```
import java.util.*;

public class powergrid {
    public static void main(String[] args) {
        List<Edge> edges = new ArrayList<>();
        edges.add(new Edge('A', 'B', 1));
        edges.add(new Edge('B', 'C', 4));
        edges.add(new Edge('B', 'D', 6));
        edges.add(new Edge('D', 'E', 5));
        edges.add(new Edge('C', 'E', 1));

        System.out.println(solve(edges, 5));
    }

    private static List<Edge> solve(List<Edge> edges, int vertices) {
        Queue<Edge> pq = new PriorityQueue<>((a, b) -> {return a.cost -
b.cost;});
        Map<Character, Set<Edge>> map = new HashMap<>();

        int minCost = Integer.MAX_VALUE;
        Edge minEdge = null;
        for  (Edge e : edges) {
            if (!map.containsKey(e.from))
                map.put(e.from, new HashSet<>());
            if (!map.containsKey(e.to))
                map.put(e.to, new HashSet<>());
            map.get(e.from).add(e);
            map.get(e.to).add(e);
            if (e.cost < minCost) {
                minCost = e.cost;
                minEdge = e;
            }
        }

        Set<Character> set = new HashSet<>();
        List<Edge> res = new ArrayList<>();
        pq.add(minEdge);

        while (set.size() != vertices) {
            Edge e = pq.poll();
            if (set.contains(e.to) && set.contains(e.from)) continue;

            set.add(e.to);
            set.add(e.from);
            res.add(e);

            Set<Edge> neighbors = new HashSet<>();
            if (map.containsKey(e.to)) {
                neighbors.addAll(map.get(e.to));
                map.remove(e.to);
            }

            if (map.containsKey(e.from)) {
                neighbors.addAll(map.get(e.from));
                map.remove(e.from);
            }
```

```
            for (Edge n : neighbors)
                pq.add(n);
        }

        return res;
    }

    private static class Edge {
        public Character from, to;
        public int cost;

        public Edge(Character from, Character to, int cost) {
            this.from = from;
            this.to = to;
            this.cost = cost;
        }

        @Override
        public String toString() {
            return "[" + from + ", " + to + ", " + cost + "]";
        }
    }
}
```