# 题目

给你一个整数 n ，表示有 n 间零售商店。总共有 m 种商品，每种商品的数目用一个下标从 0 开始的整数数组 quantities 表示，其中 quantities[i] 表示第 i 种商品的数目。

你需要将 所有商品 分配到零售商店，并遵守这些规则：

一间商店 至多 只能有 一种商品 ，但一间商店拥有的商品数目可以为 任意 件。

分配后，每间商店都会被分配一定数目的商品（可能为 0 件）。用 x 表示所有商店中分配商品数目的最大值，你希望 x 越小越好。也就是说，你想 最小化 分配给任意商店商品数目的 最大值 。

请你返回最小的可能的 x 。

示例 1：

输入：n = 6, quantities = [11,6]

输出：3

解释： 一种最优方案为：

- 11 件种类为 0 的商品被分配到前 4 间商店，分配数目分别为：2，3，3，3 。

- 6 件种类为 1 的商品被分配到另外 2 间商店，分配数目分别为：3，3 。

分配给所有商店的最大商品数目为 max(2, 3, 3, 3, 3, 3) = 3 。

示例 2：

输入：n = 7, quantities = [15,10,10]

输出：5

解释：一种最优方案为：

- 15 件种类为 0 的商品被分配到前 3 间商店，分配数目为：5，5，5 。

- 10 件种类为 1 的商品被分配到接下来 2 间商店，数目为：5，5 。

- 10 件种类为 2 的商品被分配到最后 2 间商店，数目为：5，5 。

分配给所有商店的最大商品数目为 max(5, 5, 5, 5, 5, 5, 5) = 5 。

示例 3：

输入：n = 1, quantities = [100000]

输出：100000

解释：唯一一种最优方案为：

- 所有 100000 件商品 0 都分配到唯一的商店中。

分配给所有商店的最大商品数目为 max(100000) = 100000 。

提示：

m == quantities.length

1 <= m <= n <= 105

1 <= quantities[i] <= 105

# 分析

要解决“最小化分配给任意商店商品数目的最大值”问题，核心思路是二分查找。通过二分枚举可能的最大商品数x，验证该x是否能满足“用不超过n间商店分配所有商品”的条件，最终找到最小的有效x。

解题思路

1、问题转化与二分边界：

- 目标是找到最小的x，使得所有商品能按规则分配到n间商店。

- 二分查找的下界（left）：1（每间商店至少分配1件商品，若商品数为0可忽略，但题目中quantities[i]≥1）。

- 二分查找的上界（right）：max(quantities)（最坏情况：某类商品数量极大，需单独用1间商店分配，此时x等于该商品数量）。

2、验证条件：

对某个候选x，计算分配所有商品所需的最少商店数：

- 对于第i种商品（数量为q），若每件商店最多分x件，则至少需要ceil(q / x)间商店（向上取整，如11件商品按x=3分配，需4间商店：11/3=3.666，向上取整为4）。

- 若所有商品所需的最少商店数≤n，则x是可行的，可尝试更小的x；否则x不可行，需尝试更大的x。

3、二分查找流程：

- 初始化left=1，right=max(quantities)。

- 循环计算mid=(left+right)/2，验证mid是否可行：

若可行：更新right=mid（尝试找更小的x）。

若不可行：更新left=mid+1（需找更大的x）。

- 当left==right时，left即为最小的x。

代码：

class Solution {

public:

int minimizedMaximum(int n, vector<int>& quantities) {

// 二分查找的边界：left=1，right=最大商品数量

int left = 1;

int right = \*max\_element(quantities.begin(), quantities.end());

while (left < right) {

int mid = left + (right - left) / 2; // 避免溢出的中间值计算

// 验证mid是否可行：所需商店数是否≤n

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

right = mid; // 可行，尝试更小的x

} else {

left = mid + 1; // 不可行，尝试更大的x

}

}

return left; // 最终left==right，即为最小x

}

private:

// 验证函数：当最大分配数为x时，是否能用≤n间商店分配所有商品

bool isFeasible(int x, int n, vector<int>& quantities) {

int requiredShops = 0; // 所需的最少商店数

for (int q : quantities) {

// 计算当前商品所需商店数：ceil(q / x) = (q + x - 1) / x（整数除法）

requiredShops += (q + x - 1) / x;

// 剪枝：若所需商店数已超过n，直接返回false

if (requiredShops > n) {

return false;

}

}

return requiredShops <= n;

}

};

代码解释

1、二分边界初始化：

- left=1：最小可能的x（至少分配1件商品）。

- right=\*max\_element(...)：最大可能的x（某类商品单独占1间商店，x等于该商品数量）。

2、二分查找核心循环：

- 计算mid时用left + (right - left)/2而非(left+right)/2，避免left+right溢出（虽题目中quantities[i]≤1e5，但属于通用优化）。

- 调用isFeasible验证mid：若可行，收缩右边界；否则收缩左边界，直到left==right。

3、可行性验证函数（isFeasible）：

- 对每种商品，用(q + x - 1) / x计算向上取整的结果（如q=11、x=3时，(11+3-1)/3=13/3=4，即需4间商店）。

- 剪枝优化：若计算过程中requiredShops已超过n，直接返回false，避免多余计算。

复杂度分析

- 时间复杂度：O(m log M)，其中m是商品种类数，M是quantities中的最大值（≤1e5）。

二分查找的次数为log M（约`17`次，因2^17=131072≥1e5）。

每次验证需遍历m种商品，时间为O(m)。

- 空间复杂度：O(1)，仅使用固定变量，无额外动态空间开销。