**MACHINE INTELLIGENCE LABORATORY**

**WEEK – 2**

**NAME :** SANMAT SANJAYAKUMAR PAYAGOUDAR

**SRN** **:** PES1UG20CS385

**CODE :**

**PES1UG20CS385.py**

"""

You can create any other helper funtions.

Do not modify the given functions

"""

def A\_star\_Traversal(cost, heuristic, start\_point, goals):

    """

    Perform A\* Traversal and find the optimal path

    Args:

        cost: cost matrix (list of floats/int)

        heuristic: heuristics for A\* (list of floats/int)

        start\_point: Staring node (int)

        goals: Goal states (list of ints)

    Returns:

        path: path to goal state obtained from A\*(list of ints)

    """

    path = []

    explored = []

    path = [start\_point]

    frontier = [[0 + heuristic[start\_point], path]]

    while len(frontier) > 0:

        curr\_cost, curr\_path = frontier.pop(0)

        n = curr\_path[-1]

        curr\_cost -= heuristic[n]

        if n in goals:

            return curr\_path

        explored.append(n)

        children = [i for i in range(len(cost[0]))

            if cost[n][i] not in [0, -1]]

        for i in children:

            new\_curr\_path = curr\_path + [i]

            new\_path\_cost = curr\_cost + cost[n][i] + heuristic[i]

            if i not in explored and new\_curr\_path not in [i[1] for i in frontier]:

                frontier.append((new\_path\_cost, new\_curr\_path))

                frontier = sorted(frontier, key=lambda x: (x[0], x[1]))

            elif new\_curr\_path in [i[1] for i in frontier]:

                index = search\_q(frontier, new\_curr\_path)

                frontier[index][0] = min(frontier[index][0], new\_path\_cost)

                frontier = sorted(frontier, key=lambda x: (x[0], x[1]))

    return list()

visited = list(0 for i in range(0,11))

path = []

def DFS\_Traversal(cost, start\_point, goals):

    """

    Perform DFS Traversal and find the optimal path

        cost: cost matrix (list of floats/int)

        start\_point: Staring node (int)

        goals: Goal states (list of ints)

    Returns:

        path: path to goal state obtained from DFS(list of ints)

    """

    path = []

    pathList = []

    visitedArray = [0 for \_ in range(len(cost))]

    arrayAsStack = [(start\_point, [start\_point])]

    while (len(arrayAsStack) != 0):

        last, currPath = arrayAsStack[-1]

        if visitedArray[last] == 1:

            pass

        else:

            visitedArray[last] = 1

            if last in goals:

            # success, path found

                return currPath

            for element in range(len(cost) - 1, 0, -1):

                if cost[last][element] >= 1:

                    if visitedArray[element] == 1:

                        pass

                    else:

                        tempArray = [i for i in currPath]

                        tempArray.append(element)

                        arrayAsStack.append((element, tempArray))

    if len(path) == 0:

        return path

    else:

        return []

    return pat

**OUTPUT :**

