Traveling Salesman problem using A* Algorithm

Objective

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and reaches the destination. Each city is connected to one or more cities and it can be a one-way connection. Our aim is to use A* algorithm and establish a heuristic function for shortest distance and minimum number of cities visited, such that we reach the destination from source without exploring all the possible solutions. This heuristic gives us information at every node(City) regarding which city to select next when there is a situation of multiple roads from that city. This selection is done so that minimum number of paths are explored directed towards destination rather than exploring all possible solution paths.

Inputs

Connection File – List of cities and their connections to other cities. It can be one-way connection or two-way connection.

Locations File – List of cities with their coordinates in space.

Heuristic to choose – Shortest distance or fewest links.

Source – City where the search starts from.

Destination – City where the search ends once it is reached.

Exclude Cities – Cities to be excluded before the start of the search.

Algorithm

Open Nodes : Set of nodes to be evaluated

Closed Nodes : Set of nodes already evaluated

G_cost = F_Cost + H_Cost

H_Cost : Heuristic function (Example: Straight line distance between Nodes)

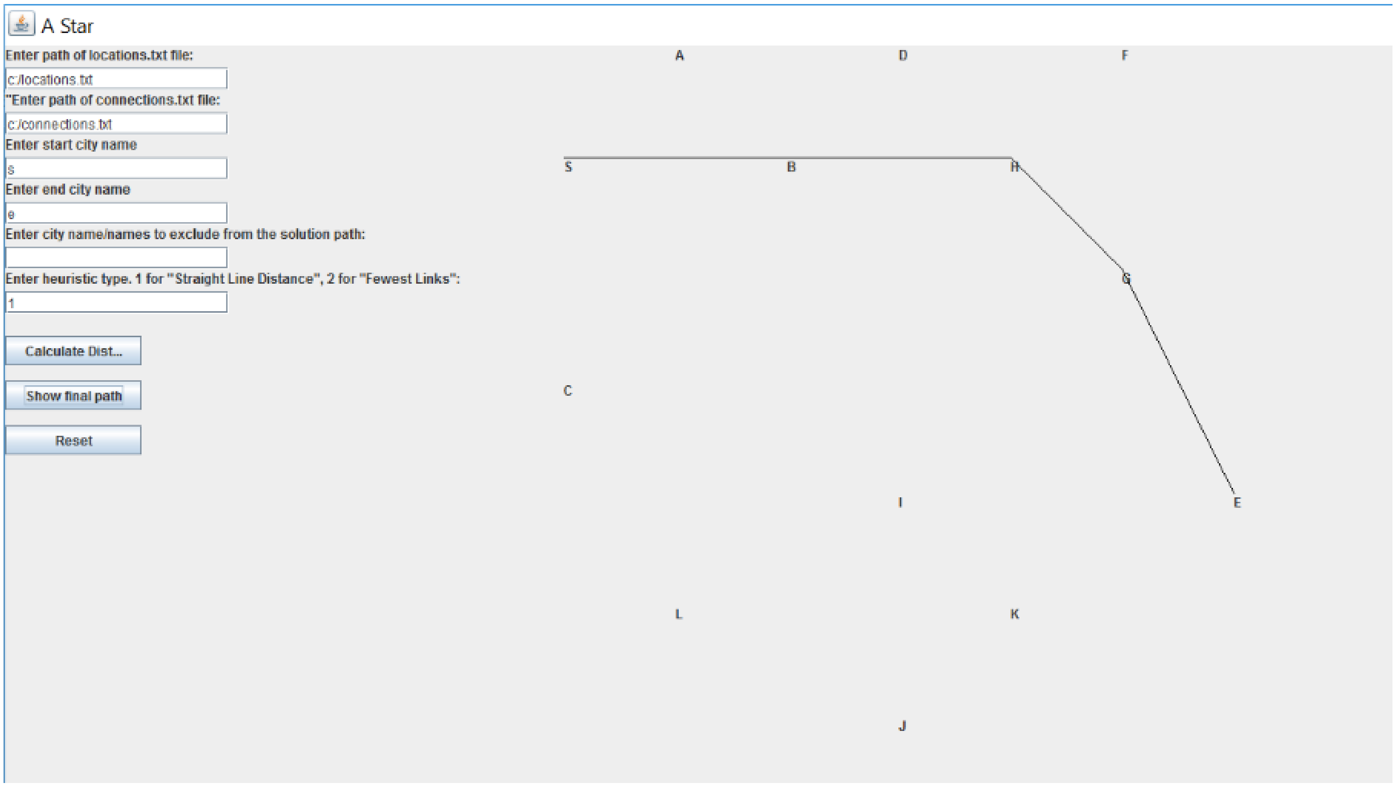
F_Cost: Distance travelled till Current Node.

Set H_cost, G_cost, Parent=null for Start City

```
Loop While CurrentCity is not the Destination City
    For each ConnectingCity (Neighbour) of the CurrentCity
        IF the ConnectingCity is in the list of closed nodes then Continue to next iteration
        ELSE (ConnectingCity is not in the list of Closed Nodes) then
            IF G_cost is already calculated for the ConnectingCity, then we check if an update is needed.
                IF the actual distance calculated from current path is greater than or equal to the actual distance calculated from previous path, we don't need to update the cost estimate for this ConnectingCity and current iteration of the loop is skipped.
                ELSE
                G_cost and parent node are updated for ConnectingCity.
            ELSE (If G_cost of ConnectingCity is not previously calculated)
                    Set H_cost, G_cost and Parent node
                    Add ConnectingCity to OpenNodes
//end for loop
    Set CurrentCity to ConnectingCity with the Node having less G_cost in Open Nodes.
//end while loop
```

Output

Step by step execution of the shortest path taken depending on the heuristic selected (Straight line distance or fewest links).



A Star

Enter path of locations.txt file:

c:/locations.txt

"Enter path of connections.txt file:

c:/connections.txt

Enter start city name

s

Enter end city name

e

Enter city name/names to exclude from the solution path:

Enter heuristic type. 1 for "Straight Line Distance", 2 for "Fewest Links":

1

Performance:

Only one optimal solution is given using knowledge(heuristic) for the shortest path without exploring all the possibilities. Exploring all possibilities takes a lot of time. Hence by using the heuristic above we have improved the performance of the search.

Task Management:

Analysis, Algorithm, Designing, Coding, Testing, UI Design, Documentation – done by Astha Sharma and Mounica Reddy