# 题目

给你一个下标从0开始的整数数组nums，该数组的大小为n，请你计算nums[j] - nums[i]能求得的 最大差值 ，其中0 <= i < j < n且nums[i] < nums[j]。

返回最大差值。如果不存在满足要求的i和j，返回-1。

示例 1：

输入：nums = [7,1,5,4]

输出：4

解释：

最大差值出现在 i = 1 且 j = 2 时，nums[j] - nums[i] = 5 - 1 = 4 。

注意，尽管 i = 1 且 j = 0 时 ，nums[j] - nums[i] = 7 - 1 = 6 > 4 ，但 i > j 不满足题面要求，所以 6 不是有效的答案。

示例 2：

输入：nums = [9,4,3,2]

输出：-1

解释：

不存在同时满足 i < j 和 nums[i] < nums[j] 这两个条件的 i, j 组合。

示例 3：

输入：nums = [1,5,2,10]

输出：9

解释：

最大差值出现在 i = 0 且 j = 3 时，nums[j] - nums[i] = 10 - 1 = 9 。

提示：

n == nums.length

2 <= n <= 1000

1 <= nums[i] <= 10^9

# 分析

## 方法一：直接遍历

思路：

我们可以使用一次遍历的方法。我们遍历数组 nums，同时跟踪到目前为止找到的最小值 min\_val。对于当前遍历到的元素 nums[j]，我们计算 nums[j] - min\_val，并更新最大差值 max\_diff 如果这个差值比当前的最大差值要大。同时，如果 nums[j] 比 min\_val 小，我们就更新 min\_val 为 nums[j]。

首先初始化 min\_val 为数组的第一个元素，max\_diff 为 -1。然后，从数组的第二个元素开始遍历，对于每个元素 nums[j]，如果它大于 min\_val，则计算差值并更新 max\_diff。无论 nums[j] 是否大于 min\_val，我们都更新 min\_val 为当前遍历到的最小值。最后返回 max\_diff。

请注意，根据题目要求，如果不存在满足 nums[i] < nums[j] 的 i 和 j，则返回 -1。在上面的代码中，如果数组是单调递减的或者所有元素都相同，max\_diff 将保持为初始值 -1，满足题目要求。

代码：

class Solution {

public:

int maximumDifference(vector<int>& nums) {

int n = nums.size();

int min\_val = nums[0]; // 初始化最小值为数组的第一个元素

int max\_diff = -1; // 初始化最大差值为 -1

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

// 如果当前元素大于之前的最小值，则计算差值并更新最大差值

if (nums[j] > min\_val) {

int diff = nums[j] - min\_val;

max\_diff = max(max\_diff, diff);

}

// 更新最小值

min\_val = min(min\_val, nums[j]);

}

return max\_diff;

}

};