https://course.acciojob.com/idle?question=87b0d2e6-d58a-4149-a832-70355e59043b

- MEDIUM
- Max Score: 30 Points
-
-
-

Diameter of a Binary Tree

Given a `root` of a binary tree, write a function to get the diameter of the tree. The diameter of a binary tree is the length of the longest path between any two nodes in a tree. This path may or may not pass through the root.

Input Format

You are given a string `s` which describes the nodes of the binary tree. (The first element corresponds to the root, the second is the left child of the root and so on). In the function, you are provided with the root of the binary tree.

Output Format

Return the diameter of the binary tree.

Example 1

Input

```
8 2 1 3 N N 5
```

Output

```
5
```

Explanation

The longest path is between 3 and 5. The diameter is 5.

Example 2

Input

```
1 2 N
```

Output

```
2
```

Explanation

The longest path is between 1 and 2. The diameter is 2.

Constraints

1 <= N <= 10^5

1 <= Node Data <= 10^5

Topic Tags

- **Recursion**
- **Trees**

My code

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

```
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 {
        BufferedReader br =new BufferedReader(new
InputStreamReader(System.in));
        String s1 = br.readLine();
        Node root1 = buildTree(s1);
        Solution g = new Solution();
        System.out.println(g.diameter(root1));
    }
}


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

class Solution {
static  int height(Node node)
    {
         if(node==null)
```

```
            return 0;
        int i=height( node.left) ;
        int j=height(node.right) ;
        if(i>j)
            return i+1;
        return j+1;
    }
   static int max=0;
    public static int diameter(Node root) {
        // Your code here
        if(root==null)
            return 0;
        int h1=0,h2=0;
        if(root.left!=null)
         h1=height(root.left);
          if(root.right!=null)
         h2=height(root.right);
        if(max<(h1+h2))
            max=h1+h2;
        diameter( root.left);
        diameter( root.right);
        return max+1;
    }
}
```