Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

ГРАФЫ

Отчет по лабораторной работе № 8 по дисциплине «Структуры и алгоритмы обработки данных в ЭВМ»

Выполнил: Обучающийся гр. 439-3
(группа)

_____ А. С. Мазовец
(подпись) *(И. О. Фамилия)*

« _____ » _____ 2021г.
(дата)

Проверил: ассистент кафедры АСУ
(должность, ученая степень, звание)

_____ _____ Я. В. Яблонский
(оценка) *(подпись)* *(И. О. Фамилия)*

« _____ » _____ 2021г.
(дата)

Томск 2021

1 Задание на лабораторную работу

1.1 Вариант 4

Напишите программу, которая с помощью алгоритма Дейкстры будет находить кратчайшие пути от фиксированной вершины графа до всех остальных его вершин. Граф задан списками смежности. Предусмотрите ввод данных из файла. После завершения работы с динамическими структурами данных необходимо освободить занимаемую ими память.

2 Листинг программы

```
 1| from math import inf
 2| from collections import defaultdict
 3|
 4| from sys import stdin
 5| input = stdin.readline
 6|
 7| class Node:
 8|     def __init__ (self):
 9|         self.opt_w  = inf
10|         self.viewed = False
11|         self.joints = set()
12|
13|     def addJoint (self, node, weight):
14|         self.joints.add(NodeJ(node, weight))
15|
16| class NodeJ:
17|     def __init__ (self, node, weight):
18|         self.node   = node
19|         self.weight = weight
20|
21| class Graph:
22|     def __init__ (self, count):
23|         self.vertCount  = count
24|         self.jointsList = defaultdict(Node)
25|
26|     def addEdge (self, start, end, w):
27|         self.jointsList[start].addJoint(end, w)
28|         self.jointsList[end].addJoint(start, w)
29|
30|
31| def dijkstra (g, start):
32|     g.jointsList[start].opt_w = 0
33|     neigb = []
34|     curr = start
35|     sn = []
```

```
    for join in g.jointsList[curr].joints:
        if not g.jointsList[join.node].viewed:
            sn.append(join)
    sn = sorted(sn, key=lambda i: i.weight)
    for i in sn:
        neigb.append(i)
                if g.jointsList[i.node].opt_w > i.weight + g.jointsList[curr].opt_w:
                    g.jointsList[i.node].opt_w = i.weight + g.jointsList[curr].opt_w
    g.jointsList[curr].viewed = True
    while len(neigb) > 0:
        curr = neigb.pop(0).node
        sn = []
        for join in g.jointsList[curr].joints:
            if not g.jointsList[join.node].viewed:
                sn.append(join)
        sn = sorted(sn, key=lambda i: i.weight)
        for i in sn:
            neigb.append(i)
                 if g.jointsList[i.node].opt_w > i.weight + g.jointsList[curr].opt_w:
                     g.jointsList[i.node].opt_w = i.weight + g.jointsList[curr].opt_w
        g.jointsList[curr].viewed = True
    for i in g.jointsList:
                    print("{:d} <-> {:d} :: opt weight {:d}".format(start, i, g.jointsList[i].opt_w))

    for i in g.jointsList:
        opt_way = []
        j = i
        if g.jointsList[i].opt_w == 0:
                            print("{:d} <-> {:d} :: opt way".format(start,start))
            print([start, start])
            print("")
        else:
            while j != start:
                for join in g.jointsList[j].joints:
                    if join.weight == g.jointsList[j].opt_w - g.jointsList[join.node].opt_w:
                        opt_way.append(join.node)
                        j = join.node
                        break
            opt_way.insert(0,i)
            print("{:d} <-> {:d} :: opt way".format(start,i))
            print(list(reversed(opt_way)))
            print("")

f = open ("I", 'r')
n = int (f.readline ())
g = Graph (n)
```

```
 82| for i in range (n):
 83|     a, b, w = map (int, f.readline ().split ())
 84|     g.addEdge (a, b, w)
 85| f.close ()
 86|
 87| dijkstra (g, 4)
```

3 Пример решения задачи

3.1 Входной файл

Построчно вводятся данные из 3 натуральных чисел — вершины ребра и длина ребра графа.

```
 1| 9
 2| 1 2 7
 3| 1 3 9
 4| 1 6 14
 5| 2 3 10
 6| 2 4 15
 7| 3 4 11
 8| 3 6 2
 9| 4 5 6
10| 5 6 9
```

3.2 Выходной файл

Сначала выводятся расчетные длины кратчайших путей от исходной вершины, далее выводится сам путь в виде вектора вершин, в порядке их оптимального прохождения.

```
 1| 4 <-> 1 :: opt weight 20
 2| 4 <-> 2 :: opt weight 15
 3| 4 <-> 3 :: opt weight 11
 4| 4 <-> 6 :: opt weight 13
 5| 4 <-> 4 :: opt weight 0
 6| 4 <-> 5 :: opt weight 6
 7|
 8| 4 <-> 1 :: opt way
 9| [4, 3, 1]
10|
11| 4 <-> 2 :: opt way
12| [4, 2]
13|
14| 4 <-> 3 :: opt way
15| [4, 3]
16|
17| 4 <-> 6 :: opt way
18| [4, 3, 6]
```

```
19|
20| 4 <-> 4 :: opt way
21| [4, 4]
22|
23| 4 <-> 5 :: opt way
24| [4, 5]
```

4 Вывод

Изучил и реализовал алгоритм Дейкстры нахождения кратчайших путей в графе. Закрепил знания реализации и работы с графами.