

12. Given a graph with weights and a potential Minimum Spanning Tree (MST), verify if the given MST is unique. If it is not unique, provide another possible MST.

Test Case 1:

Input:

n = 4

m = 5

edges = [(0, 1, 10), (0, 2, 6), (0, 3, 5), (1, 3, 15), (2, 3, 4)]

given_mst = [(2, 3, 4), (0, 3, 5), (0, 1, 10)]

Output: Is the given MST unique? True

Program:

```
class DisjointSet:
```

```
    def __init__(self, n):
```

```
        self.parent = list(range(n))
```

```
        self.rank = [0] * n
```

```
    def find(self, u):
```

```
        if self.parent[u] != u:
```

```
            self.parent[u] = self.find(self.parent[u])
```

```
        return self.parent[u]
```

```
    def union(self, u, v):
```

```
        root_u = self.find(u)
```

```
        root_v = self.find(v)
```

```
        if root_u != root_v:
```

```
            if self.rank[root_u] > self.rank[root_v]:
```

```
                self.parent[root_v] = root_u
```

```
            elif self.rank[root_u] < self.rank[root_v]:
```

```
                self.parent[root_u] = root_v
```

```
            else:
```

```
                self.parent[root_v] = root_u
```

```

        self.rank[root_u] += 1

def kruskal(n, edges):

    edges.sort(key=lambda x: x[2])

    disjoint_set = DisjointSet(n)

    mst = []

    total_weight = 0

    for u, v, weight in edges:

        if disjoint_set.find(u) != disjoint_set.find(v):

            disjoint_set.union(u, v)

            mst.append((u, v, weight))

            total_weight += weight

    return mst, total_weight

def is_mst_unique(n, edges, given_mst):

    given_mst_weight = sum(weight for _, _, weight in given_mst)

    mst, mst_weight = kruskal(n, edges)

    if mst_weight != given_mst_weight:

        return False, mst

    mst_set = set(mst)

    given_mst_set = set(given_mst)

    if mst_set == given_mst_set:

        return True, mst

    return False, mst

n = 4

m = 5

edges = [(0, 1, 10), (0, 2, 6), (0, 3, 5), (1, 3, 15), (2, 3, 4)]

given_mst = [(2, 3, 4), (0, 3, 5), (0, 1, 10)]

is_unique, alternative_mst = is_mst_unique(n, edges, given_mst)

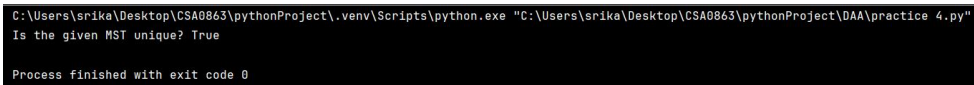
```

```
print("Is the given MST unique?", is_unique)
```

```
if not is_unique:
```

```
    print("Alternative MST:", alternative_mst)
```

Output:

A terminal window with a black background and white text. The first line shows the command: C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA\practice 4.py". The second line shows the output: Is the given MST unique? True. The third line shows the message: Process finished with exit code 0.

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA\practice 4.py"
Is the given MST unique? True

Process finished with exit code 0
```

Time complexity:

$O(m \log m)$