问题描述

22. 括号生成

难度 中等 932

数字 *n* 代表生成括号的对数，请你设计一个函数，用于能够生成所有可能的并且 **有效的** 括号组合。

示例：

```
输入：n = 3
输出：[
       "((()))",
       "(()())",
       "(())()",
       "()(())",
       "()()()"
     ]
```

通过次数 112,765 | 提交次数 149,964

在真实的面试中遇到过这道题?

参看网址：https://leetcode-cn.com/problems/generate-parentheses/

解题思路

从最开始就想用递归作，但是一直找不到递归出口。最后最认同回溯法（DFS）的方法；

//本来想写一个只有一个return的递归，但是发现这条肯定是实现不了的。不禁感叹自己的愚笨，感叹函数在不是所有分支都有返回时产生的那个警告的重要性

回溯法想的是用一个string 变量暂存当前括号搭配，考虑当前剩余需要搭配的左括号和右括号数进行添加"(" 或者 ")" 最后n对括号都使用完毕时，回溯，加入vector；

回溯像是在画一棵树，这棵树每次有两个选择。

- 1、在L<N的时候可以直接添加左括号
- 2、在L<R的时候可以直接添加右括号

在这样的情况下，为当前括号组合cur 进行添加 (或者) 然后每种情况进行回溯：（消除刚刚添加的字符）.

也可以直接在迭代时，cur传入时不进行引用，直接传入cur+‘(’作为参数。

代码实现

```cpp
#include<iostream>
#include<stdio.h>
#include<vector>
#include<queue>
#include<string>
using namespace std;

class Solution {
public:
    vector<string> str;
    vector<string> generateParenthesis(int n) {
        string cur = "";
        fun(0, 0, n, cur);
        return str;
    }

    void fun(int L, int R, int n, string &cur)
    {
        //先确定回溯出口
        if (L == n && R == n)
        {
            str.push_back(cur);
            return;
        }
        if (L < n)
        {
            cur.push_back('(');
            fun(L + 1, R, n, cur);          //跳出该函数时，必定已经将符合要求的string push进去了。
            cur.pop_back();                 //将当前添加的字符清除，以应对另一种情况
        }
        if (L > R)
        {
            cur .push_back( ')');
            fun(L, R + 1, n, cur);
            cur.pop_back();
        }
    }
};


int main()
{

    Solution S;
    S.generateParenthesis(3);
    int i = 0;
    for (auto iter = S.str.cbegin(); iter != S.str.cend(); iter++)  //用迭代器进行遍历
        cout << (*iter) << endl;                                    //cbegin与begin的区别在于,cbegin返回的是const类型的迭代器，只可r不可W
    return 0;
}
```

DFS：其实和上面一模一样。

```
class Solution {
public:
    vector<string> str;
    vector<string> generateParenthesis(int n) {
        dfs(0, 0, n, "");
        return str;
    }
    void dfs(int l, int r, int n, string cur)
    {
        if (l == n && r == n)
            str.push_back(cur);
        if (l < n)
            dfs(l + 1, r, n, cur+'(');
        if (l > r)
            dfs(l, r + 1, n, cur+')');
    }
};
```

相同的思路：Python实现如下

```
# coding=utf-8
from typing import List


class Solution:
    def generateParenthesis(self, n) -> List[str]:
        self.ans = []
        self.Iter('', n, n)
        return self.ans

    def Iter(self, str1, r, l):
        if r == 0 and l == 0:
            self.ans.append(str1)
        if l > 0:
            self.Iter(str1+'(',r,l-1)
        if l<r:
            self.Iter(str1+')',r-1,l)


if __name__ == '__main__':
    S = Solution()
    print(S.generateParenthesis(3))
```