# 开发文档

1. **原理及设计思路**

1.原理：

统计文件字节及字节出现次数，利用贪婪算法，根据每个字节权重构建最优哈夫曼树。根据哈夫曼树，出现次数多的字节用较短的01串表示，出现次数少的字节用较长的01串表示，以此实现对文件的压缩。

2.首先实现单个文件的压缩及解压，对于文件夹压缩采用递归方法，最终化为压缩或解压单个文件

3. 需要根据压缩文件头信息来解压文件，因此需要从解压文件/文件夹方向逆向考虑如何构建好文件头信息

，

1. **遇到的问题及解决方法**

**1.（最重要的问题）对Java尤其是JavaFX感到陌生了**

**答：首先重新理解一下大一写的project（虽然有点乱，好在知道怎么动手了），其次就是边学边用**

**2.使用hashMap保存字节及对应Huffman编码，在匹配文件字节对应编码以写入到压缩文件时耗费大量循环查询时间**

**答：第二次尝试->将hashMap转换成按Huffman编码长度排序的TreeMap，但依旧没多大改变**

**第三次尝试->意识到Huffman树是根据字节出现次数（即权重）构建的最优树，而每个字节只有256种情况，因此没必要使用键值对的hashMap或TreeMap。将字节对应的Huffman编码映射到长为256的字符串数组，而字节值即对应数组下标，由此消除了查找map所耗费的大量时间**

**3.解压文件夹时，如何判断当前正在解压的文件是否解压完毕**

**答：首先想到将编码前的文件大小写入文件头，但由于解压文件主体部分代码结构的原因，这种实现方法虽然可行，但不可取。其次就是将编码后文件大小写入文件头，这样挺好，但由此引出另一个问题：怎么获取编码后（已经在写压缩文件主体部分了）文件大小？**

**答：考虑到构建Huffman树时计算了每个字节出现的次数，以及根据Huffman树获得了每种字节对应的Huffman编码，因此可通过计算每个字节出现次数乘以对应Huffman编码长度的总和除以8得到编码后文件字节数（考虑到文件头结构的整齐性，额外加上2，一代表最后不足八位的一个字节，一代表补零的个数）**

**4. 解压文件夹时，当压缩文件被移动到另一文件夹时，发生文件/文件夹写入路径不存在或者写入的路径不是在解压文件夹下的错误**

**答：在压缩文件夹时，文件头的处理与在压缩单个文件时一样，没有写入相对路径**

**5. UI设计部分，在处理大文件时UI界面会显示未响应**

**答：利用多线程，为每一次压缩或解压创建一个新的线程，这样逻辑处理部分就不会总是占用Application线程的资源。但由于代码结构原因，UI界面要求显示处理时间及压缩率，而这些数据由每个线程维护，因此得到新问题：如何从子线程传参到主线程，或者如何避免Application Thread引发的不安全问题（即javafx界面只能由Application Thread修改），然后直接将子线程得到的处理时间及压缩率更新到界面。**

**答：由于各种原因，使用callable等方式从子线程返回参数会使主线程等待子线程结束，因此并不能解决未响应问题，后来了解到可以简单的调用Platform.runlater()方法避免不安全问题**

1. **主要类与方法的作用**

**1. Node**

**Huffman树的节点类**

**接口：**

**Comparable**

**类变量：**

**value（int）存储字节值，**

**count（int）存储字节出现次数，**

**leftChild（Node）节点的左子节点，**

**rightChild（Node）节点的右子节点**

**构造方法：**

**Node（int count)，初始化节点，除去叶子节点，**

**Node（int value，int count），初始化叶子节点**

**类方法：**

**类变量的getter和setter方法**

**isLeaf（boolean）判断叶子节点**

**Override方法：compareTo（Node node）比较字节出现次数**

**2. HuffmanTree**

**Huffman树**

**构造方法：系统默认方法**

**类方法：**

**toQueue（int chars）（PriorityQueue<Node>），根据字节出现次数生成优先队列**

**getHuffmanTree（PriorityQueue<Node> queue)（Node），根据优先队列生成Huffman树**

**3.Compress**

**压缩文件类**

**类变量：**

**srcFile（File），源文件，**

**chars（int []），源文件字节出现次数，**

**charsCode（String []），源文件出现字节对应Huffman编码**

**outputStream（BufferedOutputStream），输出流**

**inputStream（BufferedInputStream），输入流**

**构造方法：**

**Compress（File srcFile，BufferedOutputStream outputStream），初始化源文件、输入流、输出流**

**类方法：**

**compressFile，压缩**

**私有方法：**

**compressFile（Node root），判断文件是否为空，并且做好压缩前工作，**

**writeName（），写入文件名，**

**writeByteCount（），写入文件字节及对应次数，**

**writeBytesSize（），写入文件主体字节数，**

**writeHead（），写入文件头即执行上面三个方法，**

**write File（），写入文件主体部分，**

**4.CompressDir**

**压缩文件夹类**

**类变量：**

**file（File），源文件，**

**size（long），统计文件夹大小，用于计算压缩率，**

**output Stream（BufferedOutputStream），输出流**

**构造方法：  
 CompressDir（File file，BufferedOutputStream outputStream），初始化源文件信息以及输出流**

**类方法：**

**compressDir（），调用compressDir（File srcFile，BufferedOutputStream outputStream）私有方法，压缩文件夹**

**私有方法：**

**compressDir（File srcFile，BufferedOutputStream outputStream），判断文件类型，写入文件夹头信息，递归调用**

**5.Decompress**

**解压文件/文件夹**

**类变量：**

**srcFile（File），源文件信息，**

**input Stream（BufferedInputStream），输入流**

**构造方法：**

**Decompress（File srcFile，BufferedInputStream inputStream），初始化源文件信息以及输入流**

**类方法：**

**decompressFile（），判断是否属于可解压文件，解压文件**

**私有方法：**

**directory（File file），处理文件夹，递归调用**

**single File（File file），处理单个文件，**

**decompress（File file），判断文件类型，选择处理文件夹或单个文件，**

**getHuffmanTree（Node），根据文件头得到文件Huffman树，**

**writeFile（），根据Huffman树以及Huffman编码解码文件**

**6.FileUtil**

**一些普通、常用方法集合**

**构造方法：系统默认方法**

**类方法：**

**countByte（File file）（int []），统计文件字节及字节出现的次数，**

**getCharsCode（Node root)，根据Huffman树得到字节对应的Huffman编码，**

**bit2byte（String string）（byte），将二进制字符串转换成二进制，**

**byte2bits（int value）（String），将数字转换成对应的二进制字符串**

**7.Main**

**UI类**

**类变量：**

**各种组件及组件初始值**

**类方法：**

**Override方法：start（Stage primaryStage）**

**私有方法：**

**init（Stage primaryStage）,初始化界面,**

**getTopPane（HBox），初始化软件信息，**

**getMiddlePane（VBox），初始化文件压缩及解压时界面同步显示信息，**

**getBottomPane（HBox），初始化可点击按钮信息，及为按钮绑定点击事件**

**updateStage（int task，String path，String timeConsuming，String ratio），更新middle Pane的信息**

**16302010059**

**张健**