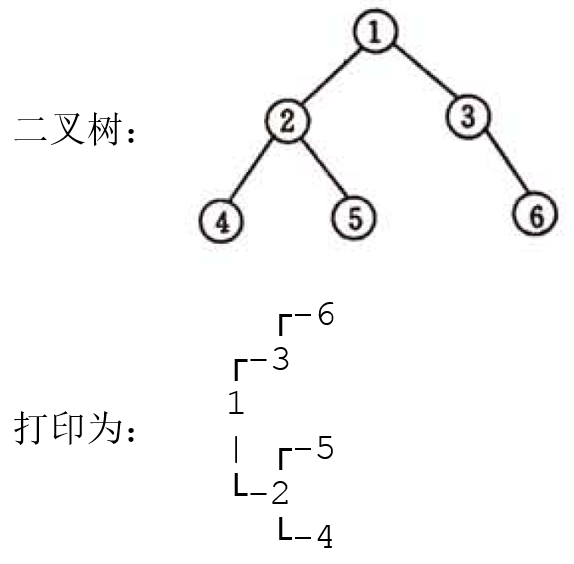
实验3 Huffman编码和解码

问题描述

Huffman编码是二叉树的典型应用之一。可对文件进行编码，压缩文件。程序可实现的功能有：以输出菜单形式可供选择打印哈夫曼树、选择指定文件压缩、解压指定文件。压缩后及解压后的文件均会保存于程序路径。具体功能如下：

1. 对文本文件统计各个字符的出现频率，构造Huffman树。
2. 以图形化的方式打印Huffman树，例如可以逆时针旋转90度打印，如下图所示，或者自己设计打印图形。  
   
3. 以Huffman树对文本文件进行编码，统计编码后的比特数，除以8得到字节数。用原文件的大小（字节数）除以编码后的字节数，即求得压缩比。

实验内容

实验中的主要数据结构为储存哈夫曼数结点的结构体，其定义为：

typedef struct {

TElemType data;

WeightType weight; // 叶子权值的类型

int parent, lchild, rchild; // 三叉静态链表

} HTNode, \*HuffmanTree;

程序中的主要函数有

1.选取哈夫曼数HT中权值最小的两个点的函数，原型为void Select(HuffmanTree HT, int s, int &l, int &r)

输入存储哈夫曼树叶子结点的数组HT，当前数组存储的有效长度s，输出最小权值点的下标l，r。

算法流程：遍历数组，记录当前最小值，与当前数值比较并记录相对小的值

算法复杂度为O(s)

2.读取文件并建立字符出现频率的数组的函数，函数原型为void ReadFile(char fc[], int fi[], int &n,int &sum)

输入字符数组与储存字符出现频率的数组地址，数组有效长度，输出储存字符信息的数组，字符总数

算法流程：遍历读取文件字符，并与当前数组内字符比较，若当前数组不存在此字符，则依次存储

3.创建哈夫曼树的函数，原型为void CreateHuffmanTree(HuffmanTree &HT, char fc[], int fi[], int n)

输入存储哈夫曼数叶子节点的数组，存储文件字符信息的数组，输出完整哈夫曼数的数组

算法流程：遍历选取当前数组内权值最小的两个点，构成一个新的结点，直至数组填满

算法复杂度O(n)

4.创建哈夫曼编码的函数，函数原型为void HuffmanCoding(HuffmanTree HT, char \*\*&HC, int n)void Coding(HuffmanTree HT, char \*\*&HC, int root, SqStack &S)

输入哈夫曼数组，储存哈夫曼码的数组首地址，数组长度，输出存储哈夫曼码的数组

算法流程：通过递归调用Coding函数向左拐计为0，右拐即为1，以字符串形式存储至HC中

算法复杂度O(nlogn)

5.压缩文件的函数，原型为void ChangeFile(HuffmanTree HT, char \*\*HC, int n,char name[],int &csum)

输入哈夫曼数组，哈夫曼码数组，数组长度，压缩文件名，输出压缩文件

算法流程：从源文件读取字符，通过哈夫曼码将其转换为二进制数，通过位操作写入压缩文件中

6.解压文件的函数，函数原型为void Decoprass(HuffmanTree HT, char \*\*HC, int n,char name2[], char name1[],int sum)

输入哈夫曼树组，哈夫曼码数组，数组长度，压缩文件名，解压文件名，字符数

算法流程：从压缩文件读取字节，通过位操作在哈夫曼数中进行遍历，找到字符后，写入解压文件

7.打印哈夫曼树的函数，原型为void print(HuffmanTree HT, char \*\*HC, int cur, int depth,int\* &dep,int count)void PrintHT(HuffmanTree HT,char \*\*HC , int n)

输入哈夫曼数数组，当前所在数组位置，当前深度，左右孩子标志位count，深度标志位数组dep，打印哈夫曼树

算法流程：通过中序遍历哈夫曼树，在dep数组中储存当前是否打印之前分支结点位置的竖线，横向打印哈夫曼树。

实验检验

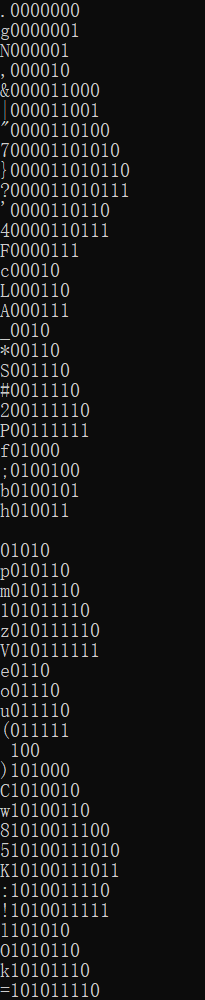
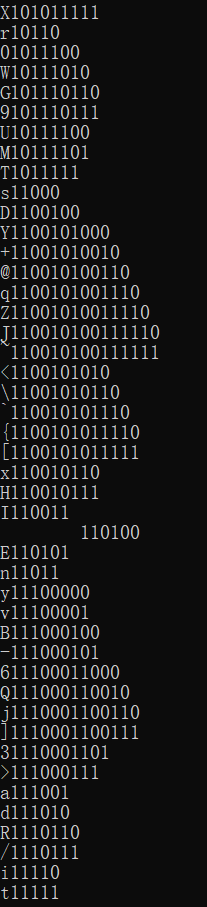
打印哈夫曼树



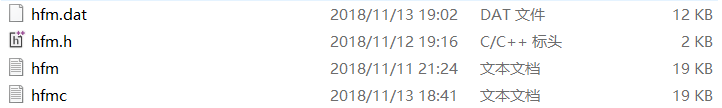




所得哈夫曼码



压缩及解压文件



用FC命令比较后hfm与hfmc完全相同

讨论与结论

构造哈夫曼树时，发现构造的哈夫曼有几个字符与其出现频次不对应，经过调试发现，在寻找最小权值的两个点时，比较中用的时>或<号，并未考虑两个权值相等的情况，改进后解决。

在压缩文件时，用位操作输入时会出现每个哈夫曼码之间会多出来一位0，经过对含有少量字符的文件压缩操作并进行人工解码后，发现，在进行位操作时左移位i的大小多了一，上限高了一，导致多存进去一位0，改进后解决。

在解压操作时，用FC命令时发现，源文件中出现了‘？’字符，而解压文件之中没有，检查后发现，在初始化时我对非叶子结点的储存值都置为了‘？’，改进后，将其都置为了0。

在解压操作时，发现解压后文件最后一位会多一个字符，经过对源文件统计字符，写进压缩文件后，解决。