[**关于数组的几道面试题**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html)

2011年2月15日更新，加入[找出绝对值最小的元素](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link18)一题

数组是最基本的数据结构，关于数组的面试题也屡见不鲜，本文罗列了一些常见的面试题，仅供参考，如果您有更好的题目或者想法，欢迎留言讨论。目前有以下18道题目，如果有好的题目，随时更新。

* [数组求和](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link01)
* [求数组的最大值和最小值](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link02)
* [求数组的最大值和次大值](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link03)
* [求数组中出现次数超过一半的元素](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link04)
* [求数组中元素的最短距离](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link05)
* [求两个有序数组的共同元素](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link06)
* [求三个数组的共同元素](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link07)
* [找出数组中唯一的重复元素](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link08)
* [找出出现奇数次的元素](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link09)
* [求数组中满足给定和的数对](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link10)
* [最大子段和](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link11)
* [最大子段积](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link12)
* [数组循环移位](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link13)
* [字符串逆序](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link14)
* [组合问题](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link15)
* [合并两个数组](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link16)
* [重排问题](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link17)
* [找出绝对值最小的元素](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#link18)

**数组求和**

给定一个含有n个元素的整型数组a，求a中所有元素的和。可能您会觉得很简单，是的，的确简单，但是为什么还要说呢，原因有二，第一，这道题要求用递归法，只用一行代码。第二，这是我人生中第一次面试时候遇到的题，意义特殊。

**分析**

简单说一下，两种情况

1. 如果数组元素个数为0，那么和为0。

2. 如果数组元素个数为n，那么先求出前n - 1个元素之和，再加上a[n - 1]即可

**代码**

// 数组求和  
int sum(int\*a, int n)  
{  
     return n == 0 ? 0 : sum(a, n -1) + a[n -1];  
}

**求数组的最大值和最小值**

给定一个含有n个元素的整型数组a，找出其中的最大值和最小值

**分析**

常规的做法是遍历一次，分别求出最大值和最小值，但我这里要说的是分治法(Divide and couquer)，将数组分成左右两部分，先求出左半部份的最大值和最小值，再求出右半部份的最大值和最小值，然后综合起来求总体的最大值及最小值。这是个递归过程，对于划分后的左右两部分，同样重复这个过程，直到划分区间内只剩一个元素或者两个元素。

**代码**

[复制代码](javascript:void(0);)

// 求数组的最大值和最小值，返回值在maxValue和minValue  
void MaxandMin(int \*a, int l, int r, int& maxValue, int& minValue)  
{  
 if(l == r) // l与r之间只有一个元素  
 {  
 maxValue = a[l] ;  
 minValue = a[l] ;  
 return ;  
 }  
  
 if(l + 1 == r) // l与r之间只有两个元素  
 {  
 if(a[l] >= a[r])  
 {  
 maxValue = a[l] ;  
 minValue = a[r] ;  
 }  
 else  
 {  
 maxValue = a[r] ;  
 minValue = a[l] ;  
 }  
 return ;  
 }  
  
 int m = (l + r) / 2 ; // 求中点  
  
 int lmax ; // 左半部份最大值  
 int lmin ; // 左半部份最小值  
 MaxandMin(a, l, m, lmax, lmin) ; // 递归计算左半部份  
  
 int rmax ; // 右半部份最大值  
 int rmin ; // 右半部份最小值  
 MaxandMin(a, m + 1, r, rmax, rmin) ; // 递归计算右半部份  
  
 maxValue = max(lmax, rmax) ; // 总的最大值  
 minValue = min(lmin, rmin) ; // 总的最小值  
}

[复制代码](javascript:void(0);)

**求数组的最大值和次大值**

给定一个含有n个元素的整型数组，求其最大值和次大值

**分析**

思想和上一题类似，同样是用分治法，先求出左边的最大值leftmax和次大值leftsecond，再求出右边的最大值rightmax和次大值rightsecond，然后合并，如何合并呢？分情况考虑

1 如果leftmax > rightmax，那么可以肯定leftmax是最大值，但次大值不一定是rightmax，但肯定不是rightsecond，只需将leftsecond与rightmax做一次比较即可。

2 如果rightmax > leftmax，那么可以肯定rightmax是最大值，但次大值不一定是leftmax，但肯定不是leftsecond，所以只需将leftmax与rightsecond做一次比较即可。

**注意**

这种方法无法处理最大元素有多个的情况，比如3,5,7,7将返回7，7而不是7,5。感谢网友 从无到有靠谁人 指出。

**代码**

[复制代码](javascript:void(0);)

// 找出数组的最大值和次大值，a是待查找的数组，left和right是查找区间，max和second存放结果  
void MaxandMin(int a[], int left, int right, int&max, int&second)  
{  
 if(left == right)  
 {  
 max = a[left] ;  
 second =  INT\_MIN;  
 }  
 elseif(left +1== right)  
 {  
 max = a[left] > a[right] ? a[left] : a[right] ;  
 second = a[left] < a[right] ? a[left] : a[right] ;  
 }  
 else  
 {  
 int mid = left + (right - left) /2 ;  
  
 int leftmax ;  
 int leftsecond ;  
 MaxandMin(a, left, mid, leftmax, leftsecond) ;  
  
 int rightmax ;  
 int rightsecond ;  
 MaxandMin(a, mid +1, right, rightmax, rightsecond) ;  
  
 if (leftmax > rightmax)  
 {  
 max = leftmax ;  
 second = leftsecond > rightmax ? leftsecond : rightmax ;  
 }  
 else  
 {  
 max = rightmax ;  
 second = leftmax < rightsecond ? rightsecond : leftmax ;  
 }  
 }  
}

[复制代码](javascript:void(0);)

**求数组中出现次数超过一半的元素**

给定一个n个整型元素的数组a，其中有一个元素出现次数超过n / 2，求这个元素。据说是百度的一道题

**分析**

设置一个当前值和当前值的计数器，初始化当前值为数组首元素，计数器值为1，然后从第二个元素开始遍历整个数组，对于每个被遍历到的值a[i]

1 如果a[i]==currentValue，则计数器值加1

2 如果a[i] != currentValue， 则计数器值减1，如果计数器值小于0，则更新当前值为a[i]，并将计数器值重置为1

**代码**

[复制代码](javascript:void(0);)

// 找出数组中出现次数超过一半的元素  
int Find(int\* a, int n)  
{  
 int curValue = a[0] ;  
 int count = 1 ;  
  
 for (int i = 1; i < n; ++i)  
 {  
 if (a[i] == curValue)  
 count++ ;  
 else  
 {  
 count-- ;  
 if (count < 0)  
 {  
 curValue = a[i] ;  
 count = 1 ;  
 }  
 }  
 }  
  
 return curValue ;  
}

[复制代码](javascript:void(0);)

另一个方法是先对数组排序，然后取中间元素即可，因为如果某个元素的个数超过一半，那么数组排序后该元素必定占据数组的中间位置。

**求数组中元素的最短距离**

给定一个含有n个元素的整型数组，找出数组中的两个元素x和y使得abs(x - y)值最小

**分析**

先对数组排序，然后遍历一次即可

**代码**

[复制代码](javascript:void(0);)

int compare(const void\* a, const void\* b)  
{  
 return \*(int\*)a - \*(int\*)b ;  
}  
  
// 求数组中元素的最短距离  
void MinimumDistance(int\* a, int n)  
{  
 // Sort  
 qsort(a, n, sizeof(int), compare) ;  
  
 int i ; // Index of number 1  
 int j ; // Index of number 2  
  
 int minDistance = numeric\_limits<int>::max() ;  
 for (int k = 0; k < n - 1; ++k)  
 {  
 if (a[k + 1] - a[k] < minDistance)  
 {  
 minDistance = a[k + 1] - a[k] ;  
 i = a[k] ;  
 j = a[k + 1] ;  
 }  
 }  
  
 cout << "Minimum distance is: " << minDistance << endl ;  
 cout << "i = " << i << " j = " << j << endl ;  
}

[复制代码](javascript:void(0);)

**求两个有序数组的共同元素**

给定两个含有n个元素的有序（非降序）整型数组a和b，求出其共同元素，比如

a = 0, 1, 2, 3, 4

b = 1, 3, 5, 7, 9

输出 1, 3

**分析**

充分利用数组有序的性质，用两个指针i和j分别指向a和b，比较a[i]和b[j]，根据比较结果移动指针，则有如下三种情况

1. a[i] < b[j]，则i增加1，继续比较

2. a[i] == b[j]，则i和j皆加1，继续比较

3. a[i] < b[j]，则j加1，继续比较

重复以上过程直到i或j到达数组末尾。

**代码**

[复制代码](javascript:void(0);)

// 找出两个数组的共同元素  
void FindCommon(int\* a, int\* b, int n)  
{  
 int i = 0;  
 int j = 0 ;  
  
 while (i < n && j < n)  
 {  
 if (a[i] < b[j])  
 ++i ;  
 else if(a[i] == b[j])  
 {  
 cout << a[i] << endl ;  
 ++i ;  
 ++j ;  
 }  
 else// a[i] > b[j]  
 ++j ;  
 }  
}

[复制代码](javascript:void(0);)

这到题还有其他的解法，比如对于a中任意一个元素，在b中对其进行Binary Search，因为a中有n个元素，而在b中进行Binary Search需要logn。所以找出全部相同元素的时间复杂度是O(nlogn)。

另外，上面的方法，只要b有序即可，a是否有序无所谓，因为我们只是在b中做Binary Search。如果a也有序的话，那么再用上面的方法就有点慢了，因为如果a中某个元素在b中的位置是k的话，那么a中下一个元素在b中的位置一定位于k的右侧，所以本次的搜索空间可以根据上次的搜索结果缩小，而不是仍然在整个b中搜索。也即如果a和b都有序的话，代码可以做如下修改，记录上次搜索时b中元素的位置，作为下一次搜索的起始点。

**求三个数组的共同元素**

给定三个含有n个元素的整型数组a,b和c，求他们最小的共同元素。

**分析**

如果三个数组都有序，那么可以设置三个指针指向三个数组的头部，然后根据这三个指针所指的值进行比较来移动指针，直道找到共同元素。

**代码**

[复制代码](javascript:void(0);)

// 三个数组的共同元素-只找最小的  
void FindCommonElements(int a[], int b[], int c[], int x, int y, int z)  
{  
 for(int i = 0, j = 0, k = 0; i < x && j < y && k < z;)  
 {  
 if(a[i] < b[j])  
 {  
 i++ ;  
 }  
 else // a[i] >= b[j]  
 {  
 if(b[j] < c[k])  
 {  
 j++ ;  
 }  
 else // b[j] >= c[k]  
 {  
 if(c[k] < a[i])  
 {  
 k++ ;  
 }  
 else // c[k] >= a[i]  
 {  
 cout << c[k] << endl ;  
 return ;  
 }  
 }  
 }  
 }  
  
 cout << "Not found!" << endl ;  
}

[复制代码](javascript:void(0);)

如果三个数组都无序，可以先对a, b进行排序，然后对c中任意一个元素都在b和c中做二分搜索。

**代码**

[复制代码](javascript:void(0);)

// 找出三个数组的共同元素  
// O(NlogN)  
int UniqueCommonItem(int \*a, int \*b, int \*c, int n)  
{  
 // sort array a  
 qsort(a, n, sizeof(int), compare) ; // NlogN  
  
 // sort array b  
 qsort(b, n, sizeof(int), compare) ; // NlogN  
  
 // for each element in array c, do a binary search in a and b  
 // This is up to a complexity of N\*2\*logN  
 for (int i = 0; i < n; i++)  
 {  
 if(BinarySearch(a, n, c[i]) && BinarySearch(b, n, c[i]))  
 return c[i] ;  
 }  
  
 return - 1 ; // not found  
}

[复制代码](javascript:void(0);)

也可以对a进行排序，然后对于b和c中任意一个元素都在a中进行二分搜索，但是这样做是有问题的，你看出来了么？感谢网友[yy\_5533](http://home.cnblogs.com/u/518392/)指正。

**代码**

[复制代码](javascript:void(0);)

// 找出三个数组唯一的共同元素  
// O(NlogN)  
int UniqueCommonItem1(int \*a, int \*b, int \*c, int n)  
{  
 // sort array a  
 qsort(a, n, sizeof(int), compare) ; // NlogN  
  
 // Space for time  
 bool \*bb = new bool[n] ;  
 memset(bb, 0, n) ;  
  
 bool \*bc = new bool[n] ;  
 memset(bb, 0, n) ;  
  
 // for each element in b, do a BS in a and mark all the common element  
 for (int i = 0; i < n; i++) // NlogN  
 {  
 if(BinarySearch(a, n, b[i]))  
 bb[i] = true ;  
 }  
  
 // for each element in c, do a BS only if b[i] is true  
 for (int i = 0; i < n; i++) // NlogN  
 {  
 if(b[i] && BinarySearch(a, n, c[i]))  
 return c[i] ;  
 }  
  
 return - 1 ; // not found  
}

[复制代码](javascript:void(0);)

排序和二分搜索代码如下

[复制代码](javascript:void(0);)

// Determine whether a contains value k  
bool BinarySearch(int \*a, int n, int k)  
{  
 int left = 0 ;  
 int right = n - 1 ;  
 while (left <= right)  
 {  
 int mid = (left + right) ;  
  
 if(a[mid] < k)  
 left = mid + 1 ;  
 if(a[mid] == k)  
 return true ;  
 else  
 right = mid - 1 ;  
 }  
  
 return false ;  
}  
  
// Compare function for qsort  
int compare(const void\* a, const void\* b)  
{  
 return \*(int\*)a - \*(int\*)b ;  
}

[复制代码](javascript:void(0);)

小小总结一下，对于在数组中进行查找的问题，可以分如下两种情况处理

1. 如果给定的数组有序，那么首先应该想到Binary Search，所需O(logn)

2. 如果给定的数组无序，那么首先应该想到对数组进行排序，很多排序算法都能在O(nlogn)时间内对数组进行排序，然后再使用二分搜索，总的时间复杂度仍是O(nlogn)。

如果能做到以上两点，大多数关于数组的查找问题，都能迎刃而解。

**找出数组中唯一的重复元素**

给定含有1001个元素的数组，其中存放了1-1000之内的整数，只有一个整数是重复的，请找出这个数

**分析**

求出整个数组的和，再减去1-1000的和

**代码**

略

**找出出现奇数次的元素**

给定一个含有n个元素的整型数组a，其中只有一个元素出现奇数次，找出这个元素。这道题实际上是一个变种，原题是找出数组中唯一一个出现一次的元素，下面的方法可以同时解决这两道提。所以题目就用这个广义的吧。

**分析**

因为对于任意一个数k，有k ^ k = 0，k ^ 0 = k，所以将a中所有元素进行异或，那么个数为偶数的元素异或后都变成了0，只留下了个数为奇数的那个元素。

**代码**

[复制代码](javascript:void(0);)

int FindElementWithOddCount(int\*a, int n)  
{  
     int r = a[0] ;  
  
     for (int i =1; i < n; ++i)  
     {  
          r ^= a[i] ;  
     }  
  
     return r ;  
}

[复制代码](javascript:void(0);)

**求数组中满足给定和的数对**

给定两个有序整型数组a和b，各有n个元素，求两个数组中满足给定和的数对，即对a中元素i和b中元素j，满足i + j = d(d已知)

**分析**

两个指针i和j分别指向数组的首尾，然后从两端同时向中间遍历。

**代码**

[复制代码](javascript:void(0);)

// 找出满足给定和的数对  
void FixedSum(int\* a, int\* b, int n, int d)  
{  
 for (int i = 0, j = n - 1; i < n && j >= 0)  
 {  
 if (a[i] + b[j] < d)  
 ++i ;  
 else if (a[i] + b[j] == d)  
 {  
 cout << a[i] << ", " << b[j] << endl ;  
 ++i ;  
 --j ;  
 }  
 else // a[i] + b[j] > d  
 --j ;  
 }  
}

[复制代码](javascript:void(0);)

**最大子段和**

给定一个整型数组a，求出最大连续子段之和，如果和为负数，则按0计算，比如1， 2， -5， 6， 8则输出6 + 8 = 14

**分析**

编程珠玑上的经典题目，不多说了。

**代码**

[复制代码](javascript:void(0);)

// 子数组的最大和  
int Sum(int\* a, int n)  
{  
 int curSum = 0;  
 int maxSum = 0;  
 for (int i = 0; i < n; i++)  
 {  
 if (curSum + a[i] < 0)  
 curSum = 0;  
 else  
 {  
 curSum += a[i] ;  
 maxSum = max(maxSum, curSum);  
 }  
 }  
 return maxSum;  
}

[复制代码](javascript:void(0);)

**最大子段积**

给定一个整型数组a，求出最大连续子段的乘积，比如 1， 2， -8， 12， 7则输出12 \* 7 = 84

**分析**

与最大子段和类似，注意处理负数的情况

**代码**

[复制代码](javascript:void(0);)

// 子数组的最大乘积  
int MaxProduct(int \*a, int n)  
{  
 int maxProduct = 1; // max positive product at current position  
 int minProduct = 1; // min negative product at current position  
 int r = 1; // result, max multiplication totally  
  
 for (int i = 0; i < n; i++)  
 {  
 if (a[i] > 0)  
 {  
 maxProduct \*= a[i];  
 minProduct = min(minProduct \* a[i], 1);  
 }  
 else if (a[i] == 0)  
 {  
 maxProduct = 1;  
 minProduct = 1;  
 }  
 else // a[i] < 0  
 {  
 int temp = maxProduct;  
 maxProduct = max(minProduct \* a[i], 1);  
 minProduct = temp \* a[i];  
 }  
  
 r = max(r, maxProduct);  
 }  
  
 return r;  
}

[复制代码](javascript:void(0);)

**数组循环移位**

将一个含有n个元素的数组向右循环移动k位，要求时间复杂度是O(n)，且只能使用两个额外的变量，这是在微软的编程之美上看到的一道题

**分析**

比如数组 1 2 3 4循环右移1位 将变成 4 1 2 3， 观察可知1 2 3 的顺序在移位前后没有改变，只是和4的位置交换了一下，所以等同于1 2 3 4 先划分为两部分

1 2 3 | 4，然后将1 2 3逆序，再将4 逆序 得到 3 2 1 4，最后整体逆序 得到 4 1 2 3

**代码**

[复制代码](javascript:void(0);)

// 将buffer中start和end之间的元素逆序  
void Reverse( int buffer[], int start, int end )  
{  
 while ( start < end )  
 {  
 int temp = buffer[ start ] ;  
 buffer[ start++ ] = buffer[ end ] ;  
 buffer[ end-- ] = temp ;  
 }  
}  
  
// 将含有n个元素的数组buffer右移k位  
void Shift( int buffer[], int n, int k )  
{  
 k %= n ;  
  
 Reverse( buffer, 0, n - k - 1) ;  
 Reverse( buffer, n - k, n - 1 ) ;  
 Reverse( buffer, 0, n - 1 ) ;  
}

[复制代码](javascript:void(0);)

稍微扩展一下,如果允许分配额外的数组,那么定义一个新的数组,然后将移位后的元素直接存入即可,也可以使用队列,将移动后得元素出对,再插入队尾即可.

**字符串逆序**

给定一个含有n个元素的字符数组a，将其原地逆序。

**分析**

可能您觉得这不是关于数组的，而是关于字符串的。是的。但是别忘了题目要求的是原地逆序，也就是不允许额外分配空间，那么参数肯定是字符数组形式，因为字符串是不能被修改的（这里只C/C++中的字符串常量），所以，和数组有关了吧，只不过不是整型数组，而是字符数组。用两个指针分别指向字符数组的首位，交换其对应的字符，然后两个指针分别向数组中央移动，直到交叉。

**代码**

[复制代码](javascript:void(0);)

// 字符串逆序  
void Reverse(char\*a, int n)  
{  
     int left =0;   
     int right = n -1;  
  
     while (left < right)  
     {  
         char temp = a[left] ;  
         a[left++] = a[right] ;  
         a[right--] = temp ;  
     }  
}

[复制代码](javascript:void(0);)

**组合问题**

给定一个含有n个元素的整型数组a，从中任取m个元素，求所有组合。比如下面的例子

a = 1, 2, 3, 4, 5

m = 3

输出

1 2 3, 1 2 4, 1 2 5, 1 3 4, 1 3 5, 1 4 5

2 3 4, 2 3 5, 2 4 5  
3 4 5

**分析**

典型的排列组合问题，首选回溯法，为了简化问题，我们将a中n个元素值分别设置为1-n

**代码**

[复制代码](javascript:void(0);)

// n选m的所有组合  
int buffer[100] ;  
  
void PrintArray(int \*a, int n)  
{  
 for (int i = 0; i < n; ++i)  
 cout << a[i] << "";  
 cout << endl ;  
}  
  
bool IsValid(int lastIndex, int value)  
{  
 for (int i = 0; i < lastIndex; i++)  
 {  
 if (buffer[i] >= value)  
 return false;  
 }  
 return true;  
}  
  
void Select(int t, int n, int m)  
{  
 if (t == m)  
 PrintArray(buffer, m);  
 else  
 {  
 for (int i = 1; i <= n; i++)  
 {  
 buffer[t] = i;  
 if (IsValid(t, i))  
 Select(t + 1, n, m);  
 }  
 }  
}

[复制代码](javascript:void(0);)

**合并两个数组**

给定含有n个元素的两个有序（非降序）整型数组a和b。合并两个数组中的元素到整型数组c，要求去除重复元素并保持c有序（非降序）。例子如下

a = 1, 2, 4, 8

b = 1, 3, 5, 8

c = 1, 2, 3, 4, 5, 8

**分析**

利用合并排序的思想，两个指针i,j和k分别指向数组a和b，然后比较两个指针对应元素的大小，有以下三种情况

1. a[i] < b[j]，则c[k] = a[i]。

2. a[i] == b[j]，则c[k]等于a[i]或b[j]皆可。

3. a[i] > b[j]，则c[k] = b[j]。

重复以上过程，直到i或者j到达数组末尾，然后将剩下的元素直接copy到数组c中即可

**代码**

[复制代码](javascript:void(0);)

// 合并两个有序数组  
void Merge(int \*a, int \*b, int \*c, int n)  
{  
 int i = 0 ;  
 int j = 0 ;  
 int k = 0 ;  
  
 while (i < n && j < n)  
 {  
 if (a[i] < b[j])// 如果a的元素小，则插入a中元素到c  
 {  
 c[k++] = a[i] ;  
 ++i ;  
 }  
 else if (a[i] == b[j])// 如果a和b元素相等，则插入二者皆可，这里插入a  
 {  
 c[k++] = a[i] ;  
 ++i ;  
 ++j ;  
 }  
 else // a[i] > b[j] // 如果b中元素小，则插入b中元素到c  
 {  
 c[k++] = b[j] ;  
 ++j ;  
 }  
 }  
  
 if (i == n) // 若a遍历完毕，处理b中剩下的元素  
 {  
 for (int m = j; m < n; ++m)  
 c[k++] = b[m] ;  
 }  
 else//j == n, 若b遍历完毕，处理a中剩下的元素  
 {  
 for (int m = i; m < n; ++m)  
 c[k++] = a[m] ;  
 }  
}

[复制代码](javascript:void(0);)

**重排问题**

给定含有n个元素的整型数组a，其中包括0元素和非0元素，对数组进行排序，要求：

1. 排序后所有0元素在前，所有非零元素在后，且非零元素排序前后相对位置不变

2. 不能使用额外存储空间

例子如下

输入 0, 3, 0, 2, 1, 0, 0

输出 0, 0, 0, 0, 3, 2, 1

**分析**

此排序非传统意义上的排序，因为它要求排序前后非0元素的相对位置不变，或许叫做整理会更恰当一些。我们可以从后向前遍历整个数组，遇到某个位置i上的元素是非0元素时，如果a[k]为0，则将a[i]赋值给a[k]，a[k]赋值为0。实际上i是非0元素的下标，而k是0元素的下标

**代码**

[复制代码](javascript:void(0);)

void Arrange(int\* a, int n)  
{  
 int k = n -1 ;  
 for (int i = n -1; i >=0; --i)  
 {  
 if (a[i] !=0)  
 {  
 if (a[k] ==0)  
 {  
 a[k] = a[i] ;  
 a[i] =0 ;  
 }  
 --k ;  
 }  
 }  
}

[复制代码](javascript:void(0);)

**找出绝对值最小的元素**

给定一个有序整数序列（非递减序），可能包含负数，找出其中绝对值最小的元素，比如给定序列 -5, -3, -1, 2, 8 则返回1。

**分析**

由于给定序列是有序的，而这又是搜索问题，所以首先想到二分搜索法，只不过这个二分法比普通的二分法稍微麻烦点，可以分为下面几种情况

* 如果给定的序列中所有的数都是正数，那么数组的第一个元素即是结果。
* 如果给定的序列中所有的数都是负数，那么数组的最后一个元素即是结果。
* 如果给定的序列中既有正数又有负数，那么绝对值得最小值一定出现在正数和负数的连接处。

为什么？因为对于负数序列来说，右侧的数字比左侧的数字绝对值小，如上面的-5, -3, -1, 而对于整整数来说，左边的数字绝对值小，比如上面的2, 8，将这个思想用于二分搜索，可先判断中间元素和两侧元素的符号，然后根据符号决定搜索区间，逐步缩小搜索区间，直到只剩下两个元素。

**代码**

单独设置一个函数用来判断两个整数的符号是否相同。

[复制代码](javascript:void(0);)

bool SameSign(int a, int b)

{

if (a \* b > 0)

return true;

else

return false;

}

[复制代码](javascript:void(0);)

主函数代码。

[复制代码](javascript:void(0);)

// 找出一个非递减序整数序列中绝对值最小的数

int MinimumAbsoluteValue(int\* a, int n)

{

// Only one number in array

if (n ==1)

{

return a[0] ;

}

// All numbers in array have the same sign

if (SameSign(a[0], a[n -1]))

{

return a[0] >=0? a[0] : a[n -1] ;

}

// Binary search

int l =0 ;

int r = n -1 ;

while(l < r)

{

if (l +1== r)

{

return abs(a[l]) < abs(a[r]) ? a[l] : a[r] ;

}

int m = (l + r) /2 ;

if (SameSign(a[m], a[r]))

{

r = m -1;

continue;

}

if (SameSign(a[l], a[m]))

{

l = m +1 ;

continue;

}

}

}

[复制代码](javascript:void(0);)

这段代码是有问题的，感谢网友[lingyunfish](http://home.cnblogs.com/u/514263/)的指正，你看出来了么？修改后的代码如下：

[复制代码](javascript:void(0);)

// 找出一个非递减序整数序列中绝对值最小的数

int MinimumAbsoluteValue(int\* a, int n)

{

// Only one number in array

if (n ==1)

{

return a[0] ;

}

// All numbers in array have the same sign

if (SameSign(a[0], a[n -1]))

{

return a[0] >=0? a[0] : a[n -1] ;

}

// Binary search

int l =0 ;

int r = n -1 ;

while(l < r)

{

if (l + 1 == r)

{

return abs(a[l]) < abs(a[r]) ? a[l] : a[r] ;

}

int m = (l + r) /2 ;

if (SameSign(a[m], a[r]))

{

r = m;

continue;

}

else

{

l = m ;

continue;

}

}

}

[复制代码](javascript:void(0);)

=THE END=

作者：[zdd](http://www.cnblogs.com/graphics/)

出处：<http://www.cnblogs.com/graphics/>

本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接，否则保留追究法律责任的权利.

分类: [数据结构与算法](http://www.cnblogs.com/graphics/category/189593.html)

[**好文要顶**](javascript:void(0);) [**关注我**](javascript:void(0);) [**收藏该文**](javascript:void(0);) **[http://common.cnblogs.com/images/icon_weibo_24.png](javascript:void(0);)** **[http://common.cnblogs.com/images/wechat.png](javascript:void(0);)**

[http://pic.cnblogs.com/face/u64257.jpg?id=21102348](http://home.cnblogs.com/u/graphics/)

[zdd](http://home.cnblogs.com/u/graphics/)  
[关注 - 21](http://home.cnblogs.com/u/graphics/followees)  
[粉丝 - 776](http://home.cnblogs.com/u/graphics/followers)

荣誉：[推荐博客](http://www.cnblogs.com/expert/)

[+加关注](javascript:void(0);)

19

2

[«](http://www.cnblogs.com/graphics/archive/2010/08/21/1805548.html)上一篇：[D3D中设备丢失的处理](http://www.cnblogs.com/graphics/archive/2010/08/21/1805548.html)  
[»](http://www.cnblogs.com/graphics/archive/2010/09/02/1816170.html)下一篇：[D3DCOLOR与D3DXCOLOR](http://www.cnblogs.com/graphics/archive/2010/09/02/1816170.html)

posted on 2010-08-24 12:58 [zdd](http://www.cnblogs.com/graphics/) 阅读(43654) 评论(96) [编辑](https://i.cnblogs.com/EditPosts.aspx?postid=1761620) [收藏](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html)

[< Prev](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html" \l "!comments)[1](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#!comments)**2**

**评论**

[**#51楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2432855)

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2432330) zdd  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2432330)@haozhou引用引用求数组中出现次数超过一半的元素算法有问题，假设数组是{ 2, 3, 3, 3, 2, 2, 2 } 你的解法给出的结果是3,其实应该是2  
我的解法给出的结果是2，没有任何问题，不要轻易相信你的眼睛，运行代码才是王道！  
  
谢谢，是我把第一个循环中的＋＋i看成i＋＋了，如果是后者那结果是错误的。楼主原来算法没错。不好意思

[支持(0)反对(0)](javascript:void(0);)

2012-07-23 10:31 | [haozhou](http://home.cnblogs.com/u/429996/) 

[**#52楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2432948)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2432855) haozhou  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2432855)@zdd引用引用@haozhou引用引用求数组中出现次数超过一半的元素算法有问题，假设数组是{ 2, 3, 3, 3, 2, 2, 2 } 你的解法给出的结果是3,其实应该是2  
我的解法给出的结果是2，没有任何问题，不要轻易相信你的眼睛，运行代码才是王道！  
  
谢谢，是我把第一个循环中的＋＋i看成i＋＋了，如果是后者那结果是错误的。楼主原来算法没错。不好意思  
不用谢，其实和i也没有关系的，在这里i++和++i结果都一样。我劝你应该多动手，而不是光凭眼睛看。

[支持(0)反对(0)](javascript:void(0);)

2012-07-23 12:27 | [zdd](http://www.cnblogs.com/graphics/) 

[**#53楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2490766)

字符串逆序  
  
这道题我有个想法：  
我们可以再O（N）完成，不用任何辅助空间--移动k位。  
采用一个循环，开始为第一个元素，将第一个移动到对应的k的位置，再将原来k个位置的数据向后移，如果大于最后一个位置的偏移量，就采用取与来做。假如这个位置偏移为1 （就是开头的数据），循环完成，否则，继续循环。

[支持(0)反对(0)](javascript:void(0);)

2012-10-11 11:29 | [飞翔的大鸟](http://home.cnblogs.com/u/453805/) 

[**#54楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2491169)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2490766) 飞翔的大鸟  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2490766)字符串逆序  
这道题我有个想法：  
我们可以再O（N）完成，不用任何辅助空间--移动k位。  
采用一个循环，开始为第一个元素，将第一个移动到对应的k的位置，再将原来k个位置的数据向后移，如果大于最后一个位置的偏移量，就采用取与来做。假如这个位置偏移为1 （就是开头的数据），循环完成，否则，继续循环。  
不好意思，我没太看懂，还是发代码上来吧。不用任何辅助空间是不可能的，起码要用一个中间变量。

[支持(0)反对(0)](javascript:void(0);)

2012-10-11 18:00 | [zdd](http://www.cnblogs.com/graphics/) 

[**#55楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2491396)

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2491169) zdd  
这个算法我这几天想了下，这种算法有bug，如果(2\*移动次数)=N，这个算法就会失败。当然解决方法就是先判断一下，如果符合这个情况就用个循环来做，在N/2次循环下完成移动（交换a0<->a(n/2)，a1<->a(n/2+1)....)。当然需要辅助空间，我理解成了空间复杂度为O(1)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | void move(int\* arry,int N,int k)  {      int j=k;                        //第一位置移动的位置      int temppre=arry[0],tempcru=0;      do      {          tempcru=arry[j]; //记下当前数据的值          arry[j]=temppre;  //把上一个元素移动到该位置          temppre=tempcru;//记下当前的值          j=j+k;          if (j>N-1)                  //当j越界后，返回到前面              j=j%N;      } while (j!=0);                 //当j=0时，表示循环终止，因为否则就无线循环了      arry[0]=temppre;                //完成最后一个数据的填充  } |

也帮我看看，这个想法可行不。

[支持(0)反对(0)](javascript:void(0);)

2012-10-11 23:54 | [飞翔的大鸟](http://home.cnblogs.com/u/453805/) 

[**#56楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2491915)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2491396) 飞翔的大鸟  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2491396)@zdd这个算法我这几天想了下，这种算法有bug，如果(2\*移动次数)=N，这个算法就会失败。当然解决方法就是先判断一下，如果符合这个情况就用个循环来做，在N/2次循环下完成移动（交换a0<->a(n/2)，a1<->a(n/2+1)....)。当然需要辅助空间，我理解成了空间复杂度为O(1)  
  
也帮我看看，这个想法可行不。  
你不是说字符串逆序么？怎么变成移位了？

[支持(0)反对(0)](javascript:void(0);)

2012-10-12 16:26 | [zdd](http://www.cnblogs.com/graphics/) 

[**#57楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2492144)

哦，抱歉，确实错了。  
博主帮我看看这个算法严密不？

[支持(0)反对(0)](javascript:void(0);)

2012-10-12 22:28 | [飞翔的大鸟](http://home.cnblogs.com/u/453805/) 

[**#58楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2492186)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2492144) 飞翔的大鸟  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2492144)哦，抱歉，确实错了。博主帮我看看这个算法严密不？  
你自己运行一下就知道结果啦。

[支持(0)反对(0)](javascript:void(0);)

2012-10-12 23:59 | [zdd](http://www.cnblogs.com/graphics/) 

[**#59楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2628065)

最大字段积问题， 若数组是 int a[ ] = {0, 0, 0, -3, 0} 返回的是1。  
所以应该考虑全0或只有一个非零且为负数的情况。

[支持(0)反对(0)](javascript:void(0);)

2013-03-05 14:54 | [krystism](http://home.cnblogs.com/u/501304/) 

[**#60楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2630279)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2628065) krystism  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2628065)最大字段积问题， 若数组是 int a[ ] = {0, 0, 0, -3, 0} 返回的是1。所以应该考虑全0或只有一个非零且为负数的情况。  
非常感谢，稍候修改一下！

[支持(0)反对(0)](javascript:void(0);)

2013-03-08 09:27 | [zdd](http://www.cnblogs.com/graphics/) 

[**#61楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2643126)

第16个，两个升序数组合并成一个升序数组，删除重复元素，貌似关于分配数组空间写的不够详细，我自己写了一个，求指点  
  
public static int[] MergeSortedArray(int[] a, int[] b)  
{  
int m = 0, n = 0, count = 0, num = 0;  
while (m < a.Length && n < b.Length)  
{  
if (a[m] < b[n])  
m++;  
else  
if (a[m] > b[n])  
n++;  
else  
if (a[m] == b[n])  
{  
m++;  
n++;  
count++;  
}  
}  
int[] c = new int[a.Length + b.Length - count];  
m = 0; n = 0;  
while (m < a.Length && n < b.Length)  
{  
if (a[m] < b[n])  
c[num++] = a[m++];  
else  
if (a[m] > b[n])  
c[num++] = b[n++];  
else  
if (a[m] == b[n])  
{  
c[num++] = a[m++];  
n++;  
count++;  
}  
}  
while (m < a.Length)  
{  
c[num++] = a[m++];  
}  
while (n < b.Length)  
{  
c[num++] = b[n++];  
}  
return c;  
}

[支持(0)反对(0)](javascript:void(0);)

2013-03-26 16:28 | [我不是杰克船长](http://www.cnblogs.com/fengye87626/) 

[**#62楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2643321)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2643126) 过期明信片  
感谢，我原文中并没有范培内存的步骤，只专注于合并的部分。你的方法很好。另外，评论里面也可以格式化代码的。输入框右上角倒数第二个图标即是。

[支持(0)反对(0)](javascript:void(0);)

2013-03-26 20:54 | [zdd](http://www.cnblogs.com/graphics/) 

[**#63楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2651632)

找出绝对值最小的元素  
查找数组a[4]={-2, -1, 3, 4}返回的是3，好像是有点问题。循环中保持的不变式应该是a[l]<0 && a[r]>0,所以比较中间元a[m]与a[l],a[r]的符号后，要保持不变式，就应该是l=m或者r=m。  
if (SameSign(a[m], a[r]))  
{  
r = m;//a[r]>0  
continue;  
}  
if(SameSign(a[l], a[m]))  
{  
l = m ;//a[l]<0  
continue;  
}  
望指正

[支持(0)反对(0)](javascript:void(0);)

2013-04-08 09:45 | [lingyunfish](http://home.cnblogs.com/u/514263/) 

[**#64楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2651706)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2651632) lingyunfish  
非常感谢您的指正，确实有问题，按您的方法修改了代码，再次感谢！

[支持(0)反对(0)](javascript:void(0);)

2013-04-08 10:32 | [zdd](http://www.cnblogs.com/graphics/) 

[**#65楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2653644)

写的内容很赞。提一个小问题，最后那个找绝对值最小的问题，其实可以转化成在数组中lower\_bound查找0，绝对值最小的值肯定是这个位置或者旁边的

[支持(0)反对(0)](javascript:void(0);)

2013-04-09 23:48 | [wtommy](http://home.cnblogs.com/u/515205/) 

[**#66楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2653722)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2653644) wtommy  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2653644)写的内容很赞。提一个小问题，最后那个找绝对值最小的问题，其实可以转化成在数组中lower\_bound查找0，绝对值最小的值肯定是这个位置或者旁边的  
不好意思，没明白你的意思，什么叫 在数组中lower\_bound查找0？

[支持(0)反对(0)](javascript:void(0);)

2013-04-10 09:14 | [zdd](http://www.cnblogs.com/graphics/) 

[**#67楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2653831)

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2653722) zdd  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2653722)@wtommy引用引用写的内容很赞。提一个小问题，最后那个找绝对值最小的问题，其实可以转化成在数组中lower\_bound查找0，绝对值最小的值肯定是这个位置或者旁边的  
不好意思，没明白你的意思，什么叫 在数组中lower\_bound查找0？  
我想说的是，你可以直接用二分思想写一个类似STL里的lower\_bound函数，直接在数组中找到第一个<=0的元素位置，复杂度log(n)。绝对值最小的元素就在那附近（左边那个或者当前那个或者右边那个），这样就不需要你这么多的分支判断分析了，会简洁一些。

[支持(0)反对(0)](javascript:void(0);)

2013-04-10 10:36 | [wtommy](http://home.cnblogs.com/u/515205/) 

[**#68楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2654164)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2653831) wtommy  
问题是数组中不一定有<=0的元素。有可能全是正整数。

[支持(0)反对(0)](javascript:void(0);)

2013-04-10 15:26 | [zdd](http://www.cnblogs.com/graphics/) 

[**#69楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2654252)

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2654164) zdd  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2654164)@wtommy问题是数组中不一定有<=0的元素。有可能全是正整数。  
嗯，是可能都是正整数，但是lower\_bound是可以返回第一个不满足<0元素的位置。lower\_bound returns the furthermost iterator i in [first, last) such that, for every iterator j in [first, i), \*j < value. 你可以参考一下这个页面http://www.sgi.com/tech/stl/lower\_bound.html

[支持(0)反对(0)](javascript:void(0);)

2013-04-10 16:34 | [wtommy](http://home.cnblogs.com/u/515205/) 

[**#70楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2654375)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2654252) wtommy  
感觉很麻烦，还是这样吧。谢谢你。

[支持(0)反对(0)](javascript:void(0);)

2013-04-10 18:29 | [zdd](http://www.cnblogs.com/graphics/) 

[**#71楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2654413)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2654252) wtommy  
说白了，你这个方法也还是离不开分支判断。

[支持(0)反对(0)](javascript:void(0);)

2013-04-10 18:54 | [zdd](http://www.cnblogs.com/graphics/) 

[**#72楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2660466)

求三个数组的最小共同元素那个题，只对a排序，b、c不排序的那个方法，有点问题。当b[i]和c[i]同时都在a里面也说明不了就是共同元素

[支持(0)反对(0)](javascript:void(0);)

2013-04-18 10:33 | [yy\_5533](http://home.cnblogs.com/u/518392/) 

[**#73楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2660484)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2660466) yy\_5533  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2660466)求三个数组的最小共同元素那个题，只对a排序，b、c不排序的那个方法，有点问题。当b[i]和c[i]同时都在a里面也说明不了就是共同元素  
非常感谢，我看了一下，确实如此，此方法是错误的，为了提醒自己，已经在文章中加以说明。再次感谢。

[支持(0)反对(0)](javascript:void(0);)

2013-04-18 10:43 | [zdd](http://www.cnblogs.com/graphics/) 

[**#74楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2758725)

谢谢博主，从这里学到了好多，求数组的最大值和次大值,最大元素有多个时，可以通过对次大值与最大值比较，如果次大值等于最大值，将次大值值为0，就能够正确的返回最大值和次大值。

[支持(0)反对(0)](javascript:void(0);)

2013-08-23 12:14 | [华电岳黎明](http://www.cnblogs.com/yueliming/) 

[**#75楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2759466)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2758725) 华电岳黎明  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2758725)谢谢博主，从这里学到了好多，求数组的最大值和次大值,最大元素有多个时，可以通过对次大值与最大值比较，如果次大值等于最大值，将次大值值为0，就能够正确的返回最大值和次大值。  
嗯，是个路子，感谢提醒。

[支持(0)反对(0)](javascript:void(0);)

2013-08-24 15:48 | [zdd](http://www.cnblogs.com/graphics/) 

[**#76楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2774873)

楼主写的很棒，学习了，想问一下，求数组中出现次数超过一半的程序中，count表示的应该是出现的次数，但是如果利用数组1，2，3，4，1，1，1，1进行带入的话，1应该出现了5次，但是程序中似乎只判断了最后四次。

[支持(0)反对(0)](javascript:void(0);)

2013-09-13 11:39 | [urshine](http://home.cnblogs.com/u/562818/) 

[**#77楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2775051)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2774873) urshine  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2774873)楼主写的很棒，学习了，想问一下，求数组中出现次数超过一半的程序中，count表示的应该是出现的次数，但是如果利用数组1，2，3，4，1，1，1，1进行带入的话，1应该出现了5次，但是程序中似乎只判断了最后四次。  
这里的count确切的说是不同值元素两两抵消之后剩下的相同元素个数值，以你的例子来说1, 2, 3, 4, 1, 1, 1, 1  
1和2抵消，3和4抵消，那么剩下的就是4个1了。所以count=4.

[支持(0)反对(0)](javascript:void(0);)

2013-09-13 15:22 | [zdd](http://www.cnblogs.com/graphics/) 

[**#78楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2787830)

最后一个题目，找绝对值最小的元素，还是有bug，因为：-3 -2 -1 0 1 2 3   
那个while死循环。所以应该为：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | int bin(int a[],int n)  {      if(n==1) return a[0];      if(a[0]\*a[n-1]>0&&a[0]>0) return a[0];      if(a[0]\*a[n-1]>0&&a[0]<0) return a[n-1];      int l=0,r=n-1;      while(l<r)      {          if(l+1==r)              return min(abs(a[l]),abs(a[r]));          int m=(l+r)>>1;          if(a[l]\*a[m]>0)              r=m;          if(a[m]\*a[r]>0)              l=m;          if(a[m]==0) return 0;      }  } |

[支持(0)反对(0)](javascript:void(0);)

2013-10-08 10:12 | [simingtu](http://home.cnblogs.com/u/572487/) 

[**#79楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2802498)

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2787830) simingtu  
你的代码也有点小问题，应该是这样

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | int bin(int a[],int n)    {        if(n==1) return a[0];        if(a[0]\*a[n-1]>0&&a[0]>0) return a[0];        if(a[0]\*a[n-1]>0&&a[0]<0) return a[n-1];        int l=0,r=n-1;        while(l<r)        {            if(l+1==r)                return min(abs(a[l]),abs(a[r]));            int m=(l+r)>>1;            if(a[l]\*a[m]>0)                l=m;            if(a[m]\*a[r]>0)               r=m;            if(a[m]==0) return 0;        }    } |

[支持(0)反对(0)](javascript:void(0);)

2013-10-25 21:13 | [qswsjs](http://home.cnblogs.com/u/446771/) 

[**#80楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2806005)

厉害啊！

[支持(0)反对(0)](javascript:void(0);)

2013-10-31 08:14 | [学习中的小毛](http://www.cnblogs.com/math-mao/) 

[**#81楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2808820)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2787830) simingtu  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2787830)最后一个题目，找绝对值最小的元素，还是有bug，因为：-3 -2 -1 0 1 2 3 那个while死循环。所以应该为：  
确实是死循环了，因为有0的时候两个if语句都不满足。把第二个if改为else即可。谢谢告知！

[支持(0)反对(0)](javascript:void(0);)

2013-11-04 10:27 | [zdd](http://www.cnblogs.com/graphics/) 

[**#82楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2808821)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2802498) qswsjs  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2802498)@simingtu你的代码也有点小问题，应该是这样  
看了一下，感觉你们俩的代码一样啊，他的错在哪里？

[支持(0)反对(0)](javascript:void(0);)

2013-11-04 10:27 | [zdd](http://www.cnblogs.com/graphics/) 

[**#83楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2808822)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2806005) 学习中的小毛  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2806005)厉害啊！  
感谢支持！

[支持(0)反对(0)](javascript:void(0);)

2013-11-04 10:27 | [zdd](http://www.cnblogs.com/graphics/) 

[**#84楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2809140)

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2808821) zdd  
我改的25、29行，他写的颠倒了

[支持(0)反对(0)](javascript:void(0);)

2013-11-04 14:36 | [qswsjs](http://home.cnblogs.com/u/446771/) 

[**#85楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2809619)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2809140) qswsjs  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2809140)@zdd 我改的25、29行，他写的颠倒了  
看见了，多谢。

[支持(0)反对(0)](javascript:void(0);)

2013-11-04 23:24 | [zdd](http://www.cnblogs.com/graphics/) 

[**#86楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2876241)

博主太棒了！非常感谢！在学习“找出数组中唯一一个出现奇数次的元素”那题时，我忽然想起在一个面试题中看到了此题的一个变种。我开始以为那题与博主的题目是一码事，后来感觉似乎复杂的多。我看到的题目是这样：给定一个数组，如果数组中出现次数为奇数的数字唯一，输出该数字，否则输出-1。  
  
博主的原题是已经假定了数组中只有一个出现次数为奇数的数字，用异或法遍历一次数组就可以得出这个数字。然而我发现如果数组中出现次数为奇数的数字不唯一的话，那么用异或的方法遍历一次数组可能得出各种千奇百怪的结果。似乎很难用异或的方法来判断数组是否只有一个元素出现奇数次。比如{1,2,3,4,5}就会得出1（因为1^2^3^4^5 == 1)，而{3,4,5}就会得出2...  
  
当然，我可以用最笨的办法先把数组排序，然后遍历一次排序后的数组，统计每一个数字的重复次数是否为奇数，若发现奇数次数的元素达到两个，就输出-1...  
  
但我相信这肯定不是原题的意图，原题肯定是按照博主的思路让面试者使用异或，但我实在想不出如何用异或判断出现次数为奇数的元素是否唯一。盼望博主能给些亮光！非常感谢！

[支持(0)反对(0)](javascript:void(0);)

2014-02-16 13:18 | [yndng](http://home.cnblogs.com/u/604506/) 

[**#87楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2975717)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#2876241) yndng  
想了一下，你这个题目用一次遍历基本是行不通的，用异或也不行，因为奇数次元素不唯一。

[支持(0)反对(0)](javascript:void(0);)

2014-07-02 11:16 | [zdd](http://www.cnblogs.com/graphics/) 

[**#88楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3071632)

感觉那个出现一半以上的少了一次循环了。是吗

[支持(0)反对(0)](javascript:void(0);)

2014-11-25 22:07 | [蒲豪杰](http://www.cnblogs.com/puhaojie/) 

[**#89楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3071732)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3071632) 蒲豪杰  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3071632)感觉那个出现一半以上的少了一次循环了。是吗  
不太明白你的意思，能否详细说说？

[支持(0)反对(0)](javascript:void(0);)

2014-11-26 09:17 | [zdd](http://www.cnblogs.com/graphics/) 

[**#90楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3123855)

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3071732) zdd  
他的意思是，Find函数返回的那个值，还得重新遍历一遍数组来确定是否真的是主元素。他说的是对的  
  
举例说吧，如果输入数组是{1, 1, 2, 3, 4}，则Find函数返回值就是4。  
这个时候就需要拿着这个4，再回去遍历一遍

[支持(0)反对(0)](javascript:void(0);)

2015-02-05 11:19 | [chipgenius](http://home.cnblogs.com/u/721305/) 

[**#91楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3124285)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3123855) chipgenius  
[引用](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3123855)@zdd他的意思是，Find函数返回的那个值，还得重新遍历一遍数组来确定是否真的是主元素。他说的是对的  
  
举例说吧，如果输入数组是{1, 1, 2, 3, 4}，则Find函数返回值就是4。  
这个时候就需要拿着这个4，再回去遍历一遍  
兄弟，我恳请你再**仔仔细细**看一下那道题的描述可以么？  
你这个例子是不对的。

[支持(0)反对(0)](javascript:void(0);)

2015-02-05 17:50 | [zdd](http://www.cnblogs.com/graphics/) 

[**#92楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3125251)

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3124285) zdd  
好吧，原来条件是确定有主元素了  
之前面试过的都是O(n)时间去确认数组中是否有主元素，如果有的话就找到它

[支持(0)反对(0)](javascript:void(0);)

2015-02-06 23:42 | [chipgenius](http://home.cnblogs.com/u/721305/) 

[**#93楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3125587)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3125251) chipgenius  
是的，否则无法O(n).

[支持(0)反对(0)](javascript:void(0);)

2015-02-07 21:10 | [zdd](http://www.cnblogs.com/graphics/) 

[**#94楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3126640)

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3125587) zdd  
再遍历一遍就是O(2n)嘛，也是O(n)......

[支持(0)反对(0)](javascript:void(0);)

2015-02-09 23:17 | [chipgenius](http://home.cnblogs.com/u/721305/) 

[**#95楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3126963)**[楼主]**

[@](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3126640) chipgenius  
哈哈，对。我的意思是只遍历一次。

[支持(0)反对(0)](javascript:void(0);)

2015-02-10 12:32 | [zdd](http://www.cnblogs.com/graphics/) 

[**#96楼**](http://www.cnblogs.com/graphics/archive/2010/08/24/1761620.html#3148765)

不知道博主还管理博客么，两个问题，  
1，找出出现奇数次的元素问题，如果数组是{3,4,4,2,4,3}，也满足你说的“找出数组中唯一一个出现一次的元素”的情况，但结果就不对了，所以算法适用我觉得应该是“只有一个元素出现奇数次，找出它”  
2，求三个数组的共同元素问题，二分搜索“int mid = (left + right) ;”  
  
还有就是想问，求三个数组的共同元素问题，为什么当b[i]和c[i]同时都在a里面也说明不了就是共同元素