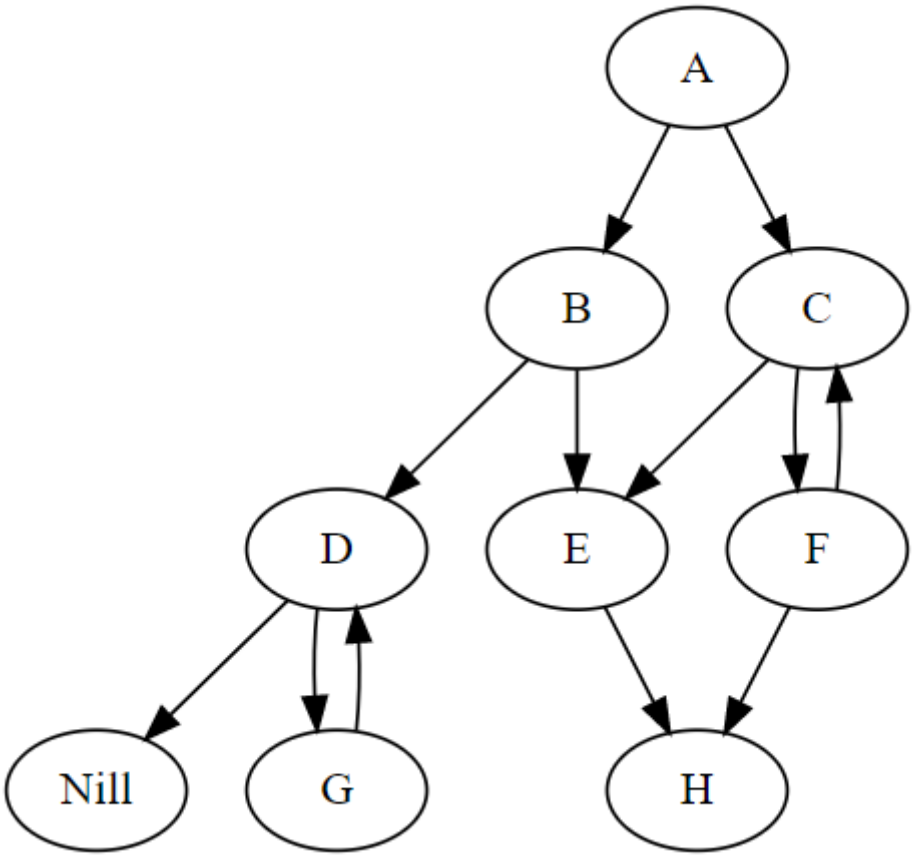
树与二叉树作业

登峰1901 张皓鸿

6.8

总结点数减去叶子节点数为非叶子节点数
kn1+1-n0=n1
移项得
n0=(k-1)n1+1

6.15



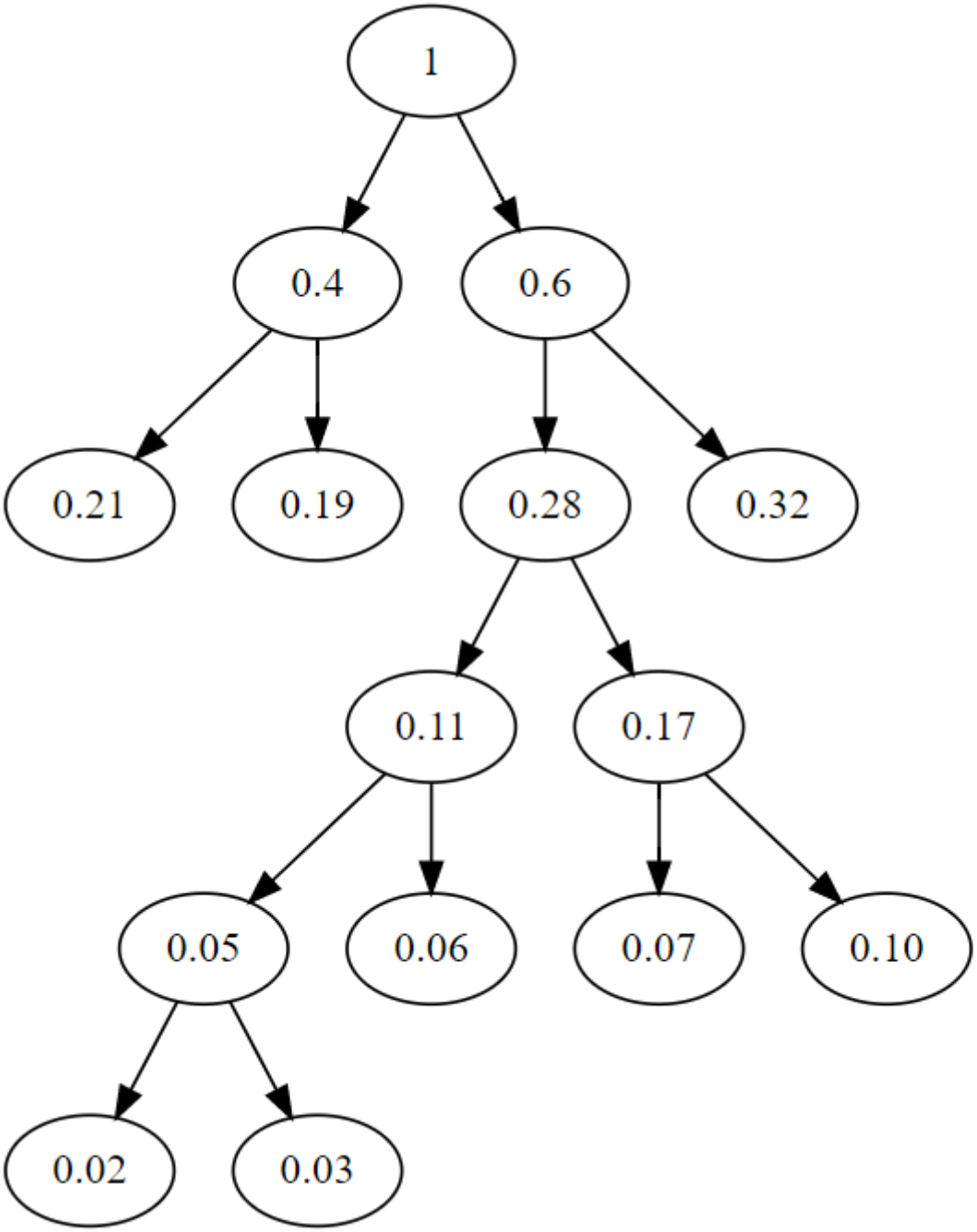
6.16

index	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Info	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Ltag	0	0	0	1	0	1	0	1	0	0	1	1	1	1

index	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Lchild	2	4	6	2	7	3	10	14	12	13	13	9	10	11
Rtag	0	0	1	1	0	0	0	1	1	1	0	1	1	1
Rchild	3	5	6	5	8	9	11	3	12	13	14	0	11	8

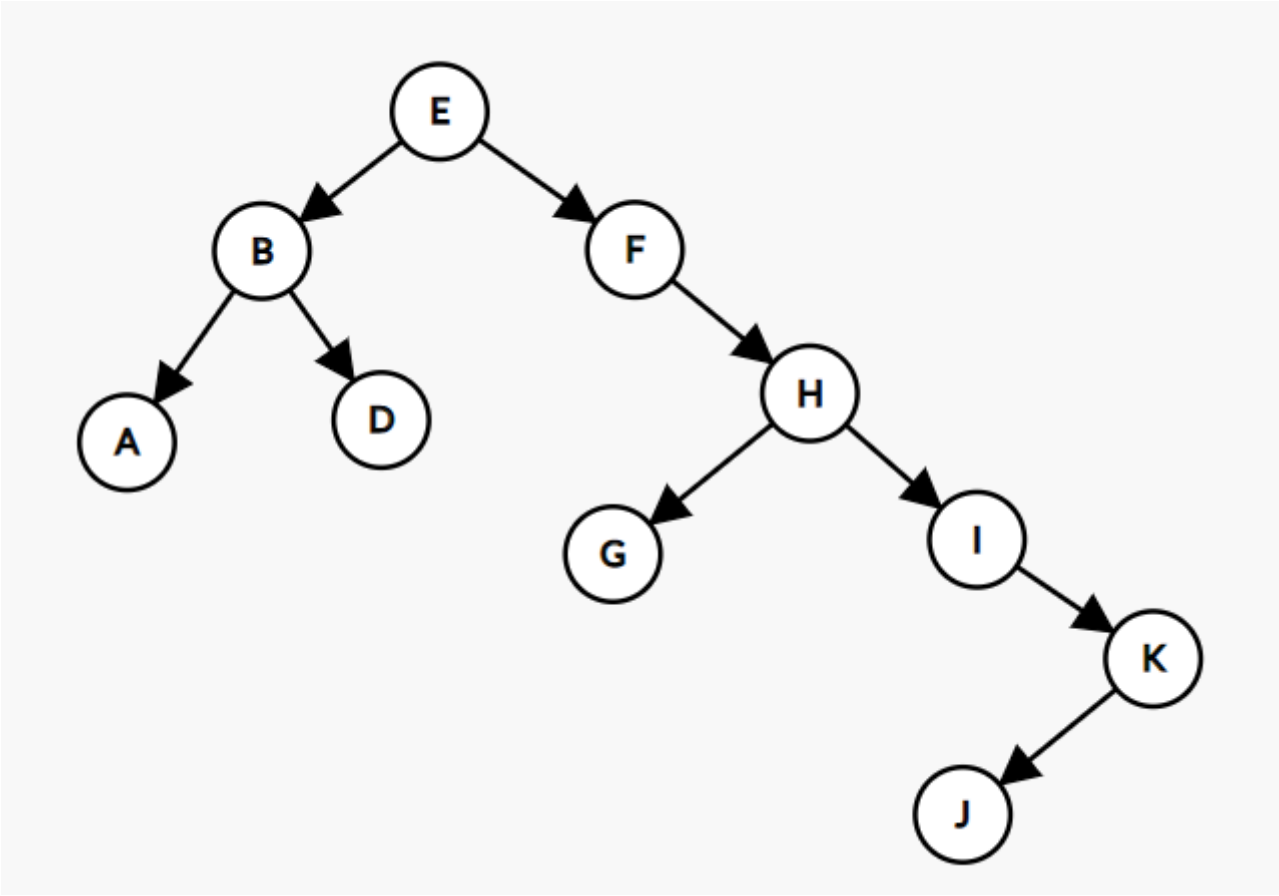
6.26

哈夫曼编码为

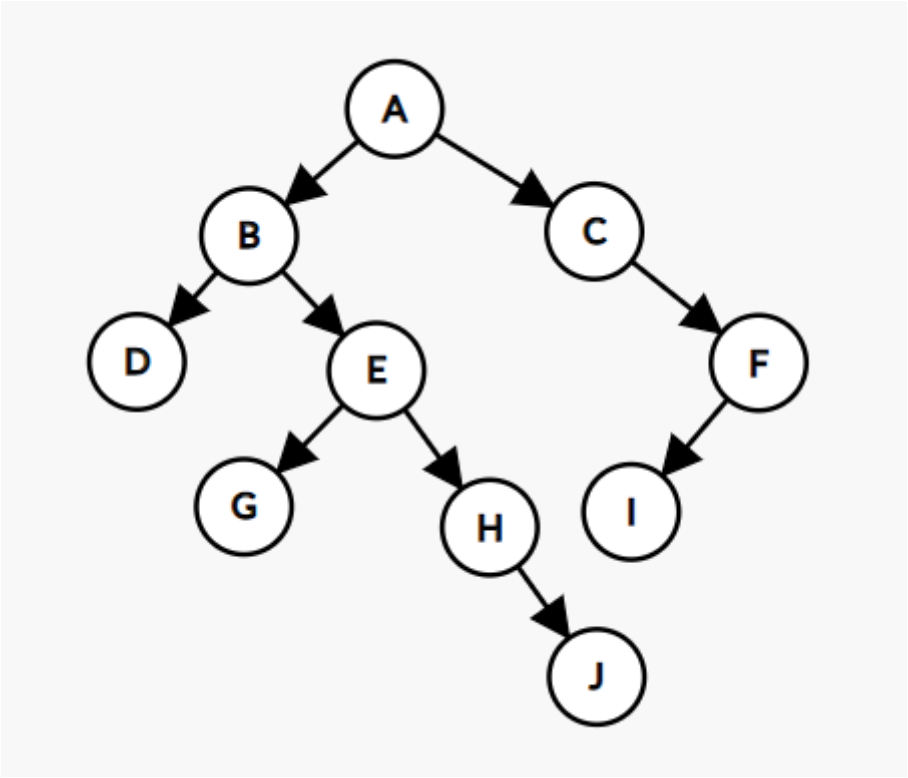


取向左为0，向右为1
00, 01, 11, 10000, 10001, 1001, 1010, 1011
相对于树的表示，不如树直观，无法定量反应距离权重，但更节省空间。

6.27



6.29



6.43

```
Status mirror_tree(bitree root){      //镜像二叉树
    if(!root)return 0;
    node *p = root->rchild;
    root->rchild = root->lchild;
    root->lchild = p;
    mirror_tree(root->lchild);
    mirror_tree(root->rchild);
}
```

6.44

```
Status find_x( bitree root, char x, bitree &tmp){       //找到x节点
    if(!root)return ERROR;
    if(root->data == x){
        tmp = root;
        return OK;
    }
    else{
        if(find_x(root->lchild, x, tmp))return OK;
        else if(find_x(root->rchild, x, tmp))return OK;
    }
}

int tree_height(bitree root){     //求二叉树高度

    if(!root)return 0;
    if(root && !root->lchild && !root->rchild)return 1;
    if(tree_height(root->lchild) >= tree_height(root->rchild))
    return tree_height(root->lchild)+1;
    else return tree_height(root->rchild)+1;
}
```

6.45

```
Status find_x( bitree root, char x, bitree &tmp){       //找到x节点
    if(!root)return ERROR;
    if(root->data == x){
        tmp = root;
        return OK;
    }
    else{
        if(find_x(root->lchild, x, tmp))return OK;
        else if(find_x(root->rchild, x, tmp))return OK;
    }
}

Status delete_child(bitree &root){       //删去子节点
    if(!root)return ERROR;
    delete_child(root->lchild);
    delete_child(root->rchild);
    root->lchild = NULL;
    root->rchild = NULL;
}
```

6.47

```
typedef struct Sqbitree{          //顺序储存二叉树
    char data [MAX_TREE_SIZE];
    int node_num;
}Sqbitree;

Status layer_display(Sqbitree root){      //层次遍历
    int i = 0;
    while(i < root.node_num){
        if(root.data[i])
            printf("%d", root.data[i]);
        i++;
    }
}
```

6.49

```
Status is_completed(bitree root){     //判断是否为完全二叉树
    if(!root)return OK;
    if((!root->rchild && root->lchild) || (root->rchild && !root->lchild)){
        printf("not completed");
        judge++;
        return 0;}
    is_completed(root->lchild);
    is_completed(root->rchild);
}

void judge_complete(){
    if(!judge)printf("is completed");
}
```

6.65

```
typedef struct{
    char ch[maxsize];
    int low, high;
}sqlist;

bitree BuTrPM(sqlist s1, sqlist s2){      //前序中序建树
    int j,l1,l2,h1,h2; char c; node *p;
    l1=s1.low;l2=s2.low;h1=s1.high;h2=s2.high;
    if(l1 > h1 || l2 > h2)   return(0);
    c=s1.ch[s1.low];
    p=new(node); p->data=c;
    for(j=s2.low;j<=s2.high;j++)
        if(c==s2.ch[j]) break;
    s1.low=l1+1; s1.high=l1+j-l2; s2.low=l2;s2.high=j-1;
    p->lchild= BuTrPM(s1,s2);
    s1.low=l1+j-l2+1; s1.high=h1; s2.low=j+1; s2.high=h2;
    p->rchild= BuTrPM(s1,s2);
    return(p);
}
```