# 题目

给你一个整数数组 nums （下标从 0 开始）。每一次操作中，你可以选择数组中一个元素，并将它增加 1 。

比方说，如果 nums = [1,2,3] ，你可以选择增加 nums[1] 得到 nums = [1,3,3] 。

请你返回使 nums 严格递增 的 最少 操作次数。

我们称数组 nums 是 严格递增的 ，当它满足对于所有的 0 <= i < nums.length - 1 都有 nums[i] < nums[i+1] 。一个长度为 1 的数组是严格递增的一种特殊情况。

示例 1：

输入：nums = [1,1,1]

输出：3

解释：你可以进行如下操作：

1) 增加 nums[2] ，数组变为 [1,1,2] 。

2) 增加 nums[1] ，数组变为 [1,2,2] 。

3) 增加 nums[2] ，数组变为 [1,2,3] 。

示例 2：

输入：nums = [1,5,2,4,1]

输出：14

示例 3：

输入：nums = [8]

输出：0

提示：

1 <= nums.length <= 5000

1 <= nums[i] <= 10^4

# 分析

## 方法一：遍历

思路：

通过遍历数组，对于每个位置i，如果nums[i]小于等于nums[i-1]，则将nums[i]增加到nums[i-1]+1，同时记录增加的次数。最后返回总的增加次数即可。

代码：

class Solution {

public:

int minOperations(vector<int>& nums) {

int operations = 0;

for (int i = 1; i < nums.size(); ++i) {

if (nums[i] <= nums[i - 1]) {

operations += nums[i - 1] - nums[i] + 1;

nums[i] = nums[i - 1] + 1;

}

}

return operations;

}

};

## 方法二：贪心

思路：

代码：

class Solution {

public:

int minOperations(vector<int>& nums) {

int pre = nums[0] - 1, res = 0;

for (int num : nums) {

pre = max(pre + 1, num);

res += pre - num;

}

return res;

}

};

复杂度分析

时间复杂度：O(n)，其中n为数组nums的长度。

空间复杂度：O(1)，仅使用常量空间。