# 题目

给你一个整数数组 nums，请你选择数组的两个不同下标 i 和 j，使 (nums[i]-1)\*(nums[j]-1) 取得最大值。

请你计算并返回该式的最大值。

**示例 1：**

输入：nums = [3,4,5,2]

输出：12

解释：如果选择下标 i=1 和 j=2（下标从 0 开始），则可以获得最大值，(nums[1]-1)\*(nums[2]-1) = (4-1)\*(5-1) = 3\*4 = 12 。

**示例 2：**

输入：nums = [1,5,4,5]

输出：16

解释：选择下标 i=1 和 j=3（下标从 0 开始），则可以获得最大值 (5-1)\*(5-1) = 16 。

**示例 3：**

输入：nums = [3,7]

输出：12

**提示：**

2 <= nums.length <= 500

1 <= nums[i] <= 10^3

# 分析

## 方法一：排序

## 方法二：逐个遍历（推荐）

class Solution {

public:

int maxProduct(vector<int>& nums) {

int maxNum = 0;

int secNum = 0;

for(auto i : nums)

{

if(i > maxNum)

{

secNum = maxNum;//此时存在一个更大的值,所以前面的最大值设置为第二大值

maxNum = i;

}

else if(i > secNum)

{

secNum = i;

}

}

return (maxNum - 1) \* (secNum - 1);

}

};

**复杂度分析：**

时间复杂度：O(n)

空间复杂度：O(n)

另一种写法：

class Solution {

public:

int maxProduct(vector<int>& nums) {

// 初始化最大和次大的数

int max1 = 0, max2 = 0;

// 遍历数组找最大和次大的数

for (int num : nums) {

if (num > max1) {

max2 = max1;

max1 = num;

} else if (num > max2) {

max2 = num;

}

}

// 计算并返回结果

return (max1 - 1) \* (max2 - 1);

}

};

## 方法三：优先队列/最大堆（推荐）

思路：

代码：

class Solution {

private:

int K = 2;

priority\_queue<int, vector<int>, greater<int> > pri\_queue;

public:

int maxProduct(vector<int>& nums) {

for (auto& num : nums) {

pri\_queue.push(num);

if (pri\_queue.size() > K) pri\_queue.pop();

}

int num1 = pri\_queue.top();

pri\_queue.pop();

int num2 = pri\_queue.top();

return (num1 - 1) \* (num2 - 1);

}

};

**复杂度：**

时间复杂度：

每次插入一个元素到优先队列需要O(log2)=O(1) 的时间。

因此，总的时间复杂度是O(n)，其中n是nums数组的长度。

空间复杂度：

空间复杂度是O(2)=O(1)，因为堆的大小最多为2。