Question:1

8 PUZZLE PROBLEM

AIM

To solve the 8 Puzzle problem using python

ALGORITHM

1. solve_puzzle(): Use A* search algorithm.
    1) Initialize an open list with the initial state, and a closed set to track visited states.
    2) While the open list is not empty:
        - Pop the state with the lowest path cost from the open list.
        - If the current state is the goal state, return the solution path.
        - Generate neighboring states by swapping the empty tile with adjacent tiles.
        - Add valid neighboring states to the open list if they have not been visited.
    3) If no solution is found, return an empty list.
2. print_state(): Print the state of the puzzle in a 3x3 grid format.
3. initial_state: Represents the initial state of the 8-puzzle problem.
4. Store the solution path obtained from the solve_puzzle() function.
5. Print the solution path if found, else print "No solution found.".

CODE

```
import heapq

def solve_puzzle(initial):
    goal = (1, 2, 3, 4, 5, 6, 7, 8, 0)
    moves = [1, -1, 3, -3]
    open_list, closed_set = [(0, initial, [])], set()

    while open_list:
        _, current, path = heapq.heappop(open_list)
        if current == goal:
            return path + [current]
        closed_set.add(current)

        empty = current.index(0)
```

```
        for m in moves:
            if 0 <= empty // 3 + m // 3 < 3 and m + empty in range(9):
                neighbor = list(current)
                neighbor[empty], neighbor[empty + m] = neighbor[empty + m], neighbor[empty]
                neighbor_tuple = tuple(neighbor)
                if neighbor_tuple not in closed_set:
                    heapq.heappush(open_list, (len(path) + 1, neighbor_tuple, path + [current]))

    return []

def print_state(state):
    for i in range(0, 9, 3):
        print(state[i:i+3])
    print('---')

initial_state = (1, 0, 3, 4, 2, 5, 7, 8, 6)
solution_path = solve_puzzle(initial_state)

if solution_path:
    print("Solution path:")
    for step in solution_path:
        print_state(step)
else:
    print("No solution found.")
```

OUTPUT

```
Solution path:
(1, 0, 3)
(4, 2, 5)
(7, 8, 6)
---
(1, 2, 3)
(4, 0, 5)
(7, 8, 6)
---
(1, 2, 3)
(4, 5, 0)
(7, 8, 6)
---
(1, 2, 3)
(4, 5, 6)
(7, 8, 0)
---
```