7 lines Easy Java Solution

242

dietpepsi
Reputation: 3447

Some used stack. Some used the depth of a stack. Here I use a different perspective. In a binary tree, if we consider null as leaves, then

- all non-null node provides 2 outdegree and 1 indegree (2 children and 1 parent), except root
- all null node provides 0 outdegree and 1 indegree (0 child and 1 parent).

Suppose we try to build this tree. During building, we record the difference between out degree and in degree `diff` = `outdegree - indegree`. When the next node comes, we then decrease `diff` by 1, because the node provides an in degree. If the node is not `null`, we increase diff by `2`, because it provides two out degrees. If a serialization is correct, diff should never be negative and diff will be zero when finished.

```
public boolean isValidSerialization(String preorder) {
    String[] nodes = preorder.split(",");
    int diff = 1;
    for (String node: nodes) {
        if (--diff < 0) return false;
        if (!node.equals("#")) diff += 2;
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
    }
    return diff == 0;
}
```

0

JulieWang
Reputation: ★ -1

Thank you for providing this very neat solution!

0

gaooo
Reputation: ★ 0

Thank you for such a neat solution!
I have a follow-up question on this: does this also work if the String given is an inorder of postorder traversal?
Thank you for your time!

2

james.wang.cccgmail.com
Reputation: ★ 8

Thanks for such a neat solution dietpepsi. I spent sometime trying to generalize this solution to InOrder and PostOrder traversal.

1. I think the condition that diff == 0 at the end holds for all three, first of all
2. Through observation, I concluded the following, which I am NOT certain if correct and complete or not.

So will really appreciate it if someone can offer some better idea how to generalize this approach to the InOrder and PostOrder:

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
public boolean isValidSerialization_PostOrder(String postorder) {
        String[] nodes = postorder.split(",");
        int diff = 1;
        for (String node : nodes) {
                diff--;
                if (!node.equals("#")) diff += 2;
                // Post-Order traversal fail criteria
                if (diff > 0) return false;
        }
        return diff == 0;
}

public boolean isValidSerialization_InOrder(String inorder) {
        String[] nodes = inorder.split(",");
        int diff = 1;
        for (String node : nodes) {
                diff--;
                if (!node.equals("#")) diff += 2;
                // In-Order traversal fail criteria
                if (diff > 1) return false;
        }
        return diff == 0;
}
```

6

dietpepsi @james.wang.cccgmail.com

Reputation: ★ 3447

generalize the idea to postorder is easy. Just reverse traversal the strings, it is the same.

But the tricky one is inorder. After all inorder is not able to deserialize even if the serialization is correct.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

For example, given "#,1,#,2,#" ◦ It can either be a tree rooted with 1 or a tree rooted with 2. Both are valid. Thus I really doubt inorder can verify easily anyhow.

46

lixx2100
Reputation: ★ 656

Nice solution. My solution is quite similar to yours.

If we treat `null`'s as leaves, then the binary tree will always be **full**. A full binary tree has a good property that `# of leaves = # of nonleaves + 1`. Since we are given a pre-order serialization, we just need to find the **shortest** prefix of the serialization sequence satisfying the property above. If such prefix does not exist, then the serialization is definitely invalid; otherwise, the serialization is valid if and only if the prefix is the **entire** sequence.

```
// Java Code
public boolean isValidSerialization(String preorder) {
    int nonleaves = 0, leaves = 0, i = 0;
    String[] nodes = preorder.split(",");
    for (i=0; i<nodes.length && nonleaves + 1 != leaves; i++)
        if (nodes[i].equals("#")) leaves++;
        else nonleaves++;
    return nonleaves + 1 == leaves && i == nodes.length;
}
```

0

ElementNotFoundException
Reputation: ★ 1098

Very nice solution. Let's assume

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
If a serialization is correct, diff should never be negative and diff will be zero when finished
```

is sounded, how do we argue the opposite is also true, i.e.,

```
if diff never becomes negative and diff is zero at the end, then the serialization is correct
```

That is, we know `a -> b` is true, but it does not necessarily mean `b -> a` is also true. The method I used also has this kind of unresolved question (if we are strict to ourselves) and I cannot explain it well yet.

0

Will
Reputation: ★ 25

"If a serialization is correct, diff should never be negative and diff will be zero when finished."
How did you get this conclusion? Sorry I'm not good at graph and indegree/outdegree thing. Thanks!

1

RainbowSecret @lixx2100
Reputation: ★ 722

Only count the number of the null-node and the non-null node can ensure the tree is OK ????? Is this condition Complete ???

0

RainbowSecret
Reputation: ★ 722

I am wondering how you think of this idea ?

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

0

lixx2100 @lixx2100
Reputation: 656

I am not just comparing the two counters, but the following actually. :)

> *we just need to find the shortest prefix of the serialization sequence satisfying the property above. If such prefix does not exist, then the serialization is definitely invalid; otherwise, the serialization is valid if and only if the prefix is the entire sequence.*

1

JerryH
Reputation: 0

Is this solution only work for full binary tree?

0

hpplayer @james.wang.cccgmail.com
Reputation: 727

Why pick int diff = 1? I guess you wanna force invalid input has diff < 0 during the program?

0

lisen
Reputation: 24

Thank you for sharing this creative solution.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

0

evilcoder
Reputation: ★ 15

thanks for sharing this wonderful solution!

0

eidision
Reputation: ★ 3

In my mind, this solution could solve preorder and postorder but inorder, just do some modify on "'if (--diff < 0) return false;"'

1

lanbeiliu @james.wang.cccgmail.com
Reputation: ★ 1

because the root provides 2 outdegree and no indegree, but in the for circulation, the root node is treated as a normal node

0

franklipolo
Reputation: ★ 2

in my mind, in-order will be quite easy. the sequence will be 01010101010101...

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

0

dietpepsi
Reputation: ★ 3447

yes. inoder is like that. But the problem itself is somewhat ill posed.

0

JulianWang12
Reputation: ★ 25

When it reaches "diff < 0" when processing the current node, it means it already reaches "diff == 0" after processing last node, which means nodes before the current one can already form a tree, thus the current node is an extra one, so do following nodes.

0

JulianWang12
Reputation: ★ 25

I think you probably made a logic error, and so does @dietpepsi in his explanation. If we use this solution here, we must assume the following is true, which I think is related to preorder traversal.

```
if diff never becomes negative and diff is zero at the end, then the serialization is correct
```

SOLUTION-SHARING 15815 EASIEST 238 JAVA 7710

44 POSTS

12878 VIEWS

Log in to reply

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com