给定一个排序数组，你需要在**原地**删除重复出现的元素，使得每个元素只出现一次，返回移除后数组的新长度。

不要使用额外的数组空间，你必须在**原地修改输入数组**并在使用 O(1) 额外空间的条件下完成。

示例 1:

给定数组 *nums* = **[1,1,2]**,

函数应该返回新的长度 **2**, 并且原数组 *nums* 的前两个元素被修改为 **1**, **2**。

你不需要考虑数组中超出新长度后面的元素。

示例 2:

给定 *nums* = **[0,0,1,1,1,2,2,3,3,4]**,

函数应该返回新的长度 **5**, 并且原数组 *nums* 的前五个元素被修改为 **0**, **1**, **2**, **3**, **4**。

你不需要考虑数组中超出新长度后面的元素。

说明:

为什么返回数值是整数，但输出的答案是数组呢?

请注意，输入数组是以**“引用”**方式传递的，这意味着在函数里修改输入数组对于调用者是可见的。

你可以想象内部操作如下:

```
// nums 是以“引用”方式传递的。也就是说，不对实参做任何拷贝
int len = removeDuplicates(nums);

// 在函数里修改输入数组对于调用者是可见的。
// 根据你的函数返回的长度, 它会打印出数组中该长度范围内的所有元素。
for (int i = 0; i < len; i++) {
    print(nums[i]);
}
```

思路：和移除元素思路差不多，遍历数组，从第二个元素开始遍历，然后若是和后一个元素相等，则移除，不等则保留，因为已经排序好了，相等的必连续。

```
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
         for(vector<int>::iterator i=nums.begin()+1;i<nums.end();i++){
            if(*i==*(i-1)){
                nums.erase(i);
                i--;
            }
        }
```

```
        return nums.size();
    }
};
```