Solution@27-11-23

515. Find Largest Value in Each Tree Row

```
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;

class TreeNode {
  int val;
  TreeNode left, right;

  public TreeNode(int val) {
    this.val = val;
    this.left = this.right = null;
  }
}

public class LargestValuesInTreeRows {
  public List<Integer> largestValues(TreeNode root) {
    List<Integer> result = new ArrayList<>();
    if (root == null) {
      return result;
    }

    Queue<TreeNode> queue = new LinkedList<>();
    queue.offer(root);

    while (!queue.isEmpty()) {
      int size = queue.size();
      int max = Integer.MIN_VALUE;
```

```
    for (int i = 0; i < size; i++) {
      TreeNode node = queue.poll();
      max = Math.max(max, node.val);

      if (node.left != null) {
        queue.offer(node.left);
      }

      if (node.right != null) {
        queue.offer(node.right);
      }
    }

    result.add(max);
  }

  return result;
}

public static void main(String[] args) {
  // Example usage:
  // Create a binary tree
  TreeNode root = new TreeNode(1);
  root.left = new TreeNode(3);
  root.right = new TreeNode(2);
  root.left.left = new TreeNode(5);
  root.left.right = new TreeNode(3);
  root.right.right = new TreeNode(9);

  LargestValuesInTreeRows solution = new LargestValuesInTreeRows();
  List<Integer> result = solution.largestValues(root);
```

```
    System.out.println("Largest values in each row: " + result);
  }
}
```

Output

```
PS C:\Users\Ajeet\Desktop\java> javac LargestValuesInTreeRows.java
PS C:\Users\Ajeet\Desktop\java> java LargestValuesInTreeRows
Largest values in each row: [1, 3, 9]
```

451. Sort Characters By Frequency

```
import java.util.*;

public class SortCharactersByFrequency {

  public static String frequencySort(String s) {
    // Step 1: Count the frequency of each character
    Map<Character, Integer> frequencyMap = new HashMap<>();
    for (char c : s.toCharArray()) {
      frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);
    }

    // Step 2: Create a priority queue to sort characters by frequency
    PriorityQueue<Character> maxHeap = new PriorityQueue<>(
        (a, b) -> frequencyMap.get(b) - frequencyMap.get(a));

    maxHeap.addAll(frequencyMap.keySet());

    // Step 3: Build the sorted string
    StringBuilder sortedString = new StringBuilder();
```

```
    while (!maxHeap.isEmpty()) {
      char currentChar = maxHeap.poll();
      int frequency = frequencyMap.get(currentChar);
      for (int i = 0; i < frequency; i++) {
        sortedString.append(currentChar);
      }
    }

    return sortedString.toString();
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter a string:");
    String inputString = scanner.nextLine();

    String result = frequencySort(inputString);

    System.out.println("Sorted String by Frequency: " + result);
  }
}
```

Output

```
PS C:\Users\Ajeet\Desktop\java> javac SortCharactersByFrequency.java
PS C:\Users\Ajeet\Desktop\java> java SortCharactersByFrequency
Enter a string:
tree
Sorted String by Frequency: eert
```