Leetcode

Leetcode 1 Solution :-

```
class Solution {
    public String frequencySort(String s) {


        Map<Character, Integer> map = new HashMap<>();
        PriorityQueue<Map.Entry<Character, Integer>> pq = new
PriorityQueue<>((a, b) -> b.getValue() - a.getValue());

        // Mapping the char with the frequency
        for (char c : s.toCharArray()) {
            map.put(c, map.getOrDefault(c, 0) + 1);
        }

        // Pushing the char with frequencies in the priority queue
        pq.addAll(map.entrySet());

        StringBuilder ans = new StringBuilder();

        // Generating the anser string
        while (!pq.isEmpty()) {
            Map.Entry<Character, Integer> entry = pq.poll();

            int freq = entry.getValue();
            char ch = entry.getKey();

            while (freq-- > 0) {
                ans.append(ch);
            }
        }

        return ans.toString();

    }
}
```

Leetcode 2 Solution :-

```
public class Codec {
    public String serialize(TreeNode root) {
        if (root == null) return "";
        Queue<TreeNode> q = new LinkedList<>();
        StringBuilder res = new StringBuilder();
        q.add(root);
```

```
        while (!q.isEmpty()) {
            TreeNode node = q.poll();
            if (node == null) {
                res.append("n ");
                continue;
            }
            res.append(node.val + " ");
            q.add(node.left);
            q.add(node.right);
        }
        return res.toString();
    }

    public TreeNode deserialize(String data) {
        if (data == "") return null;
        Queue<TreeNode> q = new LinkedList<>();
        String[] values = data.split(" ");
        TreeNode root = new TreeNode(Integer.parseInt(values[0]));
        q.add(root);
        for (int i = 1; i < values.length; i++) {
            TreeNode parent = q.poll();
            if (!values[i].equals("n")) {
                TreeNode left = new TreeNode(Integer.parseInt(values[i]));
                parent.left = left;
                q.add(left);
            }
            if (!values[++i].equals("n")) {
                TreeNode right = new TreeNode(Integer.parseInt(values[i]));
                parent.right = right;
                q.add(right);
            }
        }
        return root;
    }
}
```