O(n)时间解决的面试题（中）

七月算法　曹鹏

2015年5月11日

提纲

- 再谈O(n)
- 一些例题
  - 例1 下一个排列
  - 例2 均分01
  - 例3 X的个数
  - 例4 PAT的个数
  - 例5 最小平均值子数组
  - 例6 环形最大子数组和
  - 例7 允许交换一次的最大子数组和
- 总结

再谈O(n)

- 组合数学
  - 下一个排列(上一个排列)
  - 巧妙地证明
  - 计数 != 枚举
- 动态规划

例1 下一个排列

- 例1 (C++ STL) Next Permutation 找到字典序里的下一个排列。12345的下一个是12354,而54321的下一个认为是12345。（Leetcode 31)
  - 分析 关于字典序的理解：
    - a[0],a[1]…a[n – 1], 下一个排列是字典序比它大，最小的
    - 找到尽可能大的m, b[0] = a[0], b[1] = a[1]…b[m – 1] = a[m - 1], 而b[m] > a[m], b[m + 1..n – 1]是按照升序（不减序）排列的。

例1 续

- 目前排列是:　　(A)a[x](B)
- 下一个排列是: (A)a[y](B’)
  - A是相同的，A尽可能长
  - a[y] > a[x]
  - B’几乎是B里面的数排好顺序的结果
- 如何确定x?
  - 一个位置只要右边有数比它大就是候选的x
  - a[x]是最后一个这样的数（最右边）
    - a[x]右边的数,每个数的右边（后缀）没有比它大的
    - 所以a[x]右边的数是按照降序（不升序）排列

例1 续2

- 算法（二找、一交换、一翻转）
  - 找到最后一个严格升序的首位 $(a[i] < a[i+1])$，定义为x
    - $(A) = a[0..x-1]$ $(B) = a[x+1..n-1]$
  - 找到 $y > x$，$a[y] > a[x]$，且 $a[y]$ 最小
    - 一定存在，因为 $x+1$ 就是一个候选
    - $a[x]$后面的数都是降序，所以从后往前找到第一个大于 $a[x]$ 的位置就是y了
    - 可以二分找到y，但不影响总体时间复杂度
  - 交换 $a[x]$, $a[y]$
  - 对 $(x+1)$ 位后进行逆转
    - 交换后 $a[x+1..n-1]$ 仍然是降序(不升)
    - 逆转等于排序

例1 续3

```
public:
    void nextPermutation(vector<int>& nums) {
        int n = nums.size();
        int x;
        for (x = n - 2; (x >= 0) && (nums[x] >= nums[x + 1]); --x)
        ;
        if (x < 0) {
            reverse(nums.begin(), nums.end());
            return;
        }
        int y;
        for (y = n - 1; nums[y] <= nums[x]; --y)
        ;
        swap(nums[x], nums[y]);
        reverse(nums.begin() + x + 1, nums.end());
    }
```

例1 续4

- 思考题: 上一个排列?
- 提示:类似算法
  - 找到最后一个严格降序的首位 (a[i] > a[i + 1]), 定义为x
  - 找到y > x, a[y] < a[x], 且a[y]最大
  - 交换a[x], a[y]
  - 对(x + 1)位后进行逆转
- 优点
  - 复杂度低 O(n)
  - 可以应对有重复数的情况

例2 均分01

- 例2 给定一个01串，恰好包含2n个0和2n个1，你可以把它切成若干段，再把它们任意拼接，要求拼接出两部分，每部分恰好包含n个0，n个1，如何使得切的段数最少？
  - 举例1： 0101，从中间切一刀形成(01)，(01)， 分别作为两部分
  - 举例2: 0011， 切成3段(0) (01) (1)，把中间(01)单独作为一部分，剩余的(0) (1)作为另外一部分。

例2 续1

- 分析：下标从0开始,$f(x)$ $(x = 0,1,..2n)$,表示原串在$[x..x+2n-1]$的一段中0的个数与1的个数的差。
  - $f(0) + f(2n) = 0$ (恰好是全部的0和1)
    - 如果$f(0) = f(2n) = 0$，相当于从中间切一刀切成两段，可以完成，答案是2
    - 否则$f(0) < 0$ (或者> 0) $f(2n) > 0$ (或者<0)
      - $f(x)$是奇数还是偶数？ 始终是偶数
      - 窗口滑动 $f(x)$->$f(x + 1)$
        - 0和1的个数不变 差不变
        - 多了个0，少了个1，增加2
        - 少了个0，多了个1，减少2

例2 续2

- 窗口滑动过程中，偶数由负数到正数(或者由正数到负数)的过程中，必然能出现$f(y) = 0$，即存在$0 \le y \le 2n$使得$f(y) = 0$（不算0和2n)
  - 把$[y..2n + y - 1]$作为一段，它包含n个0和n个1
  - 剩余$[0..y - 1]$和$[2n + y..4n - 1]$一起合并为一段
  - 答案就是一共切成3段就可以
- 总之，答案是2或者3
- 所以只要看第一个窗口就可以了！

例2 续2

- 算法实现(伪代码)
  - i = 0 to 2n – 1 O(n)
    - s[i] == ‘0’ ++d;
    - s[i] ==‘1’ --d;
  - 最后d = f(0)
  - 如果d == 0 则答案是2
  - 否则 答案是3

例3 X的个数

- 例3（改自某公司online assignment) 给定一个长度为n的整数数组a，下标从0开始，再给定一个元素X，求一个位置m，满足0<=m<=n，且a[0..m – 1]中X的个数（如果m = 0表示空数组）和a[m..n – 1]中非X的个数(如果m == n，表示空数组）相等。
  - 分析：假设a中一共有x个X，给定m，假设a[0..m – 1]中有y个X，则a[m..n – 1]中非X的个数是(n – m) – (x – y) = n – m – x + y 根据题意要求n – m – x + y == y
  - 解得 m = n – x
  - 存在且唯一！

例3 续

- 直接统计一下有多少个X就可以了
  - O(n)时间，O(1)空间
  - 不需要“前缀和”之类的方法
- 思考题
  - 10个硬币，有4个是正面的，在不开灯的情况下，把它们分成两组，正面个数相等？
    - 提示：分成前6个一组和后4个一组，把后4个翻面……

例4 PAT的个数

- 例4 (PAT：Programming Ability Test)给定一个只包含P,A,T的串，求一共出现多少个“PAT”子序列？
  - 分析： 计数和枚举不同
    - p, pa, pat表示之前出现的“P”，“PA”，“PAT”的个数
    - s[i] == ‘P’ , ++p
    - s[i] == ‘A’, pa += p
    - s[i] == ‘T’, pat += pa
  - 时间复杂度O(n), 空间复杂度O(1)
  - 思考题： Leetcode 115

例5 最小平均值子数组

- 例5 (codility) 给定一个数组，求一个至少包含两个元素的子数组，满足平均值最小。输出子数组的起点，多个的时候输出最小的。
  - 分析：
    - 如果最优解长度为偶数，我们把它拆成长度为2的若干段。
    - 如果最优解长度为奇数(>2),我们把它拆成长度为2的若干段，和一段长度为3的段
    - 最优解中每一段的平均值都相等！
      - 如果优一段平均值比最优解小，至少优一段平均值比最优解大，矛盾。

例5 续

结论：只考虑长度为2和3的段就可以了。可以“滑动窗口”，也可以直接计算，因为2和3是常数……

可以使用乘法代替除法避免精度问题（前提：不溢出）

```
int solution(vector<int> &A) {
    // write your code in C++11
    int n = A.size();
    if (n <= 2) {
        return 0;
    }
    int total2 = A[1] + A[2];
    int best2;
    int start2;
    if (A[0] <= A[2]) {
        start2 = 0;
        best2 = A[0] + A[1];
    }
    else {
        start2 = 1;
        best2 = A[1] + A[2];
    }
    int total3 = A[0] + A[1] + A[2];
    int best3 = total3;
    int start3 = 0;
    for (int i = 3; i < n; ++i) {
        total2 += A[i] - A[i - 2];
        if (total2 < best2) {
            best2 = total2;
            start2 = i - 1;
        }
        total3 += A[i] - A[i - 3];
        if (total3 < best3) {
            best3 = total3;
            start3 = i - 2;
        }
    }
    int cmp = best2 * 3 - best3 * 2;
    if (cmp == 0) {
        return min(start2, start3);
    }
    return (cmp < 0)?start2 : start3;
}
```

例6 环形最大子数组和

- 例6 (itint5 9)给定一个数组，是环形的，最后一个元素和第一个元素相接，求最大子数组和。
  - 环形最大子数组和
    - 普通最大子数组和，例如 1 2 -4 5 6 -9
    - 开头和结尾的一部分， 例如1 2 -4 -5 -6 9
  - 算法
    - 求普通最大子数组和
    - 总和减去普通的最小子数组和
      - 可以考虑对原始数组取相反数，调用最大子数组和模块

例7 允许交换一次的最大子数组和

- 例7 (codility) 给定一个数组，在允许交换两个数的前提下（只允许交换一次，可以不换），求最大子数组和。
- 定义f[i]为两部分之和
  - 以a[i]结尾的最大子数组的和（可以为空）
  - 与任意a[0..i]里面单独一个元素
  - 以上两部分没有交集
  - 递推式 f[i] = max(f[i – 1] + a[i], max(a[0..i]))
- 定义g[i]:
  - 以a[i]开头的最大子数组和（非空）
    - 递推式: g[i] = max(g[i + 1], 0) + a[i]

例7 续

- 如果a[i]和a[j]交换 (j < i), 原来包含a[i]的最大子数组和变为
  - g[i] – a[i] + f[i – 1]
  - (即要换掉的元素在f[i – 1]里)
- 如果不交换，答案就是max{g[i]}
- 我们只考虑j < i的情况，对于j > i, 把a翻转再做一次就可以了。

```
int help(vector<int> &a) {
    int n = a.size();
    vector<int> f,g;
    f.resize(n);
    f[0] = a[0];
    int now = a[0];
    for (int i = 1; i < n; ++i) {
        now = max(now, a[i]);
        f[i] = max(a[i] + f[i - 1], now);
    }
    g.resize(n);
    g[n - 1] = a[n - 1];
    int answer = a[n - 1];
    for (int i = n - 2; i >= 0; --i) {
        g[i] = max(g[i + 1], 0) + a[i];
        answer = max(answer, g[i]);
    }
    for (int i = 1; i < n; ++i) {
        answer = max(answer, g[i] - a[i] + f[i - 1]);
    }
    return answer;
}

int solution(vector<int> &A) {
    // write your code in C++11
    int answer = help(A);
    reverse(A.begin(),A.end());
    answer = max(answer,help(A));
    return answer;

}
```

总结

- 多思考，多练习
- 计数 != 枚举
- 没有讲到的问题
  - $O(n^3)$优化到$O(n^2)$
  - 序列相关的问题
    - 给定一个1-n的排列，每次只能把一个数放到序列开头，至少几次能排好顺序？
    - 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？
  - 更多前缀、后缀的利用
  - …