[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/30 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200480006553.2

[43] 公开日 2006 年 4 月 12 日

[11] 公开号 CN 1759394A

[22] 申请日 2004.3.12
[21] 申请号 200480006553.2
[30] 优先权
[32] 2003. 3.13 [33] US [31] 10/386,462
[86] 国际申请 PCT/IN2004/000059 2004.3.12
[87] 国际公布 WO2004/081819 英 2004.9.23
[85] 进入国家阶段日期 2005.9.12
[71] 申请人 惠普开发有限公司
地址 美国德克萨斯州
[72] 发明人 S·K·N·V·库马
R·S·曼塔 C·S·雷乌尔
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 杨生平 刘 杰

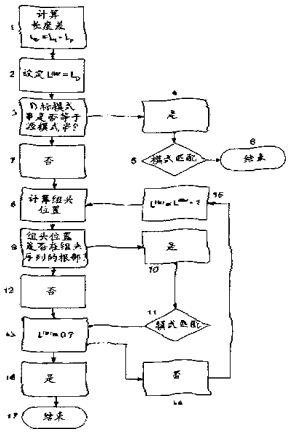
权利要求书 3 页 说明书 10 页 附图 5 页

[54] 发明名称

模式匹配的方法和系统

[57] 摘要

通过分别在可能源模式和目标模式的有序序列内只考虑源模式和目标模式的位置数可以获得模式匹配。 包含目标模式的源模式位置数形成许多组。在每个 i 组内的源模式数和组间间隔内的源模式数取决于目标模式在源模式内的位置。 目标模式的长度和字母集合中的元素数。 每个组也有位置数，它的位置在有序序列的组内。 将输入源模式的组位置数与从目标模式的位置数、目标模式的长度和字母集中的元素数得到的序列比较(9)。 如果组位置数是序列的成员(10)，那么源模式包含目标模式(11)。 可以在未编码数据或 Oh 算术编码数据上执行模式匹配。



1. 一种通过确定可能模式序列内的源模式位置是否是和包括目标模式的可能模式序列内的一个位置相关的位置来确定目标模式是否出现在由来自字母表集合的一个或多个字符组成的源模式内的方法。

2. 如权利要求 1 所述的方法，包括如下步骤：

i .根据字母表系统的库，通过将可能位置数序列划分为组来识别源模式的组头，其中组头是形成的每个组中的最高位置数；以及

ii .确定源模式的组头值是否是和包括目标模式的可能模式序列内的目标模式位置相关的组头值。

3. 如权利要求 2 所述的方法，其中通过将可能位置数的升序序列划分为具有对应于被升高到小 1 的行的幂字母表的库的组尺寸的组来为每行直接或间接确定源模式的组头值，其中所述行行对应源模式内的目标模式位置。

4. 如权利要求 3 所述的方法，其中组头位置用于确定组头值对目标模式是否有效，并且其中组头位置是在该行组间的组头的组的位置。

5. 如权利要求 2 任意一种方式所示的方法，其中根据以下算法计算源模式的组头位置：

$N=1+$ 商$((Sp-1)/b^{r-1})$

其中：

N=源模式组头位置数

Sp=源模式位置数

b=字母表库

r=行数

6. 如权利要求 5 所述的方法，其中确定在步骤 ii 中是否存在匹配，以下算法的结果必须是 0：

R=余数$((N-Tp)/b^{Tl})$

其中：

R=余数

Tp=目标模式位置数

Tl=目标模式的长度

7. 如权利要求 3 所述的方法，其中有许多行，并且将该方法连续的应用于

每一行。

8．如权利要求 3 所述的方法，其中将该方法同时应用于超过一个源模式。

9．如权利要求 3 所述的方法，其中搜索行的顺序取决于源模式的特性。

10．如权利要求 3 所述的方法，其中在行中寻找目标模式的概率被估计并用于确定搜索行的顺序。

11．如权利要求 3 所述的方法，其中在行中寻找目标模式的概率是已知的，并用于确定行的搜索顺序。

12．如权利要求 1 所述的方法，其中源模式是算术编码数据。

13．如权利要求 12 所述的方法，其中目标模式是算术编码数据。

14．如权利要求 12 所述的方法，其中源模式是是使用算术编码压缩的。

15．如权利要求 13 所述的方法，其中目标模式是使用算术编码压缩的。

16．如权利要求 1 所述的方法，其中目标模式具有 1 个以上的元素。

17．一种确定目标序列[X_1...Xm]是否在源序列[Y_1...Yq]内存在的方法，其中{X_1...Xm}和{Y_1...Yq}是有限有序集合{Z_1...Zo}的成员，包括以下步骤：

ⅰ.在库 o 中构建源子序列值的集合，{[V_1],...[V_1...V_k],...[V_1...Vn]}，其中 V_k=j:Y_k=Z_j；以及

ⅱ.确定集合的任何值是否对应目标值序列{p,...p+ko^m,...p+no^m}，其中 p 是库 o 中的数[D_1...D_n] 以便 D_k=h:X_k=Z_h。

18．如权利要求 17 所述的方法，其中确定集合的任何值是否对应步骤ⅱ中的序列，对集合的所有成员，以下算法的结果必须是 0：

R=余数((N-p)/o^m)

其中：

N=库 10 中的集合成员

19.如权利要求 17 所述的方法，其中源序列是图像数据。

20．如权利要求 17 所述的方法，其中源序列是视频数据。

21．如权利要求 17 所述的方法，其中源序列是生物医学数据。

22．一种数据处理装置，用于确定目标模式是否存在于由一个或多个字母表的字符组成的源模式中，包括：

ⅰ.第一存储器，用于存储目标模式；

ⅱ.第二存储器，用于存储源模式；

iii.处理装置，用于确定源模式在可能模式序列中的位置；

iv.处理装置，用于确定目标模式在可能模式序列中的位置；

v.处理装置，用于关联源位置和目标位置。

23．一种数据处理装置，用于确定目标序列是否存在于由一个或多个字母表的字符组成的源序列中：

i.存储目标序列位置的第一存储器，其中目标序列位置是目标序列在字母表字符的所有可能组合的词典中的位置；

ii.存储源序列位置的第二存储器，其中源序列位置是源序列在所述词典中的位置；

iii.计算源序列的子序列的位置集合的处理装置，其中子序列的位置是子序列在所述词典中的位置，并且其中子序列包括源序列的第一字符位置；

iv.确定定义序列在包括目标序列的词典中的所有位置的序列的处理装置；

v.用于关联集合和序列的处理装置。

24．如权利要求22所述的数据处理装置，其中源序列是图像数据。

25．如权利要求22所述的数据处理装置，其中源序列是视频数据。

26．如权利要求22所述的数据处理装置，其中源序列是生物医学数据。

27．如权利要求22所述的数据处理装置，其中处理装置是计算机系统。

28．一种用于按照权利要求1的方法操作的数据处理装置。

29．一种用于按照权利要求17的方法操作的数据处理装置。

30．一种执行如权利要求1方法的计算机程序。

31．一种执行如权利要求17方法的计算机程序。

32．包含如权利要求30所述的计算机程序的计算机可读介质。

33．包含如权利要求31所述的计算机程序的计算机可读介质。

34．一种包含如权利要求30所述的计算机程序的信号。

35．包含如权利要求31所述的计算机程序的计算机可读介质。

36．被编程用于按照权利要求1所述的方法操作的计算机。

37．被编程用于按照权利要求17所述的方法操作的计算机。

38．通过权利要求1所述的方法来产生的模式匹配信息。

39．通过权利要求17所述的方法来产生的模式匹配信息。

40．包含如权利要求38所述的模式匹配信息的信号。

41．包含如权利要求39所述的模式匹配信息的信号。

模式匹配的方法和系统

技术领域

本发明涉及模式匹配。模式匹配是在源模式中寻找一些或所有目标模式的出现的过程。压缩模式匹配是在不解压源模式的情况下，在压缩的源模式中寻找一些或所有目标模式的出现的过程。

背景技术

模式匹配是将分析规则应用于数据块以便识别数据块的特性。

最普遍的模式匹配问题是在较大序列元素[X1...Xn]（源模式）中寻找一些或所有序列元素[Y1...Ym]（目标模式）的出现的过程。元素来自于有限元素集—字母集。该集合可以是英语字母表，{0，1}，自然数等等。解决该问题的最常用算法是 Knuth-Morris-Pratt 算法，Boyer-Moore 算法和 Rabin-Karp 算法。

在公共领域中有许多模式匹配工具。这些中最著名的是文件内字符串查找族（grep family）。

模式匹配用于简单文本搜索，在图像数据、语音数据、视频数据、音频数据、生物-医学序列分析等中搜索数据。

数据压缩主要用于减少存储空间并加速数据传送。已知各种形式的压缩。特别的关注是算术编码压缩，对于该压缩，压缩模式匹配至今还没有成为可能。

算术编码起源于 70 年代和 80 年代（参见示例 US4，122，440）。算术编码用于多个领域，包括语音和医学图像压缩。

在压缩模式匹配中，在压缩领域执行模式匹配。简洁的输入，诸如文本串的压缩模式匹配可以表示为：

ac 是给出的压缩算法，ac(D)是 ac 的结果

压缩数据 D

输入： 压缩文本 ac(T)和压缩模式 ac(P)

输出： 当模式 P 出现时，在 T 中的所有或一些位置

Amir 和 Benson（“Efficient Two-dimensional Compressed Matching”，在第二次 IEEE 数据压缩会议中，279-288 页，1992 年 3 月）示出，与未压缩领域中的任何常规方法相比，压缩领域里的模式匹配的复杂性是降低的。

然而，认为在“算术代码”上不压缩数据而执行模式匹配还是不可能的。（“TR-COSC 07/01 Pattern Matching in Compressed Texts and Images”，Tim Bell 等，技术报告，坎特伯雷大学，访问 http://www.cosc.canterbury.ac.nz/research/reports/TechReps/2001/tr 0107.pdf 可得）

发明内容

本发明的目的是提供适用于匹配算术代码的模式匹配方法或至少为公众提供有用的选择。

根据本发明的第一方面，提供一种通过确定源模式在可能模式序列的位置是否是和在包括目标模式的可能模式序列中的一个位置相关的位置来确定目标模式是否在由一个或多个来自字母表集的字符组成的源模式中出现的方法，。

根据本发明的另一方面，提供一种确定目标序列[X1…Xm]是否在源序列[Y1…Yq]中出现的方法，其中{X1…Xm}和{Y1…Yq}是有限有序集合{Z1…Zo}的成员，包括步骤：

i.在库（base）o 中构建源子序列集合{[V_1],...[V_1...V_k],…[V_1…V_n]}，其中 V_k=j:Y_k=Z_j；以及

ii.确定集合的任意值是否对应目标值序列{p,..p+ko^m,…p+no^m}，其中 P 是库 o 中的数字[D_1,…D_n]，例如 D_k=h:X_k=Z_h。

根据本发明的另一方面，提供一种数据处理装置，用于确定目标模式是否出现在由一个或多个字母表字符组成的源模式中，包括：

i.第一存储器，用于存储目标模式；

ii.第二存储器，用于存储源模式；

iii.处理装置，用于确定源模式在可能模式序列中的位置；

iv.处理装置，用于确定目标模式在可能模式序列中的位置；

v.处理装置，用于关联源位置和目标位置；

根据本发明的另一方面，提供一种数据处理装置，用于确定目标序列是否出现在由一个或多个字母表字符组成的源序列中：

i.存储目标序列位置的第一存储器，其中目标序列位置是目标序列在字母表字符的所有可能组合的词典中的位置；

ii.存储源位置的第二存储器，其中源序列位置是源序列在词典中的位置；

iii.计算源序列的子序列的位置集合的处理装置，其中子序列的位置是子序

列在词典中的位置，并且其中子序列包括源序列的第一字符位置；

iv.确定定义序列在包括目标序列的词典中的所有位置的序列的处理装置；

v.关联集合与序列的处理装置

附图说明

现在将参考相应附图，通过示例详细描述本发明，其中：

附图 1 是包含目标模式‘1’的、长度为 4 的源模式串的位置图表，其中字母集是{0，1}；

附图 2 是包含目标模式‘b’、长度为 4 的源模式串的图表，其中字母集是{a,b,c,d,e}；

附图 3 是包含目标模式‘b’、长度为 4 的源模式串的位置数图表，其中字母集是{a,b,c,d,e}；

附图 4 是如附图 3 的图表，其中位置数被标度；以及

附图 5 是表示模式匹配算法的流程图。

具体实施方式

将和示例结合描述本发明，其中数据可以是压缩的或非压缩的。

假设源模式串 t 和目标模式串 p。这些串分别具有长度 L_t 和 L_p。并且我们需要 L_t L_p（注意，假设满足本条件，本发明可以适用于任意长度的串。）我们定义源模式串在有序序列的源模式串中的位置 P_t。例如，对于 L_t=4,并且字母集合是{a,b,c,d,e}，第一个 11 源模式串和他们在词汇顺序集的位置如表 1 所示：

源模式串	aaaa	aaab	aaac	aaad	aaae	aaba	aabb	aabc	aabd	aabe	aaca
P_t	1	2	3	4	5	6	7	8	9	10	11

表格 1

类似的，我们定义目标模式串的位置 P_p。对于 L_p=1,以及相同的字母集{a,b,c,d,e}，可能的目标模式串和他们在词汇顺序集合中的位置如表 2 所示

目标模式串	a	b	c	d	e
P_p	1	2	3	4	5

表格 2

源模式串可以包含0，一个或多个目标模式串的匹配。

附图1是为了便于描述的简单方案。在附图1中，我们开始于字母表集{0，1}以及源数据长度L_t为4。可能的源模式串是从0000到1111的所有二进制数。源模式串在数字有序集合中的位置也如附图所示。现在考虑模式匹配，目标模式为‘1’。在行1到4中，为在源模式串中的四个可能目标模式位置，表示满足模式匹配条件的源模式串的位置数。例如，在行4，目标模式‘1’占据源模式串的第一个位置。对二进制数1000到1111满足该条件，其具有从9到16的位置数。

更一般的说，注意在给出的行中，在目标模式出现处的源模式串的位置数形成一个或多个连续数目的组，组间有间断。每组中元素数取决于行，并因此取决于目标模式在源模式串中的位置。在行1，每个组包括单一元素，每组间的间隙为1。在行3，每组包括4个元素，每组间的间隙为4。

明显的是，行1的元素形成算术序列，其中开始元素b_0等于目标模式P_p的位置，而区别是被升高到目标模式的长度（也就是，目标模式中元素数目）L_p的幂的字母表集合中的元素数目N：

$$b_m=b_0+(m-1)N^{Lp} \qquad \text{等式（1）}$$

我们现在定义每组中的最高位置为组头。这样每行都有组头序列。任何行的组头序列被通过因子N（字母集中的元素数目）与邻行的组头序列相关。我们定义行1的组头序列为根组头序列。

在附图2到4中，我们呈现了第二方案。我们开始于字母表集{a,b,c,d,e}，以及源数据长度L_t为4。我们希望用目标模式‘b’来执行模式匹配，。然后，目标模式的位置数目P_p为2。附图2示出了包含目标模式‘b’的所有可能源模式串。虚线表示不是所有的源模式串已经显示在该区域中。附图3完全对应附图2，除了源模式串由他们的位置数表示。附图4对应于附图3，除了已添加位置数轴，以便位置数可以被大致地表示来标度。根据上面的等式（1），根组头序列可以表示为：

$$b_m=2+(m-1)5^1$$

$$=\{2,7,12,17\ldots\}$$

任何行的组头序列被通过因子N=5与相邻行的组头序列相关。

源数据长度为4的可能组头和长度$L_p=1$的所有可能目标模式如表3所示。

注意当目标模式是‘b’时（因此 P_p=2），只有具有组头位置{2,7,12,...}的组表示模式匹配，表格中的对应栏以阴影显示。

组头位置（序列）	1		3	4	5	6		8
行 1 组头序列	1		3	4	5	6		8
行 2 组头序列	5		15	20	25	30		40
行 3 组头序列	25		75	100	125	150		200
行 4 组头序列	125		375	500	625	（只有 625 可能源模式串）	-	-

表格 3

现在将参考 3 和 5 描述模式匹配算法，其中用括在一起的数字表示相关步骤。

该算法在源模式串中的每个可能目标模式位置搜索模式匹配，也就是一次一行。行数等于 1 加上源模式串与目标模式串之间的长度差，$L_t-L_p+1=L_D+1$，其中 $L_D=L_t-L_p$（1）。因此我们定义 L^{iter}（2），其以 1 递增或递减以用于一次移动一行。

我们计算源和目标模式串的位置数。如果 L_D 等于 0 而 P_t 等于 P_p。那么，目标模式串等同于源模式串（3）。如果这样（4），那么很清楚，我们有一个并只有一个模式匹配（5）并且算法终止（6）。否则（7），我们计算包括源模式串的组的组头位置（8）。然后，将该组头位置与根组头序列比较（9）。如果组头位置不是根组头序列的成员（12），那么处理进行到下一个步骤（13）。如果组头位置是根组头序列的成员（10），那么在对应行有模式匹配（11）。如果 L^{iter} 不等于 0（13，14），算法以递减 L^{iter} 循环剩余的行，检查每行中的模式匹配。一旦已检查所有行，L^{iter}=0（13，16），算法结束（17）。

很明显，该算法搜索源数据中的所有模式匹配。然而，类似算法可以简单的搜索单一模式匹配，发现模式匹配之后立即结束。在该情况下，涉及串的经验知识可以极大的加速该算法。搜索序列会取决于源模式串的特性。如果已知先验，向着源模式串的终点可能包含目标模式，我们应该开始假设在行 1 中包含目标模式，并向着行（L_D+1）一次渐进移动一行。另一方面，如果向着源模式串的开头可能包含目标模式，我们从行（L_D+1）渐进移动到行 1。另一类似算法可以在源模式串中的特定位置搜索模式匹配。此外，搜索顺序可以根据源

模式串的分析并按照每行的估计概率发生。可以理解的是根据源数据的特性使用搜索序列的范围。

如上所述的计算组头位置的步骤可以如以下实现。每个可能组的最低的成员位置 P_j 可以如下表示：

$$P_j=1+(j-1)N^{r-1}$$

其中 j={1,2,3…},N 是字母表中元素数目，而 r 是行数。

然后

$$(j-1)=(P_j-1)/N^{r-1}$$

以便，对于任意位置 P_t 的源模式串，寻找组头位置 n

$$n=1+\text{商}(P_t-1)/N^{r-1}$$

确定组头位置 n 是否是可以使用该等式获得的根组头序列的元素的步骤：

$$R=\text{余数}((n-P_p)/N^{Lp})$$

其中 P_p 是可能目标模式有序序列内的目标模式位置，而 L_p 是目标模式中元素数。

如果 R=0，那么组头位置是根组头序列的一个元素。

示例 1

执行上述步骤的算法在此以伪码表示。

N-字母表中的元素数。

T-源模式串。

P-目标模式串。

L_t-T 的长度。

L_p-P 的长度。

P_t-T 的位置数。

P_p-P 的位置数。

R_h-对特定串 ac 输出的范围的高值。

R_l-对特定串 ac 输出的范围的低值。

```
function  Pattern_match
 {                                   /*临时变量*/
 int L_iter,z,d, n, m, temp;
 L_d=L_t-L_p;
```

$L_{iter}=L_d$;

/*从下面给出的等式（2）计算 P_t 和 P_p*/

if (L_d=0) and (P_t=P_p) /*等长串*/

Msg（“Both match”）;

Exit Function;

/*通过 $a_n=a_0+(n-1)$ 和 $b_m=b_0+(m-1)z$ 给出两个算术序列*/

Loop for L_d+1 times

{

/*在组算术序列中寻找源数据组头位置（n）*/

$a_n=P_t$; a_0=1 d=power(N,L_{iter})

temp=quotient((a_n-a_0)/d);

n=temp+1;

/*检查‘n’是否落入根组头序列*/

b_m=n; $b_0=P_p$; z=power(N,L_p);

r=remainder((b_m-b_0)/z);

if (r=0)

Msg(“pattern match success at position:”((L_d+1)-L_{iter}));

$L_{iter}=L_{iter}$-1;

} /*循环结束*/

}

我们现在将使用如附图 2 到 4 中所示的示例讨论该算法。

考虑源模式串输入，源数据的位置 P_t=158。

首先，我们寻找行 4 中的模式匹配。

$L^{iter}=L_t-L_p=4-1=3$

/*寻找组算术序列中的源数据组头位置(n)*/

$a_n=P_t=158$ a_0=1 d=power(N,L_{iter})=5^3=125

temp =quotient((a_n-a_0)/d)=quotient((158－1)/125)=1

n=temp+1=2

/*检查‘n’是否落入根组头序列*/

b_m=n=2; $b_0=P_p$=2; z=power(N,L_p)=5;

r=remainder($(b_m-b_0)/z$)=remainder((2－2)/5)=0;

r=0 因此，在位置 1 模式匹配

接下来我们在行 3 寻找模式匹配。如果

$L^{iter}=L^{iter}-1=2$

/*在组算术序列中寻找源数据组头位置(n)*/

$a_n=P_t=158$ $a_0=1$ $d=power(N,L_{iter})=5^2=25$

temp=quotient($(a_n-a_0)/d$)=quotient((158－1)/25)=6;

n=temp+1=7;

/*检查‘n’是否落入根组头序列*/

$b_m=n=7$; $b_0=P_p=2$; $z=power(N,L_p)=5$;

r=remainder($(b_m-b_0)/z$)=remainder((7－2)/5)=0;

r=0 因此在位置 2 模式匹配

接下来我们寻找行 2 的模式匹配

$L^{iter}=L^{iter}-1=1$

/*寻找组算术序列中源数据的组头位置(n)*/

$a_n=P_t=158$ $a_0=1$ $d=power(N,L_{iter})=5^1=5$

temp=quotient($(a_n-a_0)/d$)=quotient((158－1)/5)=31;

n=temp+1=32;

/*检查‘n’是否落入根组头序列*/

$b_m=n=32$; $b_0=P_p=2$; $z=power(N,L_p)=5$;

r=remainder($(b_m-b_0)/z$)=remainder((32-2)/5)=0;

r=0 因此在位置 3 的模式匹配

最后，我们寻找行 1 中的模式匹配

$L^{iter}=L^{iter}-1=0$

/*在组算术序列中寻找源数据的组头位置(n)*/

$a_n=P_t=158$ $a_0=1$ $d=power(N,L_{iter})=5^0=1$

temp=quotient($(a_n-a_0)/d$)=quotient((158－1)/1)=157;

n=temp+1=158;

/*检查‘n’是否落入根组头序列*/

$b_m=n=158$; $b_0=P_p=2$; $z=power(N,L_p)=5$;

$r=remainder((b_m-b_0)/z)=remainder((158-2)/5)=1;$

r 0 因此在位置 4 不是模式匹配

其中目标模式由可以使用相同算法的多个字符组成。

在算术代码中，串由数字线上的间隔表示。字母表符号的概率决定间隔的大小。在等概率的情况下，计算串的位置数是简单的事情：

$$P=R_h/(R_h-R_l) \qquad \text{等式(2)}$$

其中 R_h 是间隔的最大值，而 R_l 是间隔的最小值。

在字母表的不同元素具有不同概率的情况下，虽然不象在等概率情况中一样直接，但仍然可以确定位置数目。我们考虑字母表集合每个元素的概率和源模式的长度来建立对应感兴确的间隔的位置数。因此，当操作算术上压缩的数据时，计算编码间隔的位置数。该位置数与目标模式位置数一起成为上述算法的输入。

示例 2

源序列可以被表示为具有源序列长度的字母表元素的所有可能组合一源词典的的一个列表中的位置。当以库 n（n 是字母表中的元素数）表示时，，源序列本身实际上是以库 n 表示数字的字母表的元素的位置。

可以在具有子序列长度的字母表元素的所有可能组合的词典内计算源序列的所有子序列的位置集，该位置集合包括源序列的第一字符位置。该集合包括以下元素：对于库 n 中的源序列的位置长度，库 n 中的源序列的第一数字；第一数字和第二数字；第一数字和第二和第三，等。

在源词典内，包含目标序列的“词”是 n 的分开的目标序列长度的幂。使用包含目标序列的第一个“词”的位置，从而可以确定一系列包含目标序列的“词”的位置。

需要在相同库中提供集合和序列，并然后可以运行相关过程以观察任意可能目标序列位置是否等同于源子序列位置。

如果集合是从最小元素向最大元素排序的，也与目标序列匹配的集合中的元素位置描述了目标序列最右边元素的源序列的位置。

当然，通过使用余数等式，源子序列位置的集合再次与 O(n)中的可能目标位置的序列匹配。

这样，会确定目标序列$[X_1...X_m]$是否出现在源序列$[Y_1...Y_q]$内，其中

$[X_1 \ldots X_m]$和$[Y_1 \ldots Y_q]$是有限有序集合$\{Z_1 \ldots Z_o\}$的成员：

i.构造库 o 中源子序列值的集合$\{[V_1],\ldots[V_1 \ldots V_k],\ldots[V_1 \ldots V_n]\}$，其中$V_k$=j:$Y_k=Z_j$；以及

ii.确定集合的任意值是否对应目标值序列，$\{p,\ldots p+ko^m,\ldots p+no^m\}$，其中 p 是库 o 中的数$[D_1 \ldots Dn]$，例如，$D_k$=h:$X_k=Z_h$。

为了确定集合的任意值是否对应步骤 ii 中的序列，以下算法的结果对于该集合的值必须为 0：

R=余数$((N-p)/o^m)$

其中：

N=库 10 内集合的数

可以使用大范围的标准数据处理设备或专用图像音频处理或序列设备实施本方法。例如，在 MICROSOFT WINDOWS™ 环境下工作的个人计算机或 UNIX 操作系统下操作的服务器机上使用本方法。

可以理解的是，可以在设备上执行的软件内或专用硬件内实施本方法。

熟悉本领域技术的人员会进一步理解，可以在多个计算机或服务器上分部分的实施本方法。

应该相信，本发明表示了用于不解压缩数据的算术压缩源数据的模式匹配的第一方法。本发明也可以使用压缩目标模式串操作。本发明可以在算术解码站被极好地应用，其中解码器能一般检查与现有数据串的部分或全部匹配。

也可以使用本发明寻找所有模式匹配，一次模式匹配或在源模式串中特定位置的模式匹配。本发明也可以并行执行，因为在一个特定间隔上的搜索可以被独立于在其它间隔上的搜索来执行。也就是，本发明可以同时应用于多个源串。本发明也可以在分布式系统中操作，例如通信网络上的搜索引擎。

虽然已通过描述其实施例来描述本发明，并且虽然实施例已经被详细描述，申请人的目的不是限制或以任何方式限制附属的权利要求到此细节。其他优点或修改对本领域技术人员式明显的。因此，本发明在更广的方面不局限于表示和描述的特定细节，描述的装置和方法，以及示例性示例。因此，可以在不背离申请的基本创造性概念的精神或范围下进行一些修改。

目标模式 “1” 源模式串*** 目标模式的位置数 = 2

P_t	源数据串	行4 1***	行3 *1**	行2 **1*	行1 ***1
1	0000				
2	0001				2
3	0010			3	
4	0011			4	4
5	0100		5		
6	0101		6		6
7	0110		7	7	
8	0111		8	8	8
9	1000	9			
10	1001	10			10
11	1010	11		11	
12	1011	12		12	12
13	1100	13	13		
14	1101	14	14		14
15	1110	15	15	15	
16	1111	16	16	16	16

图 1

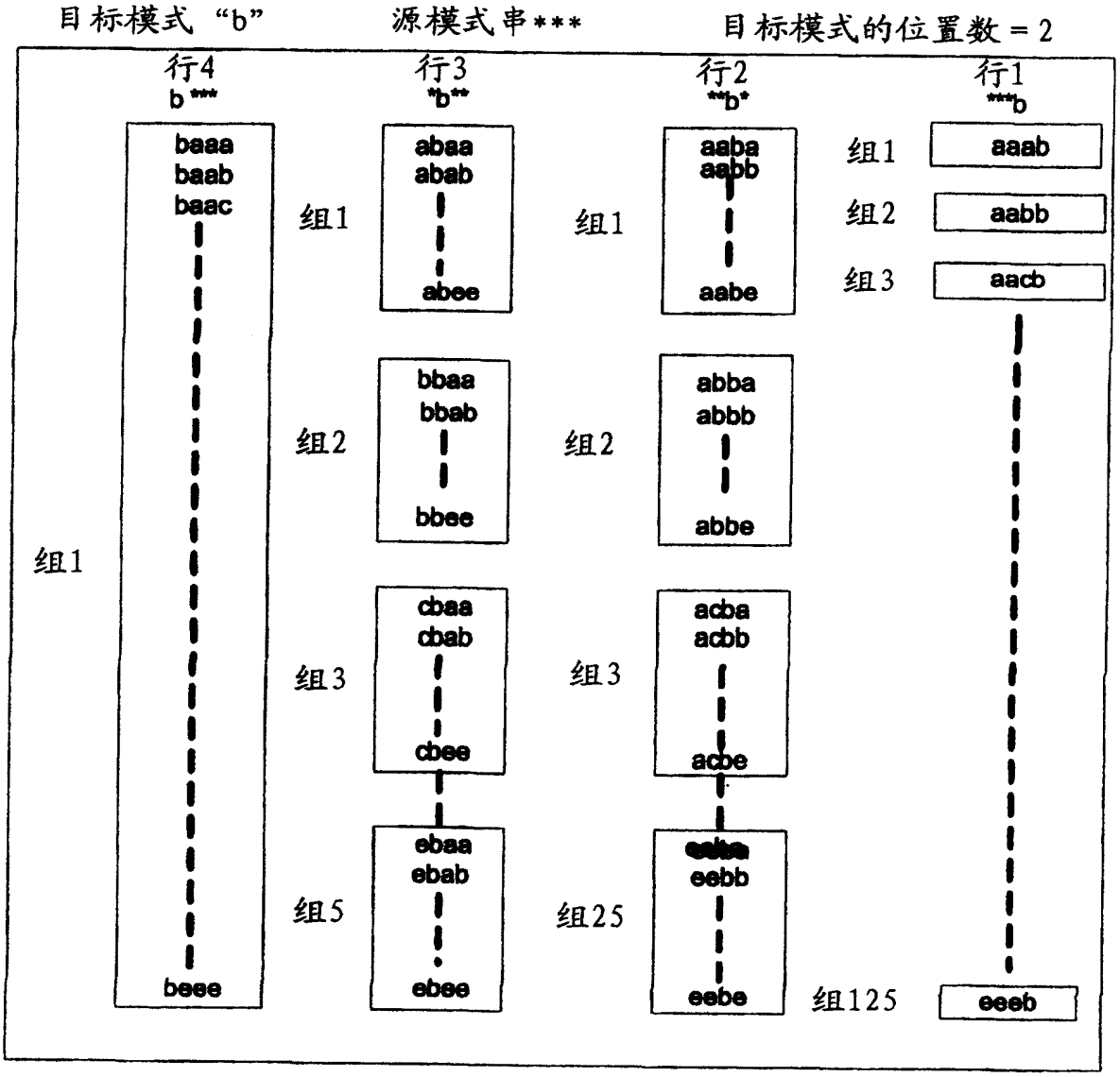

目标模式 “b”
源模式串***
目标模式的位置数 = 2
行4
b***
行3
*b**
行2
**b*
行1
***b
组1
baaa
baab
baac
beee
组1
abaa
abab
abee
组2
bbaa
bbab
bbee
组3
cbaa
cbab
cbee
组5
ebaa
ebab
ebee
组1
aaba
aabb
aabe
组2
abba
abbb
abbe
组3
acba
acbb
acbe
组25
eebb
eebe
组1
aaab
组2
aabb
组3
aacb
组125
eeeb

图 2

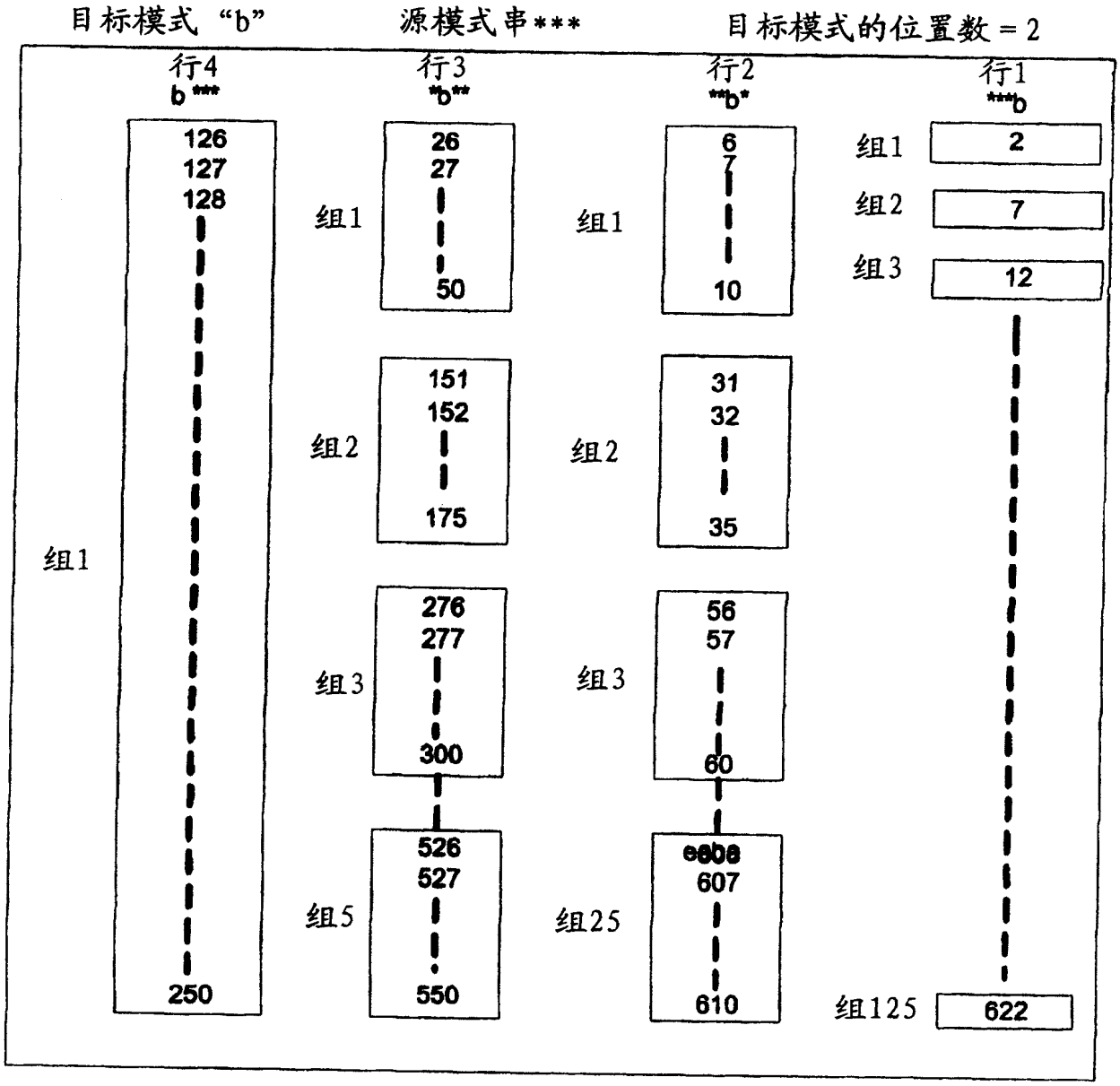

目标模式 “b”
源模式串***
目标模式的位置数 = 2
行4
b ***
行3
*b**
行2
**b*
行1
***b
组1
126
127
128
250
组1
26
27
50
组2
151
152
175
组3
276
277
300
组5
526
527
550
组1
6
7
10
组2
31
32
35
组3
56
57
60
组25
607
610
组1
2
组2
7
组3
12
组125
622

图 3

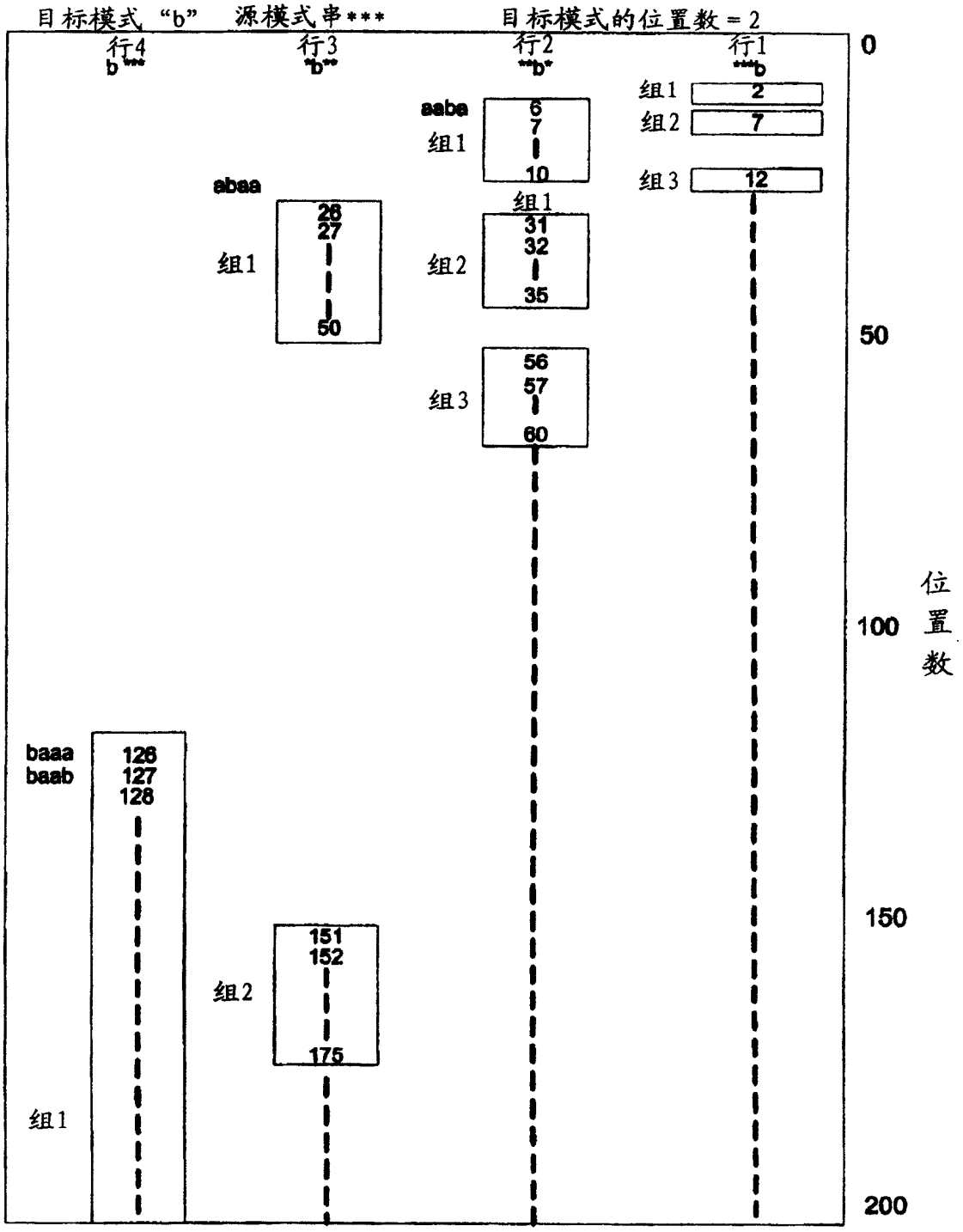

目标模式 "b" 源模式串***
目标模式的位置数 = 2
行4
b***
行3
*b**
行2
**b*
行1
***b
组1
2
组2
7
组3
12
aaba
组1
6
7
10
abaa
组1
26
27
50
组1
31
32
35
组2
56
57
60
组3
baaa
baab
126
127
128
组1
151
152
175
组2
0
50
100
150
200
位置数

图 4

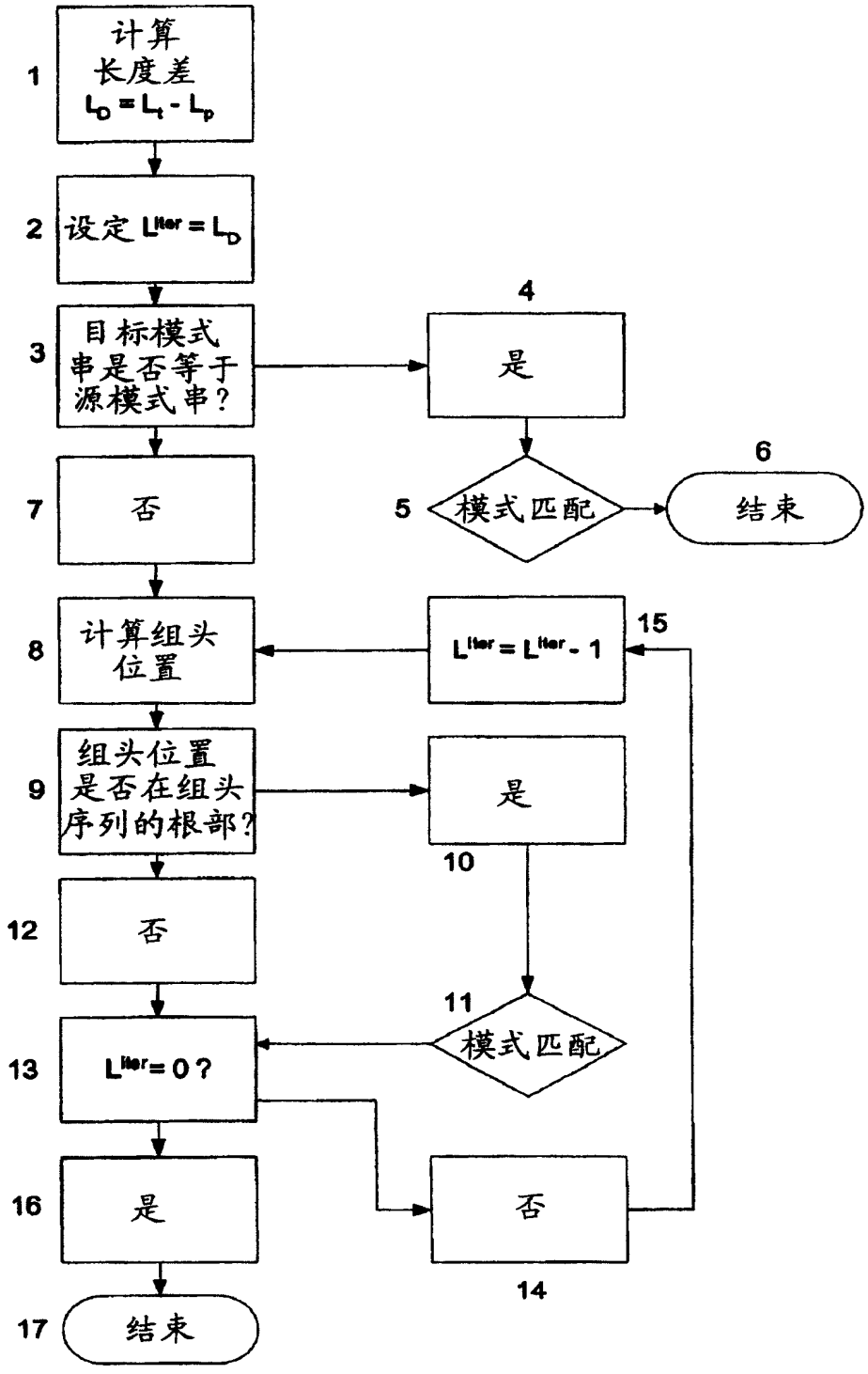

1
计算
长度差
$L_D = L_t - L_p$
2
设定 $L^{iter} = L_D$
3
目标模式串是否等于源模式串？
4
是
5
模式匹配
6
结束
7
否
8
计算组头位置
9
组头位置是否在组头序列的根部？
10
是
11
模式匹配
12
否
13
$L^{iter} = 0$？
14
否
15
$L^{iter} = L^{iter} - 1$
16
是
17
结束

图 5