文章编号: 2095-7386(2015) 04-0060-05
DOI: 10. 3969 / j. issn. 2095-7386. 2015. 04. 015

基于前缀码的快速编码算法研究

王防修
(武汉轻工大学 数学与计算机学院,湖北 武汉 430023)

摘 要: 针对目前符号序列的编码存在编码速度慢的问题,提出了一种通过减少平均查找长度来提高编码速度的算法。根据符号概率的大小,设计了顺序查找、大概率优先查找和小概率优先查找三种编码算法。通过对这三种编码算法的平均查找长度的分析比较,结果表明: 大概率优先查找算法的平均查找长度最短。根据符号本身的大小,设计了折半查找和二叉排序树查找两种编码算法。通过对这两种编码算法的平均查找长度的分析比较,结果表明折半查找编码算法的平均查找长度最短。因此,最优的编码算法应从大概率优先查找算法和折半查找算法之中选择其一。算例表明,为了提高符号序列的编码速度,对同一符号序列的编码,应从大概率优先查找算法和折半查找算法中选择平均查找长度最短的算法作为编码算法。

关键词: 顺序查找; 折半查找; 二叉排序树查找; 平均查找长度; 编码速度
中图分类号: TP 391 **文献标识码:** A

Fast encoding algorithm research based on prefix codes

WANG Fang-xiu
(School of Mathematics and Computer Science, Wuhan Polytechnic University, Wuhan 430023, China)

Abstract: In view of the current symbol sequence encoding having the problem of slow encoding speed, this paper proposes an algorithm to improve the encoding speed by reducing the average search length. According to the symbol probability, it designs three coding algorithm that are a sequential search, big probability priority first search and small probability priority first search. Through the analysis and comparison of the average search length of the three coding algorithm, teh results show that the average search length is the shortest for the big probability priority search algorithm. According to the size of the symbol itself, it designs two kinds of coding algorithm, one binary search and the other two binary sort tree search. Through the analysis of the average search length of the two coding algorithm, the results show that the average search length of the binary search algorithm is the shortest. Therefore, the optimal coding algorithm must be selected from one of big probability priority search algorithm and binary search algorithm. Examples show that, in order to improve the encoding speed of the symbol sequence ,for the same symbol sequences, one must be chosn between big probability priority search algorithm and binary search algorithm which has the shortest average search length as the coding algorithm.

Key words: sequential search; binary search; two binary sort tree search; average search length; coding speed

---

收稿日期: 2015-11-12.
作者简介: 王防修(1973-) ,男,副教授,E-mail: wfx323@126. com.
基金项目: 国家自然科学基金资助项目(61179032) .

1　引言

作为一种即时码,前缀码[1]在信息编码过程中得到广泛应用。其中,哈夫曼编码[2]、香农编码[3]和费诺编码是最常见的前缀码。设符号序列 $X = x_1x_2\cdots x_m$ 包含 n 个互异符号 $c_i(i = 1,2,\cdots,n)$,即 $\forall x_i \in X$,有 $x_i \in \{c_1,c_2,\cdots,c_n\}$。$b_i$ 是对应 c_i 的前缀码,p_i 是 c_i 在 X 中出现的概率,l_i 是 b_i 的码长。所谓编码,就是将 X 中的每个符号用对应的码字替换。对于符号序列 X 中的每个符号 x_i,如果存在 $c_j = x_i$,则用码字 b_i 替换 X 中的 x_i。当 X 中所有符号被码字全部替换,则整个编码过程结束。显然,编码时间取决于查找编码表的时间。

对于一个符号序列的编码,编码时间[4,5]的长短取决于平均查找长度的大小。如果平均查找长度大,则编码时间长; 如果平均查找长度小,则编码时间短。目前,针对编码速度的研究尚未文献报道。

为此,设计了 5 种不同的编码算法。通过对这些算法的平均查找长度的比较,找出了编码时间最优的算法。算例测试表明,通过平均查找长度的计算和比较,可以找到编码时间最短的算法。

2　编码算法

2.1　顺序查找法

对于符号序列 X 中的任何一个元素 x_i 而言,为了从编码表中找到对应的码字,需要从顺序表 $C = c_1c_2\cdots c_m$ 的第一个元素 c_1 开始,依次与元素 x_i 比较。如果存在某个符号 $c_j = x_i$,则 b_j 就是符号 x_i 的码字。将符号序列 X 中所有符号编码的过程如下。

(1) 令 $i = 1$ 和 $Y = \varphi$,它表示从 X 中的第一个符号 x_1 开始编码。

(2) 令 $j = 1$,它表示从 C 中的 c_1 开始查找 x_i 在 C 中的位置。

(3) 如果 $x_i = c_j$,则转步骤(4); 否则,转步骤(5)。

(4) 令 $Y = Y \cup \{b_j\}$,转步骤(6)。

(5) 令 $j = j + 1$。如果 $j \le n$,则转步骤(1); 否则,由于编码过程出错而终止计算机继续编码。

(6) 令 $i = i + 1$。如果 $i \le m$,则转步骤(2); 否则,编码过程结束。

如果算法中出现 $j > n$,则意味着符号序列中有某个符号无法编码。事实上,除非码字集本身存在问题,否则这种情况不可能发生。

2.2　大概率优先查找法

大概率优先查找本质上是一种顺序查找法,唯一不同的是大概率符号一定会比小概率符号要先查找到。也就说,对于任意两个不同符号 x_i 和 x_j 来说,如果 $p_i > p_j$,则查找到 x_i 所花费的时间一定要比查找到 x_j 所花费的时间短。为了做到这一点,只需要对概率序列 $P = p_1p_2\cdots p_n$ 进行降序排序即可。需要说明的是,在概率序列降序排列的过程中,顺序表 $C = c_1c_2\cdots c_m$ 和编码表 $B = b_1b_2\cdots b_m$ 都需要相应调换位置。总之,经过排序后,对 $\forall i \in \{1,2,\cdots,n\}$,符号 c_i 对应的概率是 p_i,而对应的码字是 b_i。同时,符号概率必须满足 $p_1 \ge p_2 \ge \cdots \ge p_n$。

当顺序表中的符号满足上述概率条件时,在此基础上用顺序查找法对符号序列进行编码,该编码所花费的时间一定比直接顺序查找法要短。

2.3　小概率优先查找法

跟大概率优先查找法一样,小概率优先查找法也是一种特殊的顺序查找法。顾名思义,小概率优先查找法就是小概率符号比大概率符号要先查找到。为了做到这一点,只需要将概率序列 $P = p_1p_2\cdots p_n$ 进行升序排序。同样,在排序的过程,如果出现需要 p_i 与 p_j 交换,则相应的需要进行 $c_i \leftrightarrow c_j$ 和 $b_i \leftrightarrow b_j$。显然,在顺序表 $C = c_1c_2\cdots c_m$ 中查找符号序列 X 中的任意符号时,那么小概率符号一定要比大概率符号先查找到。

2.4　折半查找法

无论是顺序查找法,还是大概率优先查找法或小概率优先查找法,整个符号序列的编码速度都与符号的概率有关. 与这些方法不同的是,折半查找法的编码速度与符号的概率无关,只取决于符号本身的大小. 如果顺序表 $C = c_1c_2\cdots c_m$ 是一个有序顺序表,则可以用折半查找法对符号序列进行编码. 如果顺序表是一个升序序列,则对符号序列的编码过程如下。

(1) 令 $i = 1$,表示从符号 x_1 开始编码. 令 $Y = \varphi$,表示对应符号序列的初始编码为空。

(2) 令 $l = 1$ 和 $h = 1$. 此时 l 指示 x_1,而 h 指示 x_n.

(3) 折半. 令 $m = \dfrac{l + h}{2}$.

(4) 如果 $x_i > c_m$,则 $l = m + 1$; 否则如果 $x_i < c_m$,则 $h = m - 1$.

(5) 如果 $x_i = c_m$,则令 $Y = Y \cup \{b_m\}$.

在上述算法中,没有考虑符号编码失败的问题.

因此,必须保证此处的码字集是正确的。

上述算法中,每个符号的编码速度只与它在有序顺序表中的位置有关,而与它自身的概率无关. 也就是说,概率大的符号的编码速度有可能比概率小的符号的编码速度小。

2.5 二叉排序树法

前面介绍的 4 种编码方法查找的对象都是顺序表,而二查排序树查找的对象是二叉树. 对于二叉排序树中的任何一个节点 p,设其左孩子为 $p->lchild$ 和右孩子为 $p->rchild$,而节点信息为符号 $p->data$ 和码字 $p->b$,其中 $p->b$ 是符号 $p->data$ 对应的码字. 则由 $C=c_1c_2\cdots c_m$ 可以建立一个二叉排序树. 如果设该二叉排序树的根节点为 t,则符号序列 X 的编码过程描述如下。

(1) 令 $i=1$,表示第一个需要编码的符号是 x_1. 令 $Y=\varphi$,它表示编码序列的初始化。

(2) 令 $s=t$,它表示需要从二叉树的根节点开始查询。

(3) 如果 $s->data=x_i$,则 $Y=Y\cup s->b$ 并转步骤(6)。

(4) 如果 $s->data>x_i$,则令 $s=s->lchild$,转步骤(3)。

(5) 如果 $s->data<x_i$,则令 $s=s->rchild$,转步骤(3)。

(6) 令 $i=i+1$. 如果 $i\le m$,则转步骤(2);否则,编码过程结束。

3 编码算法分析

算法 2.1,算法 2..2 和算法 2.3 本质上都是顺序查找,不同的是他们的平均查找长度不同。要从顺序表 $C=c_1c_2\cdots c_m$ 中查找 c_i $(i=1,2,\cdots,n)$,则查找成功的比较次数为 $l_i=i$。所以,对符号序列 X 编码的平均查找长度为:

$$L=\sum_{i=1}^{n}p_il_i=\sum_{i=1}^{n}p_ii. \tag{1}$$

由于算法 2.2 中大概率符号的查找长度短而小概率符号的查找长度长,故用它编码的平均查找的长度短。相反,由于算法 2.3 中大概率符号的查找长度长而小概率符号的查找长度短,故用它编码的平均查找的长度比较长。如果设 L_1,L_2 和 L_3 分别表示算法 2.1,算法 2.2 和算法 2.3 的平均查找长度,则他们之间的大小关系如下:

$$L_2\le L_1\le L_3. \tag{2}$$

不等式(2) 说明对于同一符号的编码速度而言,这三种算法的平均查找长度是不一样的。其中算法 2.2 的平均查找长度最短,而算法 2.3 的平均查找长度最长。

与算法 2.1,算法 2.2 和算法 2.3 有本质不同,算法 2.4 和算法 2.5 不是顺序查找,而是跳跃查找。前三种算法的平均查找长度只与符号的概率有关,而与符号本身的大小无关。算法 2.4 和算法 2.5 的平均查找长度不仅与符号的概率有关,而且与符号自身的大小有关。如果用 l_i 表示查找到符号 c_i 的比较次数,则算法 2.4 和算法 2.5 的平均查找长度为

$$L=\sum_{i=1}^{n}p_il_i. \tag{3}$$

要想计算用算法 2.4 进行编码的平均查找长度,必须求出每一个符号 c_i 在有序顺序表中的比较次数 l_i。在有序顺序表中查找 c_i 的过程如下。

(1) 令 $l_i=0$。对查找 c_i 的比较次数进行初始化。

(2) 令 $l=1$ 和 $h=n$。为第 1 次折半做准备。

(3) 令 $m=\dfrac{l+h}{2}$ 和 $l_i=l_i+1$。

(4) 如果 $c_i>c_m$,则 $l=m+1$;否则如果 $c_i<c_m$,则 $h=m-1$;否则,查找结束。

(5) 转步骤 3 重复执行步骤(3) 和步骤(4)。

以上算法是在折半查找的基础上统计查找 c_i 的比较次数。

与算法 2.4 的统计查找的比较次数不同,算法 2.5 中每个符号的查找比较次数来自于二叉排序树的查找。在二叉排序树 t 中查找符号 c_i 的比较次数的统计过程如下。

(1) 令 $l_i=1$,表示查找 c_i 时至少需要比较 1 次。

(2) 如果 $c_i<t->data$,则 $t=t->lchild$;否则如果 $c_i>t->data$,则 $t=t->rchild$;否则,查找过程结束。

(3) 令 $l_i=l_i+1$ 并转步骤(2) 重复执行。

如果二叉排序树是一棵严格平衡二叉树,那么算法 2.5 的平均查找长度与算法 2.4 相等。否则,一般情况下,算法 2.4 的平均查找长度要小于算法 2.5。如果 L_4 和 L_5 分别表示算法 2.4 和算法 2.5 的平均查找长度,那么一定有下面的关系:

$$L_4\le L_5. \tag{4}$$

通过对以上 5 种算法的分析,要想提高符号序列的编码速度,只需在算法 2.2 和算法 2.4 之间选择,即最小的平均查找长度为:

$$L = \min\{L_2, L_4\}. \quad (5)$$

因此,从对同一符号序列的编码速度而言,有时算法 2.4 的速度最快,有时算法 2.2 的速度最快。当用算法 2.5 建立的二叉树是严格平衡二叉树时,有 $L_4 = L_5$.

4 算例

例 1　已知符号序列 X 中包含 a,b,c,d,e,f 六种符号,其对应的概率分别为{0.06,0.15,0.14,0.4,0.2,0.05}..分别用上面介绍的 5 种方法求对符号序列 X 进行编码的平均查找长度。

解　1) 如果用算法 2.1 求 X 平均查找长度,则 $C = abcdef$,而 $p = \{0.06,0.15,0.14,0.4,0.2,0.05\}$,所以算法 2.1 编码的平均查找长度为 $L_1 = 0.06 \times 1 + 0.15 \times 2 + 0.14 \times 3 + 0.4 \times 4 + 0.2 \times 5 + 0.05 \times 6 = 3.78$。

2) 如果用算法 2.2 求 X 平均查找长度,则 $C = debcaf$,而 $p = \{0.4,0.2,0.15,0.14,0.06,0.05\}$,所以算法 2.2 编码的平均查找长度为 $L_2 = 0.4 \times 1 + 0.2 \times 2 + 0.15 \times 3 + 0.14 \times 4 + 0.06 \times 5 + 0.05 \times 6 = 2.49$。

3) 如果用算法 2.3 求 X 平均查找长度,则 $C = facbed$,而 $p = \{0.05,0.06,0.14,0.15,0.2,0.4\}$,所以算法 2.3 编码的平均查找长度为 $L_3 = 0.05 \times 1 + 0.06 \times 2 + 0.14 \times 3 + 0.15 \times 4 + 0.2 \times 5 + 0.4 \times 6 = 4.59$。

4) 如果用算法 2.4 求 X 平均查找长度,则 $C = abcdef$,所以 $l_3 = 1$,$l_1 = l_5 = 2$ 和 $l_2 = l_4 = l_6 = 3$。故由算法 2.4 编码的平均查找长度为 $L_1 = 0.06 \times 2 + 0.15 \times 3 + 0.14 \times 1 + 0.4 \times 3 + 0.2 \times 2 + 0.05 \times 3 = 2.46$。

5) 如果用算法 2.5 求 X 平均查找长度,则 $C = abcdef$,所以编码的平均查找长度为 $L_1 = 0.06 \times 1 + 0.15 \times 2 + 0.14 \times 3 + 0.4 \times 4 + 0.2 \times 5 + 0.05 \times 6 = 3.78$。

通过上述计算的平均查找长度比较,用算法2.4编码的速度最优。

例 2　已知符号序列 X 中包含 a,b,c,d,e,f 六种符号,其对应的概率分别为{0.01,0.02,0.03,0.04,0.05,0.85}..求对符号序列 X 进行编码的最短平均查找长度。

解　1) 如果用算法 2.2 求平均查找长度,则 $C = fedcba$ 和 $p = \{0.85,0.05,0.04,0.0.03.002,0.01\}$. 因此,由算法 2.2 求得的平均查找长度 $L_2 = 0.85 \times 1 + 0.05 \times 2 + 0.04 \times 3 + 0.03 \times 4 + 0.02 \times 5 + 0.01 \times 6 = 1.44$.

2) 如果用算法 2.4 求 X 平均查找长度,则 $C = abcdef$ 和 $p = \{0.01,0.02,0.03,0.04,0.05,0.85\}$. 由折半查找有 $l_3 = 1$,$l_1 = l_5 = 2$ 和 $l_2 = l_4 = l_6 = 3$。故由算法 2.4 编码的平均查找长度为 $L_4 = (0.01 + 0.05) \times 2 + 0.03 \times 1 + (0.02 + 0.04 + 0.85) \times 3 = 2.88$.

3) 如果用算法 2.5 求 X 平均查找长度,则由 $C = abcdef$ 建立的二叉排序树有 $l_i = i$,$i = 1,2,\cdots,6$。故由算法 2.5 编码的平均查找长度为 $L_5 = 0.01 \times 1 + 0.02 \times 2 + 0.03 \times 3 + 0.04 \times 4 + 0.05 \times 5 + 0.85 \times 6 = 5.65$.

通过上述计算的平均查找长度比较,用算法2.2编码的速度最优。

例 3　已知符号序列 X 中包含 c,a,e,b,d,f 六种符号,其对应的概率分别为{0.4,0.2,0.14,0.15,0.05,0.06}..求对符号序列 X 进行编码的最短平均查找长度。

解　1) 如果用算法 2.2 求平均查找长度,则 $C = cabefd$ 和 $p = \{0.4,0.2,0.15,0.14,0.06,0.05\}$. 因此,由算法 2.2 求得的平均查找长度 $L_2 = 0.4 \times 1 + 0.2 \times 2 + 0.15 \times 3 + 0.14 \times 4 + 0.06 \times 5 + 0.05 \times 6 = 2.49$.

2) 如果用算法 2.4 求 X 平均查找长度,则 $C = abcdef$ 和 $p = \{0.2,0.15,0.4,0.05,.14,0.06\}$. 由折半查找有 $l_3 = 1$,$l_1 = l_5 = 2$ 和 $l_2 = l_4 = l_6 = 3$。故由算法 2.4 编码的平均查找长度为 $L_4 = 0.2 \times 2 + 0.15 \times 3 + 0.4 \times 1 + 0.05 \times 3 + 0.15 \times 2 + 0.06 \times 3 = 1.88.$

3) 如果用算法 2.5 求 X 平均查找长度,则由 $C = cabefd$ 建立的二叉排序树是一棵严格平衡二叉树,故有 $l_3 = 1$,$l_1 = l_5 = 2$ 和 $l_2 = l_4 = l_6 = 3$。故由算法 2.5 编码的平均查找长度为 $L_5 = 0.2 \times 2 + 0.15 \times 3 + 0.4 \times 1 + 0.05 \times 3 + 0.15 \times 2 + 0.06 \times 3 = 1.88.$

通过上述计算的平均查找长度比较,用算法2.4和算法 2.5 编码的速度最优。

5 结束语

在对符号序列的编码速度进行深入分析的基础上,设计了 5 种不同的编码算法. 通过对这些编码算法的平均查找长度的比较,发现大概率优先查找算法或折半查找算法的平均查找长度是最短的. 算例

表明,对于一个具体的符号序列进行编码,为了得到最快的编码速度,只需要从大概率优先查找和折半查找这两种算法中选择平均查找长度最短的编码算法即可.

参考文献:

[1] 叶芝慧,沈克勤. 信息论与编码 [M]. 北京: 电子工业出版社 ,2013.

[2] 程佳佳,熊志斌. 哈夫曼算法在数据压缩中的应用 [J] 科学技术与工程, 2010,2: 35 – 37.

[3] 邵军花,刘玉红,周东梅. 香农编码的优化算法研究 [J]. 兰州交通大学学报,2010,29(6): 110 – 113.

[4] 郭蕾,李东辉 ,缪志甫. 结合压缩感知理论的快速分形编码 [J]. 计算机工程与设计,2012,9,33(9): 3494 – 3497.

[5] 汪大勇,孙世新,杨浩淼. 一种基于残差系数的快速编码算法 [J]. 电子与信息学报,2010,11,32(11): 2541 – 2546.

(上接第 59 页)

于实现,在保证滤波精度的同时,也能有效的克服由于组合导航系统维数过高而导致粒子数目增加滤波计算量增大的缺点,更加适用于组合导航系统信息的实时融合。

参考文献:

[1] Casper E S. INS and GPSintegration [D]. Lyngby: Technical University of Denmark, 2006: 60-72.

[2] 鲍其莲,周媛媛. 基于 UKF 的 GPS/SINS 伪距(伪距率)组合导航系统设计 [J]. 中国惯性技术学报, 2008, 16(1): 78-82.

[3] 周翟和,刘建业,赖际舟,等. Rao-Blackwellized 粒子滤波在 SINS/GPS 深组合导航系统中的应用研究 [J]. 宇航学报, 2009(2): 515-520.

[4] 袁俊刚,范胜林,刘建业,等. 卡尔曼/粒子组合滤波在 GPS/INS 组合导航中的应用研究 [J]. 导航与控制, 2010,09(4): 37-40.

[5] 周翟和,刘建业,赖际舟,等. 混合高斯粒子滤波在组合导航中应用的计算量分析 [J]. 中国惯性技术学报, 2010, 18(5): 595-599.

[6] 熊剑,郭杭,熊智,等. GPS/INS 组合导航系统中的高斯粒子滤波混和算法 [J]. 中国惯性技术学报,2012,20(2): 225-229.

[7] Yang T, Mehta P G,Meyn S P. Feedback particle filter [J]. IEEE Trans. onAutomatic Control, 2013, 58(10): 2465-2480.

[8] 王惠南. GPS 导航原理与应用 [M]. 北京: 科学出版社, 2003.