| 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

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7 «Работа с графами»**

Студент Романов Семен Константинович

Группа ИУ7 – 35Б

Проверено Силантьева Александра Васильевна

**Описание задачи**

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

Задан граф - не дерево. Проверить, можно ли превратить его в дерево удалением одной вершины вместе с ее ребрами.

**Техническое задание:**

**Входные данные:**

1. **Текстовый файл:** Текстовый файл содержит количество вершин и дальнейшее их представление парами целых чисел для **неориентированного графа** (далее по отчету просто “граф”)

**Выходные данные:**

1. **Результаты работы с графом:** Файл формата .png содержащий графическое изображение изначального неориентированного графа и графа, с удаленной вершиной (в случае нахождения оной)
2. **Эффективность работы структур данных:** Время работы функции, занимаемая память структуры

**Способ обращения к программе:** запускается из терминала командой ./app.exe <filename>, где <filename> - имя входного файла

**Функция программы:**

1. Вывод графа.
2. Нахождение вершины для перестройки графа.
3. Вывод графа без вышеуказаной вершины
4. Вывод характеристик структур данных, указанных в секции выходных данных.

**Аварийные ситуации:**

1. Неверное название файла или пустой файл: Сообщение “Incorrect file! Exiting the program…”
2. Некорректное содержание файла: Сообщение “Incorrect input! Exiting the program…”

**Структуры данных:**

**Структура узла для списка смежностей:**

1. typedef struct vertex
2. {

int value;

struct vertex \*next;

1. } vertex\_t;

7.

**Обозначения полей:**

* **Int value:** Номер вершины
* **Struct vertex \*next:** Ссылка на следующий узел

**Структура для графа:**

1. typedef struct graph
2. {
3. int size;
4. int edges;
5. vertex\_t \*array;
6. } graph\_t;

**Обозначения полей:**

* **Int size:** Количество вершин графа
* **Int edges:** Количество рёбер графа
* **Vertex\_t \*array:** Ссылка на массив со входами в списки смежностей

**Алгоритм**

1. При начале работы программы задается название файла, откуда будут считываться данные
2. Далее производится создание списков смежностей. Создается массив равный количеству вершин, и по индексам, равным номерам вершин, записываются вершины
3. При нахождении такого узла, при удалении которого вместе с ребрами граф перестраивается в дерево, мы руководствуемся двумя свойствами дерева:
   1. Связанность всех вершин (Связанный граф)
   2. Количество рёбер = Количество вершин – 1

То есть, мы должны найти такую вершину, при удалении которой выполняются первое и второе условие для всего графа

Обходя массив со списком смежностей циклом, ищем такие вершины, чтобы выполнялось второе свойство дерева

Затем проверяем сохранилась ли связанность списка. В данном случае алгоритм не особо важен, поскольку нам необходимо обойти все вершины, поэтому был выбран обычный обход в глубину

Если вершин, удовлетворяющих условиям, не найдены, то это означает, что нужных нам вершин нет

**Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| N | Test | Input | Output |
| 1 | Некорректное имя файла | Abc.txt  (Файл не существует) | “Incorrect file! Exiting the program” |
| 2 | Пустой файл | Abc.txt  (Файл пуст) | “Incorrect file! Exiting the program” |
| 3 | Некорректное значение полей в файле | -1 4 | “Invalid Input! Exiting the program” |
| 4 | Некорректное значение полей в файле | 2 7  (при количестве вершин равных 6) | “Invalid Input! Exiting the program” |
| 6 | Корректный ввод всех данных | Корректный файл, содержащий граф | Графы, результат работы программы  Характеристика структур данных |

**Оценка эффективности:**

Все время измеряется в тактах процессора

**Время нахождения элемента:**

|  |  |
| --- | --- |
| **Количество узлов в связанном списке** | **Время выполнения** |
| 15 | 8888 |
| 20 | 10228 |
| 40 | 13488 |

**Занимаемая память в байтах:**

Все размеры в байтах.

|  |  |
| --- | --- |
| **Количество узлов в связанном списке** | **Размер списка** |
| 15 | 240 |
| 20 | 320 |
| 40 | 640 |

**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 (BFS – Breadth First Search), обход в глубину (DFS – Depth First Search).

**5. Где используются графовые структуры?**

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

**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**

В данной реализации алгоритм состоит из двух этапов — обход списка и обход в глубину. Если V – количество вершин графа, а E – количество ребер, то обход списка имеет сложность O(V), а алгоритм перебора рёбер имеет сложность O(E + V), таким образом общая сложность алгоритма, без учёта константы будет O(E + V) (в худшем случае). Преимущество списка смежности над матрицей заключается в том, что в списке с большим количеством вершин память под список выделятся гораздо меньше, чем под матрицу. Также можно отметить, что при большой разряженности матрицы (малом количестве ребер), список смежностей будет работать быстрее, нежели матрица.