10. Regular Expression Matching

Add to List

Question | Editorial Solution | My Submissions (/problems/regular-expression-matching/submissions/)

Total Accepted: **110524** Total Submissions: **473062** Difficulty: **Hard** Contributors: **Admin**

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).

The function prototype should be:
bool isMatch(const char *s, const char *p)

Some examples:
isMatch("aa","a") → false
isMatch("aa","aa") → true
isMatch("aaa","aa") → false
isMatch("aa", "a*") → true
isMatch("aa", ".*") → true
isMatch("ab", ".*") → true
isMatch("aab", "c*a*b") → true
```

Notes

Hide Company Tags | Google (/company/google/) | Uber (/company/uber/) | Airbnb (/company/airbnb/) | Facebook (/company/facebook/) | Twitter (/company/twitter/)

Hide Tags | Dynamic Programming (/tag/dynamic-programming/) | Backtracking (/tag/backtracking/) | String (/tag/string/)

Hide Similar Problems | (H) Wildcard Matching (/problems/wildcard-matching/)

Have you met this question in a real interview? Yes No

Discuss (https://leetcode.com/discuss/questions/oj/regular-expression-matching) | Pick One (/problems/random-one-question/)

C++

```cpp
class Solution {
public:

    bool isMatch(string s, string p) {
        int M = s.length(), N = p.length();
        vector<vector<int>> dp(M+1, vector<int>(N+1));
        dp[0][0] = 1;
        for(int i = 1; i <=M; ++i) dp[i][0] = 0;
        for(int j = 1; j <=N; ++j) dp[0][j] = j > 1 && p[j-1] == '*' && dp[0][j-2];

        for(int i = 1; i <= M; ++i) {
            for(int j = 1; j <= N; ++j) {
                if(p[j-1] != '*') {
                    dp[i][j] = dp[i-1][j-1] && (s[i-1] == p[j-1] || p[j-1] == '.');
                } else {
                    dp[i][j] = dp[i][j-2] || ((s[i-1] == p[j-2] || p[j-2] == '.') && dp[i-1][j]);
                }
            }
        }
        return dp[M][N];
    }



    //http://xiaohuiliucuriosity.blogspot.com/2014/12/regular-expression-matching.html
    // bool isMatch(string s, string p) {
    //     int m = s.size(), n = p.size();
    //     vector<vector<bool>> f(m + 1, vector<bool>(n + 1, false));

    //     f[0][0] = true;
    //     for (int i = 1; i <= m; i++)
    //         f[i][0] = false;
    //     // p[0.., j - 3, j - 2, j - 1] matches empty iff p[j - 1] is '*' and p[0..j - 3] matches empty
    //     for (int j = 1; j <= n; j++)
    //         f[0][j] = j > 1 && '*' == p[j - 1] && f[0][j - 2];

    //     for (int i = 1; i <= m; i++)
    //         for (int j = 1; j <= n; j++)
    //             if (p[j - 1] != '*')
    //                 f[i][j] = f[i - 1][j - 1] && (s[i - 1] == p[j - 1] || '.' == p[j - 1]);
    //             else
    //                 // p[0] cannot be '*' so no need to check "j > 1" here
    //                 f[i][j] = f[i][j - 2] || (s[i - 1] == p[j - 2] || '.' == p[j - 2]) && f[i - 1][j];

    //     return f[m][n];

    // }

    // recursive
    // bool isMatch(string s, string p) {
    //     if(p.empty()) return s.empty();
    //     if(p.size() >= 2 && p[1] == '*') {
    //         return (isMatch(s, p.substr(2)) || (!s.empty() && (s[0] == p[0] || p[0] == '.') && isMatch(s.substr(1), p)));
    //     } else {
    //         return !s.empty() && (s[0] == p[0] || p[0] == '.') && isMatch(s.substr(1), p.substr(1));
    //     }
    // }
};
```

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

Custom Testcase ☐

Run Code | Submit Solution

Submission Result: Accepted (/submissions/detail/84545888/)

More Details › (/submissions/detail/84545888/)

Next challenges: (H) Valid Number (/problems/valid-number) (H) Integer to English Words (/problems/integer-to-english-words) (M) Can I Win (/problems/can-i-win)

Share your acceptance!

30 35

Notes

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)