**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Построение и анализ алгоритмов»**

**Тема: «Алгоритмы на графах»**

Студент гр. 6381 \_\_\_\_\_\_\_\_\_\_\_\_\_ Афийчук И.И.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_ Филатов А.Ю.

Санкт-Петербург

2018

**Цель работы.**

Целью данной лабораторной работы является применение на практике алгоритмов поиска кратчайшего пути в графе.

**Условия задания.**

**Задание 2.1:**

Разработайте программу, которая решает задачу построения пути в графе при помощи **жадного алгоритма**. Жадность в данном случае понимается следующим образом: на каждом шаге выбирается последняя посещённая вершина. Переместиться необходимо в ту вершину, путь до которой является самым дешёвым из последней посещённой вершины. Каждая вершина в графе имеет буквенное обозначение ("a", "b", "c"...), каждое ребро имеет неотрицательный вес.  
Пример входных данных:

**a e  
a b 3.0  
b c 1.0  
c d 1.0  
a d 5.0**

**d e 1.0**

В первой строке через пробел указываются начальная и конечная вершины. Далее в каждой строке указываются ребра графа и их вес.

В качестве выходных данных необходимо представить строку, в которой перечислены вершины, по которым необходимо пройти от начальной вершины до конечной. Для приведённых в примере входных данных ответом будет:

**abcde**

**Задание 2.2:**

Разработайте программу, которая решает задачу построения кратчайшего пути в графе **методом А\***. Каждая вершина в графе имеет буквенное обозначение ("a", "b", "c"...), каждое ребро имеет неотрицательный вес. В качестве эвристической функции следует взять близость символов, обозначающих вершины графа, в таблице ASCII.  
Пример входных данных:

**a e  
a b 3.0  
b c 1.0  
c d 1.0  
a d 5.0**

**d e 1.0**

В первой строке через пробел указываются начальная и конечная вершины. Далее в каждой строке указываются ребра графа и их вес.

В качестве выходных данных необходимо представить строку, в которой перечислены вершины, по которым необходимо пройти от начальной вершины до конечной. Для приведённых в примере входных данных ответом будет:

**ade**

**Описание жадного алгоритма**

Работа алгоритма основана на принятии локально оптимальных решений при выборе вершины для перехода. Все вершины и ребра записываются в матрицу смежности (вектор векторов float). Затем, пока текущая вершина не равна конечной вершине, в каждой строчке находится минимальное ребро и соседняя вершина, в которую ведет это ребро записывается в вектор выходных данных. Таким образом, на каждом этапе работы алгоритма будет выбрана вершина, смежная с последней обработанной и имеющая наименьшую стоимость перехода, что обеспечивает жадность.

**Описание алгоритма А\***

Работа алгоритма основана на поиске по первому наилучшему совпадению, т.е. расширению наиболее перспективных узлов (в данном случае имеющих минимальную стоимость полного пути), выбираемых в соответствии с эвристической функцией (стоимость перехода (вес ребра) + значение функции эвристической оценки расстояния от рассматриваемой вершины до конечной (в данном алгоритме используется разница между кодами ASCII вершин)). Начиная с начальной вершины, из которой необходимо найти путь в конечную, происходит вставка всех смежных вершин в очередь с приоритетами. В качестве приоритета используется значение эвристической функции. Таким образом, за счёт сортировки приоритетов от меньшего к большому, на каждом этапе работы алгоритма будет выбрана вершина, имеющая наименьшую цену полного пути, то есть являющуюся наиболее перспективной, что и обеспечивает поиск оптимального решения.

**Описание функций и структур данных.**

**lab2\_1.cpp:**

Структуры данных:

**vector<vector <float>> matrix** - матрица смежности;

**char start** - начальная вершина;

**char finish** - конечная вершина;

**float weight** – вес;

**vector <char> result** - массив выходных данных;

Функции:

**void PrintMatrix(vector<vector<float>>& matrix**) – печать матрицы;

**int Greedy (vector<vector<float>>& matrix, vector <char>& result, char& start, char& finish)** – жадный алгоритм;

**float MaxElement(vector<vector<float>>& matrix)** – нахождение максимального элемента;

**lab2\_2.cpp:**

Структуры данных и методы:

**struct Info** { - информация о имени вершины в которую есть путь из данной вершины и её вес;

**Info(char name, float weight): name(name), weight(weight) {};**

**char name;**

**float weight;**

**};**

**class Vertex {** - вершина с прилегающим к ней списком смежных вершин и их путь;

**public:**

**Vertex(char name) : name(name){};**

**char get\_name() const {**

**return name;**

**}**

**list<Info> list\_edge** - список всевозможных путей, в другие вершины + вес;

**private:**

**char name;**

**};**

**class Info\_prior {** - класс приоритетов для приоритетной очереди;

**public:**

**Info\_prior(Vertex vert, float prior): vert(vert), prior(prior){};**

**Vertex vert;**

**float prior;**

**};**

**class Adjacency\_list {** - список смежности;

**public:**

**Adjacency\_list() { }**

**Vertex& at(char a)** – взятие по индексу;

**float heuristic(Info temp, Vertex last)** - эвристической функция;

**void search\_result (Vertex first, Vertex last)** – алгоритм а\*;

**void print\_result(Vertex first, Vertex last, map<char, char>& prevV\_nextV)** – вывод выходных данных;

**priority\_queue<Info\_prior> queue** - приоритетная очередь

**vector<Vertex> list\_vertex** – вектор вершин;

**map <char, char> prevV\_nextV** – контейнер, в котором хранятся пары из вершин: из которой ушли и в которую пришли;

**map <char, float> vertex\_to\_weight** – контейнер, в котором хранятся пары из вершины и стоимости пути до нее из начальной вершины;

};

**Тестирование.**

1. **Тест 1:**

Входные данные:

**a e  
a b 3.0  
b c 1.0  
c d 1.0  
a d 5.0**

**d e 1.0**

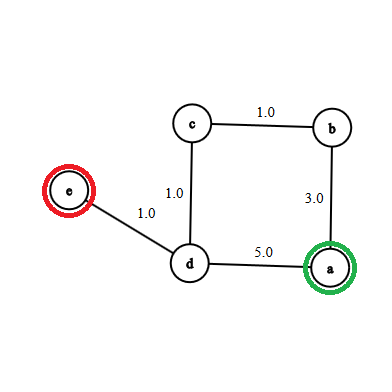
****

Рис. 1. Граф теста 1.

- Жадный алгоритм: **abcde**

- Алгоритм A\*: **ade**

1. **Тест 2:**

Входные данные:

**a e**

**a e 15.0**

**a b 1.0**

**c e 12.0**

**b d 5.0**

**b d 7.0**

**d e 1.0**

**b c 8.0**

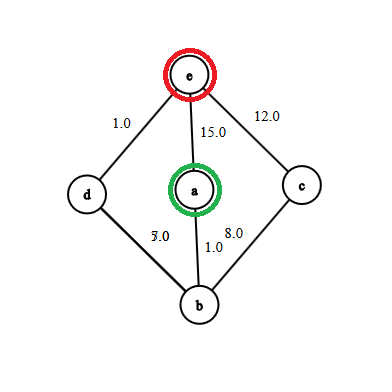
****

Рис. 2. Граф теста 2.

- Жадный алгоритм: **abde**

- Алгоритм A\*: **abde**

1. **Тест 3:**

Входные данные:

**a f**

**a b 1.0**

**c b 5.0**

**d e 4.0**

**a e 7.0**

**a c 10.0**

**c d 11.0**

**e f 3.0**

**b f 16.0**

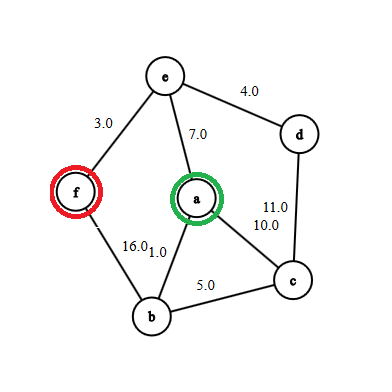
****

Рис. 3. Граф теста 3.

- Жадный алгоритм: **abcdef**

- Алгоритм A\*: **aef**

**Вывод**

В процессе выполнения лабораторной работы был реализован жадный алгоритм для поиска пути в графе и алгоритм А\* для поиска кратчайшего пути в графе.