```
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_parent(self, parent, i):
        if parent[i] == i:
            return i
        return self.find_parent(parent, parent[i])

    def union(self, parent, rank, x, y):
        xroot = self.find_parent(parent, x)
        yroot = self.find_parent(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 KruskalMST(self):
        result = []
```

```
i = 0
e = 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(parent, u)
    y = self.find_parent(parent, v)

    if x != y:
        e += 1
        result.append([u, v, w])
        self.union(parent, rank, x, y)

for u, v, weight in result:
    print("%d -- %d == %d" % (u, v, weight))
```