**广 州 商 学 院**

**实验报告（第 5 次）**

**实验名称： 哈夫曼编码基本操作演示程序的设计与实现**

**实验时间： 2021.5. 7**

1. **实验目的**

1．掌握哈夫曼树的定义和构造方法；

2．掌握哈夫曼编码的方法及应用。

1. **实验内容**

从网上查出26个英文出现的频率，以此为权构造哈夫曼树，并进行编码、译码演示。具体功能包括：

1. 以26个英文字母的频率为权构造一棵哈夫曼树；
2. 自底向上进行编码；
3. 自顶向下进行编码；
4. 对输入的文字（自己任意选择一句英语）编码；
5. 对输入的二进制串译码；

  6. 退出系统。

1. **源程序及主要算法说明**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

typedef struct //树结点定义

{

int weight;

int parent;

int lchild;

int rchild;

}HTNode,\*HuffmanTree;

static char N[100]; //用于保存正文

typedef char \*\*HuffmanCode;//哈弗曼编码，char型二级指针

typedef struct //封装最小权结点和次小权结点

{

int a;

int b;

}MinCode;

//函数声明

void Error(char \*message);

HuffmanCode HuffmanCoding(HuffmanTree &HT,HuffmanCode HC,int \*w,int n);

MinCode Select(HuffmanTree HT,int n);

void Error(char \*message) //当输入1个结点时的错误提示

{

fprintf(stderr,"错误:%s\n",message);

exit(1);

}

HuffmanCode HuffmanCoding(HuffmanTree &HT,HuffmanCode HC,int \*w,int n)//构造哈夫曼树HT，编码存放在HC中,w为权值,n为结点个数

{

int i,a=0,b=0;

HuffmanTree p;

char \*cd;

int f,c,start,m;

MinCode min;

if(n<=1)

{

Error("Code too small!");//只有一个结点不进行编码，直接exit(1)退出。非return,如果return 会造成main函数HT[i]无值

}

m=2\*n-1; //哈弗曼编码需要开辟的结点大小为2n-1

HT=new HTNode[m+1];//开辟哈夫曼树结点空间 m+1 。为了对应关系，我们第0个空间不用。

for(p=HT,i=0;i<=n;i++,p++,w++){ //初始化n个叶子结点,w[0] = 0,main函数已赋值

p->weight=\*w;

p->parent=0;

p->lchild=0;

p->rchild=0;

}

for(;i<=m;i++,p++){ //将n-1个非叶子结点的初始化

p->weight=0;

p->parent=0;

p->lchild=0;

p->rchild=0;

}

//构造哈夫曼树

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

min=Select(HT,i-1); //找出最小和次小的两个结点

a=min.a; //最小结点下标

b=min.b; //次小结点下标

HT[a].parent=i;

HT[b].parent=i;

HT[i].lchild=a;

HT[i].rchild=b;

HT[i].weight=HT[a].weight+HT[b].weight;//赋权和

}

//打印哈弗曼树

printf("HT的终态:\n");

printf("Number\t\tweight\t\tparent\t\tlchild\t\trchild\n");

for(i=1;i<=m;i++){

printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\n",i,HT[i].weight,HT[i].parent,HT[i].lchild,HT[i].rchild);

}

HC=new char\*[n+1];//从叶子结点到根节点求每个字符的哈弗曼编码

cd=new char[n];//为哈弗曼编码动态分配空间

cd[n-1]='\0';//如：3个结点编码最长为2。cd[3-1] = '\0';

//求叶子结点的哈弗曼编码

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

start=n-1;

for(c=i,f=HT[i].parent; f!=0; c=f,f=HT[f].parent){

if(HT[f].lchild==c)

cd[--start]='0';

else

cd[--start]='1';

}

HC[i]=(char \*)malloc((n-start)\*sizeof(char \*));//为第i个字符分配编码空间

strcpy(HC[i],&cd[start]); //将当前求出结点的哈弗曼编码复制到HC

}

free(cd);

return HC;

}

MinCode Select(HuffmanTree HT,int n)

{

int min,secmin;

int i,a,b;

MinCode code ;

a=1;

b=1;

min = 66666;//足够大

//找出权值weight最小的结点，下标保存在s1中

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

{

if(HT[i].weight<min && HT[i].parent==0)

{

min=HT[i].weight;

a=i;

}

}

secmin = 66666;//足够大

//找出权值weight次小的结点，下标保存在s2中

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

{

if((HT[i].weight<secmin) && (i!=a) && HT[i].parent==0)

{

secmin=HT[i].weight;

b=i;

}

}

//放进封装中

code.a=a;

code.b=b;

return code;

}

void HuffmanTranslateCoding(HuffmanTree HT, int n,char\* ch)

{//译码过程

int m=2\*n-1;

int i,j=0;

printf("译码:");

while(ch[j]!='\0'){ //ch[]:你输入的要译码的0101010串

i=m;

while(0 != HT[i].lchild && 0 != HT[i].rchild){//从顶部找到最下面

if('0' == ch[j]){ //0 往左子树走

i=HT[i].lchild;

}else{ //1 往右子树走

i=HT[i].rchild;

}

++j; //下一个路径

}

printf("%c",N[i-1]); //打印出来

}

printf("\n");

}

int main()

{

HuffmanTree HT=NULL;

HuffmanCode HC=NULL;

int \*w=NULL;

int i,n;

char tran[100];

printf("请输入要编码的字符:");

gets(N);

fflush(stdin);

n = strlen(N);

w=(int \*)malloc((n+1)\*sizeof(int \*));//开辟n+1个长度的int指针空间

w[0]=0;

printf("请输入结点的权值（weight）:\n");

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

printf("w[%d]=",i);

scanf("%d",&w[i]);

}

fflush(stdin);//构造哈夫曼树HT，编码存放在HC中,w为权值,n为结点个数

HC=HuffmanCoding(HT,HC,w,n);//输出哈弗曼编码

printf("编码:\n");

printf("Number\t\tWeight\t\tCode\n");

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

printf("%c\t\t%d\t\t%s\n",N[i-1],w[i],HC[i]);

}

fflush(stdin);//译码过程

printf("请输入需要二进制串译码:");

gets(tran);

HuffmanTranslateCoding(HT, n, tran);

printf("按任意键退出。");

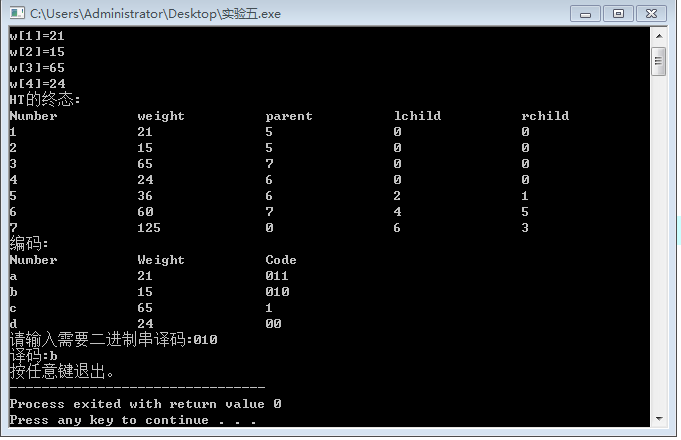
return 0;

}

**四．主要问题和解决方案**

对哈夫曼树的一些算法还不熟悉，要多加练习

1. **测试数据及结果**

****

1. **心得体会与自我评价**

课后要多加练习

1. **教师评分**