Problem 272: Closest Binary Search Tree Value II

Problem Information

Difficulty: Hard
Acceptance Rate: 60.87%
Paid Only: Yes
Tags: Two Pointers, Stack, Tree, Depth-First Search, Binary Search Tree, Heap (Priority Queue), Binary Tree

Problem Description

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:

![](https://assets.leetcode.com/uploads/2021/03/12/closest1-1-tree.jpg)

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 <= 104`. * `0 <= Node.val <= 109` * `-109 <= target <= 109`

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

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:
vector<int> closestKValues(TreeNode* root, double target, int k) {

}
};
```

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) {

}
}
```

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 closestKValues(self, root: Optional[TreeNode], target: float, k: int) ->
List[int]:
```