LeetCode Explore Problems Interview New Contest Discuss Store

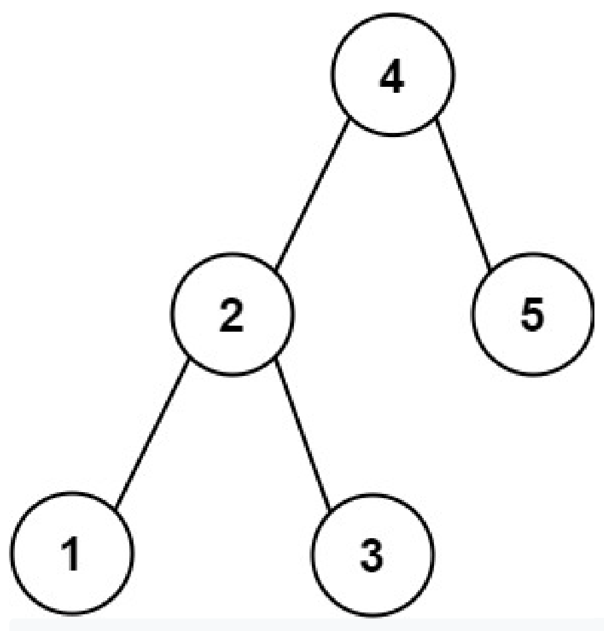
Description | Solution | Discuss (594) | Submissions

270. Closest Binary Search Tree Value

Easy 1255 83 Add to List Share

Given the `root` of a binary search tree and a `target` value, return *the value in the BST that is closest to the* `target`.

Example 1:



```
Input: root = [4,2,5,1,3], target = 3.714286
Output: 4
```

Example 2:

```
Input: root = [1], target = 4.428571
Output: 1
```

Constraints:

- The number of nodes in the tree is in the range `[1, ` 10^4`]`.
- `0 <= Node.val <= `10^9
- -10^9` <= target <= `10^9

Accepted 223,035 | Submissions 424,932

Seen this question in a real interview before? Yes No

Companies i

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook	12
Amazon	2
Goldman Sachs	2
HBO	2

Related Topics

Binary Search · Tree · Depth-First Search · Binary Search Tree · Binary Tree

Similar Questions

Count Complete Tree Nodes	Medium
Closest Binary Search Tree Value II	Hard
Search in a Binary Search Tree	Easy

Java | Autocomplete

```java
class Solution {
  public void inorder(TreeNode root, List<Integer> nums) {
    if (root == null) return;
    inorder(root.left, nums);
    nums.add(root.val);
    inorder(root.right, nums);
  }

  public int closestValue(TreeNode root, double target) {
    List<Integer> nums = new ArrayList();
    inorder(root, nums);
    return Collections.min(nums, new Comparator<Integer>() {
      @Override
      public int compare(Integer o1, Integer o2) {
        return Math.abs(o1 - target) < Math.abs(o2 - target) ? -1 : 1;
      }
    });
  }
}


class Solution {
  public int closestValue(TreeNode root, double target) {
    int val, closest = root.val;
    while (root != null) {
      val = root.val;
      closest = Math.abs(val - target) < Math.abs(closest - target) ? val : closest;
      root =  target < root.val ? root.left : root.right;
    }
    return closest;
  }
}




class Solution {
  public int closestValue(TreeNode root, double target) {
    LinkedList<TreeNode> stack = new LinkedList();
    long pred = Long.MIN_VALUE;

    while (!stack.isEmpty() || root != null) {
      while (root != null) {
        stack.add(root);
        root = root.left;
      }
      root = stack.removeLast();

      if (pred <= target && target < root.val)
        return Math.abs(pred - target) < Math.abs(root.val - target) ? (int)pred : root.val;

      pred = root.val;
      root = root.right;
    }
    return (int)pred;
  }
}
```

Problems | Pick One | Prev | /99 | Next | Console | Contribute i | Run Code | Submit