哈尔滨工业大学(深圳)2021 年春《数据结构》
第二次作业 树型结构

学号		姓名		成绩	

1、概念题

1.1 在二叉树的顺序存储结构中，实际上隐含着双亲的信息，因此可和三叉链表（含有父链指针）对应。假设每个指针域占 4 个字节，每个信息域占 k 个字节。试问：对于一棵有 n 个结点的二叉树，在顺序存储结构中最后一个节点的下标为 m，在什么条件下顺序存储结构比三叉链表更节省空间?

【参考答案】

采用三叉链表结构，需要 $n(k+12)$个字节的存储空间；

采用顺序存储结构，需要 mk 个字节的存储空间，则：

当 $mk<n(k+12)$时，即 $k<12n/(m-n)$时，采用顺序存储比采用三叉链表更节省空间。

1.2 对于二叉树 T 的两个结点 $n1$ 和 $n2$，我们应该选择二叉树 T 结点的前序、中序和后序中哪两个序列来判断结点 $n1$ 必定是结点 $n2$ 的祖先?

试给出判断的方法。(不需证明判断方法的正确性)

【参考答案】

此问题考查的知识点是二叉树的遍历及祖先的定义。

采用前序和后序两个序列来判断二叉树上结点 $n1$ 必定是结点 $n2$ 的祖先。在前序序列中某结点的祖先都排在其前。

若结点 $n1$ 是 $n2$ 的祖先，则 $n1$ 必定在问之前。而在后序序列中，某结点的祖先排在其后，即若结点 $n1$ 是 $n2$ 的祖先，则 $n1$ 必在 $n2$ 之后。根据这条规则来判断若结点 $n1$ 在前序序列中在 $n2$ 之前，在后序序列中又在 $n2$ 之后，则它必是结点叫的祖先。

1.3 一棵深度为 H 的满 k 叉树有如下性质：第 H 层上的结点都是叶子结点，其余各层上每个结点都有 k 棵非空子树。如果按层次顺序从 1 开始对全部结点编号，问：

(1)各层的结点数目是多少?

(2)编号为 p 的结点的父结点(若存在)的编号是多少?

(3)编号为 p 的结点的第 i 个儿子结点(若存在)的编号是多少?

(4)编号为 p 的结点有右兄弟的条件是什么?其右兄弟的编号是多少?

【参考答案】

(1)(k^H–1)/(k–1)

(2)如果 p 是其双亲的最小的孩子(右孩子)，则 p 减去根结点的一个结点，应是 k 的整数倍，该整数即为所在的组数，每一组为一棵满 k 叉树，正好应为双亲结点的编号。如果 p 是其双亲的最大的孩子(左孩子)，则 p+k-1 为其最小的弟弟，再减去一个根结点，除以 k，即为其双亲结点的编号。

综合来说，对于 p 是左孩子的情况，i=(p+k-2)/k;对于 p 是右孩子的情况，i=(p-1)/k，如果左孩子的编号为 p，则其右孩子编号必为 p+k-1，所以，其双亲结点的编号为 i=(p+k−2)/k，（向下取整）

(3)结点 p 的右孩子的编号为 kp+1，左孩子的编号为 kp+1-k+1=k(p-1)+2，第 i 个孩子的编号为 k(p-1)+2+i-1=kp-k+i+1。

(4)当(p-1)%k!=0 时，结点 p 有右兄弟，其右兄弟的编号为 p+1。

1.4 已知一棵度为 k 的树中有 n_1 个度为 1 的结点，n_2 个度为 2 的结点，...，n_k 个度为 k 的结点，问该树中有多少个叶子结点（n_0）？

【参考答案】

根据树的定义，在一颗树中，除树根结点外，每个结点有且仅有一个前驱结点，也就是说，每个结点与指向它的一个分支一一对应，所以除树根结点之外的结点树等于所有结点的分支数，即度数，从而可得树中的结点数等于所有结点的度数加 1。

总结点数为：

$$1+n_1+2n_2+3n_3+...+kn_k$$

而度为 0 的结点数就应为总结点数减去度不为 0 的结点数的总和，即：

$$n_0=1+n_1+2n_2+3n_3+...+kn_k-(n_1+n_2+n_3+...+n_k)=1+\sum(i-1)n_i \quad ， \quad i=1..n$$

2、算法设计

针对本部分的每一道题，要求：

（1） 采用 C 或 C++语言设计数据结构；

（2） 给出算法的基本设计思想；

（3） 根据设计思想，采用 C 或 C++语言描述算法，关键之处给出注释；

（4） 说明你所设计算法的时间复杂度和空间复杂度。

2.1 已知一棵二叉树按顺序方式存储在数组 *int A*[1..n]中。设计算法，求出下标分别为 i 和 j（i<=n，j<=n）的两个结点的最近的公共祖先结点的位置和值。

【参考答案】

该题是按完全二叉树的格式存储。利用完全二叉树双亲结点与孩子结点编号间的关系，求下标为 i 和 j 的两结点的双亲，双亲的双亲，……，等等，直至找到最近的公共祖先。

```
void    Ancestor( ElemType A[ ]，int n，int i，int j )
{   while( i! =j)
    if ( i > j)
         i = i/2;          //下标为 i 的结点的双亲结点的下标
     else
        j = j/2 ;          //下标为 j 的结点的双亲结点的下标
    printf("所查结点的最近公共祖先的下标是%d，值是%d" ，i，A[i]);
}// Ancestor
```

2.2 假设二叉树 *bt* 采用二叉链表存储，在二叉树 *bt* 中查找值为 *x* 的结点，试编写算法打印值为 *x* 的结点的所有祖先，假设值为 *x* 的结点不多于一个。试分析该算法的时间复杂度。

【参考答案】

后序遍历最后访问根结点，当访问到值为 *x* 的结点时，栈中所有元素均为该结点的祖先。

```
void Search (BiTree bt，  ElemType x)
 //在二叉树 bt 中，查找值为 x 的结点，并打印其所有祖先
{   typedef struet {
                      BiTree    t;
                      lnt tag;
                     } stack;          // tag 标志
    stack s[ ] ;      //假设核容量足够大
    top=0 ;
    while ( bt! = null || top>0)
      {   while ( bt! = n1lll && bt->data! = x)
            {   s[++top].t = bt;
                s[top].tag =0;
                bt=bt->lehild;
             }   //结点入枝，沿左分支向下
          if( hb->data = = x)
             {   printf("所查结点的所有祖先结点的值为：\n") ;       //找到 x
                 for (i=1 ; i <= top; i + +) printf(s[i].t->data) ;
                 return;
              } //输出祖先值后结束
          while(top! =0&&s[top].tag = = 1)    top-- ;     //退栈(空遍历)
          if(top!=0)
          {   s[top].tag = 1 ;
              bt=s[top].t->rehild;
          }   //沿右分支向下遍历
      }   // while ( bt! = null|| top>0)
}   //Search
```

因为查找的过程就是后序遍历的过程，使用的栈的深度不超过树的深度，算法复杂度为 $O(log_2n)$。

2.3 一棵二叉树 *T* 的繁茂度定义为各层结点数的最大值(也称二叉树的宽度)和二叉树的高度的乘积。试设计算法，求给定二叉树 *T* 的繁茂度。

【参考答案】

```
int BiTreeThrive(BiTree & T)
{
    int i，d，nn[20];
    d=BiTDepth(T);
    BiTree p=T;
    Stack s1，s2;
    InitStack(s1); InitStack(s2);
    for(i=0;i<20;i++)    nn[i]=0;            // 每层结点个数
    if(p)
        Push(s1，p);
    else
        return 0;
    for(i=0;i<d;i++)
        {    if(!StackEmpty(s1) && StackEmpty(s2))
                {   while(!StackEmpty(s1))
                        {   Pop(s1，p); nn[i]++;              // s1 中存放第 i 层的结点
                            if(p->lchild) Push(s2，p->lchild);//s2 中存放第 i+1 层结点
                            if(p->rchild) Push(s2，p->rchild);
                        } //while
                } //if
            else{   if(StackEmpty(s1) && !StackEmpty(s2))
                        {   while(!StackEmpty(s2))
                                {   Pop(s2，p);   nn[i]++;
                                    if(p->lchild)   Push(s1，p->lchild);
                                    if(p->rchild)   Push(s1，p->rchild);
                                } //while
                        } //if
                  } //else
        } //for
        int max=nn[0];
        for(i=0;i<d;i++)
            if(max<nn[i])   max=nn[i];
        return max*d;
    } / BiTreeThrive
```

2.4 设计算法，对于二叉树 *T* 中每一个元素值为 *x* 的结点，删去以它为根的子树，并释放相应的空间。

【参考答案】

```
// 删除以元素值为 x 的结点为根的子树
Status DelChildTree(BiTree& T，TElemType x)
{
    if(T){
            if(T->data==x)
            {
```

```
                DelBTree(T);
                T=NULL;
                return OK;
            }
          Else
            {    if(DelChildTree(T->lchild，x))
                        return OK;
                 else{
                        if(DelChildTree(T->rchild，x))
                           return OK;
                        else return ERROR;
                       } //else
            }//else
        }//if
    else
        return ERROR;
}

// 删除二叉树
Status DelBTree(BiTree& T)
{
    if(T){
            DelBTree(T->lchild);
            DelBTree(T->rchild);
            delete T;
            return OK;
        }
    else
        return ERROR;
}
```