

**课 程 设 计 报 告 书**

**实验课程名称 《数据结构课程设计》**

**开 课 学 院 网络空间安全学院**

**指导老师姓名**

**学 生 学 号**

**学 生 姓 名**

**2019年06月13日**

选题名称：基于哈夫曼树的数据压缩算法

一、需求分析

现如今互联网无时无刻不在传输海量的数据，但这其中很多数据是冗余的，有非常多重复的字节（字符），如果设计一个转化表，将二进制数01代表一个1字节的字符，那么传输效率将大大提升。将二进制数替代本身字符有如下几点要求：

1. 因为需要解压，所以编码不能有异议性，也就是说任意字符对应的二进制数不能“包含其它字符对应的二进制数”
2. 需要尽可能地缩短编码的平均长度，可以让出现频率高的字符对应较短的字节，出现频率较低的字符对应较长的字节

二、总体设计

根据以上需求，可以采用哈夫曼编码（Huffman Coding），哈夫曼编码是一种可变长编码，该方法完全依据字符出现概率来构造异字头的平均长度最短的码字。

考虑需要压缩的内容一般都非常长，不利于在命令行中输入及粘贴，所以考虑从文本文件（txt）中读取内容，压缩后输出在命令行中

三、详细设计

数据结构：

1. 内容结构：

typedef struct Contents

{

int size; // 读入字串的大小

char\* pt; // 读入的内容

int\* frequency; // 存储每个字符出现的次数

int\*\* huffmanCode; // 存储每个字符的哈夫曼编码

}Contents;

1. 哈夫曼树结构：

typedef struct HuffmanTreeNode

{

int weight; // 结点的权重

int value; // 结点的值

struct HuffmanTreeNode\* LChild; // 结点的左孩子

struct HuffmanTreeNode\* RChild; // 结点的右孩子

}Node,\*PNode;

函数：

主模块：

* int main(int argc, char \*argv[]); error()
* void error();

加密模块：

* void encode(char \*argv[]);
* Contents\* readFile(FILE \*fp);
* void clear(Contents\* content, PNode top);

文本分析模块：

* void frequency(Contents\* content);
* PNode\* intArray2PNodeArray(Contents\* content);
* void sort(PNode\* a,int i);

哈夫曼树模块：

* PNode createHuffmanTree(PNode \* a);
* void InorderTraversal(PNode top);
* void PreorderTraversal(PNode top);
* void PostorderTraversal(PNode top);
* void printHuffmanTree(PNode top);
* void createHuffmanCoding(PNode top, Contents\* content ,int\* code , int n);

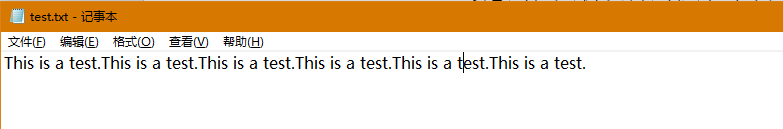
另：源文件中有所有函数的详细解释（函数名、功能、调用的函数、被调用、输入、输出、返回值···），例如：

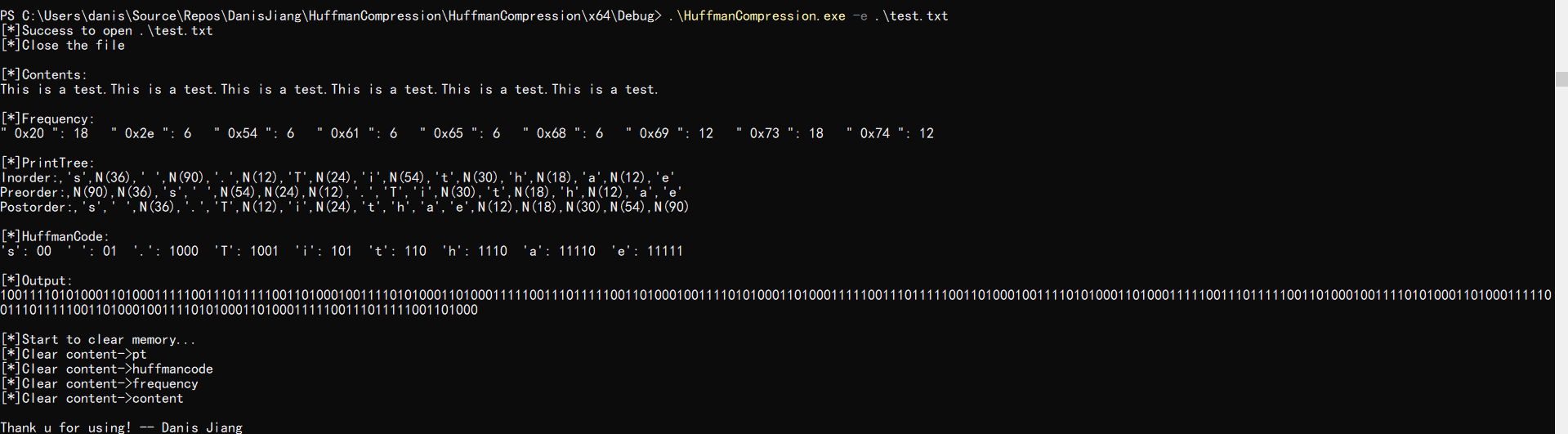


四、系统测试与分析

1. 为了方便他人操作，我对程序做了两部优化：
   1. C:\Users\danis\AppData\Local\Temp\ScreenClip.png使用命令行参数来选择功能及选择文件
   2. 考虑到他人初次得到程序时不知道如何使用，我在设计程序时对不合法输入做了处理，使程序调用error()函数来输出相关教程，非常简洁直观
2. 在考虑输出格式时考虑了windows系统下换行符由“\t”“\n”两个字符组成，如果输入每个字符的表格的话会遇到输出格式的问题，所以一开始只输出ASCII码，但随后考虑到大量的ASCII码不利于阅读，权衡之下还是采用了字符输出。
3. 考虑到一般使用时需要直接得到哈夫曼编码，如果再保存到文件中会极大影响效率，因此将最后得到的哈夫曼编码直接显示在命令行中
4. 为应对各种错误情况，我对可能出现的错误（例如：不合法的输入、不存在的文件、文件被占用、内存分配失败···等情况都做了相对应的error处理）C:\Users\danis\AppData\Local\Temp\ScreenClip.png

最后输出结果：



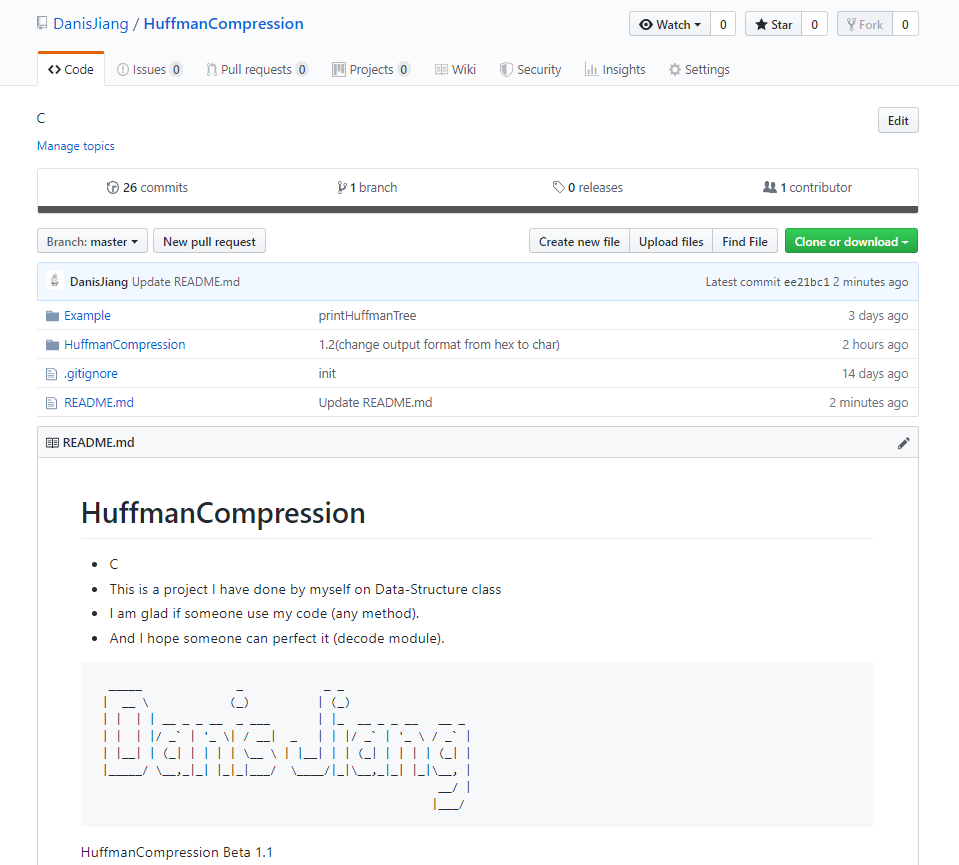


（输出包括文本内容、字符出现的个数、生成的哈弗曼树、每个字符对应的哈夫曼编码、最后压缩生成的文本，非常详尽）

五、结论与心得

此项目的所有源代码都已开源，项目地址：

<https://github.com/DanisJiang/HuffmanCompression>



这次的数据结构课程设计让我更深入地理解了树的相关知识，更熟练地使用C语言编写程序。也丰富了我在github上的项目。

同时，这次的项目从结构上更加接近正式的项目：模块化的文件、更加规范详细的注释、更加成熟的程序架构。

在这里，希望我的计算机技术更上一层楼。