# 数据结构实验报告

## 哈夫曼树的编/译码器模拟

1. 问题描述

利用哈夫曼编码进行通讯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将待传数据进行预先编码；在接受端将传来的数据进行解码（复原）。对于可以双向传输的信道，每端都要有一个完整的编/译码系统。试为这样的信息收发站写一个模拟哈夫曼的编译码系统。

实验内容及要求：

* 从终端读入字符集大小为n（即字符的个数），逐一输入n个字符和相应的n个权值（即字符出现的频度），建立哈夫曼树，将它存于文件 hfmtree 中。并将建立好的哈夫曼树以树或凹入法形式输出；对每个字符进行编码并且输出。
* 利用已建好的哈夫曼编码文件 hfmtree，对已编码的正文进行译码，输出译码后的正文。
* 采用文本文件存放文本，先统计文本中的每个字符出现的频率，然后再建立哈夫曼树，并进行编码和译码。

1. 问题分析

哈夫曼编码是一种最小权值编码，能够对字符串进行无损压缩。此问题当中需要按照给定权值完成哈夫曼树的建立，并通过其对字符串进行编码、对二进制编码进行译码，以及提供对字符串操作、命令行控制和文件读写的接口。

1. 算法设计

哈夫曼树构建时，将带有权值的节点放入连续内存中；将其调整为小顶堆结构；每次从中取出两个权值最小的节点合并放回，直到堆中剩下唯一的节点，则此节点即为构建好的哈夫曼树；在之后的使用当中，不保留哈夫曼树的权值。同时会生成每个字符对应的编码，对哈夫曼树进行DFS遍历并记录路径，从而生成编码。

开始

带权节点（放入vector）

建立小顶堆

Pop两个节点

合并

放回

DFS生成字符对应编码

结束

剩1节点？

否

是

编码字符串时，预留连续内存区域以备写入，预留 sizeof ( std::size\_t ) 字节的偏置；直接通过字符访问已经得到的编码；利用位运算将编码写入长度为1字节的缓冲区中；缓冲区满时添加到连续内存区域的尾端并复位；循环对所有字符进行编码，末尾缓冲区不满时剩余位补0；在编码过程中同时记录已经使用的二进制位数，以二进制方式写入到连续内存区域的开头（之前偏置预留的区域），在解码时通过二进制位数来控制解码停止。

开始

准备连续内存（预留偏置）

得到单个字符编码

按位写入缓冲区

记录编码位数

结束

缓冲区空？

否

是

到达末尾？

否

是

缓冲复位

得到1B编码

补0写入

下一字符

缓冲区满？

是

编码结束？

否

否

是

解码时，从连续内存区域开头读入二进制位数，控制解码循环；循环读入二进制位，通过指针访问哈夫曼树，0访问左子树，1访问右子树，直到到达叶节点得到字符，指针回到哈夫曼树根节点。

开始

读取二进制编码位数

指针回根节点

读入二进制位

结束

编码结束？

否

是

右子树

下一字符

为真(1)？

是

叶节点？

否

是

左子树

下一位

否

1. 数据结构
   1. hfm文件结构

采用二进制文件；依次为：

l1（std::size\_t类型），哈夫曼树的长度；

l1字节，哈夫曼树（格式为[0b10000000][left][root][right][0b10000001]）；

l2（std::size\_t类型），编码长度；

l2字节，哈夫编码。

* 1. class HFMTree

内部类：

Node、Weighted\_Node（继承自Node）：私有，不带权和带权的二叉树结构，用于建树；

Code：私有，继承自std::vector<bool>（通过私有继承禁用了不需要的方法），用于储存编码；

Counter：公有，继承自std::vector<std::size\_t>（通过私有继承禁用了不需要的方法），字符计数器；

成员变量：

Tree tree：指向哈夫曼树的指针（using Tree = Node\*;）从字符串建树时子节点保留为Weighted\_Node，从hfmtree文件建树时为子节点均为Node；

std::vector<Code> code：各个字符对应的编码。

主要方法：

static Tree build\_tree()：私有，根据字符出现频次建树，用于构造函数，为不同的构造函数经过多次重载；

static std::vector<Code> generate\_code(Tree)：私有，根据哈夫曼树获得每个字符的哈夫曼编码，用于构造函数；

std::vector<uint8\_t> sequence() const：私有，将哈夫曼树转化为线性序列输出；

std::vector<uint8\_t> encode(const std::string &string) const：公有，对字符串进行编码。

std::string decode() const：公有，对哈夫曼编码进行解码，为不同参数进行了重载，通过模板能够在具有随机访问迭代器的容器上广泛适配；

void print\_tree() const：公有，将sequence()的结果输出到std::cout；

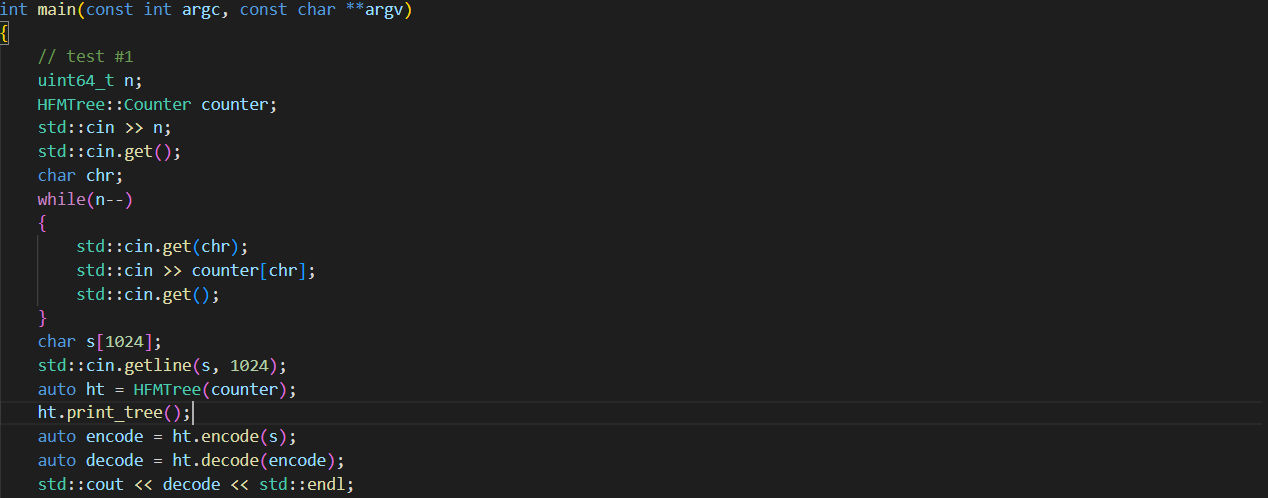
* 1. class HFMString

同时存储哈夫曼树与字符串，提供了与文件等交互的接口。

1. 源码

见文件夹No13\_HuffmanTree；

1. 测试

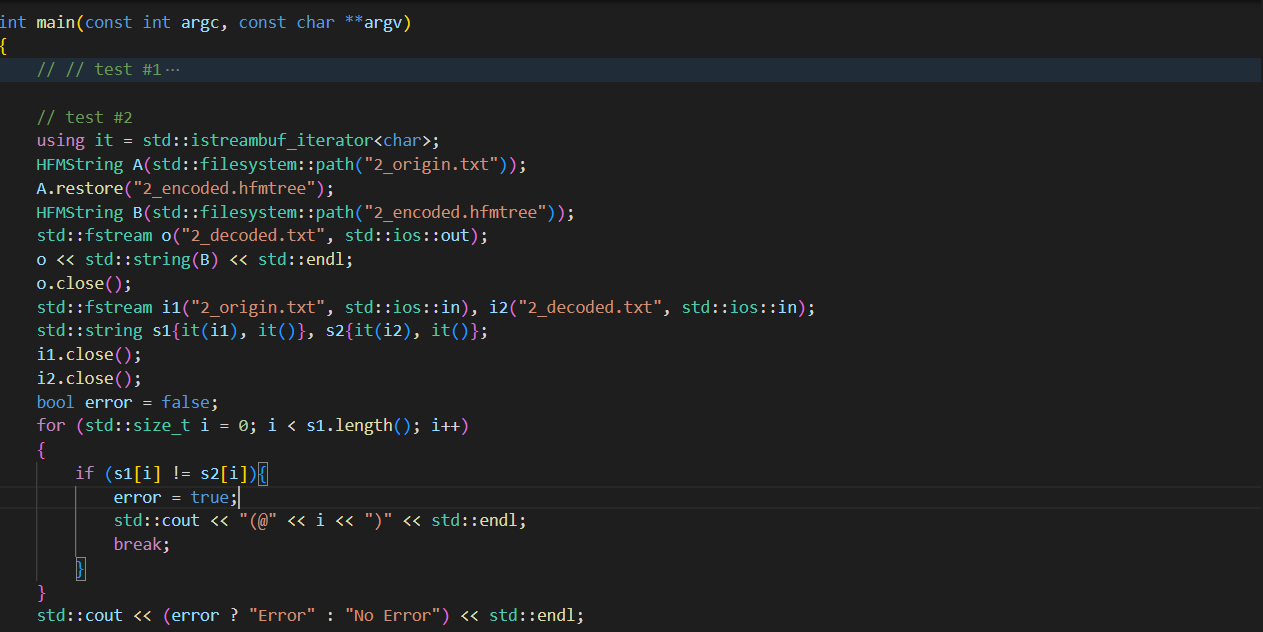
 

切换到文件所在目录HuffmanTree执行以下指令：

huffman.exe<1\_in.txt>1\_out.txt

验得结果正确；文件1\_in.txt，1\_out.txt内容见文件夹HuffmanTree；

2、采用文本文件存放文本，测试实验内容及要求的第3项。

提交时可执行文件按照2、中的主函数编译。

1. 总结

在完成本题的过程中熟悉了哈夫曼编码的原理，熟练了二叉树及相关递归算法、小顶堆的使用，熟悉了c++及其面向对象的特性。