**数据结构课程设计**

**班级：1615403**

**学号：161540205**

**姓名：张佳钰**

**指导老师：孙涵**

7、Huffman编码与解码 (选做)（Huffman编码、二叉树）

[问题描述]

对一篇英文文章，统计各字符出现的次数，实现Huffman编码，以及对编码结果的解码。

[基本要求]

（1） 输出每个字符出现的次数和编码，其中求最小权值要求用堆实现。

（2） 在Huffman编码后，要将编码表和英文文章编码结果保存到文件中，编码结果必须是二进制形式，即0 1的信息用比特位表示，不能用字符’0’和’1’表示。

（3） 提供读编码文件生成原文件的功能。

采用的数据结构：

静态数组

算法思想：

最优树

源程序：

#include<iostream>

using namespace std;

#include<fstream>

#include<stack>

#include<string.h>

#include<algorithm>

#include<cctype>

#include<string>

#include<string.h>

#include<map>

#include<queue>

#include<vector>

#define FILE\_TO\_ENCODE "file\_to\_encode/"

#define FILE\_TO\_DECODE "file\_to\_decode/"

#define HUFF\_TREE\_DAT "huff\_tree\_dat/"

#define ENCODED\_FILE "encoded\_file/"

#define DECODED\_FILE "decoded\_file/"

#define SAVED\_ENCODE "file\_to\_encode.txt"

#define SAVED\_DECODE "file\_to\_decode.txt"

struct huff

{

int data;

int left;

int right;

int par;

int chr;

huff(int a=255){chr=a;}

};

int N;

huff \*init\_huff(huff \*huff\_man,map<char,int> cnt)

{

int i=1;

huff\_man = new huff[N];

for(map<char,int>::reverse\_iterator rit=cnt.rbegin();rit!=cnt.rend();rit++)

{

// cout<<(\*rit).first<<","<<(\*rit).second<<endl;

huff\_man[i].data=(\*rit).second;

huff\_man[i].chr=(\*rit).first;

huff\_man[i].par=0;

huff\_man[i].left=0;

huff\_man[i].right=0;

cout<<i<<" "<<huff\_man[i].data<<endl;

i++;

}

return huff\_man;

}

char\* get\_file\_name(char \*pre,char \*&suffix)

{

char new\_file[30];

char \*a;

a=new\_file;

strcpy(a,pre);

strcat(a,suffix);

strcpy(suffix,a);

cout<<suffix<<endl;

return suffix;

}

int seek\_min(huff \*huff\_man,int place)

{

int i;

int j;

int min=0;

int sub=0;

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

{

if(huff\_man[i].par==0)

{

min=i;

break;

}

}

for(i;i<=(N/2);i++)

{

// cout<<"search"<<endl;

if((huff\_man[i].par==0)&&(huff\_man[i].data<huff\_man[min].data))

{

min=i;

// cout<<i<<endl;

}

}

// huff\_man[min].par=place;

// cout<<min<<endl;

return min;

}

map<char,int> read\_origin(char \*f\_origin)

{

map<char,int> cnt;

char chr;

fstream f1;

int i;

int n=0;

f1.open(f\_origin,ios::in);

cout<<f\_origin<<endl;

system("pause");

while(!f1.eof())

{

chr=f1.get();

if(!cnt.count(chr))

{

n++;

cnt[chr]=0;

}

cnt[chr]++;

cout<<chr<<endl;

}

for(map<char,int>::reverse\_iterator rit=cnt.rbegin();rit!=cnt.rend();rit++)

{

cout<<(\*rit).first<<","<<(\*rit).second<<endl;

}

f1.close();

cout<<endl<<n<<endl;

N=n\*2;

cout<<N;

return cnt;

}

void display(huff \*huff\_man)

{

int i;

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

{

cout<<i<<" "<<huff\_man[i].chr<<" "<<huff\_man[i].data<<" "<<huff\_man[i].par<<" "<<huff\_man[i].left<<" "<<huff\_man[i].right<<endl;

}

}

huff\* merge\_tree(huff \*huff\_man,int t1 ,int t2,int place)

{

huff\_man[t1].par=place;

huff\_man[t2].par=place;

huff\_man[place].par=0;

huff\_man[place].data=huff\_man[t1].data+huff\_man[t2].data;

huff\_man[place].left=t1;

huff\_man[place].right=t2;

return huff\_man;

}

string \* make\_code(huff \*huff\_man)

{

string a;

int count =0 ;

fstream f1;

f1.open("huff\_man.dat",ios::out|ios::binary);

string \*encode;

encode= new string [N/2+1];

// cout<<"llll"<<endl;

// display(huff\_man);

for(int i=1;i<=N/2;i++)

{

int par;

int p;

char choose;

p=i;

par=huff\_man[i].par;

while(par!=0)

{

if(huff\_man[par].left==p)

choose='0';

else

choose='1';

a=choose+a;

p=par;

par=huff\_man[par].par;

}

cout<<endl<<i<<" "<<a<<endl;

cout<<a<<endl;

f1<<a;

encode[i]=a;

a="";

}

f1.close();

return encode;

}

void get\_rate(huff \*huff\_man)

{

fstream f1;

char chr;

f1.open("huff\_origin.txt");

f1>>chr;

while(!f1.eof())

{

if(chr>='A'&&chr<='Z')

huff\_man[chr-'A'+1].data++;

else if(chr>='a'&&chr<='z')

huff\_man[chr-'a'+26+1].data++;

cout<<chr;

f1>>chr;

}

f1.close();

// return huff\_man;

}

void make\_huff(huff \* huff\_man)

{

int t1,t2,place;

for(place=N/2+1;place<N;place++)

{

t1=seek\_min(huff\_man,place);

huff\_man[t1].par=place;

t2=seek\_min(huff\_man,place);

huff\_man[t2].par=place;

huff\_man=merge\_tree(huff\_man,t1,t2,place);

}

}

void make\_tree(huff \*huff\_man)

{

int t1,t2,place;

for(place=N/2+1;place<N;place++)

{

t1=seek\_min(huff\_man,place);

huff\_man[t1].par=place;

t2=seek\_min(huff\_man,place);

huff\_man[t2].par=place;

huff\_man=merge\_tree(huff\_man,t1,t2,place);

}

}

void turn\_txt\_to\_huff(char \*f\_encode,char \*f\_origin)

{

fstream f1,f2,f3;

int chr;

string code;

int i=1;

f1.open("huff\_man.txt",ios::in);

map<int,string> encode;

while(!f1.eof())

{

if(f1.eof())

break;

f1>>chr;

if(f1.eof()) break;

f1>>code;

encode[chr]=code;

}

f1.close();

f2.open(f\_encode,ios::out|ios::binary);

f3.open(f\_origin,ios::in);

char ch;

while(!f3.eof())

{

ch=f3.get();

f2<<encode[(int)ch];

}

f2.close();

f3.close();

}

void write\_tree(map<char,string>en\_code,huff \* huff\_man,char \* f\_huff)

{

fstream f1,f2;

f1.open("huff\_man.dat",ios::out|ios::binary);

f2.open(f\_huff,ios::out|ios::binary);

cout<<"write:"<<endl;

cout<<sizeof(huff\_man)<<" "<<sizeof(huff)\*N<<" "<<endl;

f2.write((char\*)&N,sizeof(int));

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

f2.write((char \*)&huff\_man[i],sizeof(huff));

display(huff\_man);

for(map<char,string>::reverse\_iterator rit=en\_code.rbegin();rit!=en\_code.rend();rit++)

{

f1<<(int)(\*rit).first<<" "<<(\*rit).second<<endl;

cout<<(int)(\*rit).first<<" "<<(\*rit).second<<endl;

}

f1.close();

f2.close();

}

huff\* read\_tree(char \* f\_huff)

{

huff \*tree;

int a;

fstream f1;

f1.open(f\_huff,ios::binary|ios::in);

if(f1.fail())

{

cout<<"can't open file: "<<f\_huff<<endl;

exit(1);

}

f1.read((char \*)&N,sizeof(int));

cout<<N<<endl;

system("pause");

tree=new huff[N];

f1.read((char \*)tree,sizeof(huff)\*N);

f1.close();

cout<<"tree:"<<endl;

system("pause");

display(tree);

return tree;

}

void decode(char \*f\_encode,char \*f\_decode,char \*f\_huff)

{

fstream f1,f2;

f1.open(f\_encode,ios::in|ios::binary);

f2.open(f\_decode,ios::out);

int tmp;

char chr;

huff \*huff\_man;

huff\_man=read\_tree(f\_huff);

cout<<"huff\_tree"<<endl;

display(huff\_man);

while(!f1.eof())

{

tmp=N-1;

cout<<tmp;

while(((tmp==N-1)||tmp>N/2)&&!f1.eof())

{

f1>>chr;

cout<<chr<<endl;

if(chr=='0')

{

tmp=huff\_man[tmp].left;

}else if(chr=='1')

{

tmp=huff\_man[tmp].right;

}

}

if(!f1.eof())

{

if(huff\_man[tmp].chr!=-1)

{

f2<<(char)huff\_man[tmp].chr;

cout<<(char)huff\_man[tmp].chr;

}

}

}

f1.close();

f2.close();

}

int encode\_all\_file()

{

fstream f\_to\_encode;

char f\_o[30],f\_en[30],f\_hu[30];

char \*f\_origin,\*f\_encode,\*f\_huff;

f\_origin=f\_o;

f\_encode=f\_en;

f\_huff=f\_hu;

f\_to\_encode.open(SAVED\_ENCODE,ios::in);

if(f\_to\_encode.fail())

{

cout<<"sorry cant't open file"<<endl;

exit(1);

}

else

{

while(!f\_to\_encode.eof())

{

f\_to\_encode>>f\_origin>>f\_encode>>f\_huff;

cout<<f\_origin<<" "<<f\_encode<<" "<<f\_huff<<endl;

get\_file\_name(ENCODED\_FILE,f\_encode);

get\_file\_name(HUFF\_TREE\_DAT,f\_huff);

get\_file\_name(FILE\_TO\_ENCODE,f\_origin);

cout<<f\_origin<<" "<<f\_encode<<" "<<f\_huff<<endl;

int i=1;

map<char,int> cnt;

map<char,string> en\_code;

string \* encode;

huff \* huff\_man;

cnt = read\_origin(f\_origin);

huff\_man = init\_huff(huff\_man,cnt);

make\_tree(huff\_man);

display(huff\_man);

encode = make\_code(huff\_man);

for(map<char,int>::reverse\_iterator rit=cnt.rbegin();rit!=cnt.rend();rit++)

{

cout<<(\*rit).first<<","<<(\*rit).second<<endl;

en\_code[(\*rit).first]=encode[i++];

}

write\_tree(en\_code,huff\_man,f\_huff);

turn\_txt\_to\_huff(f\_encode,f\_origin);

}

f\_to\_encode.close();

}

return 0;

}

int decode\_all\_file()

{

fstream f\_to\_decode;

char f\_decode[30],f\_encode[30],f\_huff[30];

f\_to\_decode.open(SAVED\_DECODE,ios::in);

if(f\_to\_decode.fail())

{

cout<<"sorry cant't open file"<<endl;

exit(1);

}

else

{

char f\_d[30],f\_en[30],f\_hu[30];

char \*f\_decode,\*f\_encode,\*f\_huff;

f\_decode=f\_d;

f\_encode=f\_en;

f\_huff=f\_hu;

while(!f\_to\_decode.eof())

{

f\_to\_decode>>f\_encode>>f\_decode>>f\_huff;

get\_file\_name(DECODED\_FILE,f\_decode);

get\_file\_name(HUFF\_TREE\_DAT,f\_huff);

get\_file\_name(FILE\_TO\_DECODE,f\_encode);

cout<<f\_encode<<" "<<f\_decode<<" "<<f\_huff<<endl;

system("pause");

decode(f\_encode,f\_decode,f\_huff);

}

f\_to\_decode.close();

}

return 0;

}

int main()

{

/\*

int i=1;

map<char,int> cnt;

map<char,string> en\_code;

string \* encode;

huff \* huff\_man;

cnt = read\_origin();

huff\_man = init\_huff(huff\_man,cnt);

make\_tree(huff\_man);

display(huff\_man);

encode = make\_code(huff\_man);

\*/

/\*

for(map<char,int>::reverse\_iterator rit=cnt.rbegin();rit!=cnt.rend();rit++)

{

cout<<(\*rit).first<<","<<(\*rit).second<<endl;

en\_code[(\*rit).first]=encode[i++];

}

\*/

/\*

write\_tree(en\_code,huff\_man);

turn\_txt\_to\_huff();

\*/

//de\_huff();

int flag=1;

int choice=0;

fstream f\_to\_encode;

fstream f\_to\_decode;

while(flag)

{

cout<<"加密请输入0 \n 解密请输入1"<<endl;

cin>>choice;

if(choice==0)

{

cout<<"请将待加密文件存放于file\_to\_encode文件夹下"<<endl<<endl;

system("pause");

cout<<"请将 待加密文件名 加密后文件名 生成哈弗曼树文件名 写在 file\_to\_encode.txt中"<<endl<<endl;

cout<<"格式为 待加密文件名 加密后文件名 生成哈弗曼树文件名 "<<endl<<endl;

cout<<"如： huff\_origin.txt encoded\_1.txt huff\_1.dat"<<endl<<endl;

cout<<"请再次确认"<<endl<<endl;

system("pause");

encode\_all\_file();

}

else if(choice==1)

{

cout<<"请将待解密文件存放于file\_to\_decode文件夹下"<<endl;

system("pause");

cout<<"请将 待解密文件名 解密后文件名 哈弗曼树文件名 写在 file\_to\_decode.txt中"<<endl<<endl;

cout<<"格式为 待解密文件名 解密后文件名 哈弗曼树文件名 "<<endl<<endl;

cout<<"如： encoded\_1.txt huff\_origin\_1.txt huff\_1.dat"<<endl<<endl;

cout<<"请再次确认"<<endl<<endl;

system("pause");

decode\_all\_file();

}

cout<<"flag: "<<endl;

cin>>flag;

}

return 0;

}

注：

在file\_to\_encode.txt 写上待加密的文件名

格式 待加密文件名 （加密到） 加密后文件名 生成huff树数据存放路径 加密后存放于 encoded\_file

在file\_to\_decode.txt 写上待解密的文件名 解密后存放于 decoded\_file 哈弗曼树数据存放路径

格式 待解密文件名 （加密到） 解密后文件名 huff树数据存放路径

//规定 所有的huff树数据都在huff\_tree\_dat中

测试数据：

Huff\_origin.txt

x x xx xxx xx Most people need to hear those "three little words" I love you. Once in a while, they hear them

just in time.

Most people need to hear those "three little words" I love you. Once in a while, they hear them just in time.

I met Connie the day she was admitted to the hospice ward, where I worked as a volunteer. Her husband, Bill, stood nervously nearby as she was transferred from the gurney to the hospital bed. Although Connie was in the final stages of her fight against cancer, she was alert and cheerful. We got her settled in. I finished marking her name on all the hospital supplies she would be using, then asked if she needed anything.

"Oh, yes," she said, "would you please show me how to use the TV? I enjoy the soaps so much and I don't want to get behind on what's happening." Connie was a romantic. She loved soap operas, romance novels and movies with a good love story. As we became acquainted, she confided how frustrating it was to be married 32 years to a man who often called her "a silly woman."

"Oh, I know Bill loves me," she said, "but he has never been one to say he loves me, or send cards to me." She sighed and looked out the window at the trees in the courtyard. "I'd give anything if he'd say 'I love you,' but it's just not in his nature."

Bill visited Connie every day. In the beginning, he sat next to the bed while she watched the soaps. Later, when she began sleeping more, he paced up and down the hallway outside her room. Soon, when she no longer watched television and had fewer waking moments, I began spending more of my volunteer time with Bill.

He talked about having worked as a carpenter and how he liked to go fishing. He and Connie had no children, but they'd been enjoying retirement by traveling, until Connie got sick. Bill could not express his feelings about the fact that his wife was dying.

One day, over coffee in the cafeteria, I got him on the subject of women and how we need romance in our lives; how we love to get sentimental1 cards and love letters.

"Do you tell Connie you love her?" I asked (knowing his answer), and he looked at me as if I was crazy.

"I don't have to," he said. "She knows I do!"

early

"I'm sure she knows," I said, reaching over and touching his hands rough, carpenter's hands that were gripping the cup as if it were the only thing he had to hang onto "but she needs to hear it, Bill. She needs to hear what she has meant to you all these years. Please think about it."

We walked back to Connie's room. Bill disappeared inside, and I left to visit another patient. Later, I saw Bill sitting by the bed. He was holding Connie's hand as she slept. The date was February 12.

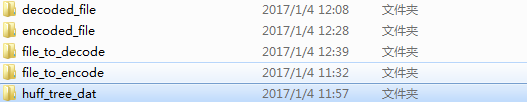
Two days later I walked down the hospice ward at noon. There stood Bill, leaning up against the wall in the hallway, staring at the floor. I already knew from the head nurse that Connie had died at 11 A.M..

When Bill saw me, he allowed himself to come into my arms for a long time. His face was wet with tears and he was trembling. Finally, he leaned back against the wall and took a deep breath.

"I have to say something," he said. "I have to say how good I feel about telling her." He stopped to blow his nose. "I thought a lot about what you said, and this morning I told her how much I loved her... and loved being married to her. You shoulda2 seen her smile!"

I went into the room to say my own goodbye to Connie. There, on the bedside table, was a large Valentine card from Bill. You know, the sentimental kind that says, "To my wonderful wife... I love you."

测试结果：



加密之后的文件在encoded\_file中

解密之后的文件在decoded\_file中

在file\_to\_encode.txt 写上待加密的文件名

格式 待加密文件名 （加密到） 加密后文件名 生成huff树数据存放路径 加密后存放于 encoded\_file

在file\_to\_decode.txt 写上待解密的文件名 解密后存放于 decoded\_file 哈弗曼树数据存放路径

格式 待解密文件名 （加密到） 解密后文件名 huff树数据存放路径

//规定 所有的huff树数据都在huff\_tree\_dat中

代码行数：

472行