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. 1304 Макки К.Ю

Преподаватель _____ Шевелева А.М.

Санкт-Петербург
2023

Цель работы.

Изучить жадный алгоритм и A*, применить их к задаче построения пути в ориентированном графе.

Задание.

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 дешёвым из последней посещенной вершины. Каждая вершина в графе имеет буквенное обозначение ("a", "b", "c"...), каждое ребро имеет неотрицательный вес.

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.

Выполнение работы.

В ходе работы было определено, что все необходимые для решения функции можно сделать методами класса *Solution*. Рассмотрим методы и переменные класса:

1) *__init__(self)* Конструктор класса где происходит инициализация переменных:

graph — словарь, представляющий граф, где ключ - вершина графа, а значение - список пар (вершина, вес ребра), соответствующих ребрам, исходящим из данной вершины.

start_node — строка, содержащая начальную вершину графа.

end_node — строка, содержащая конечную вершину графа.

correct_path — булева переменная, указывающая был ли найден правильный путь между начальной и конечной вершинами графа.

answer — строка, содержащая путь от начальной вершины до конечной вершины графа.

2)input_to_graph(self): Метод для чтения входных данных с консоли и заполнения графа. Входные данные содержат информацию о начальной и конечной вершинах, а также список смежности графа с указанием расстояния между вершинами. Метод читает входные данные, обрабатывает их и сохраняет в виде словаря с ключами - вершинами и значениями - списками смежности для каждой вершины.

3) output(self): метод, который выводит результат в переменной answer.

4) heuristic(self, cur_node: str): Метод, который вычисляет эвристическое расстояние между текущей вершиной и конечной вершиной, используя разницу в значениях кодов символов. Эвристическая функция используется в алгоритме A* для оценки оставшегося расстояния до конечной вершины.

5) greedy_algorithm(self, cur_node: str, cur_way: str): Рекурсивный метод для поиска пути между начальной и конечной вершинами с использованием жадного алгоритма. Сначала соседние вершины текущей вершины сортируются по расстоянию в порядке возрастания, затем метод проверяет, является ли текущая вершина конечной вершиной. Если да, то он устанавливает результат в переменную answer и завершает поиск. В противном случае он рекурсивно исследует соседние вершины текущей вершины. Жадный алгоритм выбирает следующую вершину на основе ее расстояния до конечной вершины без учета стоимости достижения этой вершины.

6) a_star_algorithm(self): Метод, который решает задачу поиска кратчайшего пути от начальной вершины до конечной вершины, используя алгоритм A*. Метод создает очередь, которая используется для хранения кандидатов на следующую вершину. Затем он создает словари came_from и weight, которые будут использоваться для хранения пути и веса пути от начальной вершины к каждой вершине. Алгоритм работает следующим образом: из очереди извлекается вершина с наименьшим весом и проверяется, является ли она конечной вершиной. Если да, то метод завершает поиск и

использует came_from для восстановления пути от конечной вершины до начальной вершины. Если нет, то метод рассматривает каждого соседа текущей вершины и обновляет вес пути и значение эвристической функции для каждого соседа. Каждый сосед добавляется в очередь и в словари came_from и weight

7) *solve_greedy(self)* Метод для запуска поиска пути с использованием жадного алгоритма.

8) *solve_a_start(self)* Метод для запуска поиска пути с использованием алгоритма A*.

Описание оставшихся переменных:

1) input_graph - словарь, используемый для временного хранения ребер графа, вводимых пользователем
2) cur_node - текущая вершина, используемая при поиске пути в графе
3) next_node - следующая вершина, используемая при поиске пути в графе
4) distance - расстояние между текущей вершиной и следующей вершиной, используемое при поиске пути в графе
5) queue - очередь, используемая в алгоритме A*
6) came_from - словарь, содержащий информацию о том, из какой вершины пришли в каждую вершину
7) weight - словарь, содержащий информацию о весе пути от начальной вершины до каждой другой вершины
8) priority - приоритет, используемый в алгоритме A* для выбора следующей вершины
9) new_cost - новый вес пути от начальной вершины до следующей вершины, используемый в алгоритме A* для выбора следующей вершины

Выводы.

В рамках исследования основных алгоритмов на графах были рассмотрены жадный алгоритм и A*. Сравнение этих алгоритмов показало, что, в отличие от жадного алгоритма, который выбирает локально наилучший вариант, но не всегда генерирует глобально наилучшее решение, A* использует эвристический подход для решения задачи поиска кратчайшего пути между двумя вершинами в ориентированном графе. Оба алгоритма были успешно протестированы на платформе Stepik и доказали свою эффективность в решении поставленной задачи.