**1.如果已知一棵二叉树有20个叶子结点，有10个结点仅有左孩子，15个结点仅有右孩子，求出该二叉树的结点数目。**

n0=n2+1

n2=19

n1=25

n=n0+n1+n2+1=65

**2.已知某完全二叉树有100个结点，试用三种不同的方法求出该二叉树的叶子结点数。**

二叉树高度为7

前六层有63个节点，第七层有37个

第六层有19个节点是有子叶的

第六层有32个结点，13个节点为叶子结点

共37+13=50个

若二叉树为满二叉树，有127个节点，所以第七层删除27个节点

第七层含结点37个，第六层有13个无子叶节点

共50个

总结点数N=2n（叶子结点数）

共50个

**3.设计算法以求解编号为*i*和*j*的两个结点的最近的公共祖先结点的编号。**

三叉链的二叉树，遍历node1的父节点，与node2的父节点比较，相等时停止循环

Node \*cur

while (node1 != NULL)

{

node1 = node1->\_parent;

cur = node2;

while (cur != NULL)

{

if (node1 == cur->\_parent)

{

return node1;

}

cur = cur->\_parent;

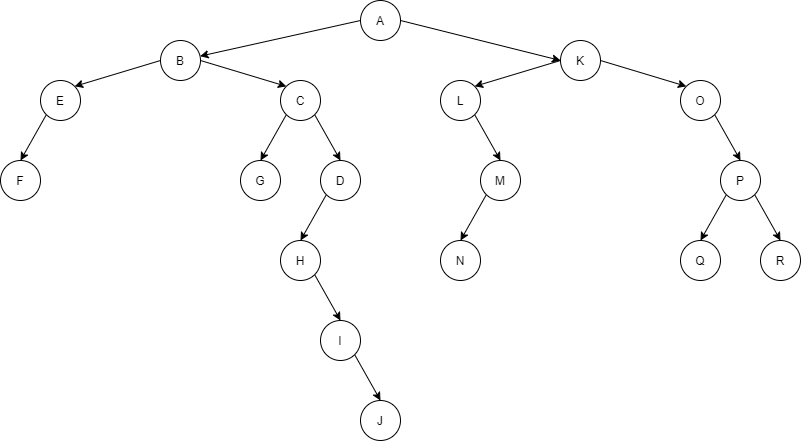
}

}

}

**4. 将下面的森林转换为对应的二叉树**。





**5. 设计算法求树/森林中所有的父子对。**

将树转换为二叉树，递归求父子。

int func(tree T) {

if (T->fristson != NULL && T->nextbrother != NULL) {

cout << "("<<T->data<<",";

if (T->fristson != NULL)

cout << T->fristson->data;

if (T->nextbrother != NULL)

cout << T->nextbrother->data;

cout << ")";

}

else

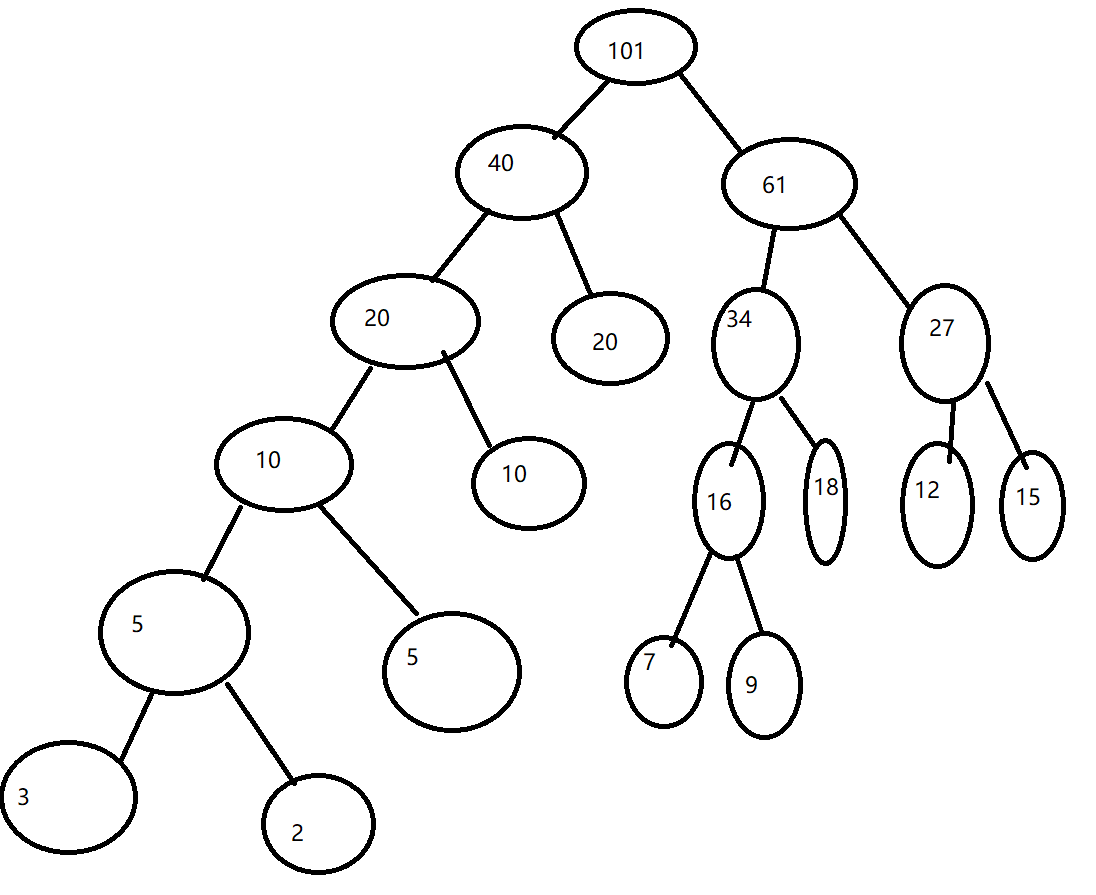
return 0;

func(T->fristson);

func(T->nextbrother);

}

**6.以集合w={2,3,5,7,9,10,12,15,18,20}中的元素为权值，构建一棵哈夫曼树，并计算其带权路径长度值。**



101+40+20+10+5++61+34+16+27=314

20\*2+10\*3+5\*4+2\*5+3\*5+18\*3+12\*3+15\*3+7\*4+9\*4=314

**7. 设计算法输出哈夫曼树各叶子结点的编码。**

递归调用子树，后继为空时，输出编码。