Звіт

з лабораторної роботи №8

з курсу алгоритми і методи обчислень.

(ПОШУК НАЙКОРОТШИХ ШЛЯХІВ НА ГРАФАХ)

Студента 1-го курсу,

Парфенюка Тимофія Павловича

2019

Завдання:

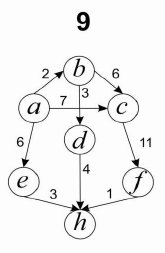
У графі знайти найкоротші шляхи від вказаної вершини до усіх інших вершин або між двома вказаними вершинами.

Алгоритм Дейкстри.

Складіть програму для реалізації алгоритму Дейкстри пошуку найкоротших шляхів до всіх вершин графа від вказаної вершини а . Спосіб представлення графа у пам’яті комп’ютера оберіть за бажанням. У програмі забезпечте:

• можливість вибору початкової вершини а (ввід з клавіатури);

• вивід на екран найкоротших шляхів у графі та їх довжин (ваг).



**Код:**

using System;

using System.Linq;

namespace Algoritm\_8

{

class Program

{

const int n = 7;

static int inf = 100000;

static string[] names = { "A", "B", "C", "D", "E", "F", "H" };

static int[,] V =

{{0,2,7,0,6,0,0},//A

{0, 0,6,3,0,0,0 },//B

{0,0,0,0,0,11,0 },//C

{0,0,0,0,0,0,4 },//D

{0,0,0,0,0,0,3 },//E

{0,0,0,0,0,0,1 },//F

{0,0,0,0,0,0,0 },//D

};//new int[n, n];

static public void PrintMas()

{

try

{

Console.Write(" ");

for (int i = 0; i < names.Length; i++)

{

Console.Write("{0}\t|", names[i]);

}

Console.WriteLine("");

for (int i = 0; i < V.GetLength(0); i++)

{

Console.Write($"{names[i]} ");

for (int j = 0; j < V.GetLength(1); j++)

{

Console.Write("{0}\t|", V[i, j]);

}

Console.WriteLine();

}

}

catch(Exception e)

{

Console.WriteLine(e.Message + "Something wrong!");

}

}

static void Dejkstra(int start)

{

int v;

int[] x = new int[n];

int[] D = new int[n];

int[] P = new int[n];

int end;

for (int i = 0; i < n; i++)

{

end = i;

if (end == start)

continue;

else

{

int u;

for (u = 0; u < n; u++)

{

D[u] = inf;

x[u] = 0;

}

P[start] = 0;

D[start] = 0;

x[start] = 1;

v = start;

do

{

for (u = 0; u < n; u++)

{

if (V[v, u] == 0) continue;

if (x[u] == 0 && D[u] > D[v] + V[v, u])

{

D[u] = D[v] + V[v, u];

P[u] = v;

}

}

int w = inf;

v = -1;

for (u = 0; u < n; u++)

{

if (x[u] == 0 && D[u] < w)

{

v = u;

w = D[u];

}

}

if (v == end)

{

Console.Write($"Shortest path from {names[start]} to {names[end]} : ");

u = end;

string outp = "";

while (u != start)

{

//Console.Write($" {names[u]} ");

outp = outp + names[u];

u = P[u];

}

outp = outp + names[start];

Console.Write($" {new string(outp.Reverse().ToArray())}");

Console.WriteLine();

Console.WriteLine($"Path length = {D[end]}");

break;

}

if (v == -1)

{

Console.WriteLine($"Сan't find: {names[start]}{names[end]} ");

break;

}

x[v] = 1;

} while (true);

}

}

}

static int FindLetter(string L)

{

for(int i = 0; i< names.Length; i++)

{

if (names[i] == L)

return i;

}

return -100;

}

static void Main()

{

PrintMas();

try

{

int start;

do

{

Console.Write("Start node: ");

start = FindLetter(Console.ReadLine());

Console.WriteLine();

} while (start < 0 || start > 7);

Dejkstra(start);

}

catch(Exception e)

{

Console.WriteLine(e.Message + "Something wrong!");

Console.WriteLine("XC");

}

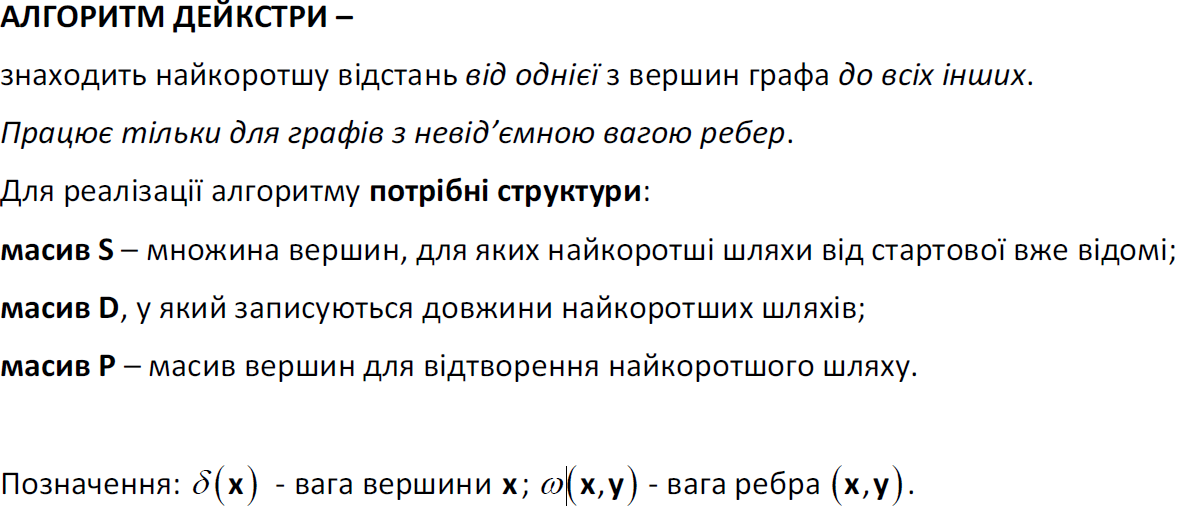
}

}

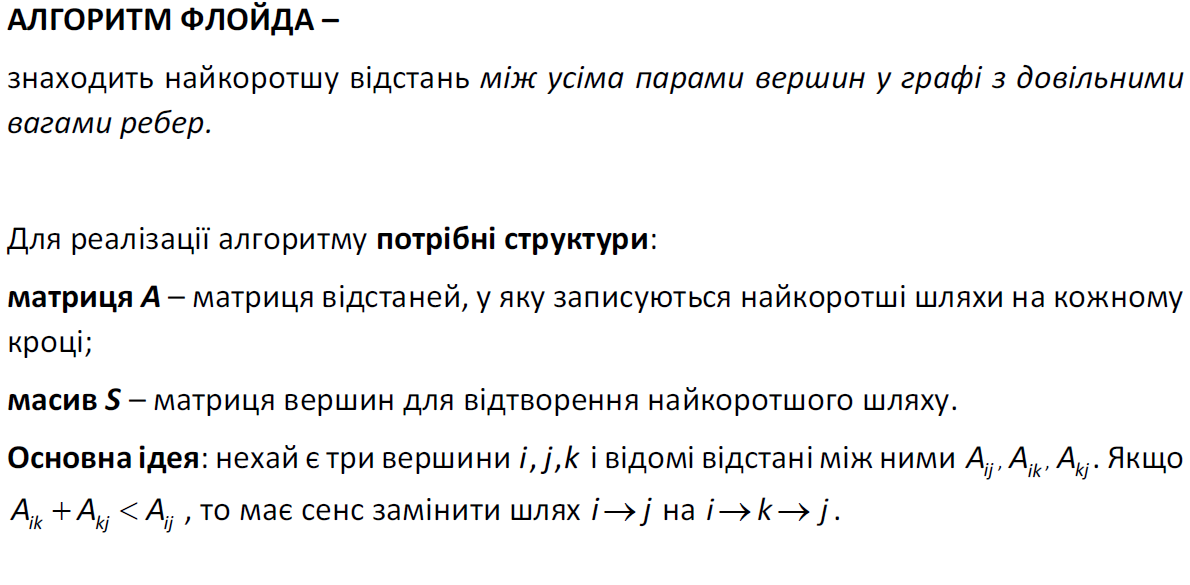
}

***Висновок:***

В теорії графів, задача про найкоротший шлях полягає в знаходженні такого шляху між двома вершинами (або вузлами) графу, що сума ваг ребер з яких він складається мінімальна. Прикладом може бути знаходження найкоротшого шляху між двома пунктами на дорожній мапі; в цьому випадку, вершинами є пункти, а ребрами — відтинки дороги із вагами, рівними часу, необхідному для подолання цього відрізку.



Реалізація алгоритму Дейкстри потребує O ( V^2 ) дій. У ній використовується масив відстаней та масив позначок. На початку алгоритму відстані заповнюються великим позитивним числом (більшим максимального можливого шляху в графі), а масив позначок заповнюється нулями. Потім відстань для початкової вершини вважається рівною нулю і запускається основний цикл. На кожному кроці циклу ми шукаємо вершину з мінімальною відстанню і прапором рівним нулю. Потім ми встановлюємо в ній позначку 1 і перевіряємо всі сусідні з нею вершини. Якщо в ній відстань більша, ніж сума відстані до поточної вершини і довжини ребра, то зменшуємо його. Цикл завершується коли позначки всіх вершин стають рівними 1.



В інформатиці, алгоритм Флойда-Воршелла використовується для розв'язання задачі про найкоротший шлях у зваженому графі з додатними або від'ємними вагами ребер (але без від'ємнозначних циклів). При звичайній реалізації алгоритм видасть довжини (сумарні ваги) найкоротших шляхів між всіма парами вершин, хоча він не видасть інформацію про самі шляхи. Різні версії алгоритму також використовуються для знаходження транзитивного замикання в відношенні R, або (враховуючи Метод Шульце), для знаходження найбільшого шляху (англ. widest path problem) між всіма парами вершин у зваженому графі.

***Вхідні та вихідні данні:***

