```
import heapq
class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = []  # For Kruskal's algorithm (edge list)
        self.adj_list = {i: [] for i in range(vertices)}  # For Prim's algorithm

    def add_edge(self, u, v, w):
        self.graph.append((w, u, v))  # Edge list for Kruskal
        self.adj_list[u].append((v, w))  # Adjacency list for Prim
        self.adj_list[v].append((u, w))  # Undirected graph

    # Kruskal's Algorithm
    def kruskal_mst(self):
        # Helper functions for Union-Find
        def find(parent, i):
            if parent[i] == i:
                return i
            return find(parent, parent[i])

        def union(parent, rank, x, y):
            xroot = find(parent, x)
            yroot = 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
```

```
    # Kruskal's algorithm implementation
    self.graph.sort()  # Sort edges by weight
    parent = []
    rank = []
    mst = []

    for node in range(self.V):
        parent.append(node)
        rank.append(0)

    for edge in self.graph:
        w, u, v = edge
        x = find(parent, u)
        y = find(parent, v)
        if x != y:
            mst.append(edge)
            union(parent, rank, x, y)

    print("Kruskal's MST:")
    for w, u, v in mst:
        print(f"{u} -- {v} == {w}")

# Prim's Algorithm
def prim_mst(self):
    visited = [False] * self.V
    min_heap = [(0, 0)]  # (weight, vertex)
    mst = []

    while min_heap:
        weight, u = heapq.heappop(min_heap)
        if visited[u]:
```

```
            continue
        visited[u] = True
        mst.append((u, weight))

        for v, w in self.adj_list[u]:
            if not visited[v]:
                heapq.heappush(min_heap, (w, v))

    print("Prim's MST:")
    for u, weight in mst[1:]:
        print(f"{u} -- {weight}")


# Example usage
# Nodes: 0 - Admin, 1 - Library, 2 - CSE Dept, 3 - ECE Dept, 4 - Hostel
g = Graph(5)
g.add_edge(0, 1, 10)
g.add_edge(0, 2, 20)
g.add_edge(1, 2, 5)
g.add_edge(1, 3, 15)
g.add_edge(2, 3, 30)
g.add_edge(3, 4, 10)
g.add_edge(2, 4, 25)

# Find MST using Kruskal's algorithm
g.kruskal_mst()

# Find MST using Prim's algorithm
g.prim_mst()
```