# Problem Classification and Structure

## Easy Problems (10)

1. Remove Letter To Equalize Frequency (#2423)

2. Distribute Money to Maximum Children (#2591)

3. Remove One Element to Make Array Strictly Increasing (#1909)

4. Can Place Flowers (#605)

5. Number of Valid Words in a Sentence (#2047)

6. X of a Kind in a Deck of Cards (#914)

7. Longest Even Odd Subarray With Threshold (#2760)

8. Long Pressed Name (#925)

9. Smallest Missing Integer Greater Than Sequential Prefix Sum (#2996)

10. Buddy Strings (#859)

## Medium Problems (10)

1. Zero Array Transformation III (#3362)

2. Count the Number of Substrings With Dominant Ones (#3234)

3. Maximum Frequency of an Element After Performing Operations I (#3346)

4. Minimum Number of Valid Strings to Form Target I (#3291)

5. Find the Lexicographically Smallest Valid Sequence (#3302)

6. Maximum Good Subarray Sum (#3026)

7. Count of Substrings Containing Every Vowel and K Consonants II (#3306)

8. Minimum Cost to Make Array Equalindromic (#2967)

9. Smallest Divisible Digit Product II (#3348)

10. Final Array State After K Multiplication Operations II (#3266)

## Hard Problems (10)

1. Find the Occurrence of First Almost Equal Substring (#3303)

2. Minimize the Maximum Adjacent Element Difference (#3357)

3. Strong Password Checker (#420)

4. Find the Original Typed String II (#3333)

5. Count Number of Balanced Permutations (#3343)

6. Find the Largest Palindrome Divisible by K (#3260)

7. Find X-Sum of All K-Long Subarrays II (#3321)

8. Length of the Longest Increasing Path (#3288)

9. String to Integer (atoi) (#8)

10. Divide Two Integers (#29)

## Problem Categories

1. String Manipulation: #2423, #2047, #925, #859, #3234, #3291, #420, #8

2. Array Operations: #1909, #605, #914, #2760, #3026, #3266

3. Number Theory: #2591, #2996, #3348, #29

4. Dynamic Programming: #3302, #3306, #3288

5. Mathematical: #2967, #3357, #3343, #3260

## Key Evaluation Metrics for Each Problem

1. Code Correctness (via LeetCode test cases)

2. Solution Performance (time and space complexity)

3. Code Quality (readability, cyclomatic complexity, duplication)

4. Language-Specific Implementation Differences

## Testing Strategy

1. Use uniform test cases across all three languages

2. Compare solution approaches between languages

3. Analyze performance characteristics

4. Document language-specific optimizations

This distribution ensures:

- Even coverage across difficulty levels

- Diverse problem types

- Comprehensive language feature usage

- Valid comparison basis across languages