114

(/user/left-peter) **left.peter (/user/left-peter)**
Reputation: ★ 156

My method is DP. First consider how to get the result f(n) from previous result f(0)...f(n-1). Actually, the result f(n) will be put an extra () pair to f(n-1). Let the "(" always at the first position, to produce a valid result, we can only put ")" in a way that there will be i pairs () inside the extra () and n - 1 - i pairs () outside the extra pair.

Let us consider an example to get clear view:

f(0): ""

f(1): "("f(0)")"

f(2): "("f(0)")"f(1), "("f(1)")"

f(3): "("f(0)")"f(2), "("f(1)")"f(1), "("f(2)")"

So f(n) = "("f(0)")"f(n-1) , "("f(1)")"f(n-2) "("f(2)")"f(n-3) ... "("f(i)")"f(n-1-i) ... "(f(n-1)")"

Below is my code:

```
public class Solution
{
    public List<String> generateParenthesis(int n)
    {
        List<List<String>> lists = new ArrayList<>();
        lists.add(Collections.singletonList(""));

        for (int i = 1; i <= n; ++i)
        {
            final List<String> list = new ArrayList<>();

            for (int j = 0; j < i; ++j)
            {
                for (final String first : lists.get(j))
                {
                    for (final String second : lists.get(i - 1 - j))
                    {
                        list.add("(" + first + ")" + second);
                    }
                }
            }

            lists.add(list);
        }

        return lists.get(lists.size() - 1);
    }
}
```

2 years ago (/topic/3474/an-iterative-method/1)

JAVA **7.4k** (/tags/java)

Log in to reply (/login) (https://leetcode.com/problems/generate-parentheses)

0

(/user/weird) ● **weird (/user/weird)**
Reputation: ★ 63

Nice! thanks.

2 years ago (/topic/3474/an-iterative-method/2)

0

(/user/jlu_chuang) ● **jlu_chuang (/user/jlu_chuang)**
Reputation: ★ 1

Nice！！！！！！！！

2 years ago (/topic/3474/an-iterative-method/3)

-8

(/user/ky512) ● **ky512 (/user/ky512)**
Reputation: ★ 21

Thanks a lot ! Your solution is perfect!!!

about a year ago (/topic/3474/an-iterative-method/4)

0

(/user/wenyan) ● **wenyan (/user/wenyan)**
Reputation: ★ 0

very good solution!

about a year ago (/topic/3474/an-iterative-method/5)

0

(/user/jingb) ● **jingb (/user/jingb)**
Reputation: ★ 0

pretty awesome

about a year ago (/topic/3474/an-iterative-method/6)

0

(/user/jackwhit3) ● **jackwhit3 (/user/jackwhit3)**
Reputation: ★ 0

I don't get this 🙁
How do you know/prove this is correct?

about a year ago (/topic/3474/an-iterative-method/7)

0

(/user/seekerwu) ● **seekerwu (/user/seekerwu)**
Reputation: ★ 0

This post is deleted!

10 months ago (/topic/3474/an-iterative-method/8)

12

(/user/w0mtea) ● **w0mTea (/user/w0mtea)**
Reputation: ★ 12

I think it's useful to prove this equation.

The equation is equivalent to the following one:

f(n) = (f(0))f(n-1) + (f(1))f(n-2) + ... + (f(n-2))f(1) + (f(n-1))f(0)

First, let f(n) to be a correct solution set when there is n pair of parentheses.
This means every combination in f(n) is a valid combination, and any combination which isn't in f(n) is not a valid combination for n.
And we can easily get the first three solution sets i.e. f(0) = {""}, f(1) = {"()"} f(2) = {"()()", "(())"}.

For any n > 2, each combination of f(n) can be divided into two parts p0 and p1.
p0 and p1 has several properties:

1. Parentheses in both p0 and p1 can match wel
2. p0 should be as short as possible but not empty. This means that p0 belongs to (f(l0-1)) where l0 is the number of pairs in p0.
   This property can be proved easily. Shortest means the first left parenthesis in this combination always matches the last right parenthesis.
   So without these two, what left is also a legal combination.

Now, let's reorganize f(n) by p0.
Put those combinations with same length of p0 into a same set, and then f(n) is divided into several subsets.
Each combination in subset s whose p0 has l0 pair of parentheses also belongs to the set (f(l0-1))f(n-l0).
So we can get f(n) belongs to (f(0))f(n-1) + (f(1))f(n-2) + ... + (f(n-2))f(1) + (f(n-1))f(0).

OK, the only thing to prove now is (f(0))f(n-1) + (f(1))f(n-2) + ... + (f(n-2))f(1) + (f(n-1))f(0) also belongs to f(n).
Notice that each combination in (f(i))f(n-1-i) is a legal combination for n, and we've declared before that each legal combination for n belongs to f(n).
So each combination in the left side of equation belongs to f(n), and the left side as a whole set belongs to f(n).

Prove complete.

10 months ago (/topic/3474/an-iterative-method/9)

0

(/user/csnerds) ● **Csnerds (/user/csnerds)**
Reputation: ★ 0

what is the time complexity for this one? O(n^4)?

10 months ago (/topic/3474/an-iterative-method/10)

0

(/user/mehere) ● **mehere (/user/mehere)**
Reputation: ★ 11

Another iterative method with complexity O(n^2)

```
        LinkedList<String> queueBracket = new LinkedList<>();
        queueBracket.add("(");

        // 0 means # of left brackets; 1 means # of right brackets
        LinkedList<List<Integer>> queueBracketNum = new LinkedList<>();
        queueBracketNum.add(Arrays.asList(new Integer[]{1, 0}));

        for (int i = 1; i <= n * 2 - 1; i++) {
            while (queueBracket.peek().length() == i) {
                String bracket = queueBracket.remove();
                List<Integer> bracketNum = queueBracketNum.remove();

                if (bracketNum.get(0) < n) {
                    queueBracket.add(bracket + "(");
                    queueBracketNum.add(Arrays.asList(new Integer[]{bracketNum.get(0) + 1,
                }

                if (bracketNum.get(0) > bracketNum.get(1) && bracketNum.get(1) < n) {
                    queueBracket.add(bracket + ")");
                    queueBracketNum.add(Arrays.asList(new Integer[]{bracketNum.get(0), bra
                }
            }
        }

        return queueBracket;
```

8 months ago (/topic/3474/an-iterative-method/11)

▲
0
▼

(/user/dragonfly9113) ● **dragonfly9113 (/user/dragonfly9113)**
Reputation: ★ 0

Thanks for the explanations!

5 months ago (/topic/3474/an-iterative-method/12)

▲
2
▼

(/user/wfxr) ● **wfxr (/user/wfxr)**
Reputation: ★ 27

Too many loops and containers. How about this:

```
vector<string> generateParenthesis(int n) {
    vector<string> result;
    if (!n) return result;

    string s(n, '(');
    s.append(n, ')');

    for (;;) {
        auto l = n, r = n;
        result.push_back(s);
        for (;;) {
            if (s.back() == ')') --r;
            else if (l < r + 2) --l;
            else break;
            s.pop_back();
            if (s.empty()) return result;
        }

        s.back()=')';
        s.append(n - (l - 1), '(');
        s.append(n - (r + 1), ')');
    };
}
```

4 months ago (/topic/3474/an-iterative-method/13)

▲ 0 ▼

(/user/neer1304) ● **neer1304 (/user/neer1304)** Reputation: ★ 0

Can you explain your approach ?

3 months ago (/topic/3474/an-iterative-method/14)

▲ 0 ▼

(/user/hongyi5) ● **hongyi5 (/user/hongyi5)** Reputation: ★ 0

I got a "Memory Limit Exceeded" using this method

19 days ago (/topic/3474/an-iterative-method/15)

JAVA 7.4k (/tags/java)

Log in to reply (/login) (https://leetcode.com/problems/generate-parentheses)