**CSE422\_01\_Lab\_Assignment01\_Turnitin\_Summer2024**

**Name : Tahmid Iqbal  | Student ID : 21201701**

**CSE 422**

**Assignment 01**

**Name: Tahmid Iqbal  
ID: 21201701**

**Sec: 06**

**import heapq**

**def *inpFunc*(*file\_path\_01*):**

**with open(file\_path\_01, 'r') as file:**

**graph = {}**

**h = {}**

**for line in file:**

**parts = line.split()**

**city = parts[0]**

**heuristic = int(parts[1])**

**neighbors = parts[2:]**

**h[city] = heuristic**

**graph[city] = {}**

**for i in range(0, len(neighbors), 2):**

**neighbor = neighbors[i]**

**distance = int(neighbors[i+1])**

**graph[city][neighbor] = distance**

**return graph, h**

**# The a\_star\_search function finds the shortest path from a start node to a goal node using a priority queue**

**def *a\_star\_search*(*graph*, *h*, *start*, *goal*):**

**priority\_Q = [] # Priority queue to store nodes to be explored**

**heapq.heappush(priority\_Q, (0 + h[start], 0, start, [start]))**

**visited\_nodes = set() # tracking da visited nodes**

**while priority\_Q:**

**f\_score, g\_score, current\_node, path = heapq.heappop(priority\_Q) #lowest f\_score node**

**if current\_node in visited\_nodes:**

**continue**

**if current\_node == goal:**

**return path, g\_score**

**visited\_nodes.add(current\_node) # Marking visited**

**for neighbor, distance in graph[current\_node].items():**

**if neighbor in visited\_nodes:**

**continue**

**tentative\_g\_score = g\_score + distance**

**f\_score = tentative\_g\_score + h[neighbor]**

**heapq.heappush(priority\_Q, (f\_score, tentative\_g\_score, neighbor, path + [neighbor]))**

**return None, float('inf')**

**def *main*():**

**file\_path\_01 = 'input.txt'**

**file\_path\_02 = 'output.txt'**

**graph, h = inpFunc(file\_path\_01)**

**start = input("Start node: ")**

**goal = input("Destination node: ")**

**path, total\_dist = a\_star\_search(graph, h, start, goal)**

**with open(file\_path\_02, 'w') as output\_file:**

**if path:**

**output\_file.write(f"Path: {' -> '.join(path)}\n")**

**output\_file.write(f"Total distance: *{total\_dist}* km\n")**

**else:**

**output\_file.write("NO PATH FOUND\n")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**