Name: - Rajlaxmi Manepatil

Roll No: -3263

Subject: - AI [LP II]

**Assignment No: 3**

**Problem Statement:** Implement Greedy search algorithm : Kruskal's Minimal Spanning Tree Algorithm

**Code**

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 = 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, u)

y = self.find(parent, v)

if x != y:

e += 1

result.append([u, v, w])

self.union(parent, rank, x, y)

print("Edges in the Minimum Spanning Tree:")

for u, v, weight in result:

print("%d - %d: %d" % (u, v, weight))

g = Graph(int(input("Enter the number of vertices in the graph: ")))

while True:

edge = input("Enter an edge in the format 'u v weight' (or 'done' to finish): ")

if edge == 'done':

break

else:

u, v, w = map(int, edge.split())

g.add\_edge(u, v, w)

g.kruskal\_mst()

**OUTPUT**

Enter the number of vertices in the graph: 4

Enter an edge in the format 'u v weight' (or 'done' to finish): 0 1 10

Enter an edge in the format 'u v weight' (or 'done' to finish): 0 2 6

Enter an edge in the format 'u v weight' (or 'done' to finish): 0 3 5

Enter an edge in the format 'u v weight' (or 'done' to finish): 1 3 15

Enter an edge in the format 'u v weight' (or 'done' to finish): done

Edges in the Minimum Spanning Tree:

0 - 3: 5

0 - 2: 6

0 - 1: 10