Shri Vile Parle Kelavani Mandal's

INSTITUTE OF TECHNOLOGY

DHULE (M.S.)

DEPARMENT OF COMPUTER ENGINEERING

Subject : Artificial Intelligence Lab

Name : Mohammad Anas Aarif Baig Mirza **Roll No. :** 40

Class : B.tech Final Year **Batch : B2** **Division: -**

Expt. No. :05 **Date : 11/09/2025**

Title : Solve 8-puzzle problem using best first search.

Remark

Signature

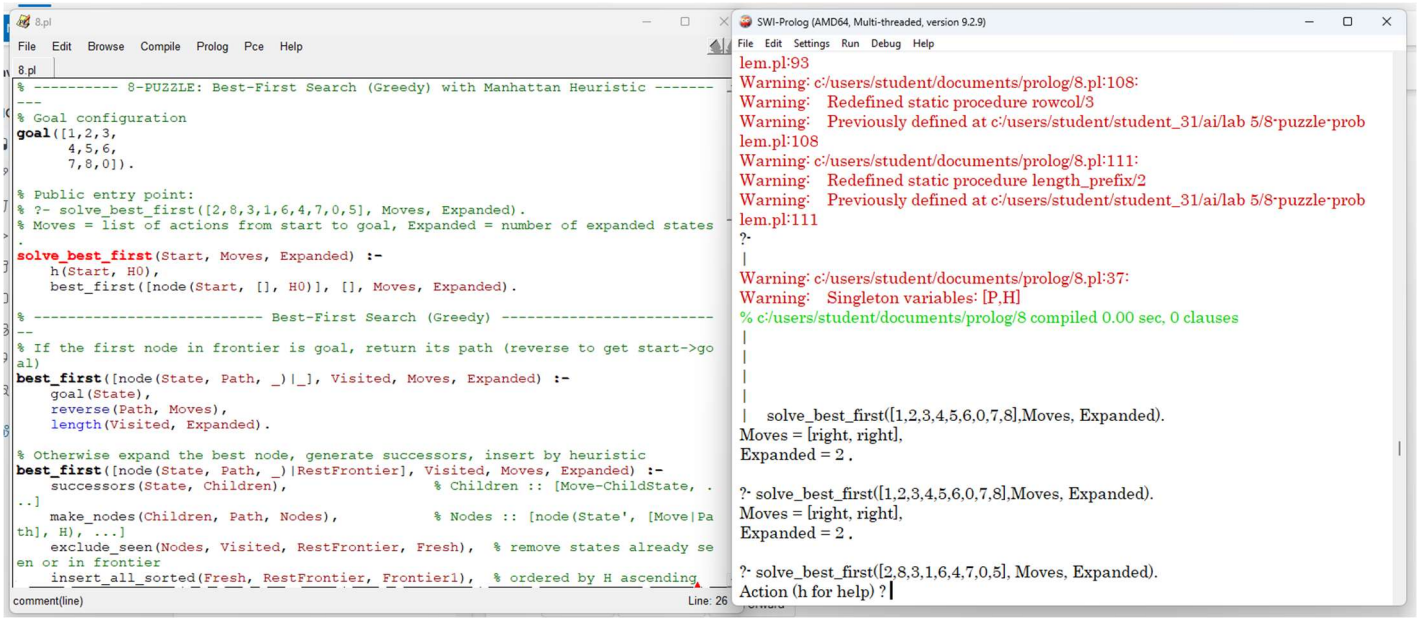
Code:

```
% ---------- 8-PUZZLE: Best-First Search (Greedy) with Manhattan Heuristic ----------
% Goal configuration
goal([1,2,3,
      4,5,6,
      7,8,0]).

% Public entry point:
% ?- solve_best_first([2,8,3,1,6,4,7,0,5], Moves, Expanded).
% Moves = list of actions from start to goal, Expanded = number of expanded states.
solve_best_first(Start, Moves, Expanded) :-
    h(Start, H0),
    best_first([node(Start, [], H0)], [], Moves, Expanded).

% --------------------------- Best-First Search (Greedy) ---------------------------
% If the first node in frontier is goal, return its path (reverse to get start->goal)
best_first([node(State, Path, _)|_], Visited, Moves, Expanded) :-
    goal(State),
Solve 8-puzzle problem using best first search.Solve 8-puzzle problem using best first
search.    reverse(Path, Moves),
    length(Visited, Expanded).

% Otherwise expand the best node, generate successors, insert by heuristic
best_first([node(State, Path, _)|RestFrontier], Visited, Moves, Expanded) :-
    successors(State, Children),                % Children :: [Move-ChildState, ...]
    make_nodes(Children, Path, Nodes),          % Nodes :: [node(State', [Move|Path], H), ...]
    exclude_seen(Nodes, Visited, RestFrontier, Fresh),  % remove states already seen or in frontier
    insert_all_sorted(Fresh, RestFrontier, Frontier1),  % ordered by H ascending
```

```
    best_first(Frontier1, [State|Visited], Moves, Expanded).

% Build nodes with updated paths and heuristic
make_nodes([], _, []).
make_nodes([Move-S|T], Path, [node(S, [Move|Path], H)|NT]) :-
    h(S, H),
    make_nodes(T, Path, NT).

% Filter out nodes whose states are already visited or present in the frontier
exclude_seen([], _, _, []).
exclude_seen([node(S,P,H)|T], Visited, Frontier, R) :-
    ( member(S, Visited) ; member_state(S, Frontier) ),
    !, exclude_seen(T, Visited, Frontier, R).
exclude_seen([N|T], Visited, Frontier, [N|R]) :-
    exclude_seen(T, Visited, Frontier, R).

% Check if a state exists inside a frontier (list of node/3)
member_state(S, [node(S,_,_)|_]).
member_state(S, [_|T]) :- member_state(S, T).

% Insert a list of nodes into a frontier (ordered by H ascending)
insert_all_sorted([], F, F).
insert_all_sorted([N|T], F, R) :-
    insert_sorted(N, F, F1),
    insert_all_sorted(T, F1, R).

insert_sorted(node(S,P,H), [], [node(S,P,H)]).
insert_sorted(node(S, P, H), [node(S2, P2, H2)|T], [node(S, P, H), node(S2, P2, H2)|T]) :-
    H =< H2, !.
insert_sorted(N, [Hn|T], [Hn|R]) :-
    insert_sorted(N, T, R).

% ---------------------------- Successor Generation -------------------------------
% All legal moves from a state with corresponding next states
successors(State, MovesStates) :-
    findall(Move-S2, move(State, Move, S2), MovesStates).

% Moves: up, down, left, right (when legal)
move(S, up,    S2) :- index0(0, S, I), I > 2,    J is I-3, swap0(S, I, J, S2).
move(S, down,  S2) :- index0(0, S, I), I < 6,    J is I+3, swap0(S, I, J, S2).
move(S, left,  S2) :- index0(0, S, I), I mod 3 =\= 0, J is I-1, swap0(S, I, J, S2).
move(S, right, S2) :- index0(0, S, I), I mod 3 =\= 2, J is I+1, swap0(S, I, J, S2).

% Find index of element E in list (0-based)
index0(E, [E|_], 0) :- !.
index0(E, [_|T], I) :- index0(E, T, I1), I is I1 + 1.
```

```
% Swap elements at indices I and J (0-based)
swap0(L, I, J, R) :-
   nth0(I, L, Ei),
   nth0(J, L, Ej),
   set_nth0(L, I, Ej, L1),
   set_nth0(L1, J, Ei, R).

% Replace element at index I with X producing R
set_nth0([_|T], 0, X, [X|T]).
set_nth0([H|T], I, X, [H|R]) :- I > 0, I1 is I - 1, set_nth0(T, I1, X, R).

% -------------------------------- Heuristic (h) ----------------------------------
% Manhattan distance for all tiles (ignoring 0)
h(State, H) :- manhattan(State, H).

manhattan(State, Sum) :-
   goal(Goal),
   manhattan_sum(State, Goal, 0, Sum).

manhattan_sum([], _, Acc, Acc).
manhattan_sum([0|T], Goal, Acc, Sum) :-          % skip blank
   manhattan_sum(T, Goal, Acc, Sum).
manhattan_sum([X|T], Goal, Acc, Sum) :-
   index0(X, [1,2,3,4,5,6,7,8,0], GoalIdx),      % goal position of tile X is fixed
   index0(X, [X|T], 0),                          % current head is X at relative index 0
   length_prefix(T, N),                          % compute absolute index of current X
   CurrIdx is 8 - N,                             % 8 - remaining length = absolute index
   rowcol(CurrIdx, CR, CC),
   rowcol(GoalIdx, GR, GC),
   D is abs(CR - GR) + abs(CC - GC),
   Acc1 is Acc + D,
   manhattan_sum(T, Goal, Acc1, Sum).

% Helper: row/col from absolute index
rowcol(I, R, C) :- R is I // 3, C is I mod 3.

% Helper: number of remaining elements
length_prefix(L, N) :- length(L, N).
```

Output:


8.pl
File Edit Browse Compile Prolog Pce Help
8.pl
% ---------- 8-PUZZLE: Best-First Search (Greedy) with Manhattan Heuristic -------
---
% Goal configuration
goal([1,2,3,
4,5,6,
7,8,0]).
% Public entry point:
% ?- solve_best_first([2,8,3,1,6,4,7,0,5], Moves, Expanded).
% Moves = list of actions from start to goal, Expanded = number of expanded states
.
solve_best_first(Start, Moves, Expanded) :-
h(Start, H0),
best_first([node(Start, [], H0)], [], Moves, Expanded).
% -------------------------- Best-First Search (Greedy) ------------------------
--
% If the first node in frontier is goal, return its path (reverse to get start->go
al)
best_first([node(State, Path, _)|_], Visited, Moves, Expanded) :-
goal(State),
reverse(Path, Moves),
length(Visited, Expanded).
% Otherwise expand the best node, generate successors, insert by heuristic
best_first([node(State, Path, _)|RestFrontier], Visited, Moves, Expanded) :-
successors(State, Children), % Children :: [Move-ChildState, .
..]
make_nodes(Children, Path, Nodes), % Nodes :: [node(State', [Move|Pa
th], H), ...]
exclude_seen(Nodes, Visited, RestFrontier, Fresh), % remove states already se
en or in frontier
insert_all_sorted(Fresh, RestFrontier, Frontier1), % ordered by H ascending
comment(line)
Line: 26
SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help
lem.pl:93
Warning: c:/users/student/documents/prolog/8.pl:108:
Warning: Redefined static procedure rowcol/3
Warning: Previously defined at c:/users/student/student_31/ai/lab 5/8-puzzle-prob
lem.pl:108
Warning: c:/users/student/documents/prolog/8.pl:111:
Warning: Redefined static procedure length_prefix/2
Warning: Previously defined at c:/users/student/student_31/ai/lab 5/8-puzzle-prob
lem.pl:111
?-
|
Warning: c:/users/student/documents/prolog/8.pl:37:
Warning: Singleton variables: [P,H]
% c:/users/student/documents/prolog/8 compiled 0.00 sec, 0 clauses
|
|
|
|
| solve_best_first([1,2,3,4,5,6,0,7,8],Moves, Expanded).
Moves = [right, right],
Expanded = 2 .
?- solve_best_first([1,2,3,4,5,6,0,7,8],Moves, Expanded).
Moves = [right, right],
Expanded = 2 .
?- solve_best_first([2,8,3,1,6,4,7,0,5], Moves, Expanded).
Action (h for help) ?