Europe by Rail

Prerequisites, Goals, and Outcomes

Prerequisites: Students should have mastered the following prerequisite skills.

Graphs - Knowledge of graph representation
Graph Algorithms - Knowledge of Dijkstra's shortest path algorithm

Goals: This assignment is designed to reinforce the student's understanding of the implementation of a fundamental graph algorithm
Outcomes: Students successfully completing this assignment would master the following outcomes.

Understand graph representation
Understand how to implement Dijkstra's shortest path algorithm

Background

Traveling through Europe by rail is a cheap and effective way to experience the sights, sounds, and culture of a wide array of countries and cities. Generally, travelers purchase rail passes that allow unlimited travel on the rail system. Individual tickets, however, can be purchased.

Description

The program for this assessment calculates the cheapest route between two cities in a mock European rail system. The graph that represents the rail system is pictured below.

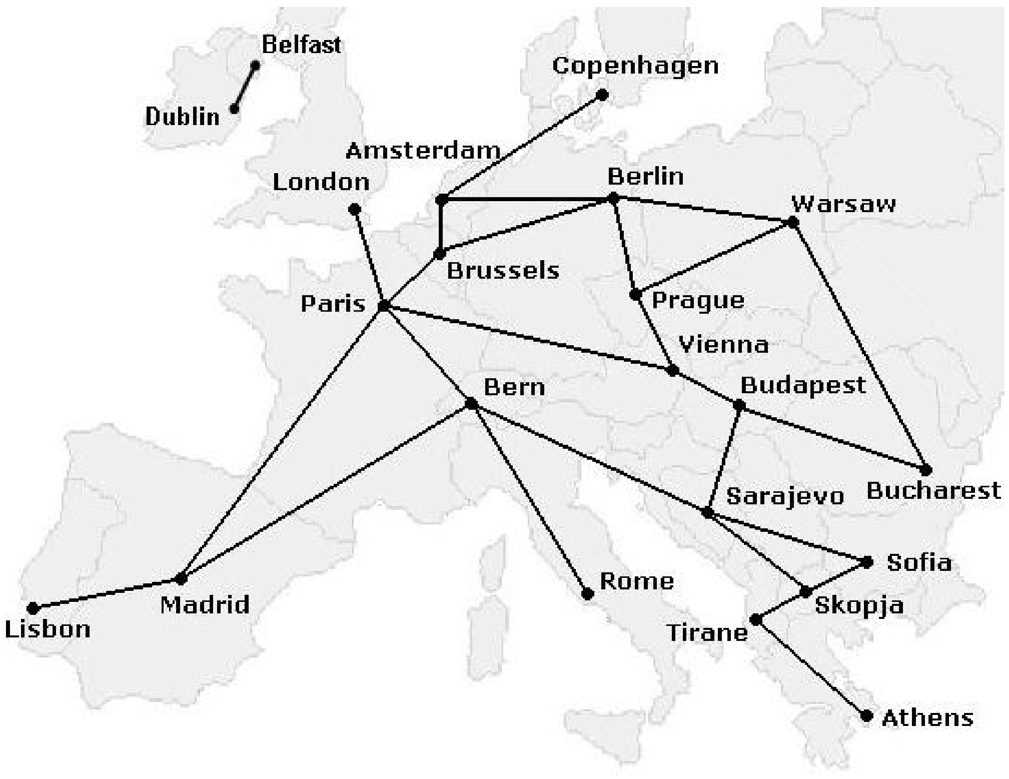
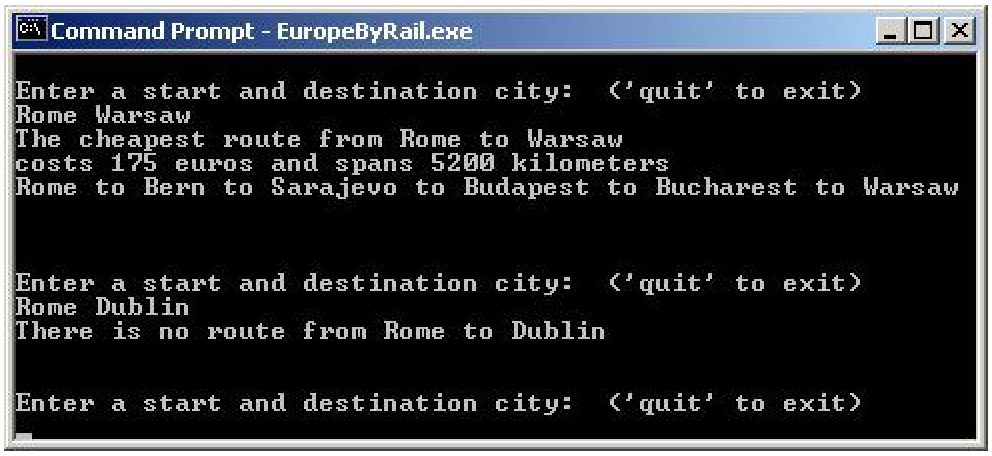


Figure 1 A mock European rail system

In this program, a weighted graph is constructed to represent rail services to and from European cities. Each service from a city has an associated destination city, a fee (in *Euros*), and a distance (in kilometers). The program processes user input of a source city and a destination city. The program then displays the cheapest route from the source city to the destination city. In addition, for each route, the total cost and total distance are displayed. The file *services.txt* contains the data for the available services.

You can use the latest assignment to construct this graph. And then calculate the cheapest path between two cities through the fee between them. Sample output from this program is shown in the figure below.



```
Enter a start and destination city:  ('quit' to exit)
Rome Warsaw
The cheapest route from Rome to Warsaw
costs 175 euros and spans 5200 kilometers
Rome to Bern to Sarajevo to Budapest to Bucharest to Warsaw


Enter a start and destination city:  ('quit' to exit)
Rome Dublin
There is no route from Rome to Dublin


Enter a start and destination city:  ('quit' to exit)
```

Figure 2 Output from a sample solution

The pseudocode of Dijkstra can be as follows:

```
Dijkstra(graph digraph, node first)
    For all nodes v of the graph
        currDist(v)=INFINITY;
    currDist(first)=0;
    toBechecked = all the nodes;
    while toBechecked is not empty
        v = the node with minimum currDist in toBechecked;
        remove v from toBechecked;
        for every u that adjoins v in toBechecked
            if currDist(u)> currDist(v)+weight(edge(vu))
                currDist(u)= currDist(v)+weight(edge(vu)) ;
                predecessor(u)=v;
```

To be more efficient, toBecheck is better to be a heap, you can use your code for the assignment of heap.

Tasks

To complete this assessment, you need to complete the follow function

- the construction of a graph
- the implementation of Dijkstra’s shortest path algorithm.
- Calculate the shortest path between two cities