# 哈弗曼编码与译码

## 实验要求

1. **实现哈弗曼编码。**
2. 实现哈弗曼译码。

## 实验步骤

1. **读取文本文件source.txt中的字符，用累加法判断频率。**

void checkfreq(map map[46])

{

fstream f;

f.open("d:\\source.txt",ios::in);

char c;

bool mark=0;

while(!f.eof())

{

c=f.get();

if(c>='a'&&c<'z')c-=32;

for(int i=0;i<46;i++)

if(c==map[i].c)

{

mark=1;

map[i].freq++;break;

}

if(!mark)map[31].freq++;

}

f.close();

}

其中map是一个结构数组，存放了可能出现的字符，并用于存放其频率与哈弗曼编码。

1. **按频率建立哈弗曼树。（哈弗曼树用堆实现）**

HuffTree\*htr[46];

for(int i=0;i<46;i++)

htr[i]=new HuffTree(map[i].c,map[i].freq);

HuffTree\*htr1=buildHuff(htr,46);

其中，buildHuff函数：

HuffTree\*buildHuff(HuffTree\*\*TreeArray,int count)

{

heap\*forest=new heap(TreeArray,count,count);

HuffTree\*temp1,\*temp2,\*temp3=NULL;

while(forest->size()>1)

{

temp1=forest->removefirst();//pull first two trees

temp2=forest->removefirst();

temp3=new HuffTree(temp1,temp2);//internal node

forest->insert(temp3);//把结点连接后，把树放到forest中

delete temp1;

delete temp2;

}

return temp3;

}

1. **将哈弗曼编码与哈弗曼树写入文本文件。**

void hufcode(HuffNode\*p)

{

if(p->isLeaf())

{

fstream f;

f.open("D:\\map.txt",ios::app);

Eweight+=strlen(p->hfcode)\*(p->wgt);

f<<p->hfcode<<"\n";

f.close();

return;

}

if(p==root)

{

strcpy\_s(((IntlNode\*)root)->lc->hfcode,"0");

strcpy\_s(((IntlNode\*)root)->rc->hfcode,"1");

}

else

{

strcpy\_s(((IntlNode\*)p)->lc->hfcode,p->hfcode);

strcat\_s(((IntlNode\*)p)->lc->hfcode,"0");

strcpy\_s(((IntlNode\*)p)->rc->hfcode,p->hfcode);

strcat\_s(((IntlNode\*)p)->rc->hfcode,"1");

}

hufcode(((IntlNode\*)p)->lc);

hufcode(((IntlNode\*)p)->rc);

}

void writecode(HuffNode\*p)

{

fstream f;

f.open("D:\\map.txt",ios::out);

f.clear();

f.close();

hufcode(p);

print();

}

void inorderprinthelp(HuffNode\*root,int level)

{

if(root==NULL)return;

if(!root->isLeaf())inorderprinthelp(((IntlNode\*)root)->lc,level+1);

fstream f;

f.open("D:\\map.txt",ios::app);

for(int i=0;i<level;i++)f<<" ";

if(root->isLeaf())

{

f<<((LeafNode\*)root)->ch<<": "<<root->wgt<<"\n";

}

else f<<root->wgt<<"\n";

f.close();

if(!root->isLeaf())inorderprinthelp(((IntlNode\*)root)->rc,level+1);

}

void print()

{

fstream f;

f.open("D:\\map.txt",ios::app);

if(root==NULL)f<<"empty"<<endl;

else inorderprinthelp(root,0);

f.close();

}

1. **将源文件根据哈弗曼编码转为01码。**

void transtobin(map map[46])

{

fstream fread,fmap,fwrite;

fread.open("D:\\source.txt",ios::in);

fmap.open("D:\\map.txt",ios::in);

fwrite.open("D:\\huffcode.txt",ios::out);

for(int i=0;i<46;i++)

{

fmap.getline(map[i].code,47);

}

fmap.close();

char c;

while(!fread.eof())

{

c=fread.get();

if(c>='a'&&c<'z')c-=32;

for(int i=0;i<46;i++)

if(c==map[i].c)

{

fwrite<<map[i].code;break;

}

}

fread.close();

fwrite.close();

}

1. **将01码根据哈弗曼编码转为文本。**

void transtotxt(map map[46])

{

fstream f1,f2;

f1.open("d:\\huffcode.txt",ios::in);

f2.open("d:\\result.txt",ios::out);

char c;

char cntsrc[2];

char cntdst[46];

cntdst[0]='\0';

while(!f1.eof())

{

c=f1.get();

cntsrc[0]=c;

cntsrc[1]='\0';

strcat\_s(cntdst,cntsrc);

for(int i=0;i<46;i++)

if(!strcmp(cntdst,map[i].code))

{

f2<<map[i].c;

cntdst[0]='\0';

break;

}

}

f1.close();

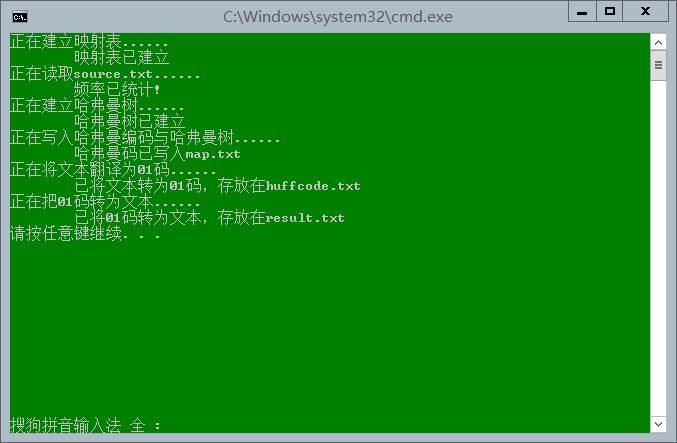
f2.close();

}

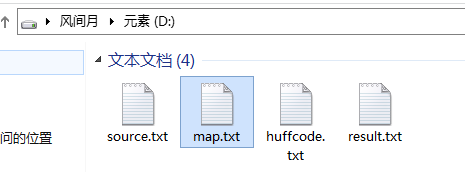
这里实际上不需要利用先前构造的哈弗曼树，因为每一个哈弗曼码唯一对应一个字符，而且不存在前导相同的情况，所以可以按字符读取，逐一比较是否存在对应的哈弗曼编码，如果存在则译码，不存在则继续读取字符。

## 三、实验结果

程序运行截图如下：



文件如下：



文件内容在压缩包中，可打开查看。完整代码见code.cpp