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*

Студентка гр. 9382 _____ Голубева В.П.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2021

Цель работы.

Понять, что такое жадный алгоритм, научиться его реализовывать на примере поиска пути в неориентированном графе.

Задание.

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.

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

Вариант 1. В A* вершины именуются целыми числами (в т.ч. отрицательными)

Описание алгоритма

Был реализован только поиск пути в ориентированном графе жадным алгоритмом.

Граф представлялся в виде словаря, ключи в котором — вершины-источники, а значения — список словарей всех вершин (пары «вершина:вес ребра»), с которыми соединена вершина источник. Пример можно будет увидеть в Приложении Б.

Текущим решением будет список, в котором будут храниться вершины, составляющие путь.

Главный цикл в алгоритме проходит по вершинам в графе. Берёт последнюю вершину из текущего пути и смотрит, путь до какой вершины наименьший. Записывает её в текущий путь, затем удаляет из графа. Таким образом, текущий путь будет содержать вершины, пути до которых минимальны. Если на очередном шаге не удалось найти путь из последней текущей вершины, это значит, что мы не дошли в заданную вершину, нужно «откатиться» и попробовать поискать решение, которое приводит нас к финишу другим путём.

Тестирование

Результаты тестирования можно посмотреть в приложении Б.

Выводы.

Было изучено что такое жадный алгоритм, написана программа, которая его реализует.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab2_1.py

```
path = input().split(' ') #list

vertice = input()
dictionary = {}
while (vertice):
    current = vertice.split(' ')
    if current[0] in dictionary:
        dictionary[current[0]].append(dict.fromkeys([current[1]],
float(current[2])))#if current source vertice alredy in dictionary
    else:
        dictionary[current[0]]=[dict.fromkeys([current[1]],
float(current[2]))]#if current source vertice not in dictionary
    try:
        vertice = input()
    except:
        break
print("Our incoming pathes")
print(dictionary)

current_path = [path[0]]#store a path of vertices
size = 1#current path size

while current_path[size-1]!=path[1]:
    print ("\nCurrent path")
    print(current_path)
    #trying to make path from the last vertice in current path
    try:
        find = dictionary[current_path[size-1]]#list of dictionaries
        print("Trying find greedy path from "+current_path[size-1])
    except:
        print("No path from "+ current_path[size-1]+", delete this
vertice from current_path")
```

```
            current_path.pop()
            size-=1
            continue

        min_size = 1000
        for i in find:# i - dict
            keys = i.keys()# get a keys for current source vertice

            for j in keys:
                if i[j] < min_size:
                    min_size = i[j]
                    min_key = j
                    item = i
        find.remove(item)#remove vertice from dictionary
        print("Find greedy path from "+min_key)
        current_path.append(min_key)
        size+=1

str_result = "\nResult path: "
for i in current_path:
    str_result+=str(i)
print(str_result)
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Входные данные	Выходные данные
a z a b 1 a c 3 b y 6 c y 1 y z 1	Our incoming pathes {'a': [{'b': 1.0}, {'c': 3.0}], 'b': [{'y': 6.0}], 'c': [{'y': 1.0}], 'y': [{'z': 1.0}]} Current path ['a'] Trying find greedy path from a Find greedy path from b Current path ['a', 'b'] Trying find greedy path from b Find greedy path from y Current path ['a', 'b', 'y'] Trying find greedy path from y Find greedy path from z Result path: abyz
a e a b 3.0 b c 1.0 c d 1.0 a d 5.0 d e 1.0	Our incoming pathes {'a': [{'b': 3.0}, {'d': 5.0}], 'b': [{'c': 1.0}], 'c': [{'d': 1.0}], 'd': [{'e': 1.0}]} Current path ['a'] Trying find greedy path from a Find greedy path from b

	Current path ['a', 'b'] Trying find greedy path from b Find greedy path from c Current path ['a', 'b', 'c'] Trying find greedy path from c Find greedy path from d Current path ['a', 'b', 'c', 'd'] Trying find greedy path from d Find greedy path from e Result path: abcde
a d a b 1.0 b c 9.0 c d 3.0 a d 9.0 a e 1.0 e d 3.0	Our incoming pathes {'a': [{'b': 1.0}, {'d': 9.0}, {'e': 1.0}], 'b': [{'c': 9.0}], 'c': [{'d': 3.0}], 'e': [{'d': 3.0}]} Current path ['a'] Trying find greedy path from a Find greedy path from b Current path ['a', 'b'] Trying find greedy path from b Find greedy path from c Current path ['a', 'b', 'c']

	Trying find greedy path from c Find greedy path from d Result path: abcd
a b a b 1.0 a c 1.0	Our incoming pathes {'a': [{'b': 1.0}, {'c': 1.0}]} Current path ['a'] Trying find greedy path from a Find greedy path from b Result path: ab
a g a b 3.0 a c 1.0 b d 2.0 b e 3.0 d e 4.0 e a 3.0 e f 2.0 a g 8.0 f g 1.0	Our incoming pathes {'a': [{'b': 3.0}, {'c': 1.0}, {'g': 8.0}], 'b': [{'d': 2.0}, {'e': 3.0}], 'd': [{'e': 4.0}], 'e': [{'a': 3.0}, {'f': 2.0}], 'f': [{'g': 1.0}]} Current path ['a'] Trying find greedy path from a Find greedy path from c Current path ['a', 'c'] No path from c, delete this vertice from current_path Current path ['a'] Trying find greedy path from a Find greedy path from b Current path

	['a', 'b'] Trying find greedy path from b Find greedy path from d Current path ['a', 'b', 'd'] Trying find greedy path from d Find greedy path from e Current path ['a', 'b', 'd', 'e'] Trying find greedy path from e Find greedy path from f Current path ['a', 'b', 'd', 'e', 'f'] Trying find greedy path from f Find greedy path from g Result path: abdefg