Majority Element II

Given an integer array of size *n*, find all elements that appear more than `⌊ n/3 ⌋` times. The algorithm should run in linear time and in O(1) space.

1. How many majority elements could it possibly have?
2. Do you have a better hint? Suggest it!

Solution 1

For those who aren't familiar with Boyer-Moore Majority Vote algorithm, I found a great article (http://goo.gl/64Nams) that helps me to understand this fantastic algorithm!! Please check it out!

The essential concepts is you keep a counter for the majority number**X**. If you find a number **Y** that is not **X**, the current counter should deduce 1. The reason is that if there is 5 **X** and 4 **Y**, there would be one (5-4) more **X** than **Y**. This could be explained as "4 **X** being paired out by 4 **Y**".

And since the requirement is finding the majority for more than ceiling of [n/3], the answer would be less than or equal to two numbers. So we can modify the algorithm to maintain two counters for two majorities.

Followings are my sample Python code:

```python
class Solution:
# @param {integer[]} nums
# @return {integer[]}
def majorityElement(self, nums):
    if not nums:
        return []
    count1, count2, candidate1, candidate2 = 0, 0, 0, 1
    for n in nums:
        if n == candidate1:
            count1 += 1
        elif n == candidate2:
            count2 += 1
        elif count1 == 0:
            candidate1, count1 = n, 1
        elif count2 == 0:
            candidate2, count2 = n, 1
        else:
            count1, count2 = count1 - 1, count2 - 1
    return [n for n in (candidate1, candidate2)
                if nums.count(n) > len(nums) // 3]
```

written by chungyushao original link here

Solution 2

```
vector<int> majorityElement(vector<int>& nums) {
    int cnt1=0, cnt2=0;
    int a,b;

    for(int n: nums){
        if (cnt1 == 0 || n == a){
            cnt1++;
            a = n;
        }
        else if (cnt2 == 0 || n==b){
            cnt2++;
            b = n;
        }
        else{
            cnt1--;
            cnt2--;
        }
    }

    cnt1=cnt2=0;
    for(int n: nums){
        if (n==a)   cnt1++;
        else if (n==b) cnt2++;
    }

    vector<int> result;
    if (cnt1 > nums.size()/3)   result.push_back(a);
    if (cnt2 > nums.size()/3)   result.push_back(b);
    return result;
}
```

written by longren original link here

Solution 3

Boyer-Moore Majority Vote algorithm generalization to elements appear more than floor(n/k) times

```
class Solution {
public:
  vector<int> majorityElement(vector<int> &a) {
    int y = 0, z = 1, cy = 0, cz = 0;
    for (auto x: a) {
      if (x == y) cy++;
      else if (x == z) cz++;
      else if (! cy) y = x, cy = 1;
      else if (! cz) z = x, cz = 1;
      else cy--, cz--;
    }
    cy = cz = 0;
    for (auto x: a)
      if (x == y) cy++;
      else if (x == z) cz++;
    vector<int> r;
    if (cy > a.size()/3) r.push_back(y);
    if (cz > a.size()/3) r.push_back(z);
    return r;
  }
};
```

written by MaskRay original link here