1. Majority Element II

Given an integer array of size *n*, find all elements that appear more than ⌊ n/3 ⌋ times.

**Note:** The algorithm should run in linear time and in O(1) space.

**Example 1:**

Input: [3,2,3]  
Output: [3]

**Example 2:**

Input: [1,1,1,3,3,2,2,2]  
Output: [1,2]

**解** 摩尔投票法

回顾求解超过一半的数字。假设超过一半的数字是x，剩下数字的集合记做是Y（），对于任意的，都能有一个x和其配对，最后还会剩下一个x或者刚好配对完。具体实现为：

实现的算法从第一个数开始扫描整个数组，有两个变量(参考第一答题者的变量名)major和count。其实这两个变量想表达的是一个“隐形的数组”array，**array存储的是“当前暂时无法删除的数字”**，我们先不要管major和count，只考虑这个array，同时再维护一个结果数组result，**result里面存储的是每次删除一对元素之后的当前结果**。[]

实现的算法从第一个数开始扫描整个数组，有两个变量(参考第一答题者的变量名)major和count。其实这两个变量想表达的是一个“隐形的数组”array，**array存储的是“当前暂时无法删除的数字”**，我们先不要管major和count，只考虑这个array，同时再维护一个结果数组result，**result里面存储的是每次删除一对元素之后的当前结果**。[https://www.zhihu.com/question/49973163/answer/235921864]

对于超过1/3的数字，必然最多有2两个。思路与求1/2的类似

class Solution {  
public:  
 vector<int> majorityElement(vector<int>& nums) {  
 int m, n, cnt\_m = 0, cnt\_n = 0;  
 for(int x : nums){  
 if(m == x){  
 cnt\_m++;  
 }else if(n == x){  
 cnt\_n++;  
 }else if(cnt\_m == 0){  
 cnt\_m = 1;  
 m = x;  
 }else if(cnt\_n == 0){  
 cnt\_n = 1;  
 n = x;  
 }else{  
 cnt\_m--;  
 cnt\_n--;  
 }  
 }  
 cnt\_m = 0, cnt\_n = 0;  
 for(int x: nums){  
 if(x == m)cnt\_m++;  
 else if(x == n)cnt\_n++;  
 }  
 vector<int>res;  
 if(cnt\_m > nums.size() / 3)res.push\_back(m);  
 if(cnt\_n > nums.size() / 3)res.push\_back(n);  
 return res;  
 }  
};