Daily Coding Problem #199

Problem

This problem was asked by Facebook.

Given a string of parentheses, find the balanced string that can be produced from it using the minimum number of insertions and deletions. If there are multiple solutions, return any of them.

For example, given "(()", you could return "(())". Given "))()(", you could return "()()()()".

Solution

To solve this problem, first note that there is no significant difference between using insertions or deletions. To see this, think about what we would need to do to balance an unmatched "(" in a string. Whether we decide to delete it, or to append a ")" immediately after, it would still take one operation. So for simplicity, we will choose to use only insertions.

Once this is clear, we can see that a greedy solution is indeed possible, similar to what we would do to determine if the string were balanced. Namely, we can keep a counter, and increment it for open parentheses and decrement it for close parentheses. If the counter is in danger of going negative (indicating an unbalanced string), we must insert an open parenthesis. Otherwise, no change is needed to the original string.

Finally, we may have unmatched open parentheses when we finish traversing the string. To deal with these, we can append a number of close parentheses corresponding to the counter value.

Since we traverse the string once and build up a new string of length at most 2*N, this algorithm has O(N) time and space complexity.

The code for this problem would look like this:

```
def get_closest_string(s):
    closest_string = []

    open_parens = 0

    for char in s:
        if char == "(":
            open_parens += 1
            closest_string.append(char)

        else:
            if not open_parens:
                closest_string += "("
            else:
                open_parens -= 1
            closest_string.append(char)

    while open_parens:
        open_parens -= 1
        closest_string.append(")")

    return ''.join(closest_string)
```