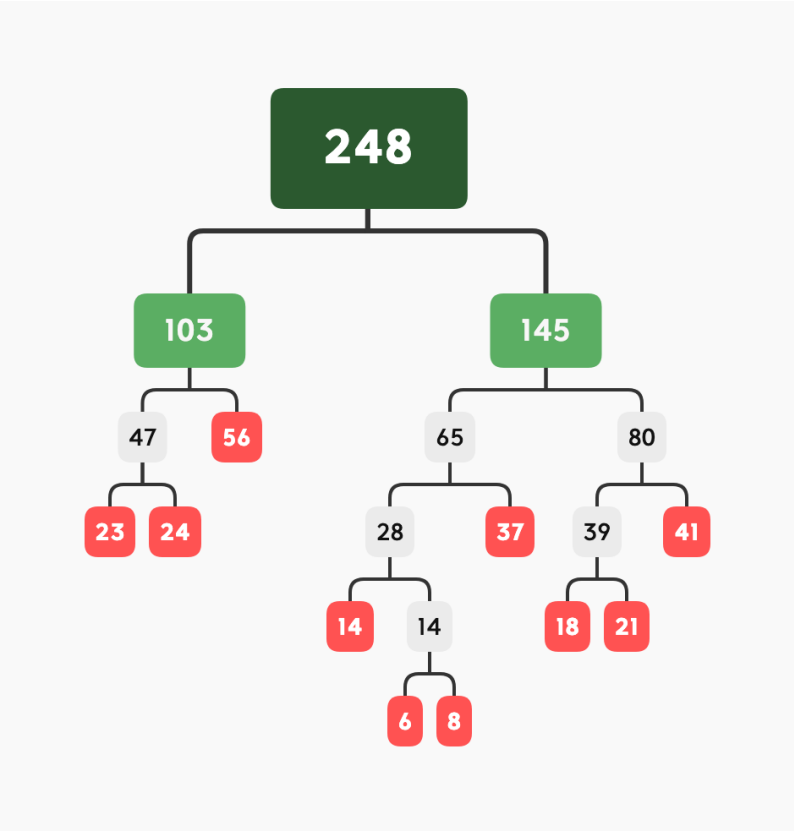
数据结构作业

姓名：叶先志　学号：18040400011　班级：1818039

第 14 题

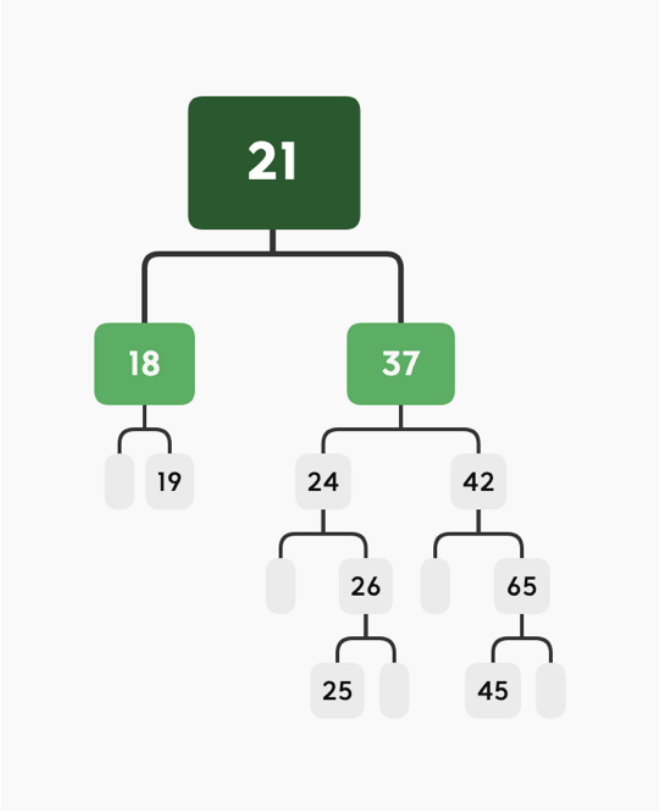
哈夫曼树：



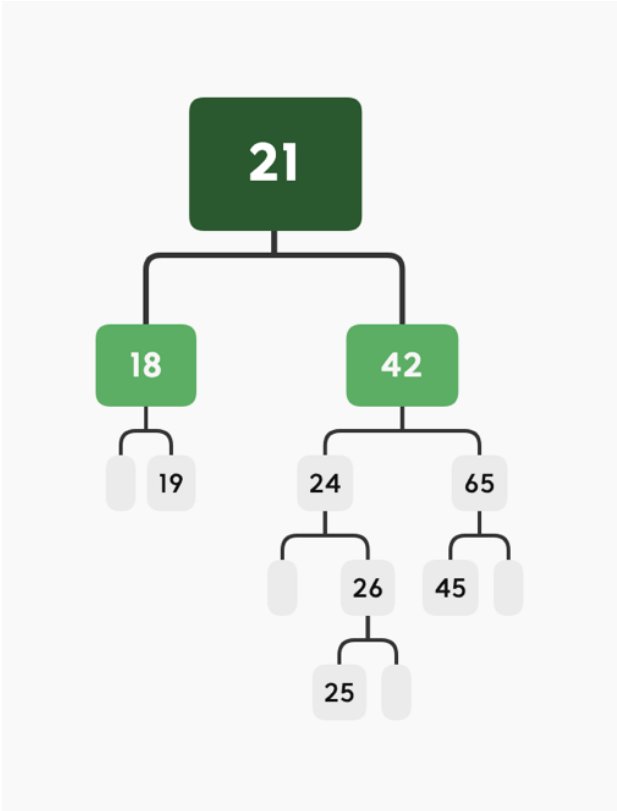
从哈夫曼树根结点开始，对左子树分配代码 0，右子树分配代码 1， 一直到达叶子结点为止，然后将从树根沿每条路径到达叶子结点的代码排列起来，便得到了哈夫曼编码：

符号编号	符号出现频率	哈夫曼编码
0	23	000
1	24	001
2	14	1000
3	6	10010
4	8	10011
5	37	101
6	56	01
7	18	1100
8	21	1101
9	41	111

第 15 题
原二叉排序树：



删除结点 37 后的二叉排序树：



第 17 题
代码：

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
typedef char elemtype;

typedef struct node          //定义二叉树
{
    elemtype ch;
    struct node *lchild,*rchild;
}Bitree;

Bitree *CreatTree()     //二叉树的建立，广度优先输入
{
    char m;
    Bitree *S,*root;
    root=NULL;
    Bitree *Q[MAX];          //指向每个元素的指针构成的队列
    int front=1,rear=0;
    while((m=getchar())!='#')
    {
        S=NULL;
        if(m!='@')
        {
            S=(Bitree*)malloc(sizeof(Bitree));
            S->ch=m;
            S->lchild=NULL;
            S->rchild=NULL;
        }
        rear++;
        Q[rear]=S;
        if(rear==1)
        {
            root=S;
        }
        else
        {
            if(S&&Q[front])
            {
                if(rear%2==0)
                {
                    Q[front]->lchild=S;
                }
```

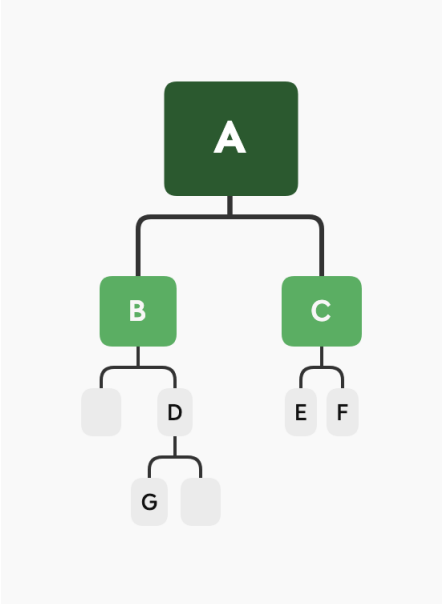
```
43.                 else
44.                 {
45.                     Q[front]->rchild=S;
46.                 }
47.             }
48.             if(rear%2==1)
49.             {
50.                 front++;
51.             }
52.         }
53.     }
54.     return root;
55. }
56.
57. void preOrder(Bitree *root)//非递归的先序遍历算法
58. {
59.     Bitree *Stack[MAX];
60.     Bitree *s;
61.     int top=-1;
62.     if(root!=NULL)
63.     {
64.         top++;
65.         Stack[top]=root;
66.         while(top!=-1)
67.         {
68.             s=Stack[top];
69.             top--;
70.             while(s)
71.             {
72.                 printf("%c ",s->ch);
73.                 if(s->rchild!=NULL)     //若有右孩子，入栈
74.                 {
75.                     top++;
76.                     Stack[top]=s->rchild;
77.                 }
78.                 s=s->lchild;
79.             }
80.         }
81.     }
82.     else
83.         printf("二叉树为空！\n");
84. }
85.
86. int main()
```

```
{
    Bitree *root;
    printf("请输入按广度优先的二叉树，@表示虚结点，#表示结束：\n");
    root=CreatTree();
    printf("按非递归的先序遍历算法可以得到：\n");
    preOrder(root);
    printf("\n");
    return 0;
}
```

程序效果：

```
请输入按广度优先的二叉树，@表示虚结点，#表示结束：
ABC@DEF@@G#
按非递归的先序遍历算法可以得到：
A B D G C E F
Program ended with exit code: 0
```

输入二叉树为：



第 21 题
代码：

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
typedef struct node          //定义二叉树
{
    char ch;
    struct node *lchild,*rchild;
} Bitree;

Bitree *creatree()           //二叉树的建立，广度优先输入
{
    char m;
    Bitree *Q[MAX];
    Bitree *s,*root;
    root=NULL;
    int front=1,rear=0;
    while((m=getchar())!='#')
    {
        s=NULL;
        if(m!='@')
        {
            s=(Bitree*)malloc(sizeof(Bitree));
            s->ch=m;
            s->lchild=NULL;
            s->rchild=NULL;
        }
        rear++;
        Q[rear]=s;
        if(rear==1)
            root=s;
        else
        {
            if(Q[front]&&s)
            {
                if(rear%2==0)
                    Q[front]->lchild=s;
                else
                    Q[front]->rchild=s;
            }
            if(rear%2==1)
                front++;
        }
```

```
    }
    return root;
}

void Layer(Bitree *root)    //按广度优先遍历输出,利用队列的先进先出来实现
{
    Bitree *Q[MAX];
    Bitree *s;
    int front=0,rear=1;
    if(root)                    //二叉树不为空时
    {
        Q[rear]=root;
        while(front<rear)       //队列从 1 开始，front 指向队头的头元素
        {
            front++;
            printf("%c ",Q[front]->ch);      //出队列
            s=Q[front];
            if(s->lchild!=NULL)             //入队列
            {
                rear++;
                Q[rear]=s->lchild;
            }
            if(s->rchild!=NULL)
            {
                rear++;
                Q[rear]=s->rchild;
            }
        }
        printf("\n");
    }
    else
        printf("二叉树为空！\n");
}

void exchange(Bitree *root)      //交换左右子树
{
    Bitree *s;
    if(root)
    {
        s=root->lchild;
        root->lchild=root->rchild;
        root->rchild=s;
        exchange(root->lchild);  //根节点的左子树交换
        exchange(root->rchild);  //根节点的右子树交换
```

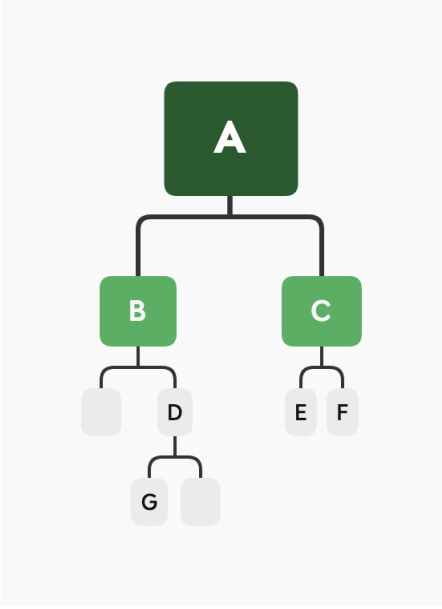
```
    }
}

int main()
{
    Bitree *root;
    printf("请输入按广度优先的二叉树，@表示虚结点，#表示结束：\n");
    root=creatree();
    printf("交换之前按广度优先遍历可以得到：\n");
    Layer(root);
    printf("交换之后按广度优先遍历可以得到：\n");
    exchange(root);
    Layer(root);
     return 0;
}
```

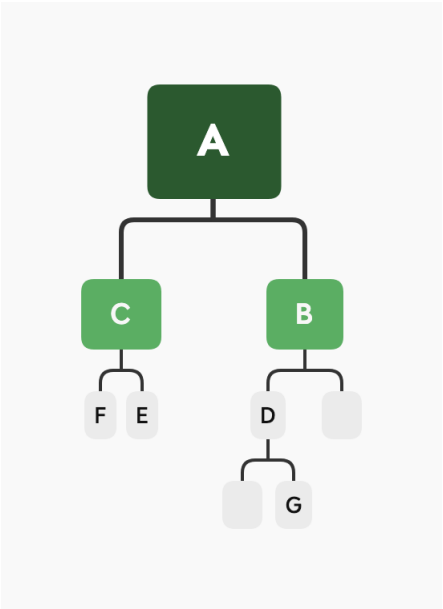
程序效果：

```
请输入按广度优先的二叉树，@表示虚结点，#表示结束：
ABC@DEF@@G#
交换之前按广度优先遍历可以得到：
A B C D E F G
交换之后按广度优先遍历可以得到：
A C B F E D G
Program ended with exit code: 0
```

输入二叉树为：



交换后输出的二叉树为：



哈夫曼编码译码程序：
代码：

```
#include <stdio.h>
#include <stdlib.h>
int num_of_leaves;      //符号个数
int num_of_nodes;       //节点个数

typedef struct
{
    char bits[50];  //编译器不允许用变量来设置数组长度
    int start;
    char ch;
}codetype;
codetype code[50];

typedef struct
{
    int lchild,rchild,parent;
    float weight;
    char ch;
}hufmtree;
hufmtree *root;

hufmtree * creathufmtree()
{
    int i,j,p1,p2,s1,s2;
```

```
    hufmtree * root;
    num_of_nodes=2*num_of_leaves-1;
    root=(hufmtree*)malloc(num_of_nodes*sizeof(hufmtree));
    for(i=0;i<num_of_nodes;i++)          //初始化
    {
        root[i].lchild=root[i].parent=root[i].rchild=root[i].weight=-1;
        root[i].ch='@';
    }

    printf("请连续输入%d 个符号（无空格）：\n",num_of_leaves);
    for(i=0;i<num_of_leaves;i++)
        scanf("%c",&root[i].ch);
    getchar();

    printf("请依次输入每个符号的权值：\n");
    for(i=0;i<num_of_leaves;i++)    //输入前 n 个点的权值
        scanf("%f",&root[i].weight);
    getchar();

    for(i=num_of_leaves;i<num_of_nodes;i++)    //合并
    {
        p1=p2=0;
        s1=s2=1000;
        for(j=0;j<i;j++)    //从第一个节点遍历，寻找权值最小的两个节点
        {
            if(root[j].parent==-1)
            {
                if(root[j].weight<s1)
                {
                    s2=s1;
                    s1=root[j].weight;
                    p2=p1;
                    p1=j;
                }
                else if(root[j].weight<s2)
                {
                    s2=root[j].weight;
                    p2=j;
                }
            }
        }
        root[p1].parent=i;
        root[p2].parent=i;
        root[i].lchild=p1;
```

```
69.         root[i].rchild=p2;
70.         root[i].weight=root[p1].weight+root[p2].weight;
71.     }
72.     return root;
73. }
74.
75. void hufmcode(hufmtree *root)
76. {
77.     int i,son,p;
78.     codetype temp;
79.     for(i=0;i<num_of_leaves;i++)
80.     {
81.         temp.start=num_of_leaves;
82.         temp.ch=root[i].ch;       //保留 root[i]的符号
83.         son=i;
84.         p=root[son].parent;
85.         while(p!=-1)
86.         {
87.             temp.start--;
88.             if(root[p].lchild==son)
89.                 temp.bits[temp.start]='0';
90.             else
91.                 temp.bits[temp.start]='1';
92.             son=p;
93.             p=root[son].parent;
94.         }
95.         code[i]=temp;
96.     }
97.     printf("生成的密码表为: \n");
98.     for(i=0;i<num_of_leaves;i++)
99.     {
100.         printf("%c \t ",code[i].ch);
101.         for(int j=code[i].start;j<num_of_leaves;j++)
102.         {
103.             printf("%c",code[i].bits[j]);
104.         }
105.         printf("\n");
106.     }
107. }
108.
109. int hufmdecode(hufmtree *root)
110. {
111.     int i,bit;
112.     i=num_of_nodes-1;       //从根节点向下遍历
```

```
113.     printf("请输入二进制代码，每位以空格间隔，最终输入 2 结束：\n");
114.     scanf("%d",&bit);
115.     while(bit!=2)
116.     {
117.         if(bit==0)
118.             i=root[i].lchild;
119.         else if(bit==1)
120.             i=root[i].rchild;
121.         else
122.         {
123.             printf("请输入 0 或 1，或输入 2 结束！\n");
124.             return 0;
125.         }
126.         if(root[i].lchild==-1)      //找到叶子,输出符号
127.         {
128.             printf("%c",root[i].ch);
129.             printf("\n");
130.             i=num_of_nodes-1;
131.         }
132.         scanf("%d",&bit);
133.     }
134.     return 0;
135. }
136. 
137. int main(int argc, const char * argv[])
138. {
139.     printf("请输入符号个数：");
140.     scanf("%d",&num_of_leaves);
141.     getchar();
142.     root=creathufmtree();
143.     /*
144.     for(int i=0;i<num_of_leaves;i++)
145.         printf("%c",root[i].ch);
146.     printf("\n");
147.      */
148.     hufmcode(root);
149.     hufmdecode(root);
150.     return 0;
151. }
```

程序效果：

```
请输入符号个数：3
请连续输入3个符号（无空格）：
abc
请依次输入每个符号的权值：
1 2 3
生成的密码表为：
a    10
b    11
c    0
请输入二进制代码，每位以空格间隔，最终输入2结束：
0
c
1 0
a
1 1
b
2
Program ended with exit code: 0
```

检测第十五题：

```
请输入符号个数：10
请连续输入10个符号（无空格）：
0123456789
请依次输入每个符号的权值：
23 24 14 6 8 37 56 18 21 41
生成的密码表为：
0    000
1    001
2    1000
3    10010
4    10011
5    101
6    01
7    1100
8    1101
9    111
请输入二进制代码，每位以空格间隔，最终输入2结束：
0 0 0
0
1 0 0 0
2
0 1
6
2
Program ended with exit code: 0
```