Regular Expression Matching

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
```

Solution 1

Please refer to my blog post if you have any comment. Wildcard matching problem can be solved similarly.

```cpp
class Solution {
public:
    bool isMatch(string s, string p) {
        if (p.empty())    return s.empty();

        if ('*' == p[1])
            // x* matches empty string or at least one character: x* -> xx*
            // *s is to ensure s is non-empty
            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.subst
r(1), p.substr(1));
    }
};

class Solution {
public:
    bool isMatch(string s, string p) {
        /**
         * f[i][j]: if s[0..i-1] matches p[0..j-1]
         * if p[j - 1] != '*'
         *      f[i][j] = f[i - 1][j - 1] && s[i - 1] == p[j - 1]
         * if p[j - 1] == '*', denote p[j - 2] with x
         *      f[i][j] is true iff any of the following is true
         *      1) "x*" repeats 0 time and matches empty: f[i][j - 2]
         *      2) "x*" repeats >= 1 times and matches "x*x": s[i - 1] == x && f[
i - 1][j]
         * '.' matches any single character
         */
        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];
    }
};
```

written by xiaohui7 original link here

Solution 2

1.'.' is easy to handle. if p has a '.', it can pass any single character in s except '\0'.

2.'' *is a totally different problem. if p has a* `'` character, it can pass any length of first-match characters in s including '\0'.

```
  class Solution {
    public:
    bool matchFirst(const char *s, const char *p){
        return (*p == *s || (*p == '.' && *s != '\0'));
    }

bool isMatch(const char *s, const char *p) {
    if (*p == '\0') return *s == '\0';  //empty

    if (*(p + 1) != '*') {//without *
        if(!matchFirst(s,p)) return false;
        return isMatch(s + 1, p + 1);
    } else { //next: with a *
        if(isMatch(s, p + 2)) return true;    //try the length of 0
        while ( matchFirst(s,p) )       //try all possible lengths
            if (isMatch(++s, p + 2))return true;
    }
}
};
```

written by enriquewang original link here

Solution 3

In the given examples, the last one `isMatch("aab", "c*a*b") → true`; don't understand why these two strings matches? Can someone please help me understand this example?

written by shawnForsythe original link here