AI ASSIGNMENT-2

The following files are being submitted, attached in the zip folder:

- report.pdf
  This file itself. Contains deliverables, steps to run the program, logic of the code, prolog features and screenshots of sample inputs.
- heuristicdistance.pl
  This file is formed when heuristics.py is run. It consists of the facts that will be used as heuristics.
- test.pl
  This file contains the main code for reading the csv file and then based on that creating facts for determining the distances and the roads. Basically, it forms the weighted graph with cities as nodes and roads as edges in prolog.
- findpath.pl
  This is the main file for running the Searching Algorithms
- roaddistance.csv
  It is a modified version of the csv file attached, just the 1 row is deleted from what is attached. It is read in the test.pl file.
- heuristics.py
  In this python file, the heuristics are formed and written as prolog code.

Steps to run the program:

- Open swi-prolog using the command swipl in the terminal.
- Run [‘test.pl’]. After this run the function start.
- Now run the python file heuristics.py. It forms data for heuristicdistance.pl file.
  Command: python heuristics.py
- Now run the prolog file in swipl: [‘heuristicdistance.pl’].
- Run the prolog file, [‘findPath.pl’]. Execute the command startSearch. to start the program for searching algorithms. After selecting the algorithm, enter the city names between which you wish to find the path.
- NOTE: Enter the city names in single quotes.
- To see multiple paths in DFS, remove the cut in line 30.
- To see multiple paths in Best First Search, remove the cut in line 56.

Logic of the code:

test.pl file reads the given CSV file and makes facts out of them. These facts correspond to weighted edges of the graph.

The Heuristic for Best-First Search used is Straight Line Path between the cities. It is found using the ‘geopy’ library of python. First the latitude and longitude coordinates of the cities are found and then straight line distance between them is found. It is then written in the ‘heuristicdistance.pl’ file.

Using these, Depth First Search and Best-First Search are implemented in ‘findPath.pl’ file.

In DFS, the max depth is defined as 4. Can be changed to get deeper recursion.

Important Prolog Features Used:

- Lists, Recursion, Backtracking
- CSV Reader using the library csv
- Cuts
- Functor
- Table
- Various in-built functions such as
    - arg() – To get member based on index
    - member() – to check list membership
    - forall() – to iterate over all list members and process them
    - assert() – to dynamically assert facts
    - retract(), retractall() – to remove facts dynamically
    - functor() – to find arity of a fact
    - findall() – for forming a list based on some conditions
    - map_list_to_pairs() – for converting elements of a list to list of pairs based on some criteria (In our case, heuristic distance)
    - append() – to append 2 lists to third
    - keysort() – to sort a list of pairs
    - pairsvalue() – to convert a pairs list back to normal list of values
- Checking ‘not’ a fact using \+

Screenshots of working program:

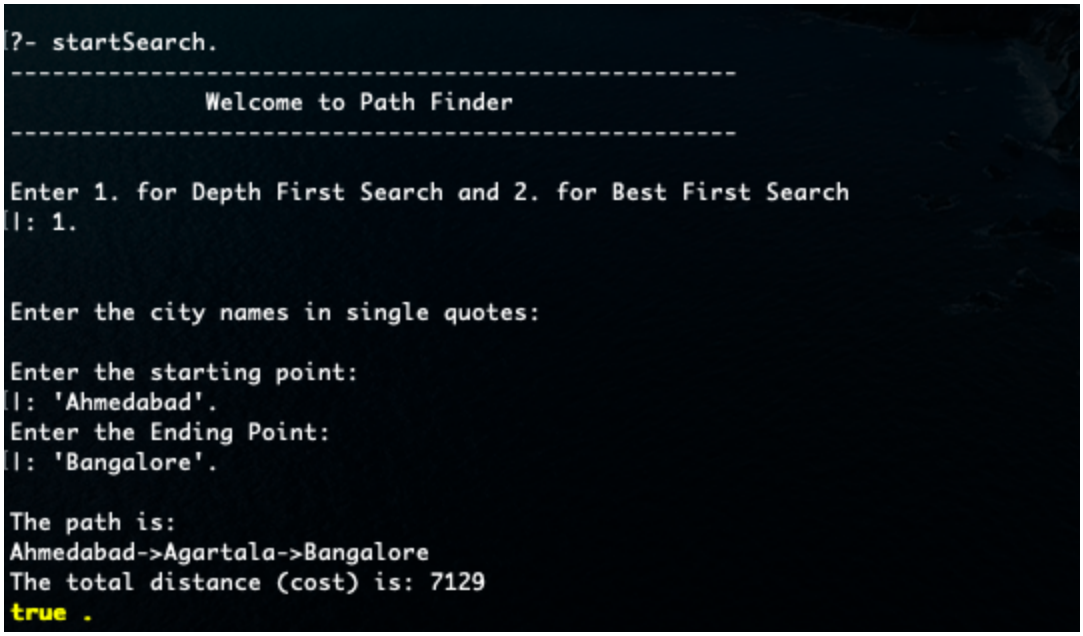
```
visheshrangwani@Visheshs-MacBook-Air Assignment 2 Final % python heuristics.py
visheshrangwani@Visheshs-MacBook-Air Assignment 2 Final % swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['heuristicdistance.pl'].
true.

?- ['test.pl'].
true.

?- start.
Hello 1
Hello 2
Hello 3
true.

?- ['findPath.pl'].
true.

?- startSearch.
-----------------------------------------------------
              Welcome to Path Finder
-----------------------------------------------------

Enter 1. for Depth First Search and 2. for Best First Search
|:
```

Depth First Search:

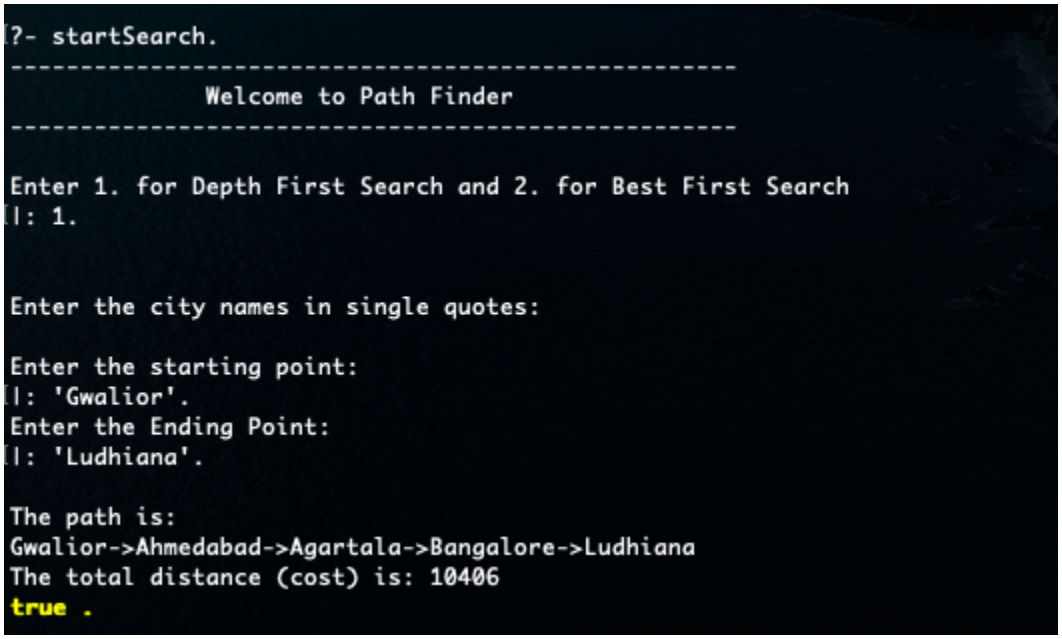
1. Source: Bangalore; Destination: Bangalore

```
?- startSearch.
-----------------------------------------------------
              Welcome to Path Finder
-----------------------------------------------------

Enter 1. for Depth First Search and 2. for Best First Search
|: 1.


Enter the city names in single quotes:

Enter the starting point:
|: 'Bangalore'.
Enter the Ending Point:
|: 'Bangalore'.

The path is:
Bangalore
The total distance (cost) is: 0
true .
```

2. Source: Ahmedabad; Destination: Bangalore



3. Source: Gwalior; Destination: Ludhiana



Best First Search:

1. Source: Bangalore; Destination: Bangalore

```
?- startSearch.
--------------------------------------------------
              Welcome to Path Finder
--------------------------------------------------

Enter 1. for Depth First Search and 2. for Best First Search
|: 2.


Enter the city names in single quotes:

Enter the starting point:
|: 'Bangalore'.
Enter the Ending Point:
|: 'Bangalore'.

The path is:
Bangalore
The total distance (cost) is: 0
true .
```

2. Source: Ahmedabad; Destination: Bangalore

```
?- startSearch.
--------------------------------------------------
              Welcome to Path Finder
--------------------------------------------------

Enter 1. for Depth First Search and 2. for Best First Search
|: 2.


Enter the city names in single quotes:

Enter the starting point:
|: 'Ahmedabad'.
Enter the Ending Point:
|: 'Bangalore'.

The path is:
Ahmedabad->Bangalore
The total distance (cost) is: 1490
true .
```

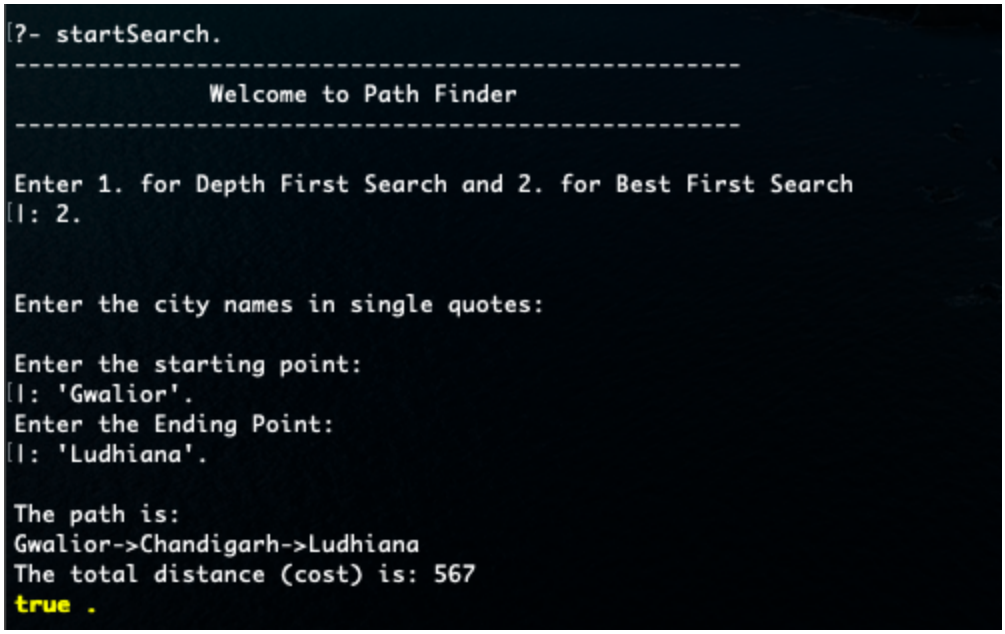
3. Source: Ahmedabad; Destination: Agra

```
?- startSearch.
-----------------------------------------------------
              Welcome to Path Finder
-----------------------------------------------------

Enter 1. for Depth First Search and 2. for Best First Search
|: 2.


Enter the city names in single quotes:

Enter the starting point:
|: 'Ahmedabad'.
Enter the Ending Point:
|: 'Agra'.

The path is:
Ahmedabad->Agra
The total distance (cost) is: 878
true .
```

4. Source: Agra; Destination: Ahmedabad

```
?- startSearch.
-----------------------------------------------------
              Welcome to Path Finder
-----------------------------------------------------

Enter 1. for Depth First Search and 2. for Best First Search
|: 2.


Enter the city names in single quotes:

Enter the starting point:
|: 'Agra'.
Enter the Ending Point:
|: 'Ahmedabad'.

The path is:
Agra->Ahmedabad
The total distance (cost) is: 878
true .
```

5. Source: Gwalior; Destination: Ludhiana


?- startSearch.
Welcome to Path Finder
Enter 1. for Depth First Search and 2. for Best First Search
|: 2.
Enter the city names in single quotes:
Enter the starting point:
|: 'Gwalior'.
Enter the Ending Point:
|: 'Ludhiana'.
The path is:
Gwalior->Chandigarh->Ludhiana
The total distance (cost) is: 567
true .