- **HARD**
- **Max Score: 40 Points**
- https://course.acciojob.com/idle?question=21d3de29-2828-4145-8aa5-c9d8ab42ee78

Right view of Binary tree

You are given a `root` pointer to the root of binary tree. You have to print the right view of the tree from top to bottom.

Note

The right view of a tree is the set of nodes that are visible from the right side.

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.

The second 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

```
1 3
```

Explanation

'1' and '3' are visible from the right side.

Example 2

Input:

```
1
1 2 3 N N 4
```

```
      1
    /   \
   2     3
         /
        4
```

Output

```
1 3 4
```

Explanation

'1', '3', and '4' are visible from the right side.

Constraints

1 <= T <= 10

1 <= N <= 10000

My code

```
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.rightView(root);
                            for(int x : arr)
                            System.out.print(x +" ");
                            System.out.println();
                    }
            }
}

class Solution{

     ArrayList<Integer> al = new ArrayList<>();
    int mxlvl =0;
     public void recur(Node root,int lvl)
     {
        if(root==null)
            return;
```

```
         if(mxlvl<lvl)
         {
              al.add(root.data);
             mxlvl = lvl;
         }
        recur(root.right,lvl+1);
        recur(root.left,lvl+1);
    }
   ArrayList<Integer> rightView(Node root) {
          if(root==null)
         return al;
       recur(root,1);
     return al;
   }
}

/*class Solution{
   ArrayList<Integer> rightView(Node root) {


   }
}
/*class Solution{

    ArrayList<Integer> al = new ArrayList<>();
   int mxlvl =0;
    public void recur(Node root,int lvl)
    {
        if(root==null)
            return;

         if(mxlvl<lvl)
         {
              al.add(root.data);
             mxlvl = lvl;
         }
        recur(root.right,lvl+1);
        recur(root.left,lvl+1);
    }
   ArrayList<Integer> rightView(Node root) {
          if(root==null)
         return al;
       recur(root,1);
```

```
        return al;
    }
}
*/
```