AI Assignment 2

Abhinav Saurabh MT20127

Steps to run the program:

1. open terminal
2. type swipl and hit enter
3. ['complete path']. then hit enter

   Eg -['/Users/abhinavsaurabh/Desktop/IIIT-DELH/soln2.pl']

4. Call knowledgebase() to load the heuristics and distances

```
?- ['/Users/abhinavsaurabh/Desktop/IIIT-DELHI/Semester 3/2. Artificial Intelligence/Assignment 2/soln2.
pl'].
true.

?- knowledgebase().
true.
```

5. Then we can use either DFS or BFS for the initial and target city. Given initial state and target state.
6. Then we call depth_first_search('agartala','pune').

```
?- depth_first_search('agartala','pune').

Path Discovered with DFS is
[agartala, --> ,ahmedabad, --> ,agra, --> ,bangalore, --> ,allahabad, --> ,bhubaneshwar, --> ,amritsar, --> ,bombay, --> ,asan
sol, --> ,calcutta, --> ,baroda, --> ,chandigarh, --> ,bhopal, --> ,cochin, --> ,calicut, --> ,delhi, --> ,coimbatore, --> ,hy
derabad, --> ,gwalior, --> ,indore, --> ,hubli, --> ,jaipur, --> ,imphal, --> ,kanpur, --> ,jabalpur, --> ,lucknow, --> ,jamsh
edpur, --> ,madras, --> ,jullundur, --> ,nagpur, --> ,kolhapur, --> ,nasik, --> ,ludhiana, --> ,panjim, --> ,madurai, --> ,pat
na, --> ,meerut, --> ,pondicherry, --> ,pune]

Total cost of the path is :
56752
true .
```

7. Then we call bestFS('agartala','pune').

```
?- bestFS('agartala','pune').
Path Discovered with Best First Search is
agartala --> pune
The total cost of discovered path:
3442
true
```

The above give solution using Depth-first search and Best First Search Solutions.

~/Desktop/IIIT-DELHI/Semester 3/2. Artificial Intelligence/Assignment 2/AI_A2_Abhinav MT20127/soln2.pl (functions)

```
% Reading the data from csv in the knowledge base
% Reading the Distance without Heuristics
knowledgebase :-  csv_read_file('/Users/abhinavsaurabh/Desktop/IIIT-DELHI/Semester 3/2. Artificial Intelligence/Assignm
maplist(assert,Distances),
% Reading the Distance with Heuristics
csv_read_file('/Users/abhinavsaurabh/Desktop/IIIT-DELHI/Semester 3/2. Artificial Intelligence/Assignment 2/BFS/dheur.cs
maplist(assert,DisHeuristics).


% To locate next node in case of DFS
next_node(Current, Next, Path) :-
% Getting distance between current node and next node
    distance(Current, Next,Dist),
    not(member(Next, Path)),
    assert(cost(Dist)).

% Depth First Search
% DFS Search for path to be found
depth_first_search(Initial_city,Target_city) :- depth_first(Initial_city, Target_city, [Initial_city]).
% Intial city is the current starting point
% Target city is the goal of the algorithm
depth_first(Target_city, Target_city, _) :- assert(cities(Target_city)),distance_list_conversion(List),nl,write("Path [
% TotalCost stores the cost of entire path then it is printed
depth_first(Initial_city, Target_city, Visited) :-
  % DFS takes input with intial node and the goal node
    next_node(Initial_city, Next_node, Visited),assert(cities(Initial_city)),assert(cities(" --> ")),
    % Moves further after printing intial nodes
    depth_first(Next_node, Target_city, [Next_node|Visited]).

% DFS exploits the path simultaneous recursively to node it encounters. Its an exhaustive search.

% further converting distances into the list format.
 distance_list_conversion([Px|Tail]):- retract(cities(Px)), distance_list_conversion(Tail).
 distance_list_conversion([]).

% further converting costs into the list format.
 cost_list_conversion([Px|Tail]):- retract(cost(Px)),  cost_list_conversion(Tail).
  cost_list_conversion([]).

% further converting costs for bfs into the list format.
 cost_list_conversion_bfs([Px|Tail]):- retract(cost_bfs(Px)),  cost_list_conversion_bfs(Tail).
  cost_list_conversion_bfs([]).
```

```
  cost_list_conversion([]).

% further converting costs for bfs into the list format.
 cost_list_conversion_bfs([Px|Tail]):- retract(cost_bfs(Px)),  cost_list_conversion_bfs(Tail).
  cost_list_conversion_bfs([]).

% cost summation on each of the steps is store in following variables.
cost_summation([],0).
cost_summation([T|R],M) :- cost_summation(R,S), M is T+S.



% Best First Search
% BFS search for path
best_first(Initial_city,Target_city) :- heuristic(Initial_city,Target_city,Value),write("Path Discovered with Best Firs
% Intial city is the current starting point
% Target city is the goal of the algorithm
bestFirstSearch(X,X,_,_) :- nl,write("The total cost of discovered path: "),nl,cost_list_conversion_bfs(CostListBfs),co
% NetCost stores the cost of entire path then it is printed for bfs
bestFirstSearch(_,_,[],_):- write("No element left in the Open List").
% BFS takes intial city and target i.e goal state as input
bestFirstSearch(Initial_city,Target,OpenList,ClosedList) :-
% Further BFS exploits the lists.
[Head1 | Tail] = OpenList,
% intial city is initialized
_-Initial_cityNode = Head1,
% Finding all the values next
findall(Value-NextNode,(distance(Initial_cityNode,NextNode,_), Initial_cityNode \== NextNode, not(member(NextNode,Close
% BFS does its search with  heuristic way here and tries to find the best path to the targets
append(NN,Tail,UpdatedOpenList),
keysort(UpdatedOpenList,SortedOpenList),
[HeadNode|_] = SortedOpenList,
_-BestNextNode = HeadNode,

% Further best path is printed here
write(" --> "),write(BestNextNode),
% Then again next best node is printed
distance(Initial_city,BestNextNode,Dist),
% Further cost is taken into the account
assert(cost_bfs(Dist)),
bestFirstSearch(BestNextNode,Target,SortedOpenList,[Initial_city|ClosedList]).

```