|  |  |
| --- | --- |
|  | **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**

Студент **Паншин Сергей Константинович**

Группа **ИУ7-33Б**

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

|  |  |
| --- | --- |
| Студент | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Паншин С. К.** |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Никульшина Т. А.** |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Барышникова М. Ю.** |

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2023 г.*

**Условие задачи**

Обработать графовую структуру в соответствии с указанным вариантом задания. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Предложить вариант реальной задачи, для решения которой можно использовать разработанную программу. Ввод данных – на усмотрение программиста. Результат выдать в графической форме.

**Техническое задание**

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

**Входные данные:**

Пункт меню (число от 0 до 9):

Menu:

1) Init graph

2) Add edge

3) Show graph

4) Is graph connected

5) Change start node

6) Find nodes, that path len from start node <= a

7) Print paths to n node from start node

8) Build minimum cost spanning trees

9) Time comparation

0) Quit program

Также путь к файлу, номера узлов и веса ребер, максимальная длина.

**Выходные данные:**

Картинка графа, номера узлов.

**Возможные аварийные ситуации:**

Некорректный ввод: пункта меню, некорректный путь, не корректный номер узла.

**Способ обращения к программе**

В папке с программой запустить команду *make run*.

**Структуры данных**

// граф

typedef struct graph

{

int n;

int \*\*paths;

int edge\_count;

} graph\_t;

// пути

typedef struct path

{

int n;

int start\_node;

int \*lens;

int \*nodes\_marked;

int \*path;

} path\_t;

// инициализация пути

// вход: указатель на указатель на путь, размер

// выход: код возврата

int path\_init(path\_t \*\*path, int n);

// очистка пути

// вход: указатель на указатель на путь

void path\_free(path\_t \*\*path);

// инициализация графа

// вход: указатель на указатель на граф, размер

// выход: код возврата

int graph\_init(graph\_t \*\*g, int n);

// очистка графа

// вход: указатель на указатель на граф

void graph\_free(graph\_t \*\*g);

// добавления ребра в граф

// вход: указатель на граф, вершины ребра и его вес

// выход: код возврата

int graph\_add\_edge(graph\_t \*g, int v, int u, int len);

// граф в дот формат

// вход: указатель на файл, имя, указатель на граф

void graph\_to\_dot(FILE \*f, const char \*graph\_name, graph\_t \*g);

// чтение ребра

// вход: размер графа, указатели на вершины ребра и его вес

void read\_edge(int n, int \*v, int \*u, int \*len);

// чтение графа из файла

// вход: указатель на указатель на граф, указатель на файл

// выход: код возврата

int graph\_read\_from\_file(graph\_t \*\*g, FILE \*file);

// поиск всех минимальных путей в графе

// вход: указатель на указатель на граф, указатель на указатель на путь, стартовая вершина

// выход: код возврата

int dijkstra(graph\_t \*g, path\_t \*\*path, int start\_node);

// вершины, до которых путь не более а

// вход: указатель на путь, макс длина

// выход: вектор вершин

vector\_t \*get\_nodes\_len\_le\_a(path\_t \*path, int a);

// путь из стартовой вершина в заданную

// вход: указатель на путь, номер вершины

// выход: вектор вершин

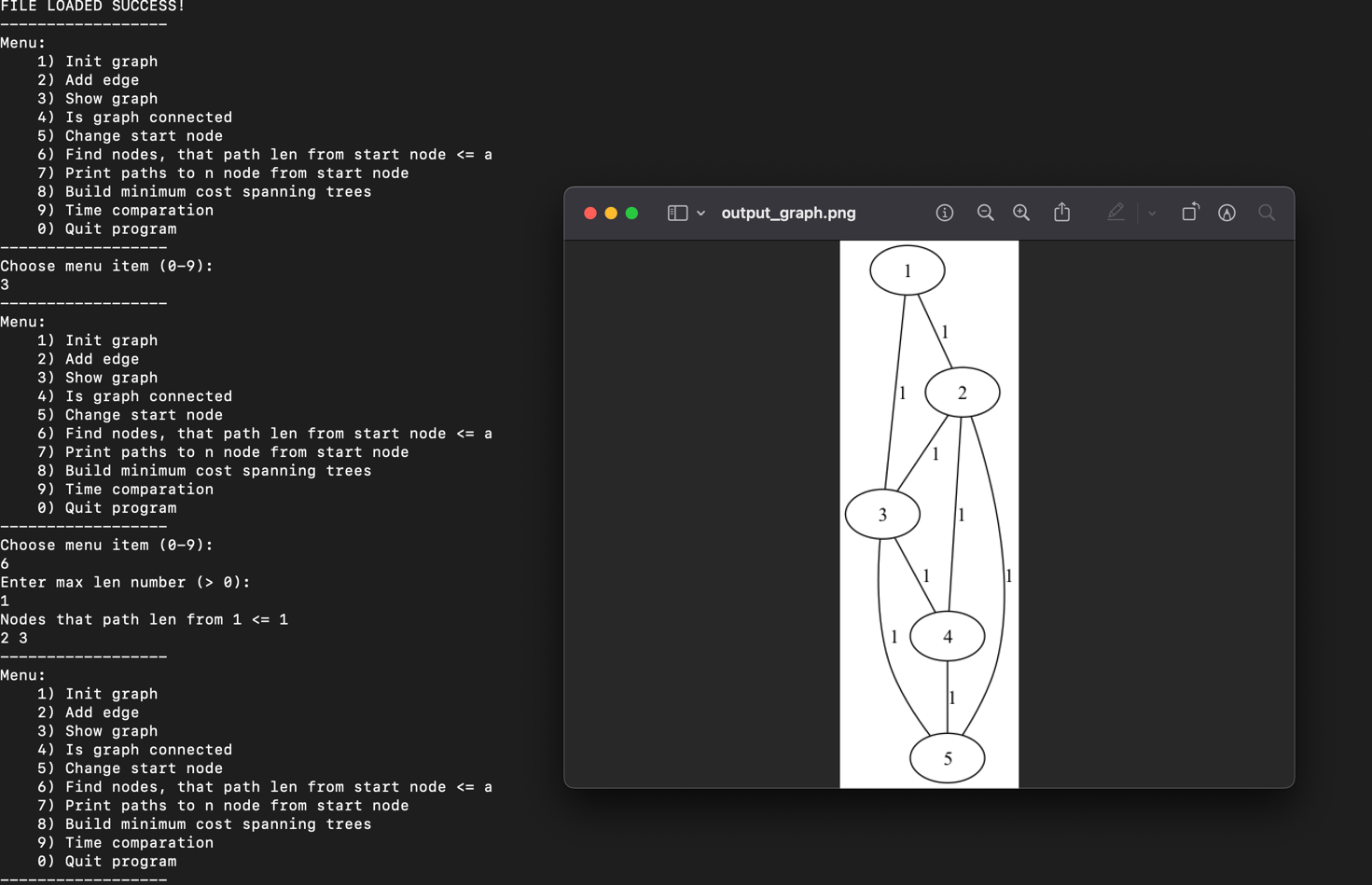
vector\_t \*get\_path\_to\_t(path\_t \*path, int to);ъ

// построение минимального остового дерева

// вход: указатель на граф

void minimum\_spanning\_tree(graph\_t \*g);

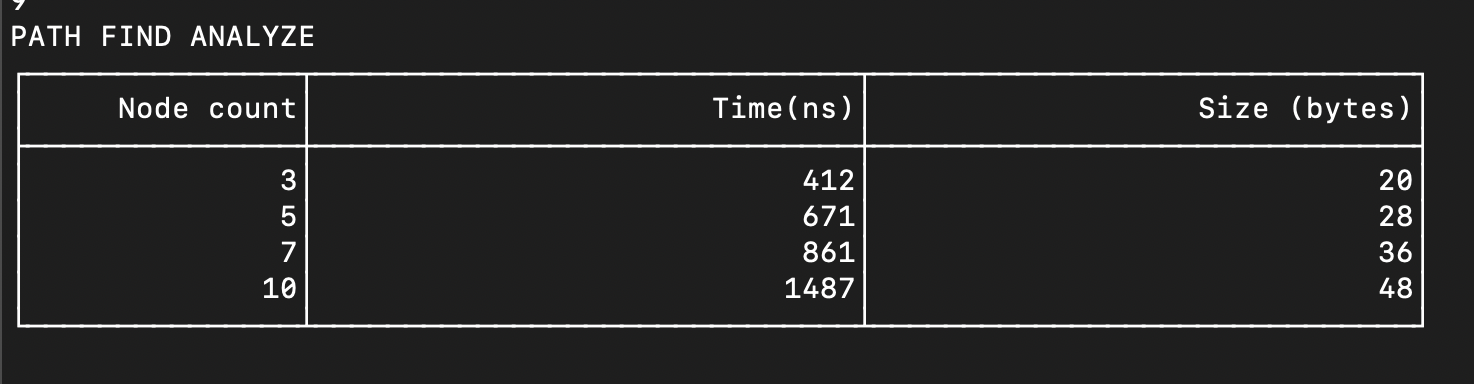
**Пример работы**

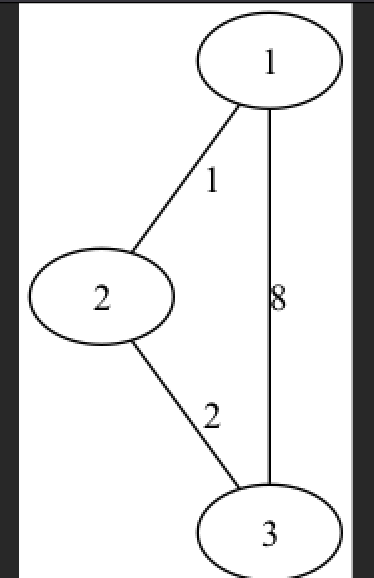
****

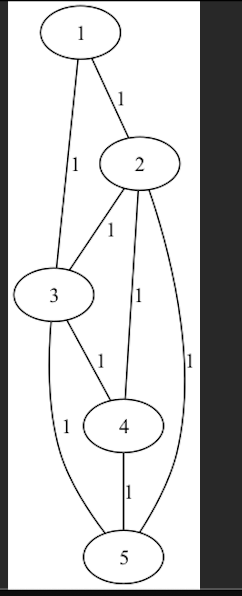
*Примечание: при попытке создать не связанный граф, не связанные вершины не отображаются.*

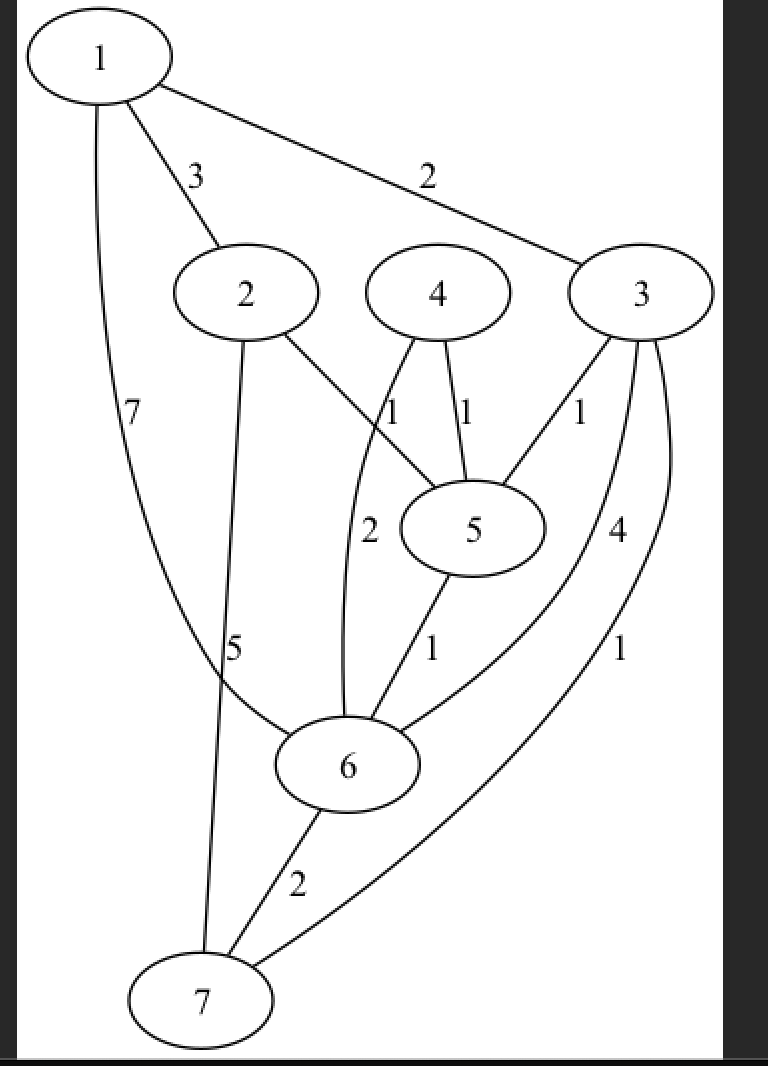
**Замеры**

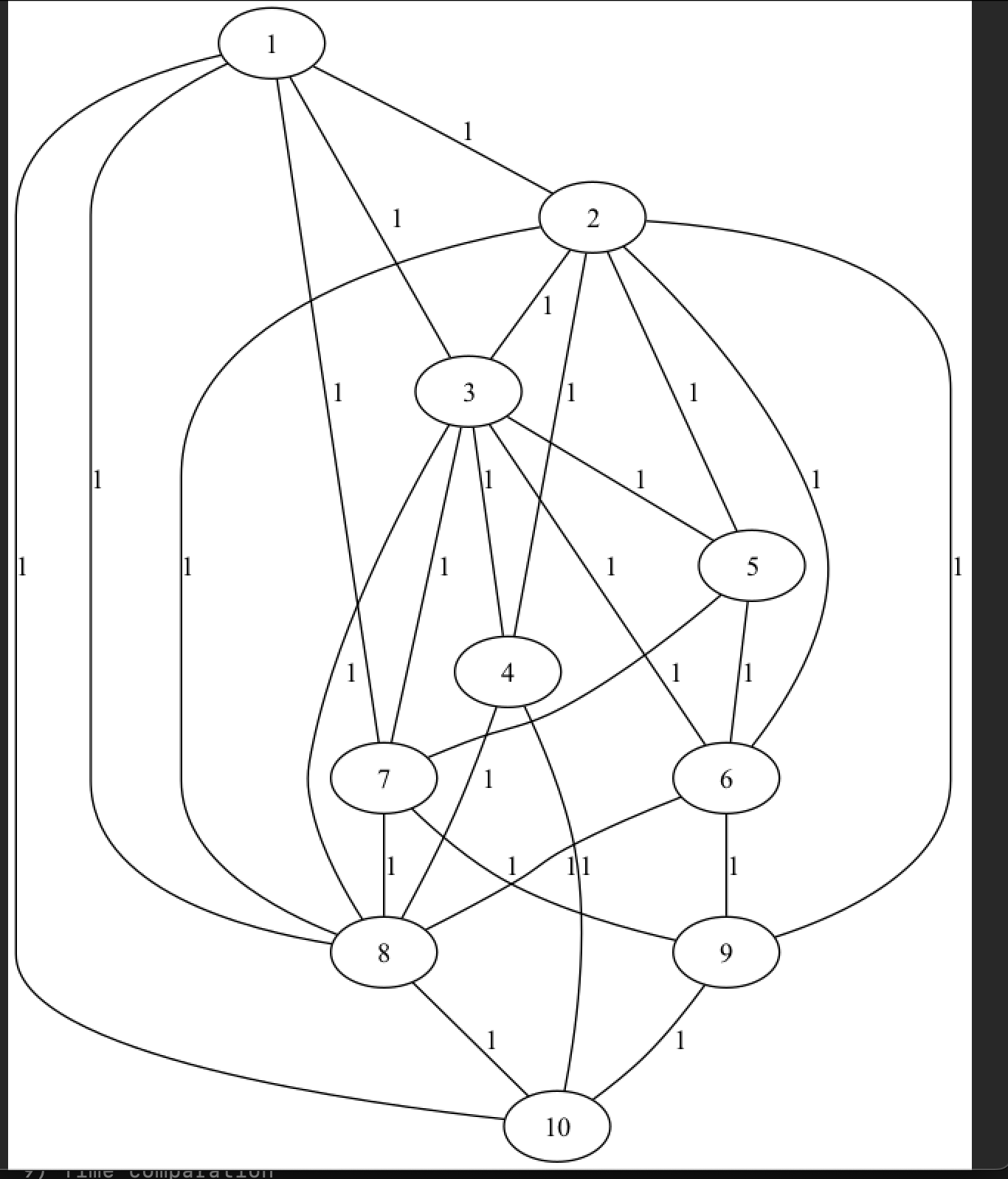
Производится 2000 итераций при замерах, на графах, представленных ниже.









****

**Выводы по проделанной работе**

Для поиска кратчайшего пути используется алгоритм Дейкстры. Он подходит под условие задачи, так как в графе не может быть отрицательных весов. Алгоритм Дейкстры не подходит для несвязного графа, и так же не подходит для графа, где есть отрицательные веса у дуг. Но, так как в моей задаче таких случаев быть не может, то данный алгоритм идеально подходит для решения моей задачи. На поиск вершины будет затрачиваться O(n) операций, а на одну релаксацию — O(1) операций, и итоговая асимптотика алгоритма составляет: O(n^2 + m). Для хранения матрицы используется матрица смежности, так как её удобно использовать для алгоритмов поиска пути.

Реальная задача в которой можно использовать реализованную программу – это посмотреть до каких пунктов может доехать курьер за путь не длинее N километров.

**Контрольные вопросы**

*Что такое граф?*

Граф – конечное множество вершин и соединяющих их ребер;

G = <V, E>. Если пары Е (ребра) имеют направление, то граф называется ориентированным; если ребро имеет вес, то граф называется взвешенным.

*Как представляются графы в памяти?*

С помощью матрицы смежности или списков смежности.

*Какие операции возможны над графами?*

Обход вершин, поиск различных путей, исключение и включение вершин.

*Какие способы обхода графов существуют?*

Обход в ширину и обход в глубину

*Где используются графовые структуры?*

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические.

*Какие пути в графе Вы знаете?*

Эйлеров путь, непростой путь, гамильтонов путь.

*Что такое каркасы графа?*

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (необязательно все) его рёбра.