МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ
“ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота № 5
з дисципліни
«Дискретна математика»
з теми :

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

Виконав:
Студент групи КН-114
Сиротюк Владислав
Викладач:
Мельникова Н.І.

Львів – 2019р.

Лабораторна робота № 5.

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

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

ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ПРИКЛАДИ РОЗВ'ЯЗАННЯ ЗАДАЧ

Задача знаходження найкоротшого шляху з одним джерелом полягає у знаходженні найкоротших(мається на увазі найоптимальніших за вагою) шляхів від деякої вершини(джерела) до всіх вершин графа G. Для розв'язку цієї задачі використовується «жадібний» алгоритм, який називається алгоритмом Дейкстри.

«Жадібними» називаються алгоритми, які на кожному кроці вибирають оптимальний із можливих варіантів.

Задача про найкоротший ланцюг. Алгоритм Дейкстри.

Дано n-вершинний граф $G=(V, E)$, у якому виділено пару вершин $v_0, v^* \in V$, і кожне ребро зважене числом $w(e) \geq 0$. Нехай $X=\{x\}$ – множина усіх простих ланцюгів, що з'єднують v_0 з v^*, $x=(V_x, E_x)$. Цільова функція $F(x)=\sum_{e \in E_x} w(e) \to \min$. Потрібно знайти найкоротший ланцюг, тобто $x_0 \in X: \ F(x_0)=\min_{x \in X} F(x)$

Перед описом алгоритму Дейкстри подамо визначення термінів "k-а найближча вершина і "дерево найближчих вершин". Перше з цих понять визначається індуктивно так.

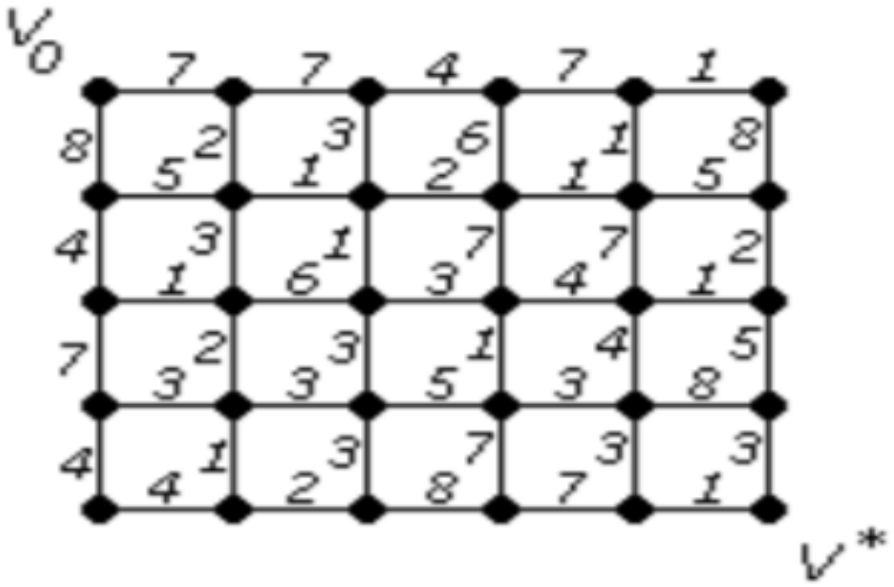
1-й крок індукції. Нехай зафіксовано вершину x_0, E_1 – множина усіх ребер $e \in E$, інцидентних v_0. Серед ребер $e \in E_1$ вибираємо ребро $e(1)=(v_0, v_1)$, що має мінімальну вагу, тобто $w(e(1))=\min_{e \in E_1} w(e)$. Тоді v_1 називаємо першою найближчою вершиною (НВ), число $w(e(1))$ позначаємо $l(1) = l(v_1)$ і називаємо відстанню до цієї НВ. Позначимо $V_1=\{v_0, v_1\}$ – множину найближчих вершин.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ

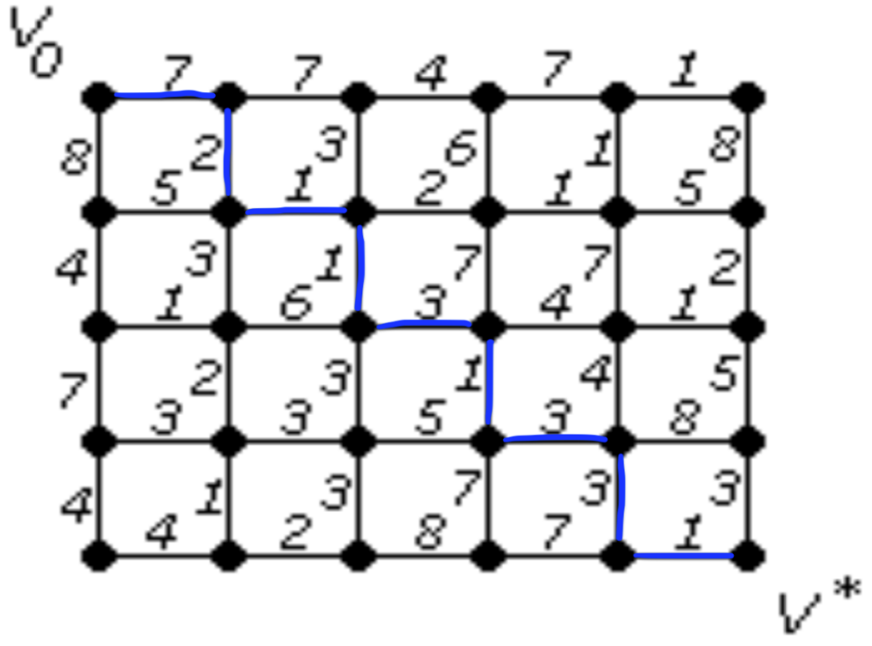
Завдання № 1. Розв'язати на графах наступні 2 задачі:

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

10

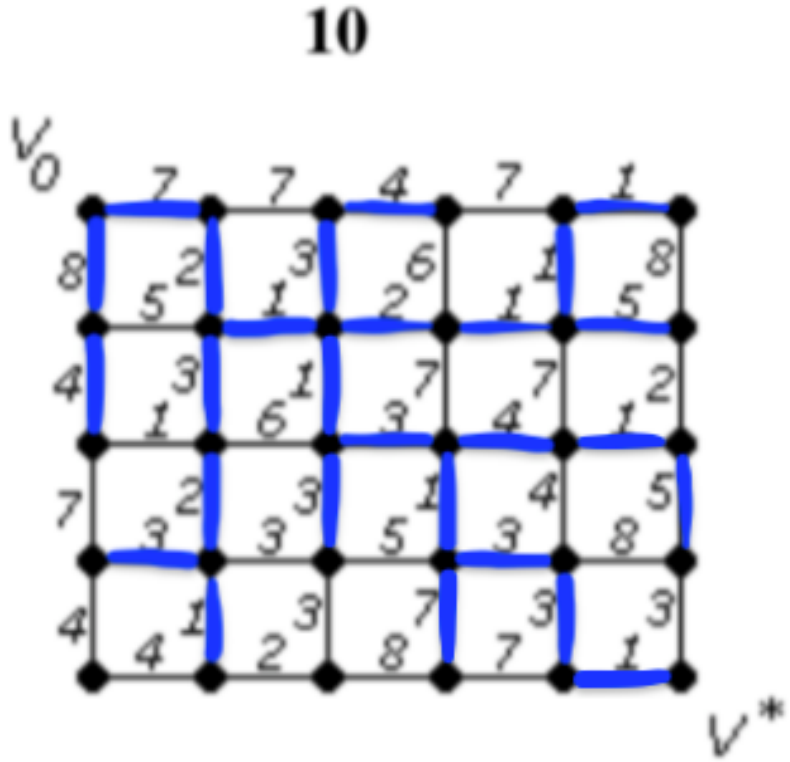


10



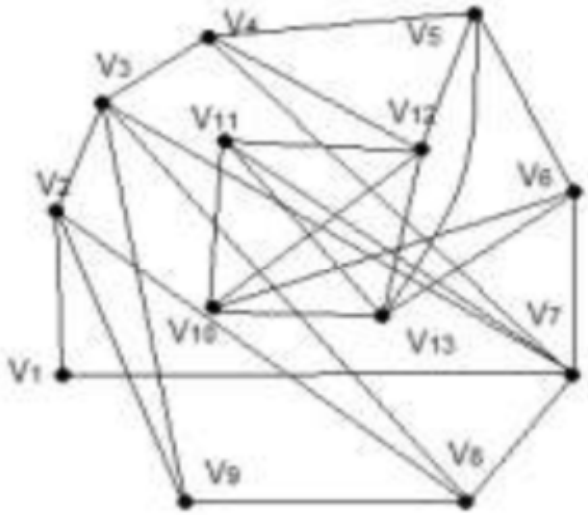
Найкоротший шлях – 1,2,8,9,15,16,22,23,29,30.
Вага шляху – 0+7+2+1+1+3+1+3+3+1 = 22.

Всі шляхи,які використовував при рішенні:



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

10



E – 27
V - 14

Для того щоб робити укладу графа потрібно перевірити чи вона взагалі можлива . Для цього існує формула : V-E+F=2, де V-кількість вершин, E- кількість ребер, F- кількість граней, що не містять всередині себе інших елементів графа.
З того, що кожна грань обмежена не більше ніж трьома ребрами, випливає, що для плоского графа:
E≤3V-6
тобто, при більшому числі ребер граф непланарний. Звідси випливає, що в планарному графі завжди можна знайти вершину степеня не більше 5.

Підставляємо значення :
27≤3*14-6.
Виходить, що 27≤36.
Отже, не можливо зробити укладку нашого графа в площині , він непланарний.

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

```
#define _CRT_SECURE_NO_WARNINGS
#include<iostream>


using namespace std;
int main()
{
  int vertex,edges;
  cout << "Count of vertex: ";
  cin >> vertex;
  cout << "Count of edges: ";
  cin >> edges;
  int **a=new int*[vertex];
  int *d=new int[vertex];
  int *visited=new int[vertex];
  int temp, minindex, min;


  int start, finish;
  cout << "From vertex :";
  cin >> start;
  start--;
  cout << "To: ";
  cin >> finish;
  finish--;

  int begin_index = start;



  for (int i = 0; i < vertex; i++)
  {
    a[i] = new int [vertex];
  }


  for (int i = 0; i < vertex; i++)
  {
    for (int j = 0; j < vertex; j++) {
      a[i][j] = 0;
    }
  }


  for (int i = 0; i < edges; i++)
  {
    int x, y,weight;
    cout << "Edge[" << i << "]=" << endl;
```

```
  cout << "Vertex1: ";
  cin >> x;
  cout << "Vertex2: ";
  cin >> y;
  cout << "Weight: ";
  cin >> weight;
  a[--x][--y] =weight;
  a[y][x] = weight;

}



for (int i = 0; i < vertex; i++)
{
  for (int j = 0; j < vertex; j++)
    cout<< a[i][j]<<" ";
  cout << endl;
}


for (int i = 0; i < vertex; i++)
{
  d[i] = 10000;
  visited[i] = 1;
}

d[begin_index] = 0;

do {
  minindex = 10000;
  min = 10000;
  for (int i = 0; i < vertex; i++)
  {
    if ((visited[i] == 1) && (d[i] < min))
    {
      min = d[i];
      minindex = i;
    }
  }
  if (minindex != 10000)
  {
    for (int i = 0; i < vertex; i++)
    {
      if (a[minindex][i] > 0)
      {
        temp = min + a[minindex][i];
        if (temp < d[i])
        {
          d[i] = temp;
        }
      }
    }
    visited[minindex] = 0;
  }
} while (minindex < 10000);
```

```
  cout<<"Minimal ways to vertex: "<<endl;
  for (int i = 0; i < vertex; i++)cout << d[i] << " ";

  bool flag = false;
  for (int i = 0; i < vertex; i++)if(d[i]!=0&&d[i]!=10000)flag = true;

  if (flag) {

    int* ver = new int[vertex];
    int end = finish;
    ver[0] = end + 1;
    int k = 1;
    int weight = d[end];

    while (end != begin_index)
    {
      for (int i = 0; i < vertex; i++)
        if (a[end][i] != 0)
        {
          int temp = weight - a[end][i];
          if (temp == d[i])
          {
            weight = temp;
            end = i;
            ver[k] = i + 1;
            k++;
          }
        }
    }


    cout << endl << "Print minimal way" << endl;
    for (int i = k - 1; i >= 0; i--)cout << ver[i] << " ";


  }
  else {
    cout <<endl<< "There isn`t such way";
  }
  return 0;
}
```

Висновок: на цій лабораторній роботі я набув практичних вмінь та навичок з використання алгоритму Дейкстри.