2015-2016 学年度第一学期软件学院 2014 级本科

《数据结构》课程考试试卷（B 卷）

专业、班级：_____ 姓名：_____学号：_____成绩：_____ 考试日期： 2016 年 1 月 24 日 考试时间： 150 分钟

题号	1	2	3	4	5	6	7	8	9	10
得分										

说明：

- 本试卷中使用的数据结构如下： typedef
  struct bINTREENODE
  {
  int data;
  struct bINTREENODE *leftChild;
  struct bINTREENODE *rightChild;
  } BINTREENODE;

- 每题 10 分，请合理安排时间。如果有编程困难，可以使用伪代码或文字表达以得到部

  分分数。

1. 一个递归函数具有如下形式

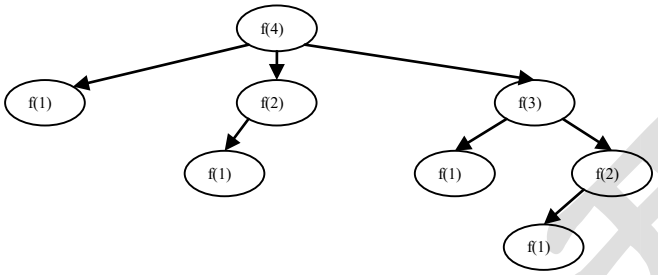
```
int func ( int x )
{
   int i, sum；

   if ( x < 0)
return (0);

   if (x == 1)
sum = 1;    else
      for (i = 1, sum = 0; i < x; i++)
sum += x*func (x);

      printf ( “%d,”, sum);
```

```
    return (sum);
}
```

请依次写出 func（1），func（2），func（3），和 func（4）的执行结果，画出递归调用树，并分析 func (n)的计算时间复杂度，以 O 的形式给出，要求给出分析过程。

func(1) = 1: 打印 1, func(2) = 1: 打印 1,1, func(3) = 3: 打印 1,1,1,3

func(4) = 12 : 打印 1,1,1,1,1,1,3,12



$T(n) = \sum_{i=0}^{n-1} T(i) + \Theta(n), T(1) = \Theta(1)$

带入年，得到

$T(n-1) = \sum_{i=0}^{n-2} T(i) + \Theta(n-1)$

两式相减，有

$T(n) - T(n-1) = T(n-1) + \Theta(1)$，即$T(n) = 2T(n-1) + \Theta(1)$

化简得到$T(n) = O(2^n)$

2，(3^200) % 5 =？写出你的计算依据和推导过程。

根据费马小定理，3^4 % 5 = 1, 而 5^200 = (5^4)^50，因此 3^200 % 5 = 1 % 5 = 1

3，编写一个 C 语言函数 int treeHeight (BINTREENODE *root)，得到并返回一棵以 root 为根节点的二叉树的高度，空树高度为-1。

```
int treeHeight (BINTREENODE *root)
{
      int h_l, h_r;

      if (root == NULL)
     return (-1);

      h_l = treeHeight (root->left);
  h_r = treeHeight (root->right);

      if (h_l < h_r)
        return (h_r + 1);

      return (h_l + 1);
}
```

4，构造一棵 n 个元素的最小堆的最坏时间复杂度用 O 表示是多少？证明你的结论。

O(n)

- 2 comparisons for an internal node, 1 for finding a smaller child, 1 with that child.
- # of moves in percolate_down = height of the node. Then we need to know the sum of heights of all internal nodes in a heap. This is between the value for a full (perfect) binary tree with height h-1 and the value for a full binary tree with height h.
- $S = \sum 2_i(h - i) = h + 2(h - 1) + \ldots + 2_{h-1}$ (1)
- $2S = 2h + 2(h - 1) + \ldots + 2*2_{h-1}$ (2)
- (2) – (1), we have $S = -h - 1 + 1 + 2 + \ldots + 2^h = (2^{h+1} - 1) - (h + 1) = n - (h + 1)$

因为 BST 节点插入为树叶，在空树中依次插入节点得到 BST 的代价等同于所有节点深度之和。令 n 个结点的 BST 的深度之和为 T(n)，则

T(n) = T(m) + T(n-1-m) + n-1，其中 m 为左子树节点个数，n-1-m 为右子树节点个数，等同于根节点令左右子树节点加深一层 对于平均情况，m 可以等概率取 0 到 n-1 中的任何值，对 n 种不同情况求和，有 $nT(n) = \sum_{m=0}^{n-1} T(m) + n(n-1)$

带入 n-1 的情况

$(n-1)T(n-1) = 2\sum_{m=0}^{n-2} T(m) + (n-1)(n-2)$

两式相减

nT(n) - (n-1)T(n-1) = 2T(n-1) + 2(n-1)

nT(n) = (n+1)T(n-1) + 2(n-1)，两边同时除以 n (n+1)，近似地

T(n) /(n+1) = T(n-1)/n + 2/ n

T(n-1) /(n) = T(n-2)/(n-1) + 2/(n-1)

……

T(1) = 1

利用调和序列求和与欧拉常数

T(n)=(n+1) ∑1/m=O(n ln n) =O(n log n)

5，设计并编写一个 C 语言函数 unsigned char isIdentical (int a[], unsigned int n)，判断给定的

长度为 n 的元素各不相同且已按升序排序的数组 a 中是否存在一个元素等于其索引值，即 a[i]=i，如果存在返回 1，否则返回 0。 要求算法的时间复杂度为 O(log n)。 分治算法，比较 a[mid] 和 mid ， 如果 a[mid] = mid，返回 1;否则如果 a[mid]< mid，在右侧找；否则在左侧找

```
unsigned char is_identical (int a[], int first,  int last)
{
   int mid = (first + last)/2;

   if (first > last)
      return (0);

   if (a[mid] == mid)
return (1);

   if (a[mid] < mid)
     return (is_identical (a, mid+1, last));
```

```
    return (is_identical (a, first, mid-1));
}

unsigned char isIdentical (int a[], unsigned int n)
{
    return (is_identical (a, 0, n-1));
}
```

6，对数组{510，88，26，16，2，100，110，777，77，92，17，78，217，18，97}进行以10 为基的基数排序。要求用图或表表示排序过程，并写出在每一趟中的回收结果。

个位

110 100						16	207 97 17 77	78 18 28	
10		92				26	7	8	
0	1	2	3	4	5	6	7	8	9

回收结果:10->100->110->92->26->16->7->77->17->97->207->8->28->18->78

十位

8 207 7 100	18 17 16 110 10	28 26					78 77		97 92
0	1	2	3	4	5	6	7	8	9

回收结果:100->7->207->8->10->110->16->17->18->26->28->77->78->92->97

百位

97 92 78 77 28 26 18 17 16 10 8 7	110 100	207							
0	1	2	3	4	5	6	7	8	9

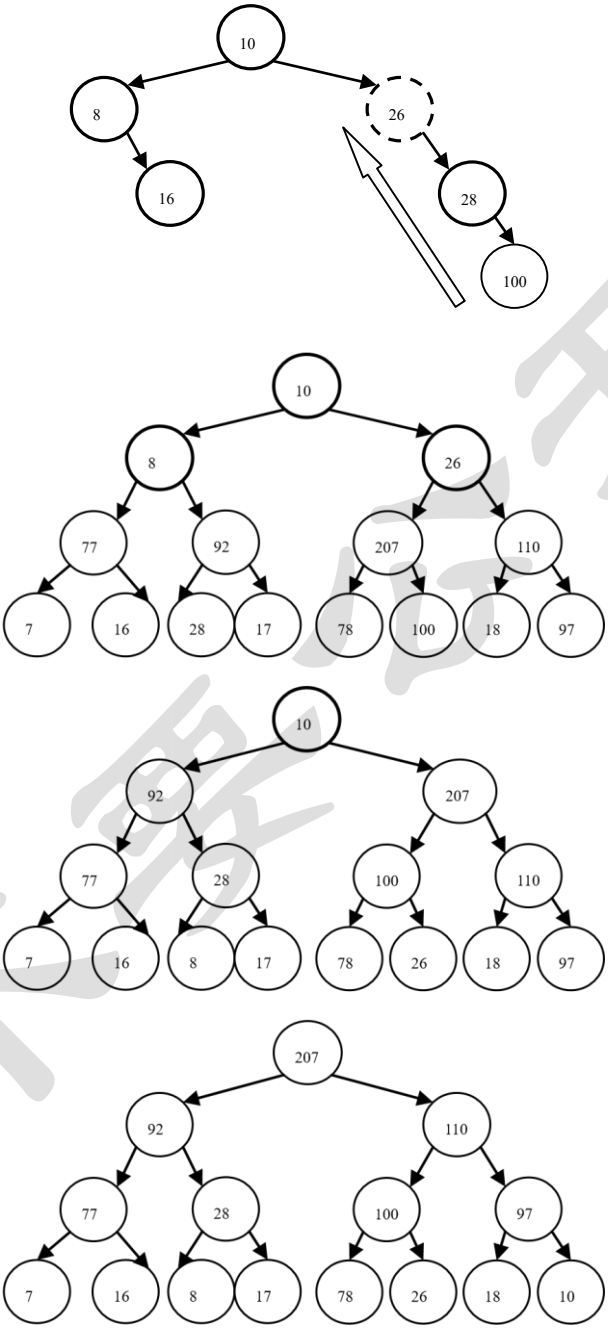
回收结果: 7->8->10->16->17->18->26->28->77->78->92->97->100->110->207

7，依次把下列数组中的元素插入一个初始为空树的 BST 树，要求用图表示调整为 AVL 树

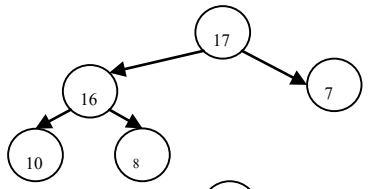
的过程。

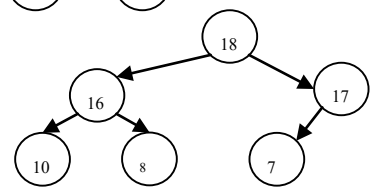
8	10	16	26	28	100	110	92	207	7	17	78	77	18	97

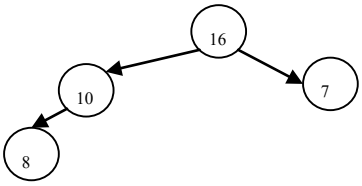
建堆过程 初始数组 树表示（加粗表
明需要调整的节点）

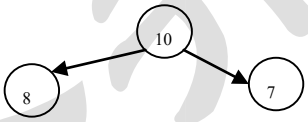


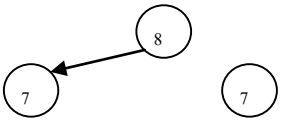
排序



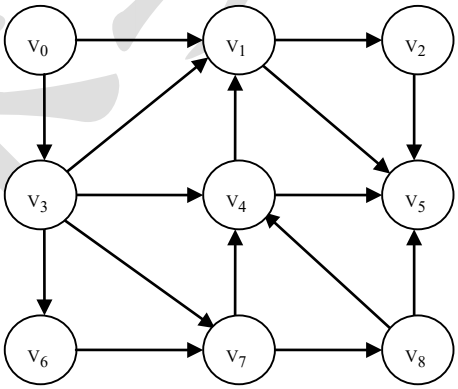








8，写出下图的邻接矩阵，给出它的一个拓扑排序。



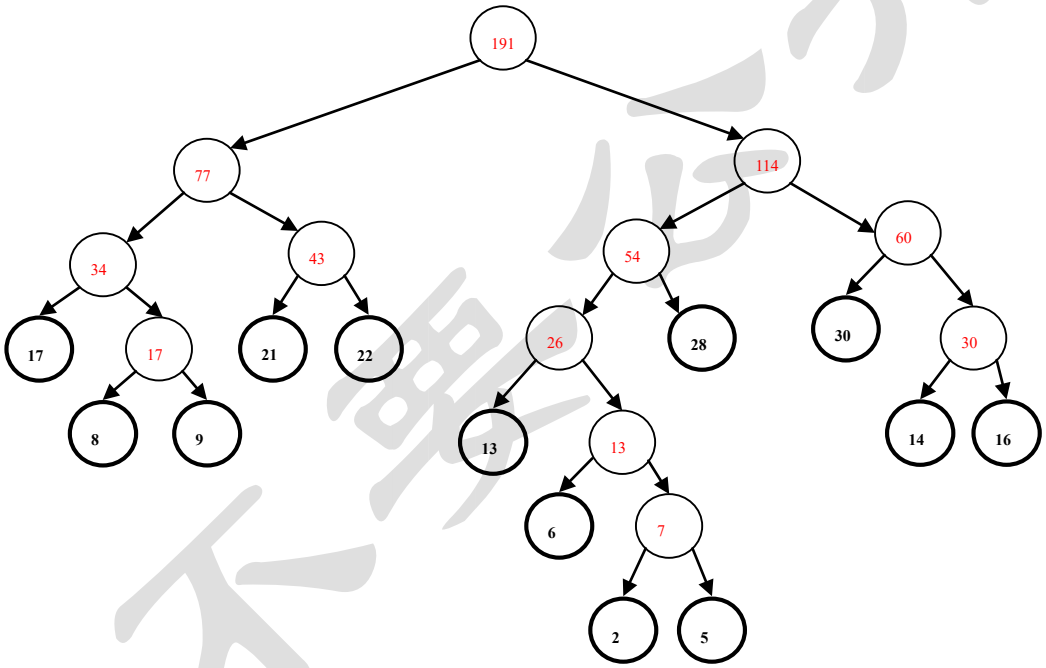
邻接矩阵

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}_{9\times 9}$$

拓扑排序： V0，V3，V6，V7，V8，V1，V2，V5

9，一组符号 S_i, i = 0..12,其出现的频率分别是 15，18，2，3，14，6，17，9，4，16，8， 30 和 21。请设计出相应的 Huffman 编码。要求画出 Huffman 树，并给出编码。编码可能不唯一。

Huffman 树



Huffman 编码(左 0 右 1)

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
100111	0010	011	1000	1110	10010	000	0011	100110	1111	101	110	010

10，设计一个算法，并编写 C 语言函数，从长度为 n 的未排序的数组 a 中同时挑选出最大值和最小值。要求算法的比较次数小于 2n，需要证明你的结果。 将数组 a 分成(n+k-1)/k 个长度不超过 k 的子数组，通过总共 n 次比较大小得出各自的最大值，共有(n+k-1)/k 个，如果(n+k-1)/k=n/k。

注意数组的长度可能是奇数，因此在起始/结束位置上应作处理（从 a[0]到 a[m/2-1]共有 m/2 个元素，而从 a[m/2]到 a[m-1]共有(m+1)/2 个元素）。

复杂度分析

最坏情况下 T(m,n)=T(m)+T(n/2)+ O(1), T(1,1)=O(1)（m 和 n 可以互换）；因此，经过 log n 次迭代得到 T(m,1)，这时的最坏情况为 T(m,n)=T(m /2)+ O(1)，经过 log m 次迭代得到 T(1,1)。总的时间复杂度为 O(log m + log n)。