9/11/23

Lab program

3) 8- Puzzle Iterative deepening search algorithm.

Algorithm

Step 1: Define puzzle node class. each node represents a state of the puzzle.

Step 2: Define helper functions

get_blank_position(state)
get_neighbors(node)
apply_action(state, action)
print_solution(solution)

Step 3: Depth limited search function

depth_limited_search(node, goal_state, depth_limit)

Step 4: Iterative deepening search function

iterative_deepening_search(initial_state, goal_state)

Step 5: Main function.

code

```
import copy

GOAL_STATE = [[1,2,3], [8,0,4], [7,6,5]]
MOVE_UP, MOVE_DOWN, MOVE_LEFT, MOVE_RIGHT
    = 'U', 'D', 'L', 'R'
MOVES = [MOVE_UP, MOVE_DOWN, MOVE_LEFT,
         MOVE_RIGHT]

def print_puzzle(state):
    for row in state:
        print(row)
    print()

def find_blank(state):
    for i in range(3):
        for j in range(3):
            if state[i][j] == 0:
                return i, j

def move(state, direction):
    i, j = find_blank(state)
    new_state = copy.deepcopy(state)

    if direction == MOVE_UP and i > 0:
        new_state[i][j], new_state[i-1][j] =
            new_state[i-1][j], new_state[i][j]
    elseif direction == MOVE_DOWN and i < 2:
        new_state[i][j], new_state[i+1][j] =
            new_state[i+1][j], new_state[i][j]

    return new_state
```

```
def is_goal(state):
    return state == GOAL_STATE

def depth_limited_search(state, depth, path):
    if depth == 0:
        return None

    if is_goal(state):
        return path

    for move direction in MOVES:
        newstate = move(state, move_direction)
        if new_state not in path:
            new_path = path + [new_state]
            result = depth_limited_search
                (new_state, depth-1,
                    new path)

    return None

def iterative_deepening_search(initial
        _state):
    depth = 0
    while True:
        result = depth_limited_search(
                initial state, depth, [initial state])
        if result is not None:
            return result
        depth += 1

initial state = [[1,2,3],[0,8,4],[7,6,5]]
```

```
if solution_path is not None:
    print("Solution found!")
else:
    print("No solution found")
```

Output

```
Initial State:
[1, 2, 3]
[0, 8, 4]
[7, 6, 5]

Solution found:

step 1:
[1, 2, 3]
[0, 8, 4]
[7, 6, 5]

step 2:
[1, 2, 3]
[8, 0, 4]
[7, 6, 5]
```