实验报告

题目：5.2哈夫曼编/译码器

班级：信卓1901班 姓名：刘可立 学号：U201913498

一、需求分析

编写具有以下功能的哈夫曼编/译码系统（接收用户的以下命令）：

1. I：初始化，读入字符集大小n，n个字符以及n个权值。建立哈夫曼树并存储。
2. E：编码，将用户输入的文件进行哈夫曼编码，并输出到EncodedFile.txt。
3. D：解码，将用户输入的文件进行解码，并输出到DecodedFile.txt。
4. P：印代码文件，将EncodedFile.txt以紧凑格式写入到终端，并同时存储到CodePrin.txt。
5. T：印哈夫曼树，将已在内存中的Huffman树以直观的凹入表显示在终端，并同时储存到TreePrint.txt。
6. 测试数据：
7. 教材例6.2的数据
8. 按教材表格给出的频度编解码：“this program is my favorite”
9. 英文文本 hamlet.txt

二、概要设计

1. 所使用的线性表的抽象数据结构定义为：

ADT HuffmanTree {

数据对象 D ：D是具有相同特性数据元素的集合

数据关系：R

基本操作：

Init(&t)

读入各结点信息

构造一个哈夫曼树并对其初始化

DestroyList(&t)

哈夫曼树t已存在

销毁哈夫曼树t，释放分配的空间

Empty(t)

若t为空树，则返回TRUE，否则返回FALSE

LeftChild(t, e)

e是t中某节点

返回e的左孩子，若e无左孩子，则返回空

RightChild(t, e)

e是t中某节点

返回e的右孩子，若e无右孩子，则返回空

Showtree(t)

输出huffman树的各个节点信息

Save(t)

将构造好的huffman树存入文件中

Read(t)

从文件中读入huffman树

}

1. 本程序包含三个模块：
2. 主程序模块：

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

初始化数据;

接受用户输入;

while (为接收到退出指令){

处理命令;

执行相应操作;

}

Return 0;

}

1. 实现哈夫曼树的抽象数据类型
2. 哈夫曼树的结点结构

三、详细设计

1. 采用顺序结构存储哈夫曼树，树和其节点的类型定义如下：

**class** HuffmanTreeNode {

**public**:

**int** weight;

**int** parent, left, right;

};

**class** HuffmanTree {

**public**:

**int** n;

HuffmanTreeNode \* nodes;

**char** \* chars;

HuffmanTreeCode codes;

}

1. Huffman树的初始化、读入、保存、编码、译码等操作的伪码如下：

**void** HuffmanTree :: BuildTree(){ //树的初始化

读入节点个数

分配存储空间

读入n个字符及其权值

**for** (**int** i = n + 1; i <= 2 \* n - 1; i++) {

在前i - 1个结点中选择两个权值最小的树作为左右子树构成新的二叉树

新二叉树的权值等于左右子树权值之和

}

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

从叶子到根逆向求每个字符编码

}

}

**void** HuffmanTree :: display(){ //主要在调试过程中使用，输出树的各个节点信息

输出n个字符以及对应编码

输出树的n个结点以及各结点的权值、父结点、左右子节点序号

}

**void** HuffmanTree :: save(){ //将树保存到文件

读入文件路径

对文件输出n个字符以及对应编码

对文件输出树的2n-1个结点以及各结点的权值、父结点、左右子节点序号

}

**void** HuffmanTree :: read(){ //从文件读入树

读入文件路径

从文件读入n个字符以及对应编码

从文件读入树的2n-1个结点以及各结点的权值、父结点、左右子节点序号

}

**void** HuffmanTree :: encode(){ //对文件进行Huffman编码

检查是否已建成树

读入输入文件路径

**if** (空文件 || 打不开){

**return**;

}

**while** (未读到文件尾) {

读入一个字符

查找是否有其对应huffman编码

若无，对文件原样输出

若有，对文件输出其编码

}

fclose(fp);

}

**void** HuffmanTree :: decode(){ //对文件进行Huffman译码

检查是否已建成树

读入输入文件路径

**if** (空文件 || 打不开){

**return**;

}

**while** (未读到文件尾) {

读入一个字符

**if** (该字符不是0或1) {

对文件原样输出

**continue**;

}

从树根开始根据字符的值找左/右子节点

**if** (找到了叶子结点){

对文件输出相应字符

}

}

fclose(fp);

}

**void** HuffmanTree :: showtree(){ //以凹入表形式打印Huffman树

调用先序遍历

输出子节点时多输出一个/t

}

**void** HuffmanTree :: print(){ //印代码文件

读入文件路径

**if** (空文件 || 打不开){

**return**;

}

**while** (未读到文件尾) {

读入一个字符

输出该字符到终端和输出文件

记录本行字符数

**if** (字符数 == 50) {

换行

字符数 = 1

}

}

fclose(fp);

}

1. 函数调用关系

在主函数根据用户指令调用各个Huffman树的初始化、读入、保存、编码、译码等函数。

四、调试分析

1. 设计过程总结反思

考虑到待编码文件的英文字符既有大写又有小写，在编码时先在字符集中查找该字符，若未查找到则将其转为小写重新查找。

为避免每次运行都要重新建立二叉树，支持将其存在文件中，并从文件中读取树。

反思：若忘记写 fclose 函数，那么在其他函数中无法再次打开该文件进行读/写

1. 算法的时空分析
2. 空间复杂度

编码、译码算法在原地工作，使用的储存空间为常数。

1. 时间复杂度
2. 建树过程

理论分析：建树过程有两层循环。外层循环次数为n – 1次，内层循环次数为n~2n-2次，故总循环次数为O(n2)。

考虑到总输入字符数存在上限，因此建树的时间也存在上限，可以认为是一个常数值。

1. 编码、译码过程

While循环的执行次数取决于文件的大小，假设文件的大小为n，则编码、译码过程的时间复杂度应为O(n)。

五、用户手册

1. 本程序的测试运行环境为MacOS，在 Terminal中定位到 a.out 所在的文件夹，输入 ./a.out 回车运行
2. 运行后程序提示输入指令，可输入 h 来获取帮助
3. 显示的用户界面：手机屏幕的截图

   描述已自动生成

(若Terminal窗口太小可能导致提示文字显示不正确)

1. 默认编码后文件：EncodedFile.txt

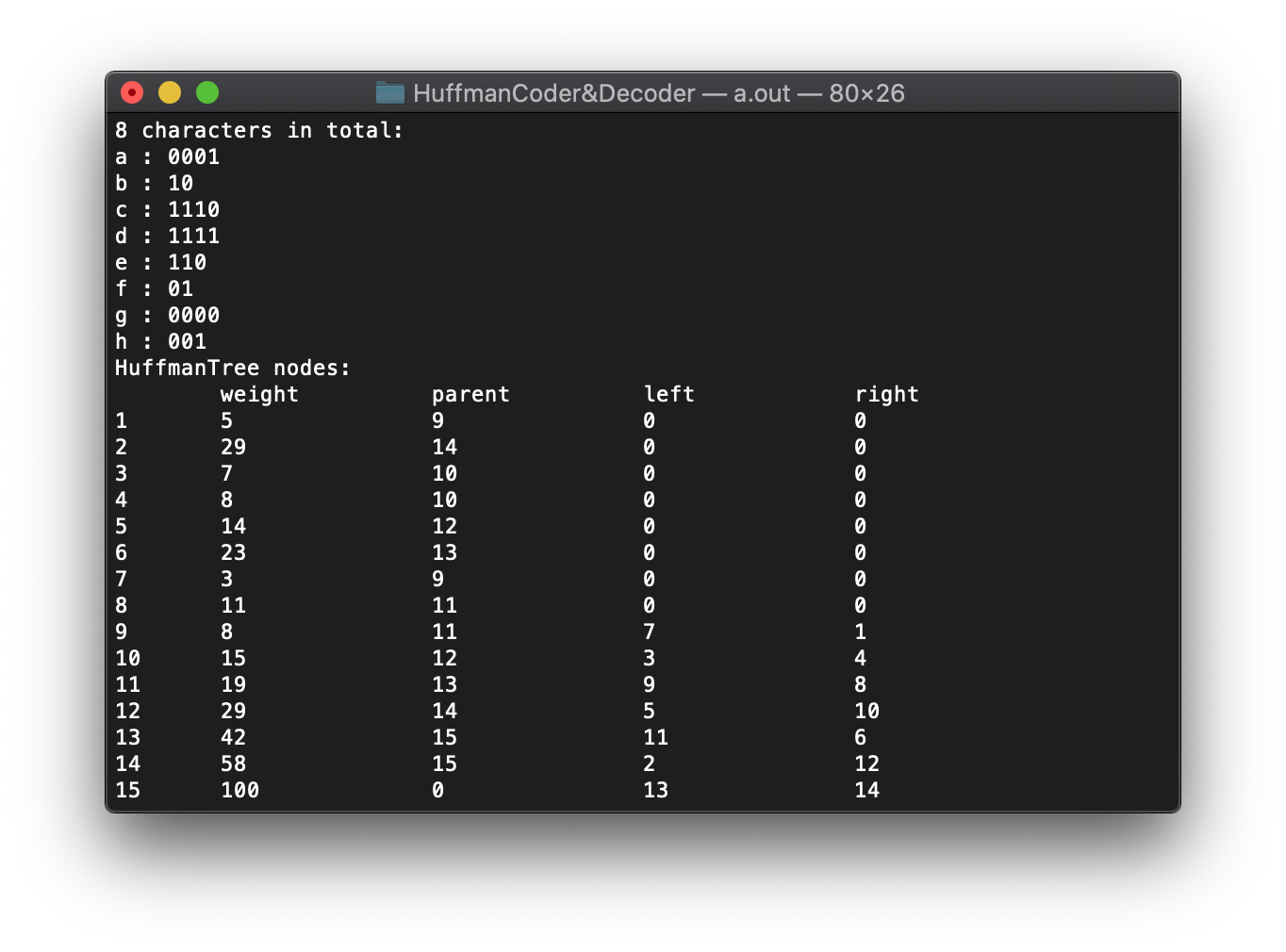
默认解码后文件：DecodedFile.txt

输入的权值应为整数

六、测试结果

1. 输入数据：8种字符（设为a~h）,频率分别为0.05，0.29，0.07，0.08，0.14，0.23，0.03，0.11

建立的哈夫曼树：



1. 按教材表格给出的频度编解码：“this program is my favorite”

哈夫曼树：(已存入alphabet.txt文件中) (第一个字符是空格)

27 characters in total:

: 111

a : 1010

b : 100000

c : 00000

d : 10110

e : 010

f : 110011

g : 100001

h : 0001

i : 0110

j : 1100001000

k : 11000011

l : 10111

m : 110010

n : 0111

o : 1001

p : 100010

q : 1100001001

r : 0010

s : 0011

t : 1101

u : 00001

v : 1100000

w : 110001

x : 1100001010

y : 100011

z : 1100001011

HuffmanTree nodes:

weight parent left right

1 186 50 0 0

2 64 45 0 0

3 13 33 0 0

4 22 37 0 0

5 32 39 0 0

6 103 48 0 0

7 21 36 0 0

8 15 33 0 0

9 47 41 0 0

10 57 43 0 0

11 1 28 0 0

12 5 31 0 0

13 32 39 0 0

14 20 36 0 0

15 57 43 0 0

16 63 44 0 0

17 15 34 0 0

18 1 28 0 0

19 48 42 0 0

20 51 42 0 0

21 80 46 0 0

22 23 37 0 0

23 8 32 0 0

24 18 35 0 0

25 1 29 0 0

26 16 34 0 0

27 1 29 0 0

28 2 30 11 18

29 2 30 25 27

30 4 31 28 29

31 9 32 30 12

32 17 35 23 31

33 28 38 3 8

34 31 38 17 26

35 35 40 32 24

36 41 40 14 7

37 45 41 4 22

38 59 44 33 34

39 64 45 5 13

40 76 46 35 36

41 92 47 37 9

42 99 47 19 20

43 114 48 10 15

44 122 49 38 16

45 128 49 2 39

46 156 50 40 21

47 191 51 41 42

48 217 51 6 43

49 250 52 44 45

50 342 52 46 1

51 408 53 47 48

52 592 53 49 50

53 1000 0 51 52

编码结果：1101000101100011111100010001010011000010010101011001011101100011111110010100011111110011101011000001001001001101101010

1. 哈夫曼树与上一个测试相同。

待编码文件：hamlet.txt

输出文件：EncodedFile.txt, DecodedFile.txt

测试教材要求的 Print , Tree printing 功能：

输出文件：CodePrin.txt, TreePrint.txt

七、附录

清单：

/ HuffmanCoder&Decoder.xcodeproj 项目文件

/ HuffmanCoder&Decoder/main.cpp 主程序

/ HuffmanCoder&Decoder/Header.h 头文件

/ HuffmanCoder&Decoder/a.out 可执行文件

以下都是测试的输出/输入

/ HuffmanCoder&Decoder/alphabet.txt

/ HuffmanCoder&Decoder/hamlet.txt

/ HuffmanCoder&Decoder/EncodedFile.txt

/ HuffmanCoder&Decoder/DecodedFile.txt

/ HuffmanCoder&Decoder/TreePrint.txt

/ HuffmanCoder&Decoder/CodePrin.txt