

# Assignment-6

## Question 1.

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        if(root == null || root == p || root == q) return root;
        TreeNode left = lowestCommonAncestor(root.left,p,q);
        TreeNode right = lowestCommonAncestor(root.right,p,q);
        if(left == null) return right;
        else if(right == null) return left;
        else return root;
    }
}
```

The screenshot displays the LeetCode interface for the problem "lowest-common-ancestor-of-a-binary-tree". The left sidebar shows the problem status as "Accepted" with 32/32 test cases passed. The runtime is 7 ms, beating 63.51% of submissions. The memory usage is 44.89 MB, beating 47.95% of submissions. The main area shows the Java code for the solution, which implements the lowestCommonAncestor method. The test result section shows "Accepted" with a runtime of 0 ms for three cases. The input for the first case is root = [3,5,1,6,2,0,8,null,null,7,4].

Runtime: 7 ms | Beats 63.51%

Memory: 44.89 MB | Beats 47.95%

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input: root = [3,5,1,6,2,0,8,null,null,7,4]

## Question 2

```
class Solution {  
  
    public int[] topKFrequent(int[] nums, int k) {  
        HashMap<Integer,Integer> map = new HashMap<>();  
        for(int num : nums){  
            map.put(num,map.getOrDefault(num,0)+1);  
        }  
        PriorityQueue<Integer> pq = new PriorityQueue<>((a,b) -> map.get(b) - map.get(a));  
        pq.addAll(map.keySet());  
        int [] result = new int[k];  
        for(int i =0;i<k;i++){  
            result[i] = pq.poll();  
        }  
        return result;  
    }  
}
```

Screenshot of a LeetCode submission for the problem "Top K Frequent Elements".

The submission is titled "Heap (Priority Queue)" and is marked as "Accepted". It shows the runtime performance: 13 ms, Beats 76.53%.

The memory usage is 48.80 MB, Beats 53.26%.

The code is written in Java and uses a Priority Queue to solve the problem.

The test result shows "Accepted" with a runtime of 1 ms. The input is "nums = [1,1,1,2,2,3]".

The code is as follows:

```
1 class Solution {  
2     public int[] topKFrequent(int[] nums, int k) {  
3         HashMap<Integer,Integer> map = new HashMap<>();  
4         for(int num : nums){  
5             map.put(num,map.getOrDefault(num,0)+1);  
6         }  
7         PriorityQueue<Integer> pq = new PriorityQueue<>((a,b) -> map.get(b) - map.get(a));  
8         pq.addAll(map.keySet());  
9         int [] result = new int[k];  
10        for(int i =0;i<k;i++){  
11            result[i] = pq.poll();  
12        }  
13        return result;  
14    }  
15 }
```