|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство образования и науки 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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7  
«ОБРАБОТКА ГРАФОВ»**

Название предмета: Типы и структуры данных

Студент: Малышев Иван Алексеевич

Группа: ИУ7-31Б

*2020 г.*

**Цель работы**: 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.

**Описание условия задачи**

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

**Техническое задание**

**Входные данные**

1. **Целое число, представляющее собой количество вершин в графе:** целое положительное число.
2. **Целые числа, представляющие собой стоимость очередного ребра:** целые числа.
3. **Целое число, представляющее собой номер вершины(столица):** целое положительное число.
4. **Целое число, представляющее собой расстояние Т:** целое положительное число.

**Выходные данные**

1. Файлы с изображением исходного графа и графа с пометками: вершина красного цвета – столица, вершины синего цвета – множество городов более чем на расстоянии Т.

**Функция программы:** программа находит решение изложенной в условии задачи и визуализирует найденное решение при помощи фреймворка GraphViz.

**Способ обращения к программе:**

К программе можно обратиться через терминал ОС по её имени «app.exe». Дальнейшие инструкции будут выведены после запуска.

**Аварийные ситуации:**

* Некорректное число вершин
* Некорректная стоимость
* Несуществующая столица
* Некорректное расстояние

Во всех указанных случаях программа завершится корректно или сообщит об ошибке.

**Структуры данных**

Реализация графа в виде матрицы смежности:

typedef struct graph

{

int \*\*adjacency; - матрица смежности/стоимостей

int \*markers; - массив меток

int vertexes\_count; - число вершин

} graph\_t;

**Описание функций**

int input\_graph(graph\_t \*\*graph) – ввод графа

int get\_min\_costs(graph\_t \*graph, int num\_of\_vrtx) – поиск минимальных длин от столицы до других городов

int is\_connective(graph\_t \*graph) – проверка связности графа

void print\_graph(graph\_t \*graph) – печать графа в файл .png

void cities\_on\_dist\_at\_least\_T(graph\_t \*graph, int capital, int T) – поиск множества городов

void free\_graph(graph\_t \*graph) – освобождение памяти из под графа

**Алгоритм**

После ввода элементов графа и прочих данных граф проверяется на связность. Так как в данной задаче реализуется обработка неориентированного графа, то для проверки его на связность достаточно найти в матрице стоимостей нулевую строку. Если такой не найдено, граф является связным.

После того, как была произведена проверка на связность, если граф является несвязным, то выдаётся соответствующее сообщение и работа программы прекращается. В противном случае, с помощью алгоритма Дейкстры находятся минимальные длины до вершин от выбранной. Далее, длины сравниваются с величиной расстояния, и те, что больше неё, их вершины помечаются.

Алгоритм Дейкстры позволяет находить кратчайшие пути от одной из вершин графа до всех остальных. По условию задачи нужно найти расстояния от выделенного города (столицы) до остальных городов, а потом сравнить их с величиной Т. Таким образом можно судить о том, находится ли город от столицы на расстоянии больше, чем на Т, и «отметить» его, если это так. Поэтому использование алгоритма Дейкстры здесь обосновано.

**Тестирование**

Положительные тесты

|  |  |  |  |
| --- | --- | --- | --- |
|  | Тест | Пользовательский ввод | Результат |
| 1 | Ввод графа с такими городами | 3  6  3  2  0  2 | Города, находящиеся от города 0 дальше, чем на 2: 1 2 |
| 2 | Ввод графа, в котором нет таких городов | 3  5  1  1  0  2 | Города, находящиеся от города 0 дальше, чем на 2: таких городов нет |

Негативные тесты

|  |  |  |  |
| --- | --- | --- | --- |
|  | Тест | Пользовательский ввод | Результат |
| 1 | Ввод несвязного графа | 3  5  0  0  0  2 | Ошибка: граф не является связным |
| 2 | Неверный ввод числа вершин | 0 | Возникли проблемы с созданием графа |
| 3 | Неверный ввод стоимости ребра | 3  1  s | Возникли проблемы с созданием графа |
| 3 | Неверный ввод столицы | 3  5  1  1  4 | Ошибка выбора столицы |
| 4 | Неверный ввод расстояния | 3  5  1  1  0  -1 | Ошибка ввода расстояния |

**Оценка эффективности**

Измерения эффективности реализаций очереди будут производиться в единицах измерения – тактах процессоров. Для измерения была специально написана ассемблерная функция, поэтому погрешность измерений минимальна.

|  |  |  |  |
| --- | --- | --- | --- |
| Кол-во вершин | 5 | 10 | 15 |
| Время выполнения (в тактах процессора) | 5776 | 10563 | 17122 |
| Занимаемая память (в байтах): | 144 | 464 | 984 |

**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.

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

С помощью матрицы смежности или списков смежности.

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

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

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

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

1. **Где используются графовые структуры?**

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

1. **Какие пути в графе Вы знаете?**

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

1. **Что такое каркасы графа?**

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

**Вывод**

Основой выполнения данной задачи является алгоритм Дейкстры – поиск минимальных длин путей от выбранной вершины до других. С помощью него можно найти расстояния от столицы до других городов, что нам и нужно было. Сложность этого алгоритма – O(V2), где V – количество вершин графа. По условию задачи мы работаем с неорграфами, поэтому здесь отлично подошла реализация графа в виде матрицы смежностей/стоимостей, т. к. она отлично отражает симметрию отношений смежности и с ней удобно работать при поиске минимальных длин путей в неорграфе.

Такая программа может подойти для решения, например, следующей задачи: есть система городов и двусторонних дорог. От того города, в котором вы находитесь, нужно уехать в какой-то другой город, находящийся более чем на определённом расстоянии, во избежание надвигающейся ракетной атаки. Зная их радиус поражения, эта программа поможет найти подходящие города.