You have n bags numbered from 0 to n - 1. You are given two **0-indexed** integer arrays capacity and rocks. The ith bag can hold a maximum of capacity[i] rocks and currently contains rocks[i] rocks. You are also given an integer additionalRocks, the number of additional rocks you can place in **any** of the bags.

Return*the****maximum****number of bags that could have full capacity after placing the additional rocks in some bags.*

**Example 1:**

**Input:** capacity = [2,3,4,5], rocks = [1,2,4,4], additionalRocks = 2

**Output:** 3

**Explanation:**

Place 1 rock in bag 0 and 1 rock in bag 1.

The number of rocks in each bag are now [2,3,4,4].

Bags 0, 1, and 2 have full capacity.

There are 3 bags at full capacity, so we return 3.

It can be shown that it is not possible to have more than 3 bags at full capacity.

Note that there may be other ways of placing the rocks that result in an answer of 3.

**Example 2:**

**Input:** capacity = [10,2,2], rocks = [2,2,0], additionalRocks = 100

**Output:** 3

**Explanation:**

Place 8 rocks in bag 0 and 2 rocks in bag 2.

The number of rocks in each bag are now [10,2,2].

Bags 0, 1, and 2 have full capacity.

There are 3 bags at full capacity, so we return 3.

It can be shown that it is not possible to have more than 3 bags at full capacity.

Note that we did not use all of the additional rocks.

**Constraints:**

* n == capacity.length == rocks.length
* 1 <= n <= 5 \* 104
* 1 <= capacity[i] <= 109
* 0 <= rocks[i] <= capacity[i]
* 1 <= additionalRocks <= 109