Q1 https://practice.geeksforgeeks.org/problems/inorder-successor-in-bst/1

Inorder Successor in BST

Easy Accuracy: **34.97%** Submissions: **104K+** Points: **2**

Given a BST, and a reference to a Node x in the BST. Find the Inorder Successor of the given node in the BST.

Example 1:

Input:

```
      2
    /   \
   1     3
K(data of x) = 2
```

Output: 3

Q2 Inorder predecessor

Q3 https://leetcode.com/problems/two-sum-iv-input-is-a-bst/

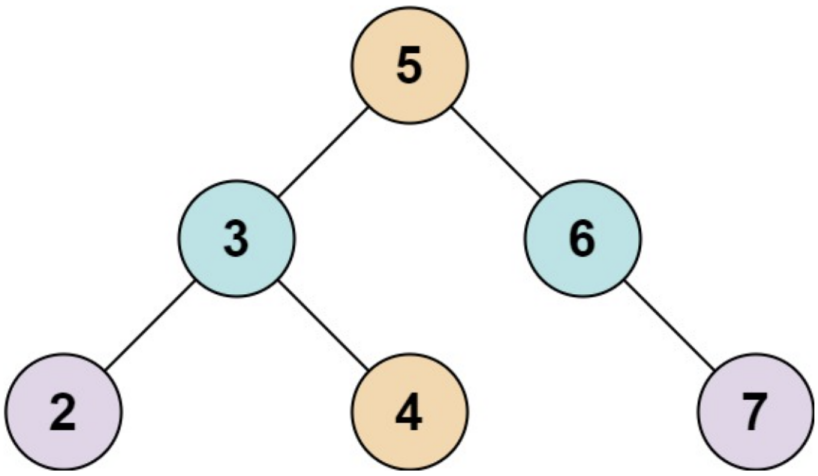
653. Two Sum IV - Input is a BST

Easy 6.4K 252

Companies

Given the `root` of a binary search tree and an integer `k`, return `true` *if there exist two elements in the BST such that their sum is equal to* `k`*, or* `false` *otherwise.*

Example 1:




```
Input: root = [5,3,6,2,4,null,7], k = 9
Output: true
```

Q4 Ceil in a BST
https://practice.geeksforgeeks.org/problems/implementing-ceil-in-bst/1

Given a BST and a number **X**, find **Ceil of X**.

Note: Ceil(X) is a number that is either equal to X or is immediately greater than X.

If Ceil could not be found, return -1.

Example 1:

```
Input:
        5
     /    \
    1      7
     \
      2
       \
        3
X = 3
Output: 3
Explanation: We find 3 in BST, so ceil
of 3 is 3.
```

Q5 Floor in a BST

Q6
https://leetcode.com/problems/construct-binary-search-tree-from-preorder-traversal/description/

1008. Construct Binary Search Tree from Preorder Traversal

Medium 5.8K 71

Companies

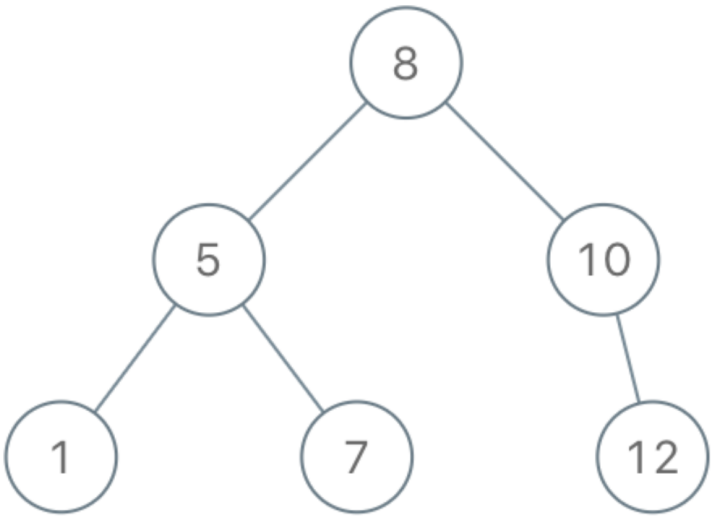
Given an array of integers preorder, which represents the **preorder traversal** of a BST (i.e., **binary search tree**), construct the tree and return *its root*.

It is **guaranteed** that there is always possible to find a binary search tree with the given requirements for the given test cases.

A **binary search tree** is a binary tree where for every node, any descendant of `Node.left` has a value **strictly less than** `Node.val`, and any descendant of `Node.right` has a value **strictly greater than** `Node.val`.

A **preorder traversal** of a binary tree displays the value of the node first, then traverses `Node.left`, then traverses `Node.right`.

Example 1:




```
Input: preorder = [8,5,1,7,10,12]
Output: [8,5,10,1,7,null,12]
```

Q7 https://leetcode.com/problems/binary-search-tree-iterator/

173. Binary Search Tree Iterator

Medium 8.1K 467

Companies

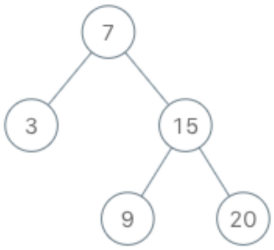
Implement the `BSTIterator` class that represents an iterator over the **in-order traversal** of a binary search tree (BST):

- `BSTIterator(TreeNode root)` Initializes an object of the `BSTIterator` class. The `root` of the BST is given as part of the constructor. The pointer should be initialized to a non-existent number smaller than any element in the BST.
- `boolean hasNext()` Returns `true` if there exists a number in the traversal to the right of the pointer, otherwise returns `false`.
- `int next()` Moves the pointer to the right, then returns the number at the pointer.

Notice that by initializing the pointer to a non-existent smallest number, the first call to `next()` will return the smallest element in the BST.

You may assume that `next()` calls will always be valid. That is, there will be at least a next number in the in-order traversal when `next()` is called.

Example 1:




```
Input
["BSTIterator", "next", "next", "hasNext", "next",
"hasNext", "next", "hasNext", "next", "hasNext"]
[[[7, 3, 15, null, null, 9, 20]], [], [], [], [], [],
[], [], [], []]
Output
[null, 3, 7, true, 9, true, 15, true, 20, false]
```

Explanation

```
BSTIterator bSTIterator = new BSTIterator([7, 3, 15,
null, null, 9, 20]);
bSTIterator.next();    // return 3
bSTIterator.next();    // return 7
bSTIterator.hasNext(); // return True
bSTIterator.next();    // return 9
bSTIterator.hasNext(); // return True
bSTIterator.next();    // return 15
bSTIterator.hasNext(); // return True
bSTIterator.next();    // return 20
bSTIterator.hasNext(); // return False
```

Q8 https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/description/

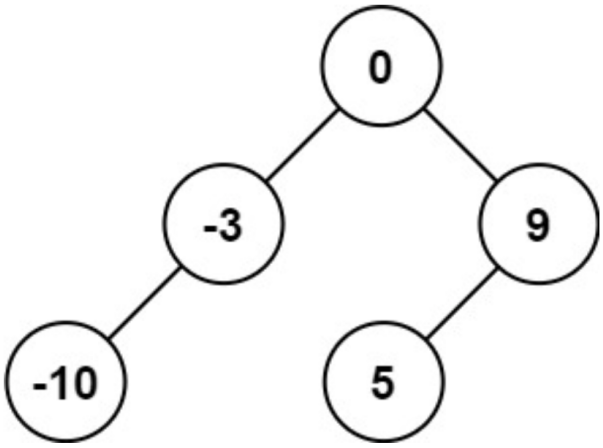
108. Convert Sorted Array to Binary Search Tree

Easy 10.3K 507

Companies

Given an integer array `nums` where the elements are sorted in **ascending order**, convert *it to a* ***height-balanced*** *binary search tree.*

Example 1:



Q8