

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

Кафедра систем штучного інтелекту

Лабораторна робота № 5

з дисципліни

«Дискретна математика»

Виконав:

Студент групи КН-113

Волошин Володимир

Викладач:

Мельникова Н.І.

Львів – 2019р.

Тема: Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи.

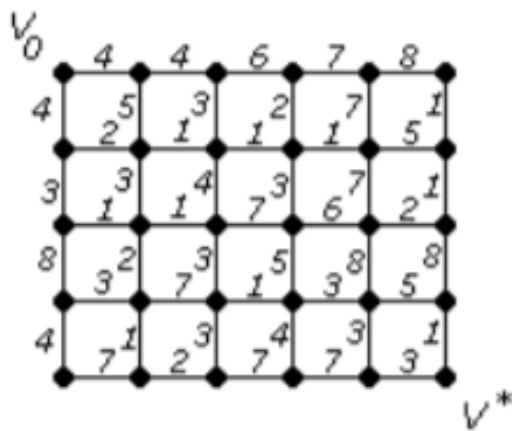
Мета роботи: набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

Варіант 7

Завдання № 1.

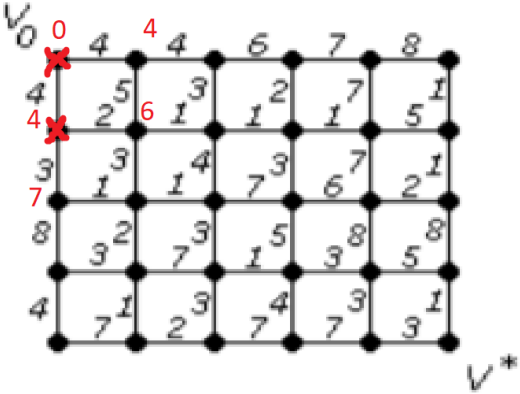
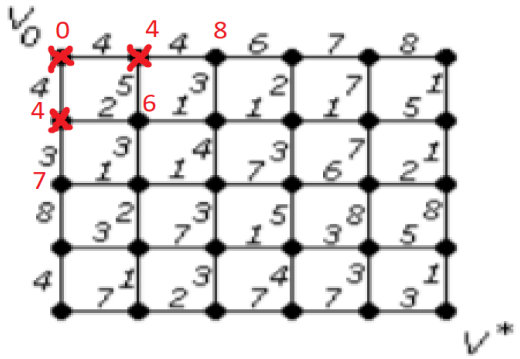
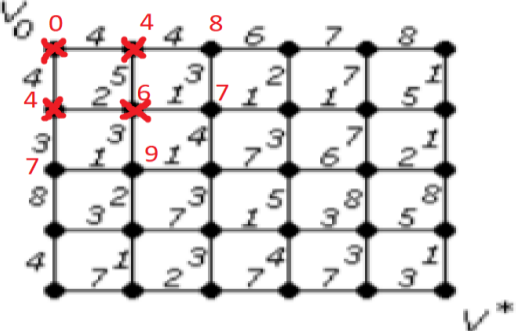
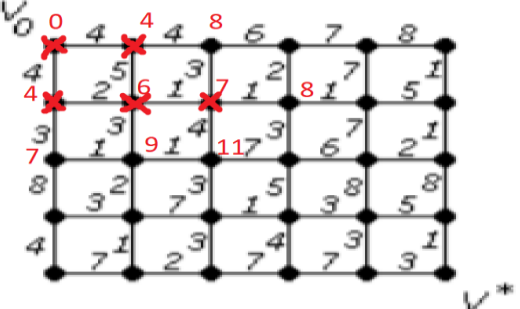
Розв'язати на графах наступні 2 задачі:

1. За допомогою алгоритму Дейкстра знайти найкоротший шлях у графі поміж парою вершин V_0 і V^*



Розв'язання:

Крок	Малюнок	Пояснення
1		<p>V_0 – початкова вершина, тому позначаємо відстань до неї – 0.</p> <p>Червоним кольором будемо позначати найкоротшу відстань до V_0 на даний момент.</p>
2		<p>Записуємо відстані до сусідніх вершин, та закреслюємо початкову вершину, оскільки всі її сусідні вершини перевірені.</p>

3		<p>Оскільки у V_0 дві суміжні ребра мають мінімальний шлях 4, то довільним чином було вибрана одна з сусідніх вершин та знайдено мінімальні відстані до сусідніх вершин вже цієї точки. Коли всі сусідні вершини перевірені, точка закреслюється. Точки, які вже викреслені, більше не перевіряються.</p>
4		<p>Серед не закреслених вершин було вибрану вершину з найменшою міткою. Від неї було знайдено відстані до сусідніх вершин. Сусідня нижня вершин мала б отримати мітку 9, але оскільки нова мітка більша за стару, то ми залишаємо стару. Якщо ж нова мітка менша за стару, то пишемо нову, видаливши стару. Так будемо діяти далі і до кінця.</p>
5		
6		

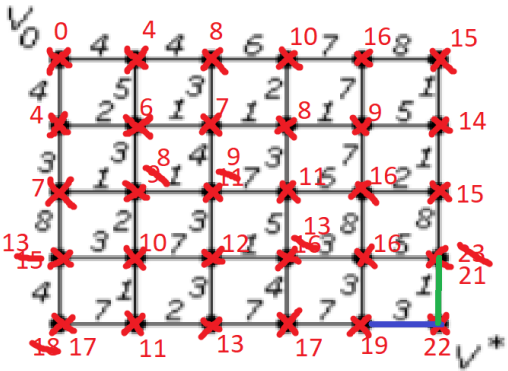
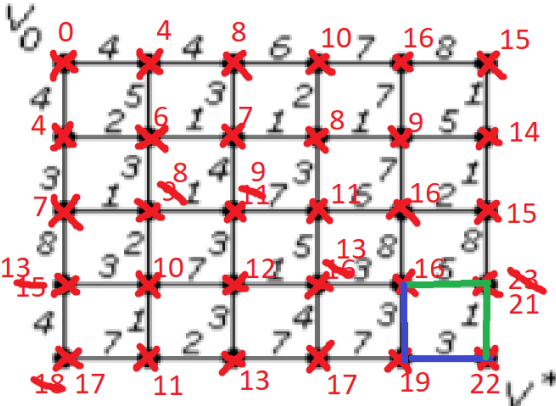
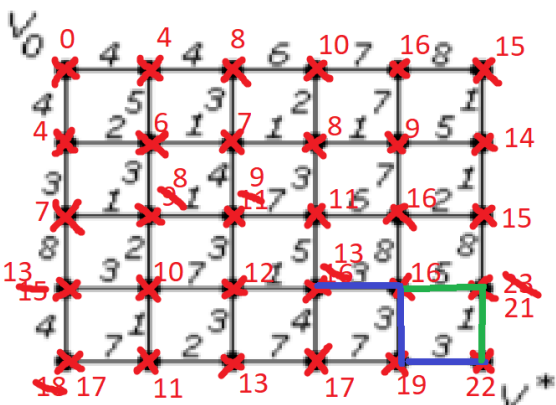
7		Нова мітка виявилась меншою за стару, тож стару ми видалили і записали нову.
8		
9		
10		
11		

12		
13		
14		
15		
16		

17	<p>A 5x5 grid of numbers. Red 'X' marks are placed on the top-left, top-middle, top-right, middle-left, middle-middle, middle-right, bottom-left, bottom-middle, and bottom-right cells. The numbers are: Row 1: 0, 4, 4, 8, 10, 16, 8; Row 2: 4, 2, 5, 6, 1, 3, 7, 2, 8, 7, 1, 9, 5, 14; Row 3: 3, 3, 3, 8, 4, 9, 3, 11, 16, 2, 1; Row 4: 8, 3, 2, 10, 7, 3, 12, 5, 13, 8, 5, 8; Row 5: 4, 7, 1, 2, 3, 7, 4, 7, 3, 3, 1. Labels: V_0 (top-left), V^* (bottom-right), 13 (bottom-left), 15 (bottom-left), 18 (bottom-left), 11 (bottom-middle), 13 (bottom-middle).</p>	
18	<p>A 5x5 grid of numbers. Red 'X' marks are placed on the top-left, top-middle, top-right, middle-left, middle-middle, middle-right, bottom-left, bottom-middle, and bottom-right cells. The numbers are: Row 1: 0, 4, 4, 8, 10, 16, 8; Row 2: 4, 2, 5, 6, 1, 3, 7, 2, 8, 7, 1, 9, 5, 14; Row 3: 3, 3, 3, 8, 4, 9, 3, 11, 16, 2, 1; Row 4: 8, 3, 2, 10, 7, 3, 12, 5, 13, 8, 5, 8; Row 5: 4, 7, 1, 2, 3, 7, 4, 7, 3, 3, 1. Labels: V_0 (top-left), V^* (bottom-right), 13 (bottom-left), 15 (bottom-left), 18 (bottom-left), 11 (bottom-middle), 13 (bottom-middle), 17 (bottom-right).</p>	
19	<p>A 5x5 grid of numbers. Red 'X' marks are placed on the top-left, top-middle, top-right, middle-left, middle-middle, middle-right, bottom-left, bottom-middle, and bottom-right cells. The numbers are: Row 1: 0, 4, 4, 8, 10, 16, 8; Row 2: 4, 2, 5, 6, 1, 3, 7, 2, 8, 7, 1, 9, 5, 14; Row 3: 3, 3, 3, 8, 4, 9, 3, 11, 16, 2, 1; Row 4: 8, 3, 2, 10, 7, 3, 12, 5, 13, 8, 5, 8; Row 5: 4, 7, 1, 2, 3, 7, 4, 7, 3, 3, 1. Labels: V_0 (top-left), V^* (bottom-right), 13 (bottom-left), 15 (bottom-left), 18 (bottom-left), 17 (bottom-left), 11 (bottom-middle), 13 (bottom-middle), 17 (bottom-right).</p>	
20	<p>A 5x5 grid of numbers. Red 'X' marks are placed on the top-left, top-middle, top-right, middle-left, middle-middle, middle-right, bottom-left, bottom-middle, and bottom-right cells. The numbers are: Row 1: 0, 4, 4, 8, 10, 16, 8; Row 2: 4, 2, 5, 6, 1, 3, 7, 2, 8, 7, 1, 9, 5, 14; Row 3: 3, 3, 3, 8, 4, 9, 3, 11, 16, 2, 1; Row 4: 8, 3, 2, 10, 7, 3, 12, 5, 13, 8, 5, 8; Row 5: 4, 7, 1, 2, 3, 7, 4, 7, 3, 3, 1. Labels: V_0 (top-left), V^* (bottom-right), 13 (bottom-left), 15 (bottom-left), 18 (bottom-left), 17 (bottom-left), 11 (bottom-middle), 13 (bottom-middle), 17 (bottom-right).</p>	
21	<p>A 5x5 grid of numbers. Red 'X' marks are placed on the top-left, top-middle, top-right, middle-left, middle-middle, middle-right, bottom-left, bottom-middle, and bottom-right cells. The numbers are: Row 1: 0, 4, 4, 8, 10, 16, 8, 15; Row 2: 4, 2, 5, 6, 1, 3, 7, 2, 8, 7, 1, 9, 5, 14; Row 3: 3, 3, 3, 8, 4, 9, 3, 11, 16, 2, 1; Row 4: 8, 3, 2, 10, 7, 3, 12, 5, 13, 8, 5, 8; Row 5: 4, 7, 1, 2, 3, 7, 4, 7, 3, 3, 1. Labels: V_0 (top-left), V^* (bottom-right), 13 (bottom-left), 15 (bottom-left), 18 (bottom-left), 17 (bottom-left), 11 (bottom-middle), 13 (bottom-middle), 17 (bottom-right), 15 (bottom-right).</p>	

22		
23		
24		
25		
26		

27		
28		
29		
30		
31		<p>Усі точки викреслено.</p> <p>Довжина мінімального шляху від V_0 до V^* - 22.</p> <p>Також знайдено всі мінімальні відстані від будь-якої вершини до V_0.</p>

32		<p>Щоб знайти сам шлях, починаємо рух з кінцевої точки та перевіряємо чи дорівнює мітка сусідньої вершини 22 відняти відстань між цими сусідніми вершинами.</p> <p>Для прикладу $19 = 22 - 3$, отже замальовуємо цей шлях синім кольором, але наступного сусіда перевіряємо також, оскільки мінімальний шлях може бути не один. $21 = 22 - 1$, отже і це ребро замальовуємо, вже зеленим кольором. Діємо так поки не дійдемо до вершини V_0. Наступна перевірка виконується з вершин які ми щойно знайшли.</p>
33		
34		

35	<p>A 7x7 grid with numbers 0-22 and red 'X' marks. A blue path connects (3,3) to (3,6) to (4,6) to (4,5) to (4,4) to (4,3) to (4,2) to (4,1) to (4,0). A green path connects (4,6) to (5,6) to (5,5) to (5,4) to (5,3) to (5,2) to (5,1) to (5,0). The grid is labeled V_0 at the top-left and V^* at the bottom-right.</p>	
36	<p>A 7x7 grid with numbers 0-22 and red 'X' marks. A blue path connects (3,3) to (3,6) to (4,6) to (4,5) to (4,4) to (4,3) to (4,2) to (4,1) to (4,0). A green path connects (4,6) to (5,6) to (5,5) to (5,4) to (5,3) to (5,2) to (5,1) to (5,0). The grid is labeled V_0 at the top-left and V^* at the bottom-right.</p>	
37	<p>A 7x7 grid with numbers 0-22 and red 'X' marks. A blue path connects (3,3) to (3,6) to (4,6) to (4,5) to (4,4) to (4,3) to (4,2) to (4,1) to (4,0). A green path connects (4,6) to (5,6) to (5,5) to (5,4) to (5,3) to (5,2) to (5,1) to (5,0). The grid is labeled V_0 at the top-left and V^* at the bottom-right.</p>	
38	<p>A 7x7 grid with numbers 0-22 and red 'X' marks. A blue path connects (3,3) to (3,6) to (4,6) to (4,5) to (4,4) to (4,3) to (4,2) to (4,1) to (4,0). A green path connects (4,6) to (5,6) to (5,5) to (5,4) to (5,3) to (5,2) to (5,1) to (5,0). The grid is labeled V_0 at the top-left and V^* at the bottom-right.</p>	

39		
40		<p>Ми повернулися до вершини V_0. Алгоритм закінчено. Знайдено два мінімальні шляхи.</p>

2. За допомогою γ -алгоритма зробити укладку графа у площині, або довести що вона неможлива.

Розв'язання:

Виділяємо довільний цикл та виписуємо всі сегменти (Контактні вершини обведені прямокутниками). Починаємо розміщати наші сегменти. Після кожного кроку викреслюємо сегменти які вже розміщені.

Крок	Граф	Сегменти
1		

2		
3		
4		
5		
6		

7		
8		

Граф укладено.

Завдання №2.

Написати програму, яка реалізує алгоритм Дейкстри знаходження найкоротшого шляху між парою вершин у графі. Протестувати розроблену програму на графі згідно свого варіанту.

Програмна реалізація:

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

//Функція виведення даних масиву та запису мінімальних значень в масив min_ways
void PrintArr_FillMin(int* arr, int SIZE, int num_of, int* min_ways)
{
    for (int i = 0; i < SIZE; i++) {
        if(i == num_of)
            cout << "|" << arr[i] << "| ";
        else
            cout << arr[i] << " ";
    }
    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 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] = 0;
        }
    }

    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;
    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;

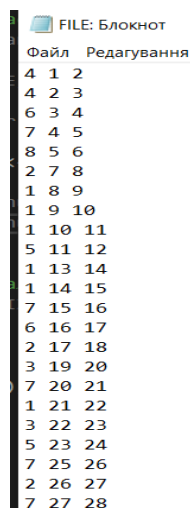
ребер
for (int i = 0; i < SIZE; i++) { // Знаходження можливих наступних мінімальних
    if (current_min[i] > 0 && current_min[i] != 99)
    {
        find_min[min_SIZE] = current_min[i];
        min_SIZE++;
        end = false;
    }
}
if (end)
    goto link;

них
minimum = FindMin(find_min, min_SIZE); // Знаходження мінімального значення з

for (int i = 0; i < SIZE; i++) { // Прирівнювання значень та знаходження
    if (current_min[i] == minimum) {
        num_of_edge = i;
        chosens[counter] = i;
        counter++;
        break;
    }
}
cout << "Step " << step << endl;
step++;
PrintArr_FillMin(current_min, SIZE, 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])
        {
            cout << j + 1 << " ";
            i = j;
            if (i == 0)
                return 0;
        }
    }
}
return 0;
}

```

Дані в файлі:



```

FILE: Блокнот
Файл  Редагування
4 1 2
4 2 3
6 3 4
7 4 5
8 5 6
2 7 8
1 8 9
1 9 10
1 10 11
5 11 12
1 13 14
1 14 15
7 15 16
6 16 17
2 17 18
3 19 20
7 20 21
1 21 22
3 22 23
5 23 24
7 25 26
2 26 27
7 27 28

```

Результат виконання програми:

```
0 0 0 0 16 99 0 0 0 0 0 14 0 0 0 0 16 99 0 0 0 |13| 99 99 17 0 13 99 99 99      Vertex#22 minimal way - |13|
Step 18
0 0 0 0 16 99 0 0 0 0 0 14 0 0 0 0 16 99 0 0 0 0 16 99 17 0 |13| 17 99 99      Vertex#27 minimal way - |13|
Step 19
0 0 0 0 16 99 0 0 0 0 0 |14| 0 0 0 0 16 99 0 0 0 0 16 99 17 0 0 17 99 99      Vertex#12 minimal way - |14|
Step 20
0 0 0 0 16 |15| 0 0 0 0 0 0 0 0 0 0 16 15 0 0 0 0 16 99 17 0 0 17 99 99      Vertex#6 minimal way - |15|
Step 21
0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 16 |15| 0 0 0 0 16 99 17 0 0 17 99 99      Vertex#18 minimal way - |15|
Step 22
0 0 0 0 |16| 0 0 0 0 0 0 0 0 0 0 16 0 0 0 0 0 16 23 17 0 0 17 99 99      Vertex#5 minimal way - |16|
Step 23
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16| 0 0 0 0 0 16 23 17 0 0 17 99 99      Vertex#17 minimal way - |16|
Step 24
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16| 23 17 0 0 17 99 99      Vertex#23 minimal way - |16|
Step 25
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 |17| 0 0 17 19 99      Vertex#25 minimal way - |17|
Step 26
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 |17| 19 99      Vertex#28 minimal way - |17|
Step 27
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 |19| 99      Vertex#29 minimal way - |19|
Step 28
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21| 0 0 0 0 0 22      Vertex#24 minimal way - |21|
Step 29
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |22|      Vertex#30 minimal way - |22|

Мінімальний шлях: 30 29 23 22 21 15 14 13 9 8 7 2 1
E:\Progects VS\Discrete5\Debug\Discrete5.exe (process 13136) exited with code 0.
```

Висновок: Виконуючи цю лабораторну роботу я набув практичних вмінь та навичок з використання алгоритму Дейкстри та укладання плоских планарні графів.