МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Лабораторна робота №5

з дисципліни

«Дискретна математика»

Виконала:

студентка групи КН-112

Максимець Віра

Перевірила:

Мельникова Н. І.

Львів-2019

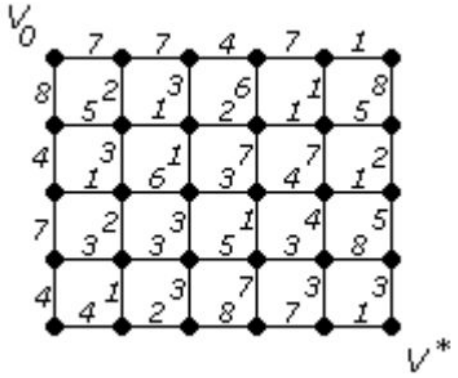
Тема: Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи

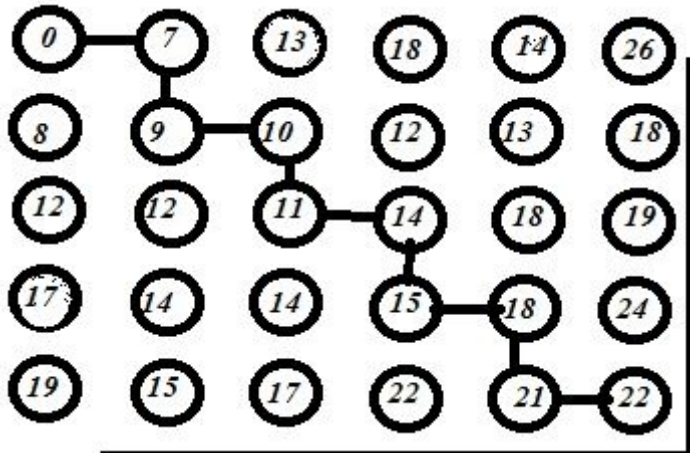
Мета роботи: набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

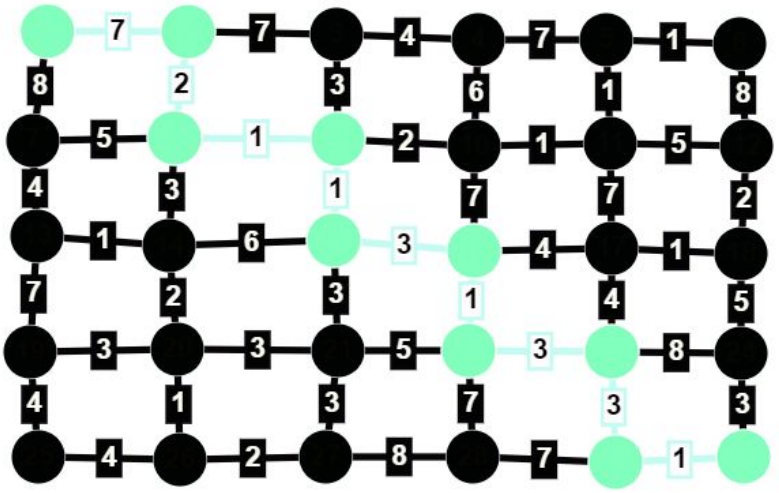
Варіант №10

Завдання №1:

1. За допомогою алгоритму Дейкстра знайти найкоротший шлях у графі поміж парою вершин V0 і V * . http://graphonline.ru/en/?graph=spCSbboHPQbUwNzDZZcst

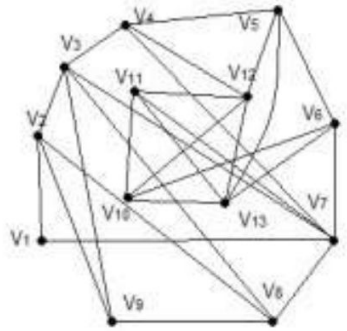




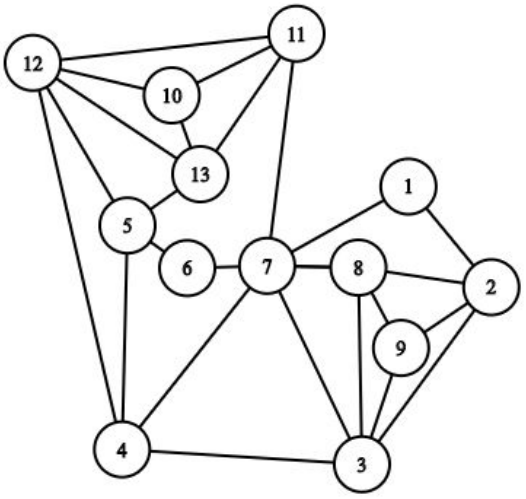


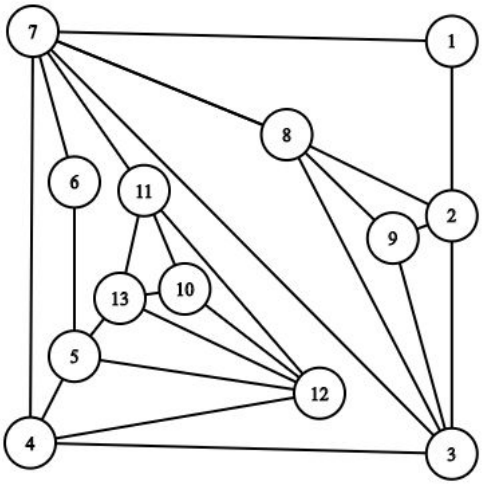
Найкоротша відстань: **22**

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

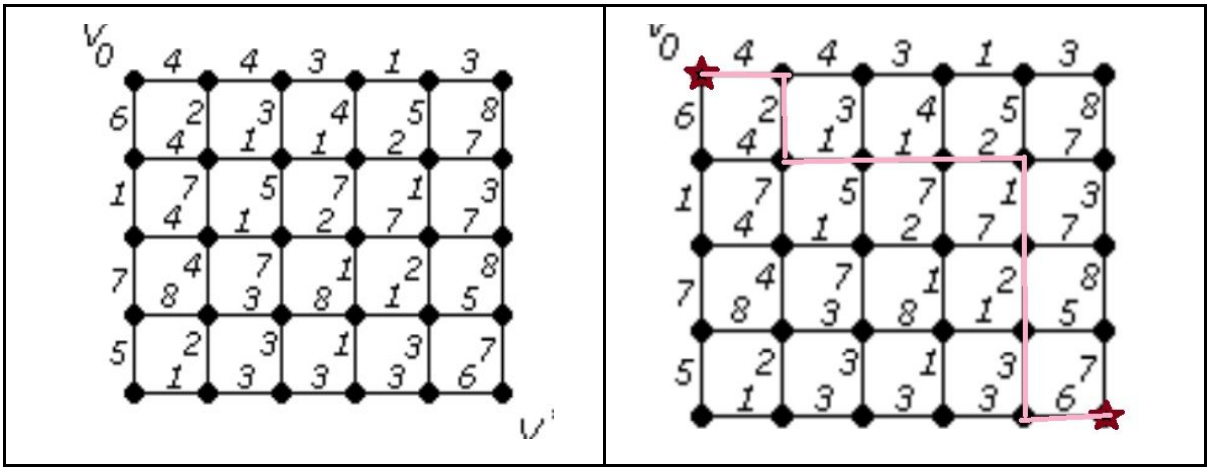


1. Виберемо деякий простий цикл С графа G і укладемо його на площині; покладемо G1= G **[1,2,3,4,5,6,7]**
2. Знайдемо грані графа G1 і сегменти відносно G1.
3. Для кожного сегмента S визначимо множину Г(S).
4. Якщо існує сегмент S, для якого мається єдина припустима грань Г, то розмістимо α- ланцюг L∈S у грань Г;





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



Функція Main:

```
#include <iostream>

using namespace std;

const int SIZE = 30;

int main()
{
int Matrix[][30] = {
        { 0,4,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*1*/
        { 4,0,4,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
        { 0,4,0,3,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*3*/
        { 0,0,3,0,1,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
        { 0,0,0,1,0,3,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*5*/
        { 0,0,0,0,3,0,0,0,0,0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
        { 6,0,0,0,0,0,0,4,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*7*/
        { 0,0,0,0,0,0,4,0,1,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
        { 0,0,3,0,0,0,0,1,0,1,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*9*/
        { 0,0,0,4,0,0,0,0,1,0,2,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
        { 0,0,0,0,5,0,0,0,0,2,0,7,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*11*/
        { 0,0,0,0,0,8,0,0,0,0,7,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0 },
        { 0,0,0,0,0,0,1,0,0,0,0,0,0,4,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0 }, /*13*/
        { 0,0,0,0,0,0,0,7,0,0,0,0,4,0,1,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0 },
        { 0,0,0,0,0,0,0,0,5,0,0,0,0,1,0,2,0,0,0,0,7,0,0,0,0,0,0,0,0,0 }, /*15*/
        { 0,0,0,0,0,0,0,0,0,7,0,0,0,0,2,0,7,0,0,0,0,1,0,0,0,0,0,0,0,0 },
        { 0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,7,0,7,0,0,0,0,2,0,0,0,0,0,0,0 }, /*17*/
        { 0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,7,0,0,0,0,0,0,8,0,0,0,0,0,0 },
        { 0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,8,0,0,0,0,5,0,0,0,0,0 }, /*19*/
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,8,0,3,0,0,0,0,2,0,0,0,0 },
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,3,0,8,0,0,0,0,2,0,0,0 }, /*21*/
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,8,0,1,0,0,0,0,1,0,0 },
```

```
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,1,0,5,0,0,0,0,3,0 }, /*23*/
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,5,0,0,0,0,0,0,7 },
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,0,1,0,0,0,0 }, /*25*/
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,1,0,3,0,0,0 },
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,3,0,3,0,0 }, /*27*/
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,3,0,3,0 },
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,3,0,6 }, /*29*/
 { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,6,0 }
        };

        cout <<  "The adjency matrix.\n" << endl;
        for (int i =0;i<SIZE; i++)
 {
    for (int j=0;j<SIZE; j++)
    {
        cout << Matrix[i][j] << " ";
    }
    cout << endl;
 }

int dis[SIZE];        // відстань
int visited[SIZE];    //відвідані вершини
int  minindex, min;
int startpoint = 0;
int split = SIZE/3;

for (int i = 0; i<SIZE; i++)
{
  dis[i] = 10000;     //відстань до інших вершин
  visited[i] = 1;     //позначити як невідвідано
}
dis[startpoint] = 0;

do {
  minindex = 10000;
  min = 10000;
  for (int i = 0; i<SIZE; i++)
  {
    if ((visited[i] == 1) && (dis[i]<min))
    {
      min = dis[i];
      minindex = i;
    }
  }

  if (minindex != 10000)
```

```
  {
    for (int i = 0; i<SIZE; i++)
    {
      if (Matrix[minindex][i] > 0)
      {
        int point = min + Matrix[minindex][i];   //додати знайдену мін вагу до існуючої ваги
вершини
        if (point < dis[i])                  // порівняти з потоною вагою
        {
          dis[i] = point;
        }
      }
    }
    visited[minindex] = 0;
  }
 } while (minindex < 10000);


 // Восстановление пути
 int endy;
 cout << "\nEnter the end-point: ";
 cin >> endy;
 int end = endy-1;

 int seen[SIZE]; // масив відвіданих вершин

 seen[0] = end + 1; // початковий элемент - кінцева вершина
 int weight = dis[end]; // вага кінцева вершини
 int k = 1; // індекс попередньої


 while (split!=0) // пока не початок
 {
  for (int i = 0; i<SIZE; i++)
  {
    if ((Matrix[end][i] != 0)&&(Matrix[end][i] != seen[k]))   // якщо вершини суміжні
    {
      int point = weight - Matrix[end][i];
      if (point == dis[i]) // якщо вага не співпадає
      {
           weight = point;
           end = i;       // зберігаємо попередню вершину
           seen[k] = i + 1; // і записуємо її в масив
           k++;
      }
    }
```

```
   }
   split--;
  }

  //Шлях
  cout<<"\nThe shortest lenght from start point to the end\n";
 cout << "1 > ";
  for (int i = k-1; i >= 0; i--)
  {
     cout << seen[i] << " > ";
  }
   cout << "\n Total weight is " << dis[SIZE-1] << endl;


  return 0;
}
```

Вивід:

```
The adjency matrix.

0 4 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 0 4 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 4 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 0 0
0 0 3 0 1 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 1 0 3 0 0 0 0 5 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 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 4 0 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 4 0 1 0 0 0 0 7 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 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 4 0 0 0 0 1 0 2 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 5 0 0 0 0 2 0 7 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 8 0 0 0 0 7 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 0 0 0 0 0 4 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 7 0 0 0 0 4 0 1 0 0 0 0 4 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 5 0 0 0 0 1 0 2 0 0 0 0 7 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 7 0 0 0 0 2 0 7 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 7 0 7 0 0 0 0 2 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 8 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 8 0 0 0 0 5 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 8 0 3 0 0 0 0 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 3 0 8 0 0 0 0 2 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 8 0 1 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 1 0 5 0 0 0 0 3
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 7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 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 2 0 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 3 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 3 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 3 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 7 0 0 0 0 6

Enter the end-point: 30

The shortest lenght from start point to the end
1 > 2 > 8 > 9 > 10 > 11 > 17 > 23 > 29 > 30 >
 Total weight is 22

Process returned 0 (0x0)   execution time : 2.602 s
Press any key to continue.
```