систем штучного інтелекту**

**Розрахункова робота**

з дисципліни

«Дискретна математика»

**Виконав:**

Студент групи КН-113

Волошин Володимир

**Викладач:**

Мельникова Н.І.

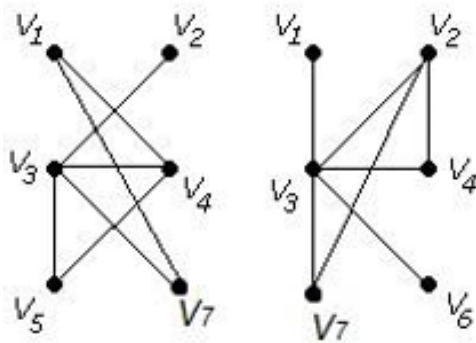
Львів – 2019р.

## Варіант 12

**Завдання № 1.** Розв'язати на графах наступні задачі:

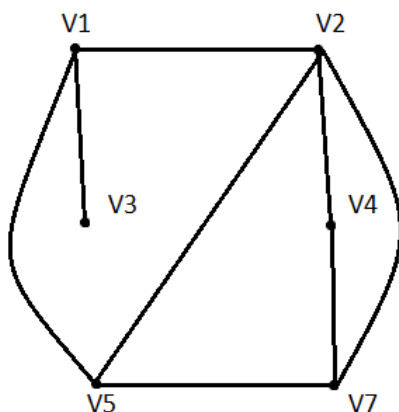
Виконати наступні операції над графами:

- 1) знайти доповнення до першого графу,
- 2) об'єднання графів,
- 3) кільцеву суму  $G_1$  та  $G_2$  ( $G_1+G_2$ ),
- 4) розщепити вершину у другому графі,
- 5) виділити підграф  $A$ , що складається з 3-х вершин в  $G_1$  і знайти стягнення  $A$  в  $G_1$  ( $G_1 \setminus A$ ),
- 6) добуток графів.

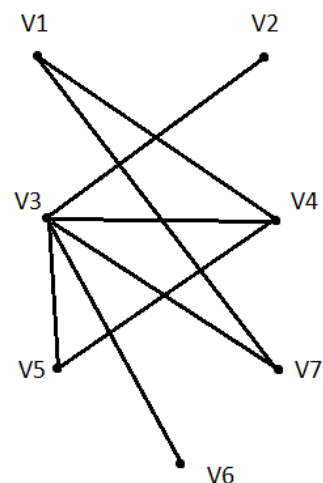


*Розв'язок:*

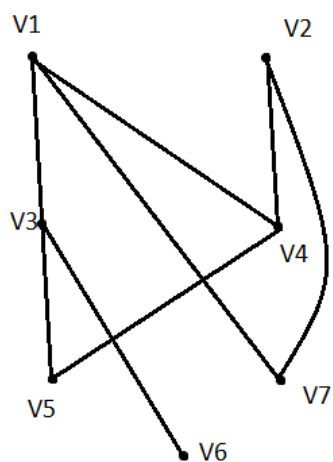
1)



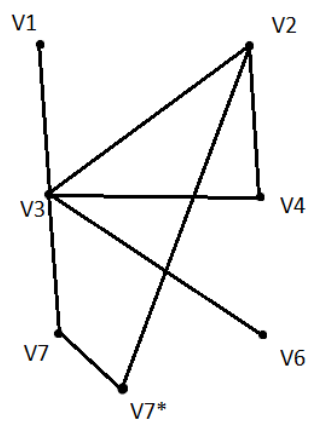
2)



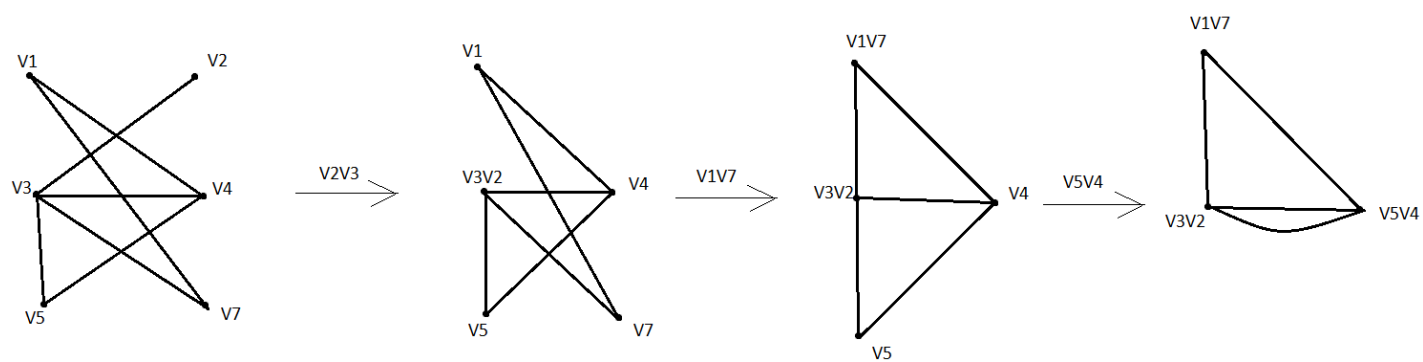
3)



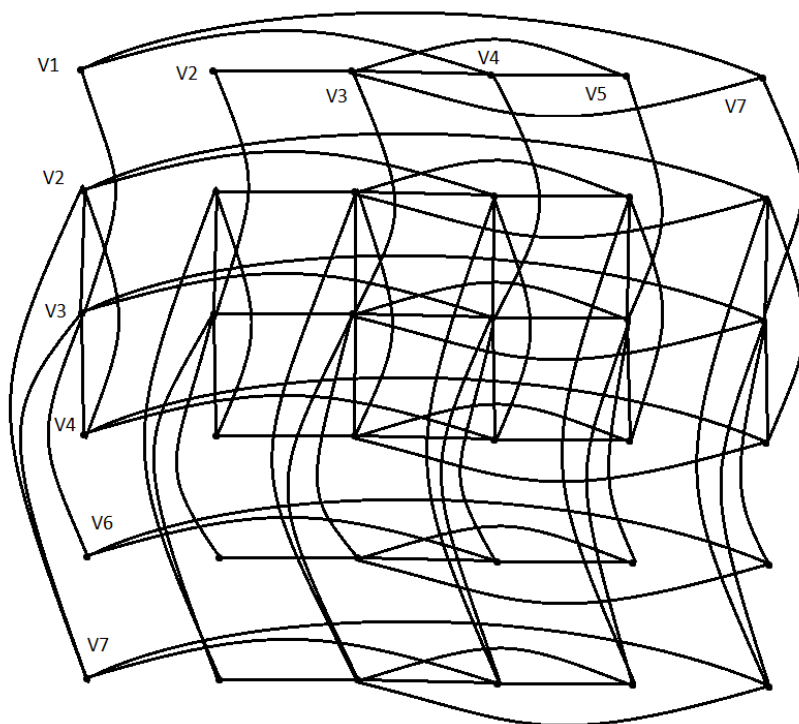
4)



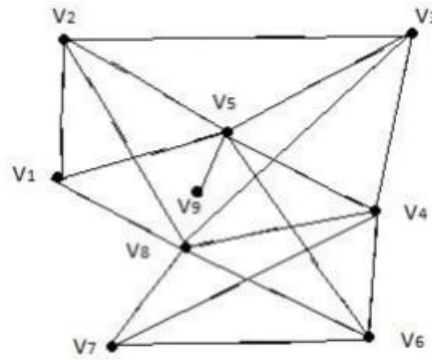
5)



6)



**Завдання № 2.** Скласти таблицю суміжності для графа.



*Розв'язок:*

Таблиця суміжності:

	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	0	0	1	0	0	1	0
V2	1	0	1	0	1	0	0	1	0
V3	0	1	0	1	1	0	0	1	0
V4	0	0	1	0	1	1	1	1	0
V5	1	1	1	1	0	1	0	0	1
V6	0	0	0	1	1	0	1	1	0
V7	0	0	0	1	0	1	0	1	0
V8	1	1	1	1	0	1	1	0	0
V9	0	0	0	0	1	0	0	0	0

**Завдання № 3.**

Для графа з другого завдання знайти діаметр.

*Розв'язок:*

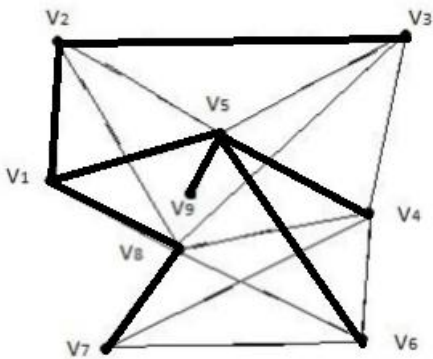
Діаметр = 3.

**Завдання № 4.**

Для графа з другого завдання виконати обхід дерева вглиб (варіант закінчується на непарне число) або вшир (закінчується на парне число).

## Розв'язок:

### Пошук вшир



Вершина	BFS номер	Вміст черги
V1	1	V1
V2	2	V1V2
V5	3	V1V2V5
V8	4	V1V2V5V8
-	-	V2V5V8
V3	5	V2V5V8V3
-	-	V5V8V3
V4	6	V5V8V3V4
V6	7	V5V8V3V4V6
V9	8	V5V8V3V4V6V9
-	-	V8V3V4V6V9
V7	9	V8V3V4V6V9V7
-	-	V3V4V6V9V7
-	-	V4V6V9V7
-	-	V6V9V7
-	-	V9V7
-	-	V7
-	-	∅

## Програмна реалізація:

```
#include <iostream>
#include <queue>
#include <fstream>
using namespace std;

int n;
int matrix[1000][1000];
bool already[1000];
queue<int> a;

void Foo(int v);

int main()
{
    ifstream fin("FILE.txt");
    fin >> n;
    for (int i = 0; i < n; i++) {
        already[i] = false;
        for (int j = 0; j < n; j++)
            fin >> matrix[i][j];
    }
}
```

```

    }
    Foo(0);
    return 0;
}

void Foo(int v) {
    already[v] = true;
    cout << v << endl;
    for (int i = 0; i < n; i++)
        if (matrix[v][i] && !already[i])
            a.push(i);
    v = a.front();
    if (!a.empty()) {
        a.pop();
        Foo(v);
    }
}

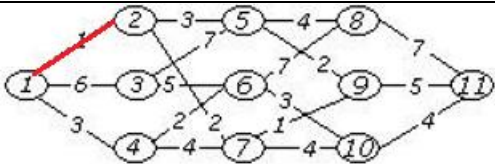
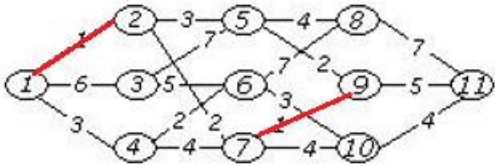
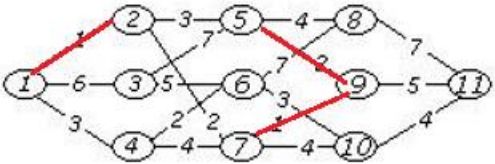
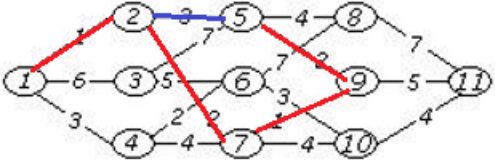
```

### Завдання № 5.

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

*Розв'язок:*

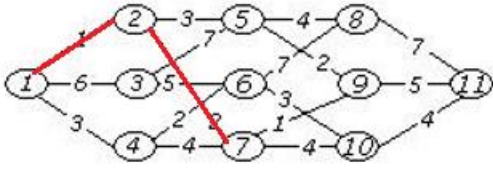
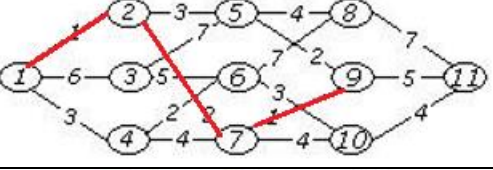
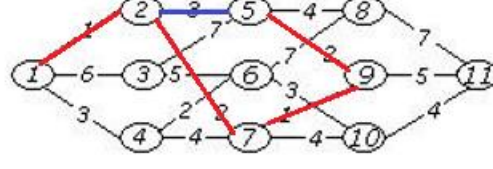
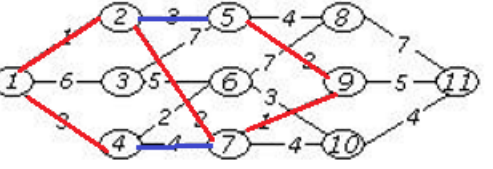
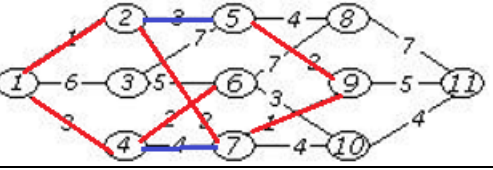
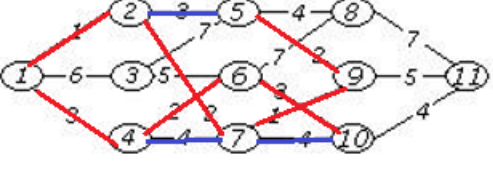
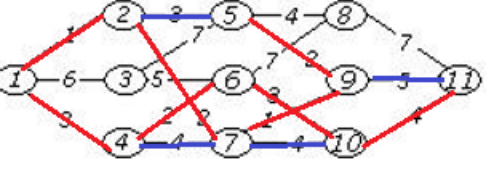
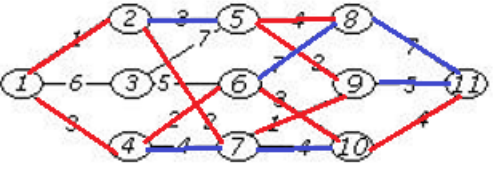
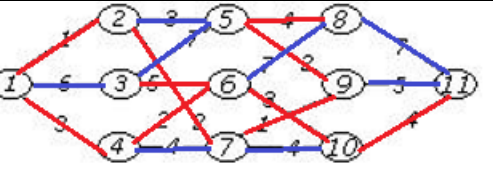
#### Метод Краскала

Крок	Малюнок	Пояснення	Вибрані ребра
1		Вибирається ребро з найменшою вагою.	(1,2)
2		Тепер найменшим є (9,7), його і вибираємо. Так будемо діяти і надалі.	(1,2), (9,7)
3			(1,2), (9,7), (5,9)
4		Після того як ми вибрали (2,7) бачимо що ребро (2,5) утворить цикл, тому його одразу	(1,2), (9,7), (5,9), (2,7)

		закреслюємо синім кольором. Так діємо і надалі.	
5			(1,2), (9,7), (5,9), (2,7), (4,6)
6			(1,2), (9,7), (5,9), (2,7), (4,6), (6,10)
7			(1,2), (9,7), (5,9), (2,7), (4,6), (6,10), (1,4)
8			(1,2), (9,7), (5,9), (2,7), (4,6), (6,10), (1,4), (10,11)
9			(1,2), (9,7), (5,9), (2,7), (4,6), (6,10), (1,4), (10,11), (5,8)
10		Всі ребра замальовані, отже мінімальне остове дерево знайдено.	(1,2), (9,7), (5,9), (2,7), (4,6), (6,10), (1,4), (10,11), (5,8), (3,6)

## Метод Прима

Крок	Малюнок	Пояснення	Вибрані ребра
1		<p>Вибирається довільна точка та прилегле ребро з найменшою вагою. Нехай це буде точка 1 і прилегле ребро (1,2).</p>	(1,2)

2		Тепер серед інцидентних ребер обираємо ребро з найменшою вагою. Так діємо і надалі.	(1,2), (2,7)
3			(1,2), (2,7), (7,9)
4		Після того як ми вибрали (5,9) бачимо що ребро (2,5) утворюють цикл, тому його одразу закреслюємо синім кольором. Так діємо і надалі.	(1,2), (2,7), (7,9), (5,9)
5			(1,2), (2,7), (7,9), (5,9), (1,4)
6			(1,2), (2,7), (7,9), (5,9), (1,4), (4,6)
7			(1,2), (2,7), (7,9), (5,9), (1,4), (4,6), (6,10)
8			(1,2), (2,7), (7,9), (5,9), (1,4), (4,6), (6,10), (10,11)
9			(1,2), (2,7), (7,9), (5,9), (1,4), (4,6), (6,10), (10,11), (5,8)
10		Всі ребра замальовані, отже мінімальне остове дерево знайдено.	(1,2), (2,7), (7,9), (5,9), (1,4), (4,6), (6,10), (10,11), (5,8), (3,6)



## Програмна реалізація:

```
#include <iostream>
#include <fstream>
using namespace std;

/*
    Алгоритм Прима знаходження мінімального остову
*/

//Функція знаходження мінімального значення
int FindMin(int* arr, int size)
{
    for (int i = 1; i < size; i++){
        if (arr[0] > arr[i])
            arr[0] = arr[i];
    }
    return arr[0];
}

int main()
{
    setlocale(LC_ALL, "Ukr");
    int size = 11;
    cout << "Введіть кількість вершин графа: ";
    cin >> size;
    int** arr1 = new int* [size]; //створення двовимірного динамічного масиву
    for (int i = 0; i < size; i++)
        arr1[i] = new int[size];

    for (int i = 0; i < size; i++){ //Ініціалізація масиву нулями
        for (int j = 0; j < size; j++)
            arr1[i][j] = 0;
    }

    int num_of_edges = 18, weight = 0, vertex1 = 0, vertex2 = 0;
    cout << "Введіть кількість ребер графа: ";
    cin >> num_of_edges;

    ifstream fin("FILE.txt"); //Зчитування даних про граф з файлу
    if (!fin.is_open())
        cout << "Error\n";
    else {
        for (int i = 0; i < num_of_edges; i++){
            fin >> weight;
            fin >> vertex1;
            fin >> vertex2;
            arr1[vertex1 - 1][vertex2 - 1] = weight;
            arr1[vertex2 - 1][vertex1 - 1] = weight;
        }
    }
    fin.close();

    int min[100], min_size = 0, counter = 0, n_of_chosens = 1;
    int* chosens = new int [size]; //Створення допоміжного динамічного масиву
    chosens[counter] = 0;
    bool end = false;
    cout << "Мінімальне остове дерево: ";
    while(!end)
    {
        end = true;
        min_size = 0;
        for (int n = 0; n < size; n++) //Видалення потрібних рядків рядків
            arr1[chosens[counter]][n] = 0;
```

```

        for (int j = 0; j < n_of_chosens; j++) {
            for (int i = 0; i < size; i++){//Пошук можливих наступних ребер
                if (arr1[i][chosens[j]] > 0){
                    min[min_size] = arr1[i][chosens[j]];
                    min_size++;
                    end = false;
                }
            }
        }
        int minimum = FindMin(min, min_size);//Знаходження мінімального значення з них
        for (int j = 0; j < n_of_chosens; j++){
            for (int i = 0; i < size; i++){
                if (arr1[i][chosens[j]] == minimum){//знаходження вибраного ребра
                    cout << "(" << i + 1 << ", " << chosens[j] + 1 << ") ";
                    n_of_chosens++;
                    counter++;
                    chosens[counter] = i;
                    goto link;
                }
            }
        }
        link:;
    }
    for (int i = 0; i < size; i++)
        delete[] arr1[i];
    delete[] arr1;
    delete[] chosens;
    return 0;
}

```

Вхідні дані:

```

1 1 2
6 1 3
3 1 4
4 4 7
2 4 6
5 3 6
7 3 5
3 2 5
2 2 7
4 5 8
2 5 9
7 6 8
3 6 10
4 7 10
1 7 9
4 10 11
5 9 11
7 8 11

```

Результат виконання програми:

```

Введіть кількість вершин графа: 11
Введіть кількість ребер графа: 18
Мінімальне остове дерево: (2,1) (7,2) (9,7) (5,9) (4,1) (6,4) (10,6) (8,5) (11,10) (3,6)

```

## Завдання № 6

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

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

Розв'язок:

Матриця	Пояснення	Шлях																																																																																	
<table><tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr><tr><td>1</td><td><del><math>\infty</math></del></td><td><del>5</del></td><td><del>6</del></td><td><del>5</del></td><td><del>4</del></td><td><del>4</del></td><td><del>5</del></td><td><del>5</del></td></tr><tr><td>2</td><td>5</td><td><math>\infty</math></td><td>1</td><td>5</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>3</td><td>6</td><td>1</td><td><math>\infty</math></td><td>1</td><td>1</td><td>3</td><td>2</td><td>1</td></tr><tr><td>4</td><td>5</td><td>5</td><td>1</td><td><math>\infty</math></td><td>5</td><td>5</td><td>7</td><td>5</td></tr><tr><td>5</td><td>4</td><td>1</td><td>1</td><td>5</td><td><math>\infty</math></td><td>3</td><td>2</td><td>5</td></tr><tr><td>6</td><td>4</td><td>1</td><td>3</td><td>5</td><td>3</td><td><math>\infty</math></td><td>5</td><td>6</td></tr><tr><td>7</td><td>5</td><td>1</td><td>2</td><td>7</td><td>2</td><td>5</td><td><math>\infty</math></td><td>1</td></tr><tr><td>8</td><td>5</td><td>1</td><td>1</td><td>5</td><td>5</td><td>6</td><td>1</td><td><math>\infty</math></td></tr></table>		1	2	3	4	5	6	7	8	1	<del><math>\infty</math></del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>	2	5	$\infty$	1	5	1	1	1	1	3	6	1	$\infty$	1	1	3	2	1	4	5	5	1	$\infty$	5	5	7	5	5	4	1	1	5	$\infty$	3	2	5	6	4	1	3	5	3	$\infty$	5	6	7	5	1	2	7	2	5	$\infty$	1	8	5	1	1	5	5	6	1	$\infty$	<p>Починаємо шлях з вершини 1.</p> <p>Закреслюємо перший рядок та у першому стовпці шукаємо вершину, до якої шлях є мінімальний. Таких декілька, тому довільним чином обираємо вершину 6.</p> <p>Обводимо елемент 6 1.</p>	1-6
	1	2	3	4	5	6	7	8																																																																											
1	<del><math>\infty</math></del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>																																																																											
2	5	$\infty$	1	5	1	1	1	1																																																																											
3	6	1	$\infty$	1	1	3	2	1																																																																											
4	5	5	1	$\infty$	5	5	7	5																																																																											
5	4	1	1	5	$\infty$	3	2	5																																																																											
6	4	1	3	5	3	$\infty$	5	6																																																																											
7	5	1	2	7	2	5	$\infty$	1																																																																											
8	5	1	1	5	5	6	1	$\infty$																																																																											

<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr><tr><th>1</th><td><del>∞</del></td><td><del>5</del></td><td><del>6</del></td><td><del>5</del></td><td><del>4</del></td><td><del>4</del></td><td><del>5</del></td><td><del>5</del></td></tr><tr><th>2</th><td>5</td><td>∞</td><td>1</td><td>5</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><th>3</th><td>6</td><td>1</td><td>∞</td><td>1</td><td>1</td><td>3</td><td>2</td><td>1</td></tr><tr><th>4</th><td>5</td><td>5</td><td>1</td><td>∞</td><td>5</td><td>5</td><td>7</td><td>5</td></tr><tr><th>5</th><td>4</td><td>1</td><td>1</td><td>5</td><td>∞</td><td>3</td><td>2</td><td>5</td></tr><tr><th>6</th><td>4</td><td>1</td><td>3</td><td>5</td><td>3</td><td>∞</td><td>5</td><td>6</td></tr><tr><th>7</th><td>5</td><td>1</td><td>2</td><td>7</td><td>2</td><td>5</td><td>∞</td><td>1</td></tr><tr><th>8</th><td>5</td><td>1</td><td>1</td><td>5</td><td>5</td><td>6</td><td>1</td><td>∞</td></tr></table>		1	2	3	4	5	6	7	8	1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>	2	5	∞	1	5	1	1	1	1	3	6	1	∞	1	1	3	2	1	4	5	5	1	∞	5	5	7	5	5	4	1	1	5	∞	3	2	5	6	4	1	3	5	3	∞	5	6	7	5	1	2	7	2	5	∞	1	8	5	1	1	5	5	6	1	∞	Тепер закреслюємо шостий рядок і шукаємо знову вершину, до якої шлях є мінімальний, вже у шостому стовпці. Це вершина 2. Обводимо елемент 2 6. Так діємо і надалі.	1-6-2
	1	2	3	4	5	6	7	8																																																																											
1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>																																																																											
2	5	∞	1	5	1	1	1	1																																																																											
3	6	1	∞	1	1	3	2	1																																																																											
4	5	5	1	∞	5	5	7	5																																																																											
5	4	1	1	5	∞	3	2	5																																																																											
6	4	1	3	5	3	∞	5	6																																																																											
7	5	1	2	7	2	5	∞	1																																																																											
8	5	1	1	5	5	6	1	∞																																																																											
<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr><tr><th>1</th><td><del>∞</del></td><td><del>5</del></td><td><del>6</del></td><td><del>5</del></td><td><del>4</del></td><td><del>4</del></td><td><del>5</del></td><td><del>5</del></td></tr><tr><th>2</th><td><del>5</del></td><td><del>∞</del></td><td><del>1</del></td><td><del>5</del></td><td><del>1</del></td><td>1</td><td><del>1</del></td><td><del>1</del></td></tr><tr><th>3</th><td>6</td><td>1</td><td>∞</td><td>1</td><td>1</td><td>3</td><td>2</td><td>1</td></tr><tr><th>4</th><td>5</td><td>5</td><td>1</td><td>∞</td><td>5</td><td>5</td><td>7</td><td>5</td></tr><tr><th>5</th><td>4</td><td>1</td><td>1</td><td>5</td><td>∞</td><td>3</td><td>2</td><td>5</td></tr><tr><th>6</th><td>4</td><td>1</td><td>3</td><td>5</td><td>3</td><td>∞</td><td>5</td><td>6</td></tr><tr><th>7</th><td>5</td><td>1</td><td>2</td><td>7</td><td>2</td><td>5</td><td>∞</td><td>1</td></tr><tr><th>8</th><td>5</td><td>1</td><td>1</td><td>5</td><td>5</td><td>6</td><td>1</td><td>∞</td></tr></table>		1	2	3	4	5	6	7	8	1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>	2	<del>5</del>	<del>∞</del>	<del>1</del>	<del>5</del>	<del>1</del>	1	<del>1</del>	<del>1</del>	3	6	1	∞	1	1	3	2	1	4	5	5	1	∞	5	5	7	5	5	4	1	1	5	∞	3	2	5	6	4	1	3	5	3	∞	5	6	7	5	1	2	7	2	5	∞	1	8	5	1	1	5	5	6	1	∞		1-6-2-8
	1	2	3	4	5	6	7	8																																																																											
1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>																																																																											
2	<del>5</del>	<del>∞</del>	<del>1</del>	<del>5</del>	<del>1</del>	1	<del>1</del>	<del>1</del>																																																																											
3	6	1	∞	1	1	3	2	1																																																																											
4	5	5	1	∞	5	5	7	5																																																																											
5	4	1	1	5	∞	3	2	5																																																																											
6	4	1	3	5	3	∞	5	6																																																																											
7	5	1	2	7	2	5	∞	1																																																																											
8	5	1	1	5	5	6	1	∞																																																																											
<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr><tr><th>1</th><td><del>∞</del></td><td><del>5</del></td><td><del>6</del></td><td><del>5</del></td><td><del>4</del></td><td><del>4</del></td><td><del>5</del></td><td><del>5</del></td></tr><tr><th>2</th><td><del>5</del></td><td><del>∞</del></td><td><del>1</del></td><td><del>5</del></td><td><del>1</del></td><td>1</td><td><del>1</del></td><td><del>1</del></td></tr><tr><th>3</th><td>6</td><td>1</td><td>∞</td><td>1</td><td>1</td><td>3</td><td>2</td><td>1</td></tr><tr><th>4</th><td>5</td><td>5</td><td>1</td><td>∞</td><td>5</td><td>5</td><td>7</td><td>5</td></tr><tr><th>5</th><td>4</td><td>1</td><td>1</td><td>5</td><td>∞</td><td>3</td><td>2</td><td>5</td></tr><tr><th>6</th><td>4</td><td>1</td><td>3</td><td>5</td><td>3</td><td>∞</td><td>5</td><td>6</td></tr><tr><th>7</th><td>5</td><td>1</td><td>2</td><td>7</td><td>2</td><td>5</td><td>∞</td><td>1</td></tr><tr><th>8</th><td>5</td><td>1</td><td>1</td><td>5</td><td>5</td><td>6</td><td>1</td><td>∞</td></tr></table>		1	2	3	4	5	6	7	8	1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>	2	<del>5</del>	<del>∞</del>	<del>1</del>	<del>5</del>	<del>1</del>	1	<del>1</del>	<del>1</del>	3	6	1	∞	1	1	3	2	1	4	5	5	1	∞	5	5	7	5	5	4	1	1	5	∞	3	2	5	6	4	1	3	5	3	∞	5	6	7	5	1	2	7	2	5	∞	1	8	5	1	1	5	5	6	1	∞		1-6-2-8-7
	1	2	3	4	5	6	7	8																																																																											
1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>																																																																											
2	<del>5</del>	<del>∞</del>	<del>1</del>	<del>5</del>	<del>1</del>	1	<del>1</del>	<del>1</del>																																																																											
3	6	1	∞	1	1	3	2	1																																																																											
4	5	5	1	∞	5	5	7	5																																																																											
5	4	1	1	5	∞	3	2	5																																																																											
6	4	1	3	5	3	∞	5	6																																																																											
7	5	1	2	7	2	5	∞	1																																																																											
8	5	1	1	5	5	6	1	∞																																																																											

<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr><tr><th>1</th><td><del>∞</del></td><td><del>5</del></td><td><del>6</del></td><td><del>5</del></td><td><del>4</del></td><td><del>4</del></td><td><del>5</del></td><td><del>5</del></td></tr><tr><th>2</th><td><del>5</del></td><td><del>∞</del></td><td><del>1</del></td><td><del>5</del></td><td><del>1</del></td><td>1</td><td><del>1</del></td><td><del>1</del></td></tr><tr><th>3</th><td>6</td><td>1</td><td>∞</td><td>1</td><td>1</td><td>3</td><td>2</td><td>1</td></tr><tr><th>4</th><td>5</td><td>5</td><td>1</td><td>∞</td><td>5</td><td>5</td><td>7</td><td>5</td></tr><tr><th>5</th><td>4</td><td>1</td><td>1</td><td>5</td><td>∞</td><td>3</td><td>2</td><td>5</td></tr><tr><th>6</th><td>4</td><td>1</td><td>3</td><td>5</td><td>3</td><td>∞</td><td>5</td><td>6</td></tr><tr><th>7</th><td><del>5</del></td><td><del>1</del></td><td><del>2</del></td><td><del>7</del></td><td><del>2</del></td><td><del>5</del></td><td>∞</td><td>1</td></tr><tr><th>8</th><td><del>5</del></td><td>1</td><td>1</td><td>5</td><td>5</td><td>6</td><td>1</td><td>∞</td></tr></table>		1	2	3	4	5	6	7	8	1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>	2	<del>5</del>	<del>∞</del>	<del>1</del>	<del>5</del>	<del>1</del>	1	<del>1</del>	<del>1</del>	3	6	1	∞	1	1	3	2	1	4	5	5	1	∞	5	5	7	5	5	4	1	1	5	∞	3	2	5	6	4	1	3	5	3	∞	5	6	7	<del>5</del>	<del>1</del>	<del>2</del>	<del>7</del>	<del>2</del>	<del>5</del>	∞	1	8	<del>5</del>	1	1	5	5	6	1	∞		1-6-2-8- 7-5
	1	2	3	4	5	6	7	8																																																																											
1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>																																																																											
2	<del>5</del>	<del>∞</del>	<del>1</del>	<del>5</del>	<del>1</del>	1	<del>1</del>	<del>1</del>																																																																											
3	6	1	∞	1	1	3	2	1																																																																											
4	5	5	1	∞	5	5	7	5																																																																											
5	4	1	1	5	∞	3	2	5																																																																											
6	4	1	3	5	3	∞	5	6																																																																											
7	<del>5</del>	<del>1</del>	<del>2</del>	<del>7</del>	<del>2</del>	<del>5</del>	∞	1																																																																											
8	<del>5</del>	1	1	5	5	6	1	∞																																																																											
<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr><tr><th>1</th><td><del>∞</del></td><td><del>5</del></td><td><del>6</del></td><td><del>5</del></td><td><del>4</del></td><td><del>4</del></td><td><del>5</del></td><td><del>5</del></td></tr><tr><th>2</th><td><del>5</del></td><td><del>∞</del></td><td><del>1</del></td><td><del>5</del></td><td><del>1</del></td><td>1</td><td><del>1</del></td><td><del>1</del></td></tr><tr><th>3</th><td>6</td><td>1</td><td>∞</td><td>1</td><td>1</td><td>3</td><td>2</td><td>1</td></tr><tr><th>4</th><td>5</td><td>5</td><td>1</td><td>∞</td><td>5</td><td>5</td><td>7</td><td>5</td></tr><tr><th>5</th><td><del>4</del></td><td><del>1</del></td><td><del>1</del></td><td><del>5</del></td><td><del>∞</del></td><td><del>3</del></td><td>2</td><td><del>5</del></td></tr><tr><th>6</th><td>4</td><td>1</td><td>3</td><td>5</td><td>3</td><td>∞</td><td>5</td><td>6</td></tr><tr><th>7</th><td><del>5</del></td><td><del>1</del></td><td><del>2</del></td><td><del>7</del></td><td><del>2</del></td><td><del>5</del></td><td>∞</td><td>1</td></tr><tr><th>8</th><td><del>5</del></td><td>1</td><td>1</td><td>5</td><td>5</td><td>6</td><td>1</td><td>∞</td></tr></table>		1	2	3	4	5	6	7	8	1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>	2	<del>5</del>	<del>∞</del>	<del>1</del>	<del>5</del>	<del>1</del>	1	<del>1</del>	<del>1</del>	3	6	1	∞	1	1	3	2	1	4	5	5	1	∞	5	5	7	5	5	<del>4</del>	<del>1</del>	<del>1</del>	<del>5</del>	<del>∞</del>	<del>3</del>	2	<del>5</del>	6	4	1	3	5	3	∞	5	6	7	<del>5</del>	<del>1</del>	<del>2</del>	<del>7</del>	<del>2</del>	<del>5</del>	∞	1	8	<del>5</del>	1	1	5	5	6	1	∞		1-6-2-8- 7-5-3
	1	2	3	4	5	6	7	8																																																																											
1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>																																																																											
2	<del>5</del>	<del>∞</del>	<del>1</del>	<del>5</del>	<del>1</del>	1	<del>1</del>	<del>1</del>																																																																											
3	6	1	∞	1	1	3	2	1																																																																											
4	5	5	1	∞	5	5	7	5																																																																											
5	<del>4</del>	<del>1</del>	<del>1</del>	<del>5</del>	<del>∞</del>	<del>3</del>	2	<del>5</del>																																																																											
6	4	1	3	5	3	∞	5	6																																																																											
7	<del>5</del>	<del>1</del>	<del>2</del>	<del>7</del>	<del>2</del>	<del>5</del>	∞	1																																																																											
8	<del>5</del>	1	1	5	5	6	1	∞																																																																											
<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr><tr><th>1</th><td><del>∞</del></td><td><del>5</del></td><td><del>6</del></td><td><del>5</del></td><td><del>4</del></td><td><del>4</del></td><td><del>5</del></td><td><del>5</del></td></tr><tr><th>2</th><td><del>5</del></td><td><del>∞</del></td><td><del>1</del></td><td><del>5</del></td><td><del>1</del></td><td>1</td><td><del>1</del></td><td><del>1</del></td></tr><tr><th>3</th><td><del>6</del></td><td><del>1</del></td><td><del>∞</del></td><td><del>1</del></td><td>1</td><td>3</td><td>2</td><td>1</td></tr><tr><th>4</th><td>5</td><td>5</td><td>1</td><td>∞</td><td>5</td><td>5</td><td>7</td><td>5</td></tr><tr><th>5</th><td><del>4</del></td><td><del>1</del></td><td><del>1</del></td><td><del>5</del></td><td><del>∞</del></td><td><del>3</del></td><td>2</td><td><del>5</del></td></tr><tr><th>6</th><td>4</td><td>1</td><td>3</td><td>5</td><td>3</td><td>∞</td><td>5</td><td>6</td></tr><tr><th>7</th><td><del>5</del></td><td><del>1</del></td><td><del>2</del></td><td><del>7</del></td><td><del>2</del></td><td><del>5</del></td><td>∞</td><td>1</td></tr><tr><th>8</th><td><del>5</del></td><td>1</td><td>1</td><td>5</td><td>5</td><td>6</td><td>1</td><td>∞</td></tr></table>		1	2	3	4	5	6	7	8	1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>	2	<del>5</del>	<del>∞</del>	<del>1</del>	<del>5</del>	<del>1</del>	1	<del>1</del>	<del>1</del>	3	<del>6</del>	<del>1</del>	<del>∞</del>	<del>1</del>	1	3	2	1	4	5	5	1	∞	5	5	7	5	5	<del>4</del>	<del>1</del>	<del>1</del>	<del>5</del>	<del>∞</del>	<del>3</del>	2	<del>5</del>	6	4	1	3	5	3	∞	5	6	7	<del>5</del>	<del>1</del>	<del>2</del>	<del>7</del>	<del>2</del>	<del>5</del>	∞	1	8	<del>5</del>	1	1	5	5	6	1	∞		1-6-2-8- 7-5-3-4
	1	2	3	4	5	6	7	8																																																																											
1	<del>∞</del>	<del>5</del>	<del>6</del>	<del>5</del>	<del>4</del>	<del>4</del>	<del>5</del>	<del>5</del>																																																																											
2	<del>5</del>	<del>∞</del>	<del>1</del>	<del>5</del>	<del>1</del>	1	<del>1</del>	<del>1</del>																																																																											
3	<del>6</del>	<del>1</del>	<del>∞</del>	<del>1</del>	1	3	2	1																																																																											
4	5	5	1	∞	5	5	7	5																																																																											
5	<del>4</del>	<del>1</del>	<del>1</del>	<del>5</del>	<del>∞</del>	<del>3</del>	2	<del>5</del>																																																																											
6	4	1	3	5	3	∞	5	6																																																																											
7	<del>5</del>	<del>1</del>	<del>2</del>	<del>7</del>	<del>2</del>	<del>5</del>	∞	1																																																																											
8	<del>5</del>	1	1	5	5	6	1	∞																																																																											

	1	2	3	4	5	6	7	8
1	99	5	6	5	4	4	5	5
2	5	99	1	5	1	1	1	1
3	6	1	99	1	1	3	2	1
4	5	5	1	99	5	5	7	5
5	4	1	1	5	99	2	2	5
6	4	1	3	5	3	99	5	6
7	5	1	2	7	2	5	99	1
8	5	1	1	5	5	6	1	99

Ми пройшлися по всіх вершинах, але нам ще потрібно повернутись у початкову. Дивимось яка відстань від 4 до 1, та обводимо елемент 1 4.

1-6-2-8-7-5-3-4-1.

Тож кінцевий шлях: 1-6-2-8-7-5-3-4-5.

Програмна реалізація:

```
#include <iostream>
using namespace std;
#define SIZE 8

int FindMin(int* arr, int size)
{
    for (int i = 1; i < size; i++) {
        if (arr[0] > arr[i])
            arr[0] = arr[i];
    }
    return arr[0];
}

int main()
{
    int arr[SIZE][SIZE] = { 99,5,6,5,4,4,6,5,
                             5,99,1,5,1,1,1,1,
                             6,1,99,1,1,3,2,1,
                             5,5,1,99,5,5,7,5,
                             4,1,1,5,99,3,2,1,
                             4,1,3,5,3,99,5,6,
                             5,1,2,7,2,5,99,1,
                             5,1,1,5,5,6,1,99 };

    int tempArr[SIZE];
    int min;

    int num_of = 0;
    cout << 1 << " ";
    while (1) {
        for (int n = 0; n < SIZE; n++)
            arr[n][num_of] = 99;
        for (int i = 0; i < SIZE; i++)
        {
            tempArr[i] = arr[num_of][i];
        }
        min = FindMin(tempArr, SIZE);
        if (min == 99)
            break;
        for (int i = 0; i < SIZE; i++)
        {
            if (min == arr[num_of][i]) {
```

```

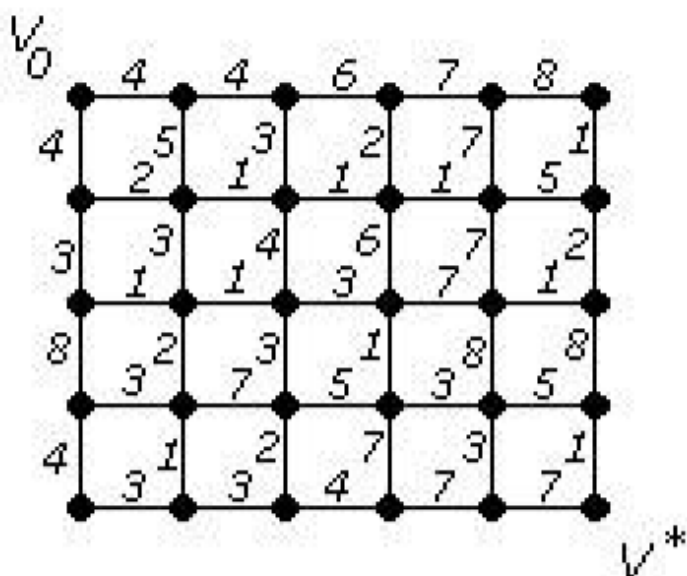
        num_of = i;
        cout << i + 1 << " ";
        break;
    }
}
cout << 1;
}

```

Результат виконання програми:

1 5 2 3 4 6 7 8 1

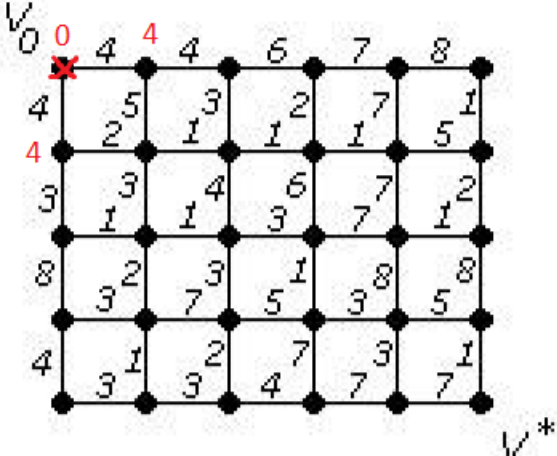
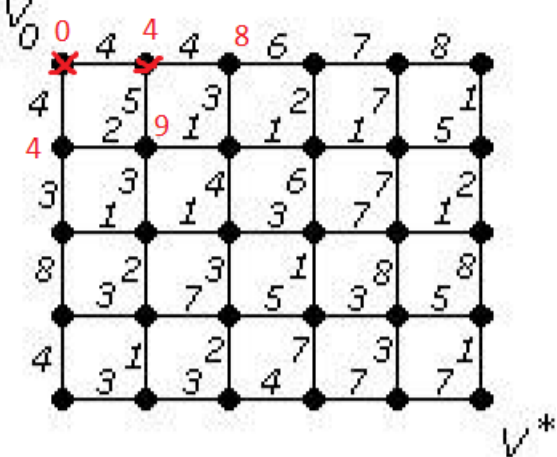
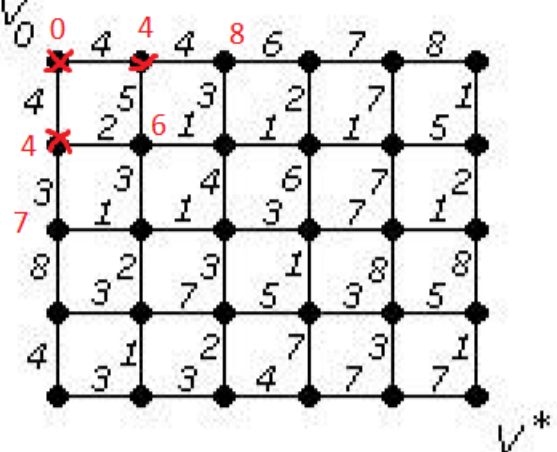
### Завдання № 7.



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

Розв'язок:

Крок	Малюнок	Пояснення
1		<p><math>V_0</math> – початкова вершина, тому позначаємо відстань до неї – 0.</p> <p>Червоним кольором будемо позначати найкоротшу відстань до <math>V_0</math> на даний момент.</p>

2		<p>Записуємо відстані до сусідніх вершин, та закреслюємо початкову вершину, оскільки всі її сусідні вершини перевірені.</p>
3		<p>Наступною вибирається сусідня точка, відстань до якої є найменшою. Оскільки є дві точки відстань від яких до <math>V_0</math> є однаковою і мінімальною, то довільним чином вибираємо одну з них. Перевіряємо відстань до її сусідів. Якщо відстань до сусідів, які вже мали мітки є меншою, то заміняєм стару мітку на нову, якщо більшою, то залишаємо стару. Діємо так допоки не перейдемося по всіх вершинах.</p>
4		



31		<p>Усі точки викреслено. Довжина мінімального шляху від <math>V_0</math> до <math>V^*</math> - 22. Також знайдено всі мінімальні відстані від будь-якої вершини до <math>V_0</math>.</p>
----	--	--

### Пошук мінімального шляху

1		<p>Щоб знайти сам шлях, починаємо рух з кінцевої точки та перевіряємо чи дорівнює мітка сусідньої вершини різниці відстаней мітки поточної вершини та відстані між цими вершинами. Для прикладу <math>21 = 22 - 1</math>, отже замальовуємо цей шлях синім кольором, але іншого сусіда перевіряємо також, оскільки мінімальний шлях може бути не один. Діємо так поки не дійдемо до вершини <math>V_0</math>. Наступна перевірка виконується з вершин які ми щойно знайшли.</p>
2		

3		
9		<p>Ми повернулися до вершини V<sub>0</sub>. Алгоритм закінчено. Мінімальний шлях знайдено.</p>

### Програмна реалізація:

```
#include <iostream>
#include <fstream>
using namespace std;

//Функція виведення мінімальної відстані до вершини та запису мінімальних значень в масив min_ways
void PrintArr_FillMin(int* arr, int num_of, int* min_ways)
{
    cout << "\t\tVertex#" << num_of + 1 << " minimal way - |" << arr[num_of] << "|" << endl;
    min_ways[num_of] = arr[num_of];
}

//Функція знаходження мінімального значення
int FindMin(int* arr, int SIZE)
{
    for (int i = 1; i < SIZE; i++) {
        if (arr[0] > arr[i])
            arr[0] = arr[i];
    }
    return arr[0];
}

int main()
{
    setlocale(LC_ALL, "Ukr");

    const int SIZE = 30;
    int end_arr[SIZE];
    int min_ways[SIZE];
    min_ways[0] = 0;
```

```

int arr1[SIZE][SIZE];
int arr2[SIZE][SIZE];

for (int i = 0; i < SIZE; i++) { //Ініціалізація масивів початковими значеннями
    for (int j = 0; j < SIZE; j++) {
        arr1[i][j] = 99;
        arr2[i][j] = 99;
    }
}

int num_of_edges = 49;
int weight = 0, vertex1 = 0, vertex2 = 0;

ifstream fin("FILE.txt"); //Зчитування даних про граф з файлу та заповнення масиву
if (!fin.is_open())
    cout << "Error\n";
else {
    for (int i = 0; i < num_of_edges; i++) { //arr2 це копія масиву arr1
        fin >> weight;
        fin >> vertex1;
        fin >> vertex2;
        arr1[vertex1 - 1][vertex2 - 1] = weight;
        arr1[vertex2 - 1][vertex1 - 1] = weight;
        arr2[vertex1 - 1][vertex2 - 1] = weight;
        arr2[vertex2 - 1][vertex1 - 1] = weight;
    }
}
fin.close();

int* current_min = new int[SIZE]; //Створення динамічного масиву
for (int i = 1; i < SIZE; i++)
{
    current_min[i] = 99;
}
current_min[0] = 0;

int find_min[100], min_SIZE = 0, step = 1, num_of_edge = 0, minimum = 0, counter = 0,
counter1 = 0;
int* chosens = new int[SIZE]; //Створення динамічного масиву
bool end = false;

while (true)
{
    for (int n = 0; n < SIZE; n++) //Видалення потрібних стовпців
        arr1[n][num_of_edge] = 0;

    for (int i = 0; i < SIZE; i++) {
        if (arr1[num_of_edge][i] > 0)
        {
            if (current_min[i] > arr1[num_of_edge][i] + minimum)
                current_min[i] = arr1[num_of_edge][i] + minimum;
        }
    }
    end = true;
    min_SIZE = 0;

    for (int i = 0; i < SIZE; i++) { // Знаходження можливих наступних мінімальних
ребер
        if (current_min[i] > 0 && current_min[i] != 99)
        {
            find_min[min_SIZE] = current_min[i];
            min_SIZE++;
            end = false;
        }
    }
    if (end)
        goto link;
}

```

```

        minimum = FindMin(find_min, min_SIZE); //Знаходження мінімального значення з
них
        for (int i = 0; i < SIZE; i++) { //Прирівнювання значень та знаходження
вибраного ребра та номеру рядка
            if (current_min[i] == minimum)
            {
                num_of_edge = i;
                chosens[counter] = i;
                counter++;
                break;
            }
        }
        cout << "Step " << step;
        step++;
        PrintArr_FillMin(current_min, num_of_edge, min_ways);
        current_min[num_of_edge] = 0;
    }
link:
    cout << "\nМінімальний шлях: "; // Вивід мінімального шляху
    int i = SIZE - 1;
    while (true) {
        for (int j = SIZE - 1; j >= 0; j--){
            if (min_ways[i] - min_ways[j] == arr2[i][j])
            {
                end_arr[counter1] = j + 1;
                counter1++;
                i = j;
                if (i == 0)
                    goto link2;
            }
        }
    }
link2:
    for ( i = counter1-1; i >= 0; i--)
    {
        cout << end_arr[i]<<" ";
    }
    cout << SIZE;
    delete[] current_min;
    delete[] chosens;
    return 0;
}

```

**Результат виконання програми:**

```

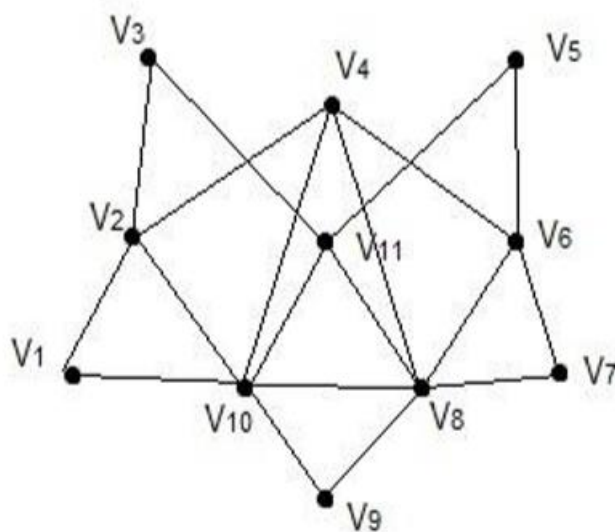
Step 1      Vertex#2 minimal way - |4|
Step 2      Vertex#7 minimal way - |4|
Step 3      Vertex#8 minimal way - |6|
Step 4      Vertex#9 minimal way - |7|
Step 5      Vertex#13 minimal way - |7|
Step 6      Vertex#3 minimal way - |8|
Step 7      Vertex#10 minimal way - |8|
Step 8      Vertex#14 minimal way - |8|
Step 9      Vertex#11 minimal way - |9|
Step 10     Vertex#15 minimal way - |9|
Step 11     Vertex#4 minimal way - |10|
Step 12     Vertex#20 minimal way - |10|
Step 13     Vertex#26 minimal way - |11|
Step 14     Vertex#16 minimal way - |12|
Step 15     Vertex#21 minimal way - |12|
Step 16     Vertex#19 minimal way - |13|
Step 17     Vertex#22 minimal way - |13|
Step 18     Vertex#12 minimal way - |14|
Step 19     Vertex#25 minimal way - |14|
Step 20     Vertex#27 minimal way - |14|
Step 21     Vertex#6 minimal way - |15|
Step 22     Vertex#5 minimal way - |16|
Step 23     Vertex#17 minimal way - |16|
Step 24     Vertex#18 minimal way - |16|
Step 25     Vertex#23 minimal way - |16|
Step 26     Vertex#28 minimal way - |18|
Step 27     Vertex#29 minimal way - |19|
Step 28     Vertex#24 minimal way - |21|
Step 29     Vertex#30 minimal way - |22|

Мінімальний шлях: 1 7 13 14 15 16 22 23 24 30

```

## Завдання № 8

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

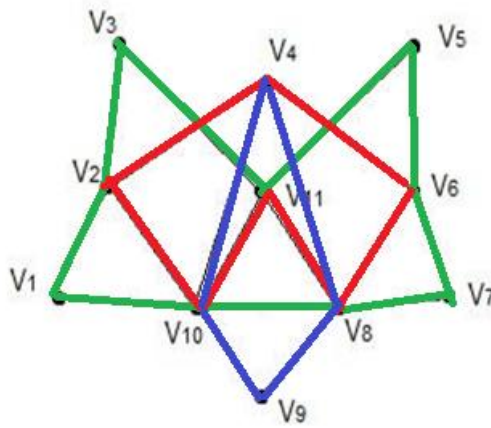


### Розв'язок:

А) Будемо видаляти ребра одне за одним, перевіряючи кожен раз перед цим чи не є це ребро мостом.

Отож шлях ейлерового циклу:  $v_2 v_{10} v_1 v_2 v_3 v_{11} v_5 v_6 v_7 v_8 v_{11} v_{10} v_9 v_8 v_4 v_{10} v_8 v_6 v_4 v_2$ . Всі ребра пройдено і ми повернулись у початкову вершину.

Б) Поступово будемо виділяти елементарні цикли та об'єднувати їх. Виділимо різні елементарні цикли різними кольорами.



Шлях:  $v_{10} v_1 v_2 v_3 v_{11} v_5 v_6 v_7 v_8 v_{10} v_4 v_8 v_9 v_{10} v_2 v_4 v_6 v_8 v_{11} v_{10}$ .

### Програмна реалізація:

```
/*  
Алгоритм Флері для знаходження ейлерового циклу  
*/  
#include<iostream>  
#include<vector>  
#define NODE 11  
using namespace std;  
int graph[NODE][NODE]  
= { {0,1,0,0,0,0,0,0,0,1,0},  
    {1,0,1,1,0,0,0,0,0,1,0},  
    {0,1,0,0,0,0,0,0,0,0,1},  
    {0,1,0,0,0,1,0,1,0,1,0},  
    {0,0,0,0,0,1,0,0,0,0,1},  
    {0,0,0,1,1,0,1,1,0,0,0},  
    {0,0,0,0,0,1,0,1,0,0,0},  
    {0,0,0,0,0,1,1,0,1,1,1},  
    {0,0,0,0,0,0,0,1,0,1,0},  
    {1,1,0,0,0,0,0,1,1,0,1},  
    {0,0,1,0,1,0,0,1,0,1,0},  
};  
int tempGraph[NODE][NODE];  
int findStartVert() {  
    for (int i = 0; i < NODE; i++) {
```

```

        int deg = 0;
        for (int j = 0; j < NODE; j++) {
            if (tempGraph[i][j])
                deg++;
        }
        if (deg % 2 != 0)
            return i;
    }
    return 0;
}

bool isBridge(int u, int v) {
    int deg = 0;
    for (int i = 0; i < NODE; i++)
        if (tempGraph[v][i])
            deg++;
    if (deg > 1) {
        return false;
    }
    return true;
}

int edgeCount() {
    int count = 0;
    for (int i = 0; i < NODE; i++)
        for (int j = i; j < NODE; j++)
            if (tempGraph[i][j])
                count++;
    return count;
}

void fleuryAlgorithm(int start) {
    static int edge = edgeCount();
    for (int v = 0; v < NODE; v++) {
        if (tempGraph[start][v]) {
            if (edge <= 1 || !isBridge(start, v)) {
                cout << start + 1 << "--" << v + 1 << " ";
                tempGraph[start][v] = tempGraph[v][start] = 0;
                edge--;
                fleuryAlgorithm(v);
            }
        }
    }
}

int main() {
    for (int i = 0; i < NODE; i++)
        for (int j = 0; j < NODE; j++)
            tempGraph[i][j] = graph[i][j];
    fleuryAlgorithm(findStartVert());
}

```

Результат виконання програми:

```

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

```

## Завдання № 9

Спростити формулу (привести до скороченої ДНФ).

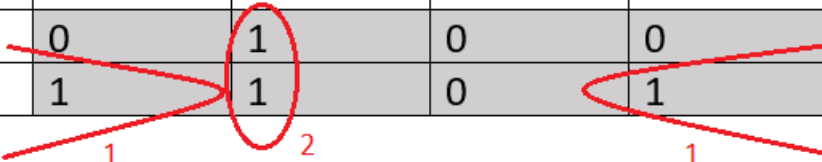
$\bar{x}y \vee x\bar{y}\bar{z}$

Розв'язок:

Побудуємо карту Карно

z	0	0	1	1
y\ x	0	1	1	0
0	0	1	0	0
1	1	1	0	1

z	0	0	1	1
y\ x	0	1	1	0
0	0	1	0	0
1	1	1	0	1



Виділяємо прямокутні групи одиниць.

В першій групі незмінними є  $x$  та  $y$ , вони дорівнювали 0 та 1 відповідно, в другій –  $x$  та  $z$ , вони дорівнювали 1 та 0.

Отже скорочена ДНФ буде виглядати так:

$$\bar{x}y \vee x\bar{z}$$