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 №7

Название «Графы»

Дисциплина «Типы и структуры данных»

Вариант 10

Студент ИУ7-31Б _____ Корниенко К.Ю.
(подпись, дата) (Фамилия И.О.)

Преподователь _____ Барышникова М.Ю.
(подпись, дата) (Фамилия И.О.)

Москва, 2021

Содержание

Введение 3
1 Аналитический раздел 4
1.1 Техническое задание 4
1.2 Теоритические сведения 4
1.3 Представление графа в памяти 4
1.3.1 Матрица смежностей 4
1.3.2 Список смежностей 4
1.3.3 Вывод 5
1.4 Алгоритмы поиска кратчайшего пути 5
1.4.1 Алгоритм Дейкстры 5
1.4.2 Алгоритм Беллмана-Форда 5
1.4.3 Алгоритм Флойда-Уоршалла 5
1.4.4 Вывод 5
2 Конструкторский раздел 6
2.1 Структуры данных 6
2.2 Используемые функции 6
2.3 Схемы алгоритмов 7
3 Технологический раздел 9
3.1 Требования к ПО 9
3.2 Тестирование ПО 9
4 Контрольные вопросы 10
Заключение 11
Список использованных источников 12

Введение

Целью данной работы является реализация алгоритма поиска кратчайшего пути в графовой структуре.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- проанализировать графовые структуры;
- выполнить постановку задачи
- описать используемые структуры данных;
- описать реализуемый алгоритм;
- описать требования к разрабатываемому ПО;
- реализовать описанные алгоритмы;
- протестировать разработанное ПО.

1 Аналитический раздел

В данном разделе представлены базовые сведения о графовых структурах, обоснован выбор используемой графовой структуры и описано техническое задание.

1.1 Техническое задание

Задана система двусторонних дорог. Определить, можно ли, построив еще три новые дороги, из заданного города добраться до каждого из остальных городов, проезжая расстояние не более T единиц.

1.2 Теоритические сведения

Граф – это конечное множество вершин и ребер, соединяющих их:

$$G =< V, E >, \tag{1.1}$$

где V – конечное непустое множество вершин; E – множество ребер (пар вершин).

Если пары E (ребра) имеют направление, то граф называется ориентированным (орграф), если иначе - неориентированный (неорграф). Если в пары E входят только различные вершины, то в графе нет петель. Если ребро графа имеет вес, то граф называется взвешенным. Степень вершины графа равна числу ребер, входящих и выходящих из нее (инцидентных ей). Неорграф называется связным, если существует путь из каждой вершины в любую другую.

1.3 Представление графа в памяти

1.3.1 Матрица смежностей

Графы в памяти могут представляться различным способом. Один из видов представления графов – это матрица смежности $B(n \times n)$; В этой матрице элемент $b[i,j] = 1$, если ребро, связывающее вершины V_i и V_j существует и $b[i,j] = 0$, если ребра нет. У неориентированных графов матрица смежности всегда симметрична.

1.3.2 Список смежностей

Во многих случаях удобнее представлять граф в виде так называемого списка смежностей. Список смежностей содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с этой вершиной. Каждый элемент ($ZAP[u]$) списка смежностей является записью, содержащей данную вершину и указатель на следующую запись в списке (для последней записи в списке этот указатель – пустой). Входы в списки смежностей для каждой вершины графа хранятся в таблице (массиве) ($BEG[u]$)

1.3.3 Вывод

Для решения поставленной задачи было решено хранить матрицу в виде матрицы смежности, так как работа осуществляется с взвешенным неориентированным графом. Ребер в рассматриваемых графах достаточно много, так что реализация списком не будет выигрывать по памяти.

1.4 Алгоритмы поиска кратчайшего пути

1.4.1 Алгоритм Дейкстры

Поиск кратчайших путей до всех вершин из одной указанной вершины для взвешенного орграфа (имеющего значение, т.е. вес или стоимость, ребра) с неотрицательными ребрами осуществляется с использованием алгоритма Дейкстры. Алгоритм Дейкстры основан на выборе для включения в путь всякий раз той вершины, которая имеет наименьшую оценку кратчайшего пути (по весам ребер), то есть наименьший путь до этой вершины из всех возможных путей.

1.4.2 Алгоритм Беллмана-Форда

Алгоритм Беллмана-Форда позволяет решить задачу о поиске кратчайших путях из одной выбранной вершины ко всем остальным вершинам при любых весах ребер, в том числе и отрицательных. Сначала ищется путь от выбранной вершины ко всем вершинам, связными с ней (аналогично поиску в ширину), а затем – кратчайший путь ко всем остальным вершинам, с попыткой последовательно пройти к ним, т.е. сначала через первую вершину, затем через вторую, через третью и так далее до последней вершины. Кроме того алгоритм возвращает TRUE, если в графе нет цикла отрицательного веса, достижимого из данной вершины.

1.4.3 Алгоритм Флойда-Уоршалла

Для поиска кратчайших путей между всеми вершинами используется алгоритм Флойда-Уоршалла. По алгоритму Флойда-Уоршалла сначала ищется кратчайший путь от одной вершины ко всем вершинам, доступным из нее, затем проводятся те же действия, но пытаясь пройти от этой вершины ко всем доступным из нее, проходя каждый раз через новую вершину (сначала через первую, затем – через вторую и т.д.). Таким образом обрабатываются все вершины.

1.4.4 Вывод

Для решения реализуемым программным обеспечением поставленной задачи необходимо использовать алгоритм Дейкстры, так как осуществляется поиск кратчайших путей до всех вершин из данной, а также веса графов положительные.

2 Конструкторский раздел

В данном разделе описаны используемые структуры данных, приведен листинг используемых функций, а также описан алгоритм Дейкстры.

2.1 Структуры данных

Используемые структуры данных представлены ниже, на листинге 2.1

Листинг 2.1 — Используемые структуры данных

```
1  // Структура очереди
2
3  typedef struct node
4  {
5      int key;
6      struct node *next;
7  } node_t;
8
9
10 typedef struct
11 {
12     node_t *first;
13     node_t *last;
14 } queue_t;
15
16 // Структура графа
17
18 typedef struct graph
19 {
20     unsigned int size;
21     int **adjmat;
22 } graph_t;
```

2.2 Используемые функции

Ниже, на листинге 2.2, представлены функции для работы с очередью.

Листинг 2.2 — Функции для работы с очередью

```
1  queue_t queue_empty(void);
2
3  void queue_push(queue_t *queue, int key);
4
5  int queue_pop(queue_t *queue);
6
7  void queue_destroy(queue_t *queue);
```

Ниже, на листинге 2.3, представлены функции для работы с графом.

Листинг 2.3 — Функции для работы с графом

```
graph_t graph_init(unsigned int vertices);

void graph_output(const graph_t *graph, const char *graphname);

void graph_free(graph_t *graph);

int graph_input(graph_t *graph, const char *filename);

void graph_output(const graph_t *graph, const char *out);

void debug_out(const graph_t *graph);

bool graph_reachable(const graph_t *graph, int T, int src);
```

2.3 Схемы алгоритмов

Ниже, на рисунке 2.1, представлена схема алгоритма Дейкстры.

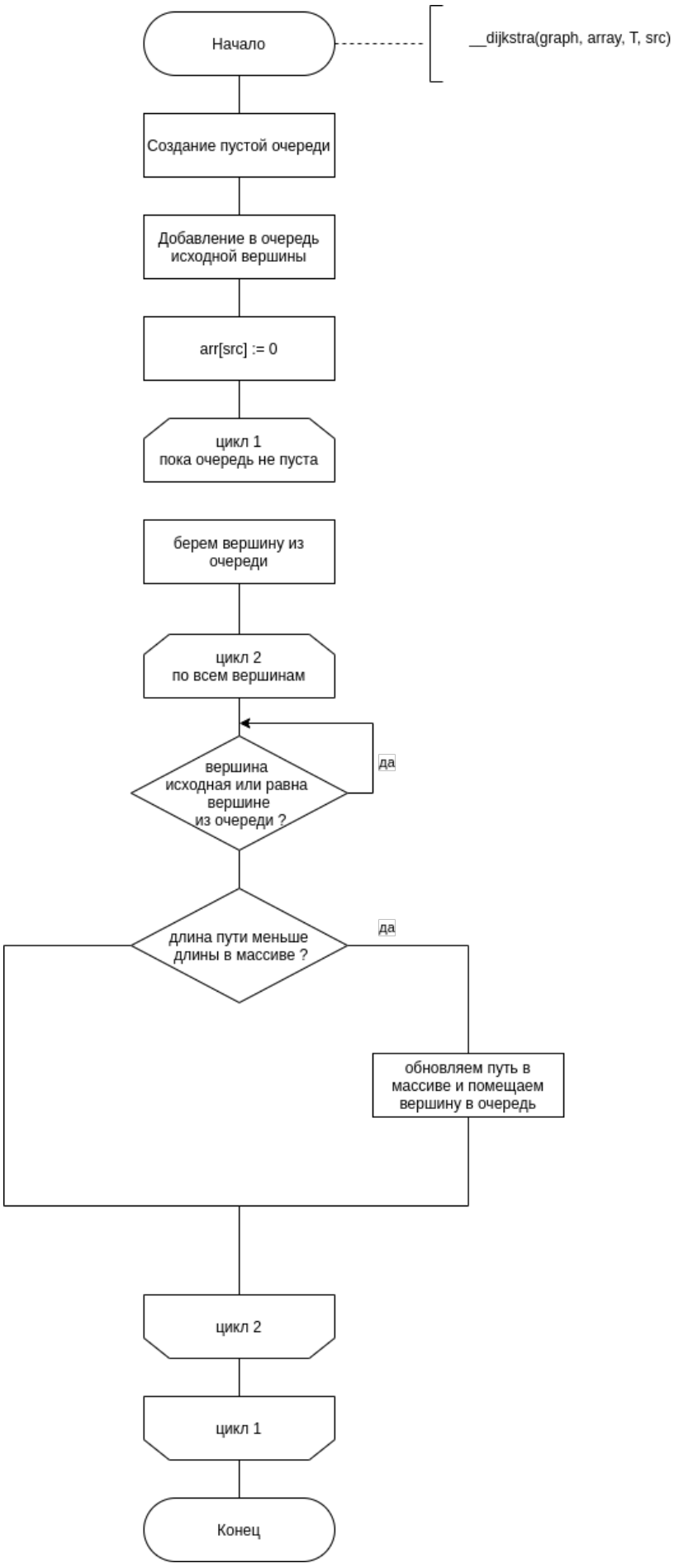

Рисунок 2.1 — Алгоритм Дейкстры

3 Технологический раздел

В данном разделе представлены предъявляемые к ПО требования, а также приведены функциональные тесты к разработанному ПО.

3.1 Требования к ПО

К разрабатываемому ПО предъявлены следующие требования:

- На вход программе подается имя текстового файла, содержащего размер графа и всевозможные связи, а также максимальная длина пути и длины достраиваемых дорог.
- На выход программа синтезирует графическое представление введенного графа, а также графа с достроенными дорогами и результат: возможно ли добраться до каждой вершины из исходной за T.

3.2 Тестирование ПО

Ниже, на таблице 3.1, представлены функциональные тесты к разрабатываемому ПО.

Таблица 3.1 — Функциональные тесты

Входные данные	Выходные данные	Ожидаемый результат
Пустой файл	5 graph.txt	Incorrect input
Некорректный файл	5 g.jpg	Incorrect input
Некорректный размер графа	5 graph.txt	Incorrect input
Некорректные ребра графа	5 Incorrect input	Incorrect input
Некорректные вершины графа	5 graph.txt	Incorrect input
Некорректный максимальный путь	-10	Incorrect input
Исходная вершина больше размера графа	10 graph.txt 222	Incorrect input
Обычный тест	10 graph.txt 0	Reachable

Все тесты пройдены успешно. [1]

4 Контрольные вопросы

1) Что такое граф?

Граф - это конечный набор вершин и соединяющих их рёбер. Если ребра имеют направление, то граф называется ориентированным.

2) Как представляются графы в памяти?

В зависимости от поставленной задачи графы могут представляться в виде списка вершин и матрицы смежности этих вершин, либо с помощью списков смежности. Выбор зависит от соотношения между количеством вершин и количеством дуг в графе.

3) Какие операции возможны над графами?

В графах можно производить поиск отдельных вершин или рёбер, находить кратчайшие расстояния между вершинами, определять компоненты связности графа, а также его ацикличность.

4) Какие способы обхода графов существуют?

Обход в глубину (DFS - Depth First Search). При таком обходе, начиная с произвольной вершины v_0 ищется ближайшая смежная вершина v, для которой, в свою очередь, осуществляется поиск в глубину (т.е. снова ищется ближайшая, смежная с ней вершина) 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 $v = v_0$.

Обход в ширину (BFS - Breadth First Search). Обработка вершины v осуществляется путем просмотра сразу всех новых соседей этой вершины. При этом полученный путь является кратчайшим путем из одной вершины в другую.

5) Где используются графовые структуры?

Графовые структуры активно применяются для представления сетевых отношений в базах данных. Также распростанено их пременение в нейронных сетях.

6) Какие пути в графе Вы знаете?

Путь — последовательность вершин графа, соеденeнных ребрами. Простой путь — путь без повторения вершин. Цепь — путь без повторения ребер. Простая цепь — цепь без повторения вершин. Цикл — путь, начальная и конечная вершина которого совпадают. Простой цикл — цикл, который не проходит дважды по одной вершине. Гамильтонов путь (цикл) — простая цепь (простой цикл), содержащая(ий) все вершины графа без повторений.

7) Что такое каркасы графа?

Каркас графа (остовное дерево) — связный подграф, содержащий все вершины графа и не имеющий циклов. Количество ребер в каркасе связного графа всегда на единицу меньше количества вершин графа.

Заключение

В результате выполнения работы был реализован алгоритм поиска кратчайшего пути в графе. Были проанализированны графовые структуры, разработаны схемы алгоритмов и описаны используемые структуры данных. Также было произведено тестирование ПО.

Были сделаны следующие выводы:

Хранение графа в виде списков смежности подходит для задач, в которых средняя степень вершины не будет зависеть от числа вершин. Или, другими словами, число ребер будет намного меньше, чем половина от квадрата числа вершин. В случае с поставленной задачей, выгоднее хранить граф в виде матрицы смежности, так как число ребер в большинстве случаев довольно велико.

Выбирать матрицу смежности для хранения графа оправдано в тех случаях, когда число ребер будет близко к числу ребер в полном графе.

Выбор конкретного алгоритма обработки графа зависит от его структуры. При большом числе вершин, но малом числе ребер следует выбирать алгоритмы с временной сложностью $O(N \times M)$ (алгоритм Беллмана - Форда), $O(N \times log(N + M))$ (алгоритм Дейкстры с использованием доп. структур). А если же обрабатываемый граф содержит ребра практически между всеми парами вершин, то будут хорошо работать алгоритмы со сложностью $O(N^2)$ (алгоритм Дейкстры со списком), $O(N^3)$ (алгоритм Флойда-Уоршелла) и $O(N \times M + N^2 * logN)$ (алгоритм Джонсона).

В жизни реализованная задача может применяться для проектировании городской инфраструктуры при планировании застройки города администрацией. Информация о самых используемых, густонаселенных или востребованных мест служит отправной точкой для решения проблемы застройки.

Список использованных источников

1. <title>. — (дата обращения: ...). <url>.