Artificial Intelligence | Assignment 2

Report

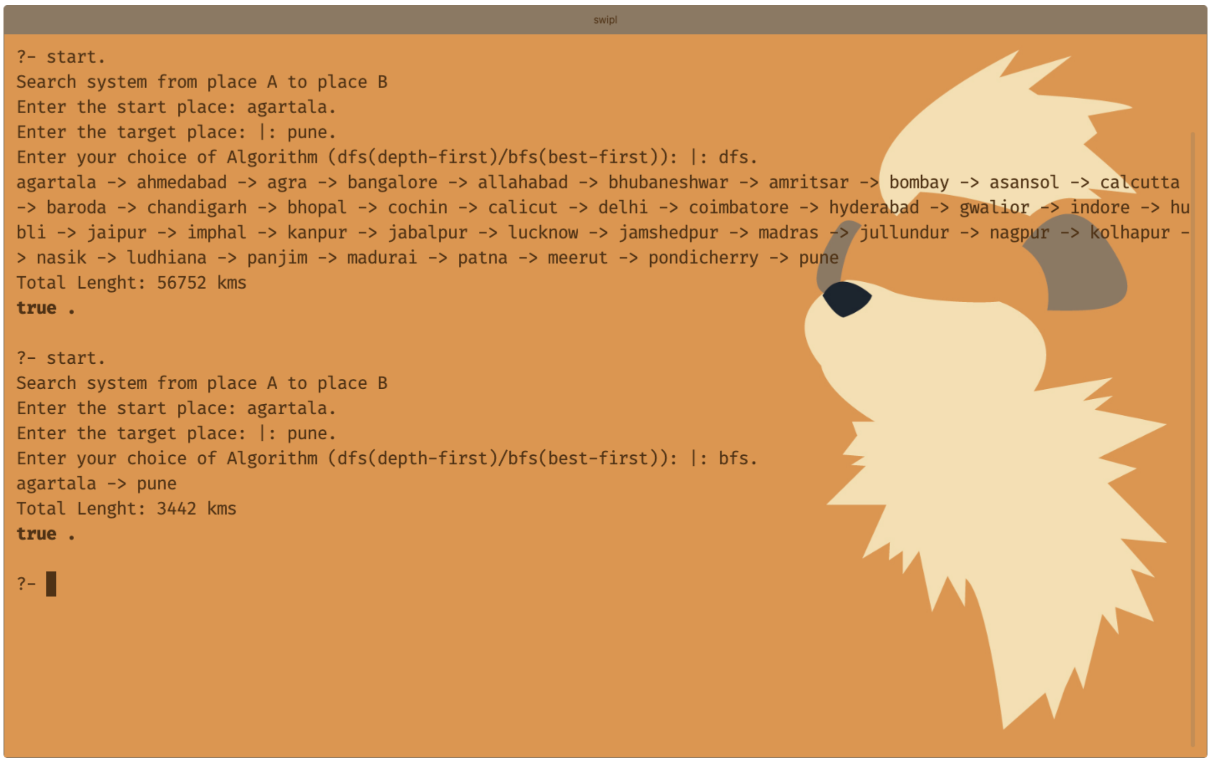
Data Preprocessing

- From the roaddistance.csv file, the first empty column and the last empty column was removed.
- To generate heuristics.csv file, python was used.
- Using BING API, latitudes and longitudes for each city were generated.
- Using the geopy library, Euclidean distance, rounded off to the nearest integer in kms was calculated and used as the final heuristic.

Working

- The program first reads the heuristic.csv file.
- To make sure that the heuristics are consistent, the heuristics are taken as the straight line distance between the two cities, whose integer division is done by 4.
- After this, the program reads roaddistance.csv file.
- It is made sure that the first line is not read while parsing the csv and a bidirectional graph is generated using that.
- After that the Start place and the Goal place are taken from the user input.
- The program ensures that both the places are actually present in the database, and if not then the program prompts the user to enter them again.
- The program then asks the user to enter the algorithm to be used for calculating the path.
- In this program, the available algorithms are:
    - Depth First Search
    - Best First Search
- The program makes sure that the algorithm entered by the user is among the above ones else, it prompts the user to enter it again.
- Best First Search is implemented using sorting method.
- Once the path is calculated, the path along with the path length is printed.
- The program works for both the cases when there is a direct connection between the two cities as well as when there is no direct connection between the two cities.

Agartala -> Pune (Direct edge exists)

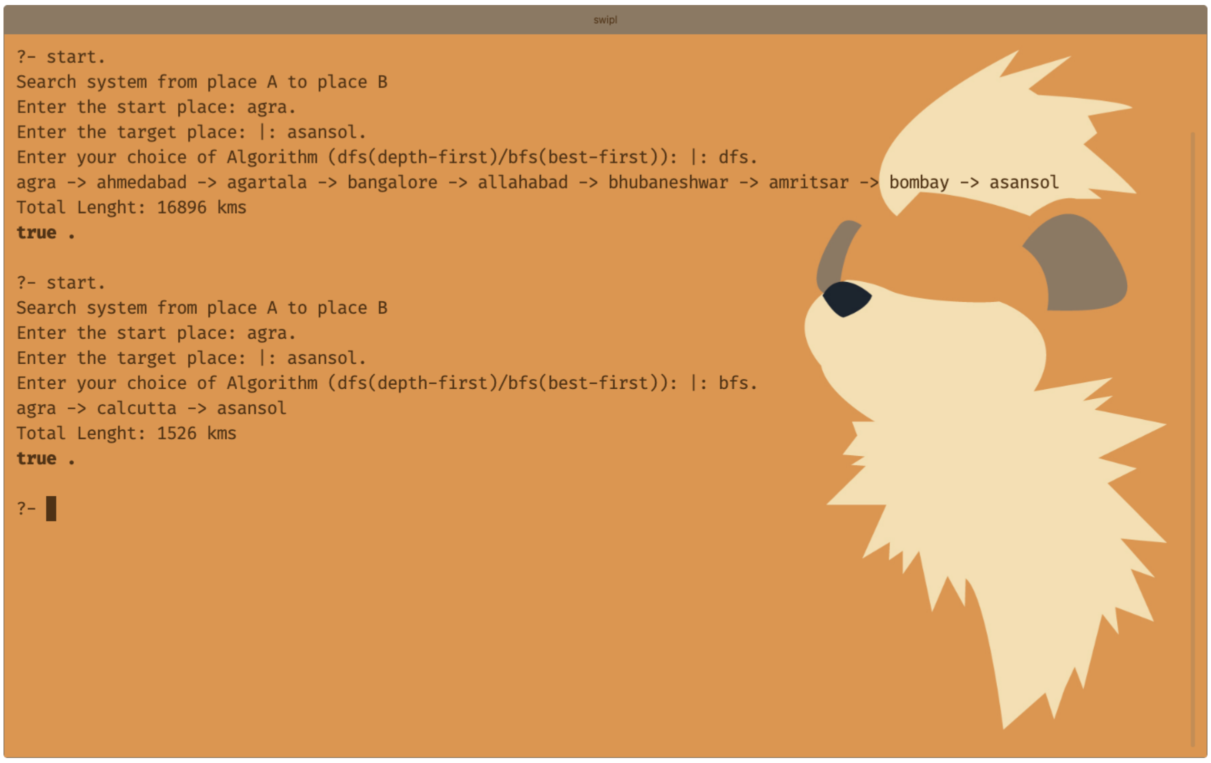


```
?- start.
Search system from place A to place B
Enter the start place: agartala.
Enter the target place: |: pune.
Enter your choice of Algorithm (dfs(depth-first)/bfs(best-first)): |: dfs.
agartala -> ahmedabad -> agra -> bangalore -> allahabad -> bhubaneshwar -> amritsar -> bombay -> asansol -> calcutta
-> baroda -> chandigarh -> bhopal -> cochin -> calicut -> delhi -> coimbatore -> hyderabad -> gwalior -> indore -> hu
bli -> jaipur -> imphal -> kanpur -> jabalpur -> lucknow -> jamshedpur -> madras -> jullundur -> nagpur -> kolhapur -
> nasik -> ludhiana -> panjim -> madurai -> patna -> meerut -> pondicherry -> pune
Total Lenght: 56752 kms
true .

?- start.
Search system from place A to place B
Enter the start place: agartala.
Enter the target place: |: pune.
Enter your choice of Algorithm (dfs(depth-first)/bfs(best-first)): |: bfs.
agartala -> pune
Total Lenght: 3442 kms
true .

?- 
```

Agra -> Asansol (Direct edge does not exist)



```
?- start.
Search system from place A to place B
Enter the start place: agra.
Enter the target place: |: asansol.
Enter your choice of Algorithm (dfs(depth-first)/bfs(best-first)): |: dfs.
agra -> ahmedabad -> agartala -> bangalore -> allahabad -> bhubaneshwar -> amritsar -> bombay -> asansol
Total Lenght: 16896 kms
true .

?- start.
Search system from place A to place B
Enter the start place: agra.
Enter the target place: |: asansol.
Enter your choice of Algorithm (dfs(depth-first)/bfs(best-first)): |: bfs.
agra -> calcutta -> asansol
Total Lenght: 1526 kms
true .

?-
```