改进的四进制哈夫曼算法

胡智宏，尹小正，路立平
（郑州轻工业学院 电气信息工程学院，河南 郑州 450002）

摘 要：提出了一种改进的四进制哈夫曼树的生成算法，通过分析算法的平均码长和编码效率，论证了算法相对于传统的四进制算法的优点。并用C语言分别实现两种算法，进行了压缩比和压缩时间的比较，证明了改进算法在压缩比和压缩速度上的提升。

关键词：数据压缩；哈夫曼；四叉树

中图分类号：TP301.6 **文献标识码**：A **文章编号**：1674-7720(2012)10-0065-02

An improved quaternary Huffman algorithm

Hu Zhihong, Yin Xiaozheng, Lu Liping
(College of Electric and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China)

Abstract: This paper proposed an improved quaternary Huffman tree algorithm, demonstrated the advantages of improved quaternary algorithm by analyzing the average code length and coding efficiency compared to conventional quaternary algorithms. And implemented the two algorithms by C language, made a comparison between the two algorithms in ratio and compression time, proved algorithm compression ratio and compression speed of the upgrade.

Key words: data compression; Huffman; quadtree

数据压缩是计算机科学中的一项重要技术[1]。Huffman算法作为通用数据压缩算法[2]已经成为大多数数据压缩程序的基础[3]。目前应用最多的是二进制Huffman算法。由于四进制Huffman算法每次可以处理2 bit的数据，相对于二进制有占用内存空间小，解码速度快的优点，因此在一些应用场合用四进制Huffman算法比二进制Huffman算法可以获得更好的性能。

Huffman树是一种带权路径WPL(Weighted Path Length)最小的树。衡量一棵Huffman树性能的重要指标有平均码长和编码效率[4]。

平均码长：

$$l_p=\sum_{i=1}^{n} l_i p_i \tag{1}$$

其中，l_i表示各符号的码长；p_i为各符号的概率。

编码效率：

$$\eta=\frac{H(X)}{l_p}=-\sum_{i=1}^{N} P(a_i)\lg P(a_i) \tag{2}$$

式中，$H(X)$表示信源的熵；$P(a_i)$表示符号a_i的概率。

1 四进制Huffman算法

传统的四进制Huffman算法与二进制Huffman算法类似，都是采用递归调用的方法，区别是二进制Huffman每次取最小的2个节点构造新节点，四进制是取最小的4个节点构造新节点。传统的四进制Huffman算法过程如下：

(1)将字符按照概率由大到小的顺序排列；建立未处理节点集合；

(2)将4个最小的概率组合作为一个新的节点；将4个最小的概率从未处理节点集合中去除，并加入新组合的节点，重新排好；

(3)重复步骤(2)直到剩余一个根节点为止。

这种算法有一种情况没有考虑到：当最后剩余的未编码节点不是4个的时候，就会产生在根部的子节点不是4个的情况，也就是权重最小的节点没有使用，而使用了权重最大的节点。例如，有12个待编码字符(A，B，C，D，E，F，G，H，I，J，K，L)，其概率分别是(0.23，0.22，0.13，0.12，0.07，0.06，0.05，0.04，0.03，0.025，0.015，0.01)，传统算法生成的四进制Huffman树如图1所示。

由图1可看出，根节点只有3个子节点，这就造成一个权重小的节点未使用而浪费，增加了平均码长，降低了编码效率。

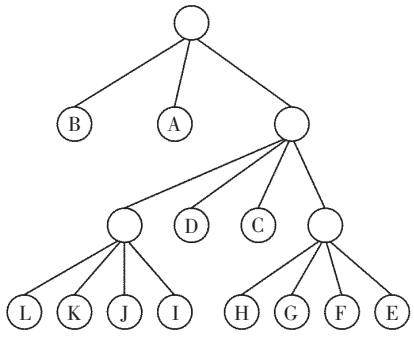

图 1 四进制 Huffman 树

由式(1)可知图 1 的 Huffman 树平均码长为：

$$l_p=\sum_{i=1}^{12} l_i p_i=3.70 \text{ bit}$$

由式(2)可知图 1 Huffman 树的编码效率为：

$$\eta=\frac{H(X)}{l_p}=83.08\%$$

2 改进的四进制 Huffman 算法

在传统四进制 Huffman 算法中，树根可能的子树有 2 棵、3 棵或者 4 棵，在树根的子树为 2 棵和 3 棵时，均有高层次的节点没有利用，不能保证权重较大（出现频率较高）的节点出现在较高的层次中，也就不能保证得到的编码长度是最短的。由图 1 可看出树根只有 3 个子节点，剩下的一个为空，这样就造成了浪费，增加了平均码长，减小了编码效率。

为了使编码长度最短，需要调整树的结构，上移权重较大的节点，保证树根和高层次节点都有 4 个子节点，只在最底层出现空缺。

n 个树叶的四进制 Huffman 树共有 $n+\text{ent}(\frac{n+1}{3})$ 个节点，除去树根，剩下的节点除以 4 的余数即为不能构成完全 4 个子节点的个数 m。当 m 为 0 时，可以构成每个父节点都有 4 个孩子的四进制 Huffman 树；当 m 不为 0 时，肯定有一个父亲节点没有完全的 4 个子节点，此时，要保证空缺出现在 Huffman 树最底层。

改进后的算法如下：

(1)根据 n 个字符（树叶）计算节点的总个数；

(2)计算不能构成完全 4 个子节点的个数 m；

(3)判断 m 是否为 0。若是，执行步骤(5)；若否，执行步骤(4)；

(4)先用权重最小的 m 个节点生成它们的父节点 M。在节点集合中删除 m 个节点并加入它们的父节点 M，构成节点集合 n^*；

(5)用 n^* 按照传统的四进制 Huffman 的算法构造剩下的 Huffman 树。

用改进的算法重新构造图 1 的四进制 Huffman 树，如图 2 所示。

由图 2 看到，根节点具有 4 个子节点，而且空缺移到了最底层。

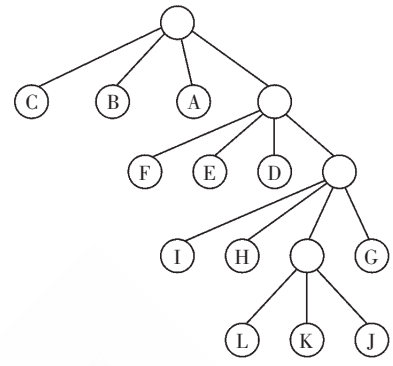

图 2 改进后的四进制 Huffman 树

由式(1)可知图 2 所示 Huffman 树的平均码长为：

$$l_p=\sum_{i=1}^{12} l_i p_i=3.28 \text{ bit}$$

由式(2)可知图 2 所示 Huffman 树的编码效率为：

$$\eta=\frac{H(X)}{l_p}=93.72\%$$

虽然改进后的 Huffman 树的层数多了 1 层，但是平均码长却小了，编码效率比改进前的 Huffman 树提高了 10.64%。同时由于压缩比提高，需要输出的数据和压缩耗时也会相应地减少。

在实际情况中，如果不能构成完全 4 个子节点的个数为 0，即树中每个父节点都有 4 个子节点，则改进前后的算法结果一样。

3 算法比较

用 C 语言实现了改进前后的四进制 Huffman 压缩算法，随机选取了 7 个不同类型的文件，并用随机数产生了由 256 个节点构成的文件，分别压缩这 8 个文件，比较它们的压缩比和压缩时间，比较结果如表 1 所示。

由表 1 可以看出，在不能构成完全 4 个子节点的个数不为 0 的情况下，改进后的算法比传统的算法在压缩比上有较大提高，压缩速度也提高很多；在不能构成完全 4 个子节点的个数为 0 的情况下，两者的压缩比没有差别，压缩速度差别也不大；而且两种压缩算法的解压缩过程完全相同。

Huffman 算法是一种有效的无损压缩算法，针对传统四进制 Huffman 算法的不足，提出了改进的四进制 Huffman 算法。主要改进了在不能构成完全 4 个子节点的个数不为 0 的情况下的压缩效率和压缩速度。这种改进不论出现哪种情况，用算法生成的四进制 Huffman 树是平均码长最短、编码效率最高的四进制 Huffman 树。最后将两种算法用 8 个大小不同的文件进行了对比，实验表明，改进的算法在压缩比和压缩速度上有绝对的优势。

参考文献

[1] 孙学琛，李新洁.哈夫曼树的图形化算法设计[J].山东理工大学学报(自然科学版)，2008，22(6)：108-110.

表 1 改进 Huffman 和传统 Huffman 算法的比较

源文件大小/B	节点个数	传统 Huffman 算法		改进 Huffman 算法		改进百分比/%	
		压缩后/B	耗时/s	压缩后/B	耗时/s	压缩率	耗时
2 917 888	257	1 965 029	0.404 6	1 717 725	0.307 2	12.59	24.07
5 362 751	256	5 363 233	0.718 5	5 363 233	0.713 6	0.00	0.68
4 431 872	257	5 074 429	0.712 9	4 723 634	0.644 6	6.91	9.58
2 049 851	257	2 358 333	0.300 4	2 198 059	0.268 5	6.80	10.62
2 246 656	257	1 681 223	0.339 1	1 519 091	0.237 8	9.64	29.87
16 103 967	257	17 665 428	2.666	15 606 291	2.368	11.66	11.18
34 320 370	257	38 457 272	5.816	34 011 078	4.999	11.56	14.05
330 895 360	257	380 603 556	74.969 2	329 902 602	62.882 2	13.32	16.12

[2] SALOMON D.数据压缩原理与应用(第 2 版)[M].吴乐南,译.北京:电子工业出版社,2003.

[3] 张凤林,刘思峰.Huffman*:一个改进的 Huffman 数据压缩算法[J].计算机工程与应用,2007,43(2):73-74.

[4] 吴家安.数据压缩技术及应用[M].北京:科学技术出版社,2009.

(收稿日期:2012-01-20)

作者简介:

胡智宏,男,1974年生,硕士,副教授,主要研究方向:电子信息及嵌入式系统应用。

尹小正,男,1978年生,硕士研究生,主要研究方向:数据压缩的应用研究。

路立平,男,1961年生,硕士,副教授,主要研究方向:传感器,电子测量和检测控制。