数组高频面试题精讲

七月算法　**曹鹏**

2015年4月22日

提纲

- 数组简介
- 面试题总体分析
  - 选题原则
  - 难度
  - 经典(常见)
  - 新颖
- 一些例题
  - 例1 局部最小值
  - 例2 第一个缺失的正整数
  - 例3 元素间的最大距离
  - 例4 只出现一次的数
  - 例5 众数问题
  - 例6 “前缀和”的应用
- 总结

数组简介

- 数组(array)
  - java : [], ArrayList
  - C++ : STL vector, []
  - C：只有[]
  - 理解:输入的数组通常理解为集合，我们自己可以排序，查找
  - 注意
    - C++ STL中vector的一种实现
    - 数组下标是一种特殊的hash…做计数
    - 理解数组与map
    - 给数组“顺序”

面试题总体分析

- 查找和排序
  - 二分查找
  - 元素交换
  - 排序，中位数
  - 归并
  - 位运算
  - 前缀和的应用
- 动态规划
- 排列组合

例1 局部极小值

- 例1 一个给定的不包含相同元素的整数数组，每个，局部极小值的定义是一个值比左右相邻的（如果存在）都小的值，求它的一个局部最小值（国外某公司面试题）
  - 分析：局部最小值的存在性，全部数组的最小值显然是一个解。O(n)?
  - 我们规定数组下标a[1..n]，并定义a[0] = a[n + 1] = ∞, 我们有a[1] < a[0], a[n] < a[n + 1]
  - 结论：子数组a[x..y] 若 a[x] < a[x – 1] , a[y] < a[y + 1]，则它包含一个局部极小值

例1——续

- mid = (x + y) / 2,二分，两个子数组a[x..mid], a[mid + 1..y]
  - 若a[mid] < a[mid + 1], 则子数组a[x..mid]满足a[x] < a[x - 1], a[mid] < a[mid + 1]
  - 反之a[mid] > a[mid + 1], 则子数组a[mid + 1..y]满足a[mid + 1] < a[mid], a[y] < a[y + 1]
- 复杂度 O(logn)
- 思考题
  - 循环有序数组最小值、查找元素x （Leetcode 153,154)
  - 一个严格单增的数组，查找a[x] == x的位置

例2 第一个缺失的正整数

- 给一个数组，找到从1开始第一个不在里面的正整数。

例如[3,4,-1,1]输出2。（Leetcode 41）

- 分析：数组下标从0开始
- 让a[i] == i + 1
- 每次循环
  - 要么i + 1
  - 要么n – 1
  - 要么有一个数被放到正确的位置

```
class Solution {
public:
    int firstMissingPositive(int A[], int n) {
        // [0..i) is 1..i
        for (int i = 0; i < n;) {
            if (A[i] == i + 1) {
                ++i;
            }
            else if ((A[i] <= i) || (A[i] > n) || (A[A[i] - 1] == A[i])) {
                A[i] = A[--n];
            }
            else {
                swap(A[i], A[A[i] - 1]);
            }
        }
        return n + 1;

    }
};
```

例3 元素最大间距离

- 给定一个整数数组(n > 1)，求把这些整数表示在数轴上，相邻两个数差的最大值。(Leetcode 164)
  - 显然排序是一个思想。有更好的方法么？
    - 最大值x, 最小值y, 如果x == y显然答案是0
    - 把数放进(n + 1)个桶
      - 每个桶大小是d = (x – y) / (n + 1) （浮点数）
      - 每个桶区间是[y + i * d, y + (i + 1) * d) (i=0,1,..n)
        - 注意是左闭右开的区间，最后一个桶是双闭区间
        - 最小的数在0号桶里，最大的数在n号桶里
        - 第一个桶非空，最后一个桶非空

例3 续

- 中间有空桶，空桶左右两侧肯定有元素！
- 最大间隙出现在一个非空桶的最大值和下一个非空桶的最小值之间
- 如何判断数r在哪个桶里？
  - $(r - y) * (n + 1) / (x - y)$（整数运算），注意r == x的时候，答案取n
  - 记录每个桶的最大值和最小值即可，时间空间都是O(n)

例4 只出现1次的数

- 一个数组，所有元素都出现了两次，只有两个数只出现了一次，求这两个数。
  - 分析：所有数做异或，则出现两个次的数相抵消，那么最终的结果就是那两个出现一次的数x和y的异或结果，即x xor y，且这个值非0
  - 既然x xor y非0，我们可以找到二进制表示中某一个为1的位（bit）（例如最低位），把所有的数按这位为1和为0分开。
  - 在该位为0和为1的数中，各有一个数只出现一次。（一个是x，另一个是y)

例4 续

伪代码:

```
int xXory = 0;
for (int i = 0; i < n; ++i) xXory ^= a[i];
int mask = 1;
for (; (xXory & mask) == 0; mask <<= 1);
int x = 0, y = 0;
for (int i = 0; i < n; ++i)
        if (a[i] & mask) x ^= a[i];
        else y ^= a[i];
```

- 思考题
  - Leetcode 137 除一个外，所有数出现了3次，求那个数 * （难）
  - 1-100，缺少了两个数，求这两个数？ 位运算？解方程？

例5 众数问题

- 找出超过一半的数
  - 分析：众数出现的次数大于其他所有数出现次数之和
    - 每次扔掉两个不同的数，众数不变
      - 如果扔掉一个众数，和一个非众数
      - 如果扔掉两个非众数
    - 如何实现？和x不同就扔掉，表示扔掉了一个x和一个y?

```
int count = 0, x;
for (int i = 0; i < n; ++i)
        if (count == 0) {x = a[i]; count = 1;}
        else if (x == a[i]) ++count;
        else --count;
//注意有的题目要数一下x出现次数是否确实超过一半。（众数可能不存在）
```

- 思考题：如何找到所有出现次数严格大于总数1 / k的数？ 提示：保存(k – 1)个数，如何查找？hash? map?

例6 前缀和的应用

- 给定浮点数组a,求一个数组b, b[i] = a[0] * a[1] *…*a[i – 1] * a[i + 1] * …*a[n – 1],不能使用除法，不允许再开数组
  - 先求“后缀积”

    for (int i = n – 1; i >= 0; --i) b[i] = a[i] * ((i == n – 1)?1:b[i + 1]);
  - 顺带求“前缀积”

    for (int i = 0, j = 1; i < n; j *= a[i++]) b[i] = j * ((i == n – 1)?1:b[i + 1]);

例6 续

- 关于前缀和的性质
  - a[i] + a[i + 1] + … + a[j] = sum[j] – sum[i – 1]
- 思考题
  - 求数组中连续一段和，绝对值最小？(codility)
    - 提示：用前缀和排序
  - 把一个数组从中间p位置分开，使得a[0] + ..+ a[p – 1]与a[p] + a[p + 1] + …+ a[n – 1]差值最小？
    - 提示：前缀和，与总和减去该前缀和的差最小，枚举
    - 如果都是非负数，可以采取“两头扫”的方法，和较小的那一边多加一个数（国外某公司的面试题）

总结

- 利用序
  - 理解二分查找
- 利用前缀和
  - 查找、计算、排序
- 理解数组
  - map
- 用数组实现高级数据结构
  - 一般树：存每个节点的父亲 （并查集）
  - 二叉树：下标从1开始a[i]的儿子是a[i * 2]和a[i * 2 + 1] (堆）
- 抓住简单题
  - 分治法求逆序对数
  - 有序数组归并
  - 两个有序数组的中位数
  - 两头扫的方法 （2-SUM, 3-SUM)