面试中较难的 Follow Up 问题

主讲人 侯卫东

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
微博: http://www.weibo.com/ninechapter
知乎: http://zhuanlan.zhihu.com/jiuzhang
官网: http://www.jiuzhang.com

Overview

- Subarray Sum以及follow up
- Continuous Subarray Sum以及follow up
- Partition以及follow up
- Iterator以及follow up

Subarray sum及FollowUp

Subarray sum

http://www.lintcode.com/problem/subarray-sum/
http://www.jiuzhang.com/solutions/subarray-sum/

LintCode 138

九章算法

- 给定N个数，找到一段子数组使得和是0

- 例子：
- 输入：[-3, 1, 2, -3, 4]
- 输出：[0, 2]

分析：

- 利用前缀和，A[i] +…+ A[j] = S[j] – S[i - 1]
- 如果A[i] +…+ A[j]=0，说明S[j]=S[i-1]
- 利用哈希表

Subarray Sum Closest

https://www.lintcode.com/problem/subarray-sum-closest/

https://www.jiuzhang.com/solution/subarray-sum-closest/

LintCode 139

九章算法

- 给定N个数，找到一段子数组使得和最接近0

- 例子：
- 输入：[-3, 1, 1, -3, 5]
- 输出：[0, 2]

分析：

- 利用前缀和，A[i] +…+ A[j] = S[j] – S[i - 1]
- S[j]与S[i-1]差得越小，A[i] +…+ A[j]越接近0
- 将所有S值从小到大排序，检查每个相邻的对子

Submatrix Sum

http://www.lintcode.com/problem/submatrix-sum/
http://www.jiuzhang.com/solutions/submatrix-sum/

LintCode 405

- 给定一个整数矩阵，找到一个子矩阵使得数字之和为0

- 例子：
- 输入：
  - [[1 ,5 ,7],
  - [3 ,7 ,-8],
  - [4 ,-8 ,9]]
- 输出：
  - [(1, 1), (2, 2)]

分析：

九章算法

- 枚举子矩阵的上边界U，下边界D
- 将U和D之间的数按照列求和，然后仿照Subarray Sum解决
- 时间复杂度$O(N^3)$

Subarray Sum II

http://www.lintcode.com/problem/subarray-sum-ii/
http://www.jiuzhang.com/solutions/subarray-sum-ii/

LintCode 404

九章算法

- 给定一个正整数数组，求出在所有子数组中，数字之和在给定区间内的个数

- 例子：
- 输入：
  - [1, 2, 3, 4]
  - 范围 [1, 3]
- 输出：
  - 4

分析：

- 利用前缀和，A[i] +…+ A[j] = S[j] – S[i - 1]
- A都是正整数→S严格递增
- 每个S[j]要找到在[S[j]-end，S[j]-start]中的S[i]个数
  - 同向双指针！
- 时间复杂度:O(N)

编程

Continuous Subarray Sum 及FollowUp

Continuous Subarray Sum

www.lintcode.com/problem/continuous-subarray-sum/
http://www.jiuzhang.com/solutions/continuous-subarray-sum/

LintCode 402

- 给定一个整数数组，求出在所有子数组中，数字之和最大的一个

- 例子：
- 输入：
  - [-3, 1, 3, -3, 4]
- 输出：
  - [1, 4]

分析：

- 利用前缀和，A[i] +…+ A[j] = S[j] – S[i - 1]
- 每个S[j]需要找到之前最小的S[i-1]
- 时间复杂度:O(N)

Continuous Subarray Sum II

http://www.lintcode.com/problem/continuous-subarray-sum-ii/
http://www.jiuzhang.com/solutions/continuous-subarray-sum-ii/

LintCode 403

- 给定一个循环整数数组，求出在所有子数组中，数字之和最大的一个

- 例子：
- 输入：
  - [3, 1, -100, -3, 4]
- 输出：
  - [4, 1]

分析：

- 先按照Continuous Subarray Sum求出无环情况下的最大非空子数组和S_1
- 再用类似方法求出无环情况下的最小非空子数组和S_2
- 答案即为$\max\{S_1, total_sum - S_2\}$
- 特殊情况：如果选择了最小非空子数组而它是整个数组，那么选取无环情况下的最大非空子数组和S_1
- 时间复杂度:O(N)

编程

Partition Follow Up

Quick select

http://www.lintcode.com/problem/kth-largest-element/

http://www.jiuzhang.com/solutions/kth-largest-element/

LintCode 5: Kth Largest

- PriorityQueue
- 时间复杂度O(nlogk)
- 更适合动态维护Topk

- QuickSelect
- 时间复杂度O(n)
- 更适合静态第k大

Wiggle Sort

http://www.lintcode.com/problem/wiggle-sort/

http://www.jiuzhang.com/solutions/wiggle-sort/

LintCode 508

九章算法

- 给定一个数组，要求按照如下条件排
- nums[0]<=nums[1]>=nums[2]<=nums[3]....

- 例子：
- 输入：
  - [3, 5, 2, 1, 6, 4]
- 输出：
  - [1, 6, 2, 5, 3, 4]

分析：

- 要求nums[0] <= nums[1] >= nums[2] <= nums[3]....，即：
  - 当i为奇数时，nums[i] >= nums[i - 1]
  - 当i为偶数时，nums[i] <= nums[i - 1]
- 从左到右，依次处理每个数字
- 如果和条件不满足，与之前的数交换位置
- 时间复杂度O(N)

Wiggle Sort II

http://www.lintcode.com/problem/wiggle-sort-ii/

http://www.jiuzhang.com/solutions/wiggle-sort-ii/

LintCode 507

九章算法

- 给定一个数组，要求按照如下条件排
- nums[0]<nums[1]>nums[2]<nums[3]....

- 例子：
- 输入：
  - [3, 5, 2, 1, 6, 4]
- 输出：
  - [1, 6, 2, 5, 3, 4]

分析：

- 如果获得了中位数，小于中位数的放在nums[0,2,4,..]，大于中位数的放在nums[1,3,5,…]
- 用quickselect获得中位数：O(N)

课间休息五分钟

Nuts & Bolts Problem

http://www.lintcode.com/problem/nuts-bolts-problem/

http://www.jiuzhang.com/solutions/nuts-bolts-problem/

LintCode 399

九章算法

- 给定一组 n 个不同大小的 nuts 和 n 个不同大小的 bolts
- nuts 和 bolts 一一匹配。 不允许将 nut 之间互相比较，也不允许将 bolt 之间互相比较，只许将 nut 与 bolt 进行比较， 或将 bolt 与 nut 进行比较
- 输出对应的 nuts 与 bolts

- 例子：
- 输入：
  - Given nuts = ['ab','bc','dd','gg'], bolts = ['AB','GG', 'DD', 'BC']
- 输出：
  - 一种可能输出： nuts = ['ab','bc','dd','gg'], bolts = ['AB','BC','DD','GG']

分析：

- 如果nuts之间可以比较，bolts之间可以比较，那么就可以分别快速排序。只能比较nuts和bolts，就可以利用一方排序另一方
- 先用bolts中的任意一个，B，去将nuts分成三部分：
  - 小于B的nuts；正好对应B的nut；大于B的nuts
- 然后用这个中间的nut，N，去将bolts分成三部分：
  - 小于N的bolts；B；大于N的bolts
- 分别递归
- 平均时间复杂度：O(nlogn)

编程

Iterator Problem

Iterator is Non-recursion

必须要用的一个数据结构是什么？

栈

Flatten List

http://www.lintcode.com/problem/flatten-list/

http://www.jiuzhang.com/solutions/flatten-list/

LintCode 22

九章算法

- 给定一个列表，该列表中的每个要素要么是列表，要么是整数
- 要求将其扁平化，即变成一个只包含整数的简单列表
- 要求不使用递归

- 例子：
- 输入：
  - [4,[3,[2,[1]]]]
- 输出：
  - [4,3,2,1]

分析：

- 如果可以使用递归，遇到整数就塞入结果，遇到list就递归

- 不使用递归，可以用栈模拟
  - 遇到整数就输出，遇到list就将本层iterator塞入栈，然后处理list的iterator

```
public List<Integer> flatten2(List<NestedInteger> nestedList) {
        Iterator<NestedInteger> cur = nestedList.iterator();
        List<Integer> res = new ArrayList<>();
        Stack<Iterator<NestedInteger>> stack = new Stack<>();
        while(!stack.isEmpty() || cur.hasNext()){
            if(cur.hasNext()){
                NestedInteger ni = cur.next();
                if(ni.isInteger()){
                    res.add(ni.getInteger());
                }else{
                    stack.push(cur);
                    cur = ni.getList().iterator();
                }
            }else{
                cur = stack.pop();
            }
        }
        return res;
    }
```

Flatten Nested List Iterator

http://www.lintcode.com/problem/flatten-nested-list-iterator/
http://www.jiuzhang.com/solutions/flatten-nested-list-iterator/

LintCode 528

九章算法

- 给定一个列表，该列表中的每个要素要么是列表，要么是整数
- 要求输出一个扁平化后的iterator，即如果不停调用这个iterator的getNext(), 返回只包含整数的简单列表
- 要求不使用递归

- 例子：
- 输入：
  - [4,[3,[2,[1]]]]
- 输出：（反复调用iterator）
  - [4,3,2,1]

分析：

- 和Flatten List想法类似， 使用栈
- 先将List里的元素倒序放入栈，即List第一个元素在栈顶
- 每次调用getNext时
  - 遇到整数输出结果并pop
  - 遇到list就倒序放入栈，继续，直到遇到整数

Flatten 2D Vector

http://www.lintcode.com/problem/flatten-2d-vector/
http://www.jiuzhang.com/solutions/flatten-2d-vector/

LintCode 601

九章算法

- 给定一个二维列表
- 要求输出一个扁平化后的iterator，即如果不停调用这个iterator的getNext(), 返回只包含整数的简单列表
- 要求不使用递归

- 例子：
- 输入：
  - [[1,2], [3], [4,5,6]]
- 输出：（反复调用iterator）
  - [1, 2, 3, 4, 5, 6]

分析：

- 存储Input List的iterator i
- i每次后移一位，代表处理下一个一维链表
- 元素的iterator为j
  - j一开始指向i指向链表的第一个元素
  - 每次如果j是null，就后移i，j指向i所指链表的第一个元素
- 类似二维for loop

Binary Search Tree Iterator

http://www.lintcode.com/problem/binary-search-tree-iterator/

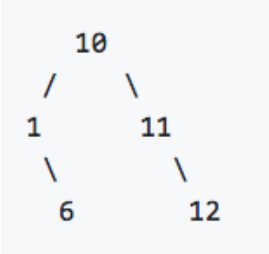
http://www.jiuzhang.com/solutions/binary-search-tree-iterator/

LintCode 86

九章算法

- 给定一个二叉搜索树
- 要求输出一个扁平化后的iterator，即如果不停调用这个iterator的getNext(), 返回整棵树中数字，按照从小到大的顺序，即按照树的中序遍历
- 要求不使用递归，hasNext和next的平摊时间复杂度O(1)

- 例子：
- 输入：



- 输出：（反复调用iterator）
  - [1, 6, 10, 11, 12]

分析：

- 用栈模拟中序遍历dfs
- 首先将root，root的左儿子，root的左儿子的左儿子，…依次放入栈
- 每次输出栈顶p
  - 如果栈顶p有右儿子r，将r，r的左儿子，r左儿子的左儿子，…依次放入栈
  - 如果栈顶p没有右儿子，则不停pop栈顶，直到栈为空，或者刚pop的元素是新任栈顶的左儿子

编程

Follow Up 常见方式

九章算法

- 一维转二维
  - 可以套相同的思路试一试
    - Trapping Water I/II
    - Subarray Sum/Submatrix Sum
- 数组变成循环数组
  - 循环数组小技巧
    - Continuous Subarray Sum
- 题目条件加强
  - 可能题目的解题方法会变化
    - Wiggle Sort I/II

Follow Up 常见方式

九章算法

- 换马甲(变一个描述，本质不变)
  - 本质不变
    - Number of airplanes on the Sky/ Meeting Room
    - BackPack Problem

- 描述完全不一样，但是方法相同
  - 这种题目得去分析
    - Quick Sort/ Bolts and Nuts Problem

课程总结

1. 透析热门IT公司中的FollowUp面试题
2. 数据结构（上）—— Union Find, Trie
3. 数据结构（下）—— Heap, Deque, 单调 Stack
4. 二分法第四层境界 + 扫描线算法
5. 动态规划（上）——滚动数组，划分、博弈、区间型动态规划
6. 动态规划（下）——双序列型动态规划，背包动态规划
7. 如何解决困难的 Follow Up 问题 —— Iterator, Subarray Sum, Wiggle Sort

课程总结

- 谢谢大家！
- 祝各位同学面试顺利，拿到自己理想的Offer
- 如果您喜欢这门课程，请推荐给您的朋友
- 希望大家参与课程反馈，给侯老师留下宝贵的建议和意见！
  - 课程问卷调查
  - https://www.jiuzhang.com/course/5/questionnaire/?term=476