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

ОТЧЕТ
по лабораторной работе №3
по дисциплине «Алгоритмы и Структуры Данных»
Тема: «Алгоритмы на графах»
Вариант 1

Студент гр. 8301 _____ Урбан М.Ф.

Преподаватель _____ Тутуева А.В.

Санкт-Петербург
2020

1. Цель работы

Реализовать алгоритм Дейкстры на графе, представляемом при помощи списков смежности. А также найти наиболее эффективный по стоимости перелет из города ***i*** в город ***j***, используя список возможных авиарейсов, данный в текстовом формате в .txt файле.

2. Описание программы

Идея алгоритма состоит в следующем:

Каждой вершине V графа G сопоставляется метка – минимальное известное расстояние от этой вершины до стартовой вершины S. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Метка самой вершины S полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от S до других вершин пока неизвестны. Все вершины графа помечаются как не посещённые.

Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина V, имеющая минимальную метку. Рассматриваются всевозможные маршруты, в которых V является предпоследним пунктом. Для каждого соседа вершины V, кроме отмеченных как посещённые, рассматривается новая длина пути, равная сумме значений текущей метки V и длины ребра, соединяющего V с этим соседом. Если полученное значение длины меньше значения метки соседа, значение метки заменяется полученным значением длины. Рассмотрев всех соседей, вершина V помечается как посещённая и шаг алгоритма повторяется.

В данной программе используются следующие структуры данных:

- Класс Graph:
  Данный класс реализует граф, основанный на списках смежности. Этот класс хранит: список смежности adjList (динамический массив из контейнеров Map), который помимо номера смежной вершины хранит ещё и вес ребра; список namesList, являющийся связью между названием города и номером вершины; очередь с приоритетами markList, которая является связью между номером вершины, её меткой и информацией о том, посещена вершина или нет.

- Класс List:
  Шаблонный двусвязный список. Используется для хранения введенной информации, списка названий городов, для хранения результирующего маршрута и ещё много чего.

- Класс Map:
  Шаблонный ассоциативный массив. Используется в качестве списка смежности для одновременного хранения смежной вершины и веса ребра, направленного к этой вершине.

- Класс priorityQueue:
  Шаблонная очередь с приоритетами. Функционал очереди дополнен возможностью получать приоритет элементов очереди, а также получать элемент с минимальным приоритетом, не удаляя его из очереди. Используется для хранения меток вершин. Приоритет элементов выступает в качестве метки, а наличие элемента в очереди – в качестве информации о том, была посещена вершина или нет.

3. Оценка временной сложности методов

N – количество строк входной информации
M – количество символов в строке
V – количество вершин (городов)
E – количество ребер (возможных перелетов между городами)

1) ***convert*** имеет временную сложность O($N*M$)

2) ***getUniqueNames*** имеет временную сложность O($N*V$);
$N*(V*1 + V*1) = N*V$

3) ***Graph*** имеет временную сложность O(V);

4) ***makeGraphFromList*** имеет временную сложность O($N^2 + NV + N*log(E)$);
$N*(N + V + V + 2*log(E)) = N^2 + NV + N*log(E)$

5) ***makePath*** имеет временную сложность O($VE+E^3$);
$log(E)+log(E)+V+2V+3V+V+E(E+E(E+log(E)+2V)+V)=$
$= log(E)+V+E^2+VE+ E^3 + ElogE +EV = EV+E*log(E)+ E^3$

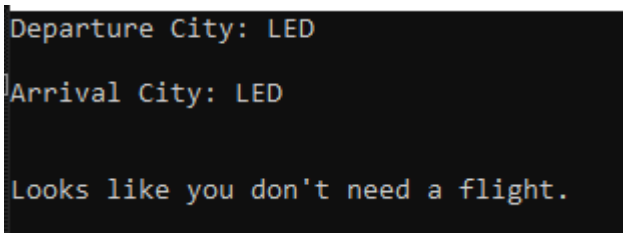
6) ***getShortestPath*** имеет временную сложность O($E^3+V^2*E*log(V)+V*E*log(E)+V*E^2+V^2*E$);
$2V+V*log(V)+V(V+log(V)+E+E(E+V+log(E)+V+V*log(V))+log(V))+VE+ E^3+2V=$
$= E^3+V^2*E*log(V)+V*E*log(E)+V*E^2+V^2*E$

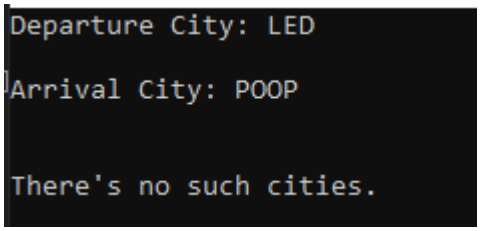
4. Примеры работы

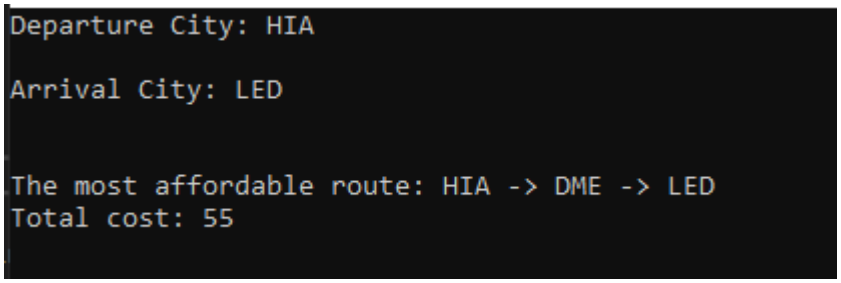
Используем следующую базу:

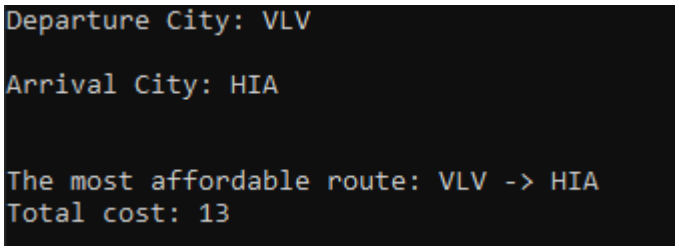
```
LED;DME;10;20
DME;HIA;40;35
LED;HIA;14;N/A
VLV;HIA;13;8
VLV;LED;N/A;20
```

Примеры входных и выходных данных:









5. Листинг, как и прежде, на GitHub