7. Minimum Spanning Tree

Code:

```
import heapq
def prim_mst(graph):
    start_node = 0
    pq = [(0, start_node)]
    visited = set()
    mst_cost = 0
    mst_edges = []
    while pq:
        cost, u = heapq.heappop(pq)
        if u in visited:
            continue
        visited.add(u)
        mst_cost += cost
        if cost != 0:
            mst_edges.append((u, cost))
        for v, weight in graph[u]:
            if v not in visited:
                heapq.heappush(pq, (weight, v))
    return mst_cost, mst_edges
graph = {
    0: [(1, 10), (2, 1), (3, 4)],
    1: [(0, 10), (2, 3), (4, 0)],
    2: [(0, 1), (1, 3), (3, 2), (4, 8)],
    3: [(0, 4), (2, 2), (4, 2), (5, 7)],
    4: [(1, 0), (2, 8), (3, 2), (5, 1)],
    5: [(3, 7), (4, 1)]
}
mst_cost, mst_edges = prim_mst(graph)
print("MST cost:", mst_cost)
```

print("MST edges:", mst_edges)

output:

```
PS C:\Users\karth>
PS C:\Users\karth> & C:/Users/karth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/karth/OneDrive/Documents/OriginLab/daa.py
MST cost: 6
MST edges: [(2, 1), (3, 2), (4, 2), (5, 1)]
PS C:\Users\karth>
```

Time complexity:

F(n)=o(nlogn)