https://course.acciojob.com/idle?question=f74897ab-cbb8-4302-b031-f69c5d31c1ea

MEDIUM

Max Score: 40 Points

Maximum Width of tree

Given the `root` of a binary tree, return the maximum width of the given tree.

The maximum width of a tree is the maximum width among all levels.

The width of one level is defined as the length between the end-nodes (the leftmost and rightmost non-null nodes), where the null nodes between the end-nodes that would be present in a complete binary tree extending down to that level are also counted into the length calculation.

Input Format

The first line of input contains a number `n`.

The second line of input contains `n` space seperated integer.

Output Format

Return the maximum width of the given tree.

Example 1

Input

```
7
1 2 3 4 5 6 7
```

Output

```
4
```

Explanation

```
The given binary tree is
            1
          /   \
         2     3
        / \   / \
       4   5 6   7
the maximum width is at 3rd level {4,5,6,7}
```

Example 2

Input

```
4
1 2 3 4
```

Output

```
2
```

Explanation

```
The given tree is
            1
          /   \
         2     3
        /
       4
The maximum width exists in the second level with length 2 (3,2).
```

Constraints

1 <= n <= 10^3

Topic Tags

BFS

DFS

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;
}

public static void main(String[] args) throws IOException {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
```

```
        sc.nextLine();
        String s = sc.nextLine();
        Node root = buildTree(s);
        Solution tree = new Solution();
        int ans = tree.solve(root);
        System.out.println(ans);

        sc.close();
    }
}

class Solution {
 static int with(Node r)
        {
                if(r==null)
                        return 0;
                Queue<Node>q=new LinkedList<>();
                q.add(r);
                int ans=0;
                while(!q.isEmpty())
                        {
                        int t=q.size();
                                if(t>ans)ans=t;
                                for(int i=0;i<t;i++)
                                        {
                                                Node n=q.remove();
                                                if(n.left!=null)
                                                        q.add(n.left);
                                                if(n.right!=null)
                                                        q.add(n.right);
```

```
                                    }
                            }
                    return ans;
            }
        public int solve(Node root) {
            // your code here
         return   with(root);

        }
}
```