Description | Editorial | Solutions (7.9K) | Submissions

98. Validate Binary Search Tree

Medium | 15.6K | 1.3K

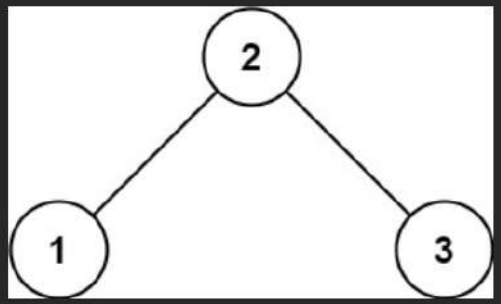
Companies

Given the `root` of a binary tree, *determine if it is a valid binary search tree (BST).*

A **valid BST** is defined as follows:

- The left subtree of a node contains only nodes with keys **less than** the node's key.
- The right subtree of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

Example 1:




```
Input: root = [2,1,3]
Output: true
```

Example 2:

Java | Auto

```java
*/
class Solution {
    public boolean isValidBST(TreeNode root) {
        return isValid(root, null, null);
    }

    boolean isValid(TreeNode root, Integer max, Integer min) {

        if(root == null)
            return true;

        if((max != null && root.val >= max) || (min != null && root.val <= min)) {
            return false;
        }

        return isValid(root.left, root.val, min) && isValid(root.right, max, root.val);
    }
}
```

Ln 32, Col 2

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

root =
[2,1,3]

Output

true

Console | Run | Submit

Description | Editorial | Solutions (2.6K) | Submissions

97. Interleaving String

Medium 7.5K 434

Companies

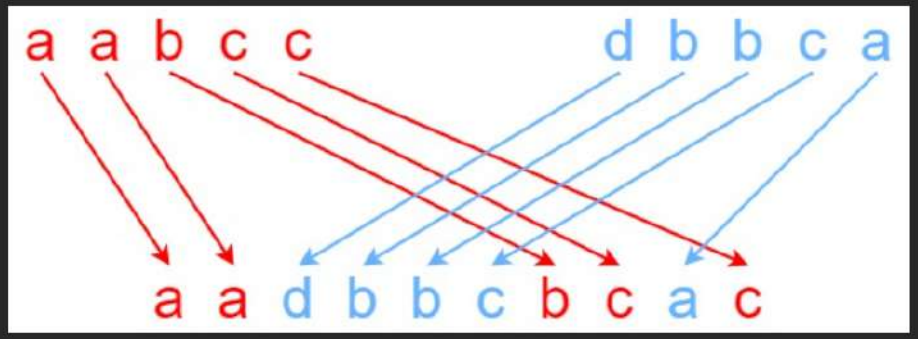
Given strings `s1`, `s2`, and `s3`, find whether `s3` is formed by an **interleaving** of `s1` and `s2`.

An **interleaving** of two strings `s` and `t` is a configuration where `s` and `t` are divided into `n` and `m` substrings respectively, such that:

- $s = s_1 + s_2 + ... + s_n$
- $t = t_1 + t_2 + ... + t_m$
- $|n - m| <= 1$
- The **interleaving** is $s_1 + t_1 + s_2 + t_2 + s_3 + t_3 + ...$ or $t_1 + s_1 + t_2 + s_2 + t_3 + s_3 + ...$

Note: `a + b` is the concatenation of strings `a` and `b`.

Example 1:



Java | Auto

```java
public class Solution {
    public boolean isInterleave(String s1, String s2, String s3) {
        int m = s1.length(), n = s2.length(), l = s3.length();
        if (m + n != l) return false;

        boolean[] dp = new boolean[n + 1];
        dp[0] = true;

        for (int j = 1; j <= n; ++j) {
            dp[j] = dp[j - 1] && s2.charAt(j - 1) == s3.charAt(j - 1);
        }

        for (int i = 1; i <= m; ++i) {
            dp[0] = dp[0] && s1.charAt(i - 1) == s3.charAt(i - 1);
            for (int j = 1; j <= n; ++j) {
                dp[j] = (dp[j] && s1.charAt(i - 1) == s3.charAt(i + j - 1)) || (dp[j - 1] && s2.charAt(j - 1) == s3.charAt(i + j - 1));
            }
```

Ln 22, Col 2

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

s1 =
"aabcc"

s2 =
"dbbca"

Console | Run | Submit