O(n)时间解决的面试题(下)

七月算法　曹鹏

2015年5月

提纲

- 例题汇总
  - 例1 最大子数组（和、乘积）
  - 例2 循环移位
  - 例3 快排partition
  - 例4 众数问题
  - 例5 单调堆栈
  - 例6 单调队列
  - 例7 树
  - 例8 滑动窗口
  - 例9 其他
- 结束语

例1 最大子数组

- 例1.1 给一个数组，求最大的连续子数组和。(动态规划实战例2，leetcode 53)
  - 方法1 记录最小前缀和——两个前缀和的差就是一段连续的子数组
  - 方法2 动态规划，记录以每个位置结束的最大子数组的和。
- 例1.2 给一个数组，求最大的连续子数组乘积(leetcode 152)

例1 续

- 类似例1.1
- 考虑的问题：
  - 溢出——没有溢出
  - 到当前项乘积最大
    - 之前乘积绝对值大
      - 保存之前最小乘积
      - 保存之前最大乘积

例1 续2

```
class Solution {
public:
    int maxProduct(vector<int>& nums) {
        int n = nums.size();
        if (n == 0) {
            return 1;
        }
        int mini = nums[0], maxi = nums[0],ansmin = nums[0],ansmax = nums[0];
        for (int i = 1; i < n; ++i) {
            int tempmin = min(nums[i], min(mini * nums[i], maxi * nums[i]));
            int tempmax = max(nums[i], max(mini * nums[i], maxi * nums[i]));
            mini = tempmin;
            maxi = tempmax;
            ansmin = min(mini, ansmin);
            ansmax = max(maxi, ansmax);
        }
        return ansmax;
    }
};
```

例2 循环移位

- 例2.1 一个数组，比如{1,2,3,4,5}循环移动一位就是{2,3,4,5,1},再移动一位变为{3,4,5,1,2}。
  - 分析： 长度为n, 把它移动m位,和移动m % n位是一样的。
    - 翻转前m位
    - 翻转后(n – m)位
    - 总体再翻转
    - 翻转可以O(n)做到：
      - for (int i = from, j = to; i < j; swap(a[i++], a[j--]));

例2 续

- 例2.2 单词翻转 (字符串高频面试题 例5)
- 例2.3 回文判断

例3 快排partition

- 例3.1 荷兰国旗问题 (排序查找实战例3 leetcode 75 ）
- 例3.2 奇偶数分开，正负数分开
- 例3.3 01交换排序（字符串高频面试题 例1）
- 例3.4 交换星号（字符串高频面试题 例3）
- 例3.5第一个缺失的整数（数组高频面试题 例2 leetcode 41）
- 例3.6 中位数、第k大(小)的数、最小的k个数

例3 续

- 找第k小的数关键
  - 5数取中做pivot（三数会退化）
  - Partition分三段（有相同数会退化）
    - 算法导论“偷懒”了
- 找到最小的k个数
  - 基于partition的方法找到的数是无序的

例4 众数问题

- 例5.1 找出出现次数超过一半的数 (数组高频面试题例5)
- 例5.2 推广找出出现次数大于1/k的数，用(k – 1)个map (hash table)，复杂度O(k * n)，注意k是常数的时候就是O(n)

例5 单调堆栈

- 例5 最大直方图（栈和队列例5）
  - 如栈时左边界确定
  - 出栈时右边界确定
  - 每块只出入一次
    - O(n)

例6 单调队列

- 例6.1 滑动窗口最值（栈和队列 例6）
- 例6.2 给定一个数组A和整数K,问有多少对下标i <= j满足max(A[i..j]) – min(A[i..j]) <= K
  - 分析：如果(i,j)满足条件，则(i + 1,j) (i + 2, j)…都满足条件。
    - 对每个i，找到第一个不满足条件的j
    - 如何求[i..j]的最大最小值？
      - 单调队列
      - 滑动窗口——两个边界都只增大不减
        - 滑动出去的不会进来

例6 续

```
int solution(int K, vector<int> &A) {
    // write your code in C++98
    deque<int> qmin,qmax;
    int answer = 0;
    for (int i = 0, j = 0; i < A.size(); ++i) {
        while (j < A.size()) {
            while ((!qmin.empty()) && (A[qmin.back()] >= A[j])) {
                qmin.pop_back();
            }
            qmin.push_back(j);
            while ((!qmax.empty()) && (A[qmax.back()] <= A[j])) {
                qmax.pop_back();
            }
            qmax.push_back(j);
            if (A[qmax.front()] - A[qmin.front()] <= K) {
                ++j;
            }
            else {
                break;
            }
        }
        if (qmin.front() == i) {
            qmin.pop_front();
        }
        if (qmax.front() == i) {
            qmax.pop_front();
        }

        answer += j - i;
        if (answer >=  1000000000) {
            return 1000000000;
        }
    }
    return answer;

}
```

例7 树相关

- 例7.1 树的高度
- 例7.2 二叉树对称判断
- 例7.3 二叉树平衡判断
- 例7.4 二叉树的最小深度
- 例7.5 指定和的路径
- 例7.6 二叉树双向链表转换
- 例7.7 前中后序遍历

例7 续

- 例7.8 给定一个树(无向无环图)，求距离最远的两个点(树的直径)
- 简单、巧妙地贪心
  - 以任意一点为根，找到距离它最远的节点x
  - 以x为根找到距离x最远的点y
  - (x,y)就是一条直径
- 如何找最远的点？ dfs求深度
- 思考题:算法证明？

例8 滑动窗口相关

- 例8 Leetcode 209 给定一个数组，里面全是正整数，再给一个正整数s，求数组里面最少多少个连续的数，满足总和不小于s
  - 核心，大窗口不满足条件，它的任意小窗口也不满足条件
  - 窗口[i..j]
    - 过小——++j
    - 过大——--i

例8 续

```
class Solution {
public:
    int minSubArrayLen(int s, vector<int>& nums) {
        int n = nums.size();
        int answer = n;
        for (int i = 0, j = 0, sum = 0, length = 0; j < n; ) { //[i..j - 1]
            while ((sum < s) && (j < n)) {
                sum += nums[j++];
            }
            if (sum >= s) {
                for (;sum >= s; sum -= nums[i++])
                ;
                answer = min(answer, j - i + 1);
            }
        }
        return (answer >= n)?0:answer;

    }
};
```

例8 续2

- 例8.2 子串变位词（字符串高频面试题精讲例4）
- 思考题1 最短子串包含全部字母 Leetcode 76
- 思考题2 无重复字符的最长子串

例9 其他

- 例9.1 2-SUM (Leetcode 1) 找到数组里和为s的两个数。
  - 方法1：排序，经典两头扫（排序不是O(n))
  - 方法2: 使用hash查找，对于x，查找s－x。
  - 如果hash足够好——总体复杂度是O(n)

例9 续

- 例9.2给定一个1-n的排列，每次只能把一个数放到序列末尾，至少几次能排好顺序？
  - 为什么要移动1？其他都排好了，1自然就好了
  - 如果要移动x，则我们必须把(x + 1), (x + 2).. n都移动了。
  - 从1-(x-1)是有序的
    - x有多大？

```
int solution(vector<int> &a) {
int n = a.size(), want = 1;
    for (int i = 0; i < n; ++i) {
        if (a[i] == want) {
            ++want;
        }
    }
    // want .. n must be moved
    return n - want + 1;
}
```

例9 续2

- 例9.3 给定一个1-n的排列，每次可以把一个数放到序列开头，也可以放到结尾，至少几次能排好序？
  - 分析
    - 把1..y移动到开头
    - 把x..n移动到末尾
    - [y + 1.. x – 1]

    必须按顺序出现
    - dp[x]表示从x开始在原
      - 即x, x + 1, …x + dp[x] – 1按顺序出现
      - 倒着循环i， dp[a[i]] = dp[a[i] + 1] + 1

```
int solution(vector<int> &a) {
int n = a.size(), m = 0;
vector<int> dp(n + 2, 0);  //使用1..n + 1 注意下标范围
    for (int i = n - 1; i >= 0; --i) {
        m = max(m, dp[a[i]] = dp[a[i] + 1] + 1);

    }
    return n - m;
}
```

结束语

- O(n)很神奇
- 多思考，勤练习
- 多写代码，多实践

谢谢大家

- 更多视频尽在：
  - http://www.julyedu.com/
    - 免费视频
    - 直播课程
    - 面试问答
- Contact us：微博
  - @七月算法
  - @七月算法问答
  - @曹鹏博士