Previous Next

100. Same Tree

March 7, 2019 | 94K views

Average Rating: 4.41 (46 votes)

Given two binary trees, write a function to check if they are the same or not.

Two binary trees are considered the same if they are structurally identical and the nodes have the same value.

Example 1:

```
Input:     1         1
          / \       / \
         2   3     2   3

        [1,2,3],   [1,2,3]

Output: true
```

Example 2:

```
Input:     1         1
          /           \
         2             2

        [1,2],     [1,null,2]

Output: false
```

Example 3:

```
Input:     1         1
          / \       / \
         2   1     1   2

        [1,2,1],   [1,1,2]

Output: false
```

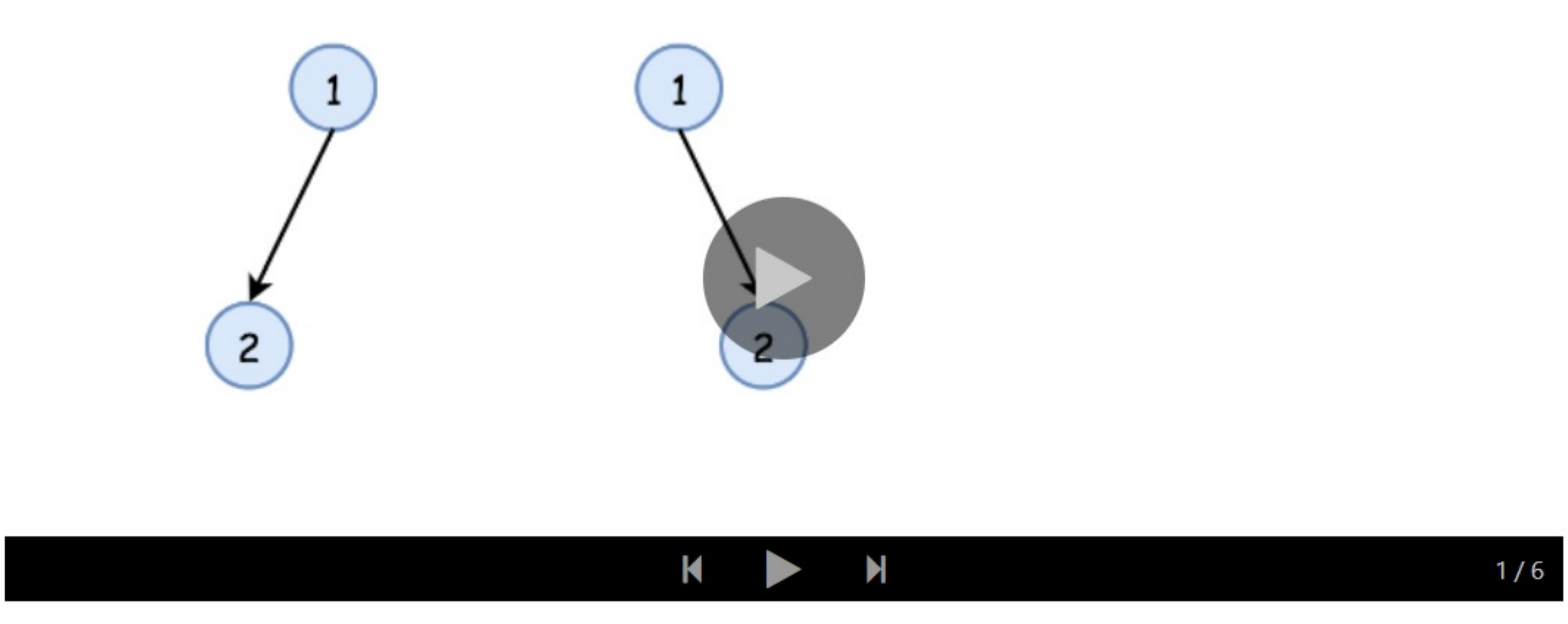
Solution

Approach 1: Recursion

Intuition

The simplest strategy here is to use recursion. Check if `p` and `q` nodes are not `None`, and their values are equal. If all checks are OK, do the same for the child nodes recursively.

Implementation



Java Python Copy

```
class Solution:
    def isSameTree(self, p, q):
        """
        :type p: TreeNode
        :type q: TreeNode
        :rtype: bool
        """
        # p and q are both None
        if not p and not q:
            return True
        # one of p and q is None
        if not q or not p:
            return False
        if p.val != q.val:
            return False
        return self.isSameTree(p.right, q.right) and \
               self.isSameTree(p.left, q.left)
```

Complexity Analysis

- Time complexity : $\mathcal{O}(N)$, where N is a number of nodes in the tree, since one visits each node exactly once.
- Space complexity : $\mathcal{O}(\log(N))$ in the best case of completely balanced tree and $\mathcal{O}(N)$ in the worst case of completely unbalanced tree, to keep a recursion stack.

Approach 2: Iteration

Intuition

Start from the root and then at each iteration pop the current node out of the deque. Then do the same checks as in the approach 1 :

- `p` and `p` are not `None`,
- `p.val` is equal to `q.val`,

and if checks are OK, push the child nodes.

Implementation

Java Python Copy

```
from collections import deque
class Solution:
    def isSameTree(self, p, q):
        """
        :type p: TreeNode
        :type q: TreeNode
        :rtype: bool
        """
        def check(p, q):
            # if both are None
            if not p and not q:
                return True
            # one of p and q is None
            if not q or not p:
                return False
            if p.val != q.val:
                return False
            return True

        deq = deque([(p, q),])
        while deq:
            p, q = deq.popleft()
            if not check(p, q):
                return False

            if p:
                deq.append((p.left, q.left))
```

Complexity Analysis

- Time complexity : $\mathcal{O}(N)$ since each node is visited exactly once.
- Space complexity : $\mathcal{O}(\log(N))$ in the best case of completely balanced tree and $\mathcal{O}(N)$ in the worst case of completely unbalanced tree, to keep a deque.

Rate this article: ★★★★★

Previous Next

Comments: 38 Sort By

Type comment here... (Markdown is supported)

Preview Post

none0 ★ 308 June 7, 2019 4:51 AM

An easier and intuitive iterative solution (beats 100% both):
The idea is to store both the root values in a queue, and later dequeue both two compare them.

```
class Solution {
```

Read More

136 Share Reply

SHOW 9 REPLIES

kevinhynes ★ 286 June 28, 2019 6:32 PM Report

Many comments here are questioning the space complexity of the iterative approach, and I agree this should be reviewed by the experts. I believe the space complexity is dependent on using an iterative DFS (stack) vs BFS (queue). Here's what I think, with code samples/test cases below:

Read More

20 Share Reply

SHOW 1 REPLY

mengmengli100 ★ 22 April 20, 2019 7:19 AM Report

A question regards the space complexity of the iterative solution:
Why is O(log(N)) in the best case? I thought it was also O(N).
I tried myself and found that the ArrayDeque "deqP" always keeps N/2's items in the best case of completely balanced tree. Because in every while loop, we remove one item from and add two items (left and right) into deqP.

Read More

18 Share Reply

SHOW 8 REPLIES

terrible_whiteboard ★ 626 May 19, 2020 6:21 PM

I made a video if anyone is having trouble understanding the solution (clickable link)
https://youtu.be/G9wwY-cmuiE

Read More

17 Share Reply

lxnn ★ 282 October 4, 2019 5:54 AM Report

It isn't necessary to use a queue in the iterative implementation. In fact the two solutions are the same except that we use the call stack in one and make our own stack in the other.

```
class Solution:
```

Read More

9 Share Reply

SHOW 1 REPLY

mohanmunisifreddy ★ 6 July 23, 2019 5:16 AM

class Solution {
public boolean isSameTree(TreeNode p, TreeNode q) {
if(p==null || q==null) return p==q;
return p.val==q.val && isSameTree(p.left, q.left) && isSameTree(p.right, q.right);
}

Read More

6 Share Reply

SHOW 2 REPLIES

shAdow2654 ★ 12 October 2, 2019 7:15 PM Report

Can anyone please explain the space complexity in both the recursive and iterative case please!!!

3 Share Reply

SHOW 1 REPLY

JAMESJJ78 ★ 203 April 10, 2019 7:45 PM

Yessirr

3 Share Reply

SHOW 1 REPLY

Marinebattery ★ 3 March 22, 2019 9:06 PM

beauty

3 Share Reply

chenmengjie ★ 2 March 28, 2019 9:39 PM Report

great

2 Share Reply

‹ 1 2 3 4 ›