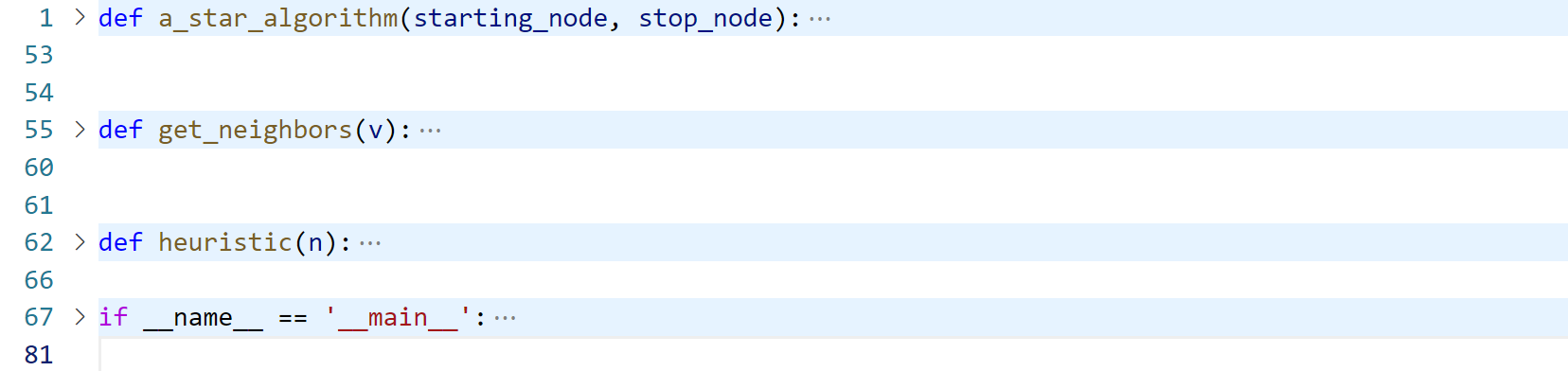
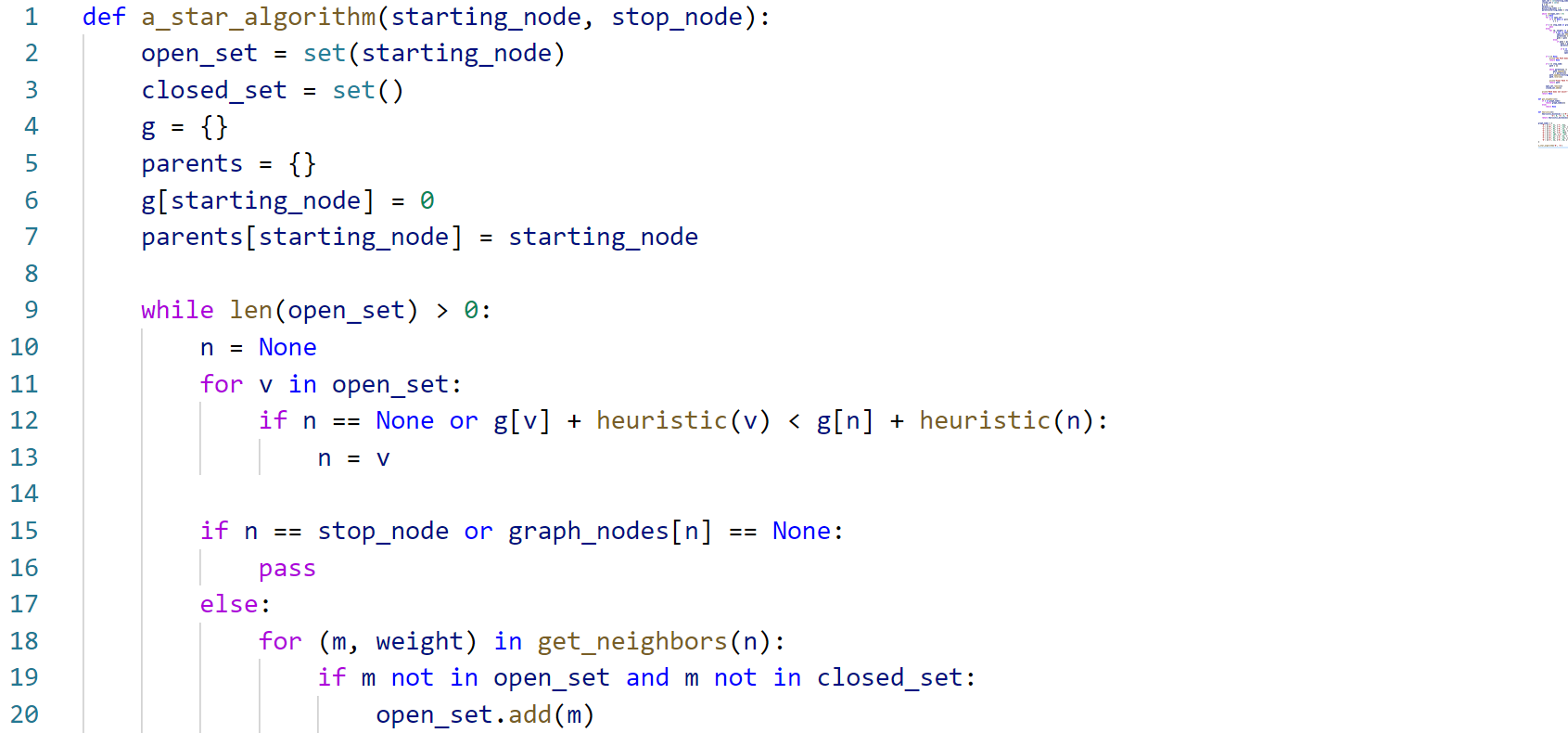
Advanced Algorithms Lab-Min term

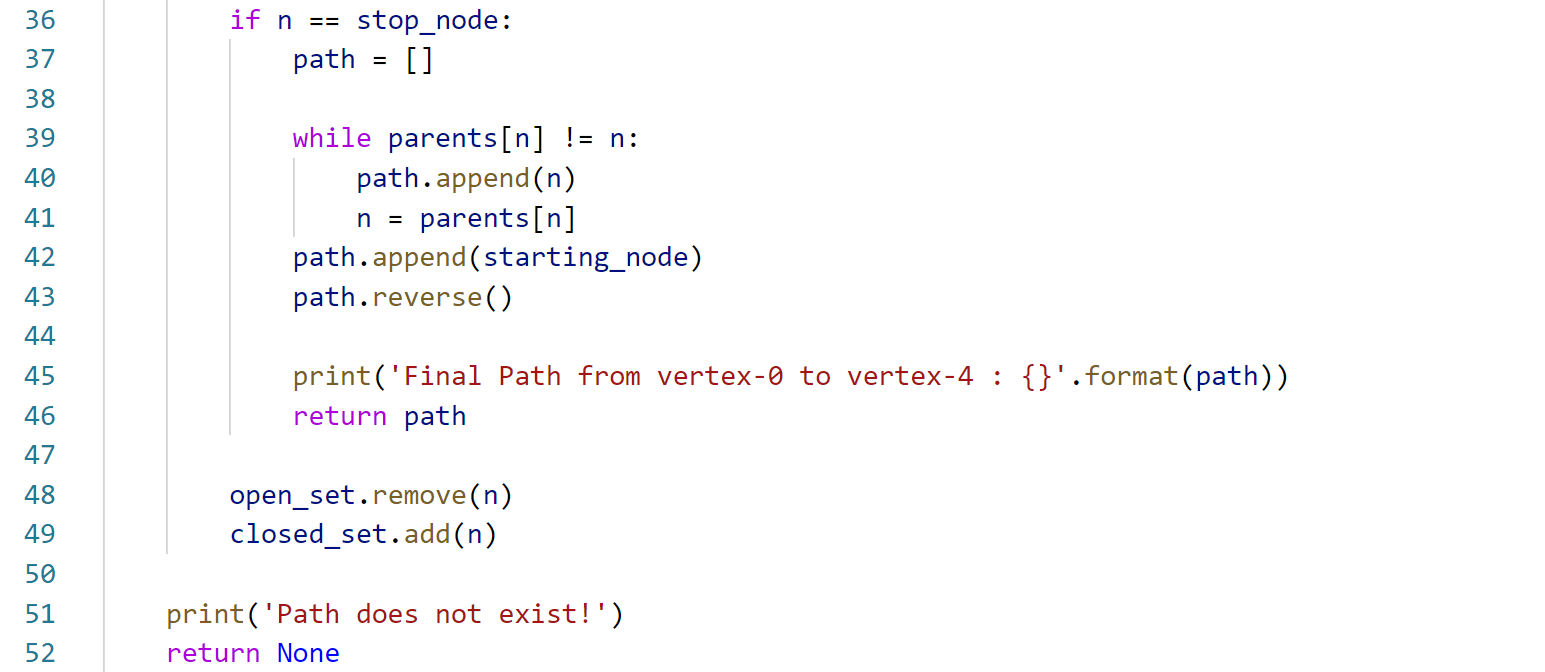
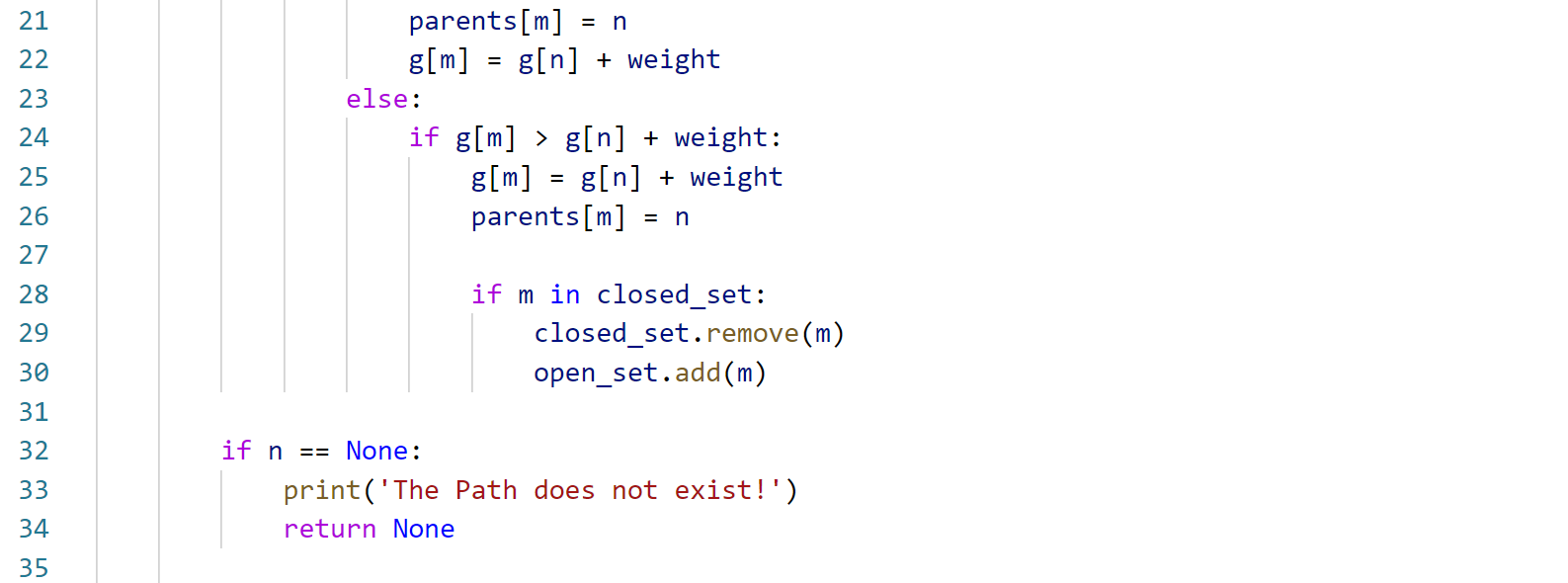
Prashanth.S   
 (19MID0020)

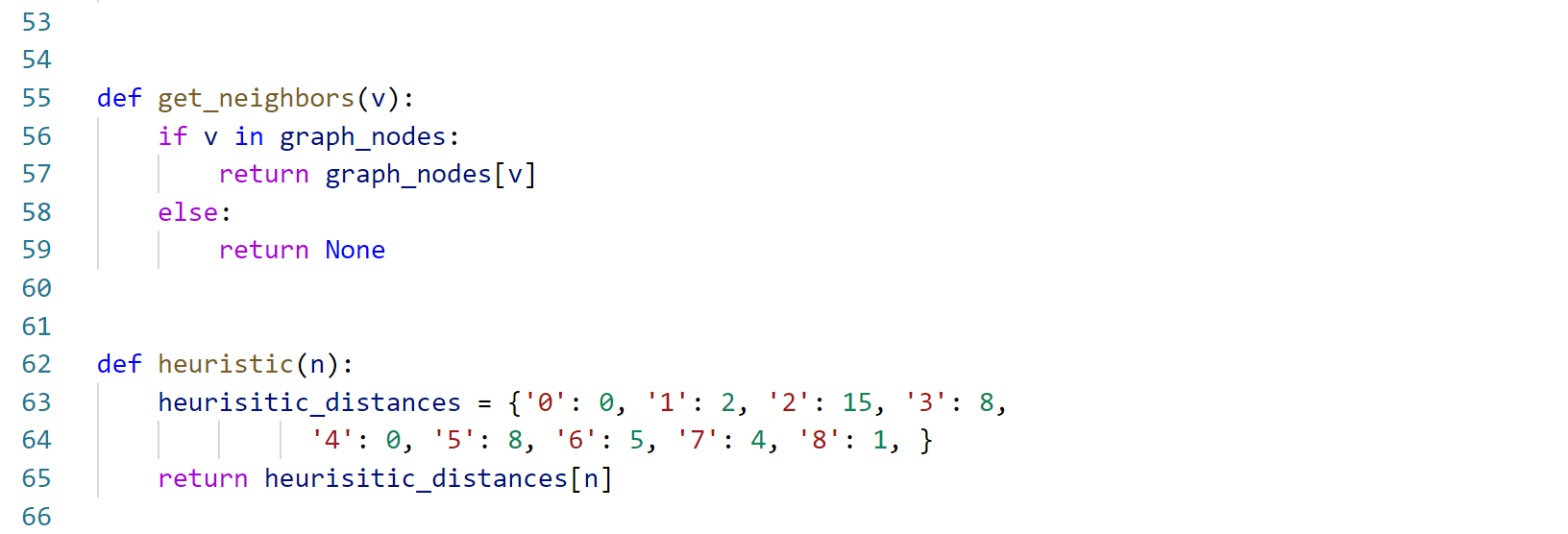
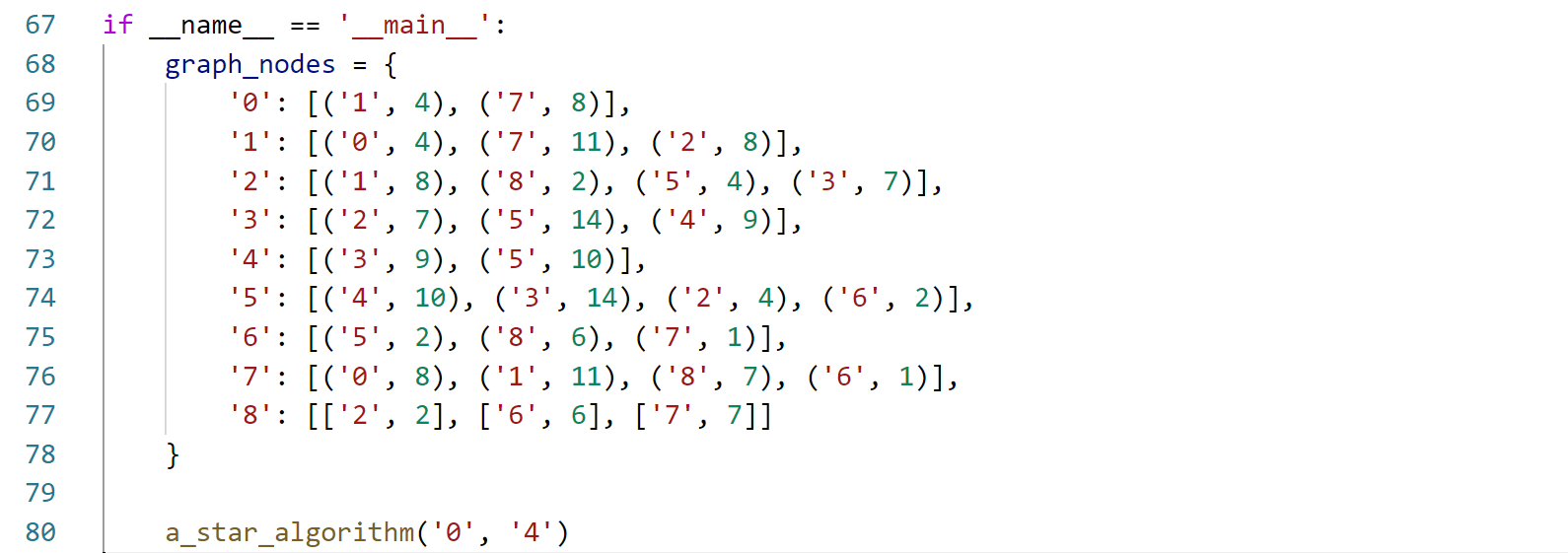
**Code Structure**



**Code Snippet**





**Output**



**Code**

def a\_star\_algorithm(starting\_node, stop\_node):

    open\_set = set(starting\_node)

    closed\_set = set()

    g = {}

    parents = {}

    g[starting\_node] = 0

    parents[starting\_node] = starting\_node

    while len(open\_set) > 0:

        n = None

        for v in open\_set:

            if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):

                n = v

        if n == stop\_node or graph\_nodes[n] == None:

            pass

        else:

            for (m, weight) in get\_neighbors(n):

                if m not in open\_set and m not in closed\_set:

                    open\_set.add(m)

                    parents[m] = n

                    g[m] = g[n] + weight

                else:

                    if g[m] > g[n] + weight:

                        g[m] = g[n] + weight

                        parents[m] = n

                        if m in closed\_set:

                            closed\_set.remove(m)

                            open\_set.add(m)

        if n == None:

            print('The Path does not exist!')

            return None

        if n == stop\_node:

            path = []

            while parents[n] != n:

                path.append(n)

                n = parents[n]

            path.append(starting\_node)

            path.reverse()

            print('Final Path from vertex-0 to vertex-4 : {}'.format(path))

            return path

        open\_set.remove(n)

        closed\_set.add(n)

    print('Path does not exist!')

    return None

def get\_neighbors(v):

    if v in graph\_nodes:

        return graph\_nodes[v]

    else:

        return None

def heuristic(n):

    heurisitic\_distances = {'0': 0, '1': 2, '2': 15, '3': 8,

              '4': 0, '5': 8, '6': 5, '7': 4, '8': 1, }

    return heurisitic\_distances[n]

if \_\_name\_\_ == '\_\_main\_\_':

    graph\_nodes = {

        '0': [('1', 4), ('7', 8)],

        '1': [('0', 4), ('7', 11), ('2', 8)],

        '2': [('1', 8), ('8', 2), ('5', 4), ('3', 7)],

        '3': [('2', 7), ('5', 14), ('4', 9)],

        '4': [('3', 9), ('5', 10)],

        '5': [('4', 10), ('3', 14), ('2', 4), ('6', 2)],

        '6': [('5', 2), ('8', 6), ('7', 1)],

        '7': [('0', 8), ('1', 11), ('8', 7), ('6', 1)],

        '8': [['2', 2], ['6', 6], ['7', 7]]

    }

    a\_star\_algorithm('0', '4')