AI- Assignment 1

A road route finding system

Assumptions:

- All connections are bi-directional

Heuristic:

We define heuristic function of a node as the minimum cost of the node and its neighbours.

Checking for admissible and consistent.

As the h(s) = minimum edge cost of the node and its neighbours.

- ➔ h(s) ≤ h*(s), heuristic would be always less than equal to true cost.
- ➔ Heuristic is admissible.

And h(s) < cost(s,t)+h(t)

Here h(s) is the minimum cost and h(t) > 0

- ➔ So the above condition will always be true.
- ➔ Heuristic is consistent.

Change directory using - cd('c:/Users/Sharm/Desktop/Courses/AI/AI-A2-Mohit-2020086'). to move into working directory.

Change the address according to the location of the files

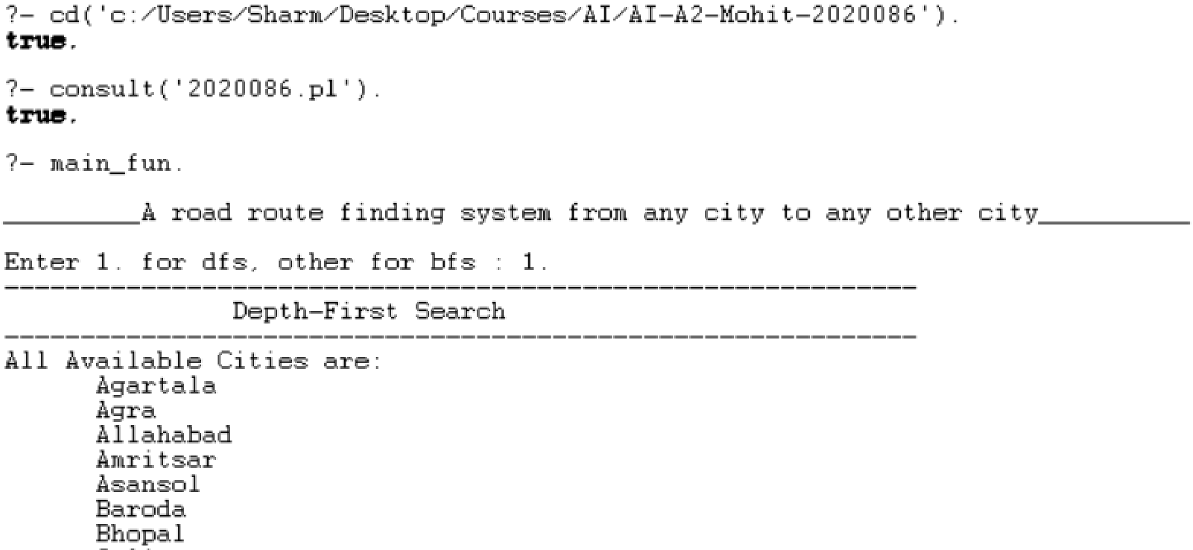
Source Code

```
:-style_check(-singleton).

[library(csv)].                % Importing library for reading csv file
[library(lists)].              % library for list usage
[library(aggregate)].

:- dynamic distance/3.         % creating dynamic fact named distance of the form distance(starting_point, ending_point,
distance)
:- dynamic visited/1.
:- dynamic heuristic/2.

main_fun:-
   nl,write('_________A road route finding system from any city to any other city__________'),nl,nl,
   setting_env(All_city),
   write('Enter 1. for dfs, other for bfs : '),
   read(Option),
   dfs_env(All_city,Option) ,bfs_env(All_city,Option).

setting_env(All_city):-                                  % helper funtion used to set all facts and rules
   retractall(name(_)),
   retractall(distance(_,_,_)),
   retractall(heuristic(_,_)),
   csv_to_facts('roaddistance.csv',All_city),
   retractall(distance(_,'','')).

show_route([]) :- true.
show_route([H]) :- write(H);true.
show_route([H|T]):-
      write(H),write(' -> '),
      show_route(T);true.

show_city_list([]).
show_city_list([H|T]):-
      write('      '),write(H),nl,
      show_city_list(T);true.

print_ans(Path,Cost):-
   write('Path to Reach Goal: '), nl,
   show_route(Path),nl,
   write('Cost to Reach Goal : '),
   write(Cost),nl.

csv_to_facts(FILENAME,All_city):-                        % Converts given CSV file in FILENAME to facts
   csv:csv_read_file(FILENAME,[_,First|Remaining]),       % Ignoring the first Row, Storing the Row with city names in
first
   helper2(First,First_list),
   helper3(First_list,Remaining, All_city).

helper2(Row,Row_l):-                                     % Function to convert Row to list
   Row =..[_,_|Row_l].

helper3([_|First_list],[],All_city) :- All_city = First_list;true.

helper3(First_list,[Row|Remaining],All_city_up):-
   helper2(Row,Row_l),
   lists:nth1(1,Row_l,A),                                % Get city name from the list
   Flag_1=(lists:nth1(Index,Row_l,Distance)),                % Get a distance from the list
   Flag_2=( Index > 1),
   Flag_3=(lists:nth1(Index,First_list,B)),                 % Get Corresponding City name
```

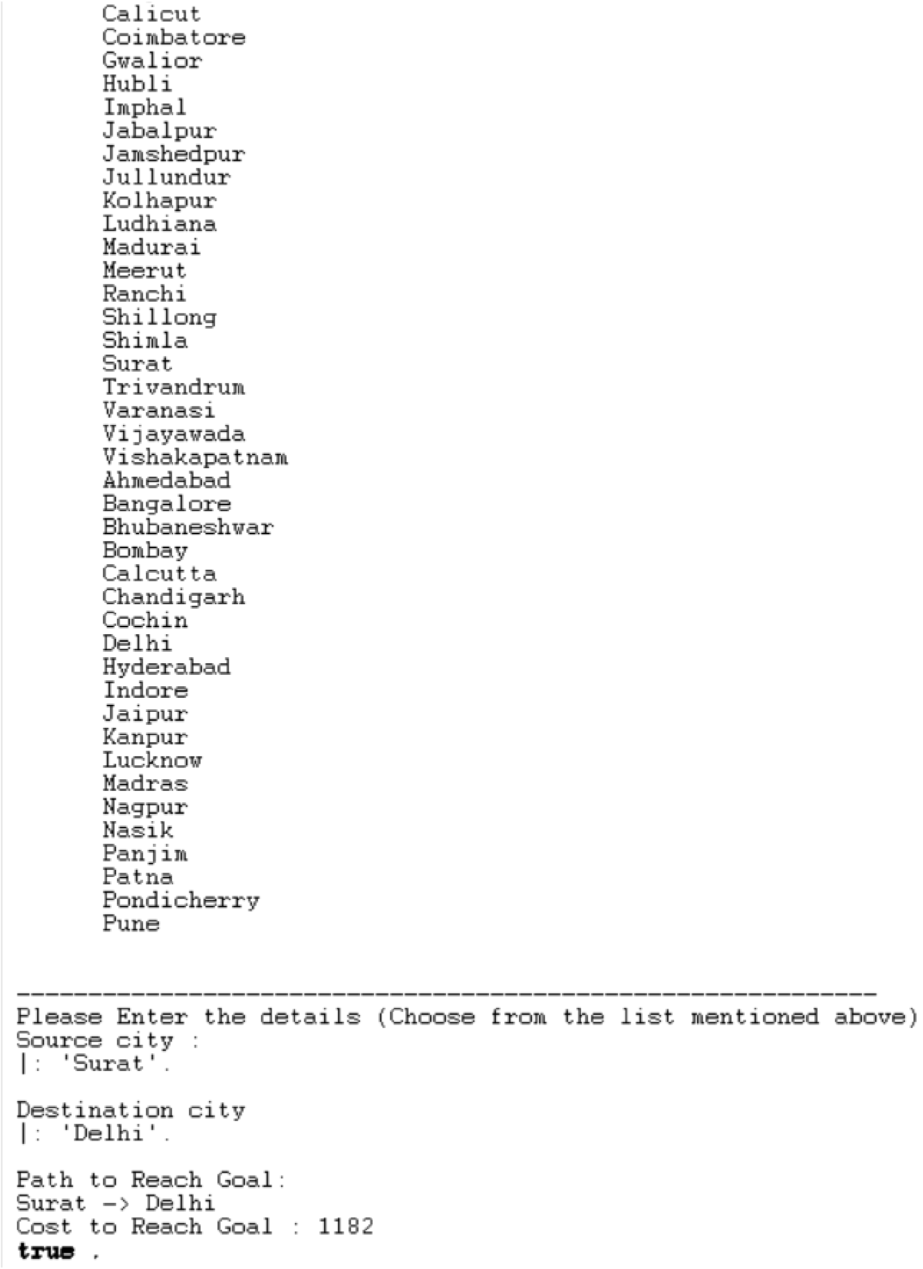
```
    Flag=(Flag_1,Flag_2,Flag_3),                    % Check if all the conditions satisfies ie all the arguments are intentiated
    helper4(A),helper4(B),
    Flag_as=assertz(distance(A,B, Distance)),             % Add the fact at the end
    forall(Flag,Flag_as),
    helper3(First_list, Remaining, All_city),
    helper5(All_city, A, All_city_up).

/* Also add distance(A,A,0) Type in the facts*/
helper4(City_name):-
    \+ distance(City_name,City_name,_),
    assert(distance(City_name,City_name,0))
    ;true.

helper5(All_city, A, All_city_up):-
    \+member(A,All_city) -> All_city_up = [A|All_city] ;  All_city_up = All_city
    ;true.

dfs_env(All_city,Option):-
    Option == 1,
    write('----------------------------------------------------------'),nl,
    write('                Depth-First Search'),nl,
    write('----------------------------------------------------------'),nl,
    write('All Available Cities are: '),nl,
    show_city_list(All_city),nl,
    write('----------------------------------------------------------'),nl,
    write('Please Enter the details (Choose from the list mentioned above)'),nl,
    write('Source city : '),nl,
    read(S),nl,
    writef("Destination city"),nl,
    read(G),nl,
    dfs(S,G,Path,Cost),
    print_ans(Path,Cost);true.

/*      dfs Code is Implemented from the pseudocode given in the class

            Algorithm DEPTH: Depth first search in state space
                1. Init lists OPEN = {Si}, CLOSED = {}
                2. if OPEN = {}
                    then return FAIL
                3. Remove first node S from OPEN and insert it into CLOSED
                4. Expand node S
                    4.1. Generate all successors Sj of node S
                    4.2. for each succesor Sj of S do
                    4.2.1. Set link Sj → S
                    4.2.2. if Sj is final state
                        then
                    i. Solution is (Sj,.., Si)
                    ii. return SUCCESS
                    4.2.3. Insert Sj in OPEN, at the beginning
                5. repeat from 2
            end.
*/

connected(S,G,C):-
    distance(S,G,C);distance(G,S,C).

generate_neigh_list(M,Neighbour_list) :-
    findall(T,(connected(M,T,_),\+ visited(T)), Neighbour_list).

dfs(S,G,Path,Cost):-
    retractall(visited(_)),
    dfs1(S, G,[S],P1,Cost),
    Path = P1.

dfs1(_, _, [], Path, Cost):-
    fail.

dfs1(S, G, [M|Open], Path, Cost):-
    connected(M,G,_) -> ifpart_dfs(S, G, [M|Open], Path, Cost); ifpart_dfs(S, G, [M|Open], Path, Cost).

ifpart_dfs(S, G, [M|_], Path, Cost):-
    connected(M,G,C),
    Path = [M,G],
    connected(S,M,C1),
    Cost is C + C1.

ifpart_dfs(S, G, [M|Open], Path, Cost):-
    assert(visited(M)),
    generate_neigh_list(M, Neighbour_list),
    append(Neighbour_list, Open, Updated_open),
    dfs1(M, G, Updated_open, Path_rest, Cost_rest),
    Path = [M|Path_rest],
    connected(S,M,C),
    Cost is C+Cost_rest.


bfs_env(All_city,Option):-
    Option == 2,
    write('------------------------------------------------------------'),nl,
    write('                Best-First Search'),nl,
    create_heuristics(All_city),
    write('------------------------------------------------------------'),nl,
    write('All Available Cities are: '),nl,
    show_city_list(All_city),nl,
    write('------------------------------------------------------------'),nl,
    write('Please Enter the details (Choose from the list mentioned above)'),nl,
    write('Source city : '),nl,
    read(S),nl,
    writef("Destination city"),nl,
    read(G),nl,
    bfs(S,G,Path,Cost),
    print_ans(Path,Cost);true.

/*      bfs Code is Implemented from the pseudocode given in the class

    Algorithm BEST-FIRST: BEST first search in state space
    1. Init lists OPEN = {Si}, CLOSED = {}
    2. if OPEN = {}
        then return FAIL
    3. Sort nodes in OPEN with the minimum estimated cost one to goal first
    4. Remove first node S from OPEN and insert it in CLOSED
    5. Expand node S
        5.1. Generate all direct successors Sj of node S
        5.2. for each successor Sj of S do
        5.2.1. Make link Sj → S
        5.2.2. if Sj is final state
            then
            i. Solution is (Sj, S, .., Si)
            ii. return SUCCESS
        5.2.3. Insert Sj in OPEN, at the end
    6.repeat from 2
    end.

*/
```

```
create_heuristics([]).
create_heuristics([H|T]):-
   \+ heuristic(H,_),
   retractall(distance(H,H,_)),
   smallest_distance(H,D),
   assert(heuristic(H,D)),
   assert(distance(H,H,0)),
   create_heuristics(T);true.
create_heuristics([H|T]):-
   heuristic(H,_),
   create_heuristics(T);true.

smallest_distance(City, Distance):-
   connected(City,_,Distance),
   \+ (connected(City,_,Distance2), Distance2 < Distance).

sorting_open(List,List_up) :-
   helper6(List,List_temp),
   !,
   sorting_open(List_temp,List_up).
sorting_open(List,List).

helper6([H1,H2|T],[H2,H1|T]) :-
   heuristic(H1,Distance1),
   heuristic(H2,Distance2),
   Distance1 > Distance2.
helper6([H|T],[H|T1]) :-
   helper6(T,T1).

bfs(S,G,Path,Cost):-
   retractall(visited(_)),
   bfs1(S, G,[S],P1,Cost),
   Path = P1.

bfs1(_, _, [], Path, Cost):-
   fail.

bfs1(S, G, [M|Open], Path, Cost):-
   connected(M,G,_) -> ifpart_bfs(S, G, [M|Open], Path, Cost); elsepart_bfs(S, G, [M|Open], Path, Cost).

ifpart_bfs(S, G, [M|_], Path, Cost):-
   connected(M,G,C),
   Path = [M,G],
   connected(S,M,C1),
   Cost is C + C1.

elsepart_bfs(S, G, [M|Open], Path, Cost):-
   assert(visited(M)),
   generate_neigh_list(M, Neighbour_list),
   append(Neighbour_list, Open, Updated_open),
   write(Open),nl,
   sorting_open(Updated_open,Sorted_open),
   bfs1(M, G, Sorted_open, Path_rest, Cost_rest),
   Path = [M|Path_rest],
   connected(S,M,C),
   Cost is C+Cost_rest.
```

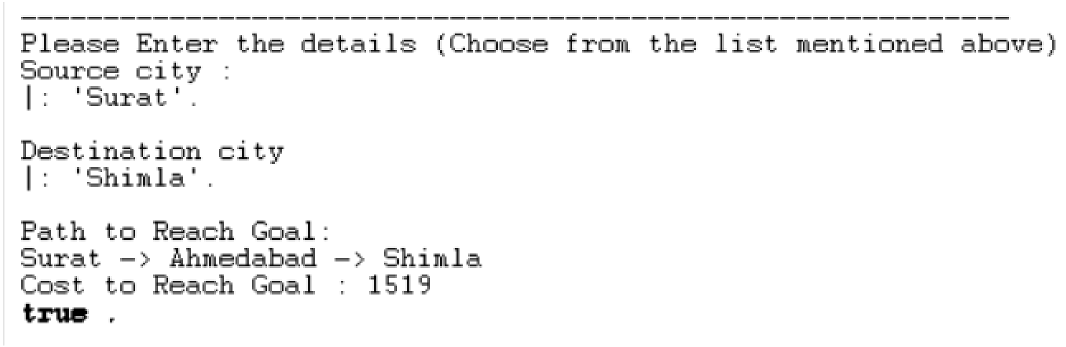
Screenshots: (dfs)



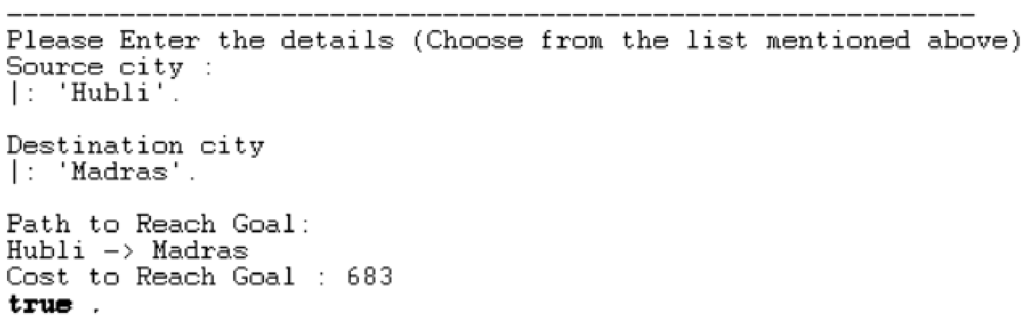
```
?- cd('c:/Users/Sharm/Desktop/Courses/AI/AI-A2-Mohit-2020086').
true.

?- consult('2020086.pl').
true.

?- main_fun.

_________A road route finding system from any city to any other city_________

Enter 1. for dfs, other for bfs : 1.
------------------------------------------------------------
               Depth-First Search
------------------------------------------------------------
All Available Cities are:
      Agartala
      Agra
      Allahabad
      Amritsar
      Asansol
      Baroda
      Bhopal
```



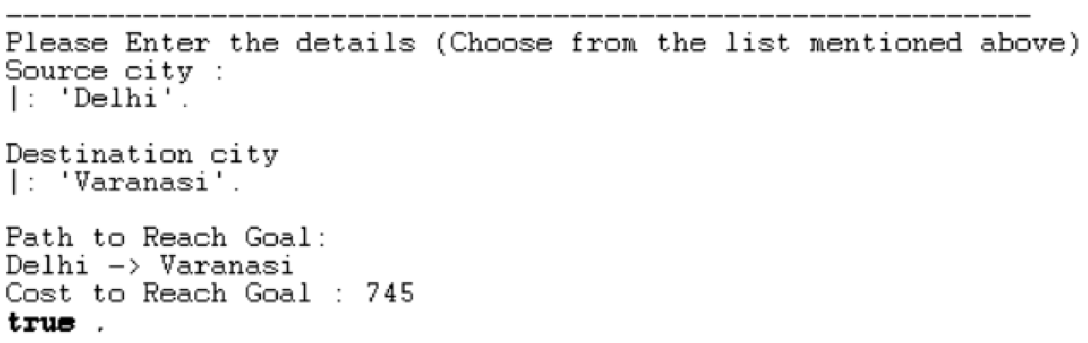
```
      Calicut
      Coimbatore
      Gwalior
      Hubli
      Imphal
      Jabalpur
      Jamshedpur
      Jullundur
      Kolhapur
      Ludhiana
      Madurai
      Meerut
      Ranchi
      Shillong
      Shimla
      Surat
      Trivandrum
      Varanasi
      Vijayawada
      Vishakapatnam
      Ahmedabad
      Bangalore
      Bhubaneshwar
      Bombay
      Calcutta
      Chandigarh
      Cochin
      Delhi
      Hyderabad
      Indore
      Jaipur
      Kanpur
      Lucknow
      Madras
      Nagpur
      Nasik
      Panjim
      Patna
      Pondicherry
      Pune


------------------------------------------------------------
Please Enter the details (Choose from the list mentioned above)
Source city :
|: 'Surat'.

Destination city
|: 'Delhi'.

Path to Reach Goal:
Surat -> Delhi
Cost to Reach Goal : 1182
true .
```



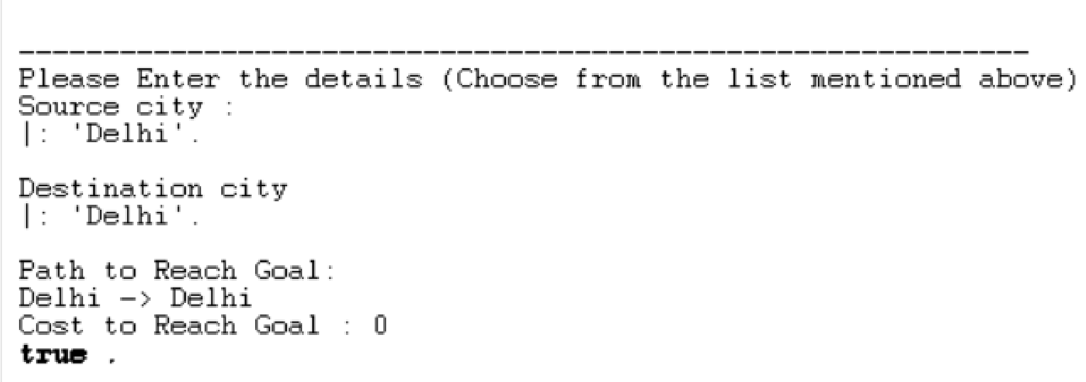
```
-------------------------------------------------------------
Please Enter the details (Choose from the list mentioned above)
Source city :
|: 'Surat'.

Destination city
|: 'Shimla'.

Path to Reach Goal:
Surat -> Ahmedabad -> Shimla
Cost to Reach Goal : 1519
true .
```



```
-------------------------------------------------------------
Please Enter the details (Choose from the list mentioned above)
Source city :
|: 'Hubli'.

Destination city
|: 'Madras'.

Path to Reach Goal:
Hubli -> Madras
Cost to Reach Goal : 683
true .
```

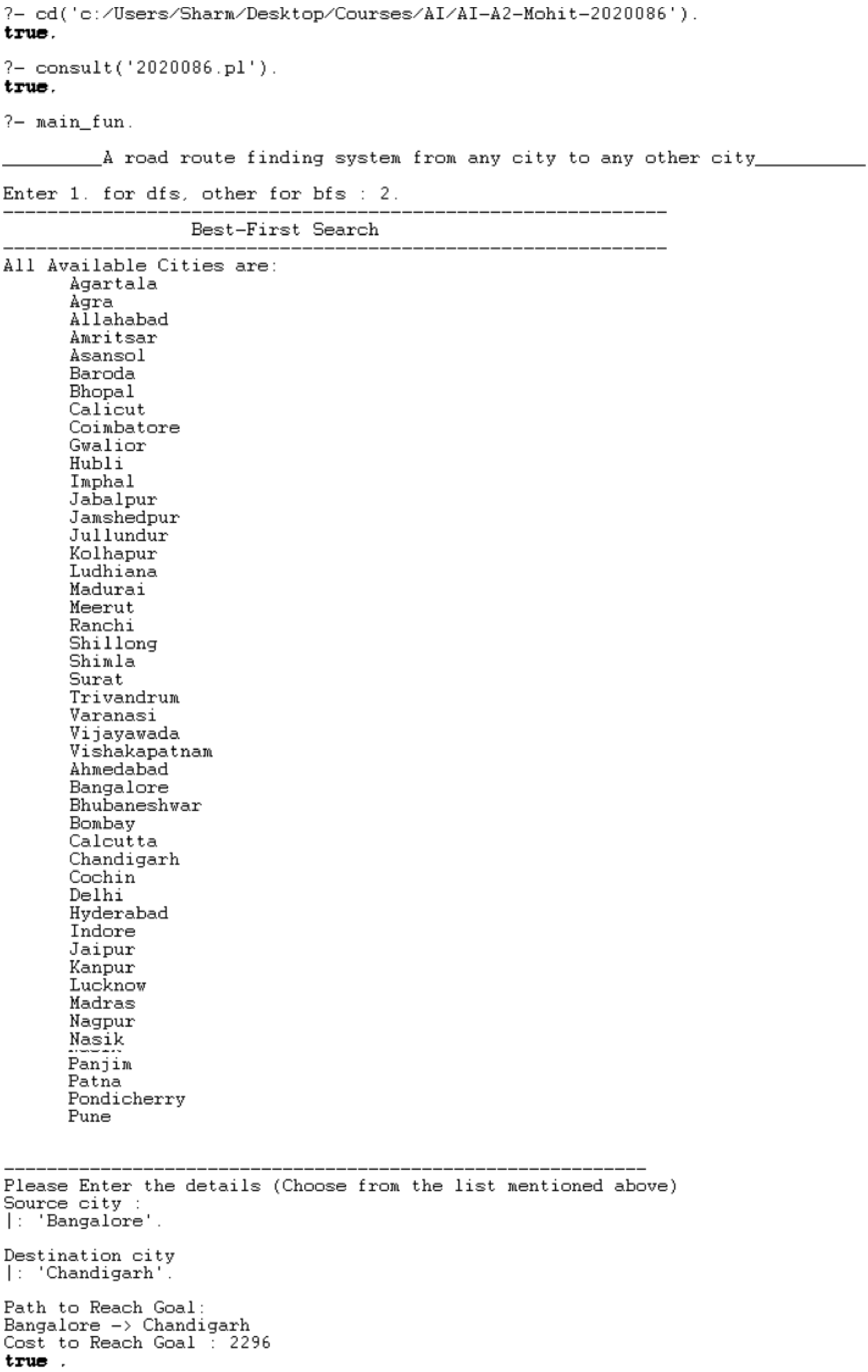


```
-------------------------------------------------------------
Please Enter the details (Choose from the list mentioned above)
Source city :
|: 'Delhi'.

Destination city
|: 'Varanasi'.

Path to Reach Goal:
Delhi -> Varanasi
Cost to Reach Goal : 745
true .
```

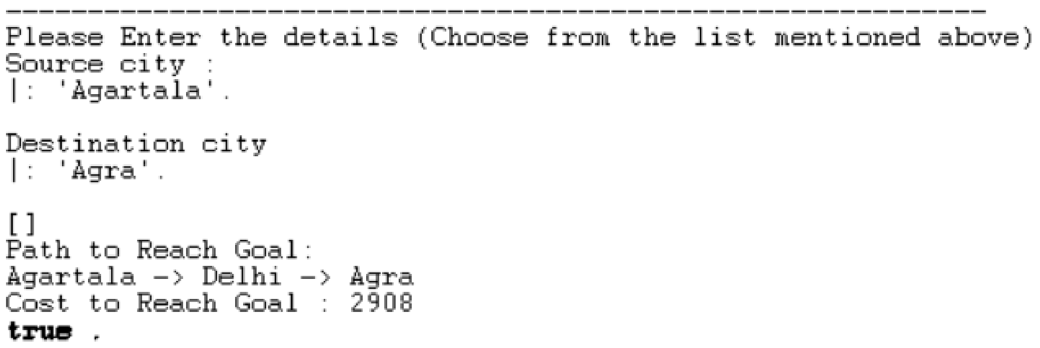


```
-------------------------------------------------------------
Please Enter the details (Choose from the list mentioned above)
Source city :
|: 'Delhi'.

Destination city
|: 'Delhi'.

Path to Reach Goal:
Delhi -> Delhi
Cost to Reach Goal : 0
true .
```

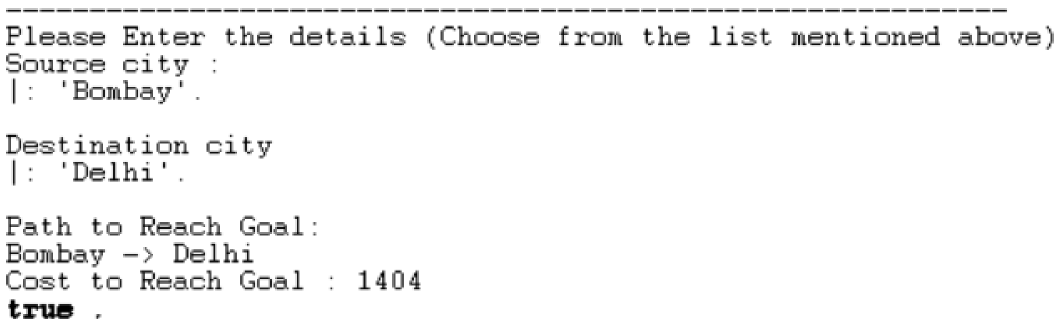
Screenshots: (bfs)



```
?- cd('c:/Users/Sharm/Desktop/Courses/AI/AI-A2-Mohit-2020086').
true.

?- consult('2020086.pl').
true.

?- main_fun.

_________A road route finding system from any city to any other city__________

Enter 1. for dfs, other for bfs : 2.
-------------------------------------------------------------
                 Best-First Search
-------------------------------------------------------------
All Available Cities are:
      Agartala
      Agra
      Allahabad
      Amritsar
      Asansol
      Baroda
      Bhopal
      Calicut
      Coimbatore
      Gwalior
      Hubli
      Imphal
      Jabalpur
      Jamshedpur
      Jullundur
      Kolhapur
      Ludhiana
      Madurai
      Meerut
      Ranchi
      Shillong
      Shimla
      Surat
      Trivandrum
      Varanasi
      Vijayawada
      Vishakapatnam
      Ahmedabad
      Bangalore
      Bhubaneshwar
      Bombay
      Calcutta
      Chandigarh
      Cochin
      Delhi
      Hyderabad
      Indore
      Jaipur
      Kanpur
      Lucknow
      Madras
      Nagpur
      Nasik
      Panjim
      Patna
      Pondicherry
      Pune


-----------------------------------------------------------
Please Enter the details (Choose from the list mentioned above)
Source city :
|: 'Bangalore'.

Destination city
|: 'Chandigarh'.

Path to Reach Goal:
Bangalore -> Chandigarh
Cost to Reach Goal : 2296
true .
```



```
-------------------------------------------------------------
Please Enter the details (Choose from the list mentioned above)
Source city :
|: 'Agartala'.

Destination city
|: 'Agra'.

[]
Path to Reach Goal:
Agartala -> Delhi -> Agra
Cost to Reach Goal : 2908
true .
```



```
-------------------------------------------------------------
Please Enter the details (Choose from the list mentioned above)
Source city :
|: 'Bombay'.

Destination city
|: 'Delhi'.

Path to Reach Goal:
Bombay -> Delhi
Cost to Reach Goal : 1404
true .
```