22. Generate Parentheses

1779 114

Description (/problems/generate-parentheses/description/) Hints (/problems/generate-parentheses/hints/) Submissions (/problems/generat

Quick Navigation View in Article (/articles/generate-parentheses/) Notes

Approach 1: Brute Force

Intuition

We can generate all 2^{2n} sequences of `'('` and `')'` characters. Then, we will check if each one is valid.

Algorithm

To generate all sequences, we use a recursion. All sequences of length `n` is just `'('` plus all sequences of length `n-1`, and then `')'` plus all sequences of length `n-1`.

To check whether a sequence is valid, we keep track of `balance`, the net number of opening brackets minus closing brackets. If it falls below zero at any time, or doesn't end in zero, the sequence is invalid - otherwise it is valid.

Java Python Copy

```java
class Solution {
    public List<String> generateParenthesis(int n) {
        List<String> combinations = new ArrayList();
        generateAll(new char[2 * n], 0, combinations);
        return combinations;
    }

    public void generateAll(char[] current, int pos, List<String> result) {
        if (pos == current.length) {
            if (valid(current))
                result.add(new String(current));
        } else {
            current[pos] = '(';
            generateAll(current, pos+1, result);
            current[pos] = ')';
            generateAll(current, pos+1, result);
        }
    }

    public boolean valid(char[] current) {
        int balance = 0;
        for (char c: current) {
            if (c == '(') balance++;
            else balance--;
            if (balance < 0) return false;
        }
        return (balance == 0);
```

Complexity Analysis

- Time Complexity : $O(2^{2n}n)$. For each of 2^{2n} sequences, we need to create and validate the sequence, which takes $O(n)$ work.
- Space Complexity : $O(2^{2n}n)$. Naively, every sequence could be valid. See Approach 3 for development of a tighter asymptotic bound.

Approach 2: Backtracking

Intuition and Algorithm

Instead of adding `'('` or `')'` every time as in Approach 1, let's only add them when we know it will remain a valid sequence. We can do this by keeping track of the number of opening and closing brackets we have placed so far.

We can start an opening bracket if we still have one (of `n`) left to place. And we can start a closing bracket if it would not exceed the number of opening brackets.

Java Python Copy

Complexity Analysis

Our complexity analysis rests on understanding how many elements there are in `generateParenthesis(n)`. This analysis is outside the scope of this article, but it turns out this is the `n`-th Catalan number $\dfrac{1}{n+1}\binom{2n}{n}$, which is bounded asymptotically by $\dfrac{4^n}{n\sqrt{n}}$.

- Time Complexity : $O(\dfrac{4^n}{\sqrt{n}})$. Each valid sequence has at most `n` steps during the backtracking procedure.
- Space Complexity : $O(\dfrac{4^n}{\sqrt{n}})$, as described above, and using $O(n)$ space to store the sequence.

Approach 3: Closure Number

Intuition

To enumerate something, generally we would like to express it as a sum of disjoint subsets that are easier to count.

Consider the *closure number* of a valid parentheses sequence `S`: the least `index >= 0` so that `S[0], S[1], ..., S[2*index+1]` is valid. Clearly, every parentheses sequence has a unique *closure number*. We can try to enumerate them individually.

Algorithm

For each closure number `c`, we know the starting and ending brackets must be at index `0` and `2*c + 1`. Then, the `2*c` elements between must be a valid sequence, plus the rest of the elements must be a valid sequence.

Java Python Copy

Complexity Analysis

- Time and Space Complexity : $O(\frac{4^n}{\sqrt{n}})$. The analysis is similar to Approach 2.

Comments: 38

Sort By

Approach 3 follows grammar rule A -> E | A(A)

Preview Post

slz250 (/slz250) ★ 15 2 days ago

Can someone explain solution 3? I'm so lost lol

(/slz250)

0 Share Reply

windliang (/windliang) ★ 80 October 10, 2018 10:36 AM

详细分析了下解法三，以及时间复杂度的来由和卡塔兰数相关，分享一下 https://leetcode.windliang.cc/leetCode-22-Generate-Parentheses.html (https://leetcode.windliang.cc/leetCode-22-Generate-Parentheses.html)

(/windliang)

0 Share Reply

itallen (/itallen) ★ 0 October 10, 2018 2:42 AM

DP solution:

(/itallen)

```
class Solution:
    def generateParenthesis(self, n):
        """
```

Read More

0 Share Reply

vishalshah3584 (/vishalshah3584) ★ 8 October 5, 2018 12:44 AM

Why can't we go with DP Fibonacci series kind of solution?

(/vishalshah3584)

Read More

0 Share Reply

SHOW 1 REPLY