|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 №8**

**«ГРАФЫ»**

**по курсу «Типы и структуры данных»**

**Вариант 12**

Студент: Писаренко Дмитрий Павлович

Группа: ИУ7-34Б

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Писаренко Д.П.

*подпись, дата фамилия, и.о.*

Преподаватель \_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Рыбкин Ю.А.

*подпись, дата фамилия, и.о.*

# Цель работы

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

# Условие задачи

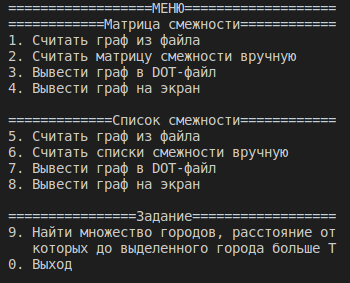
Обработать графовую структуру в соответствии с указанным вариантом задания. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Ввод данных – на усмотрение программиста. Результат выдать в графической форме.

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

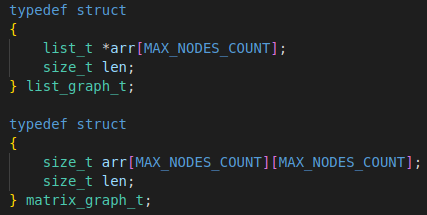
# Техническое задание

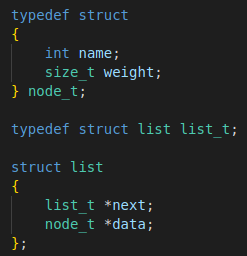
## Исходные данные

*Выбор действия*: целое число от 0 до 9.



## Структуры данных





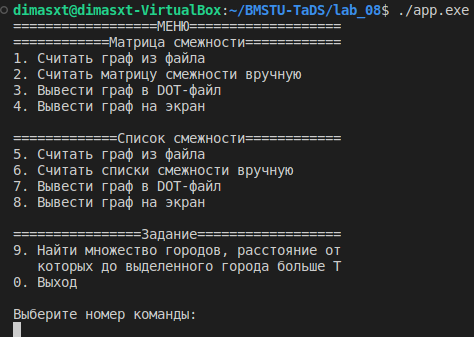
# Способ обращения к программе

Работа с программой осуществляется с помощью консоли.

Сборка осуществляется c помощью команды **make release**

Запуск выполняется с помощью команды **./app.exe**

Дальнейшая работа производится с помощью меню:



# Тестирование

## Позитивные тесты

|  |  |  |  |
| --- | --- | --- | --- |
| # | Входные данные | Выходные данные | Результат |
| 1 | Ключ = 1  Файл: data/graph2.txt | Сообщение:  Файл успешно прочитан | Ожидание следующего ключа |
| 2 | Ключ = 3  Ключ = 4 | Выводится png файл графа с помощью скрипта | Ожидание следующего ключа |
| 3 | Ключ = 9  Длина пути T: 2  Столица: 2 | На экран выводятся вершины, от которых до столицы длина больше T | Ожидание следующего ключа |
| 4 | Ключ = 2  Кол-во вершин: 3  0 1 2  2 0 1  1 0 0 | Сообщение:  Матрица успешно прочитана | Ожидание следующего ключа |

## Негативные тесты

|  |  |  |  |
| --- | --- | --- | --- |
| # | Входные данные | Выходные данные | Результат |
| 1 | Ключ = 1  Файл: data/rwgdf.txt | Сообщение:  Введено неверное имя файла | Ожидание следующего ключа |
| 2 | Ключ = 2  Вершины: dsss | Сообщение:  При вводе возникала ошибка | Код ошибки 2 |
| 3 | Ключ = dsfw | Номер команды – целое число от 0 до 9 | Код ошибки 5 |

# Таблица с результатами измерения времени и памяти

Время в таблице указано в секундах, память в байтах.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Размер** | **Матрица** | | **Списки** | |
| **Время** | **Память** | **Время** | **Память** |
| **3** | 0.0002 | 72 | 0.0003 | 312 |
| **10** | 0.0011 | 200 | 0.0025 | 488 |
| **20** | 0.0572 | 3200 | 0.3633 | 12320 |
| **50** | 0.9624 | 20000 | 5.2412 | 122410 |

# Контрольные вопросы

*1. Что такое граф?*

Граф – это конечное множество вершин и ребер, соединяющих их, G = < V,E >, где V – конечное непустое множество вершин; Е – множество ребер (пар вершин).

Если пары Е (ребра) имеют направление, то граф называется ориентированным (орграф), если иначе - неориентированный (неорграф). Если в пары Е входят только различные вершины, то в графе нет петель. Если ребро графа имеет вес, то граф называется взвешенным.

Неорграф называется связным, если существует путь из каждой вершины в любую другую.

*2. Как представляются графы в памяти?*

В памяти удобно представлять граф в виде матрицы смежности или списка смежности.

Матрица смежности B(n\*n) – элемент b[i,j]=1, если существует ребро, связывающее вершины i и j, и =0, если ребра не существует.

Список смежностей – содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с ней. Входы в списки смежностей могут храниться в отдельной таблице, либо же каждая вершина может хранить свой список смежностей.

*3. Какие операции возможны над графами?*

Обход вершин и поиск различных путей: поиск кратчайшего пути от одной вершины к другой (если он есть), поиск кратчайшего пути, поиск эйлерова пути, поиск гамильтонова пути.

*4. Какие способы обхода графов существуют?*

Обход в ширину (BFS – Breadth First Search) - обработка вершины V осуществляется путём просмотра сразу всех «новых» соседей этой вершины, которые последовательно заносятся в очередь просмотра.

Обход в глубину (DFS – Depth First Search) - начиная с некоторой вершины v0, ищется ближайшая смежная ей вершина v, для которой в свою очередь осуществляется поиск в глубину до тех пор, пока не встретится ранее просмотренная вершина, или не закончится список смежности вершины v (то есть вершина полностью обработана). Если нет новых вершин, смежных с v, то вершина v считается использованной, идет возврат в вершину, из которой попали в вершину v, и процесс продолжается до тех пор, пока не получим v = v0. При просмотре используется стек.

*5. Где используются графовые структуры?*

Графовые структуры могут использоваться в задачах, где между элементами могут быть установлены произвольные связи. Наиболее распространенное использование таких структур — при решении различных задачах о путях.

*6. Какие пути в графе Вы знаете?*

Эйлеровый путь - путь в графе, проходящий через каждое ребро ровно один раз. (путь может проходить по некоторым вершинам несколько раз – в этом случае он является непростым)

Гамильтонов путь - путь, проходящий через каждую вершину ровно один раз.

Такие пути могут не существовать в графах.

*7. Что такое каркасы графа?*

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (не обязательно все) его рёбра. Для построения каркасов графа используются алгоритмы Крускала и Прима.

# Вывод

Хранение графа в виде матрицы смежности оказалось выгоднее как по времени, так и по памяти.