```python
class Solution(object):
    def isMatch(self, s, p):
        """
        :type s: str
        :type p: str
        :rtype: bool
        """
        m, n = len(s), len(p)

        # DP table: dp[i][j] = True if s[:i] matches p[:j]
        dp = [[False] * (n + 1) for _ in range(m + 1)]
        dp[0][0] = True  # Empty string matches empty pattern

        # Handle patterns like a*, a*b*, a*b*c* matching empty string
        for j in range(2, n + 1):
            if p[j - 1] == '*':
                dp[0][j] = dp[0][j - 2]

        # Fill DP table
        for i in range(1, m + 1):
            for j in range(1, n + 1):
                if p[j - 1] == '.' or p[j - 1] == s[i - 1]:
                    dp[i][j] = dp[i - 1][j - 1]  # Match current character
                elif p[j - 1] == '*':
                    # Match zero occurrence of previous char
                    dp[i][j] = dp[i][j - 2]
                    # Match one or more occurrences if previous char matches
                    if p[j - 2] == '.' or p[j - 2] == s[i - 1]:
                        dp[i][j] = dp[i][j] or dp[i - 1][j]

        return dp[m][n]
```