Number of Longest Increasing Subsequence

```
 class Solution {
public int findNumberOfLIS(int[] nums) {
  int n = nums.length;
  int[] dp = new int[n];
  int[] count = new int[n];

  Arrays.fill(dp, 1);
  Arrays.fill(count, 1);

  int maxLen = 1;

  for (int i = 1; i < n; i++) {
    for (int j = 0; j < i; j++) {
      if (nums[i] > nums[j]) {
        if (dp[j] + 1 > dp[i]) {
          dp[i] = dp[j] + 1;
          count[i] = count[j];
        } else if (dp[j] + 1 == dp[i]) {
          count[i] += count[j];
        }
      }
    }
    maxLen = Math.max(maxLen, dp[i]);
  }
  int res = 0;
  for (int i = 0; i < n; i++) {
    if (dp[i] == maxLen) {
      res += count[i];
    }
  }
```

```
        return res;
    }
}
```

Wildcard Matching

```
class Solution {
    public boolean isMatch(String s, String p) {
        int m = s.length(), n = p.length();
       boolean[][] dp = new boolean[m + 1][n + 1];
        dp[0][0] = true;
        for (int j = 1; j <= n; j++) {
            if (p.charAt(j - 1) == '*')
                dp[0][j] = dp[0][j - 1];
        }
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                char sc = s.charAt(i - 1);
                char pc = p.charAt(j - 1);

                if (pc == sc || pc == '?') {
                    dp[i][j] = dp[i - 1][j - 1];
                } else if (pc == '*') {

                    dp[i][j] = dp[i][j - 1] || dp[i - 1][j];
                }
            }
        }
        return dp[m][n];
    }
}
```