



**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**

**Вариант 4**

Студент **Ильченко Е. А.**

Группа **ИУ7-34Б**

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

Студент \_\_\_\_\_ **Ильченко Е. А.**

Преподаватель \_\_\_\_\_ **Силантьева А. В.**

**2024 г.**

# Описание условия задачи

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

## Описание ТЗ

### 1. Описание исходных данных и результатов работы программы

#### Входные данные:

Пользовательская команда из доступных и необходимые аргументы определенного сценария:

- 1: Ввести граф вручную
- 2: Ввести граф из файла
- 3: Вывести матрицу смежности графа
- 4: Вывести граф
- 5: Найти все вершины графа, к которым от заданной вершины можно добраться по пути не длиннее  $A$
- 6: Меню
- 0: Выход

Также программа принимает на вход текстовый файл, содержащий целочисленные значения, если пользователь вводит граф из файла. При ручном вводе программа принимает количество вершин в графе и для каждого ребра: первую вершину, вторую вершину, вес ребра. На основе этих данных создается граф в виде матрицы. При поиске всех вершин, к которым от заданной вершины можно добраться по пути не длиннее  $A$  программа принимает на вход начальную вершину и максимальное расстояние.

#### Выходные данные:

Граф, измененный в соответствии с выбранной операцией; матрица смежности графа; png и dot файл с графом; вершины графа, к которым можно добраться от заданной вершины по пути не длиннее  $A$ .

### 2. Описание задачи, реализуемой в программе

Реализовать алгоритмы обработки графовых структур: поиск различных путей, проверка связности, построение остовых деревьев минимальной стоимости.

### 3. Способ обращения к программе

Запуск исполняемого файла

```
./app.exe
```

Далее выбирается, какой пункт меню выполнить

### 4. Описание возможных аварийных ситуаций и ошибок пользователя

1. Неверный ввод пункта меню: “Неверная команда”
2. Ошибка ввода названия файла: “Ошибка ввода файла”
3. Ошибка открытия файла: “Ошибка открытия файла”
4. Ошибка ввода начальной вершины: “Ошибка ввода начальной вершины”
5. Ошибка ввода максимального расстояния: “Ошибка ввода максимального расстояния”
6. Ошибка ввода количества вершин графа: “Ошибка ввода”
7. Ошибка ввода данных ребра: “Ошибка: некорректные данные”
8. Ошибка чтения данных из файла: “Ошибка: некорректные данные в файле”

### 5. Описание внутренних структур данных

```
typedef int **graph_t;
```

Тип данных, представляющий граф в виде матрицы смежности.

Каждый элемент массива `graph_t[i][j]` хранит вес пути из города `i` в город `j`.

Если путь между городами отсутствует, то значение равно `INF`. Между городами с одинаковым индексом (`graph_t[i][i]`) значение пути `0`.

### 6. Описание функций

```
int **input_graph(int *n);
```

Ручной ввод графа

```
int **read_graph_from_file(FILE *file, int *n);
```

Чтение графа из файла

```
void print_adjacency_matrix(graph_t graph, int n);
```

Вывод матрицы смежности из графа

```
void save_to_png_from_graphviz(graph_t graph, int n);
```

Сохранение графа в png

```
void find_reachable_vertices(graph_t graph, int n, int start, int  
max_distance);
```

Поиск всех вершин графа, к котором от заданной вершины можно добраться по пути не менее A

```
void free_graph(graph_t graph, int n);
```

Освобождение памяти из под графа

## 7. Описание алгоритмов

Алгоритм поиска всех вершин графа, к котором от заданной вершины можно добраться по пути не менее A

Инициализация начальных данных:

1. Выделяется память для двух массивов:
  - a. `dist`: хранит минимальные расстояния от начальной вершины до каждой из остальных вершин, изначально заполнен значением `INF` (бесконечность), кроме начальной вершины, расстояние до которой равно 0.
  - b. `visited`: хранит флаги, показывающие, посещена ли вершина, изначально заполнен нулями (все вершины не посещены).

Поиск ближайшей непосещённой вершины:

1. Для каждой итерации выбирается вершина `u`, минимальное расстояние до которой меньше всех остальных среди непосещённых.
2. Если все непосещённые вершины недостижимы (расстояние равно `INF`), алгоритм завершает выполнение.

Обход соседей текущей вершины:

1. Все соседи вершины `u` (по данным графа `graph`) проверяются на достижимость:
  - a. Если расстояние до соседа `v` через `u` меньше, чем текущее записанное в `dist[v]`, то расстояние обновляется.
2. После проверки всех соседей вершина `u` отмечается как посещённая.

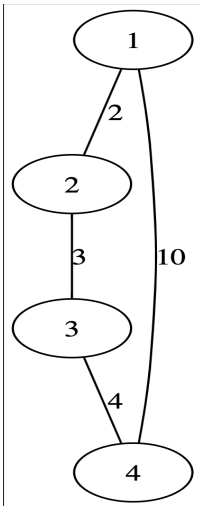
Вывод достижимых вершин:

1. Алгоритм проверяет каждую вершину графа.
2. Если расстояние до вершины не превышает `max_distance` и вершина не является стартовой, она считается достижимой.
3. Для каждой такой вершины выводится её номер и расстояние.

Завершение:

1. Освобождается память, выделенная для массивов `dist` и `visited`, чтобы избежать утечек.

# Тесты

Тест	Входные данные	Выходные данные																									
Ввод графа вручную	Количество вершин: 4 Рёбра графа: 1 2 5 2 3 3 3 4 2 4 1 1 -1 -1 -1	Успешно введенный граф																									
Ввод графа из файла	Имя файла: test_graph.txt	Успешно считанный граф																									
Поиск достижимых вершин	Количество вершин: 4 Рёбра графа: 1 2 2 2 3 3 3 4 4 1 4 10 -1 -1 -1 Начальная вершина: 1 Максимальное расстояние: 5	Вершины, достижимые от вершины 1 с длиной пути <= 5: Вершина 2 (расстояние: 2) Вершина 3 (расстояние: 5)																									
Вывести граф	Количество вершин: 4 Рёбра графа: 1 2 5 2 3 3 3 4 2 4 1 1 -1 -1 -1																										
Вывести матрицу смежности графа	Количество вершин: 4 Рёбра графа: 1 2 5 2 3 3 3 4 2 4 1 1 -1 -1 -1	<table><tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>1</td><td>0</td><td>2</td><td>INF</td><td>10</td></tr><tr><td>2</td><td>2</td><td>0</td><td>3</td><td>INF</td></tr><tr><td>3</td><td>INF</td><td>3</td><td>0</td><td>4</td></tr><tr><td>4</td><td>10</td><td>INF</td><td>4</td><td>0</td></tr></table>		1	2	3	4	1	0	2	INF	10	2	2	0	3	INF	3	INF	3	0	4	4	10	INF	4	0
	1	2	3	4																							
1	0	2	INF	10																							
2	2	0	3	INF																							
3	INF	3	0	4																							
4	10	INF	4	0																							

Ошибка ввода данных ребра	Количество вершин: 3 Рёбра графа: 1 5 10 1 -1 3 -1 -1 -1	Ошибка: некорректные данные
Ввод неверной команды	Команда: "а"	Неверная команда
Ошибка ввода количества вершин	Количество вершин: -1	Ошибка ввода

## 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?

- Список смежности: Для каждой вершины хранится список соседних вершин.
- Матрица смежности: Двумерный массив, где элемент  $[i][j]$  равен весу ребра между вершинами  $i$  и  $j$  или 1 – если связь есть, иначе – 0.
- Матрица инцидентности: двумерная матрица, где строки – вершины, а столбцы – рёбра, 1 – если вершина инцидентна ребру, иначе – 0.

### 3. Какие операции возможны над графами?

- Добавление/удаление вершин
- Добавление/удаление ребер
- Поиск кратчайшего пути
- Проверка связности графа
- Поиск пути между вершинами
- Поиск минимального остовного дерева
- Обход графа

### 4. Какие способы обхода графов существуют?

- Поиск в глубину DFS
- Поиск в ширину BFS

### 5. Где используются графовые структуры?

Графовые структуры данных широко используются в областях науки и техники, для моделирования сложных взаимосвязей и зависимостей

- Сетевые технологии
- Алгоритмы и оптимизация

- Социальные сети
- Логистика и транспорт
- Искусственный интеллект

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

- Простой путь – путь, в котором все вершины уникальны.
- Эйлеров путь – проходит через каждое ребро графа ровно один раз.
- Эйлеров цикл – Эйлеров путь, который начинается и заканчивается в одной вершине.
- Гамильтонов путь – проходит через каждую вершину ровно один раз.
- Гамильтонов цикл – Гамильтонов путь с возвращением в начальную вершину.
- Кратчайший путь – путь с минимальной суммарной стоимостью рёбер.

#### **7. Что такое каркасы графа?**

Каркас графа (минимальное остовное дерево) — это подграф, соединяющий все вершины исходного графа минимальным числом рёбер без образования циклов.