647 回文子串

Label：字符串 动态规划
给定一个字符串，你的任务是计算这个字符串中有多少个回文子串。
具有不同开始位置或结束位置的子串，即使是由相同的字符组成，也会被视作不同的子串。

输入："abc"
输出：3
解释：三个回文子串："a", "b", "c"
输入："aaa"
输出：6
解释：6个回文子串："a", "a", "a", "aa", "aa", "aaa"
输入的字符串长度不会超过 1000 。

- 暴力法

```
class Solution {
    public int countSubstrings(String s) {
        int count = 0;
        List<String> list = new ArrayList<>();
        for (int i = 0; i < s.length(); i++) {
            for (int j = i; j < s.length(); j++) {
                list.add(s.substring(i,j+1));
            }
        }

        for (String str: list) {
            if (isP(str)) count++;
        }
        return count;
    }

    private boolean isP(String str) {
        if (str.length()==1) return true;

        StringBuilder sb = new StringBuilder(str);
        String strRe = sb.reverse().toString();

        for (int i = 0; i < str.length(); i++)
            if (str.charAt(i) != strRe.charAt(i)) return false;

        return true;
    }
}
```

- 暴力法 优化一下

```
class Solution {
    public int countSubstrings(String s) {
        int count = 0;
        List<String> list = new ArrayList<>();
        for (int i = 0; i < s.length(); i++) {
            for (int j = i; j < s.length(); j++) {
                list.add(s.substring(i,j+1));
            }
        }

        for (String str: list) {
            if (isP(str)) count++;
        }

        return count;
    }

    private boolean isP(String str) {   // 换一种判断回文的方式
        if (str.length()==1) return true;

        int left = 0;
        int right = str.length()-1;
        while (left < right) {
            if (str.charAt(left++) != str.charAt(right--)) return false;
        }

        return true;
    }
}
```

- 从左到右，从中间到两边 （中心扩展法的变体，更好理解一些）

```
class Solution {
    int num = 0;
    public int countSubstrings(String s) {
        for (int i = 0; i < s.length(); i++){  // 不判断，两个都执行一下，反正会快速失
败
            count(s, i, i); // 回文串长度为奇数
            count(s, i, i+1); //回文串长度为偶数
        }
        return num;
    }

    public void count(String s, int start, int end) {
        while(start >= 0 && end < s.length() && s.charAt(start) ==
s.charAt(end)){
            num++;
            start--;
            end++;
        }
    }

}
```

- 动态规划

```java
class Solution {
    public int countSubstrings(String s) {
        int count = 0;
        int n = s.length();
        boolean[][] dp = new boolean[n][n];
        for(int i = 0; i < s.length(); i++){
            for(int j = 0; j <= i; j++){
                // 只要首尾相同时长度是3或3以内或者减去首尾的dp[i-1][j+1]还是真（即以j+1为首，i-1为尾字符串还是回文串
                if(s.charAt(i) == s.charAt(j) && (i - j <= 2 || dp[i-1][j+1])){
                    dp[i][j] = true;
                    count++;
                }
            }
        }
        return count;
    }
}
```