```
/*
给定一个整数数组 a，其中1 ≤ a[i] ≤ n （n为数组长度），其中有些元素出现两次而其他元素出现一次。

找到所有出现两次的元素。

你可以不用到任何额外空间并在O(n)时间复杂度内解决这个问题吗？

示例：

输入:
[4,3,2,7,8,2,3,1]

输出:
[2,3]

https://leetcode-cn.com/classic/problems/find-all-duplicates-in-an-array/description/
*/
```

---

分析:

- 方法一:遍历数组,使用map数据结构保存元素出现的次数,出现2次的就直接返回;
- 方法二:由于$1 - 1 \leq a[i] - 1 \leq n - 1$,我们可以利用索引和元素的一一对应来过滤只出现一次的元素,找到了就把索引位置上的元素给对应给负值,如果对应索引位置上的值已经为负数了就找到了.

---

方法一:C++_map

```
class Solution
{
    public:
        vector<int> findDuplicates(vector<int>& nums)
        {
            map<int,int> mii;
            int i = 0;
            vector<int> ret_val ;

            for(i=0;i<nums.size();i++)
            {
                if(mii.count(nums[i]))
                {
                    ret_val.push_back(nums[i]);
                }
                else
                {
                    mii[nums[i]] = 1;
                }
            }

            return ret_val;
        }
};
```

```
/*
执行结果：
通过
显示详情
执行用时 :232 ms, 在所有 cpp 提交中击败了25.34%的用户
内存消耗 :27.8 MB, 在所有 cpp 提交中击败了5.55%的用户
*/
```

方法二:C++_索引法

```
class Solution
{
    public:
        vector<int> findDuplicates(vector<int>& nums)
        {
            int i = 0;
            vector<int> ret_val ;

            for(i=0;i<nums.size();i++)
            {
                /*已经被置设置为负数过*/
                if(nums[abs(nums[i])-1] <0)
                {
                    ret_val.push_back(abs(nums[i]));
                }
                else
                {
                    nums[abs(nums[i])-1] *= -1;
                }

            }
            return ret_val;
        }
};


/*
执行结果：
通过
显示详情
执行用时 :116 ms, 在所有 cpp 提交中击败了96.78% 的用户
内存消耗 :14.7 MB, 在所有 cpp 提交中击败了93.33%的用户
*/
```

AlimyBreak
2019.10.29