5.11 KRUSKAL'S ALGORITHM FOR MINIMUM SPANNING TREE (MST)

Question:

Given a graph represented by an edge list, implement Kruskal's Algorithm to find the Minimum Spanning Tree (MST) and its total weight.

AIM

To construct the MST of a weighted undirected graph using Kruskal's algorithm and return the total weight of the MST.

ALGORITHM

1. Sort all edges in ascending order of weight.
2. Initialize a Union-Find (Disjoint Set Union) structure to track connected components.
3. Iterate through sorted edges:
    - If the edge connects two different components, add it to the MST.
    - Union the two components.
4. Stop when the MST contains n - 1 edges (for n vertices).
5. Return the total weight of the MST.

PROGRAM

```
def find(parent, i):
    if parent[i] != i:
        parent[i] = find(parent, parent[i])
    return parent[i]

def union(parent, rank, x, y):
    xroot = find(parent, x)
    yroot = find(parent, y)
    if rank[xroot] < rank[yroot]:
        parent[xroot] = yroot
    else:
        parent[yroot] = xroot
        if rank[xroot] == rank[yroot]:
            rank[xroot] += 1

def kruskal(n, edges):
    edges.sort(key=lambda x: x[2])
    parent = list(range(n))
    rank = [0] * n
    mst = []
    for u, v, w in edges:
        if find(parent, u) != find(parent, v):
            union(parent, rank, u, v)
            mst.append((u, v, w))
    total = sum(w for _, _, w in mst)
    return mst, total

n = int(input("Enter number of vertices: "))
m = int(input("Enter number of edges: "))
edges = []
for _ in range(m):
    u, v, w = map(int, input("Edge: ").split())
    edges.append((u, v, w))
mst, total = kruskal(n, edges)
print("Edges in MST:", mst)
print("Total weight of MST:", total)
```

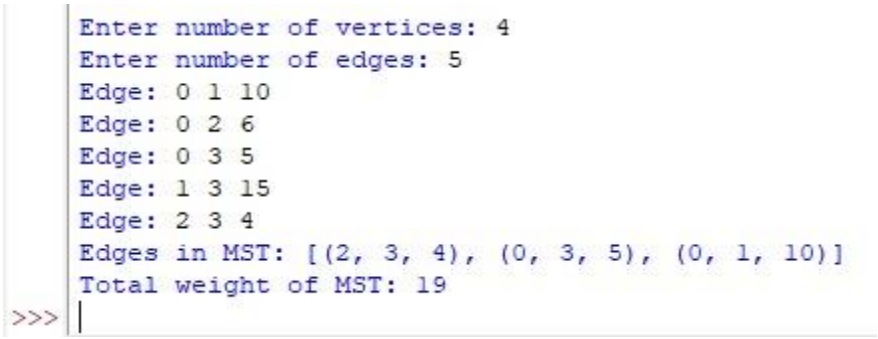
Input:

Enter number of vertices: 4

Enter number of edges: 5

Edges: 0 1 10
Edges: 0 2 6
Edges: 1 3 15
Edges: 2 3 4

Output:



RESULT:

Thus program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

· Time Complexity: O(E log E), where *E* is the number of edges (due to sorting)

· Space Complexity: O(N), for Union-Find structure