实验八　树及二叉树的应用实验

【实验目的】

（1）掌握哈夫曼树的建立及应用。

（2）掌握二叉排序树的建立及查找。

（3）掌握树的逻辑结构及其孩子兄弟链表存储结构。

【实验背景】

1、 哈夫曼树和哈夫曼编码

在具有 n 个叶子节点、且叶子节点的权值分别为 w_1，w_2，…，w_n 的所有二叉树中，带权路径长度 WPL 最小的二叉树被称为最优二叉树或哈夫曼树(Huffman tree)。

假设给定 n 个实数 w_1，w_2，…，w_n，构造拥有 n 个叶子节点的哈夫曼树，且这 n 个叶子节点的权值分别为给定的实数，则哈夫曼树的构造方法为：

⑴ 根据给定的 n 个实数，构造 n 棵单节点二叉树，各二叉树的根节点的权值分别为 w_1，w_2，…，w_n；令这 n 棵二叉树构成一个二叉树的集合 M。

这 n 棵单节点的二叉树中，这些节点既是根节点又是叶子节点。

⑵ 在集合 M 中筛选出两个根节点的权值最小的二叉树作为左、右子树，构造一棵新二叉树，且新二叉树根节点的权值为其左、右子树根节点权值之和；

⑶ 从集合 M 中删除被选取的两棵二叉树，并将新二叉树加入该集合；

⑷ 重复②、③步，直至集合 M 中只剩一棵二叉树为止，则该二叉树即为哈夫曼树。

由于一棵有 n 个叶子节点的哈夫曼树上共有 2n-1 个节点，可以采用长度为 2n-1 的数组顺序存储节点信息。每一个节点应包括 4 个域：存放该节点权值的 weight 域、分别存放其左右孩子节点在数组中下标的 lchild 域和 rchild 域，以及记录该节点的父节点信息的 parent 域。

这样，节点的类型描述为：

```
typedef struct{
      float weight;
   int parent，lchild，rchild;
}hufmtree;
```

若给定 n 个权值，则可定义数组 tree[ ]存储哈夫曼树上的节点：

```
hufmtree tree[2*n-1];
```

基于上述存储结构的哈夫曼算法分析如下：

⑴ 初始化数组 tree[2*n-1]；读入给定的 n 个权值，分别放入数组的前 n 个分量的 weight 域中，并将数组中所有分量的 lchild 域、rchild 域和 parent 域置 0；

⑵ 从数组的前 n 个分量中选择权值最小和次小的两个节点(假设下标分别为 p1 和 p2)合并，产生新节点，将新节点的信息存放在第 n+1 个分量中；新节点的权值 weight 为这两个节点的权值之和，左右孩子域中的值分别修改为 p1 和 p2；同时，改变下标为 p1 和 p2 节点的 parent 域中的值，使其等于 n+1；

⑶ 重复②，每次均从 parent 域的值为 0 的所有节点中选择权值最小和次小的两个节点合并，产生的新节点顺次存放在 weight 域值为 0 的分量中，同时修改该分量的左右孩子域值和被合并的两个节点的 parent 域值，直到数组的第 2n-1 个分量的 weight 域、lchild 域和 rchild 域中的值被修改为止。

在通信及数据传输中多采用二进制编码。为了使电文尽可能的缩短，可以对电文采用哈夫曼编码。

以电文中每个字符的概率值作为给定的权值，构造哈夫曼树。这样，哈夫曼树上的每个

叶子节点分别代表字符集 D 中的不同字符。然后约定哈夫曼树的所有左分支标记为 1，所有右分支标记为 0（或者所有左分支标记为 0，所有右分支标记为 1）；则从根节点到叶子节点的路径上所有分支标记将组成一个代码序列，该序列就是该叶子节点所对应的字符的编码。

2、 二叉排序树

二叉排序树或者是一棵空树，或者是具有如下性质的二叉树：

(1) 若其左子树非空，则左子树上所有节点的值均小于根节点的值；

(2) 若其右子树非空，则右子树上所有节点的值均大于根节点的值；

(3) 其左右子树也分别为二叉排序树。

我们可以使用二叉链表作为存储结构，其节点结构说明如下：

```
typedef struct node{
    keytype key；//关键字项
    datatype other；    //其它数据项
    struct node *lchild，*rchild；    //左右孩子指针
}Bstnode；
```

在二叉排序树中可以进行结点的查找和插入操作。二叉排序树的动态查找思想描述为：

(1) 若二叉排序树为空，则插入待查元素节点；

(2) 否则，将根节点关键字的值与待查关键字进行比较，若相等，则查找成功，若根节点关键字值大于待查值，则进入左子树重复此步骤，否则，进入右子树重复此步骤。

由于查找过程是从根节点开始逐层向下进行的，因此，容易写出该过程的非递归算法：

```
Bstnode *Bsearch(Bstnode *t，keytype x){
    Bstnode *p；int flag=0；
    p = t；
    while(p != NULL){
        if(p->key = = x){
            flag=1；return(p)；break；
        }
    if(x < p->key) p=p->lchild；
    else p = p->rchlid；
    }
  if(flag==0){
        printf(“ 找不到值为%x 的节点! ”，x)；
    return(NULL)；}
 }
```

二叉排序树的插入算法可在上述查找算法上修改得到。

【实验任务】

1、程序验证

（1）采用孩子兄弟表示法建立一棵树。

（2）基于孩子兄弟表示法存储的树实现前序遍历操作。

（3）阅读程序，写出程序功能，并通过运行验证之。

```
typedef struct{
    char data；
    CSTreeNode    *firstChild；
    CSTreeNode    *nextSibling；
```

```
} CSTreeNode;
int   dep(CSTreeNode   *root ) {
    if( !root )   return   0;
    else   {
            n1 = dep( root-> firstChild);
            n2 = dep( root-> nextSibling);
            return   max( n1+1, n2);
           }
}
```

（3）对给定的一组无序序列，建立一棵二叉排序树。

（4）对建立的二叉排序树实现查找操作。

2、算法填空

完善下述算法，并通过运行来验证：求给定结点在二叉排序树中所在的层数。

```
int   level(Bstnode   *root,   Bstnode   *p) {
    if( !p)   return   0;
    if( p = = root )   return   1;
    else     if ( p->key < root->key )
                return   _____ ;
             else   return   _____ ;
}
```

3、算法设计

（1）试写一算法判别给定的二叉树是否为二叉排序树，设此二叉树以二叉链表存储，且树中结点的关键字均不相同。

（2）以孩子兄弟链表作为存储结构，求树中结点 x 的第 i 个孩子。

（3）编写算法，在二叉排序树上找出任意两个不同结点的最近公共祖先。

（4）在二叉树中查找值为 x 的结点，试设计打印值为 x 的结点的所有祖先结点的算法。

提示：①对二叉树进行先序遍历，查找值为 x 的结点；

②找到值为 x 的结点时，栈中存放的是 x 的所有祖先结点。

4、实例演练：哈夫曼编码

[问题描述]

设某编码系统共有 n 个字符，使用频率分别为{ w1, w2, …, wn}，设计一个不等长的编码方案，使得该编码系统的空间效率最好。

[基本要求]

① 设计数据结构；

② 设计编码算法；

③ 分析时间复杂度和空间复杂度。

[实现提示]

利用哈夫曼编码树求得最佳的编码方案。利用哈夫曼算法建立哈夫曼树，在哈夫曼树中，设左分支为 0，右分支为 1，从根结点出发，遍历整棵哈夫曼树，求得各叶子结点所表示字符的哈夫曼编码。

[思维扩展]

对于采用哈夫曼编码树进行的编码，如何设计解码算法？