**Exercise - 6**

**Kruskal’s Algorithm**

**Aim:** To write a python code to implement Kruskal’s algorithms

**Algorithm:**

1. Sort all the edges of the graph in non-decreasing order of their weights.
2. Create an empty set called MST to store the edges of the Minimum Spanning Tree.
3. Initialize a disjoint set data structure to keep track of disjoint sets of vertices.
4. For each vertex in the graph, create a disjoint set with that vertex as the only element.
5. Iterate through the sorted edges in ascending order of their weights:

* If adding the current edge to the MST does not create a cycle (i.e., the two vertices of the edge belong to different disjoint sets), add the edge to the MST set and merge the disjoint sets of the two vertices.

1. After iterating through all the edges, the MST set will contain the edges of the Minimum Spanning Tree.

**Program Code:**

class Graph:

    def \_\_init\_\_(self, vertices):

        self.V = vertices

        self.graph = []

    def addEdge(self, u, v, w):

        self.graph.append([u, v, w])

    def find(self, parent, i):

        if parent[i] != i:

            parent[i] = self.find(parent, parent[i])

        return parent[i]

    def union(self, parent, rank, x, y):

        if rank[x] < rank[y]:

            parent[x] = y

        elif rank[x] > rank[y]:

            parent[y] = x

        else:

            parent[y] = x

            rank[x] += 1

    def KruskalMST(self):

        result = []

        i = 0

        e = 0

        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 less than to V-1

        while e < self.V - 1:

            u, v, w = self.graph[i]

            i = i + 1

            x = self.find(parent, u)

            y = self.find(parent, v)

            if x != y:

                e = 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)

**Sample Input and Output:**

Input:    g = Graph(4)

    g.addEdge(0, 1, 10)

    g.addEdge(0, 2, 6)

    g.addEdge(0, 3, 5)

    g.addEdge(1, 3, 15)

    g.addEdge(2, 3, 4)

    g.KruskalMST()

Output:

A picture containing text, font, screenshot, receipt

Description automatically generated

**Result:**

Thus, Kruskal’s algorithm has been successfully implemented using Python code and the output is verified.