# 实习报告

题目：实现霍夫曼树的创建，编码，译码

班级： 电信1809 姓名： 胡伊曼 学号： U201813486 完成日期：2019.4.22

1. **需求分析**
2. 给定权值，字符
3. 输入字符数n
4. 创建霍夫曼树，对给定字符进行编码（coding）译码（decode）
5. **概要设计**
6. **主程序**

Void main（）{

初始化；

输入n；

输入权值数组weight；

建树，编码，译码操作

输出给定字符串的编码，输入一组二进制码，输出译码

}

1. **详细设计**

/\*创建霍夫曼树：

1.HT数组长度2n，第0位不用，1~n用来存放n个字符的数据，n+1~2n-1用来表示字符结点的parent结点

根据他们之间的parent child关系 建立出霍夫曼树，数组只是这棵树的存储结构

2.selet用来找出parent == 0的两个权值最小的结点，从n+1~2n-1按顺序作为每次选出的最小结点的parent，

将他们之间的parent child关系输入

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct

{

int value;

int weight;

int parent,lchild,rchild;

char \* string;

}HT;

void select(HT \* tree,int \*ps1,int \*ps2,int n)

{

int i,s1,s2;

s1 = 1;

for(i = 1;i<n;i++)

{

if(tree[s1].parent!=0)

{

s1++;

}

else if(tree[s1].weight>=tree[i].weight&&tree[i].parent==0)

s1 = i;

}

\*ps1 = s1;

s2 = 1;

for(i = 2;i<n;i++)

{

if(i == s1)i++;

if(s2 == s1) s2++;

if(tree[s2].parent!=0)

{

s2++;

}

else if(tree[s2].weight>=tree[i].weight&&tree[i].parent == 0)

s2 = i;

}

\*ps2 = s2;

}

void initialize(HT \* tree,int \* w,char \* c,int n)

{

int i;

int s1,s2;

for(i=1;i<n+1;i++)//把n个字符的value和weight存入HT数组 ,HT[0]不存放数据

{

tree[i].value = c[i-1];

tree[i].weight = w[i-1];

tree[i].parent = 0;

tree[i].lchild = 0;

tree[i].rchild = 0;

tree[i].string = NULL;

}

for(;i<2\*n;i++)//把n+1~2n-1的位置上的数据全部置零

{

tree[i].value = 0;

tree[i].parent = 0;

tree[i].lchild = 0;

tree[i].rchild = 0;

tree[i].weight = 0;

tree[i].string = NULL;

}

for(i = n+1;i<2\*n;i++)

{

select(tree,&s1,&s2,i-1);

tree[i].lchild = s1;

tree[i].rchild = s2;

tree[s1].parent = i;

tree[s2].parent = i;

tree[i].weight = tree[s1].weight + tree[s2].weight;

}

}

void addcoding(HT \*tree,int n)

{

char tmp[27];

tmp[n-1]=0;

int start,i,c,f;

for(i=1;i<=n;i++)

{

start = n-1;

for(c=i,f=tree[c].parent;f;c=f,f=tree[c].parent)

{

if(c==tree[f].lchild) tmp[--start] = '0';

else tmp[--start] = '1';

}

strcpy(tree[i].string,&tmp[start]);

}

}

int findpos(char \*c,char k)

{

int i;

for(i=0;i<27;i++)

{

if(k==c[i])return i+1;

}

}

void coding(HT \*tree,char \* s,char \*c,char \*cs)

{

int i,m;

char k;

for(i=0;i<20;i++)

{

k = s[i];

m = findpos(c,k);

strcat(cs,tree[m].string);

}

}

/\*void decode(HT \*tree,char \*codes,char \*chars)

{

char m[27];

int i,j;

for(i=0;codes[i];i++)

{

if(codes[i]=='0')

{

for(j=0;j<=i;j++)

{

m[j]=codes[j];

}

m[j]=0;

for(j=1;j<=27;j++)

{

if(strcmp(tree[j].string,m)==0)

{

\*chars = tree[j].value;

chars++;

}

else

{

printf("error");

exit(1);

}

}

}

}

\*chars = 0;

}

\*/

void decode(HT \*tree,char \*codes,char \*chars)

{

int i = 2\*27-1;

for(;\*codes;codes++)

{

if(tree[i].lchild == 0)

{

\*chars = tree[i].value;

chars++;

}

if(\*codes == '0')

{

i = tree[i].lchild;

}

else if(\*codes == '1')

{

i = tree[i].rchild;

}

}

}

int main()

{

/\*int n,i;

printf("please input the number of chars");

scanf("%d",&n);//输入字符个数

int \*w = (int \*)malloc(n\*sizeof(int));

char \*c = (char \*)malloc(n\*sizeof(char));

for(i=0;i<n;i++)

{

printf("please input the value and weight");

scanf("%c",&c[i]);//输入每个字符

getchar();

scanf("%d",&w[i]);//输入每个字符的权值

}

HT \* tree = (HT \*)malloc(2\*n\*sizeof(HT));//开一个HT数组

initialize(tree,w,c,n);

//int m = tree[1].parent;

//printf("%d",tree[m].rchild);

\*/

int n,i;

n = 27;

int w[27] = {186,64,13,22,32,103,21,15,47,57,1,5,32,20,57,63,15,1,48,51,80,23,8,18,1,16,1};

char c[28] = {" ABCDEFGHIJKLMNOPQRSTUVWXYZ"};

HT \*tree = (HT \*)malloc(2\*n\*sizeof(HT));

initialize(tree,w,c,n);

for(i=1;i<=n;i++)

{

tree[i].string = (char \*)malloc(n\*sizeof(char));

}

addcoding(tree,n);

char s[30] = {"THIS PROGRAM IS MY FAVORITE"};

char codestring[1000];

codestring[0]=0;

coding(tree,s,c,codestring);

printf("%s",codestring);

char codes[50] = {"0010001"};

char chars[10];

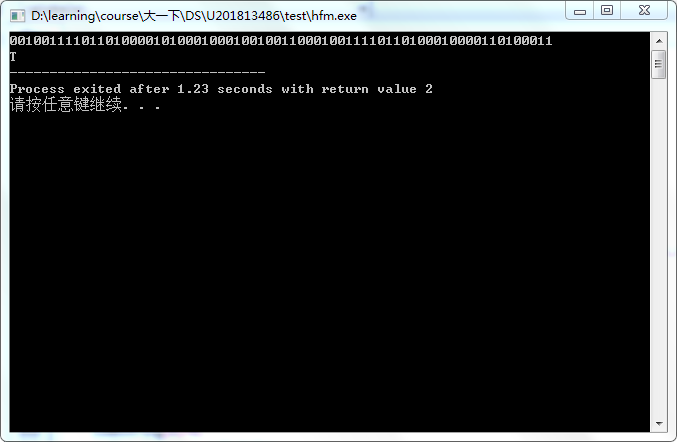
decode(tree,codes,chars);

printf("\n%s",chars);

}

1. **调试分析**
2. 译码是从头结点走到叶结点，0往左1往右
3. 编码是从叶结点开始，直到parent = 0（头结点），若当前是leftchild就是0，其余为1

**五、 测试结果**

****