//Assignment 5 (a)

```
# Kruskal's Algorithm for Minimum Spanning Tree

class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = []  # List to store edges (u, v, w)
    # Add edge to graph
    def add_edge(self, u, v, w):
        self.graph.append([u, v, w])
    # Find set of an element (with path compression)
    def find(self, parent, i):
        if parent[i] != i:
            parent[i] = self.find(parent, parent[i])
        return parent[i]
    # Union of two sets
    def union(self, parent, rank, x, y):
        xroot = self.find(parent, x)
        yroot = self.find(parent, y)
        if rank[xroot] < rank[yroot]:
            parent[xroot] = yroot
        elif rank[xroot] > rank[yroot]:
            parent[yroot] = xroot
        else:
            parent[yroot] = xroot
            rank[xroot] += 1
    # Kruskal’s Algorithm
    def kruskal_mst(self):
        result = []  # Store MST edges
        i = 0  # index for sorted edges
        e = 0  # index for result[]
        # Step 1: Sort edges by weight
        self.graph = sorted(self.graph, key=lambda item: item[2])
        parent = []
        rank = []
        # Create V subsets with single elements
        for node in range(self.V):
            parent.append(node)
            rank.append(0)

        # Number of edges to be taken is V-1
        while e < self.V - 1:
            # Step 2: Pick smallest edge
            u, v, w = self.graph[i]
            i = i + 1
            x = self.find(parent, u)
            y = self.find(parent, v)
            # If including this edge does not cause a cycle
            if x != y:
                e = e + 1
                result.append([u, v, w])
                self.union(parent, rank, x, y)
        # Print MST
        print("\nEdges in the Minimum Spanning Tree (Kruskal’s Algorithm):")
        min_cost = 0
        for u, v, weight in result:
            print(f"{u} -- {v} == {weight}")
            min_cost += weight
        print("Minimum Spanning Tree Cost:", min_cost)
# ----------------- Main Program -----------------
if __name__ == "__main__":
    # Example: 5 departments
    # 0 = Main Gate, 1 = CS Dept, 2 = IT Dept, 3 = Library, 4 = Hostel
    g = Graph(5)
    # Add edges with distances (weights)
    g.add_edge(0, 1, 10)  # Gate - CS
    g.add_edge(0, 2, 8)   # Gate - IT
    g.add_edge(1, 2, 5)   # CS - IT
    g.add_edge(1, 3, 3)   # CS - Library
    g.add_edge(2, 3, 7)   # IT - Library
    g.add_edge(2, 4, 6)   # IT - Hostel
    g.add_edge(3, 4, 4)   # Library - Hostel

    # Run Kruskal’s Algorithm
    g.kruskal_mst()
```

```
//OUTPUT
Edges in the Minimum Spanning Tree (Kruskal’s Algorithm):
1 -- 3 == 3
3 -- 4 == 4
1 -- 2 == 5
0 -- 2 == 8
Minimum Spanning Tree Cost: 20
```