# 题目

给你一个下标从 0 开始的整数数组 nums 。

请你从所有满足 i < j < k 的下标三元组 (i, j, k) 中，找出并返回下标三元组的最大值。如果所有满足条件的三元组的值都是负数，则返回 0 。

下标三元组 (i, j, k) 的值等于 (nums[i] - nums[j]) \* nums[k] 。

示例 1：

输入：nums = [12,6,1,2,7]

输出：77

解释：下标三元组 (0, 2, 4) 的值是 (nums[0] - nums[2]) \* nums[4] = 77 。

可以证明不存在值大于 77 的有序下标三元组。

示例 2：

输入：nums = [1,10,3,4,19]

输出：133

解释：下标三元组 (1, 2, 4) 的值是 (nums[1] - nums[2]) \* nums[4] = 133 。

可以证明不存在值大于 133 的有序下标三元组。

示例 3：

输入：nums = [1,2,3]

输出：0

解释：唯一的下标三元组 (0, 1, 2) 的值是一个负数，(nums[0] - nums[1]) \* nums[2] = -3 。因此，答案是 0 。

提示：

3 <= nums.length <= 100

1 <= nums[i] <= 10^6

# 分析

## 方法一：枚举j

思路：

枚举j，为了让(nums[i]−nums[j])∗nums[k]尽量大，我们需要知道j左侧元素的最大值，和j右侧元素的最大值。

也就是nums的前缀最大值preMax和后缀最大值sufMax，这都可以用递推预处理出来：

preMax[i]=max(preMax[i−1],nums[i]),

sufMax[i]=max(sufMax[i+1],nums[i])

代码实现时，可以只预处理sufMax数组，preMax可以在计算答案的同时算出来。

代码：

class Solution {

public:

long long maximumTripletValue(vector<int> &nums) {

int n = nums.size();

vector<int> suf\_max(n + 1, 0);

for (int i = n - 1; i > 1; i--) {

suf\_max[i] = max(suf\_max[i + 1], nums[i]);

}

long long ans = 0;

int pre\_max = nums[0];

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

ans = max(ans, (long long) (pre\_max - nums[j]) \* suf\_max[j + 1]);

pre\_max = max(pre\_max, nums[j]);

}

return ans;

}

};

复杂度分析

时间复杂度：O(n)，其中n为nums的长度。

空间复杂度：O(n)。

## 方法二：枚举k

思路：

枚举k，我们需要知道k左边nums[i]−nums[j]的最大值。

类似121.买卖股票的最佳时机，我们可以在遍历的过程中，维护nums[i]的最大值preMax，同时维护preMax减当前元素的最大值maxDiff，这就是k左边nums[i]−nums[j]的最大值。

代码：

class Solution {

public:

long long maximumTripletValue(vector<int> &nums) {

long long ans = 0;

int max\_diff = 0, pre\_max = 0;

for (int x : nums) {

ans = max(ans, (long long) max\_diff \* x);

max\_diff = max(max\_diff, pre\_max - x);

pre\_max = max(pre\_max, x);

}

return ans;

}

};

复杂度分析

时间复杂度：O(n)，其中n为nums的长度。

空间复杂度：O(n)。