



**Министерство науки и высшего образования Российской Федерации**  
**Федеральное государственное бюджетное образовательное учреждение**  
**высшего образования**  
**«Московский государственный технический университет**  
**имени Н.Э. Баумана**  
**(национальный исследовательский университет)»**  
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

## **Отчёт по лабораторной работе № 7** **«Графы»**

Студент Шелия София Малхазовна

Группа ИУ7 – 35Б

2020 г.

**Цель работы** – реализовать алгоритмы обработки графовых структур: поиск различных путей, проверку связности, построение остовых деревьев минимальной стоимости.

## 1. Описание условия задачи.

### **Вариант №12**

Задана система двусторонних дорог. Найти множество городов, расстояние от которых до выделенного города (столицы) больше, чем Т.

## 2. Техническое задание

### **Исходные данные и результат**

#### **Ввод:**

1. Пользователь вводит цифру 1 или 2 (1 – чтение данных из файла, 2 – ручное заполнение графа)  
При ручном вводе также вводится количество вершин и матрица смежности
2. Пользователь вводит цифру
  - 1 - Оценить эффективность выполнения задачи
  - 2 - Выполнить задачу с помощью матрицы и алгоритма Дейкстры
  - 3 - Выполнить задачу с помощью матрицы и алгоритма Беллмана
  - 4 - Выполнить задачу с помощью списка смежности и алгоритма Дейкстры
  - 5 - Выполнить задачу с помощью списка смежности и алгоритма Беллмана
  - 6 - Вывести текущий граф

Для пунктов 2 – 4 также вводится значение вершины столицы и значение Т (расстояние больше которого должен быть путь из столицы в какой-то город)

#### **Вывод: (для пунктов 1 - 6)**

1. Выводится сравнительная таблица времени выполнения задачи для разных вариантов хранения и обработки графов.
2. Выводятся все расстояния от столицы до остальных городов. Города, удовлетворяющие условие – отмечены.
3. Выводятся все расстояния от столицы до остальных городов. Города, удовлетворяющие условие – отмечены.
4. Выводятся все расстояния от столицы до остальных городов. Города, удовлетворяющие условие – отмечены.
5. Выводятся все расстояния от столицы до остальных городов. Города, удовлетворяющие условие – отмечены.
6. Происходит вывод графа 3 разными способами

### **Описание задачи, реализуемой программой**

1. Программа считывает граф в матрицу и массив, элементами которого являются связные списки.
2. Программа вычисляет минимальные расстояния от введенной пользователем столицы до остальных городов, а также отмечает расстояния, который больше заданного T.
3. Программа сравнивает время выполнения задачи для разных вариантов хранения и обработки графов.

### **Способ обращения к программе**

Запуск приложения возможен через терминал MSYS2, а именно.

1. `gcc -std=c99 -Wall -Werror -c *.c`
2. `gcc -o main.exe *.o`
3. `./main.exe`

## **Описание возможных аварийных ситуаций и ошибок пользователя**

### **Ошибки пользователя при выборе пункта меню**

- Пустой ввод
- Вводятся посторонние символы (например, буквы)
- Вводится отрицательное число
- Вводится не целое число
- Вводится отсутствующий пункт меню

### **Ошибки пользователя при вводе значения столицы**

- Пустой ввод
- Вводятся посторонние символы (например, буквы)
- Вводится не целое число
- Вводится отрицательное число
- Вводится отсутствующая вершина

### **Ошибки пользователя при вводе количества вершин графа, элементов матрицы смежности и значения T**

- Пустой ввод
- Вводятся посторонние символы (например, буквы)
- Вводится не целое число
- Вводится отрицательное число

## **3. Описание внутренних структур данных**

Для хранения графа я использовала динамический массив указателей на элементы следующего структурного типа

```
typedef struct list_graph
{
    int value;
    int path_length;
    struct list_graph *next;
```

```
}list_graph;
```

Второй вариант хранения – динамическая матрица.

## **4. Алгоритм**

### **Алгоритм Дейкстры**

1. Создаем массив размера size (в нем будут храниться кратчайшие расстояния от заданной вершины)
2. Всем элементам этого массива присваиваем значение INF, кроме того, который является столицей (то есть городом, от которого мы ищем расстояния), ему присваиваем значение 0
3. Также создаем массив размера size, все элементы которого – 1. С помощью этого массива определяем была ли посещена вершина.
4. Далее начинается цикл, который закончится, когда все вершины будут посещены
  - Ищем не посещенную вершину p с минимальным значением
  - В цикле проверяем значения вершин, с которыми связана p. Если сумма пути до p – ой вершины и веса ребра до заданной вершины меньше его текущего значения, то мы заменяем его текущее значение на данную сумму.
  - Отмечаем вершину как пройденную.

### **Беллмана-Форда**

1. Создаем массив размера size (в нем будут храниться кратчайшие расстояния от заданной вершины)
2. Всем элементам этого массива присваиваем значение INF, кроме того, который является столицей (то есть городом, от которого мы ищем расстояния), ему присваиваем значение 0
3. Начинается цикл – на каждой его итерации просматриваются все ребра графа.
  - На первой итерации ищется путь от столицы ко всем вершинам
  - Далее – кратчайший путь, с помощью попытки прохождения через i вершину

## **5. Тесты**

### **Негативные тесты**

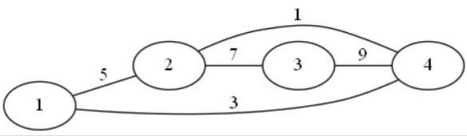
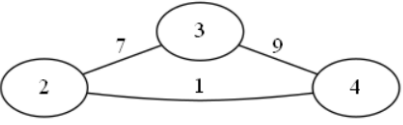
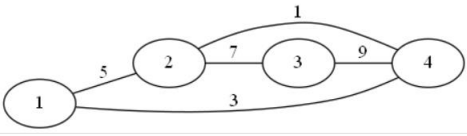
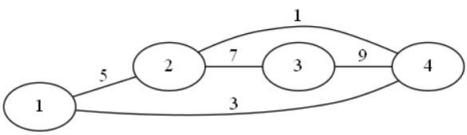
Входные данные	Результат	Условие проверки
<b>Для выбора пункта меню</b>		
Выберите действие: 1 - Оценить эффективность выполнения задачи 2 - Выполнить задачу с помощью матрицы и алгоритма Дейкстры 3 - Выполнить задачу с помощью матрицы и алгоритма Беллмана	Некорректный ввод.	Пустой ввод

<p>4 - Выполнить задачу с помощью списка смежности и алгоритма Дейкстры</p> <p>5 - Выполнить задачу с помощью списка смежности и алгоритма Беллмана</p> <p>6 - Вывести текущий граф</p> <p>Выбор:</p>		
<p>Выберите действие:</p> <p>1 - Оценить эффективность выполнения задачи</p> <p>2 - Выполнить задачу с помощью матрицы и алгоритма Дейкстры</p> <p>3 - Выполнить задачу с помощью матрицы и алгоритма Беллмана</p> <p>4 - Выполнить задачу с помощью списка смежности и алгоритма Дейкстры</p> <p>5 - Выполнить задачу с помощью списка смежности и алгоритма Беллмана</p> <p>6 - Вывести текущий граф</p> <p>Выбор: ргшд</p>	Некорректный ввод.	Вводятся посторонние символы (например, буквы)
<p>Выберите действие:</p> <p>1 - Оценить эффективность выполнения задачи</p> <p>2 - Выполнить задачу с помощью матрицы и алгоритма Дейкстры</p> <p>3 - Выполнить задачу с помощью матрицы и алгоритма Беллмана</p> <p>4 - Выполнить задачу с помощью списка смежности и алгоритма Дейкстры</p> <p>5 - Выполнить задачу с помощью списка смежности и алгоритма Беллмана</p> <p>6 - Вывести текущий граф</p> <p>Выбор:-56</p>	Некорректный ввод.	Вводится отрицательное число
<p>Выберите действие:</p> <p>1 - Оценить эффективность выполнения задачи</p> <p>2 - Выполнить задачу с помощью матрицы и алгоритма Дейкстры</p> <p>3 - Выполнить задачу с помощью матрицы и алгоритма Беллмана</p> <p>4 - Выполнить задачу с помощью списка смежности и алгоритма Дейкстры</p> <p>5 - Выполнить задачу с помощью списка смежности и алгоритма Беллмана</p> <p>6 - Вывести текущий граф</p> <p>Выбор: 89</p>	Некорректный ввод.	Вводится отсутствующий пункт меню
<p>Выберите действие:</p> <p>1 - Оценить эффективность выполнения задачи</p> <p>2 - Выполнить задачу с помощью матрицы и алгоритма Дейкстры</p> <p>3 - Выполнить задачу с помощью матрицы и алгоритма Беллмана</p>	Некорректный ввод.	Вводится не целое число

4 - Выполнить задачу с помощью списка смежности и алгоритма Дейкстры 5 - Выполнить задачу с помощью списка смежности и алгоритма Беллмана 6 - Вывести текущий граф Выбор: 78.1		
<b>Для ввода значения элемента</b>		
Введите целое число - значение элемента для добавления/удаления:	Некорректный ввод. Введите целое число:	Пустой ввод
Введите целое число - значение элемента для добавления/удаления: 1ава	Некорректный ввод. Введите целое число:	Вводятся посторонние символы (например, буквы)
Введите целое число - значение элемента для добавления/удаления: 34.12	Некорректный ввод. Введите целое число:	Вводится не целое число
<b>Для ввода столицы</b>		
Введите вершину, которая будет являться столицей ( $\leq 10$ ):	Некорректный ввод. Введите целое число:	Пустой ввод
Введите вершину, которая будет являться столицей ( $\leq 10$ ): щыв	Некорректный ввод. Введите целое число:	Вводятся посторонние символы (например, буквы)
Введите вершину, которая будет являться столицей ( $\leq 10$ ): 45.1	Некорректный ввод. Введите целое число:	Вводится не целое число
Введите вершину, которая будет являться столицей ( $\leq 10$ ): 8390	Некорректный ввод. Введите целое число:	Вводится число больше заданного
Введите вершину, которая будет являться столицей ( $\leq 10$ ): -45	Некорректный ввод. Введите целое число:	Вводится отрицательное число
<b>Для ввода Т</b>		
Введите Т - расстояние больше которого должен быть путь до столицы (целое неотрицательное число $< 100$ ):	Некорректный ввод. Введите целое число:	Пустой ввод
Введите Т - расстояние больше которого должен быть путь до столицы (целое неотрицательное число $< 100$ ): io	Некорректный ввод. Введите целое число:	Вводятся посторонние символы (например, буквы)
Введите Т - расстояние больше которого должен быть путь до столицы (целое неотрицательное число $< 100$ ): 67.2	Некорректный ввод. Введите целое число:	Вводится не целое число
Введите Т - расстояние больше которого должен быть путь до столицы (целое неотрицательное число $< 100$ ): 456	Некорректный ввод. Введите целое число:	Вводится число больше заданного
Введите Т - расстояние больше которого должен быть путь до столицы (целое неотрицательное число $< 100$ ): -2	Некорректный ввод. Введите целое число:	Вводится отрицательное число

## Позитивные тесты

Входные данные	Результат	Условие проверки
----------------	-----------	------------------

 <p>Введите вершину, которая будет являться столицей(<math>\leq 4</math>): 1 Введите T - расстояние больше которого должен быть путь до столицы (целое неотрицательное число <math>&lt; 100</math>): 1</p>	<p>Столицей является вершина: 1, расстояние для сравнения: 1 Расстояние до города 2: 4 Расстояние до города 3: 11 Расстояние до города 4: 3</p>	<p>Обычный тест</p>
 <p>Введите вершину, которая будет являться столицей(<math>\leq 4</math>): 1 Введите T - расстояние больше которого должен быть путь до столицы (целое неотрицательное число <math>&lt; 100</math>): 1</p>	<p>Столицей является вершина: 1, расстояние для сравнения: 1 Расстояние до города 2: 0 (Не подходит) Расстояние до города 3: 0 (Не подходит) Расстояние до города 4: 0 (Не подходит)</p>	<p>Из столицы нет дорог</p>
 <p>Введите вершину, которая будет являться столицей(<math>\leq 4</math>): 3 Введите T - расстояние больше которого должен быть путь до столицы (целое неотрицательное число <math>&lt; 100</math>): 30</p>	<p>Столицей является вершина: 3, расстояние для сравнения: 30 Расстояние до города 1: 11 (Не подходит) Расстояние до города 2: 7 (Не подходит) Расстояние до города 4: 8 (Не подходит)</p>	<p>Все города не подходят</p>
 <p>Введите вершину, которая будет являться столицей(<math>\leq 4</math>): 1 Введите T - расстояние больше которого должен быть путь до столицы (целое неотрицательное число <math>&lt; 100</math>): 1</p>	<p>Столицей является вершина: 3, расстояние для сравнения: 0 Расстояние до города 1: 11 Расстояние до города 2: 7 Расстояние до города 4: 8</p>	<p>Все города подходят</p>

## 6. Сравнение эффективности

Количество итераций 1000000

Время

Дейкстра

Количество элементов	Количество ребер	Матрица	Список
----------------------	------------------	---------	--------

10	40	0.0000002740	0.0000002850
10	21	0.0000001670	0.0000002230
20	173	0.0000002600	0.0000001470
20	93	0.0000001530	0.0000003090
30	384	0.0000001910	0.0000001550
30	201	0.0000002520	0.0000001680
40	693	0.0000001600	0.0000002600
40	355	0.0000001680	0.0000001340

### Беллмана-Форда

Количество элементов	Количество ребер	Матрица	Список
10	40	0.0000026000	0.0000027000
10	21	0.0000022000	0.0000013000
20	173	0.0000220000	0.0000256000
20	93	0.0000200000	0.0000131000
30	384	0.0000725000	0.0000837000
30	201	0.0000631000	0.0000414000
40	693	0.0001747000	0.0002280000
40	355	0.0001540000	0.0001026000

### Занимаемая память

Количество элементов	Количество ребер	Матрица	Список
10	40	400	1280
10	21	400	672
20	173	1600	5536
20	93	1600	2976
30	384	3600	12288
30	201	3600	6432
40	693	6400	22176
40	355	6400	11360

### Выводы по проделанной работе

Как видно из результатов, алгоритм Дейкстры оказался значительно быстрее Беллмана, но у Беллмана есть другие преимущества – в отличие от Дейкстры он также работает с графами с отрицательными ребрами.

Также стоит отметить, что Беллман работает быстрее если граф “разрежен”, то есть содержит небольшое кол-во ребер



Что касается памяти, то тут абсолютным лидером стала динамическая матрица, которой требуется существенно меньше памяти, чем списку.

Также стоит отметить, что особой разницы по времени при обработке матрицы или списка не отметились.

## **Ответы на контрольные вопросы**

### **1. Что такое граф?**

Граф – конечное множество вершин и ребер, которые их соединяют  $G = (V, E)$

$V$  – конечное непустое множество вершин

$E$  – множество ребер

### **2. Как представляются графы в памяти?**

Существует два основных способа: матрица смежности и список смежности.

Матрица смежности: элемент матрицы  $a[i][j] = 1$ , если ребро между  $i$  и  $j$  вершиной существует, иначе – 0.

Список смежности: массив, элементами которого являются связные списки. В данных связных списках представлена информация о вершинах, в которые есть доступ у  $i$  вершины.

### **3. Какие операции возможны над графами?**

- Поиск кратчайшего вершины от определенной вершины ко всем
- Поиск кратчайшего пути от одной вершины к другой
- Поиск кратчайшего пути между всеми вершинами
- Поиск эйлера и гамильтонова пути

### **4. Какие способы обхода графов существуют?**

- Обход в глубину
- Обход в ширину

### **5. Где используются грифовые структуры?**

Графы могут использоваться в ГИС, логистике, социальных сетях, магазинах, метро и т.д. То есть в любых сферах, где необходимо установить произвольные связи.

### **6. Какие пути в графе Вы знаете?**

Эйлеров, гамильтонов

### **7. Что такое каркасы графа?**

Каркас – остоновый ациклический подграф, его области связности совпадают с областями связности исходного графа.