10. Regular Expression Matching

Hard 4294 704 Add to List Share

Given an input string (`s`) and a pattern (`p`), implement regular expression matching with support for `'.'` and `'*'`.

```
'.' Matches any single character.
'*' Matches zero or more of the preceding element.
```

The matching should cover the **entire** input string (not partial).

Note:

- `s` could be empty and contains only lowercase letters `a-z`.
- `p` could be empty and contains only lowercase letters `a-z`, and characters like `.` or `*`.

Example 1:

```
Input:
s = "aa"
p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
```

Example 2:

```
Input:
s = "aa"
p = "a*"
Output: true
Explanation: '*' means zero or more of the preceding element, 'a'. Therefore, by
repeating 'a' once, it becomes "aa".
```

```java
class Solution {
    public boolean isMatch(String s1, String s2) {
        boolean isMatched = false;

        int row = s1.length() + 1;
        int col = s2.length() + 1;

        String[][] table2d = new String[row][col];
        table2d[0][0] = "T";

        // initialize each row with false value
        for (int i = 1; i < table2d.length; i++) {
            table2d[i][0] = "F";
        }
        // initialize each column with false value
        for (int i = 1; i < table2d[0].length; i++) {
            table2d[0][i] = "F";
        }
        for (int i = 1; i < table2d[0].length; i++) {
            if (s2.charAt(i - 1) == '*') {
                table2d[0][i] = table2d[0][i - 2];
            }
```

Your previous code was restored from your local storage. Reset to default

Testcase Run Code Result Debugger

Accepted Runtime: 0 ms

Your input	"aa" "a"
Output	false
Expected	false

Diff