Artificial intelligence

Lab - 1

Shubham Sharma
RA1911003010649

8 Puzzle Problem

⇒ Start State

1	5	3
2	4	0
8	7	6

Goal State

1	2	3
4	5	6
7	8	

Algorithm →

```
① Define a function find_next() that accepts a node.
② moves == map defining moves as a list corresponding
   to each value {0: [1,3], 1: [0,2,4], 2: [1,5]; 3: [0,4,6],
                  4: [1,3,5,7], 5: [2,4,8], 6: [3,7], 7: [4,6,8]
                  8: [5,7]}
③ results = a new list.
④   pos_0 = first value of node.
⑤ for each move in moves [pos_0] do
        • new_node = a new list from node.
        • swap new_node [move] and new node [pos_0]
        • insert a new tuple from new node at the
          end of result.
⑥ return result.
⑦ Define a function get_paths(). This will take dict.
```

```
cnt: =0
8  Do the following infintely, do.
9      • current-nodes := a list where value is same
             as cnt
       • if size of current-nodes is same as 0, then
             return -1;

       • for each node in current_nodes, do
           • next_moves := find_next(node)
           • for each move in next_move, do.
               • if move is not present in dict,
                   dict[move] := cnt + 1
               • if move is same as(0,1,2,3,4,5,6,7,8) then
                   return cnt + 1
• From the main method do the following:
• dict: = a new map, flatten := a new list.
• for i in range 0 to row count of board, do
      • flatten:= flatten + board [i]

• flatten := a copy of listen.
• dict [flatten] := 0.
• if flatten is same as (0,1,2,3,4,5,6,7,8), then
      • return 0.
• return get-paths(dict)
```

Result :- Hence, the implementation of 8 puzzle problem is Succesfully executed

Scanned by TapScanner