**深 圳 大 学 实 验 报 告**

**课程名称： 算法分析与设计**

**实验项目名称： 赫夫曼编码**

**学院： 计算机与软件学院**

**专业： 软件工程**

**指导教师： 李荣华**

**报告人： 洪继耀 学号： 2014150120 班级： 02**

1. 实验目的

(1)掌握Huffman编码。

(2)掌握文档压缩的基本原理和应用。

1. 算法详解

1. 霍夫曼编码使用**变长编码表**对源符号（如文件中的一个字母）进行编码，其中变长编码表是通过一种评估来源符号出现概率的方法得到的，出现概率高的字母使用较短的编码，反之出现概率低的则使用较长的编码，这便使编码之后的字符串的平均长度、期望值降低，从而达到无损压缩数据的目的。

2. 例如，在英文中，e的出现概率最高，而z的出现概率则最低。当利用霍夫曼编码对一篇英文进行压缩时，e极有可能用一个比特来表示，而z则可能花去25个比特（不是26）。用普通的表示方法时，每个英文字母均占用一个字节，即8个比特。二者相比，e使用了一般编码的1/8的长度，z则使用了3倍多。倘若我们能实现对于英文中各个字母出现概率的较准确的估算，就可以大幅度提高无损压缩的比例。

3. **霍夫曼树又称最优二叉树，是一种带权路径长度最短的二叉树。**所谓树的带权路径长度，就是树中所有的叶结点的权值乘上其到根结点的路径长度（若根结点为0层，叶结点到根结点的路径长度为叶结点的层数）。树的路径长度是从树根到每一结点的路径长度之和，记为WPL=（W1\*L1+W2\*L2+W3\*L3+...+Wn\*Ln），N个权值Wi（i=1,2,...n）构成一棵有N个叶结点的二叉树，相应的叶结点的路径长度为Li（i=1,2,...n）。可以证明**霍夫曼树的WPL是最小的**。

4. 实现霍夫曼编码的方式主要是创建一个二叉树和其节点。

(a)这些树的节点可以存储在数组里，数组的大小为符号数的大小n，而节点分为是叶节点与内部节点。

一开始，所有的节点都是叶节点，节点内有三个字段：符号 权重 指向父节点的链接。而内部节点内有四个字段： 权重 指向两个子节点的链接 指向父节点的链接

(b)基本上，我们用'0'与'1'分别代表指向左子节点与右子节点，最后为完成的二叉树共有n个叶子节点与n-1个内部节点。**过程中，每个叶子节点都包含着一个权重，两两叶子节点结合会产生一个新节点，新节点的权重是由两个权重最小的终端节点权重之总和，并持续进行此过程直到只剩下一个节点为止。**

实现霍夫曼树的方式有很多种，可以使用优先队列简单达成这个过程，给与权重较低的符号较高的优先级（Priority）

(c)**算法的时间复杂度为O（n log n）；因为有n个终端节点，所以树总共有2n-1个节点，而使用优先队列每个循环需要O（log n）**

1. 实验代码

1. 以字母(Character)为基础的压缩

1. 文本解析：将cacm.all文件分解成一个个的字母

代码：



1. 字频统计：统计每个字母出现的词频

代码：

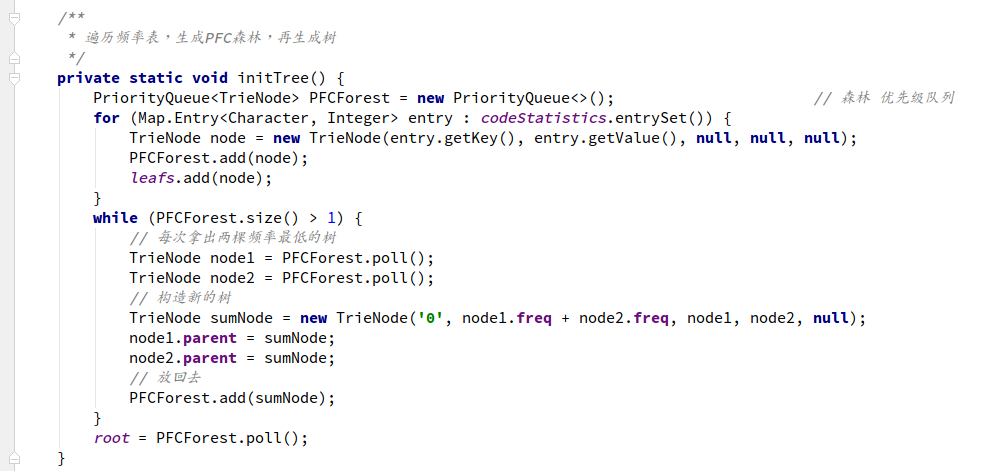


读取结果在文件夹“编码表和频率表”里

1. Huffman编码：根据Huffman编码的原理，对每个字母进行编码。给出一个编码字典。

代码：

1. 建树

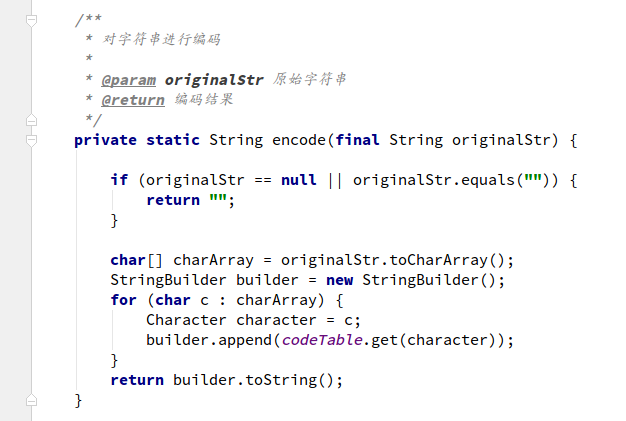


1. 构建编码字典

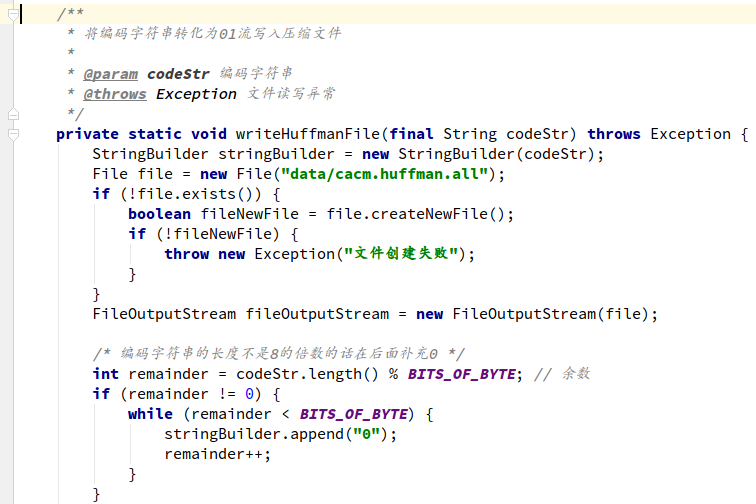


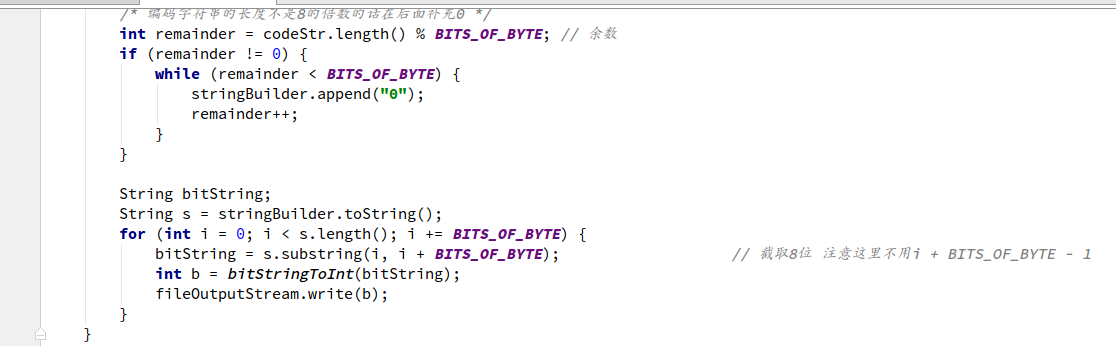
1. 文档压缩：根据Huffman编码，压缩文件。

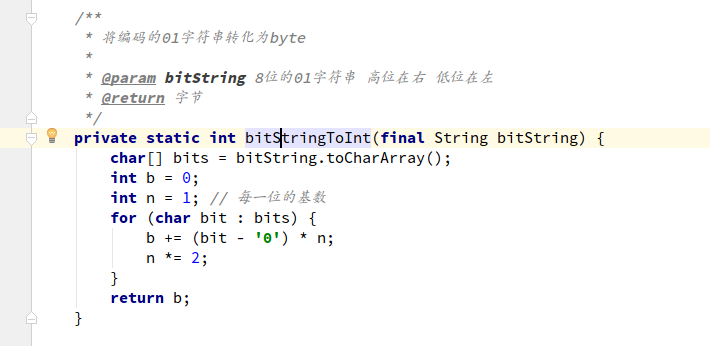
(a)字符串编码



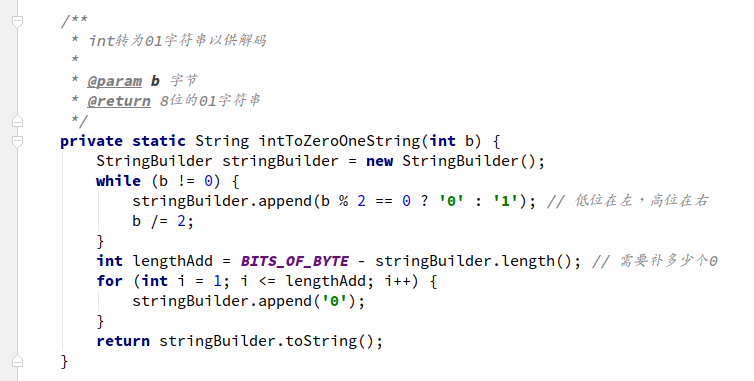
(b)文件读写



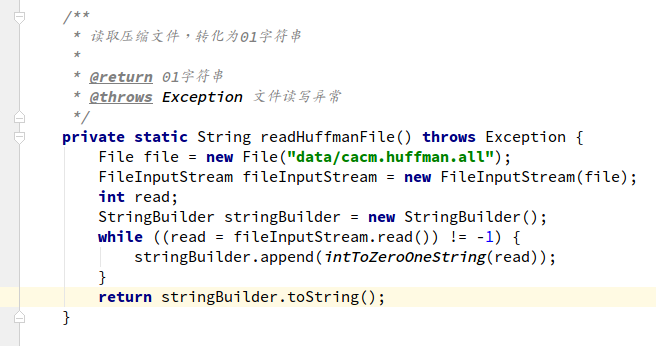




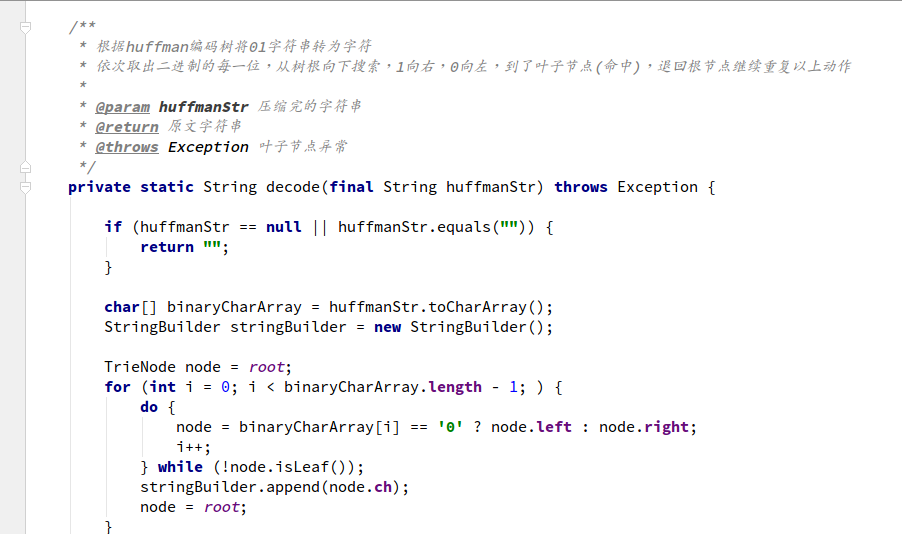
1. 文档还原：对压缩后的文档进行解压缩。
   1. 转换函数

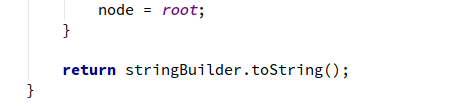


* 1. 文件读写

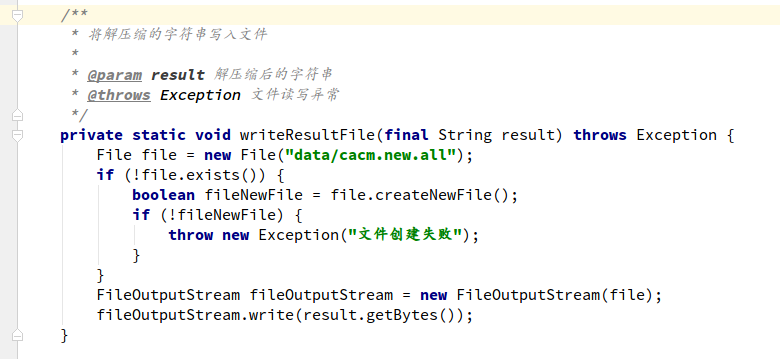


* 1. 解压缩





* 1. 写入结果



2. 比较

比较不同的压缩方法的效率

|  |  |  |
| --- | --- | --- |
|  | huffman字母压缩 | Rar压缩 |
| 时间效率 | 165ms（4次平均） | 532ms |
| 压缩效率 | 压缩比62.5% | 压缩比25.8% |
| 解压效率 | 104ms（4次平均） | 160ms |

三、实验总结

终于明白压缩的原理之一了。

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。