Міністерство освіти і науки України

Національний університет
“Львівська політехніка”

Кафедра систем штучного інтелекту

Лабораторна робота № 5

з дисципліни

«Дискретна математика»

Виконав:

студент групи КН-115
Поставка Маркіян

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.

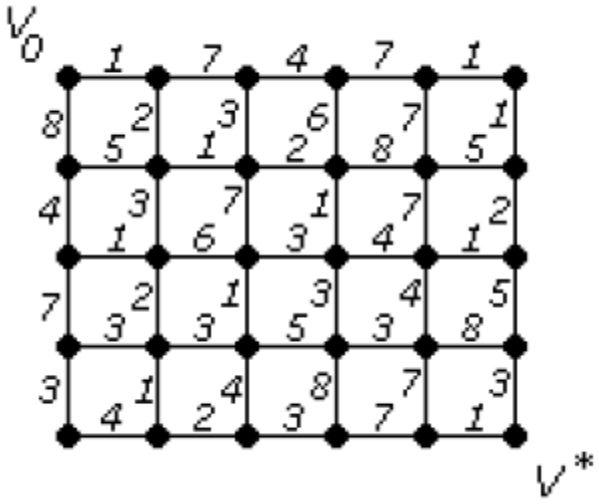
Додаток 1:

Варіант 11.

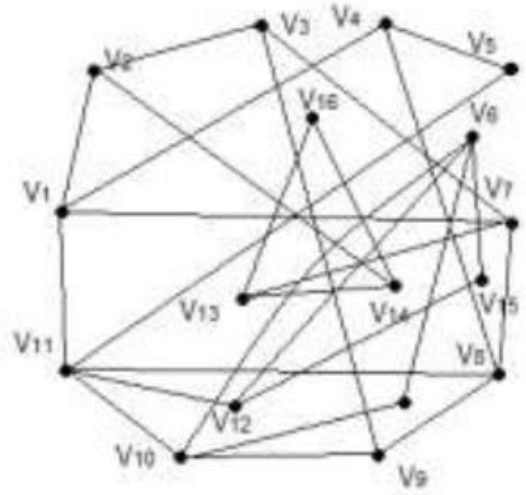
Постановка задачі:

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^*.

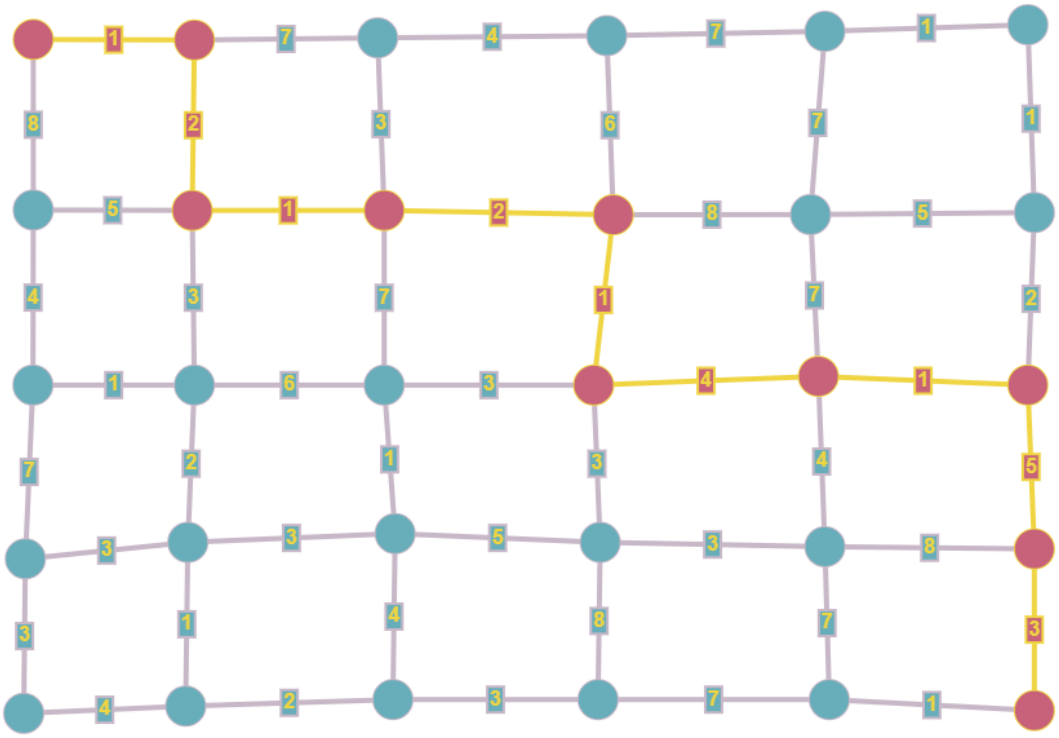


2. За допомогою γ-алгоритма зробити укладку графа у площині, або довести що вона неможлива.



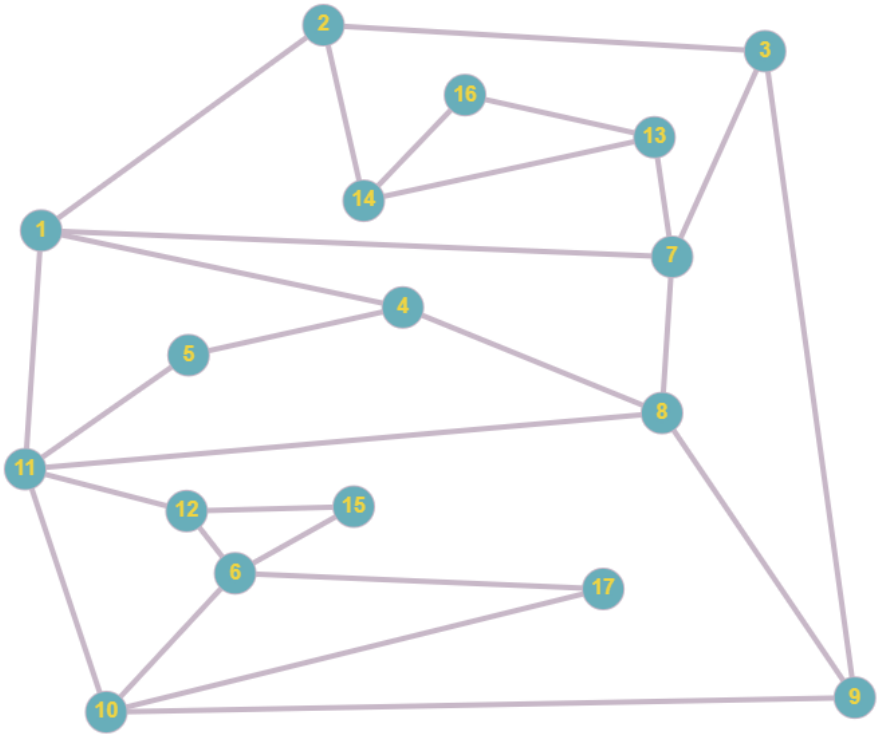
Рішення:

1)



Відстань між вершинами = 20.

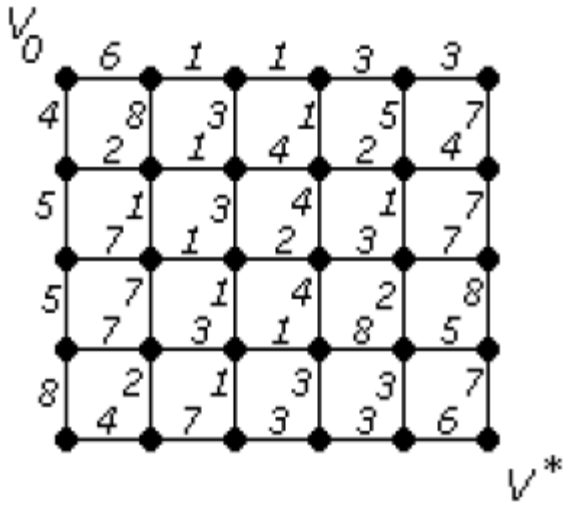
2)



Додаток 2:

Постановка задачі:

Завдання №2. Написати програму, яка реалізує алгоритм Дейкстри знаходження найкоротшого шляху між парою вершин у графі. Протестувати розроблену програму на графі згідно свого варіанту.



Код програми:

```
#include "stdafx.h"
#include <iostream>

using namespace std;

const int N = 30; // Num of nodes
const int inf = 1000; // infinity

void Dijkstra(int[N][N], int, int);

int main()
{
	int start_node;
	int end_node;

	// Initializing here beacuse to complicated to do this in function
	int gaph[N][N] =
	{ { 0, 6, 4, 0, 0, 0, 0, 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, 0, 8, 1, 0, 0, 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, 2, 0, 0, 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, 8, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
	{ 0, 1, 0, 0, 0, 1, 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, 0, 0, 0, 1, 0, 3, 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, 0, 0, 0, 3, 0, 0, 0, 5, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
	{ 0, 0, 0, 1, 3, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
	{ 0, 0, 0, 0, 0, 1, 0, 4, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
	{ 0, 0, 0, 0, 0, 0, 5, 0, 2, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
	{ 0, 0, 0, 0, 0, 0, 0, 3, 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, 0, 0, 0, 0, 4, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0 },
	{ 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 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, 0, 0, 5, 0, 8, 0, 0, 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, 8, 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, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 7, 0, 0, 0 },
	{ 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 1, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0 },
	{ 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 },
	{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0 },
	{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 7, 0, 0, 0, 2, 0, 0, 0, 0, 0 },
```

```
        { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0 },
        { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 2, 7, 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, 0, 0, 1, 0, 0, 0, 3, 0, 1, 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, 0, 0, 4, 0, 0, 0, 0, 1, 0, 8, 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, 0, 2, 0, 0, 0, 8, 0, 5, 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, 8, 0, 0, 0, 5, 0, 0, 0, 0, 7 },
        { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 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, 0, 0, 0, 0, 0, 3, 0, 0, 3, 0, 6 },
        { 0, 0, 0, 0, 0, 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, 6, 0 } };

        cout << "Enter start node: ";    cin >> start_node;
        cout << "Enter end node: ";              cin >> end_node;

        Dijkstra(gaph, start_node - 1, end_node-1);

        system("pause");
        return 0;
}

void Dijkstra(int g[N][N], int st, int ed)
{
        int weight[N]; // Array of nodes weights
        int k;
        int i;
        int u; // Active node
        bool ck[N]; // If visited

        for (i = 0; i < N; i++)
        {
                weight[i] = inf; ck[i] = false;
        }

        weight[st] = 0;
        for (k = 0; k < N - 1; k++)
        {
                int min = inf;
                for (i = 0; i < N; i++)
                        if (!ck[i] && weight[i] <= min) // If not checked and weight != inf
                        {
                                min = weight[i]; u = i;
                        }

                ck[u] = true;
                for (i = 0; i < N; i++)
                {
                        if (!ck[i] && (g[u][i]) && (weight[u] != inf) &&         // If !ck,
sumish(g[u][i]!=0), d[u]!=max -> pereN na shumishnist,
                                (weight[u] + g[u][i] < weight[i]))                                      //
If actiNe_w+cur_w<inf || actiNe_w+cur_w<preN_w //(if current_node weight is higher than base+current_side
weight we change current_node weight)
                        {
                                weight[i] = weight[u] + g[u][i];
                        }
                }

        }

        cout <<"\nPath from start to all weight\n";
        for (i = 0; i < N; i++) if (weight[i] != inf)
                cout << st + 1 << " -> " << i + 1 << " = " << weight[i] << endl;
        else cout << st + 1 << " > " << i + 1 << " = " << "No way from start to this node" << endl;

        cout <<"\nPath from start to entered node\n";
        if (weight[ed] != inf)
                cout << st + 1 << " -> " << ed + 1 << " = " << weight[ed] << endl;
        else cout << st + 1 << " -> " << ed + 1 << " = " << "No way from start to this node" << endl;
        cout << endl;
}
```

Результат роботи програми:

На вході програми програмно задано матрицю суміжності даного графа.

```
int gaph[N][N] =
{ { 0, 6, 4, 0, 0, 0, 0, 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, 0, 8, 1, 0, 0, 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, 2, 0, 0, 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, 8, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 1, 0, 0, 0, 1, 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, 0, 0, 0, 0, 1, 0, 3, 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, 0, 0, 3, 0, 0, 0, 5, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 1, 3, 0, 0, 0, 4, 0, 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, 1, 0, 4, 0, 2, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 5, 0, 2, 0, 0, 4, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 3, 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, 0, 0, 0, 0, 4, 7, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 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, 0, 0, 0, 5, 0, 8, 0, 0, 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, 8, 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, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0 },
{ 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 1, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 7, 0, 0, 0, 2, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 2, 7, 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, 0, 1, 0, 0, 0, 3, 0, 1, 0, 0, 1, 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, 1, 0, 8, 0, 0, 3, 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, 8, 0, 5, 0, 0, 3, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 5, 0, 0, 0, 0, 7 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 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, 0, 0, 0, 0, 0, 0, 0, 0, 3, 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, 0, 0, 0, 3, 0, 0, 3, 0, 6 },
{ 0, 0, 0, 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, 6, 0 } };
```

На виході отримуємо найкоротші маршрути з вказаної вершини до всіх вершин та конкретно з вказаної до вказаної вершини, за умовою завдання.

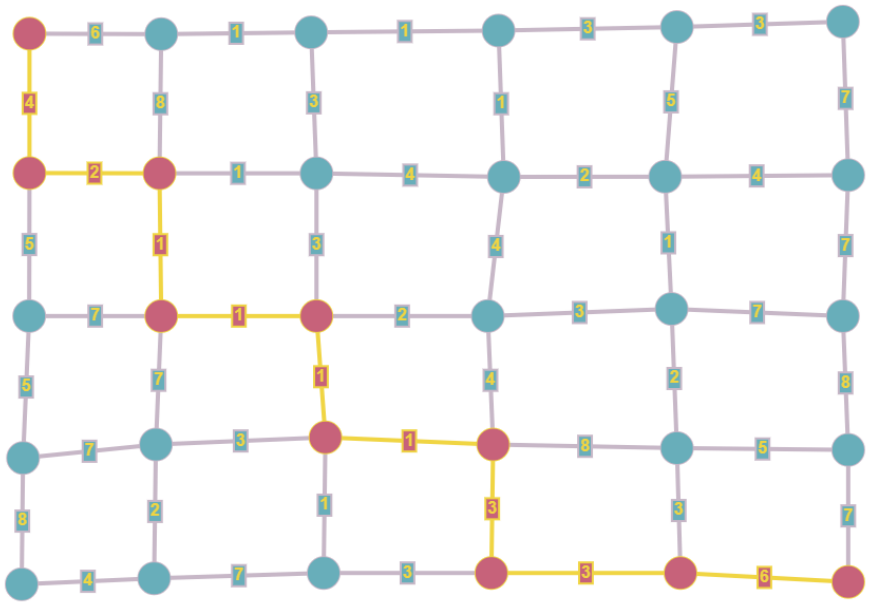
```
C:\Windows\system32\cmd.exe
Enter start node: 1
Enter end node: 30

Path from start to all weight
1 -> 1 = 0
1 -> 2 = 6
1 -> 3 = 4
1 -> 4 = 6
1 -> 5 = 7
1 -> 6 = 8
1 -> 7 = 11
1 -> 8 = 7
1 -> 9 = 9
1 -> 10 = 11
1 -> 11 = 14
1 -> 12 = 15
1 -> 13 = 9
1 -> 14 = 14
1 -> 15 = 18
1 -> 16 = 14
1 -> 17 = 7
1 -> 18 = 8
1 -> 19 = 10
1 -> 20 = 12
1 -> 21 = 19
1 -> 22 = 12
1 -> 23 = 9
1 -> 24 = 10
1 -> 25 = 14
1 -> 26 = 19
1 -> 27 = 10
1 -> 28 = 13
1 -> 29 = 16
1 -> 30 = 22

Path from start to entered node
1 -> 30 = 22

Для продолжения нажмите любую клавишу . . .
```

Візуальний вигляд найкоротшого маршруту для цього графа за умовою задачі.



Довжина такого маршруту = 22.

Висновок: Я набув практичних вмінь та навичок з використання алгоритму Дейкстри.