# 题目

给你一个非递减的 有序 整数数组，已知这个数组中恰好有一个整数，它的出现次数超过数组元素总数的 25%。

请你找到并返回这个整数

示例：

输入：arr = [1,2,2,6,6,6,6,7,10]

输出：6

提示：

1 <= arr.length <= 10^4

0 <= arr[i] <= 10^5

# 分析

## 方法一：遍历

思路：

我们可以利用题目中给出的条件：数组是非递减的，并且恰好有一个整数出现的次数超过了数组元素总数的25%。

这意味着这个特殊的整数在数组中连续出现的次数至少为 n / 4 + 1，其中 n 是数组的长度。由于数组是非递减的，我们可以遍历数组，并检查每个连续子数组的长度。一旦找到长度大于或等于 n / 4 + 1 的子数组，我们就可以返回该子数组中的任意一个元素，因为它们都是相同的特殊整数。

代码：

class Solution {

public:

int findSpecialInteger(vector<int>& arr) {

int n = arr.size();

int targetCount = n / 4 + 1; // 目标长度，至少为 n/4 + 1

int count = 1; // 当前连续子数组的长度

// 遍历数组，检查连续子数组的长度

for (int i = 1; i < n; ++i) {

if (arr[i] == arr[i - 1]) {

// 如果当前元素与前一个元素相同，则增加连续子数组的长度

count++;

// 如果连续子数组长度达到目标长度，则返回该元素

if (count >= targetCount) {

return arr[i];

}

} else {

// 如果当前元素与前一个元素不同，则重置连续子数组的长度

count = 1;

}

}

// 如果循环结束仍未找到特殊整数，则返回最后一个元素（理论上这种情况不会发生，因为题目保证了存在这样的整数）

return arr.back();

}

};