九章独家《Amazon OA 九题》答案详解

1.最大子树

1.1.题目描述：

给你一棵二叉树，找二叉树中的一棵子树，他的所有节点之和最大。
返回这棵子树的根节点。（数据保证有且仅有唯一的解。）

1.2.样例：

给出如下二叉树

```
     1
   /   \
 -5     2
 /\    /  \
0   3 -4  -5
```

返回值为 3 的节点。

1.3.题目链接：

http://www.lintcode.com/zh-cn/problem/maximum-subtree/

1.4.答案链接：

http://www.jiuzhang.com/solutions/maximum-subtree/

1.5.思路解析：

求最大的子树，需要把所有的子树找出来，把他们的子树节点的和算出来，找一个最

大的即可。

如何实现这个过程呢？我们可以从分治的角度考虑。把当前子树所包含的所有子树分为三个部分：

1. 左子树中包含的所有子树
2. 右子树中包含的所有子树
3. 以当前节点为根节点的子树

计算当前节点为根节点的子树需要左右子树提供什么呢？子树和。有了左子树的子树和，右子树的子树和，然后再加上当前节点的值即可以算出当前子树的子树和。然后在 左子树的最大子树，右子树的最大子树，当前子树 中取最大的那一个子树即可。

如果只使用使用分治，就需要使用ResultType，因为需要返回三个值，（当前子树和，最大子树的根节点，最大子树的子树和）

一般这种情况下，都可以考虑能不能使用遍历来优化代码，一般做法是使用全局变量来避免使用ResultType。在这题中，因为最大子树的根节点 和 最大子树的子树和 这两个变量其实和位置和没有关系的，就可以考虑设定两个全局变量，一个是最大子树的根节点，一个是最大子树的子树和。然后每次当前节点的子树 和 最大子树比较即可。

1.6.面试官角度分析：

能够正确的使用分治和遍历的思想来分析问题并且写出正确的代码就能够达到hire。

1.7.相关题目：

http://www.lintcode.com/zh-cn/problem/binary-tree-maximum-path-sum/

2.最长回文串

2.1.题目描述：

给出一个包含大小写字母的字符串。求出由这些字母构成的最长的回文串的长度是多少。
数据是大小写敏感的，也就是说，"Aa" 并不会被认为是一个回文串。（假设字符串的长度不会超过 1010。）

2.2.样例：

给出 s = "abcccdd" 返回 7
一种可以构建出来的最长回文串方案是 "dccaccd"。

2.3.题目链接：

http://www.lintcode.com/zh-cn/problem/longest-palindrome/

2.4.答案链接：

http://www.jiuzhang.com/solutions/longest-palindrome/

2.5.思路解析：

回文串，很显然左右对称，那么也就是说如果这个回文串是偶数长度，那么一定是所有的元素出现的次数都是偶数，如果是奇数长度，那么只有最中间的字符出现的次数不是偶数次。所以解法就是，对于每一个元素，我们可以先求出在给定串中出现的次数num，在回文串中出现的次数是不大于num的最大的偶数sum1（不考虑奇数中间的情况），然后累加sum1，得到ans，最后如果有元素出现过奇数个数的话，那么我们最后把其加入到回文串的中间，也就是ans++。

2.6.面试官角度分析：

此题回文是通过奇数偶数构造回文，如果想出这种快速构造的方法就可以达到hire。

2.7.相关题目：

http://www.lintcode.com/problem/valid-palindrome

3.矩形重叠

3.1.题目描述：

给定两个矩形，判断这两个矩形是否有重叠。
l1代表第一个矩形的左上角
r1代表第一个矩形的右下角
l2代表第二个矩形的左上角
r2代表第二个矩形的右下角
保证：l1 != r2 并且 l2 != r2

3.2.样例：

给定 l1 = [0, 8], r1 = [8, 0], l2 = [6, 6], r2 = [10, 0], 返回 true
给定 l1 = [0, 8], r1 = [8, 0], l2 = [9, 6], r2 = [10, 0], 返回 false

3.3.题目链接：

http://www.lintcode.com/zh-cn/problem/rectangle-overlap/

3.4.答案链接：

http://www.jiuzhang.com/solutions/rectangle-overlap/

3.5.思路解析：

首先，两个矩形重叠会有很多种重叠方式，可能一个矩形包含另一个矩形的一个顶点，两个顶点，三个顶点，四个顶点，甚至重叠部分不包含任意一个顶点，所以如果我们要讨论重叠的话需要讨论很多方面，有一种说法叫做正难则反，所以我们可以直接反过来考虑不重叠的情况。其实不重叠的情况只有四种

1.第二个矩形在第一个矩形的左边 r1.x < l2.x
2.第二个矩形在第一个矩形的右边 l1.x > r2.x
3.第二个矩形在第一个矩形的上边 r1.y > l2.y
4.第二个矩形在第一个矩形的下边 l1.y < r2.y

3.6.面试官角度分析：

能够进行反向思考简化判断过程就可以达到hire。

3.7.相关题目：

http://www.lintcode.com/zh-cn/problem/max-points-on-a-line/

4.滑动窗口内数的和

4.1.题目描述：

给你一个大小为n的整型数组和一个大小为k的滑动窗口，将滑动窗口从头移到尾，输出从开始到结束每一个时刻滑动窗口内的数的和。

4.2.样例：

对于数组 [1,2,7,8,5] ，滑动窗口大小k= 3 。

1 + 2 + 7 = 10
2 + 7 + 8 = 17
7 + 8 + 5 = 20
返回 [10,17,20]

4.3.题目链接：

http://www.lintcode.com/zh-cn/problem/window-sum/

4.4.答案链接：

http://www.jiuzhang.com/solutions/window-sum/

4.5.思路解析：

这题我们可以维护一个数组存储前缀和(从0到当前位置所有元素的和)，用sum[pos]表示pos位置的前缀和，维护的方法就是令sum[0] = nums[0]，令sum[i] = sum[i-1] + nums[i]，这样i从0~n-1循环一次，就能达到sum[i]表示从0到i位置所有元素和的维护效果。这样计算索引i到j的和就是sum[j] - sum[i-1]。我们需要求的是所有连续k个元素的和，也就是说我们需要求的是所有sum[i] - sum[i-k]的集合，那么我们只需要把i从k移动到n-1，就可以得到答案。

4.6.面试官角度分析：

懂得用前缀和的方式在O（n）的复杂度下求出所有解的答案是此题的hire程度。

4.7.相关题目：

http://www.lintcode.com/problem/sliding-window-median
http://www.lintcode.com/problem/sliding-window-maximum
http://www.lintcode.com/problem/minimum-window-substring

5.安排课程

5.1.题目描述：

你需要去上n门九章的课才能获得offer，这些课被标号为 0 到 n-1 。
有一些课程需要“前置课程”，比如如果你要上课程0，你需要先学课程1，我们用一个匹配来表示他们： [0,1]
给你课程的总数量和一些前置课程的需求，返回你为了学完所有课程所安排的学习顺序。

可能会有多个正确的顺序，你只要返回一种就可以了。如果不可能完成所有课程，返回一个空数组。

5.2.样例：

给定 n = 2, prerequisites = [[1,0]]
返回[0,1]
给定 n = 4, prerequisites = [1,0],[2,0],[3,1],[3,2]]
返回 [0,1,2,3] or [0,2,1,3]

5.3.题目链接：

http://www.lintcode.com/zh-cn/problem/course-schedule-ii/

5.4.答案链接：

http://www.jiuzhang.com/solutions/course-schedule-ii/

5.5.思路解析：

很自然可以想到，如果一门课没有任何的先修课程，那我就可以先学这门课程。接下来的一步非常重要，我们要移除这门课对其他课的影响，也就是学了这门课我就可以解锁把这门课当先修课程的课了。然后接着找没有任何先修课程的课。如此循环。
为了解决这个问题，引入图论中度的概念，度分为入度和出度，入度是连向我的点有多少个，出度是我连向的点有多少个。这里我们只需要入度，我们把“B的先修课程是A”当做是“A连向B的一条有向边”。并且把B的入度加1。这里入度的具体含义是我修这门课的时候有几门先修课程。
所以具体的算法就是：

1. 初始化每个节点(也就是每门课程)的入度
2. 找一个入度为0的节点u。如果能找到，转到3，否则，转到4
3. 把u连向的所有节点入度-1,转到2
4. 判断课程有没有修满。如果修满了，返回结果。如果没有修满，返回空数组。

上述算法就叫做拓扑排序。这解决这一类问题的通用方法。

5.6.面试官角度分析：

能够使用拓扑排序的做法解出此题就可以达到hire。

5.7.相关题目：

http://www.lintcode.com/en/problem/course-schedule/
http://www.lintcode.com/en/problem/sequence-reconstruction/

6.优秀成绩

6.1.题目描述：

每个学生有两个属性 id 和 scores。找到每个学生最高的5个分数的平均值。

6.2.样例：

给出 results = [[1,91],[1,92],[2,93],[2,99],[2,98],[2,97],[1,60],[1,58],[2,100],[1,61]]
返回：
1: 72.40
2: 97.40

6.3.题目链接：

http://www.lintcode.com/zh-cn/problem/high-five/

6.4.答案链接：

www.jiuzhang.com/solutions/high-five/

6.5.思路解析：

因为题目需要的是每个学生最高的5个分数的平均值。这个过程可以分为两步，首先，求出这个学生最高的5个分数，然后求平均值。所以我们可以使用Map对id和5个最高的分数之间做一个映射。因为需要的分数只有5个，这里选用一个比较简单的实现方法。直接使用数组存储前5个数，后面的数如果比这5个数中最小的那个数大的话，就替代5个数中最小的那个数。当然，你可以使用优先队列等方法来实现这个过程。

6.6.面试官角度分析：

想到正确的解法并且写出来就可以达到hire。

6.7.相关题目：

http://www.lintcode.com/zh-cn/problem/top-k-largest-numbers/

7.K个最近的点

7.1.题目描述：

给定一些 points 和一个 origin，从 points 中找到 k 个离 origin 最近的点。按照距离由小到大返回。如果两个点有相同距离，则按照x值来排序；若x值也相同，就再按照y值排序。

7.2.样例：

给出 points = [[4,6],[4,7],[4,4],[2,5],[1,1]], origin = [0, 0], k = 3
返回 [[1,1],[2,5],[4,4]]

7.3.题目链接：

http://www.lintcode.com/zh-cn/problem/k-closest-points/

7.4.答案链接：

http://www.jiuzhang.com/solutions/k-closest-points/

7.5.思路解析：

求前k大的数，有很多种思路，最直接的方法就是先把所有的点离原点的距离求出来，然后按距离的大小进行排序，得出前k个。那么另一种思路就是维护一个最大容量为k的优先队列，每次将当前点加入队列，如果此时队列容量大于k，就弹出队头的元素，最后队内剩下的就是答案所求的。

7.6.相关题目：

http://www.lintcode.com/problem/k-closest-numbers-in-sorted-array

8.复制带随机指针的链表

8.1.题目描述：

给出一个链表，每个节点包含一个额外增加的随机指针可以指向链表中的任何节点或空的节点。
返回一个深拷贝的链表。

8.2.题目链接：

http://www.lintcode.com/zh-cn/problem/copy-list-with-random-pointer/

8.3.答案链接：

http://www.jiuzhang.com/solutions/copy-list-with-random-pointer/

8.4.思路解析：

可以将复制分为二个部分。
深复制链表的节点部分
深复制链表的随机指针部分
那么可不可以复制节点的时候将随机指针部分也复制过去呢?答案是不行。因为这个随机指针并不是绝对的位置，而是相对的位置。Node1.RamdomNode = Node2，那么结果应该是newNode1.RamdomNode = newNode2。如果直接复制过去的结果是newNode1.RamdomNode = Node2。
答案中给出了两种方法：
第一种方法是使用map对新的节点和旧的节点使用做一个对应关系，复制链表节点过后，再使用对应关系复制随机指针。
第二种方法做得非常的巧妙：首先让Node1->Node2变成Node1->newNode1->Node2->newNode2，这样做的目的首先是复制了节点，然后是做了对应关系，不过这个对应关系不是第一种方法的map，而是前面一个节点，就对应这个节点的下一个节点。同样是使用对应关系复制随机指针，最后使用两个dummy node将这个链表分开就可以得到答案。

8.5.面试官角度分析：
能够想到使用map作为对应关系就可以达到hire，如果能够想到第二种方法，就可以达到strong hire.

8.6.相关题目：

http://www.lintcode.com/zh-cn/problem/clone-graph/
http://www.lintcode.com/zh-cn/problem/clone-binary-tree/

9.最小生成树

9.1.题目描述：

给出一些Connections，即Connections类，找到一些能够将所有城市都连接起来并且花费最小的边。如果说可以将所有城市都连接起来，则返回这个连接方法；不然的话返回一个空列表。
（返回cost最小的连接方法，如果cost相同就按照city1进行排序，如果city1也相同那么就按照city2进行排序。）

9.2.样例：

给出 connections = ["Acity","Bcity",1], ["Acity","Ccity",2], ["Bcity","Ccity",3]
返回 ["Acity","Bcity",1], ["Acity","Ccity",2]

9.3.题目链接：

http://www.lintcode.com/zh-cn/problem/minimum-spanning-tree/

9.4.答案链接：

http://www.jiuzhang.com/solutions/minimum-spanning-tree/

9.5.思路解析：

最小生成树的概念是在一个无向图中，找出一些边，组成一棵树，使得这些边的权值加起来最小。
最小生成树是一类问题，有很多解决方法。这里我们介绍Kruskal算法。
Kruskal的逻辑非常简单。首先按照每条边的边权对边进行从小到大排序。然后依次判断每条边是否属于最小生成树。下面是具体步骤：
判断边的两个端点是否都已经在最小生成树中了，如果在，说明连这条边就会形成一个环，所以就不能连这条边。如果至少有一方不在最小生成树中，就把这条边加入最小生成树中。
判断两个端点是否在一个集合的方法叫做并查集。
并查集的具体做法见下面博客：
http://blog.csdn.net/dm_vincent/article/details/7655764

9.6.面试官角度分析：

使用Kruskal解决最小生成树问题就可以达到hire。

9.7.**相关题目：(以下题目都使用并查集算法)**

http://www.lintcode.com/zh-cn/problem/connecting-graph/
http://www.lintcode.com/zh-cn/problem/connecting-graph-ii/
http://www.lintcode.com/zh-cn/problem/connecting-graph-iii/
http://www.lintcode.com/zh-cn/problem/number-of-islands-ii/