**KRUSKAL’S ALGORITHM:**

class Graph:  
    def \_\_init\_\_(self, vertices):  
        self.V = vertices  
        self.graph = []  
  
    def add\_edge(self, u, v, w):  
        self.graph.append([u, v, w])  
  
    def find(self, parent, i):  
        if parent[i] == i:  
            return i  
        return self.find(parent, parent[i])  
  
    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  
  
    def kruskal\_mst(self):  
        result = []  
        i, e = 0, 0  
        self.graph = sorted(self.graph, key=lambda item: item[2])  
        parent = []  
        rank = []  
  
        for node in range(self.V):  
            parent.append(node)  
            rank.append(0)  
  
        while e < self.V - 1:  
            u, v, w = self.graph[i]  
            i += 1  
            x = self.find(parent, u)  
            y = self.find(parent, v)  
  
            if x != y:  
                e += 1  
                result.append([u, v, w])  
                self.union(parent, rank, x, y)  
  
        minimumCost = 0  
        print("Edges in the constructed MST")  
        for u, v, weight in result:  
            minimumCost += weight  
            print("%d -- %d == %d" % (u, v, weight))  
        print("Minimum Spanning Tree", minimumCost)  
  
  
def main():  
    num\_vertices = int(input("Enter the number of vertices: "))  
    num\_edges = int(input("Enter the number of edges: "))  
  
    g = Graph(num\_vertices)  
  
    for i in range(num\_edges):  
        u = int(input(f"Enter source vertex for edge {i+1}: "))  
        v = int(input(f"Enter destination vertex for edge {i+1}: "))  
        w = int(input(f"Enter weight for edge {i+1}: "))  
        g.add\_edge(u, v, w)  
  
    g.kruskal\_mst()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
    main()

OUTPUT:

Enter the number of vertices: 4  
Enter the number of edges: 5  
Enter source vertex for edge 1: 0  
Enter destination vertex for edge 1: 1  
Enter weight for edge 1: 10  
Enter source vertex for edge 2: 0  
Enter destination vertex for edge 2: 2  
Enter weight for edge 2: 6  
Enter source vertex for edge 3: 0  
Enter destination vertex for edge 3: 3  
Enter weight for edge 3: 5  
Enter source vertex for edge 4: 1  
Enter destination vertex for edge 4: 3  
Enter weight for edge 4: 15  
Enter source vertex for edge 5: 2  
Enter destination vertex for edge 5: 3  
Enter weight for edge 5: 4

Edges in the constructed MST  
0 -- 3 == 5  
0 -- 2 == 6  
2 -- 3 == 4  
Minimum Spanning Tree 15