3. 8 puzzle bfs

```
from collections import deque


GOAL_STATE = (1, 2, 3, 4, 5, 6, 7, 8, 0)


def find_empty(state):
    return state.index(0)


def get_neighbors(state):
    neighbors = []
    empty_index = find_empty(state)
    row, col = divmod(empty_index, 3)
    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
    for dr, dc in directions:
        new_row, new_col = row + dr, col + dc
        if 0 <= new_row < 3 and 0 <= new_col < 3:
            new_index = new_row * 3 + new_col
            new_state = list(state)
            new_state[empty_index], new_state[new_index] = new_state[new_index],
new_state[empty_index]
            neighbors.append(tuple(new_state))
    return neighbors


def bfs(initial_state):
```

```
    queue = deque([(initial_state, [])])
    visited = set()
    visited.add(initial_state)
    visited_count = 1  # Initialize visited count
    while queue:
        current_state, path = queue.popleft()
        if current_state == GOAL_STATE:
            return path, visited_count  # Return path and count
        for neighbor in get_neighbors(current_state):
            if neighbor not in visited:
                visited.add(neighbor)
                queue.append((neighbor, path + [neighbor]))
                visited_count += 1  # Increment visited count
    return None, visited_count  # Return count if no solution found


def input_start_state():
    print("Enter the starting state as 9 numbers (0 for the empty space):")
    input_state = input("Format: 1 2 3 4 5 6 7 8 0\n")
    numbers = list(map(int, input_state.split()))
    if len(numbers) != 9 or set(numbers) != set(range(9)):
        print("Invalid input. Please enter numbers from 0 to 8 with no duplicates.")
        return input_start_state()
    return tuple(numbers)
```

```
def print_matrix(state):
    for i in range(0, 9, 3):
        print(state[i:i+3])


if _name_ == "_main_":
    initial_state = input_start_state()
    print("Initial state:")
    print_matrix(initial_state)
    solution, visited_count = bfs(initial_state)
    print(f"Number of states visited: {visited_count}")
    if solution:
        print("\nSolution found with the following steps:")
        for step in solution:
            print_matrix(step)
            print()
    else:
        print("No solution found.")
```

```
Enter the starting state as 9 numbers (0 for the empty space):
Format: 1 2 3 4 5 6 7 8 0
2 3 5 1 6 4 8 0 7
Initial state:
(2, 3, 5)
(1, 6, 4)
(8, 0, 7)
Number of states visited: 24445

Solution found with the following steps:
(2, 3, 5)
(1, 0, 4)
(8, 6, 7)

(2, 3, 5)
(1, 4, 0)
(8, 6, 7)

(2, 3, 5)
(1, 4, 7)
(8, 6, 0)

(2, 3, 5)
(1, 4, 7)
(8, 0, 6)

(2, 3, 5)
(1, 0, 7)
(8, 4, 6)

(2, 3, 5)
(1, 7, 0)
(8, 4, 6)

(2, 3, 0)
(1, 7, 5)
(8, 4, 6)
```

```
(2, 0, 3)
(1, 7, 5)
(8, 4, 6)

(0, 2, 3)
(1, 7, 5)
(8, 4, 6)

(1, 2, 3)
(0, 7, 5)
(8, 4, 6)

(1, 2, 3)
(7, 0, 5)
(8, 4, 6)

(1, 2, 3)
(7, 4, 5)
(8, 0, 6)

(1, 2, 3)
(7, 4, 5)
(0, 8, 6)

(1, 2, 3)
(0, 4, 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)
```