Dynamic Programming problem: regex matching

This is a classic dynamic programming problem: regex matching with support for . and *.

Pattern Matching Rules:
- 'a' matches 'a' exactly.
- '.' matches any single character.
- 'a*' matches zero or more 'a's.
- '.*' matches any sequence of characters, including the empty string.
- matching must consume the entire string S.

Strategy: Dynamic Programming (DP):

Define a DP table: dp[i][j] = true if s[0...i-1] matches p[0,...j-1]

Use a 2D boolean table dp[s.length()+1][p.length()+1], where:
- dp[0][0] = true (empty string matches empty pattern)
- fill the table bottom up

Transitions: For each dp[i][j], consider:

1. When p[j-1] != '*': You can match s[i-1] and p[j-1] only if: s[i-1] == p[j-1] or p[j-1] == '.' and dp[i-1][j-1] is true
2. When p[j-1] == '*': You need to look at the preceding character p[j-2] and consider:
   - dp[i][j-2]: match 0 of the preceding element
   - dp[i-1][j] && match(s[i-1], p[j-2]): use * to match more.

Key Helper: You often need a matches(i, j) function:

s[i-1] == p[j-1] || p[j-1] == '.'

Edge Cases:
- pattern like "a*b*c" matches any empty string
- don't access s[i-1] & p[j-1] when i==0 & j==0