



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»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8 ПО ДИСЦИПЛИНЕ: ТИПЫ И СТРУКТУРЫ ДАННЫХ

Графы

Студент **Попов Ю.А.**

Группа **ИУ7-32Б**

Вариант 4

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Попов Ю.А.**

Преподаватель _____ **Барышникова М.Ю.**

Оглавление

Оглавление	2
Описание условия задачи	3
Описание технического задания	3
Входные данные:	3
Выходные данные:	3
Действие программы:	3
Обращение к программе:	3
Описание структуры данных	4
Описание алгоритма	5
Описание основных функций	5
Набор тестов	6
Оценка эффективности	8
Объем занимаемой памяти (байты)	8
Замер времени выполнения (нс)	9
Ответы на контрольные вопросы	10
1. Что такое граф?	10
2. Как представляются графы в памяти?	10
3. Какие операции возможны над графами?	10
4. Какие способы обхода графов существуют?	10
5. Где используются графовые структуры?	10
6. Какие пути в графе Вы знаете?	10
7. Что такое каркасы графа?	10
Вывод	11

Описание условия задачи

Найти все вершины графа, к которым от заданной вершины можно добраться по пути не длиннее A .

Описание технического задания

Входные данные:

- Файл, в котором содержится граф в формате матрицы смежностей;
- число, стартовая вершина;
- число, максимальная длина пути;

Выходные данные:

- Вершины, расстояние до которых от стартовой вершины меньше указанного значения;
- изображение дерева;

Действие программы:

Программа работает линейно. Сначала пользователь должен ввести путь к файлу, после этого программа запрашивает 2 числа — номер стартовой вершины, и максимальное расстояние. Программа выводит вершины, до которых можно добраться по пути меньше максимального расстояния. В конце, программа визуализирует граф и сохранит изображение.

Обращение к программе:

Программа запускается через терминал, в командной строке, с помощью команды `./app.exe`.

Описание структуры данных

Структура, описывающая граф представлена на листинге 1

count_vertices - Количество вершин

matrix - Матрица смежностей

```
struct _graph_type_  
{  
    size_t count_vertices; // Количество вершин  
    int **matrix;           // Матрица смежности  
};
```

листинг 1. Структура описания графа.

Матрица смежности - структура данных, хранящая длину пути из одной вершины в другую. $b[i,j] \in \mathbb{Z}$, если ребро, связывающее вершины V_i и V_j существует и $b[i,j]=0$, если ребра не существует. Визуализации матрицы смежности для графа с рисунка 1 представлена на рисунке 2

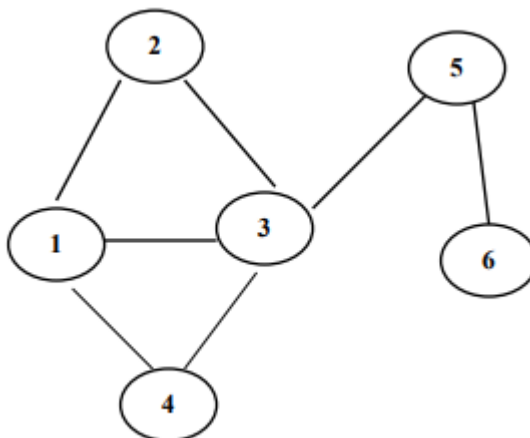


Рисунок 1. Неориентированный граф

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	1	0	0	0
3	1	1	0	1	1	0
4	1	0	1	0	0	0
5	0	0	1	0	0	1
6	0	0	0	0	1	0

Рисунок 2. Матрица смежности для неориентированного графа

Описание алгоритма

1. Программа считывает граф из файла, который был введен пользователем
2. Пользователь вводит 2 числа — номер начальной вершины, и максимальное расстояние.
3. С помощью алгоритма Беллмана-Форда программа считает расстояние от начальной вершины до каждой вершины графа.
4. После этого определяется какие вершины лежат на расстоянии меньше максимального, и дерево выводится на экран.

Описание основных функций

```
graph_t *create_graph(size_t vert_count);
```

Функция для инициализации графа

Входные параметры: Количество вершин

Возвращаемый результат: Указатель на структурную переменную графа

```
void free_graph(graph_t *graph);
```

Освобождение всей памяти, выделенное под граф

Входные параметры: указатель на граф

```
int load_graph(graph_t **graph, char *filename);
```

Загрузка графа из файла

Входные параметры: указатель на граф

Входные параметры: название файла

Выходные параметры: граф, заполненный информацией из файла

Возвращаемый результат: код возврата

```
void show_graph(graph_t *graph, int const *distance);
```

Функция для вывода на экран графа

Входные параметры: указатель на граф

Входные параметры: массив расстояний до вершин

Выходные параметры: вывод графа на экран

```
int bellman_ford_alg(graph_t *graph, int start_vertex, int **output);
```

Реализация алгоритма Беллмана-Форда

Входные параметры: указатель на граф

Входные параметры: начальная вершина

Входные параметры: массив расстояние от начальной вершины до всех остальных

Выходные параметры: заполненный массив расстояний

Возвращаемый результат: код возврата.

```
void calc_memory_usage(graph_t *graph)
```

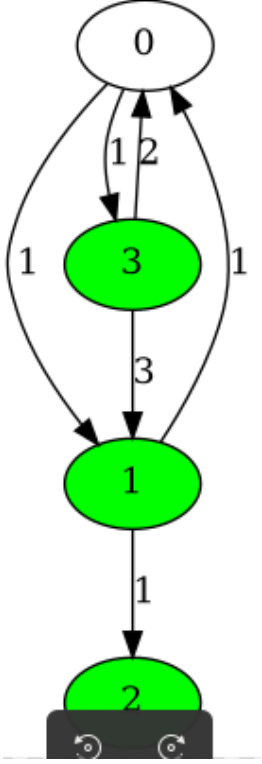
Подсчет использования памяти

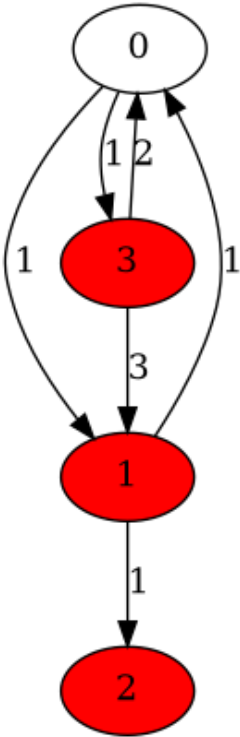
Входные параметры: указатель на граф

Выходные параметры: вывод количества занятой памяти

Набор тестов

№	Описание теста	Ввод	Вывод
1	Пустая строка вместо файла		Ошибка при вводе строки
2	Неверное имя файла		Ошибка чтения файла
3	Пустой файл		Ошибка при считывании матрицы смежности из файла
4	В файле символ вместо числа	a 1	Ошибка при считывании матрицы смежности из файла

5	Матрица смежности введена не полностью	$\begin{matrix} 2 \\ 1 & 1 \\ 1 \end{matrix}$	Ошибка при считывании матрицы смежности из файла
6	Граф пустой или состоит из 1 элемента	$\begin{matrix} 1 \\ 1 \end{matrix}$	Пустой граф
7	Неверный ввод начальной вершины или максимальной дистанции	-1	Ошибка ввода числа
8	Неверный ввод начальной вершины или максимальной дистанции	a	Ошибка ввода числа
9	Корректный ввод	0 2	

10	Ввод длины 0	0 0	
11	Граф с отрицательным весом		Граф содержит отрицательный цикл

Оценка эффективности

Объем занимаемой памяти (байты)

Объем, занимаемой памяти очередью в зависимости от их размера

Количество элементов	Память, занимаемая деревом (байт)
2	48
10	496
50	10416
100	40816
1000	400080016

Таблица 1. Оценка занимаемой памяти, относительно размера.

Можно рассчитать размер графа в памяти по формуле

```
sizeof(graph) + sizeof(graph→matrix) + sizeof(graph→matrix[0]) *  
graph→count_vertices * graph→count_vertices;  
// 16 + n * 8 + n^2 * 4, где n - это количество вершин
```

Замер времени выполнения (нс)

Временные замеры осуществлялись с помощью библиотеки time.h.

Ниже представлены таблица времени работы алгоритма Беллмана-Форда.

Количество вершин	Время выполнения (мкс)
2	0.341
3	0.541
4	0.831
5	1.133
10	5.330
25	62.086

Таблица 2. Время выполнения алгоритма Беллмана-Форда.

Сложность алгоритма Беллмана-Форда $O(V \cdot E)$, где V - число вершин, а E - число ребер. Он был выбран, потому что способен работать с отрицательными весами, определять отрицательные циклы. Кроме этого алгоритм подходит для работы с ориентированными и неориентированными графами.

Матричная форма записи была выбрана из-за простоты доступа к ребрам, упрощения реализации алгоритма, и эффективности для “плотных” графов.

Разработанную программу можно использовать в навигационных сервисах. Например для решения задачи, хватит ли топлива до заправки.

Ответы на контрольные вопросы

1. Что такое граф?

Граф – это конечное множество вершин и ребер, соединяющих их, т. е.: $G = \langle V, E \rangle$, где V – конечное непустое множество вершин; E – множество ребер (пар вершин).

2. Как представляются графы в памяти?

С помощью матрицы смежности или списков смежности

3. Какие операции возможны над графами?

Обход вершин (обход в глубину и обход в ширину), поиск различных путей, исключение и включение вершин.

4. Какие способы обхода графов существуют?

DFS - поиск в глубину (Depth First Search)

BFS - поиск в ширину (Breadth First Search)

5. Где используются графовые структуры?

Навигационные системы, социальные связи. Графы используют в задачах, где между элементами установлены произвольные связи.

6. Какие пути в графе Вы знаете?

Эйлеров путь, простой путь, сложный путь

7. Что такое каркасы графа?

Каркас графа - дерево, в которое входят все вершины графа и некоторые ребра.

Вывод

В ходе лабораторной работы была написана программа для поиска “достижимых путей”. В задаче использовался алгоритм Беллмана-Форда, который работает $O(VE)$. Хранить граф в списке смежности эффективно только при малом количестве вершин. Матрица смежности работает намного быстрее (в доступе), но занимает много бОльший объём, тк хранит V^2 элементов.