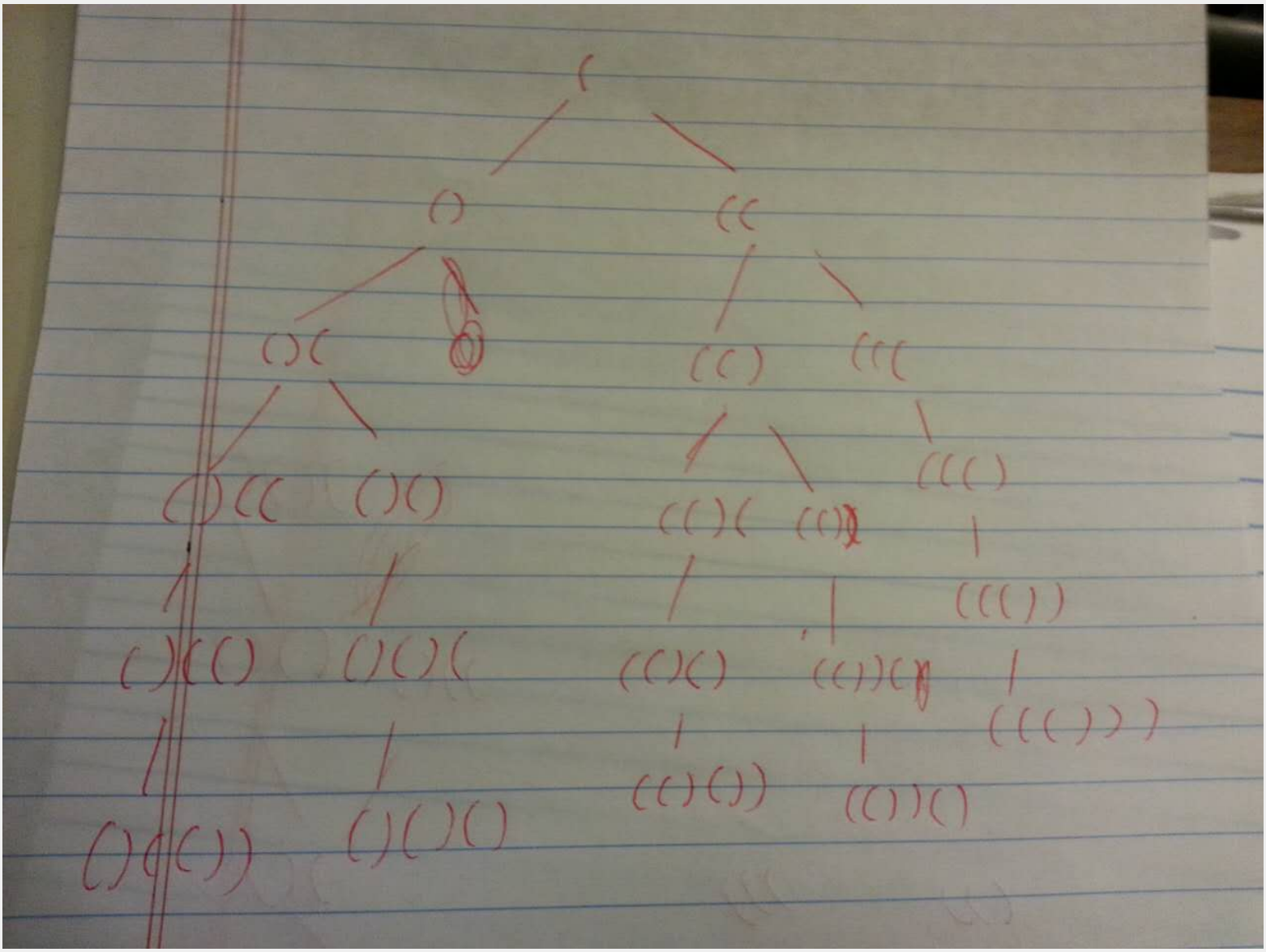
problems:

Generate Parentheses

Total Accepted: **70009** Total Submissions: **202994** Difficulty: **Medium**

> Given *n* pairs of parentheses, write a function to generate all combinations of well-formed parentheses.
>
> For example, given *n* = 3, a solution set is:
>
> `"((()))", "(()())", "(())()", "()(())", "()()()"`



This is a tree, recursion should be the first thing that comes to our mind (theoretically...though), and we can see that there should be two recursive calls: one for adding left parentheses “(” and one for add right parentheses “)”.

There two conditions that stops the recursive call: 1) No. of right parentheses is larger than the No. of left parentheses. 2) The left parentheses are more than the preset number ‘n’. One thing to note that, though, one recursive call is

stopped, we still have another one, which means that once the No. of left parentheses reaches n, we stop adding left parentheses but we still need to finish the string by adding right parentheses.

Code:

```
class Solution(object):
    def generateParenthesis(self, n):
        res = []
        cand = ''
        self.gp(0, 0, cand, res, n)
        return res

    def gp(self, left, right, cand, res, n):
        if left < right or left > n:   # when to stop the recursive calls
            return
        elif left == n and right == n:
            # if adding both parentheses are done, put the string into the result
            res.append(cand)
        else:
            # we should always start adding left parentheses first
            self.gp(left+1, right, cand + '(', res, n)

            # keep adding right parentheses
            self.gp(left, right + 1, cand + ')', res, n)
```