|  |  |
| --- | --- |
| 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)** |

**Типы и структуры данных.**

**Лабораторная работа №7.**

**«Графы»**

Студент **Леонов Владислав Вячеславович**

Группа **ИУ7-36Б**

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Леонов В.В.

*подпись, дата фамилия, и.о.*

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

*подпись, дата фамилия, и.о.*

*2020 г.*

Оглавление

[Условие и техническое задание 3](#_Toc59586688)

[Цель работы 3](#_Toc59586689)

[Задание 3](#_Toc59586690)

[Входные данные 3](#_Toc59586691)

[Выходные данные 3](#_Toc59586692)

[Обращение к программе 3](#_Toc59586693)

[Функции программы 4](#_Toc59586694)

[Аварийные ситуации 4](#_Toc59586695)

[Реализация 5](#_Toc59586696)

[Структуры данных 5](#_Toc59586697)

[Структура ***graph\_matrix\_t*** 5](#_Toc59586698)

[Структура ***graph\_list\_t*** 5](#_Toc59586699)

[Структура ***node\_t*** 6](#_Toc59586700)

[Алгоритм 7](#_Toc59586701)

[Тестирование 8](#_Toc59586702)

[Аварийные ситуации 8](#_Toc59586703)

[Штатное поведение 9](#_Toc59586704)

[Оценка эффективности использования графов в виде матриц и списков смежности. 11](#_Toc59586705)

[Контрольные вопросы 13](#_Toc59586706)

[Вывод о проделанной работе 14](#_Toc59586707)

# Условие и техническое задание

## Цель работы

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

## Задание

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 два города и соединяющий их путь, который проходит через каждую из дорог системы только один раз.

## Входные данные

1. **Имя файла** для чтения данных.
2. **Опция меню.**

## Выходные данные

1. Печать графа в формате .**png**.
2. Сравнительная таблица анализа структур.
3. Время выполнения операции поиска эйлерова пути.
4. Объем памяти, необходимый для хранения неориентированного графа в виде матрицы и списка смежности.

## Обращение к программе

Обращение к программе **app.exe** осуществляется через запуск из терминала.

## Функции программы

1. Чтение данных из файла.
2. Печать исходных данных в графическое представление.
3. Поиск пути, проходящего через каждую из дорог системы только 1 раз.
4. Сравнительный анализ структур.
5. Завершить работу.

## Аварийные ситуации

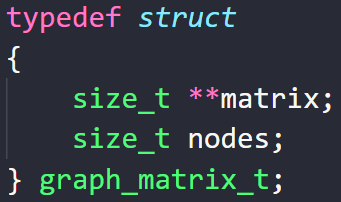
1. Некорректная опция меню (буква или значение, несоответствующее ни одному пункту из меню.
2. Некорректное название файла.
3. Некорректное содержимое файла.
4. Ошибка выделения памяти операционной системой.

# Реализация

## Структуры данных

### Структура ***graph\_matrix\_t***

Для хранения графа в виде матрицы смежности используется агрегированный тип ***graph\_matrix\_t.***

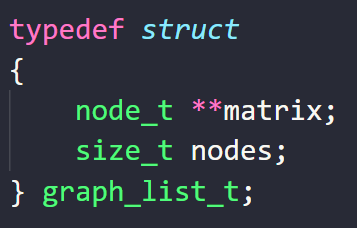


Поле ***matrix*** представляет собой матрицу смежности для заданного неориентированного графа.

Поле ***nodes*** представляет собой беззнаковое целое число и служит для хранения количества вершин в матрице.

### Структура ***graph\_list\_t***

Для хранения графа в виде списка смежности используется агрегированный тип ***graph\_list\_t.***

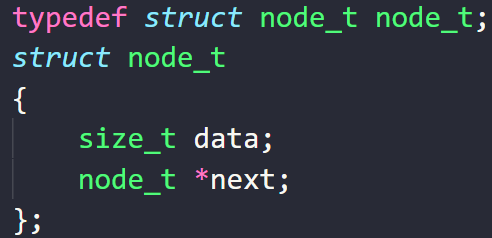


Поле ***matrix*** представляет собой список смежности для заданного неориентированного графа.

Поле ***nodes*** представляет собой беззнаковое целое число и служит для хранения количества вершин в матрице.

### Структура ***node\_t***

Для хранения узла списка используется агрегированный тип ***node\_t.***



Поле ***data*** представляет собой целое число и служит для хранения данных в узле списка.

Поле ***next*** представляет собой указатель и служит для хранения адреса следующего узла списка.

## Алгоритм

1. Пользователь вводит номер команды из меню.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия из меню программы.

Чтение данных выполняется из файла (функции ***graph\_list\_fread*** и ***graph\_matrix\_fread***). Требуется, чтобы в начале файла было написано число вершин в графе, а затем следовала матрица смежности. Заметим, что матрица обязательно должна быть симметрична и главная диагональ должна быть нулевая.

Известно, что эйлеров путь существует не всегда в графе, поэтому предварительно проверяются несколько условий:

1. Граф должен быть связанным (функции ***graph\_list\_is\_connected*** и ***graph\_matrix\_is\_connected***). Выполняется поиск в глубину и проверяется достижимость всех вершин из одной (функции ***graph\_list\_dfs*** и ***graph\_matrix\_dfs***).

2. В графе должно быть не более 2 вершин нечетных степеней связанным (функции ***graph\_list\_is\_eulerian*** и ***graph\_matrix\_is\_eulerian***).

В случае выполнения заданных условий выполняется постепенное удаление ребер из неориентированного графа в порядке, при котором не будут возникать новые компоненты связности, пока граф не окажется пустым (функции ***graph\_list\_find\_eulerian\_path*** и ***graph\_matrix\_ find\_eulerian\_path***).

# Тестирование

## Аварийные ситуации

|  |  |  |
| --- | --- | --- |
| № | Описание теста | Поведение программы |
| 1 | Некорректная опция из меню |  |
| 2 | Некорректный файл |  |
| 3 | Пустой файл |  |

## Штатное поведение

|  |  |  |
| --- | --- | --- |
| № | Описание теста | Демонстрация работы программы |
| 1 | Чтение данных из файла. |  |
| 2 | Печать исходных данных в графическое представление |  |
| 3 | Поиск эйлерова пути  (существует) |  |
| 4 | Поиск эйлерова пути  (не существует) |  |
| 5 | Сравнительный анализ структур |  |
| 6 | Завершение работы |  |

# Оценка эффективности использования графов в виде матриц и списков смежности.

Время в тактах процессора, объем памяти в байтах.

|  |  |  |  |
| --- | --- | --- | --- |
| Размер | Тип структуры | Время выполнения операции поиска эйлерова пути | Объем занимаемой памяти структурой |
| 16x16 | Матрица смежности | 606537 | 2048 |
| Список смежности | 684817 | 1910 |
| 32x32 | Матрица смежности | 674181 | 8192 |
| Список смежности | 968854 | 7910 |
| 64x64 | Матрица смежности | 607912 | 32768 |
| Список смежности | 2576101 | 32822 |
| 128x128 | Матрица смежности | 981659 | 131072 |
| Список смежности | 16928412 | 130012 |
| 256х256 | Матрица смежности | 2833457 | 524288 |
| Список смежности | 1252022425 | 522585 |

Данные результаты были получены при нормальном распределении количества ребер, то есть количество ребер приблизительно равно половине от максимального количества.

Пользуясь данными, полученными в ходе предыдущих лабораторных работ, можно сделать вывод о том, что при росте количества ребер относительно данного соотношения список будет сильнее проигрывать по памяти матрице за счет необходимости хранения дополнительных указателей (проигрыш в 2 раза), а при уменьшении количества ребер список будет более эффективен, однако в этом случае можно использовать не обычные матрицы, а разреженные, что позволит уменьшить объем занимаемой памяти.

Также явно прослеживается закономерность, при увеличении размерности список начинает сильно проигрывать по времени. В случае большего количества ребер, очевидно, проигрыш будет еще более весомым, т.к. потребуется еще больше обращений к динамической памяти, операции которой очень затратны по времени. В случае уменьшения количества ребер можно использовать разреженный вид хранения матрицы, и она также будет эффективнее по времени, чем список, за счет меньшего количества обращений к динамической памяти.

Таким образом, представление графов в виде матриц более эффективно, чем в виде списков.

# Контрольные вопросы

**1. 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.

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

С помощью матрицы смежности или списков смежности.

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

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

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

Обход в ширину, обход в глубину.

**5. Где используются графовые структуры?**

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи. Например, задачи о навигации.

**6. Какие пути в графе Вы знаете?**

Эйлеров путь, простой путь, сложный путь, гамильтонов путь.

**7. Что такое каркасы графа?**

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

# Вывод о проделанной работе

В результате лабораторной работы были изучены структуры данных – неориентированные графы на основе списков и массивов, методы их реализации на основе известных ранее структур, а также особенности обработки подобных структур данных. Были закреплены навыки работы с динамической памятью, а также освоены методы и способы представления деревьев в графическом виде на языке DOT.

Пользуясь данными, полученными в ходе предыдущих лабораторных работ, можно сделать вывод о том, что при росте количества ребер относительно нормального соотношения список будет сильнее проигрывать по памяти матрице за счет необходимости хранения дополнительных указателей (проигрыш в 2 раза), а при уменьшении количества ребер список будет более эффективен, однако в этом случае можно использовать не обычные матрицы, а разреженные, что позволит уменьшить объем занимаемой памяти.

При увеличении размерности список начинает сильно проигрывать по времени. В случае большего количества ребер, очевидно, проигрыш будет еще более весомым, т.к. потребуется еще больше обращений к динамической памяти, операции которой очень затратны по времени. В случае уменьшения количества ребер можно использовать разреженный вид хранения матрицы, и она также будет эффективнее по времени, чем список, за счет меньшего количества обращений к динамической памяти.

Таким образом, представление графов в виде матриц более эффективно, чем в виде списков.