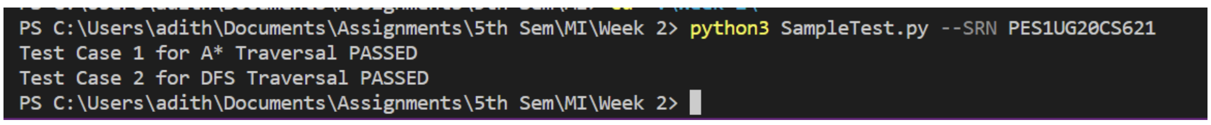
# Name: Adithya M    Section: K    SRN: PES1UG20CS621

Code:

```
"""
You can create any other helper funtions.
Do not modify the given functions
"""

from encodings import search_function


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 = []
    starting = [start_point]
    frntr = [[0 + heuristic[start_point], starting]]
    while len(frntr) > 0:
        cr_cc, cr_cp = frntr.pop(0)
        m = cr_cp[-1]
        cr_cc -= heuristic[m]
        if m in goals:
            return cr_cp
        path.append(m)

        brach = [i for i in range(len(cost[0])) if cost[m][i] not in [0, -1]]

        for i in brach:
            new_cr_cp = cr_cp + [i]
            new_pc = cr_cc + cost[m][i] + heuristic[i]

            if i not in path and new_cr_cp not in [i[1] for i in frntr]:
                frntr.append((new_pc, new_cr_cp))
                frntr = sorted(frntr, key=lambda x: (x[0], x[1]))

            elif new_cr_cp in [i[1] for i in frntr]:
                index = search_function(frntr, new_cr_cp)
                frntr[index][0] = min(frntr[index][0], new_pc)
                frntr = sorted(frntr, key=lambda x: (x[0], x[1]))

    print(path)
    return 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 = []
lt = [False for i in range(0, len(cost))]
stack = []
stack.append(start_point)
while len(stack):
    s = stack[-1]
    stack.pop()
    if not lt[s]:
        path.append(s)
        lt[s] = True
        if s in goals:
            break

    for i in range(0, len(cost[s])):
        if cost[s][len(cost[s]) - i - 1] > 0 and not lt[len(cost[s]) - i - 1]:
            stack.append(len(cost[s]) - i - 1)

return path
```

**Output:**



```
PS C:\Users\adith\Documents\Assignments\5th Sem\MI\Week 2> python3 SampleTest.py --SRN PES1UG20CS621
Test Case 1 for A* Traversal PASSED
Test Case 2 for DFS Traversal PASSED
PS C:\Users\adith\Documents\Assignments\5th Sem\MI\Week 2>
```