5. Longest Palindromic Substring

题目描述：https://leetcode.com/problems/longest-palindromic-substring/

> 给定一个字符串，求其最长回文子串
> 注意不能用reverse再判断substr的方法，是错的
> 例如：abcdefgmnpfedcba -> abcdefpnmedcba
> substr最长为abcedf但是并不是回文的

解题思路：

方法1： 动态规划
dp[i][j]的意思是从i到j是不是回文的,如果是则为true，否则为false，则动态转移方程如下：
长度为1和2的都初始化
dp[i][j] = dp[i-1][j+1] , a[i] == a[j] j-i+1 > 2
dp[i][j] = false
方法2： 中心扩展法
以i为中心向两边扩展
注意：偶数长度的从字母之间的空位开始扩展

代码V1：

```
class Solution {
public:
    string longestPalindrome(string s) {
        vector<vector<bool> > f(s.size(), vector<bool>(s.size(), false));
        for(int i = 0; i < s.size(); i++){
            f[i][i] = true;
        }
        for(int i = 1; i < s.size(); i++){
            if(s.at(i) == s.at(i-1))
                f[i-1][i] = true;
        }
        int l = 3;
        while(l <= s.size()){
            for(int i = 0; i+l-1 < s.size(); i++){
                int j = i+l-1;
                if(s.at(i) == s.at(j)){
                    f[i][j] = f[i+1][j-1];
                }
            }
            l++;
        }
        l = s.size();
        while(l >= 0){
            for(int i = 0; i+l-1 < s.size(); i++){
                int j = i+l-1;
                if(f[i][j]){
                    return s.substr(i, l);
                }
            }
            l--;
        }
        return "";
    }
};
```

代码**V2**:

```
class Solution {
public:
    string longestPalindrome(string s) {
        int maxLen = 1;
        int startLoc = 0, endLoc = 0;
        int d1 = 0, d2 = 0;
        for(int k = 0; k < s.size()*2; k++){
            int i = k / 2;
            if(k % 2 == 0){
                d1 = 0;
                d2 = d1;
            }
            if(k % 2 == 1){
                d1 = 0;
                d2 = 1;
            }
            while(i-d1 >= 0 && i+d2 < s.size()){
                if(s.at(i-d1) != s.at(i+d2)){
                    break;
                }
                d1++; d2++;
            }
            if(maxLen < d2+d1-1){
                maxLen = d2+d1-1;
                startLoc = i-d1+1;
                endLoc = i+d2-1;
            }
        }
        return s.substr(startLoc, endLoc-startLoc+1);
    }
};
```