正则表达式匹配

请实现一个函数用来匹配包含'.'和'*'的正则表达式。模式中的字符'.'表示任意一个字符，而'*'表示它前面的字符可以出现任意次（含0次）。在本题中，匹配是指字符串的所有字符匹配整个模式。例如，字符串"aaa"与模式"a.a"和"ab*ac*a"匹配，但与"aa.a"和"ab*a"均不匹配。

示例 1:

输入:
s = "aa"
p = "a"
输出:false
解释:"a" 无法匹配 "aa" 整个字符串。

```
/*class Solution {
public:
    bool isMatch(string s, string p) {

    }
};*/

class Solution {
public:
    bool isMatch(string s, string p) {
        //定义 dp 数组 dp[sLen+1][pLen+1]。dp[i][j]表示：s 中长度为 i 的部分可以匹配上 p 中长度为 j 的部分  sLen 和 pLen 均表示长度大小
        int sLen = s.length(), pLen = p.length();
        //“s 中长度为 i 的部分”，指的是 s[0]到 s[i-1]，因为 i、j 是表示长度的，所以具体到哪一位需要 -1
        //默认设置 dp 初始化为 false
        vector<vector<bool> > dp(sLen + 1, vector<bool>(pLen + 1, false));
        //先全部初始化为 false，然后让 dp[0][0]为 true（s 和 p 都为空时必然是 true）；
        //当 s 为空，p 满足一定条件（比如 p = "a*b*c*"）时，让 dp[0][i]为 true；
        //当 s 不为空，p 为空时，必然为 false。此情况不必再处理，因为一开始就初始化了整个 dp 为 false 了；
        //当 s 不为空，p 也不为空时，需要开始执行动态规划过程。

        dp[0][0] = true;
        /*
        s 和 p 都从 1 开始遍历（因为它们为 0 的情况都已经处理过了）。p 中长度为 j 的元素（即 p[j-1]）只有可能会是：正常字符、'.' 或 '*'。此时开始分情况讨论：
        如果 p[j-1]是正常字符且等于 s[i-1]，或者 p[j-1]是'.'，判断 dp[i][j]需要看的是它们前面的全部是否为 true，即 dp[i][j] = dp[i - 1][j - 1]。
```

如果 p[j-1]是''，那么我们需要根据 p[j-2]的值来分情况讨论，因为''的作用是让其前一位字符（即 p[i-2]）出现任意次（0 次、1 次或多次）：

假如 p[j-2]是一个正常字符而且与 s[i-1]不相等，且 p[j-2]不等于'.'，比如 s："b"与 p："a*"，我们只能让'*'的作用是“使 a 出现 0 次”，从而 dp[i][j]取决于 dp[i][j-2]；

否则假如 p[j-2]是一个正常字符且与 s[i-1]相等，或者 p[j-2]等于'.'，那么情况有三种：'*'让 p[j-2]出现 0 次、1 次或多次：

0 次：比如说 s："a"与 p："aa*“。即使 s[i-1]与 p[j-2]相等，但是 s[i-1]早就与 p[j-3]相等了（格林：在你来之前我们就是总冠军了），那么'*'也只能忍痛让 p[j-2]出现 0 次。

1 次：比如说 s："a"与 p："a*"。这时候'*'只需要让其前面的"a"出现一次，一切就会是刚刚好（詹姆斯：我一人一城，带你们成为总冠军）。

多次：比如说 s："aabb“与 p："aab*"。这时候只需要判断 s 的"aab"与 p 的"aab*”是否匹配，因为如果匹配的话，即使 s 再多加一个 b 也没事，因为 p 中的"*"可以保证 b 的量足够（C 罗：我比你少几个金球奖我就要再拿几个金球奖！）。

假如 p[j-1]是一个正常字符且不等于 s[i-1]，dp[i][j]必为 false。

*/

```
        for(int i=1; i<=pLen; ++i)
        {
            if(i >= 2 && p[i - 1] == '*' && dp[0][i - 2] == true) // 记得加 i >= 2 的判断
                dp[0][i] = true;
        }
        for(int i=1; i<=sLen; ++i)
        {
            for(int j=1; j<=pLen; ++j)
            {
                if(s[i - 1] == p[j - 1] || p[j - 1] == '.')
                    dp[i][j] = dp[i - 1][j - 1];
                else if(p[j - 1] == '*' && j >= 2) // 记得加 j >= 2 的判断，否则 s = aa, p = *a 过不去测试
                {
                    if(p[j - 2] != s[i - 1] && p[j - 2] != '.')
                        dp[i][j] = dp[i][j - 2];
                    else
                        dp[i][j] = dp[i][j - 2] || dp[i][j - 1] || dp[i - 1][j];
                }
                else dp[i][j] = false;
            }
        }
        return dp[sLen][pLen];
    }
```

```
};
```