<https://leetcode.com/problems/minimized-maximum-of-products-distributed-to-any-store/description/>

You are given an integer n indicating there are n specialty retail stores. There are m product types of varying amounts, which are given as a **0-indexed** integer array quantities, where quantities[i] represents the number of products of the ith product type.

You need to distribute **all products** to the retail stores following these rules:

A store can only be given **at most one product type** but can be given **any** amount of it.

After distribution, each store will have been given some number of products (possibly 0). Let x represent the maximum number of products given to any store. You want x to be as small as possible, i.e., you want to **minimize** the **maximum** number of products that are given to any store.

Return the minimum possible x.

**Example 1:**

**Input:** n = 6, quantities = [11,6]

**Output:** 3

**Explanation:** One optimal way is:

- The 11 products of type 0 are distributed to the first four stores in these amounts: 2, 3, 3, 3

- The 6 products of type 1 are distributed to the other two stores in these amounts: 3, 3

The maximum number of products given to any store is max(2, 3, 3, 3, 3, 3) = 3.

**Example 2:**

**Input:** n = 7, quantities = [15,10,10]

**Output:** 5

**Explanation:** One optimal way is:

- The 15 products of type 0 are distributed to the first three stores in these amounts: 5, 5, 5

- The 10 products of type 1 are distributed to the next two stores in these amounts: 5, 5

- The 10 products of type 2 are distributed to the last two stores in these amounts: 5, 5

The maximum number of products given to any store is max(5, 5, 5, 5, 5, 5, 5) = 5.

**Example 3:**

**Input:** n = 1, quantities = [100000]

**Output:** 100000

**Explanation:** The only optimal way is:

- The 100000 products of type 0 are distributed to the only store.

The maximum number of products given to any store is max(100000) = 100000.

**Constraints:**

m == quantities.length

1 <= m <= n <= 10^5

1 <= quantities[i] <= 10^5

**Attempt 1: 2024-12-07**

**Solution 1: Binary Search + Greedy (10 min)**

class Solution {

    public int minimizedMaximum(int n, int[] quantities) {

        // The minimum possible maximum is 1.

        int lo = 1;

        // The maximum possible maximum is max⁡(quantities)

        // because no single store can get more than all the

        // products of one type.

        int hi = 1;

        for(int q : quantities) {

            hi = Math.max(hi, q);

        }

        while(lo <= hi) {

            int mid = lo + (hi - lo) / 2;

            if(canDistribute(n, quantities, mid)) {

                hi = mid - 1;

            } else {

                lo = mid + 1;

            }

        }

        return lo;

    }

    // For a given maximum value 'minMaxNumOfProduct', calculate how many stores

    // are needed to distribute the products without exceeding 'minMaxNumOfProduct'

    // products per store.

    private boolean canDistribute(int n, int[] quantities, int minMaxNumOfProduct) {

        int storeCount = 0;

        for(int q : quantities) {

            // Ceiling division

            storeCount += ((q - 1) / minMaxNumOfProduct + 1);

            // More stores required than available, early termination

            if(storeCount > n) {

                return false;

            }

        }

        // If the number of required stores is ≤n, then x is feasible.

        return true;

    }

}

Time Complexity: O(nlogn)

Space Complexity: O(1)

**Refer to chatGPT**

Binary Search:

The minimum possible maximum is 1.

The maximum possible maximum is max⁡(quantities) because no single store can get more than all the products of one type.

Feasibility Check:

For a given maximum value x, calculate how many stores are needed to distribute the products without exceeding x products per store.

If the number of required stores is ≤n, then x is feasible.

class Solution {

public int minimizedMaximum(int n, int[] quantities) {

int left = 1, right = 0;

// Determine the range for binary search

for (int quantity : quantities) {

right = Math.max(right, quantity);

}

// Perform binary search

while (left < right) {

int mid = left + (right - left) / 2;

if (canDistribute(n, quantities, mid)) {

right = mid; // Try for a smaller maximum

} else {

left = mid + 1; // Increase the maximum

}

}

return left;

}

private boolean canDistribute(int n, int[] quantities, int maxPerStore) {

int requiredStores = 0;

for (int quantity : quantities) {

requiredStores += (quantity + maxPerStore - 1) / maxPerStore; // Ceiling division

if (requiredStores > n) {

return false; // More stores required than available

}

}

return true; // Distribution is feasible

}

}

**Explanation:**

Binary Search:

Start with the smallest and largest possible values for the maximum products per store.

Use binary search to minimize this value.

canDistribute **Function**:

For each product type, calculate how many stores are needed if no store can have more than maxPerStore products.

If the total required stores exceed nnn, return false.

Ceiling Division:

To calculate the number of stores needed for a given product type qqq with a limit xxx:
storesNeeded=⌈q/x⌉=(q+x−1)/x

**Refer to**

[L410.Split Array Largest Sum (Ref.L1011,L704,L1482)](note://WEB4d9d6f26a965381b137615128e2648be)

[L1011.Capacity To Ship Packages Within D Days (Ref.L410,L1482)](note://WEB87ca8f9dd9a14fb4535856a4e0f8f4b1)

[L1283.Find the Smallest Divisor Given a Threshold (Ref.L410,L1011,L1482)](note://76952362C4844CB3B68E28EA09F2EE58)

[L1482.Minimum Number of Days to Make m Bouquets (Ref.L410,L1011)](note://WEBd5ce394160da45a79e92dbbc806111ec)

[L1802.Maximum Value at a Given Index in a Bounded Array (Ref.L410)](note://WEB7d060d486a37a8c92696be41f18fb27c)