CS211 Final Project Overview

Submitted by Shane Hyland Pierce – 18409052

Setup

I decided to write my solution in Java as I wanted to use an object-oriented approach to store the data, and this is the object-oriented language I am most comfortable with. My first goal was to figure out how to take the input from the csv file and store it using a data structure that would allow for an effective solution.

Input

The input was a csv file with pairs of latitudes and longitudes on each line. I used the java scanner to read through the csv file and I stored each pair of latitudes and longitudes in airport objects. Each airport object contained the lats and longs of the airport and also the airport's id number which would later be used when generating a solution string I stored all of the airport objects in an array.

Graphs

My solution involves the use of a weighted graph to find the shortest path between all the airports, so I had to find a way of representing this information on a graph in java. I decided to use an adjacency matrix and I represented this in java with a 2D array. In the array each slot contains the weight between two nodes, these nodes being the indices of the array. Each node represents an airport and each weight represents a distance between the two airports e.g. the value of the matrix at [0][3] will be the distance between airport 0 and airport 3. This means that every node will be connected to every other node so that the distance between every airport can be measured. I created a graph class to store the graph information. This graph contains the matrix and methods to add edges, get distances and mark airports as already visited.

Filling the Graphs

In the main class I have a method called fillGraph which takes in the graph, the number of airports and the array of airport objects. It uses a nested loop to find the distance between every airport and add this information to the matrix. However because there is a condition that a plane cannot go from one airport to another than is 100km or less away, if the distance between one airport to another is <= 100 I set the distance to be equal to 0 so that my algorithm doesn't match these airports in the solution.

Greedy Algorithm with Randomness

My approach to solving this problem was to use the nearest neighbour greedy algorithm. This algorithm chooses the nearest airport (over 100km away) to go to each time, and then marks the airports that have already been passed through by setting the distance between it and every other airport to be 0. I implement this algorithm in my tsp method which takes in the graph and returns a solution string. I start with airport 0 and get the distance between 0 and every other airport. I check if each distance is 0 as this means the airport is either less than 100km away or has already been visited and cannot be the next airport. When the airport with the shortest distance is found I mark it as visited, add it to the solution string and then do the same with this airport, checking the distance between it and all other unvisited airports to find the shortest distance. I do this until every airport has been visited. I then add airport 0 to the end of the solution string and return it. The problem with this algorithm is it is deterministic and will always end up having multiple airports at the end that are all within 100km of each other. To fix this problem I add a random distance between 0 and 400km to each airport. When I call the tsp method, if the 2nd and 3rd last numbers in the solution string are the same, I know that there is less than 100km between the last few airports. I check this condition and if it is true I call the tsp method again and keep calling it until a valid solution string is produced. However because the randomness may give a path that is not short, I check the total distance travelled in the solution string and if this distance is not below a certain threshold I call tsp again until a good solution is found. This threshold can be changed easily however if it is too low the program may take an extremely long time, or even may never find a solution under this threshold. I went with the threshold of 495000km as the program took too long to calculate a path with a distance of less.

Solution

My final solution can be seen below in both HackerRank and Eclipse. The program took 42.658 seconds to come up with this solution, however due to randomness this time can variate greatly each time the program is run.

```
Solution String: 0,795,790,4,809,791,671,808,807,804,763,80
Distance Travelled: 490343.08300976024km
Program took: 42.6580233 seconds
```

Eclipse Solution

Your Output (stdout)

```
1  The total distance travelled by the plane is 490561 km
2  The time spent travelling is 1114 hours
```

HackerRank Solution