**郑 州 轻 工 业 大 学**

**实 验 报 告**

**课程名称：** 赫夫曼编码及其应用

**实验名称：** 常用的内部排序算法

**院 （系）：** 计算机与通信工程学院

**姓 名：** 刘亚炜

**学 号：** 541707090128

**专业班级：** 物联网工程17-1

**指导教师：** 孟颍辉

2019 **年** 4 **月** 25 **日**

**实验报告成绩评定表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 评定项目 | 内 容 | 满 分 | 评 分 | 总 分 |
| 实验态度 | 实验认真，态度端正，遵守纪律，出勤情况。 | 20 |  |  |
| 实验过程 | 按实验要求完成各种功能或操作，代码书写规范，注释清晰，设计严谨，运行结果正确。 | 40 |  |
| 报告撰写 | 报告字迹整洁、内容丰富、条理清楚；图、表、文字表达准确规范，上交及时。 | 40 |  |
| 总成绩 | 采用五级分制：优、良、中、及格、不及格 |  | | |
| **评语：**  **指导老师签字： 年 月 日** | | | | |

实验报告正文

1. **实验目的**

掌握赫夫曼（Huffman）树和赫夫曼编码的基本思想和应用。

1. **实验内容及要求**
2. 任务描述

实现文件中数据的加解密与压缩。

1. 主要数据类型与变量

typedef struct huffNode

{

int weight;

int Lchild,Rchild,parent;

}HTNode,\*HuffmanTree;

typedef char \*\*HuffmanCode;

1. 算法或程序模块

void CountStringChar(string s);

void Printttime();

void TempToArray();

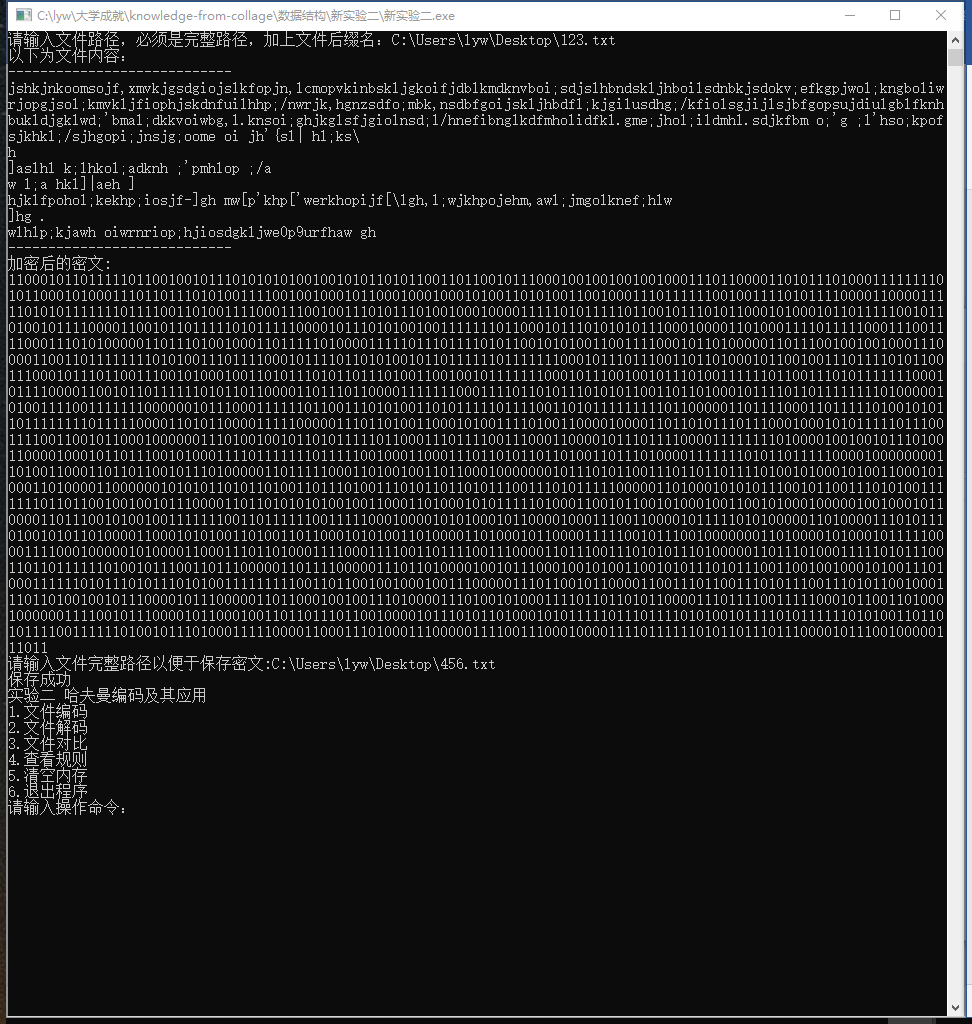
void PrintCode();

1. **测试**
   1. 方案

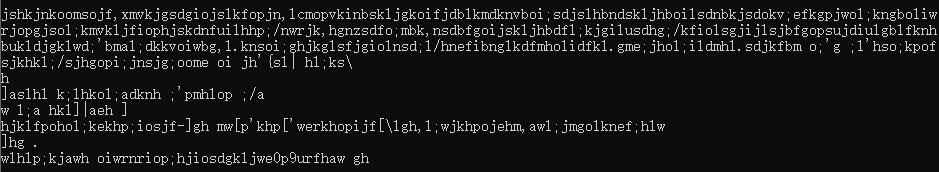
创建一个txt文件对其进行编码，输出另一个文件

* 1. 结果

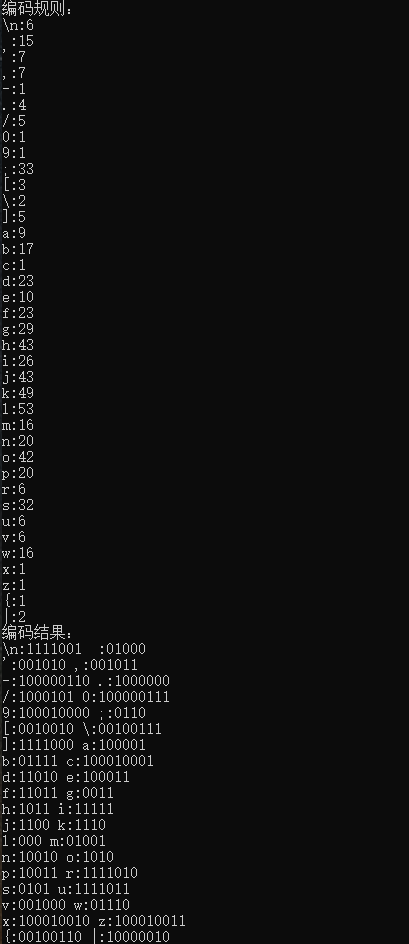
对其进行加密操作：



对其进行解码操作：



查看编码规则：



1. **总结与讨论**

本次实验用的哈夫曼树构建是书上给的标程，在进行文件操作时使用c++文件流，以字符串形式承载内容，在进行每次比对时，自动保存了规则，以便之后可以进行查看，但是由于每个文件加密的方法都不同，加密方法又保存到了内存中，因此就会出现退出了就找不到这个文件是如何进行加密的，暂时未找到解决此问题的方法。

**附：程序的源代码**

**Main.cpp**

**#include <bits/stdc++.h>**

**#include "solve.h"**

**using namespace std;**

**const int max\_n = 1000;**

**string dtext;**

**string ctext;**

**string textdecode;**

**string path,savePath;**

**map<char,int> ttime;**

**map<char,int> :: iterator timeIterator;**

**map<char,string> tcode;**

**map<char,string> :: iterator codeIterator;**

**char key[max\_n];**

**int w[max\_n];**

**HuffmanTree HT;**

**HuffmanCode HC;**

**int num = 0;**

**void CountStringChar(string s);**

**void Printttime();**

**void TempToArray();**

**void PrintCode();**

**string getCode(string data);**

**string getDecode(string data);**

**void init();**

**void contrastFile(string code,string decode);**

**int main()**

**{**

**bool flag = true;**

**int choose = -1;**

**while(1){**

**welcome();**

**cout<<"请输入操作命令：";**

**cin>>choose;**

**switch(choose){**

**case 1:**

**init();**

**cout<<"请输入文件路径，必须是完整路径，加上文件后缀名：";**

**cin>>path;**

**dtext = readFile(path);**

**cout<<"以下为文件内容："<<"\n"**

**<<"----------------------------\n"**

**<<dtext<<"\n"**

**<<"----------------------------\n";**

**CountStringChar(dtext);**

**TempToArray();**

**HuffmanToCode(HT,HC,w,num);**

**ctext = getCode(dtext);**

**cout<<"加密后的密文:"<<endl;**

**cout<<ctext<<endl;**

**cout<<"请输入文件完整路径以便于保存密文:";**

**cin>>savePath;**

**saveFile(savePath,ctext);**

**cout<<"保存成功"<<endl;**

**break;**

**case 2:**

**cout<<"请输入文件路径，必须是完整路径，加上文件后缀名：";**

**cin>>path;**

**dtext = readFile(path);**

**textdecode = getDecode(dtext);**

**cout<<"明文:"<<endl;**

**cout<<"-----------------------------"<<endl;**

**cout<<textdecode<<endl;**

**cout<<"-----------------------------"<<endl;**

**break;**

**case 3:**

**cout<<"文件对比会清空已有规则"<<endl;**

**contrastFile(path,savePath);**

**break;**

**case 4:**

**cout<<"编码规则："<<endl;**

**Printttime();**

**cout<<"编码结果："<<endl;**

**PrintCode();**

**break;**

**case 5:**

**init();**

**cout<<"清空成功！"<<endl;**

**break;**

**case 6:**

**flag = false;**

**cout<<"退出..."<<endl;**

**break;**

**default:**

**cout<<"命令异常"<<endl;**

**break;**

**}**

**if(!flag) break;**

**}**

**return 0;**

**}**

**void contrastFile(string code,string decode){**

**init();**

**string data = readFile(code);**

**CountStringChar(data);**

**TempToArray();**

**HuffmanToCode(HT,HC,w,num);**

**string temp1 = getCode(data);**

**string temp2 = readFile(decode);**

**if(temp1 == temp2){**

**cout<<"密文文件与明文文件一致"<<endl;**

**}else{**

**cout<<"密文文件与明文文件不一致"<<endl;**

**}**

**}**

**void init(){**

**dtext="";**

**ctext="";**

**textdecode="";**

**path="";**

**savePath="";**

**num = 0;**

**ttime.clear();**

**tcode.clear();**

**}**

**string getDecode(string data)**

**{**

**string ans = "",temp;**

**int L=0,R=1,length = data.size();**

**while(L<length&&R<=length){**

**temp = data.substr(L,R-L);**

**char c;**

**bool flag = false;**

**for( codeIterator = tcode.begin(); codeIterator!=tcode.end(); codeIterator++){**

**if(codeIterator->second == temp){**

**c = codeIterator->first;**

**flag = true;**

**break;**

**}**

**}**

**if(flag){**

**L = R;**

**R = L+1;**

**ans += c;**

**}**

**else{**

**R++;**

**}**

**}**

**return ans;**

**}**

**string getCode(string data){**

**string s="",temp;**

**int length = data.size();**

**tcode.clear();**

**for (int i = 1; i <=num; i++){**

**temp = "";**

**for(int j=0;HC[i][j]!='\0';j++){**

**temp += HC[i][j];**

**}**

**tcode[key[i]] = temp;**

**}**

**for(int i = 0;i<length;i++){**

**s += tcode[data[i]];**

**}**

**return s;**

**}**

**void PrintCode(){**

**for (int i = 1; i <=num; i++){**

**if(key[i] == '\n'){**

**cout<<"\\n"<<":"<<HC[i]<<" ";**

**}else**

**cout<<key[i]<<":"<<HC[i]<<" ";**

**if(++i<=num){**

**if(key[i] == '\n'){**

**cout<<"\\n"<<":"<<HC[i]<<endl;**

**}else**

**cout<<key[i]<<":"<<HC[i]<<endl;**

**}else{**

**cout<<endl;**

**}**

**}**

**}**

**void TempToArray(){**

**int i = 1;**

**num = ttime.size();**

**for(timeIterator = ttime.begin();timeIterator!=ttime.end();timeIterator++){**

**key[i] = timeIterator->first;**

**w[i] = timeIterator->second;**

**i++;**

**}**

**}**

**void Printttime(){**

**for(timeIterator = ttime.begin();timeIterator!=ttime.end();timeIterator++){**

**if(timeIterator->first == '\n'){**

**cout<<"\\n"<<":"<<timeIterator->second<<endl;**

**}**

**else cout<<timeIterator->first<<":"<<timeIterator->second<<endl;**

**}**

**}**

**void CountStringChar(string s){**

**ttime.clear();**

**if(s.size() == 0) return ;**

**int length = s.size();**

**for(int i=0; i<length; i++){**

**ttime[s[i]]++;**

**}**

**}**

**string readFile(string path)**

**{**

**const char \* filepath = path.data();**

**ifstream ifile(filepath);**

**ostringstream buf;**

**char ch;**

**while(buf&&ifile.get(ch))**

**buf.put(ch);**

**ifile.close();**

**return buf.str();**

**}**

**void saveFile(string savePath,string data){**

**ofstream outFile;**

**const char \* path = savePath.data();**

**outFile.open(path);**

**outFile << data;**

**outFile.close();**

**}**

**void select(const HuffmanTree &HT,int n,int &s1,int &s2){**

**s1 = s2 = 0;**

**int min1 = INF,min2 = INF;**

**for (int i=1; i<=n; ++i){**

**if ( HT[i].parent == 0 ){**

**if ( HT[i].weight < min1 ){**

**min2 = min1;**

**s2 = s1;**

**min1 = HT[i].weight;**

**s1 = i;**

**}**

**else if ( (HT[i].weight >= min1) && (HT[i].weight < min2) ){**

**min2 = HT[i].weight;**

**s2 = i;**

**}**

**}**

**}**

**}**

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

**{**

**int s1,s2;**

**int m = 2\*n-1;**

**int i,c,f,j;**

**HT = (HuffmanTree)malloc((m+1)\*sizeof(HTNode));**

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

**HT[i] = {w[i],0,0,0};**

**for (int i=n+1; i<=m; i++)**

**HT[i] = {0,0,0,0};**

**for (int i=n+1; i<=m; i++){**

**select(HT,i-1,s1,s2);**

**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;**

**}**

**HC = (HuffmanCode)malloc((n)\*sizeof(char \*));**

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

**code[n-1] = '\0';**

**for (int i = 1; i <= n; i++ ){**

**int start = n-1;**

**for (int c = i,f = HT[c].parent; f!=0; c=HT[c].parent,f=HT[c].parent){**

**if ( HT[f].Lchild == c )**

**code[--start] = '0';**

**else if(HT[f].Rchild == c)**

**code[--start] = '1';**

**}**

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

**strcpy(HC[i],&code[start]);**

**}**

**}**

**void welcome(){**

**cout<<"实验二 哈夫曼编码及其应用"<<endl;**

**cout<<"1.编码\n2.解码"<<endl;**

**cout<<"3.对比\n4.查看规则"<<endl;**

**cout<<"5.清空内存\n6.退出程序"<<endl;**

**}**

**Solve.h**

**#include <bits/stdc++.h>**

**using namespace std;**

**#ifndef SOLVE\_H\_INCLUDED**

**#define SOLVE\_H\_INCLUDED**

**#endif**

**typedef struct huffNode**

**{**

**int weight;**

**int Lchild,Rchild,parent;**

**}HTNode,\*HuffmanTree;**

**typedef char \*\*HuffmanCode;**

**void select(const HuffmanTree &HT,int n,int &s1,int &s2);**

**void HuffmanToCode(HuffmanTree &HT,HuffmanCode &HC,int \*w,int n);**

**string readFile(string path);**

**void saveFile(string savepath,string data);**

**void welcome();**