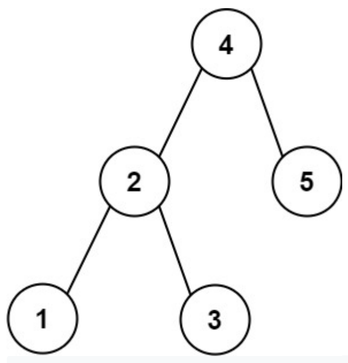
272. Closest Binary Search Tree Value II

Hard 905 26 Add to List Share

Given the `root` of a binary search tree, a `target` value, and an integer `k`, return *the* `k` *values in the BST that are closest to the* `target`. You may return the answer in **any order**.

You are **guaranteed** to have only one unique set of `k` values in the BST that are closest to the `target`.

Example 1:




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

Example 2:

```
Input: root = [1], target = 0.000000, k = 1
Output: [1]
```

Constraints:

- The number of nodes in the tree is `n`.
- `1 <= k <= n <= 10^4`.
- `0 <= Node.val <= 10^9`
- `-10^9 <= target <= 10^9`

Follow up: Assume that the BST is balanced. Could you solve it in less than `O(n)` runtime (where `n = total nodes`)?

Accepted 82,026 | Submissions 149,278

Seen this question in a real interview before? Yes No

Companies

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

LinkedIn | 11

Related Topics

Two Pointers Stack Tree Depth-First Search Binary Search Tree Heap (Priority Queue) Binary Tree

Similar Questions

Binary Tree Inorder Traversal Easy

Closest Binary Search Tree Value Easy

Hide Hint 1

Consider implement these two helper functions:

i. `getPredecessor(N)`, which returns the next smaller node to N.

ii. `getSuccessor(N)`, which returns the next larger node to N.

Hide Hint 2

Try to assume that each node has a parent pointer, it makes the problem much easier.

Hide Hint 3

Without parent pointer we just need to keep track of the path from the root to the current node using a stack.

Hide Hint 4

You would need two stacks to track the path in finding predecessor and successor node separately.

Java Autocomplete

```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 List<Integer> closestKValues(TreeNode root, double target, int k) {

    }
}
```