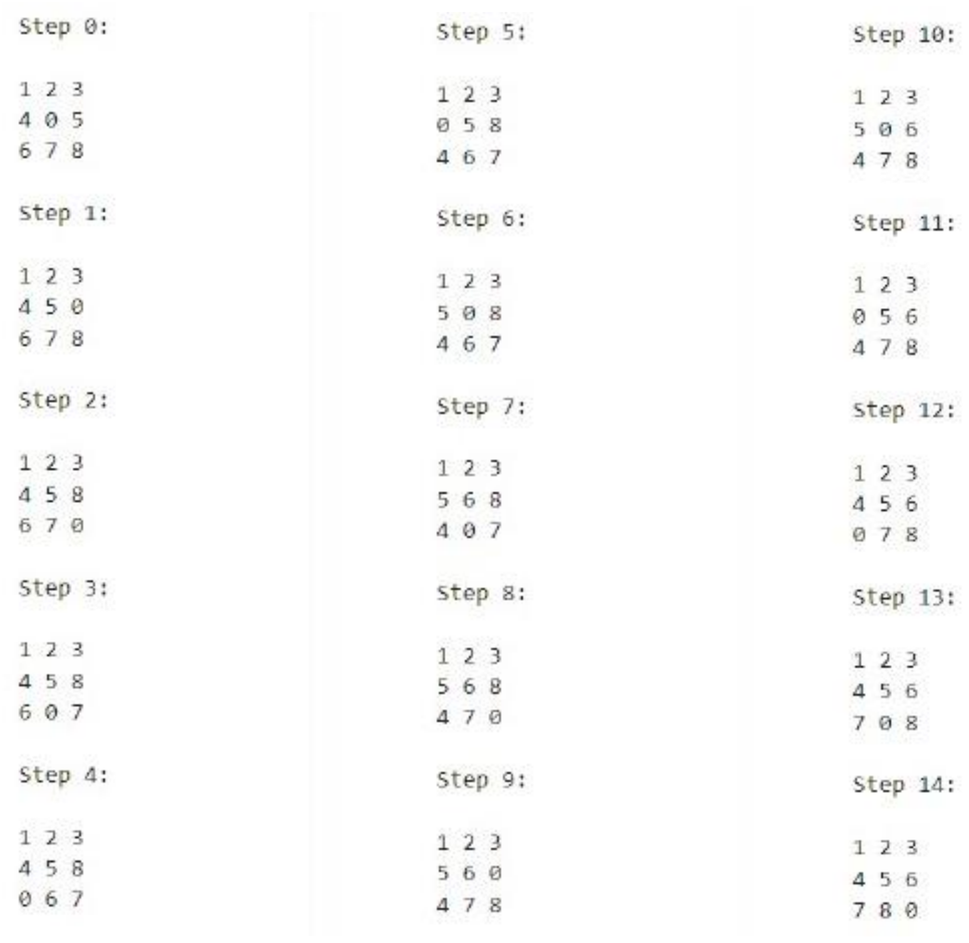
1. Solve 8 Puzzle problem using DFS and BFS.

BFS:

```
from collections import deque
class EightPuzzleBFS:
    def __init__(self, start_state, goal_state):
        self.start_state = start_state
        self.goal_state = goal_state
        self.visited = set()
    def get_neighbors(self, state):
        zero_idx = state.index(0)
        neighbors = []
  row, col = divmod(zero_idx, 3)
   directions = {
            'up': (-1, 0),
            'down': (1, 0),
            'left': (0, -1),
            'right': (0, 1)
        }
        for dr, dc in directions.values():
            new_row, new_col = row + dr, col + dc
            if 0 <= new_row < 3 and 0 <= new_col < 3:
                new_idx = new_row * 3 + new_col
                new_state = list(state)
 new_state[zero_idx], new_state[new_idx] = new_state[new_idx], new_state[zero_idx]
                neighbors.append(new_state)
    def solve(self):
        queue = deque([(self.start_state, [self.start_state])])
        self.visited.add(tuple(self.start_state))
        while queue:
            current_state, path = queue.popleft()
            if current_state == self.goal_state:
                return path  # Return the full path of states
            for neighbor in self.get_neighbors(current_state):
                if tuple(neighbor) not in self.visited:
                    self.visited.add(tuple(neighbor))
                    queue.append((neighbor, path + [neighbor]))
        return "No solution found"
def print_puzzle(state):
    """Print the puzzle state in a 3x3 grid format."""
    print("\n".join([" ".join(map(str, state[i:i + 3])) for i in range(0, 9, 3)]))
start_state = [1, 2, 3, 4, 0, 5, 6, 7, 8]  # Initial configuration
goal_state = [1, 2, 3, 4, 5, 6, 7, 8, 0]   # Goal configuration
solver = EightPuzzleBFS(start_state, goal_state)
solution = solver.solve()
if solution != "No solution found":
    for idx, state in enumerate(solution):
        print(f"Step {idx}:\n")
        print_puzzle(state)
        print()  # Add a blank line for better separation
else:
    print(solution)
```

Output:



```
Step 0:

1 2 3
4 0 5
6 7 8

Step 1:

1 2 3
4 5 0
6 7 8

Step 2:

1 2 3
4 5 8
6 7 0

Step 3:

1 2 3
4 5 8
6 0 7

Step 4:

1 2 3
4 5 8
0 6 7

Step 5:

1 2 3
0 5 8
4 6 7

Step 6:

1 2 3
5 0 8
4 6 7

Step 7:

1 2 3
5 6 8
4 0 7

Step 8:

1 2 3
5 6 8
4 7 0

Step 9:

1 2 3
5 6 0
4 7 8

Step 10:

1 2 3
5 0 6
4 7 8

Step 11:

1 2 3
0 5 6
4 7 8

Step 12:

1 2 3
4 5 6
0 7 8

Step 13:

1 2 3
4 5 6
7 0 8

Step 14:

1 2 3
4 5 6
7 8 0
```