Міністерство освіти і науки України

Національному університеті "Львівська Політехніка"

Кафедра систем штучного інтелекту

Лабораторна робота № 5

з дисципліни

≪ Дискретна математика ≫

Виконав:

студент групи КН-114

Павлик Богдан

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

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 $v \in V$ *

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

$w\ e\ w\ e$

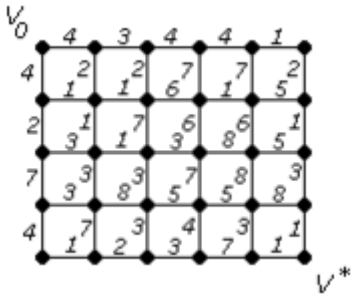
$e \in E$

= . Тоді

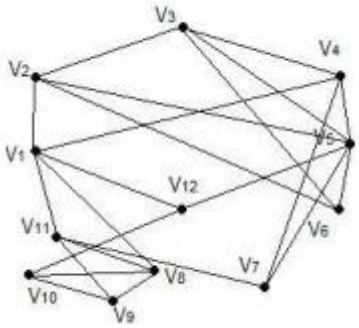
v1 називаємо першою найближчою вершиною (НВ), число w(e(1)) позначаємо l(1) = l(v1) і називаємо відстанню до цієї НВ. Позначимо V1={v0, v1} – множину найближчих вершин.
2-й крок індукції. Позначимо E2 – множину усіх ребер e=(*v*′,*v*′′), e ∈E, таких що *v*′∈V1, *v*′′∈(V \V1). Найближчим вершинам v∈V1 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 $\cup$ *vs*+1 і зростаюче дерево Ds+1 =Ds $\cup$ (*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.

Завдання варіанту №3

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.



Розв'язання

1.

l(v1) = 4

l(v2) = 4

l(v3) = 5

l(v4) = 6

l(v5) = 6

l(v6) = 6

l(v7) = 7

l(v8) = 7

l(v9) = 9

l(v10) = 10

l(v11) = 10

l(v12) = 11

l(v13) = 12

l(v14) = 12

l(v15) = 13

l(v16) = 13

l(v17) = 15

l(v18) = 15

l(v19) = 15

l(v20) = 16

l(v21) = 16

l(v22) = 16

l(v23) = 18

l(v24) = 18

l(v25) = 19

l(v26) = 20

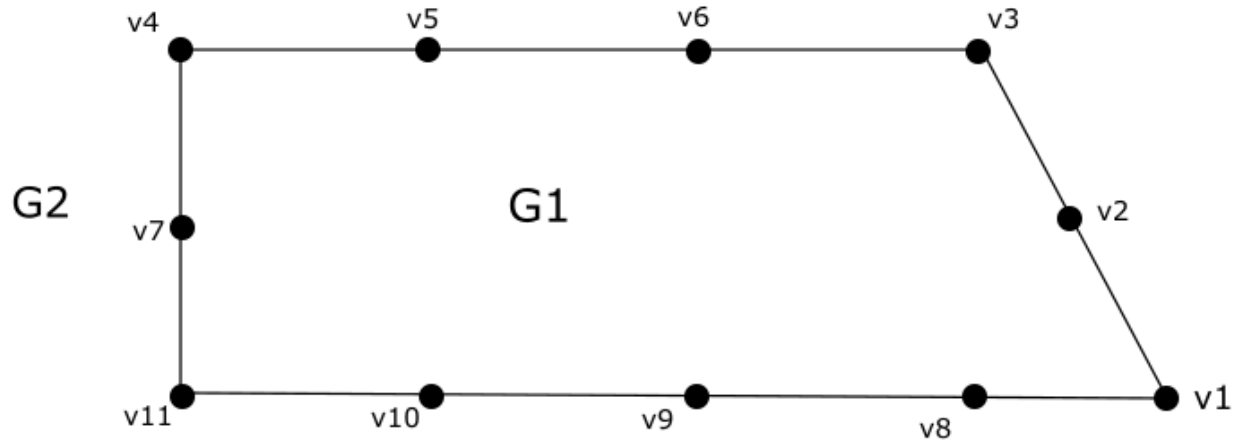
l(v27) = 22

l(v28) = 23

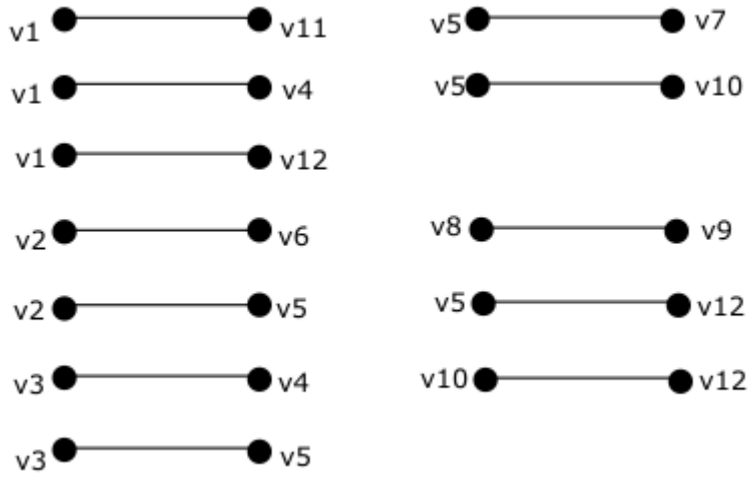
l(v*) = 23

Найкоротша відстань від вершини v0 до v* проходить через вершини { v2, v8, v12, v17, v20, v23, v25, v27} та ця відстань дорівнює 23.

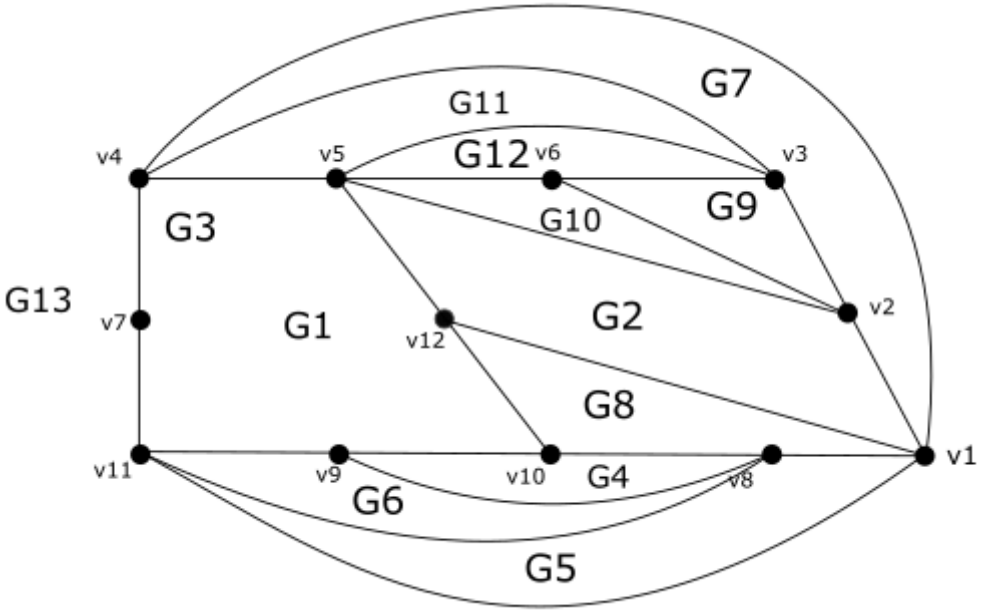
2. Виділяємо цикл v1 v2 v3 v6 v5 v4 v7 v11 v10 v9 v8



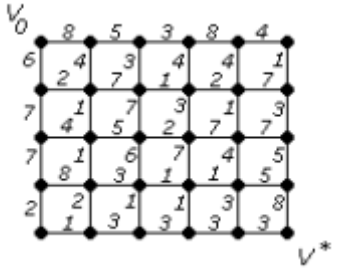
Ось ребра, які не входять до циклу:



Отже, укладений граф 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.



```cpp
#include<iostream>
using namespace std;
int n;
int i, j, q;
int dist[40];
bool visited[40];
int pred[40];
void createGraph(int c[40][40])
{
    int g1, g2;
    cout << "Enter the number of vertices: ";
    cin >> n;
    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++)
        {
            c[i][j] = 0;
        }
    }
    cout << "Enter size of (n*m) : ";
    cin >> g1 >> g2;

    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++)
        {
            if (j == i + 1 || j == i + g1) {
                cout << "Enter the length from x" << i+1 << " to x" << j+1 << ": ";
                cin >> c[i][j];
            }
            else {
                c[i][j] = 0;
            }
        }
    }
}
int minDistance()
{
    int minimum = 10000, minDist;
    for (int v = 0; v < n; v++)
        if (visited[v] == false && dist[v] <= minimum)
        {
            minimum = dist[v];
            minDist = v;
        }
    return minDist;
}
void printPath(int j)
{
    if (pred[j] == -1)
        return;
    printPath(pred[j]);
    cout << "X" << j+1 << " -> ";
}
void dijkstra(int c[40][40])
{
        int start;
        cout << "Enter the first node : ";
        cin >> start;
        for (int i = 0; i < n; i++)
        {
            pred[0] = -1;
            dist[i] = 10000;
```

```
63              visited[i] = false;
64          }
65          dist[start-1] = 0;
66          for (int count = 0; count < n - 1; count++)
67          {
68              int u = minDistance();
69              visited[u] = true;
70              for (int v = 0; v < n; v++)
71                  if (!visited[v] && c[u][v] &&
72                      dist[u] + c[u][v] < dist[v])
73                  {
74                      pred[v] = u;
75                      dist[v] = dist[u] + c[u][v];
76                  }
77          }
78          cout << "The least way is: ";
79          cout << dist[29] << endl;
80          cout << "The way is: ";
81          cout << "X1 -> ";
82              printPath(29);
83              cout << "The end!)" << endl;
84          //}
85      }
86  int main()
87  {
88      int c[40][40];
89      createGraph(c);
90      dijkstra(c);
91      return 0;
92  }
```

```
Enter the number of vertices: 30
Enter size of (n*m) : 6 5
Enter the length from x1 to x2: 8
Enter the length from x1 to x7: 6
Enter the length from x2 to x3: 5
Enter the length from x2 to x8: 4
Enter the length from x3 to x4: 3
Enter the length from x3 to x9: 3
Enter the length from x4 to x5: 8
Enter the length from x4 to x10: 4
Enter the length from x5 to x6: 4
Enter the length from x5 to x11: 4
Enter the length from x6 to x7: 0
Enter the length from x6 to x12: 1
Enter the length from x7 to x8: 2
Enter the length from x7 to x13: 7
Enter the length from x8 to x9: 7
Enter the length from x8 to x14: 1
Enter the length from x9 to x10: 1
Enter the length from x9 to x15: 7
Enter the length from x10 to x11: 2
Enter the length from x10 to x16: 3
Enter the length from x11 to x12: 7
Enter the length from x11 to x17: 1
Enter the length from x12 to x13: 0
Enter the length from x12 to x18: 3
Enter the length from x13 to x14: 4
Enter the length from x13 to x19: 7
Enter the length from x14 to x15: 5
Enter the length from x14 to x20: 1
Enter the length from x15 to x16: 2
Enter the length from x15 to x21: 6
Enter the length from x16 to x17: 7
Enter the length from x16 to x22: 7
Enter the length from x17 to x18: 7
Enter the length from x17 to x23: 4
Enter the length from x18 to x19: 0
Enter the length from x18 to x24: 5
Enter the length from x19 to x20: 8
Enter the length from x19 to x25: 2
Enter the length from x20 to x21: 3
Enter the length from x20 to x26: 2
Enter the length from x21 to x22: 1
Enter the length from x21 to x27: 1
Enter the length from x22 to x23: 1
Enter the length from x22 to x28: 1
Enter the length from x23 to x24: 5
Enter the length from x23 to x29: 3
Enter the length from x24 to x25: 0
Enter the length from x24 to x30: 8
Enter the length from x25 to x26: 1
Enter the length from x26 to x27: 3
Enter the length from x27 to x28: 3
Enter the length from x28 to x29: 3
Enter the length from x29 to x30: 3
Enter the first node : 1
The least way is: 21
The way is: X1 -> X7 -> X8 -> X14 -> X20 -> X21 -> X22 -> X28 -> X29 -> X30 -> T
he end!)
```