Leetcode problem number 10

10. Regular Expression Matching

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

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

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

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".

Example 3:

Input: s = "ab", p = ".*"

Output: true

Explanation: ".*" means "zero or more (*) of any character (.)".

Constraints:

- 1 <= s.length <= 20
- 1 <= p.length <= 20
- s contains only lowercase English letters.
- p contains only lowercase English letters, '.', and '*'.
- It is guaranteed for each appearance of the character '*', there will be a previous valid character to match.

Explantaion

यह **Regular Expression Matching** की problem है।

हमें दिया गया है:

- एक **string s**
- एक **pattern p**

हमें check करना है कि **पूरा string s, पूरा pattern p से match करता है या नहीं**

👉 **Partial match allowed नहीं है**

🔹 **Pattern में दो special characters हैं**

1️⃣ **. (dot)**

- यह **किसी भी एक single character** से match करता है

Example:

s = "a"

p = "."

✓ Match (true)

2️⃣ *** (star)**

- यह अपने **previous character** को
  0 या उससे ज्यादा बार repeat कर सकता है

Example:

p = "a*"

➞ "" (0 बार a)

➞ "a"

➞ "aa"

➞ "aaa"

🔹 **Important Rule**

- Match **पूरे string पर होना चाहिए**, बीच में छोड़कर नहीं

📌 **Examples Explanation**

✅ **Example 1**

s = "aa"

p = "a"

❌ false

क्यों?

Pattern "a" सिर्फ **एक 'a'** match करता है

लेकिन string में **दो 'a'** हैं

पूरा string match नहीं हुआ

✅ **Example 2**

s = "aa"

p = "a*"

✓ true

क्यों?

a* का मतलब = 'a' को कितनी भी बार repeat कर सकते हैं

तो "aa" match हो गया

✅ **Example 3**

s = "ab"

p = ".*"

✓ true

क्यों?

- . → कोई भी character
- * → कितनी भी बार

.* का मतलब = **कोई भी string**

इसलिए "ab" match हो गया

🧠 **Problem आसान भाषा में**

👉 तुम्हें यह check करना है कि

pattern p पूरे string s को पूरी तरह से match करता है या नहीं,

जहाँ:

- . = कोई भी 1 character
- * = पिछले character की 0 या ज्यादा repetitions

Code

Java

```
enum Result {
    TRUE, FALSE
}

class Solution {
    Result[][] memo;

    public boolean isMatch(String text, String pattern) {
        memo = new Result[text.length() + 1][pattern.length() + 1];
        return dp(0, 0, text, pattern);
    }

    public boolean dp(int i, int j, String text, String pattern) {
        if (memo[i][j] != null) {
            return memo[i][j] == Result.TRUE;
        }
        boolean ans;
        if (j == pattern.length()){
            ans = i == text.length();
        } else{
            boolean first_match = (i < text.length() &&
                                   (pattern.charAt(j) == text.charAt(i) ||
                                    pattern.charAt(j) == '.'));

            if (j + 1 < pattern.length() && pattern.charAt(j+1) == '*'){
                ans = (dp(i, j+2, text, pattern) ||
                       first_match && dp(i+1, j, text, pattern));
            } else {
                ans = first_match && dp(i+1, j+1, text, pattern);
            }
        }
        memo[i][j] = ans ? Result.TRUE : Result.FALSE;
```

```
        return ans;
    }
}
```

C++

```
class Solution {
public:
    bool isMatch(string s, string p) {
        int n = s.length(), m = p.length();
        bool dp[n+1][m+1];
        memset(dp, false, sizeof(dp));
        dp[0][0] = true;

        for(int i=0; i<=n; i++){
            for(int j=1; j<=m; j++){
                if(p[j-1] == '*'){
                    dp[i][j] = dp[i][j-2] || (i > 0 && (s[i-1] == p[j-2] || p[j-2] == '.') && dp[i-1][j]);
                }
                else{
                    dp[i][j] = i > 0 && dp[i-1][j-1] && (s[i-1] == p[j-1] || p[j-1] == '.');
                }
            }
        }

        return dp[n][m];
    }
};
```

Python

```
class Solution(object):
    def isMatch(self, s, p):
        """
```

```
:type s: str
:type p: str
:rtype: bool
"""
m, n = len(s), len(p)
dp = [[False] * (n + 1) for _ in range(m + 1)]
dp[0][0] = True

for j in range(1, n + 1):
    if p[j - 1] == '*':
        dp[0][j] = dp[0][j - 2]

for i in range(1, m + 1):
    for j in range(1, n + 1):
        if p[j - 1] in {s[i - 1], '.'}:
            dp[i][j] = dp[i - 1][j - 1]
        elif p[j - 1] == '*':
            dp[i][j] = dp[i][j - 2]
            if p[j - 2] in {s[i - 1], '.'}:
                dp[i][j] = dp[i][j] or dp[i - 1][j]

return dp[m][n]
```