一、题目说明

这个题目是22. Generate Parentheses，简单来说，输入一个数字n，输出n对匹配的小括号。

简单考虑了一下，n=0，输出"";n=1，输出“（）”；n=2，输出“()()”,"(())"...

二、我的解法

我考虑了一下，基本上2种思路，其1是暴力破解法，就是列出所有可能情况，然后判断哪些是匹配的。

汗颜的是，我没做出来。找到的答案：

```
#include<iostream>
#include<vector>
#include<string>
using namespace std;

class Solution {
public:
    //Brute Force
    vector<string> generateParenthesis(int n) {
        int num = 2*n;
        string cur(num,'0');
        vector<string>  ans;
        //1左移num位得到2的2*n次方
        for (int v = (1<<num)-1; v > 0; v--) {
            //生成 所有的0、1序列
//            for (int i = 0; i < num; i++) {
//              if(v & 1<<i) cur[i] = '(';
//              else     cur[i]=')';
//            }
//            ans.push_back(cur);

            int cnt = 0;
            for (int i = 0; i < num; i++) {
                if (v&1<<i) { cur[i]='('; cnt++; }
                else        { cur[i]=')'; cnt--; }
                if (cnt < 0 || cnt > n) break;
            }
            if (cnt == 0) ans.push_back(cur);
        }
        return ans;
    };

    void test(int n){
        string cur(n,'0');
        for(int v=(1<<n)-1;v>=0;v--){
            for(int j=0;j<n;j++){
                if(v & 1<<j) cur[j] = '1';
                else cur[j] = '0';
            }
            cout<<cur<<"\n";
        }
    }
};
```

```
int main(){
    Solution s;
//  s.test(2);
    vector<string> r = s.generateParenthesis(1);
    cout<<r.size()<<endl;
    for(int i=0;i<r.size();i++){
        cout<<r[i]<<endl;
    }
    return 0;
}
```

其2是回溯法，汗颜的是，我也没做出来，网上找到的答案：

```
#include<iostream>
#include<vector>
using namespace std;

class Solution {
public:

    //Backtracking
    vector<string> generateParenthesis(int n)
    {
        if(!n) return vector<string>(1, "");
        if(n == 1) return vector<string>(1, "()");

        vector<string> result;

        for(int i = n-1; i >=0; i--)
        {
            vector<string> inner = generateParenthesis(i);
            vector<string> outer = generateParenthesis(n-i-1);

            for(int i=0; i < inner.size(); i++)
                for(int j=0; j < outer.size(); j++)
                    result.push_back( "(" + inner[i] + ")" + outer[j] );
        }

        return result;
    }
};
int main(){
    Solution s;
    vector<string> r = s.generateParenthesis(3);
    for(int i=0;i<r.size();i++){
        cout<<r[i]<<endl;
    }
    return 0;
}
```

三、优化措施

另外，学过离散数学的，还有一种方法就是“闭包”。似曾相识，直接上答案：

```
class Solution {
```

```
public:

    //Imporved Closure Number
    vector<string> generateParenthesis(int n) {
        map<int, vector<string>> map;
        map[0].push_back("");
        for(int i = 1; i <= n; i++)
            for(int c = 0; c < i; c++)
                for(string left: map[c])
                    for(string right: map[i - 1 - c])
                        map[i].push_back("(" + left + ")" + right);
        return map[n];
    }
};
```