一、题目说明

题目是32. Longest Valid Parentheses，求最大匹配的括号长度。题目的难度是Hard

二、我的做题方法

简单理解了一下，用栈就可以实现。实际上是我考虑简单了，经过5次提交终于正确了。

性能如下：

```
Runtime: 8 ms, faster than 61.76% of C++ online submissions for Longest Valid
Parentheses.
Memory Usage: 9.8 MB, less than 10.71% of C++ online submissions for Longest
Valid Parentheses.
```

代码如下：

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

class Solution {
public:
    int longestValidParentheses(string s){
        if(s.size()<=1) return 0;

        stack<char> st;
        vector<int> result;

        for(int t=0;t<s.size();t++){
            if(s[t]=='('){
                st.push(s[t]);
                result.push_back(0);
            }else if(s[t]==')'){
                if(st.empty()){
                    result.push_back(0);
                }else if(st.top() == '('){
                    st.pop();
                    if(result.back()>0){
                        int s = result.size()-1;
                        //合并所有非 0
                        while(s>0 && result[s]>0){
                            s--;
                        }
                        if(s < result.size()-2){
                            int sum = 0,currSize = result.size();
                            for(int j=s+1;j<currSize-1;j++){
                                sum += result.back();
                                result.pop_back();
                            }
                            result.back() += sum;
                        }
```

```
                        int tmp = result.back() + 2;
                        if(result.size()>0){
                            result.pop_back();
                        }
                        result.back() += tmp;

                    }else{
                        result.back() += 2;
                    }

                }
            }
        }


        int max=0,curMax=0;
        for(int i=0;i<result.size();i++){
            curMax = 0;
            int t = i;
            while(t<result.size() && result[t]>0){
                curMax += result[t];
                t++;
            }
            max = curMax>max ? curMax : max;
        }

        return max;
    }
};
int main(){
    Solution s;
    cout<<(2==s.longestValidParentheses("(()"))<<endl;
    cout<<(4==s.longestValidParentheses(")()())"))<<endl;
    cout<<(2==s.longestValidParentheses("()(()"))<<endl;
    cout<<(6==s.longestValidParentheses("()(())"))<<endl;
    cout<<(4==s.longestValidParentheses("(()()"))<<endl;
    cout<<(4==s.longestValidParentheses("(())"))<<endl;
    cout<<(6==s.longestValidParentheses("(()())"))<<endl;
    cout<<(10==s.longestValidParentheses(")(())(()())("))<<endl;
    cout<<(8==s.longestValidParentheses("((()))())"))<<endl;
    return 0;
}
```

三、优化措施

题解给了4种方法，这4种方法都比较好理解，我上述实现方法属于第3种“栈”，只不过把问题搞复杂了。惭愧！！！

1.暴力法，枚举所有子串，判断合法性，求出最长。

2.动态规划，这一直是我的软肋。

用数组dp表示，其中第 i 个元素表示以下标为 i 的字符结尾的最长有效子字符串的长度。

3.栈

4.不需要额外空间：这种方法非常巧妙，非常考验智商！理解起来不难！

用left和right分别统计左括号和右括号数量，先从左到右统计，遇到左括号left++，遇到右括号right++，如果right==left求最大值，如果left<right则将left=right=0；再从右到左来一遍。