(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 101589390 B
(45)授权公告日 2012.07.04

(21)申请号 200780050445.9
(22)申请日 2007.10.25
(30)优先权数据
013211/2007 2007.01.24 JP
(85)PCT申请进入国家阶段日
2009.07.24
(86)PCT申请的申请数据
PCT/JP2007/001172 2007.10.25
(87)PCT申请的公布数据
WO2008/090588 JA 2008.07.31
(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 国分光裕
(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)
*G06F 12/00*(2006.01)

(56)对比文件
CN 1516155 A,2004.07.28,
JP 2001-202277 A,2001.07.27,
JP 2001-357070 A,2001.12.26,
JP 2003-224581 A,2003.08.08,

审查员 李东

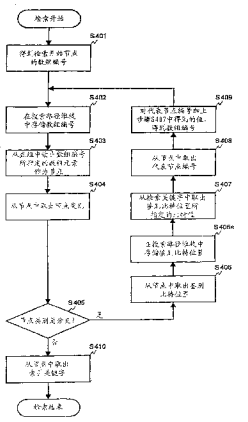
权利要求书 6 页 说明书 19 页 附图 22 页

(54)发明名称

比特序列检索装置、检索方法

(57)摘要

为了即使配对节点树的规模变大也可减轻利用了配对节点树的处理的效率降低，而在基本检索、最大值或最小值检索中，在存储检索历史的搜索路径堆栈中不仅存储配置了节点的地址信息，还存储在检索路径下搜索到的分支节点的鉴别比特位置。



CN 101589390 B

1. 一种比特序列检索装置，所述比特序列检索装置包括：

存储装置，其存储有配对节点树，该配对节点树是用于比特序列检索的树，由根节点、以及在相邻的存储区域中配置的分支节点和叶节点、或者分支节点之间或叶节点之间的节点对构成，所述根节点是表示树的起点的节点，当该树的节点为一个时所述根节点是所述叶节点，当树的节点为两个以上时所述根节点是所述分支节点，所述分支节点包含进行比特序列检索的检索关键字的鉴别比特位置、和表示作为链接目的地节点对中的一个节点的代表节点的位置的位置信息，所述叶节点包含由检索对象比特序列构成的索引关键字，

数据处理装置，其将所述配对节点树的任意节点作为检索开始节点，在所述分支节点中，根据该分支节点中包含的鉴别比特位置的检索关键字的比特值，依次反复地链接到链接目的地节点对的代表节点或与其相邻的存储区域中所配置的节点，直至到达所述叶节点为止，由此将存储在所述叶节点中的索引关键字作为检索结果关键字，所述检索结果关键字是所述配对节点树的以所述检索开始节点为根节点的任意部分树的、基于所述检索关键字的检索结果，

其中，所述数据处理装置依次在堆栈中保持以下三种信息：配置了所述检索开始节点的存储区域的地址信息、配置了从所述检索开始节点到所述叶节点的链接目的地节点的存储区域的地址信息、以及从所述检索开始节点到所述叶节点的链接路径的分支节点的鉴别比特位置。

2. 根据权利要求 1 所述的比特序列检索装置，其特征在于，

所述配对节点树存储在数组中，所述位置信息是所述数组的存储了与该位置信息对应的所述代表节点的数组元素的数组编号，

配置了所述检索开始节点以及链接目的地节点的存储区域的地址信息是存储了所述检索开始节点以及链接目的地节点的数组元素的数组编号。

3. 根据权利要求 2 所述的比特序列检索装置，其特征在于，

通过所述数组编号与所述检索关键字的鉴别比特位置的比特值的运算，来求出存储了链接目的地节点的数组元素的数组编号。

4. 根据权利要求 1 所述的比特序列检索装置，其特征在于，

所述分支节点与叶节点包含有表示各个节点的类别的数据。

5. 一种索引关键字插入方法，该方法在配对节点树中插入包含由期望比特序列构成的索引关键字的叶节点，

该配对节点树是用于比特序列检索的树，由根节点、以及在相邻的存储区域中配置的分支节点和叶节点、或者分支节点之间或叶节点之间的节点对构成，所述根节点是表示树的起点的节点，当该树的节点为一个时所述根节点是所述叶节点，当树的节点为两个以上时所述根节点是所述分支节点，所述分支节点包含进行比特序列检索的检索关键字的鉴别比特位置、和表示作为链接目的地节点对中的一个节点的代表节点的位置的位置信息，所述叶节点包含由检索对象比特序列构成的索引关键字，

该配对节点树构成为以所述树的任意节点作为检索开始节点，在所述分支节点中，根据该分支节点中包含的鉴别比特位置的检索关键字的比特值，依次反复地链接到链接目的地节点对的代表节点或与其相邻的存储区域中所配置的节点，直至到达所述叶节点为止，由此将存储在所述叶节点中的索引关键字作为检索结果关键字，所述检索结果关键字是所

述树的以所述检索开始节点为根节点的任意部分树的、基于所述检索关键字的检索结果，

所述索引关键字插入方法的特征在于，具有以下步骤：

将所述索引关键字作为所述检索关键字，从所述配对节点树中检索相应的叶节点，并且在堆栈中依次存储：到达该叶节点为止所搜索到的链接路径的分支节点以及配置了该叶节点的存储区域的地址信息、和该分支节点的鉴别比特位置；

在所述检索关键字与所述相应的叶节点所包含的索引关键字之间进行大小比较和比特序列比较；

根据在比特序列比较中成为不同比特值的开头比特位置与存储在所述堆栈中的鉴别比特位置之间的相对位置关系，来决定由包含被插入索引关键字的叶节点和另一个节点构成的节点对的插入位置；以及

根据所述大小比较来决定将包含待插入索引关键字的叶节点设为所述被插入节点对中的哪个节点。

6. 根据权利要求 5 所述的索引关键字插入方法，其特征在于，

所述配对节点树存储在数组中，所述位置信息是所述数组的存储了与该位置信息对应的所述代表节点的数组元素的数组编号，

配置了所述检索开始节点以及链接目的地节点的存储区域的地址信息是存储了所述检索开始节点以及链接目的地节点的数组元素的数组编号。

7. 一种比特序列检索方法，该方法采用配对节点树，

该配对节点树是用于比特序列检索的树，由根节点、以及在相邻的存储区域中配置的分支节点和叶节点、或者分支节点之间或叶节点之间的节点对构成，所述根节点是表示树的起点的节点，当该树的节点为一个时所述根节点是所述叶节点，当树的节点为两个以上时所述根节点是所述分支节点，所述分支节点包含进行比特序列检索的检索关键字的鉴别比特位置、和表示作为链接目的地节点对中的一个节点的代表节点的位置的位置信息，所述叶节点包含由检索对象比特序列构成的索引关键字，

该方法将所述配对节点树的任意节点作为检索开始节点，在所述分支节点中，根据该分支节点中包含的鉴别比特位置的检索关键字的比特值，依次反复地链接到链接目的地节点对的代表节点或在与其相邻的存储区域中配置的节点，直至到达所述叶节点为止，由此将存储在所述叶节点中的索引关键字作为检索结果关键字，所述检索结果关键字是所述配对节点树的以所述检索开始节点为根节点的任意部分树的、基于所述检索关键字的检索结果，

所述比特序列检索方法的特征在于，

在堆栈中依次保持以下三种信息：配置了所述检索开始节点的存储区域的地址信息、配置了从所述检索开始节点到所述叶节点的链接目的地节点的存储区域的地址信息、以及从所述检索开始节点到所述叶节点的链接路径的分支节点的鉴别比特位置。

8. 根据权利要求 7 所述的比特序列检索方法，其特征在于，

所述配对节点树存储在数组中，所述位置信息是所述数组的存储了与该位置信息对应的所述代表节点的数组元素的数组编号，

配置了所述检索开始节点以及链接目的地节点的存储区域的地址信息是存储了所述检索开始节点以及链接目的地节点的数组元素的数组编号。

9. 根据权利要求 8 所述的比特序列检索方法，其特征在于，

通过所述数组编号与所述检索关键字的鉴别比特位置的比特值的运算，来求出存储了链接目的地节点的数组元素的数组编号。

10. 一种比特序列检索方法，该方法采用配对节点树，

该配对节点树是用于比特序列检索的树，由根节点、以及在相邻的存储区域中配置的分支节点和叶节点、或者分支节点之间或叶节点之间的节点对构成，所述根节点是表示树的起点的节点，当该树的节点为一个时所述根节点是所述叶节点，当树的节点为两个以上时所述根节点是所述分支节点，所述分支节点包含进行比特序列检索的检索关键字的鉴别比特位置、和表示作为链接目的地节点对中的一个节点的代表节点的位置的位置信息，所述叶节点包含由检索对象比特序列构成的索引关键字，

该方法将所述配对节点树的任意节点作为检索开始节点，仅链接所述节点对中构成该节点对的 2 个节点中的代表节点、或在与代表节点相邻的存储区域中配置的节点，而到达叶节点，由此求出以所述检索开始节点为根节点的任意部分树的检索关键字的最小值或最大值，

所述比特序列检索方法的特征在于，

依次在堆栈中保持以下三种信息 :配置了所述检索开始节点的存储区域的地址信息、配置了从所述检索开始节点到所述叶节点的链接目的地节点的存储区域的地址信息、以及从所述检索开始节点到所述叶节点的链接路径的分支节点的鉴别比特位置。

11. 根据权利要求 10 所述的比特序列检索方法，其特征在于，

所述配对节点树存储在数组中，所述位置信息是所述数组的存储了与该位置信息对应的所述代表节点的数组元素的数组编号，

配置了所述检索开始节点以及链接目的地节点的存储区域的地址信息是存储了所述检索开始节点以及链接目的地节点的数组元素的数组编号。

12. 一种配对节点树分割方法，

该配对节点树是用于比特序列检索的树，由根节点、以及在相邻的存储区域中配置的分支节点和叶节点、或者分支节点之间或叶节点之间的节点对构成，所述根节点是表示树的起点的节点，当该树的节点为一个时所述根节点是所述叶节点，当树的节点为两个以上时所述根节点是所述分支节点，所述分支节点包含进行比特序列检索的检索关键字的鉴别比特位置、和表示作为链接目的地节点对中的一个节点的代表节点的位置的位置信息，所述叶节点包含由检索对象比特序列构成的索引关键字，

该配对节点树构成为以所述树的任意节点作为检索开始节点，在所述分支节点中，根据该分支节点中包含的鉴别比特位置的检索关键字的比特值，依次反复地链接到链接目的地节点对的代表节点或者与其相邻的存储区域中所配置的节点，直至到达所述叶节点为止，由此将存储在所述叶节点中的索引关键字作为检索结果关键字，所述检索结果关键字是所述树的以所述检索开始节点为根节点的任意部分树的、基于所述检索关键字的检索结果，

所述配对节点树分割方法的特征在于，该配对节点树分割方法具有以下步骤 :

取得分割关键字的步骤，该分割关键字决定对作为分割对象的处理源配对节点树进行分割的索引关键字 ;

处理源最小值或最大值取得步骤，通过权利要求 10 所述的比特序列检索方法来求出所述处理源配对节点树的索引关键字的最小值或最大值；

比较步骤，对所述分割关键字与所述最小值或最大值的大小进行比较，如果该最小值大于所述分割关键字或该最大值小于所述分割关键字，则结束处理；

生成步骤，如果所述比较结果为该最小值不大于所述分割关键字或该最大值不小于所述分割关键字，则通过权利要求 5 所述的索引关键字插入方法来插入该最小值或该最大值的索引关键字，生成新的处理目标配对节点树；以及

删除步骤，从所述处理源配对节点树中删除所述最小值或所述最大值的索引关键字，

反复进行将删除了所述最小值或所述最大值的索引关键字后的所述处理源配对节点树作为新的处理源配对节点树的、所述处理源最小值或最大值取得步骤、所述比较步骤、所述生成步骤以及所述删除步骤，直到在所述处理源最小值或最大值取得步骤中取得的最小值大于所述分割关键字、或取得的最大值小于所述分割关键字为止。

13. 根据权利要求 12 所述的配对节点树分割方法，其特征在于，

所述配对节点树存储在数组中，所述位置信息是所述数组的存储了与该位置信息对应的所述代表节点的数组元素的数组编号，

配置了所述检索开始节点以及链接目的地节点的存储区域的地址信息是存储了所述检索开始节点以及链接目的地节点的数组元素的数组编号，

所述生成步骤包含以下步骤：

将在所述处理源最小值或最大值取得步骤中求出的最小值或最大值设定为所述处理目标配对节点树的插入关键字，

执行通过权利要求 11 所述的比特序列检索方法来取得所述处理目标配对节点树的索引关键字的最大值或最小值的处理目标最大值或最小值取得步骤，

在所述插入关键字与所述索引关键字的最大值或最小值之间进行比特序列比较，求出成为不同比特值的开头比特位置，

根据该比特位置与存储在所述堆栈中的鉴别比特位置之间的相对位置关系，决定处理目标父节点，所述处理目标父节点是包含了保持所述插入关键字的叶节点的节点对的插入位置，

将该处理目标父节点的所述位置信息设为存储如下节点对的代表节点的数组元素的数组编号，该节点对包含了保持所述插入关键字的叶节点。

14. 根据权利要求 13 所述的配对节点树分割方法，其特征在于，

所述生成步骤还包含以下步骤：将所述处理目标父节点设定为插入了包含插入关键字的叶节点的插入完毕节点，

所述处理目标最大值或最小值取得步骤是将所述插入完毕节点作为检索开始节点，取得所述处理目标配对节点树的索引关键字的最大值或最小值的步骤。

15. 根据权利要求 14 所述的配对节点树分割方法，其特征在于，

所述删除步骤包含以下步骤：

将包含所述处理源最小值或最大值取得步骤中求出的最小值或最大值作为索引关键字的叶节点，设定为所述处理源配对节点树的删除节点，

将与所述删除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

点中，以及

删除该节点对。

16. 一种配对节点树结合方法，用于 2 个配对节点树的结合，

该配对节点树是用于比特序列检索的树，由根节点、以及在相邻的存储区域中配置的分支节点和叶节点、或者分支节点之间或叶节点之间的节点对构成，所述根节点是表示树的起点的节点，当该树的节点为一个时所述根节点是所述叶节点，当树的节点为两个以上时所述根节点是所述分支节点，所述分支节点包含进行比特序列检索的检索关键字的鉴别比特位置、和表示作为链接目的地节点对中的一个节点的代表节点的位置的位置信息，所述叶节点包含由检索对象比特序列构成的索引关键字，

该配对节点树构成为以所述树的任意节点作为检索开始节点，在所述分支节点中，根据该分支节点中包含的鉴别比特位置的检索关键字的比特值，依次反复地链接到链接目的地节点对的代表节点或与其相邻的存储区域中所配置的节点，直至到达所述叶节点为止，由此将存储在所述叶节点中的索引关键字作为检索结果关键字，所述检索结果关键字是所述树的以所述检索开始节点为根节点的任意部分树的、基于所述检索关键字的检索结果，

所述配对节点树结合方法的特征在于，该配对节点树结合方法具有以下步骤：

处理源最小值或最大值取得步骤，通过权利要求 10 所述的比特序列检索方法来求出所述 2 个配对节点树中一方的处理源配对节点树的索引关键字的最小值或最大值；

插入步骤，通过权利要求 5 所述的索引关键字插入方法，在所述 2 个配对节点树的另一方的处理目标配对节点树中，插入所述最小值或最大值的索引关键字；以及

删除步骤，从所述处理源配对节点树中删除所述最小值或所述最大值的索引关键字，

反复进行将删除了所述最小值或所述最大值的索引关键字后的所述处理源配对节点树作为新的处理源配对节点树的、所述处理源最小值或最大值取得步骤、所述插入步骤以及所述删除步骤，直到所述处理源配对节点树被全部删除。

17. 根据权利要求 16 所述的配对节点树结合方法，其特征在于，

所述配对节点树存储在数组中，所述位置信息是所述数组的存储了与该位置信息对应的所述代表节点的数组元素的数组编号，

配置了所述检索开始节点以及链接目的地节点的存储区域的地址信息是存储了所述检索开始节点以及链接目的地节点的数组元素的数组编号，

所述插入步骤包含以下步骤：

将所述最小值或所述最大值设定为所述处理目标配对节点树的插入关键字，

执行通过权利要求 11 所述的比特序列检索方法来取得所述处理目标配对节点树的索引关键字的最大值或最小值的处理目标最大值或最小值取得步骤，

在所述插入关键字与所述索引关键字的最大值或最小值之间进行比特序列比较，求出成为不同比特值的开头比特位置，

根据该比特位置与存储在所述堆栈中的鉴别比特位置之间的相对位置关系，决定作为如下节点对的插入位置的父节点，该节点对包含了保持所述插入关键字的叶节点，

将该父节点的所述位置信息设为存储如下节点对的代表节点的数组元素的数组编号，该节点对包含了保持所述插入关键字的叶节点。

18. 根据权利要求 17 所述的配对节点树结合方法，其特征在于，

所述删除步骤包含以下步骤：

将包含所述处理源最小值或最大值取得步骤中求出的最小值或最大值作为索引关键字的叶节点，设定为所述处理源配对节点树的删除节点，

将与所述删除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中，以及

删除该节点对。

# 比特序列检索装置、检索方法

## 技术领域

[0001] 本发明涉及使用存储比特序列的树状数据结构从比特序列的集合中检索期望的比特序列的检索方法，特别涉及使用了本申请人在日本特愿 2006-187827 中提出的配对节点树的比特序列检索装置、检索方法以及程序。

## 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被利用起来。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作为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可视为文档的索引关键字。

[0003] 而且，由于这些索引关键字利用比特序列来表达，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，一直以来对存储比特序列的数据结构进行了各种研究。作为这种研究之一，公知的是 Patricia 树这样的树结构。

[0005] 图 1 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。虽未作明示，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访问的信息。

[0006] 在图 1 的例子中，保持索引关键字“100010”的节点 1750a 是根节点，其检查比特位置 1730a 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 1730c 是 3。节点 1750d 保持的索引关键字是“011010”，检查比特位置 1730d 是 2。

[0008] 从节点 1750c 开始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 的虚线的链接目的地表示指针所指示的地址，在这里表示右指针 1741c 指向节点 1750c。

[0009] 节点 1750d 的右指针 1741d 指向节点 1750d 自身，左链接 1740d 与节点 1750e 连接。节点 1750e 保持的索引关键字是“010010”，检查比特位置 1730e 是 5。节点 1750e 的左指针 1740e 指向节点 1750b，右指针 1741e 指向节点 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 指向节点 1750a，右指针 1741g 指向节点 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 指向节点 1750f，右指针 1741h 指向节点 1750h。

[0013] 在图 1 的例子中，形成这样的结构：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，是 1 时搜索右链接，是 0 时搜索左链接。然后，当链接目的地节点的检查比特位置不大于链接源节点的检查比特位置时，即，链接目的地不是下方而是回到上方时（将图 1 中虚线所示的该后退的链接称为反向链接），对链接目的地节点的索引关键字与检索关键字进行比较。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：只通过必要比特的检查即可进行检索，以及关键字全体的比较只需一次，然而 具有以下等缺点：由于必定有从各节点起的 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，由于反向链接而返回、从开头与索引关键字进行比较而造成的检索处理延迟以及追加删除等数据维护困难。

[0016] 作为解决 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，通过在连续的区域中存储下位的左右节点来削减指针的存储容量，并通过在各节点中设置表示下一链接是否是反向链接的比特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，将下位的左右节点存储在连续的区域中而采用 1 个指针，因而对于例如图 1 所示的 Patricia 树的最下段部分即左指针 1740c、右指针 1741h 等部分，也必须分配与节点相同容量的存储区域，存储容量的削减效果不怎么好。并且，也没有改善反向链接引起的检索处理的延迟问题，并且很难进行追加删除等处理。

[0018] 专利文献 1：日本特开 2001-357070 号公报

[0019] 作为解决上述现有检索方法中的问题点的方案，本申请人在日本特愿 2006-187827 中提出了使用配对节点树的比特序列检索，该配对节点树是由根节点、和在相邻的存储区域中配置的分支节点和叶节点或者分支节点之间或叶节点之间的节点对构成的比特序列检索用的树，根节点是表示树的起点的节点，在该树的节点为一个时，根节点为叶节点，在树的节点为两个以上时，根节点为所述分支节点，所述分支节点包含进行比特序列检索的检索关键字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位置信息，所述叶节点包含由检索对象比特序列构成的索引关键字。

[0020] 在上述申请中，示出了根据给定的索引关键字的集合来生成配对节点树的方法、以及从配对节点树检索单个索引关键字的方法等使用配对节点树的基本检索方法。

[0021] 并且，在比特序列的检索中，存在求出最小值、最大值或求出某个范围的值等各种检索要求。因此，本申请人在日本特愿 2006-293619 中 提出了求出配对节点树的任意部分树中包含的索引关键字的最大值 / 最小值的方法等。

[0022] 此外，本申请人在日本特愿 2006-319407 中提出了配对节点树的分割 / 结合方法。

[0023] 在上述三个申请中提出的检索方法是以下述操作为基础的，即：从检索开始节点开始依次搜索分支节点，直至叶节点，并求出叶节点中存储的索引关键字。作为检索历史，即配置了从检索开始节点到叶节点的节点的存储区域的地址信息，在堆栈中存储对该节点

进行存储的数组元素的数组序号。然后，在检索处理后的各种处理中进行下述处理：以存储在堆栈中的地址信息为基础，参照在该地址信息所示的存储区域中存储的节点来取出鉴别比特位置。

[0024] 作为上述地址信息，在将配对节点树存储在数组中的情况下，可以采用该数组中的数组编号，可以削减表示地址信息的比特量，但是随着配对节点树的规模变大，实际存储节点的地址范围也相应地变大，从而在计算机上执行各种处理时引起缓存错误的可能性变高，可能会降低处理效率。

## 发明内容

[0025] 因此，本发明的目的是提供一种即使配对节点树的规模变大也可减轻利用了配对节点树的处理的效率降低的方法。

[0026] 根据本发明，在基本检索、最大值或最小值检索的检索处理中，在存储检索历史的探索路径堆栈中，不仅存储配置了节点的存储区域的地址信息，还存储在检索路径中搜索到的分支节点的鉴别比特位置。

[0027] 另外，在对检索处理所搜索到的分支节点的鉴别比特位置进行搜索的索引关键字插入及配对节点树分割结合处理中，从搜索路径堆栈中取出鉴别比特位置。

[0028] 根据本发明，当需要在检索处理中搜索到的节点的鉴别比特位置时，该信息可以从存储检索历史的探索路径堆栈中获得，这样即使配对节点树的规模变大，也能够减小缓存错误发生的可能性。

## 附图说明

[0029] 图 1 是示出以往的检索中使用的 Patricia 树的一例的图。

[0030] 图 2A 是说明存储在数组中的配对节点树的结构例的图。

[0031] 图 2B 是概念性示出配对节点树的树结构的图。

[0032] 图 3 是说明用于实施本发明的硬件结构例的图。

[0033] 图 4A 是示出本发明中的比特序列检索的基本动作的流程图。

[0034] 图 4B 是利用配对节点树来说明本发明中的比特序列检索的基本动作的图。

[0035] 图 5A 是示出本发明中的作为插入处理前级的检索处理的处理流程的图。

[0036] 图 5B 是说明准备用于待插入节点对的数组元素的处理的处理流程图。

[0037] 图 5C 是示出求得插入节点对的位置并写入节点对的各节点的内容以完成插入处理的处理流程的图。

[0038] 图 6 是说明包含本发明中的根节点插入处理在内的、追加索引关键字时的整个节点插入处理的处理流程图。

[0039] 图 7A 是示出作为删除处理前级的检索处理的处理流程的图。

[0040] 图 7B 是说明删除处理后级的处理流程的图。

[0041] 图 8 是示出本发明的求取配对节点树中存储的索引关键字的最小值的处理的流程图。

[0042] 图 9 是利用配对节点树来说明本发明的求取索引关键字的最小值的处理的图。

[0043] 图 10 是示出本发明的求取配对节点树中存储的索引关键字的最大值的处理的流

程图。

[0044] 图 11 是利用配对节点树来说明本发明的求取索引关键字的最大值的处理的图。

[0045] 图 12 是说明配对节点树的第 1 分割处理流程的图。

[0046] 图 13 是说明配对节点树的第 2 分割处理流程的图。

[0047] 图 14 是说明第 2 分割处理中的节点插入处理流程的图。

[0048] 图 15A 是说明第 2 分割处理中的根节点设定处理的流程图。

[0049] 图 15B 是说明第 2 分割处理中的在父节点中插入包含插入关键字的节点对的处理流程的图。

[0050] 图 16 是说明第 2 分割处理中的删除处理的流程图。

## 具体实施方式

[0051] 首先，关于本申请人在之前的上述申请中提出的作为本发明前提的配对节点树，说明将配对节点树存储在数组中的例子。分支节点保持的表示链接目的地位置的数据也可以作为存储装置的地址信息，不过通过使用由可存储分支节点或叶节点中的所占有区域的存储容量更大一方的数组元素构成的数组，可以用数组编号来表示节点位置，这样可以削减位置信息的信息量。

[0052] 图 2A 是说明存储在数组中的配对节点树的结构例的图。

[0053] 参照图 2A，节点 101 配置在数组 100 的数组编号为 10 的数组元素内。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 中存储有 1。在代表节点编号 104 中存储有链接目的地节点对的代表节点的数组编号 20。另外，以下为了简化说明，有时把存储在代表节点编号中的数组编号称为代表节点编号。并且，存储在代表节点编号中的数组编号有时利用赋予给该节点的符号或赋予给节点对的符号来表示。

[0054] 在数组编号 20 的数组元素中存储有节点对 111 的代表节点即节点 [0]112。然后在相邻的下一数组元素（数组编号 20+1）中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 中存储有 0，在鉴别比特位置 115 中存储有 3，在代表节点编号 116 中存储有 30。并且，在节点 [1]113 的节点类别 117 中存储有 1，表示节点 [1]113 是叶节点。在索引关键字 118 中存储有“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访问的信息，然而省略说明。

[0055] 另外，有时用节点 [0] 表示代表节点，用节点 [1] 表示与其成对的节点。此外，有时将存储在某个数组编号的数组元素中的节点称为该数组编号的节点，将存储有节点的数组元素的数组编号称为节点的数组编号。

[0056] 省略了由存储在数组编号 30 和 31 的数组元素中的节点 122 和节点 123 构成的节点对 121 的内容。

[0057] 分别赋予给存储有节点 [0]112、节点 [1]113、节点 122 以及节点 123 的数组元素的 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。链接到将位于前级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上。

[0058] 因此，通过将前级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节点的数组元素的数组编号。

[0059] 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，然而显然也可采用大的一方。

[0060] 图 2B 是概念性地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 1 所例示的 Patricia 树的索引关键字相同。

[0061] 符号 210a 所示的是根节点。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。

[0062] 作为树结构，在根节点 210a 之下配置有节点对 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 之下配置有节点对 201d，并在节点对 201d 之下配置有节点对 201e。

[0063] 附在各节点前面的符号 0 或 1 与在图 2A 中所说明的附在数组元素前面的符号相同。根据检索关键字的鉴别比特位置的比特值来搜索树，从而找到检索对象叶节点。

[0064] 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 示出为 0。代表节点编号是 220a，该编号是存储了节点对 201b 的代表节点 210b 的数组元素的数组编号。

[0065] 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中存储有存储了节点对 201c 的代表节点 210c 的数组元素的数组编号 220b。

[0066] 由于在节点 210c 的节点类别 260c 中存储有 1，因而该节点是叶节点，因此包含索引关键字 250c。在索引关键字 250c 中存储有“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，在代表节点编号中存储有存储了节点对 201d 的代表节点 210d 的数组元素的数组编号 221c。

[0067] 节点 210d 的节点类别 260d 是 0，鉴别比特位置 230d 是 5，在代表节点编号中存储有存储了节点对 201e 的代表节点 210e 的数组元素的数组编号 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 中存储有“011010”。

[0068] 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，在各自的索引关键字 250e、251e 中存储有“010010”和“010011”作为索引关键字。

[0069] 在节点对 201b 的另一个节点即节点 211b 的鉴别比特位置 231b 中存储有 2，在链接目的地的代表节点编号中存储有存储了节点对 201f 的代表节点 210f 的数组元素的数组编号 221b。

[0070] 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。在各自的鉴别比特位置 230f、231f 中存储有 5、3。在节点 210f 的代表节点编号中存储有存储了节点对 201g 的代表节点 210g 的数组元素的数组编号 220f，在节点 211f 的代表节点编号中存储有存储了节点对 201h 的代表节点即节点 [0]210h 的数组元素的数组编号 221f。

[0071] 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，在各自的索引关键字 250g、251g 中存储有“100010”和“100011”。

[0072] 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，在各自的索引关键字 250h、251h 中存

储有“101011”和“101100”。

[0073] 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、...。

[0074] 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元素中存储的节点 211b。由于在节点 211b 的鉴别比特位置 231b 中存储有 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 的数组元素中存储的节点 210f。

[0075] 由于在节点 210f 的鉴别比特位置 230f 中存储有 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 的数组元素中存储的节点 210g。

[0076] 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 并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。

[0077] 下面，参照图 2B 说明配对节点树的结构的意义。

[0078] 配对节点树的结构是由索引关键字的集合规定的。在图 2B 的例子中，根节点 210a 的鉴别比特位置是 0，这是因为在图 2B 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 之下，第 0 比特是 1 的索引关键字的组被分类在节点 211b 之下。

[0079] 节点 211b 的鉴别比特位置是 2，这反映了索引关键字集合的性质，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 比特全都是 0，从第 2 比特开始才有不同。

[0080] 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。

[0081] 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 的鉴别比特位置中存储 5。

[0082] 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 中存储有“101011”和“101100”。

[0083] 假设在索引关键字的集合内包含有“101101”或“101110”来取代“101100”，但是到第 3 比特为止与“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。

[0084] 如以上说明的那样，配对节点树的结构是由索引关键字集合中包含的各索引关键字的各比特位置的比特值所决定的。

[0085] 进而也可以说，对于成为不同比特值的每个比特位置，分支为比特值是“1”的节点和比特值是“0”的节点，所以当使节点“1”侧和树的深度方向优先来搜索叶节点时，在它们

中存储的索引关键字成为节点 211h 的索引关键字 251h 的“101100”、节点 210h 的索引关键字 250h 的“101011”、…、节点 210c 的索引关键字 250c 的“000111”，并以降序的方式排序。

[0086] 即，在配对节点树中，对索引关键字进行排序并将其配置在树上。

[0087] 当利用检索关键字进行检索时，搜索在配对节点树上配置了索引关键字的根，例如，如果检索关键字是“101100”，则可到达节点 211h。并且，根据上述说明还可以想到，即使在将“101101”或“101110”作为检索关键字的情况下，也可以搜索到节点 211h，通过与索引关键字 251h 进行比较可知晓检索失败。

[0088] 并且，例如在利用“100100”进行检索的情况下，在节点 210a、211b、210f 的链接路径上不使用检索关键字的第 3 比特和第 4 比特、并且由于“100100”的第 5 比特是 0，所以与利用“100010”进行检索的情况同样，到达节点 210g。这样，使用与存储在配对节点树中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。

[0089] 图 3 是说明用于实施本发明的硬件结构例的图。

[0090] 本发明的检索装置的检索处理和数据维护是通过数据处理装置 301 使用数据存储装置 308 来实施的，该数据处理装置 301 至少具有中央处理装置 302 和缓存 303。数据存储装置 308 具有对配对节点树进行配置的数组 309 和搜索路径堆栈 310，该搜索路径堆栈 310 存储对在检索中搜索到的节点进行存储的数组元素的数组编号之外，还存储节点内的信息。该数据存储装置 308 可以利用主存储装置 305 或外部存储装置 306 来实现，或者还可以使用经由通信装置 307 而连接的配置在远方的装置。

[0091] 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，不过连接方法不仅限于此。另外，还可以使主存储装置 305 在数据处理装置 301 内，或可以将搜索路径堆栈 310 实现为中央处理装置 302 内的硬件。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，选择适当的硬件结构，例如，使数组 309 在外部存储装置 306 内、使搜索路径堆栈 310 在主存储装置 305 内等。

[0092] 并且，尽管未作特别图示，为了能在后面的处理中使用处理中途所获得的各种值，当然可以使用与各个处理对应的临时存储装置。

[0093] 接着，说明利用了在上述申请中由本申请人提出的以下处理的本发明：采用了配对节点树的基本检索处理、配对节点树中的插入删除处理、求得配对节点树所包含的索引关键字的最大值 / 最小值的处理等应用处理、以及在配对节点树的分割 / 结合处理中，在存储检索历史的堆栈中不仅存储配置节点的存储区域的地址信息，还存储在各种处理中采用的分支节点的鉴别比特位置。

[0094] 图 4A 是示出本发明的比特序列检索的基本动作的流程图。图 4A 所示的流程图是在本申请人申请的上述日本特愿 2006-293619 所提出的表示比特序列检索的基本动作的流程图中追加了在搜索路径堆栈 310 中存储鉴别比特位置的步骤 S406a 的处理。

[0095] 首先，在步骤 S401 中取得检索开始节点的数组编号。与取得的数组 编号对应的数组元素存储有构成配对节点树的任意节点。在后面说明的各种应用检索中对检索开始节点进行指定。

[0096] 接着，在步骤 S402 中，在搜索路径堆栈 310 中存储所取得的数组编号，在步骤 S403 中，将与该数组编号对应的数组元素作为应该参照的节点进行读出。然后，在步骤 S404 中，

从读出的节点中取出节点类别，在步骤 S405 中，判定节点类别是否是分支节点。

[0097] 在步骤 S405 的判定中，当读出的节点是分支节点时，进入步骤 S406，从节点中取出关于鉴别比特位置的信息。

[0098] 接着在步骤 S406a 中，将步骤 S406 所取出的鉴别比特位置存储在搜索路径堆栈 310 中，并且在步骤 S407，从检索关键字中取出与取出的鉴别比特位置对应的比特值。然后，在步骤 S408，从节点中取出代表节点编号，在步骤 S409，将从检索关键字取出的比特值与代表节点编号相加，得到新的数组编号，然后返回步骤 S402。

[0099] 此后，在步骤 S405 的判定中判定为叶节点而进入步骤 S410 之前，重复从步骤 S402 到步骤 S409 的处理。在步骤 S410 中，从叶节点中取出索引关键字，结束处理。

[0100] 图 4B 是通过图 2B 所例示的配对节点树来说明图 4A 的流程图所示的本实施方式的比特序列检索的基本动作的图。

[0101] 在图 4B 中示出了配对节点树、检索关键字设定区域 270 以及搜索路径堆栈 310。以下，采用图 4B 来说明在配对节点树上所参照的节点以及搜索路径堆栈 310 的状态。

[0102] 另外，在图 4B 中针对图 2B 所示的配对节点树，仅示出用于说明图 4A 的处理所需的一部分，并省略其余的节点（代表节点编号 221b 以下的节点）。在以下的说明中，对于用于说明在配对节点树上参照的节点的附图也是同样的。

[0103] 首先，设定数组编号 220 作为开始检索的节点的数组编号。此时，把对应的数组编号 220 压入 (push) 搜索路径堆栈 310，并且参照数组元素的各种信息。

[0104] 根据存储在数组中的信息识别为数组编号 220 的节点是分支节点而 没有存储索引关键字时，进一步参照在该数组编号 220 的数组中存储的信息（代表节点编号及鉴别比特位置）等，来算出下一个应该参照的数组编号。

[0105] 这里，首先将数组编号 220 存储在搜索路径堆栈 310 中，从数组编号 220 的节点 210a 中取出节点类别 260a。取出的节点类别 260a 如图所示为″ 0″，表示节点 210a 为分支节点，因此从节点 210a 取出鉴别比特位置 230a 的值″ 0″，并与数组编号 220 一起存储在搜索路径堆栈 310 中。

[0106] 此外，取出检索关键字设定区域 270 的检索关键字″ 011010″中的鉴别比特位置 0 的比特值″ 0″。然后，取出在节点 210a 中存储的代表节点编号 220a 的值，与先前从检索关键字的鉴别比特位置取出的比特值″ 0″相加，将相加得到的数组编号 220a 存储在搜索路径堆栈 310 中。

[0107] 接着，当从数组编号 220a 的节点中读出节点类别 260b 时，判定数组编号 220a 的节点 210b 是分支节点，在搜索路径堆栈 310 中存储节点 210b 的鉴别比特位置 230b 的值″ 1″，从检索关键字″ 011010″取出与鉴别比特位置 1 对应的比特值，值是″ 1″。因此，将节点 210b 的代表节点编号 220b 与得到的比特值″ 1″相加，在搜索路径堆栈 310 中存储 220b+1。

[0108] 从数组编号 (220b+1) 的节点 211c 读出的节点类别 261c 表示分支节点，所以将节点 211c 的鉴别比特位置 231c 的值″ 2″存储在搜索路径堆栈 310 中，从检索关键字″ 011010″中取出与鉴别比特位置 2 对应的比特值时，值为″ 1″。将节点 211c 的代表节点编号 221c 与得到的比特值″ 1″相加后得到的值 221c+1 存储在搜索路径堆栈 310 中。

[0109] 当从数组编号 (221c+1) 的节点 211d 中取出节点类别 261d 时可知，节点 211d 是叶节点。因此，从节点 211d 中取出索引关键字″ 011010″，并结束处理。

[0110] 这样，执行依次参照检索关键字的比特信息和各节点的信息的链接处理，由此在搜索路径堆栈 310 中，按照链接顺序压入从作为检索开始节点的节点 210a 的数组编号 220 到叶节点 211d 的数组编号 (221c+1)、和从节点 210a 的鉴别比特位置 230a 到节点 221c 的鉴别比特位置 231c 的信息。

[0111] 接着，利用图 5A ～图 5C、图 6 来说明本发明的配对节点树中的节点插入处理。图 5A ～图 5C 用于说明常规的插入处理，图 6 用于说明根节点的插入处理。利用根节点的插入处理和常规的插入处理，可生成配对节点树，所以节点插入处理的说明也是配对节点树的生成处理的说明。

[0112] 图 5A 是示出作为插入处理前级的检索处理的处理流程的图，相当于在图 4A 所示的检索处理中将插入关键字作为检索关键字。步骤 S501 相当于在图 4A 的步骤 S401 中将检索开始节点作为根节点，步骤 S502 至步骤 S510 的处理与图 4A 的步骤 S402 至步骤 S410 完全对应，所以省略说明。

[0113] 在图 5A 的步骤 S511 中对插入关键字与索引关键字进行比较，如果等同，则插入关键字已经存在于配对节点树中，所以插入失败，结束处理。如果不等同，则进入下一处理，即图 5B 的步骤 S512 以下的处理。

[0114] 图 5B 是说明准备用于待插入节点对的数组元素的处理的处理流程图。

[0115] 在步骤 S512 中，从数组求得空的节点对，取得该节点对中的应成为代表节点的数组元素的数组编号。

[0116] 进到步骤 S513，对插入关键字与在步骤 S510 所得到的索引关键字的大小进行比较，当插入关键字大时得到值为 1 的布尔值、当插入关键字小时得到值为 0 的布尔值。

[0117] 进到步骤 S514，取得将步骤 S512 所得到的代表节点的数组编号与步骤 S513 所得到的布尔值相加得到的数组编号。

[0118] 进到步骤 S515，取得将步骤 S512 所得到的代表节点的数组编号与步骤 S513 所得到的布尔值的逻辑反转值相加得到的数组编号。

[0119] 步骤 S514 所得到的数组编号是存储有将插入关键字作为索引关键字的叶节点的数组元素的数组编号，步骤 S515 所得到的数组编号是存储了与该叶节点成对的分支节点的数组元素的数组编号。

[0120] 即，根据在前级的检索处理中得到的叶节点中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中存储保持插入关键字的叶节点。

[0121] 例如在图 2B 的配对节点树内插入“011011”的情况下，检索结果的索引关键字为存储在节点 211d 中的“011010”。通过插入关键字“011011”与存储在节点 211d 中的索引关键字“011010”的大小比较来求出布尔值，在这里的例子中由于插入关键字大，因而得到布尔值 1，在插入节点对的代表节点编号加 1 得到的数组元素中，存储保持插入关键字的叶节点。另一方面，把索引关键字“011010”存储在通过大小比较得到的布尔值的逻辑反转值与代表节点编号相加得到的数组编号的数组元素内。

[0122] 此时，由于索引关键字“011010”与插入关键字“011011”在第 5 比特不同，因而节点 211d 成为如下的分支节点：鉴别比特位置为 5、代表节点编号为所插入的节点对的代表

节点的数组编号。

[0123] 并且，在图 2B 的配对节点树中插入“011001”的情况下，检索结果的索引关键字成为存储在节点 211d 中的“011010”。在该情况下，由于插入关键字小，因而得到布尔值 0，在插入节点对的代表节点编号加上 0 后得到的数组元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”与插入关键字“011001”在第 4 比特不同，因而节点 211d 成为如下的分支节点：鉴别比特位置为 4、代表节点编号为所插入的节点对的代表节点的数组编号。

[0124] 接着进入图 5C 的步骤 S516 以下的处理。

[0125] 图 5C 是示出将节点存储在图 5B 中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。

[0126] 步骤 S516 ～步骤 S523 的处理是求出待插入节点对在配对节点树上的位置的处理，步骤 S524 以下的处理是对各节点设定数据来完成插入处理的处理。

[0127] 在步骤 S516 中，利用例如逻辑“异或”来进行插入关键字与步骤 S510 所得到的索引关键字之间的比特序列比较，得到差分比特序列。

[0128] 进到步骤 S517，根据步骤 S516 所得到的差分比特序列，得到从上位第 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 中通过向该 CPU 输入差分比特序列，来得到不一致的比特位置。并且，也能以软件方式进行与优先编码器同等的处 理，得到第一个不一致比特的比特位置。

[0129] 然后进到步骤 S518，判定搜索路径堆栈的堆栈指针是否指向根节点的数组编号。如果指向根节点的数组编号则转到步骤 S524，如果没有指向根节点的数组编号则进到步骤 S519a。

[0130] 在步骤 S519a 中，使搜索路径堆栈的堆栈指针后退 1，取出在此存储的数组编号。

[0131] 在作为在先申请的上述日本特愿 2006-187827 所提出的插入处理中，存储在搜索路径堆栈中的只有数组编号，所以取出数组编号，在数组中从该数组编号指向的数组元素所存储的节点中，取出鉴别比特位置，但是在本发明中，通过图 5A 所示的步骤 S506a，在搜索路径堆栈中存储了鉴别比特位置，因此无需访问数组就能够取出鉴别比特位置。

[0132] 接着进入步骤 S522，判定在步骤 S519a 取出的鉴别比特位置是否是比在步骤 S517 取得的比特位置上位的位置关系。这里所谓上位的位置关系是指比特序列的更左侧的位置，即比特位置的值较小的位置。

[0133] 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 使搜索路径堆栈的堆栈指针前进 1，转到步骤 S524 以下的处理。

[0134] 在上述步骤 S516 ～步骤 S523 所说明的处理是这样的处理：为了决定待插入节点对的插入位置，在待插入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，调查在比特序列比较中成为不同比特值的、开头的（最上位的）比特位置与存储在搜索路径堆栈中的分支节点的鉴别比特位置之间的相对位置关系，将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待插入节点对的插入位置。

[0135] 例如当在图 2B 的配对节点树中插入“111000”时，检索结果的索引关键字为存储在节点 210h 中的“101011”。通过插入关键字“111000”与存储在节点 210h 中的索引关键

字“101011”之间的比特序列比较，得到成为不同比特值的最上位的比特位置 1。依次反向搜索搜索路径堆栈，直到所得到的比特位置 1 与蓄积在搜索路径堆栈中的数组编号的数组元素所存储 的分支节点的鉴别比特位置之间的位置关系成为鉴别比特位置在上位时，到达根节点 210a。在此，使搜索路径堆栈的指针前进 1，得到节点 211b 的数组编号。把插入关键字“111000”插入到节点 211b 的链接目的地。

[0136] 并且，即使对搜索路径堆栈进行反向搜索而到达根节点，但与先前求出的在比特序列比较中成为不同比特值的、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中，与根节点的鉴别比特位置相比为上位的比特值全部相等。并且是指如下情况：在待插入的索引关键字中，存在一开始就与根节点的鉴别比特位置的上位比特值不同的比特值。因此，待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成插入关键字的作为与已有索引关键字不同的值的最上位比特的位置。

[0137] 下面，对步骤S524以下的在各节点中设定数据来完成插入处理的处理进行说明。

[0138] 在步骤 S524 中，从搜索路径堆栈中取出堆栈指针所指的数组编号。

[0139] 在步骤 S525 中，在步骤 S514 中得到的数组编号所指定的数组元素的节点类别中写入 1（叶节点），在索引关键字中写入插入关键字。

[0140] 进到步骤 S526，从数组中读出在步骤 S524 中所得到的数组编号的数组元素。

[0141] 然后在步骤 S527 中，在步骤 S515 中得到的数组编号的数组元素中写入在步骤 S526 中读出的内容。

[0142] 最后在步骤 S528 中，在步骤 S524 中得到的数组编号所指定的数组元素的节点类别中写入 0（分支节点），在鉴别比特位置中写入在步骤 S517 中得到的比特位置，在代表节点编号中写入步骤 S512 中得到的数组编号，结束处理。

[0143] 在上述图 2B 的配对节点树中插入“111000”的例子中，在所取得的空节点对的节点 [0] 中写入节点 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 的鉴别比特位置中存储在比特序列比较中成为不同比特值的、最上位的比特

[0144] 图 6 是说明包含根节点插入处理在内的追加索引关键字时的整个节点插入处理的处理流程图。图 6 所示的流程本身与说明作为在先申请的上述日本特愿 2006-187827 中提出的整个节点插入处理的处理流程是同样的。

[0145] 在步骤 S601 中，对要求取得的配对节点树的根节点的数组编号是否登记完毕进行判定。如果登记完毕，则进行使用图 5A ～图 5C 中说明的常规的插入处理。

[0146] 如果步骤 S601 的判定为未登记完毕，则开始全新的配对节点树的登记、生成。

[0147] 首先，在步骤 S602 中，由数组求得空节点对，取得该节点对中的应成为代表节点的数组元素的数组编号。然后在步骤 S603 中，求出对步骤 S602 中得到的数组编号加上 0 后的数组编号（实际上，等于在步骤 S602 中取得的数组编号）。然后在步骤 S604 中，针对在步骤 S603 中得到的数组编号的数组元素，对待插入根节点的节点类别写入 1（叶节点），并对索引关键字写入插入关键字，在步骤 S605，登记在步骤 S602 中取得的根节点的数组编号，结束处理。

[0148] 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，反

复进行图 6 和图 5A ～图 5C 的处理，从而可以构建与索引关键字的集合对应的本发明的配对节点树。

[0149] 接着参照图 7A、图 7B 来说明从配对节点树的索引关键字集合删除特定的索引关键字的处理流程。图 7A，图 7B 所示的流程本身与本申请人申请的上述日本特愿 2006-187827 中提出的删除处理的流程是同样的。

[0150] 图 7A 是示出作为删除处理前级的检索处理的处理流程的图，基本上相当于在图 4A 所示的检索处理中将删除关键字作为检索关键字，不过在图 4A 的步骤 S406a 的搜索路径堆栈中存储鉴别比特位置的处理被省略。如果继续进行如频繁利用作为删除处理前级的检索处理的历史中的鉴别比特位置这样的处理，则可以追加在搜索路径堆栈中存储鉴别比特位置 的处理。

[0151] 步骤 S701 相当于在图 4A 的步骤 S401 中将检索开始节点作为根节点，步骤 S702 至步骤 S710 的处理除了上述点之外与图 4A 的步骤 S402 至步骤 S410 完全对应，所以省略说明。

[0152] 在图 7A 的步骤 S711 中对删除关键字与索引关键字进行比较，如果不相等，则在配对节点树内不存在待删除索引关键字，因而删除失败，结束处理。如果相等，则进到后面的处理，即图 7B 的步骤 S712 以下的处理。

[0153] 图 7B 是说明删除处理后级的处理流程的图。

[0154] 首先，在步骤 S712 判定在搜索路径堆栈中是否存储有 2 个以上的数组编号。所谓未存储有 2 个以上的数组编号，换句话说只存储有 1 个数组编号，该数组编号是存储了根节点的数组元素的数组编号。在该情况下，转到步骤 S718，删除在步骤 S701 中得到的根节点的数组编号涉及的节点对。然后进到步骤 S719，删除所登记的根节点的数组编号，结束处理。

[0155] 当在步骤 S712 中判定为在搜索路径堆栈中存储有 2 个以上的数组编号时，进到步骤 S713，将在步骤 S708 中得到的代表节点编号加上步骤 S707 中得到的比特值的反转值，得到相加后的数组编号。该处理是求出与存储了删除对象索引关键字的叶节点成对的节点所配置在的数组编号的处理。

[0156] 然后在步骤 S714 中，读出在步骤 S713 中得到的数组编号的数组元素的内容，在步骤 S715 中使搜索路径堆栈的堆栈指针后退 1，取出数组编号。

[0157] 接着进到步骤 S716，将步骤 S714 中读出的数组元素的内容重写到步骤 S715 中得到的数组编号的数组元素中。该处理是将作为链接到存储有删除对象索引关键字的叶节点的链接源的分支节点置换成与上述叶节点成对的节点的处理。

[0158] 最后在步骤 S717 中删除在步骤 S708 中得到的代表节点编号涉及的节点对，结束处理。

[0159] 图 8 是示出本发明的求得配对节点树（包含部分树）中存储的索引关键字的最小值的处理的流程图。图 8 所示的流程图是在本申请人申请的上述日本特愿 2006-293619 提出的索引关键字最小值的流程图中，追加了从节点取出鉴别比特位置的处理和在搜索路径堆栈中存储从节点取出的鉴别比特位置的处理。

[0160] 求出索引关键字最小值的处理相当于根据如先前所述的索引关键字在树上的配置，在树上从检索开始节点起搜索节点 [0] 直至叶节点。

[0161] 首先，从步骤 S801 的检索开始节点的数组编号的取得到步骤 S805 的节点类别的判定，分别与上述图 4A 的步骤 S401 至步骤 S405 的处理同样。

[0162] 在步骤 S805 的节点类别判定中，当节点类别被判定为分支节点时，进入步骤 S806，从节点中取出数组的代表节点编号。

[0163] 接着进入步骤 S806a，从节点中取出鉴别比特位置，在步骤 S806b，将该取出的鉴别比特位置存储在搜索路径堆栈中。

[0164] 接着在步骤 S807，将步骤 S806 中取出的代表节点编号与值“0”相加，将其结果作为新的数组编号，并返回步骤 S802。以后，重复从步骤 S802 到步骤 S807 的处理，直到在步骤 S805 中该节点被判定为叶节点为止，在步骤 S808，从叶节点取出索引关键字，结束处理。

[0165] 在上述图 8 所示的处理中，为了搜索节点 [0]，一律将代表节点编号与″ 0″相加。即，根据图 8 的处理，链接目的地节点一定是节点对中的节点 [0]，并分支到存储有更小值的索引关键字的节点。由此，能够取出树结构为如先前所述排列结构的配对节点树的最小索引关键字。

[0166] 图 9 是利用图 2B 中例示的配对节点树来说明图 8 的流程图所示的求出索引关键字最小值的处理的图。图 9 示出将数组编号 220 的节点 210a 作为检索开始节点时的链接路径以及搜索路径堆栈 310 的状态。

[0167] 因为检索开始节点是节点 210a，所以把该数组编号 220 与鉴别比特位置 230a 的比特值“0”压入搜索路径堆栈 310，接着链接到由代表节点编号 220a 表示的节点对 201b 中的作为节点 [0] 的节点 210b，把该代表节点编号 220a 和鉴别比特位置 230b 的比特值″ 1″存储在搜索路径堆栈 310 中，进而到达叶节点 210c。取出叶节点 210c 的索引关键字″ 000111″，结束处理。

[0168] 如以上所说明的那样，依次将节点对中的节点 [0] 的数组编号和分支节点的鉴别比特位置压入搜索路径堆栈 310 中。

[0169] 图 10 是示出求得配对节点树（包含部分树）中存储的索引关键字的最大值的处理的流程图。图 10 所示的流程图是在本申请人申请的上述日本特愿 2006-293619 中提出的求得索引关键字最大值的流程图中，追加了从节点取出鉴别比特位置的处理和在搜索路径堆栈中存储从节点取出的鉴别比特位置的处理。

[0170] 求出索引关键字最大值的处理相当于针对树上节点中的节点 [1] 依次进行搜索，直至叶节点。以下，针对求出任意部分树的最大索引关键字的处理，一边与求出上述最小索引关键字的处理进行比较，一边以不同点为中心进行说明。

[0171] 图 10 所示的一连串处理中，从步骤 S1001 到步骤 S1006b 以及步骤 S1008 分别与图 8 中的步骤 S801 到步骤 S806b 以及步骤 S808 对应，并执行同样的处理。与图 8 的求出最小值的处理不同之处是，在步骤 S1007 中将代表节点编号与值″ 1″相加。由此，始终链接代表节点编号所表示的节点对中的节点 [1]，依次重复从步骤 S1002 到步骤 S1007 的处理，直到到达叶节点，由此能够得到索引关键字的最大值。

[0172] 图 11 是利用图 2B 中例示的配对节点树来说明图 10 的流程图所示的求出索引关键字最大值的处理的图。如图 11 所示，与图 9 的例子相同，将检索开始节点设为数组编号 220 的节点 210a。因此，检索开始节点的数组编号是数组编号 220，在搜索路径堆栈 310 中存储数组编号 220，并读出数组编号 220 的数组元素中存储的节点 210a。节点 210a 的节点

类别 260a 是″ 0″，为分支节点，所以取出代表节点编号 220a 和鉴别比特位置 230a，并在搜索路径堆栈 310 中存储鉴别比特位置 230a 的值″ 0″。

[0173] 接着，由于在最大值检索中始终链接节点 [1]，因此将代表节点编号 220a 与值″ 1″相加后所得的值作为链接目的地的数组编号而链接到节点 211b。然后将数组编号 220a+1 存储在搜索路径堆栈 310 中，并且读出节 点 211b。

[0174] 因为节点 211b 的节点类别 261b 是″ 0″，为分支节点，所以取出代表节点编号 221b 和鉴别比特位置 231b，并在搜索路径堆栈 310 中存储鉴别比特位置 231b 的值″ 2″。

[0175] 接着，将代表节点编号 221b 与值″ 1″相加所得的值作为链接目的地的数组编号而链接到节点 211f。然后将数组编号 221b+1 存储在搜索路径堆栈 310 中，并且读出节点 211f。

[0176] 因为节点 211f 的节点类别 261f 是″ 0″，为分支节点，所以取出代表节点编号 221f 和鉴别比特位置 231f，并在搜索路径堆栈 310 中存储鉴别比特位置 231f 的值″ 3″。

[0177] 接着，将代表节点编号 221f 与值″ 1″相加所得的值作为链接目的地的数组编号而链接到节点 211h。然后将数组编号 221f+1 存储到搜索路径堆栈 310 中，并读出节点 211h。

[0178] 因为节点 211h 的节点类别 261h 是″ 1″，为叶节点，所以从节点 211h 中取出索引关键字 251h 的值″ 101100″，作为最大值，并结束处理。

[0179] 如图 4A 至图 11 所示，在执行对与检索关键字一致的索引关键字进行检索的基本动作及索引关键字的最小值 / 最大值的检索处理时，在搜索路径堆栈 310 中依次存储所参照的数组的数组编号和分支节点的鉴别比特位置。

[0180] 另外，在参照了上述图 8 以及图 10 的索引关键字最小值 / 最大值的检索处理中，对配对节点树被存储到数组中进行了说明，但显而易见，配对节点树不一定要存储在数组中，通过仅链接构成节点对的 2 个节点中的代表节点、或者仅链接配置在与代表节点相邻的存储区域中的节点，而到达叶节点，也能够进行索引关键字的最小值 / 最大值的检索。

[0181] 接着，对配对节点树的分割 / 结合方法进行说明。

[0182] 本发明的配对节点树的分割是指，在指定了由某个比特序列构成的分割关键字时，根据与该分割关键字之间的大小关系将配对节点树中包含的索引关键字分为 2 组，生成由属于各个组的索引关键字构成的 2 个配对节点树。

[0183] 针对基于大小关系的分割，在以下的说明中，分割成大于分割关键字的组和小于等于分割关键字的组，但是，即使在分割成大于等于分割关键字的组和小于分割关键字的组的情况下，也能够同样地进行分割 / 结合，这一点根据以下的说明能够容易地理解。

[0184] 总之，分割关键字是为了决定在哪里分割配对节点树而使用的关键字。

[0185] 并且，配对节点树的结合是指，根据与 2 个索引关键字的集合对应的 2 个配对节点树，生成与 2 个索引关键字的集合的并集对应的配对节点树。在本发明中，前提是 2 个索引关键字的集合的交集为空。此外，在以下的说明中，有时将配对节点树简称为树。

[0186] 图 12 是说明本发明的配对节点树的第 1 分割处理流程的图。图 12 所示的流程的步骤结构与本申请人作为在先申请的日本特愿 2006-319407 中提出的分割处理的实施例 1 的流程相同，不过步骤内容中存在不同的内容。

[0187] 在第 1 分割处理中，取出作为分割对象的处理源树（以下，有时简称为处理源。）

的索引关键字的最小值，将取出的索引关键字的最小值插入通过处理源的分割而生成的处理目标树（以下，有时简称为处理目标。），在最小值为分割关键字的期间，反复进行从处理源树中删除索引关键字的最小值的处理，由此从作为分割对象的处理源树中分割出处理目标树。

[0188] 在最初的步骤 S1201 中，将指定的分割关键字设定为处理源的分割关键字。分割关键字的指定可能是操作者进行的外部输入的情况，基于某计算机程序的处理结果的情况，基于来自远方的指令的情况等。指定的分割关键字被设定在对处理源的分割关键字进行保持的存储器的区域中。

[0189] 接着在步骤 S1202 中，将处理源的根节点设定为处理源的检索开始节点，进入步骤 S1203。

[0190] 在步骤 S1203 中，判定处理源树是否已登记。该判定结果为没有登记意味着处理源树已全部删除完毕，所以是分割关键字大于等于处理源树的索引关键字的最大值的例外情况，在该情况下结束处理。

[0191] 如果处理源树已登记，则转移到步骤 S1204，从在步骤 S1202 中设定为检索开始节点的根节点开始，执行图 8 所示的处理来得到索引关键字的最小值。此时，如先前所说明的，在搜索路径堆栈中存储有数组编号和鉴别比特位置。

[0192] 接着，进入步骤 S1205，判定在步骤 S1204 所得到的最小值是否大于分割关键字。如果最小值大于分割关键字，则树的分割完成，所以结束处理，如果最小值不大于分割关键字，则执行以下说明的步骤 S1206 ～步骤 S1209 的处理目标树的生成和从处理源树删除节点的处理，返回步骤 S1203。

[0193] 在步骤 S1206 中，将在步骤 S1204 中得到的最小值设定为处理目标插入关键字。

[0194] 接着，在步骤 S1207 中，通过图 5A ～图 5C、图 6 所示的树的生成、插入处理，执行基于插入关键字的处理目标树生成。

[0195] 本发明的配对节点树的第 1 分割处理中的生成、插入处理因为可以参照在先前的最小值检索处理中存储在搜索路径堆栈中的鉴别比特位置，所以与本申请人作为在先申请的日本特愿 2006-319407 中提出的分割处理的实施例 1 相比，能够加快处理。

[0196] 接着，在步骤 S1208 中，将步骤 S1207 中的插入关键字设定为处理源的删除关键字，在步骤 S1209 中，通过图 7A、图 7B 所示的删除处理，从处理源树中删除包含删除关键字的叶节点。

[0197] 在上述分割处理的说明中，从处理源的索引关键字的最小值起依次进行删除，但是，对于本领域技术人员来说，显然，同样能够从索引关键字的最大值起依次进行删除。该情况下，步骤 S1204 成为求出索引关键字的最大值的处理，步骤 S1205 成为最大值与分割关键字的大小关系的判定处理，在步骤 S1206 中将最大值设定为处理目标的插入关键字。

[0198] 以上说明了分割处理，但是，也能够通过图 12 所示的处理流程来执行结合处理。

[0199] 关于结合处理，将待结合的两个树中的某一个作为处理源树，如果分割关键字大于等于处理源树的索引关键字的最大值，则相当于上述的 例外处理，删除处理源树，与处理目标树结合。另外，在处理源树的索引关键字的最大值未知的情况下，预先通过图 10 所示的最大值检索处理来求出分割关键字。

[0200] 然后，由于处理源的分割关键字大于等于处理源树的索引关键字的最大值，因此

在步骤 S1205 的大小比较中，分割关键字始终大于最小值而分支到步骤 S1206，所以可省略步骤 S1205。如果这样，则设定分割关键字没有意义，所以也不需要步骤 S1202，仅通过反复进行最小值检索、插入处理和删除处理，就能够进行结合处理。

[0201] 另外，如分割处理所述，显然也可以通过反复进行最大值检索、插入处理和删除处理来进行结合处理。

[0202] 图 13 是说明本发明的配对节点树的第 2 分割处理流程的图。图 13 所示流程的步骤结构与本申请人作为在先申请的日本特愿 2006-319407 中提出的分割处理的实施例 2 的流程相同，不过步骤内容中存在不同的内容。

[0203] 关于第 2 分割处理，以索引关键字为单位来进行插入删除，这一点与第 1 分割处理的相同，在搜索待插入删除索引关键字时利用搜索路径堆栈，减少了在执行插入处理以及删除处理时的处理步骤数量。

[0204] 图 13 中从步骤 S1301 到步骤 S1306 的处理与图 12 所示的第 1 分割处理的从步骤 S1201 到步骤 S1206 的处理完全相同，因此省略说明。另外，在步骤 S1304 的最小值检索处理中，在搜索路径堆栈中存储鉴别比特位置和数组编号，这也与步骤 S1204 中的最小值检索同样。

[0205] 在步骤 S1307 中，利用插入关键字在处理目标中插入节点。该处理是与图 12 所示的步骤 S1207 的插入处理不同的、第 2 分割处理特有的处理，后面参照图 14、图 15A、图 15B 来详细说明。

[0206] 接着，在步骤 S1308 中，将包含步骤 S1304 中得到的最小值的节点设定为处理源删除节点，在步骤 S1309 中，将删除节点从处理源删除，得到处理源父节点，该处理源父节点复制了与删除节点成对的节点的内容。

[0207] 接着，在步骤 S1310 中，将步骤 S1309 中得到的处理源父节点设定 为处理源检索开始节点，返回步骤 S1303。

[0208] 如以后说明的那样，处理源父节点是删除节点的最近的上位分支节点。删除节点包含处理源索引关键字的最小值，根据上述的索引关键字的顺序性，下次检索到的最小值位于处理源父节点的下位。

[0209] 因此，第 2 次及第 2 次之后的步骤 S1304 中的最小值检索的检索开始节点不是根节点而是处理源父节点，因此能够减少处理步骤数量。

[0210] 后面参照图 16 来详细说明步骤 S1308 至步骤 S1309 和处理源父节点。

[0211] 图 14 是说明与图 13 所示的步骤 S1306 和步骤 S1307 对应的节点插入处理流程的图。

[0212] 最初的步骤 S1401 中将插入关键字设定为处理目标插入关键字的步骤是与图 13 所示的步骤 S1306 对应的处理，将在图 13 所示的步骤 S1304 中得到的最小值设定到处理目标插入关键字设定区域中。

[0213] 接着，在步骤 S1402 中，判定处理目标是否已登记完毕。

[0214] 如果未登记完毕，则转移到步骤 S1403，在步骤 S1403 中，将包含插入关键字的节点对设定为处理目标根节点，作为处理目标根节点进行登记。接着，进入步骤 S1404，将处理目标根节点设定为插入完毕节点，结束处理。

[0215] 如果上述步骤 S1402 的判定结果为已登记完毕，则转移到步骤 S1405。在步骤

S1405 中，判定插入完毕节点是否已设定完毕。该判定处理在后面说明的树结合处理中是必要的。在树分割处理中，在最初的插入处理中，在步骤 S1404 中将根节点设定为插入完毕节点，所以步骤 S1406 的判定始终为“是”。因此，如果仅是分割处理，则步骤 S1405 和步骤 S1407 不是必须的。

[0216] 如果步骤 S1405 的判定结果为“是”，则进入步骤 S1406，将设定为插入完毕节点的节点设定为处理目标检索开始节点，进入步骤 S1408。

[0217] 如果步骤 S1405 的判定结果为“否”，则进入步骤 S1407，将根节点设定为处理目标检索开始节点，进入步骤 S1408。

[0218] 在步骤 S1408 中，通过图 10 所示的最大值检索处理，从步骤 S1406 或步骤 S1407 中设定的检索开始节点开始，求出处理目标索引关键字的最大值。此时，如之前说明的那样，在搜索路径堆栈中存储有数组编号和鉴别比特位置。

[0219] 接着，在步骤 S1409 中，根据在步骤 S1401 中设定的插入关键字和在步骤 S1407 中得到的最大值，求出待插入节点对的处理目标父节点，在该父节点中插入包含所述插入关键字的节点对。

[0220] 接着，进入步骤 S1410，将在步骤 S1409 中插入了包含插入关键字的节点对的处理目标父节点设定为插入完毕节点，结束处理。

[0221] 接着，详细说明上述步骤 S1403 和步骤 S1409。

[0222] 图 15A 是说明图 14 所示的步骤 S1403 的根节点设定处理的流程图。

[0223] 首先，在步骤 S1501 中，从数组中取得节点对为空的代表节点编号。

[0224] 接着，在步骤 S1502 中，将步骤 S1501 中取得的代表节点编号加上值“0”而得到的数组编号设定为插入节点的数组编号。

[0225] 进入步骤 S1503，为了形成叶节点，在节点类别中设定叶，在索引关键字中设定插入关键字，存储到步骤 S1502 中设定的数组编号的数组元素中。

[0226] 接着，在步骤 S1504 中，将插入节点的数组编号登记为根节点的数组编号，结束处理。

[0227] 另外，关于上述步骤 S1501 ～步骤 S1504 的处理，在参照图 6 说明的配对节点树的生成处理中，对应于根节点的数组编号未登记完毕时的步骤 S602 至步骤 S605。

[0228] 图 15B 是说明图 14 所示的步骤 S1409 的求出待插入节点对的处理目标父节点并在该父节点中插入包含所述插入关键字的节点对的处理的流程图。图 15B 所示的流程是先前参照图 5C 而示出的插入处理的应用，步骤 S1505 的处理对应于图 5C 所示的步骤 S516 和步骤 S517 的处理，其中，步骤 S1505 的处理为 :将插入关键字和最大值作为比特序列进行比较而求出从上位第 0 比特开始所发现的第一个不一致比特位置，将其设定在存储差分比特位置的区域中。

[0229] 步骤 S1505 之后进入步骤 S1506a。另外，步骤 S1506a 至步骤 S1512 的处理块相当于由图 5C 所示的步骤 S519a 至步骤 S523 构成的处理块，步骤 S1513 至步骤 S1518 的处理相当于图 5C 所示的步骤 S524 至步骤 S528。

[0230] 在步骤 S1506a 中，使先前图 14 所示的步骤 S1408 的最大值检索处理中的处理目标搜索路径堆栈的堆栈指针后退 1，取出鉴别比特位置。

[0231] 接着在步骤 S1509 中，比较在步骤 S1505 中设定的差分比特位置与在步骤 S1506a

中取出的鉴别比特位置，判定鉴别比特位置与差分比特位置相比前者是否为上位的位置关系。

[0232] 如果该判定结果为鉴别比特位置与差分比特位置相比是上位的位置关系，则进入步骤 S1512，使处理目标搜索路径堆栈的堆栈指针前进 1，取出数组编号，将其设定在父节点的数组编号区域中，转移到步骤 S1513。

[0233] 如果步骤 S1509 的判定结果为鉴别比特位置与差分比特位置相比不是上位的位置关系，则进入步骤 S1510。

[0234] 在步骤 S1510 中，判定处理目标搜索路径堆栈的堆栈指针是否指向根节点的数组编号。

[0235] 在判定结果为处理目标搜索路径堆栈的堆栈指针没有指向根节点的数组编号的情况下，返回步骤 S1506。

[0236] 在判定结果为处理目标搜索路径堆栈的堆栈指针指向根节点的数组编号的情况下，在步骤 S1511 中，从处理目标搜索路径堆栈中取出堆栈指针所指向的数组编号，将其设定存储处理目标父节点的数组编号的区域中，转移到步骤 S1513。

[0237] 在以上说明的步骤 S1506a 至步骤 S1512 的处理中，因为能够参照在先前的最大值检索处理中存储在搜索路径堆栈中的鉴别比特位置，所以与本申请人作为在前申请的日本特愿 2006-319407 提出的分割处理的实施例 2 相比，可加快处理。

[0238] 在步骤 S1513 中，从数组中取得节点对为空的代表节点编号。

[0239] 接着在步骤 S1514 中，将代表节点编号加 1 得到的数组编号设定到存储插入节点的数组编号的区域中。

[0240] 接着在步骤 S1515 中，为了形成叶节点，在节点类别中设定叶，在 索引关键字中设定插入关键字，存储到步骤 S1514 中设定的数组编号的数组元素中。

[0241] 接着在步骤 S1516 中，将代表节点编号加 0 得到的数组编号设定到对与插入节点成对的对节点的数组编号进行存储的区域中。

[0242] 接着在步骤 S1517 中，读出在步骤 S1511 或步骤 S1512 中设定的处理目标父节点的数组编号所指定的数组元素的内容，将其存储到步骤 S1516 中设定的对节点的数组编号所指定的数组元素中。

[0243] 接着在步骤 S1518 中，为了形成分支节点，在节点类别中设定分支，在鉴别比特位置中设定步骤 S1505 中设定的差分比特位置，在代表节点编号中设定步骤 S1516 中设定的对节点的数组编号，存储到步骤 S1511 或步骤 S1512 中设定的处理目标父节点的数组编号所指定的数组元素中，结束处理。

[0244] 图 16 是示出对图 13 所示的处理流程中的删除处理进行说明的处理流程例的图。

[0245] 最初的步骤即步骤 S1601 相当于图 13 所示的步骤 S1308。删除节点中包含的删除关键字是在图 13 所示的步骤 S1304 中通过最小值检索而求出的，所以，在处理源搜索路径堆栈中存储有存储了删除关键字的数组元素的数组编号，因此根据该数组编号来读出删除节点并将其设定到存储删除节点的区域中。

[0246] 在接下来的步骤 S1602 中，判定在处理源搜索路径堆栈中是否存储了 2 个以上的数组编号。该步骤 S1602 到步骤 S1608 的处理对应于图 7B 所示的删除处理后级的处理。

[0247] 如果在处理源搜索路径堆栈中没有存储 2 个以上的数组编号，则转移到步骤

S1607，删除在步骤 S1601 中设定的删除节点的数组编号所指定的节点对，进到步骤 S1608，删除根节点的数组编号，结束处理。

[0248] 如果在处理源搜索路径堆栈中存储了 2 个以上的数组编号，则进到步骤 S1603，求出与删除节点成对的节点的数组编号，将其设定到存储对节点的数组编号的区域中。

[0249] 接着在步骤 S1604 中，使处理目标搜索路径堆栈的堆栈指针后退 1，取出数组编号，将其设定到作为删除节点的最近上位分支节点的处理源父节点的数组编号存储区域中。

[0250] 接着在步骤 S1605 中，读出在步骤 S1603 中设定的对节点的数组编号所指定的数组元素的内容，将其存储到步骤 S1604 中设定的处理源父节点的数组编号所指定的数组元素中。

[0251] 接着在步骤 S1606 中，将删除节点的代表节点编号所指定的节点对删除，结束处理。

[0252] 以上，对第 2 分割处理进行了说明，但在第 2 分割处理中也可以与第 1 分割处理的情况相同，从索引关键字的最大值开始依次进行删除。

[0253] 另外，与第 1 分割处理的情况相同，能够在树结合处理中应用分割处理的处理流程。可以进行以下处理，将待结合的两个树中的某个作为处理源树，将分割关键字设为处理源树的索引关键字的最大值以上或最小值以下，来进行处理源树的删除处理，并且将已删除的节点插入到处理目标树中。

[0254] 显然，可以通过使计算机执行以上说明的本发明实施方式的比特序列检索处理、索引关键字插入处理、配对节点树分割结合处理以及与这些等同的处理的程序，来实现本发明的比特序列检索方法、索引关键字插入方法、配对节点树分割方法以及结合方法。

[0255] 因此，本发明的实施方式包括上述程序、以及存储有程序的计算机可读存储介质。

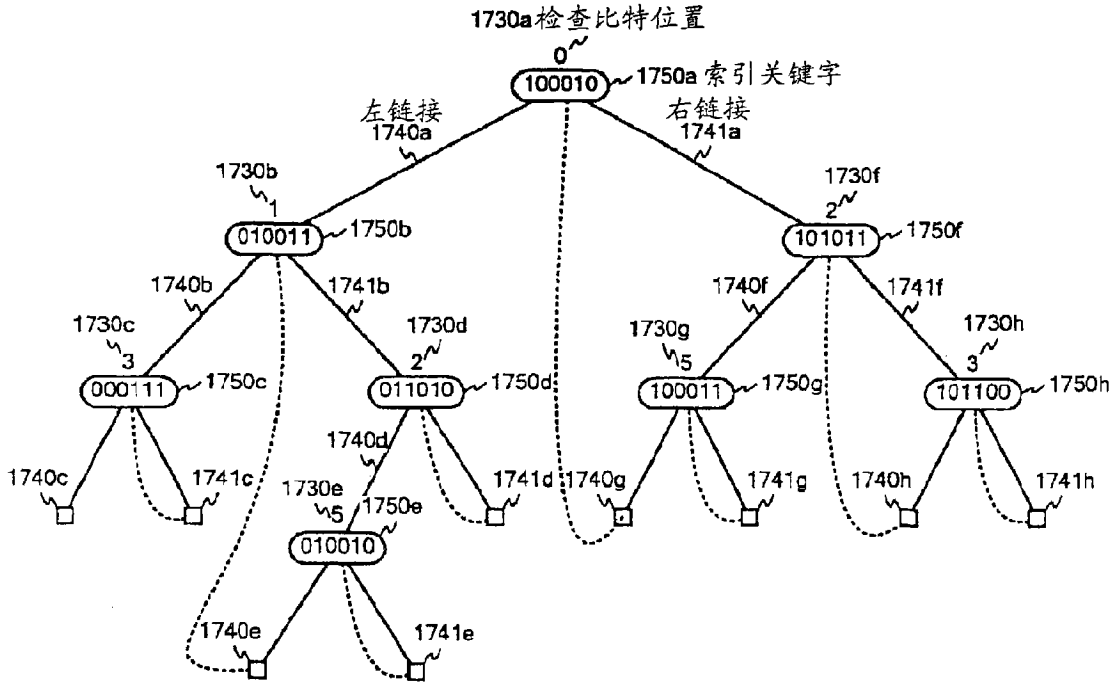

1730a检查比特位置
0
100010
1750a索引关键字
左链接
1740a
右链接
1741a
1730b
1
010011
1750b
1730f
2
101011
1750f
1740b
1741b
1740f
1741f
1730c
3
000111
1750c
1730d
2
011010
1750d
1730g
5
100011
1750g
1730h
3
101100
1750h
1740d
1740c
1741c
1730e
5
1750e
1741d
1740g
1741g
1740h
1741h
010010
1740e
1741e

图 1

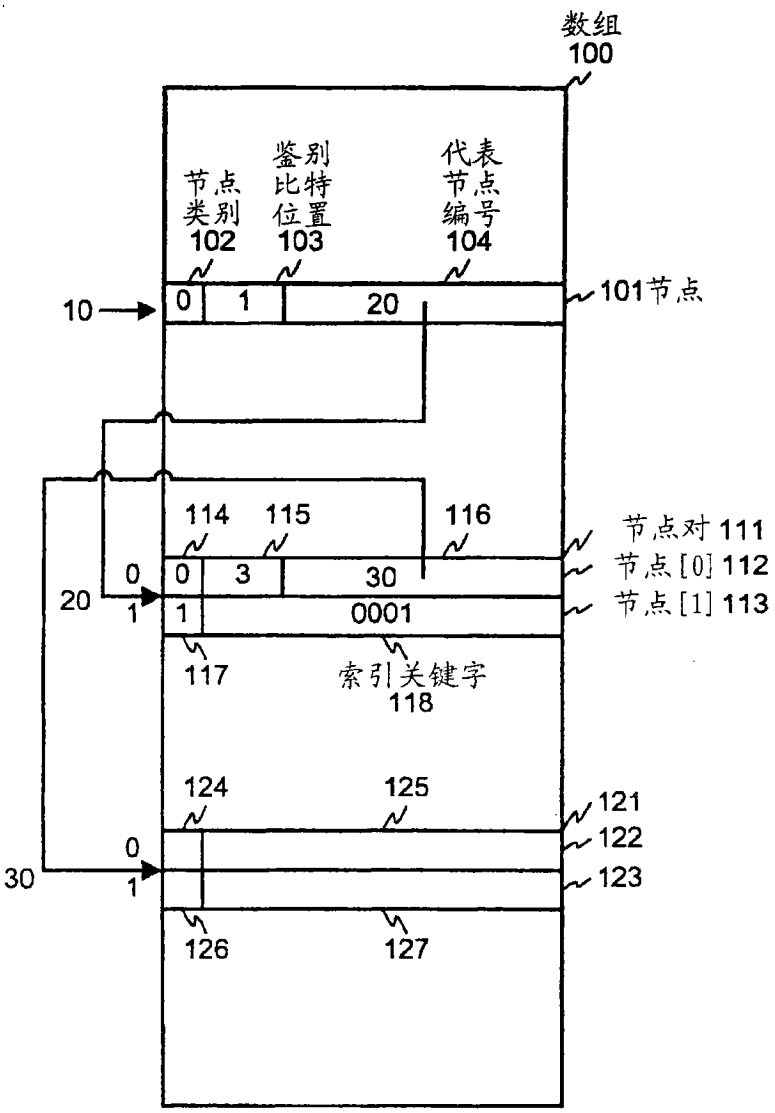

数组
100
节点类别
102
鉴别比特位置
103
代表节点编号
104
101节点
10
0
1
20
114
115
116
节点对 111
节点[0] 112
节点[1] 113
0
3
30
20
1
1
0001
117
索引关键字
118
124
125
121
122
123
0
30
1
126
127

图 2A

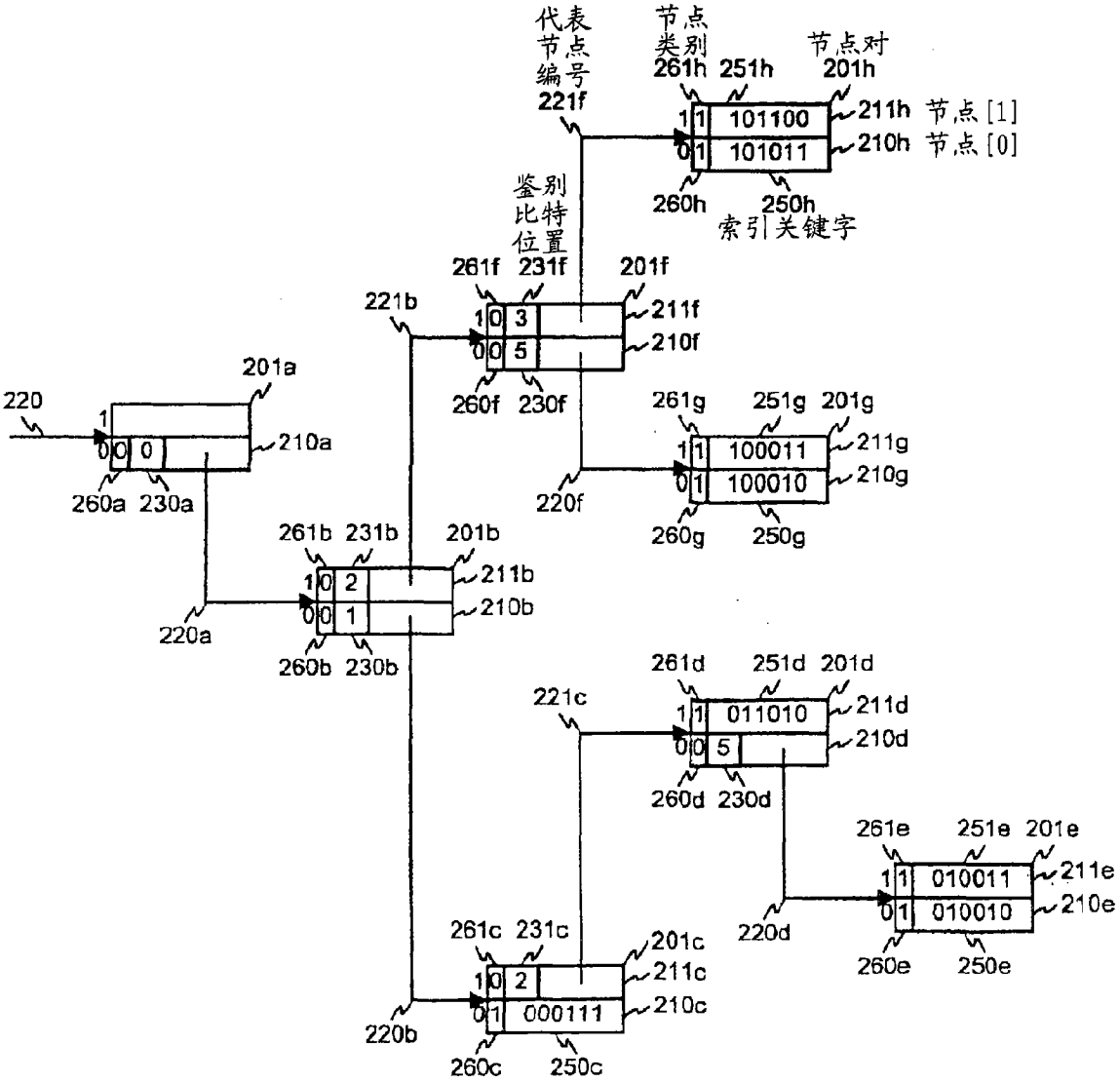

代表节点编号
221f
节点类别
261h 251h
节点对
201h
101100
211h 节点[1]
101011
210h 节点[0]
260h
250h
索引关键字
鉴别比特位置
261f 231f
201f
221b
3
211f
5
210f
260f 230f
201a
220
210a
260a 230a
261g 251g 201g
100011
211g
100010
210g
220f
260g 250g
261b 231b
201b
2
211b
1
210b
220a
260b 230b
261d 251d 201d
221c
011010
211d
5
210d
260d 230d
261e 251e 201e
010011
211e
010010
210e
220d
260e 250e
261c 231c
201c
2
211c
000111
210c
220b
260c 250c

图 2B

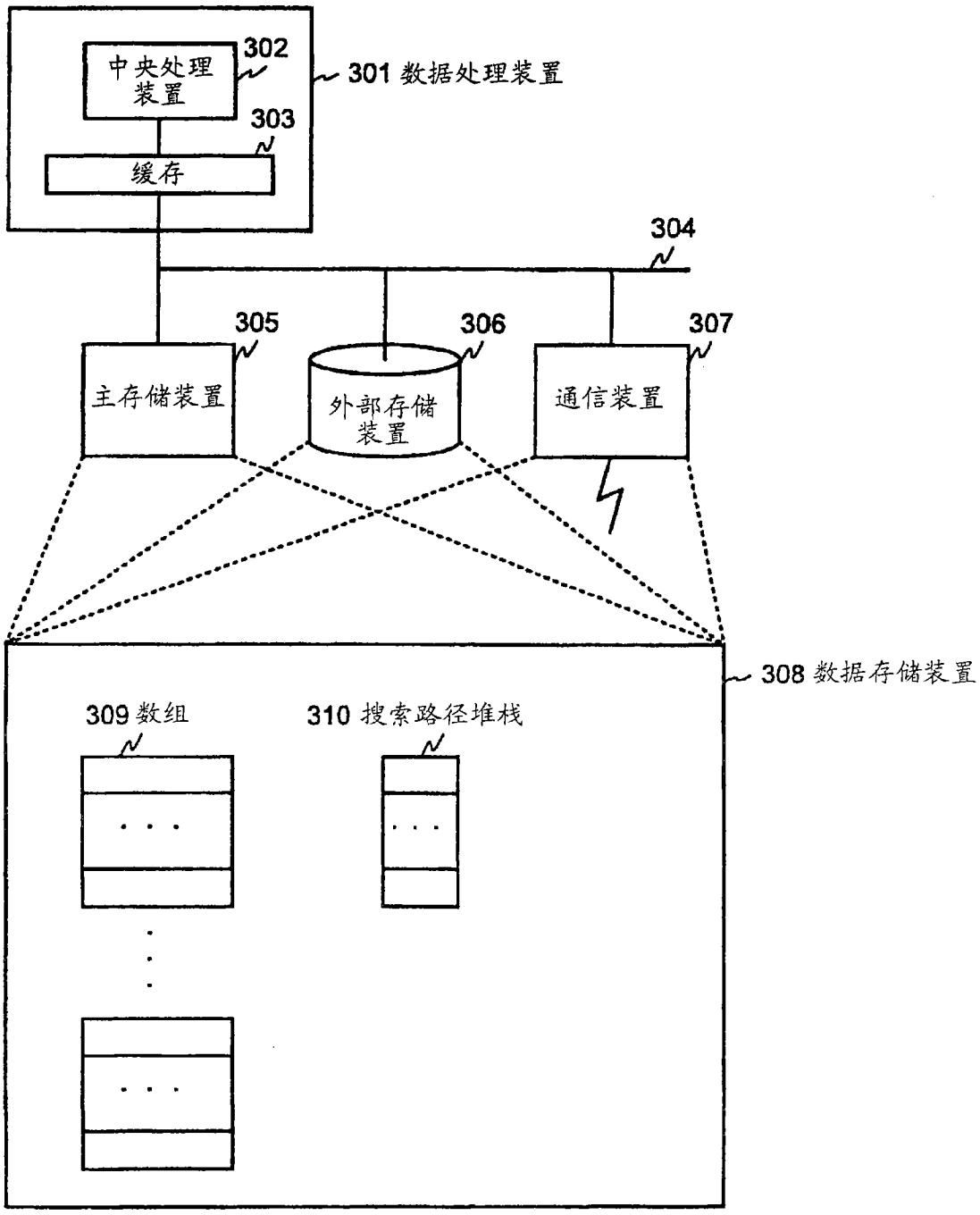

中央处理装置
302
301 数据处理装置
303
缓存
304
305
306
307
主存储装置
外部存储装置
通信装置
308 数据存储装置
309 数组
310 搜索路径堆栈

图 3

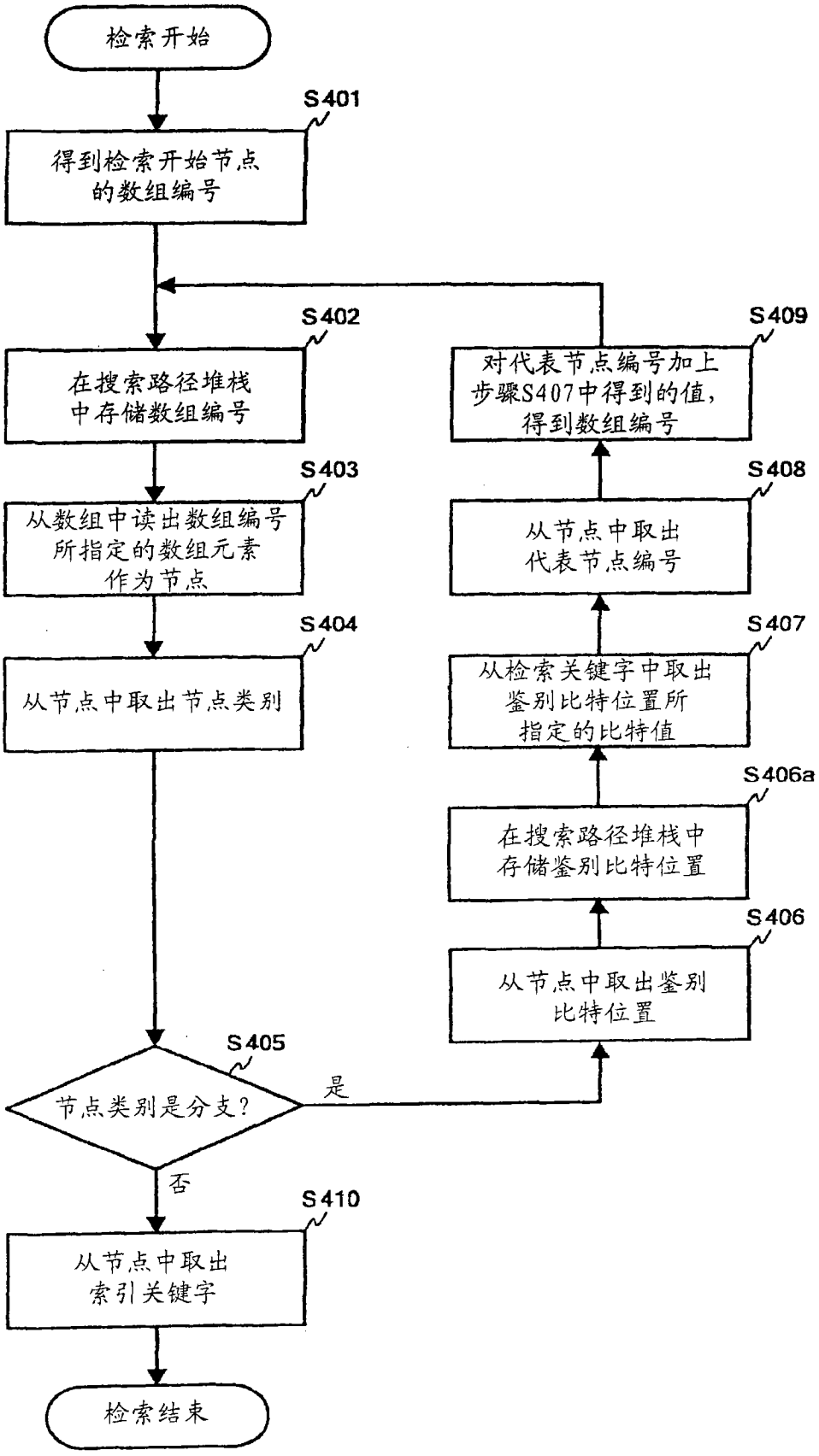

检索开始
S401
得到检索开始节点的数组编号
S402
在搜索路径堆栈中存储数组编号
S403
从数组中读出数组编号所指定的数组元素作为节点
S404
从节点中取出节点类别
S405
节点类别是分支?
是
否
S406
从节点中取出鉴别比特位置
S406a
在搜索路径堆栈中存储鉴别比特位置
S407
从检索关键字中取出鉴别比特位置所指定的比特值
S408
从节点中取出代表节点编号
S409
对代表节点编号加上步骤S407中得到的值，得到数组编号
S410
从节点中取出索引关键字
检索结束

图 4A

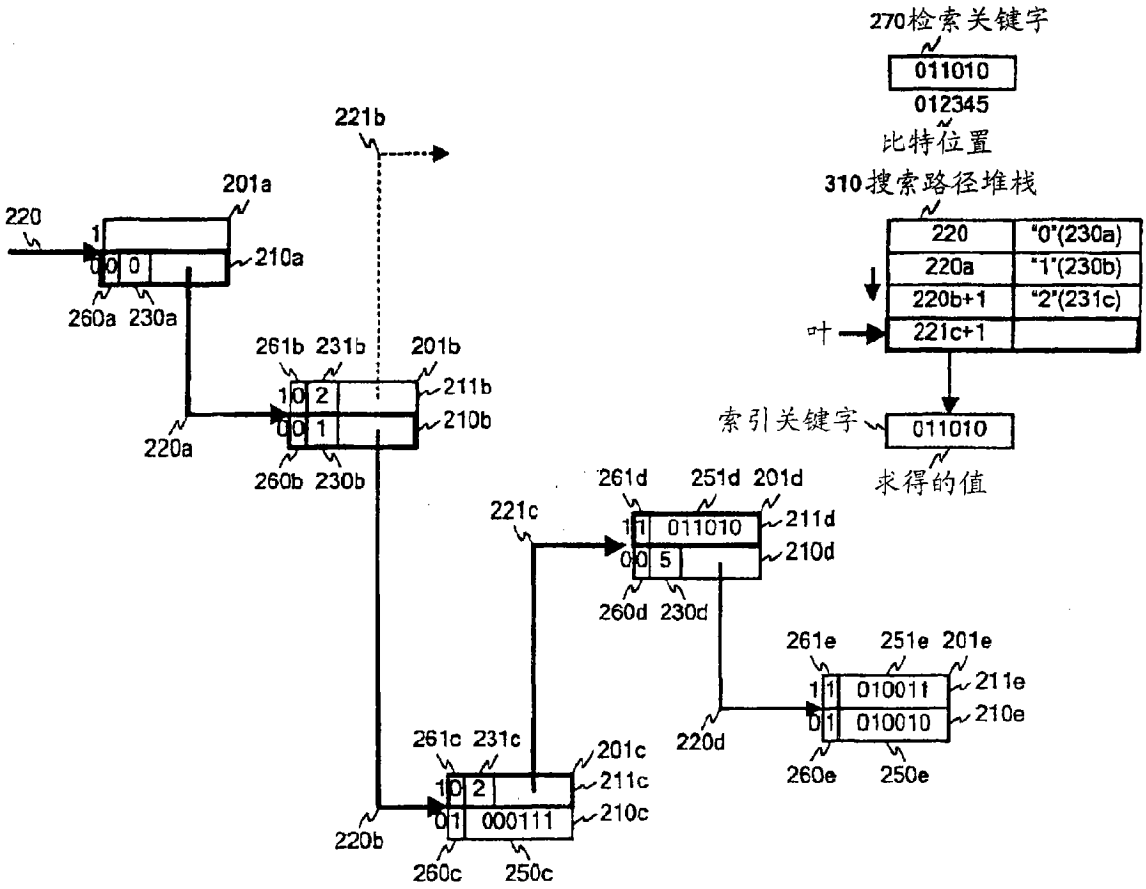

270检索关键字
011010
012345
比特位置
310搜索路径堆栈
220
"0"(230a)
220a
"1"(230b)
220b+1
"2"(231c)
叶
221c+1
索引关键字
011010
求得的值
221b
201a
220
210a
260a 230a
261b 231b 201b
211b
210b
220a
260b 230b
221c
261d 251d 201d
011010
211d
210d
260d 230d
261e 251e 201e
010011
211e
010010
210e
220d
260e 250e
261c 231c 201c
211c
210c
220b
000111
260c 250c

图 4B

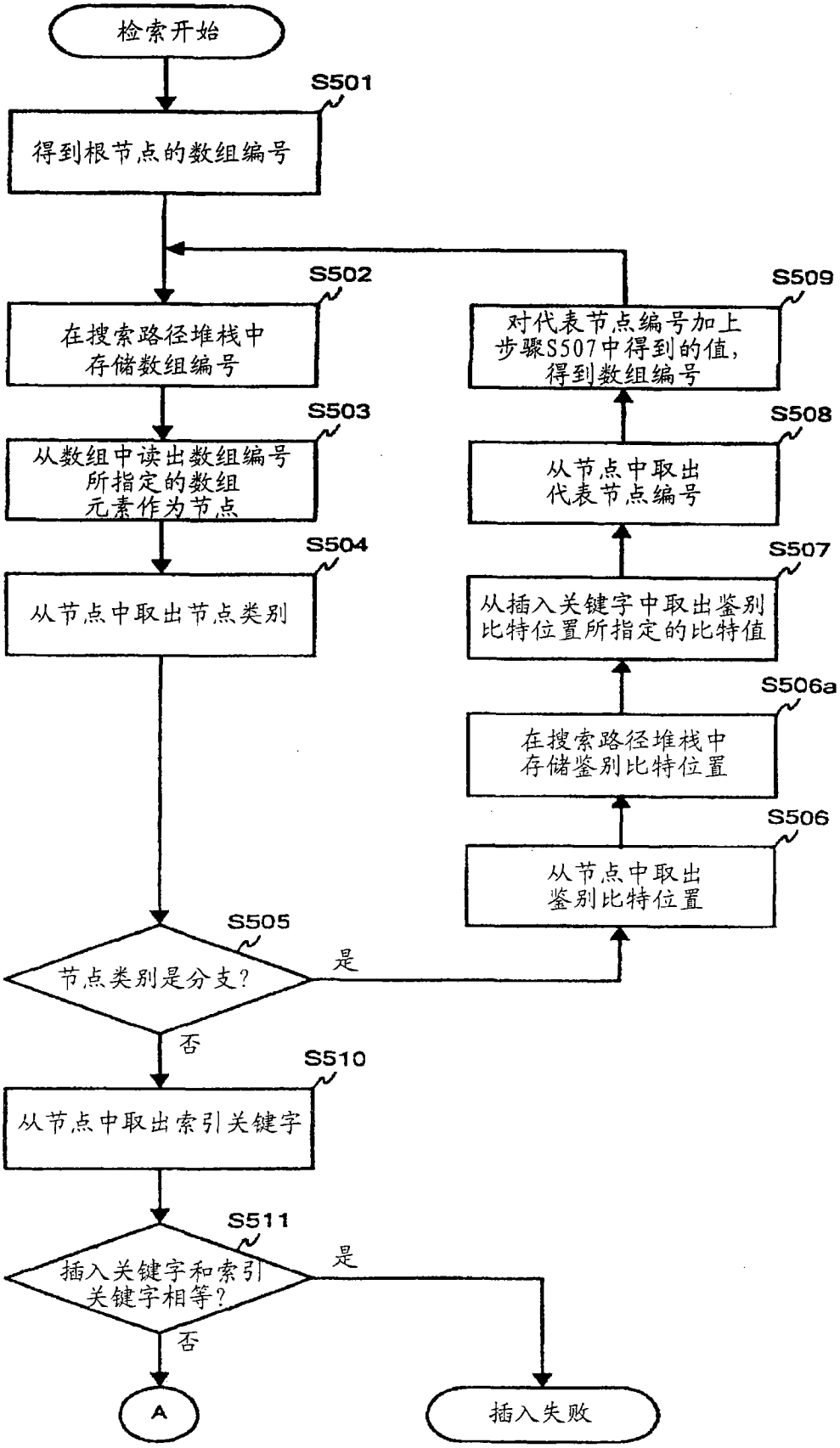

检索开始
S501
得到根节点的数组编号
S502
在搜索路径堆栈中
存储数组编号
S503
从数组中读出数组编号
所指定的数组
元素作为节点
S504
从节点中取出节点类别
S505
节点类别是分支?
是
否
S506
从节点中取出
鉴别比特位置
S506a
在搜索路径堆栈中
存储鉴别比特位置
S507
从插入关键字中取出鉴别
比特位置所指定的比特值
S508
从节点中取出
代表节点编号
S509
对代表节点编号加上
步骤S507中得到的值,
得到数组编号
S510
从节点中取出索引关键字
S511
插入关键字和索引
关键字相等?
是
否
A
插入失败

图 5A

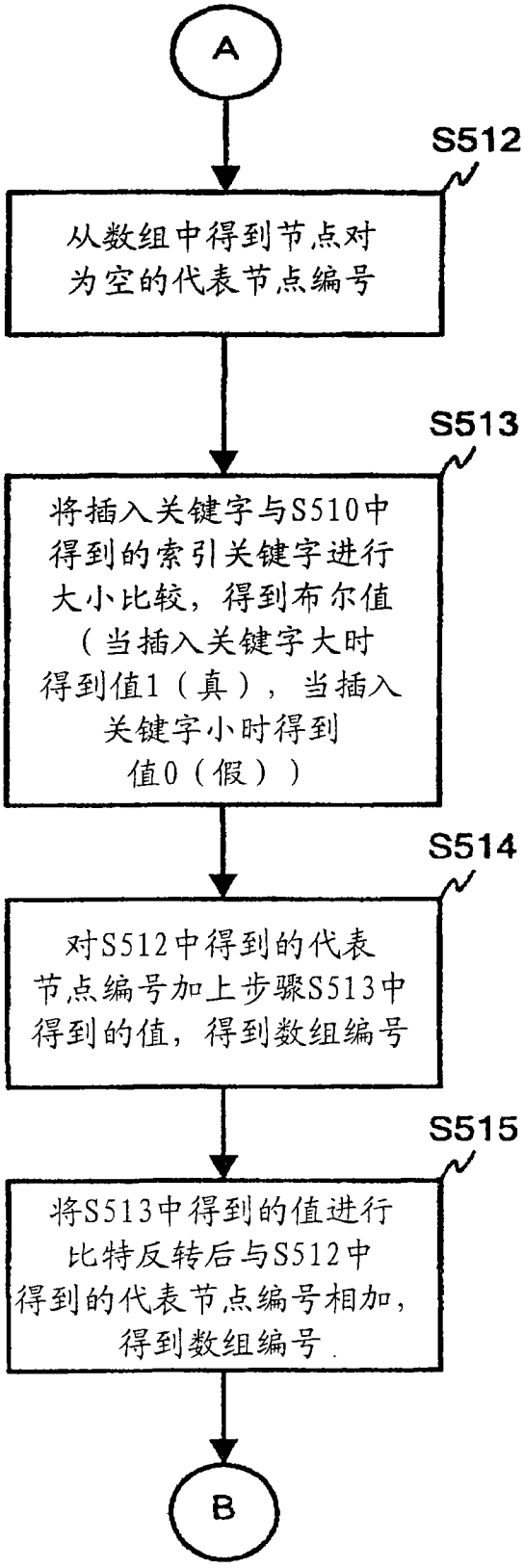

A
S512
从数组中得到节点对为空的代表节点编号
S513
将插入关键字与S510中得到的索引关键字进行大小比较，得到布尔值（当插入关键字大时得到值1（真），当插入关键字小时得到值0（假））
S514
对S512中得到的代表节点编号加上步骤S513中得到的值，得到数组编号
S515
将S513中得到的值进行比特反转后与S512中得到的代表节点编号相加，得到数组编号
B

图 5B

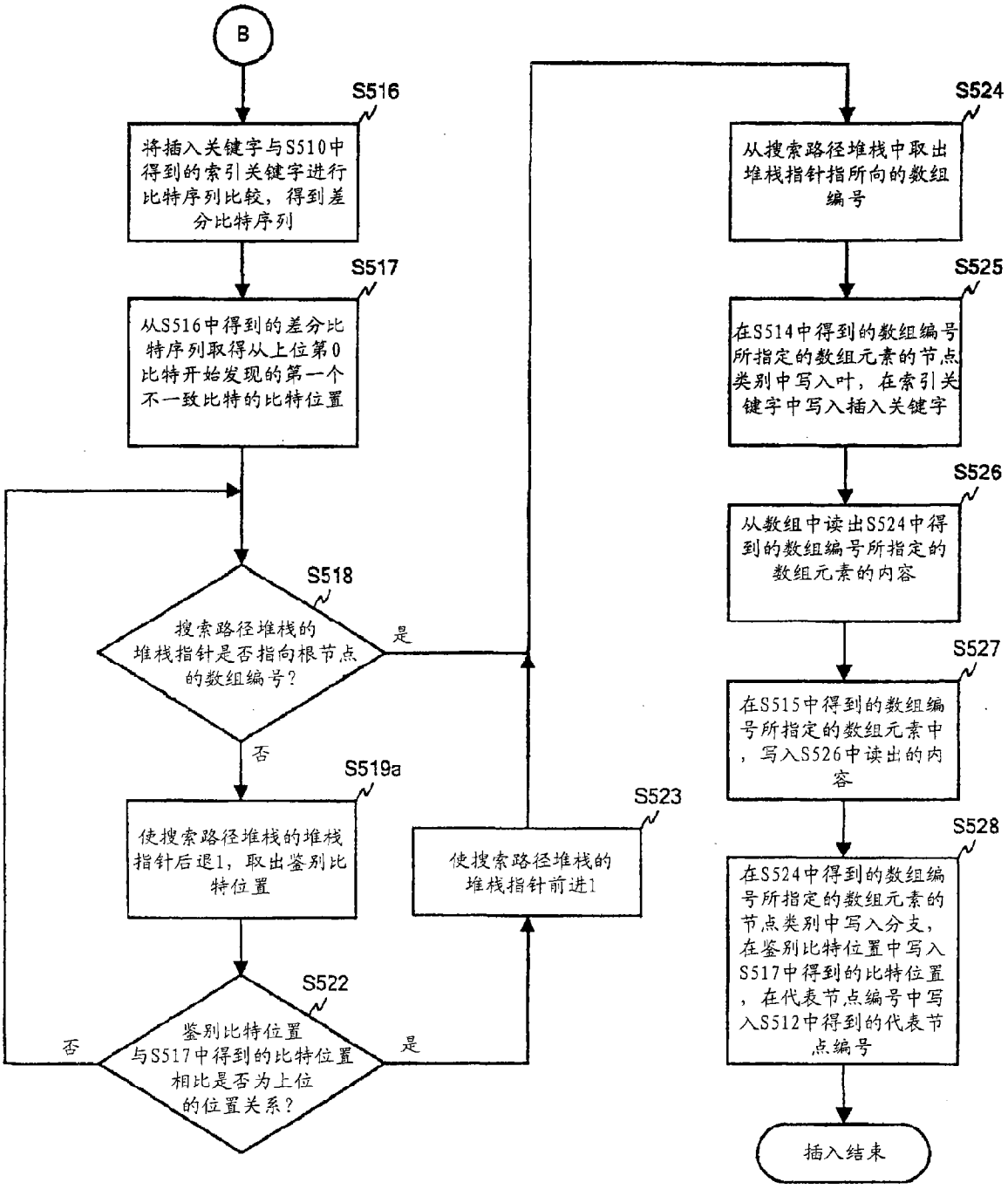

B
S516
将插入关键字与S510中得到的索引关键字进行比特序列比较，得到差分比特序列
S517
从S516中得到的差分比特序列取得从上位第0比特开始发现的第一个不一致比特的比特位置
S518
搜索路径堆栈的堆栈指针是否指向根节点的数组编号？
是
否
S519a
使搜索路径堆栈的堆栈指针后退1，取出鉴别比特位置
S522
鉴别比特位置与S517中得到的比特位置相比是否为上位的位置关系？
否
是
S523
使搜索路径堆栈的堆栈指针前进1
S524
从搜索路径堆栈中取出堆栈指针指所向的数组编号
S525
在S514中得到的数组编号所指定的数组元素的节点类别中写入叶，在索引关键字中写入插入关键字
S526
从数组中读出S524中得到的数组编号所指定的数组元素的内容
S527
在S515中得到的数组编号所指定的数组元素中，写入S526中读出的内容
S528
在S524中得到的数组编号所指定的数组元素的节点类别中写入分支，在鉴别比特位置中写入S517中得到的比特位置，在代表节点编号中写入S512中得到的代表节点编号
插入结束

图 5C

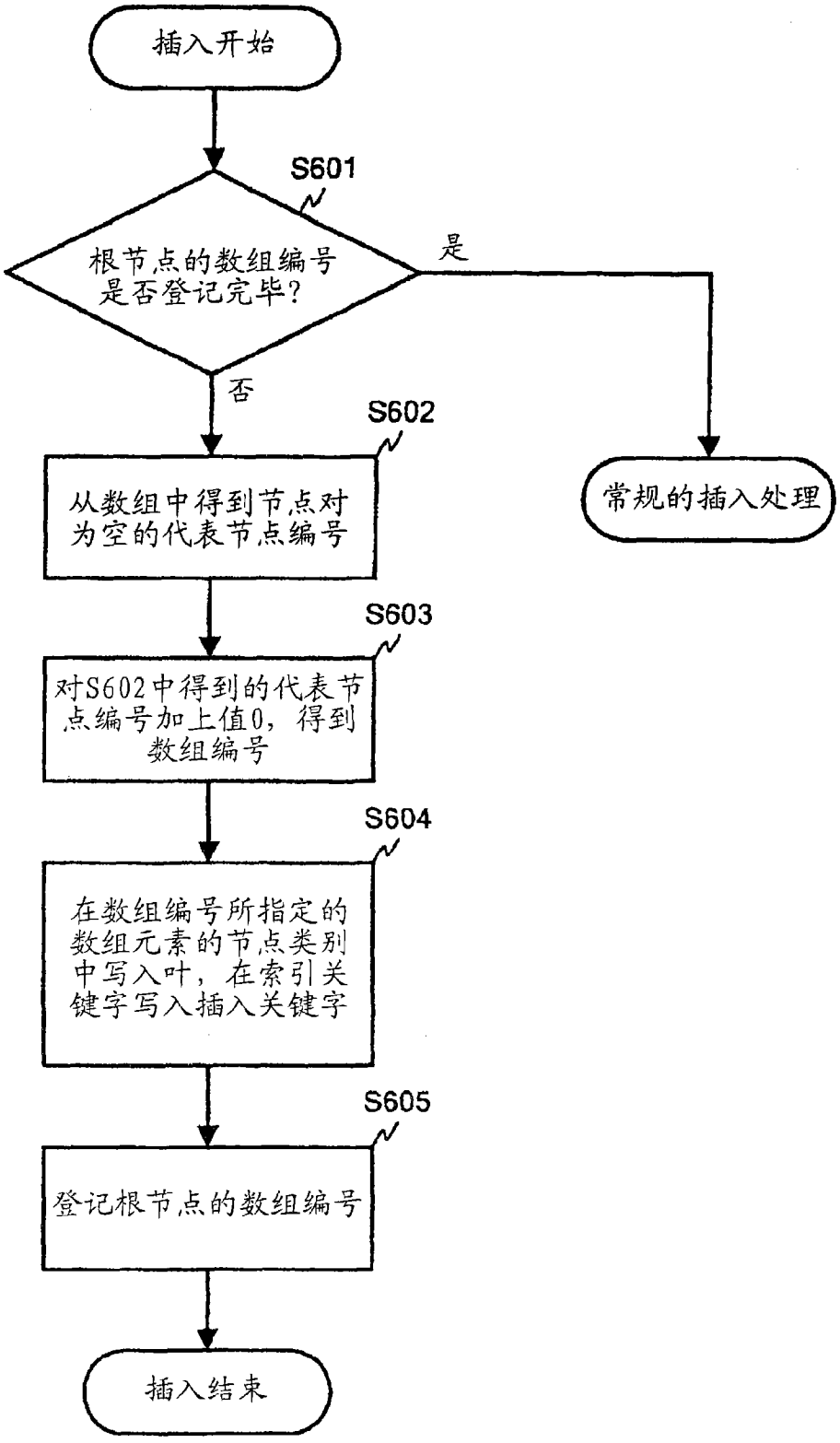

插入开始
S601
根节点的数组编号是否登记完毕?
是
否
常规的插入处理
S602
从数组中得到节点对为空的代表节点编号
S603
对S602中得到的代表节点编号加上值0，得到数组编号
S604
在数组编号所指定的数组元素的节点类别中写入叶，在索引关键字写入插入关键字
S605
登记根节点的数组编号
插入结束

图 6

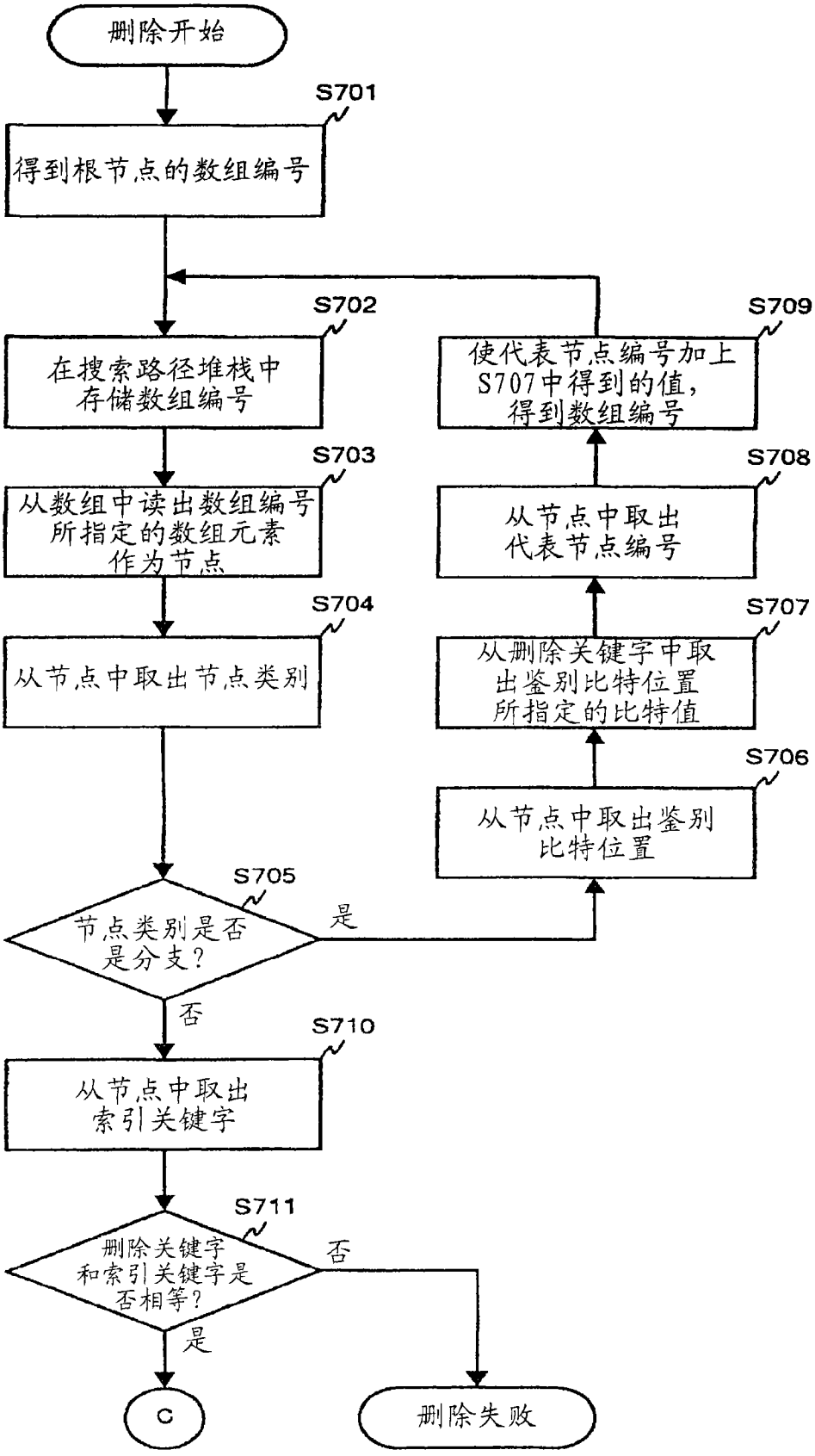

删除开始
S701
得到根节点的数组编号
S702
在搜索路径堆栈中存储数组编号
S703
从数组中读出数组编号所指定的数组元素作为节点
S704
从节点中取出节点类别
S705
节点类别是否是分支？
是
否
S706
从节点中取出鉴别比特位置
S707
从删除关键字中取出鉴别比特位置所指定的比特值
S708
从节点中取出代表节点编号
S709
使代表节点编号加上S707中得到的值，得到数组编号
S710
从节点中取出索引关键字
S711
删除关键字和索引关键字是否相等？
否
是
C
删除失败

图 7A

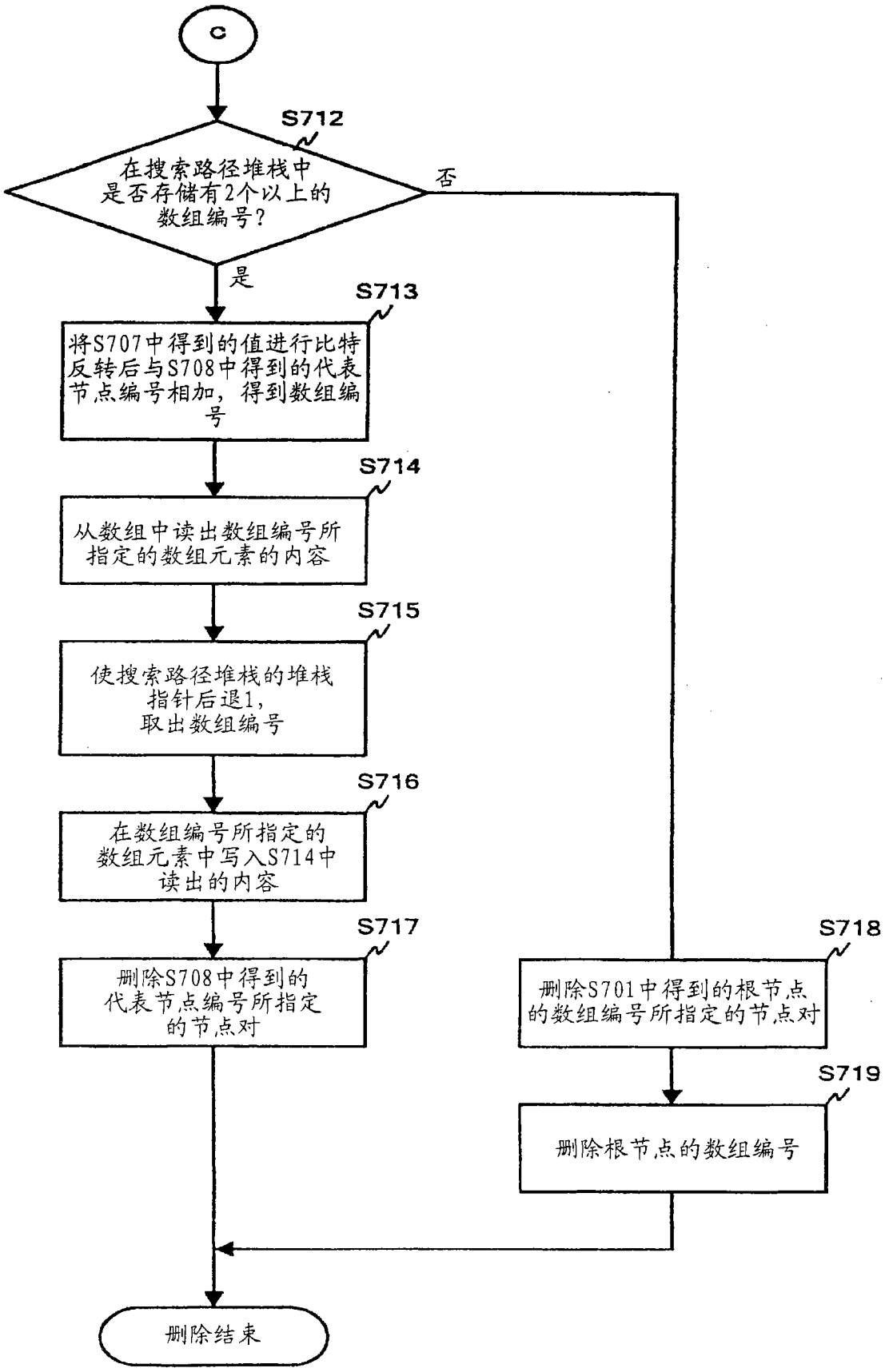

C
S712
在搜索路径堆栈中是否存储有2个以上的数组编号？
否
是
S713
将S707中得到的值进行比特反转后与S708中得到的代表节点编号相加，得到数组编号
S714
从数组中读出数组编号所指定的数组元素的内容
S715
使搜索路径堆栈的堆栈指针后退1，取出数组编号
S716
在数组编号所指定的数组元素中写入S714中读出的内容
S717
删除S708中得到的代表节点编号所指定的节点对
S718
删除S701中得到的根节点的数组编号所指定的节点对
S719
删除根节点的数组编号
删除结束

图 7B

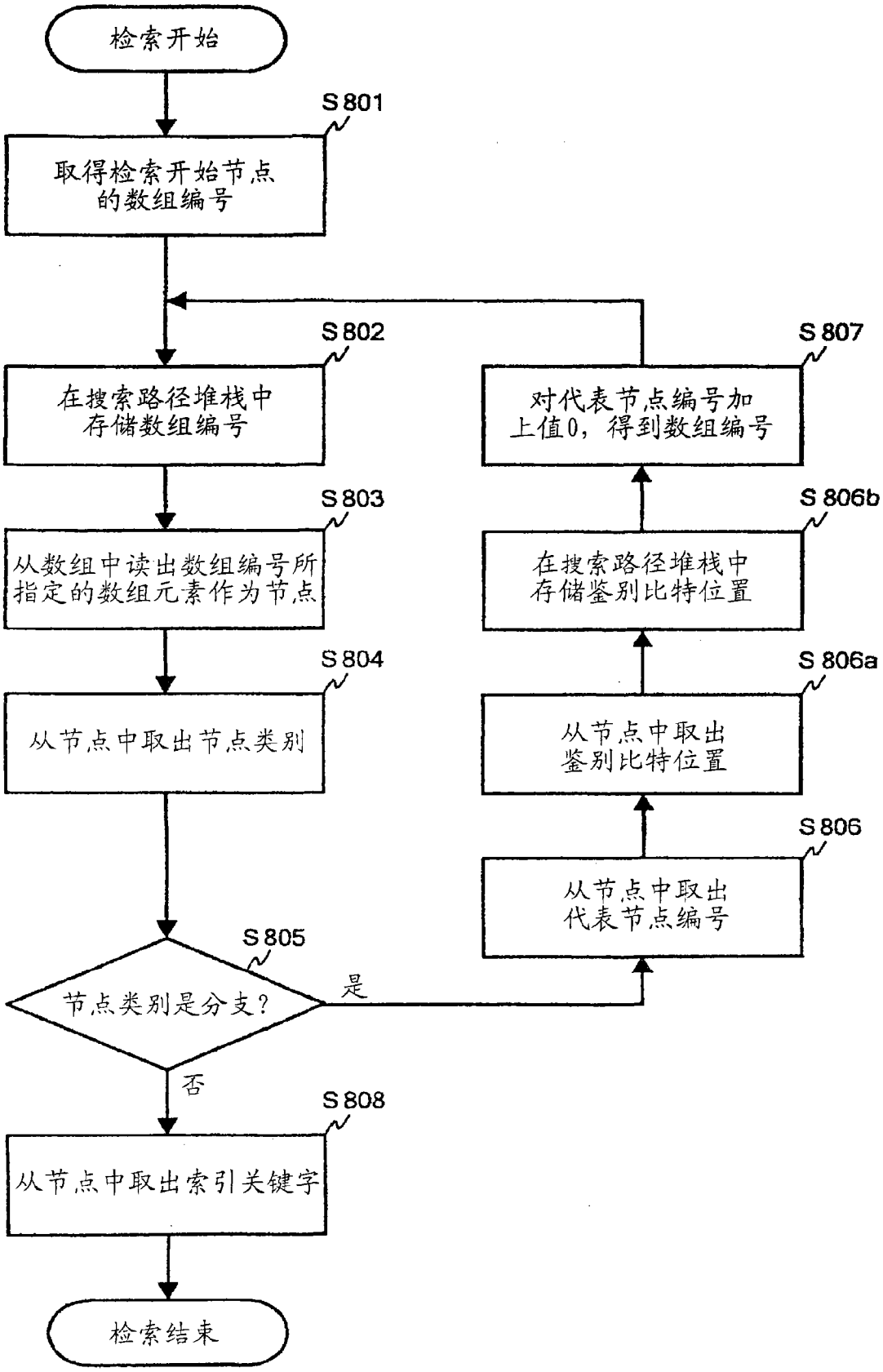

检索开始
S801
取得检索开始节点的数组编号
S802
在搜索路径堆栈中存储数组编号
S803
从数组中读出数组编号所指定的数组元素作为节点
S804
从节点中取出节点类别
S805
节点类别是分支？
是
否
S806
从节点中取出代表节点编号
S806a
从节点中取出鉴别比特位置
S806b
在搜索路径堆栈中存储鉴别比特位置
S807
对代表节点编号加上值0，得到数组编号
S808
从节点中取出索引关键字
检索结束

图 8

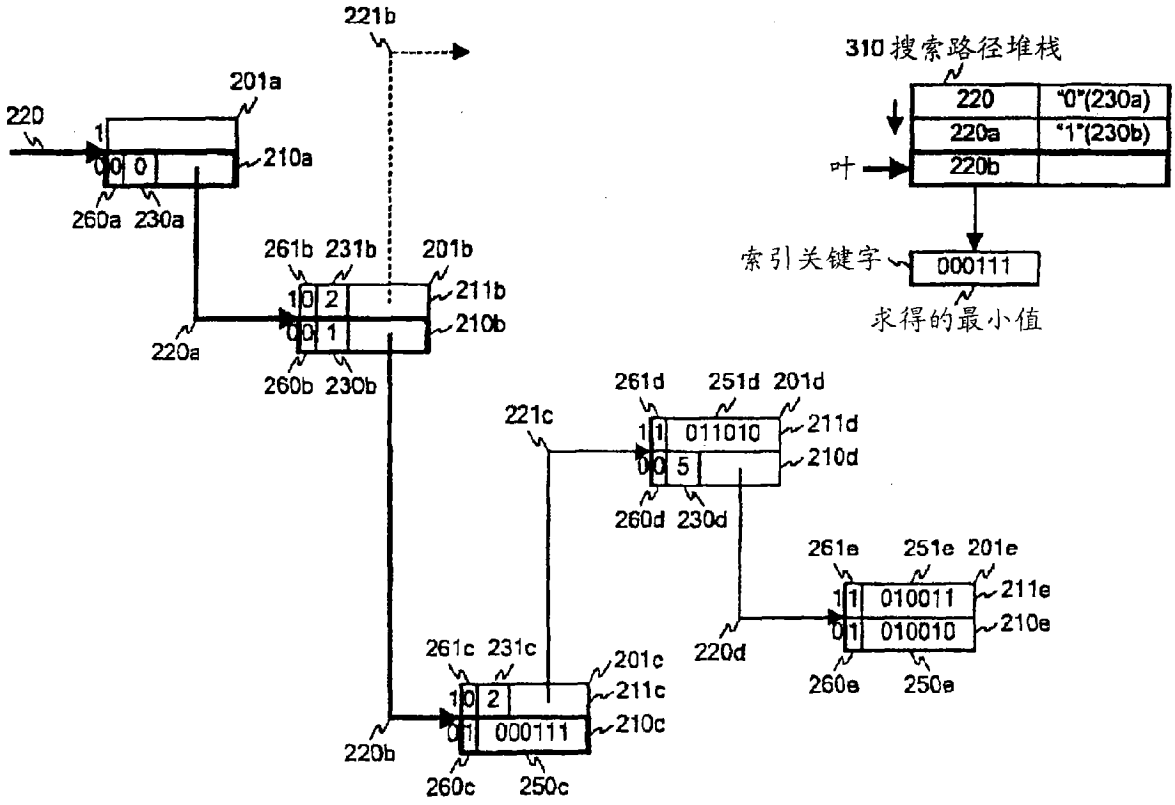

221b
201a
220
210a
260a 230a
261b 231b 201b
211b
210b
220a
260b 230b
310 搜索路径堆栈
220 "0"(230a)
220a "1"(230b)
叶
220b
索引关键字
000111
求得的最小值
261d 251d 201d
221c
011010
211d
210d
260d 230d
261e 251e 201e
010011
211e
010010
210e
220d
260e 250e
261c 231c
201c
211c
210c
220b
000111
260c 250c

图 9

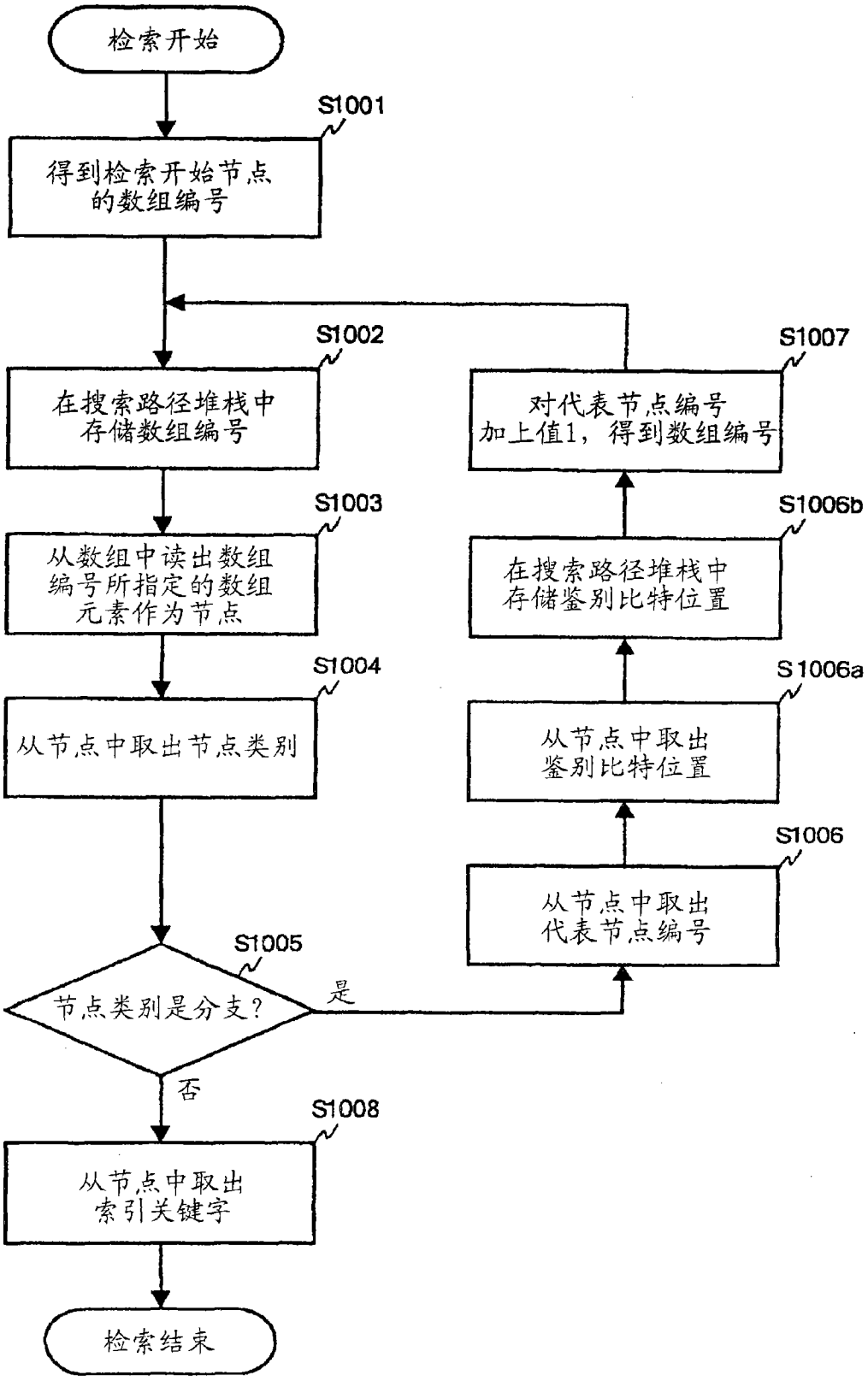

检索开始
S1001
得到检索开始节点的数组编号
S1002
在搜索路径堆栈中存储数组编号
S1003
从数组中读出数组编号所指定的数组元素作为节点
S1004
从节点中取出节点类别
S1005
节点类别是分支?
是
否
S1006
从节点中取出代表节点编号
S1006a
从节点中取出鉴别比特位置
S1006b
在搜索路径堆栈中存储鉴别比特位置
S1007
对代表节点编号加上值1，得到数组编号
S1008
从节点中取出索引关键字
检索结束

图 10

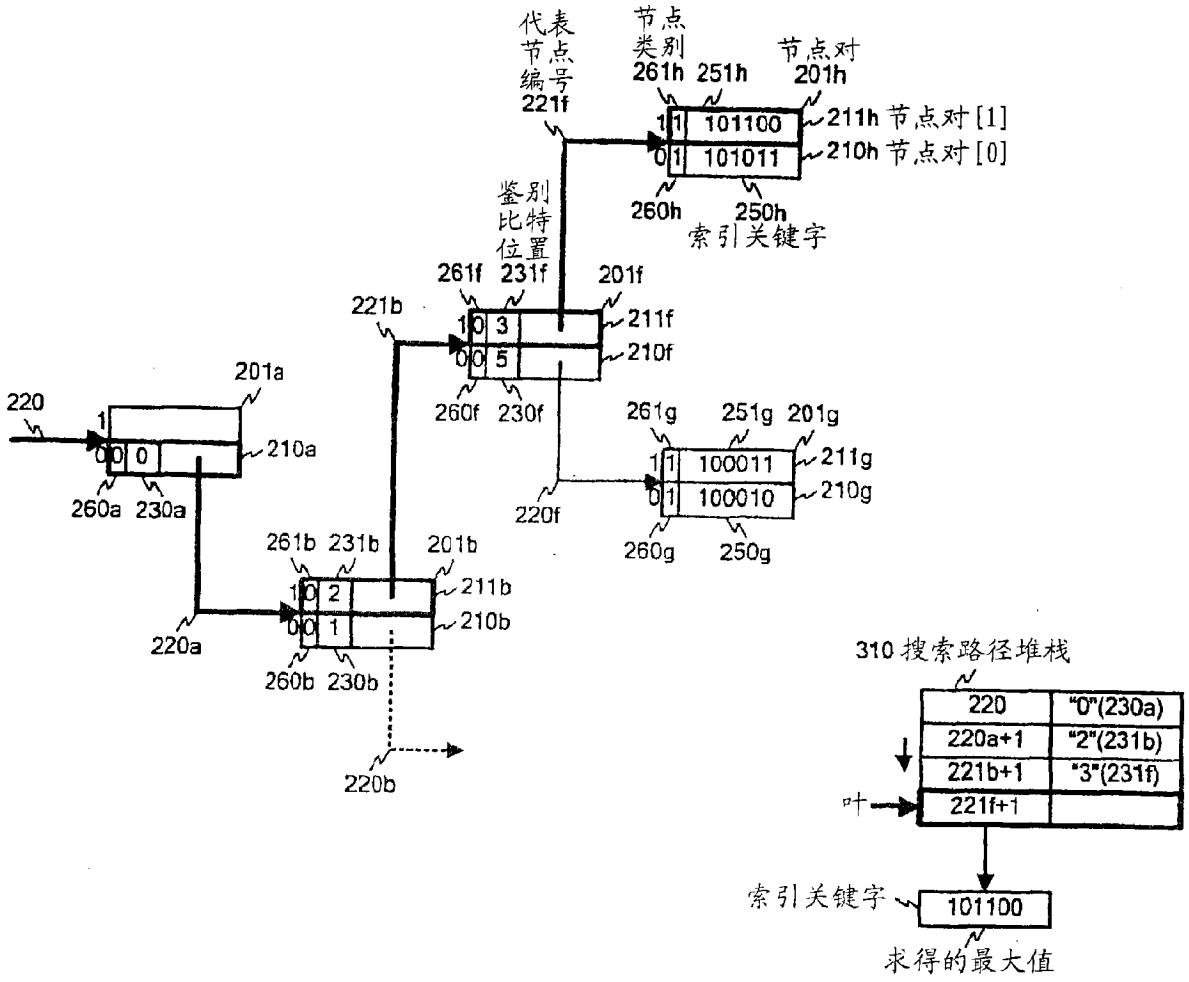

代表节点编号
221f
节点类别
261h 251h
节点对
201h
101100
211h 节点对[1]
101011
210h 节点对[0]
260h
250h
索引关键字
鉴别比特位置
261f 231f
201f
221b
3
211f
5
210f
260f 230f
201a
220
210a
260a 230a
261g 251g 201g
100011
211g
100010
210g
220f
260g 250g
261b 231b 201b
2
211b
1
210b
220a
260b 230b
220b
310 搜索路径堆栈
220 "0"(230a)
220a+1 "2"(231b)
221b+1 "3"(231f)
叶
221f+1
索引关键字
101100
求得的最大值

图 11

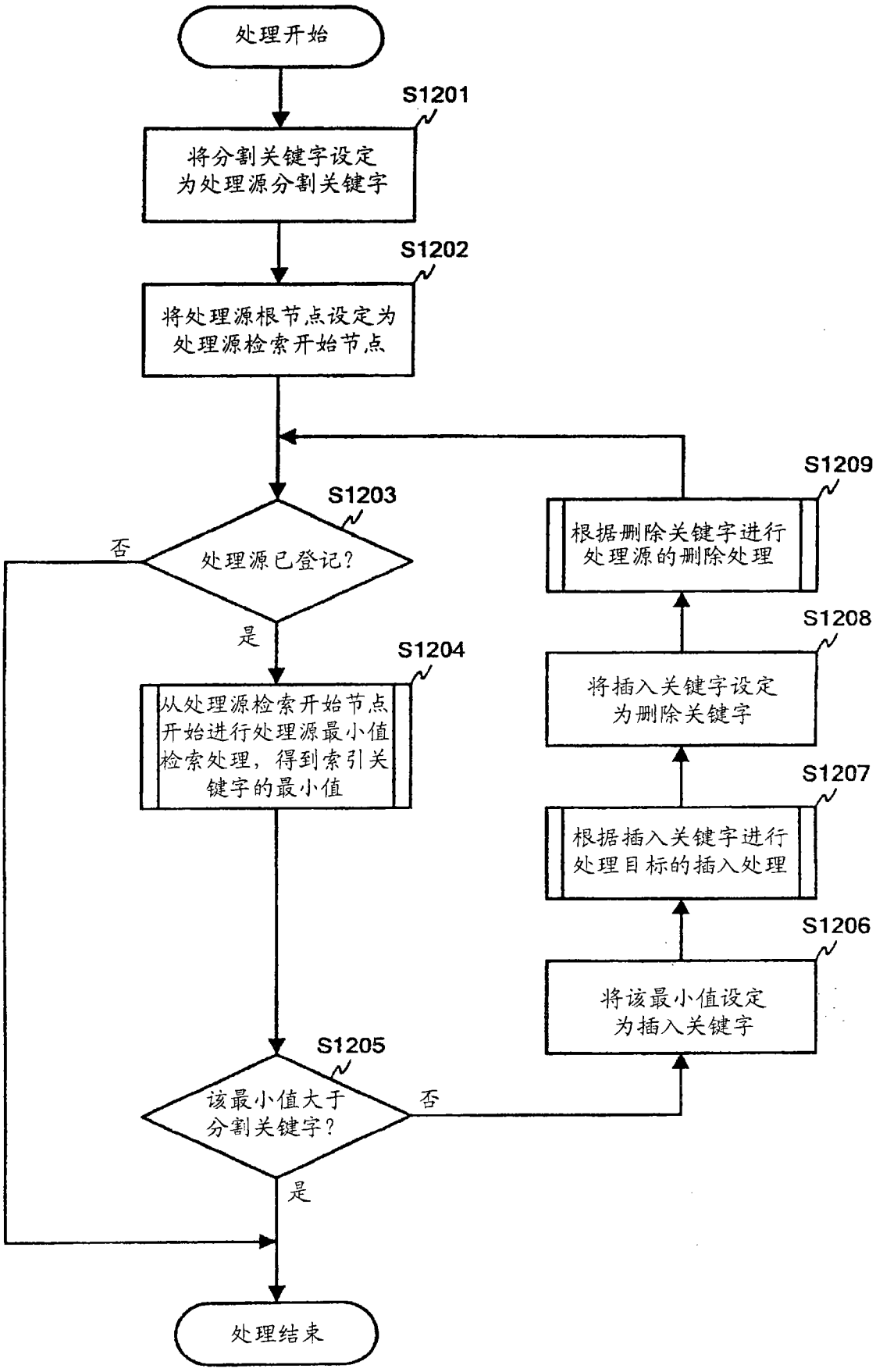

处理开始
S1201
将分割关键字设定为处理源分割关键字
S1202
将处理源根节点设定为处理源检索开始节点
S1203
处理源已登记？
否
是
S1204
从处理源检索开始节点开始进行处理源最小值检索处理，得到索引关键字的最小值
S1205
该最小值大于分割关键字？
否
是
S1206
将该最小值设定为插入关键字
S1207
根据插入关键字进行处理目标的插入处理
S1208
将插入关键字设定为删除关键字
S1209
根据删除关键字进行处理源的删除处理
处理结束

图 12

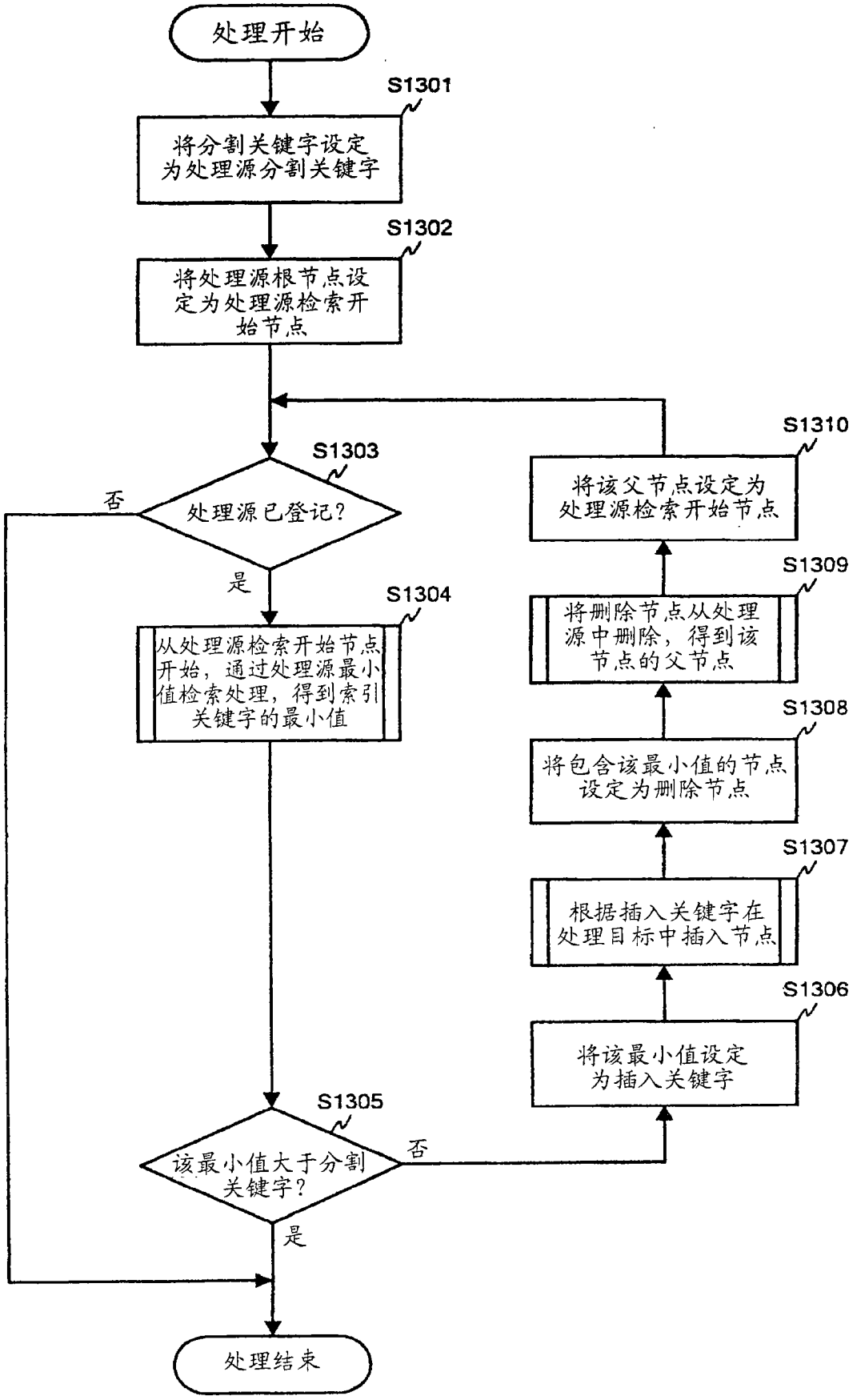

处理开始
S1301
将分割关键字设定为处理源分割关键字
S1302
将处理源根节点设定为处理源检索开始节点
S1303
处理源已登记?
否
是
S1304
从处理源检索开始节点开始，通过处理源最小值检索处理，得到索引关键字的最小值
S1305
该最小值大于分割关键字?
否
是
S1306
将该最小值设定为插入关键字
S1307
根据插入关键字在处理目标中插入节点
S1308
将包含该最小值的节点设定为删除节点
S1309
将删除节点从处理源中删除，得到该节点的父节点
S1310
将该父节点设定为处理源检索开始节点
处理结束

图 13

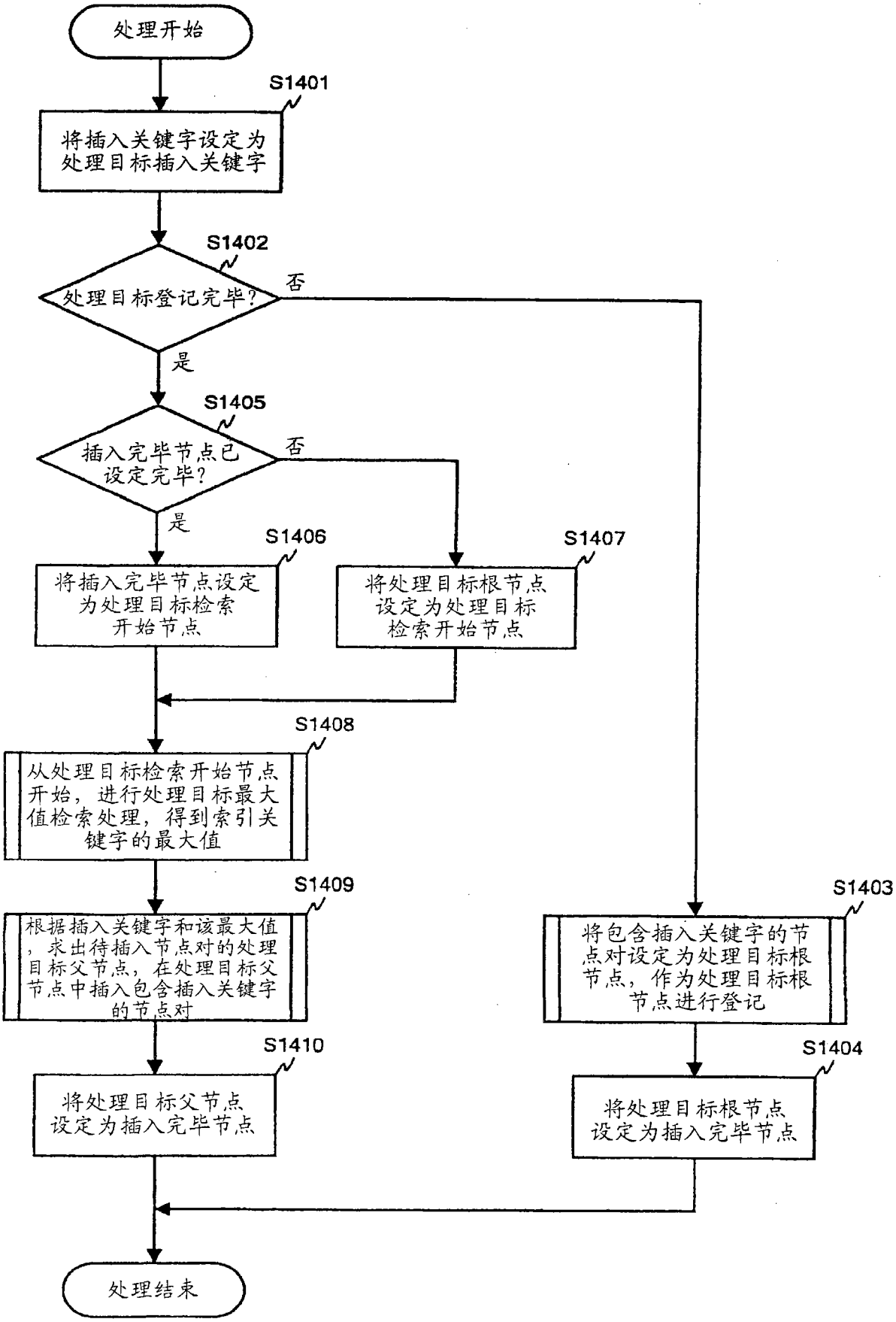

处理开始
S1401
将插入关键字设定为处理目标插入关键字
S1402
处理目标登记完毕?
否
是
S1405
插入完毕节点已设定完毕?
否
是
S1406
将插入完毕节点设定为处理目标检索开始节点
S1407
将处理目标根节点设定为处理目标检索开始节点
S1408
从处理目标检索开始节点开始，进行处理目标最大值检索处理，得到索引关键字的最大值
S1409
根据插入关键字和该最大值，求出待插入节点对的处理目标父节点，在处理目标父节点中插入包含插入关键字的节点对
S1410
将处理目标父节点设定为插入完毕节点
S1403
将包含插入关键字的节点对设定为处理目标根节点，作为处理目标根节点进行登记
S1404
将处理目标根节点设定为插入完毕节点
处理结束

图 14

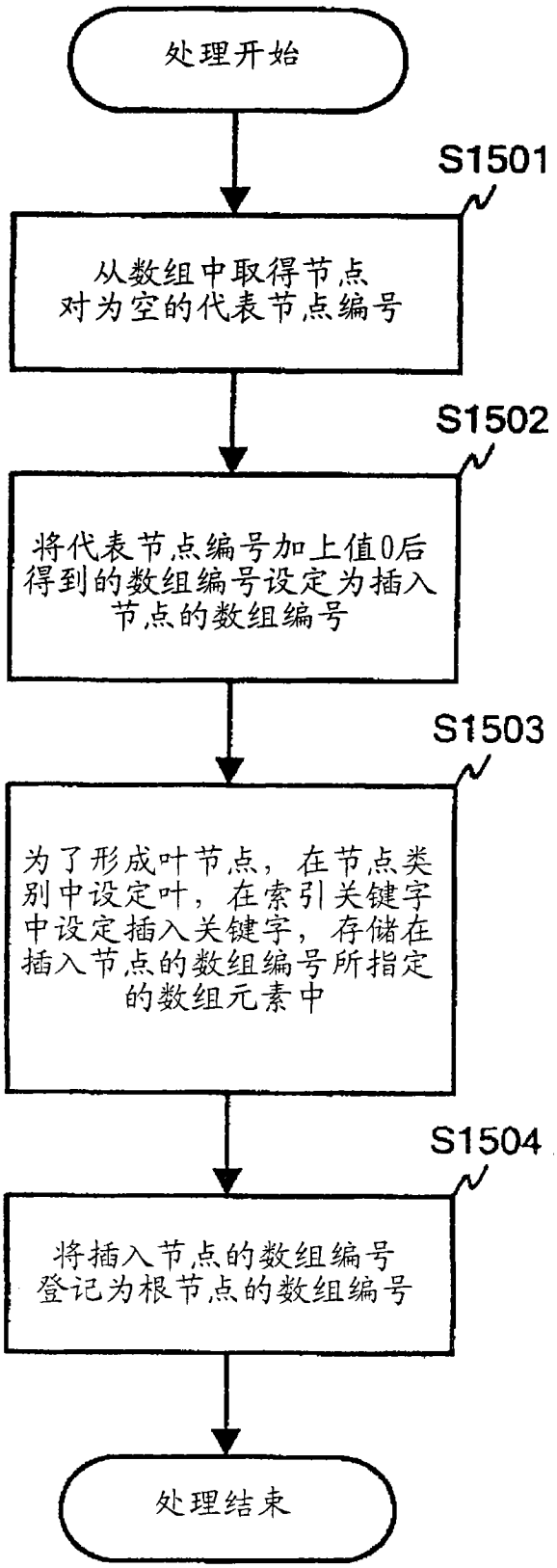

处理开始
S1501
从数组中取得节点对为空的代表节点编号
S1502
将代表节点编号加上值0后得到的数组编号设定为插入节点的数组编号
S1503
为了形成叶节点，在节点类别中设定叶，在索引关键字中设定插入关键字，存储在插入节点的数组编号所指定的数组元素中
S1504
将插入节点的数组编号登记为根节点的数组编号
处理结束

图 15A

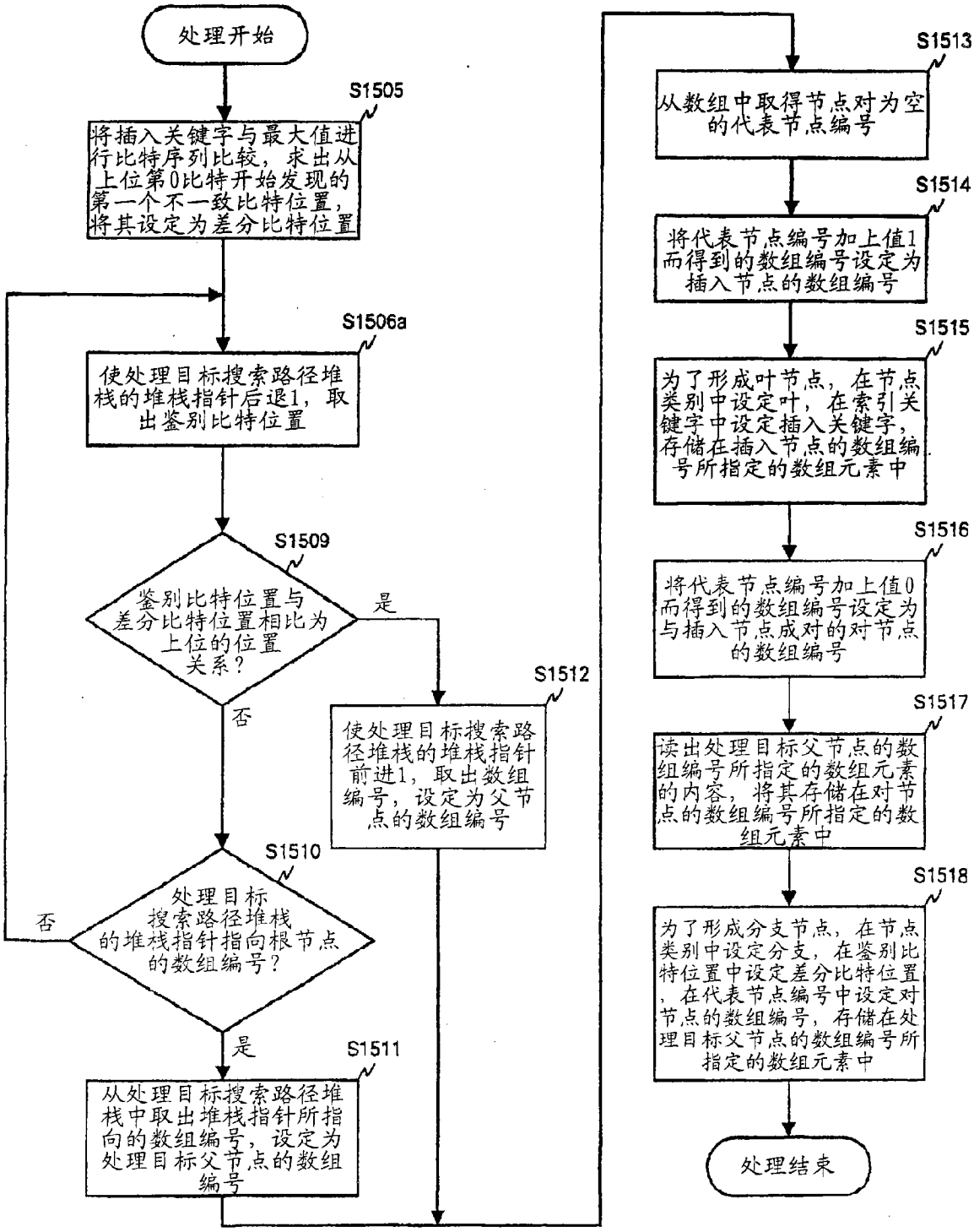

处理开始
S1505
将插入关键字与最大值进行比特序列比较，求出从上位第0比特开始发现的第一个不一致比特位置，将其设定为差分比特位置
S1506a
使处理目标搜索路径堆栈的堆栈指针后退1，取出鉴别比特位置
S1509
鉴别比特位置与差分比特位置相比为上位的位置关系？
是
否
S1512
使处理目标搜索路径堆栈的堆栈指针前进1，取出数组编号，设定为父节点的数组编号
S1510
处理目标搜索路径堆栈的堆栈指针指向根节点的数组编号？
否
是
S1511
从处理目标搜索路径堆栈中取出堆栈指针所指向的数组编号，设定为处理目标父节点的数组编号
S1513
从数组中取得节点对为空的代表节点编号
S1514
将代表节点编号加上值1而得到的数组编号设定为插入节点的数组编号
S1515
为了形成叶节点，在节点类别中设定叶，在索引关键字中设定插入关键字，存储在插入节点的数组编号所指定的数组元素中
S1516
将代表节点编号加上值0而得到的数组编号设定为与插入节点成对的对节点的数组编号
S1517
读出处理目标父节点的数组编号所指定的数组元素的内容，将其存储在对节点的数组编号所指定的数组元素中
S1518
为了形成分支节点，在节点类别中设定分支，在鉴别比特位置中设定差分比特位置，在代表节点编号中设定对节点的数组编号，存储在处理目标父节点的数组编号所指定的数组元素中
处理结束

图 15B

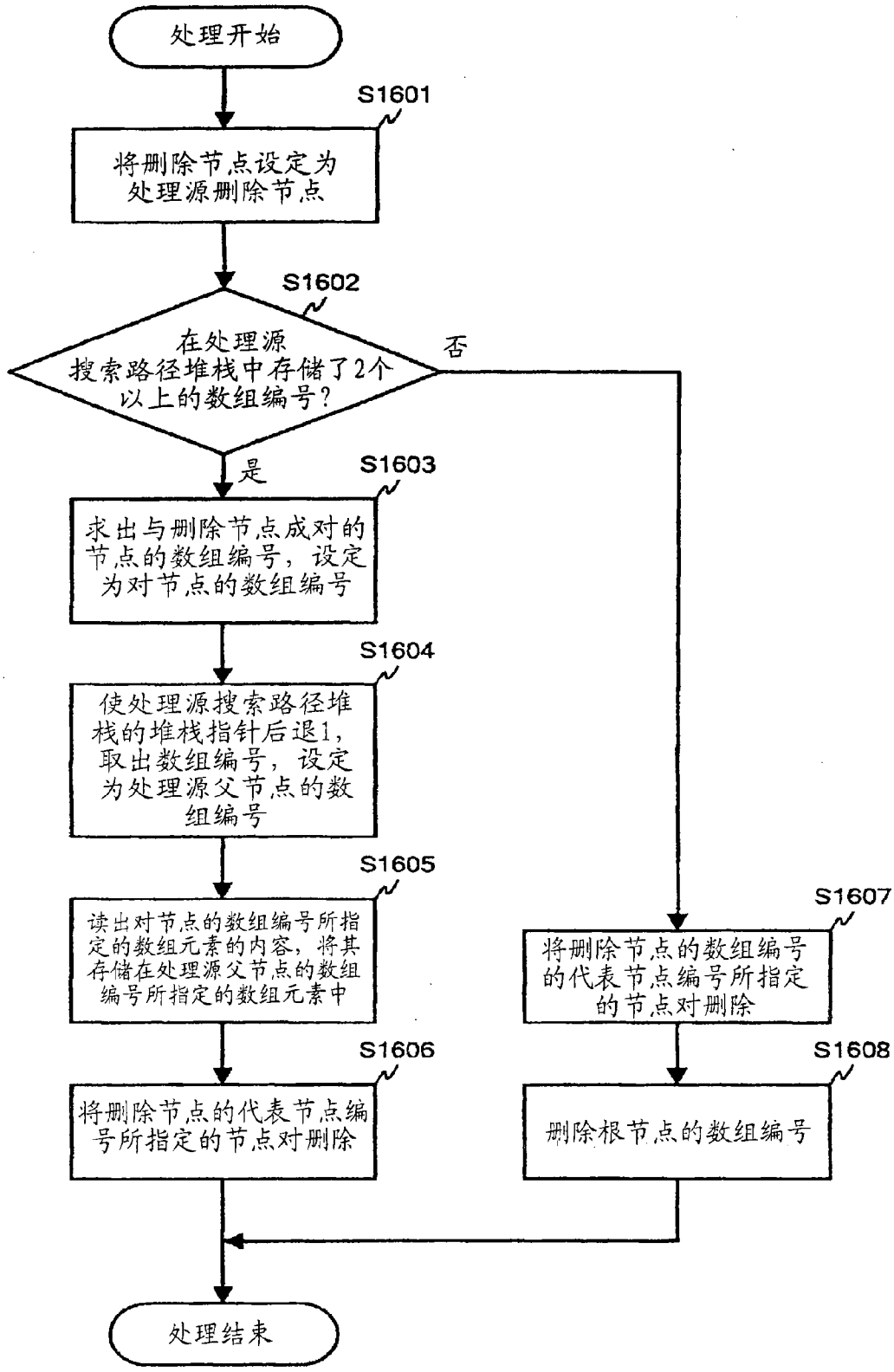

处理开始
S1601
将删除节点设定为处理源删除节点
S1602
在处理源搜索路径堆栈中存储了2个以上的数组编号?
否
是
S1603
求出与删除节点成对的节点的数组编号，设定为对节点的数组编号
S1604
使处理源搜索路径堆栈的堆栈指针后退1，取出数组编号，设定为处理源父节点的数组编号
S1605
读出对节点的数组编号所指定的数组元素的内容，将其存储在处理源父节点的数组编号所指定的数组元素中
S1606
将删除节点的代表节点编号所指定的节点对删除
S1607
将删除节点的数组编号的代表节点编号所指定的节点对删除
S1608
删除根节点的数组编号
处理结束

图 16