CS 284 B: Tree Exercise
Spring 2022

1 Assignment Policies

Collaboration Policy. Homework will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. Excerpts of code presented in class can be used.

Your code must include a comment with your name and section .

2 Assignment

Parsing balanced parenthesis.
In class we have learned that every sequence of balanced parenthesis can be generated using the following context-free grammar:

$$\begin{aligned} E &\rightarrow EE \\ E &\rightarrow (E) \\ E &\rightarrow \oslash \end{aligned}$$

For example, the sequence `(())()` can be parsed as the following tree:

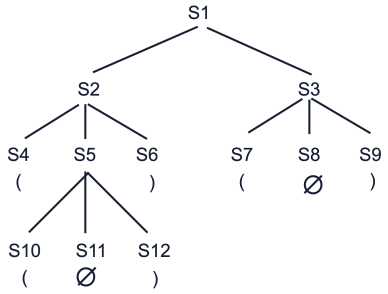

Figure 1: The parsing tree of the sequence `(())()`

You job in this problem is to implement the method `parse` of the class `BalancedParenTree.java`.

`BalancedParenTree.java` is the class for parsing the balanced parenthesis string. It contains the following methods and data fields:

(1) A private class `Node` which defines every node in the parsing tree. The `Node` class contains 2 data fields:

- `String terminal;`
- `Node[] children;`

In a non-leaf node (i.e., S_1, S_2, S_3, S_5 in Figure 1), `terminal` is set to null, e.g., S_1`.terminal = null;` in a leaf node (i.e., $S_4, S_6, ..., S_{12}$ in Figure 1), `terminal` is the terminal string, which can be one of the following 3 symbols: (1) '`(`', (2) '`)`', (3) '`N`'. For example, S_8`.terminal = 'N'`. In a non-leaf node, `children` stores the reference to a non-leaf node's children, e.g., S_1`.children =`$\{S_2, S_3\}$; in a leaf node, `children` is set to null, e.g., S_8`.children = null`.

(2) A public data field `Node root;` which stores the reference to the root of the parse tree, e.g., in Figure 1, `root` refers to node S_1;

(3) A public method `parse`:

```
public void parse(String paren_str);
```

where `paren_str` is the balanced parenthesis string, e.g., `(())()`. You may assume the sequence contains only `(` and `)`. You need to implement this method. The method `parse` in your code should at least generate the parse tree for the following balanced parentheses: `()`, `()()`, `(())`, `(())()`, `(())()()`. An example of the parsing tree is shown in Figure 1.

Important: Since when we test your code, we need to compare the expected result with the return value of the method `toString` which needs `this.root` as the parameter. Therefore, **at the end of the function `parse`, set the data field `root` (defined in (2)) to the `Node` object which references the *root* of the parsing tree**:

```
this.root = root_node; // root_node references the root of the parsing tree
```

(4) A public method `toString` which creates the string representation of BalancedParenTree by calling the method `toStringHelper`. For example, the method `toString` should return `(())()` for the tree in Figure 1. You don't need to modify this method;

(5) A private method `toStringHelper` which traverses the tree starting from `this.root` and returns the terminal symbols from left to right. You don't need to modify this method.

3 Submission instructions

- You can create other constructors you need in the class `BalancedParenTree.java`.
- You should submit `BalancedParenTree.java` to Gradescope.