**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.

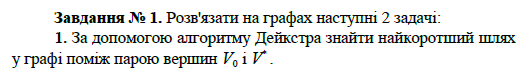
**Тема**: “Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи”

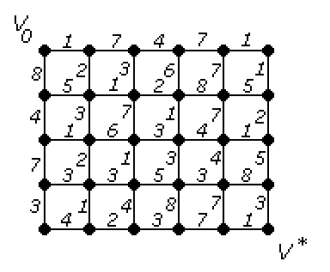
**Мета роботи:** набуття практичних вмінь та навичок з використання алгоритму Дeйкстри.

**Додаток 1:**

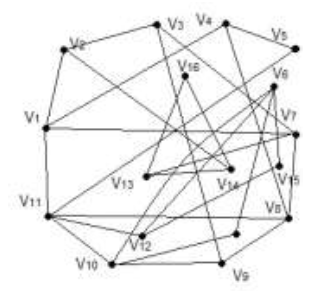
Варіант 11.

Постановка задачі:

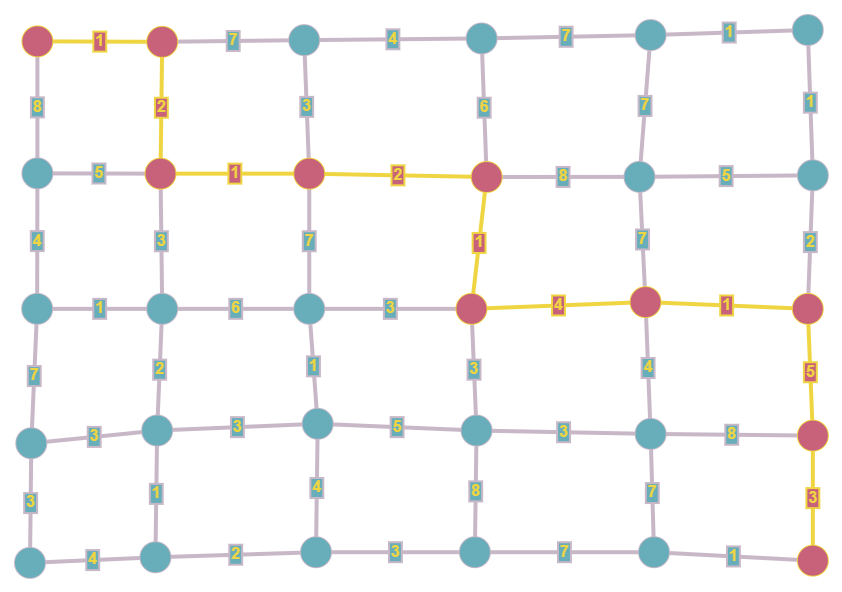






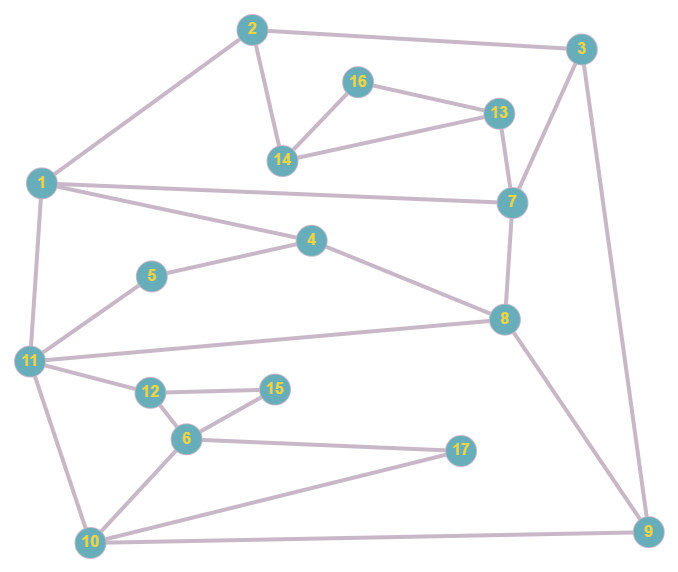


Рішення:

1. 

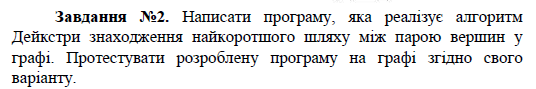
Відстань між вершинами = 20.

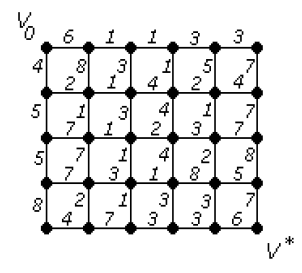
2)



**Додаток 2:**

Постановка задачі:





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, 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, 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, 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, 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, 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, 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, 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, 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;

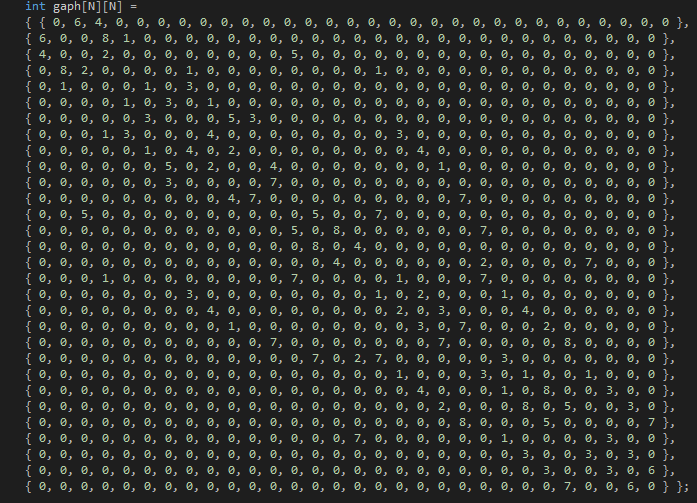
else cout << st + 1 << " -> " << ed + 1 << " = " << "No way from start to this node" << endl;

cout << endl;

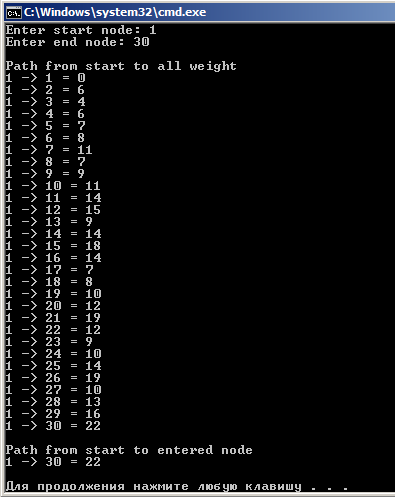
}

Результат роботи програми:

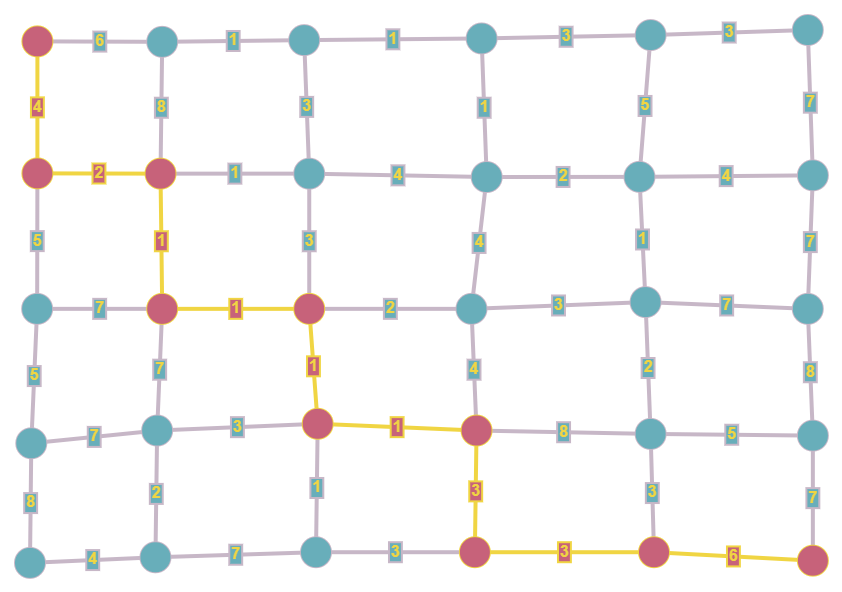
На вході програми програмно задано матрицю суміжності даного графа.



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



Візуальний вигляд найкоротшого маршруту для цього графа за умовою задачі.



Довжина такого маршруту = 22.

**Висновок:** Янабув практичних вмінь та навичок з використання алгоритму Дейкстри.