path = astar\_search(romania\_map, start\_node, goal\_node)

if path:

print("Path found:", path)

else:

print("No path found")

return None, float('inf')

def get\_neighbors(state):

# Define the successors for each state with their associated costs (simplified example).

successors = {

1: [(2, 3), (3, 5)],

2: [(1, 3), (4, 7)],

3: [(1, 5), (5, 2)],

4: [(2, 7), (6, 4)],

5: [(3, 2), (7, 6)],

6: [(4, 4), (8, 8)],

7: [(5, 6), (8, 5)],

8: [(6, 8), (7, 5)],

}

return successors.get(state, [])

if \_\_name\_\_ == '\_\_main\_\_':

start\_state = 1

goal\_state = 8

path, cost = rbfs(start\_state, goal\_state)

if path is not None:

print(f"Optimal path from {start\_state} to {goal\_state}:")

print(" -> ".join(map(str, path)))

print(f"Total cost: {cost}")

else:

print("No path found.")