(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101484895 B
(45)授权公告日 2011.12.28

(21)申请号 200780025563.4
(22)申请日 2007.06.15
(30)优先权数据
187827/2006 2006.07.07 JP
(85)PCT申请进入国家阶段日
2009.01.06
(86)PCT申请的申请数据
PCT/JP2007/000639 2007.06.15
(87)PCT申请的公布数据
WO2008/004335 JA 2008.01.10
(73)专利权人 新叶股份有限公司
地址 日本千叶县
(72)发明人 新庄敏男
(74)专利代理机构 北京三友知识产权代理有限公司 11127
代理人 黄纶伟
(51)Int.Cl.
G06F 17/30(2006.01)
(56)对比文件
CN 1050630 A,1991.04.10,
CN 1432943 A,2003.07.30,
JP 2001357070 A,2001.12.26,
Kikuta M..Patricia Tree.《Journal of Japanese Society of Artificial Intelligence》.1996,第11卷(第2期),337-339.
审查员 刘琳

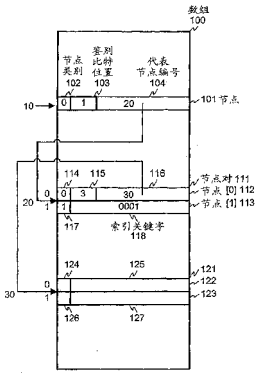
权利要求书 5 页 说明书 18 页 附图 17 页

(54)发明名称

比特序列检索装置以及检索方法

(57)摘要

本发明提供一种具有如下的数据结构的比特序列检索方法,该数据结构是检索对象的比特序列的数据结构,所需要的存储容量小,检索速度高,数据维护容易,该比特序列检索方法具有配对节点树,该树由根节点、以及配置在相邻的存储区域内的分支节点和叶节点或分支节点之间或叶节点之间的节点对构成,分支节点包含表示检索关键字的鉴别比特位置和链接目的地的节点对中的一个节点的位置的信息,叶节点包含由检索对象的比特序列构成的索引关键字,在从根节点到分支节点中,根据分支节点内包含的鉴别比特位置的检索关键字的比特值依次重复进行链接到链接目的地的节点对中的任一节点的动作直到到达叶节点,来执行基于检索关键字的索引关键字的检索。



CN 101484895 B

1. 一种比特序列检索装置，该比特序列检索装置根据存储有由作为检索对象的比特序列构成的索引关键字的树的数据结构，通过由比特序列构成的检索关键字来检索所述索引关键字，其特征在于，

该比特序列检索装置具有：

配对节点树，其具有作为所述树的起点的根节点、以及具有配置在相邻存储区域中的两个节点的作为树的构成要素的节点对，所述节点具有存储表示该节点是分支节点还是叶节点的节点类别的区域，所述分支节点除了包含所述节点类别，还包含存储所述检索关键字的鉴别比特位置的区域和存储表示链接目的地节点对中的一个节点的位置的信息的区域，所述叶节点除了包含所述节点类别，还包含存储由所述检索对象比特序列构成的索引关键字的区域，

根节点读取单元，其取得表示所述根节点的位置的信息，并根据该取得的表示根节点的位置的信息来读取根节点；

节点类别判定单元，其从所述节点的存储节点类别的区域中读取相应的节点类别，并判定该节点类别是表示所述叶节点还是表示所述分支节点；

索引关键字读取单元，其从所述叶节点的存储索引关键字的区域中读取相应的索引关键字；以及

链接单元，其从所述分支节点中取出鉴别比特位置，并从所述检索关键字中取得所述取出的鉴别比特位置的比特值，从所述分支节点中取出表示链接目的地节点对中的代表节点的位置的信息，将该取得的鉴别比特位置的比特值与所述取出的表示链接目的地节点对中的代表节点的位置的信息相加，来求取表示链接目的地节点的位置的信息，从由该求取出的表示链接目的地节点的位置的信息所表示的存储区域中，读取配置在该存储区域的节点作为链接目的地节点，

由所述节点类别判定单元来判定由所述根节点读取单元读取出的根节点的节点类别，如果该节点类别表示叶节点，则通过所述索引关键字读取单元从该叶节点读取索引关键字；如果该节点类别表示分支节点，则通过所述链接单元读取所述链接目的地节点，并由所述节点类别判定单元来判定该读取出的链接目的地节点的节点类别，反复进行以上操作，直到该节点类别表示叶节点为止，通过所述索引关键字读取单元从该叶节点读取索引关键字，

对通过所述索引关键字读取单元读取出的索引关键字与所述检索关键字进行比较，如果一致则检索成功，如果不一致则检索失败。

2. 根据权利要求 1 所述的比特序列检索装置，其特征在于，所述配对节点树被存储在数组内，表示所述链接目的地节点对中的一个节点的位置的信息是所述数组的存储有该节点的数组要素的数组编号。

3. 一种索引关键字插入方法，在该方法中比特序列检索装置根据存储有由作为检索对象的比特序列构成的索引关键字的树的数据结构，通过由比特序列构成的检索关键字来检索所述索引关键字，该比特序列检索装置在该树中存储由所希望的比特序列构成的插入关键字作为索引关键字，其特征在于，

所述树是配对节点树，是存储在数组中的树，其具有作为该树的起点的根节点、以及具有配置在所述数组的相邻的数组要素中的两个节点的、作为树的构成要素的节点对，所述

节点具有存储表示该节点是分支节点还是叶节点的节点类别的区域，所述分支节点除了包含所述节点类别，还包含存储所述检索关键字的鉴别比特位置的区域和存储所述数组的对链接目的地节点对中的一个节点进行存储的数组要素的数组编号的区域，所述叶节点除了包含所述节点类别，还包含存储由所述检索对象比特序列构成的索引关键字的区域，

该比特序列检索装置具有：

根节点读取单元，其取得所述数组的存储有所述根节点的数组要素的数组编号，并从该取得的数组编号的数组要素中读取根节点；

节点类别判定单元，其读取所述节点的节点类别，并判定该节点类别是表示所述叶节点还是表示所述分支节点；

索引关键字读取单元，其从所述叶节点的存储所述索引关键字的区域中读取相应的索引关键字；以及

链接单元，其从所述分支节点中取出鉴别比特位置，并从所述检索关键字中取得所述取出的鉴别比特位置的比特值，从所述分支节点中取出表示链接目的地节点对中的代表节点的位置的信息，将该取得的鉴别比特位置的比特值与所述取出的表示链接目的地节点对中的代表节点的位置的信息相加，来求取表示链接目的地节点的位置的信息，从由该求取出的表示链接目的地节点的位置的信息所表示的存储区域中，读取配置在该存储区域的节点作为链接目的地节点，

该索引关键字插入方法将所述插入关键字作为所述检索关键字，具有以下步骤：

检索步骤，由所述根节点读取单元读取根节点，由所述节点类别判定单元来判定该读取出的根节点的节点类别，如果该节点类别表示叶节点，则通过所述索引关键字读取单元从该叶节点读取索引关键字；如果该节点类别表示分支节点，则通过所述链接单元读取所述链接目的地节点，并通过所述节点类别判定单元来判定该读取出的链接目的地节点的节点类别，反复上述处理，直到该节点类别表示叶节点为止，通过所述索引关键字读取单元从该叶节点读取索引关键字，同时在堆栈中依次存储对直到该叶节点为止的链接路径中的分支节点和该叶节点进行存储的数组要素的数组编号；

比较步骤，在通过所述检索步骤读取出的索引关键字与所述插入关键字之间进行大小比较和比特序列比较；

空节点对取得步骤，从所述数组中取得存储节点对的空数组要素组，并取得该组的一个数组要素的数组编号；

叶节点存储位置确定步骤，通过所述比较步骤中的所述大小比较，来确定把包含插入关键字的叶节点存储在通过所述空节点对取得步骤取得的空数组要素组的哪一个空数组要素中；

节点对插入位置确定步骤，根据在所述比较步骤的比特序列比较中成为不同比特值的最初比特位置与存储在所述堆栈中的数组编号的数组要素所存储的分支节点的鉴别比特位置之间的相对位置关系，读取存储在该堆栈中的数组编号，将存储在该数组编号的数组要素中的节点作为存储在通过所述空节点对取得步骤取得的空数组要素组中的节点对的链接源，来确定该节点对的插入位置；

插入节点对生成步骤，通过下述方式来生成插入节点对：在要配置到通过所述叶节点存储位置确定步骤确定的空数组要素中的叶节点中、在存储节点类别的区域中写入表示是

叶节点的节点类别，在存储索引关键字的区域中写入所述插入关键字，并读出通过所述节点对插入位置确定步骤从所述堆栈读取出的数组编号的数组要素中所存储的节点内容，写入到另一个空数组要素中；

分支节点生成步骤，将通过所述节点对插入位置确定步骤从所述堆栈读取出的数组编号的数组要素中所存储的节点作为分支节点，在其存储节点类别的区域中写入表示是分支节点的节点类别，在存储鉴别比特位置的区域中写入在所述比较步骤的比特序列比较中成为不同比特值的最初比特位置，在存储所述数组的对链接目的地节点对中的一个节点进行存储的数组要素的数组编号的区域中写入通过所述空节点对取得步骤取得的数组编号。

4. 一种配对节点树生成方法，在该方法中比特序列检索装置生成树，该比特序列检索装置根据存储有由作为检索对象的比特序列构成的索引关键字的树的数据结构，通过由比特序列构成的检索关键字来检索所述索引关键字，其特征在于，

所述树是权利要求 3 所述的配对节点树，

所述比特序列检索装置是权利要求 3 所述的比特序列检索装置，

从存储在所述配对节点树中的索引关键字的集合中取出一个索引关键字，并生成将包含该索引关键字的叶节点作为根节点的配对节点树，

再从所述索引关键字的集合中依次取出索引关键字，根据权利要求 3 所述的索引关键字插入方法将该索引关键字插入到所述配对节点树中，由此生成配对节点树。

5. 一种索引关键字插入方法，在该方法中比特序列检索装置删除索引关键字，该比特序列检索装置根据存储有由作为检索对象的比特序列构成的索引关键字的树的数据结构，通过由比特序列构成的检索关键字来检索所述索引关键字，该比特序列检索装置删除存储在所述树中的、与由所希望的比特序列构成的删除关键字相同的索引关键字，其特征在于，

所述树是存储在数组中的配对节点树，其具有作为所述树的起点的根节点、以及具有配置在所述数组的相邻的数组要素中的两个节点、作为树的构成要素的节点对，所述节点具有存储表示该节点是分支节点还是叶节点的节点类别的区域，所述分支节点除了包含所述节点类别，还包含存储所述检索关键字的鉴别比特位置的区域和存储所述数组的对链接目的地节点对中的一个节点进行存储的数组要素的数组编号的区域，所述叶节点除了包含所述节点类别，还包含存储由所述检索对象比特序列构成的索引关键字的区域，

该比特序列检索装置具有：

根节点读取单元，其取得所述数组的存储所述根节点的数组要素的数组编号，并从该取得的数组编号的数组要素中读取根节点；

节点类别判定单元，其读取所述节点的节点类别，并判定该节点类别是表示所述叶节点还是表示所述分支节点；

索引关键字读取单元，其从所述叶节点的存储所述索引关键字的区域中读取相应的索引关键字；以及

链接单元，其从所述分支节点中取出鉴别比特位置，并从所述检索关键字中取得所述取出的鉴别比特位置的比特值，从所述分支节点中取出表示链接目的地节点对中的代表节点的位置的信息，将该取得的鉴别比特位置的比特值与所述取出的表示链接目的地节点对中的代表节点的位置的信息相加，来求取表示链接目的地节点的位置的信息，从由该求取出的表示链接目的地节点的位置的信息所表示的存储区域中，读取配置在该存储区域的节

点作为链接目的地节点，

该索引关键字删除方法将所述删除关键字作为所述检索关键字，具有以下步骤：

检索步骤，由所述根节点读取单元读取根节点，由所述节点类别判定单元来判定该读取出的根节点的节点类别，如果该节点类别表示叶节点，则通过所述索引关键字读取单元从该叶节点读取索引关键字；如果该节点类别表示分支节点，则通过所述链接单元读取所述链接目的地节点，并由所述节点类别判定单元来判定该读取出的链接目的地节点的节点类别，反复进行上述处理，直到该节点类别表示叶节点为止，通过所述索引关键字读取单元从该叶节点读取索引关键字，同时在堆栈中依次存储对直到该叶节点为止的链接路径中的分支节点和该叶节点进行存储的数组要素的数组编号；

当通过所述检索步骤读取出的索引关键字与所述删除关键字相同时，该索引关键字删除方法还具有以下步骤：

节点读取步骤，读取存储了与保持该索引关键字的所述叶节点一起构成同一节点对的节点的数组要素的内容；

写入步骤，从该堆栈中读取存储在所述堆栈中的、比存储所述叶节点的数组要素的数组编号前一个的数组编号，在该数组编号的数组要素中写入通过所述节点读取步骤读取出的数组要素的内容；以及

节点对删除步骤，释放存储有所述节点对的数组要素的组。

6. 一种比特序列检索方法，该方法中，比特序列检索装置根据存储有由作为检索对象的比特序列构成的索引关键字的树的数据结构，通过由比特序列构成的检索关键字来检索所述索引关键字，其特征在于，

所述树是配对节点树，其具有作为该树的起点的根节点、以及具有配置在相邻的存储区域内的两个节点、作为树的构成要素的节点对，所述节点具有存储表示该节点是分支节点还是叶节点的节点类别的区域，所述分支节点除了包含所述节点类别，还包含存储所述检索关键字的鉴别比特位置的区域和存储表示链接目的地节点对中的一个节点的位置的信息的区域，所述叶节点除了包含所述节点类别，还包含存储由所述检索对象比特序列构成的索引关键字的区域，

该比特序列检索方法包含以下步骤：

根节点读取步骤，取得表示所述根节点的位置的信息，并根据该取得的表示根节点的位置的信息来读取根节点；

节点类别判定步骤，从所述节点的存储节点类别的区域中读取相应的节点类别，并判定该节点类别是表示所述叶节点还是表示所述分支节点；

索引关键字读取步骤，从所述叶节点的存储所述索引关键字的区域中读取相应的索引关键字；以及

链接步骤，从所述分支节点的存储鉴别比特位置的区域和存储表示链接目的地节点对中的一个节点的位置的信息的区域中，分别读取相应的鉴别比特位置和表示链接目的地节点对中的一个节点的位置的信息，通过所述检索关键字的该读取出的鉴别比特位置的比特值和所述读取出的表示链接目的地节点对中的一个节点的位置的信息之间的运算，来求取表示节点位置的信息，从由该求取出的表示节点位置的信息所表示的存储区域中，读取配置在该存储区域中的节点，作为链接目的地节点，

通过所述节点类别判定步骤来判定通过所述根节点读取步骤读取出的根节点的节点类别，如果该节点类别表示叶节点，则通过所述索引关键字读取步骤从该叶节点读取索引关键字；如果该节点类别表示分支节点，则通过所述链接步骤读取所述链接目的地节点，并通过所述节点类别判定步骤来判定该读取出的链接目的地节点的节点类别，反复进行上述处理，直到该节点类别表示叶节点为止，通过所述索引关键字读取步骤从该叶节点读取索引关键字，

对通过所述索引关键字读取步骤读取出的索引关键字与所述检索关键字进行比较，如果一致则检索成功，如果不一致则检索失败。

7. 根据权利要求 6 所述的比特序列检索方法，其特征在于，所述配对节点树被存储在数组内，表示所述链接目的地节点对中的一个节点的位置的信息是存储有该节点的所述数组的数组要素的数组编号。

比特序列检索装置以及检索方法

技术领域

[0001] 本发明涉及从比特序列的集合中检索期望的比特序列的检索装置，特别是涉及对存储比特序列的数据结构进行设计来实现检索速度等的提高的技术领域。

背景技术

[0002] 近年，社会的信息化不断发展，大规模的数据库在各处利用起来。为了从这种大规模的数据库中检索记录，通常是将与存储有各记录的地址相对应的记录内的项目用作索引关键字来进行检索，检索出期望的记录。并且，全文检索中的字符串也能视为文件的索引关键字。

[0003] 而且，由于这些索引关键字利用比特序列来表现，因而数据库的检索可归结于比特序列的检索。

[0004] 为了高速进行上述比特序列的检索，以往做法是，对存储比特序列的数据结构进行种种设计。作为这种设计之一，公知的是 Patricia（帕特里希亚）树这样的树结构。

[0005] 图 17 示出在上述现有的检索处理中使用的 Patricia 树的一例。Patricia 树的节点构成为包含索引关键字、检索关键字的检查比特位置、以及左右的链接指针。尽管未作明示，然而当然在节点内包含有用于对与索引关键字对应的记录进行存取的信息。

[0006] 在图 17 的例子中，保持索引关键字“100010”的节点 1750a 为根节点，其检查比特位置是 0。节点 1750a 的左链接 1740a 与节点 1750b 连接，右链接 1741a 与节点 1750f 连接。

[0007] 节点 1750b 保持的索引关键字是“010011”，检查比特位置 1730b 是 1。节点 1750b 的左链接 1740b 与节点 1750c 连接，右链接 1741b 与节点 1750d 连接。节点 1750c 保持的索引关键字是“000111”，检查比特位置是 3。节点 1750d 保持的索引关键字是“011010”，检查比特位置是 2。

[0008] 从节点 1750c 用实线连接的部分表示节点 1750c 的左右链接指针，未进行虚线连接的左指针 1740c 表示该栏是空栏。进行了虚线连接的右指针 1741c 的虚线的连接目的地表示指针所指示的地址，在当前情况下表示右指针指定节点 1750c。

[0009] 节点 1750d 的右指针 1741d 指定节点 1750d 自身，左链接 1740d 与节点 1750e 连接。1750e 保持的索引关键字是“010010”，检查比特位置是 5。节点 1750e 的左指针 1740e 指定节点 1750b，右指针 1741e 指定节点 1750e。

[0010] 并且，节点 1750f 保持的索引关键字是“101011”，检查比特位置 1730f 是 2。节点 1750f 的左链接 1740f 与节点 1750g 连接，右链接 1741f 与节点 1750h 连接。

[0011] 节点 1750g 保持的索引关键字是“100011”，检查比特位置 1730g 是 5。节点 1750g 的左指针 1740g 指定节点 1750a，右指针 1741g 指定节点 1750g。

[0012] 节点 1750h 保持的索引关键字是“101100”，检查比特位置 1730h 是 3。节点 1750h 的左指针 1740h 指定节点 1750f，右指针 1741h 指定节点 1750h。

[0013] 在图 17 的例子中，采用这样的结构：随着从根节点 1750a 开始对树进行向下遍历，

各节点的检查比特位置增大。

[0014] 当使用某检索关键字进行检索时，从根节点开始依次检查由各节点所保持的检索关键字的检查比特位置，判定检查比特位置的比特值是 1 还是 0，在是 1 时搜索右链接，在是 0 时搜索左链接。然后，当链接目的地的节点的检查比特位置不大于链接源的节点的检查比特位置时，即，链接目的地回到上方而不是下方时（将图 17 中虚线所示的该后退的链接称为反向链接），进行链接目的地的节点的索引关键字和检索关键字的比较。能够保证在比较结果是相同时检索成功，在比较结果是不相同时检索失败。

[0015] 如上所述，在使用 Patricia 树的检索处理中，有以下等优点：只通过所需要的比特的检查就能进行检索，以及关键字全体的比较只要一次就行，然而具有以下等缺点：由于必定有来自各节点的 2 个链接而使存储容量增大，由于反向链接的存在而使判定处理复杂化，在通过反向链接返回之后才与索引关键字进行比较而使得检索处理延迟以及使追加删除等数据维护困难。

[0016] 作为消除 Patricia 树的这些缺点的技术，例如有下述专利文献 1 所公开的技术。在下述专利文献 1 所记载的 Patricia 树中，通过将下位的左右节点存储在连续的区域内来削减指针的存储容量，并通过将表示下一链接是否是反向链接的比特设置在各节点内来减轻反向链接的判定处理。

[0017] 然而，在下述专利文献 1 所公开的技术中，由于 1 个节点必定占据索引关键字的区域和指针的区域，以及将下位的左右节点存储在连续的区域内而采用 1 个指针，因而需要将与节点相同容量的存储区域分配给例如图 17 所示的 Patricia 树的最下段部分即左指针 1740c、右指针 1741h 等部分等，存储容量的削减效果不怎么好。并且，反向链接导致的检索处理的延迟问题和追加删除等的处理困难的情况也未得到改善。

[0018] 专利文献 1：日本特开 2001-357070 号公报

发明内容

[0019] 因此，本发明要解决的课题是提供一种具有如下的数据结构的比特序列检索装置和检索方法，该数据结构是检索对象的比特序列的数据结构，所需要的存储容量小，检索速度高，数据维护容易。

[0020] 本发明的一方面是一种比特序列检索装置，该比特序列检索装置根据存储有由作为检索对象的比特序列构成的索引关键字的树的数据结构，通过由比特序列构成的检索关键字来检索所述索引关键字，其特征在于，

[0021] 该比特序列检索装置具有：

[0022] 配对节点树，其具有作为所述树的起点的根节点、以及具有配置在相邻存储区域中的两个节点的作为树的构成要素的节点对，所述节点具有存储表示该节点是分支节点还是叶节点的节点类别的区域，所述分支节点除了包含所述节点类别，还包含存储所述检索关键字的鉴别比特位置的区域和存储表示链接目的地节点对中的一个节点的位置的信息的区域，所述叶节点除了包含所述节点类别，还包含存储由所述检索对象比特序列构成的索引关键字的区域，

[0023] 根节点读取单元，其取得表示所述根节点的位置的信息，并根据该取得的表示根节点的位置的信息来读取根节点；

[0024] 节点类别判定单元，其从所述节点的存储节点类别的区域中读取相应的节点类别，并判定该节点类别是表示所述叶节点还是表示所述分支节点；

[0025] 索引关键字读取单元，其从所述叶节点的存储索引关键字的区域中读取相应的索引关键字；以及

[0026] 链接单元，其从所述分支节点中取出鉴别比特位置，并从所述检索关键字中取得所述取出的鉴别比特位置的比特值，从所述分支节点中取出表示链接目的地节点对中的代表节点的位置的信息，将该取得的鉴别比特位置的比特值与所述取出的表示链接目的地节点对中的代表节点的位置的信息相加，来求取表示链接目的地节点的位置的信息，从由该求取出的表示链接目的地节点的位置的信息所表示的存储区域中，读取配置在该存储区域的节点作为链接目的地节点，

[0027] 由所述节点类别判定单元来判定由所述根节点读取单元读取出的根节点的节点类别，如果该节点类别表示叶节点，则通过所述索引关键字读取单元从该叶节点读取索引关键字；如果该节点类别表示分支节点，则通过所述链接单元读取所述链接目的地节点，并由所述节点类别判定单元来判定该读取出的链接目的地节点的节点类别，反复进行以上操作，直到该节点类别表示叶节点为止，通过所述索引关键字读取单元从该叶节点读取索引关键字，

[0028] 对通过所述索引关键字读取单元读取出的索引关键字与所述检索关键字进行比较，如果一致则检索成功，如果不一致则检索失败。

[0029] 本发明的一方面是一种索引关键字插入方法，在该方法中比特序列检索装置根据存储有由作为检索对象的比特序列构成的索引关键字的树的数据结构，通过由比特序列构成的检索关键字来检索所述索引关键字，该比特序列检索装置在该树中存储由所希望的比特序列构成的插入关键字作为索引关键字，其特征在于，

[0030] 所述树是配对节点树，是存储在数组中的树，其具有作为该树的起点的根节点、以及具有配置在所述数组的相邻的数组要素中的两个节点的、作为树的构成要素的节点对，所述节点具有存储表示该节点是分支节点还是叶节点的节点类别的区域，所述分支节点除了包含所述节点类别，还包含存储所述检索关键字的鉴别比特位置的区域和存储所述数组的对链接目的地节点对中的一个节点进行存储的数组要素的数组编号的区域，所述叶节点除了包含所述节点类别，还包含存储由所述检索对象比特序列构成的索引关键字的区域，

[0031] 该比特序列检索装置具有：

[0032] 根节点读取单元，其取得所述数组的存储有所述根节点的数组要素的数组编号，并从该取得的数组编号的数组要素中读取根节点；

[0033] 节点类别判定单元，其读取所述节点的节点类别，并判定该节点类别是表示所述叶节点还是表示所述分支节点；

[0034] 索引关键字读取单元，其从所述叶节点的存储所述索引关键字的区域中读取相应的索引关键字；以及

[0035] 链接单元，其从所述分支节点中取出鉴别比特位置，并从所述检索关键字中取得所述取出的鉴别比特位置的比特值，从所述分支节点中取出表示链接目的地节点对中的代表节点的位置的信息，将该取得的鉴别比特位置的比特值与所述取出的表示链接目的地节点对中的代表节点的位置的信息相加，来求取表示链接目的地节点的位置的信息，从由该

求取出的表示链接目的地节点的位置的信息所表示的存储区域中，读取配置在该存储区域的节点作为链接目的地节点，

[0036] 该索引关键字插入方法将所述插入关键字作为所述检索关键字，具有以下步骤：

[0037] 检索步骤，由所述根节点读取单元读取根节点，由所述节点类别判定单元来判定该读取出的根节点的节点类别，如果该节点类别表示叶节点，则通过所述索引关键字读取单元从该叶节点读取索引关键字；如果该节点类别表示分支节点，则通过所述链接单元读取所述链接目的地节点，并通过所述节点类别判定单元来判定该读取出的链接目的地节点的节点类别，反复上述处理，直到该节点类别表示叶节点为止，通过所述索引关键字读取单元从该叶节点读取索引关键字，同时在堆栈中依次存储对直到该叶节点为止的链接路径中的分支节点和该叶节点进行存储的数组要素的数组编号；

[0038] 比较步骤，在通过所述检索步骤读取出的索引关键字与所述插入关键字之间进行大小比较和比特序列比较；

[0039] 空节点对取得步骤，从所述数组中取得存储节点对的空数组要素组，并取得该组的一个数组要素的数组编号；

[0040] 叶节点存储位置确定步骤，通过所述比较步骤中的所述大小比较，来确定把包含插入关键字的叶节点存储在通过所述空节点对取得步骤取得的空数组要素组的哪一个空数组要素中；

[0041] 节点对插入位置确定步骤，根据在所述比较步骤的比特序列比较中成为不同比特值的最初比特位置与存储在所述堆栈中的数组编号的数组要素所存储的分支节点的鉴别比特位置之间的相对位置关系，读取存储在该堆栈中的数组编号，将存储在该数组编号的数组要素中的节点作为存储在通过所述空节点对取得步骤取得的空数组要素组中的节点对的链接源，来确定该节点对的插入位置；

[0042] 插入节点对生成步骤，通过下述方式来生成插入节点对：在要配置到通过所述叶节点存储位置确定步骤确定的空数组要素中的叶节点中、在存储节点类别的区域中写入表示是叶节点的节点类别，在存储索引关键字的区域中写入所述插入关键字，并读出通过所述节点对插入位置确定步骤从所述堆栈读取出的数组编号的数组要素中所存储的节点内容，写入到另一个空数组要素中；

[0043] 分支节点生成步骤，将通过所述节点对插入位置确定步骤从所述堆栈读取出的数组编号的数组要素中所存储的节点作为分支节点，在其存储节点类别的区域中写入表示是分支节点的节点类别，在存储鉴别比特位置的区域中写入在所述比较步骤的比特序列比较中成为不同比特值的最初比特位置，在存储所述数组的对链接目的地节点对中的一个节点进行存储的数组要素的数组编号的区域中写入通过所述空节点对取得步骤取得的数组编号。

[0044] 本发明的一方面是一种索引关键字插入方法，在该方法中比特序列检索装置删除索引关键字，该比特序列检索装置根据存储有由作为检索对象的比特序列构成的索引关键字的树的数据结构，通过由比特序列构成的检索关键字来检索所述索引关键字，该比特序列检索装置删除存储在所述树中的、与由所希望的比特序列构成的删除关键字相同的索引关键字，其特征在于，

[0045] 所述树是存储在数组中的配对节点树，其具有作为所述树的起点的根节点、以及

具有配置在所述数组的相邻的数组要素中的两个节点、作为树的构成要素的节点对，所述节点具有存储表示该节点是分支节点还是叶节点的节点类别的区域，所述分支节点除了包含所述节点类别，还包含存储所述检索关键字的鉴别比特位置的区域和存储所述数组的对链接目的地节点对中的一个节点进行存储的数组要素的数组编号的区域，所述叶节点除了包含所述节点类别，还包含存储由所述检索对象比特序列构成的索引关键字的区域，

[0046] 该比特序列检索装置具有：

[0047] 根节点读取单元，其取得所述数组的存储所述根节点的数组要素的数组编号，并从该取得的数组编号的数组要素中读取根节点；

[0048] 节点类别判定单元，其读取所述节点的节点类别，并判定该节点类别是表示所述叶节点还是表示所述分支节点；

[0049] 索引关键字读取单元，其从所述叶节点的存储所述索引关键字的区域中读取相应的索引关键字；以及

[0050] 链接单元，其从所述分支节点中取出鉴别比特位置，并从所述检索关键字中取得所述取出的鉴别比特位置的比特值，从所述分支节点中取出表示链接目的地节点对中的代表节点的位置的信息，将该取得的鉴别比特位置的比特值与所述取出的表示链接目的地节点对中的代表节点的位置的信息相加，来求取表示链接目的地节点的位置的信息，从由该求取出的表示链接目的地节点的位置的信息所表示的存储区域中，读取配置在该存储区域的节点作为链接目的地节点，

[0051] 该索引关键字删除方法将所述删除关键字作为所述检索关键字，具有以下步骤：

[0052] 检索步骤，由所述根节点读取单元读取根节点，由所述节点类别判定单元来判定该读取出的根节点的节点类别，如果该节点类别表示叶节点，则通过所述索引关键字读取单元从该叶节点读取索引关键字；如果该节点类别表示分支节点，则通过所述链接单元读取所述链接目的地节点，并由所述节点类别判定单元来判定该读取出的链接目的地节点的节点类别，反复进行上述处理，直到该节点类别表示叶节点为止，通过所述索引关键字读取单元从该叶节点读取索引关键字，同时在堆栈中依次存储对直到该叶节点为止的链接路径中的分支节点和该叶节点进行存储的数组要素的数组编号；

[0053] 当通过所述检索步骤读取出的索引关键字与所述删除关键字相同时，该索引关键字删除方法还具有以下步骤：

[0054] 节点读取步骤，读取存储了与保持该索引关键字的所述叶节点一起构成同一节点对的节点的数组要素的内容；

[0055] 写入步骤，从该堆栈中读取存储在所述堆栈中的、比存储所述叶节点的数组要素的数组编号前一个的数组编号，在该数组编号的数组要素中写入通过所述节点读取步骤读取出的数组要素的内容；以及

[0056] 节点对删除步骤，释放存储有所述节点对的数组要素的组。

[0057] 本发明的一方面是一种比特序列检索方法，该方法中，比特序列检索装置根据存储有由作为检索对象的比特序列构成的索引关键字的树的数据结构，通过由比特序列构成的检索关键字来检索所述索引关键字，其特征在于，

[0058] 所述树是配对节点树，其具有作为该树的起点的根节点、以及具有配置在相邻的存储区域内的两个节点、作为树的构成要素的节点对，所述节点具有存储表示该节点是分

支节点还是叶节点的节点类别的区域，所述分支节点除了包含所述节点类别，还包含存储所述检索关键字的鉴别比特位置的区域和存储表示链接目的地节点对中的一个节点的位置的信息的区域，所述叶节点除了包含所述节点类别，还包含存储由所述检索对象比特序列构成的索引关键字的区域，

[0059] 该比特序列检索方法包含以下步骤：

[0060] 根节点读取步骤，取得表示所述根节点的位置的信息，并根据该取得的表示根节点的位置的信息来读取根节点；

[0061] 节点类别判定步骤，从所述节点的存储节点类别的区域中读取相应的节点类别，并判定该节点类别是表示所述叶节点还是表示所述分支节点；

[0062] 索引关键字读取步骤，从所述叶节点的存储所述索引关键字的区域中读取相应的索引关键字；以及

[0063] 链接步骤，从所述分支节点的存储鉴别比特位置的区域和存储表示链接目的地节点对中的一个节点的位置的信息的区域中，分别读取相应的鉴别比特位置和表示链接目的地节点对中的一个节点的位置的信息，通过所述检索关键字的该读取出的鉴别比特位置的比特值和所述读取出的表示链接目的地节点对中的一个节点的位置的信息之间的运算，来求取表示节点位置的信息，从由该求取出的表示节点位置的信息所表示的存储区域中，读取配置在该存储区域中的节点，作为链接目的地节点，

[0064] 通过所述节点类别判定步骤来判定通过所述根节点读取步骤读取出的根节点的节点类别，如果该节点类别表示叶节点，则通过所述索引关键字读取步骤从该叶节点读取索引关键字；如果该节点类别表示分支节点，则通过所述链接步骤读取所述链接目的地节点，并通过所述节点类别判定步骤来判定该读取出的链接目的地节点的节点类别，反复进行上述处理，直到该节点类别表示叶节点为止，通过所述索引关键字读取步骤从该叶节点读取索引关键字，

[0065] 对通过所述索引关键字读取步骤读取出的索引关键字与所述检索关键字进行比较，如果一致则检索成功，如果不一致则检索失败。

[0066] 根据本发明，提供一种名叫配对节点树 (coupled node tree) 的具有以下数据结构的树，在本发明的比特序列检索装置中，使用该配对节点树来进行索引关键字的检索。

[0067] 本发明的配对节点树具有：具有链接目的地的数据的分支节点、以及具有作为检索对象的索引关键字的叶节点。并且，该树结构由根节点、以及配置在相邻的存储区域内的分支节点和叶节点或分支节点之间或叶节点之间的节点对构成。

[0068] 分支节点包含检索关键字的鉴别比特位置以及用于链接到链接目的地的节点对中的一方即代表节点的代表节点编号，所述叶节点包含由作为检索对象的比特序列构成的索引关键字。除了树的节点仅为 1 个时以外，根节点是分支节点。

[0069] 检索关键字的鉴别比特位置在使用检索关键字的该位置的比特值方面与 Patricia 树的检查比特位置相同，然而不同点是，在 Patricia 树中，判定检查比特位置的比特值来求出链接目的地，而在本发明的配对节点树中，将鉴别比特位置的比特值用于求出链接目的地的节点的运算。

[0070] 使用检索关键字执行的检索是在包含根节点在内的各分支节点中，通过根据该分支节点内包含的鉴别比特位置的检索关键字的比特值依次重复进行链接到链接目的地的

节点对中的一个节点的动作直到到达叶节点来执行的。

[0071] 当到达叶节点时，将叶节点保持的索引关键字和检索关键字进行比较，如果一致则检索成功。如果不一致，则在检索对象的索引关键字内没有与检索关键字一致的索引关键字。

[0072] 在对本发明的配对节点树追加新的索引关键字的情况下，首先将该索引关键字用作检索关键字来进行检索，取得所找到的叶节点的索引关键字。并且，将搜索到叶节点为止的分支节点的编号从根节点开始依次存储在堆栈内。

[0073] 在要追加的索引关键字和通过检索而取得的索引关键字之间进行大小比较和比特序列比较，根据作为通过比特序列比较而不同的比特值的、开头的（最上位的）比特位置与存储在堆栈内的分支节点的鉴别比特位置之间的相对位置关系，决定所追加的节点对的插入位置，根据大小关系决定将包含要追加的索引关键字的叶节点作为待追加的节点对中的哪个节点。

[0074] 当从本发明的配对节点树中删除某索引关键字时，通过这样来进行，即：使用要删除的关键字进行检索，将与保持删除对象的索引关键字的节点构成同一节点对的节点的内容写入到该节点对的链接源的分支节点内，删除该节点对。

[0075] 本发明的配对节点树使用节点对来构成树结构，可针对索引关键字的集合使其大小紧凑化。并且，由于将节点分离为包含指针的分支节点和包含索引关键字的叶节点，因而使需要指针信息的节点具有指针信息，不需要索引关键字用的区域，需要索引关键字的节点不需要指针用的区域，因而不会在存储区域内产生浪费。并且，也无需 Patricia 树那样的反向链接的处理。

[0076] 在将配对节点树存储在数组内的情况下，指针可采用数组编号，可进一步减少所需要的存储容量。

[0077] 根据本发明，如上所述可减少存储用于检索的树结构体的存储容量，从而可减小检索处理负担，而且分支节点配置成保持由配对节点树内包含的索引的比特序列结构所规定的鉴别比特位置，因而处理仅是在所需要的比特位置处的处理，分支处理的负担也小。并且，鉴别比特位置的比特值不会用于判定处理，而用于运算处理，因而在这一点上减轻了 CPU 处理负担。

[0078] 并且，配对节点树的插入处理和删除处理也简单，维护负担也轻。

[0079] 如上所述，根据本发明，可提供一种能高速执行比特序列检索、数据维护容易的比特序列检索装置。

附图说明

[0080] 图 1 是说明存储在数组中的配对节点树的结构例的图。

[0081] 图 2 是说明配对节点树的树结构的图。

[0082] 图 3 是说明用于实施本发明的硬件结构例的图。

[0083] 图 4 是示出本发明的一实施方式中的检索处理的流程图。

[0084] 图 5 是示出本发明的一实施方式中的插入处理的前段即检索处理的处理流程的图。

[0085] 图 6 是说明本发明的一实施方式中的插入处理中的准备要插入的节点对用的数

组要素的处理流程的图。

[0086] 图 7 是示出求出插入节点对的位置、写入节点对的各节点的内容来完成插入处理的处理流程的图。

[0087] 图 8 是说明包含本发明的一实施方式中的根节点的插入处理在内的在追加索引关键字的情况下的整个节点插入处理的处理流程图。

[0088] 图 9 是示出本发明的一实施方式中的删除处理的前段即检索处理的处理流程的图。

[0089] 图 10 是说明本发明的一实施方式中的删除处理的后段的处理流程的图。

[0090] 图 11A 是说明删除处理前的配对节点树和删除关键字“011010”的图。

[0091] 图 11B 是说明删除处理后的配对节点树的图。

[0092] 图 12A 是说明插入处理前的配对节点树和插入关键字“0011”的图。

[0093] 图 12B 是说明插入处理后的配对节点树的图。

[0094] 图 13 是说明图 5 所示的插入处理的前段即检索处理中的处理框的流程的图。

[0095] 图 14 是说明图 6 所示的为了要插入的节点对而准备数组要素的处理框的流程的图。

[0096] 图 15 是说明图 7 所示的处理中的前半部分，即求出要插入的节点对在配对节点树上的位置的处理流程的图。

[0097] 图 16 是说明图 7 所示的处理中的后半部分，即在各节点设定数据来完成插入处理的处理流程的图。

[0098] 图 17 是示出以往的检索中使用的 Patricia 树的一例的图。

具体实施方式

[0099] 以下，说明用于实施本发明的最佳方式，即将配对节点树存储在数组内的例子。作为分支节点保持的表示链接目的地的位置的数据，还能使用存储装置的地址信息，然而通过使用由可存储分支节点或叶节点中占有区域的存储容量大的一方的数组要素构成的数组，可使用数组编号表示节点位置，可削减位置信息的信息量。

[0100] 图 1 是说明存储在数组内的配对节点树的结构例的图。

[0101] 参照图 1，节点 101 配置在数组 100 的数组编号为 10 的数组要素内。节点 101 由节点类别 102、鉴别比特位置 103 以及代表节点编号 104 构成。节点类别 102 是 0，表示节点 101 是分支节点。在鉴别比特位置 103 内存储有 1。在代表节点编号 104 内存储有链接目的地的节点对的代表节点的数组编号 20。另外，以下为了简化表述，有时把存储在代表节点编号内的数组编号称为代表节点编号。并且，存储在代表节点编号内的数组编号有时利用付加给该节点的符号或付加给节点对的符号来表示。

[0102] 在数组编号 20 的数组要素内存储有节点对 111 的代表节点即节点 [0]112。然后在相邻的下一数组要素（数组编号 20+1）内存储有与代表节点成对的节点 [1]113。在节点 [0]112 的节点类别 114 内存储有 0，在鉴别比特位置 115 内存储有 3，在代表节点编号 116 内存储有 30。并且，在节点 [1]113 的节点类别 117 内存储有 1，表示节点 [1]113 是叶节点。在索引关键字 118 内存储有“0001”。与之前针对 Patricia 树描述的一样，当然在叶节点内包含有对与索引关键字对应的记录进行存取的信息，然而省略表述。

[0103] 另外，有时代表节点利用节点 [0] 表示，与其成对的节点利用节点 [1] 表示。

[0104] 省略了由存储在数组编号 30 和 31 的数组要素内的节点 122 和节点 123 构成的节点对 121 的内容。

[0105] 分别赋予给存储有节点 [0]112、节点 [1]113、节点 122 以及节点 123 的数组要素的 0 或 1 表示在使用检索关键字进行检索的情况下链接到节点对的哪个节点。链接到将位于前段的分支节点的鉴别比特位置上的检索关键字的比特值 0 或 1 与代表节点编号相加所得到的数组编号的节点。

[0106] 因此，通过将前段的分支节点的代表节点编号与检索关键字的鉴别比特位置的比特值相加，可求出存储有链接目的地的节点的数组要素的数组编号。

[0107] 另外，在上述例子中代表节点编号采用配置有节点对的数组编号中的小的一方，然而显然也能采用大的一方。

[0108] 图 2 是概念性示出配对节点树的树结构的图。图示的 6 比特的索引关键字与图 17 所例示的 Patricia 树的相同。

[0109] 利用符号 210a 表示的是根节点。在图示的例子中，根节点 210a 作为配置在数组编号 220 上的节点对 201a 的代表节点。

[0110] 作为树结构，在根节点 210a 的下方配置有节点对 201b，在节点对 201b 的下层配置有节点对 201c 和节点对 201f，在节点对 201f 的下层配置有节点对 201h 和节点对 201g。在节点对 201c 的下方配置有节点对 201d，并在节点对 201d 的下方配置有节点对 201e。

[0111] 附在各节点的前方的符号 0 或 1 与在图 1 中所说明的附在数组要素的前方的符号相同。根据检索关键字的鉴别比特位置的比特值来搜索树，从而找到检索对象的叶节点。

[0112] 在图示的例子中，根节点 210a 的节点类别 260a 是 0，表示是分支节点，鉴别比特位置 230a 表示 0。代表节点编号是 220a，该编号是存储有节点对 201b 的代表节点 210b 的数组要素的数组编号。

[0113] 节点对 201b 由节点 210b 和 211b 构成，它们的节点类别 260b、261b 都是 0，表示是分支节点。在节点 210b 的鉴别比特位置 230b 上存储有 1，在链接目的地的代表节点编号内存储有存储了节点对 201c 的代表节点 210c 的数组要素的数组编号 220b。

[0114] 由于在节点 210c 的节点类别 260c 内存储有 1，因而该节点是叶节点，因此包含索引关键字。在索引关键字 250c 内存储有“000111”。另一方面，节点 211c 的节点类别 261c 是 0，鉴别比特位置 231c 是 2，在代表节点编号内存储有存储了节点对 201d 的代表节点 210d 的数组要素的数组编号 221c。

[0115] 节点 210d 的节点类别 260d 是 0，鉴别比特位置 230d 是 5，在代表节点编号内存储有存储了节点对 201e 的代表节点 210e 的数组要素的数组编号 220d。与节点 210d 成对的节点 211d 的节点类别 261d 是 1，在索引关键字 251d 内存储有“011010”。

[0116] 节点对 201e 的节点 210e、211e 的节点类别 260e、261e 都是 1，表示双方都是叶节点，在各个索引关键字 250e、251e 内存储有“010010”和“010011”作为索引关键字。

[0117] 在节点对 201b 的另一个节点即节点 211b 的鉴别比特位置 231b 内存储有 2，在链接目的地的代表节点编号内存储有存储了节点对 201f 的代表节点 210f 的数组要素的数组编号 221b。

[0118] 节点对 201f 的节点 210f、211f 的节点类别 260f、261f 都是 0，表示双方都是分支

节点。在各个鉴别比特位置 230f、231f 内存储有 5、3。在节点 210f 的代表节点编号内存储有存储了节点对 201g 的代表节点 210g 的数组要素的数组编号 220f，在节点 211f 的代表节点编号内存储有存储了节点对 201h 的代表节点即节点 [0]210h 的数组要素的数组编号 221f。

[0119] 节点对 201g 的节点 210g、211g 的节点类别 260g、261g 都是 1，表示双方都是叶节点，在各个索引关键字 250g、251g 内存储有“100010”和“100011”。

[0120] 并且同样，节点对 201h 的代表节点即节点 [0]210h 和与其成对的节点 [1]211h 的节点类别 260h、261h 都是 1，表示双方都是叶节点，在各个索引关键字 250h、251h 内存储有“101011”和“101100”。

[0121] 以下，简单说明从上述的树中检索索引关键字“100010”的处理流程。鉴别比特位置从左起为 0、1、2、...。

[0122] 首先，将比特序列“100010”用作检索关键字，从根节点 210a 开始处理。由于根节点 210a 的鉴别比特位置 230a 是 0，因而当查看检索关键字“100010”的鉴别比特位置 0 的比特值时是 1。因此，链接到在对存储有代表节点编号的数组编号 220a 加上 1 后的数组编号的数组要素内存储的节点 211b。由于在节点 211b 的鉴别比特位置 231b 内存储有 2，因而当查看检索关键字“100010”的鉴别比特位置 2 的比特值时是 0，因此链接到在存储有代表节点编号的数组编号 221b 的数组要素内存储的节点 210f。

[0123] 由于在节点 210f 的鉴别比特位置 230f 内存储有 5，因而当查看检索关键字“100010”的鉴别比特位置 5 的比特值时是 0，因此链接到在存储有代表节点编号的数组编号 220f 的数组要素内存储的节点 210g。

[0124] 由于节点 210g 的节点类别 260g 是 1，表示是叶节点，因而当读出索引关键字 250g 来与检索关键字进行比较时，双方都是“100010”是一致的。这样进行使用配对节点树的检索。

[0125] 下面，参照图 2 说明配对节点树的结构的意义。

[0126] 配对节点树的结构是根据索引关键字的集合来规定的。在图 2 的例子中，根节点 210a 的鉴别比特位置是 0，这是因为在图 2 所例示的索引关键字中有第 0 比特是 0 的索引关键字以及第 0 比特是 1 的索引关键字。第 0 比特是 0 的索引关键字的组被分类在节点 210b 的下方，第 0 比特是 1 的索引关键字的组被分类在节点 211b 的下方。

[0127] 节点 211b 的鉴别比特位置是 2，这反映了索引关键字的集合的性质，即：存储在节点 211h、210h、211g、210g 内的第 0 比特是 1 的索引关键字的第 1 比特全部相同是 0，从第 2 比特开始才有不同。

[0128] 以下与第 0 比特的情况一样，第 2 比特是 1 的索引关键字被分类在节点 211f 侧，第 2 比特是 0 的索引关键字被分类在节点 210f 侧。

[0129] 然后，由于第 2 比特是 1 的索引关键字中存在第 3 比特不同的索引关键字，因而在节点 211f 的鉴别比特位置上存储 3，由于在第 2 比特是 0 的索引关键字中第 3 比特和第 4 比特均相同而第 5 比特不同，因而在节点 210f 的鉴别比特位置上存储 5。

[0130] 在节点 211f 的链接目的地中，由于第 3 比特是 1 的索引关键字和第 3 比特是 0 的索引关键字分别只有一个，因而节点 210h、211h 成为叶节点，分别在索引关键字 250h 和 251h 内存储有“101011”和“101100”。

[0131] 假设在索引关键字的集合内包含有“101101”或“101110”而取代“101100”，到第3比特为止也与“101100”相等，因而只是存储在节点211h内的索引关键字改变，树结构自身不会改变。然而，当除了“101100”以外还包含有“101101”时，节点211h成为分支节点，其鉴别比特位置成为5。当所追加的索引关键字是“101110”时，鉴别比特位置为4。

[0132] 如以上说明那样，配对节点树的结构是根据索引关键字的集合内包含的各索引关键字的各比特位置的比特值来决定的。当使用检索关键字进行检索时，索引关键字搜索配置在配对节点树上的根上，例如当检索关键字是“101100”时，可到达节点211h。并且，从上述说明还可以想象出，即使在将“101101”或“101110”用作检索关键字的情况下，也搜索到节点211h，通过与索引关键字251h进行比较可知检索失败。

[0133] 并且，例如在使用“100100”进行检索的情况下，在节点210a、211b、210f的链接路径上不会使用检索关键字的第3比特和第4比特，“100100”的第5比特是0，因而与使用“100010”进行检索的情况一样到达节点210g。这样，使用与存储在配对节点树内的索引关键字的比特结构对应的鉴别比特位置来进行分支。

[0134] 另外，后面详细说明检索的详细算法和针对索引关键字的集合如何构成配对节点树。

[0135] 图3是说明用于实施本发明的硬件结构例的图。

[0136] 本发明的检索装置的检索处理和数据维护是通过至少具有中央处理装置302和高速缓存303的数据处理装置301使用数据存储装置308来实施的。具有配置配对节点树的数组309和探索路径堆栈310的数据存储装置308可使用主存储装置305或外部存储装置306来实现，或者还能使用通过通信装置307连接的配置在远方的装置来实现，该探索路径堆栈310用于保存存储有在检索中搜索的节点的数组要素的数组编号。

[0137] 在图3的例示中，主存储装置305、外部存储装置306以及通信装置307通过一根总线304与数据处理装置301连接，然而连接方法不限于此。并且，也能使主存储装置305为数据处理装置301内的装置，也能将探索路径堆栈310作为中央处理装置302内的硬件来实现。或者，显然可以根据可使用的硬件环境、索引关键字集合的大小等适当选择使数组309在外部存储装置306内、使探索路径堆栈310在主存储装置305内等硬件结构。

[0138] 并且，尽管未作特别图示，然而为了在后面的处理中使用在处理途中所获得的各种值，当然可以使用与各个处理对应的暂时存储装置。

[0139] 图4是示出本发明的一实施方式中的检索处理的流程图。适当参照图2，说明检索处理的算法。

[0140] 当开始检索处理时，取得作为检索对象的配对节点树的根节点的数组编号（步骤S401）。由于配对节点树可根据其根节点的数组编号来识别，因而配对节点树的管理可使用根节点的数组编号来进行。因此，假定在配对节点树的管理单元内登记有配对节点树的根节点的数组编号。

[0141] 另外，图示的检索处理显然可使用计算机上的程序来实现，然而上述配对节点树的管理单元例如可以为使计算机执行检索处理的程序上的存储区域。

[0142] 并且，在一个计算机系统上搭载了使用多个配对节点树的检索系统的情况下，还可以在各个检索程序的外部具有该检索系统。

[0143] 在图2的例子中，相当于取得数组编号220。另外，根节点的数组编号的取得不限

于上述方法，可以从键盘等直接输入数组编号，也可以作为固定值预先在程序中设定。

[0144] 然后在步骤 S402 中，将在步骤 S401 所取得的数组编号存储在探索路径堆栈内。进到步骤 S403，从数组中读出堆在探索路径堆栈的最上部的数组编号的数组要素作为链接目的地的节点。在步骤 S404 中，从在步骤 S403 所读出的节点中取出节点类别。

[0145] 然后在步骤 S405 中进行节点类别的判定。当步骤 S405 的判定是分支节点时，转到步骤 S406。在步骤 S406 中，从节点中取出鉴别比特位置。然后在步骤 S407 中，从检索关键字中取得在步骤 S406 所取出的鉴别比特位置的比特值。然后进到步骤 S408，从节点中取得链接目的地的节点对的代表节点的数组编号。接下来进到步骤 S409，将在步骤 S408 所取得的数组编号与在步骤 S407 所取得的比特值相加，取得链接目的地节点的数组编号，回到步骤 S402。

[0146] 在步骤 S402 中将在步骤 S409 所取得的数组编号存储在探索路径堆栈内。

[0147] 重复以上步骤 S402 至 S409 的循环处理，直到步骤 S405 的判定为叶节点。

[0148] 当在步骤 S405 判定为节点类别是叶节点时，转到步骤 S410，从节点中取出索引关键字。然后进到步骤 S411，判定检索关键字和索引关键字是否相同。如果相同则检索成功，如果不相同则检索失败。

[0149] 在上述说明中，为了识别处理中的节点的数组编号而使用了堆栈，然而如果只是为了检索，则也能将普通的工作区域用作暂时存储。然而，在以下说明的插入处理中，由于使用堆栈是有效的，因而优选的是在检索处理中也使用堆栈。

[0150] 下面，使用图 5 ～图 8 说明配对节点树中的节点插入处理。图 5 ～图 7 是说明常规的插入处理的图，图 8 是说明根节点的插入处理的图。由于通过根节点的插入处理和常规的插入处理来生成配对节点树，因而节点插入处理的说明也是配对节点树的生成处理的说明。

[0151] 图 5 是示出插入处理的前段即检索处理的处理流程的图，相当于在图 4 所示的检索处理中将插入关键字用作检索关键字的处理。由于步骤 S501 ～步骤 S510 的处理完全对应于图 4 的步骤 S401 ～步骤 S410，因而省略说明。

[0152] 在图 5 的步骤 S511 中将插入关键字和索引关键字进行比较，如果相同，则插入关键字已存在于配对节点树内，因而插入失败，结束处理。如果不相同，则进到后面的处理，即图 6 的步骤 S512 以下的处理。

[0153] 图 6 是说明准备要插入的节点对用的数组要素的处理的处理流程图。

[0154] 在步骤 S512 中，根据数组求出空的节点对，取得该节点对中的应成为代表节点的数组要素的数组编号。

[0155] 进到步骤 S513，将插入关键字与在步骤 S510 所获得的索引关键字的大小进行比较，当插入关键字大时获得值 1、当插入关键字小时获得值 0 的布尔值。

[0156] 进到步骤 S514，获得将在步骤 S512 所获得的代表节点的数组编号与在步骤 S513 所获得的布尔值相加后的数组编号。

[0157] 进到步骤 S515，获得将在步骤 S512 所获得的代表节点的数组编号与在步骤 S513 所获得的布尔值的逻辑“非”值相加后的数组编号。

[0158] 在步骤 S514 所获得的数组编号是存储具有插入关键字作为索引关键字的叶节点的数组要素的数组编号，在步骤 S515 所获得的数组编号是存储与该叶节点成对的节点的

数组要素的数组编号。

[0159] 即，根据在前段的检索处理中获得的叶节点内所存储的索引关键字和插入关键字的大小，决定在插入的节点对中的哪个节点内存储保持插入关键字的叶节点。

[0160] 例如在图 2 的配对节点树内插入“011011”的情况下，检索结果的索引关键字也为存储在节点 211d 内的“011010”。通过插入关键字“011011”和存储在节点 211d 内的索引关键字“011010”的大小比较来求出布尔值，在当前的例子中由于插入关键字大，因而获得布尔值 1，在将插入的节点对的代表节点编号加上 1 后的数组要素内存储保持插入关键字的叶节点。另一方面，索引关键字“011010”被存储在将通过大小比较所获得的布尔值进行了逻辑反转后的值与代表节点编号相加所得到的数组编号的数组要素内。

[0161] 此时，由于索引关键字“011010”和插入关键字“011011”在第 5 比特不同，因而节点 211d 成为如下的分支节点 :将鉴别比特位置设为 5、将代表节点编号设为所插入的节点对的代表节点的数组编号。

[0162] 并且，在要将“011001”插入到图 2 的配对节点树内的情况下，检索结果的索引关键字为存储在节点 211d 内的“011010”。在该情况下，由于插入关键字小，因而获得布尔值 0，在将插入的节点对的代表节点编号加上 0 后的数组要素内存储保持插入关键字的叶节点。然后，由于索引关键字“011010”和插入关键字“011001”在第 4 比特不同，因而节点 211d 成为如下的分支节点 :将鉴别比特位置设为 4、将代表节点编号设为所插入的节点对的代表节点的数组编号。然后进到图 7 的步骤 S516 以下的处理。

[0163] 图 7 是示出将节点存储在图 6 所准备的数组内并求出其插入位置、变更已有节点的内容来完成插入处理的处理流程的图。

[0164] 步骤 S516 ～步骤 S523 的处理是求出要插入的节点对在配对节点树上的位置的处理，步骤 S524 以下的处理是在各节点设定数据来完成插入处理的处理。

[0165] 在步骤 S516 中，通过使用例如“异或”来进行插入关键字与在步骤 S510 所获得的索引关键字的比特序列比较，获得差分比特序列。

[0166] 进到步骤 S517，根据在步骤 S516 所获得的差分比特序列，获得从上位第 0 比特开始观察到的第一个不一致比特的比特位置。该处理可在例如具有优先编码器的 CPU 中通过向 CPU 输入差分比特序列，来获得不一致的比特位置。并且，也能以软件方式进行与优先编码器同等的处理，获得第一个不一致比特的比特位置。

[0167] 然后进到步骤 S518，判定探索路径堆栈的堆栈指针是否指定根节点的数组编号。在指定根节点的数组编号时转到步骤 S524，在未指定根节点的数组编号时进到步骤 S519。

[0168] 在步骤 S519 中，使探索路径堆栈的堆栈指针后退 1 个，取出堆栈在其内的数组编号。

[0169] 进到步骤 S520，从数组中读出在步骤 S519 所取出的数组编号的数组要素作为节点。

[0170] 进到步骤 S521，从在步骤 S520 所读出的节点中取出鉴别比特位置。

[0171] 然后进到步骤 S522，判定在步骤 S521 所取出的鉴别比特位置与在步骤 S517 所获得的比特位置相比是否为上位的位置关系。假定这里上位的位置关系是指比特序列的靠左侧的位置，即比特位置的值小的位置。

[0172] 在步骤 S522 的判定结果是否定时，回到步骤 S518，重复执行直到步骤 S518 的判定

为肯定或者步骤 S522 的判定为肯定。当步骤 S522 的判定为肯定时，在步骤 S523 使探索路径堆栈的堆栈指针前进 1 个，转到步骤 S524 以下的处理。

[0173] 在上述步骤 S516 ～步骤 S523 所说明的处理是这样的处理：为了决定要插入的节点对的插入位置，在要插入的索引关键字与通过检索所取得的索引关键字之间进行比特序列比较，调查作为通过比特序列比较而不同的比特值的、开头的（最上位的）比特位置与存储在探索路径堆栈内的分支节点的鉴别比特位置之间的相对位置关系，将以鉴别比特位置为上位的分支节点的下一分支节点的链接目的地设为要插入的节点对的插入位置。

[0174] 例如当把“111000”插入到图 2 的配对节点树内时，检索结果的索引关键字为存储在节点 210h 内的“101011”。获得作为通过插入关键字“111000”与存储在节点 210h 内的索引关键字“101011”的比特序列比较而不同的比特值的、最上位的比特位置 1。当在探索路径堆栈内依次反向搜索所获得的比特位置 1 与在堆栈在探索路径堆栈中的数组编号的数组要素内所存储的分支节点的鉴别比特位置之间的位置关系直到鉴别比特位置为上位时，到达根节点 210a。因此，使探索路径堆栈的指针前进 1 个，获得节点 211b 的数组编号。插入关键字“111000”被插入到节点 211b 的链接目的地。

[0175] 并且，即使对探索路径堆栈进行反向搜索并到达根节点，与先前求出的作为通过比特序列比较而不同的比特值的、最上位的比特位置相比，根节点的鉴别比特位置也不是上位的比特位置是指以下的情况：在该配对节点树的索引关键字的上位比特，与根节点的鉴别比特位置相比为上位的比特值全部相同。并且是指如下情况：在要插入的索引关键字中，在最初的与根节点的鉴别比特位置相比为上位的比特值中存在不同的比特值。因此，要插入的节点对成为根节点的直接链接目的地，根节点的鉴别比特位置变为与已有的索引关键字不同的值即插入关键字的最上位比特的位置。

[0176] 下面，对步骤 S524 以下的在各节点设定数据来完成插入处理的处理进行说明。

[0177] 在步骤 S524 中，从探索路径堆栈中取出堆栈指针所指定的数组编号。

[0178] 在步骤 S525 中，向在步骤 S514 所获得的数组编号所指定的数组要素的节点类别内写入 1（叶节点），向索引关键字内写入插入关键字。

[0179] 进到步骤 S526，从数组中读出在步骤 S524 所获得的数组编号的数组要素。

[0180] 然后在步骤 S527 中，向在步骤 S515 所获得的数组编号的数组要素内写入在步骤 S526 所读出的内容。

[0181] 最后在步骤 S528 中，向在步骤 S524 所获得的数组编号所指定的数组要素的节点类别内写入 0（分支节点），向鉴别比特位置内写入在步骤 S517 所获得的比特位置，向代表节点编号内写入在步骤 S512 所获得的数组编号，结束处理。

[0182] 在向上述图 2 的配对节点树内插入“111000”的例子中，向所取得的空节点对的节点 [0] 内写入节点 211b 的内容（步骤 S527），将节点 [1] 设为保持插入关键字“111000”的叶节点（步骤 S525）。然后，在节点 211b 的鉴别比特位置上存储作为通过比特序列比较而不同的比特值的、最上位的比特位置 1，将存储所取得的节点对的代表节点的数组要素的数组编号存储在代表节点编号内（步骤 S528）。

[0183] 图 8 是说明包含本发明的一实施方式中的根节点的插入处理在内的在追加索引关键字的情况下的整个节点插入处理的处理流程图。

[0184] 在步骤 S551 中，判定要求取得的配对节点树的根节点的数组编号是否已登记。如

果已登记，则进行使用图 5 ～图 7 所说明的常规的插入处理。

[0185] 如果步骤 S551 的判定为未登记，则开始全新的配对节点树的登记、生成。

[0186] 首先，在步骤 S552 中，根据数组求出空节点对，取得该节点对中的应成为代表节点的数组要素的数组编号。然后在步骤 S553 中，求出在步骤 S552 所获得的数组编号加上 0 后的数组编号（实际上，等于在步骤 S552 所取得的数组编号）。然后在步骤 S554 中，向要插入的根节点的节点类别内写入 1（叶节点），并向索引关键字内写入插入关键字，在步骤 S553 所获得的数组编号的数组要素内登记在步骤 S556、步骤 S552 所取得的根节点的数组编号，结束处理。

[0187] 如前所述，显然，当有索引关键字的集合时，从该集合中依次取出索引关键字，通过重复进行图 8 和图 5 ～图 7 的处理，可构建与索引关键字的集合对应的本发明的配对节点树。

[0188] 下面，参照图 9 和图 10，对从本发明的一实施方式中的配对节点树涉及的索引关键字的集合中删除特定的索引关键字的处理流程进行说明。

[0189] 图 9 是示出删除处理的前段即检索处理的处理流程的图，相当于在图 4 所示的检索处理中把删除关键字作为检索关键字的处理。由于步骤 S901 ～步骤 S910 的处理与图 4 的步骤 S401 ～步骤 S410 完全对应，因而省略说明。

[0190] 在图 9 的步骤 S911 中将删除关键字与索引关键字进行比较，如果不相同，则要删除的索引关键字不存在于配对节点树内，因而删除失败，结束处理。如果相同，则进到后面的处理，即图 10 的步骤 S912 以下的处理。

[0191] 图 10 是说明删除处理的后段的处理流程的图。

[0192] 首先，在步骤 S912 判定在探索路径堆栈内是否存储有 2 个以上的数组编号。未存储有 2 个以上的数组编号，换句话说只存储有 1 个数组编号，则该数组编号是存储有根节点的数组要素的数组编号。在该情况下，转到步骤 S918，删除在步骤 S901 所获得的根节点的数组编号涉及的节点对。然后进到步骤 S919，删除所登记的根节点的数组编号，结束处理。

[0193] 当在步骤 S912 中判定为在探索路径堆栈内存储有 2 个以上的数组编号时，进到步骤 S913，获得将步骤 S908 中获得的代表节点编号与使步骤 S907 中所获得的比特值反转后的值相加所得到的数组编号。该处理是求出配置有与存储了删除对象的索引关键字的叶节点成对的节点的数组编号的处理。

[0194] 然后在步骤 S914 中，读出在步骤 S913 所获得的数组编号的数组要素的内容，在步骤 S915 中使探索路径堆栈的堆栈指针后退 1 个，取出数组编号。

[0195] 然后进到步骤 S916，将在步骤 S914 所读出的数组要素的内容改写为在步骤 S915 所获得的数组编号的数组要素。该处理是将作为链接到存储有删除对象的索引关键字的叶节点的链接源的分支节点置换为与上述叶节点成对的节点的处理。

[0196] 最后在步骤 S917 中删除在步骤 S908 所获得的代表节点编号涉及的节点对，结束处理。

[0197] 图 11A 和图 11B 是对从图 2 所例示的配对节点树中删除索引关键字“011010”的处理进行说明的图。

[0198] 图 11A 所示的配对节点树的节点对 201f 以下的节点省略了记载。删除对象的索引关键字“011010”被存储在作为暂时存储区域的删除关键字 270 内。探索路径堆栈 280

的堆栈指针指定数组编号 221c+1，表示检索处理完成。图中被粗框包围的节点是在检索处理中所被搜索过的节点，其数组编号从根节点 210a 的数组编号到叶节点 211d 的数组编号被堆栈在探索路径堆栈 280 中。

[0199] 在使用删除关键字的检索处理中，首先开始取得根节点 210a 的数组编号 220，将其存储在探索路径堆栈 280 内。由于根节点 210a 的鉴别比特位置 230a 是 0，删除关键字的比特位置 0 的比特值是 0，因而代表节点编号 220a+0 ＝ 220a 被存储在探索路径堆栈 280 内。

[0200] 于是读出节点 210b，由于鉴别比特位置 230b 是 1，删除关键字的比特位置 1 的比特值是 1，因而代表节点编号 220b+1 被存储在探索路径堆栈 280 内。

[0201] 然后读出节点 211c，由于鉴别比特位置 231c 是 2，删除关键字的比特位置 2 的比特值是 1，因而代表节点编号 221c+1 被存储在探索路径堆栈 280 内。存储在数组编号是 221c+1 的数组要素内的节点 211d 的节点类别 261d 是 1，表示是叶节点，因而当取出索引关键字 251d 时，其值是“011010”，与存储在删除关键字 270 内的删除对象的索引关键字一致。

[0202] 在图 11A 所示的状态下，读出与具有删除对象的索引关键字的节点 211d 成对的节点 210d 的内容，该内容被写入到在使探索路径堆栈 280 的堆栈指针后退 1 个的位置中所存储的数组编号 220b+1 的数组要素（节点 221c）内。之后删除节点对 201d。删除了节点对的数组要素为空，可再利用。

[0203] 图 11B 所示的配对节点树是删除处理结束后的配对节点树。在节点 211c 的节点类别 261c、鉴别比特位置 231c、代表节点编号 221c 内，如括弧所示，直接存储有在节点 210d 内存储的值。

[0204] 下面，根据图 12A、图 12B 和图 13 ～图 16，列举具体例来进一步说明插入处理。

[0205] 图 12A 所示的是具有比特序列“0100”、“0001”、“0000”作为索引关键字的配对节点树以及保持了此后要插入的插入关键字“0011”的暂时存储区域 1250。图示的树由节点对 1201a、1201b、1201c 构成。

[0206] 节点对 1201a 的代表节点是根节点 1210a，在鉴别比特位置保持有 1。节点对 1201a 的下位的节点对 1201b 的代表节点 1210b 是分支节点，在鉴别比特位置保持有 3，与代表节点 1210b 成对的节点 1211b 是叶节点，保持有索引关键字“0100”。作为分支节点的节点 1210b 链接到节点对 1201c。

[0207] 构成节点对 1201c 的节点 1210c、1211c 都是叶节点，分别保持了索引关键字“0000”、“0001”。

[0208] 图 12B 是示出插入了插入关键字“0011”的配对节点树。新的节点对 1201d 被插入在节点对 1201b 与节点对 1201c 之间。

[0209] 将图 12A 和图 12B 进行比较，所插入的节点 1210d 的内容是插入前的节点 1210b 的内容，插入后的节点 1210b 的鉴别比特位置从 3 变化为 2。

[0210] 以下，适当参照图 5 ～图 7，根据图 13 ～图 16 对图 12A 和图 12B 所例示的插入处理进行说明。在图 13 ～图 16 中粗线所示的处理流程是依照图 12A 和图 12B 的例示的处理流程。并且，S501 等的符号表示该部分的处理对应于图 5 ～图 7 记载的步骤 S501 等中的处理。

[0211] 图 13 是说明图 5 所示的插入处理的前段即检索处理中的处理块的流程的图。

[0212] 如图 13 所示，在最初的处理块中进行根节点设定。通过取得根节点的数组编号来设定根节点的位置。将所取得的数组编号 1210a 存储在探索路径堆栈内。另外，这里，表示数组编号的符号兼用作表示节点的符号。

[0213] 然后，在由步骤 S503、步骤 S504 和步骤 S505 构成的节点读出 / 判定块中判定节点 1210a 是分支节点，进行分支处理。

[0214] 在分支处理中，将插入关键字“0011”的第 1 比特的“0”与存储在节点 1210a 的代表节点编号内的数组编号 1201b 相加来计算链接目的地的节点位置。另外，这里，利用链接目的地的节点对的符号来表示存储在代表节点编号内的数组编号。计算出的节点位置 1210b 被存储在探索路径堆栈内。

[0215] 根据在分支处理中计算出的节点位置再次进行节点读出 / 判定处理，读出节点 1210b，判定为是分支节点。

[0216] 因此再次进行分支处理，将插入关键字“0011”的第 3 比特的“1”与存储在节点 1210b 的代表节点编号内的数组编号 1201c 相加来计算链接目的地的节点位置。计算出的节点位置 1211c 被存储在探索路径堆栈内。

[0217] 在后面的节点读出 / 判定处理中读出节点 1211c，判定为是叶节点。

[0218] 因此处理分支到叶处理，进行节点 1211c 的索引关键字“0001”和插入关键字“0011”的比特序列比较，如果一致则插入失败，如果不一致则进到后面的处理。

[0219] 另外，虚线箭头指向处所记载的符号 D513 等的出口标记表示与虚线的一方连接的数据值在相同符号的入口点中被使用。D513 中的 513 表示在步骤 S513 中被使用。以上，在图 14 ～图 16 中也是一样。

[0220] 图 14 是对准备图 6 所示的要插入的节点对用的数组要素的处理块的流程进行说明的图。

[0221] 在空节点对取得块中，从数组中取得连续区域的 2 个空数组要素的数组编号涉及的代表节点编号 1201d。

[0222] 然后在节点存储位置计算块中，进行插入关键字“0011”和在步骤 S510 所获得的索引关键字“0001”的大小比较，决定在空节点对取得块中所取得的 2 个数组要素中的哪个数组要素内存储保持插入关键字的叶节点。由于比较结果是插入关键字大，因而在代表节点编号 1201d 加上 1 后的数组编号的数组要素内存储保持插入关键字的叶节点作为节点 1211d。

[0223] 图 15 是说明图 7 所示的处理中的前半部分，即求出要插入的节点对在配对节点树上的位置的处理流程的图。

[0224] 在差分比特位置检测块中，运算插入关键字“0011”和在步骤 S510 所获得的索引关键字“0001”的各比特的“异或”值“0010”，求出差分比特位置是第 2 比特。

[0225] 在插入节点位置检索块中，一边将存储在探索路径堆栈的指针所指定的数组编号的数组要素内的节点的鉴别比特位置和差分比特位置进行比较，一边使指针依次后退直到鉴别比特位置成为上位。在途中指针指定了根节点的数组编号的情况下，插入位置为根节点的紧下方。

[0226] 在开始插入节点位置的探索之前的阶段中，由于探索路径堆栈的指针指定数组编号 1211c，因而使指针后退 1 个开始插入节点位置的探索处理，取出节点 1210b 的存储位置，

读出节点 1210b。由于节点 1210b 的鉴别比特位置是 3，与差分比特位置 2 相比为下位，节点 1210b 不是根节点，因而再继续探索插入节点位置。

[0227] 当再使指针后退 1 个时，取出节点 1210a 的存储位置，读出节点 1210a。由于节点 1210a 的鉴别比特位置是 1，与差分比特位置 2 相比为上位，因而使指针的位置前进 1 个来进行插入节点位置读出块的处理。

[0228] 当进行插入位置读出时，探索路径堆栈的指针指定数组编号 1210b，因而作为插入节点的插入位置，读出数组编号 1210b。

[0229] 图 16 是说明图 7 所示的处理中的后半部分，即在各节点设定数据来完成插入处理的处理流程的图。

[0230] 在叶节点装配 / 写入块中，装配在节点类别内存储 1、在索引关键字内存储插入关键字“0011”的叶节点，将其存储在图 14 所示的节点存储位置计算块中计算出的数组编号 1211d 的数组要素内。

[0231] 然后，在插入位置的节点读出 / 写入块中，读出在图 15 所示的插入节点位置读出块中所读出的数组编号的节点 1210b，将其内容存储在图 14 所示的节点存储位置计算块中计算出的数组编号 1210d 的数组要素内。

[0232] 然后，在分支节点装配 / 写入框中，装配进如下的分支节点 :在节点类别内存储 0，在鉴别比特位置内存储在图 15 所示的差分比特位置检测块中检测出的差分比特位置 2，在代表节点编号内存储在图 14 所示的空节点对取得块中所取得的代表节点编号 1201d。然后，将所装配的分支节点写入到在图 15 所示的插入节点位置读出块中所读出的数组编号 1210b 的数组要素内。

[0233] 通过所述处理，在图 12A 所示的配对节点树内插入了插入关键字“0011”，完成图 12B 所示的配对节点树。

[0234] 以上详细说明了用于实施本发明的最佳方式，然而对本领域技术人员来说，本发明的实施方式显然不限于此而能进行各种变形。

[0235] 并且，显然，本发明的比特序列检索装置可利用存储配对节点树的存储单元和使计算机执行图 4 所示的处理的程序来构建在计算机上。

[0236] 而且，显然，能利用使计算机执行图 8 和图 5 ～图 7 所示的索引关键字插入处理及其等同物的程序来实现本发明的索引关键字插入方法，并能利用使计算机执行图 9 和图 10 所示的索引关键字删除处理及其等同物的程序来实现本发明的索引关键字删除方法。然后，利用这些程序在计算机上实现分支节点和叶节点的识别单元、根据分支节点的鉴别比特位置链接到链接目的地的节点对的哪个节点的单元等。

[0237] 因此，所述程序以及存储有程序的计算机可读取的存储介质包含在本发明的实施方式内。而且，本发明的配对节点树的数据结构也包含在本发明的实施方式内。

[0238] 通过使用以上详细说明的本发明提供的全新的数据结构即配对节点树，可进行更高速的比特序列数据的检索，而且也能容易地执行比特序列数据的追加删除。

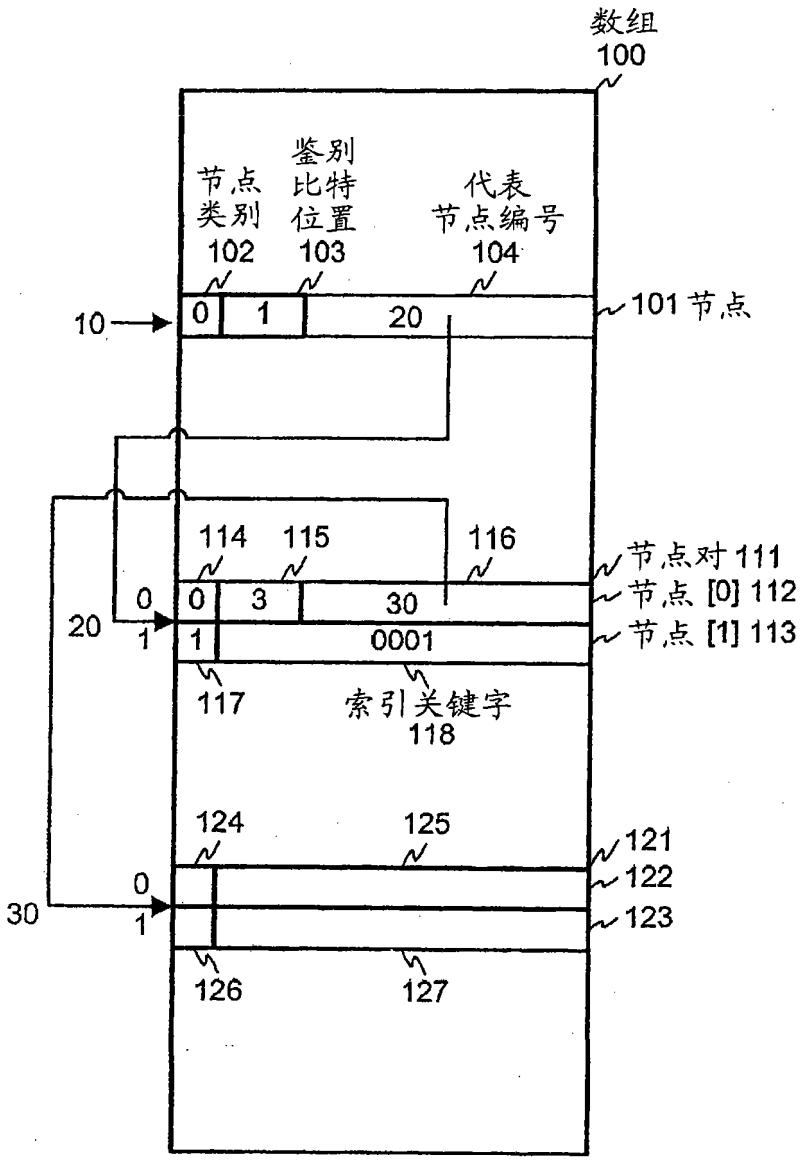

数组
100
节点
类别
102
鉴别
比特
位置
103
代表
节点编号
104
101 节点
10
0
1
20
114
115
116
节点对 111
节点 [0] 112
节点 [1] 113
0
3
30
20
1
0001
117
索引关键字
118
124
125
121
122
123
30
126
127

图 1

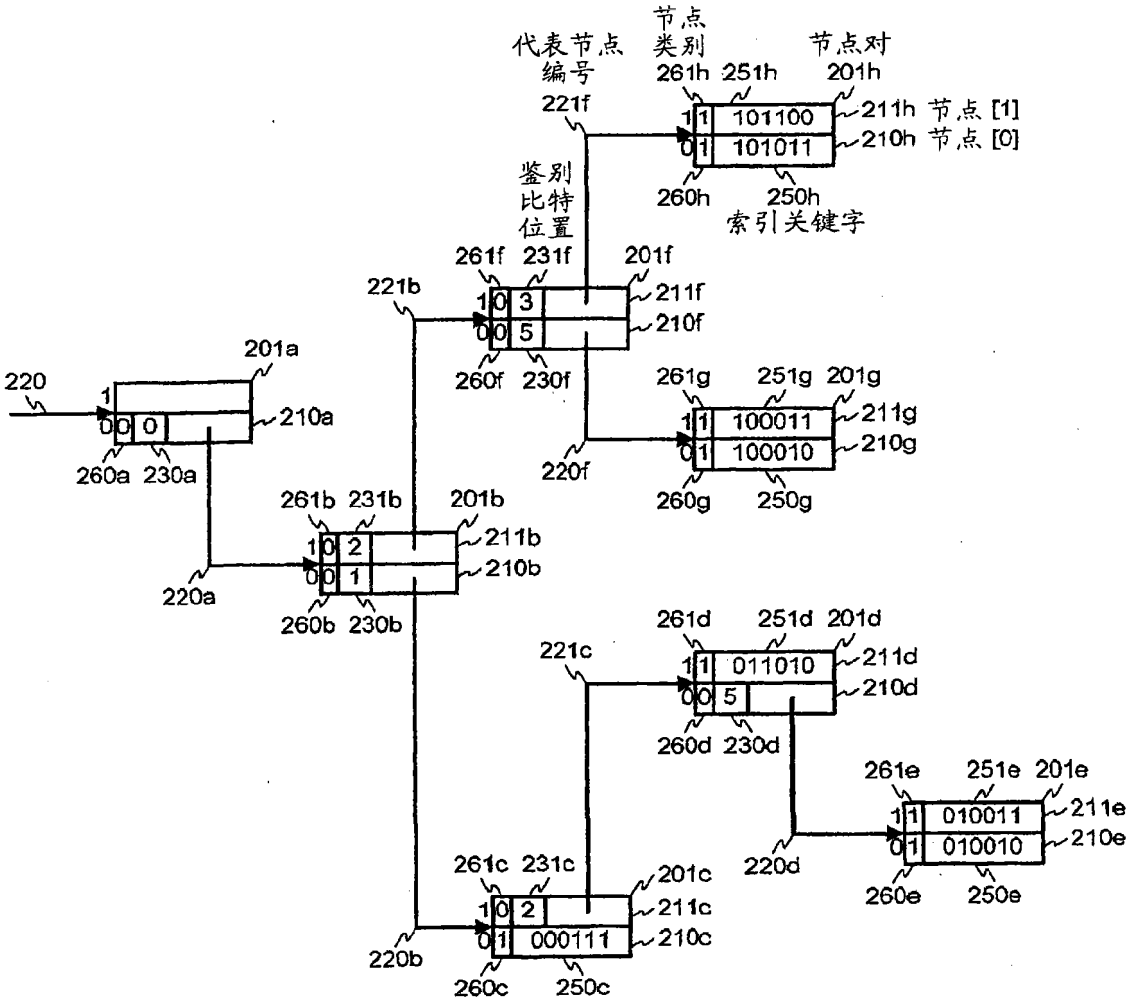

节点
代表节点
编号
类别
节点对
261h 251h 201h
221f
101100
211h 节点 [1]
101011
210h 节点 [0]
260h
250h
索引关键字
鉴别
比特
位置
261f 231f 201f
221b
211f
210f
260f 230f
261g 251g 201g
100011
211g
100010
210g
220f
260g 250g
201a
220
210a
260a 230a
261b 231b 201b
211b
210b
220a
260b 230b
261d 251d 201d
221c
011010
211d
210d
260d 230d
261e 251e 201e
010011
211e
010010
210e
220d
260e 250e
261c 231c 201c
211c
210c
000111
220b
260c 250c

图 2

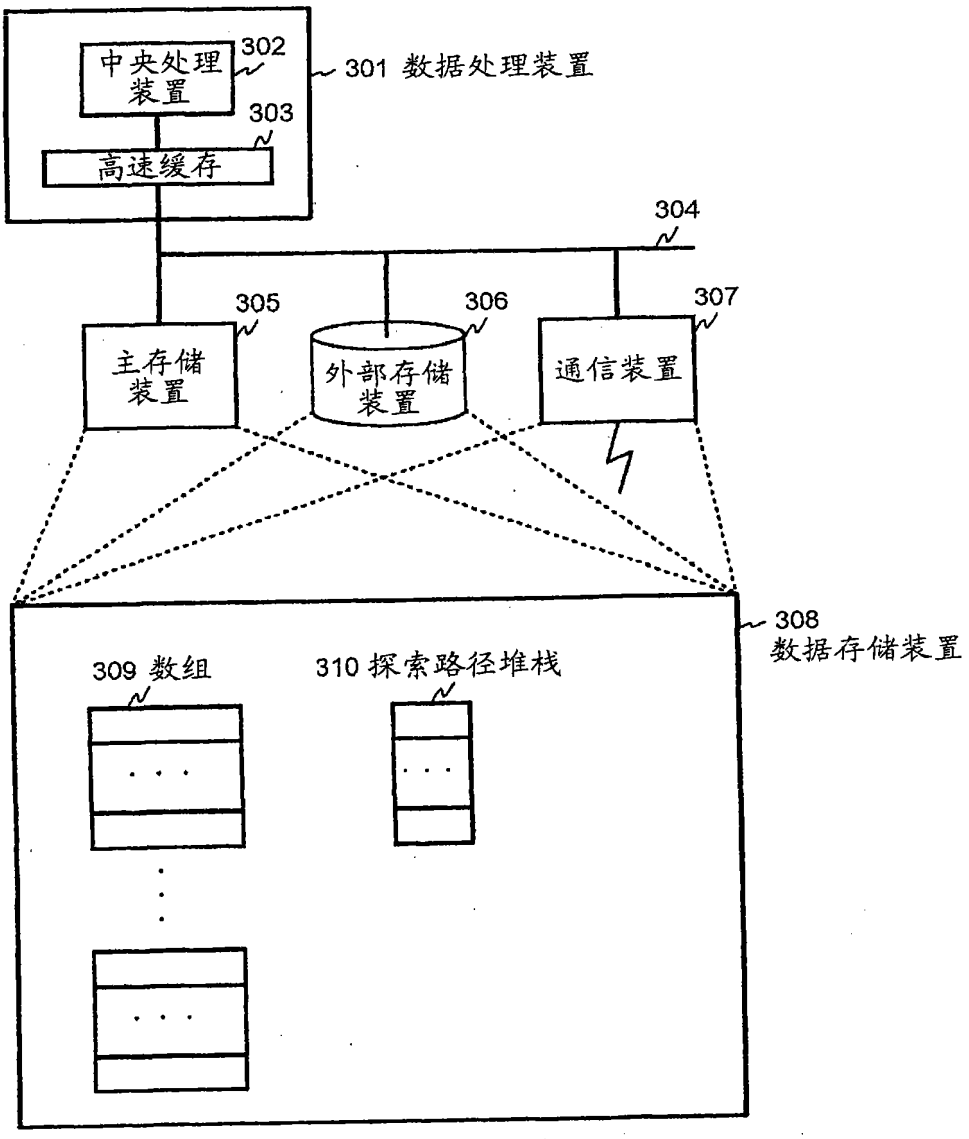

中央处理装置
302
301 数据处理装置
303
高速缓存
304
305
306
307
主存储装置
外部存储装置
通信装置
308
数据存储装置
309 数组
310 探索路径堆栈

图 3

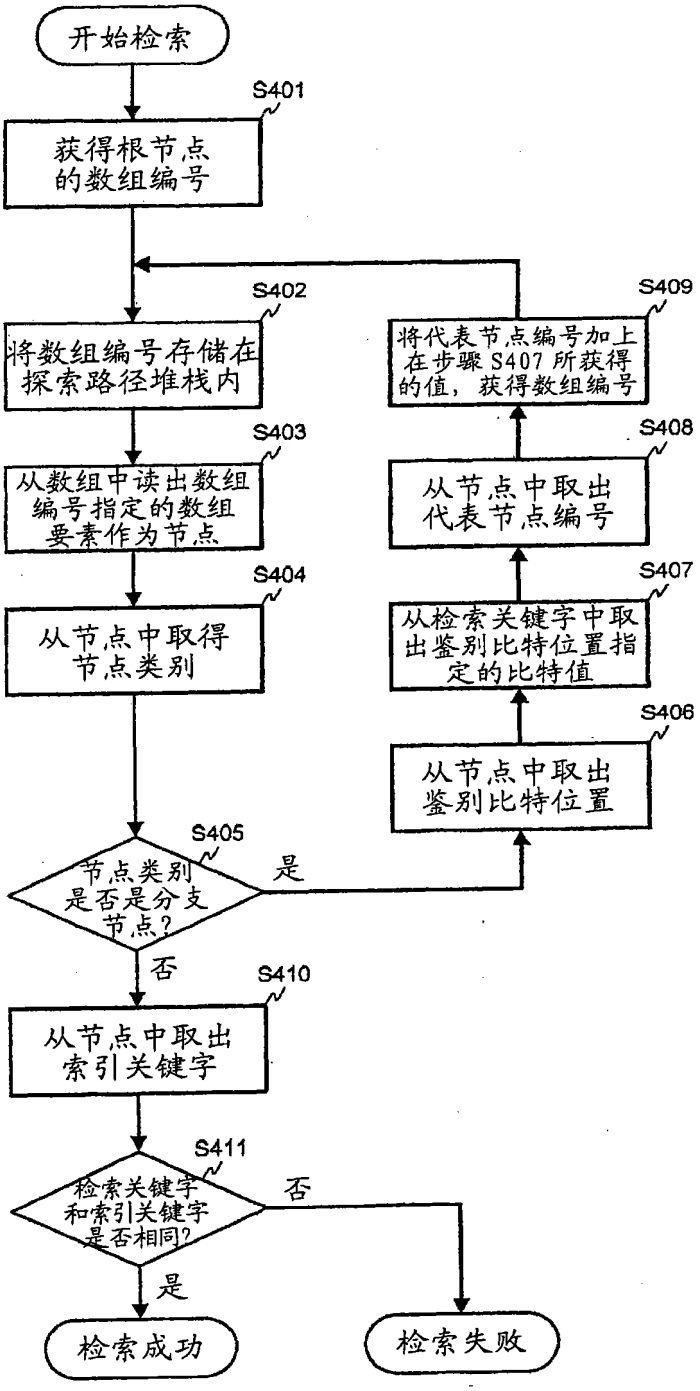

开始检索
S401
获得根节点的数组编号
S402
将数组编号存储在探索路径堆栈内
S403
从数组中读出数组编号指定的数组要素作为节点
S404
从节点中取得节点类别
S405
节点类别是否是分支节点?
是
否
S406
从节点中取出鉴别比特位置
S407
从检索关键字中取出鉴别比特位置指定的比特值
S408
从节点中取出代表节点编号
S409
将代表节点编号加上在步骤S407所获得的值，获得数组编号
S410
从节点中取出索引关键字
S411
检索关键字和索引关键字是否相同?
否
是
检索成功
检索失败

图 4

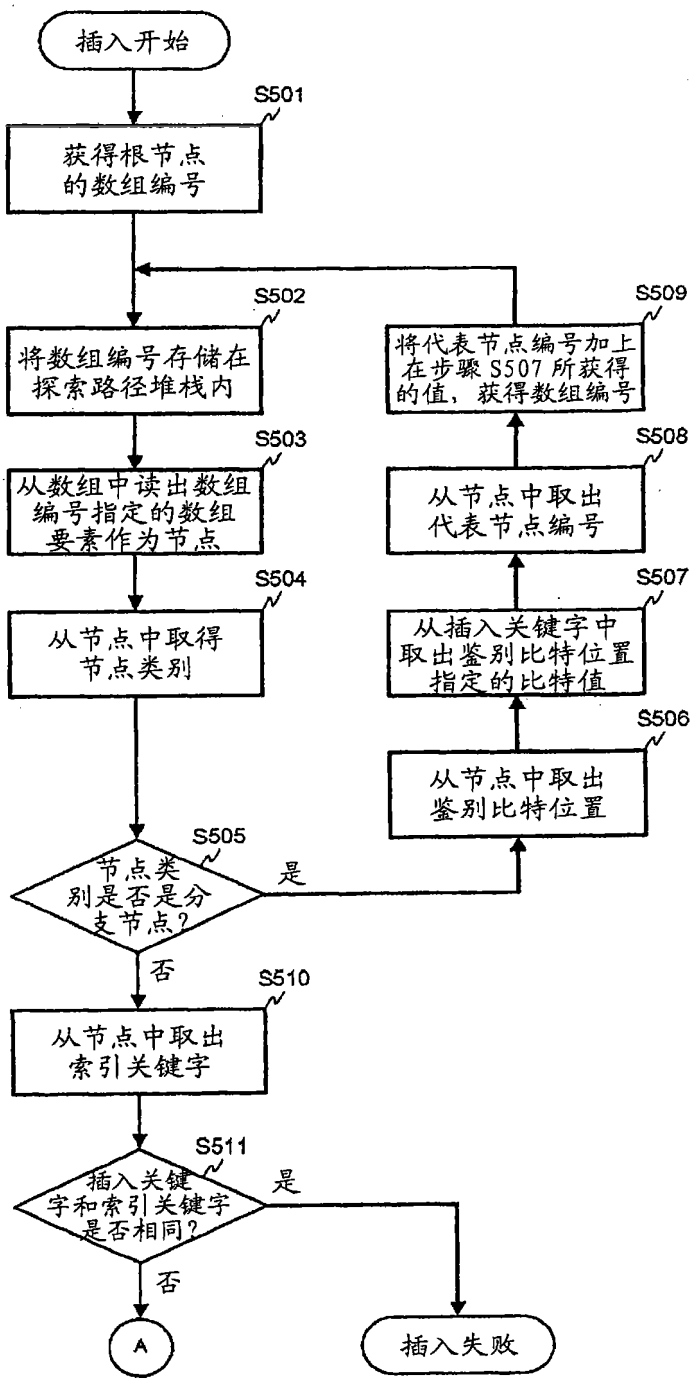

插入开始
S501
获得根节点的数组编号
S502
将数组编号存储在探索路径堆栈内
S503
从数组中读出数组编号指定的数组要素作为节点
S504
从节点中取得节点类别
S505
节点类别是否是分支节点?
是
否
S506
从节点中取出鉴别比特位置
S507
从插入关键字中取出鉴别比特位置指定的比特值
S508
从节点中取出代表节点编号
S509
将代表节点编号加上在步骤 S507 所获得的值，获得数组编号
S510
从节点中取出索引关键字
S511
插入关键字和索引关键字是否相同?
是
否
A
插入失败

图 5

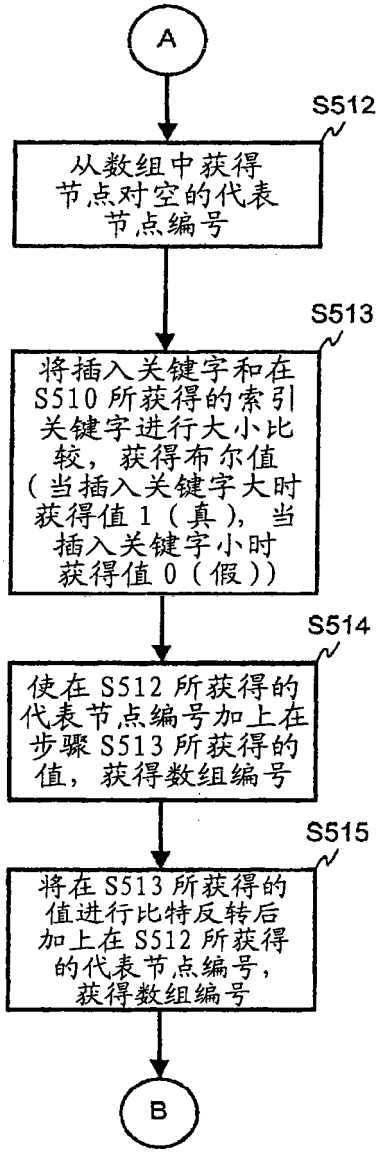

A
S512
从数组中获得节点对空的代表节点编号
S513
将插入关键字和在S510所获得的索引关键字进行大小比较，获得布尔值（当插入关键字大时获得值1（真），当插入关键字小时获得值0（假））
S514
使在S512所获得的代表节点编号加上在步骤S513所获得的值，获得数组编号
S515
将在S513所获得的值进行比特反转后加上在S512所获得的代表节点编号，获得数组编号
B

图 6

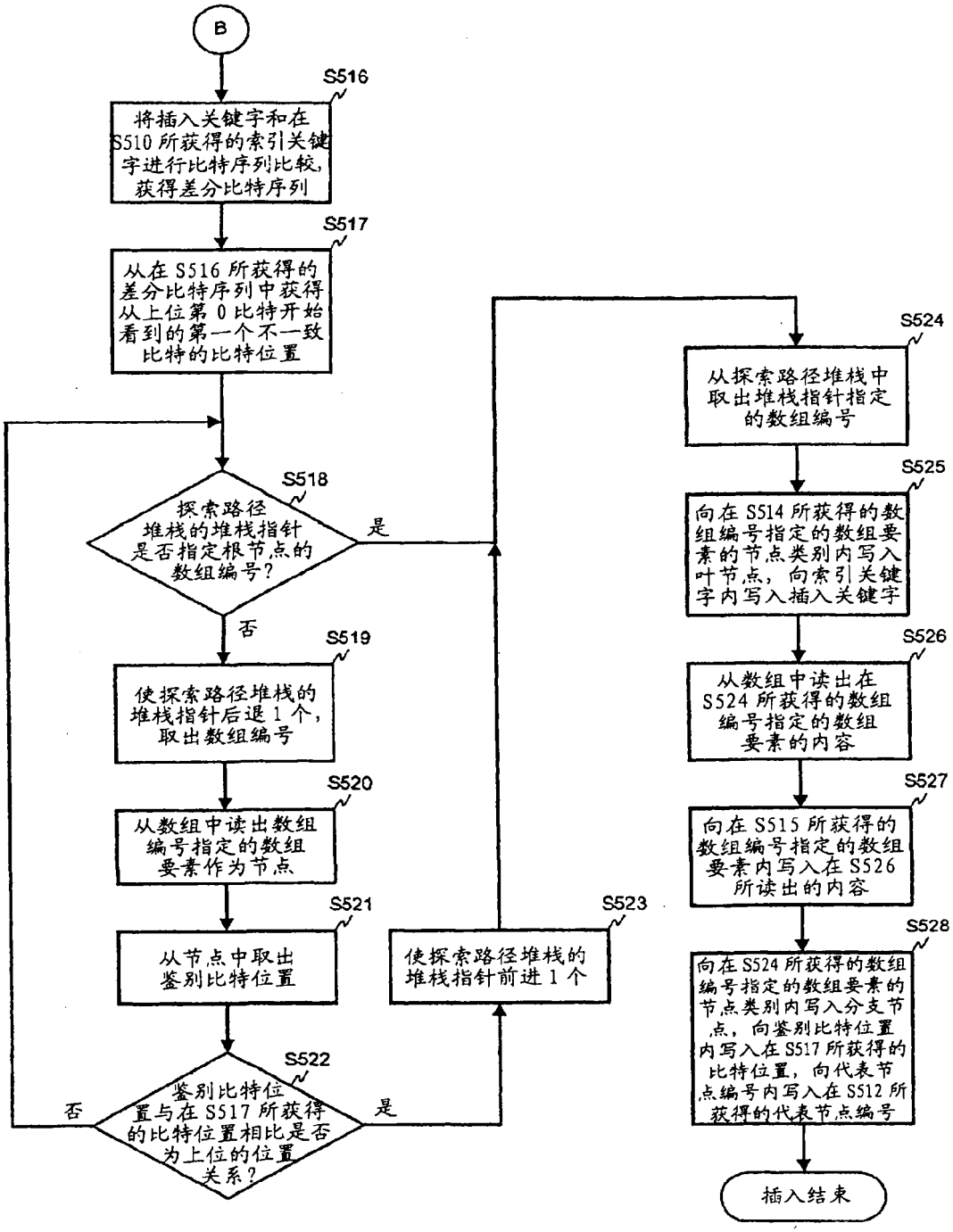

B
S516
将插入关键字和在S510所获得的索引关键字进行比特序列比较，获得差分比特序列
S517
从在S516所获得的差分比特序列中获得从上位第0比特开始看到的第一个不一致比特的比特位置
S518
探索路径堆栈的堆栈指针是否指定根节点的数组编号？
是
否
S519
使探索路径堆栈的堆栈指针后退1个，取出数组编号
S520
从数组中读出数组编号指定的数组要素作为节点
S521
从节点中取出鉴别比特位置
S522
鉴别比特位置与在S517所获得的比特位置相比是否为上位的位置关系？
否
是
S523
使探索路径堆栈的堆栈指针前进1个
S524
从探索路径堆栈中取出堆栈指针指定的数组编号
S525
向在S514所获得的数组编号指定的数组要素的节点类别内写入叶节点，向索引关键字内写入插入关键字
S526
从数组中读出在S524所获得的数组编号指定的数组要素的内容
S527
向在S515所获得的数组编号指定的数组要素内写入在S526所读出的内容
S528
向在S524所获得的数组编号指定的数组要素的节点类别内写入分支节点，向鉴别比特位置内写入在S517所获得的比特位置，向代表节点编号内写入在S512所获得的代表节点编号
插入结束

图 7

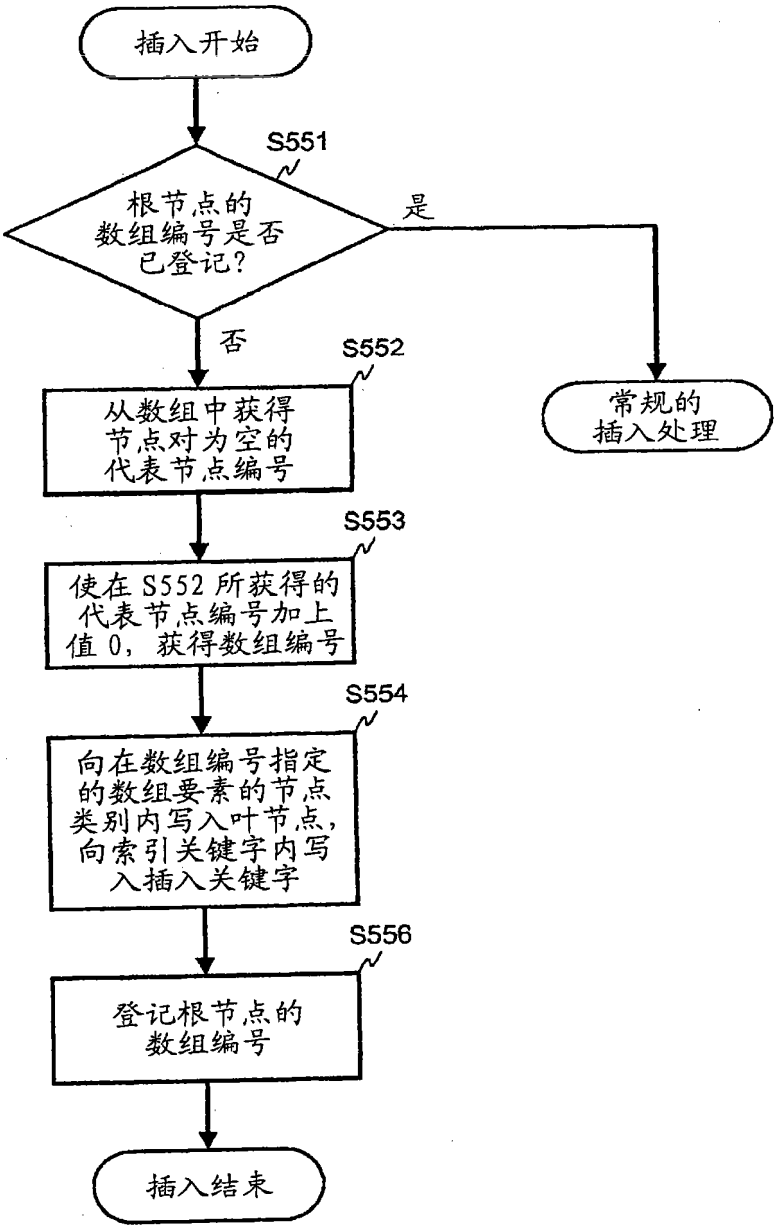

插入开始
S551
根节点的
数组编号是否
已登记?
是
否
常规的
插入处理
S552
从数组中获得
节点对为空的
代表节点编号
S553
使在 S552 所获得的
代表节点编号加上
值 0，获得数组编号
S554
向在数组编号指定
的数组要素的节点
类别内写入叶节点，
向索引关键字内写
入插入关键字
S556
登记根节点的
数组编号
插入结束

图 8

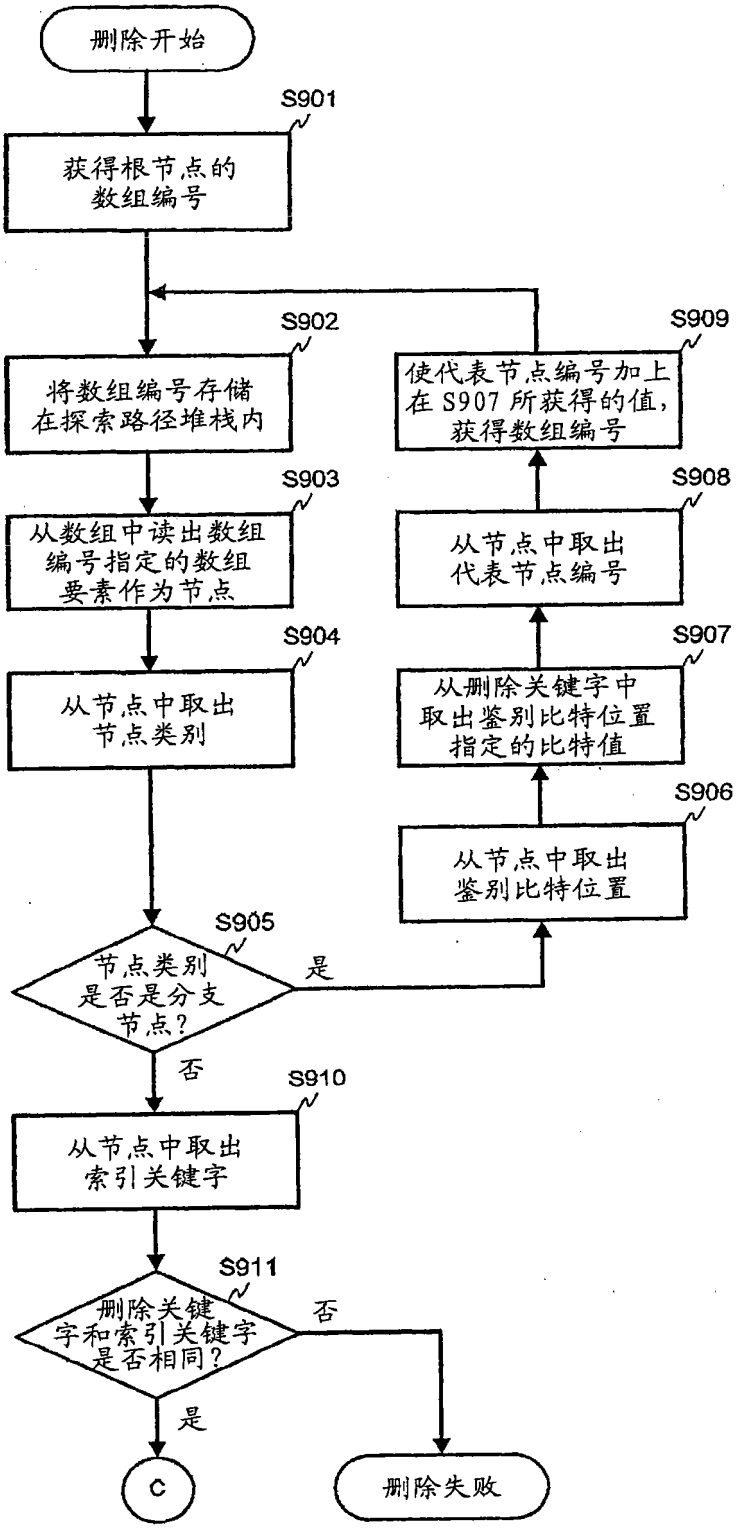

删除开始
S901
获得根节点的数组编号
S902
将数组编号存储在探索路径堆栈内
S903
从数组中读出数组编号指定的数组要素作为节点
S904
从节点中取出节点类别
S905
节点类别是否是分支节点？
是
否
S906
从节点中取出鉴别比特位置
S907
从删除关键字中取出鉴别比特位置指定的比特值
S908
从节点中取出代表节点编号
S909
使代表节点编号加上在 S907 所获得的值，获得数组编号
S910
从节点中取出索引关键字
S911
删除关键字和索引关键字是否相同？
否
是
C
删除失败

图 9

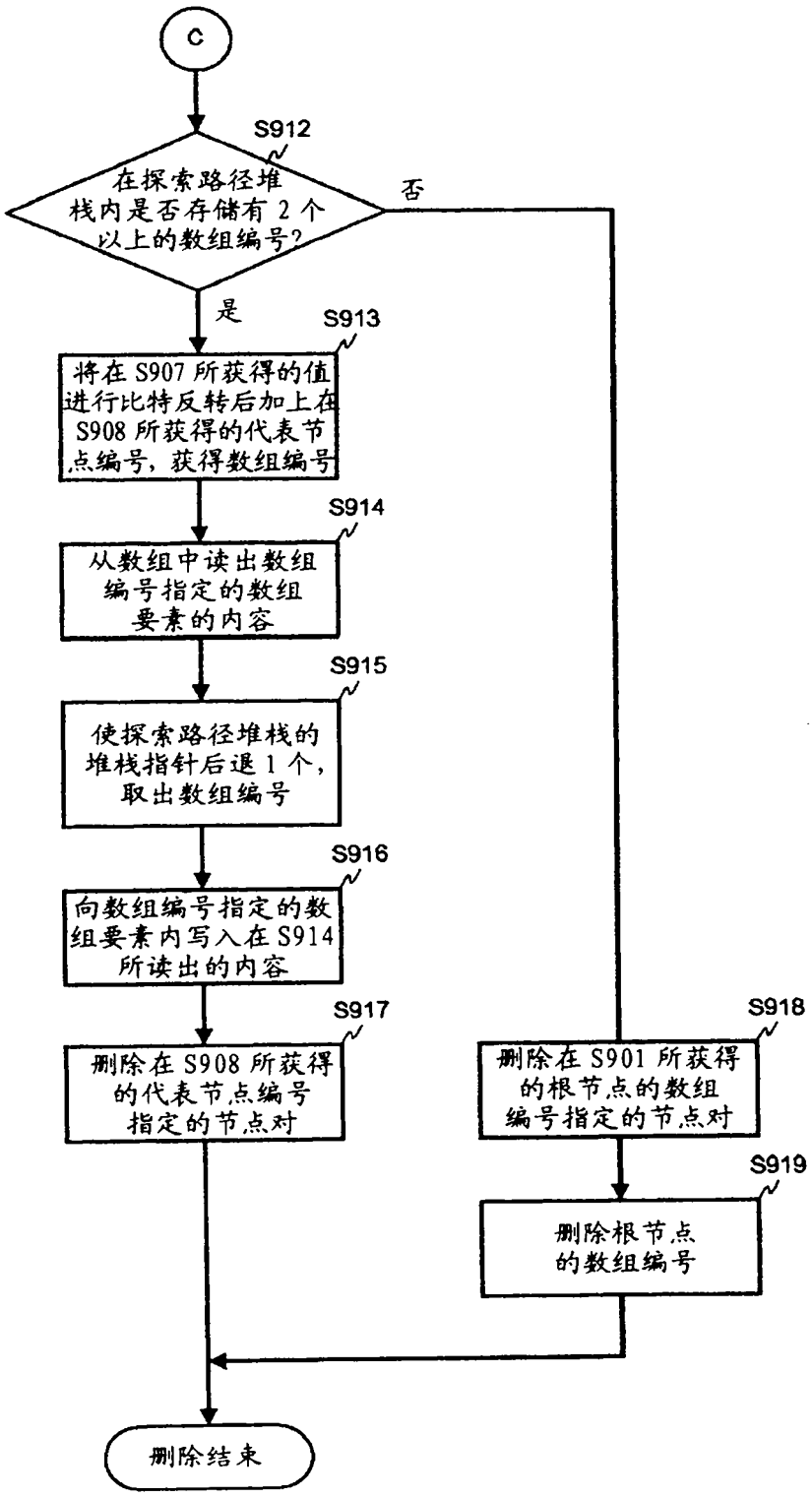

C
S912
在探索路径堆栈内是否存储有 2 个以上的数组编号?
否
是
S913
将在 S907 所获得的值进行比特反转后加上在 S908 所获得的代表节点编号，获得数组编号
S914
从数组中读出数组编号指定的数组要素的内容
S915
使探索路径堆栈的堆栈指针后退 1 个，取出数组编号
S916
向数组编号指定的数组要素内写入在 S914 所读出的内容
S917
删除在 S908 所获得的代表节点编号指定的节点对
S918
删除在 S901 所获得的根节点的数组编号指定的节点对
S919
删除根节点的数组编号
删除结束

图 10

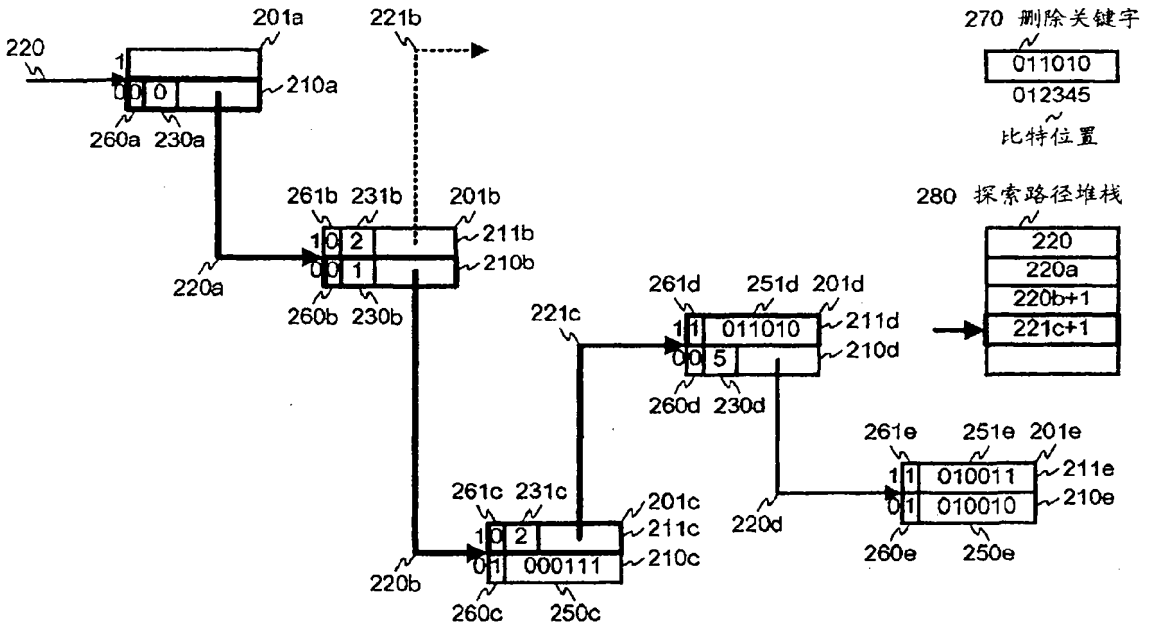

220
201a
221b
210a
260a 230a
270 删除关键字
011010
012345
比特位置
261b 231b 201b
211b
210b
220a
260b 230b
280 探索路径堆栈
220
220a
220b+1
221c+1
261d 251d 201d
221c
011010
211d
210d
260d 230d
261e 251e 201e
010011
211e
010010
210e
220d
260e 250e
261c 231c
201c
211c
210c
000111
220b
260c 250c

图 11A

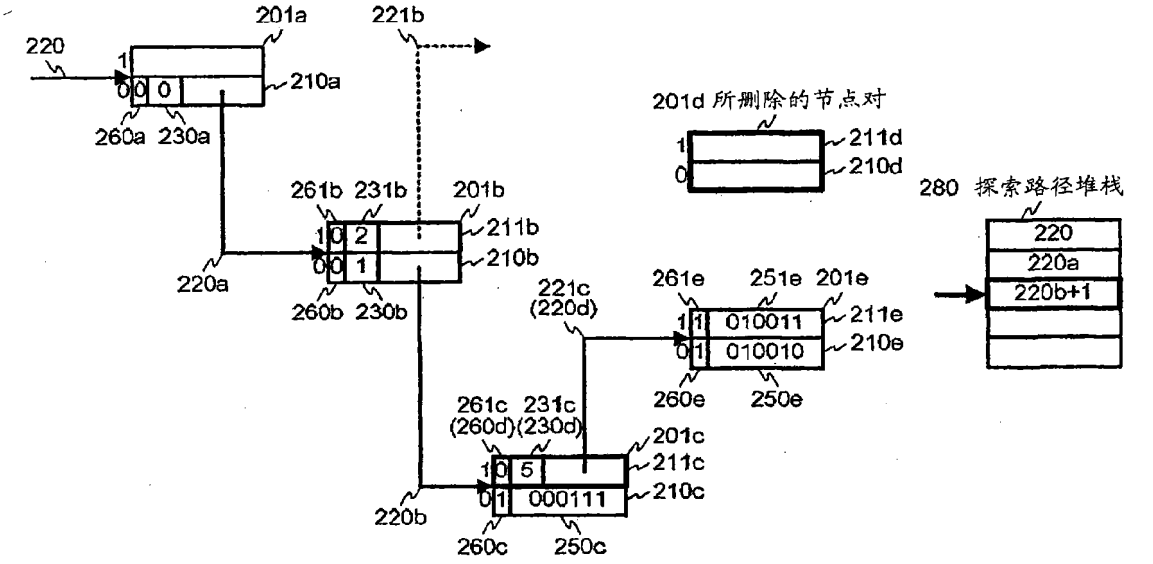

220
201a
221b
210a
260a 230a
201d 所删除的节点对
211d
210d
261b 231b 201b
211b
210b
220a
260b 230b
280 探索路径堆栈
220
220a
220b+1
221c
(220d)
261e 251e 201e
010011
211e
010010
210e
260e 250e
261c 231c
(260d)(230d)
201c
211c
210c
000111
220b
260c 250c

图 11B

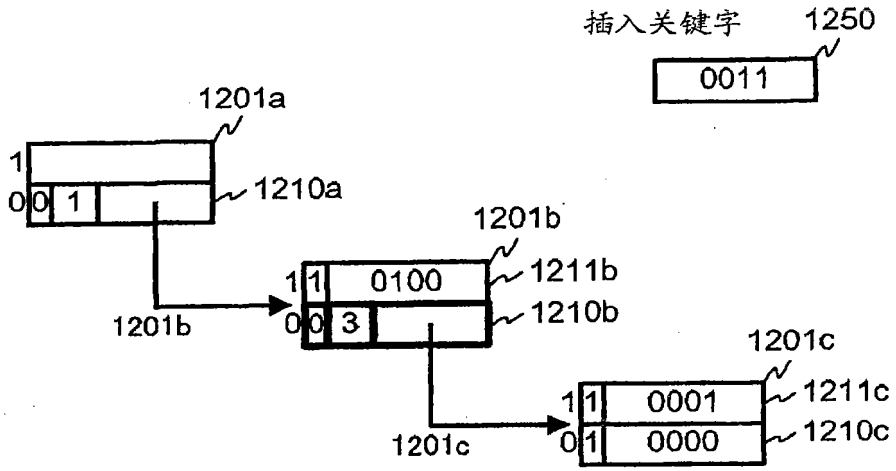

插入关键字
1250
0011
1201a
1
0 0 1
1210a
1201b
1 1 0100
1211b
0 0 3
1210b
1201b
1201c
1 1 0001
1211c
0 1 0000
1210c
1201c

图 12A

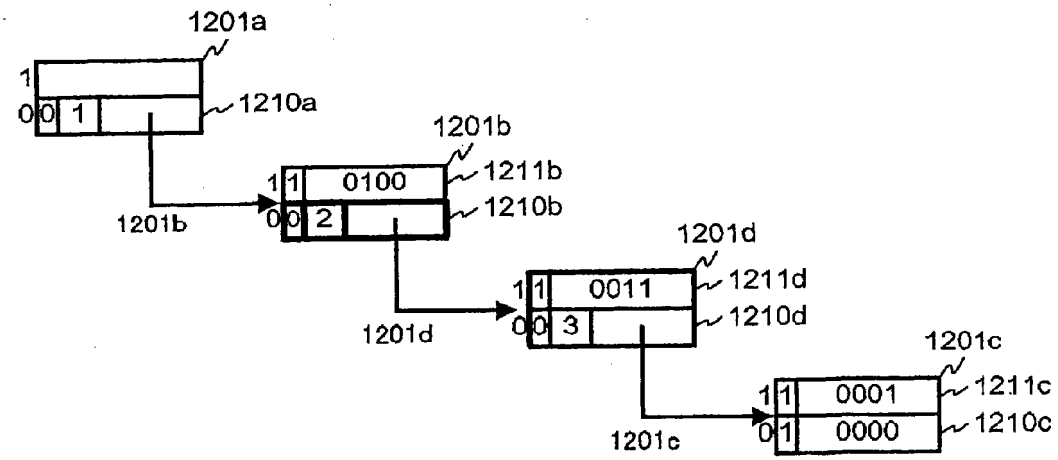

1201a
1
0 0 1
1210a
1201b
1 1 0100
1211b
0 0 2
1210b
1201b
1201d
1 1 0011
1211d
0 0 3
1210d
1201d
1201c
1 1 0001
1211c
0 1 0000
1210c
1201c

图 12B

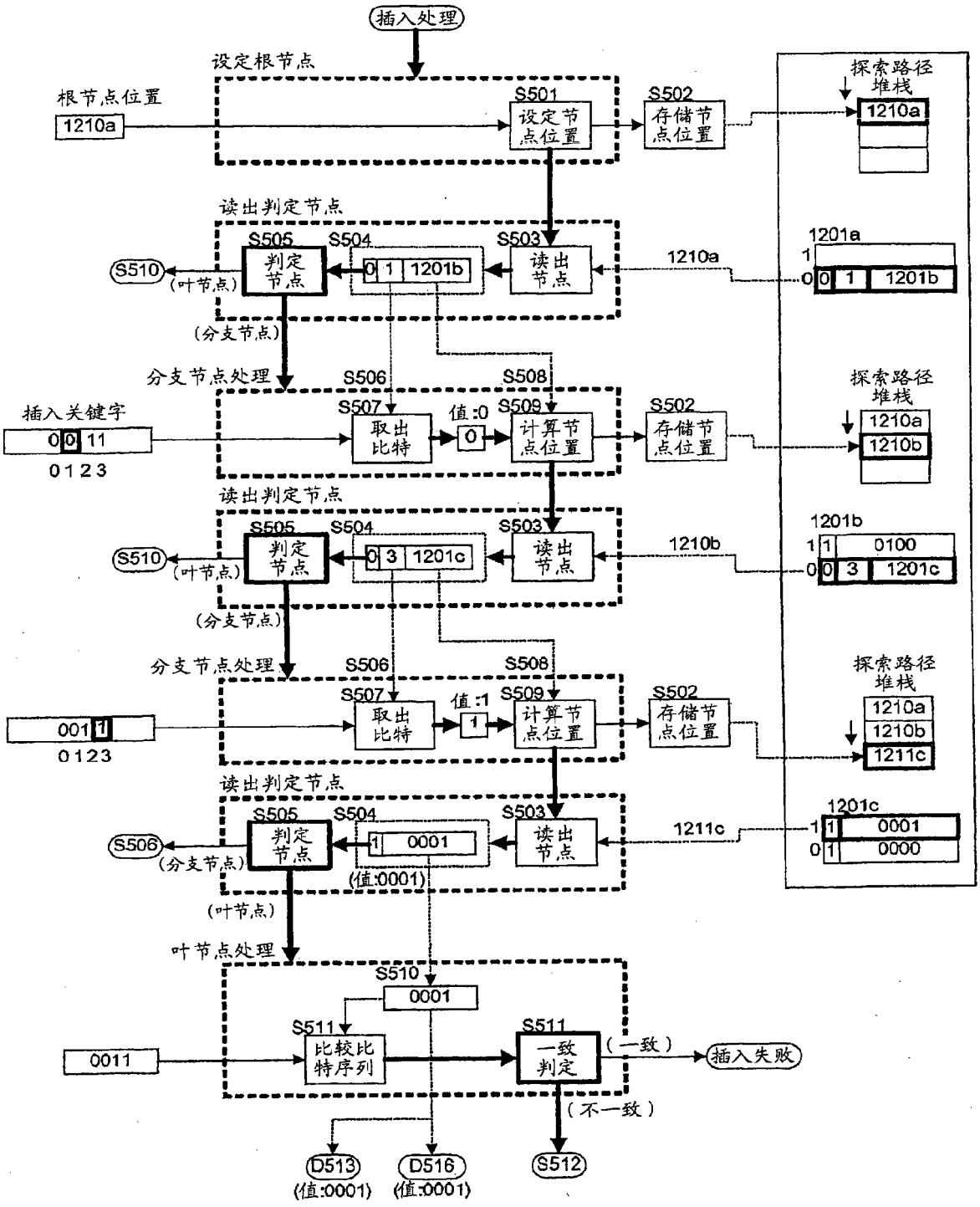

插入处理
设定根节点
根节点位置
1210a
S501
设定节点位置
S502
存储节点位置
探索路径堆栈
1210a
读出判定节点
S505
判定节点
S504
0 1 1201b
S503
读出节点
S510
(叶节点)
(分支节点)
1210a
1201a
1
0 0 1 1201b
分支节点处理
S506
S508
插入关键字
0 0 11
0 1 2 3
S507
取出比特
值:0
0
S509
计算节点位置
S502
存储节点位置
探索路径堆栈
1210a
1210b
读出判定节点
S505
判定节点
S504
0 3 1201c
S503
读出节点
S510
(叶节点)
(分支节点)
1210b
1201b
1 1 0100
0 0 3 1201c
分支节点处理
S506
S508
001 1
0 1 2 3
S507
取出比特
值:1
1
S509
计算节点位置
S502
存储节点位置
探索路径堆栈
1210a
1210b
1211c
读出判定节点
S505
判定节点
S504
1 0001
S503
读出节点
S506
(分支节点)
(值:0001)
1211c
1201c
1 1 0001
0 1 0000
(叶节点)
叶节点处理
S510
0001
S511
比较比特序列
S511
一致判定
(一致)
插入失败
0011
(不一致)
D513
(值:0001)
D516
(值:0001)
S512

图 13

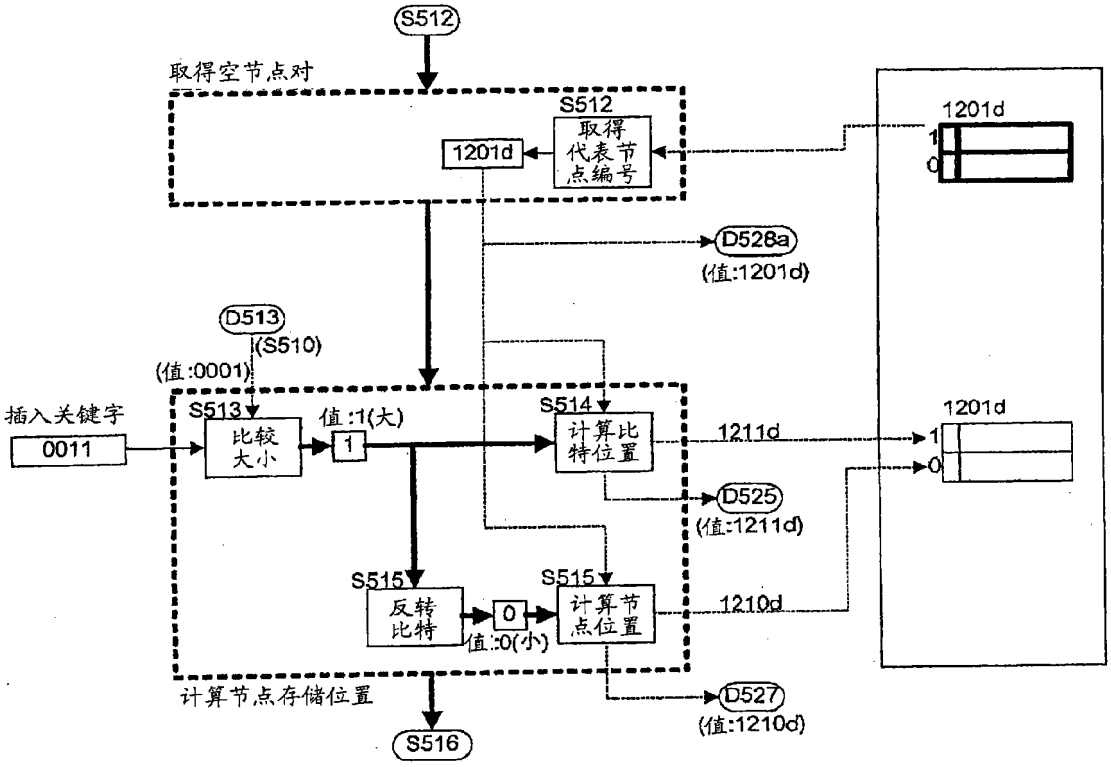

S512
取得空节点对
S512
取得代表节点编号
1201d
1201d
1
0
D528a
(值:1201d)
D513
(S510)
(值:0001)
插入关键字
0011
S513
比较大小
值:1(大)
1
S514
计算比特位置
1211d
1201d
1
0
D525
(值:1211d)
S515
反转比特
0
值:0(小)
S515
计算节点位置
1210d
计算节点存储位置
D527
(值:1210d)
S516

图 14

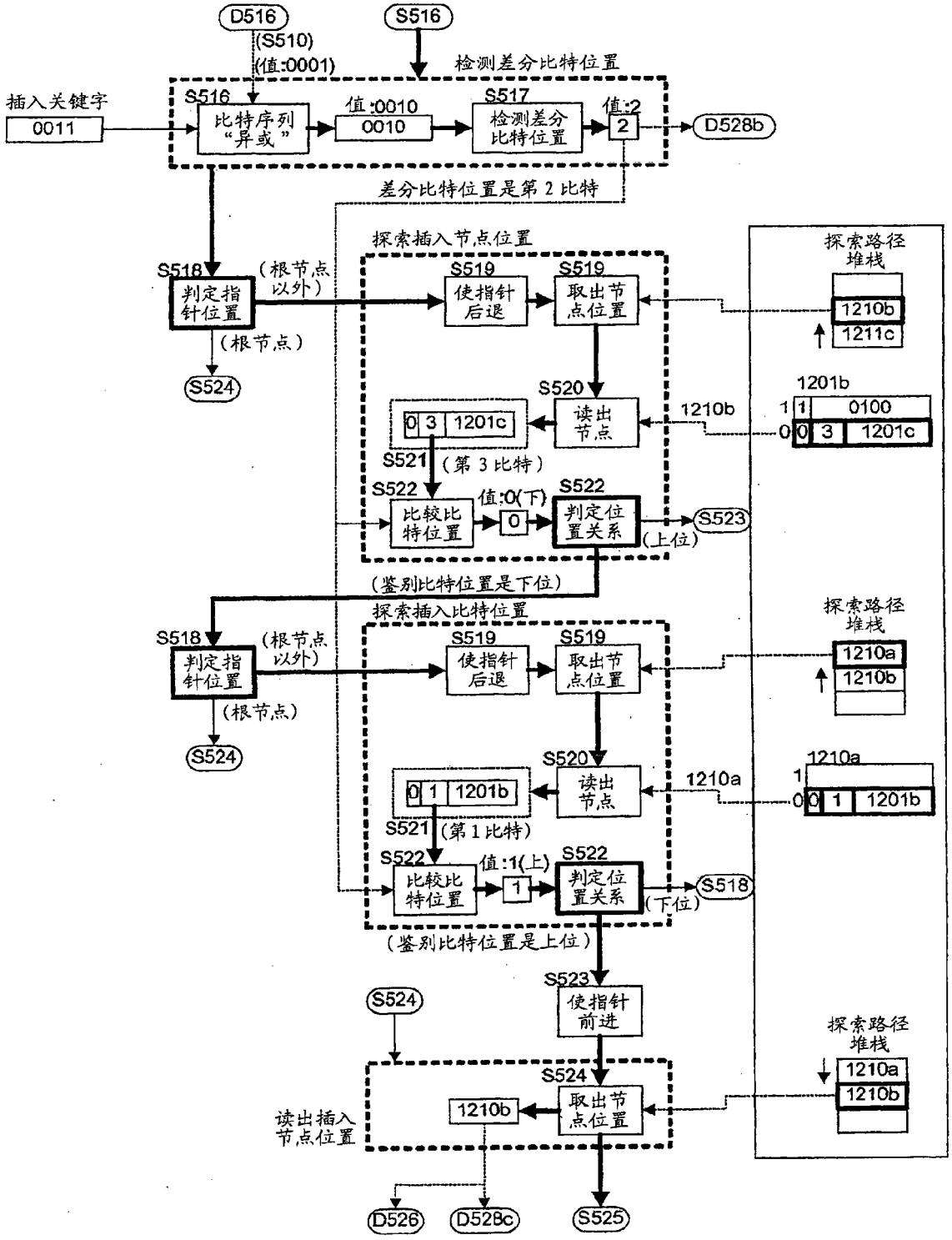

D516
(S510)
(值:0001)
S516
检测差分比特位置
插入关键字
0011
S516
比特序列
"异或"
值:0010
0010
S517
检测差分
比特位置
值:2
2
D528b
差分比特位置是第 2 比特
探索插入节点位置
探索路径
堆栈
S518
判定指
针位置
(根节点
以外)
(根节点)
S524
S519
使指针
后退
S519
取出节
点位置
1210b
1211c
S520
读出
节点
1210b
1201b
1 1 0100
0 0 3 1201c
0 3 1201c
S521
(第 3 比特)
S522
比较比
特位置
值:0(下)
0
S522
判定位
置关系
S523
(上位)
(鉴别比特位置是下位)
探索插入比特位置
探索路径
堆栈
S518
判定指
针位置
(根节点
以外)
(根节点)
S524
S519
使指针
后退
S519
取出节
点位置
1210a
1210b
S520
读出
节点
1210a
1210a
1
0 0 1 1201b
0 1 1201b
S521
(第 1 比特)
S522
比较比
特位置
值:1(上)
1
S522
判定位
置关系
S518
(下位)
(鉴别比特位置是上位)
S523
使指针
前进
S524
S524
取出节
点位置
探索路径
堆栈
1210a
1210b
1210b
读出插入
节点位置
D526
D528c
S525

图 15

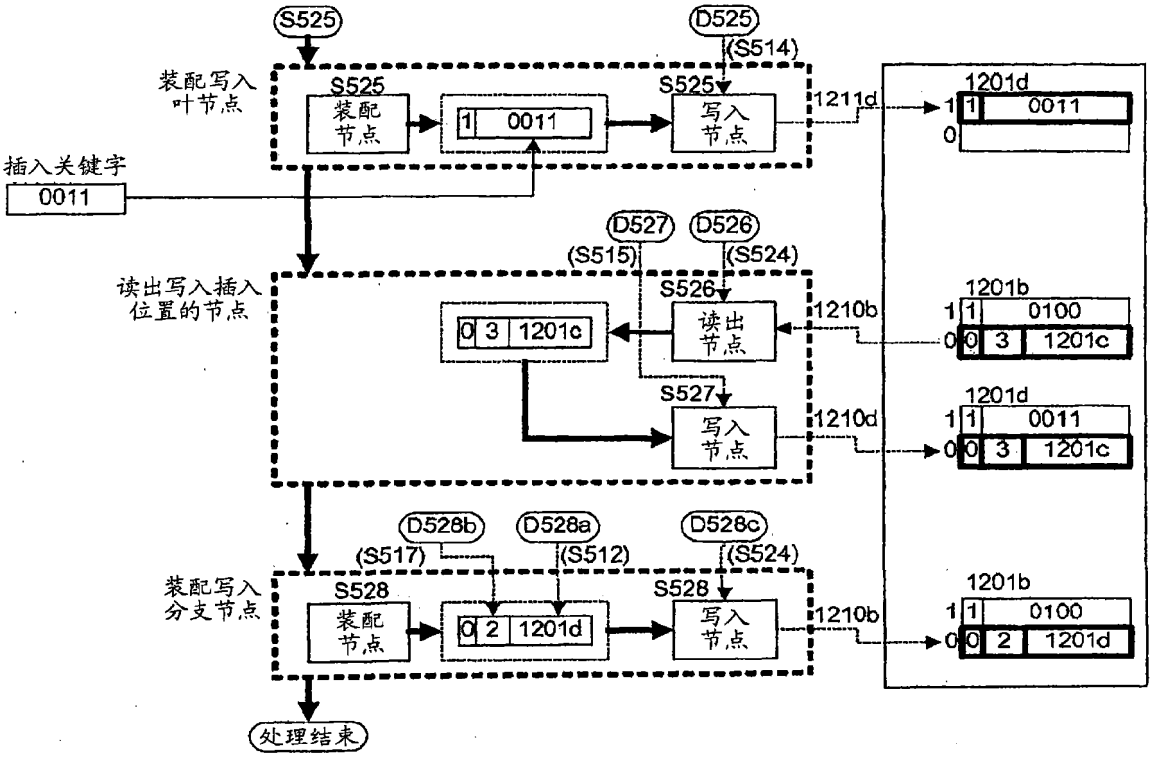

S525
D525
(S514)
装配写入
叶节点
S525
装配
节点
1 0011
S525
写入
节点
1211d
1201d
1 1 0011
0
插入关键字
0011
D527
D526
(S515)
(S524)
读出写入插入
位置的节点
S526
0 3 1201c
读出
节点
1210b
1201b
1 1 0100
0 0 3 1201c
S527
写入
节点
1210d
1201d
1 1 0011
0 0 3 1201c
D528b
D528a
D528c
(S517)
(S512)
(S524)
装配写入
分支节点
S528
装配
节点
0 2 1201d
S528
写入
节点
1210b
1201b
1 1 0100
0 0 2 1201d
处理结束

图 16

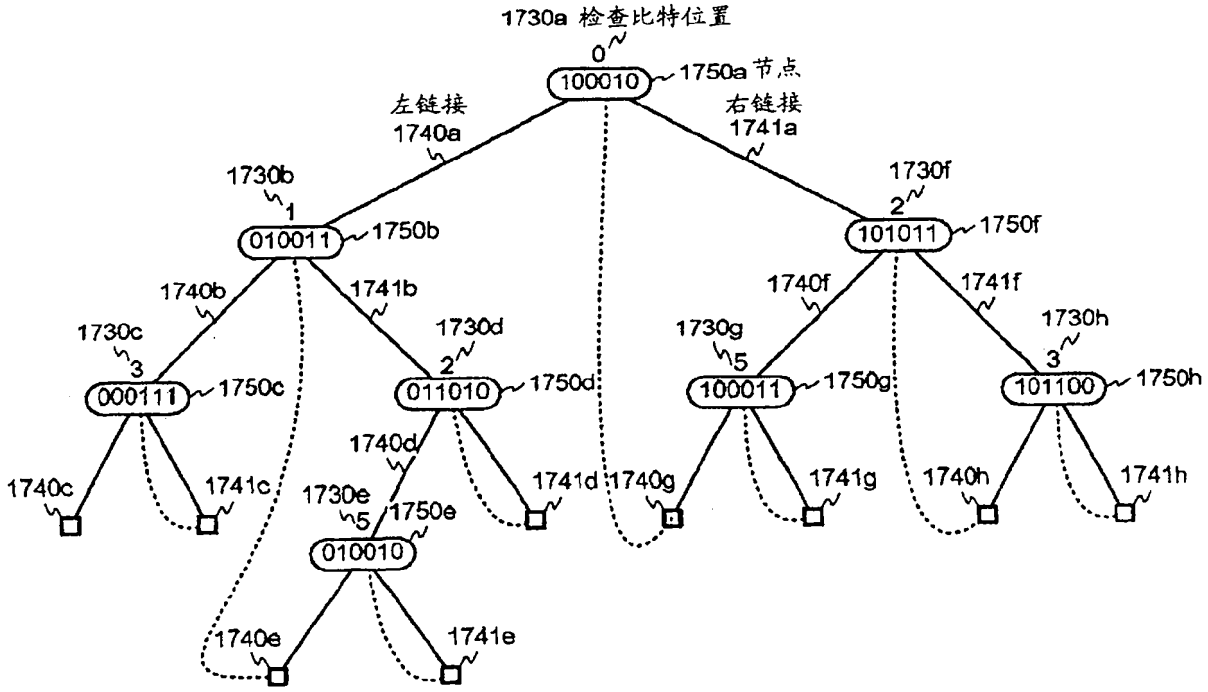

1730a 检查比特位置
0
100010
1750a 节点
左链接
1740a
右链接
1741a
1730b
1
010011
1750b
1730f
2
101011
1750f
1740b
1741b
1740f
1741f
1730c
3
000111
1750c
1730d
2
011010
1750d
1730g
5
100011
1750g
1730h
3
101100
1750h
1740d
1740c
1741c
1730e
5
1750e
1741d
1740g
1741g
1740h
1741h
010010
1740e
1741e

图 17