

# Problem 1676: Lowest Common Ancestor of a Binary Tree IV

## Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

Given the

root

of a binary tree and an array of

TreeNode

objects

nodes

, return

the lowest common ancestor (LCA) of

all the nodes

in

nodes

. All the nodes will exist in the tree, and all values of the tree's nodes are

unique

.

Extending the

definition of LCA on Wikipedia

: "The lowest common ancestor of

$n$

nodes

$p$

1

,

$p$

2

, ...,

$p$

$n$

in a binary tree

$T$

is the lowest node that has every

$p$

$i$

as a

descendant

(where we allow

a node to be a descendant of itself

) for every valid

i

". A

descendant

of a node

x

is a node

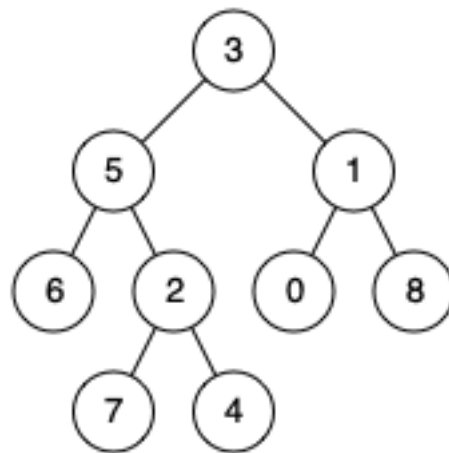
y

that is on the path from node

x

to some leaf node.

Example 1:



Input:

root = [3,5,1,6,2,0,8,null,null,7,4], nodes = [4,7]

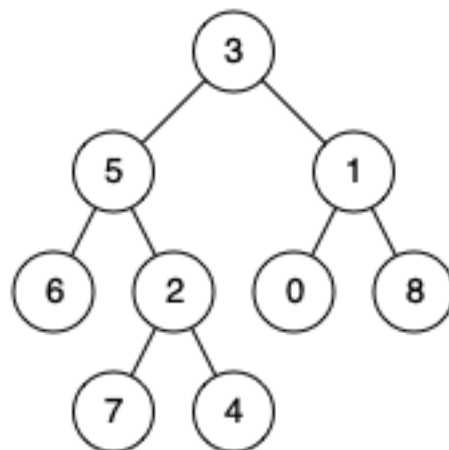
Output:

2

Explanation:

The lowest common ancestor of nodes 4 and 7 is node 2.

Example 2:



Input:

root = [3,5,1,6,2,0,8,null,null,7,4], nodes = [1]

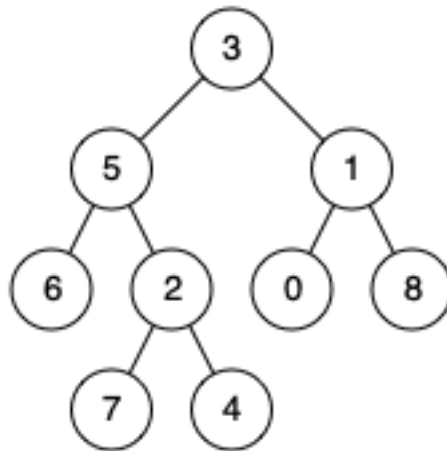
Output:

1

Explanation:

The lowest common ancestor of a single node is the node itself.

Example 3:



Input:

root = [3,5,1,6,2,0,8,null,null,7,4], nodes = [7,6,2,4]

Output:

5

Explanation:

The lowest common ancestor of the nodes 7, 6, 2, and 4 is node 5.

Constraints:

The number of nodes in the tree is in the range

[1, 10

4

]

.

-10

9

$\leq \text{Node.val} \leq 10$

9

All

Node.val

are

unique

.

All

nodes[i]

will exist in the tree.

All

nodes[i]

are distinct.

## 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* lowestCommonAncestor(TreeNode* root, vector<TreeNode*> &nodes) {

    }
};
```

### Java:

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *   int val;
 *   TreeNode left;
 *   TreeNode right;
 *   TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode[] nodes) {

    }
}
```

### Python3:

```

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

class Solution:
def lowestCommonAncestor(self, root: 'TreeNode', nodes: 'List[TreeNode]') ->
'TreeNode':

```

## Python:

```

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

class Solution(object):
def lowestCommonAncestor(self, root, nodes):
    """
    :type root: TreeNode
    :type nodes: List[TreeNode]
    """

```

## JavaScript:

```

/**
 * Definition for a binary tree node.
 * function TreeNode(val) {
 *   this.val = val;
 *   this.left = this.right = null;
 * }
 */
/**
 * @param {TreeNode} root
 * @param {TreeNode[]} nodes
 * @return {TreeNode}
 */
var lowestCommonAncestor = function(root, nodes) {

```

```
};
```

## C#:

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     public int val;
 *     public TreeNode left;
 *     public TreeNode right;
 *     public TreeNode(int x) { val = x; }
 * }
 */
public class Solution {
    public TreeNode LowestCommonAncestor(TreeNode root, TreeNode[] nodes) {

    }
}
```

## Solutions

### C++ Solution:

```
/*
 * Problem: Lowest Common Ancestor of a Binary Tree IV
 * Difficulty: Medium
 * Tags: array, tree, hash, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * 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* lowestCommonAncestor(TreeNode* root, vector<TreeNode*> &nodes) {

}
};

```

### Java Solution:

```

/**
 * Problem: Lowest Common Ancestor of a Binary Tree IV
 * Difficulty: Medium
 * Tags: array, tree, hash, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * 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(int x) { val = x; }
 * }
 */
class Solution {
public
TreeNode lowestCommonAncestor(TreeNode root, TreeNode[] nodes) {

}

}

```

### Python3 Solution:

```

"""
Problem: Lowest Common Ancestor of a Binary Tree IV
Difficulty: Medium
Tags: array, tree, hash, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

class Solution:
def lowestCommonAncestor(self, root: 'TreeNode', nodes: 'List[TreeNode]') ->
'TreeNode':
# TODO: Implement optimized solution
pass

```

## Python Solution:

```

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

class Solution(object):
def lowestCommonAncestor(self, root, nodes):
"""
:type root: TreeNode
:type nodes: List[TreeNode]
"""

```

## JavaScript Solution:

```

/**
 * Problem: Lowest Common Ancestor of a Binary Tree IV
 * Difficulty: Medium
 * Tags: array, tree, hash, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height
 */

/**
 * Definition for a binary tree node.
 * function TreeNode(val) {
 *   this.val = val;
 *   this.left = this.right = null;
 * }
 */
/**
 * @param {TreeNode} root
 * @param {TreeNode[]} nodes
 * @return {TreeNode}
 */
var lowestCommonAncestor = function(root, nodes) {

};

```

## C# Solution:

```

/*
 * Problem: Lowest Common Ancestor of a Binary Tree IV
 * Difficulty: Medium
 * Tags: array, tree, hash, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * 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 x) { val = x; }
* }
*/
public class Solution {
public TreeNode LowestCommonAncestor(TreeNode root, TreeNode[] nodes) {

}
}
```