至少是其他数字两倍的最大数

给你一个整数数组 nums ，其中总是存在 **唯一的** 一个最大整数 。

请你找出数组中的最大元素并检查它是否 **至少是数组中每个其他数字的两倍** 。如果是，则返回 **最大元素的下标** ，否则返回 -1 。

示例 1：

输入： nums = [3,6,1,0]

输出： 1

解释： 6 是最大的整数，对于数组中的其他整数，6 大于数组中其他元素的两倍。6 的下标是 1 ，所以返回 1 。

示例 2：

输入： nums = [1,2,3,4]

输出： -1

解释： 4 没有超过 3 的两倍大，所以返回 -1 。

示例 3：

输入： nums = [1]

输出： 0

解释： 因为不存在其他数字，所以认为现有数字 1 至少是其他数字的两倍。

```
class Solution:
    def dominantIndex(self, nums: List[int]) -> int:
    #遍历数组分别找到数组的最大值 和次大值。如果最大值大于 2 倍的次大值则最大值至少是数组其余数字的两倍，此时返回最大值的下标，否则返回 -1-1。
        firstmax=-1 #最大值
        secondmax=-1 #次大值
        idx=-1 #记录下标索引
        for i in range(len(nums)):
            if nums[i]>firstmax:
                secondmax=firstmax
                firstmax=nums[i]
                idx=i
            elif nums[i]>secondmax:
                secondmax=nums[i]
        if firstmax>=2*secondmax:
```

```
            return idx
        else:
            return -1
```