

Problem 2790: Maximum Number of Groups With Increasing Length

Problem Information

Difficulty: Hard

Acceptance Rate: 22.72%

Paid Only: No

Tags: Array, Math, Binary Search, Greedy, Sorting

Problem Description

You are given a **0-indexed** array `usageLimits` of length `n`.

Your task is to create **groups** using numbers from `0` to `n - 1`, ensuring that each number, `i`, is used no more than `usageLimits[i]` times in total **across all groups**. You must also satisfy the following conditions:

- * Each group must consist of **distinct** numbers, meaning that no duplicate numbers are allowed within a single group.
- * Each group (except the first one) must have a length **strictly greater** than the previous group.

Return _an integer denoting the**maximum** number of groups you can create while satisfying these conditions._

Example 1:

Input: usageLimits = [1,2,5] **Output:** 3 **Explanation:** In this example, we can use 0 at most once, 1 at most twice, and 2 at most five times. One way of creating the maximum number of groups while satisfying the conditions is: Group 1 contains the number [2]. Group 2 contains the numbers [1,2]. Group 3 contains the numbers [0,1,2]. It can be shown that the maximum number of groups is 3. So, the output is 3.

Example 2:

Input: usageLimits = [2,1,2] **Output:** 2 **Explanation:** In this example, we can use 0 at most twice, 1 at most once, and 2 at most twice. One way of creating the maximum number of

groups while satisfying the conditions is: Group 1 contains the number [0]. Group 2 contains the numbers [1,2]. It can be shown that the maximum number of groups is 2. So, the output is 2.

****Example 3:****

****Input:**** usageLimits = [1,1] ****Output:**** 1 ****Explanation:**** In this example, we can use both 0 and 1 at most once. One way of creating the maximum number of groups while satisfying the conditions is: Group 1 contains the number [0]. It can be shown that the maximum number of groups is 1. So, the output is 1.

****Constraints:****

* `1 <= usageLimits.length <= 105` * `1 <= usageLimits[i] <= 109`

Code Snippets

C++:

```
class Solution {  
public:  
    int maxIncreasingGroups(vector<int>& usageLimits) {  
  
    }  
};
```

Java:

```
class Solution {  
public int maxIncreasingGroups(List<Integer> usageLimits) {  
  
}  
}
```

Python3:

```
class Solution:  
    def maxIncreasingGroups(self, usageLimits: List[int]) -> int:
```