# 题目

给你一个非负整数数组nums。在一步操作中，你必须：

* 选出一个正整数x，x需要小于或等于nums中最小的非零元素。
* nums中的每个正整数都减去x。

返回使nums中所有元素都等于0需要的最少操作数。

示例 1：

输入：nums = [1,5,0,3,5]

输出：3

解释：

第一步操作：选出 x = 1 ，之后 nums = [0,4,0,2,4] 。

第二步操作：选出 x = 2 ，之后 nums = [0,2,0,0,2] 。

第三步操作：选出 x = 2 ，之后 nums = [0,0,0,0,0] 。

示例 2：

输入：nums = [0]

输出：0

解释：nums 中的每个元素都已经是 0 ，所以不需要执行任何操作。

提示：

1 <= nums.length <= 100

0 <= nums[i] <= 100

# 分析

## 方法一：排序+模拟

## 方法二：哈希表（推荐）

**思路：**

由于每次操作都将数组中的所有非零元素减少一个相同的值，因此数组中的相等元素减少到0的操作数相等，数组中的不相等元素减少到0的操作数不相等。

又由于使用贪心策略操作时，每次操作都会将数组中的最小非零元素减少到 0，因此最少操作数等于数组中的不同非零元素的个数。

使用哈希集合存储数组中的所有非零元素，则哈希集合的大小等于数组中的不同非零元素的个数，即为最少操作数。

需要注意的是，由于目标是将数组中的所有元素减为0，如果数组中的一个元素已经是0则不需要对该元素执行操作，因此只需要考虑数组中的不同非零元素的个数。

代码：

class Solution {

public:

int minimumOperations(vector<int>& nums) {

unordered\_set<int> hashSet;

for (int num : nums) {

if (num > 0) {

hashSet.emplace(num); // 更高效

}

}

return hashSet.size();

}

};

**复杂度分析：**

时间复杂度：O(n)，其中n是数组nums的长度。需要遍历数组一次，每个非零元素加入哈希集合的时间是O(1)。

空间复杂度：O(n)，其中n是数组nums的长度。哈希集合需要O(n)的空间。

**说明：**



emplace 是 insert 的更高效版本，但两者在简单场景下可以互换使用。当需要优化性能、避免临时对象的构造时，优先选择 emplace。

**另一种写法：**

class Solution {

public:

int minimumOperations(vector<int>& nums) {

unordered\_set<int> set;

for (auto& num : nums) {

if (num > 0) set.insert(num);

}

return set.size();

}

};