数据结构上机报告

题目：哈夫曼编/译码器

班级：04班 姓名： 学号： 完成日期：2020.12.05

一、需求分析

1.问题描述

问题描述：利用哈夫曼编码进行通信可以大大提高信道利用率，缩短信息传输时间，降低传输成本。但是，这要求在发送端通过一个编码系统对待传数据预先编码，在接收端将传来的数据进行译码（解码）。对于双工信道（即可以双向传输信息的信道），每端都需要一个完整的编/译码系统。试为这样的信息收发站设计一个哈夫曼编译码系统。

2.基本要求

一个完整的系统应具有以下功能：

（1）I：初始化（Initialization）。从终端读入字符集大小n，以及n个字符和n个权值，建立哈夫曼树，并将它存于文件hfmTree中。

（2）E：编码（Encoding）。利用已建好的哈夫曼树（如不在内存，则从文件hfmTree中读入），对文件ToBeTran中的正文进行编码，然后将结果存入文件CodeFile中。

（3）D：译码（Decoding）。利用已建好的哈夫曼树将文件CodeFile中的代码进行译码，结果存入文件TextFile中。

（4）P：印代码文件（Print）。将文件CodeFile以紧凑格式显示在终端上，每行50个代码。同时将此字符形式的编码文件写入文件CodePrin中。

（5）T：印哈夫曼树（Tree printing）。将已在内存中的哈夫曼树以直观的方式（树或凹入表形式）显示在终端上，同时将此字符形式的哈夫曼树写入文件TreePrint中。

3.预设需求

（1）编码结果以文本方式存储在文件CodeFile中。

（2）用户界面可以设计为“菜单”方式：显示上述功能符号，再加上“Q”，表示退出运行Quit。请用户键入一个选择功能符。此功能执行完毕后再显示此菜单，直至某次用户选择了“Q”为止。

二、概要设计:

程序用结构体数组类型存储哈夫曼树，先根据结点数初始化哈夫曼树，再进行哈夫曼树的构造及编码操作，并得出所需数据。

循环链表实现

1. 哈夫曼树的定义类型

typedef struct { //Huffman树的定义

char name;

unsigned int w; //权值

unsigned int pa; //父节点

unsigned int lch; //左子树

unsigned int rch; //右子树

} HTnode;

2、基本操作函数：

void Reverse(char \*str) { //字符串倒置，如此便不需要反向编码

int i,j;

char ch;

for(i=0,j=strlen(str)-1; i<j; i++,j--) {

ch=str[i];

str[i]=str[j];

str[j]=ch;

}

}

int min\_node(HTnode \*HT,int n) { //寻找Node中最小的节点

int i,min;

for(i=1; HT[i].pa; i++); //寻找根节点

min=i;

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

if(!HT[i].pa)

if(HT[i].w<HT[min].w)

min=i;

HT[min].pa=1;

return min;

}

3、主程序流程

a.输入叶子节点个数和操作方式，并从DataFile.txt及ToBeTran.txt中读取数据。

b.进行循环，先生成哈夫曼树，再按照给定的操作方式，进行编码/译码/输出哈夫曼树信息等操作。

c.将输出的数据存入给定文件中。

三、详细设计

各操作实现伪码

void Output\_HT(HTnode \*HT) { //输出Huffman树

int i;

printf(" 编号 名称 权值 父节点 左子树 右子树\n");

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

printf(" |%-5d %c %-5d %-5d %-6d %-4d|\n",i,

HT[i].name,HT[i].w,

HT[i].pa,HT[i].lch,

HT[i].rch);

}

}

void PrintHC(HCnode \*HC) { //输出Huffman编码

FILE \*f1,\*f2;

int i=0,j=0,x=0,n=0,wpl=0; //wpl带权路径长度

char temp[50];

if((f1=fopen("ToBeTran.txt","r"))==NULL) { //打开ToBeTran.txt

printf("open failed!\n");

exit(0);

}

if((f2=fopen("Code.txt","w"))==NULL) { //打开Code.txt

printf("open failed!\n");

exit(0);

}

printf("输出Huffman编码\n");

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

printf("%c-->%s",HC[i].name1,HC[i].pre);

x=strlen(HC[i].pre);

wpl+=(HC[i].data\*x);

printf("\n");

}

i=0;

while(!feof(f1)) { //从ToBeTran中读取字符并写入Code

temp[i]=fgetc(f1);

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

if(temp[i]==HC[j].name1) {

fprintf(f2,"%s",HC[j].pre);

}

i++;

}

fprintf(f2,"\b");

fclose(f1);

fclose(f2);

}

HTnode \*Create() { //构造Huffman树

FILE \*fp;

int i,Si,Sj; //找出来的两个最小子树保存在Si和Sj里面

HTnode \*HT;

HT=(HTnode \*)malloc((2\*N)\*sizeof(HTnode)); //给HT分配2N-1这里分配2N个

if((fp=fopen("DataFile.txt","rt"))==NULL) { //读取DataFile的文件

printf("can not find file DataFile failed!\n");

exit(0);

}

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

fscanf(fp,"%c,%d",&HT[i].name,&HT[i].w); //将Data的名称和权值赋值给HT

HT[i].lch=HT[i].rch=HT[i].pa=0; //初始化

}

printf("\n");

for(; i<=M; i++)

HT[i].lch=HT[i].rch=HT[i].pa=HT[i].w=0; //对其它多余的树进行初始化

for(i=N+1; i<=M; i++) {

Si=min\_node(HT,i); //寻找最小的节点

Sj=min\_node(HT,i);

HT[i].w=HT[Si].w+HT[Sj].w; //创建

HT[Si].pa=HT[Sj].pa=i;

HT[i].lch=Si;

HT[i].rch=Sj;

}

return HT;

}

HCnode \*Coding(HTnode \*HT) { //Huffman编码

int i,j,c,f;

HCnode \*HC;

char data[MAXSIZE];

HC=(HCnode \*)malloc(N\*sizeof(HCnode)); //叶子节点的个数

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

memset(data,'\0',MAXSIZE);

HC[i].name1=HT[i].name; //将树的名称给HC

HC[i].data=HT[i].w; //权值复制

for(c=i,j=0,f=HT[c].pa; f; c=f,f=HT[f].pa,j++) { //根节点循环终止，HC[f]的父节点给f，反向Huffman编码

if(HT[f].lch==c) data[j]='0'; //左子树为0，右子树为1

else data[j]='1';

}

HC[i].pre=(char \*)malloc((strlen(data)+1)\*sizeof(char));

Reverse(data); //逆置Huffman编码

for(j=0; j<strlen(data)+1; j++)

HC[i].pre[j]=data[j];

HC[i].pre[j]='\0'; //将Huffman编码给HC.pre[]

}

return HC;

}

void Decoding(HTnode \*HT) { //解码

FILE \*fp1,\*fp2;

char ch[50];

char cod[200];

int f,root,i=0,j=0;

if((fp1=fopen("CodeFile.txt","r"))==NULL) {

printf("open file failed");

exit(0);

}

if((fp2=fopen("Textfile.txt","w"))==NULL) {

printf("open file failed");

exit(0);

}

while(!feof(fp1)) { //将Codefile的文件给cod[]

fscanf(fp1,"%c",&cod[i]);

i++;

}

i=1;

while(HT[i].pa) {

i++; //寻找根节点

}

root=i;

i=0;

while(cod[i]!='\b') { //结尾

f=root;

while(HT[f].lch!=NULL) { //不为空

if(cod[i]=='0') f=HT[f].lch;

else f=HT[f].rch;

i++;

}

ch[j++]=HT[f].name;

}

ch[j]='\0';

printf("\nCodeFile.txt解码为:%s\n",ch);

i=0;

fprintf(fp2,"%s\b",ch);

fclose(fp1);

fclose(fp2);

}

void Output() { //输出Textfile.txt、ToBeTran.txt、Code.txt、CodeFile.txt、Textfile.txt

int c;

char b;

FILE \*fp1,\*fp2,\*fp3,\*fp4,\*fp5;

if((fp1=fopen("Datafile.txt","r"))==NULL) {

printf("open file0 failed!\n");

exit(0);

}

if((fp1=fopen("Datafile.txt","r"))==NULL) {

printf("open file0 failed!\n");

exit(0);

}

if((fp2=fopen("ToBeTran.txt","r"))==NULL) {

printf("open file ToBeTran failed!\n");

exit(0);

}

if((fp3=fopen("Code.txt","r"))==NULL) {

printf("open file Code failed!\n");

exit(0);

}

if((fp4=fopen("CodeFile.txt","r"))==NULL) {

printf("open file CodeFile failed!\n");

exit(0);

}

if((fp5=fopen("Textfile.txt","r"))==NULL) {

printf("open file Textfile failed!\n");

exit(0);

}

printf("\n输出DataFlie中的字符及其权值:\n");

while(!feof(fp1)) {

fscanf(fp1,"%c,%d", &b,&c);

printf("'%c'---%d\n",b,c);

}

printf("\n\n输出ToBeTran:\n");

while(!feof(fp2)) {

fscanf(fp2,"%c",&b);

printf("%c",b);

}

printf("\n\n输出Code.txt:\n");

while(!feof(fp3)) {

fscanf(fp3,"%c",&b);

printf("%c",b);

}

printf("\n\n输出CodeFile.txt:\n");

while(!feof(fp4)) {

fscanf(fp4,"%c",&b);

printf("%c",b);

}

printf("\n\n输出Textfile.txt:\n");

while(!feof(fp5)) {

fscanf(fp5,"%c",&b);

printf("%c",b);

}

printf("\n");

fclose(fp1);

fclose(fp2);

fclose(fp3);

fclose(fp4);

fclose(fp5);

}

void Output2(int n) { //输出Textfile.txt、ToBeTran.txt、Code.txt、CodeFile.txt、Textfile.txt

int c;

char b;

FILE \*fp1,\*fp2,\*fp3,\*fp4,\*fp5;

switch(n) {

case 2:

printf("\n输出ToBeTran:\n");

if((fp2=fopen("ToBeTran.txt","r"))==NULL) {

printf("open file ToBeTran failed!\n");

exit(0);

}

while(!feof(fp2)) {

fscanf(fp2,"%c",&b);

printf("%c",b);

}

fclose(fp2);

break;

case 3:

printf("\n\n输出Code.txt:\n");

if((fp3=fopen("Code.txt","r"))==NULL) {

printf("open file Code failed!\n");

exit(0);

}

while(!feof(fp3)) {

fscanf(fp3,"%c",&b);

printf("%c",b);

}

fclose(fp3);

break;

default:

printf("ERROR!");

}

printf("\n");

}

四、调试分析

1、调试问题

a.初期实现程序时，对文件指针操作不太熟悉，未能正确读取数据。

b.各操作实现时没有连贯性，重复操作造成函数重复调用会导致bug。

1. 算法改进设想

目前程序存在交互性较差、时间复杂度较高等问题，后期可通过完善图形化界面，增强程序健壮性，优化编码算法提高程序的可用性。

1. 经验和体会

熟悉了文件指针操作和几种读入方式，对哈夫曼树的构造和哈夫曼编码有了更深的了解。学习了前缀编码方式。

五、用户使用说明

用户需要将所需数据存入相应文件中，运行所给文件夹下exe程序，输入叶子结点数，以及所要进行的操作，即可得到结果。

六、测试结果

输入样例

权值数据及待编码文本分别存储在DataFile.txt及ToBeTran.txt中

从终端输入：

27 T Q

27 E Q

27 D Q

输出样例

读取DataFile中的字符及权值，建立Huffman树:

读取27个叶节点的权值并生成Hufman Tree表格形式:

编号 名称 权值 父节点 左子树 右子树

|1 A 64 45 0 0 |

|2 B 13 33 0 0 |

|3 C 22 37 0 0 |

|4 D 32 39 0 0 |

|5 E 103 48 0 0 |

|6 F 21 36 0 0 |

|7 G 15 33 0 0 |

|8 H 47 41 0 0 |

|9 L 57 43 0 0 |

|10 J 1 28 0 0 |

|11 K 5 31 0 0 |

|12 L 32 39 0 0 |

|13 M 20 36 0 0 |

|14 N 57 43 0 0 |

|15 O 63 44 0 0 |

|16 P 15 34 0 0 |

|17 Q 1 28 0 0 |

|18 R 48 42 0 0 |

|19 S 51 42 0 0 |

|20 T 80 46 0 0 |

|21 U 23 37 0 0 |

|22 V 8 32 0 0 |

|23 W 18 35 0 0 |

|24 X 1 29 0 0 |

|25 Y 16 34 0 0 |

|26 Z 1 29 0 0 |

|27 186 50 0 0 |

|28 2 30 10 17 |

|29 2 30 24 26 |

|30 4 31 28 29 |

|31 9 32 30 11 |

|32 17 35 22 31 |

|33 28 38 2 7 |

|34 31 38 16 25 |

|35 35 40 32 23 |

|36 41 40 13 6 |

|37 45 41 3 21 |

|38 59 44 33 34 |

|39 64 45 4 12 |

|40 76 46 35 36 |

|41 92 47 37 8 |

|42 99 47 18 19 |

|43 114 48 9 14 |

|44 122 49 38 15 |

|45 128 49 1 39 |

|46 156 50 40 20 |

|47 191 51 41 42 |

|48 217 51 5 43 |

|49 250 52 44 45 |

|50 342 52 46 27 |

|51 408 53 47 48 |

|52 592 53 49 50 |

|53 1000 0 51 52 |

对Huffman树进行编码:

输出Huffman编码

A-->1010

B-->100000

C-->00000

D-->10110

E-->010

F-->110011

G-->100001

H-->0001

L-->0110

J-->1100001000

K-->11000011

L-->10111

M-->110010

N-->0111

O-->1001

P-->100010

Q-->1100001001

R-->0010

S-->0011

T-->1101

U-->00001

V-->1100000

W-->110001

X-->1100001010

Y-->100011

Z-->1100001011

-->111

带权路径长度(WPL) is 4124

-------------------------------------------------------------------

将ToBeTran的内容改写成编码写入Code.txt

输出ToBeTran:

THIS PAROGRAM IS MY FAVORITE

输出Code.txt:

11010001001111110001010100010100110000100101010110010111001111111001010001111111001110101100000100100101101010111

-------------------------------------------------------------------

对文件CodeFile.data中的代码进行解码形成原文，结果存入文件Textfile.txt中

1. 附录

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

#define MAXSIZE 50

typedef struct { //Huffman树的定义

char name;

unsigned int w; //权值

unsigned int pa; //父节点

unsigned int lch; //左子树

unsigned int rch; //右子树

} HTnode;

typedef struct { //Huffman编码

int data;

char name1;

char \*pre;

} HCnode;

int N,M;

HCnode \*HC;

HTnode \*HT;

void Reverse(char \*str) { //字符串倒置，如此便不需要反向编码

int i,j;

char ch;

for(i=0,j=strlen(str)-1; i<j; i++,j--) {

ch=str[i];

str[i]=str[j];

str[j]=ch;

}

}

int min\_node(HTnode \*HT,int n) { //寻找Node中最小的节点

int i,min;

for(i=1; HT[i].pa; i++); //寻找根节点

min=i;

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

if(!HT[i].pa)

if(HT[i].w<HT[min].w)

min=i;

HT[min].pa=1;

return min;

}

void Output\_HT(HTnode \*HT) { //输出Huffman树

int i;

printf(" 编号 名称 权值 父节点 左子树 右子树\n");

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

printf(" |%-5d %c %-5d %-5d %-6d %-4d|\n",i,

HT[i].name,HT[i].w,

HT[i].pa,HT[i].lch,

HT[i].rch);

}

}

void PrintHC(HCnode \*HC) { //输出Huffman编码

FILE \*f1,\*f2;

int i=0,j=0,x=0,n=0,wpl=0; //wpl带权路径长度

char temp[50];

if((f1=fopen("ToBeTran.txt","r"))==NULL) { //打开ToBeTran.txt

printf("open failed!\n");

exit(0);

}

if((f2=fopen("Code.txt","w"))==NULL) { //打开Code.txt

printf("open failed!\n");

exit(0);

}

printf("输出Huffman编码\n");

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

printf("%c-->%s",HC[i].name1,HC[i].pre);

x=strlen(HC[i].pre);

wpl+=(HC[i].data\*x);

printf("\n");

}

i=0;

while(!feof(f1)) { //从ToBeTran中读取字符并写入Code

temp[i]=fgetc(f1);

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

if(temp[i]==HC[j].name1) {

fprintf(f2,"%s",HC[j].pre);

}

i++;

}

fprintf(f2,"\b");

fclose(f1);

fclose(f2);

}

HTnode \*Create() { //构造Huffman树

FILE \*fp;

int i,Si,Sj; //找出来的两个最小子树保存在Si和Sj里面

HTnode \*HT;

HT=(HTnode \*)malloc((2\*N)\*sizeof(HTnode)); //给HT分配2N-1这里分配2N个

if((fp=fopen("DataFile.txt","rt"))==NULL) { //读取DataFile的文件

printf("can not find file DataFile failed!\n");

exit(0);

}

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

fscanf(fp,"%c,%d",&HT[i].name,&HT[i].w); //将Data的名称和权值赋值给HT

HT[i].lch=HT[i].rch=HT[i].pa=0; //初始化

}

printf("\n");

for(; i<=M; i++)

HT[i].lch=HT[i].rch=HT[i].pa=HT[i].w=0; //对其它多余的树进行初始化

for(i=N+1; i<=M; i++) {

Si=min\_node(HT,i); //寻找最小的节点

Sj=min\_node(HT,i);

HT[i].w=HT[Si].w+HT[Sj].w; //创建

HT[Si].pa=HT[Sj].pa=i;

HT[i].lch=Si;

HT[i].rch=Sj;

}

return HT;

}

HCnode \*Coding(HTnode \*HT) { //Huffman编码

int i,j,c,f;

HCnode \*HC;

char data[MAXSIZE];

HC=(HCnode \*)malloc(N\*sizeof(HCnode)); //叶子节点的个数

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

memset(data,'\0',MAXSIZE);

HC[i].name1=HT[i].name; //将树的名称给HC

HC[i].data=HT[i].w; //权值复制

for(c=i,j=0,f=HT[c].pa; f; c=f,f=HT[f].pa,j++) { //根节点循环终止，HC[f]的父节点给f，反向Huffman编码

if(HT[f].lch==c) data[j]='0'; //左子树为0，右子树为1

else data[j]='1';

}

HC[i].pre=(char \*)malloc((strlen(data)+1)\*sizeof(char));

Reverse(data); //逆置Huffman编码

for(j=0; j<strlen(data)+1; j++)

HC[i].pre[j]=data[j];

HC[i].pre[j]='\0'; //将Huffman编码给HC.pre[]

}

return HC;

}

void Decoding(HTnode \*HT) { //解码

FILE \*fp1,\*fp2;

char ch[50];

char cod[200];

int f,root,i=0,j=0;

if((fp1=fopen("CodeFile.txt","r"))==NULL) {

printf("open file failed");

exit(0);

}

if((fp2=fopen("Textfile.txt","w"))==NULL) {

printf("open file failed");

exit(0);

}

while(!feof(fp1)) { //将Codefile的文件给cod[]

fscanf(fp1,"%c",&cod[i]);

i++;

}

i=1;

while(HT[i].pa) {

i++; //寻找根节点

}

root=i;

i=0;

while(cod[i]!='\b') { //结尾

f=root;

while(HT[f].lch!=NULL) { //不为空

if(cod[i]=='0') f=HT[f].lch;

else f=HT[f].rch;

i++;

}

ch[j++]=HT[f].name;

}

ch[j]='\0';

printf("\nCodeFile.txt解码为:%s\n",ch);

i=0;

fprintf(fp2,"%s\b",ch);

fclose(fp1);

fclose(fp2);

}

void Output() { //输出Textfile.txt、ToBeTran.txt、Code.txt、CodeFile.txt、Textfile.txt

int c;

char b;

FILE \*fp1,\*fp2,\*fp3,\*fp4,\*fp5;

if((fp1=fopen("Datafile.txt","r"))==NULL) {

printf("open file0 failed!\n");

exit(0);

}

if((fp1=fopen("Datafile.txt","r"))==NULL) {

printf("open file0 failed!\n");

exit(0);

}

if((fp2=fopen("ToBeTran.txt","r"))==NULL) {

printf("open file ToBeTran failed!\n");

exit(0);

}

if((fp3=fopen("Code.txt","r"))==NULL) {

printf("open file Code failed!\n");

exit(0);

}

if((fp4=fopen("CodeFile.txt","r"))==NULL) {

printf("open file CodeFile failed!\n");

exit(0);

}

if((fp5=fopen("Textfile.txt","r"))==NULL) {

printf("open file Textfile failed!\n");

exit(0);

}

printf("\n输出DataFlie中的字符及其权值:\n");

while(!feof(fp1)) {

fscanf(fp1,"%c,%d", &b,&c);

printf("'%c'---%d\n",b,c);

}

printf("\n\n输出ToBeTran:\n");

while(!feof(fp2)) {

fscanf(fp2,"%c",&b);

printf("%c",b);

}

printf("\n\n输出Code.txt:\n");

while(!feof(fp3)) {

fscanf(fp3,"%c",&b);

printf("%c",b);

}

printf("\n\n输出CodeFile.txt:\n");

while(!feof(fp4)) {

fscanf(fp4,"%c",&b);

printf("%c",b);

}

printf("\n\n输出Textfile.txt:\n");

while(!feof(fp5)) {

fscanf(fp5,"%c",&b);

printf("%c",b);

}

printf("\n");

fclose(fp1);

fclose(fp2);

fclose(fp3);

fclose(fp4);

fclose(fp5);

}

void Output2(int n) { //输出Textfile.txt、ToBeTran.txt、Code.txt、CodeFile.txt、Textfile.txt

int c;

char b;

FILE \*fp1,\*fp2,\*fp3,\*fp4,\*fp5;

switch(n) {

case 2:

printf("\n输出ToBeTran:\n");

if((fp2=fopen("ToBeTran.txt","r"))==NULL) {

printf("open file ToBeTran failed!\n");

exit(0);

}

while(!feof(fp2)) {

fscanf(fp2,"%c",&b);

printf("%c",b);

}

fclose(fp2);

break;

case 3:

printf("\n\n输出Code.txt:\n");

if((fp3=fopen("Code.txt","r"))==NULL) {

printf("open file Code failed!\n");

exit(0);

}

while(!feof(fp3)) {

fscanf(fp3,"%c",&b);

printf("%c",b);

}

fclose(fp3);

break;

default:

printf("ERROR!");

}

printf("\n");

}

int main() {

int i;

char c;

FILE \*fp;

char \*\*code; //构建一个二维数组解码用

char ope;

printf("根据Data文件请输入叶节点个数 : ");

scanf("%d",&N);

printf("请输入所要进行的操作：");

c = getchar();

c = getchar();

while(c != 'Q') {

if(N>0&&N<=500) {

M=2\*N-1;

code=(char\*\*)malloc(100\*sizeof(char));

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

code[i]=(char \*)malloc(8\*sizeof(char));

HT=Create();

if(c == 'T') Output\_HT(HT);

if(c == 'E') {

printf("对Huffman树进行编码:\n");

HC=Coding(HT);

PrintHC(HC);

printf("-------------------------------------------------------------------\n");

printf("将ToBeTran的内容改写成编码写入Code.txt\n");

Output2(3);

}

if(c == 'D') {

Output2(2);

printf("-------------------------------------------------------------------\n");

printf("对文件CodeFile.data中的代码进行解码形成原文，结果存入文件Textfile.txt中\n");

}

printf("\n请输入所要进行的操作：");

c = getchar();

c = getchar();

}

else printf("请重新输入:\n");

}

return 0;

}