给定一个非空数组，返回此数组中第三大的数。如果不存在，则返回数组中最大的数。要求算法时间复杂度必须是O(n)。

示例 1:

输入：`[3, 2, 1]`

输出：`1`

解释：第三大的数是 `1`.

示例 2:

输入：`[1, 2]`

输出：`2`

解释：第三大的数不存在， 所以返回最大的数 `2` .

示例 3:

输入：`[2, 2, 3, 1]`

输出：`1`

解释：注意，要求返回第三大的数，是指第三大且唯一出现的数。
存在两个值为2的数，它们都排第二。

思路，用一个数组记录最大的前1，2，3的值，然后动态更新前1，2，3的状态。用nums[0]作为数组初始值。
当个数小于3个时，把条件划分为大于当前最大值，小于最小值和在两个之间。大于等于3个时，需判断大于第三大的且于第1，2不等。然后根据个数返回判断值

```
class Solution {
public:
    int thirdMax(vector<int>& nums) {
        int max[3]={nums[0],nums[0],nums[0]};
        int time=1;
        int n=nums.size();
        for(int i=1;i<n;++i){
            if(time<3){
                if(nums[i]!=max[0]&&nums[i]!=max[2]){
                    if(nums[i]>max[2]){
                        max[1]=max[2];
                        max[2]=nums[i];
                    }
                    else if(nums[i]<max[0]){
                        max[1]=max[0];
                        max[0]=nums[i];
                    }else{
```

```
                        max[1]=nums[i];
                    }
                    ++time;
                }
            }else if(nums[i]>max[0]&&nums[i]!=max[1]&&nums[i]!=max[2]){
                for(int j=0;j<2;++j){
                    max[j]=max[j+1];
                    if(max[j+1]>nums[i]){
                        max[j]=nums[i];
                        break;
                    }
                }
                if(max[2]<nums[i]){
                    max[2]=nums[i];
                }
            }
        }
        if(time==3){
            return max[0];
        }
        return max[2];
    }
};
```