

专业课程实验报告

课程名称： 交互式媒体

计算机与信息科学学院 软件学院

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | | 创作第一个网页 | | | |
| 实验时间 | | 2018.12.16 | 实验类型 | | □验证性 ☑设计性 □综合性 |
| 1. 实验目的  * 编程实现各种无损压缩算法并进行比较分析   + 编程实现huffman，自适应huffman，算术编码和LZW编码算法   + 自己设置三种不同的统计数据源来进行测试，就每种数据源的压缩率，比较和评价各种算法的性能  1. 实验要求  * 统计数据源要求： * String长度大于30,构成String的每个element出现频率大于4次 * 输入输出要求： * 输入可以通过文本或者屏幕输入\读取文件 * 输出可以保存到文本或者屏幕显示均可，需要输出每个字符的编码，以及整个输入string的整体编码，string对应算法的压缩率，算法的运行时间 | | | | | |
| 1. 测试数据和执行结果 （在给定数据下，执行操作、算法和程序的结果，可使用数据、图表、截图等给出）   **综述：**  哈夫曼算法相对比较简单，所以我使用了winform窗口应用程序和C#；其他因为多处用到STL的vector，map，deque等所以使用的C++；压缩效率我是用的公式为：（压缩前位数-压缩后位数）/压缩前位数；时间以毫秒为单位。测试字符串第一个是乱码，第二个是有意义的句子，第三个是含了多种符号的句子。  **Huffman Coding:**   1. 拖进所需的控件后在button中进行编码。先新建字符串、各字符权重和出现次数的数组，再接受用户输入的字符串。在开始编码时就记下当前时间以便结束计算编码耗时。      1. 先用struct设置节点，先找到两个最小值赋给min1,min2。 2. 统计各个字符出现的次数记为权重放在weight数组中。接着初始化树。建树时先找出最小的两位数，然后不断用strcut的min.s1, min.s2给它们的父母节点和左右孩子节点赋值。 3. 将输入字符利用所建的树进行编码。这里比较简单，直接遍历树找到树所在的遍历序列即为本字符的编码。输出总的编码需要新建一个总编码数组，赋值使其不断加上新的到的单个字符编码。 4. 结束编码，再获得一个当前时间。将此时间和开始时间相减获得总时间。 5. 效率的计算首先统计编码前位数。在ASCII码中26字母均为8位2进制尾数表示，所以直接长度\*8即可；而编码后位数则在显示各个字符位数时，不断加上个各字符编码长度即可。 6. 最后显示各个属性：效率、时间、总编码。   **代码：**            **LZW Coding:**   1. 在字典算法里我是用了C++语言。因为需要用到字典这个数据结构，所以在头文件先引用了map, iterator, vector、计算时间的头文件:time, sys/timeb.h, math.h 2. 编码代码在compress()函数里。首先得到当前时间，这里的时间因为计算到毫秒，所以首先用timeb的变量做个简单计算得到：。 开始新建一个字典大小：256，ASCII码范围为0-255，所以初始化的字典先存着单独的字符，利用map<string,int>结构的字典将256字符放入此字典 3. 增加接收的字符串到字典中：首先得到的字符为char c，再将下一个字符string w与其相加得到wc串判断是否在字典中，如果不在则加入字典并将dictSize++作为其编码；否则，将对应的编码push\_back进compressed向量得到最终编码。结束编码时再记录时间可得编码总时间。 4. 在main函数里统计压缩比。压缩前位数直接长度\*8可得；压缩后位数需要利用一个string的迭代器遍历compressed向量，并判断每个字符是否是ASCII码：是则加8，否则加上计算的编码位数（二进制位数由log2 计算）   **Code:**            **Arithmetic Coding:**   1. 算术编码需要利用字符出现的概率区间来决定编码。所以首先在probe()中统计单个字符的概率并输出。Encode函数中先设置好range的最低和最高点。 2. 开始先给第一个字符安排区间。之后再将每个字符一个一个比对，判断：如果读取字符是存取字符的第一个，那区间的最低值就是此字符的概率区间最低值，最高值为此字符概率乘以接收字符概率+最低值；否则最低值为此字符概率乘以接收下一个字符概率+此字符区间最低值，最高值在最低值的下一个字符概率基础上再加自己概率。 3. 简而言之，2步骤为每读入一个字符就在原有的区间基础上等按比例缩小到最新得到的区间。 4. 编码时间获得步骤同其他算法。在main函数中输出各个属性值。而编码位数需要再一次转换乘2进制，因为编码是小数所以利用decimal()得到   **Code:**                **Adaptive Huffman Coding:**   1. FGK算法肯定需要遍历树并且更新树，所以首先定义了一个二叉树以便之后的操作。二叉树的域有：定义的节点、swap交换节点、增加节点、删除、查找节点、得到兄顶性质、得到根和得到祖先。哈夫曼树的方法中有最重要的权重赋值、找到最大值。并且哈夫曼树用到一个泛型向量以获得相应节点的值即为接收的字符。 2. 哈夫曼树中，得到编码的方法中（p\_n为参数节点）首先分为两个：p\_n是根节点（第一个节点），p\_n不是根节点。p\_n不是根节点时需要一直遍历到根，左孩子都为0右孩子为1。 3. FinLarge（）方法是找到权重大的节点。在encode方法中首先设置nyt为根节点，然后接受字符时不断将字符编码赋值存入findCode数组。在赋值后如果存在树中则直接获得权重以便下一次比较，不存在树中时则新建字数代替原nyt节点，得到的新字符放进buffers中并向上依次增加节点的权重，再设置新nyt节点为原nyt的左孩子。最后总编码由finalCode数组获得。 4. Main中获得各属性：时间、效率和总编码并显示。   **Code:**  **Huffman.cpp**                      **Main.cpp**    **实验结果：**  **三个测试字符串:**  1.TOBdEOrtBEOrTOBdwEOtrNwOTETBdwdtrwEtBwrdt  2.Thestudy,whichcontrolsfourdemographicvariables,providesmostofthefactorsthat  3.(Daviesetal.,1995;Magnussonetal.,2001,2003;Padel&Foster,2005;Lea&Worsley,2005;)  **Huffman Coding:**            **3.**        **LZW Coding:**  **1.**      **2.**          **3.**            **Adptive Huffman Coding:**  **1.**      **2.**                    **Arithmetic Coding:**  **1.**    **2.**    **3.** | | | | | |
| 五、实验结果分析及总结（对实验的结果是否达到预期进行分析，总结实验的收获和存在的问题等）  **关于选取测试串：**  三个测试串分别为普通乱码和有意义的字段。第一个为较乱且无意义的串，可进行初步测试，后面两个字符串模拟了有意义的文本内容且又分为两类：第二个为一般的长句子，只由26字母构成的；第三个为加了许多特殊符号的字符串**。**   1. 根据算法来看明显哈夫曼的算法是最简单的，效率也较高。算术算法需要使用概率区间，相对来说不是最优的，因为此算法开始必须先扫描统计以便所有字符出现的概率，之后才不断更新区间。字典算法需要用到map的结构来构建字典，在每次接收字符时与上一个合并获得新的串加入字典中。 2. 自适应哈夫曼算法最常用的有两种，这里使用了FGK算法。它在哈夫曼的算法上给每个节点加了权重，赋予了兄弟性质。同时可以满足动态接收字符串赋值，比哈夫曼算法更高效。在算法中0节点的使用在于接收之前没接受过的字符，一旦接收新字符就在原有左节点增加两个孩子：左为0，右为此字符。在接收已有的字符时，如果破坏了兄弟性质则需交换子树。相对来说更加动态。 3. 在算术算法的效率中，因为编码结果是概率区间，转化成二进制小数时有可能出现无限小数所以需要设定结果编码长度以防死循环。由此造成结果编码长度不准确，导致压缩率计算不准确。 4. 关于三种算法的统计平均值统计：     由表可看出压缩效率最高的算法是算术算法，但因为计算编码长度有差异所以待定。LZW的压缩效率最低，因为在不断接收字符时需要新建字符串加入字典中，如果字符串重复较高时字典才优先选择此算法。  时间上最短是哈夫曼算法；而自适应哈夫曼算法相对来说较高，因动态接收字符时可以随时得到二叉树的编码，可因为需要随时检查是否违法兄弟性质，在时间上最耗时。同时哈夫曼算法是无损算法，时间较短，效率相对较高随意比较适合短字符串的压缩。 | | | | | |
| 教  师  评  阅 | 实验内容和设计（含操作过程、算法或代码）（A-D）： | | |  | |
| 实验结果（A-D）： | | |  | |
| 实验分析和总结（A-D）： | | |  | |
| 实验成绩（A-D）：  反馈评语： | | | | |