

Министерство образования Российской Федерации
Пензенский государственный университет
Кафедра "Вычислительная техника"

Пояснительная Записка

К курсовому проектированию

По курсу "Логика и основы алгоритмизации

в инженерных задачах"

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

21.12.22
хорошо
Фед

Выполнил:

Студент группы 21ВВ2

Щеглов Д.А.

Приняли:

Митрохин М.А.

Юрова О.В.

Пенза 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

«__»__ 20__

ЗАДАНИЕ

на курсовое проектирование по курсу

Логика и основы алгоритмизации в инженерных задачах
Студенту Мельников Дмитрий Александрович Группа 21ВВ2
Тема проекта Реализация алгоритма Флойда-Уоршелла
нахождения кратчайших путей в графе

Исходные данные (технические требования) на проектирование

Разработка алгоритмов и программного обеспечения
в соответствии с данным заданием курсового проекта
Пояснительная записка должна содержать:

- 1) Пояснение задачи
- 2) Постановка задачи
- 3) Описание алгоритма нахождения кратчайших путей
- 4) Пример работы расчета задачи и результаты
- 5) Описание работы программы
- 6) Тесты
- 7) Список литературы
- 8) Заключение
- 9) Резюме работы программы

Объем работы по курсу

1. Расчетная часть

Другой объем работы алгоритм

2. Графическая часть

Схема алгоритма в формате Блок-схемы

3. Экспериментальная часть

Тестирование программы
Тестирование работы программы на тестовых данных

Срок выполнения проекта по разделам

- 1 Изучение исходных данных задачи к
- 2 Разработка алгоритма программы к
- 3 Разработка программы к
- 4 Тестирование и завершение разработки программы к
- 5 Оформление лабораторной работы к
- 6
- 7
- 8

Дата выдачи задания "18" Октября

Дата защиты проекта " " "

Руководитель

Задание получил

"18" Октября

2022 г.

Студент

Шелев Дмитрий Алексеевич

Содержание

Реферат.....	Error! Bookmark not defined.
Введение.....	4
1.Постановка задачи	5
2.Теоретическая часть.....	6
3.Описание алгоритма программы.....	9
4.Описание программы.....	10
5.Тестирование программы.....	18
6.Ручной расчет задачи.....	21
Заключение	23
Список литературы	24
Приложение А. Листинг.....	25

Реферат

Отчет 32 страница, 22 рисунка

Граф, теория графов, Алгоритм Флойда-Уоршелла, алгоритм поиска кратчайших путей

Цель исследования – разработка программы ,способная рассчитать кратчайший путь между вершинами графа

В работе используется алгоритм Флойда-Уоршелла, с его помощью можно найти кратчайший путь между вершинами графа

Введение

В математической теории графов и информатике граф — это совокупность непустого множества вершин и множества пар вершин. Объекты представляются как вершины, или узлы графа, а связи — как дуги, или рёбра. Для разных областей применения виды графов могут различаться направленностью, ограничениями на количество связей и дополнительными данными о вершинах или рёбрах.

Многие структуры, представляющие практический интерес в математике и информатике, могут быть представлены графами.

Математические модели в виде графов широко используются при моделировании разнообразных явлений, процессов и систем. Многие теоретические и реальные прикладные задачи могут быть решены при помощи тех или иных процедур анализа графовых моделей. В качестве примера можно привести задачу составления маршрута движения транспорта между различными городами при заданном расстоянии между населёнными пунктами.

В данной работе рассматривается алгоритм Флойда — Уоршелла — алгоритм для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа. Разработан в 1962 году Робертом Флойдом и Стивеном Уоршеллом.

1.Постановка задачи

Требуется разработать программу, которая реализует поиск кратчайших путей во взвешенном графе с помощью алгоритма Флойда-Уоршелла.

Программа должна содержать в себе:

- Текстовое или графическое меню
- Возможность задания пользователем размера графа
- Возможность выбора автоматического или ручного задания графа
- Возможность сохранения результатов работы программы

2. Теоретическая часть

Обозначим длину кратчайшего пути между вершинами u и v , содержащего, помимо u и v , только вершины из множества:

$$\{1..i\} \text{ как } d_{uv}^{(i)}, d_{uv}^{(0)} = \omega_{uv}.$$

На каждом шаге алгоритма, мы будем брать очередную вершину (пусть её номер — i) и для всех пар вершин u и v вычислять:

$$d_{uv}^{(i)} = \min(d_{uv}^{(i-1)}, d_{ui}^{(i-1)} + d_{iv}^{(i-1)}).$$

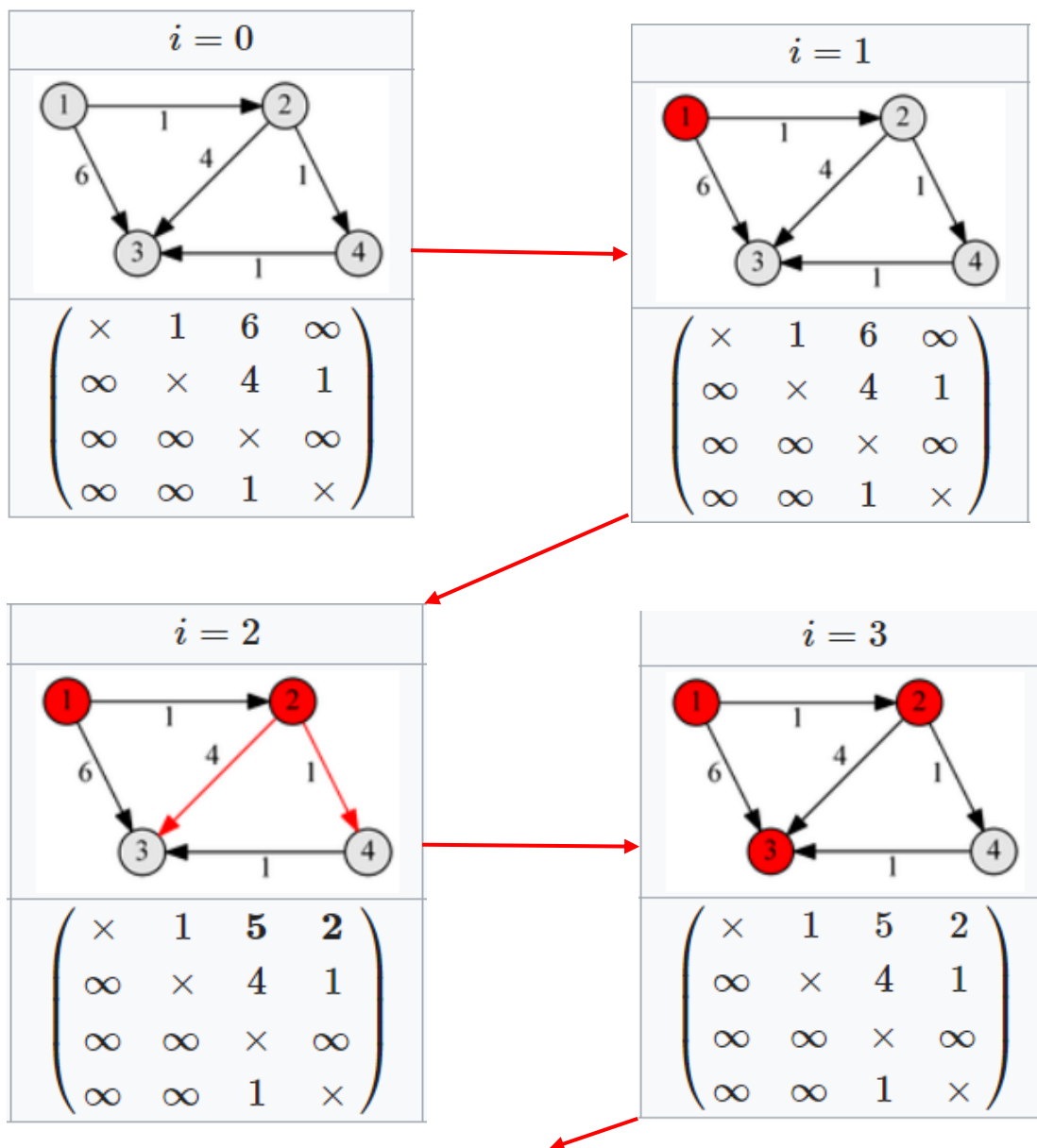
То есть, если кратчайший путь из u в v , содержащий только вершины из множества $\{1..i\}$, проходит через вершину i , то кратчайшим путем из u в v является кратчайший путь из u в i , объединенный с кратчайшим путем из i в v . В противном случае, когда этот путь не содержит вершины i , кратчайший путь из u в v , содержащий только вершины из множества $\{1..i\}$ является кратчайшим путем из u в v , содержащим только вершины из множества $\{1..i-1\}$

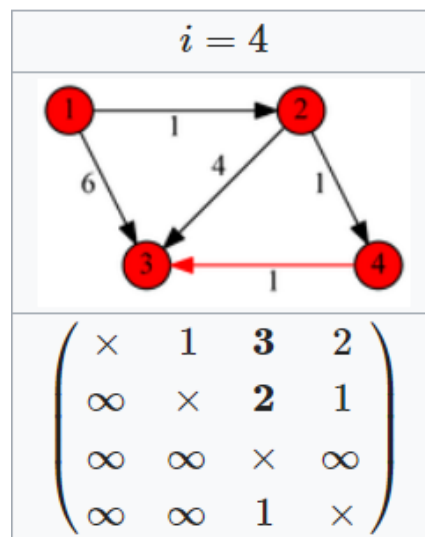
```
 $d_{uv}^{(0)} = w$ 
for  $i \in V$ 
  for  $u \in V$ 
    for  $v \in V$ 
       $d_{uv}^{(i)} = \min(d_{uv}^{(i-1)}, d_{ui}^{(i-1)} + d_{iv}^{(i-1)})$ 
```

В итоге получаем, что матрица $d^{(n)}$ и является искомой матрицей кратчайших путей, поскольку содержит в себе длины кратчайших путей между всеми парами вершин, имеющих в качестве промежуточных вершин

вершины из множества $\{1..n\}$, что есть попросту все вершины графа. Такая реализация работает за $\Theta(n^3)$ времени и использует $\Theta(n^3)$ памяти.

Пример работы алгоритма:





3.Описание алгоритма программы

При заполнении матрицы (ручном или автоматическом) используется динамический двумерный массив. Если между вершинами графа отсутствует прямой путь, то в массив записывается значение =10000.

После того как матрица выводится на экран начинает работать алгоритм Флойда-Уоршелла.

С помощью вложенных циклов производится поиск кратчайших путей. Если соблюдается условие $mas[i][j] > mas[i][k] + mas[k][j]$, то в $mas[i][j]$ записывается сумма $mas[i][k] + mas[k][j]$.

После вышеперечисленных манипуляций измененная матрица выводится на экран и пользователю предлагается сохранить результат в файл формата .txt.

Ниже представлен псевдокод для данного алгоритма :

для $k=0$ пока $k < N$ $k=k+1$

 для $i=0$ пока $i < N$ $i=i+1$

 для $j=0$ пока $j < N$ $j=j+1$

 если $(mas[i][j] \text{ больше } mas[i][k] + mas[k][j])$

 то $mas[i][j] = mas[i][k] + mas[k][j]$

 конец условия

4.Описание программы

Для написания данной программы использован язык программирования Си. Язык программирования Си - универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

Проект был создан в виде консольного приложения Win32 (Visual C++).

При запуске программы пользователю предоставляется возможность выбрать один из двух видов заполнения матрицы: ручного или автоматического.

```
cout << "1-Автоматическое заполнение матрицы смежности" << endl;  
cout << "2-Ручной ввод матрицы смежности" << endl;
```

Рисунок 1-меню программы.

Далее, при выборе варианта “Автоматическое заполнение матрицы смежности” пользователь должен ввести размер заполняемой матрицы ($N \times N$).

Далее происходит инициализация динамического массива и его заполнение случайными числами. (Если связи между вершинами графа нет, то пишется число 10000)

Вывод матрицы смежности

	1	2	3	4	5
1	0	40	10000	47	59
2	40	0	97	10000	10000
3	10000	97	0	10000	78
4	47	10000	10000	0	10000
5	59	10000	78	10000	0

Рисунок 2-заполнение массива случайными числами

```
mas = (int**)malloc(N * sizeof(int*));  
for (int i = 0; i < N; i++)  
{  
    mas[i] = (int*)malloc(N * sizeof(int));  
}  
for (int i = 0; i < N; i++)  
{  
    mas[i][i] = 0;  
    for (int j = i + 1; j < N; j++)  
    {  
        mas[i][j] = 1 + rand() % 201;  
        mas[j][i] = mas[i][j];  
        if (mas[i][j] > 100)  
        {  
            mas[i][j] = 10000;  
            mas[j][i] = mas[i][j];  
        }  
    }  
}
```

Рисунок 3-код автоматического заполнения массива

После заполнения производится вывод матрицы в консоль.

Затем происходит непосредственная реализация алгоритма Флойда-Уоршелла для поиска кратчайших путей между вершинами взвешенного графа.

```
for (int k = 0; k < N; k++)
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            if (mas[i][j] > mas[i][k] + mas[k][j])
                mas[i][j] = mas[i][k] + mas[k][j];
        }
    }
}
```

Рисунок 4-код алгоритма Флойда-Уоршелла

По завершении работы данного алгоритма результат выводится на экран и пользователю предлагается сохранить полученный результат в файл формата txt.

	1	2	3	4	5
1	0	40	137	47	59
2	40	0	97	87	99
3	137	97	0	184	78
4	47	87	184	0	106
5	59	99	78	106	0

Рисунок 5-результат работы алгоритм

```

cout << "Записать результат в файл файл" << endl;
cout << "1-да" << endl;
cout << "2-нет" << endl;
cin >> quest;
if (quest == 1)
{
    _file.open("файл.txt", ios_base::app);
    if (_file.is_open())
    {
        for (int i = 0; i < N; i++)
        {
            _file << i + 1 << "\t";
        }
        _file << endl;
        _file << endl;
        for (int i = 0; i < N; i++)
        {
            _file << i + 1 << "\t";
            for (int j = 0; j < N; j++)
            {
                _file << mas[i][j] << "\t";
            }
            _file << endl;
            _file << endl;
        }
        _file << "-----";
    }
    else
    {
        cout << "не удалось записать данные в файл!!";
    }
    _file.close();
}

```

Рисунок 6-код для записи результата в файл

Для записи результата работы программы в файл следует ввести '1'.

Для просмотра содержимого файла нужно зайти в папку с проектом

Debug	25.12.2022 22:56	Папка с файлами	
enc_temp_folder	26.12.2022 1:38	Папка с файлами	
ipch	26.12.2022 1:29	Папка с файлами	
Курсовая	25.12.2022 22:57	Папка с файлами	
Курсовая	26.12.2022 1:30	SQL Server Compa...	8 468 КБ
Курсовая	25.12.2022 22:43	Microsoft Visual St...	1 КБ

Рисунок 7 -путь в папку с сохраненным файлом

Далее в папку "Курсовая"

Debug	26.12.2022 1:29	Папка с файлами	
ReadMe	25.12.2022 22:43	Текстовый докум...	3 КБ
stdafx	25.12.2022 22:43	C++ Source	1 КБ
stdafx	25.12.2022 22:43	C/C++ Header	1 КБ
targetver	25.12.2022 22:43	C/C++ Header	1 КБ
Курсовая	25.12.2022 23:20	C++ Source	5 КБ
Курсовая	25.12.2022 22:43	VC++ Project	5 КБ
Курсовая.vcxproj	25.12.2022 22:43	VC++ Project Filte...	2 КБ
Курсовая.vcxproj.user	25.12.2022 22:43	VisualStudio.user.9...	1 КБ

Рисунок 8- путь в папку с сохраненным файлом

Открыть файл “Файлик.txt”

*Файлик – Блокнот

—

□

×

Файл

Правка

Формат

Вид

Справка

1

2

3

4

5

1

0

40

137

47

59

2

40

0

97

87

99

3

137

97

0

184

78

4

47

87

184

0

106

5

59

99

78

106

0

<

Стр 1, столб 2

100%

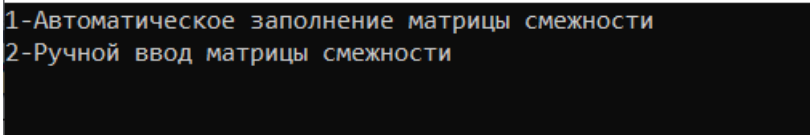
Windows (CRLF)

UTF-8

>

Рисунок 9-файл с результатом работы программы

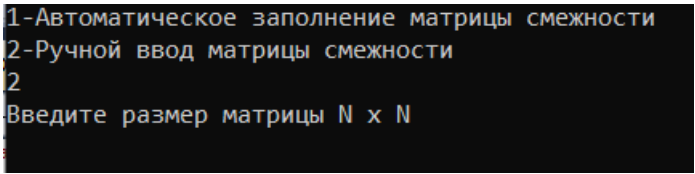
При выборе в меню цифры “2-Ручной ввод матрицы смежности”



```
1-Автоматическое заполнение матрицы смежности
2-Ручной ввод матрицы смежности
```

Рисунок 10-меню программы

Пользователь также должен ввести данные о размере массива($N \times N$) как и в случае автоматического заполнения :



```
1-Автоматическое заполнение матрицы смежности
2-Ручной ввод матрицы смежности
2
Введите размер матрицы N x N
```

Рисунок 11-установка размера матрицы при ручном вводе

Для наглядности составим небольшой граф вручную в тетради и перенесем данные в программу

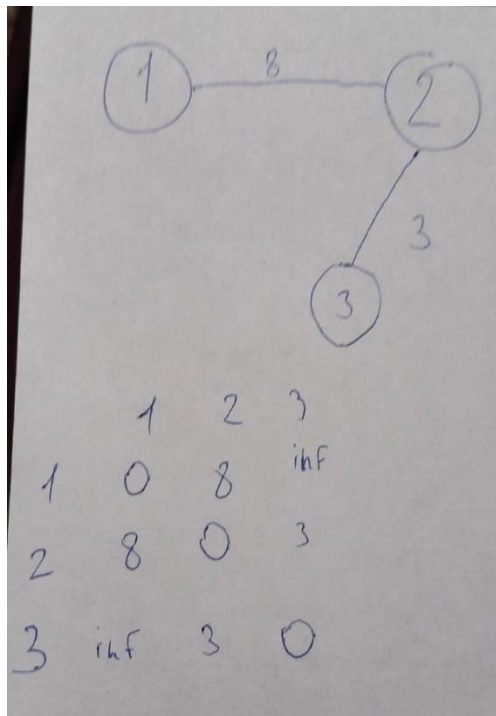


Рисунок 12-ручное заполнение графа на бумаге

```
Заполните массив:  
Если вершины не связаны следует вписать значение больше 100!!  
элемент [0][0]=0  
элемент [0][1]=8  
элемент [0][2]=10000  
элемент [1][0]=8  
элемент [1][1]=0  
элемент [1][2]=3  
элемент [2][0]=10000  
элемент [2][1]=3  
элемент [2][2]=0
```

Рисунок 13-ручное заполнение графа в программе

Программа вывела сначала необработанный массив, затем измененный массив

	1	2	3
1	0	8	10000
2	8	0	3
3	10000	3	0

	1	2	3
1	0	8	11
2	8	0	3
3	11	3	0

Рисунок 14-вывод матриц на экран

Пользователь, при ручном заполнении, также может сохранить в файл полученную в результате работы программы матрицу

```
Записать результат в файл Файлик?
1-да
2-нет
```

Рисунок 15-меню сохранения в файл

	1	2	3
1	0	8	11
2	8	0	3
3	11	3	0

Рисунок 16-файл с сохраненным результатом

Запись производится в конец файла.

5.Тестирование программы

Среда разработки Microsoft Visual Studio 2010 предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

Ниже продемонстрирован результат тестирования программы:

```
1-Автоматическое заполнение матрицы смежности
2-Ручной ввод матрицы смежности
1
Введите размер матрицы N x N
5
Вывод матрицы смежности
      1      2      3      4      5
1      0      67      54      10000      53
2      67      0      10000      69      10000
3      54      10000      0      10000      48
4      10000      69      10000      0      10000
5      53      10000      48      10000      0

      1      2      3      4      5
1      0      67      54      136      53
2      67      0      121      69      120
3      54      121      0      190      48
4      136      69      190      0      189
5      53      120      48      189      0

Записать результат в файл Файлик ?
1-да
2-нет
```

Рисунок 17-тестирование программы

Тестирование автоматического заполнения при N=5

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	10000	9	10000	10000	10000	10000	10000	20	29	10000	46	10000	59
2	10000	0	68	10000	42	10000	54	98	57	61	89	10000	10000	10000
3	9	68	0	15	10000	10000	10000	10000	10000	100	46	10000	45	10000
4	10000	10000	15	0	80	10	46	48	48	10000	2	10000	20	10000
5	10000	42	10000	80	0	10000	10000	10000	10000	10000	16	10000	65	10000
6	10000	10000	10000	10	10000	0	55	10000	29	89	10000	66	89	10000
7	10000	54	10000	46	10000	55	0	10000	10000	47	79	10000	28	10000
8	10000	98	10000	48	10000	10000	10000	0	11	33	17	10000	19	49
9	20	57	10000	48	10000	29	10000	11	0	10000	10000	74	90	24
10	29	61	100	10000	10000	89	47	33	10000	0	10000	10000	61	10000
11	10000	89	46	2	16	10000	79	17	10000	10000	0	10000	90	45
12	46	10000	10000	10000	10000	66	10000	10000	74	10000	10000	0	10000	41
13	10000	10000	45	20	65	89	28	19	90	61	90	10000	0	36
14	59	10000	10000	10000	10000	10000	10000	49	24	10000	45	41	36	0

Рисунок 18-тестирование программы

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	77	9	24	42	34	70	31	20	29	26	46	44	44
2	77	0	68	60	42	70	54	68	57	61	58	122	80	81
3	9	68	0	15	33	25	61	34	29	38	17	55	35	53
4	24	60	15	0	18	10	46	19	30	52	2	70	20	47
5	42	42	33	18	0	28	64	33	44	66	16	88	38	61
6	34	70	25	10	28	0	55	29	29	62	12	66	30	53
7	70	54	61	46	64	55	0	47	58	47	48	105	28	64
8	31	68	34	19	33	29	47	0	11	33	17	76	19	35
9	20	57	29	30	44	29	58	11	0	44	28	65	30	24
10	29	61	38	52	66	62	47	33	44	0	50	75	52	68
11	26	58	17	2	16	12	48	17	28	50	0	72	22	45
12	46	122	55	70	88	66	105	76	65	75	72	0	77	41
13	44	80	35	20	38	30	28	19	30	52	22	77	0	36
14	44	81	53	47	61	53	64	35	24	68	45	41	36	0

Рисунок 19 -тестирование программы

Тестирование программы при N=14

Произвели тестирование ручного заполнения программы

```
C:\Users\user\documents\visual studio 2010\Projects\Курсовая\Debug\Курсовая.exe

Заполните массив:
Если вершины не связаны следует вписать значение больше 100!!
элемент [0][0]=0
элемент [0][1]=8
элемент [0][2]=10000
элемент [1][0]=8
элемент [1][1]=0
элемент [1][2]=3
элемент [2][0]=10000
элемент [2][1]=3
элемент [2][2]=0

      1      2      3
1      0      8     10000
2      8      0      3
3     10000    3      0

      1      2      3
1      0      8     11
2      8      0      3
3     11      3      0

Записать результат в файл Файлик?
1-да
2-нет
```

Рисунок 20-тестирование программы

Для удобства заполнения $N=3$. При любом способе ввода (ручном или автоматическом) программа работает корректно.

6. Ручной расчет задачи

Для удобства ручного расчета, возьмем граф с количеством вершин равным 5.

С помощью программы заполним матрицу случайными числами

Вывод матрицы смежности

	1	2	3	4	5
1	0	94	10000	10000	76
2	94	0	32	10000	18
3	10000	32	0	37	10000
4	10000	10000	37	0	10000
5	76	18	10000	10000	0

Рисунок 21-заполнение матрицы

Для наглядности нарисовали граф в тетради. для несоединенных вершин написали значение inf (infinite)

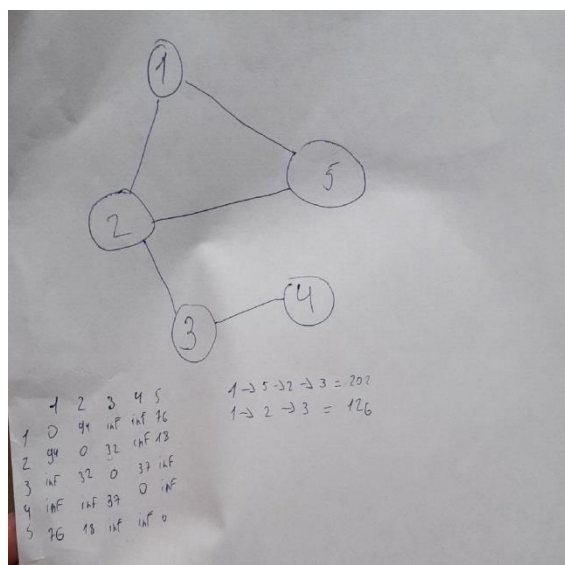


Рисунок 22-представление графа на бумаге

Отсюда видно, что вершины 1 и 3 не соединены между собой напрямую. Путем перебора различных вариантов самым коротким оказался следующий путь:

-Из вершины 1 в вершину 2

-Из вершины 2 в вершину 3

Полученный путь равен $=126$

Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Флойда-Уоршелла в Microsoft Visual Studio 2010.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежностей, основанных на теории орграфов. Приобретены навыки по осуществлению алгоритма поиска кратчайших путей. Углублены знания языка программирования Си.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

Список литературы

- Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ — 2-е изд — М.: Издательский дом «Вильямс», 2009. — ISBN 978-5-8459-0857-5.
- Романовский И. В. Дискретный анализ: Учебное пособие для студентов, специализирующихся по прикладной математике и информатике. Изд. 3-е. — СПб.: Невский диалект, 2003. — 320 с. — ISBN 5-7940-0114-3.

Приложение А. Листинг

```
#include "StdAfx.h"

#include <iostream>

#include <locale.h>

#include <cstdlib>

#include <conio.h>

#include <iostream>

#include <algorithm>

#include <fstream>

#include <stdlib.h>

#include <time.h>

#include <ctime>

#include <fstream> // работа с файлами

#include <iomanip> // манипуляторы ввода/вывода

//Максимальное значение веса = 100

#define INF 101

using namespace std;

typedef struct Node {

int vertex;

struct Node* next;

} Node;

int main() {

setlocale(LC_ALL, "RUS");

srand(time(NULL));

fstream _file;

int** mas, N; // N-количество вершин

int menu, quest;

cout << "1-Автоматическое заполнение матрицы смежности" << endl;

cout << "2-Ручной ввод матрицы смежности" << endl;

cin >> menu;

if (menu == 1) //Автоматическое заполнение
```

```

{
cout << "Введите размер матрицы N x N" << endl;
cin >> N;
Node** head = (Node * *)malloc(N * sizeof(Node*));
for (int i = 0; i < N; i++) {
head[i] = NULL;
}
bool* visited = (bool*)malloc(N * sizeof(bool));
for (int i = 0; i < N; i++)
{
visited[i] = false;
}
mas = (int**)malloc(N * sizeof(int*));
for (int i = 0; i < N; i++)
{
mas[i] = (int*)malloc(N * sizeof(int));
}
for (int i = 0; i < N; i++)
{
mas[i][i] = 0;
for (int j = i + 1; j < N; j++)
{
mas[i][j] = 1 + rand() % 201;
mas[j][i] = mas[i][j];
if (mas[i][j] > 100)
{
mas[i][j] = 10000;
mas[j][i] = mas[i][j];
}
}
}
printf("\nВывод матрицы смежности\n ");

```

```

printf("\n ");
printf("\t");
for (int i = 0; i < N; i++) {
printf("%d\t", i + 1);
}
cout << endl;

for (int i = 0; i < N; i++) { // вывод матриц смежности
printf("\n");
printf("%d\t", i + 1);
for (int j = 0; j < N; j++) {
printf("%d\t", mas[i][j]);
}
}
cout << endl;
cout << endl;
for (int k = 0; k < N; k++)
{
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
if (mas[i][j] > mas[i][k] + mas[k][j])
mas[i][j] = mas[i][k] + mas[k][j];
}
}
}
cout << "\t";
for (int i = 0; i < N; i++)
{
cout << i+1 << "\t";
};
cout << endl;

```

```

cout << endl;

for (int i = 0; i < N; i++)
{
    cout << i+1 << "\t";
    for (int j = 0; j < N; j++)
    {
        cout << mas[i][j] << "\t";
    }
    cout << endl;
}

cout << endl;

cout << "Записать результат в файл Файлик ?" << endl;

cout << "1-да" << endl;

cout << "2-нет" << endl;

cin >> quest;

if (quest == 1)
{
    _file.open("Файлик.txt", ios_base::app);
    if (_file.is_open())
    {
        for (int i = 0; i < N; i++)
        {
            _file << i + 1 << "\t";
        };
        _file << endl;
        _file << endl;
        for (int i = 0; i < N; i++)
        {
            _file << i + 1 << "\t";
            for (int j = 0; j < N; j++)
            {
                _file << mas[i][j] << "\t";
            }
        }
    }
}

```

```

}
_file << endl;
_file << endl;
}
_file << "-----";
}
else
{
cout << "не удалось записать данные в файл!!";
}
_file.close();
}
}
if (menu == 2) // Ручное заполнение массива
{
cout << "Введите размер матрицы N x N" << endl;
cin >> N;
cout << endl;
cout << endl;
mas = (int**)malloc(N * sizeof(int*));
for (int i = 0; i < N; i++)
{
mas[i] = (int*)malloc(N * sizeof(int));
}
cout << "Заполните массив:" << endl;
cout << "Если вершины не связаны следует вписать значение больше 100!!";
cout << endl;
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
cout << "элемент [" << i << "][" << j << "]=";

```

```

cin >> mas[i][j];
}
}
cout << endl;
cout << "\t";
for (int i = 0; i < N; i++)
{
cout << i + 1 << "\t";
}
cout << endl;
cout << endl;
for (int i = 0; i < N; i++)
{
cout << i + 1 << "\t";
for (int j = 0; j < N; j++)
{
cout << mas[i][j]<<"\t";
}
cout << endl;
}
for (int k = 0; k < N; k++)
{
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
if (mas[i][j] > mas[i][k] + mas[k][j])
mas[i][j] = mas[i][k] + mas[k][j];
}
}
}
cout << endl;

```

```

cout << endl;

cout << "\t";

for (int i = 0; i < N; i++)
{
    cout << i + 1 << "\t";

};

cout << endl;

cout << endl;

for (int i = 0; i < N; i++)
{
    cout << i + 1 << "\t";

    for (int j = 0; j < N; j++)
    {
        cout << mas[i][j] << "\t";
    }

    cout << endl;
}

cout << endl;

cout << "Записать результат в файл Файлик?" << endl;

cout << "1-да" << endl;

cout << "2-нет" << endl;

cin >> quest;

if (quest == 1)
{
    _file.open("Файлик", ios_base::app);

    if (_file.is_open())
    {
        for (int i = 0; i < N; i++)
        {
            _file << i + 1 << "\t";

        };

        _file << endl;
    }
}

```

```

_file << endl;
for (int i = 0; i < N; i++)
{
_file << i + 1 << "\t";
for (int j = 0; j < N; j++)
_file << mas[i][j] << "\t";
}
_file << endl;
_file << endl;
}
_file << "-----";
}
else
{
cout << "не удалось записать данные в файл!!";
}
_file.close();
}
}

```