Description | Editorial | Solutions (7.1K) | Submissions

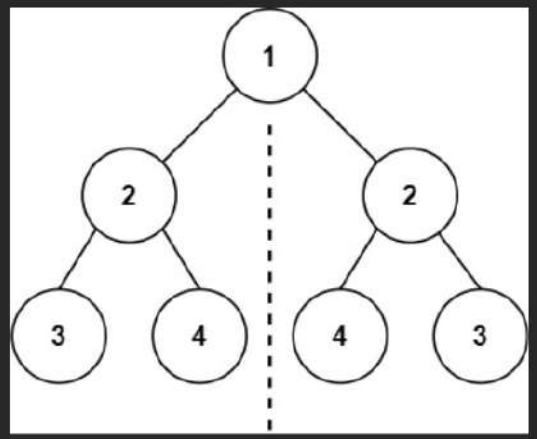
101. Symmetric Tree

Easy | 13.7K | 310

Companies

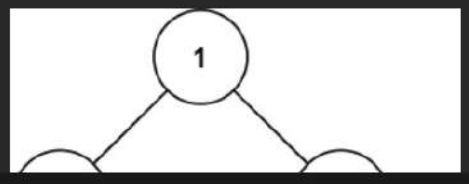
Given the `root` of a binary tree, *check whether it is a mirror of itself* (i.e., symmetric around its center).

Example 1:




```
Input: root = [1,2,2,3,4,4,3]
Output: true
```

Example 2:



Java | Auto

```java
 * }
 */
class Solution {
    public boolean isSymmetric(TreeNode root) {
        if (root == null) {
            return true;
        }
        return isMirror(root.left, root.right);
    }

    private boolean isMirror(TreeNode node1, TreeNode node2) {
        if (node1 == null && node2 == null) {
            return true;
        }
        if (node1 == null || node2 == null) {
            return false;
        }
        return node1.val == node2.val && isMirror(node1.left, node2.right) && isMirror(node1.right, node2.left);
    }
}
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

root =
[1,2,2,3,4,4,3]

Output

true

Expected

Console | Run | Submit

Description | Editorial | Solutions (903) | Submissions

1027. Longest Arithmetic Subsequence

Medium | 3.5K | 156

Companies

Given an array `nums` of integers, return *the length of the longest arithmetic subsequence in* `nums`.

Note that:

- A **subsequence** is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements.
- A sequence `seq` is arithmetic if `seq[i + 1] - seq[i]` are all the same value (for `0 <= i < seq.length - 1`).

Example 1:

```
Input: nums = [3,6,9,12]
Output: 4
Explanation: The whole array is an arithmetic sequence with steps of length = 3.
```

Example 2:

```
Input: nums = [9,4,7,2,10]
Output: 3
Explanation: The longest arithmetic subsequence is [4,7,10].
```

Example 3:

```
Input: nums = [20,1,15,3,10,5,8]
Output: 4
Explanation: The longest arithmetic subsequence is [20,15,10,5].
```

Java | Auto

```java
class Solution {
    public int longestArithSeqLength(int[] nums) {
        int n = nums.length;
        if (n <= 2)
            return n;

        int longest = 2;
        Map<Integer, Integer>[] dp = new HashMap[n];

        for (int i = 0; i < n; i++) {
            dp[i] = new HashMap<>();
            for (int j = 0; j < i; j++) {
                int diff = nums[i] - nums[j];
                dp[i].put(diff, dp[j].getOrDefault(diff, 1) + 1);
                longest = Math.max(longest, dp[i].get(diff));
            }
        }

        return longest;
    }
}
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

nums =
[3,6,9,12]

Output

4

Expected

Console | Run | Submit