|  |  |
| --- | --- |
| Shri Vile Parle Kelavani Mandal's  **INSTITUTE OF TECHNOLOGY**  **DHULE (M.S.)**  **DEPARMENT OF COMPUTER ENGINEERING** | |
| **Name:** Anushka Paras Jain **Roll no.:** 01  **Subject :** Artificial Intelligence Lab **Subject Code :** BTCOL707  **Class:** Final Year Comp. Engg. **Expt. No. :** 04  **Title :** Solve any problem using best first search. | |
| **Problem Statement:**  **Software Required :**  **Theory :**  **Conclusion:** | Solve any problem using best first search.    Prolog  A broad search algorithm called Best-First Search (BFS) uses a heuristic evaluation function to guide it across a search space. Let's use Prolog's Best-First Search technique to solve the well-known "8-puzzle" problem. To get from a starting state to a target state in the 8-puzzle, you must rearrange numbered tiles in a 3x3 grid.  % Define the initial state and goal state  initial\_state([2, 8, 3, 1, 6, 4, 7, 0, 5]).  goal\_state([1, 2, 3, 8, 0, 4, 7, 6, 5]).  % Define the heuristic function (Manhattan distance)  heuristic(State, H) :-  goal\_state(Goal),  findall(D, (nth1(I, State, Tile), nth1(I, Goal, GoalTile), manhattan(Tile, GoalTile, D)), Distances),  sum\_list(Distances, H).  manhattan(X/Y, X1/Y1, D) :-  D is abs(X - X1) + abs(Y - Y1).  % Operators to move tiles  move(State, NewState) :-  select(0, State, X, TempState),  select(T, TempState, 0, NewTempState),  append([X, T], NewTempState, NewState).  % Define a predicate to solve the puzzle using Best-First Search  solve\_best\_first(State, State, [], \_).  solve\_best\_first(CurrentState, GoalState, [Action | Actions], Visited) :-  findall((NewState, Action, H), (  move(CurrentState, NewState),  \+ member(NewState, Visited),  heuristic(NewState, H)  ), Successors),  keysort(Successors, SortedSuccessors),  member((NextState, Action, \_), SortedSuccessors),  solve\_best\_first(NextState, GoalState, Actions, [NextState | Visited]).  % Entry point to solve the puzzle  solve\_puzzle :-  initial\_state(InitialState),  goal\_state(GoalState),  solve\_best\_first(InitialState, GoalState, Actions, [InitialState]),  write('Solution Actions: '), nl,  print\_actions(Actions).  % Predicate to print the sequence of actions  print\_actions([]).  print\_actions([Action | Rest]) :-  print\_state(Action),  print\_actions(Rest).  % Predicate to print a single state  print\_state([A, B, C, D, E, F, G, H, I]) :-  format('~d ~d ~d~n~d ~d ~d~n~d ~d ~d~n', [A, B, C, D, E, F, G, H, I]).  % Start the solver  :- solve\_puzzle.  This code solves the 8-puzzle problem using Best-First Search with the Manhattan distance heuristic. It calculates the heuristic value for each state and explores the states with the lowest heuristic values first. The solve\_puzzle predicate initiates the search and prints the sequence of actions to reach the goal state from the initial state.  This practical illustrated the application of BFS in Prolog for solving pathfinding problems. The graph was represented as a list of lists, and the Manhattan distance heuristic was used to estimate the distance between two positions. The BFS algorithm was implemented to find the shortest path from the start position to the goal, ensuring optimal solutions. |