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

Студент **Паламарчук А.Н.**

Группа **ИУ7-33Б**

Предмет **Типы и структуры данных**

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

Студент _____ **Паламарчук А.Н.**

Преподаватель _____ **Никульшина Т. А.**

Преподаватель _____ **Барышникова М. Ю.**

2023 г.

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

Техническое задание
Найти самый длинный простой путь в графе.

Входные данные

- Команда (число) из меню программы (см. меню программы) – обозначение необходимой операции.
- Текстовый файл

Выходные данные

- Таблицы эффективности
- Файл в DOT формате
- Файл png
- Самый длинный простой путь в графе

Возможные аварийные ситуации

- Некорректный исходный файл
- Ошибка открытия файла
- Отсутствие исходного файла
- Некорректная команда

Описание внутренних структур данных

```
typedef struct graph
{
    int matrix[MAX_SIZE][MAX_SIZE];
    size_t size;
} graph_t;

typedef struct simple_path
{
    bool visited[MAX_SIZE];
    int path[MAX_SIZE];
    size_t size;
} simple_path_t;
```

Константы

```
 #define _POSIX_C_SOURCE 199309L
 #define COUNT_TESTS 10
 #define MAX_SIZE 100
```

```
#define FILE_SRC "graph.txt"
#define FILE_DOT "graph.gv"
#define FILE_PNG "graph.png"
```

Используемые функции

```
int read_graph_from_file(graph_t *dst, const char *path_src);
```

Считывает граф из файла

```
void export_to_dot(FILE *f, const char *graph_name, graph_t *a);
```

Формирует DOT файл по графу

```
void find_max_simple_path(graph_t *graph, simple_path_t *max_path_info);
```

Поиск максимального простого пути

```
void print_path(simple_path_t *info);
```

Выводит путь

```
int is_graph_connected(graph_t *a);
```

Проверяет граф на связность

```
int minimum_spanning_tree(graph_t *a);
```

Строит остовое дерево минимальной стоимости

```
void table_efficiency(void);
```

Выводит таблицу эффективности

Обоснование выбора структуры и алгоритма

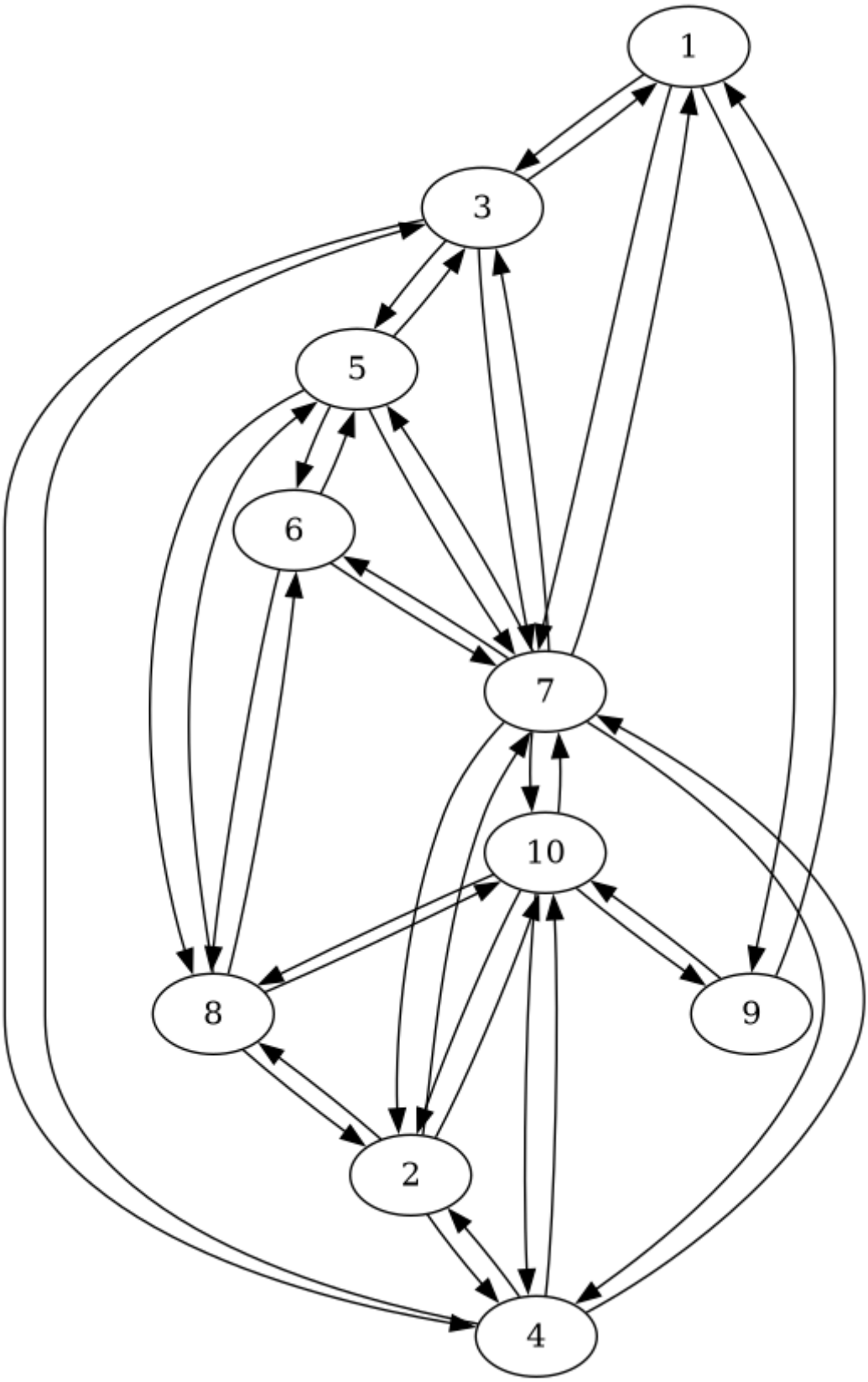
Для представления графов была выбрана матрица стоимостей, поскольку она удобна и наглядна для представления (так как граф неориентированный), а также удобна для дальнейшего поиска максимального простого пути.
Для решения задачи был выбран обход в глубину (DFS), поскольку путь, полученный методом поиска в глубину, в общем случае не является кратчайшим путем из вершины v в вершину u, что подходит под условие задачи – поиск максимального простого пути.

Меню программы

Программа обрабатывает нужную команду:

```
Комманды:
1 - Считать граф из файла
2 - Вывести граф
3 - Найти самый длинный простой путь в графе
4 - Проверить связность графа
5 - Построить остовое дерево минимальной стоимости
6 - Вывести таблицу эффективности
0 - Завершить работу программы
```

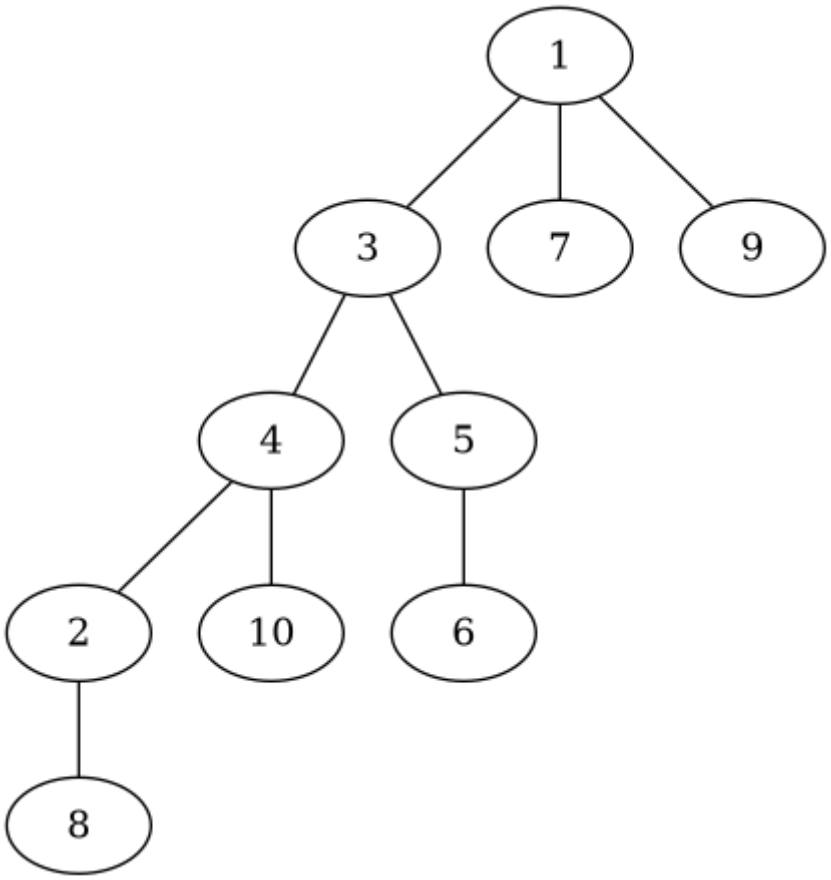
Пример работы программы


1
3
5
6
7
10
8
9
2
4

```
Комманды:
1 - Считать граф из файла
2 - Вывести граф
3 - Найти самый длинный простой путь в графе
4 - Проверить связность графа
5 - Построить остовое дерево минимальной стоимости
6 - Вывести таблицу эффективности
0 - Завершить работу программы

3
Самый длинный простой путь в графе:
1 -> 3 -> 4 -> 2 -> 7 -> 5 -> 6 -> 8 -> 10 -> 9
```

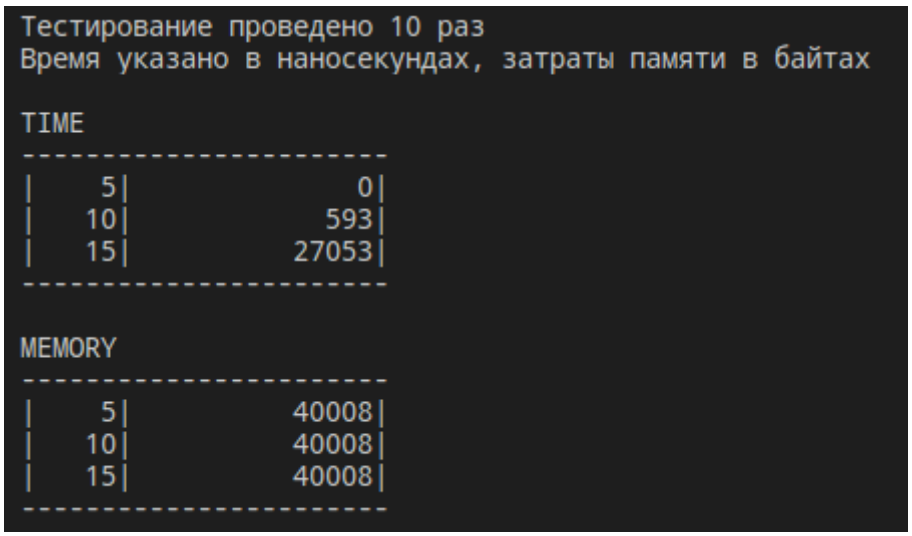
```
Комманды:
1 - Считать граф из файла
2 - Вывести граф
3 - Найти самый длинный простой путь в графе
4 - Проверить связность графа
5 - Построить остовое дерево минимальной стоимости
6 - Вывести таблицу эффективности
0 - Завершить работу программы

4
Граф связный
```

```
Комманды:
1 - Считать граф из файла
2 - Вывести граф
3 - Найти самый длинный простой путь в графе
4 - Проверить связность графа
5 - Построить остовое дерево минимальной стоимости
6 - Вывести таблицу эффективности
0 - Завершить работу программы

5
Общая стоимость остовного дерева: 9
```



Алгоритм поиска максимального простого пути
Начинаем с вершины 0, вызываем функцию для поиска максимального пути, начинающегося с этой вершины. Выполняет поиск в глубину, сохраняя информацию о текущем пути и максимальном найденном пути.
Отмечается текущая вершина как посещенная и добавляет ее в текущий путь. Проверяется, если текущий путь длиннее максимального найденного пути, то обновляется максимальный путь. Затем рекурсивно вызывается для всех смежных вершин, которые еще не были посещены. Так повторяется для всех вершин графа, чтобы убедиться, что максимальный путь начинается с каждой вершины.

Производительность



Алгоритм обхода в глубину имеет сложность O(n*n), что дает нам резко возрастающие затраты времени на его выполнение.

Пример использования

Одной из задач, которую программа по поиску максимального простого пути в графе может решить, является оптимизация пути доставки. Представим, что у компании есть множество складов, множество точек назначения и существует несколько возможных путей между ними. Каждый путь имеет свою стоимость доставки и различные ограничения, такие как время доставки или вместимость транспорта. Основная задача программы по поиску максимального простого пути в графе состоит в нахождении оптимального маршрута, который минимизирует стоимость доставки товаров, учитывая все ограничения. Программа будет анализировать различные пути и искать такой маршрут, который обеспечивает оптимальную доставку.

Выводы по проделанной работе

Для реализации данной задачи был использован алгоритм обхода дерева в глубину. Данный алгоритм имеет сложность O(n*n), что делает его достаточно медленным. Реализация алгоритма обхода в глубину достаточно проста.

Контрольные вопросы

1. **Что такое граф?**

Граф – это конечное множество вершин и ребер, соединяющих их.

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

Графы в памяти могут представляться различным способом. Один из видов представления графов – это матрица смежности B(n*n); В этой матрице элемент b[i,j]=1, если ребро, связывающее вершины Vi и Vj

существует и b[i,j]=0, если ребра нет. У неориентированных графов матрица смежности всегда симметрична. Во многих случаях удобнее представлять граф в виде так называемого списка смежностей. Список смежностей содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с этой вершиной.

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

Над графами можно выполнять операции добавления/удаления вершин и ребер, поиска кратчайшего пути, определения связности графа.

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

Существуют различные способы обхода графов, такие как обход в ширину (BFS) и обход в глубину (DFS).

5. **Где используются графовые структуры?**

Графовые структуры используются в различных областях, включая информационные технологии, транспортные сети, социальные сети, биоинформатику и т. д.

6. **Какие пути в графе Вы знаете?**

В графе можно найти различные типы путей, такие как кратчайший путь, цикл, простой путь и др.

- Эйлеров путь - произвольный путь в графе, проходящий через каждое ребро графа точно один раз.
- Гамильтонов путь - путь в графе, проходящий в точности один раз через каждую вершину графа (а не каждое ребро).
- Простой путь - путь, который не содержит повторяющихся вершин.

Что такое каркасы графа?

Каркас графа - это подграф, содержащий все вершины и некоторые из ребер исходного графа, образующий дерево без циклов. Каркасы используются, например, для поиска минимального остовного дерева взвешенного графа