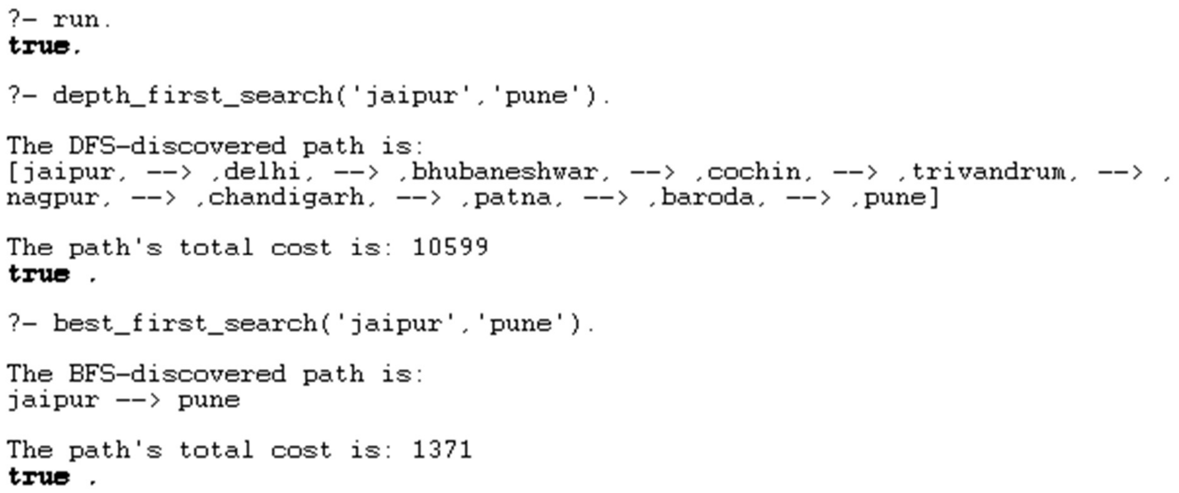
AI Assignment2

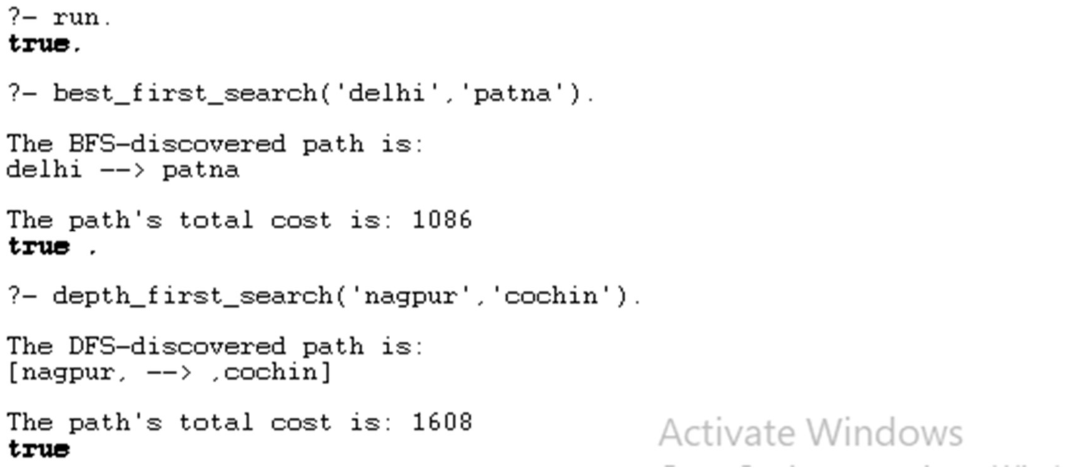
Yearagra Paliwal (SP22004)

<u>Example 1:</u>



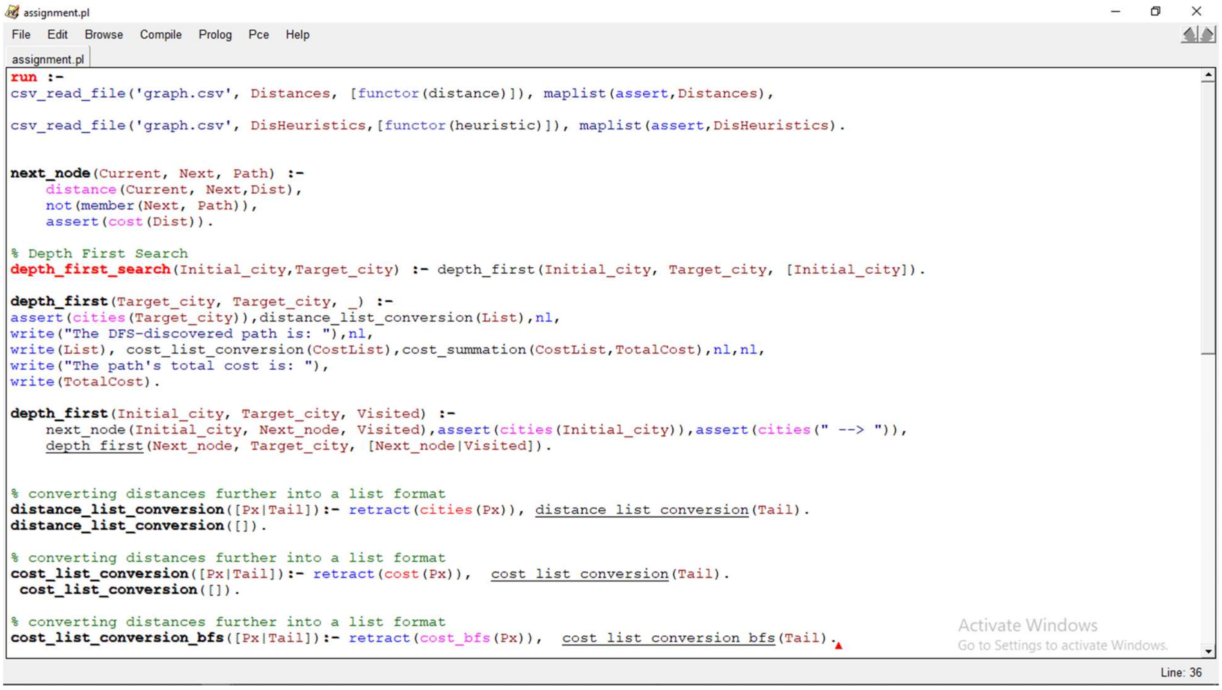
```
?- run.
true.

?- depth_first_search('jaipur','pune').

The DFS-discovered path is:
[jaipur, --> ,delhi, --> ,bhubaneshwar, --> ,cochin, --> ,trivandrum, --> ,
nagpur, --> ,chandigarh, --> ,patna, --> ,baroda, --> ,pune]

The path's total cost is: 10599
true .

?- best_first_search('jaipur','pune').

The BFS-discovered path is:
jaipur --> pune

The path's total cost is: 1371
true .
```

<u>Example 2:</u>

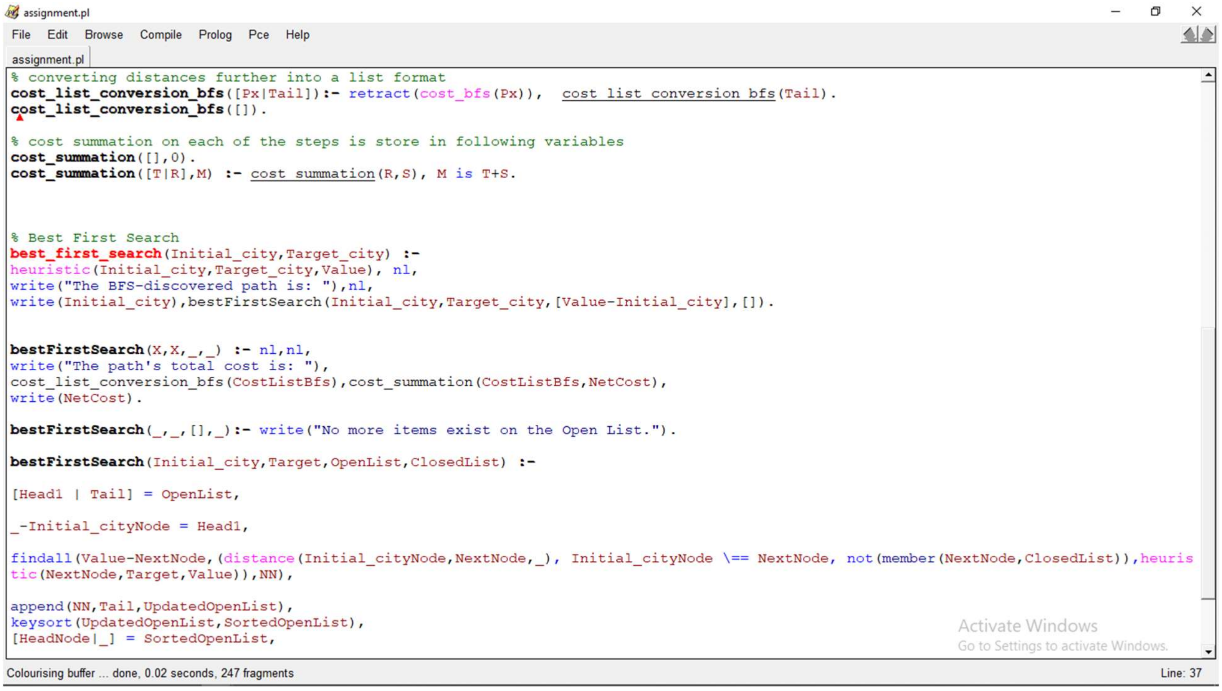


```
?- run.
true.

?- best_first_search('delhi','patna').

The BFS-discovered path is:
delhi --> patna

The path's total cost is: 1086
true .

?- depth_first_search('nagpur','cochin').

The DFS-discovered path is:
[nagpur, --> ,cochin]

The path's total cost is: 1608
true
```

Code:

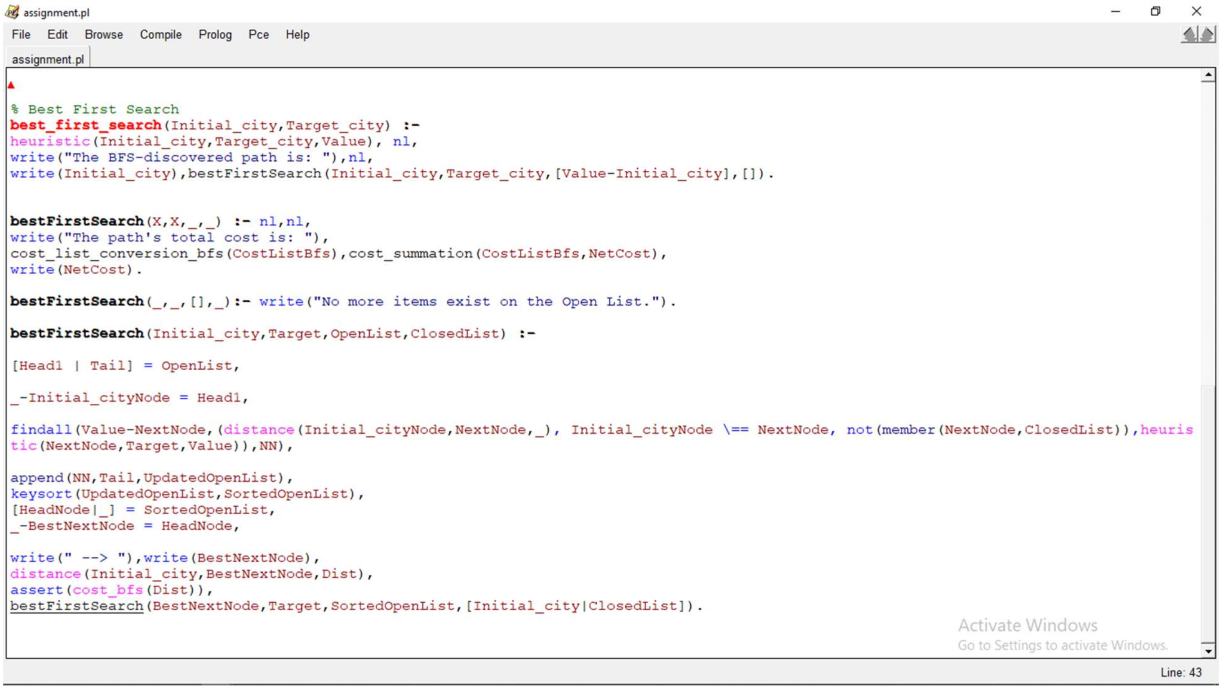

assignment.pl
File Edit Browse Compile Prolog Pce Help
assignment.pl
Activate Windows
Go to Settings to activate Windows.
Line: 36

```prolog
run :-
csv_read_file('graph.csv', Distances, [functor(distance)]), maplist(assert,Distances),

csv_read_file('graph.csv', DisHeuristics,[functor(heuristic)]), maplist(assert,DisHeuristics).


next_node(Current, Next, Path) :-
    distance(Current, Next,Dist),
    not(member(Next, Path)),
    assert(cost(Dist)).

% Depth First Search
depth_first_search(Initial_city,Target_city) :- depth_first(Initial_city, Target_city, [Initial_city]).

depth_first(Target_city, Target_city, _) :-
assert(cities(Target_city)),distance_list_conversion(List),nl,
write("The DFS-discovered path is: "),nl,
write(List), cost_list_conversion(CostList),cost_summation(CostList,TotalCost),nl,nl,
write("The path's total cost is: "),
write(TotalCost).

depth_first(Initial_city, Target_city, Visited) :-
    next_node(Initial_city, Next_node, Visited),assert(cities(Initial_city)),assert(cities(" --> ")),
    depth_first(Next_node, Target_city, [Next_node|Visited]).


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

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

% converting distances further into a list format
cost_list_conversion_bfs([Px|Tail]):- retract(cost_bfs(Px)),  cost_list_conversion_bfs(Tail).
```


assignment.pl
File Edit Browse Compile Prolog Pce Help
assignment.pl
Activate Windows
Go to Settings to activate Windows.
Line: 37

```prolog
% converting distances further into a 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
best_first_search(Initial_city,Target_city) :-
heuristic(Initial_city,Target_city,Value), nl,
write("The BFS-discovered path is: "),nl,
write(Initial_city),bestFirstSearch(Initial_city,Target_city,[Value-Initial_city],[]).


bestFirstSearch(X,X,_,_) :- nl,nl,
write("The path's total cost is: "),
cost_list_conversion_bfs(CostListBfs),cost_summation(CostListBfs,NetCost),
write(NetCost).

bestFirstSearch(_,_,[],_):- write("No more items exist on the Open List.").

bestFirstSearch(Initial_city,Target,OpenList,ClosedList) :-

[Head1 | Tail] = OpenList,

_-Initial_cityNode = Head1,

findall(Value-NextNode,(distance(Initial_cityNode,NextNode,_), Initial_cityNode \== NextNode, not(member(NextNode,ClosedList)),heuris
tic(NextNode,Target,Value)),NN),

append(NN,Tail,UpdatedOpenList),
keysort(UpdatedOpenList,SortedOpenList),
[HeadNode|_] = SortedOpenList,
```

Colourising buffer ... done, 0.02 seconds, 247 fragments



```prolog
% Best First Search
best_first_search(Initial_city,Target_city) :-
heuristic(Initial_city,Target_city,Value), nl,
write("The BFS-discovered path is: "),nl,
write(Initial_city),bestFirstSearch(Initial_city,Target_city,[Value-Initial_city],[]).


bestFirstSearch(X,X,_,_) :- nl,nl,
write("The path's total cost is: "),
cost_list_conversion_bfs(CostListBfs),cost_summation(CostListBfs,NetCost),
write(NetCost).

bestFirstSearch(_,_,[],_):- write("No more items exist on the Open List.").

bestFirstSearch(Initial_city,Target,OpenList,ClosedList) :-

[Head1 | Tail] = OpenList,

_-Initial_cityNode = Head1,

findall(Value-NextNode,(distance(Initial_cityNode,NextNode,_), Initial_cityNode \== NextNode, not(member(NextNode,ClosedList)),heuris
tic(NextNode,Target,Value)),NN),

append(NN,Tail,UpdatedOpenList),
keysort(UpdatedOpenList,SortedOpenList),
[HeadNode|_] = SortedOpenList,
_-BestNextNode = HeadNode,

write(" --> "),write(BestNextNode),
distance(Initial_city,BestNextNode,Dist),
assert(cost_bfs(Dist)),
bestFirstSearch(BestNextNode,Target,SortedOpenList,[Initial_city|ClosedList]).
```

Thank You