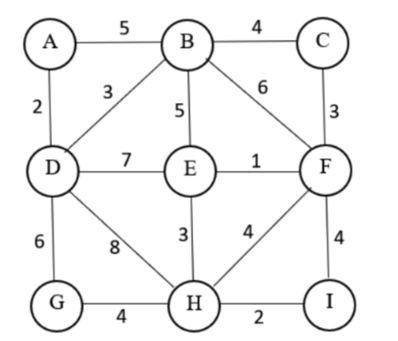
**22AIE203 – DATA STRUCTURES & ALGORITHMS 2**

**ASSIGNMENT**

Name : Anuvind MP

Roll no.: AM.EN.U4AIE22010

----------------------------------------------------------------------------------------------------------

****

**Graph:**

graph = {

     'A': [('B', 5), ('D', 2)],

     'B': [('A', 5), ('C', 4), ('D', 3), ('E', 5), ('F', 6)],

     'C': [('B', 4), ('F', 3)],

     'D': [('A', 2), ('B', 3), ('E', 7), ('G', 6)],

     'E': [('B', 5), ('D', 7), ('F', 1), ('H', 3)],

     'F': [('B', 6), ('C', 3), ('E', 1), ('H', 4), ('I', 4)],

     'G': [('D', 6), ('H', 4)],

     'H': [('D', 8), ('E', 3), ('F', 4), ('G', 4), ('I', 2)],

     'I': [('F', 4), ('H', 2)]

}

**Prims :**

**Code :**

def prim(graph):

    mst = set()

    visited = set()

    start = list(graph.keys())[0]

    priorityQueue = [(0, start)]

    totalCost = 0

    while priorityQueue:

        cost, current = priorityQueue.pop(0)

        if current not in visited:

            visited.add(current)

            mst.add((current, cost))

            totalCost += cost

            for neighbor, edgeCost in graph[current]:

                if neighbor not in visited:

                    priorityQueue.append((edgeCost, neighbor))

                    priorityQueue.sort()

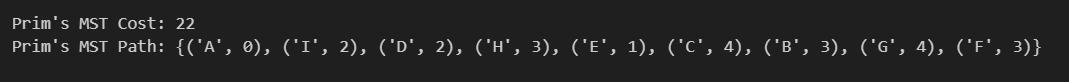
    return totalCost, mst

prim\_cost, prim\_path = prim(graph)

print("Prim's MST Cost:", prim\_cost)

print("Prim's MST Path:", prim\_path)

**Output :**

****

**Kruskal:**

**Code :**

def kruskal(graph):

    edges = []

    for node in graph:

        for neighbor, cost in graph[node]:

            edges.append((cost, node, neighbor))

    edges.sort()

    mst = set()

    total\_cost = 0

    parent = {}

    def find(node):

        if parent[node] != node:

            parent[node] = find(parent[node])

        return parent[node]

    def union(node1, node2):

        root1 = find(node1)

        root2 = find(node2)

        parent[root1] = root2

    for node in graph:

        parent[node] = node

    for edge in edges:

        cost, node1, node2 = edge

        if find(node1) != find(node2):

            mst.add((node1, node2, cost))

            total\_cost += cost

            union(node1, node2)

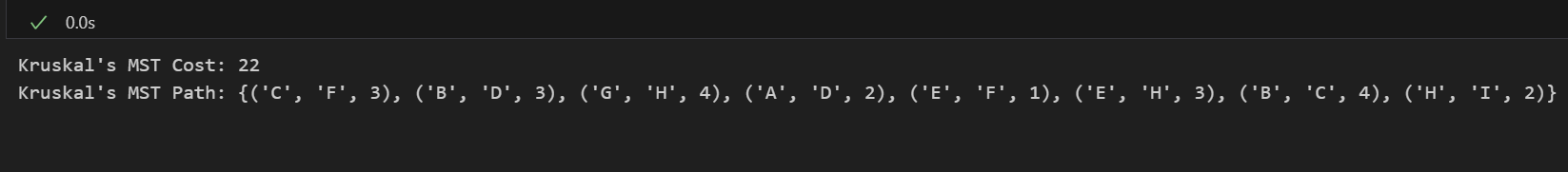
    return total\_cost, mst

kruskal\_cost, kruskal\_path = kruskal(graph)

print("Kruskal's MST Cost:", kruskal\_cost)

print("Kruskal's MST Path:", kruskal\_path)

**Output :**

****