MyZip开发文档

概念简述

MyZip是一款基于**Huffman**算法的具备压缩和解压缩功能的软件。便携语言为java。**哈夫曼编码(Huffman Coding)**，又称霍夫曼编码，是一种编码方式，哈夫曼编码是**可变字长编码(VLC)**的一种。Huffman于1952年提出一种编码方法，该方法完全依据字符出现概率来构造异字头的平均长度最短的码字，有时称之为最佳编码，一般就叫做Huffman编码（有时也称为霍夫曼编码）。

https://en.wikipedia.org/wiki/Huffman_coding

Design and Implementation

> **基本原理:**
>
> - 读入待压缩文件，计算文件中各个字符的权重。
> - 根据权重构建Huffman树

- 根据Huffman树获得各个字符的Huffman编码，建立编码与字符之间的映射关系
- 哈夫曼编码生成以后,压缩文件是将文件中每一个字节读出后用其哈夫曼编码替换,满八位后写入压缩文件,不满八位的读出下一字节凑成八位,这样边读边写,直到文件末尾.
- 解压缩过程与压缩过程相反,从压缩文件中读一字节(八位)缓存,然后一位一位的解码,直到读到一个哈夫曼编码,用其对应的字节数据替换写入解压文件,这样边读边解码边写,直到文件末尾.
- 当然写压缩文件内容前,要先写入码表(原文件的编码信息:字节—-频率)用于解压缩时还原哈夫曼树及哈夫曼编码。

✎ 创建Huffman树:

- 创建Huffman树的节点:： class Node;包含左右节点（left，right），父节点（right），权重（weight），以及 对应的值（value）
- 计算字符的权重： class Frequency ;对于一个输入流（InputStream input)，8位可能有256种可能，创建一个int counts[] = new int[256];对于每次出现相应字符就增加，直到文件流终止。并建立get与set方法，以备后续调用。
- 创建Huffman树： class Huffman; 包含Frequency counts对象，根节点以及Node[]存储所有的字符对应的节点。调用buildTree()方法
    1. 如果字符在输入流中存在（counts.getFrequency(i) != 0）；
       创建节点，加入到nodes[i]; 利用PriorityQueue对所有节点按照权重进行排序；
       PriorityQueue queue = new PriorityQueue<>(257, new

Comparator() {
@Override
public int compare(Node o1, Node o2) {
return o1.weight - o2.weight;
}
});

2. 处理输入流结尾： nodes[END] = new Node(END, 1);
3. 当存在除输入流结尾之外的节点时，取出queue的两个节点 （poll()）,创建parent节点，value为预先定义的TEMP_NODE，权重为取出的节点权重之和，放入queue中，直到queue.size()==0；完成了树的构建，取出root；
4. 获取节点的编码：Code（int chars）
   如果nodes[chars]存在，且其父节点存在，判断cur(当前节点)是父节点的左节点还是右节点，如果为左，str = “0” + str;否则 str = “1” + str; 设置cur = cur.parent;循环直到根节点，此时完成了对于各个节点的Huffman编码。
5. 获取编码对应的节点的字符值：Value(String code)
   设置cur(当前节点)为根节点，依次取出二进制的每一位，从根节点向下搜索，1向右，0向左，到了叶子节点(命中)，退回根节点继续重复以上动作。此时cur.value为对应的字符值。
6. 将树写入压缩文件流中：writeEncoding(DataOutputStream out)
   写入字符与对应权重的映射
7. 读取并重构树： readEncoding(DataInputStream in)
   chars = in.readByte(); 读取字符
   number = in.readInt(); 读取权重
   构建树buildTree();

压缩Compress

- 压缩的输出：文件路径，文件流长度，文件流内容
- 判断是否为文件夹并分情况进行压缩处理：

    1. 是文件夹
       a. 空文件夹：file.listFiles().length == 0， 在输出流中写入文件路径
       dos.writeUTF(pathOfFile)； 写 入 长 度 ， -1 ：dos.writeInt(-1);
       b. 非空文件夹：递归调用，直到是单个文件，进入单个文件的压缩
    2. 不是文件夹，是单个文件
       a. 首先写入文件路径
       b. 如果是空文件，写入长度，0；
       c. 非空文件，建立输入流，读取并写入输出，利用ByteArrayOutputStream baos，写入长度为baos.size();
- 按位读取并且输出：
  一 个 字 符 即 一 个 byte 是 8 个 bit, 建 立 class BitOutputStream（），按位读取，读满一个8bit 就flush,直到读完，输出
- 压缩文件里的输出：
  写入映射，以及通过继承BitOutputStream（）来实现按位输出。

解压缩Dempress

- 解压缩的输出：

- 以“ / “（unix系统下）作为文件的分割符，读取需要解压缩的文件的路径名氛围文件夹folder（如果有的话）

    1. 读取文件路径， readUTF();
    2. 读取内容长度，readInt()；
        a. 空文件夹：readInt() == -1；创建空文件夹；
        b. 空文件：readInt() == 0；创建空文件；
        c. 非空文件，创建新文件，读取长度为readInt()的byte；解码并且输出
- 按位读取并且输出：
  一个字符即一个byte是8个bit，建立class BitInputStream（），按位获取，读满一个8bit就flush,直到读完，输出
- 解压缩文件里的输出：
  建立输入流继承class BitInputStream（），实现按位读取，解码输出

GUI实现

- 界面：
  BorderPane：top(vBox), center(gridPane)
  vBox: label help(提供用户使用帮助)，label authority（提供版权信息）
  GridPane: 选择文件，文件夹的按钮控件，显示文件路径的文本框控件，压缩与解压缩按钮控件。
- 进程结束后，提示任务成功以及耗时或者提供错误信息

MyZip性能测试：

文件类型	文件名称	压缩时间（s）	解压缩时间（s）	原文件大小	压缩文件大小	压缩率
test1 - single file compression	a_chemical.pdb	0.022ms	0.033	35KB	15KB	42.86%
	a_document.evy	0.005	0.009	10KB	11KB	110%
	a_document.gz	0.001	0.001	143 bytes	656 bytes	458%
	a_document.hpgl	0.024	0.024	97 KB	46 KB	47.4%
	a_document.ma	0.050	0.029	216 KB	126 KB	58.3%
	a_document.pdf	0.030	0.025	130 KB	105 KB	80.8%
	a_document.sgml	0.004	0.004	18 KB	12 KB	66.7%
	a_htm.htm	0.135	0.098	685 KB	352 KB	51.4%
	a_image.cgm	0.008	0.016	12 KB	12 KB	100%
	a_image.g3f	0.006	0.003	7 KB	7 KB	100%
	a_image.gif	0.012	0.002	3 KB	4 KB	133%
	a_image.jpg	0.008	0.005	21 KB	23 KB	109%
	a_image.png	0.001	0.001	1 KB	2 KB	200%
	a_image.ps	0.018	0.015	106 KB	63 KB	59.4%
	a_image.svg	0.003	0.002	5 KB	4 KB	80%
	a_image.tif	0.005	0.004	16 KB	17 KB	106%
	a_image.xbm	0.001	0.001	2 KB	960 bytes	46.9%
	a_model.msh	0.048	0.027	222 KB	144 KB	64.9%
	a_movie.mov	0.412	0.349	1.7 MB	1.6 MB	94.1%
	a_movie.mpeg	0.032	0.022	111 KB	110 KB	99.1%
	a_object.igs	0.119	0.096	1.4 MB	560 KB	39.1%
	a_object.v5d	0.201	0.120	531 KB	498 KB	93.8%
	a_object.wrl	0.003	0.003	19 KB	12 KB	63.2%
	a_sound.aiff	0.006	0.003	22 KB	15 KB	68.2%
	a_sound.au	0.004	0.002	10 KB	8 KB	80%

	a_sound.mp3	0.305	0.308	1.3 MB	1.3 MB	100%
	a_sound.ra	0.015	0.012	28 KB	29 KB	103%
	a_sound.wav	0.005	0.002	5 KB	6 KB	120%
	a_sound1.ram	0.001	0.001	35 bytes	158 bytes	451%
	a_sound2.aiff	0.022	0.016	45 KB	41 KB	91.1%
	a_sound3.aiff	0.072	0.060	178 KB	160 KB	90%
	a_sound4.aifc	0.096	0.060	389 KB	341 KB	87.7%
	a_tsv.tsv	0.002	0.002	10 KB	5 KB	50%
	a_video.avi	0.009	0.006	34 KB	31 KB	91.2%
	anna_karenina.txt	0.226	0.193	2 MB	1.1 MB	55%
test2 - empty file and empty folder	empty_file	0	0.001	Zero bytes	16 bytes	infinite
	empty_folder	0.001	0	Zero bytes	18 bytes	infinite
test3 - folder compression	1	0.385	0.345	2.4 MB	1.5 MB	62.5%
	2	1.342	1.156	9.9 MB	6.2 MB	62.6%
	4	0.338	0.307	2.6 MB	1.7 MB	65.4%
test4 - single large file compression	1.jpg	5.218	3.541	20.7 MB	20.7 MB	100%
	2.csv	59.65	52.92	441.8 MB	277.1 MB	62.7%
	3.csv	83.859	71.728	643.4 MB	411.7 MB	64%
	4	49.132	22.141	1.02 GB	127.1 MB	12.2%

性能分析：

1. 对于不同类型的问价，压缩率差别很大，对于相同类别的压缩率，压缩性能相似。对于给定的测试文件压缩时间在0～90s不等，解压缩时间在0～70s不等。
   压缩率平均60%左右。
2. 对于空文件，空文件夹，以及jpg等原先已经被压缩的文件，由于往里面写入了文件路径，长度等信息，压缩的文件大小反而增大，与主流软件性能一致。