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: Жадный алгоритм и A*

Студент гр. 8382 Гордиенко А.М.

Преподаватель Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить и реализовать жадный алгоритм поиска кратчайшего пути в графе и алгоритм A* в графе между двумя заданными вершинами.

Постановка задачи.

1) Разработайте программу, которая решает задачу построения пути в ориентированном графе при помощи жадного алгоритма. Жадность в данном случае понимается следующим образом: на каждом шаге выбирается последняя посещенная вершина. Переместиться необходимо в ту вершину, путь до которой является самым дешевым из последней посещенной вершины. Каждая вершина в графе имеет буквенное значение (a, b, c, ...), каждое ребро имеет неотрицательный вес.

Пример входных данный:

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 приведенных в примере входных данных ответом будет abcde.

2) Разработайте программу, которая решает задачу построения кратчайшего пути в ориентированном графе методом A*. 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 начальная и конечная вершины.

В качестве выходных данных необходимо представить строку, в которой перечислены вершины, по которым необходимо пройти от начальной вершины до конечной.

Для приведенных выше входных данных ответом будет ade.

Индивидуальное задание.

Вариант 8. Перед выполнением A* выполнять предобработку графа: для каждой вершины отсортировать список смежных вершин по приоритету.

Описание алгоритма.

Жадный алгоритм.

Для удобства в начале работы жадного алгоритма поиска пути в ориентированном графе, список ребер сортируется по не убыванию их весов. Алгоритм начинает поиск из заданной вершины. Текущая просматриваемая вершина добавляется в список просмотренных.

В отсортированном списке ребер выбирается первое, которое начинается в просматриваемой вершине, если эта вершина не просмотрена, то текущей вершиной становится та, к которой ведет это ребро. Если вершина уже просмотрена, то выбирается следующее ребро. Если в какой-то момент из текущей вершины нет путей, то происходит откат на шаг

назад, и в предыдущей вершине выбирается другое ребро, если это возможно.

Алгоритм заканчивает свою работу, когда текущей вершиной становится заданная искомая, или когда были просмотрены все ребра, которые начинаются из исходной вершины.

*A**.

Поиск начинается из исходной вершины. В текущие возможные пути добавляются все ребра из начальной вершины. Происходит выбор минимального пути, где учитывается эвристическая близость вершины к искомой (в нашем случае это близость в таблице ASCII), если в выбранном пути последняя вершина уже была просмотрена, то этот путь удаляется из открытого списка, и снова происходит выбор минимального пути.

Из всех ребер выбирается те, которые начинаются из последней вершины в этом пути. Эта вершина добавляется к этому пути, и новый путь заносится в список возможных путей, с увеличением стоимости, равной переходу по этому ребру. Когда были выбраны все ребра, которые начинаются из последней вершины в этом пути, то эта вершина добавляется в закрытый список, а сам путь удаляется из открытого списка путей. Дальше снова происходит выбор минимального пути.

Алгоритм заканчивает работу, когда достигнута искомая коненая вершина.

Оценка сложности.

Сложность Жадного алгоритма.

По памяти.

Так как в памяти хранится только список ребер, то сложность составляет O(|E|).

По времени.

Сложность составляет O(|V| * |E|).

Сложность A*.

По памяти.

Так же, как и в жадном алгоритме, так как структура хранения такая же O(|E|).

По времени.

Без оптимизации сложность будет квадратичная O(|V| * |V|). С оптимальной эвристикой O(|V| * log|V|).

Описание функций и структур.

Общие.

Структура-ребро графа. Имеет поля начало ребра from, конец ребра to, и вес(цену) ребра - weight.

```
struct Edge {
	char from;
	char to;
	double weight;
};
```

Структура-путь графа. Имеет поле самого пути - path, длину (суммарный вес ребер) этого пути - len, и наименование конечной вершины пути end.

```
struct Step {
	string path;
	double len;
	char end;
};
```

Функции void readGraph() для считывания графа из потока.

Функция void Search() выполняет соответствующий поиск пути в графе.

A*.

Имеет функции is_visible(char value) ищет ребро к вершине с именем value.

Жадный алгоритм.

Имеет функции void solve(), которая запускает поиск решение.

Тестирование.

Входные данные.

a g

a b 1

b c 1

c f 2

e b 1

a d 1.49

d e 1.39

e g 2

Результат поиска жадного алгоритма.

abcfg

Результат поиска алгоритма A*.

adeg

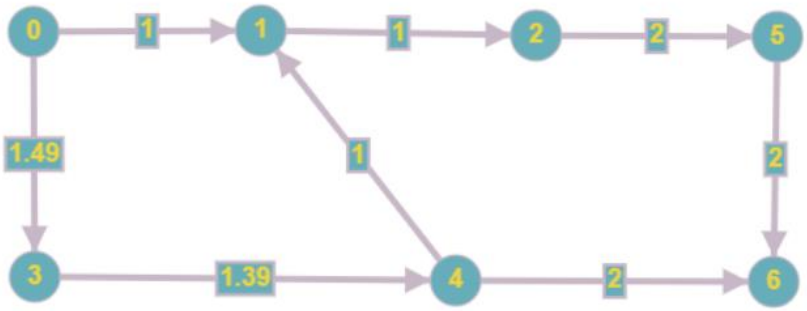

Рисунок 1. Графическое представление графа из тестирования.

Выводы.

В ходе работы были изучены и реализованы два алгоритма. На основе тестирования можно сделать выводы об исполнении алгоритмов. Видно, что жадный алгоритм быстро находит решение, которое не всегда является верным. В то время как алгоритм A* уступает по времени, но находит точно верное решение, если оно есть.