题目：

Given a string s, find the longest palindromic subsequence's length in s. You may assume that the maximum length of s is 1000.

Example 1:

Input:

```
"bbbab"
```

Output:

```
4
```

One possible longest palindromic subsequence is "bbbb".

Example 2:

Input:

```
"cbbd"
```

Output:

```
2
```

One possible longest palindromic subsequence is "bb".

Subscribe to see which companies asked this question.

1.时间：O（2^N）；空间：O（N）　　-->>超时

```
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        if (s.empty()) return 0;
        int maxLen = 0;
        dfs(s, 0, "", maxLen);
        return maxLen;
    }
private:
    inline bool is_palindromic(const std::string& str){
        if (str.empty()) return false;
        const int len = str.size();
        for (int i = 0; i < len / 2; ++i){
            if (str[i] != str[len - 1 - i]) return false;
        }
        return true;
    }
    void dfs(const std::string& str, const int index, std::string sub_sequence, int&
maxLen){
        if (is_palindromic(sub_sequence))
            maxLen = std::max(maxLen, (int)sub_sequence.size());
```

```
        if (index == str.size()) return;
        dfs(str, index + 1, sub_sequence + str[index], maxLen);
        dfs(str, index + 1, sub_sequence, maxLen);
    }};
```

2.时间：O（N^2）;空间：O（N^2）

```
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        if (s.empty()) return 0;
        std::vector<std::vector<int>> dp(s.size(), std::vector<int>(s.size(), 0));
        for (int i = 0; i < dp.size(); ++i) dp[i][i] = 1;
        for (int k = 1; k < s.size(); ++k){
            for (int i = k - 1; i >= 0; --i){
                if (s[i] == s[k]){
                    dp[i][k] = i + 1 < k ? 2 + dp[i + 1][k - 1] : 2;
                } else{
                    dp[i][k] = std::max(dp[i + 1][k], dp[i][k - 1]);
                }
            }
        }
        return dp[0][s.size() - 1];
    }};
```