

Problem 129: Sum Root to Leaf Numbers

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given the

root

of a binary tree containing digits from

0

to

9

only.

Each root-to-leaf path in the tree represents a number.

For example, the root-to-leaf path

1 -> 2 -> 3

represents the number

123

Return

the total sum of all root-to-leaf numbers

. Test cases are generated so that the answer will fit in a

32-bit

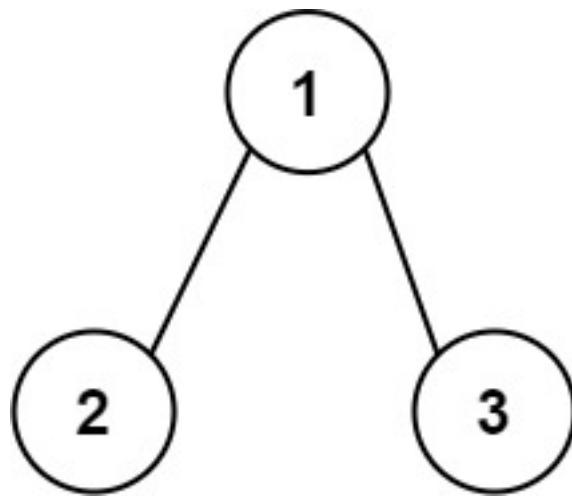
integer.

A

leaf

node is a node with no children.

Example 1:



Input:

root = [1,2,3]

Output:

25

Explanation:

The root-to-leaf path

1->2

represents the number

12

. The root-to-leaf path

1->3

represents the number

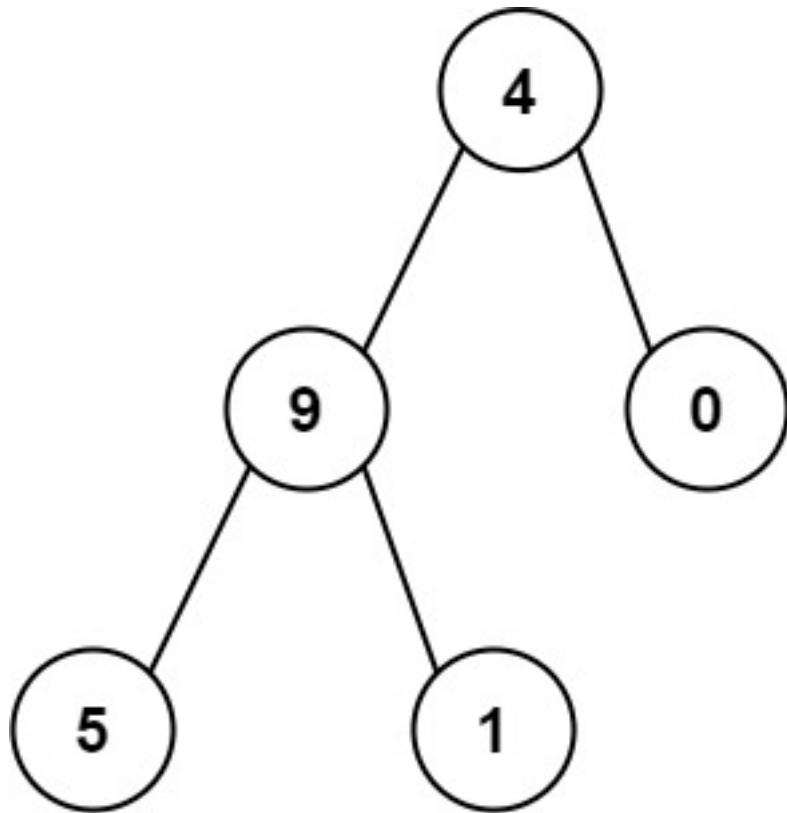
13

. Therefore, sum = 12 + 13 =

25

.

Example 2:



Input:

root = [4,9,0,5,1]

Output:

1026

Explanation:

The root-to-leaf path

4->9->5

represents the number 495. The root-to-leaf path

4->9->1

represents the number 491. The root-to-leaf path

4->0

represents the number 40. Therefore, sum = 495 + 491 + 40 =

1026

Constraints:

The number of nodes in the tree is in the range

[1, 1000]

0 <= Node.val <= 9

The depth of the tree will not exceed

10

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:  
int sumNumbers(TreeNode* root) {  
  
}  
};
```

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 int sumNumbers(TreeNode root) {  
  
}  
}
```

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 sumNumbers(self, root: Optional[TreeNode]) -> int:
```

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 sumNumbers(self, root):
        """
        :type root: Optional[TreeNode]
        :rtype: int
        """

```

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
 * @return {number}
 */
var sumNumbers = function(root) {

};


```

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)
        *
    }
}

function sumNumbers(root: TreeNode | null): number {
}

```

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 int SumNumbers(TreeNode root) {
        }
    }
}

```

C:

```

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

```

```
}
```

Go:

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

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 sumNumbers(root: TreeNode?): Int {  
  
    }  
}
```

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 sumNumbers(_ root: TreeNode?) -> Int {
}
}

```

Rust:

```

// Definition for a binary tree node.
// #[derive(Debug, PartialEq, Eq)]
// pub struct TreeNode {
//   pub val: i32,
//   pub left: Option<Rc<RefCell<TreeNode>>,
//   pub right: Option<Rc<RefCell<TreeNode>>,
// }
//
// impl TreeNode {
//   #[inline]
//   pub fn new(val: i32) -> Self {
//     TreeNode {
//       val,
//       left: None,
//       right: None
//     }
//   }
// }
use std::rc::Rc;
use std::cell::RefCell;
impl Solution {
  pub fn sum_numbers(root: Option<Rc<RefCell<TreeNode>>>) -> i32 {
}

```

```
}
```

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
# @return {Integer}
def sum_numbers(root)

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 $root
 * @return Integer
 */
function sumNumbers($root) {
```

```
}
```

```
}
```

Dart:

```
/**  
 * Definition for a binary tree node.  
 * class TreeNode {  
 * int val;  
 * TreeNode? left;  
 * TreeNode? right;  
 * TreeNode([this.val = 0, this.left, this.right]);  
 * }  
 */  
class Solution {  
int sumNumbers(TreeNode? root) {  
  
}  
}
```

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 sumNumbers(root: TreeNode): Int = {  
  
}  
}
```

Elixir:

```
# Definition for a binary tree node.  
#  
# defmodule TreeNode do
```

```

# @type t :: %__MODULE__{
#   val: integer,
#   left: TreeNode.t() | nil,
#   right: TreeNode.t() | nil
# }
# defstruct val: 0, left: nil, right: nil
# end

defmodule Solution do
@spec sum_numbers(root :: TreeNode.t | nil) :: integer
def sum_numbers(root) do

end
end

```

Erlang:

```

%% Definition for a binary tree node.

%%
%% -record(tree_node, {val = 0 :: integer(),
%% left = null :: 'null' | #tree_node{},
%% right = null :: 'null' | #tree_node{}}).

-spec sum_numbers(Root :: #tree_node{} | null) -> integer().
sum_numbers(Root) ->
.
```

Racket:

```

; Definition for a binary tree node.
#|

; val : integer?
; left : (or/c tree-node? #f)
; right : (or/c tree-node? #f)
(struct tree-node
  (val left right) #:mutable #:transparent)

; constructor
(define (make-tree-node [val 0])
  (tree-node val #f #f))

```

```
| #  
  
(define/contract (sum-numbers root)  
  (-> (or/c tree-node? #f) exact-integer?)  
)
```

Solutions

C++ Solution:

```
/*  
 * Problem: Sum Root to Leaf Numbers  
 * 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) {}  
 // TODO: Implement optimized solution  
 return 0;  
 }  
 * TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}  
 // TODO: Implement optimized solution  
 return 0;  
 }  
 * TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),  
 right(right) {}  
 // TODO: Implement optimized solution  
 return 0;  
 }  
 * };
```

```

*/
class Solution {
public:
int sumNumbers(TreeNode* root) {

}
};


```

Java Solution:

```

/**
 * Problem: Sum Root to Leaf Numbers
 * 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 int sumNumbers(TreeNode root) {

```

```
}
```

```
}
```

Python3 Solution:

```
"""
Problem: Sum Root to Leaf Numbers
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 sumNumbers(self, root: Optional[TreeNode]) -> int:
        # 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 sumNumbers(self, root):
        """
:type root: Optional[TreeNode]
:rtype: int
"""


```

JavaScript Solution:

```
/**  
 * Problem: Sum Root to Leaf Numbers  
 * 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  
 * @return {number}  
 */  
var sumNumbers = function(root) {  
  
};
```

TypeScript Solution:

```
/**  
 * Problem: Sum Root to Leaf Numbers  
 * 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)
}
*/
function sumNumbers(root: TreeNode | null): number {
}

```

C# Solution:

```

/*
 * Problem: Sum Root to Leaf Numbers
 * 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 int SumNumbers(TreeNode root) {  
  
    }  
}
```

C Solution:

```
/*  
 * Problem: Sum Root to Leaf Numbers  
 * 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;  
 * };  
 */  
int sumNumbers(struct TreeNode* root) {  
  
}
```

Go Solution:

```
// Problem: Sum Root to Leaf Numbers  
// 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 sumNumbers(root *TreeNode) int {

}

```

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 sumNumbers(root: TreeNode?): Int {
        ...
    }
}

```

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 sumNumbers(_ root: TreeNode?) -> Int {

}
}

```

Rust Solution:

```

// Problem: Sum Root to Leaf Numbers
// 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.
// #[derive(Debug, PartialEq, Eq)]
// pub struct TreeNode {
// pub val: i32,
// pub left: Option<Rc<RefCell<TreeNode>>>,
// pub right: Option<Rc<RefCell<TreeNode>>>,
// }
//
// impl TreeNode {
// #[inline]
// pub fn new(val: i32) -> Self {
// TreeNode {
// val,
// left: None,
// right: None
// }
// }
// }
use std::rc::Rc;
use std::cell::RefCell;

```

```

use std::cell::RefCell;
impl Solution {
    pub fn sum_numbers(root: Option<Rc<RefCell<TreeNode>>>) -> i32 {
        ...
    }
}

```

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
# @return {Integer}
def sum_numbers(root)

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 $root
 * @return Integer
 */
function sumNumbers($root) {

}
}

```

Dart Solution:

```

/**
 * Definition for a binary tree node.
 * class TreeNode {
 * int val;
 * TreeNode? left;
 * TreeNode? right;
 * TreeNode([this.val = 0, this.left, this.right]);
 * }
 */
class Solution {
int sumNumbers(TreeNode? root) {

}
}

```

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 sumNumbers(root: TreeNode): Int = {

}

```

```
}
```

Elixir Solution:

```
# Definition for a binary tree node.  
#  
# defmodule TreeNode do  
#   @type t :: %__MODULE__{  
#     val: integer,  
#     left: TreeNode.t() | nil,  
#     right: TreeNode.t() | nil  
#   }  
#   defstruct val: 0, left: nil, right: nil  
# end  
  
defmodule Solution do  
  @spec sum_numbers(root :: TreeNode.t | nil) :: integer  
  def sum_numbers(root) do  
  
  end  
end
```

Erlang Solution:

```
%% Definition for a binary tree node.  
%%  
%% -record(tree_node, {val = 0 :: integer(),  
%%   left = null :: 'null' | #tree_node{},  
%%   right = null :: 'null' | #tree_node{}}).  
  
-spec sum_numbers(Root :: #tree_node{} | null) -> integer().  
sum_numbers(Root) ->  
.
```

Racket Solution:

```
; Definition for a binary tree node.  
#|  
  
; val : integer?  
; left : (or/c tree-node? #f)
```

```
; right : (or/c tree-node? #f)
(struct tree-node
  (val left right) #:mutable #:transparent)

; constructor
(define (make-tree-node [val 0])
  (tree-node val #f #f))

|# 

(define/contract (sum-numbers root)
  (-> (or/c tree-node? #f) exact-integer?))
```