95. Minimum Spanning Tree

Code:

```
import heapq

def prim(n, edges):
    adj = [[] for _ in range(n)]
    for u, v, weight in edges:
        adj[u].append((weight, v))
        adj[v].append((weight, u))

    mst = []
    visited = [False] * n
    min_heap = [(0, 0)]

    while min_heap:
        weight, u = heapq.heappop(min_heap)
        if visited[u]:
            continue
        visited[u] = True
        mst.append((u, weight))

        for next_weight, v in adj[u]:
            if not visited[v]:
                heapq.heappush(min_heap, (next_weight, v))

    return mst
```

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

mst = prim(n, edges)
print("Edges in the MST:")
for u, weight in mst:
    if weight != 0:
        print(f"({u}, weight: {weight})")
```

Output:

```
Edges in the MST:
(3, weight: 5)
(2, weight: 4)
(1, weight: 10)
```

Time Complexity:

- T(n)= O(E logv)