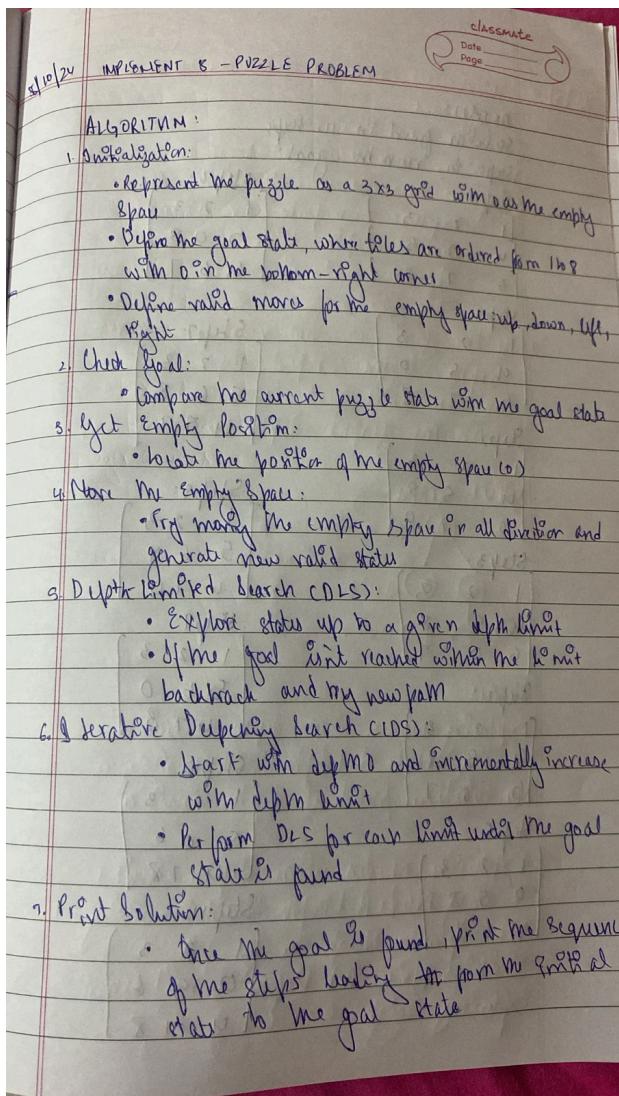


Date: 8/10/24

Program Title: Implement 8 Puzzle Problem

Algorithm:



OOPS!!

Solution found in 11 steps
Steps to reach the goal:

Step 0:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 0 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

Step 6:

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 0 & 8 \\ 4 & 6 & 7 \end{bmatrix}$$

Step 1:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 0 \\ 6 & 7 & 8 \end{bmatrix}$$

Step 7:

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 8 \\ 4 & 0 & 7 \end{bmatrix}$$

Step 2:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 8 \\ 6 & 7 & 0 \end{bmatrix}$$

Step 8:

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 8 \\ 4 & 7 & 0 \end{bmatrix}$$

Step 3:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 8 \\ 6 & 0 & 7 \end{bmatrix}$$

Step 9:

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 0 \\ 4 & 7 & 8 \end{bmatrix}$$

Step 4:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 8 \\ 6 & 0 & 7 \end{bmatrix}$$

Step 10:

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 0 & 6 \\ 4 & 7 & 8 \end{bmatrix}$$

Step 5:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 8 \\ 4 & 6 & 7 \end{bmatrix}$$

Step 11:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 4 & 7 & 8 \end{bmatrix}$$

Date _____
Page _____

Step 12:	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 7 & 8 \end{bmatrix}$
Step 13:	$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 7 & 0 & 8 \end{bmatrix}$
Step 14:	$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 2 & 8 & 0 \end{bmatrix}$

Code:

```

import copy
moves = {'up': (-1, 0), 'down': (1, 0), 'left': (0, -1), 'right': (0, 1)}
def is_goal(state, goal_state):
    return state == goal_state
def get_empty_position(state):
    for i in range(3):
        for j in range(3):
            if state[i][j] == 0:
                return i, j
def move_tile(state, direction):
    new_state = copy.deepcopy(state)
    empty_i, empty_j = get_empty_position(state)
    di, dj = moves[direction]
    new_i, new_j = empty_i + di, empty_j + dj
    if 0 <= new_i < 3 and 0 <= new_j < 3:
        new_state[empty_i][empty_j], new_state[new_i][new_j] = new_state[new_i][new_j], new_state[empty_i][empty_j]
    return new_state
    return None
def depth_limited_search(state, goal_state, depth_limit, path):
    if is_goal(state, goal_state):
        return state, path
    if depth_limit == 0:
        return None, []
    empty_i, empty_j = get_empty_position(state)
    for direction in moves:
        new_state = move_tile(state, direction)
        if new_state is not None and new_state not in path: # Avoid loops
            result, new_path = depth_limited_search(new_state, goal_state, depth_limit - 1, path + [new_state])
            if result:
                return result, new_path
    return None, []
def iterative_deepening_search(initial_state, goal_state):
    """
    def iterative_deepening_search(initial_state, goal_state):
        depth = 0
        while True:
            result, path = depth_limited_search(initial_state, goal_state, depth, [initial_state])
            if result is not None:
                return path, depth
            depth += 1
    def print_state(state):
        for row in state:
            print(row)
        print()
    initial_state = [
        [1, 2, 3],
        [4, 0, 5],
        [6, 7, 8]
    ]
    goal_state = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 0]
    ]
    solution_path, depth = iterative_deepening_search(initial_state, goal_state)
    print(f"Solution found in {depth} steps.\n")
    print("Steps to reach the goal:")
    for i, state in enumerate(solution_path):
        print(f"Step {i}:")
        print_state(state)
    import copy
    moves = {'up': (-1, 0), 'down': (1, 0), 'left': (0, -1), 'right': (0, 1)}  


```

```

import copy
moves = {'up': (-1, 0), 'down': (1, 0), 'left': (0, -1), 'right': (0, 1)}  


```

```

def is_goal(state, goal_state):
    return state == goal_state
def get_empty_position(state):
    for i in range(3):
        for j in range(3):
            if state[i][j] == 0:
                return i, j
def move_tile(state, direction):
    new_state = copy.deepcopy(state)
    empty_i, empty_j = get_empty_position(state)
    di, dj = moves[direction]
    new_i, new_j = empty_i + di, empty_j + dj
    if 0 <= new_i < 3 and 0 <= new_j < 3:
        new_state[empty_i][empty_j], new_state[new_i][new_j] = new_state[new_i][new_j],
        new_state[empty_i][empty_j]
    return new_state
    return None
def depth_limited_search(state, goal_state, depth_limit, path):
    if is_goal(state, goal_state):
        return state, path
    if depth_limit == 0:
        return None, []
    empty_i, empty_j = get_empty_position(state)
    for direction in moves:
        new_state = move_tile(state, direction)
        if new_state is not None and new_state not in path: # Avoid loops
            result, new_path = depth_limited_search(new_state, goal_state, depth_limit - 1, path +
            [new_state])
            if result:
                return result, new_path
    return None, []
def iterative_deepening_search(initial_state, goal_state):
    depth = 0
    while True:
        result, path = depth_limited_search(initial_state, goal_state, depth, [initial_state])
        if result is not None:
            return path, depth
        depth += 1
def print_state(state):
    for row in state:
        print(row)
    print()
initial_state = [
    [1, 2, 3],

```

```
[4, 0, 5],  
[6, 7, 8]  
]  
goal_state = [  
    [1, 2, 3],  
    [4, 5, 6],  
    [7, 8, 0]  
]  
solution_path, depth = iterative_deepening_search(initial_state, goal_state)  
print(f"Solution found in {depth} steps.\n")  
print("Steps to reach the goal:")  
for i, state in enumerate(solution_path):  
    print(f"Step {i}:")  
    print_state(state)
```

Output:

	Step 6:	
→ Solution found in 14 steps.	[1, 2, 3]	
	[5, 0, 8]	
	[4, 6, 7]	
Steps to reach the goal:		
Step 0:	Step 7:	
[1, 2, 3]	[1, 2, 3]	
[4, 0, 5]	[5, 6, 8]	
[6, 7, 8]	[4, 0, 7]	
Step 1:	Step 8:	
[1, 2, 3]	[1, 2, 3]	
[4, 5, 0]	[5, 6, 8]	
[6, 7, 8]	[4, 7, 0]	
Step 2:	Step 9:	
[1, 2, 3]	[1, 2, 3]	
[4, 5, 8]	[5, 6, 0]	
[6, 7, 0]	[4, 7, 8]	
Step 3:	Step 10:	
[1, 2, 3]	[1, 2, 3]	
[4, 5, 8]	[5, 0, 6]	
[6, 0, 7]	[4, 7, 8]	
Step 4:	Step 11:	Step 13:
[1, 2, 3]	[1, 2, 3]	[1, 2, 3]
[4, 5, 8]	[0, 5, 6]	[4, 5, 6]
[0, 6, 7]	[4, 7, 8]	[7, 0, 8]
Step 5:	Step 12:	Step 14:
[1, 2, 3]	[1, 2, 3]	[1, 2, 3]
[0, 5, 8]	[4, 5, 6]	[4, 5, 6]
[4, 6, 7]	[0, 7, 8]	[7, 8, 0]

OOPS!!

Solution found in 11 steps
Steps to reach the goal:

Step 0:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 0 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

Step 6:

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 0 & 8 \\ 4 & 6 & 7 \end{bmatrix}$$

Step 1:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 0 \\ 6 & 7 & 8 \end{bmatrix}$$

Step 7:

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 8 \\ 4 & 0 & 7 \end{bmatrix}$$

Step 2:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 8 \\ 6 & 7 & 0 \end{bmatrix}$$

Step 8:

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 8 \\ 4 & 7 & 0 \end{bmatrix}$$

Step 3:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 8 \\ 6 & 0 & 7 \end{bmatrix}$$

Step 9:

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 0 \\ 4 & 7 & 8 \end{bmatrix}$$

Step 4:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 8 \\ 6 & 0 & 7 \end{bmatrix}$$

Step 10:

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 0 & 6 \\ 4 & 7 & 8 \end{bmatrix}$$

Step 5:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 8 \\ 4 & 6 & 7 \end{bmatrix}$$

Step 11:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 4 & 7 & 8 \end{bmatrix}$$

Step 12:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 7 & 8 \end{bmatrix}$$

Class
Date _____
Page _____

Step 13:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 7 & 0 & 8 \end{bmatrix}$$

Step 14:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$