一、题目说明

题目647. Palindromic Substrings，给定一个字符串，计算所有子串中回文的数量。难度是Medium!

二、我的解答

这个题目，能想到的是brute force方法：

```
class Solution{
    public:
        int countSubstrings(string s){
            int len = s.length();
            if(len<1) return 0;
            else if(len<2) return 1;
            int sum = 0;
            for(int i=0;i<len;i++){
                sum += dfs(s,i);
            }
            return sum;
        }

        int dfs(string&s,int start){
            int len = s.size();
            if(start == len-1){
                return 1;
            }

            int sum = 0;
            for(int i=start;i<len;i++){
                if(checkPalindromic(s.substr(start,i-start+1))){
                    sum++;
                }
            }
            return sum;
        }

        bool checkPalindromic(string s){
            int len = s.size();
            int mid = len /2;
            for(int i=0;i<=mid;i++){
                if(s[i]!=s[len-i-1]){
                    return false;
                }
            }
            return true;
        }
};
```

性能如下：

```
Runtime: 492 ms, faster than 7.13% of C++ online submissions for Palindromic
Substrings.
Memory Usage: 458.4 MB, less than 8.00% of C++ online submissions for
Palindromic Substrings.
```

三、优化措施

可以用“中心拓展法”，“动态规划法”。

```
class Solution{
    public:
        //中心拓展法
        int countSubstrings(string s){
            res = 0;
            if(s.size()==0) return 0;
            for(int i=0;i<s.size();i++){
                expandAroundCenter(s,i,i);//以i个元素为中心扩展
                expandAroundCenter(s,i,i+1);// 以i、i+1为中心扩展
            }
            return res;
        }
        void expandAroundCenter(string s,int begin,int end){
            while(begin>=0 && end<s.size() && s[begin]==s[end]){
                begin--;
                end++;
                res++;
            }
        }
    private:
        int res;
};
```

性能：

```
Runtime: 8 ms, faster than 68.00% of C++ online submissions for Palindromic
Substrings.
Memory Usage: 15.6 MB, less than 12.00% of C++ online submissions for
Palindromic Substrings.
```

用dp的代码及性能如下：

```
class Solution{
    public:
        //dp[i][j]表示从第i个元素到第j个元素是否是回文
        //if(dp(i+1)(j-1)==true&&s[i]=s[j]) dp(i)(j)=true
        //从相邻的元素出发
        int countSubstrings(string s){
            int res = 0;
            int len = s.size();
            vector<vector<bool>> dp(len,vector<bool>(len));

            for(int j=0;j<len;j++){
                for(int i=j;i>=0;i--){
                    if(s[i]==s[j] && ((j-i<2)|| dp[i+1][j-1])){
                        dp[i][j] = true;
                        res ++;
                    }
                }
            }
```

```
            return res;
        }
};
```

```
Runtime: 20 ms, faster than 42.04% of C++ online submissions for Palindromic Substrings.
Memory Usage: 9.9 MB, less than 48.00% of C++ online submissions for Palindromic Substrings.
```