МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА“

Кафедра систем штучного інтелекту

Лабораторна робота №5

З дисципліни
«Дискретна математика»

Виконала:
Студентка групи КН-115
Галік Вікторія
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.

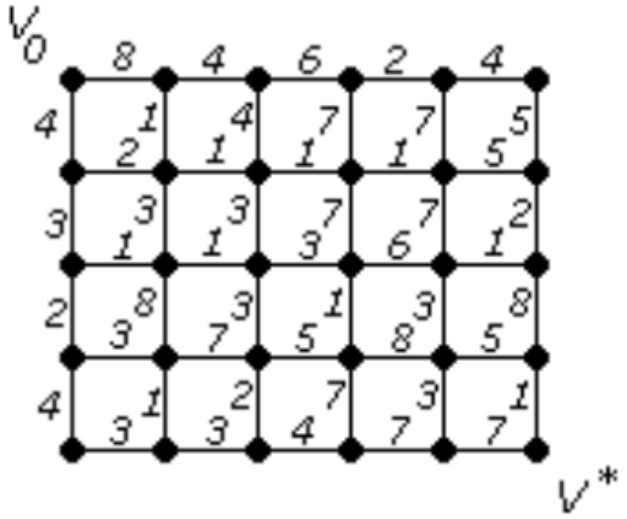
Варіант 5

Індивідуальні завдання :

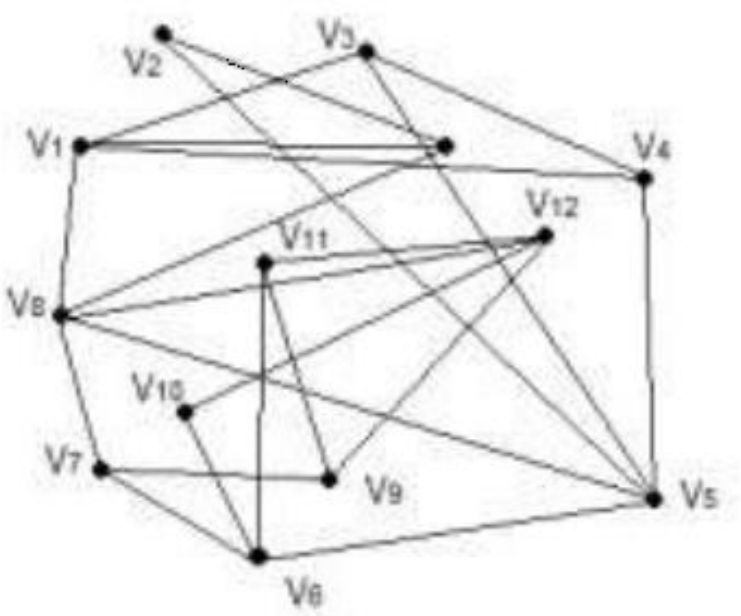
Завдання № 1.

Розв'язати на графах наступні 2 задачі:

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

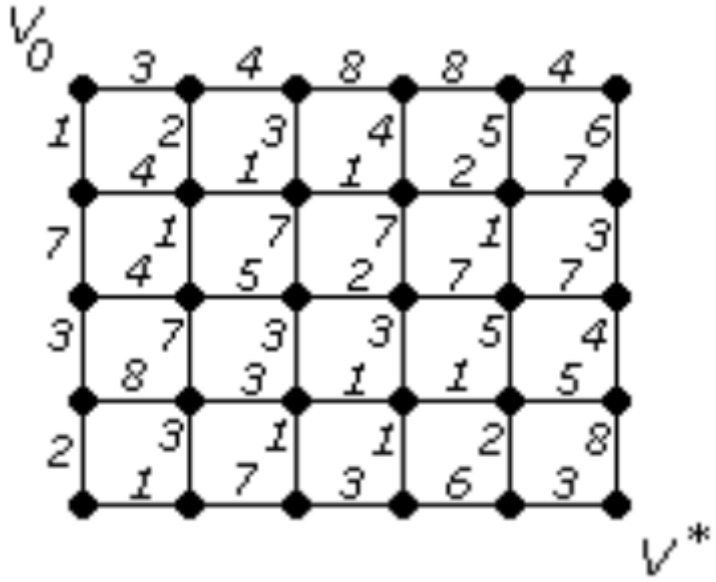


2. За допомогою γ -алгоритма зробити укладку графа у площині, або довести що вона неможлива.



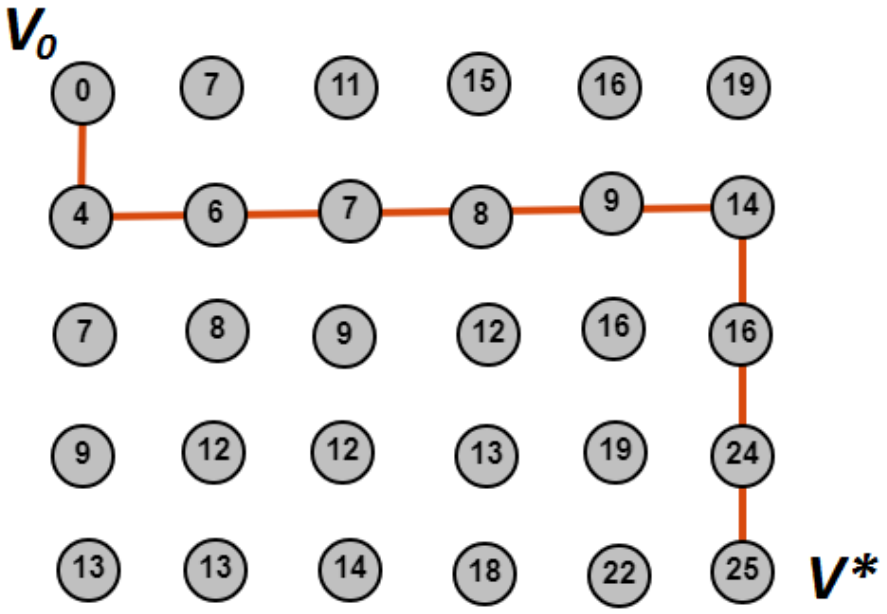
Завдання №2.

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



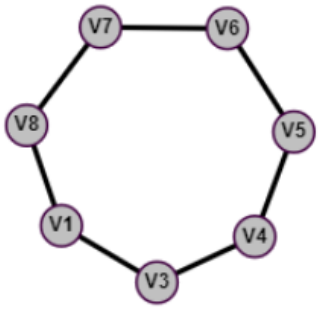
Розв’язання :

Завдання 1.1

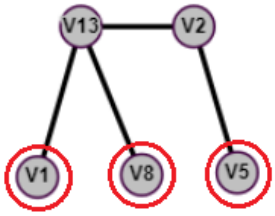


Найкоротша відстань : 4 + 2 +1 +1 + 1 + 5 + 2 + 8 + 1 **= 25**

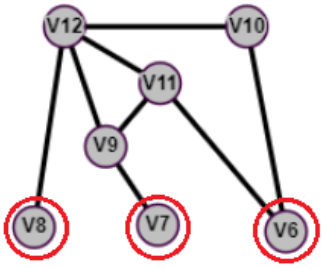
Завдання 1.2



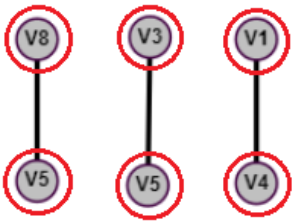
$\tilde{G}$



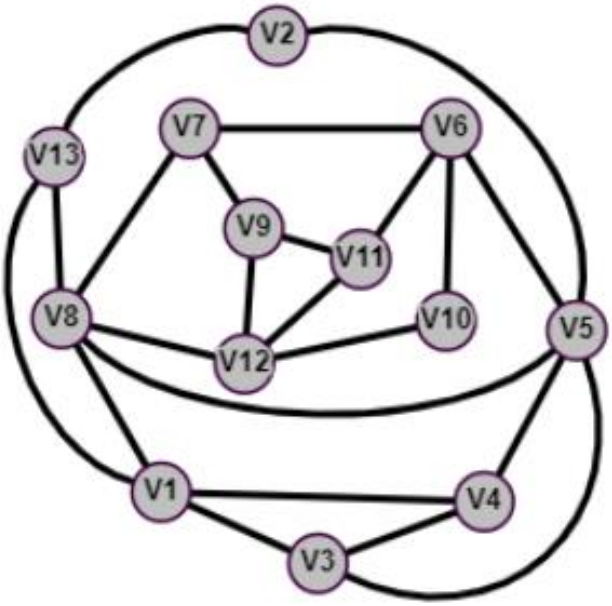
S1



S2



S3 *S4* *S5*



$\tilde{G} \cup S1 \cup S2 \cup S3 \cup S4 \cup S5$

Завдання №2

Програмна реалізація :

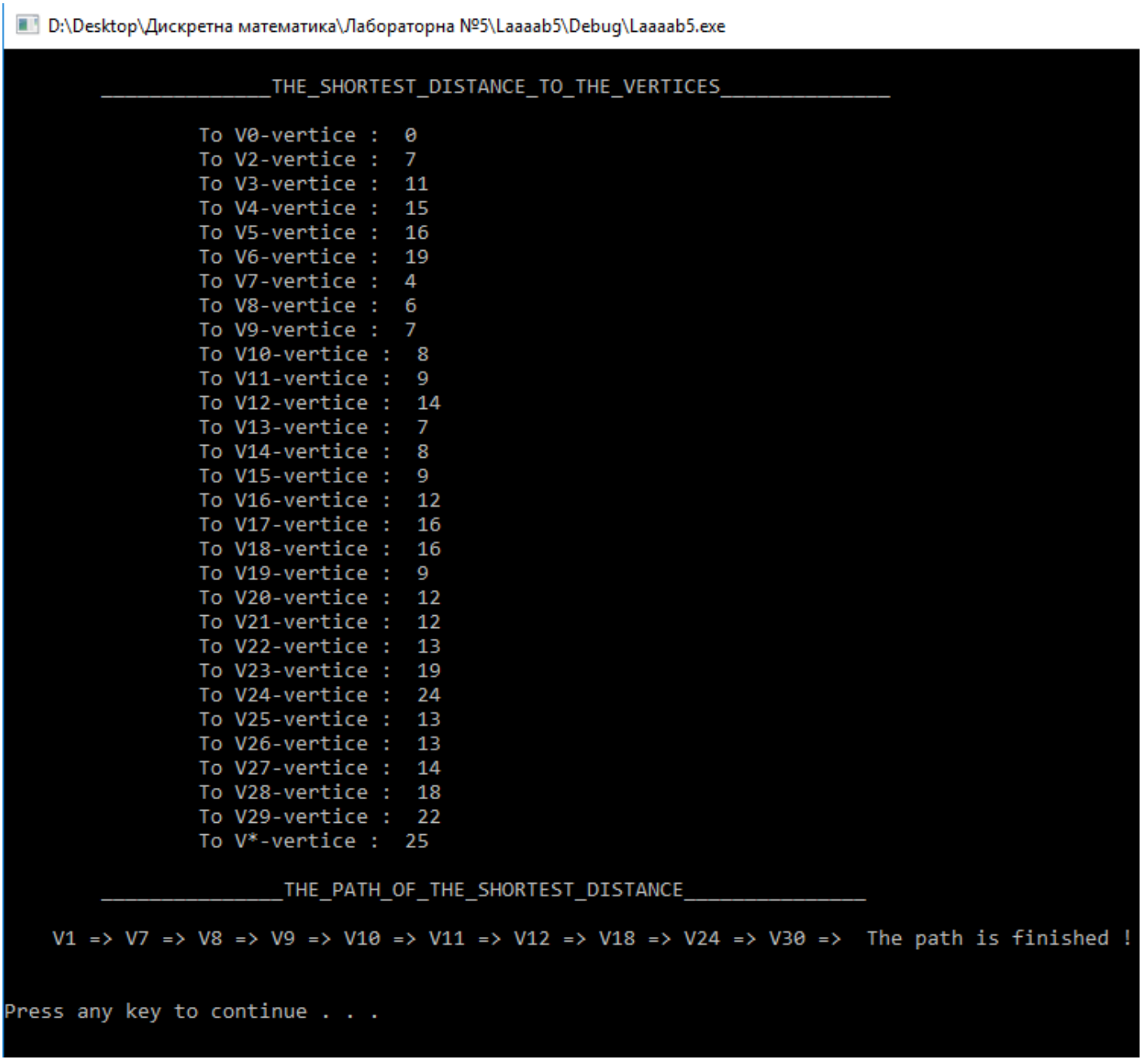
```
#include <iostream>
#define SIZE 30
#define INF 1000
using namespace std;

int main()
{
	int adjacency[SIZE][SIZE] = { {0,8,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{8,0,4,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,4,0,6,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,6,0,2,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,2,0,4,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,4,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{4,0,0,0,0,0,0,2,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
```

```
{0,1,0,0,0,0,2,0,1,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,4,0,0,0,0,1,0,1,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,7,0,0,0,0,1,0,1,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,7,0,0,0,0,1,0,5,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,5,0,0,0,0,5,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,3,0,0,0,0,0,0,1,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,3,0,0,0,0,1,0,1,0,0,0,0,8,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,3,0,0,0,0,1,0,3,0,0,0,0,3,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,7,0,0,0,0,3,0,6,0,0,0,0,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,6,0,1,0,0,0,0,3,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,1,0,0,0,0,0,0,8,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,3,0,0,0,0,4,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,3,0,7,0,0,0,0,1,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,7,0,5,0,0,0,0,2,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,5,0,8,0,0,0,0,7,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,8,0,5,0,0,0,0,3,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,5,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,0,3,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,3,0,3,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,3,0,4,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,4,0,7,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,7,0,7},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,7,0},   };
	int mindistance[SIZE];				// мінімальна відстань
	int visited[SIZE];				// пройдені вершини
	int min, minindex, temp;
	int begin_index = 0;				// початкова вершина

	for (int i = 0; i < SIZE; i++)
	{
		mindistance[i] = INF;			// ініціалізація відстаней як невідомих
		visited[i] = 1;				// 1 - непройдена вершина
	}
	mindistance[begin_index] = 0;			// відстань до початкової вершини

	do {
		minindex = INF;
		min = INF;
		for (int i = 0; i < SIZE; i++)
		{
			if ((visited[i] == 1) && (mindistance[i] < min))		//перевірка чи
вершина пройдена
			{
				min = mindistance[i];					//
переприсвоєння мінімального значення
				minindex = i;
```

```
                }
            }

            if (minindex != INF)
            {
                for (int i = 0; i < SIZE;  i++)
                {
                    if (adjacency[minindex][i] > 0)
                    {
                        temp = min + adjacency[minindex][i];                    // додаємо
знайдену мінімальну відстань до поточної відстані
                        if (temp < mindistance[i])
                        {
                            mindistance[i] = temp;                              //
остаточна мінімальна відстань до i-тої вершини
                        }
                    }
                }
                visited[minindex] = 0;                                          // позначення
вершини як пройденої
            }
        } while (minindex < INF);


        cout << endl << "\t_____________THE_SHORTEST_DISTANCE_TO_THE_VERTICES_____________" << endl
<< endl;
        for (int i = 0; i < SIZE; i++) {
            if (i == 0)
            {
                cout << "\t\tTo V0-vertice :  " << mindistance[i] << "\n";
                continue;
            }
            if (i == SIZE-1)
            {
                cout << "\t\tTo V*-vertice :  " << mindistance[i] << "\n";
                break;
            }
            cout << "\t\tTo V" << i + 1 << "-vertice :  " << mindistance[i] << "\n";
        }


        int vis[SIZE];                                                          // пройдені вершини
        int end = SIZE - 1;                                                     // індекс кінцевої вершини
        int prev = 1;
        int weight = mindistance[end];                                          // відстань до кінцевої
вершини
        vis[0] = end + 1;                                                       // початок - кінцева вершина

        while (end != begin_index)
        {
            for (int i = 0; i < SIZE; i++) {
                if (adjacency[end][i] != 0)                                     // якщо є сумісність -
перевірка на знаходження відстані, знайденої раніше
                {
                    int temp = weight - adjacency[end][i];
                    if (temp == mindistance[i])
                    {
                        weight = temp;
                        end = i;
                        vis[prev] = i + 1;                                      // збереження
відстані, через яку був перехід у найкоротшому шляху
                        prev++;
                    }
                }
            }
        }

        cout << endl << "\t_____________THE_PATH_OF_THE_SHORTEST_DISTANCE_____________" << endl
<< endl;
```

```
	cout << "    ";
	for (int i = prev - 1; i >= 0; i--) {

		cout << "V" << vis[i] << " => ";
	}
	cout << " The path is finished !";
	cout << endl << endl << endl;
	system("pause");
	return 0;
}
```

Результат програми :



Висновок : ми набуття практичних вмінь та навичок з використання алгоритму Дейкстри, програмно реалізували алгоритм