**1.问题描述**

哈弗曼树的编码与译码

— 功能：实现对任何类型文件的压缩与解码

— 输入：源文件，压缩文件

— 输出：解码正确性判定，统计压缩率、编码与解码速度

— 要求： 使用边编码边统计符号概率的方法（自适应Huffman编码） 和事先统计概率的方法（静态Huffman编码）

**2.1程序清单**

**程序书签：**

1. [**main函****数**](file:///C:\Users\dell\Desktop\哈弗曼树的文件压缩和解压实验报告(C语言).doc#主函数)
2. [**压缩函****数**](file:///C:\Users\dell\Desktop\哈弗曼树的文件压缩和解压实验报告(C语言).doc#压缩函数)
3. [**select****函数**](file:///C:\Users\dell\Desktop\哈弗曼树的文件压缩和解压实验报告(C语言).doc#select函数)
4. [**encode****函数**](file:///C:\Users\dell\Desktop\哈弗曼树的文件压缩和解压实验报告(C语言).doc#encode编码函数)
5. [**解压****函数**](file:///C:\Users\dell\Desktop\哈弗曼树的文件压缩和解压实验报告(C语言).doc#解压函数)

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**#include <conio.h>**

**#include <time.h>**

**struct node{**

**long weight; //权值**

**unsigned char ch;//字符**

**int parent,lchild,rchild;**

**char code[256];//编码的位数最多为256位**

**int CodeLength;//编码长度**

**}hfmnode[512];**

**void compress();**

**void uncompress();**

**//主函数**

**void main()**

**{**

**int choice;**

**printf("请选择1~3：\n");**

**printf("1.压缩文件\n");**

**printf("2.解压文件\n");**

**printf("3.退出！\n");**

**scanf("%d",&choice);**

**if(choice==1)compress();**

**else if(choice==2)uncompress();**

**else if(choice==3)return;**

**else printf("输入错误！");**

**}**

**//压缩函数**

**void compress()**

**{**

**int i,j;**

**char infile[20],outfile[20];**

**FILE \*ifp,\*ofp;**

**unsigned char c;//**

**long FileLength,filelength=0;**

**int n,m;//叶子数和结点数**

**int s1,s2; //权值最小的两个结点的标号**

**char codes[256];**

**long sumlength=0;**

**float rate,speed;**

**int count=0;**

**clock\_t start1, start2,finish1,finish2;**

**double duration1,duration2;**

**void encode(struct node \*nodep,int n);//编码函数**

**int select(struct node \*nodep,int pose);//用于建哈弗曼树中选择权值最小的结点的函数**

**printf("请输入要压缩的文件名:");**

**scanf("%s",infile);**

**ifp=fopen(infile,"rb");**

**if(ifp==NULL)**

**{**

**printf("文件名输入错误，文件不存在!\n");**

**return;**

**}**

**printf("请输入目标文件名:");**

**scanf("%s",outfile);**

**ofp=fopen(outfile,"wb");**

**if(ofp==NULL)**

**{**

**printf("文件名输入错误，文件不存在!\n");**

**return;**

**}**

**start1=clock() ;//开始计时1**

**//统计文件中字符的种类以及各类字符的个数**

**//先用字符的ASCII码值代替结点下标**

**FileLength=0;**

**while(!feof(ifp))**

**{**

**fread(&c,1,1,ifp);**

**hfmnode[c].weight++;**

**FileLength++;**

**}**

**FileLength--; //文件中最后一个字符的个数会多统计一次，所以要减一**

**hfmnode[c].weight--;**

**//再将ASCII转换为字符存入到结点的ch成员里，同时给双亲、孩子赋初值-1**

**n=0;**

**for(i=0;i<256;i++)**

**if(hfmnode[i].weight!=0)**

**{**

**hfmnode[i].ch=(unsigned char)i;**

**n++;//叶子数**

**hfmnode[i].lchild=hfmnode[i].rchild=hfmnode[i].parent=-1;**

**}**

**m=2\*n-1;//哈弗曼树结点总数**

**j=0;**

**for(i=0;i<256;i++)//去掉权值为0的结点**

**if(hfmnode[i].weight!=0)**

**{**

**hfmnode[j]=hfmnode[i];**

**j++;**

**}**

**for(i=n;i<m;i++)//初始化根结点**

**{**

**hfmnode[i].lchild=hfmnode[i].rchild=-1;**

**hfmnode[i].parent=-1;**

**}**

**//建立哈弗曼树**

**for(i=n;i<m;i++)**

**{**

**s1=select(hfmnode,i-1);**

**hfmnode[i].lchild=s1;**

**hfmnode[s1].parent=i;**

**s2=select(hfmnode,i-1);**

**hfmnode[i].rchild=s2;**

**hfmnode[s2].parent=i;**

**hfmnode[i].weight=hfmnode[s1].weight+hfmnode[s2].weight;**

**}**

**//编码**

**encode(hfmnode,n);**

**finish1=clock();**

**duration1=(double)(finish1- start1) / CLOCKS\_PER\_SEC;**

**/\*printf( "哈弗曼树编码用时为：%f seconds\n", duration1 );\*/**

**printf("编码完成，是否查看编码信息: y or n?\n");**

**c=getch();**

**if(c=='y')**

**{ printf("\n");**

**printf("叶子数为%d,结点数为%d\n",n,m);**

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

**printf("%d号叶子结点的权值为：%ld,双亲为：%d,左右孩子：%d,编码为：%s\n",**

**i,hfmnode[i].weight,hfmnode[i].parent,hfmnode[i].lchild,hfmnode[i].code);**

**}**

**start2=clock() ;//开始计时2**

**fseek(ifp,0,SEEK\_SET);//将ifp指针移到文件开头位置**

**fwrite(&FileLength,4,1,ofp);//将FileLength写入目标文件的前4个字节的位置**

**fseek(ofp,8,SEEK\_SET);//再将目标文件指针ofp移到距文件开头8个字节位置**

**codes[0]=0;**

**//将编码信息写入目标文件**

**while(!feof(ifp))**

**{**

**fread(&c,1,1,ifp);**

**filelength++;**

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

**if(c==hfmnode[i].ch) break; //ch必须也为unsigned 型**

**strcat(codes,hfmnode[i].code);**

**while(strlen(codes)>=8)**

**{**

**for(i=0;i<8;i++)//将codes的前8位01代码表示的字符存入c**

**{**

**if(codes[i]=='1')**

**c=(c<<1)|1;**

**else c=c<<1;**

**}**

**fwrite(&c,1,1,ofp); //将新的字符写入目标文件**

**sumlength++;**

**strcpy(codes,codes+8);//更新codes的值**

**}**

**if(filelength==FileLength) break;**

**}**

**//再将剩余的不足8位的01代码补全8位，继续写入**

**if(strlen(codes)>0)**

**{**

**strcat(codes,"00000000");**

**for(i=0;i<8;i++)**

**{**

**if(codes[i]=='1')**

**c=(c<<1)|1;**

**else c=c<<1;**

**}**

**fwrite(&c,1,1,ofp);**

**sumlength++;**

**}**

**sumlength+=8;**

**printf("编码区总长为：%ld个字节\n",sumlength-8);**

**//将sumlength和n的值写入目标文件，为的是方便解压**

**fseek(ofp,4,SEEK\_SET);**

**fwrite(&sumlength,4,1,ofp);//把sumlength写进目标文件的第5-8个字节里**

**fseek(ofp,sumlength,SEEK\_SET);**

**fwrite(&n,4,1,ofp);//把叶子数n写进编码段后面的4个字节的位置**

**//为方便解压，把编码信息存入n后面的位置**

**//存储方式为：n\*（字符值（1个字节）+该字符的01编码的位数（1个字节）+编码（字节数不确定，用count来计算总值））**

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

**{**

**fwrite(&(hfmnode[i].ch),1,1,ofp);**

**c=hfmnode[i].CodeLength;//编码最长为256位，因此只需用一个字节存储**

**fwrite(&c,1,1,ofp);**

**//写入字符的编码**

**if(hfmnode[i].CodeLength%8!=0)**

**for(j=hfmnode[i].CodeLength%8;j<8;j++)//把编码不足8位的在低位补0,赋值给C，再把C写入**

**strcat(hfmnode[i].code,"0");**

**while(hfmnode[i].code[0]!=0)//开始存入编码，每8位二进制数存入一个字节**

**{**

**c=0;**

**for(j=0;j<8;j++)**

**{**

**if(hfmnode[i].code[j]=='1')**

**c=(c<<1)|1;**

**else c=c<<1;**

**}**

**strcpy(hfmnode[i].code,hfmnode[i].code+8);//编码前移8位，继续存入编码**

**count++; //编码占的字节数的总值**

**fwrite(&c,1,1,ofp);**

**}**

**}**

**printf("\n");**

**finish2=clock();**

**duration2=(double)(finish2- start2) / CLOCKS\_PER\_SEC;**

**/\*printf( "写入目标文件用时为：%f seconds\n", duration2);\*/**

**printf( "压缩用时为：%f seconds\n", duration1+duration2);**

**speed=(float)FileLength/(duration1+duration2)/1000;**

**printf("\n压缩速率为：%5.2f KB/S\n",speed);**

**printf("\n");**

**printf("源文件长度为：%ld个字节\n",FileLength);**

**sumlength=sumlength+4+n\*2+count; //计算压缩后文件的长度**

**printf("压缩后文件长度为：%ld个字节\n",sumlength);**

**rate=(float)sumlength/(float)FileLength;**

**printf("压缩率(百分比)为：%4.2f%%%\n",rate\*100);**

**fclose(ifp);**

**fclose(ofp);**

**return;**

**}**

[**//返回书签**](file:///C:\Users\dell\Desktop\哈弗曼树的文件压缩和解压实验报告(C语言).doc#_top)

**//建立哈弗曼树中用于选择最小权值结点的函数**

**int select(struct node \*nodep,int pose)**

**{**

**int i;**

**int s1;**

**long min=2147483647;//s初值为long型的最大值**

**for(i=0;i<=pose;i++)**

**{**

**if(nodep[i].parent!=-1)continue;**

**if(nodep[i].weight<min)**

**{**

**min=nodep[i].weight;**

**s1=i;**

**}**

**}**

**return s1;**

**}**

[**//返回书签**](file:///C:\Users\dell\Desktop\哈弗曼树的文件压缩和解压实验报告(C语言).doc#_top)

**//哈弗曼编码函数**

**void encode(struct node \*nodep,int n)**

**{ //从叶子向根求每个字符的哈弗曼编码**

**int start;**

**int i,f,c;**

**char codes[256];**

**codes[n-1]='\0'; //编码结束符**

**for(i=0;i<n;i++) //逐个字符求哈弗曼编码**

**{**

**start=n-1;**

**for(c=i,f=nodep[i].parent;f!=-1;c=f,f=nodep[f].parent)**

**{**

**start--;**

**if(nodep[f].lchild==c)**

**codes[start]='0';**

**else codes[start]='1';**

**}**

**strcpy(nodep[i].code,&codes[start]);**

**nodep[i].CodeLength=strlen(nodep[i].code);**

**}**

**}**

[**//返回书****签**](file:///C:\Users\dell\Desktop\哈弗曼树的文件压缩和解压实验报告(C语言).doc#_top)

**//解压函数**

**void uncompress() //解压文件**

**{**

**clock\_t start, finish;**

**double duration;**

**FILE \*ifp,\*ofp;**

**char infile[20],outfile[20];**

**long FileLength,sumlength,filelength;**

**int n,m;**

**int i,j,k;**

**char buf[256],codes[256];**

**unsigned char c;**

**int maxlength;**

**float speed;**

**printf("请输入要解压的文件名:");**

**scanf("%s",infile);**

**ifp=fopen(infile,"rb");**

**if(ifp==NULL)**

**{**

**printf("文件名输入错误，文件不存在!\n");**

**return;**

**}**

**printf("请输入目标文件名:");**

**scanf("%s",outfile);**

**ofp=fopen(outfile,"wb");**

**if(ofp==NULL)**

**{**

**printf("文件名输入错误，文件不存在!\n");**

**return;**

**}**

**start=clock() ;//开始计时**

**fread(&FileLength,4,1,ifp);//从压缩文件读出FileLength、sumlength**

**fread(&sumlength,4,1,ifp);**

**fseek(ifp,sumlength,SEEK\_SET); //利用sumlength读出n的值**

**fread(&n,4,1,ifp);**

**printf("\n解码信息:源文件长度为%d个字节,字符种类n=%d\n",FileLength,n);**

**for(i=0;i<n;i++)//读结点信息**

**{**

**fread(&hfmnode[i].ch,1,1,ifp);//字符**

**fread(&c,1,1,ifp);//编码长度**

**hfmnode[i].CodeLength=c;**

**hfmnode[i].code[0]=0;**

**if(hfmnode[i].CodeLength%8>0) m=hfmnode[i].CodeLength/8+1;//m为编码占的字节数**

**else m=hfmnode[i].CodeLength/8;**

**for(j=0;j<m;j++)//根据字节长度m读出编码**

**{**

**fread(&c,1,1,ifp);//此处c为01编码转换成的字符**

**itoa(c,buf,2);//字符型编码转换成二进制型（首位为1）**

**//如果编码不够8位，则说明缺少了8-k位0，因此应先在前面空缺位写0**

**for(k=8;k>strlen(buf);k--)**

**{**

**strcat(hfmnode[i].code,"0");**

**}**

**//再把二进制编码存进hfmnode.code中**

**strcat(hfmnode[i].code,buf);**

**}**

**hfmnode[i].code[hfmnode[i].CodeLength]=0;//去掉编码中多余的0**

**}**

**//找出编码长度的最大值**

**maxlength=0;**

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

**if(hfmnode[i].CodeLength>maxlength)**

**maxlength=hfmnode[i].CodeLength;**

**//开始写入目标文件**

**fseek(ifp,8,SEEK\_SET); //指针指向编码区，开始解码**

**filelength=0;**

**codes[0]=0;**

**buf[0]=0;**

**while(1)**

**{**

**while(strlen(codes)<maxlength)//codes小于编码长度的最大值时，继续读码**

**{**

**fread(&c,1,1,ifp);**

**itoa(c,buf,2);//还原编码**

**for(k=8;k>strlen(buf);k--)**

**{**

**strcat(codes,"0");//把缺掉的0补上**

**}**

**strcat(codes,buf);//codes中此时存的为一串01编码**

**}**

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

**{ //在codes中查找能使其前weight位和hfmnode.code相同的i值，weight即为codelength**

**if(memcmp(hfmnode[i].code,codes,(unsigned int)hfmnode[i].CodeLength)==0) break;**

**}**

**strcpy(codes,codes+hfmnode[i].CodeLength);//更新codes的值**

**c=hfmnode[i].ch;**

**fwrite(&c,1,1,ofp);**

**filelength++;**

**if(filelength==FileLength) break;//写入结束**

**}**

**finish = clock();**

**duration = (double)(finish - start) / CLOCKS\_PER\_SEC;**

**printf( "\n解压完成，解压用时为：%f seconds\n", duration );**

**fseek(ifp,0,SEEK\_SET);**

**FileLength=0;**

**while(!feof(ifp))**

**{**

**fread(&c,1,1,ifp);**

**FileLength++;**

**}**

**FileLength--;**

**speed=(float)FileLength/duration/1000;**

**/\*printf("此文件长度为：%ld个字节\n",FileLength);\*/**

**printf("\n解压速度为：%5.2fKB/S\n",speed);**

**fclose(ifp);**

**fclose(ofp);**

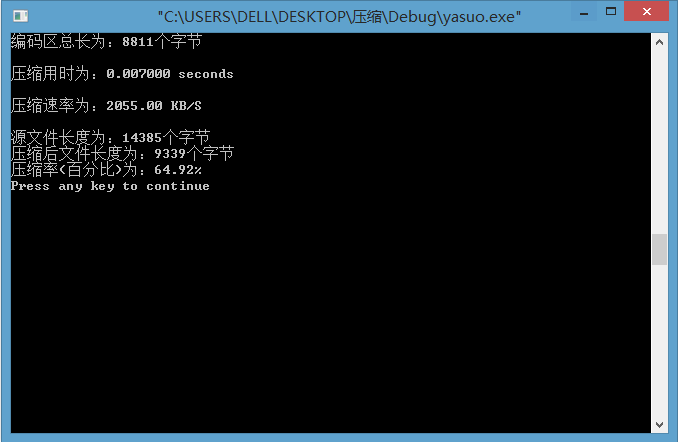
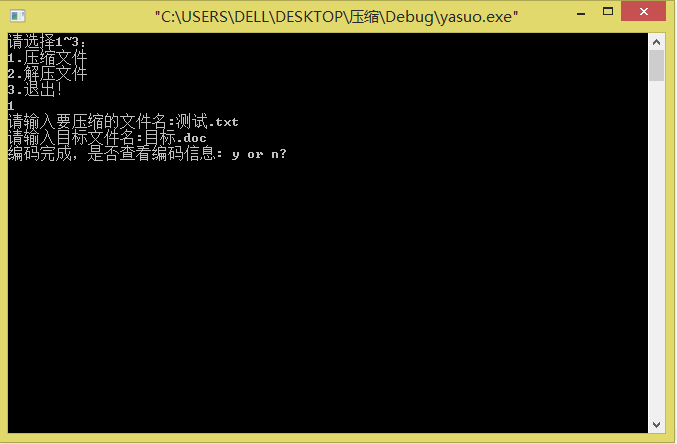
**return;**

**}**

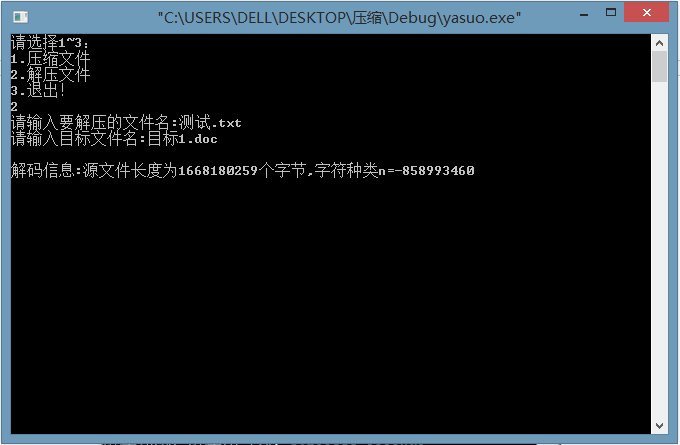
**2.2程序运行结果：**

1.对文件”测试.txt”进行压缩，压缩后存储在文件”目标.doc”中，压缩速率为：2055.00KB/S，压缩率为64.92%。

程序运行结果截图如下：



2.再对”测试.txt“文件进行解压，目标文件为“目标1.doc “。程序运行结果如下：



**2.3算法描述**

1. **压缩文件**

压缩文件时要先对源文件进行统计，统计字符的种类及出现的次数（即权值）。统计完成之后，建立哈弗曼树：每次选取权值最小且无parent的结点作为左右孩子，建成一棵二叉树，且设置新的二叉树的根结点的权值为其左右孩子的权值之和。直至建成含有**2\*n-1**个结点的哈弗曼树。

给每种字符进行编码。按照从叶子到根的顺序求其编码。算法和图示如下：

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

**{**

**start=n-1;**

**for(c=i,f=nodep[i].parent;f!=-1;c=f,f=nodep[f].parent)**

**{**

**start--;**

**if(nodep[f].lchild==c)**

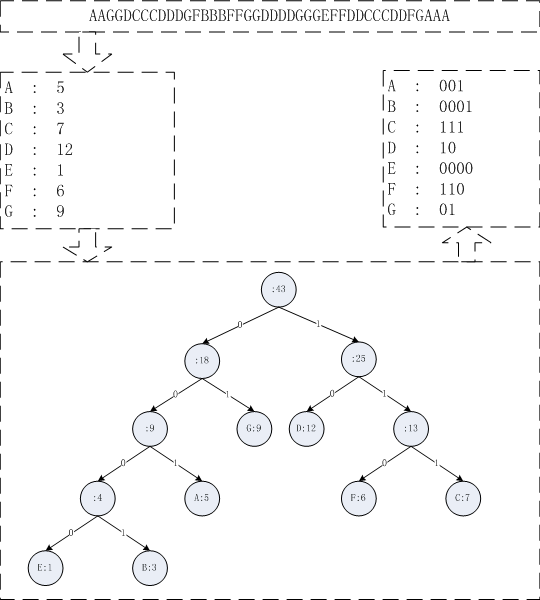
**codes[start]='0';**

**else codes[start]='1';**

**}**

**strcpy(nodep[i].code,&codes[start]);**

**}**

****

编码完成之后，开始对源文件进行压缩。

1. 从源文件读一个字符，从叶子结点中找出和此字符相同的字符结点，将其编码写入一个临时字符组codes；
2. 当codes的长度大于等于8时，将其前8位转换成字符写入目标文件中；
3. 重复1和2此过程，直至读完源文件中的所有字符；
4. 若codes最后还有剩余的不足8位的01代码，则将其低位补0至8位，再写入目标文件。

同时为了便于解码，将源文件的长度FileLength、编码区的长度以及叶子结点的个数n、每个叶子结点的信息也存入目标文件。存储方式如下图所示：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| FileLength  4B | Sumlength  4B | 源文件编码区 | 叶子数n  4B | 叶子结点信息 | | | |
| 字符值1B | 字符的编码位数1B | 字符的编码 | ............... |

|—— 1个结点的信息——|

**sumlength**

**（2）解压文件**

从被压缩的文件中读出FileLength、n的值，以及每个叶子结点的信息：字符、字符对应的编码。

开始解码：

1.从被压缩的文件编码区读出一个字符，将其值转化成二进制形式（不足8位的高位要补0），存入codes中，直至codes的长度不小于所有叶子结点的编码的长度；

2.用for循环查找出第一个和codes的01字符串匹配的叶子结点编码，将该叶子结点的字符写入目标文件，并将codes的字符串前移，前移位数=该叶子结点编码的长度。

3.重复1和2过程，直至写入的字符数与源文件的长度FileLength相同。