

DescriptionSolutionDiscuss (398)Submissions

536. Construct Binary Tree from String

Medium👍 787🗨 129💖 Add to List🔖 Share

You need to construct a binary tree from a string consisting of parenthesis and integers.

The whole input represents a binary tree. It contains an integer followed by zero, one or two pairs of parenthesis. The integer represents the root's value and a pair of parenthesis contains a child binary tree with the same structure.

You always start to construct the **left** child node of the parent first if it exists.

Example 1:

```
graph TD; 4((4)) --- 2((2)); 4 --- 6((6)); 2 --- 3((3)); 2 --- 1((1)); 6 --- 5((5))
```

Input: s = "4(2(3)(1))(6(5))"
Output: [4,2,6,3,1,5]

Example 2:

Input: s = "4(2(3)(1))(6(5)(7))"
Output: [4,2,6,3,1,5,7]

Example 3:

Input: s = "-4(2(3)(1))(6(5)(7))"
Output: [-4,2,6,3,1,5,7]

Constraints:

- 0 <= s.length <= 3 * 10⁴
- s consists of digits, '(', ')', and '-' only.

Accepted 63,120 | Submissions 115,759

Seen this question in a real interview before?

Yes

No

Companies

Facebook | 12

Related Topics

StringTreeDepth-First SearchBinary Tree

Similar Questions

Construct String from Binary Tree

JavaAutocomplete

```
1 class Solution {
2     public TreeNode str2tree(String s) {
3         return this.str2treeInternal(s, 0).getKey();
4     }
5
6     public Pair<Integer, Integer> getNumber(String s, int index) {
7
8         boolean isNegative = false;
9
10        // A negative number
11        if (s.charAt(index) == '-') {
12            isNegative = true;
13            index++;
14        }
15
16        int number = 0;
17        while (index < s.length() && Character.isDigit(s.charAt(index))) {
18            number = number * 10 + (s.charAt(index) - '0');
19            index++;
20        }
21
22        return new Pair<Integer, Integer>(isNegative ? -number : number, index);
23    }
24
25    public Pair<TreeNode, Integer> str2treeInternal(String s, int index) {
26
27        if (index == s.length()) {
28            return new Pair<TreeNode, Integer>(null, index);
29        }
30
31        // Start of the tree will always contain a number representing
32        // the root of the tree. So we calculate that first.
33        Pair<Integer, Integer> numberData = this.getNumber(s, index);
34        int value = numberData.getKey();
35        index = numberData.getValue();
36
37        TreeNode node = new TreeNode(value);
38        Pair<TreeNode, Integer> data;
39
40        // Next, if there is any data left, we check for the first subtree
41        // which according to the problem statement will always be the left child.
42        if (index < s.length() && s.charAt(index) == '(') {
43            data = this.str2treeInternal(s, index + 1);
44            node.left = data.getKey();
45            index = data.getValue();
46        }
47
48        // Indicates a right child
49        if (node.left != null && index < s.length() && s.charAt(index) == ')') {
50            data = this.str2treeInternal(s, index + 1);
51            node.right = data.getKey();
52            index = data.getValue();
53        }
54
55        return new Pair<TreeNode, Integer>(node, index < s.length() && s.charAt(index) == ')' ? index + 1 : index);
56    }
57 }
58
59 }
```

Problems

✖ Pick One

< Prev

8/99

Next >

Console

Contribute!

▶ Run Code

Submit