Description | Editorial | Solutions (1.7K) | Submissions

669. Trim a Binary Search Tree

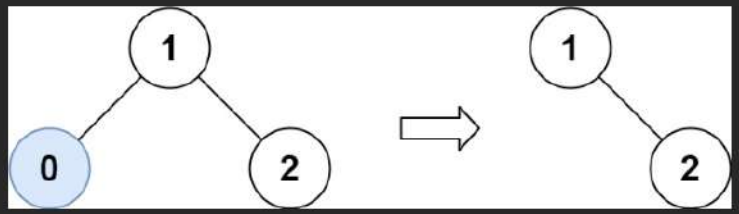
Medium 5.5K 248

Companies

Given the `root` of a binary search tree and the lowest and highest boundaries as `low` and `high`, trim the tree so that all its elements lies in `[low, high]`. Trimming the tree should **not** change the relative structure of the elements that will remain in the tree (i.e., any node's descendant should remain a descendant). It can be proven that there is a **unique answer**.

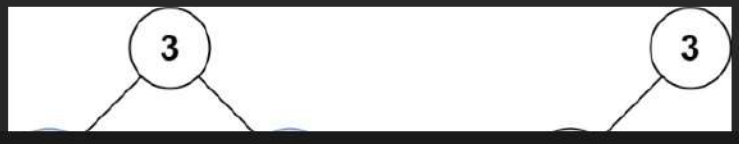
Return *the root of the trimmed binary search tree*. Note that the root may change depending on the given bounds.

Example 1:




```
Input: root = [1,0,2], low = 1, high = 2
Output: [1,null,2]
```

Example 2:



Java | Auto

```java
class Solution {
    public TreeNode trimBST(TreeNode root, int low, int high) {
        Queue<TreeNode> queue = new LinkedList<>();
        queue.offer(root);
        boolean found = false;
        while (!queue.isEmpty()) {
            TreeNode curr = queue.poll();
            if (low <= curr.val && curr.val <= high) {
                root = curr;
                found = true;
                break;
            } else {
                if (curr.left != null) queue.offer(curr.left);
                if (curr.right != null) queue.offer(curr.right);
            }
        }
        if (!found) return low <= root.val && root.val <= high ? root : null;
        trimBSTDFS(root, low, high);
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

root =
[1,0,2]

low =
1

high =

Console | Run | Submit

Description | Editorial | Solutions (1K) | Submissions

673. Number of Longest Increasing Subsequence

Medium 5.9K 238

Companies

Given an integer array `nums`, return *the number of longest increasing subsequences.*

Notice that the sequence has to be **strictly** increasing.

Example 1:

```
Input: nums = [1,3,5,4,7]
Output: 2
Explanation: The two longest increasing subsequences are [1, 3, 4, 7] and [1, 3, 5, 7].
```

Example 2:

```
Input: nums = [2,2,2,2,2]
Output: 5
Explanation: The length of the longest increasing subsequence is 1, and there are 5 increasing subsequences of length 1, so output 5.
```

Constraints:

- `1 <= nums.length <= 2000`
- `-10^6 <= nums[i] <= 10^6`

Java | Auto

```java
class Solution {
    public int findNumberOfLIS(int[] nums) {
        int n = nums.length;
        int[] lis = new int[n];
        int[] fq = new int[n];
        lis[0] = 1;
        fq[0] = 1;
        int lo = 1;

        for (int i = 1; i < nums.length; i++) {
            int mx = 0;
            int c = 1;
            for (int j = 0; j < i; j++) {
                if (nums[j] < nums[i]) {
                    if (lis[j] > mx) {
                        mx = lis[j];
                        c = fq[j];
                    } else if (lis[j] == mx) {
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

```
nums =
[1,3,5,4,7]
```

Output

```
2
```

Console | Run | Submit