22. Generate Parentheses (/problems/generate-parentheses/)

Dec. 17, 2017 | 287.3K views

Average Rating: 4.40 (100 votes)

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:

```
[
  "((()))",
  "(()())",
  "(())()",
  "()(())",
  "()()()"
]
```

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

```python
class Solution(object):
    def generateParenthesis(self, n):
        def generate(A = []):
            if len(A) == 2*n:
                if valid(A):
                    ans.append("".join(A))
            else:
                A.append('(')
                generate(A)
                A.pop()
                A.append(')')
                generate(A)
                A.pop()

        def valid(A):
            bal = 0
            for c in A:
                if c == '(': bal += 1
                else: bal -= 1
                if bal < 0: return False
            return bal == 0

        ans = []
        generate()
        return ans
```

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 Articles > 22. Generate Parentheses Copy

```python
class Solution(object):
    def generateParenthesis(self, N):
        ans = []
        def backtrack(S = '', left = 0, right = 0):
            if len(S) == 2 * N:
                ans.append(S)
                return
            if left < N:
                backtrack(S+'(', left+1, right)
            if right < left:
                backtrack(S+')', left, right+1)

        backtrack()
        return ans
```

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 $\frac{1}{n+1}\binom{2n}{n}$, which is bounded asymptotically by $\frac{4^n}{n\sqrt{n}}$.

- Time Complexity : $O(\frac{4^n}{\sqrt{n}})$. Each valid sequence has at most `n` steps during the backtracking procedure.
- Space Complexity : $O(\frac{4^n}{\sqrt{n}})$, 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

```
class Solution(object):
    def generateParenthesis(self, N):
        if N == 0: return ['']
        ans = []
        for c in xrange(N):
            for left in self.generateParenthesis(c):
                for right in self.generateParenthesis(N-1-c):
                    ans.append('({}){}'.format(left, right))
        return ans
```

Complexity Analysis

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

Rate this article:

Previous (/articles/number-of-corner-rectangles/) Next (/articles/largest-number-at-least-twice-of-others/)

Comments: 111 Sort By

Type comment here... (Markdown is supported)

Preview Post

(/bhaveshm) bhaveshm (/bhaveshm) ★ 693 September 16, 2018 6:38 PM

How should one explain the time complexity during the interview?
There is no chance that I can come up with time complexity of this particular solution unless I have read this already or I am a mathematician.

217 Share Reply

SHOW 6 REPLIES

(/s961206)

s961206 (/s961206) ★ 655 February 12, 2019 11:44 PM

I dont understand the colsure number solution. .. Could someone please help us explain?

61 Share Reply

SHOW 2 REPLIES

(/calvinchankf)

calvinchankf (/calvinchankf) ★ 2621 February 18, 2019 2:44 AM

If an interviewer ask me about the time complexity of the backtracking approach, how should I answer?

45 Share Reply

SHOW 7 REPLIES

(/dpinto)

dpinto (/dpinto) ★ 57 August 16, 2018 9:05 PM

what does it mean outside the scope of the problem , you guys should be explaining exactly these kind of stuff.

42 Share Reply

SHOW 3 REPLIES

(/vannada)

vannada (/vannada) ★ 30 March 30, 2019 9:55 PM

Isn't the approach 2 recursion? Why is it called backtracking? We aren't getting back in any case. Can someone please explain?

27 Share Reply

SHOW 12 REPLIES

(/sherlockholo)

SherlockHolo (/sherlockholo) ★ 25 July 11, 2018 5:38 AM

I think i found a codes mistakes, at Approach 2, the java codes: the 9 lines: `if (str.length() == max * 2) {` it should be `if (cur.length() == max * 2) {` because there is no a variable's name is `str` , the variable should be `cur`

25 Share Reply

SHOW 1 REPLY

(/panwu5588)

panwu5588 (/panwu5588) ★ 43 April 7, 2018 12:38 AM

BAD test cases. The order should not matter.
For example, the following should be the same:
["(())","()()"]
["()()","(())"]

25 Share Reply

SHOW 5 REPLIES

alexishe (/alexishe) ★ 207 September 24, 2018 4:17 PM

Articles › 22. Generate Parentheses

My simple Java solution:

(/alexishe)

```
class Solution {
    public List<String> generateParenthesis(int n) {
        if(n==1) return new ArrayList<String>(Arrays.asList("()"));
```

Read More

22 Share Reply

SHOW 3 REPLIES

YichaoCai (/yichaocai) ★ 14 August 21, 2018 6:25 AM

For Approach 3 in Python3, I understand it in this way. Please correct me if I am wrong!
If we enumerate all the possibilities of N = 3. We can get:

(/yichaocai)

```
[
'((()))'
```

Read More

14 Share Reply

ohcrapitspanic (/ohcrapitspanic) ★ 11 February 20, 2019 5:54 PM

An improvement to time in approach 3 involves using dynamic programming to store calculated values to avoid recalculation.

(/ohcrapitspanic)

11 Share Reply

SHOW 2 REPLIES

‹ 1 2 3 4 5 6 ... 11 12 ›