100. Same Tree ☐ (/problems/same-tree/)

March 7, 2019 | 104K views

Average Rating: 4.41 (54 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                      ≣ Articles  >  100. Same Tree  ▾
           / \       / \
          2   1     1   2

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

Output: false
```
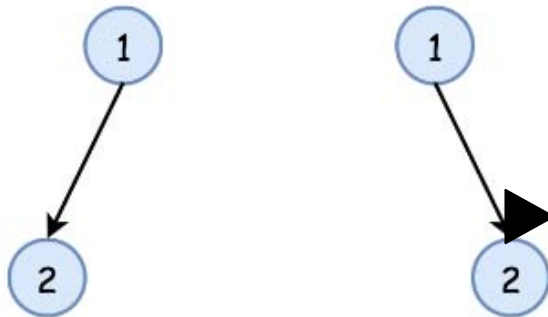
Solution

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

    ◀   ▶   ▶| 1 / (

```
Java    Python                                                    ☰ Articles  ›  100: Same Tree ▼     📋 Copy

  1    class Solution:
  2        def isSameTree(self, p, q):
  3            """
  4            :type p: TreeNode
  5            :type q: TreeNode
  6            :rtype: bool
  7            """
  8            # p and q are both None
  9            if not p and not q:
 10                return True
 11            # one of p and q is None
 12            if not q or not p:
 13                return False
 14            if p.val != q.val:
 15                return False
 16            return self.isSameTree(p.right, q.right) and \
 17                    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**   ☰ Articles › 100: Same Tree ▾

📋 Copy

```python
1   from collections import deque
2   class Solution:
3       def isSameTree(self, p, q):
4           """
5           :type p: TreeNode
6           :type q: TreeNode
7           :rtype: bool
8           """
9           def check(p, q):
10              # if both are None
11              if not p and not q:
12                  return True
13              # one of p and q is None
14              if not q or not p:
15                  return False
16              if p.val != q.val:
17                  return False
18              return True
19
20          deq = deque([(p, q),])
21          while deq:
22              p, q = deq.popleft()
23              if not check(p, q):
24                  return False
25
26              if p:
27                  deq.append((p.left, q.left))
28                  deq.append((p.right, q.right))
29
30          return True
```

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 (/articles/sliding-window-maximum/)   Next ❭ (/articles/gas-station/)

Comments: ⓐ 40   Sort By ▾

Type comment here... (Markdown is supported)

👁 Preview   Post

none0 (/none0)   ★ 321   ⊙ June 6, 2019 7:21 PM

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

149 ∧ ∨   ⓒ Share   ↩ Reply

SHOW 9 REPLIES

kevinhynes (/kevinhynes)   ★ 295   ⊙ June 28, 2019 9:02 AM

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

23 ∧ ∨   ⓒ Share   ↩ Reply

SHOW 1 REPLY

mengmengli100 (/mengmengli100)   ★ 22   ⊙ April 19, 2019 9:49 PM

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

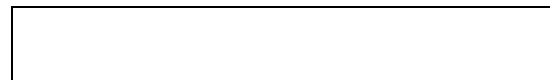18 ∧ ∨   ⓒ Share   ↩ Reply

SHOW 8 REPLIES

terrible_whiteboard (/terrible_whiteboard)   ★ 629   ⊙ May 19, 2020 8:51 AM

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

Read More

17 ∧ ∨   ⓒ Share   ↩ Reply

lxnn (/lxnn)   ★ 286   ⊙ October 3, 2019 8:24 PM

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.

Read More

10 ∧ ∨   ⓒ Share   ↩ Reply

SHOW 1 REPLY

mohanmunisifreddy (/mohanmunisifreddy) ★ 6 ⏲ July 22, 2019 7:46 PM

```
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 (/shadow2654) ★ 12 ⏲ October 2, 2019 9:45 AM

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

3 ∧ ∨ | ⤤ Share | ↩ Reply

SHOW 1 REPLY

JAMESJJ78 (/jamesjj78) ★ 207 ⏲ April 10, 2019 10:15 AM

Yessirr

3 ∧ ∨ | ⤤ Share | ↩ Reply

SHOW 1 REPLY

Marinebattery (/marinebattery) ★ 3 ⏲ March 22, 2019 11:36 AM

beauty

3 ∧ ∨ | ⤤ Share | ↩ Reply

chenmengjie (/chenmengjie) ★ 2 ⏲ March 28, 2019 12:09 PM

great

2 ∧ ∨ | ⤤ Share | ↩ Reply

‹ ① 2 3 4 ›

Help Center (/support/) | Terms (/terms/) | Privacy (/privacy/)

United States (/region/)