Practical 1

Aim: To implement BFS Algorithm.

Code:

graph = {

    "oradea": [("zerind", 46), ("sibiu", 137)],

    "zerind": [("oradea", 46), ("arad", 43)],

    "sibiu": [("oradea", 137), ("arad", 121), ("rimnicu vikea", 54), ("fagaras", 98)],

    "arad": [("zerind", 43), ("sibiu", 121), ("timisoara", 82)],

    "rimnicu vikea": [("sibiu", 54), ("craiova", 124), ("pitesti", 97)],

    "fagaras": [("sibiu", 98), ("bucharest", 155)],

    "timisoara": [("arad", 82), ("lugoj", 77)],

    "craiova": [("pitesti", 104), ("rimnicu vikea", 124), ("dobreta", 89)],

    "pitesti": [("rimnicu vikea", 97), ("bucharest", 90), ("craiova", 104)],

    "bucharest": [("fagaras", 155), ("pitesti", 90), ("giurgiu", 62), ("urziceni", 61)],

    "lugoj": [("timisoara", 77), ("mehadia", 40)],

    "dobreta": [("craiova", 89), ("mehadia", 40)],

    "giurgiu": [("bucharest", 62)],

    "urziceni": [("bucharest", 61), ("vaslui", 108), ("hirsova", 78)],

    "mehadia": [("lugoj", 40), ("dobreta", 40)],

    "vaslui": [("urziceni", 108), ("lasi", 72)],

    "hirsova": [("urziceni", 78), ("eforic", 64)],

    "lasi": [("vaslui", 72), ("neamt", 74)],

    "eforic": [("hirsova", 64)],

    "neamt": [("lasi", 74)],

}

def bfs\_connected\_component(graph, start, goal):

    explored = []

    queue = [start]

    cost = [0]

    weight = 0

*if* start == goal:

*return* "Goal Found"

*while* queue:

        node = queue.pop(0)

        a = cost.pop(0)

*if* node not in explored:

            explored.append(node)

            weight += a

            neighbours = graph[node]

*for* neighbour *in* neighbours:

*if* neighbour[0] not in explored:

*if* neighbour[0] not in queue:

                        queue.append(neighbour[0])

                        cost.append(neighbour[1])

            print(queue)

            print(cost)

*if* node == goal:

*return* (explored, weight)

*return* "Goal not found"

print(bfs\_connected\_component(graph, "arad", "bucharest"))

Output:

