|  |  |
| --- | --- |
| **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 **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 8 |

Графы

**Дисциплина:** Типы и структуры данных

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ7-35Б |  | А. В. Толмачев |
|  | (Группа) |  | (И.О. Фамилия) |
|  |  |  |  |

Москва, 2022

**Описание условия задачи**

Заданы две системы двухсторонних дорог с одним и тем же множеством городов (железные и шоссейные дороги). Найти минимальный по длине путь из города A в город B, который может проходить как по железной, так и по шоссейной дорогам, и места пересадок с одного вида транспорта на другой на этом пути.

**Описание ТЗ**

1. **Описание исходных данных**

Исходными данными является путь до текстового файла, содержащего список смежности неориентированного графа.

Ограничения исходных данных:

1. Путь до файла должен быть указан как аргумент командной строки
2. Первым числом в файле должно быть указано количество вершин
3. Допустимы только ребра с неотрицательным весом
4. Указание несуществующего файла является некорректным вводом
5. Если файл содержит любые символы, кроме цифр, пробельных символов и переводов строк, то он считается некорректным.
6. Граф задается списками смежности в формате:

№исходной вершины №связанной вершины вес\_ребра тип\_дороги

1. Тип дороги указывается целым числом:

0 – автомобильная дорога, 1 – железная

1. Вершины графы именуются натуральным числом
2. Недопустимо непоследовательное обозначение вершин графа:

Например, если присутствует вершина с номером 6, то должны присутствовать вершины со всеми номерами, меньше 6

1. При поиске пути указывается сначала исходная вершина, после, через пробел, конечная вершина
2. Недопустима работа с нецелыми весами и вершинами
3. Пункт меню задается целым числом. Возможен выбор одновременно только 1 пункта меню.
4. Ввод отсутствующего пункта меню является некорректным вводом.
5. **Описание задачи, реализуемой программой**

Программа позволяет искать минимальный по цене путь между двумя вершинами в неориентированном графе с дорогами двух типов.

**Выходные данные**

1. Графическое представление графа, построенное с помощью graphviz
2. Длина минимального пути между двумя вершинами
3. Путь между двумя вершинами
4. **Способ обращения к программе**

Для запуска программы необходимо из консоли вызвать исполняемый файл ./app.exe.

1. **Описание возможных аварийных ситуаций и ошибок пользователя**

Аварийные ситуации:

1. Ошибка выделения памяти

Ошибки пользователя:

1. Ввод данных, не удовлетворяющих ограничениям исходных данных, описанных ранее
2. Выбор несуществующего пункта меню
3. Поиск пути между не существующими вершинами

**Описание внутренних структур данных:**

Граф:

**typedef struct** **graph\_t**

{

**Struct node\_t** \*\*adj\_list;

**size\_t count\_edge**;

**char \*visited**;

} graph\_t;

Узел списка смежности:

**typedef struct** **node\_t**

{

**char type**;

**int dest**;

int weight;

struct node\_t \*next;

};

**Набор тестов**

1. Позитивные тесты
   1. Считывание корректно заданного графа
   2. Поиск пути между существующими вершинами
   3. Корректный выбор пункта меню
2. Негативные тесты

Тесты с входными данными, не удовлетворяющими ограничениям исходных данных, записанным ранее.

Алгоритм Дейкстры:

* + - 1. Заводится дополнительный массив d, в котором для каждой вершины будет храниться длина кратчайшего пути на текущий момент.
      2. Изначальной все элементы инициализируются максимальным значением, которое заведомо больше возможной максимальной длины пути.
      3. Элемент массива, соответствующий начальной вершине, устанавливается равным 0.
      4. Массив посещенных вершин инициализируется 0.
      5. Выполняется n итераций – до тех пор, пока все вершины не будут посмотрены.
      6. На каждой итерации выбирается вершина с наименьшей длиной пути, которая хранится в массиве d
      7. Выбранная вершина отмечается посещенной
      8. Просматриваются все ребра, исходящие из вершины, для каждой из которой проверяется текущая длина пути до него. Если из просматриваемой вершины в новую вершину можно попасть по более короткому пути, значение в соответствующей ячейке массива d обновляется.
      9. После выполнения всех итераций ячейка массива, соответствующая конечной вершине, будет содержать длину кратчайшего пути до него.

**Вывод:**

В рамках данной лабораторной работы была выполнена работа с графами. Для поиска пути между вершинами был выбран алгоритм Дейкстры. Данный алгоритм удовлетворяет поставленной задаче, несмотря на отсутствие поддержки отрицательных меток, так как пути между городами должны иметь положительный вес. При этом данный алгоритм ищет минимальный путь только между заданными вершинами, а не между всем, как например алгоритм Флойда-Уоршалла. Асимптотическая сложность алгоритма Дейкстры в данной реализации: O(n2)

В программе граф хранится в виде списка смежности. Этот вариант является оптимальным, так как в случае реализации матрицы смежности – сама матрица будет симметричной для неориентированного графа и будет излишнее дублирование данных, что хуже по памяти. Так же в случае относительно разреженных графов при представлении графа списком смежности не нужно хранить нулевые элементы, что так же позволит сэкономить память.

**Контрольные вопросы:**

1. Что такое граф?

Граф – это совокупность двух конечных множеств: множества точек и множества линий, попарно соединяющих некоторые из этих точек.

Граф с ребрами, имеющими направление, называется ориентированным, иначе – неориентированным.

Если ребра графа имеют вес, то граф называется взвешенным.

2. Как представляются графы в памяти?

В памяти граф может быть представлен списком смежности, матрицей смежности, матрицей инцидентности, списком инцидентности, списком ребер.

3. Какие операции возможны над графами?

Обход вершин графа. Поиск кратчайших пути от одной вершины до другой, поиск кратчайшего пути между всеми вершинами, поиск Гамильтоновых путей, поиск Эйлеровых путей, поиск вершин в графе.

4. Какие способы обхода графов существуют?

Поиск в глубину, поиск в ширину

5. Где используются графовые структуры?

Для нахождения путей между различными объектами, представления различных связей и схем.

6. Какие пути в графе вы знаете?

Эйлеров путь – произвольный путь в графе, проходящий через каждое ребро ровно 1 раз. При это путь может проходить через 1 вершину несколько раз.

Путь, проходящий через каждую вершину ровно 1 раз – Гамильтонов путь.

Простой путь – путь без повторяющихся вершин.

7. Что такое каркасы графа?

Каркас графа – подграф, которой содержит все вершины графа и является деревом