МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №5

з дисципліни

«Дискретна математика»

Виконала:

студентка групи КН-114

Олескевич Софія

Викладач:

Мельникова Н.І.

Львів – 2019р.

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

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

Теоретичні відомості:

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.

Алгоритм Дейкстри.

Дано n-вершинний граф $G = (V, E)$, у якому виділено пару вершин v $v \in V$ *

0 , , і кожне ребро зважене числом $w(e) \geq 0$. Нехай

$X = \{x\}$ – множина усіх простих ланцюгів, що з'єднують 0 v з * v ,

( ) x x E V x , = . Цільова функція min) ( ) ($\to =\sum$

$\in e$ Ex

F x w e . Потрібно

знайти найкоротший ланцюг, тобто : 0 $x \in X$ ( ) min ( ) 0 F x F x

$x \in X$

=

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 x0, E1 – множина усіх ребер e ∈E, інцидентних v0. Серед ребер e ∈E1 вибираємо ребро e(1) =(v0, v1), що має мінімальну вагу, тобто ((1)) min ( )

1

w e w e

$e \in E$

= . Тоді

v1 називаємо першою найближчою вершиною (НВ), число w(e(1)) позначаємо l(1) = l(v1) і називаємо відстанню до цієї НВ. Позначимо V1={v0, v1} – множину найближчих вершин.

2-й крок індукції. Позначимо E2 – множину усіх ребер e=(v',v''), e ∈E, таких що v'∈V1, v''∈(V \V1). Найближчим вершинам v∈V1 приписано відстані l(v) до кореня v0 , причому l(v0)=0. Введемо позначення: 1 V – множина таких вершин v''∈(V \V1), що ∃ ребра виду e =(v, v''), де v∈V1. Для всіх ребер e∈E2 знаходимо таке ребро e2=(v', v2), що величина l(v')+w(e2) найменша. Тоді v2 називається другою

найближчою вершиною, а ребра e1, e2 утворюють зростаюче дерево для виділених найближчих вершин D2 ={e1, e2}.
(s+1)-й крок індукції. Нехай у результаті s кроків виділено множину найближчих вершин Vs={v0, v1, …, vs} і відповідне їй зростаюче дерево Ds={e1, e2, …, es}... Для кожної вершини v∈Vs обчислена відстань l(v) від кореня v0 до v; *s V* – множина вершин v∈(V\Vs), для яких існують ребра вигляду e =(vr, v), де vr ∈Vs , v∈(V\Vs). На кроці s+1 для кожної вершини vr∈Vs обчислюємо відстань до вершини vr : (1)( ) ( ) min (,) *
*
L s v l v w v v r
v V
r r
$\in$ *s*
$+ = +$, де min
береться по всіх ребрах e=(vr, v∗), *v* $\in$*V s* *
, після чого знаходимо min
серед величин L(s+1)(vr). Нехай цей min досягнуто для вершин *vr*0і відповідної їй *v* $\in$*V s* * , що назвемо vs+1. Тоді вершину vs+1 називаємо (s+1)-ю НВ, одержуємо множину Vs+1 =Vs Υ *vs*+1 і зростаюче дерево Ds+1 =Ds Υ (*vr*0,*vs*+1). (s+1)-й крок завершується перевіркою: чи є чергова НВ vs+1 відзначеною вершиною, що повинна бути за умовою задачі зв'язано найкоротшим ланцюгом з вершиною v0 . Якщо так, то довжина шуканого ланцюга дорівнює l(vs+1)=l(*vr*0)+w(*vr*0, vs+1); при цьому шуканий ланцюг однозначно відновлюється з ребер зростаючого дерева Ds+1. У противному випадку випливає перехід до кроку s+2.

ІНДИВІДУАЛЬНИЙ ВАРІАНТ №2

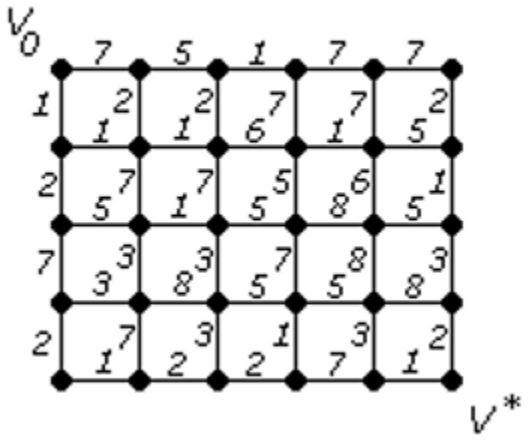
Завдання № 1.

Розв'язати на графах наступні 2 задачі:

1. За допомогою алгоритму Дейкстра знайти найкоротший шлях у графі поміж парою вершин V0 і V* .
2. За допомогою у-алгоритма зробити укладку графа у площині, або довести що вона неможлива.

1

2



Знаходжу найменшу відстань до кожної вершини:

L(v1)=1

L(v2)=2

L(v3)=3

L(v4)=3

L(v5)=4

L(v6)=5

L(v7)=6

L(v8)=8

L(v9)=9

L(v10)=9

L(v11)=10

L(v12)=10

L(v13)=11

L(v14)=12

L(v15)=12

L(v16)=13

L(v17)=13

L(v18)=14

L(v19)=15

L(v20)=15

L(v21)=16

L(v22)=16

L(v23)=17

L(v24)=17

L(v25)=17

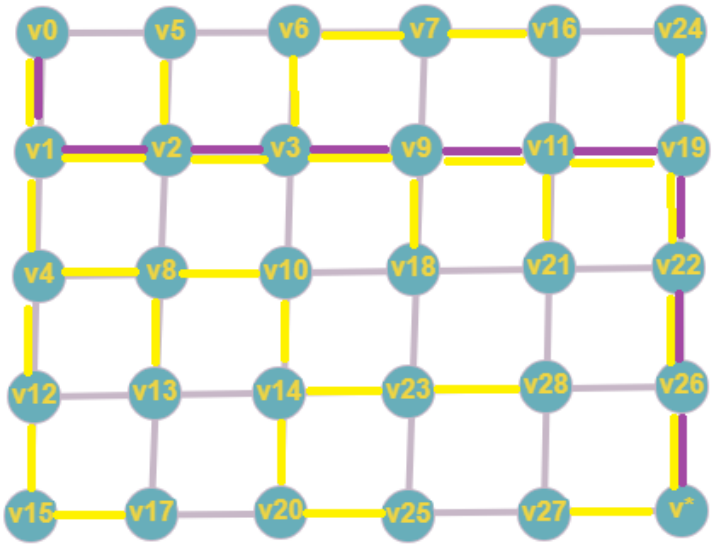
L(v26)=19

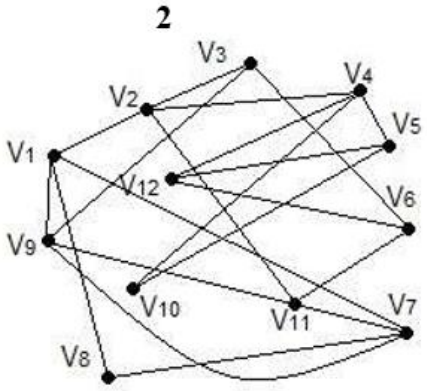
L(v27)=22

L(v28)=22

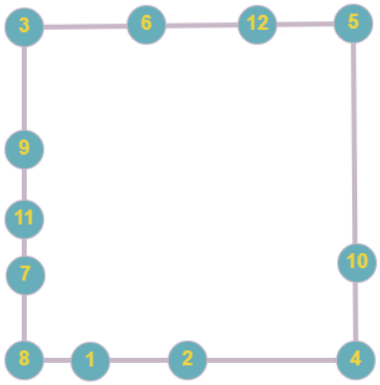
L(v*)=21

Відповідь: найкоротша відстань від V0 до V* дорівнює 21 і проходить через вершини { v0 v5 v6 v7 v16 v24 v19 v22 v26 v*}

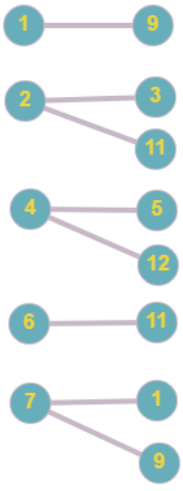




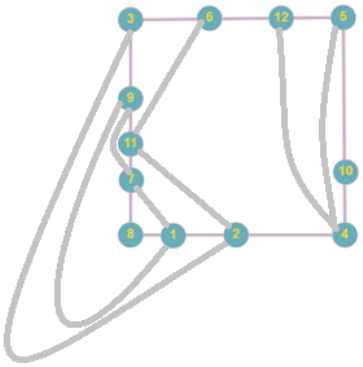
Виділяю цикл v1, v12, v4, v2, v3, v9, v11, v6, v5,v8,v7.



Записую інші вершини та їх ребра, які не входять в цикл.



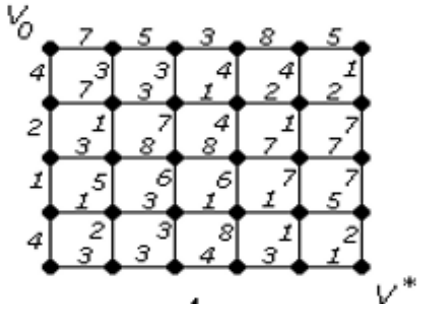
Укладка графа виглядає так:



Завдання №2.

Написати програму, яка реалізує алгоритм Дейкстри знаходження найкоротшого шляху між парою вершин у графі.

Протестувати розроблену програму на графі згідно свого варіанту.



Скріни коду програми:

```cpp
#include <iostream>
using namespace std;

int n;
int g[50][50];

bool visited[50];
int dist[50];
int pred[50];

void graf(int g[50][50])
{
    //int g[50][50];
    cout<<"Number of tops?";
    cin>>n;
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            g[i][j]=0;
        }
    }

int t1, t2;
cout<<"Number of rows and columns?"; //6 5
cin>>t1>>t2;

for (int i=0; i<n; i++){
    for(int j=i+1; j<n; j++){
      if(j==i+1 || j==i+t1){
        cout<<"top"<<i+1<<" to top"<<j+1<<" :";
        cin>>g[i][j];
      }else { g[i][j]=0; }
    }
 }
}

int distance()
{
```

```
36  int distance()
37  {
38
39    int minimum = 10000, minDist;
40    for (int z = 0; z < n; z++)
41      if (visited[z] == false && dist[z] <= minimum)
42      {
43       minimum = dist[z];
44       minDist = z;
45      }
46    return minDist;
47  }
48
49
50  void printPath(int j)
51  {
52    if (pred[j] == -1)
53      return;
54    printPath(pred[j]);
55    cout << "Top" << j+1 << " -> ";
56  }
57
58  void dijkstra(int g[50][50])
59  {
60      //int g[50][50];
61      int src;
62      cout << "Enter the Source Node : ";
63      cin >> src;
64      for (int i = 0; i < n; i++)
65      {
66        pred[0] = -1;
67        dist[i] = 10000;
68        visited[i] = false;
69      }
70      dist[src-1] = 0;
71      for (int count = 0; count < n - 1; count++)
72      {
```

```
68        visited[i] = false;
69      }
70      dist[src-1] = 0;
71      for (int count = 0; count < n - 1; count++)
72      {
73        int u = distance();
74        int q;
75        visited[u] = true;
76        for (int z = 0; z < n; z++)
77          if (!visited[z] && g[u][z] && dist[u] + g[u][z] < dist[z])
78          {
79            pred[z] = u;
80            q++;
81            dist[z] = dist[u] + g[u][z];
82          }
83      }
84      cout << "The least way is: ";
85      cout << dist[29] << endl;
86      cout << "The way is: ";
87      cout << "Top -> ";
88        printPath(29);
89        cout << "finish" << endl;
90
91    }
92
93
94
95  int main()
96  {
97
98  int g[50][50];
99  graf(g);
100 dijkstra(g);
101 return 0;
102
103 }
104
```

```
top16 to top17 :7
top16 to top22 :6
top17 to top18 :7
top17 to top23 :7
top18 to top19 :0
top18 to top24 :7
top19 to top20 :1
top19 to top25 :4
top20 to top21 :3
top20 to top26 :2
top21 to top22 :1
top21 to top27 :3
top22 to top23 :1
top22 to top28 :8
top23 to top24 :5
top23 to top29 :1
top24 to top25 :0
top24 to top30 :2
top25 to top26 :3
top26 to top27 :3
top27 to top28 :4
top28 to top29 :3
top29 to top30 :1
Enter the Source Node : 1
The least way is: 15
The way is: Top -> Top7 -> Top13 -> Top19 -> Top20 -> Top21 -> Top22 -> Top23 -> Top29 -> Top30 -> finish

Process returned 0 (0x0)   execution time : 160.406 s
Press any key to continue.
```

Висновок:

На цій лабораторній роботі я освоїла алгоритм Дейкстри та зуміла зробити гама-укладку графа .