**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**

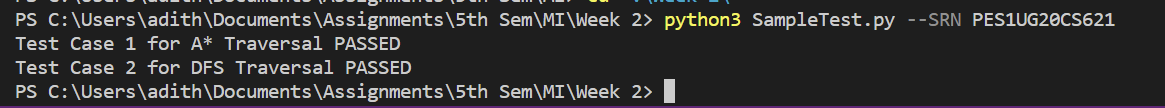
**for i in range(0, len(cost[s])):**

**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:**

****