Finding the Shortest Route using Dijkstra's Algorithm

Due May. 7 noon

Objectives

In this assignment, you will implement the classic Dijkstra's shortest path algorithm to find the shortest path between two cities for an airline company.

Background: Dijkstra's Shortest Path Algorithm

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959, is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. This algorithm is often used in routing and as a subroutine in other graph algorithms.

For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. For more information, please refer to the Chapter 14 Graphs in the Prichard text.

Task Description

Your software should provide three types of information: (1) neighboring cities, (2) the ***route*** with the shortest distance between two cities, and (3) the shortest ***distance*** between the cities.

(1) Neighboring cities
For a specified city, your method should retrieve all of the cities that you can reach without intermediate stops. For this output, sorting is not required.

(2) The shortest route
For specified cities (city A and city B), your method should find the shortest route from the city A to the city B. Your method should use the Dijkstra's shortest path algorithm explained in the Prichard text (p.807). ***There can be multiple shortest routes.***

(3) The shortest distance
For the specified cities, your method should return the total distance to fly if the client chooses the shorted route in (2).

Parsing: Example File

The example file that has been provided has multiple tuples of (city A, city B, distance). In the text file, each of the tuples follows the format:
[city A] [city B] [distance]

[city A] and [city B] include the name of the city. This record does not consider the direction between these cities. i.e**. if there is a route from city A to city B, we assume that there is a route from city B to city A, and that the distances are the same.** Students should convert this record to either:

(1) A directed graph representing both directions.
(2) An undirected graph and modify Dijkstra's algorithm.

Test Cases

The test cases listed below are provided to assist you in verifying the correctness of your software. You are also required to test your software with different test cases that you will build yourself. Actual grading will be done by test cases using the same commands; however, the values that the specified input arguments take will be different.

(1) Test case 1

Print all the neighboring cities. These cities do not need to be sorted. Sorting is not required.

```
% java PA5_Test example.txt printNeighbors Munich
Barcelona
Belgrade
Berlin
Brussels
Bucharest
Copenhagen
Hamburg
Istanbul
Kiev
London
Milan
Moscow
%
% java PA5_Test example.txt printNeighbors Budapest
Barcelona
Belgrade
Berlin
Brussels
Bucharest
%
%java PA5_Test example.txt printNeighbors Dublin
Barcelona
Belgrade
Berlin
Brussels
Bucharest
Copenhagen
```

(2) Test case 2

Print the names of the cities on the shortest path between two cities. There can be multiple shortest routes. You may have different answer. You can validate your answer by adding total distance from your path and compare it with total distance provided by example (3).

```
% java PA5_Test example.txt printShortestPath Barcelona Istanbul
Barcelona
Sofia
Istanbul
%
% java PA5_Test example.txt printShortestPath Hamburg Moscow
Hamburg
Berlin
Warsaw
Moscow
%
% java PA5_Test example.txt printShortestPath London Sofia
London
Paris
Sofia
```

(3) Test case 3

Print the shortest distance between two cities.

```
% java PA5_Test example.txt printShortestDistance Barcelona Istanbul
2254
%
%java PA5_Test example.txt printShortestDistance Hamburg Moscow
1919
%
%java PA5_Test example.txt printShortestDistance  London Sofia
2098
%
```

PA5_Test.java, and example file will be posted on the class web site along with this document. **DO NOT MODIFY the main() of PA5_Test.java that has been provided to you.**

Deliverables

Submit a tar ball of your java source code including:

```
Travel.java
Vertex.java
Edge.java
Graph.java
ShortestPath.java
InformationParser.java
```

Please include all files your created for any subclass.

Keep all of your source code in a single flat directory. The skeleton files are provided. Please do not modify PA5_Test.java.

Note: You are required to work as a team in this assignment. You and your teammate should submit only ONE copy of the assignment. Please write down the implementer's name(s) on top of each of the source code.

Grading

This assignment will account for 5% of your final grade. The grading itself will be done on a 50 point scale.

Late Policy

Please check the late policy available from the course web page