Problem 3477: Fruits Into Baskets II

Problem Information

Difficulty: Easy
Acceptance Rate: 70.27%
Paid Only: No
Tags: Array, Binary Search, Segment Tree, Simulation, Ordered Set

Problem Description

You are given two arrays of integers, `fruits` and `baskets`, each of length `n`, where `fruits[i]` represents the **quantity** of the `ith` type of fruit, and `baskets[j]` represents the **capacity** of the `jth` basket.

From left to right, place the fruits according to these rules:

* Each fruit type must be placed in the **leftmost available basket** with a capacity **greater than or equal** to the quantity of that fruit type. * Each basket can hold **only one** type of fruit. * If a fruit type **cannot be placed** in any basket, it remains **unplaced**.

Return the number of fruit types that remain unplaced after all possible allocations are made.

Example 1:

Input: fruits = [4,2,5], baskets = [3,5,4]

Output: 1

Explanation:

* `fruits[0] = 4` is placed in `baskets[1] = 5`. * `fruits[1] = 2` is placed in `baskets[0] = 3`. * `fruits[2] = 5` cannot be placed in `baskets[2] = 4`.

Since one fruit type remains unplaced, we return 1.

Example 2:

Input: fruits = [3,6,1], baskets = [6,4,7]

Output: 0

Explanation:

* `fruits[0] = 3` is placed in `baskets[0] = 6`. * `fruits[1] = 6` cannot be placed in `baskets[1] = 4` (insufficient capacity) but can be placed in the next available basket, `baskets[2] = 7`. * `fruits[2] = 1` is placed in `baskets[1] = 4`.

Since all fruits are successfully placed, we return 0.

Constraints:

* `n == fruits.length == baskets.length` * `1 <= n <= 100` * `1 <= fruits[i], baskets[i] <= 1000`

Code Snippets

C++:

```
class Solution {
public:
int numOfUnplacedFruits(vector<int>& fruits, vector<int>& baskets) {

}
};
```

Java:

```
class Solution {
public int numOfUnplacedFruits(int[] fruits, int[] baskets) {

}
}
```

Python3:

```
class Solution:
def numOfUnplacedFruits(self, fruits: List[int], baskets: List[int]) -> int:
```