632. Binary Tree Maximum Node

* [Description](http://lintcode.com/en/problem/binary-tree-maximum-node/" \l "description)
* [Notes](http://lintcode.com/en/problem/binary-tree-maximum-node/#note)
* [Testcase](http://lintcode.com/en/problem/binary-tree-maximum-node/#testcase)
* [Judge](http://lintcode.com/en/problem/binary-tree-maximum-node/#judge)

Find the maximum node in a binary tree, return the node.

Have you met this question in a real interview?

Yes

**Example**

Given a binary tree:

1

/ \

-5 2

/ \ / \

0 3 -4 -5

return the node with value 3.

<http://lintcode.com/en/problem/binary-tree-maximum-node/#>

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package javaapplication1;

import java.util.\*;

/\*\*

\*

\* @author Usuario

\*/

class TreeNode {

int val;

TreeNode left, right;

public TreeNode(int item) {

val = item;

left = right = null;

}

}

public class JavaApplication1 {

/\*\*

\* @param root

\* @param args the command line arguments

\* @return

\*/

public static TreeNode maxNode(TreeNode root) {

// write your code here

if (root == null) {

return null; //return Integer.MIN\_VALUE;

}

//keep the nodes in the path that are waiting to be visited

Stack<TreeNode> stack = new Stack();

TreeNode node = root;

//first node to be visited will be the left one

while (node != null) {

stack.push(node);

node = node.left;

}

TreeNode max = root;

// traverse the tree

while (stack.size() > 0) {

// visit the top node

node = stack.pop();

//System.out.print(node.data + " ");

//max = Math.max(max, node.data );

if(max != null && node.val > max.val) {

max = node;

}

if (node.right != null) {

node = node.right;

// the next node to be visited is the leftmost

while (node != null) {

stack.push(node);

node = node.left;

}

}

}

return max;

}

public static void main(String[] args) {

// TODO code application logic here

TreeNode tree = new TreeNode(1);

tree.left = new TreeNode(2);

tree.right = new TreeNode(3);

tree.left.left = new TreeNode(4);

tree.left.right = new TreeNode(5);

TreeNode max = maxNode(tree);

if(max != null) {

System.out.println(max.val);

}else{

System.out.println(0);

}

}

}