МИНОБРНАУКИ РОССИИ САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА) Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Построение и анализ алгоритмов»

Тема: Жадный алгоритм и А*

Студент гр. 7304	Нгуен К.Х.
Преподаватель	Филатов А.К

Санкт-Петербург 2019

Цель работы.

Исследование алгоритмов Greedy и A * для поиска путей в графах, реализация программы для решения задач с использованием этих алгоритмов

Задание

Жадный алгоритм

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 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.

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.

А* алгоритм

Разработайте программу, которая решает задачу построения кратчайшего пути в ориентированном 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.

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

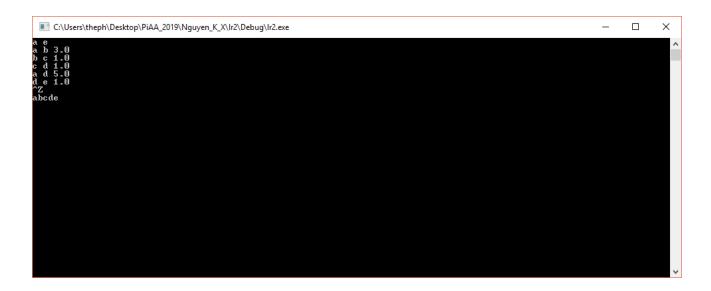
В качестве выходных данных необходимо представить строку, в которой перечислены вершины, по которым необходимо пройти от начальной вершины до конечной.

Ход работы

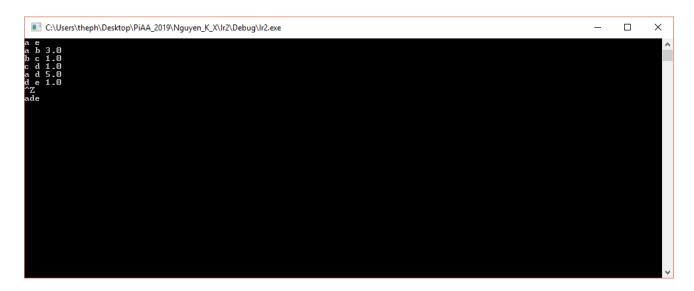
- 1. Создайте класс Edge для хранения информации о ребрах графа. Для каждого ребра нам нужно сохранить начальную и конечную точки, а также вес ребра
- 2. Создайте класс Peak для хранения информации о пиках графика. Для каждого пика мы сохраняем его имя, gSroce и fScore (фактическое расстояние и эвристическую точку для алгоритма А *), а также их соседние пики.
- 3. Напишите входную часть, которая получает ввод с клавиатуры, постройте векторы для их сохранения.
- 4. Напишите функцию distBetween, чтобы просмотреть вектор ребер и вернуть расстояние между двумя заданными пиками.
- 5. Напишите функцию calHeuristic для расчета эвристической точки (для алгоритма A *)
- 6. Напишите функцию Greedy, которая применяет алгоритм жадного поиска на графике. Он получает начальный пик, конечный пик и набор ребер. В результате этой функции поле «сатеFrom» пиков будет заполнено. Нам нужно создать еще одну функцию для отслеживания от конечного пика и возврата пути.
- 7. Напишите функцию «reconstruct», которая переводит указатель на конечный пик и возвращает путь.
- 8. Для алгоритма А * нам просто нужно изменить функцию Greedy на функцию Astar, которая применяет алгоритм А * к графу, остальные процессы такие же как в случае с Жадным алгоритмом.

Экспериментальные результаты.

Greedy algorithm



A* algorithm



Выводы.

В результате работы программы был исследован алгоритм работы поиска с возвратом при помощи рекурсии. Также была написана программа, реализующая заполнение квадратной области минимальным количеством квадратов.

Мы видим, что алгоритм A * возвращает лучший результат, чем алгоритм Greedy (более короткий путь).