**开发文档**

***——哈弗曼树压缩***

**一、核心算法：**

利用哈弗曼树用较短二进制位表示出现频率较高的字符，而用较长二进制位表示出现频率较低的字符的特点，压缩本来所有字符等长的文件，达到减小文件大小的效果。

**二、实现：**

**1、一个哈弗曼节点类（HuffmanNode）用以构造文件的哈弗曼树**

* left、right左右儿子节点
* data 节点所代表的字符（未必有）
* weight 节点的权重，即出现次数（一定有）

**2、两个运行类编码与解码（Encode & Decode）用以压缩/解压缩文件**

2.1 Encode类：

* 构造时需要传入文件路径与压缩路径
* oOut ObjectOutputstream用以输出编码后的压缩文件
* fileSize、zipSize压缩文件前后的大小用以计算压缩率
* getCodes 递归方法用以计算哈弗曼树中每个字符的哈弗曼编码
* encode 主要方法，本身也是递归方法，需要传入压缩的文件（夹）及其相对路径

先判断传入的是否为文件，若为文件，则读取它，对其按字符进行遍历，计算每种不同字符的出现次数，将其记录于长度为256的数组中。

按这个数组构造每个不同字符的哈弗曼节点，置于一个列表中，并按权重进行排序。之后每次取最后两个节点（权重最小），相加构造一个新节点，并将他们作为左右儿子。删除两个旧节点，加入一个新节点，并对他们进行再排序，直到整个列表只剩一个节点（根节点）。

根据哈弗曼树，调用getCodes方法递归求出每个字符的哈弗曼编码，存储于byteStringMap中。根据byteStringMap，重新遍历源文件的字符数组，将他们转化为字节数组。按顺序输出：文件名（包括相对路径），压缩编码，哈弗曼编码表，最后一个字符长度。

若非文件，即为文件夹，则先写入文件夹名，然后遍历其下的所有文件（夹），递归调用encode对他们进行压缩。

2.2 Decode类：

* 构造时需要传入压缩文件路径和输出路径
* oIn ObjectInputStream用以读入编码后的压缩文件
* outputPath用以记录输出路径
* decode 主要方法

先读入文件名（不管是文件夹还是文件都有的部分），判断是否为文件，若为文件夹则按路径创建文件夹并继续读下一个文件。

若为文件，则依次读入编码、哈弗曼编码表以及最后一位字符长度，并重构哈弗曼树。

然后对整个编码字符串进行一次遍历，一边按树查找，找到相应字符（叶子节点）并写入，然后回归根节点，不断循环。按输出路径输出解压文件。

**3、Main方法**

调用javafx的start函数，start里主要做页面构建、调用处理、成功与否判断及输出，不做过多赘述。

1. **遇到问题：**

1、压缩时总长不是8的倍数，导致最后一个字节不足八位，而解压缩时读取这个字节会自动补齐8位，导致有概率在解码的最后阶段失败报错。

解决：用一个变量k记录最后一个字符的长度，并在写入文件时一并写入，读取解压时前面的字节按8位处理，最后一个按k位处理即可。

2、压缩时需要可以同时选择文件与文件夹，fileChooser无法完成。

解决：上网查资料后选择JFileChooser并将其setFileSelectionMode设置为JFileChooser.FILES\_AND\_DIRECTORIES即可。

**四、优化：**

构造哈弗曼树速度非常快，时间主要花费在读入文件时的处理和编码的转化上。

1、开始时用一个map记录每个字符出现的次数，耗费时间很长，现改为用一个长度为256的数组。由于每个字符用8位二进制表示，最多只有256种可能性（-128--127），大幅加快了压缩速度。

2、编码转换时一开始使用了字符串相关方法（substring、append等），耗时很长。现改为数组存储+位运算，规避了字符串的使用，压缩与解压都大幅加速。

3、输出时使用了ObjectOutputstream类，直接按对象输出一个文件名字符串+一个编码数组+一个编码表map+一个末位int，读取时按顺序读取，非常方便

**五、测试数据：**

**1、单个文件：**

|  |  |  |  |
| --- | --- | --- | --- |
| 文件名 | 压缩率(%) | 压缩时间(s) | 解压时间(s) |
| 1.txt | 56.6 | 0.31 | 0.14 |
| 2.pdb | 44.19 | 0.06 | 0.02 |
| 3.evy | 98.14 | 0.09 | 0.02 |
| 4.gz | 81.12 | 0.03 | 0.02 |
| 5.hpgl | 47.42 | 0.06 | 0.05 |
| 6.ma | 57.97 | 0.06 | 0.2 |
| 7.pdf | 80.43 | 0.04 | 0.21 |
| 8.sgml | 65.22 | 0.03 | 0.02 |
| 9.htm | 69.61 | 0.12 | 0.12 |
| 10.cgm | 89.2 | 0.03 | 0.02 |
| 11.g3f | 83.49 | 0.03 | 0.02 |
| 12.gif | 92.89 | 0.04 | 0.06 |
| 13.jpg | 99.88 | 0.03 | 0.09 |
| 14.png | 76.44 | 0.03 | 0.05 |
| 15.ps | 59.36 | 0.03 | 0.04 |
| 16.svg | 66.51 | 0.02 | 0.05 |
| 17.tif | 96.68 | 0.05 | 0.06 |
| 18.xbm | 33.01 | 0.03 | 0.01 |
| 19.msh | 64.08 | 0.05 | 0.07 |
| 20.mov | 94.72 | 0.22 | 0.15 |
| 21.mpeg | 98.18 | 0.05 | 0.08 |
| 22.jgs | 40.96 | 0.09 | 0.04 |
| 23.v5d | 93.49 | 0.07 | 0.06 |
| 24.wrl | 61.3 | 0.02 | 0.01 |
| 25.aiff | 63.93 | 0.03 | 0.03 |
| 26.au | 79.2 | 0.03 | 0.01 |
| 27.mp3 | 99.14 | 0.13 | 0.12 |
| 28.ra | 98.88 | 0.03 | 0.03 |
| 29.wav | 96.72 | 0.04 | 0.02 |
| 30.ram | 57.14 | 0.03 | 0.01 |
| 31.aiff | 88.56 | 0.04 | 0.04 |
| 32.aiff | 88.71 | 0.06 | 0.07 |
| 33.aifc | 87.3 | 0.05 | 0.09 |
| 34.tsv | 49.5 | 0.03 | 0.03 |
| 35.avi | 87.1 | 0.03 | 0.06 |

**2、文件夹：**

|  |  |  |  |
| --- | --- | --- | --- |
| 文件夹名 | 压缩率(%) | 压缩时间(s) | 解压时间(s) |
| 1 | 61.07 | 0.31 | 0.45 |
| 2 | 63.43 | 0.80 | 1.06 |
| 3 | 64.53 | 0.25 | 0.34 |

**3、大文件：**

|  |  |  |  |
| --- | --- | --- | --- |
| 文件名 | 压缩率(%) | 压缩时间(s) | 解压时间(s) |
| 1.jpg | 99.72 | 1.16 | 0.9 |
| 2.csv | 64.15 | 4.47 | 5.80 |
| 3.csv | 63.7 | 3.15 | 3.26 |