正则表达式匹配

给你一个字符串 s 和一个字符规律 p，请你来实现一个支持 '.' 和 '*' 的正则表达式匹配。

- '.' 匹配任意单个字符
- '*' 匹配零个或多个前面的那一个元素

所谓匹配，是要涵盖 **整个** 字符串 s 的，而不是部分字符串。

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。

示例 2:

输入：s = "aa" p = "a*"

输出：true

解释：因为 '*' 代表可以匹配零个或多个前面的那一个元素，在这里前面的元素就是 'a'。因此，字符串 "aa" 可被视为 'a' 重复了一次。

示例 3：

输入：s = "ab" p = ".*"

输出：true

解释：".*" 表示可匹配零个或多个（'*'）任意字符（'.'）。

示例 4：

输入：s = "aab" p = "c*a*b"

输出：true

解释：因为 '*' 表示零个或多个，这里 'c' 为 0 个，'a' 被重复一次。因此可以匹配字符串 "aab"。

示例 5：

输入：s = "mississippi" p = "mis*is*p*."

输出：false

		A	*	B	.	A	*	B
	T	F	T	F	F	F	F	F
A	F	T	T	F	F	F	F	F
A	F	F	T	F	F	F	F	F
A	F	F	T	F	F	F	F	F
B	F	F	F	T	F	F	F	F
C	F	F	F	F	T	F	T	F
A	F	F	F	F	F	T	T	F
A	F	F	F	F	F	F	T	F
B	F	F	F	F	F	F	F	T

FORMULA

```
const p = pattern.charAt(col - 1);
const prev = pattern.charAt(col - 2);

if (p === '.' || p === t) {
  table[row][col] = table[row - 1][col - 1];
} else if (p === '*') {
  if (table[row][col - 2] === true) {
    table[row][col] = true
  } else if (prev === '.' || prev === t) {
    table[row][col] = table[row - 1][col];
  }
} else {
  table[row][col] = false;
}
```

CLOSE

```
class Solution {
public:
    bool isMatch(string s, string p) {
        vector<vector<bool>> table(s.size()+1,vector<bool>(p.size()+1));
        table[0][0]=true;

        for (int col=1; col<table[0].size(); col++) {
            char ch = p[col-1];
            if (col > 1) {
                if (ch == '*') {
                    table[0][col] = table[0][col-2];
```

```
                } else {
                    table[0][col] = false;
                }
            } else {
                if (ch == '*') {
                    table[0][col] = true;
                }
            }
        }

        for (int row=1; row<table.size(); row++) {
            char ch1 = s[row-1];
            for (int col=1; col<table[row].size(); col++) {
                char ch2 = p[col-1];
                if (ch1==ch2 || ch2 == '.') {
                    table[row][col] = table[row-1][col-1];
                } else if (ch2 == '*') {
                    if(col > 1) {
                        if (table[row][col-2]) {
                            table[row][col] = true; // * 前面的字符出现 0 次
                        } else {
                            char prev = p[col-2];
                            if (prev== ch1 || prev == '.') {
                                table[row][col] = table[row - 1][col]; // *
 前面的字符出现多次
                            }
                        }

                    }
                }
            }
        }

    vector<bool>lastRow= table[table.size()-1];
    return lastRow[lastRow.size()-1];
    }
};
```