Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ»

Кафедра САПР ВС

К защите

Руководитель проекта

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ**

по дисциплине

«Лингвистическое и программное обеспечение САПР»

по теме

«Пути на графах»

Выполнила студентка группы 546

Леонова И.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата сдачи на проверку, подпись

Руководитель проекта

д.т.н., проф. Скворцов С.В. \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

оценка дата защиты, подпись

Рязань 2018

**Содержание**

[Введение 3](#_Toc531715419)

[1. Постановка задачи 4](#_Toc531715420)

[2. Алгоритм решения задачи 5](#_Toc531715421)

[3. Программная реализация алгоритма 8](#_Toc531715422)

[4. Тестирование программы 9](#_Toc531715423)

[5. Программная документация 11](#_Toc531715424)

[Заключение 12](#_Toc531715425)

[Библиографический список 13](#_Toc531715426)

[Приложение 1. 14](#_Toc531715427)

# **Введение**

Нахождение кратчайшего пути на сегодняшний день является важной задачей и используется практически везде, начиная от нахождения оптимального маршрута между двумя объектами на местности (например, кратчайший путь от дома до университета), в системах автопилота, для нахождения оптимального маршрута при перевозках,  для нахождения кратчайшего пути прохождения Internet-пакета по сети и т.п.

Кратчайший путь рассматривается при помощи некоторого математического объекта, называемого графом. Поиск кратчайшего пути ведется между двумя заданными вершинами в графе. Результатом является путь — последовательность вершин, в которой каждая вершина соединена со следующим ребром.

Целью данной курсовой работы является разработка программы нахождения кратчайших путей между всеми парами вершин методом динамического программирования. Необходимо составить развернутую схему алгоритма, разработать программу, обеспечивающую выполнение заданного алгоритма, затем выполнить ее отладку.

1. **Постановка задачи**

Рассмотрим связный ориентированный граф G = (X, U), где Х = {х1, х2,..., хn} - множество вершин, U= {u1, u2,..., um} - множество дуг вида Up (xi, xj); xi, xj ϵX .

Наиболее часто для заданного графа G = (X, U) требуется определить кратчайший путь µ[s, t] из некоторой выделенной вершины s (источника) в другую заранее заданную вершину t (сток), где s,t ϵ X. Если значения ɭ(xi, xj) ≥ 0 рассматривать как стоимости прохождения единицы потока по дугам (xi, xj) ϵ U, то задачу можно сформулировать так: определить поток единичной мощности из источника s в сток t, который имеет минимальную стоимость.

Задача о кратчайшем пути — задача поиска самого короткого пути (цепи) между двумя точками (вершинами) на графе, в которой минимизируется сумма весов рёбер, составляющих путь.

Обобщение представленной выше задачи: для заданной начальной вершины s ϵ X найти кратчайшие пути µ[s, xi] между s и всеми другими вершинами xi ϵ Х\ s.

Задача о кратчайшем пути является одной из важнейших классических задач теории графов. Сегодня известно множество алгоритмов для её решения, например, алгоритм Форда-Беллмана, алгоритм Дейкстры, динамического программирования и др.

Я буду рассматривать метод динамического программирования для нахождения кратчайших путей между парами вершин в графе.

1. **Алгоритм решения задачи**

Алгоритмы поиска кратчайших путей в бесконтурных орграфах, основанные на методе динамического программирования, используют следующий принцип оптимальности Беллмана: "любой подпуть минимального пути является минимальным путем между соответствующими вершинами".

Описание алгоритма:

Данные:

Бесконтурный орграф G = (X, U) с n вершинами, имеющими правильную нумерацию;

Выделенный источник s = х1, произвольные веса дуг ɭ(xi, xj).

Результаты:

Длины \*(xi) кратчайших путей µ[s, xi] от источника s до вершин xi (i = 2, n).

Шаг 1 (присвоение начальных значений)

Задать пометки (xi) = ∞ вершинам (xi) ϵ X \ s, v и \*(x1) = 0 для x1 = s. Присвоить j=1.

Шаг 2 (изменение пометок) Присвоить j = j + 1 и для вершины xj вычислить окончательное значение пометки:

,

где T=Г-1 (). При T= Ø пометка вершины xj не изменяется.

Шаг 3 (проверка условия окончания) Если j = n, то конец алгоритма. Иначе переход к шагу 2.

Схема алгоритма представлена на рисунках 1 и 2.

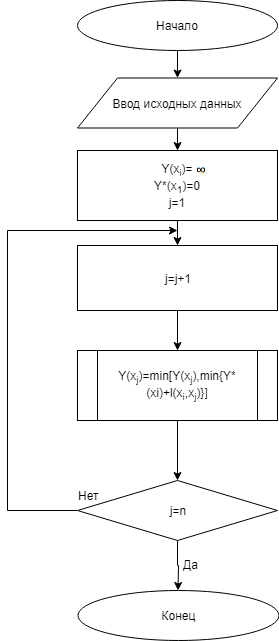
****

Рисунок 1 – Схема алгоритма.

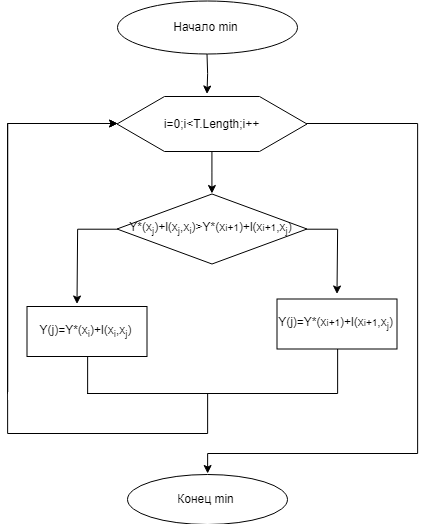


Рисунок 2 – Функция min.

1. **Программная реализация алгоритма**

Для программной реализации был выбран язык C# в среде Visual studio 2017. C# — изящный объектно-ориентированный язык со строгой типизацией, позволяющий разработчикам создавать различные безопасные и надежные приложения, работающие на платформе .NET Framework.

Входные данные задаются в текстовом файле «in.txt». Выходные данные формируются в отдельном текстовом файле «out.txt».

Для работы программы необходимо, чтобы в директории « D:\temps » , находился текстовый документ: «in.txt»

При выполнении программы ведется диагностика ошибок.

Все вычисления производятся в программе автоматически и не требуют вмешательства пользователя.

1. **Тестирование программы**

Пусть ориентированный граф, в котором отсутствуют контуры, имеет вид, показанный на рисунке 3. Необходимо определить кратчайший путь µ[x1, x9] .

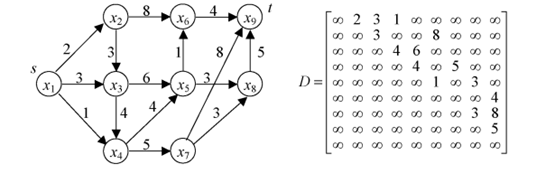
****

Рисунок 3 – Ориентированный граф.

Поиск кратчайшего пути ручным способом представлен ниже:

j=1

j=2

j=3

j=4

j=5

j=6

j=7

j=8

j=9

Результаты работы алгоритма показаны в виде вектора пометок , который имеет вид: (0, 2,3,1,5, 6, 6, 9, 10).

Кратчайший путь µ[x1, x9] имеет вид: (x1,x4,x5,x6,x9).

Далее продемонстрируем решение контрольного примера с применением разработанной программы. Для этого заполним документ «in.txt» исходными данными (рисунок 4). и запустим программу. Результат выводится в текстовый документ (рисунок 5). и в консоль (рисунок 6).

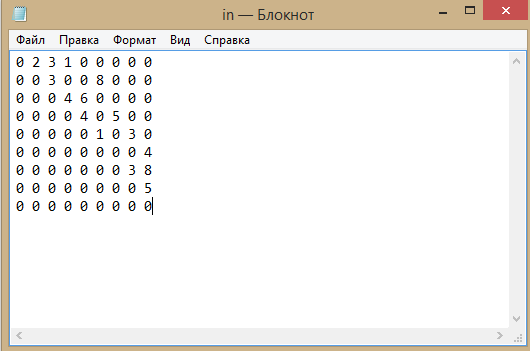


Рисунок 4 – Исходные данные.

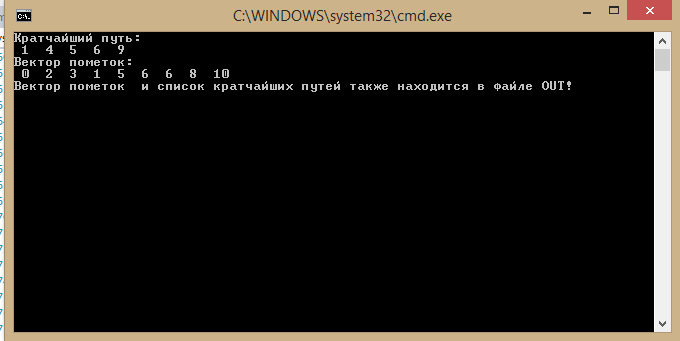


Рисунок 5 – Результат работы программы в консоли.

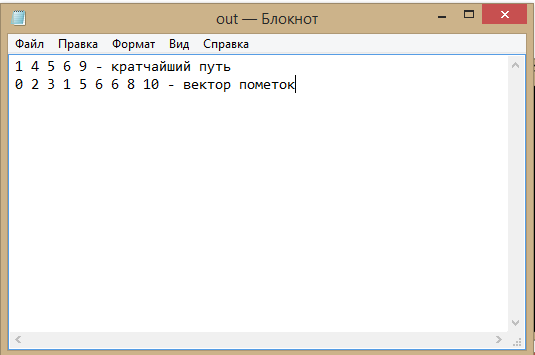


Рисунок 6 – Результат работы программы в «out.txt».

1. **Программная документация**

Назначение программы:

Наименование программы: WaysOnGraphs. Программа предназначена для построения кратчайших путей между всеми парами вершин на ориентированном графе методом динамического программирования.

Условия применения:

Минимальный состав технических средств: персональный компьютер, включающий в себя:

* 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц или выше;
* 1 ГБ (для 32-разрядного процессора) или 2 ГБ (для 64-разрядного процессора) ОЗУ.

На данном ПК должна быть установлена лицензионная версия ОС линейки MS Windows. Предпочтительная версия: Windows 8.1 (64-разрядная).

Сообщения оператору:

Введём заведомо некорректные исходные данные, например, отрицательные значения длин дуг. Программа выдаёт окно с ошибкой (Рисунок 7). Также ведётся диагностика и других ошибок, например, отсутствие файла с входными данными.

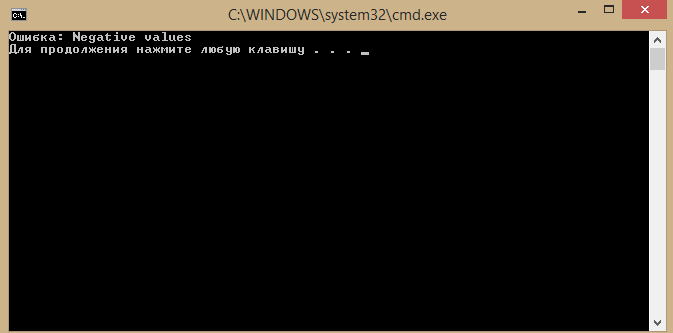


Рисунок 7 – Пример ошибки.

# **Заключение**

В данной курсовой работе была освещена задача поиска кратчайших путей на графе и рассмотрен алгоритм динамического программирования для ее решения. Была реализована программа, построения кратчайших путей между всеми парами вершин на ориентированном графе.

# **Библиографический список**

1. Лингвистическое и программное обеспечение САПР: Методические указания к курсовой работе, РГРТА.; сост.: С.В. Скворцов, В.И. Хрюкин. Рязань, 2001.
2. Алгоритмы построения путей на графах: Методические указания к практическим занятиям, РГРТА. Сост.: С.В. Скворцов, В.И. Хрюкин, Л.Б. Михеева. Рязань, 2004.
3. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов. – М.: Мир, 1981.

# **Приложение 1.**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WaysOnGraphs

{

class Program

{

static double Min(double[] ways, int n)

{

double result = Double.PositiveInfinity;

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

{

if (ways[i] < result)

{

result = ways[i];

}

}

return result;

}

static void IndexPrint(int[] index, int n)

{

string s2 = "";

int[] distIndex = index.Distinct().ToArray();

distIndex[distIndex.Length - 1] = n;

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

{

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

s2 += distIndex[i];

}

s2 += " Кратчайший путь";

string writePath = @"D:\temp\out.txt";

using (StreamWriter sw = new StreamWriter(writePath, false, System.Text.Encoding.Default))

{

sw.WriteLine(s2);

}

} static int Index(double[] ways, int n, double result)

{

int index = 0;

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

{

if (ways[i] == result)

{

index = i + 1;

}

}

return index;

}

static void InitialValues(double[] lable, int n, double[,] x)

{

double inf = Double.PositiveInfinity;

lable[0] = 0;

for (int i = 1; i < n; i++)

{

lable[i] = inf;

}

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

{

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

{

if (x[i, j] == 0)

{

x[i, j] = inf;

}

else if (x[i, j] < 0)

{

throw new Exception("Negative values");

}

}

}

Sort(x);

}

static double[,] ReadFromFile(int n)

{

double[,] x = new double[n, n];

string path = @"D:\temp\in.txt";

try

{

string[] mass = File.ReadAllLines(path);

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

{

double[] m = mass[p].Split(new char[] { ' ' },

StringSplitOptions.RemoveEmptyEntries).Select(s => double.Parse(s)).ToArray();

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

{

x[p, i] = m[i];

}

}

return x;

}

catch { throw new Exception("Error read from file"); }

}

static double[] MinWay(double[] lable, int n, double[,] x)

{

double[] ways = new double[n];

int id = 0; int[] index = new int[n];

for (int i = 1; i < n; i++)

{

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

{

ways[j] = x[j, i] + lable[j];

}

lable[i] = Min(ways, n);

index[id] = Index(ways, n, lable[i]);

id++;

if (i == n - 1) { IndexPrint(index, n); }

}

return lable;

}

static void Sort(double[,] x)

{

int count = 0;

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

{

for (int j = 0; j < x.Length; j++)

{

if (x[i, j] == Double.PositiveInfinity)

{

count++;

}

if (count == x.Length)

{

x.RemoveAt(count);

}

}

count = 0;

}

}

static void Print(double[] lable, int n)

{

string s = "";

Console.WriteLine();

Console.WriteLine("Вектор пометок и список кратчайших путей так же находится в файле OUT!");

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

{

s += lable[i].ToString() + " ";

}

s += " Вектор пометок";

string writePath = @"D:\temp\out.txt";

using (StreamWriter sw = new StreamWriter(writePath, false, System.Text.Encoding.Default))

{

sw.WriteLine(s);

}

}

static void Main(string[] args)

{

const int n = 9;

double[] lable = new double[n];

double[,] x = new double[n, n];

x = ReadFromFile(n);

InitialValues(lable, n, x);

lable = MinWay(lable, n, x);

Print(lable, n);

Console.ReadLine();

}

}

}