Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Реализация алгоритма Флойда-Уоршелла поиска кратчайших путей в графе»

Выполнил: студент группы 21ВВ1

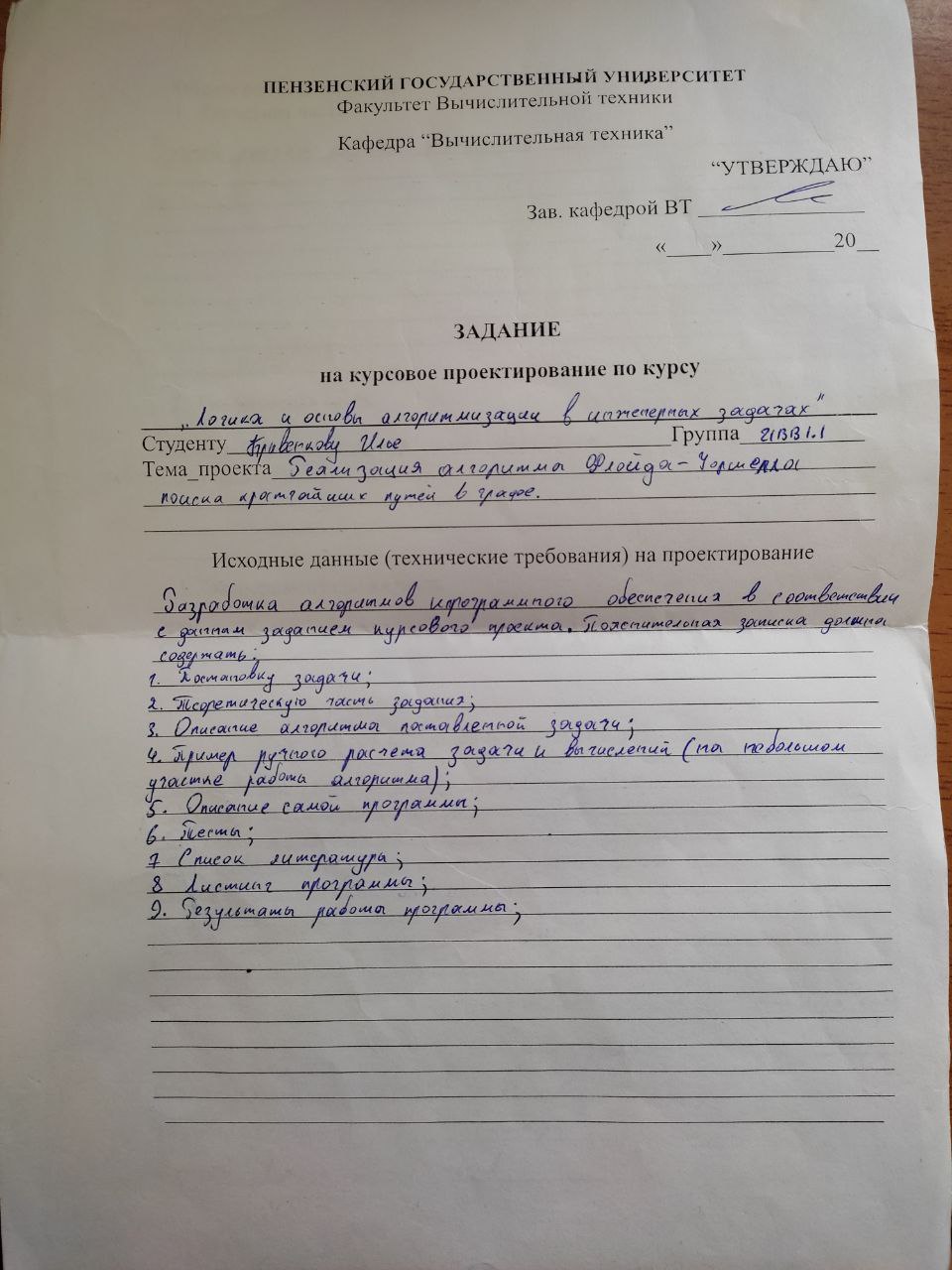
Кривенков И.В.

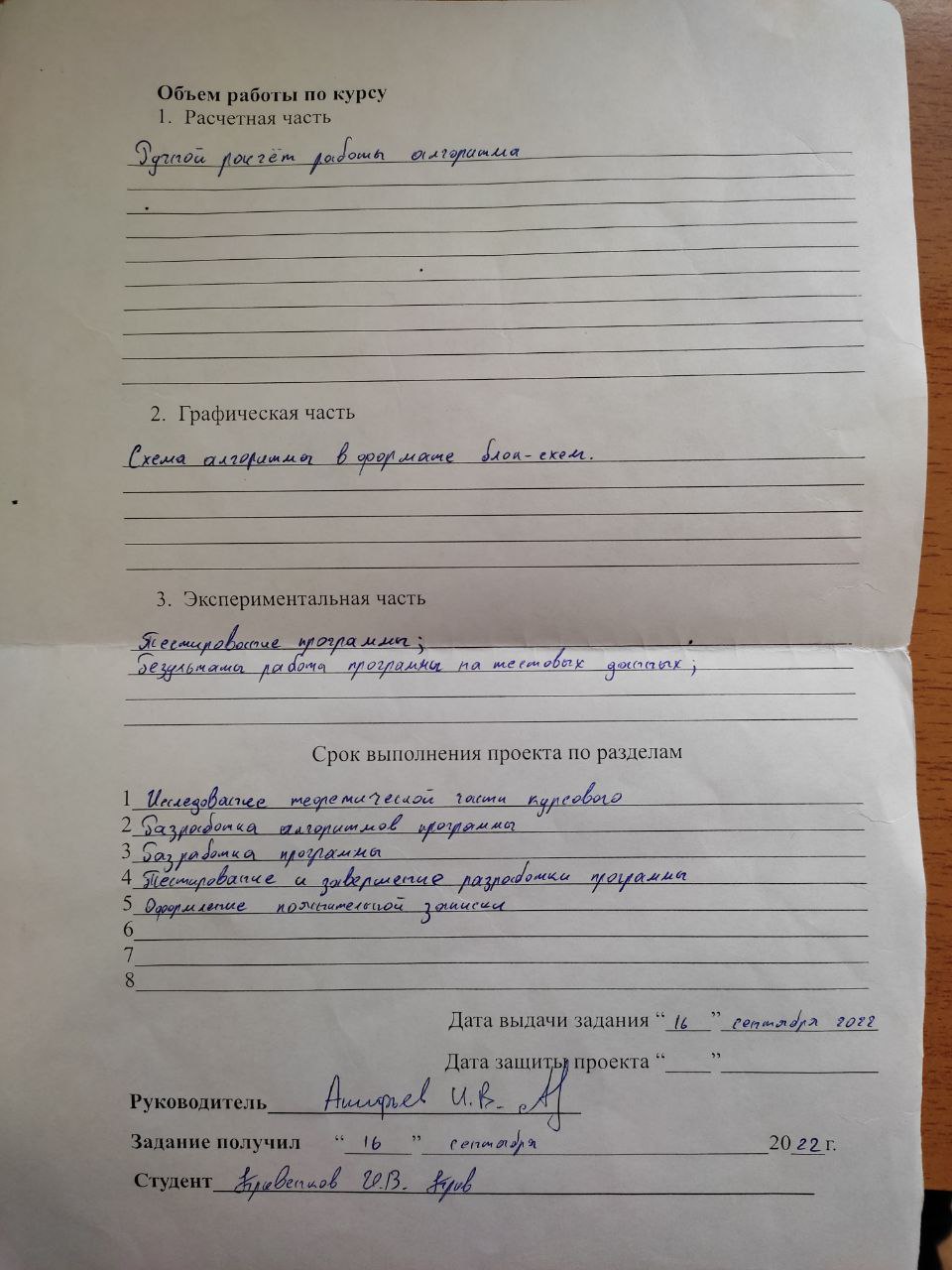
Приняли:

д.т.н. профессор Митрохин М.А.

к.т.н. доцент Юрова О.В.

Пенза 2022





**Содержание**

Реферат 5  
[Введение 6](#_Toc121615137)

1. [Постановка задачи 7](#_Toc121615138)

2. [Теоретическая часть 8](#_Toc121615139)

3. [Описание алгоритма программы 9](#_Toc121615140)

4. [Описание программы 10](#_Toc121615141)

5. [Тестирование 11](#_Toc121615142)

6. [Ручной расчет задачи 12](#_Toc121615143)

[Заключение 14](#_Toc121615144)

[Список литературы 15](#_Toc121615145)

[Приложение A. Листинг программы 16](#_Toc121615146)

**Реферат**

Отчет 17 страниц, 9 рисунков.

ГРАФ, ФЛОЙД-УОРШЕЛ, ПОИСК КРАТЧАЙШИХ ПУТЕЙ,

Цель исследования – разработка программы, поиска кратчайших путей алгоритмом Флойд-Уоршела.

В работе рассмотрены правила поиска в глубину, на основе которых

находится компонента сильной связности орграфа. Установлено, что с

помощью данного алгоритма можно найти кратчайшие пути между вершинами графа

# Введение

Алгоритм Флойда–Уоршелла- это алгоритм поиска кратчайших путей во взвешенном ориентированном графе с положительным или отрицательным весом ребер (но без отрицательных циклов). За одно выполнение алгоритма будут найдены длины (суммарные веса) кратчайших путей между всеми парами вершин.

Алгоритм Флойда–Уоршелла является примером динамического программирования и был опубликован в своей ныне признанной форме Робертом Флойдом в 1962 году. Однако он по сути такой же, как алгоритмы, ранее опубликованные Бернардом Роем в 1959 году, а также Стивеном Уоршеллом в 1962 году для поиска транзитивного замыкания графа, и тесно связан с алгоритмом Клини (опубликовано в 1956 г.) для преобразования детерминированного конечного автомата в регулярное выражение. Современная формулировка алгоритма в виде трех вложенных циклов for была впервые описана Питером Ингерманом также в 1962 году.

В качестве среды разработки была выбрана среда MicrosoftVisualStudio, язык программирования – C++.

Целью курсовой работы является реализации алгоритма Флойда-Уоршелла поиска кратчайших путей в графе.

# Постановка задачи

Требуется разработать программу, реализующая алгоритм Флойда-Уоршеллапоиска кратчайших путей в графе.

Исходный граф в программе должен задаваться матрицей смежности, причем при генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы смежности. После обработки этих данных на экран должна выводиться матрица смежности графа, кратчайшие расстояния для каждой вершины. Устройство ввода – клавиатура и мышь.

# Теоретическая часть

Граф G (рисунок 1) задается множеством вершин X1, X2, ..., Xn. и множеством ребер, которым сопоставимы числа. Граф с подобными ребрами называется взвешенным. Длина пути во взвешенном графе - это сумма длин (весов) тех ребер, из которых состоит путь.

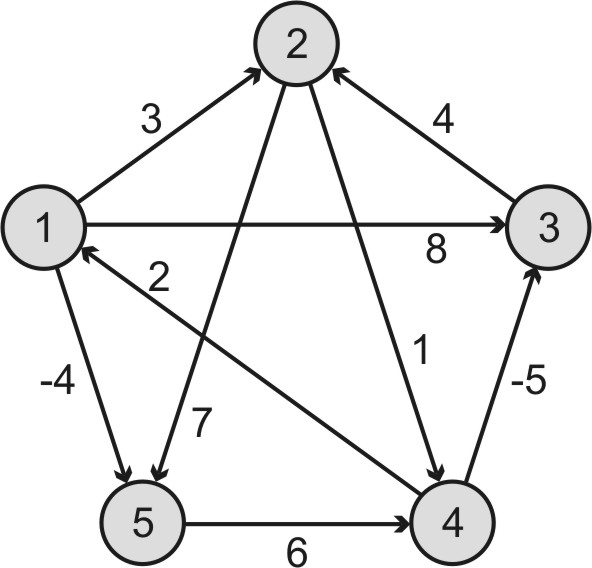


Рисунок 1

При представлении взвешенного графа матрицей смежности информация о ребрах графа хранится в квадратной матрице, где присутствие пути из одной вершины в другую обозначается весом ребра, иначе бесконечно большой величиной.

Алгоритмов нахождения кратчайшего пути в графе достаточно много: волновой алгоритм, алгоритм Левита, переборный алгоритм, алгоритм Беллмана — Форда, алгоритм Джонсона и другие. Есть более популярные методы, которые обширно применяются в наше время, например алгоритм Дейкстры или Флойда-Уоршелла. Остановимся на алгоритме Флойда-Уоршелла.

Дан ориентированный или неориентированный взвешенный граф G с n вершинами. Требуется найти значения всех величин dij — длины кратчайшего пути из вершины i в вершину j.

Предполагается, что граф не содержит циклов отрицательного веса (тогда ответа между некоторыми парами вершин может просто не существовать — он будет бесконечно маленьким).

Кратчайший путь из вершины i в вершину j может проходить, как только через них самих, так и через множество других вершин m∈(1, …, |V|). Оптимальнымиз i в j будетпутьилинепроходящийчерезm, илипроходящий. Заключитьоналичиивторогослучая, значитустановить, что такой путь идет из i до m, а затем из m до j, поэтому должно заменить, значение кратчайшего пути D[i][j] суммой D[i][m]+D[m][j].

# Описание алгоритма программы

Нам даны матрица смежностиadj[n][n] орграфа D. Нужно найти расстояния между всеми парами вершин D[i,j] = d(vi,vj).

1. Для всех i = 1,…,n, j = 1,…,n положим D[i,j] = adj[i][j] .
2. Для всех i = 1,…,n положим D[i,i] = 0.
3. Для m = 0 пока k<nделать m++
4. Дляi = 0 покаi<nделать i++

5. для j = 0 пока j<nделать j++

6. D[i,j] = min (D[i,j], D[i,m] + D[m,j]).

7. конец циклов

Полученные значения D[i,j] дают расстояния между вершинами vi и vj.

# Описание программы

Для написания данной программы использован язык программирования C++. Проект был создан в виде консольного приложения Win32.

Работа программы начинается с ввода количества вершин графа.

Далее происходит генерация взвешенной матрицы смежности.

voidrandAdjMatrix(int\*\* matrix, intnumberOfVert) {

int min = 0, max = 50, num;

srand(time(NULL));

for (inti = 0; i<numberOfVert; i++) {

for (int j = 0; j <numberOfVert; j++) {

num = 0 + rand() % (1 - 0 + 1);

if (num == 1) {

num = min + rand() % (max - min + 1);

matrix[i][j] = num;

}

elsematrix[i][j] = INF;

}

matrix[i][i] = 0;

}

}

Далее выводятся элементы массива, а именно матрица смежности графа.

for (inti = 0; i<numberOfVert; i++) {

for (int j = 0; j <numberOfVert; j++) {

if (matrix[i][j] == INF) {

cout<<"INF"<<" ";

}

else {

cout<<matrix[i][j] <<" ";

}

}

cout<<endl;

}

После выводится матрица кратчайших путей (рисунок 2)

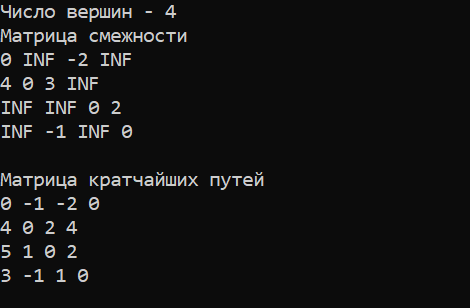


Рисунок 2

# Тестирование

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено проблемы, связанных с вводом данных, алгоритмом программы.

Ниже продемонстрирован результат тестирования программы при вводе из файла различных матриц смежности.

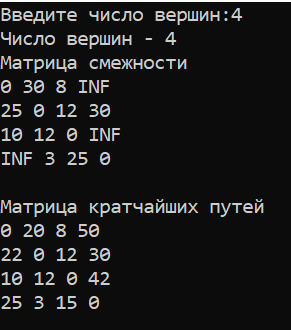


Рисунок 3. Тестирование при вводе матрицы смежности с количество вершин = 4

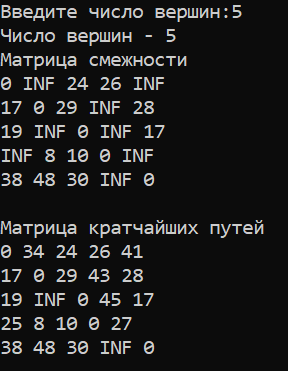


Рисунок 4. Тестирование при вводе матрицы смежности с количество вершин = 5

# Ручной расчет задачи

Определим длины минимальных путей между любыми парами вершин орграфа, изображенного на рисунке 5. Все вычисления будем проводить с помощью матриц *D*.

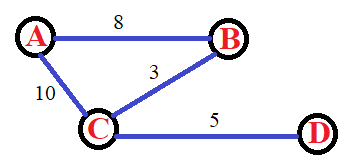


Рисунок 5

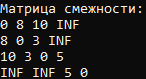


Рисунок 6

Положим m = 0. Если i = 0 или j = 0, то элементы матрицы остаются без изменения, т.к.

D[i,j] = min (D[i,j], D[i,m] + D[m,j]).

Поэтому рассмотрим случай, когдаm=2, i = 0, а j = 3, но будет изменяться. ТогдаD[0,3] = min (D[0,3], D[0,2] + D[2,3]) = min (∞,10+5) = 15. Так как 15 <∞, мы находим кратчайший путь из точки А в точку D через С. Так же находим кратчайший путь из точки В в точку D через Спри m=2, i = 1, j = 3:  
D[1,3] = min (D[1,3], D[1,2] + D[2,3]) = min (∞,3+5) = 8.  
   
  
 Продолжаем процесс до тех пор, пока i ≤4 и j ≤4. Положим m = 2 и продолжим рассуждения дальше.

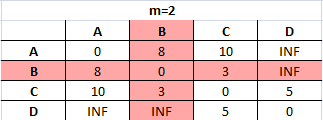
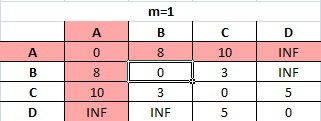


Рисунок 7

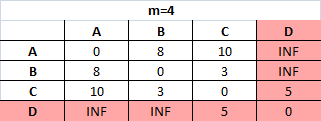
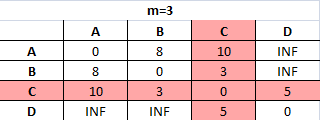


Рисунок 8

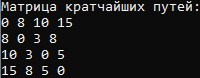


Рисунок 9 – конечный результат

Таким образом, сделав ручной подсчет на данном примере, мы удостоверились в правильности данного алгоритма.

# Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Флойда-Уоршелла поиска кратчайших путей в графе.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежностей. Приобретены навыки по осуществлению алгоритма Флойда-Уоршелла. Углублены знания языка программирования C++.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

# Список литературы

1. Электронный ресурс: https://ru.wikipedia.org/wiki/ Дата обращения: 08.12.2022.
2. Электронный ресурс: http://www.cyberforum.ru/ Дата обращения: 08.12.2022.
3. Дискретная математика для программистов, Р. Хаггарти ISBN 5-94836-016-4, 0-201-73047-2, 2005.

# Приложение A. Листинг программы

#include<iostream>

#include<algorithm>

#include<fstream>

#defineINF 51

usingnamespace std;

void newMatrix(int\*\*&matrix, intnumberOfVert) {

matrix = newint\* [numberOfVert];

for (int i = 0; i <numberOfVert; i++) {

matrix[i] = newint[numberOfVert];

}

}

void randAdjMatrix(int\*\* matrix, intnumberOfVert) {

int min = 0, max = 50, num;

srand(time(NULL));

for (int i = 0; i <numberOfVert; i++) {

for (int j = 0; j <numberOfVert; j++) {

num = 0 + rand() % (1 - 0 + 1);

if (num == 1) {

num = min + rand() % (max - min + 1);

matrix[i][j] = num;

}

elsematrix[i][j] = INF;

}

matrix[i][i] = 0;

}

}

void printMatrix(int\*\* matrix, intnumberOfVert) {

for (int i = 0; i <numberOfVert; i++) {

for (int j = 0; j <numberOfVert; j++) {

if (matrix[i][j] == INF) {

cout <<"INF"<<" ";

}

else {

cout <<matrix[i][j] <<" ";

}

}

cout << endl;

}

}

void originalFloydWarshall(int\*\* matrix, intnumberOfVert) {

for (int k = 0; k <numberOfVert; k++) {

for (int i = 0; i <numberOfVert; i++) {

for (int j = 0; j <numberOfVert; j++) {

matrix[i][j] = min(matrix[i][j], matrix[i][k] + matrix[k][j]);

}

}

}

}

int main() {

setlocale(LC\_ALL, "rus");

int numberOfVert;

int flag;

int\*\* matrix = NULL;

ifstream fin("123.txt");

cout <<"Выберитережимработы: 1 - ручнойвводматрицы, 2 - вводматрицыизфайла"<< endl;

cin >> flag;

switch (flag) {

case 1:

cout <<"Введите число вершин:";

cin >> numberOfVert;

cout <<"Числовершин - "<< numberOfVert << endl;

newMatrix(matrix, numberOfVert);

randAdjMatrix(matrix, numberOfVert);

fin.close();

break;

case 2:

fin >> numberOfVert;

newMatrix(matrix, numberOfVert);

for (int i = 0; i < numberOfVert; i++) {

for (int j = 0; j < numberOfVert; j++) {

fin >> matrix[i][j];

}

}

break;

default:

cout <<"Ошибка. Попробуйтеснова"<< endl;

break;

}

if (flag < 0 || flag > 1)

return 0;

cout <<"Матрицасмежности:"<< endl;

printMatrix(matrix, numberOfVert);

cout << endl;

originalFloydWarshall(matrix, numberOfVert);

cout <<"Матрица кратчайших путей:"<< endl;

printMatrix(matrix, numberOfVert);

cout << endl;

return 0;

}