实验四 哈夫曼编码

班级:软件工程一班 姓名:秦源 学号:1525161007

1. 需求分析

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（复原）。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写一个哈夫曼码的编／译码系统。

1. 概要设计

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）。利用以建好的哈夫曼树（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。

三、 详细设计

//

// main.cpp

// 赫夫曼编码

//

// Created by 秦源.

// Copyright © 2016年 QinYuan. All rights reserved.

//

#include <iostream>

using namespace std;

//--赫夫曼树和赫夫曼编码表的存储表示--

typedef struct{

unsigned int weight;

unsigned int parent,lchild,rchild;

}HTNode,\* HuffmanTree;//动态分配数组存储赫夫曼树

typedef char \*\* HuffmanCode;//动态分配数组存储赫夫曼编码表

//--Select 算法--

int \* Select(HuffmanTree HT,int t){

int \*s=new int[5];

s[0]=s[1]=1000;//储存序号

s[2]=s[3]=1000;//储存权值用于比较

for(int i=1;i<=t;i++){

if(HT[i].parent!=0)

continue;

if(HT[i].weight<s[2]){

s[3]=s[2];

s[2]=HT[i].weight;

s[1]=s[0];

s[0]=i;

continue;

}

if(HT[i].weight<s[3]){

s[3]=HT[i].weight;

s[1]=i;

}

}

if(s[0]>s[1]){

int temp=s[0];

s[0]=s[1];

s[1]=temp;

}

return s;

}

//--HuffmanCoding--

void HuffmanCoding(HuffmanTree & HT,HuffmanCode & HC,int \* w,int n){

//w 存放n个字符的权值(均>0)，构造赫夫曼树HT，并求出n个字符的赫夫曼编码表HC。

if(n<=1)

return;

int m=2\*n-1;//表的列数

HT=(HuffmanTree)malloc((m+1)\*sizeof(HTNode));//0号单元未用

HTNode \*p;

int i;

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

\*p={(unsigned)\*w,0,0,0};

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

\*p={0,0,0,0};

for(i=n+1;i<=m;++i){//建赫夫曼树

//在HT[1..i-1]选择parent为0且weight最小的两个节点，其序号分别为s1和s2。

int \*s=Select(HT,i-1);

int s1=s[0];

int s2=s[1];

HT[s1].parent=i;

HT[s2].parent=i;

HT[i].lchild=s1;

HT[i].rchild=s2;

HT[i].weight=HT[s1].weight+HT[s2].weight;

}

/\*

cout<<"打印HT的终态 ： "<<endl;

for(int k=1;k<=m;k++){

cout<<HT[k].weight<<"\t"<<HT[k].parent<<"\t"<<HT[k].lchild<<"\t"<<HT[k].rchild<<endl;

}

\*/

//--从叶子的根逆向求每个字符的赫夫曼编码--

HC=(HuffmanCode)malloc((n+1)\*sizeof(char \*));//分配n个字符编码的头指针向量

char \* cd=(char\*)malloc(n\*sizeof(char));//分配求编码的工作空间

cd[n-1]='\0';//编码结束符

for(i=1;i<=n;++i){//逐个字符求赫夫曼编码

int start=n-1;

int c,f;

for(c=i,f=HT[i].parent;f!=0;c=f,f=HT[f].parent){//从叶子到根逆向求编码

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);

}//HuffmanCoding

void print\_HC(HuffmanCode & HC,int n){

int i=0;

while(HC[n][i]!='\0'){

cout<<HC[n][i];

i++;

}

cout<<endl;

}

HuffmanTree HT;

HuffmanCode HC;

void Initialization(){

cout<<"请输入字符种类"<<endl;

int n;

cin>>n;

int \* w=new int[n];

for(int i=0;i<n;i++){

cout<<"请输入第"<<i+1<<"个字符的权值:";

cin>>w[i];

}

HuffmanCoding(HT,HC, w, n);

}

void Print(){

cout <<"HC 表:"<<endl;

for(int i=1;i<=8;i++){

cout<<"第"<<i<<"个字符对应的编码为： ";

print\_HC(HC, i);

}

}

int main(int argc, const char \* argv[]) {

Initialization();

Print();

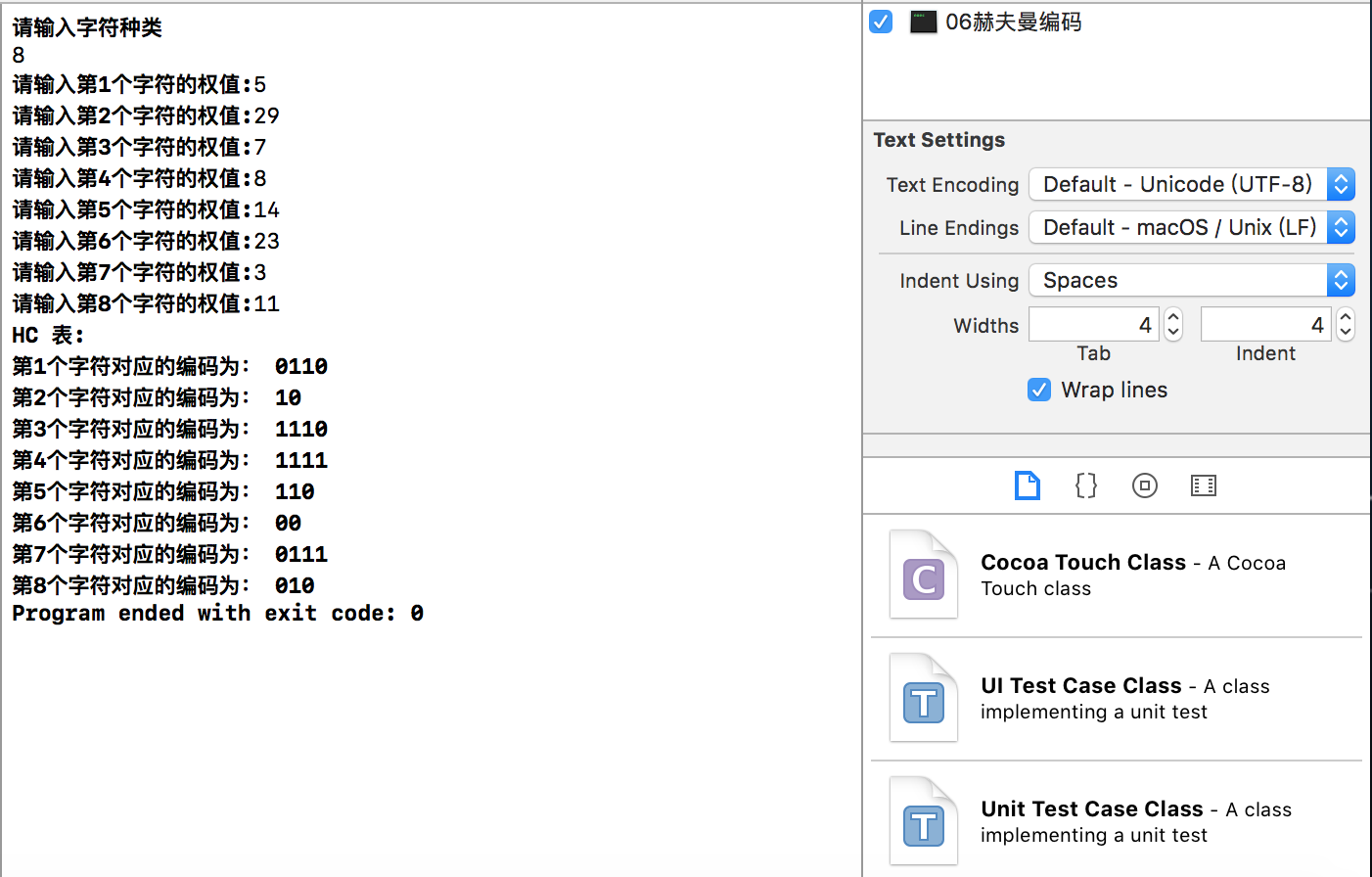
return 0;

}

四、用户使用说明

测试数据：见数据结构教科书Ｐ149测试数据。

五、 测试结果



六、 附录

仅实现了哈夫曼编码系统中数据结构部分：哈夫曼编码。