







Omer Goldberg December 05, 2020 · ( ) 11 min

# Problem 🤥

Given an array where elements are sorted in ascending order, convert it to a height balanced BST.

For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

Example

Input

[-10, -3, 0, 5, 9]

Ouput

One possible answer is: [0, -3, 9, -10, null, 5], which represents the following height balanced BST:

```
0
    / \
  -3 5
-10
        9
```

## Real World Applications



I love this question because it illustrates so clearly the relation between a sorted array and a binary search tree! Once this sinks in hopefully things click:)

# The Finer Details $\nearrow$



### **Example 1**

Input

Output



#### 1357

## Example 2

Input

[1,2], len 1, middle 0

1 \

By loooking at these examples, one of the first things that stands out is that the root is always the middle of the array! This makes - sense, a min height binary tree will equal (almost) amount of nodes on each side. Therefore, the logical split here is to pick the root as the middle element.

Great, so since this is a recursive problem in nature let's figure out the base case.

#### Base case

Array of size 0 or 1? Perfect, all done here!

Array of size 2? We could do this in two ways. Let's make the larger one the root and the smaller the left child.

#### **Recurrence Relation**

Let's use our recursive function to get the left and right sub tree.

## Recursive Solution ?



```
class Solution:
    def sortedArrayToBST(self, nums):
        arrLen = len(nums)
        if arrLen == 0:
            return
        if arrLen == 1:
            return TreeNode(nums[0], None, None)
        if arrLen == 2:
            if nums[0] <= nums[1]:</pre>
                leftChild = TreeNode(nums[0], None, None)
                return TreeNode(nums[1], leftChild, None)
            else:
                rightChild = TreeNode(nums[0], None, None)
                return TreeNode(nums[1], None, rightChild)
        m = int(arrLen / 2)
        1 = self.sortedArrayToBST(nums[0:m])
        r = self.sortedArrayToBST(nums[m + 1:])
        return TreeNode(nums[m], 1, r)
```

## Complexity Analysis

## Time complexity

O(n)

One set of constant operations for every node.

**Space complexity**  $O(\log(n))$ 

Log(n) recurive calls that take the stack space.

# Takeaways 🏂 🥙



It's verrrry easy to go from a sorted array to binary search tree and vice versa! The only difference between them is how we choose to represent the data, so don't be intimidiated!

## Resources 듣 💻





#### Leetcode

#recursion **Tags** 

#binarysearchtrees

trees

leetcode









0 Comments

#### What do you think?

0 Responses











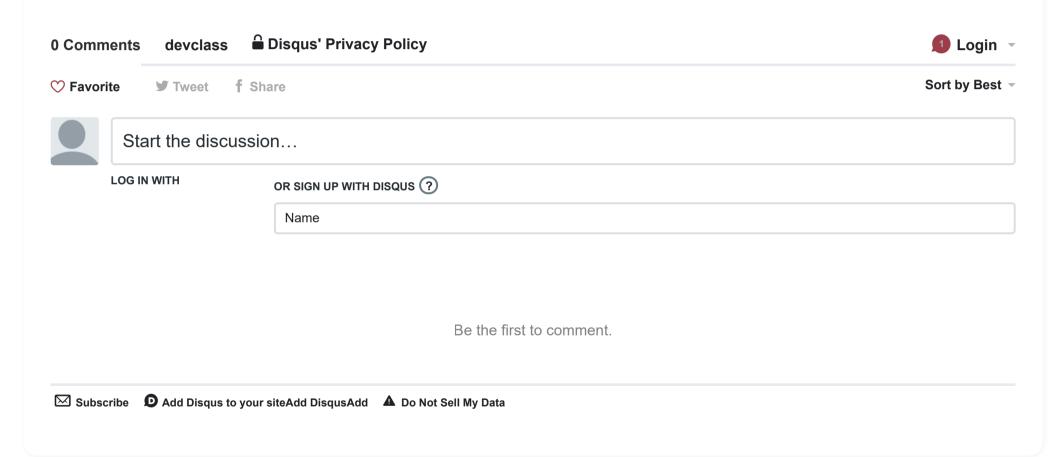
Upvote

Funny

Love

Surprised

Sad





## **Omer Goldberg**

#### **Tech Lead**

crafting mobile experiences @ instagram. tech Lead @ facebook. former algorithm and fullstack Teacher @ israel tech challenge. crypto crazy.

Expertise	Social Media
Algorithms	o instagram
System Design	<b>in</b> linkedin
Android	website
Blockchain	

## Related Posts 18

**Binary Search Trees** 

## **Delete Node from Binary Search Tree**

Assaf Elovic December 05, 2020

**Binary Search Trees** 

### Find Min and Max Elements in Binary Search Tree

Omer Goldberg December 05, 2020

**Binary Search Trees** 

## **Find Predecessor in Binary Search Tree**

Assaf Elovic December 05, 2020

**Binary Search Trees** 

### **Find Second Largest in Binary Search Tree**

Assaf Elovic December 05, 2020

**Binary Search Trees** 

### **Insert Node into Binary Search Tree**

Assaf Elovic December 05, 2020

**Binary Search Trees** 

## Is Valid Binary Search Tree

Omer Goldberg December 05, 2020

github linkedin facebook

Privacy Policy Cookie Policy Terms Of Use

Home Learn Product Instructors

// devclass.io

© 2022, All Rights Reserved.