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

def is_goal(state):
    return state == goal_state

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

def swap(state, i1, j1, i2, j2):
    new_state = [row[:] for row in state]
    new_state[i1][j1], new_state[i2][j2] = new_state[i2][j2], new_state[i1][j1]
    return new_state

def get_neighbors(state):
    neighbors = []
    i, j = find_blank(state)

    if i > 0:
        neighbors.append(swap(state, i, j, i - 1, j))
    if i < 2:
        neighbors.append(swap(state, i, j, i + 1, j))
    if j > 0:
        neighbors.append(swap(state, i, j, i, j - 1))
    if j < 2:
        neighbors.append(swap(state, i, j, i, j + 1))

    return neighbors

def dfs(state, visited, path):

    state_tuple = tuple(tuple(row) for row in state)

    if state_tuple in visited:
        return None
    visited.add(state_tuple)


    if is_goal(state):
        return path

    for neighbor in get_neighbors(state):
        result = dfs(neighbor, visited, path + [neighbor])
        if result is not None:
```

```
            return result

    return None

initial_state = [[1, 2, 3],
                 [4, 0, 6],
                 [7, 5, 8]]

visited = set()
solution = dfs(initial_state, visited, [])

if solution:
    print("Solution found in", len(solution), "steps:")
    for step in solution:
        for row in step:
            print(row)
        print()
else:
    print("No solution found.")
```

```
Solution found in 30 steps:
[1, 0, 3]
[4, 2, 6]
[7, 5, 8]

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

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

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

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

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

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

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

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

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

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

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

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

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

[5, 7, 3]
```