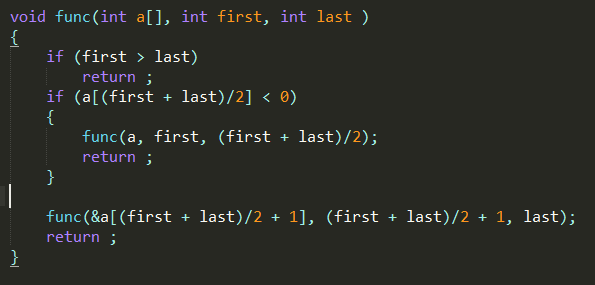
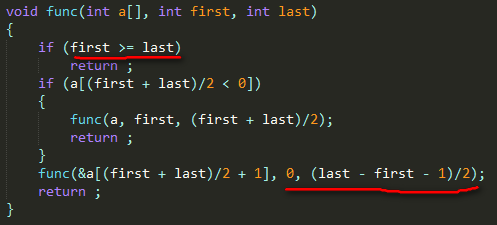
**数据结构2017试题**

**By 赵楠**

1. **一个函数具有下列形式，判断递归调用是否正确，如果不正确，应该如何修改？分析该函数的平均时间复杂度，以O的方式给出，给出分析过程。**



分析：本题就是一个二分调用的过程，不过并未实现什么功能，同时代码存在一些问题，现做修改如下



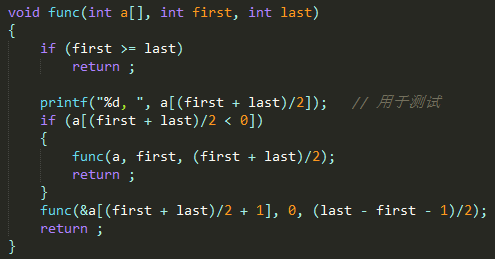
解释：

1. first >= last，不难理解在后面的调用中first <= last总是成立，所以很难终止，故修改为first >= last。
2. 第二处修改，由于数组指针已经修改，这里会超界，举例如下：

假设 a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}，first = 0, last = 9, 那么原始代码会执行func(&a[5], 5, 9)，这时传入函数的数组头指针已经修改为a+5，而不是a，因此后面的调用a[(first+last)/2]之时就是 a+5+7，超过原始界限，其中a+5为传入函数的指针。因此需要修改first = 0，last = (last – first – 1)/2

注：读者可自行打印a[(first + last)/2]测试，笔者代码中测试了两种极限情况

主函数：

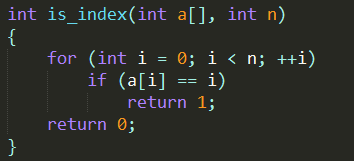


具体代码见：<https://github.com/tofar/data-structure/tree/master/数据结构复习卷>/2017\_1.c

1. **将数组 {32，145，632，88，55，25，135，132，65，13，165，335} 调整为最大堆并使用堆排序将其排为升序数组。要求用图形和数组表示过程。**
2. **编写C语言函数 int is\_index(int a[], int n) 判断升序排列的整数数组a（长度为n）中是否存在a[i] = i；如果存在，返回TRUE，否则返回FALSE。说明程序的时间复杂度是多少。**

题解：

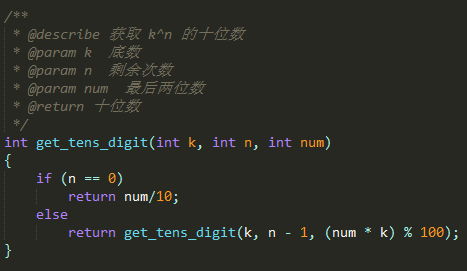
本题只需使用一个for循环遍历数组是否存在满足 a[i] == i条件的即可



具体代码见：<https://github.com/tofar/data-structure/tree/master/数据结构复习卷>/2017\_3.c

1. **对下列整数进行基数排序，{32，145，632，88，55，25，135，132，65，13，165，335}，要求标出所使用的基以及每一趟桶式排序中的元素变化过程。**
2. **用分治算法的思想计算3192的十位数是多少？说明算法的步骤，给出计算过程。**

本题较为简单只需使用一个递归即可解决，主要代码如下：



注：num最初始为1

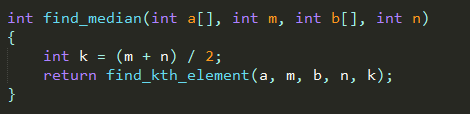
具体代码见：<https://github.com/tofar/data-structure/tree/master/数据结构复习卷/2017_5.c>

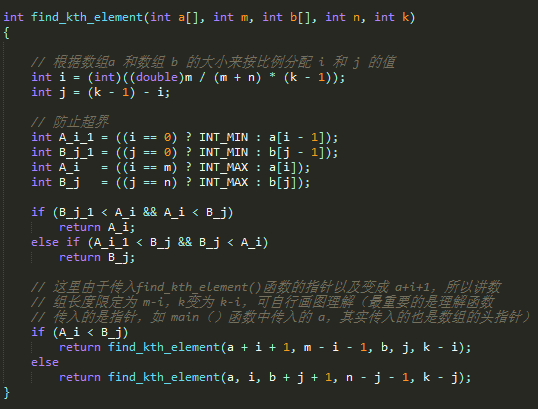
1. **利用分治算法的思想编写C语言函数int find\_median(int a[], int m, int b[], int n)，在长度分别为m和n的两个排序整型数组中找出总体的中位值，其中a为升序排列，b为降序排列。说明时间复杂度。中位值定义为第**

**（m+n）/ 2 小的元素。**

本题参照我的2016年的第六题题解即可，只是在开始的时候计算时，取k = （m+n）/ 2，其他一模一样

主要代码如下：





具体代码见：<https://github.com/tofar/data-structure/tree/master/数据结构复习卷>/2017\_6.c

1. **已知一棵二叉树的先序遍历结果为ABCDEFGHI，中序遍历结果为CEDBAGFHI，结点由字母表示。这棵树是否存在？如果存在，请构造出这棵树并标出构造过程。如果不存在，请说明为什么。**

不存在，

原因如下：

1. 先序遍历第一个字母为A，中序遍历的第一个字母为C，故可以确定最深的树序列为 A->B->C (左子树)，由中序遍历可知C的父节点为E，与前面的矛盾（C的父节点为B），故不存在。
2. **一组符号Si，i = 1 .. 12，其出现的频率分别是3，14，6， 8，15，25，35，12，65，13，33和5。请设计出相应的Huffman编码。要求画出Huffman树，并给出编码。**

哈夫曼编码过程如下：

1. 首先排序如下：3, 5, 6, 8, 12, 13, 14, 15, 25, 33, 35, 65
2. 第一次合并： 6, 8, \*8(3+5), 12, 13, 14, 15, 25, 33, 35, 65
3. 第二次合并： \*8(3+5), 12, 13, 14, \*14(6+8), 15, 25, 33, 35, 65
4. 第三次合并： 13, 14, \*14(6+8), 15, \*20(8+12), 25, 33, 35, 65
5. 第四次合并： \*14(6+8), 15, \*20(8+12), 25, \*27(13+14), 33, 35, 65
6. 第五次合并： \*20(8+12), 25, \*27(13+14), \*29(14+15), 33, 35, 65
7. 第六次合并： \*27(13+14), \*29(14+15), 33, 35, \*45(20+25), 65
8. 第七次合并： 33, 35, \*45(20+25), \*56(27+29), 65
9. 第八次合并： \*45(20+25), \*56(27+29), 65, \*68(33+35)
10. 第九次合并： 65, \*68(33+35), \*101(45+56)
11. 第十次合并： \*101(45+56), \*133(65+68)
12. 第十一次合并： \*234（101+133）

Huffman树如下（其中数字代表频率）：



哈夫曼编码：

|  |  |  |  |
| --- | --- | --- | --- |
| 符号 | 代码 | 频率 | 总比特数 |
| S1 | 00000 | 3 | 15 |
| S2 | 0101 | 14 | 56 |
| S3 | 01100 | 6 | 30 |
| S4 | 01101 | 8 | 40 |
| S5 | 0111 | 15 | 60 |
| S6 | 001 | 25 | 75 |
| S7 | 111 | 35 | 105 |
| S8 | 0001 | 12 | 48 |
| S9 | 10 | 65 | 130 |
| S10 | 0100 | 13 | 52 |
| S11 | 110 | 33 | 99 |
| S12 | 00001 | 5 | 25 |
| 总和 | 735 | | |

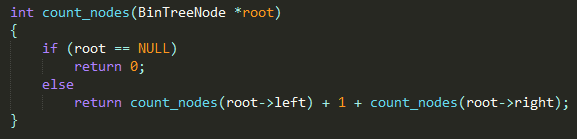
1. **编写一个C语言函数int count\_nodes(BinTreeNode \*root)返回以root为根的二叉树中节点的个数。**

题解：

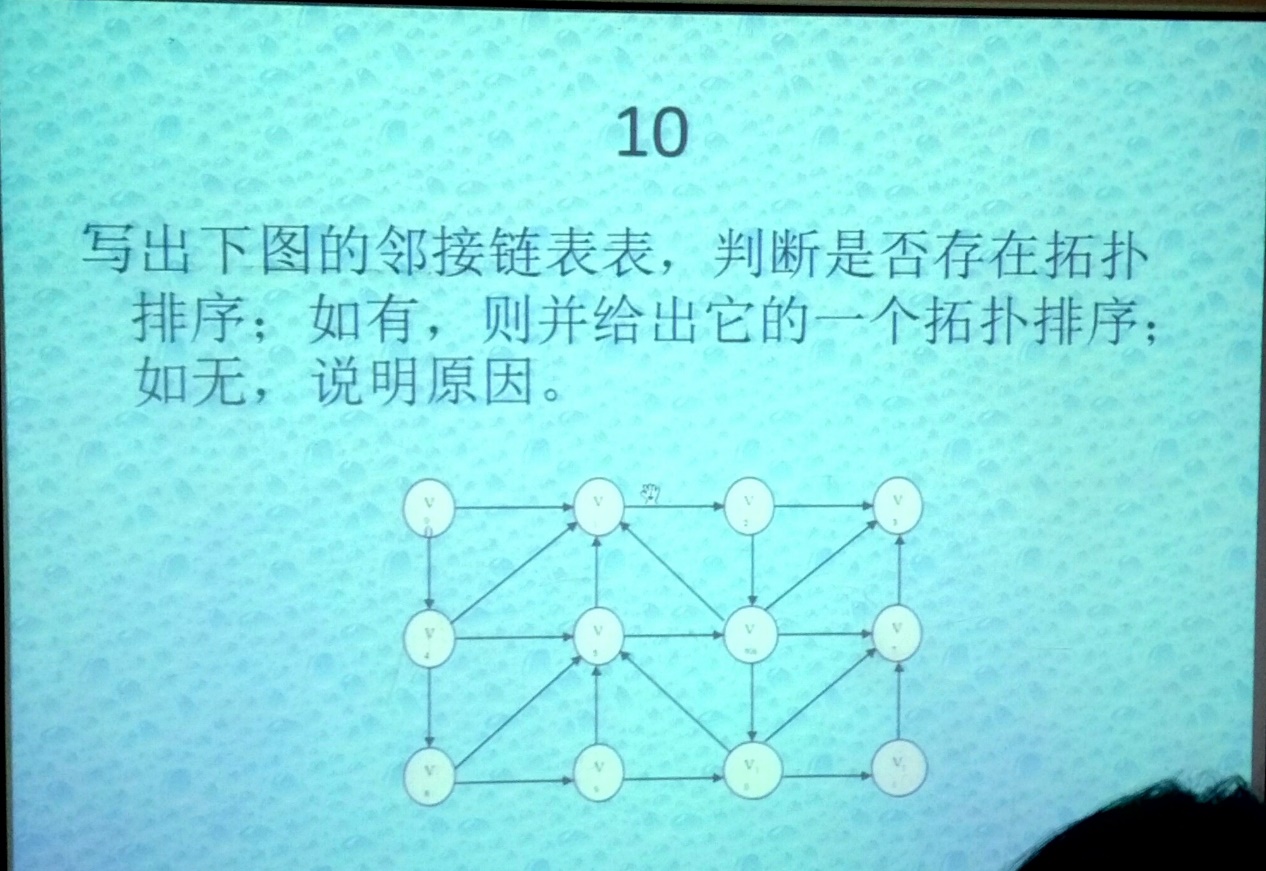
本题只需使用一个递归返回左子树和右子树的节点数再加 1即可

*count\_nodes(root->left) + 1 + count\_nodes(root->right)*

*代码如下：*



1. **写出下图的邻接链表表，判断是否存在拓扑排序；如果存在，则给出它的一个拓扑排序；否则，说明原因。**

****

题解：

参照课本的217页的拓扑排序

注：由于本人看不清那个顺序，这里直接按照从左到右从上到下标号，即

1 2 3 4

5 6 7 8

9 10 11 12

本题不存在一个拓扑排序，由于途中v2->v3->v7->v2构成一个圈，故不存在