Construct Binary Tree from String

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:

```
Input: "4(2(3)(1))(6(5))"
Output: return the tree root node representing the following tree:

       4
     /   \
    2     6
   / \   /
  3   1 5
```

Note:

1. There will only be `'('`, `')'`, `'-'` and `'0'` ~ `'9'` in the input string.
2. An empty tree is represented by `""` instead of `"()"`.

Solution 1

```
public TreeNode str2tree(String s) {
    if (s == null || s.length() == 0) return null;
    int firstParen = s.indexOf("(");
    int val = firstParen == -1 ? Integer.parseInt(s) : Integer.parseInt(s.substri
ng(0, firstParen));
    TreeNode cur = new TreeNode(val);
    if (firstParen == -1) return cur;
    int start = firstParen, leftParenCount = 0;
    for (int i=start;i<s.length();i++) {
        if (s.charAt(i) == '(') leftParenCount++;
        else if (s.charAt(i) == ')') leftParenCount--;
        if (leftParenCount == 0 && start == firstParen) {cur.left = str2tree(s.su
bstring(start+1,i)); start = i+1;}
        else if (leftParenCount == 0) cur.right = str2tree(s.substring(start+1,i)
);
    }
    return cur;
}
```

written by compton_scatter original link here

Solution 2

```
public class Solution {
    public TreeNode str2tree(String s) {
        // Base case
        if (s.length() == 0) return null;

        // Create root
        int i = 0, j = 0;
        while (j < s.length() && (Character.isDigit(s.charAt(j)) || s.charAt(j) =
= '-')) j++;
        TreeNode root = new TreeNode(Integer.parseInt(s.substring(i, j)));

        // Left child
        if (j < s.length()) {
            i = j;
            int count = 1;
            while (j + 1 < s.length() && count != 0) {
                j++;
                if (s.charAt(j) == ')') count--;
                if (s.charAt(j) == '(') count++;
            }
            root.left = str2tree(s.substring(i + 1, j));
        }

        j++;
        // Right child
        if (j < s.length()) {
            root.right = str2tree(s.substring(j + 1, s.length() - 1));
        }

        return root;
    }
}
```

written by shawngao original link here

Solution 3

```
public class Solution {
    public TreeNode str2tree(String s) {
        Stack<TreeNode> stack = new Stack<>();
        for(int i = 0, j = i; i < s.length(); i++, j = i){
            char c = s.charAt(i);
            if(c == ')')    stack.pop();
            else if(c >= '0' && c <= '9' || c == '-'){
                while(i + 1 < s.length() && s.charAt(i + 1) >= '0' && s.charAt(i
+ 1) <= '9') i++;
                TreeNode currentNode = new TreeNode(Integer.valueOf(s.substring(j
, i + 1)));
                if(!stack.isEmpty()){
                    TreeNode parent = stack.peek();
                    if(parent.left != null)    parent.right = currentNode;
                    else parent.left = currentNode;
                }
                stack.push(currentNode);
            }
        }
        return stack.isEmpty() ? null : stack.peek();
    }
}
```

written by fallcreek original link here