

# Wildcard Matching

For this problem we need to match string  $s$  against pattern  $p$  with rules:

- $?$  matches any single character.
- $*$  matches zero or more characters.

We must match the entire string.

## Approaches:

### 1. Dynamic Programming ( $O(n \times m)$ ):

We define  $dp[i][j]$ :

- Whether  $s[0, \dots, i-1]$  matches  $p[0, \dots, j-1]$ .

Recurrence:

1. If  $p[j-1] == '*'$ :  
 $dp[i][j] = dp[i][j-1] \text{ OR } dp[i-1][j]$
2. If  $p[j-1] == '?'$  or  $p[j-1] == s[i-1]$ :  
 $dp[i][j] = dp[i-1][j-1]$
3. Else false.

Initialization:

·  $dp[0][0] = \text{true}$

·  $dp[0][j] = dp[0][j-1]$  if  $p[j-1] == '*'$  (empty string matched by  $*$ ).

Complexity:

Time:  $O(n \times m)$

Space:  $O(n \times m)$  (can optimize to  $O(m)$  with rolling arrays).

### 2. Greedy Approach ( $O(n+m)$ ):

There's also a two-pointer + backtracking greedy solution:

- Use pointers for  $s$  and  $p$ , track last  $*$ .

- If mismatch occurs, backtrack to last  $*$  and extend its match.

This is faster and uses  $O(1)$  space.