noj实验7报告

0.0 题目//需求分析

需求分析：即输入一个串然后把他编码输出，编码方式是哈夫曼树

1.0 实验思路

1.先建立一棵哈夫曼树，在建立时需要将叶结点对应的字符记录在树中。

2.我们需要建立一套译码系统，将叶结点中存储字符对应的译码存储到字符串数组hc中。建立时，我们需要从叶结点开始进行记录，即从下到上，我们将每次得到的译码先存储在字符串cd中（cd长度不会超过叶结点个数），并且时从cd最后（结束符\0前）开始记录（因为之前是反的，负负得正），之后整合在hc中。

3.之后我们要在建立一套反译码系统。众所周知，哈夫曼编码是前缀编码，所以我们只需要将记录报文的字符串code与刚才字符串数组hc中每一项比较hc此项的长度len_hc（用strncmp函数），直到找到将其strcat到字符串recode中，再将code向后推len_hc位，直至比较完。

4.输出即可

2.0 代码

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

typedef struct HTNode
{
    int weight;
    int parent,lchild,rchild;
```

```
    char data;
}HTNode;
typedef struct HCNode
{
    int bit[200];
    int start;
}HCNode;

HTNode ht[1005];
HCNode hc[200];
int str[1005]={0};
int num=0;

void Select(int pos,int *x1,int *x2)  //找两个最小权的无父结点的结点
{
   int m1=1000,m2=1000;
   int j;
   for(j=1;j<pos;j++)
   {
      if(ht[j].weight<m1&&ht[j].parent==0)
      {
         m2=m1;*x2=*x1;
         m1=ht[j].weight;
         *x1=j;
      }
      else if(ht[j].weight<m2&&ht[j].parent==0)
      {
         m2=ht[j].weight;
         *x2=j;
      }
    }
}

void init(int n)  //树初始化
{
    int i,j,x1,x2;
    char c;
    for(i=1;i<=2*n-1;i++)
    {
        ht[i].weight=0;
        ht[i].lchild=0;
        ht[i].parent=0;
        ht[i].rchild=0;
    }
    for(i=1;i<=n;i++){
        getchar();
        scanf("%c",&ht[i].data);
    }
    for(i=1;i<=n;i++) scanf("%d",&ht[i].weight);
    for(i=1;i<n;i++)
    {
        Select(n+i,&x1,&x2);
        ht[x1].parent=n+i;
        ht[x2].parent=n+i;
        ht[n+i].weight=ht[x1].weight+ht[x2].weight;
        ht[n+i].lchild=x1;
        ht[n+i].rchild=x2;
    }
```

```
}

void getnum(int n)  //编码
{
    int i,j;
    HCNode x;
    for(i=1;i<=n;i++)
    {
        x.start=n;
        int cur=i;
        int par=ht[cur].parent;
        while(par!=0)
        {
            if(ht[par].lchild==cur) x.bit[x.start]=0;
            else x.bit[x.start]=1;
            x.start--;
            cur=par;
            par=ht[cur].parent;
        }
        for(j=x.start+1;j<=n;j++) hc[i].bit[j]=x.bit[j];
        hc[i].start=x.start+1;
    }
}

void print(int n)  //输出编码
{
    char code[1000];
    int i,j,k;
    scanf("%s",code);
    for(i=0;i<strlen(code);i++)
    {
        for(j=1;j<=n;j++)
        {
            if(code[i]==ht[j].data)
            {
                for(k=hc[j].start;k<=n;k++)
                {
                    printf("%d",hc[j].bit[k]);
                    str[num]=hc[j].bit[k];
                    num++;
                }
            }
        }
    }
    printf("\n");
}

void decode(int n)  //译码并输出
{
    int i=0;
    int t;
    while(i<num)
    {
        t=2*n-1;
        while(ht[t].lchild!=0&&ht[t].rchild!=0)
        {
            if(str[i]==0) t=ht[t].lchild;
            else t=ht[t].rchild;
```

```
            i++;
        }
        printf("%c",ht[t].data);
    }
}

int main()
{
    int n;
    scanf("%d",&n);
    init(n);
    getnum(n);
    print(n);
    decode(n);
    return 0;
}
```

3.0 心得

在纸上的分析确实很重要，刚开始毫无思路，到后来纸上分析后，就简化为构建树遍历树访问结点的操作。

遭遇了野指针问题，通过debug得以解决。

注：参考了网上资料思路。