**实习报告**

题目：霍夫曼树

班级：电信1809

姓名：叶杰栋

学号：U201813497

完成日期：2019/4/23

1、需求分析

（1）用一张26个大写字母和空格符的使用频率表，将这27个字符通过建立霍夫曼树进行编码。使用该方式编码可以降低传输成本，缩短传输时间。

（2）编码完成之后，尝试对一段报文进行编码和译码，并输出编码和译码的内容。报文可以通过用户输入，也可以在程序内直接定义。

2、概要设计

（1）霍夫曼树的抽象数据类型定义：

ADT HuffmanTree{

数据对象：HuffmanTree型结构体数组，每个结构体数组元素包括：字符、权值、双亲节点位置、左右孩子节点位置。

数据关系：每个数组元素最多可以和其他三个元素建立关系，分别为：A是B的双亲、C是B的左孩子、D是B的右孩子。

基本操作：

InitHuffmanTree()

操作结果：新建一个长度为n的HuffmanTree型结构体数组，并把结构体元素一一存入其中。

Select(HuffmanTree，n，s1，s2)

操作结果：从霍夫曼树前n个元素中找到权值最小且无双亲的两个元素，并用s1、s2返回其下标。

Connect(i，s1，s2)

操作结果：分别将s1、s2变为i的左右孩子，i变为s1、s2的双亲。

Coding(HuffmanTree，HuffmanCode)

操作结果：用元素间已经建立好联系的霍夫曼树，对每个字符进行编码，

并用HuffmanCode返回其值。

Encoding(HuffmanCode，s，c)

操作结果：用储存着字符编码值的HuffmanCode对原字符串s进行编码，

用c返回编码值。

Decoding(HuffmanCode，c，s)

操作结果：用储存字符编码值的HuffmanCode对编码字符串c进行译码，

并用s返回译码结果。

}ADT HuffmanTree

1. 本程序主要有两个模块：主程序模块，霍夫曼树建立和编码译码模块。

调用关系为：主程序模块调用霍夫曼树建立和编码译码模块。

1. 详细设计
2. 霍夫曼树结构体类型

typedef struct \_HuffmanTree

{

char ch;

int weight;

int parent,lchild,rchild;

}\*HuffmanTree,HTree;

1. 初始化霍夫曼树，并赋初值的函数

int m=2\*n-1;

HT=(HuffmanTree)malloc((m+1)\*sizeof(HTree));//HT为结构体指针变量

HuffmanTree p=HT+1;

int i;

for(i=1;i<=n;i++,p++,element++)

{

p->ch=element->a; //element为储存着27个字符及其使用

频率的结构体数组的首地址

p->weight=element->fre;

p->parent=0;

p->lchild=0;

p->rchild=0;

}

for( ;i<=m;i++,p++)

{

p->ch='#';

p->weight=0;

p->parent=0;

p->lchild=0;

p->rchild=0;

}

1. 从前n个结构体元素中找到两个权值最小且无双亲的元素位置的函数

void Select(HuffmanTree HT,int n,int\* s1,int\* s2)

{

int i,m1,m2;

HuffmanTree begin=HT;

m1=m2=1000;//初始化为一个很大的量（无穷大）

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

{

if(HT->parent!=0)

continue;

else

{

if(HT->weight<=m1)

{

m1=HT->weight;

\*s1=i;

}

}

}

HT=begin;

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

{

if(HT->parent!=0)

continue;

else

{

if(HT->weight<=m2&&i!=\*s1)

{

m2=HT->weight;

\*s2=i;

}

}

}

}

1. 通过建立好的霍夫曼树给每个字符编码的函数

HC=(HuffmanCode)malloc((n+1)\*sizeof(char\*));//HC是一个char型的二级指针

char\* cd;

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

cd[n-1]='\0';

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));

strcpy(HC[i],&cd[start]);

}

free(cd);

1. 将原字符串编译为“01”字符码的函数

char STRING[200]={'\0'};

for(i=0;i<strlen(string);i++)

{

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

{

if(string[i]==HT[j].ch)

{

printf("%s",HC[j]);

strcat(STRING,HC[j]);//用STRING来存储“01”码

}

}

}

1. 将“01”字符码翻译回原字符串的函数

char code[20]={'\0'};

for(i=0,j=0;i<strlen(STRING);i++,j++)//STRING中存着“01”码

{

code[j]=STRING[i];

code[j+1]='\0';

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

{

if(strcmp(code,HC[k])==0)

{

code[0]='\0'; //相当于将code置空

j=-1;

printf("%c",HT[k].ch);

}

}

} InitHuffmanTree

1. 函数调用关系图 Select

Connect

main HuffmanCoding Encoding

Decoding

Coding

1. 调试分析
2. 开始编写程序时本想用多个函数分别进行霍夫曼树的初始化、建立霍夫曼树、编码27个字符、编码字符串和译码操作。但由于CodeBlocks的C语言编译器中无法使用“引用”，且受一次只能返回某一个变量的限制，想要做到用多个函数显得十分困难。所以除了select函数，其他的函数操作我都写进了HuffmanCoding中。
3. 调试过程中大多数的BUG都源于select函数未能正确找到当前结构体数组中权值最小的两个元素的位置，原因如下：m1,m2的初值不够大，导致循环完整个数组都没有找到更小的数值来更新m1,m2。
4. Select函数中应该可以只遍历一次数组就可以找到最小的两个值的，程序还可以改进。
5. 用户使用说明

无特殊说明，本程序未设置输入函数，程序中的字符串都是已经定义好的。

1. 测试结果

用课本中的字符频率表建立霍夫曼树得出的各字符编码，以及给定报文的编码译码输出如下：

:111 //此为空格符

A:1011

B:100000

C:00000

D:10101

E:010

F:110011

G:100010

H:0001

I:0111

J:1100001001

K:11000011

L:10100

M:110010

N:0110

O:1001

P:100001

Q:1100001000

R:0010

S:0011

T:1101

U:00001

V:1100000

W:110001

X:1100001011

Y:100011

Z:1100001010

1101000101110011111100001001010011000100010101111001011101110011111110010100011111110011101111000001001001001111101010

THIS PROGRAM IS MY FAVORITE

1. 附录

完整代码已同时上传GitHub，文件名为Huffman\_Tree.c