PYTHON PROGRAM

```
1  #Program to implement Krukal Minimum Spanning Tree
2  from collections import defaultdict
3
4
5  class Graph:
6      def __init__(self, vertices):
7          self.v = vertices
8          self.graph = []
9
10     def addEdge(self, u, v, w):
11         self.graph.append([u, v, w])
12
13     def find(self, parent, i):
14         if parent[i] == i:
15             return i
16         return self.find(parent, parent[i])
17
18     def union(self, parent, rank, x, y):
19         xroot = self.find(parent, x)
20         yroot = self.find(parent, y)
21
22         if rank[xroot] < rank[yroot]:
23             parent[xroot] = yroot
24         elif rank[xroot] > rank[yroot]:
25             parent[yroot] = xroot
26
27         else:
28             parent[yroot] = xroot
29             rank[xroot] += 1
30
31     def KruskalMST(self):
32         result = []
33         i = 0
34         e = 0
35
36         self.graph = sorted(self.graph, key=lambda item: item[2])
37
38         parent = []
39         rank = []
```

PYTHON PROGRAM

```
40
41          for node in range(self.v):
42              parent.append(node)
43              rank.append(0)
44
45          while e < self.v - 1:
46              u, v, w = self.graph[i]
47
48              i += 1
49              x = self.find(parent, u)
50              y = self.find(parent, v)
51
52              if x != y:
53                  e += 1
54                  result.append([u, v, w])
55                  self.union(parent, rank, x, y)
56          # printing
57          for u, v, weight in result:
58              print("%d -- %d == %d" % (u, v, weight))
59  if __name__ == "__main__":
60      g = Graph(4)
61      g.addEdge(0, 1, 10)
62      g.addEdge(0, 2, 6)
63      g.addEdge(0, 3, 5)
64      g.addEdge(1, 3, 15)
65      g.addEdge(2, 3, 4)
66      g.KruskalMST()
```