4.2 单元检测

一、单项选择题

1.C 2.B 3.B 4.D 5.B 6.C 7.B 8.D 9.A 10.C 11.C 12.B 13.B 14.D 15.C 16.A 17.C

二、填空题

1. ABCDEFGH 2. 2^k-1 3. 68 4. n2+2n3+3n4+1 5. (n+1)/2 6. 2^{i-1}；2^k-1；2^{k-1}
7. O(n)；O(logn) 8. 完全二叉树；斜二叉树 9. h+k-1；(k^h-1)/(k-1) 10. 9

三、应用题

1.【解析】设二叉树中叶子结点、度为 1、度为 2 的结点数目分别 n0、 n1、n2，

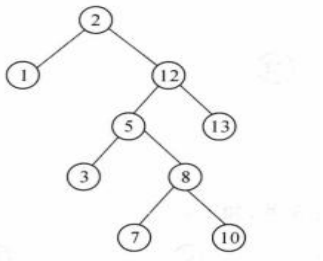
则有： n1 = 0 或 n1 = 1

n0 + n1 + n2 = 9999

n0 = n2 + 1

解方程组得：n0 = 5000, n1 = 0, n2 = 4999

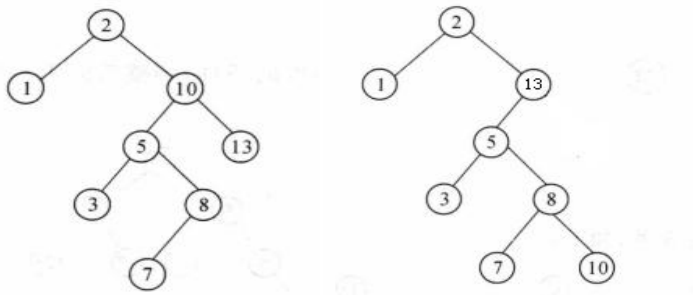
2.【解析】（1）构造的二叉排序树如下：



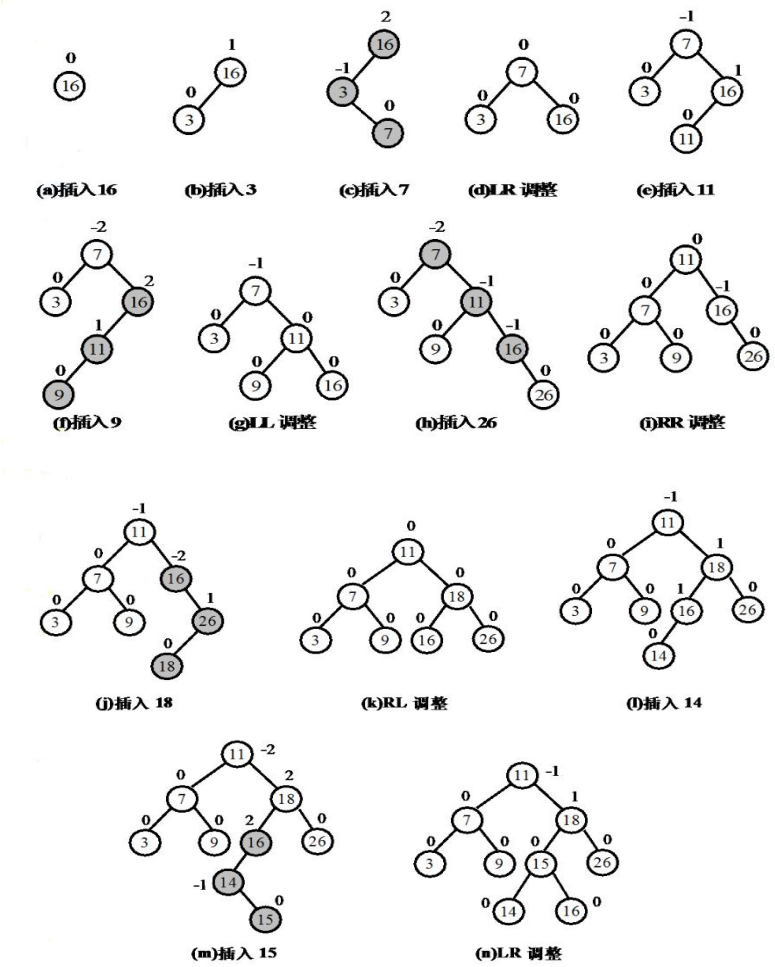
（2）中序遍历为：1，2，3，5，7，8，10，12，13

后序遍历为：1，3，7，10，8，5，13，12，2

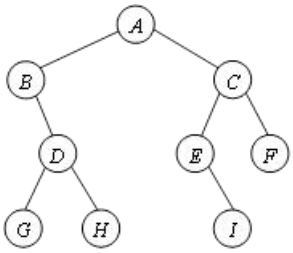
（3）删除“12”后的二叉排序树如下：



3.【解析】

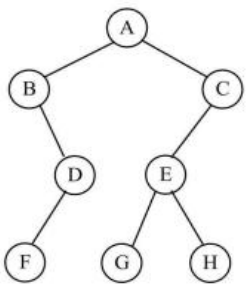


4.【解析】最后构造的二叉树如图所示。

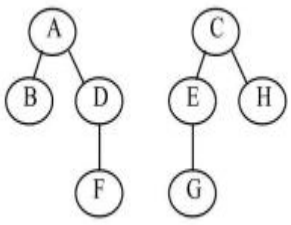


其后序遍历序列为 GHDBIEFCA。

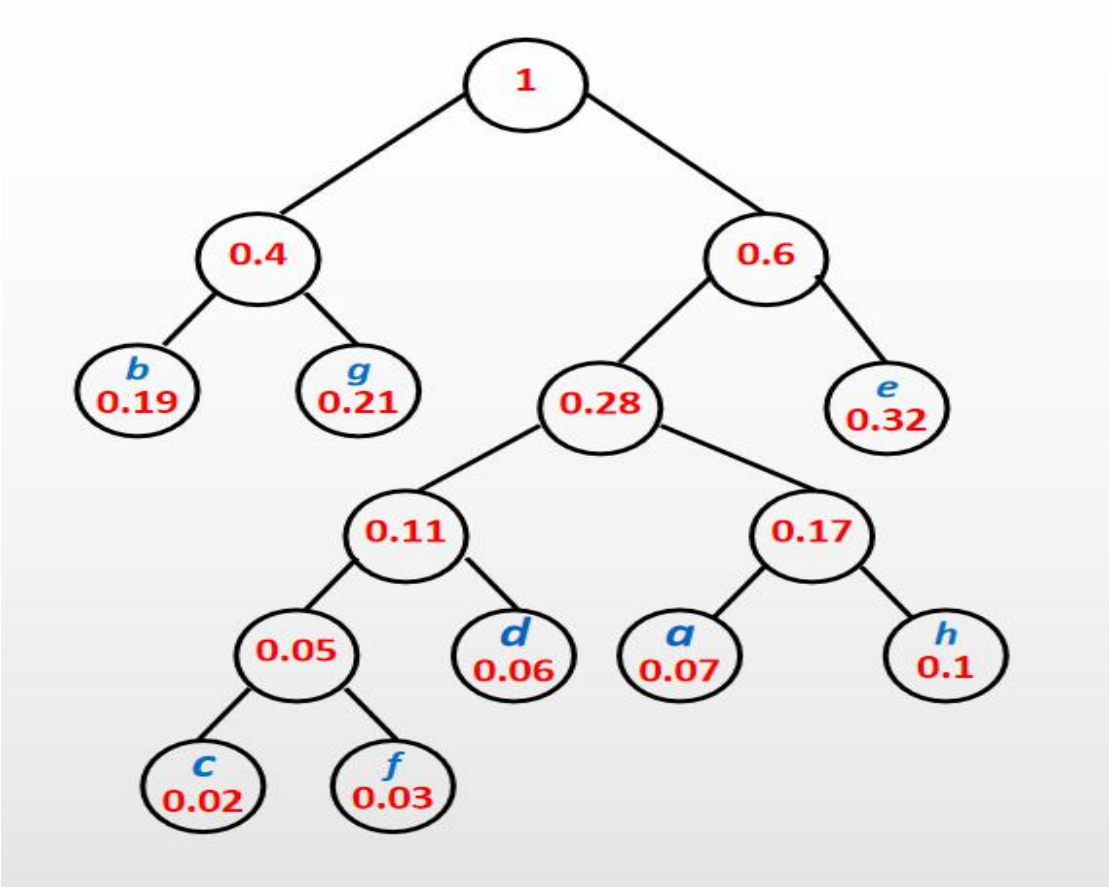
5.【解析】（1）



（2）



6. 【解析】（1）对应的哈夫曼树如下：



（2）对应的哈夫曼编码：

a：1010　b：00　c：10000　d：1001　e：11　f：10001　g：01　h：1011

7.【解析】（1）初始状态如图 1 所示，由于希望先输出最大值，故应调整为最大堆，调整后如图 2 所示。

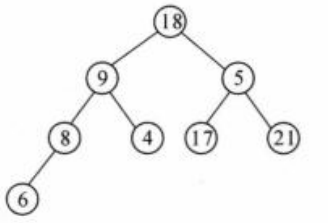


图 1

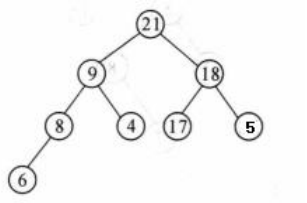


图 2

（2）输出最大值后．用最后一个元素值替换根节点，继续调整为最大堆，则此时根节点为次大值，如图 3 所示。

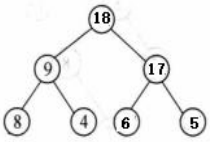


图 3

四、算法设计题

1.

```
int n1=0;
void preorder(bstnode *t)
{
    if(t == NULL)
        return;
    else
    {
        if((t->lchild == NULL && t->rchild != NULL)||
        (t->rchild == NULL && t->lchild != NULL))
            n1++;
        preorder(t->lchild);
        preorder(t->rchild);
    }
}
```

2.

```
BTNode *Findx(BTNode *b,char x)    //在二叉树 b 中查找值为 x 的结点
{   BTNode *p;
    if (b == NULL)
        return NULL;       //空树返回 NULL
    else
    {   if (b->data == x)
            return b;        //结点值等于 x，返回其地址
        p=Findx(b->lchild,x);
        if (p!=NULL)
            return p;
        return Findx(b->rchild,x);
    }
}
```

3.

```
int NodeCount( BTNode *BT )
{
    if( BT ) {
        return NodeCount(BT->lchild)+NodeCount(BT->rchild)+1;
    }
    else
        return 0;
}
```

4.

```
int level(BTNode *t,BTNode *p)        //在二叉排序树 t 中查找结点 p 所在的层次
{    int count = 0;
     if (t==NULL)
          return 0;
     else
     {    count++;
          while(t->data != p->data)
          {
               if(t->data < p->data)
                    t = t->rchild;
               else
                    t = t->lchild;
               count++;
            }
            return count;
     }
}
```

5.

```
void ChangeLR(BTNode &*T)    //交换二叉树中每个结点的左右孩子
{    BTNode *temp;
     if (T->lchild == NULL && T->rchild == NULL)
          return ;
     else
     {    temp = T->lchild;
          T->lchild = T->rchild;
          T->rchild = temp;
     }
     ChangeLR(T->lchild);
     ChangeLR(T->rchild);
}
```

6.

```
ElemType FindMax( BTNode *BST )
{
     if( BST )/*沿左分支一直向下，直到最左端点*/
          while( BST->lchild )
               BST = BST->lchild;
     return BST->data;
}
```