# Problem 1602: Find Nearest Right Node in Binary Tree

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

Given the

root

of a binary tree and a node

u

in the tree, return

the

nearest

node on the

same level

that is to the

right

of

u

, or return

null

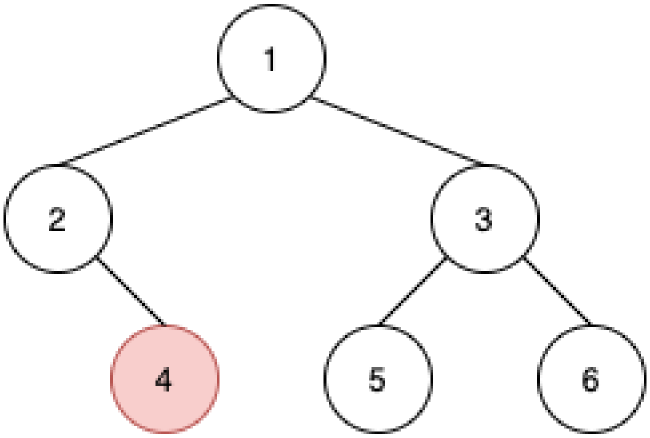
if

u

is the rightmost node in its level

.

Example 1:



Input:

root = [1,2,3,null,4,5,6], u = 4

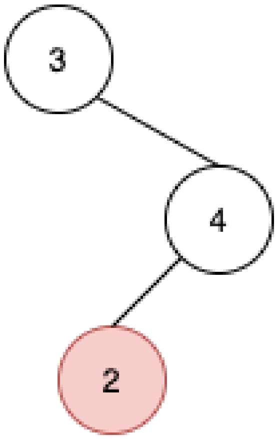
Output:

5

Explanation:

The nearest node on the same level to the right of node 4 is node 5.

Example 2:



Input:

root = [3,null,4,2], u = 2

Output:

null

Explanation:

There are no nodes to the right of 2.

Constraints:

The number of nodes in the tree is in the range

[1, 10

5

]

.

1 <= Node.val <= 10

5

All values in the tree are

distinct

.

u

is a node in the binary tree rooted at

root

.

## Code Snippets

**C++:**

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
* };
*/
class Solution {
public:
TreeNode* findNearestRightNode(TreeNode* root, TreeNode* u) {

}
};
```

**Java:**

```
/**
* Definition for a binary tree node.
* public class TreeNode {
```

```
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public TreeNode findNearestRightNode(TreeNode root, TreeNode u) {

}
}
```

### Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def findNearestRightNode(self, root: TreeNode, u: TreeNode) ->
Optional[TreeNode]:
```

### Python:

```
# Definition for a binary tree node.
# class TreeNode(object):
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution(object):
def findNearestRightNode(self, root, u):
"""
:type root: TreeNode
```

```
:type u: TreeNode
:rtype: TreeNode
"""
```

**JavaScript:**

```
/**
* Definition for a binary tree node.
* function TreeNode(val, left, right) {
* this.val = (val===undefined ? 0 : val)
* this.left = (left===undefined ? null : left)
* this.right = (right===undefined ? null : right)
* }
*/
/**
* @param {TreeNode} root
* @param {TreeNode} u
* @return {TreeNode}
*/
var findNearestRightNode = function(root, u) {

};
```

**TypeScript:**

```
/**
* Definition for a binary tree node.
* class TreeNode {
* val: number
* left: TreeNode | null
* right: TreeNode | null
* constructor(val?: number, left?: TreeNode | null, right?: TreeNode | null)
{
* this.val = (val===undefined ? 0 : val)
* this.left = (left===undefined ? null : left)
* this.right = (right===undefined ? null : right)
* }
* }
*/
/**
* @param {TreeNode} root
* @param {TreeNode} u
* @return {TreeNode}
*/
function findNearestRightNode(root: TreeNode, u: TreeNode): TreeNode | null {

};
```

**C#:**

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* public int val;
* public TreeNode left;
* public TreeNode right;
* public TreeNode(int val=0, TreeNode left=null, TreeNode right=null) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
public class Solution {
public TreeNode FindNearestRightNode(TreeNode root, TreeNode u) {

}
}
```

**C:**

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* struct TreeNode *left;
* struct TreeNode *right;
* };
*/



struct TreeNode* findNearestRightNode(struct TreeNode* root, struct TreeNode*
u){
}
```

### Go:

```
/**
* Definition for a binary tree node.
* type TreeNode struct {
* Val int
* Left *TreeNode
* Right *TreeNode
* }
*/
func findNearestRightNode(root *TreeNode, u *TreeNode) *TreeNode {

}
```

### Kotlin:

```
/**
* Example:
* var ti = TreeNode(5)
* var v = ti.`val`
* Definition for a binary tree node.
* class TreeNode(var `val`: Int) {
* var left: TreeNode? = null
* var right: TreeNode? = null
* }
*/
class Solution {
fun findNearestRightNode(root: TreeNode?, u: TreeNode?): TreeNode? {

}
}
```

### Swift:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* public var val: Int
* public var left: TreeNode?
* public var right: TreeNode?
* public init() { self.val = 0; self.left = nil; self.right = nil; }
* public init(_ val: Int) { self.val = val; self.left = nil; self.right = nil; }
* public init(_ val: Int, _ left: TreeNode?, _ right: TreeNode?) {
* self.val = val
* self.left = left
* self.right = right
* }
* }
*/
class Solution {
func findNearestRightNode(_ root: TreeNode?, _ u: TreeNode?) -> TreeNode? {

}
}
```

### Ruby:

```
# Definition for a binary tree node.
# class TreeNode
# attr_accessor :val, :left, :right
# def initialize(val = 0, left = nil, right = nil)
# @val = val
# @left = left
# @right = right
# end
# end
# @param {TreeNode} root
# @param {TreeNode} u
# @return {TreeNode}
def find_nearest_right_node(root, u)

end
```

### PHP:

```
/**
* Definition for a binary tree node.
* class TreeNode {
* public $val = null;
* public $left = null;
* public $right = null;
* function __construct($val = 0, $left = null, $right = null) {
* $this->val = $val;
* $this->left = $left;
* $this->right = $right;
* }
* }
*/
class Solution {

/**
* @param TreeNode $root
* @param TreeNode $u
* @return TreeNode
*/
function findNearestRightNode($root, $u) {

}
}
```

### Scala:

```
/**
* Definition for a binary tree node.
* class TreeNode(_value: Int = 0, _left: TreeNode = null, _right: TreeNode = null) {
* var value: Int = _value
* var left: TreeNode = _left
* var right: TreeNode = _right
* }
*/
object Solution {
def findNearestRightNode(root: TreeNode, u: TreeNode): TreeNode = {

}
}
```

## Solutions

### C++ Solution:

```
/*
* Problem: Find Nearest Right Node in Binary Tree
* Difficulty: Medium
* Tags: tree, search
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
TreeNode* findNearestRightNode(TreeNode* root, TreeNode* u) {

}
};
```

### Java Solution:

```
/**
* Problem: Find Nearest Right Node in Binary Tree
* Difficulty: Medium
* Tags: tree, search
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
```

```
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {
// TODO: Implement optimized solution
return 0;
}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public TreeNode findNearestRightNode(TreeNode root, TreeNode u) {

}
}
```

### Python3 Solution:

```
"""
Problem: Find Nearest Right Node in Binary Tree
Difficulty: Medium
Tags: tree, search

Approach: DFS or BFS traversal
Time Complexity: O(n) where n is number of nodes
Space Complexity: O(h) for recursion stack where h is height
"""

# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
```

```
class Solution:
def findNearestRightNode(self, root: TreeNode, u: TreeNode) ->
Optional[TreeNode]:
# TODO: Implement optimized solution
pass
```

### Python Solution:

```
# Definition for a binary tree node.
# class TreeNode(object):
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution(object):
def findNearestRightNode(self, root, u):
"""
:type root: TreeNode
:type u: TreeNode
:rtype: TreeNode
"""
```

### JavaScript Solution:

```
/**
* Problem: Find Nearest Right Node in Binary Tree
* Difficulty: Medium
* Tags: tree, search
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* function TreeNode(val, left, right) {
* this.val = (val===undefined ? 0 : val)
* this.left = (left===undefined ? null : left)
* this.right = (right===undefined ? null : right)
* }
```

```
*/
/**
* @param {TreeNode} root
* @param {TreeNode} u
* @return {TreeNode}
*/
var findNearestRightNode = function(root, u) {

};
```

### TypeScript Solution:

```
/**
* Problem: Find Nearest Right Node in Binary Tree
* Difficulty: Medium
* Tags: tree, search
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* class TreeNode {
* val: number
* left: TreeNode | null
* right: TreeNode | null
* constructor(val?: number, left?: TreeNode | null, right?: TreeNode | null) {
* this.val = (val===undefined ? 0 : val)
* this.left = (left===undefined ? null : left)
* this.right = (right===undefined ? null : right)
* }
* }
*/
/**
* @param {TreeNode} root
* @param {TreeNode} u
* @return {TreeNode}
*/
```

```
function findNearestRightNode(root: TreeNode, u: TreeNode): TreeNode | null {

};
```

### C# Solution:

```
/*
* Problem: Find Nearest Right Node in Binary Tree
* Difficulty: Medium
* Tags: tree, search
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* public class TreeNode {
* public int val;
* public TreeNode left;
* public TreeNode right;
* public TreeNode(int val=0, TreeNode left=null, TreeNode right=null) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
public class Solution {
public TreeNode FindNearestRightNode(TreeNode root, TreeNode u) {

}
}
```

### C Solution:

```
/*
* Problem: Find Nearest Right Node in Binary Tree
* Difficulty: Medium
* Tags: tree, search
```

```
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* struct TreeNode *left;
* struct TreeNode *right;
* };
*/


struct TreeNode* findNearestRightNode(struct TreeNode* root, struct TreeNode* u){

}
```

### Go Solution:

```
// Problem: Find Nearest Right Node in Binary Tree
// Difficulty: Medium
// Tags: tree, search
//
// Approach: DFS or BFS traversal
// Time Complexity: O(n) where n is number of nodes
// Space Complexity: O(h) for recursion stack where h is height

/**
* Definition for a binary tree node.
* type TreeNode struct {
* Val int
* Left *TreeNode
* Right *TreeNode
* }
*/
func findNearestRightNode(root *TreeNode, u *TreeNode) *TreeNode {

}
```

### Kotlin Solution:

```
/**
* Example:
* var ti = TreeNode(5)
* var v = ti.`val`
* Definition for a binary tree node.
* class TreeNode(var `val`: Int) {
* var left: TreeNode? = null
* var right: TreeNode? = null
* }
*/
class Solution {
fun findNearestRightNode(root: TreeNode?, u: TreeNode?): TreeNode? {

}
}
```

### Swift Solution:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* public var val: Int
* public var left: TreeNode?
* public var right: TreeNode?
* public init() { self.val = 0; self.left = nil; self.right = nil; }
* public init(_ val: Int) { self.val = val; self.left = nil; self.right =
nil; }
* public init(_ val: Int, _ left: TreeNode?, _ right: TreeNode?) {
* self.val = val
* self.left = left
* self.right = right
* }
* }
*/
class Solution {
func findNearestRightNode(_ root: TreeNode?, _ u: TreeNode?) -> TreeNode? {
}
}
```

### Ruby Solution:

```
# Definition for a binary tree node.
# class TreeNode
# attr_accessor :val, :left, :right
# def initialize(val = 0, left = nil, right = nil)
# @val = val
# @left = left
# @right = right
# end
# end
# @param {TreeNode} root
# @param {TreeNode} u
# @return {TreeNode}
def find_nearest_right_node(root, u)

end
```

### PHP Solution:

```
/**
* Definition for a binary tree node.
* class TreeNode {
* public $val = null;
* public $left = null;
* public $right = null;
* function __construct($val = 0, $left = null, $right = null) {
* $this->val = $val;
* $this->left = $left;
* $this->right = $right;
* }
* }
*/
class Solution {

/**
* @param TreeNode $root
* @param TreeNode $u
* @return TreeNode
*/
function findNearestRightNode($root, $u) {

}
}
```

### Scala Solution:

```
/**
* Definition for a binary tree node.
* class TreeNode(_value: Int = 0, _left: TreeNode = null, _right: TreeNode =
null) {
* var value: Int = _value
* var left: TreeNode = _left
* var right: TreeNode = _right
* }
*/
object Solution {
def findNearestRightNode(root: TreeNode, u: TreeNode): TreeNode = {

}
}
```