user: sakura

第1题——数组中的重复数字

在一个长度为 n 的数组 nums 里的所有数字都在 0～n-1 的范围内。数组中某些数字是重复的，但不知道有几个数字重复了，也不知道每个数字重复了几次。请找出数组中任意一个重复的数字。

示例 1：

```
输入：[2, 3, 1, 0, 2, 5, 3]
```

```
输出：2 或 3
```

限制：2 <= n <= 100000

解题思路：

哈希表映射（1）

使用一个等长数组map，初值均为0；使用calloc实现。

在0-numsSize范围内，每碰到一次非重复的数字，将改数字作为等长数组的下标，赋值为1；循环条件为，该下标原本的值为0；

如果该值不为0，则该元素是重复的，返回；

代码实现

```
int findRepeatNumber(int* nums, int numsSize){

    int i=0;
    char *map=calloc(numsSize,sizeof(int));
    for(i=0;i<numsSize;i++)
    {
        if(map[nums[i]]==0)
        {
            map[nums[i]]=1;
        }
        else
        {
            return nums[i];
        }
    }
    return -1;
}
```

map	1	1	2	1	1				
i	0	1	2	3	4	5	6	7	
nums	3	2	1	4	5	2	7	6	

可以看到。当map的元素>1时，该元素是重复的，则返回该元素的下标（nums）

可以看到map[2]==2；因此，该值是重复的。

我们返回nums[2];

原地排序（2）

```
int findRepeatNumber(int* nums, int numsSize){
    if(numsSize)
    {
        for(int i=0;i<numsSize;i++)
        {
            if(nums[i]==i)//如果nums[i]==i,则不需要换位置
            {
                continue;
            }
            else if(nums[i]==nums[nums[i]])
            {
                return nums[i];
            }
            else
            {
                int temp=nums[nums[i]];
                nums[nums[i]]=nums[i];
                nums[i]=temp;
            }
        }
    }
    return -1;

}
```

算法核心：

如果nums[i]==i,则不需要换位置

如果nums[i]==nums[nums[i]],则碰到重复数据，我们根据题目，返回该值

如果以上两个条件均不满足，那么我们需要将两个元素互换位置。

C++哈希表实现

我们使用的是unordered_map容器

```
class Solution {
public:
    int findRepeatNumber(vector<int>& nums) {
        unordered_map<int, bool> map;
        for(int num:nums)
        {
            if(map[num])//如果map中已有num元素
            {
                return num;
            }
            else
            {
                map[num]=true;//该元素 未出现过则将其bool值置为1
            }
        }
        return  -1;
    }
};
```