PJ1 开发文档

吕昌泽 18302010026

1.代码部分：

1）Main 文件：通过 FileChooser 和 DirectoryChooser 来选择文件与文件夹。对按钮的点击事件进行处理。

2）GUI 文件：前端部分，包含 label、背景图、button、text 等结点。

3）compress 文件：分为文件读写和文件夹读写，中心思路是先把文件名等文件结构信息写入压缩文件，然后写入哈夫曼编码。对于文件，我使用了 compressFile()方法，它通过 BufferInputStream 来读取文件，进行 Huffman 编码后，通过字 StringBuilder 的 append 方法把每个 byte 的新编码以及转换 byte 数组等操作，用 BufferOutputStream 输出 byte。文件的最后要补足不够一个 byte 的 0 的位数。对文件夹，我用 compressDir（）方法的递归来处理，如果子文件位文件夹，继续调用该方法，如果为文件，则调用 compressFile()方法。

4）decompress 文件：先读文件信息，之后构造文件结构，还是用递归处理文件文件夹，解码的时候需要将每个读入的 byte 处理它的八个位，这个我用了 boolean 数组的方式，并利用 HuffmanTree 来进行搜索。

5）fileStructure 文件：记录所有子文件夹和文件的 parent 的树结构,从而通过 N 个 int 来构造出文件的结构，顺便解决了绝对路径的问题。

6）FILE 文件：做 fileStructure 类的 root 和子文件。取名是为了区别于 java 自带的 File。

7）HuffmanTree 文件：通过输入的每个 byte 出现的次数，借助优先队列的数据结构来构造 huffman 树，每个 byte 出现的次数要写入文件结构信息，从而在解码时能重构树。并且这里计算了编码前后文件的位长度，给后面精准读入 byte 提供了很大的帮助。

8）HuffmanNode 文件：包含权重 rate、字符的 ASCII 码值的 data、Huffman 编码 code、编码长度 codelen、左节点 left、右节点 right

2.解决的问题：

1）刚刚开始压缩超级慢。解决办法：把 FileInput 换成 BufferInputStream、把 FileOutput 换成 BufferOutputStream、把 String 连接换成 StringBuilder 的 append、在重构树的时候用优先队列、int 转 byte 用位操作，不进行数学计算。最重要的是，我在 CSDN 上看到了有人用了并发编程，就是再创建一个 byte 数组 byte[8192]，用 Bufferio 来读入和输出，并且把数据存入该数组，进行并发编程。有一说一，这个东西我是最后几天压大文件的时候发现我压的很慢，才来看这种方法的，我花了两天时间，看懂这个模式，在 PJ 中写入了简单的并发，有一点点难……

2）对压缩文件里应该存什么的疑惑。我刚刚开始压缩文件里存的是：每个字符的 Huffman 编码组成的表、总 Huffman 编码转化来的 byte 码。后来我发现这样子在解码的时候，非常难以识别我已经读到什么位置了。所以我改变了想法，压缩时和解压时都构建 Huffman 树，获得编码。这样子虽然有点浪费时间，但是用了优先队列来建树，感觉也不会浪费太多。

3.大文件压缩率：

Fold1：原大小：2.40 MB (2,521,912 字节)压缩后大小：1.50 MB (1,573,315 字节)
压缩时间：801ms 压缩率：减少了 37.50% 解压时间：258ms

Fold2：原大小：9.34 MB (9,797,013 字节)压缩后大小：6.01 MB (6,309,021 字节)
压缩时间：2487ms 压缩率：减少了 35.65% 解压时间：788ms
Fold3：原大小：2.46 MB (2,587,851 字节)压缩后大小：1.61 MB (1,694,867 字节)
压缩时间：741ms 压缩率：减少了 34.55% 解压时间：436ms
2.csv：原大小：111 MB (116,596,836 字节)压缩后大小：71.3 MB (74,797,029 字节)
压缩时间：4644ms 压缩率：减少了 35.77% 解压时间：5011ms
3.csv：原大小：111 MB (116,523,008 字节)压缩后大小：70.7 MB (74,230,776 字节)
压缩时间：4656ms 压缩率：减少了 36.31% 解压时间：4018ms

4.与传统压缩软件比较：
360 压缩：
Fold1：原大小：2.40 MB (2,521,912 字节)压缩后大小：571 KB (584,887 字节)
压缩率：减少了 76.81%
Fold2：原大小：9.34 MB (9,797,013 字节)压缩后大小：2.13 MB (2,237,922 字节)
压缩率：减少了 77.19%
Fold3：原大小：2.46 MB (2,587,851 字节)压缩后大小：554 KB (568,162 字节)
压缩率：减少了 78.05%
2.csv：原大小：111 MB (116,596,836 字节)压缩后大小：27.9 MB (29,312,945 字节)
压缩率：减少了 74.86% 解压时间：15 秒
3.csv：原大小：111 MB (116,523,008 字节)压缩后大小：12.1 MB (12,790,974 字节)
压缩率：减少了 89.10% 解压时间：15 秒
总结：就压缩率来看，360 压缩碾压了我的 PJ，这个结果在意料之中，让我欣喜的是，在压缩大文件的时候，我的 PJ 压缩时间远远小于 360 压缩！可喜可贺！