https://course.acciojob.com/idle?question=2e9cb785-6601-4de7-90c3-0e79638c9f5b

- EASY
- Max Score: 30 Points

Top view of Binary tree

You are given a pointer to the head of the binary tree. You have to print the top view of the tree from left to right.

NOTE: Top view of a binary tree is the set of nodes visible when the tree is viewed from the top.

You need to complete the given function. The input and printing of output will be handled by the driver code.

Input Format

The first line contains the number of test cases.

For each test case: The line contains a string giving array representation of a tree, if the root has no children give `N` in input.

Output Format

For each test case print the right view of the binary tree.

Example 1

Input

```
1
1 2 3
```

```
      1
    /   \
   2     3
```

Output

```
2 1 3
```

Explanation

'2', '1' and '3' are visible from the top view.

Example 2

Input

```
1
1 2 3 4

      1
    /   \
  2      3
 /
4
```

Output

```
4 2 1 3
```

Explanation

'4', '2', '1' and '3' are visible from the top view.

Constraints

1 <= T <= 10

1 <= N <= 10000

Topic Tags

- **Trees**

My code

```
// in java
import java.util.LinkedList;
import java.util.Queue;
import java.io.*;
import java.util.*;

class Node{
    int data;
    Node left;
    Node right;
    Node(int data){
        this.data = data;
        left=null;
        right=null;
    }
}


class Main {
    static Node buildTree(String str){
        if(str.length()==0 || str.charAt(0)=='N'){
            return null;
        }
        String ip[] = str.split(" ");
        Node root = new Node(Integer.parseInt(ip[0]));
        Queue<Node> queue = new LinkedList<>();
        queue.add(root);
        int i = 1;
        while(queue.size()>0 && i < ip.length) {
            Node currNode = queue.peek();
            queue.remove();
            String currVal = ip[i];
            if(!currVal.equals("N")) {
                currNode.left = new Node(Integer.parseInt(currVal));
                queue.add(currNode.left);
            }
            i++;
            if(i >= ip.length)
                break;
            currVal = ip[i];
            if(!currVal.equals("N")) {
                currNode.right = new Node(Integer.parseInt(currVal));
                queue.add(currNode.right);
            }
            i++;
        }
        return root;
    }
    void inOrder(Node node) {
        if (node == null) {
            return;
        }
        inOrder(node.left);
        System.out.print(node.data + " ");
        inOrder(node.right);
    }

        public static void main (String[] args) throws IOException{
                BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
```

```
                int t=Integer.parseInt(br.readLine());
                while(t-- > 0){
                        String s = br.readLine();
                        Node root = buildTree(s);
                        Solution tree = new Solution();
                        ArrayList<Integer> arr = tree.topView(root);
                        for(int x : arr)
                        System.out.print(x +" ");
                        System.out.println();
                }
        }
}


class qObj
{
    Node node;
    int level;
    qObj(Node n,int l)
    {
        node=n;
        level=l;
    }
}

class Solution
{
    public ArrayList<Integer> topView(Node root){
            Queue<Pair>q=new ArrayDeque<>();
        ArrayList<Integer>a=new ArrayList<>();
            TreeMap<Integer, Integer>hm=new TreeMap<>();
            q.add(new Pair(0, root));
            while(!q.isEmpty())
                {
                    Pair c=q.poll();
                    if(!hm.containsKey(c.d))
                    hm.put(c.d,c.nd.data);

                    if(c.nd.left!=null)
                        q.add(new Pair(c.d-1,c.nd.left));
                    if(c.nd.right!=null)
                        q.add(new Pair(c.d+1,c.nd.right));


                }
            for(int x: hm.keySet())
                a.add(hm.get(x));
            return a;
        }
          class Pair{
              int d;
              Node nd;
              Pair(int d,Node nd)
              {
                  this.d=d;
                  this.nd=nd;
              }
          }


    }
```