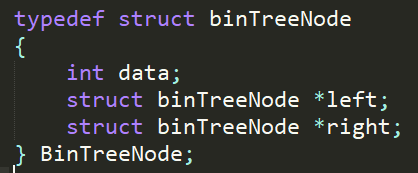
**数据结构2016试题**

**By 赵楠**

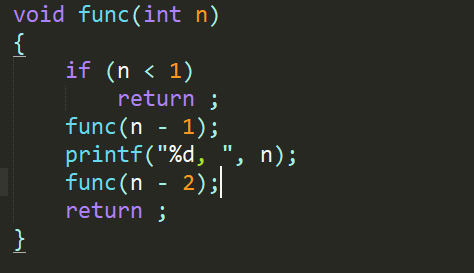
**说明：**

**本试卷中使用的数据结构如下：**



**每题10分，请合理安排时间，如果有编程困难，可以使用伪代码或者文字表达以得到部分分数。**

1. **一个函数具有如下形式，请写出func(5) 的执行结果，画出其递归调用树，并分析函数的计算时间复杂度范围，以 O 的方式给出，并给出分析过程。**



调用树：



时间复杂度分析：

此函数时间复杂度满足此式 F(n) = F(n-1) + F(n-2)，故由往期经验可知，这种类似与斐波那契数列的增长呈指数增长，此处姑且假定 F(n) = O(2^n)，证明如下：

使用方法：数学归纳法

（1）当n = 1时 F(n) = 1 满足条件

（2）假定对于 n = 1, 2, …, n成立，只需证明F(n+1) = O(x^(n+1))即可

F(n+1) = F(n) + F(n-1) = O(2^n) + O(2^(n-1)) = O(2^(n+1))

由（1）（2）可知假定成立，证毕

1. **在长度为11的Hash表中使用平方探测的开放地址法依次插入 12、28、 15、 82、 215、 88、 913，采用的Hash函数 形式为 hash(key) = key % 11。是否能将所有元素放入表中？画出插入过程。如果不信，说明原因。**

能将所有元素放入表中，插入过程如下：

（1）首先插入以下数字不会冲突：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | 12 |  |  | 15 | 82 | 28 |  |  |  |  |

（2）当插入215时，215 mod 11 = 6，而6那个位置已经被占了，于是I = 1，(215 + i^2) mod 11 = 7，所以插入7那个位置。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | 12 |  |  | 15 | 82 | 28 | 215 |  |  |  |

（3）插入88，直接插入0那个位置，没有冲突；

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 88 | 12 |  |  | 15 | 82 | 28 | 215 |  |  |  |

（4）插入913之时，由于 913 mod 11 = 0，而0已经被占，于是I = 1，（913 + i^2）mod 11 = 1，然而1已经被占，于是 i++，（913 + i^2）mod 11 = 4，然而4已经被占，于是i++，（913 + i^2）mod 11 = 9，无冲突，故插入9那个为位置，结束。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 88 | 12 |  |  | 15 | 82 | 28 | 215 |  | 913 |  |

1. **编写C语言函数void move(int a[], int n) 将整数数组 a （长度为n）中所有负数移到所有正数之前（0不需要移动），要求时间复杂度为O(n)，并且只是用常数长度的辅助存储。**

参考课本快速排序P177

题解：

具体如下：

1. 首先寻找枢纽元pivotValue，这里由于是区分正数和负数，只需寻找最靠近0的非负整数即可
2. 寻找完枢纽元之后只需对当前数组进行遍历，每次遍历的时候寻找左侧第一个小于pivotValue的数以及右侧第一个大于等于pivotValue的数（右侧从最右边开始按照降序寻找）
3. 之后交换寻找的两个数即可，这样当left < right（索引） 不满足的时候就完成了整个算法。

主函数如下：

void move(int a[], int n)

{

int pivotIndex = find\_pivot\_index(a, n);

int pivotValue = a[pivotIndex];

int left = 0, right = n - 1;

while (left < right) {

while (a[left] < pivotValue && left < right)

left++;

while (a[right] >= pivotValue && left < right)

right--;

swap(&a[left], &a[right]);

}

}

具体代码见：<https://github.com/tofar/data-structure/tree/master/数据结构复习卷/2016_3.c>

1. **将数组 2、8、12、3、15、26、7、9、23、16、18、22 分别调整为最大堆和最小堆，用图表示建堆过程。**
2. **只使用元素间的比较对一个由任意8个数组成的数组排序至少需要几次比较？为什么？如果使用归并排序算法，最多和最少需要几次比较？为什么？**
3. **编写C语言函数 int find\_kth\_element(int a[], int m, int b[], int n, int k)，在长度分别为m和n的两个已经按顺序排列的整型数组中找到总体上第k小的元素。要求时间复杂度为 O(log m + log n)。**

题解：

二分查找每次会把搜索空间折半，为了达到O(log m + log n)的复杂度，我们必须在每轮迭代时将A和B的搜索空间折半。

思路：

首先有如下结论：

*前提：i + j = k – 1*

*若 B[j-1] < A[i] < B[j], 那么 A[i] 就是第k小的元素,*

*若 A[i-1] < B[j] < A[i], 那么 B[j] 就是第k小的元素*

于是有了下面的方法：

1. 二分搜索数组A，i的初值: i = (int)((double)m / (m+n) \* (k-1)); 令j = k-1-i;

注：这里 i = (int)((double)m / (m+n) \* (k-1))，是为了按照数组a和数组b的大小的比例来分配i和j的初值。

2. 如果 B[j-1] < A[i] < B[j] 则返回A[i]， 如果 A[i-1] < B[j] < A[i] 则返回B[j];

3. 如果不满足步骤2中条件,则比较A[i]和B[j]

如果A[i]<B[j],则我们可以排除比A[i]小的元素和比B[j]大的元素,即在A[i+1, m]和B[0, j]中查找第k-i小的元素。

否则可以排除比B[j]小的元素和比A[i]大的元素,即在A[0, i]和B[j+1, n]中查找第k-j小的元素。

注：这里为了达到缩小数组查找范围的目的，每次可以将调用函数中数组指针的头指针位置移动，以及修改数组长度的参数，和k（和指针移动有关），具体看代码

函数如下：

int find\_kth\_element(int a[], int m, int b[], int n, int k)

{

// 根据 数组a 和数组 b 的大小来按比例分配 i 和 j 的值

int i = (int)((double)m / (m + n) \* (k - 1));

int j = (k - 1) - i;

// 防止超界

int Ai\_1 = ((i == 0) ? INT\_MIN : a[i - 1]);

int Bj\_1 = ((j == 0) ? INT\_MIN : b[j - 1]);

int Ai = ((i == m) ? INT\_MAX : a[i]);

int Bj = ((j == n) ? INT\_MAX : b[j]);

if (Bj\_1 < Ai && Ai < Bj)

return Ai;

else if (Ai\_1 < Bj && Bj < Ai)

return Bj;

if (Ai < Bj)

return find\_kth\_element(a + i + 1, m - i - 1, b, j, k - i);

else

return find\_kth\_element(a, i, b + j + 1, n - j - 1, k - j);

}

具体代码见：<https://github.com/tofar/data-structure/tree/master/数据结构复习卷/2016_6.c>

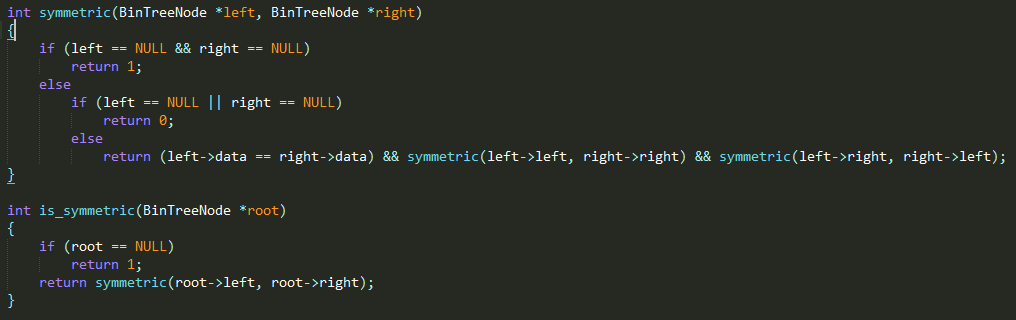
1. **一组符号 Si，I = 0 .. 11出现的频率分别是2、8、12、3、15、26、7、9、23、16、18、22，请设计其Huffman编码，计算平均编码长度。要求画出Huffman树，并列出编码结果。**
2. **使用基数排序对数组 {23，123，37，107，87，217，83，53，273，63，75，35} 排序，画出排序过程（每一趟桶式排序分配和回收的结果）。**
3. **编程一个C语言函数 int is\_symmetric(BinTreeNode \*root) 判断一棵以root为根的二叉树是否对称（左右子树完全相等），如果是则返回1，否则返回0。**

注：这里我觉得老师的题目定义不太好，写了对称，怎么有注释左右子树完全相等。。。。。。

题解：

本题先判断root的左右子树是否相等，再通过一个递归调用判断root的左右子树是否对称。

这里判断左右子树是否对称是通过判断左子树的左子树和右子树的右子树是否相等，左子树的右子树和右子树的左子树是否相等来判定。



具体代码见：<https://github.com/tofar/data-structure/tree/master/数据结构复习卷>/2016\_9.c

1. **用下列邻接矩阵表示一个无向图，请画出这个图。请画出使用Prim’ s算法得出该图的最小生成树的过程，标明每一步的依据。**