Step-by-Step Explanation (Prim's Algorithm using Priority Queue)

1. Define a graph with an adjacency list and Edge class to hold destination and weight.
2. Initialize arrays: key[] (to store min edge weight), parent[] (to store MST tree), and inMST[] (to track included nodes).
3. Set all key[] values to ∞, except key[0] = 0 to start from vertex 0.
4. Use a min-heap (PriorityQueue) to always pick the edge with the smallest weight.
5. Add the starting vertex (0) to the priority queue.
6. While the queue is not empty, extract the vertex u with the minimum key value.
7. Mark u as included in the MST.
8. For each adjacent edge (u-v) of u, update key[v] and parent[v] if a better edge is found.
9. Add updated vertex v to the priority queue.
10. After building the MST, print edges from parent[] with corresponding weights.

Time and Space Complexity

- **Time Complexity:** O(E log V)
    - Using a priority queue with V vertices and E edges (log V for heap operations).
- **Space Complexity:** O(V + E)
    - For the adjacency list, key array, parent array, and MST tracking.