Project 6

Plan a Flight

Introduction:

In this project, you will determine the shortest/cheapest flight plans for a person wishing to travel between two different cities serviced by an airline (assuming a path exists), using ***Dijkstra's algorithm with priority queues***. You will also calculate the total cost incurred for all parts of the trip. For this project, you will use information from two different input files in order to calculate the trip plan and total cost.

1. **Data File** – This file will contain a sequence of city pairs representing different legs of flights that can be considered in preparing a flight plan. For each leg, the file will also contain a dollar cost for that leg and a time to travel[1].
2. **Query File** – This file will contain a sequence of origin/destination city pairs. For each pair, your program will determine if the flight is or is not possible. If it is possible, it will output to a file the cheapest/shortest flight plan with the total cost for the flight. If it is not possible, then a suitable message will be written to the output file.

The names of the two input files as well as the output file will be provided via command line arguments. The first strings will be the data file name, second string will be query file name and third one will be the output file name

Flight Data:

Consider a flight from Detroit to El Paso. It's possible that there is a direct flight, or it may be the case that a stop must be made in Austin. One stop in Austin would mean the flight would have two legs. We can think of the complete set of flights between different cities serviced by our airline as a directed graph. An example of a directed graph is given in Figure 1.

In this example, an arrow from one city to another indicates the direction of travel. The opposite direction is not possible unless a similar arrow is present in the graph. For this programming

[1] In the spirit of simplicity, we will not consider layovers in this project.

challenge, each arrow or flight path would also have a cost associated with it. If we wanted to travel from El Paso to city Chicago, we would have to pass through Austin and Detroit. This would be a trip with three legs (El Paso to Austin, Austin to Detroit and Detroit to Chicago). It is possible that there might not be a path from one city to another city. In this case, you'd print an error message "NO FLIGHT AVAILABLE FOR THE REQUEST", exactly.

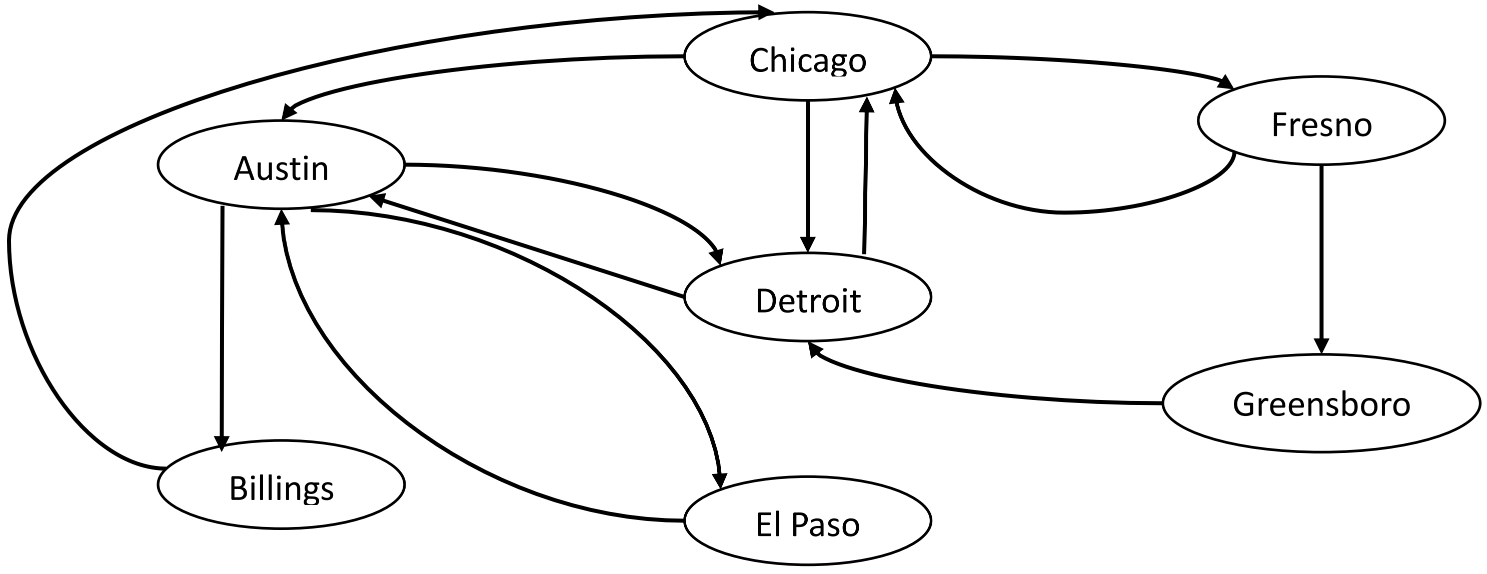

Figure 1 - Sample Directed Graph

The input file for flight data will represent a sequence of origin/destination city pairs with a cost of that flight.

Sample Data

Flight Data:

Here is an example of a flight data input file (it is not one that goes with Figure 1):

```
Dallas|Austin|98|47
Austin|Houston|95|39
Dallas|Houston|101|51
Austin|Chicago|144|192
Chicago|Austin|155|200
Austin|Dallas|100|50
Houston|Dallas|100|50
```

Each line will contain two city names (from-city and to-city; both strings), the cost of the flight (an integer number), and the number of minutes of the flight (an integer number). Each field will be separated with a pipe (shift-\ on most keyboards).

Requested Flight Plans:

A sample input file for requested flight plans is shown below. Each subsequent lines will contain a pipe-delimited list of city pairs with a trailing character to indicate sorting the output of flights by time (T) or cost (C). Your solution will find the shortest/cheapest flight path between these two cities (if one exists) and calculate the total cost of the flights and the total time in the air.

```
Dallas|Houston|T
Chicago|Dallas|C
NewYork JFK|Dallas|C
```

Output File:

For each flight in the Requested Flight Plans file, your program will print the most efficient flight plan available based on whether the request was to order by time or cost. Each line in the output file will be the solution to the corresponding query in the query file, line number should match. The file is pipe delimited list of solutions, in each solution indicating no of legs in travel followed by the cities in path from source to destination each city separated by the pipe symbol, followed by the cost of the path and then followed by the time taken for the path.

The line format of solution file will be;

No. of legs |Source| next city| next city|------|Destination |Cost of the travel |Time of the travel

Note the number of cities in the path will be always 1 + the number of legs. If you are unable to find the solution because there is no path from source to the destination or one or both the cities is missing in the graph, then just print the error message:

NO FLIGHT AVAILABLE FOR THE REQUEST

Here is an example:

```
1|Dallas|Houston|101|51
2|Chicago|Austin|Dallas|255|250
NO FLIGHT AVAILABLE FOR THE REQUEST
```