МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №5

з дисципліни «Дискретна математика»

Виконала:

студент групи КН-114

Серкіз Людмила

Викладач:

Мельникова Н.І.

Львів – 2019 р.

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.

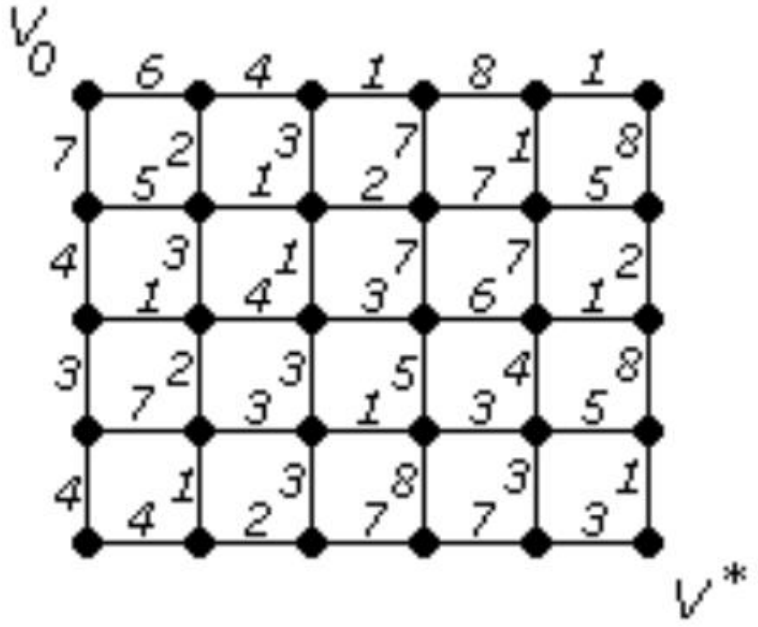
2-й крок індукції. Позначимо E_2 – множину усіх ребер $e=(v', v'')$, $e \in E$, таких що $v' \in V_1$, $v'' \in (V \setminus V_1)$. Найближчим вершинам $v \in V_1$ приписано відстані $l(v)$ до кореня v_0, причому $l(v_0)=0$. Введемо позначення: $\overline{V_1}$ – множина таких вершин $v'' \in (V \setminus V_1)$, що $\exists$ ребра виду $e=(v, v'')$, де $v \in V_1$. Для всіх ребер $e \in E_2$ знаходимо таке ребро $e_2=(v', v_2)$, що величина $l(v')+w(e_2)$ найменша. Тоді v_2 називається другою найближчою вершиною, а ребра e_1, e_2 утворюють зростаюче дерево для виділених найближчих вершин $D_2=\{e_1, e_2\}$.

(s+1)-й крок індукції. Нехай у результаті s кроків виділено множину найближчих вершин $Vs=\{v_0, v_1, \ldots, v_s\}$ і відповідне їй зростаюче дерево $D_s=\{e_1, e_2, \ldots, e_s\}$... Для кожної вершини $v \in V_s$ обчислена відстань $l(v)$ від кореня v_0 до v; $\overline{V_s}$ – множина вершин $v \in (V \setminus V_s)$, для яких існують ребра вигляду $e=(v_r, v)$, де $v_r \in V_s$, $v \in (V \setminus V_s)$. На кроці s+1 для кожної вершини $v_r \in V_s$ обчислюємо відстань до вершини v_r : $L(s+1)(v_r) = l(v_r) + \min_{v^* \in V_s} w(v_r, v^*)$, де min береться по всіх ребрах $e=(v_r, v*)$, $v^* \in \overline{V}_s$, після чого знаходимо min серед величин $L(s+1)(v_r)$. Нехай цей min досягнуто для вершин v_{r0} і відповідної їй $v^* \in \overline{V}_s$, що назвемо v_{s+1}. Тоді вершину v_{s+1} називаємо (s+1)-ю НВ, одержуємо множину $V_{s+1}=V_s \cup v_{s+1}$ і зростаюче дерево $D_{s+1}=D_s \cup (v_{r0}, v_{s+1})$. (s+1)-й крок завершується перевіркою: чи є чергова НВ v_{s+1} відзначеною вершиною, що повинна бути за умовою задачі зв'язано найкоротшим ланцюгом з вершиною v_0. Якщо так, то довжина шуканого ланцюга дорівнює $l(v_{s+1})=l(v_{r0})+w(v_{r0}, v_{s+1})$; при цьому шуканий ланцюг однозначно відновлюється з ребер зростаючого дерева D_{s+1}. У противному випадку випливає перехід до кроку s+2.

Алгоритм γ укладання графа G являє собою процес послідовного приєднання до деякого укладеного підграфа $\tilde{G}$ графа G нового ланцюга, обидва кінці якого належать $\tilde{G}$. При цьому в якості початкового плоского графа $\tilde{G}$ вибирається будь-який простий цикл графа G. Процес продовжується доти, поки не буде побудовано плоский граф, ізоморфний графові G, або приєднання деякого ланцюга виявиться неможливим. В останньому випадку граф G не є планарним.

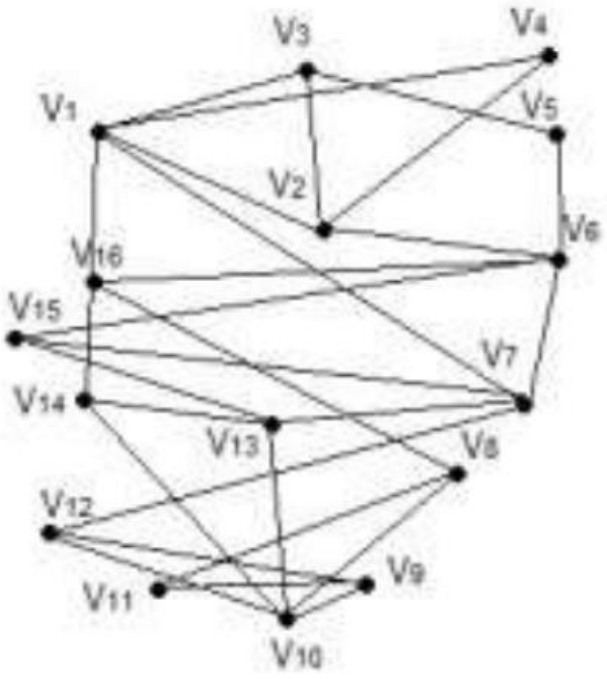
Завдання № 1. Розв'язати на графах наступні 2 задачі:

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

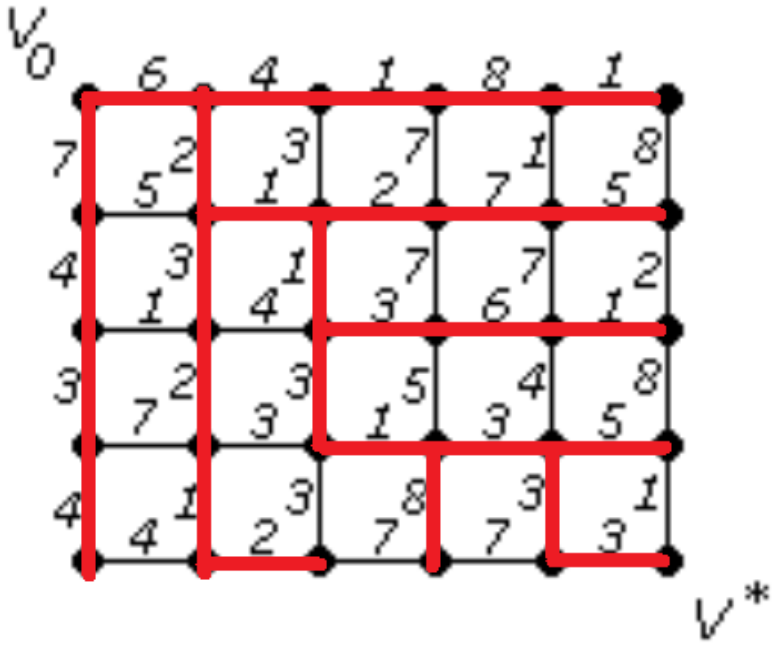


---

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

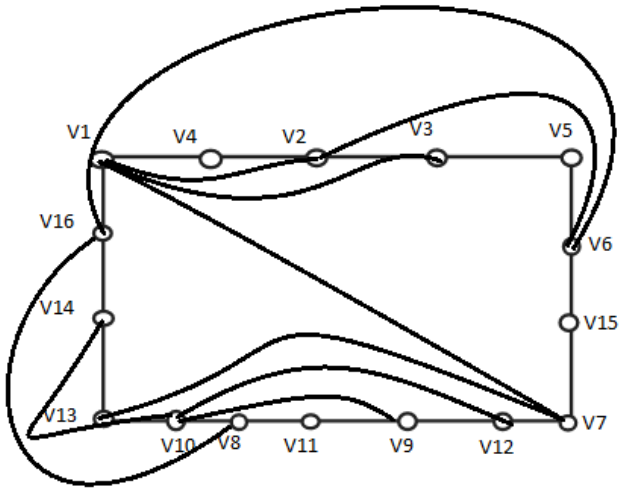


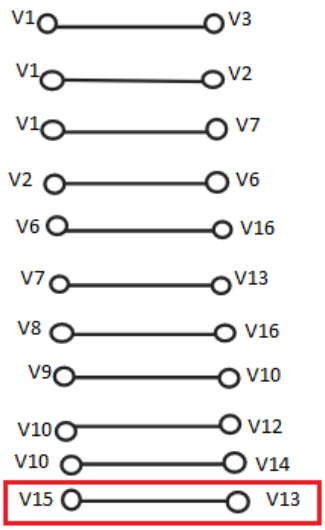
Розвязування:



S(V0,V*)=6+2+1+1+3+1+3+3+3=23

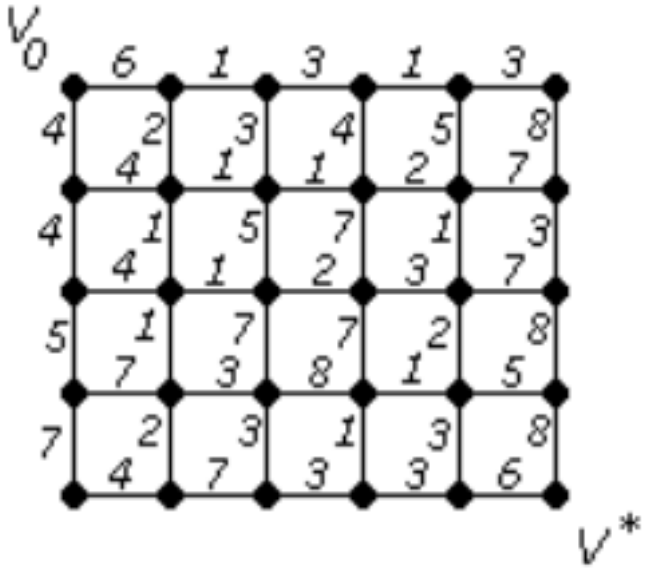
2)





Укладка графа в площині неможлива

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



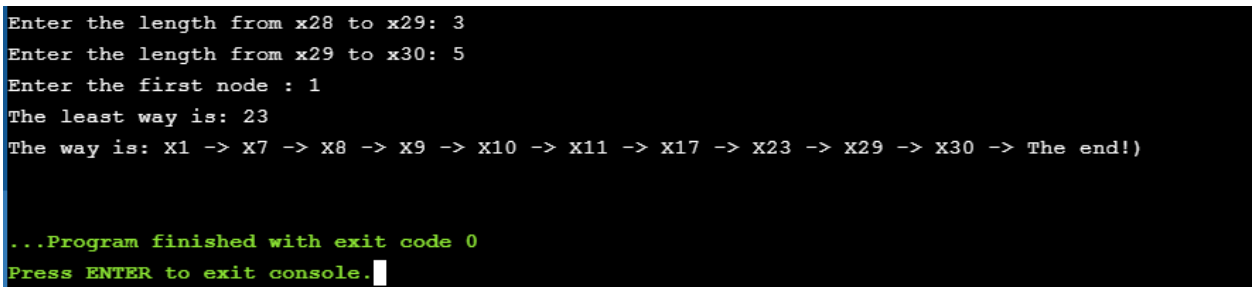

```cpp
#include<iostream>
using namespace std;
int n;
int i, j, q;
int dist[40];
bool visited[40];
int pred[40];
void createGraph(int c[40][40])
{
  int g1, g2;
  cout << "Enter the number of vertices: ";
  cin >> n;
  for (int i = 0; i < n; i++) {

    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;
      visited[i] = false;
    }
    dist[start-1] = 0;
    for (int count = 0; count < n - 1; count++)
    {
      int u = minDistance();
      visited[u] = true;
      for (int v = 0; v < n; v++)
        if (!visited[v] && c[u][v] &&
          dist[u] + c[u][v] < dist[v])
        {
          pred[v] = u;
          dist[v] = dist[u] + c[u][v];
        }
    }
    cout << "The least way is: ";
    cout << dist[29] << endl;
    cout << "The way is: ";
    cout << "X1 -> ";
      printPath(29);
      cout << "The end!)" << endl;
    //}
  }
int main()
{
  int c[40][40];
  createGraph(c);
  dijkstra(c);
  return 0;
}
```

```
Enter the number of vertices: 30
Enter size of (n*m) : 6
5
Enter the length from x1 to x2: 6
Enter the length from x1 to x7: 4
Enter the length from x2 to x3: 1
Enter the length from x2 to x8: 2
Enter the length from x3 to x4: 3
Enter the length from x3 to x9: 3
Enter the length from x4 to x5: 1
Enter the length from x4 to x10: 4
Enter the length from x5 to x6: 3
Enter the length from x5 to x11: 5
Enter the length from x6 to x7: 0
Enter the length from x6 to x12: 8
Enter the length from x7 to x8: 4
Enter the length from x7 to x13: 4
Enter the length from x8 to x9: 1
Enter the length from x8 to x14: 1
Enter the length from x9 to x10: 1
Enter the length from x9 to x15: 5
Enter the length from x10 to x11: 2
Enter the length from x10 to x16: 7
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: 5
Enter the length from x14 to x15: 1
Enter the length from x14 to x20: 1
Enter the length from x15 to x16: 2
Enter the length from x15 to x21: 7
Enter the length from x16 to x17: 3
Enter the length from x16 to x22: 7
Enter the length from x17 to x18: 7
Enter the length from x17 to x23: 2
Enter the length from x18 to x19: 0
Enter the length from x18 to x24: 8
Enter the length from x19 to x20: 7
Enter the length from x19 to x25: 7
Enter the length from x20 to x21: 3
Enter the length from x20 to x26: 2
Enter the length from x21 to x22: 8
Enter the length from x21 to x27: 3
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: 4
Enter the length from x26 to x27: 7
Enter the length from x27 to x28: 3
```



Висновок: я набула практичних вмінь та навичок з використання алгоритму Дейкстри.