200910127

文時黏貼條碼

771069

發明專利說明書

（本申請書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※申請案號：97124937

※申請日期：97 年 07 月 02 日　　※IPC 分類：G06F 17/30

一、發明名稱：

(中) 位元列檢索方法及程式產品

(英)

二、申請人：(共 1 人)

1.姓　名：（中）新叶股份有限公司
（英）S.GRANTS CO., LTD.
代表人：（中）1.新庄容子
（英）1.SHINJO, YOKO
地　址：（中）日本國千葉縣千葉市美濱區高洲三丁目五番三棟一二一〇號
（英）3-5-3-1210, Takasu, Mihama-ku, Chiba-shi, Chiba 261-0004, Japan
國籍：（中英）日本　　JAPAN

三、發明人：(共 2 人)

1.姓　名：（中）新庄敏男
（英）SHINJO, TOSHIO
國　籍：（中）日本
（英）JAPAN

2.姓　名：（中）國分光裕
（英）KOKUBUN, MITSUHIRO
國　籍：（中）日本
（英）JAPAN

四、聲明事項:

◎本案申請前已向下列國家（地區）申請專利□主張國際優先權：
【格式請依：受理國家（地區）；申請日；申請案號數　順序註記】
1.日本　　；2007/07/03；2007-175570 ☑有主張優先權

200910127

文時黏貼條碼

771069

發明專利說明書

（本申請書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※申請案號：97124937

※申請日期：97 年 07 月 02 日　　※IPC 分類：G06F 17/30

一、發明名稱：

(中) 位元列檢索方法及程式產品

(英)

二、申請人：(共 1 人)

1.姓　名：（中）新叶股份有限公司
（英）S.GRANTS CO., LTD.
代表人：（中）1.新庄容子
（英）1.SHINJO, YOKO
地　址：（中）日本國千葉縣千葉市美濱區高洲三丁目五番三棟一二一〇號
（英）3-5-3-1210, Takasu, Mihama-ku, Chiba-shi, Chiba 261-0004, Japan
國籍：（中英）日本　　JAPAN

三、發明人：(共 2 人)

1.姓　名：（中）新庄敏男
（英）SHINJO, TOSHIO
國　籍：（中）日本
（英）JAPAN

2.姓　名：（中）國分光裕
（英）KOKUBUN, MITSUHIRO
國　籍：（中）日本
（英）JAPAN

四、聲明事項:

◎本案申請前已向下列國家（地區）申請專利□主張國際優先權：
【格式請依：受理國家（地區）；申請日；申請案號數　順序註記】
1.日本　　；2007/07/03；2007-175570 ☑有主張優先權

九、發明說明

【發明所屬之技術領域】

本發明，係有關於使用記憶位元列之樹狀的資料構造，而從位元列之集合中來檢索出所期望的位元列之技術者。

【先前技術】

近年，社會之資訊化係在進展，而大規模之資料庫亦成爲被利用在各個領域中。爲了從此種大規模之資料庫而檢索出記錄，通常係將與各記錄所被記憶之位址附加有對應關係的記錄內之項目作爲索引鍵並進行檢索，而找出所期望的記錄。又，在全文檢索中之文字列，亦可看作爲文章的索引鍵。

而，由於此些之索引鍵係作爲位元列而被表現，因此資料庫的檢索係可回歸於位元列之檢索。

爲了高速進行上述位元列之檢索，從先前起，即已進行有對記憶位元列之資料構造所做的各種設計。作爲此些之其中一例，係周知有所謂的帕翠西雅樹（Patricia tree）之樹構造。

圖16，係爲展示被使用於上述之先前的檢索處理中之帕翠西雅樹之其中一例者。帕翠西雅樹之節點，係包含有：索引鍵、檢索鍵之檢查位元位置、左右之連結指標，而被構成。雖並未明示，但是，不用說，在節點中，係被包含有用以對對應於索引鍵的記錄作存取（access）之資訊

。

在圖 16 之例中，保持索引鍵“100010”之節點 1750a 係成爲根節點，該檢查位元位置 1730a 係爲 0。在節點 1750a 之左連結 1740a，係被連接有節點 1750b，在右連結 1741a，係被連接有節點 1750f。

節點 1750b 所保持之索引鍵係爲“010011”，檢查位元位置 1730b 係爲 1。在節點 1750b 之左連結 1740b，係被連接有節點 1750c，在右連結 1741b，係被連接有節點 1750d。節點 1750c 所保持之索引鍵係爲“000111”，檢查位元位置 1730c 係爲 3。節點 1750d 所保持之索引鍵係爲“011010”，檢查位元位置 1730d 係爲 2。

從節點 1750c 而以實線連接之部分，係爲表示節點 1750c 之左右的連結指標者，未被連接有虛線的左指標 1740c，係表示該欄係爲空欄。被連接有虛線之右指標 1741c 的虛線之連接目標，係表示指標所展示的位址，在現在的情況，係表示右指標 1741c 爲指定節點 1750c。

節點 1750d 之右指標 1741d，係指定有節點 1750d 本身，在左連結 1740d，係被連接有節點 1750e。節點 1750e 所保持之索引鍵係爲“010010”，檢查位元位置 1730e 係爲 5。節點 1750e 之左指標 1740e，係指向節點 1750b，右指標 1741e，係指向節點 1750e。

又，節點 1750f 所保持之索引鍵係爲“101011”，檢查位元位置 1730f 係爲 2。在節點 1750f 之左連結 1740f，係被連接有節點 1750g，在右連結 1741f，係被連接有節點

1750h。

節點 1750g 所保持之索引鍵係爲“100011”，檢查位元位置 1730g 係爲 5。節點 1750g 之左指標 1740g，係指向節點 1750a，右指標 1741g，係指向節點 1750g。

節點 1750h 所保持之索引鍵係爲“101100”，檢查位元位置 1730h 係爲 3。節點 1750h 之左指標 1740h，係指向節點 1750f，右指標 1741h，係指向節點 1750h。

於圖 16 之例中，係以隨著從根節點 1750a 而在樹上下降，各節點之檢查位元位置爲變大的方式而被構成。

當以某一檢索鍵而進行檢索時，係從根節點起而依序對被保持於各節點之檢索鍵的檢查位元位置作檢查，並進行關於檢查位元位置之位元值係爲 0 或是 1 的判定，若是爲 1，則到達右連結，若是 0，則到達左連結。而後，若是連結目標之檢查位元位置並未較連結源頭之節點的檢查位元位置大，亦即是，若是連結目標係並非爲朝向下方而係回到上方（於圖 16 中，將以虛線所示之此倒退的連結稱爲退後連結），則進行連結目標之節點的索引鍵與檢索鍵之比較。比較之結果，若是相等，則係保證檢索爲成功，若是不相等，則係保證檢索爲失敗。

如上述所示，在使用有帕翠西雅樹的檢索處理中，雖然具有：僅需對必要之位元作檢查，即可進行檢索；以及鍵全體之比較係只需一次即可等的優點，但是亦會有：由於從各節點必定會分出有 2 個的連結，因此會使記憶容量增大；或是因爲退後連接之存在所造成的判定處理之複雜

化；因爲必須在藉由退後連結而倒退的情況下才會進行與索引鍵間之比較所致的檢索處理之延遲；以及在追加、削除等之資料維護上的困難性等等的缺點。

作爲用以解決此些之帕翠西雅樹的缺點者，例如係有在下述之專利文獻 1 中所揭示的技術。在下述之專利文獻 1 中所記載的帕翠西雅樹中，係藉由將下位之左右的節點記憶在連續之區域中，而削減指標之記憶容量，同時，藉由將用以表示接下來之連結是否爲退後連結的位元設置在各節點中，而減輕退後連結之判定處理的負擔。

然而，在下述專利文獻 1 中所揭示者，亦係由於一個的節點係必定佔據有索引鍵之區域與指標之區域，以及由於將下位之左右的節點記憶在連續之區域中，而將指標設爲 1 個，因此造成就算是在例如圖 1 所示之帕翠西雅樹的最下段之左指標 1740c、右指標 1741h 等的部分，亦需要分配有與節點相同容量之記憶區域等的原因，而使得記憶容量之削減效果並不是很大。又，關於因退後連結所致之檢索處理的延遲問題，或是追加削除等之處理係爲困難的事態，亦並未被改善。

作爲解決上述之先前的檢索手法中之問題點的方法，本申請人，係在日本特願 2006-187827 中，提案有一種使用有耦合節點樹（coupled node tree）之位元列檢索，該耦合節點樹，係爲由根節點（root node）、與被配置於相鄰接之記憶區域中的分支節點（branch node）與葉節點（leaf node）、又或是分支節點彼此又或是葉節點彼此而成

之節點對所成的使用於位元列檢索之樹（tree），根節點係爲表示樹之起點的節點，當該樹之節點係爲 1 個時，係身爲葉節點，而當該樹之節點係爲 2 個以上時，則係爲前述分支節點，前述分支節點，係包含有進行位元列檢索之檢索鍵的辨別位元位置、以及代表身爲連接（link）目標之節點對的其中一方之代表節點的位置之位置資訊，前述葉節點，係包含有由檢索對象之位元列所成的索引鍵（index key）。

在上述申請中，係展示有：從被給予之索引鍵的集合中，來產生耦合節點樹之方法，和從耦合節點樹來檢索出單一的索引鍵之手法等，使用有耦合節點樹之基本的檢索手法。又，關於耦合節點樹之構成係藉由索引鍵之集合而被唯一性規定一事、以及在耦合節點樹上，索引鍵係被排序而配置一事，亦作了說明。

又，在位元列之檢索中，係存在有例如求取最大、最小值，或是求取某範圍內之值等的各種的檢索要求。於此，本申請人，係在日本特願 2006-293619 中，提案了：求取出耦合節點樹之任意的部分木中所包含之索引鍵的最大值／最小值之手法、以及將被儲存在耦合節點樹中之索引鍵以昇順又或是降順來取出的手法等。

進而，亦存在有以下之檢索要求：亦即是，就算是並不存在有與作爲檢索鍵而被賦予之位元列完全一致的索引鍵，若是存在有與檢索鍵部分一致之索引鍵，則也希望能夠將該索引鍵作爲檢索結果而取得。於此，本申請人，係

在日本特願 2007-132289 中，提案有使用有耦合節點樹之最長一致檢索與最短一致檢索之手法。

又，在上述各申請案中，亦揭示有：將耦合節點樹配置於配列中一事、和將在上述所提案之各檢索處理中的從檢索開始節點起之樹上的探索路徑之節點的配列號碼依序堆疊在探索路徑堆疊中，並使用有被堆疊在探索路徑中之配列號碼的處理。

另一方面，依存於使用位元列之處理的內容，亦有希望將一部份之位元並不作爲有意義者來作處理而欲將其作爲隨意位元（don't care bit）來作處理的情況。例如，在路由器之選路表（routing table）的檢索中，在代表 IP（Internet Protocol）位址之位元列中，係期望將網路位址（network address）之部分作爲有意位元來處理，並將主機位址（host address）作爲隨意位元來處理。

在專利文獻 2 中，係針對將使用有派翠西雅樹之最長一致檢索高速化的最長一致檢索電路而有所記載，並圖示有派翠西雅樹之節點與包含有隨意位元之位元列間之對應關係。又，在專利文獻 2 中，亦記載有：該最長一致檢索電路，係適合使用於選路表檢索系統中。

另一方面，不僅是 IP 位址，在各種之分類碼中，亦有對應於位元列之值與位元列所表示之內容的階層性之分類，而以更爲上位之位元來表現更爲上位之分類的情況。在使用有此種位元列之檢索中，係有欲將下位之一部分的位元爲隨意位元的位元列作爲檢索鍵來使用的情況。

此係因爲，係希望能夠知道：相當於作爲檢索鍵而被指定之位元列的分類，是否作爲索引鍵而存在一事，以及當並不存在的情況時，係希望能夠知道包含有相當於作爲檢索鍵而被指定之位元列的上位分類中，其最下位之分類、亦即是與檢索鍵所代表之分類最爲接近的分類，係爲何種分類之故。

如此地，係期望能夠在考慮有隨意位元的同時而進行最長一致檢索。又，在一般的文字列檢索等之中，係亦被要求有：經由在作爲檢索鍵而使用之位元列、以及身爲檢索之對象的索引鍵的雙方中，允許其包含有隨意位元，而實現更具柔軟性之檢索。

另一方面，在用以高速進行最長一致檢索之既存的手法中，係有必要進行繁雜之預處理，且在資料之維護上所耗費之成本係爲大。但是，滿足以下所有之條件、亦即是能夠進行考慮有隨意位元之最長一致檢索、且檢索係爲高速、且在維護上所耗費之成本係爲低的手法，係並未被發現。

例如，在專利文獻 3 中所記載之 IP 位址檢索表作成方法中，係當第 1 登錄（entry）之 IP 位址的範圍爲包含有第 2 登錄之 IP 位址的範圍時，將第 1 登錄分解爲被包含在第 1 登錄內之第 3 與第 4 登錄，並將第 1 登錄消去。而後，將第 3 與第 4 登錄分別與第 2 登錄作比較，並將一致之分解登錄消去。

分解，係一面藉由在子網遮罩（subnet mask）處將值

成爲“1”的範圍一次增加 1 位元，一面反覆進行此，直到分解登錄與第 2 登錄成爲一致爲止。藉由分解，由於 2 個的登錄間之包含關係係被解除，因此，在硬體所致之高速檢索裝置中，係成爲能夠恆常地正確實行最長一致檢索。但是，在每次之登錄的追加或削除中，由於包含關係會變動，因此，在資料之維護上所耗費之成本係爲高。

又，在專利文獻 4 中，係記載有：將在自然言語處理中之單字字典等的最長一致檢索高速化之方法。在此方法中，首先係將記錄（record）群以鍵項目（被登錄於字典中之單字）來作昇順排序。而後，對於各記錄，在較該當記錄爲更前方之記錄中，將具備有與該當記錄之鍵項目成爲最長一致的鍵項目之記錄的號碼，作爲「下一指標」而設定。

若是使用此種被設定有下一指標之字典，則藉由使用檢索關鍵字來進行二進碼檢索，並將該檢索結果之記錄中的鍵項目與檢索關鍵字作比較，若是並不一致，則對下一指標依序作遍歷，而能夠進行最長一致檢索。藉由此，係成爲在能夠活用二進碼檢索之高速性的狀態下，來進行最長一致檢索。但是，在每次之記錄的追加或是削除中，係有必要重新設定記錄號碼或是下一指標。

〔專利文獻 1〕日本特開 2001-357070 號公報

〔專利文獻 2〕日本特開 2003-224581 號公報

〔專利文獻 3〕日本特開平 11-103318 號公報

〔專利文獻 4〕日本特開平 10-177582 號公報

【發明內容】

〔發明所欲解決之課題〕

於此，本發明之目的，係爲活用耦合節點樹之長處、亦即是在能夠進行高速檢索的同時，在資料之維護上所耗費之成本亦爲低的優點，並在隨意位元係並不存在於較有意位元爲更上位之位置處的前提下，使用包含有隨意位元之位元列，而使對於包含有對包括隨意位元之位元列作表現的索引鍵之耦合節點樹作檢索一事成爲可能。

〔用以解決課題之手段〕

若藉由本發明，則係提供一種具備有以下之資料構造的耦合節點樹，並進行使用有該耦合節點樹之索引鍵的檢索。又，若藉由本發明，則係提供一種對於該耦合節點樹，而進行將對應於所指定之位元列的葉節點作插入或削除之方法。

本發明之耦合節點樹，係由根節點，和由被配置於鄰接之記憶區域的分支節點與葉節點、又或是分支節點彼此又或是葉節點彼此而成的節點對所構成。

前述根節點，係爲表示樹之起點的節點，當該樹之節點係爲 1 個時，係身爲前述葉節點，而當該樹之節點係爲 2 個以上時，則係爲前述分支節點。

前述分支節點，係包含有用以進行位元列檢索之檢索鍵的辨別位元位置、和代表身爲連結目標之節點對的其中

一方之節點的代表節點之位置的位置資訊。

又，前述索引鍵，係由將前方有意位元列作編碼後之編碼位元列所成，該前方有意位元列，係爲僅由有意位元所成之位元列、或是僅由任意位元值之隨意位元（don't care bit）所成之位元列、又或是在 1 位元以上之有意位元處連續有 1 位元以上之隨意位元的位元列中之任一者。

前述檢索鍵，係由將前方有意位元列編碼後之編碼位元列所成。

前述編碼位元列，係爲對應於構成前述前方有意位元列之各位元的位元對，該位元對，係由表現該位元係爲隨意位元亦或是有意位元一事之識別位元、和當該位元係爲隨意位元時則成爲預先被決定之值；當該位元係爲有意位元時則爲代表該位元之值的資料位元所成。

亦有將編碼前之前方有意位元列稱爲元位元列、並將對應於身爲編碼位元列之索引鍵或是檢索鍵的被編碼前之前方有意位元列，分別稱爲元索引鍵與元檢索鍵之情況。

此種耦合節點樹之檢索，係如下述一般地進行。

首先，在初期檢索步驟中，係將前述耦合節點樹之任意的部分樹之根節點作爲檢索開始節點，在前述分支節點中，因應於被包含在該分支節點中之辨別位元位置之前述檢索鍵的位元值，而對連結目標之節點對的代表節點或者是與該代表節點成對之非代表節點作連結，並藉由在前述編碼位元列中之識別位元所存在的位置處至少儲存前述辨別位元位置所相當之分支節點的位址資訊，而一面對作了

連結並遍歷（traverse）了的路徑作記憶，一面依序反覆進行前述連結，直到到達前述葉節點爲止。

而後，在索引鍵取得步驟中，從藉由前述初期檢索步驟所到達的前述葉節點中，取得索引鍵。

接下來，在差分位元位置取得步驟中，係在前述所取得之前述索引鍵與前述檢索鍵之間，對於從位元列之前端起，直到在前述所取得之前述索引鍵處的將末尾端之有意位元編碼後之位元對的位置、和在前述檢索鍵中，將前端之隨意位元作了編碼後之位元對的位置的兩者中，較接近於前述編碼位元列之前端的位元位置者爲止的範圍內，進行位元列比較，並將成爲相異之位元值的前端之位元位置，作爲差分位元位置而取得。

而後，在第 1 最長一致鍵取得步驟中，若是前述所取得之前述索引鍵與前述檢索鍵，在前述範圍中係爲一致，則將前述所取得之前述索引鍵，作爲最長一致鍵而取得。

於前述比較之結果，若是前述所取得之前述索引鍵與前述檢索鍵並非爲一致，則藉由分支節點選擇步驟，來將辨別位元位置爲相當於在前述編碼位元列中的識別位元所存在之位置，且該辨別位元位置爲相較於前述差分位元位置而更接近於前述編碼位元列之前端的位元位置之前述路徑上之分支節點中，前述辨別位元位置爲最爲接近前述編碼位元列之尾端的位元位置之分支節點，作爲分支節點而選擇。而後，在第 2 最長一致鍵取得步驟中，係將從前述所選擇之前述分支節點而被連結之節點對中，身爲對應於

代表前述隨意位元之前述識別位元之值而被連結的葉節點之終端側節點中所包含的索引鍵，作爲最長一致鍵而取得。

另外，前述葉節點之前述索引鍵，係亦可代替前述編碼位元列，而爲由編碼前之前述前方有意位元列所成。

於此情況，在前述差分位元位置取得步驟以及前述第1最長一致鍵取得步驟中，代替前述所取得之前述索引鍵，係將把前述所取得之索引鍵編碼後的編碼位元列與前述檢索鍵作比較。

又，爲了將包含有由身爲編碼位元列之以「編碼位元列」又或是「被編碼前之前方有意位元列」的形式而被指定的所期望之位元列所成的索引鍵之葉節點，插入至前述耦合節點樹中，係進行以下之處理。

首先，當前述所期望之位元列係經由前述編碼位元列而被指定的情況時，則係將所指定之該編碼位元列作爲檢索鍵；當前述所期望之位元列係經由前述前方有意位元列而被指定的情況時，則係取得將所指定之前述前方有意位元列編碼後的編碼位元列，並將所取得之該編碼位元列作爲檢索鍵。

而後，在前述分支節點中，因應於被包含在該分支節點中之辨別位元位置之前述檢索鍵的位元值，而對連結目標之節點對的代表節點或者是與該代表節點成對之非代表節點作連結，並從前述根節點起而一面將路徑作記憶一面依序反覆進行前述連結，直到到達前述葉節點爲止。

接下來，在前述葉節點之索引鍵與與前述檢索鍵之間，進行大小比較與位元列比較。

進而，藉由身爲在前述位元列比較中成爲相異之位元值的前端之位元位置的差分位元位置、和前述經過路徑上之分支節點的辨別位元位置，其兩者間之相對性的位置關係，來決定由所插入之前述葉節點與另外一方之節點所成的節點對之插入位置。

而後，藉由前述大小關係，來決定將包含有由前述所期望之位元列所成的索引鍵之前述葉節點，作爲前述節點對之何者的節點來插入。

另外，前述葉節點之前述索引鍵，係亦可代替前述編碼位元列，而爲由前述前方有意位元列所成。

於此情況，代替在前述大小比較與前述位元列比較之前便將前述索引鍵編碼並取得編碼位元列，再在前述葉節點之索引鍵與前述檢索鍵之間進行前述大小比較與前述位元列比較，而改爲在所取得之前述編碼位元列與前述檢索鍵之間，進行前述大小比較與前述位元列比較。

又，爲了將包含有由身爲編碼位元列之以「編碼位元列」又或是「被編碼前之前方有意位元列」的形式而被指定的所期望之位元列所成的索引鍵之葉節點，從前述耦合節點樹中作削除，係進行以下之處理。

首先，當前述所期望之位元列係經由前述編碼位元列而被指定的情況時，則係將所指定之該編碼位元列作爲檢索鍵；當前述所期望之位元列係經由前述前方有意位元列

而被指定的情況時，則係取得將所指定之前述前方有意位元列編碼後的編碼位元列，並將所取得之該編碼位元列作爲檢索鍵。

而後，在前述分支節點中，因應於被包含在該分支節點中之辨別位元位置之前述檢索鍵的位元值，而對連結目標之節點對的代表節點或者是與該代表節點成對之非代表節點作連結，並從前述根節點起而依序反覆進行前述連結，直到到達前述葉節點爲止。

接下來，將與前述葉節點成爲節點對之節點的內容，儲存在該當節點對之連結源頭的分支節點中。而後，將該當節點對削除。

又，若藉由本發明，則係亦提供一種程式，其係使電腦實行上述之檢索方法、插入方法以及削除方法。

〔發明之效果〕

隨意位元之值，係無法被唯一性的決定，而根據值係無法唯一性地被決定之位元，係無法在分支節點處而唯一性地決定連結目標。於此，在本發明中，係如同上述一般，而利用有被編碼後之檢索鍵。

若藉由本發明，則在被編碼前之前方有意位元列處的隨意位元，於身爲編碼位元列之檢索鍵處，係亦被編碼爲具備有唯一值之位元。又，被包含於葉節點中之索引鍵，無論其係爲編碼位元列或是被編碼前之前方有意位元列，在本發明中，分支節點之辨別位元位置，均係代表在編碼

後的狀態下之位元的位置。

故而，若是藉由本發明，則就算是成爲檢索鍵之根源的編碼前之前方有意位元列，係包含有隨意位元，於分支節點中，連結目標亦係被唯一性地決定，而能夠進行檢索。

又，經由本發明之檢索方法而作爲最長一致鍵所取得之索引鍵，係代表一種位元列之集合，不論成爲檢索鍵之根源的編碼前之前方有意位元列的隨意位元具體上是成爲何種值，在隨意位元處被設定有值之該位元列，均係被包含在該該位元列之集合中。

故而，本發明之耦合節點樹，係適合於使用有包含隨意位元之位元列的檢索中。

又，藉由上述之插入方法，係可產生本發明之耦合節點樹，藉由上述之插入方法與削除方法，係可對本發明之耦合節點樹作維護。

進而，本發明所致之耦合節點樹，係由節點對所構成，分支節點係包含有辨別位元位置與位置資訊，而葉節點係包含有索引鍵，此點，係與先前所申請之耦合節點樹的構成爲類似。與先前之申請相類似之此些的構成，係亦爲成爲耦合節點樹之特徵、亦即是可進行高速之檢索，且可以低成本來進行插入或削除等之特徵的要因之構成。

故而，若藉由本發明，則係提供有一種耦合節點樹，其係在繼承有檢索係爲高速，且在節點之插入或削除上所耗費之計算成本係爲低之優點的同時，亦適合於使用有包

含隨意位元之位元列的檢索中。

【實施方式】

首先，針對由本申請人在先前的上述申請案中所提案之成爲本發明的前提之耦合節點樹，對將耦合節點樹儲存於配列中之例作說明。作爲表示分支節點所保持之連結目標的位置之資料，亦可使用記憶裝置之位址資訊，但是，藉由使用由可將分支節點或是葉節點之中所佔有的區域之記憶容量較大者作儲存之配列要素所成的配列，能將節點之位置以配列號碼來表示，而能削減位置資訊之資訊量。

圖 1，係爲說明被儲存於配列中之耦合節點樹的構成例之圖。

若參考圖 1，則節點 101 係被配置在配列 100 之配列號碼 10 的配列要素中。節點 101，係由節點種類別 102、辨別位元位置 103 以及代表節點號碼 104 所構成。節點種類別 102 係爲 0，並表示節點 101 係爲分支節點。在辨別位元位置 103 中，係被儲存有 1。在代表節點號碼 104 中，係被記憶有連結目標之節點對的代表節點之配列號碼 20。另外，於以下，爲了表記之簡略化，亦有將被儲存於代表節點號碼中之配列號碼稱爲代表節點號碼的情形。又，亦有將被儲存於代表節點號碼中之配列號碼，以附加於該節點之符號或者是附加於節點對之符號來作代表的情況。

在配列號碼 20 之配列要素中，係被儲存有身爲節點對 111 之代表節點的節點〔0〕112。而，在相鄰接之下一

個配列要素（配列號碼 20＋1）中，係被儲存有與代表節點成爲一對之節點〔1〕113。在節點〔0〕112之節點種類別 114 中係被儲存有 0；在辨別位元位置 115 中係被儲存有 3；在代表節點號碼 116 中係被儲存有 30。又，在節點〔1〕113 之節點種類別 117 中係被儲存有 1，而表示節點〔1〕113 係爲葉節點。在索引鍵 118 中，係被儲存有“0001”。當然，與針對帕翠西雅樹而在先前所述相同地，於葉節點中係包含有對與索引鍵相對應之記錄進行存取的資訊，但是，於表記中係省略。

另外，將代表節點以節點〔0〕來表示，將與其成對之節點以節點〔1〕來表示。又，亦有將被儲存於某配列號碼之配列要素中的節點，稱爲該配列號碼之節點的情況，而也有將被儲存有節點之配列要素的配列號碼，稱爲節點之配列號碼的情況。

由被儲存於配列號碼 30 以及 31 之配列要素中的節點 122 與節點 123 所成之節點對 121 的內容，係被省略。

在被儲存有節點〔0〕112、節點〔1〕113、節點 122、以及節點 123 的配列要素中分別被附加的 0 或是 1，係爲在以檢索鍵來進行檢索的情況時，用以表示對節點對之何一節點作連結。將在前段之分支節點的辨別位元位置中之檢索鍵的位元值 0 或是 1，加到代表節點號碼上，並對此配列號碼之節點作連結。

故而，藉由在前段之分支節點的代表節點號碼上，再加上檢索鍵之辨別位元位置的位元值，而能求取出連結目

標之節點所被儲存之配列要素的配列號碼。

另外，在上述之例中，係採用節點對之被配置的配列號碼中之較小的一方作爲代表節點號碼，但是，不用說，亦可採用較大的一方。

圖 2，係爲將耦合節點樹之樹狀構造作概念展示的圖。圖示之 6 位元的索引鍵，係與圖 16 所例示之帕翠西雅樹者相同。

符號 210a 所示，係爲根節點。於圖示之例中，根節點 210a 係作爲被配置於配列號碼 220 中之節點對 201a 的代表節點。

作爲樹狀構造，於根節點 210a 之下方係配置有節點對 201b，而於其下層係配置有節點對 201c 與節點對 201f，在節點對 201f 之下層，係被配置有節點對 201h 與節點對 201g。在節點對 201c 之下方係被配置有節點對 201d，並更進而在其下方被配置有節點對 201e。

被附加在各節點之前的 0 或是 1 的符號，係與於圖 1 中所說明之被附加在配列要素之前的符號相同。因應於檢索鍵之辨別位元位置的位元值而遍歷樹，並找出檢索對象之葉節點。

在圖示之例中，根節點 210a 之節點種類別 260a 係爲 0，表示其係爲分支節點，而辨別位元位置 230a 係顯示 0。代表節點號碼係爲 220a，此係爲儲存有節點對 201b 之代表節點 210b 的配列要素之配列號碼。

節點對 201b 係以節點 210b 與 211b 所構成，此些之

節點種類別 260b、261b 係均爲 0，而表示其係爲分支節點。在節點 210b 之辨別位元位置 230b，係被儲存有 1，在連結目標之代表節點號碼中，係被儲存有節點對 201c 之代表節點 210c 所被儲存之配列要素的配列號碼 220b。

在節點 210c 之節點種類別 260c 中，由於係被儲存有 1，因此此節點係爲葉節點，故而，係包含有索引鍵 250c。在索引鍵 250c 中，係被儲存有“000111”。另一方面，節點 211c 之節點種類別 261c 係爲 0，辨別位元位置 231c 係爲 2，在代表節點號碼中，係被儲存有節點對 201d 之代表節點 210d 所被儲存之配列要素的配列號碼 221c。

節點 210d 之節點種類別 260d 係爲 0，辨別位元位置 230d 係爲 5，在代表節點號碼中，係被儲存有節點對 201e 之代表節點 210e 所被儲存之配列要素的配列號碼 220d。與節點 210d 成對之節點 211d 的節點種類別 261d 係爲 1，在索引鍵 251d 中，係被儲存有“011010”。

節點對 201e 之節點 210e、211e 的節點種類別 260e、261e 係均爲 1，表示其雙方均爲葉節點，而在各別的索引鍵 250e、251e 中，作爲索引鍵，係被儲存有“010010”和“010011”。

在身爲節點 201b 之另外一方的節點 211b 之辨別位元位置 231b，係被儲存有 2，在連結目標之代表節點號碼中，係被儲存有節點對 201f 之代表節點 210f 所被儲存之配列要素的配列號碼 221b。

節點對 201f 之節點 210f、211f 的節點種類別 260f、

261f 係均爲 0，表示其雙方均爲分支節點。在各別之辨別位元位置 230f、231f 中，係被儲存有 5、3。在節點 210f 之代表節點號碼中，係被儲存有節點對 201g 之代表節點 210g 所被儲存之配列要素的配列號碼 220f，在節點 211f 之代表節點號碼中，係被儲存有身爲節點對 201h 之代表節點的節點〔0〕211h 所被儲存之配列要素的配列號碼 221f。

節點對 201g 之節點 210g、211g 的節點種類別 260g、261g 係均爲 1，表示其雙方均爲葉節點，而在各別的索引鍵 250g、251g 中，係被儲存有“100010”和“100011”。

又，同樣的，身爲節點對 201h 之代表節點的節點〔0〕210h、與和其成對之節點〔1〕211h 的節點種類別 260h、261h 係均爲 1，表示其雙方均爲葉節點，而在各別的索引鍵 250h、251h 中，係被儲存有“101011”和“101100”。

以下，針對從上述之樹中對索引鍵“100010”作檢索的處理之流程作簡單說明。辨別位元位置，係設爲從左起而爲 0、1、2、...。

首先，將位元列“100010”作爲檢索鍵，而從根節點 210a 起開始進行處理。根節點 210a 之辨別位元位置 230a 由於係爲 0，因此若是檢查檢索鍵“100010”之辨別位元位置 0 的位元值，則發現其係爲 1。於此，對將儲存有代表節點號碼之配列號碼 220a 中加上 1 之後的配列號碼之配列要素中所儲存的位元 211b 作連結。在節點 211b 之辨別位元位置 231b 中，由於係被儲存有 2，因此，若是檢查檢

索鍵“100010”之辨別位元位置 2 處的位元值，則發現其係爲 0，故而，對儲存有代表節點號碼之配列號碼 221b 的配列要素中所儲存之節點 210f 作連結。

在節點 210f 之辨別位元位置 230f 中，由於係被儲存有 5，因此，若是檢查檢索鍵“100010”之辨別位元位置 5 的位元值，則發現其係爲 0，故而，對代表節點號碼之被儲存的配列號碼 220f 之配列要素中所儲存的節點 210g 作連結。

由於節點 210g 之節點種類別 260g 係爲 1，而代表其係爲葉節點，因此，將索引鍵 250g 讀出，並與檢索鍵作比較，其結果，兩者係均爲“100010”，而爲一致。如此地，進行使用有耦合節點樹之檢索。

接下來，參考圖 2，並針對耦合節點樹之構成的意義作說明。

耦合節點樹之構成，係藉由索引鍵之集合而被規定。在圖 2 之例中，根節點 210a 之辨別位元位置爲 0 的原因，係因爲在圖 2 中所例示之索引鍵中，具備有第 0 位元爲 0 者以及爲 1 者之故。第 0 位元爲 0 之索引鍵的群組，係被分類至節點 210b 之下，而第 0 位元爲 1 之索引鍵的群組，係被分類至節點 211b 之下。

節點 211b 之辨別位元位置爲 2 的原因，係因爲其係反映有：在節點 211h、210h、211g、210g 中所儲存之第 0 位元爲 1 的索引鍵，其第 1 位元係全部等於 0，而爲在第 2 位元才開始有所不同者一般之索引鍵的集合之性質之故

。

以下，與第 0 位元的情況相同地，第 2 位元爲 1 者，係被分類至節點 211f 側，而第 2 位元爲 0 者，則被分類至節點 210f 側。

而，第 2 位元爲 1 的索引鍵，由於係爲從第 3 位元而開始有所不同者，因此在節點 211f 之辨別位元位置中，係被儲存有 3，而在第 2 位元爲 0 的索引鍵中，由於其第 3 位元與第 4 位元均爲相等，而從第 5 位元才開始有所不同，因此，節點 210f 之辨別位元位置中，係被儲存有 5。

在節點 211f 之連結目標中，由於第 3 位元爲 1 者與爲 0 者係分別各只有 1 個，因此節點 210h、211h 係成爲葉節點，並分別在索引鍵 250h 與 251h 中被儲存有‘101011’與‘101100’。

就算假設在索引鍵之集合中，替代“101100”而被包含有“101101”或是“101110”，亦由於其之到第 3 位元爲止係與“101100”相等，因此僅會使被儲存在節點 211h 中之索引鍵改變，而不會使樹狀構造之本身改變。但是，若是除了“101100”之外，再加上亦包含有“101101”，則節點 211h 係成爲分支節點，而其辨別位元位置係成爲 5。若是所追加之索引鍵係爲“101110”，則辨別位元位置係成爲 4。

如以上所說明一般，耦合節點樹之構造，係藉由被包含於索引鍵之集合中的各索引鍵之各位元位置的位元值而被決定。

而，若更進一步作說明，則由於在每一成爲相異位元

值之位元位置處，均分歧有位元值爲“1”之節點與位元值爲“0”之節點，因此，若是以節點〔1〕側與樹之深度方向爲優先而探索葉節點，而於該些中所儲存之索引鍵，係成爲節點 211h 之索引鍵 251h 的“101100”、節點 210h 之索引鍵 250h 的“101011”、¡K、節點 210c 之索引鍵 250c 的“000111”，而被以降順來排序。

亦即是，在耦合節點樹中，索引鍵係被排序而配置在樹上。

當以檢索鍵來作檢索時，則係成爲遍歷索引鍵之在耦合節點樹上所被配置之路徑，例如，若是檢索鍵係爲“101100”，則係能到達節點 211h。又，從上述之說明，可以想像到，就算是當使用“101101”或是“101110”來作爲檢索鍵的情況時，亦會到達節點 211h，但是，藉由將其與索引鍵 251h 作比較，檢索會成爲失敗。

又，就算是在例如以“100100”來作檢索的情況時，亦由於在節點 210a、211b、210f 的連結路徑中檢索鍵的第 3 位元與第 4 位元係不會被使用，且“100100”的第 5 位元係爲 0，因此係和以“100010”來檢索的情況時相同，而成爲到達節點 210g。如此這般，係成爲使用因應於被儲存在耦合節點樹中之索引鍵的位元構成之辨別位元位置而進行分歧。

圖 3，係爲說明用以實施本發明之硬體的構成例之圖。

本發明之檢索裝置所致的檢索處理以及資料維護，係

藉由至少具備有中央處理裝置 302 以及快取記憶體 303 的資料處理裝置 301，並使用資料儲存裝置 308 而被實施。資料儲存裝置 308，係具備有：耦合節點樹所被配置之配列 309、以及記憶有在檢索中所遍歷之節點所被儲存的配列要素之配列號碼的探索經路堆疊 310，該資料儲存裝置 308，係可藉由主記憶裝置 305 又或是外部記憶裝置 306 而實現，或者是，亦可使用經由通訊裝置 307 而被連接之被配置在遠方的裝置。圖 1 之配列 100，係爲配列 309 之其中一例。

在圖 3 之例示中，主記憶裝置 305、外部記憶裝置 306 以及通訊裝置 307，雖係藉由一條的匯流排 304 而被連接於資料處理裝置 301，但是，連接方法係並不限定於此。又，亦可將主記憶裝置 305 設爲在資料處理裝置 301 內者，且亦可將探索路徑堆疊 310 作爲中央處理裝置 302 內之硬體而實現之。或者是，不用說，亦可使外部記憶裝置 306 具有配列 309，或是使主記憶裝置 305 具備有探索路徑堆疊 310 等，而因應於可使用之硬體環境、索引鍵集合之大小等，來適宜地對硬體構成作選擇。

又，雖並未特別作圖示，但是，當然的，爲了將在處理之途中所得到的各種值使用於後面之處理中，係因應於各別之處理而使用有暫時記憶裝置。

接下來，在爲了理解本發明所需要之範圍內，對在上述之申請中，由本申請人所提案之使用有耦合節點樹之基本的檢索處理作介紹。

圖 4，係爲展示在身爲本申請人所致之申請的上述日本特願 2006-293619 中所提案之位元列檢索的基本動作之流程圖。

首先，在步驟 S401 中，取得檢索開始節點之配列號碼。對應於所取得之配列號碼的配列要素，係爲儲存構成耦合節點樹之任意的節點者。檢索開始節點之指定，係在最大值檢索等之各種應用檢索中被進行。

接下來，在步驟 S402 中，將所取得之配列號碼儲存在探索經路 310 中，在步驟 S403 中，將對應於該配列號碼之配列要素作爲應參考之節點而讀出。而後，在步驟 S404 中，從所讀出之節點，來取出節點種類別，在步驟 S405 中，判定節點種類別是否爲分支節點。

在步驟 S405 之判定中，當所讀出之節點係爲分支節點的情況時，前進至步驟 S406，並從節點取出關於辨別位元位置之資訊，進而，在步驟 S407 中，將對應於所取出之辨別位元位置的位元值從檢索鍵中取出。而後，在步驟 S408 中，從節點而取出代表節點號碼，在步驟 S409 中，將從檢索鍵所取出之位元值與代表節點號碼作加算，得到新的配列號碼，而回到步驟 S402。

之後，反覆進行從步驟 S402 起直到步驟 S409 爲止之處理，直到在步驟 S405 之判定中被判定爲葉節點，而前進至步驟 S410 爲止。在步驟 S410 中，從葉節點中取出索引鍵，並結束處理。

另外，耦合節點樹，係可藉由反覆進行插入葉節點之

處理而產生。若依據上述日本特願 2006-187827，則在該插入處理中，首先，係將所指定之插入鍵作爲檢索鍵，並對耦合節點樹而檢索該當之葉節點。此時，將直到到達該葉節點爲止所遍歷了的經過路徑上之分支節點以及該當葉節點所被儲存之配列要素的配列號碼，依序儲存在堆疊中。而後，在檢索鍵與被包含於葉節點中之索引鍵之間，進行大小比較與位元列比較。接下來，藉由在位元列比較中成爲相異之位元值的前端之位元位置，和被儲存在堆疊中之分支節點的辨別位元位置，其兩者間之相對性的位置關係，來決定由包含有插入鍵之葉節點與另外一方之節點所成的新的節點對之插入位置。而後，藉由大小關係，來決定將包含有插入鍵之葉節點，作爲所插入之節點對的何者之節點，並插入節點對。

又，將包含有所指定之索引鍵的葉節點從耦合節點樹而削除之方法，亦係記載於上述日本特願 2006-187827 中。若藉由此，則藉由將削除鍵作爲檢索鍵而從耦合節點樹來檢索出該當之葉節點，並將與該葉節點一同構成節點對的節點之內容，寫入至該節點對之連結源頭的分支節點中，而削除該節點對，藉由此，能夠將包含有所指定之索引鍵的葉節點削除。

又，在日本特願 2006-293619 中，係亦記載有將所有之與被指定有 0～n（n≧0）位元之值的前方一致鍵在 0～n 位元爲完全一致之索引鍵取出的前方一致檢索之方法。例如，當作爲前方一致鍵而指定有“10＊＊＊＊”的情況時

，係爲 n＝1，而第 0～1 位元爲與前方一致鍵一致之索引鍵，係全部被取出。

其具體之方法，係如下所述。首先，取得將前方一致鍵之所有的“＊”置換爲“0”後的下限鍵，和將前方一致鍵之所有的“＊”置換爲“1”後的上限鍵。而後，將藉由下限鍵與上限鍵所指定之檢索範圍的索引鍵，以昇順來從耦合節點樹中取出。

將所指定之檢索範圍的索引鍵以昇順來從耦合節點樹中取出之處理，係如下述一般而進行。首先，從下限鍵而取得身爲下限鍵以上之索引鍵的最小值之下限值，並從上限鍵而取得身爲上限鍵以下之索引鍵的最大值之上限值。

下限值之取得，係藉由一面以將節點〔0〕側以及深度方向爲優先之深度優先探索順序，來對耦合節點樹之葉節點作探索，一面將葉節點之索引鍵與下限鍵作比較，而進行之。上限值之取得方法，係除了將探索順序設爲以節點〔1〕側以及深度方向爲優先之深度優先探索順序之外，爲採用與下限值之取得相同之方法。

而後，從儲存有下限值之索引鍵的葉節點而開始，並以將節點〔0〕側以及深度方向爲優先之深度優先探索順序，來對耦合節點樹之葉節點作探索，而依序從葉節點來取出索引鍵，直到找到上限值之索引鍵爲止。

藉由以上之處理，能夠將第 0～n 位元爲與前方一致鍵完全一致之索引鍵全部取出。

又，在日本特願 2007-132289 中，係記載有：從與被

指定之最長一致檢索鍵部分一致之索引鍵中，檢索出差分位元位置成爲最下位之索引鍵的最長一致檢索方法。另外，在此，所謂差分位元位置，係指將 2 個的位元列作比較時，在位元値爲不一致之不一致位元中的最爲上位之位置。

其具體之方法，係如下所述。首先，將最長一致檢索鍵作爲檢索鍵，並將根節點作爲檢索開始節點，而進行圖 4 之檢索，並將其結果所得到之索引鍵與最長一致檢索鍵作比較，而求取出差分位元位置。

而後，對在圖 4 之檢索中所記憶之探索路徑堆疊作回遡，並將接在辨別位元位置爲較差分位元位置更上位之分支節點中的辨別位元位置爲最下位之分支節點後而被記憶在探索路徑堆疊中之節點，作爲最長一致節點而取得。

由耦合節點樹之構造可以得知，在將如同上述一般所取得之最長一致節點作爲根節點的部分樹中所包含之所有的索引鍵，均係滿足「與最長一致檢索鍵部分一致之索引鍵中，差分位元位置成爲最下位者」之條件。故而，只要因應於需要而從此部分樹中適宜地將索引鍵取出即可。

以上，雖係對於成爲本發明之前提的技術作了說明，但是，若是有必要，則請參考上述之專利申請案的說明書以及圖面的記載。

另外，若是對上述日本特願 2006-293619 所致之前方一致檢索以及日本特願 2007-132289 所致之最長一致檢索的與本發明所致之檢索間的關連性作說明，則係如同下述

一般。

上述之前方一致檢索中的前方一致鍵，係包含有隨意位元。又，上述之最長一致檢索，係有將與最長一致檢索鍵僅有部分一致的索引鍵作爲結果而得到的情況，關於此點，亦可視爲將不一致之範圍作爲隨意位元來處理的情況。

但是，上述之前方一致檢索與最長一致檢索，係兩者均並非爲取得被唯一性訂定之索引鍵的處理。

亦即是，藉由上述之前方一致檢索所取出的索引鍵之數，係爲 2 以上。又，在上述之最長一致檢索中，將最長一致節點作爲根節點之部分樹，係亦有具備複數之葉節點的情況，而關於從此部分樹中而取出何者之索引鍵一事，係爲任意。

亦即是，在上述之前方一致檢索與最長一致檢索中，係會有在滿足檢索之條件的前提下，而存在有條件相互對等之複數的索引鍵的情況。故而，依存於檢索之目的，會有需要以某種之基準來從此些之複數的索引鍵中來選擇出 1 個的情況。

另一方面，在路由器中之選路表的檢索等一般之要求有非常高速之處理的技術領域中，如同上述一般而進行檢索與選擇之兩階段之處理一事，係非常不理想。因此，就算是在如同前方一致檢索或是最長一致檢索一般之基於部分性之一致的檢索中，亦期望能夠提供一種：可唯一性地訂定最適當之 1 個的索引鍵，而不需要進行第 2 階段之選

擇處理，並取得該唯一性之索引鍵的方法。

如同以下所說明一般，本發明所致之考慮有隨意位元之檢索，係爲用以滿足此種要求者。上述之前方一致檢索、最長一致檢索、以及以下所述之考慮有隨意位元之檢索，係應該因應於檢索之目的而分別作使用。

接下來，針對本發明之實施形態作詳細說明。以下，係先對位元列之編碼方法的例子以及耦合節點樹之例子作說明，而後針對檢索、插入、削除之各處理作說明。

在以下之說明中，係將編碼前與編碼後之位元列，分別稱爲「元位元列」和「編碼位元列」。「索引鍵」與「檢索鍵」之名詞，若是未特別作說明，則係代表其爲身爲編碼位元列之狀態，當代表元位元列之狀態的情況時，則係如同「元索引鍵」或是「元檢索鍵」一般地作標記。

又，將值被訂定爲“0”或是“1”之位元，稱爲「有意位元」，將值係可爲“0”或是“1”之任一者，而並未被唯一性地訂定之位元，稱爲「隨意位元」。另外，隨意位元，係與藉由“0”又或是“1”來作表示之有意位元作區分，而藉由符號“＊”來作標記。

又，於以下，係將隨意位元並不存在於較有意位元爲更前方一事作爲前提。亦即是，元位元列，係僅由隨意位元所成，或是僅有有意位元所成，又或是爲由在 1 位元以上之有意位元之後而接續有 1 位元以上之隨意位元所成。

爲了將此種具備包含隨意位元之可能性的元位元列作編碼，係採用將各位元編碼爲 2 位元之編碼方法。

編碼位元列，係將由 2 位元所成之位元對作爲單位。各位元對之第 0 位元，係爲代表其爲隨意位元或是有意位元一事的識別位元。各位元對之第 1 位元，係稱爲資料位元。

識別位元，當值係爲“0”時，係代表隨意位元，而當值爲“1”時，係代表有意位元。

當識別位元之值爲“0”時，與其成對之資料位元的值，係必定爲預先被作了決定之值，在以下之實施形態中，係作爲預先被決定之該值，而採用“0”。當識別位元之值爲“1”時，資料位元係代表元位元列之位元的值。

亦即是，資料位元之各位元，當其係爲值“0”之有意位元時，係被編碼爲“10”之位元對，而當當其係爲值“1”之有意位元時，係被編碼爲“11”之位元對，當其係爲隨意位元時，則係被編碼爲“00”之位元對。

若藉由此編碼方法，則可以從編碼位元列，來簡單的判斷出在元位元列中之後續是否存在有有意位元。此係因爲，編碼位元列中之任意的位元，係經由該位元之位置爲偶數亦或是奇數一事，而能夠簡單地區別出該位元係爲識別位元或是資料位元，且從上述之前提，可以得知隨意位元係不會存在於較有意位元爲更前方處之故。

接下來，參考圖 5，並針對利用有編碼位元列之耦合節點樹之例作說明。圖 5 之（a），係對對應於 6 個的未被編碼之元索引鍵“＊＊＊＊＊＊”、“1＊＊＊＊＊”、“101＊＊”、“1011＊”、“1100＊”、“111＊＊”之集合的耦合節點

樹之概念性構造作展示。

另外，圖 5 雖係亦包含有用以對後述之檢索處理作說明之部分（b）以及（c），但是，於此，係先僅對耦合節點樹之構造作說明。

圖 5 之（a）的耦合節點樹之各葉節點，係作爲索引鍵，而儲存有將上述之 6 個的元索引鍵分別作了編碼後之編碼位元列。於圖 5 中，爲了幫助理解，各元索引鍵之值，係在各索引鍵之近旁的括弧內作展示。

除了被儲存於索引鍵中的係爲編碼位元列一點以外，由於圖 5 與圖 2 係爲相同，故在圖 5 中，亦使用與圖 2 相同之符號。

亦即是，根節點 210a 係被配置於配列號碼 220 中，而爲節點對 201a 的代表節點。作爲樹狀構造，於根節點 210a 之下方係配置有節點對 201b，而於其下層係配置有節點對 201c，並更進而在其下層而被配置有節點對 201d。而，在節點對 201d 之下層，係被配置有節點對 201e 與節點對 201f。

根節點 210a 之節點種類別 260a 係爲 0，代表其係爲分支節點，辨別位元位置 230a 係爲 0，在代表節點號碼中，係被儲存有節點對 201b 之代表節點 210b 所被儲存之配列要素的配列號碼 220a。

節點對 201b，係由節點種類別 260b 爲 1 而身爲葉節點之節點 210b、和節點種類別 261b 爲 0 而身爲分支節點之節點 211b 所構成。

在節點 210b 之索引鍵 250b 中，係被儲存有將元索引鍵“＊＊＊＊＊”編碼後之“0000000000”。

另一方面，在節點 211b 之辨別位元位置 231b，係被儲存有 2，在代表節點號碼中，係被儲存有節點對 201c 之代表節點 210c 所被儲存之配列要素的配列號碼 221b。

節點對 201c，係由節點種類別 260c 爲 1 而身爲葉節點之節點 210c、和節點種類別 261b 爲 0 而身爲分支節點之節點 211c 所構成。

在節點 210c 之索引鍵 250c 中，係被儲存有將元索引鍵“1＊＊＊＊”編碼後之“1100000000”。

另一方面，在節點 211c 之辨別位元位置 231c，係被儲存有 3，在代表節點號碼中，係被儲存有節點對 201d 之代表節點 210d 所被儲存之配列要素的配列號碼 221c。

節點對 201d 係以節點 210d 與 211d 所構成，此些之節點種類別 260d、261d 係均爲 0，而表示其係爲分支節點。

在節點 210d 之辨別位元位置 230d，係被儲存有 6，在代表節點號碼中，係被儲存有節點對 201e 之代表節點 210e 所被儲存之配列要素的配列號碼 220d。

另一方面，在節點 211d 之辨別位元位置 231d，係被儲存有 5，在代表節點號碼中，係被儲存有節點對 201f 之代表節點 210f 所被儲存之配列要素的配列號碼 221d。

節點對 201e 係以節點 210e 與 211e 所構成，此些之節點種類別 260e、261e 係均爲 1，而表示其係爲葉節點。

在節點 210e 之索引鍵 250e 中，係被儲存有將元索引鍵“101＊＊”編碼後之“1110110000”。

另一方面，在節點 211e 之索引鍵 251e 中，係被儲存有將元索引鍵“1011＊”編碼後之“1110111100”。

節點對 201f 係以節點 210f 與 211f 所構成，此些之節點種類別 260f、261f 係均爲 1，而表示其係爲葉節點。

在節點 210f 之索引鍵 250f 中，係被儲存有將元索引鍵“1100＊”編碼後之“1111101000”。

另一方面，在節點 211f 之索引鍵 251f 中，係被儲存有將元索引鍵“111＊＊”編碼後之“1111110000”。

當作爲檢索鍵而被使用之編碼位元列，爲與被包含在耦合節點樹中之索引鍵中的 1 個爲完全一致的情況時，藉由與圖 4 相同之處理，能夠將與檢索鍵爲相等之索引鍵作爲檢索結果而得出。

但是，若是考慮隨意位元，則係有應將與檢索鍵部分一致之索引鍵作爲檢索結果而得出的情況。關於此種依據部分一致之檢索的詳細內容，係於後參考圖 6～圖 11 而作說明。

接下來，針對圖 5 之（a）的耦合節點樹之構成的意義作說明。如同圖 5 之（a）一般，就算是在使用編碼位元列的情況中，亦與圖 2 相同的，耦合節點樹之構成，係藉由索引鍵之集合而被規定。

在圖 5 之（a）的例中，根節點 210a 之辨別位元位置爲 0 的原因，係因爲在圖 5 中之索引鍵中，具備有第 0 位

元為 0 者以及為 1 者之故。第 0 位元為 0 之索引鍵，由於係僅有 1 個，因此係被分類至節點 210b 之下，而第 0 位元為 1 之索引鍵，係被分類至節點 211b 之下層。

節點 211b 之辨別位元位置 231b 為 2 的原因，係因為其係反映有：在節點 211b 之下層的各節點處所儲存之第 0 位元為 1 的索引鍵，其第 1 位元係全部等於 1，而為在第 2 位元才開始有所不同者一般之索引鍵的集合之性質之故。

以下，同樣的，第 0～1 位元為“11”之索引鍵中，第 2 位元為 0 之索引鍵，由於係僅有 1 個，因此係被分類至節點 210c，而第 2 位元為 1 之索引鍵，係被分類至節點 211c 之下層。節點 211c 之辨別位元位置 231 為 3 的原因，係因為其係反映有：在被儲存於節點 211c 之下層的各節點處之索引鍵中，係存在有第 3 位元為 0 者與第 3 位元為 1 者一事。

節點 210d 之辨別位元位置 230d 為 6 的原因，係因為其係反映有：在節點 210d 之下層的各節點處所儲存之第 0～3 位元為“1110”的索引鍵，其第 4～5 位元係全部為“11”而相等，並為在第 6 位元才開始有所不同者一般之索引鍵的集合之性質之故。

在節點 210d 之連結目標處，其第 6 位元為 0 之索引鍵與為 1 之索引鍵，係分別各只有 1 個。故而，節點 210e 與 211e 係為葉節點，在節點 210e 與 211e 之索引鍵 250e、251e 中，係分別被儲存有身為“101＊＊“之編碼位元列

的“1110110000”和身爲“1011＊”之編碼位元列的“1110111100”。

另一方面，節點211d之辨別位元位置231d爲5的原因，係因爲其係反映有：在節點211d之下層的各節點處所儲存之第0～3位元爲“1111”的索引鍵，其第4位元係全部爲“1”而相等，並爲在第5位元才開始有所不同者一般之索引鍵的集合之性質之故。

在節點211d之連結目標處，其第5位元爲0之索引鍵與爲1之索引鍵，係分別各只有1個。故而，節點210f與211f係爲葉節點，在節點210f與211f之索引鍵250f、251f中，係分別被儲存有身爲“1100＊”之編碼位元列的“1111101000”和身爲“111＊＊”之編碼位元列的“1111110000”。

如此這般，在圖5之（a）中，身爲編碼位元列之索引鍵的集合，係與耦合節點樹之樹構造相對應。

又，在本實施形態之編碼方法中，2個的元位元列之第0～m（0≦m）位元爲一致一事，和將該2個的元位元列編碼後之2個的編碼位元列之第0～（2m＋1）位元爲一致一事，係爲同值。故而，耦合節點樹之樹構造，係亦可藉由編碼前之元索引鍵的集合而被規定，而在樹構造與元索引鍵之集合之間，係存在有對應關係。

亦即是，在元索引鍵之中，第0位元爲隨意位元者，係僅有“＊＊＊＊＊”，因此，“＊＊＊＊＊”係被分類至節點210a之下的節點210b處，而第0位元爲有意位元之元

索引鍵，係被分類至與節點 210b 成對之節點 211b 的下層。

被分類至節點 211b 之下層的元索引鍵，第 0 位元係全部等於“1”。故而，關於第 0 位元係爲“0”或是“1”一事之分類係成爲不必要。

另一方面，在此些之元索引鍵之中，第 1 位元爲隨意位元者，由於係僅有“1＊＊＊＊”之元索引鍵，因此，此係被分類至節點 210c 處，而第 1 位元爲有意位元之元索引鍵，係被分類至與節點 210c 成對之節點 211c 的下層。

在被分類至節點 211c 之下層的至少第 0～1 位元係爲有意位元之元索引鍵中，係存在有第 1 位元爲“0”者與爲“1”者。於此，第 0～1 位元爲“10”者，係被分類至節點 210d 之下層，而第 0～1 位元爲“11”者，係被分類至節點 211d 之下層。

被分類至節點 210d 之下層的元索引鍵，係爲“101＊＊”與“1011＊”之 2 個。此 2 個的元索引鍵，其第 0～2 位元係完全一致，而在第 3 位元係爲隨意位元或是有意位元一事上係爲相異。

在本實施形態中，係將隨意位元“＊”編碼爲“00”。故而，如此這般，當第 n 位元（n≧1）以後係爲隨意位元之元索引鍵，和第 n 位元係爲有意位元之元索引鍵，在第 0～（n-1）位元之範圍內係爲完全一致的情況時，則前者係被分類至節點〔0〕側。

故而，元索引鍵“101＊＊”係被分類至節點 210e，

“1011＊”係被分類至節點 211e。

另一方面，被分類至節點 211d 之下層的元索引鍵，係爲“1100＊”與“111＊＊”之 2 個。此 2 個的元索引鍵，其第 0～1 位元係完全一致，但在第 2 位元之值係爲相異。

如此這般，當從前端之第 0 位元起而依序比較，並當尚未到達任一者之元索引鍵的隨意位元之前便發現值爲相異之有意位元的情況時，該有意位元之值爲“0”的元索引鍵，係被分類至節點〔0〕側，而該有意位元之值爲“1”之元索引鍵，則係被分類至節點〔1〕側。

故而，元索引鍵“1100＊”係被分類至節點 210f，“111＊＊”係被分類至節點 211f。

如上述一般，將編碼位元列分類所得到之階層構造，與將元位元列分類所得之階層構造，係爲同值。

又，由於隨意位元之值係爲不定，因此，關於隨意位元與有意位元之值中何者係爲較大一事，原本係並不一定。但是，便利上，若是將隨意位元“＊”定義爲較“0”值之有意位元爲更小，則耦合節點樹之構造，係可以說是反映了元索引鍵之順序性者。

亦即是，與圖 2 相同的，在圖 5 之耦合節點樹中，若是採用以節點〔1〕側爲優先之深度優先探索順序來遍歷葉節點，則被儲存於該些之葉節點中的索引鍵所表示之元索引鍵，係以降順而被排序。換言之，各索引鍵，係以對應於將元索引鍵以降順來作了排序後之順序，而被配置於

耦合節點樹上。

另外，在本實施形態中，將代表隨意位元之識別位元的值設爲“0”的原因，係爲了配合上述之“＊”爲較“0”更小之定義之故。

接下來，針對在本發明之其中一種實施形態中的檢索處理，參考圖6之流程圖來作說明。在圖6之檢索處理中，係將身爲未被編碼之元位元列的原檢索鍵作爲輸入而指定，而使例如圖5一般之使用有編碼位元列之耦合節點樹被作檢索。

圖6之檢索處理，係爲：若是滿足以下所述之「最長一致鍵」的條件之索引鍵係存在於耦合節點樹中，則取得該最長一致鍵的處理。而，若是在耦合節點樹中係並不包含有該滿足最長一致鍵的條件之索引鍵，則檢索係爲失敗並結束。

在本實施形態中之所謂最長一致鍵，係代表相當於下述之（a）與（b）的任一者之索引鍵。

（a）與將原檢索鍵編碼後之編碼位元列爲完全相同之索引鍵。

（b）在不存在有有意位元、或是雖存在有較元檢索鍵爲更少數量之有意位元，但是所有之有意位元之值係爲與元檢索鍵者爲一致之元索引鍵中，將有意位元數爲最大者作了編碼後之索引鍵。

另外，上述之（a）的優先度係較（b）爲更高。亦即是，若是存在有該當於（a）之索引鍵，則該索引鍵係爲

最長一致鍵，此時，就算是存在有該當於（b）之索引鍵，其亦並非爲最長一致鍵。

在開始圖 6 之具體說明前，針對上述（a）與（b）之定義，而舉出數個例子來作說明。

上述（a）之例，在圖 5 之（a）的耦合節點樹中，當原檢索鍵係爲“1100＊”的情況時，則係爲身爲將“1100＊”編碼後之編碼位元列的節點 210f 之索引鍵 250f。

又，在圖 5 之（a）的耦合節點樹中，若是列舉出該當於上述（b）之最長一致鍵的 3 個例子，則係如下述一般。

第 1 個例子，係爲當元檢索鍵爲“11001”時之例。此元檢索鍵之有意位元的數量，係爲 5。

在圖 5 之（a）中，係存在有不存在有意位元之元索引鍵“＊＊＊＊＊”。又，在存在有較 5 爲更少數量之有意位元，但所有之有意位元之值爲與原檢索鍵者一致之元索引鍵中，係存在有“1＊＊＊＊”和“1100＊”。

在此些之 3 個的元索引鍵中，將有意位元之數量爲最大之“1100＊”編碼後之節點 210f 的索引鍵 250f，係爲最長一致鍵。

第 2 個例子，係爲當元檢索鍵爲“11＊＊＊”時之例。此元檢索鍵之有意位元的數量，係爲 2。

在圖 5 中，係存在有不存在有意位元之元索引鍵“＊＊＊＊＊”。又，在存在有較 2 爲更少數量之有意位元，但所有之有意位元之值爲與原檢索鍵者一致之元索引鍵中

，係存在有“1＊＊＊＊”。

在此些之 2 個的元索引鍵中，將有意位元之數量爲最大之“1＊＊＊＊”編碼後之節點 210c 的索引鍵 250c，係爲最長一致鍵。

另外，在耦合節點樹內，係亦存在有將元索引鍵“1100＊”或“111＊＊”編碼後之索引鍵 250f 以及 251f。與“11＊＊＊”一致之範圍，一見之下，相較於“1＊＊＊＊”，“1100＊”或是“111＊＊”看起來係爲更廣。

但是，索引鍵 250f 以及 251f，係從最長一致鍵之定義而脫離。若是對將最長一致鍵作此種定義的理由作補足，則係如同下述所述一般。

由於隨意位元係代表任意值，因此，包含有隨意位元之元位元列，係可視爲代表位元列之集合。例如，“11＊＊＊”所代表之集合，係包含有“11011”之元位元列等。

但是，“11011”，係與“1100＊”在第 3 位元爲相異，而與“111＊＊”在第 2 位元爲相異。亦即是，“11011”，係並不被包含於“1100＊”或是“111＊＊”中。故而，對於元檢索鍵“11＊＊＊”，將索引鍵 250f 又或是 251f 作爲最長一致鍵係爲不適當。

另一方面，不論將元檢索鍵“11＊＊＊”之隨意位元取何種值，在隨意位元處被設定有值之該位元列，係必定被包含於元索引鍵“＊＊＊＊＊”所代表之集合、或是元索引鍵“1＊＊＊＊”所代表之集合中。

而，在 2 個的元索引鍵中，具備有更多數之有意位元

的元索引鍵“1＊＊＊＊”所代表的集合係爲較小，而更接近於元檢索鍵“11＊＊＊”所代表之集合。故而，對於元檢索鍵“11＊＊＊”之最長一致鍵，係爲元索引鍵“1＊＊＊＊”。

第 3 個例子，係爲當元檢索鍵爲“0＊＊＊＊”時之例。在圖 5 之（a）的耦合節點樹中，係並不包含有將從“0”開始之元索引鍵作了編碼後之索引鍵。

但是，此時，若是依據上述（b）之定義，則身爲將元索引鍵“＊＊＊＊＊”編碼後之編碼位元列的節點 210b 之索引鍵 250b，係爲最長一致鍵。實際上，不論將元檢索鍵“0＊＊＊＊”之隨意位元取何種值，在隨意位元處被設定有值之該位元列，係必定被包含於元索引鍵“＊＊＊＊＊”所代表之集合中。

取得如同上述一般所被定義之最長一致鍵一事，例如，係適合於如同 IP 位址一般，在位元列之值與位元列所代表之內容的階層性分類相對應一般的情況中，選擇出最爲適當的分類。

接下來，一面參考圖 6 之流程圖，一面對圖 6 之處理作詳細說明。藉由圖 6 中所示之處理，只要存在有藉由上述之（a）以及（b）所定義之最長一致鍵，則係可得到，而若是不存在，則係得到檢索失敗之結果。

在步驟 S601 中，從所指定的元檢索鍵，經由上述之編碼方法而作成檢索鍵。步驟 S601 之詳細內容，係於後參考圖 7 而作說明。例如，在圖 5 之（b）中，係例示有

由 5 位元之元檢索鍵“10100”所作成之 10 位元的檢索鍵 270。

接下來，在步驟 S602 中，將檢索對象之耦合節點樹的根節點，設定爲檢索開始節點。

接下來，在步驟 S603 中，將儲存耦合節點樹之節點的配列，藉由檢索鍵來從檢索開始節點而進行檢索，並得到作爲檢索結果之索引鍵。步驟 S603 之處理的詳細內容，係於後配合圖 8 而作說明。爲了在後來的步驟 S606 中作參考，在步驟 S603 中，係如同圖 5 之（c）中所例示一般，將配列號碼儲存在探索路徑堆疊 310 中。

在接下來的步驟 S604 中，係求取出檢索鍵與在步驟 S603 所取得之索引鍵，其兩者間的差分位元位置。差分位元位置之定義、以及步驟 S604 之詳細內容，係於後配合圖 9 以及圖 10 而作說明。

接下來，在步驟 S606 中，係依據檢索鍵與在步驟 S603 所取得之索引鍵，而回遡探索路徑堆疊，並得到最長一致鍵。步驟 S606 之處理的詳細內容，係於後配合圖 11 而作說明。

若是在步驟 S606 中得到了最長一致鍵，則檢索係爲成功，而圖 6 之處理係結束。若是在步驟 S606 中並未得到最長一致鍵，則檢索係爲失敗，而圖 6 之處理係結束。

接下來，參考圖 7 之流程圖，針對將作爲輸入位元列而被賦予之元位元列作編碼並輸出編碼位元列的編碼處理作說明。圖 7 之編碼處理，係相當於圖 6 之步驟 S601。

又，如後述一般，在插入處理以及削除處理中，亦係被實行有圖 7 之編碼處理。

在步驟 S701 中，將身爲輸入位元列之有意位元部分之長度的有意位元長度，作爲位元長度而作記憶。

又，在步驟 S702 中，將代表輸入位元列中之下一個應處理的位元之位置的位元位置初期化。在本實施形態中，爲了從第 0 位元而依序作處理，在步驟 S702 中，係將位元位置初期化爲 0。

而後，在步驟 S703 中，將輸出位元列初期化並作爲空位元列。位元位置與輸出位元列之值，係在後述之步驟 S704～步驟 S708 之迴圈中被作更新。

接下來，在步驟 S704 中，判定現在之位元位置是否爲較在步驟 S701 中所記憶之位元長度更小，若是現在的位元位置爲較小，則前進至步驟 S705，當此之外的情況時，則前進至步驟 S709。

在步驟 S705 中，係從輸入位元列中將現在之位元位置所指向之位元值取出。

接下來，在步驟 S706 中，在輸出位元列之尾端連結位元值 1。而後，在步驟 S707 中，在輸出位元列之尾端連結於步驟 S705 中所得到的位元值。

而後，在步驟 S708 中，將位元位置更新爲下一個的位置，並回到步驟 S704。另外，在本實施形態中，由於係從左方起而依序計數第 0、第 1、第 2…位元，因此，所謂「下一個位置」，係指現在的位元位置之右邊的鄰接位

置。

在步驟 S709 中，在輸出位元列之尾端，連結輸入位元列中的隨意位元之數量個的“00”之位元對，並結束處理。

藉由以上之處理，例如，從元位元列“101＊＊”，係可得到編碼位元列“1110110000”。

另外，在本實施形態中，元位元列中之有意位元的數量與隨意位元的數量，在輸入位元列其本身中，係設爲分開作指定者。但是，當僅對固定長度之位元列作處理的情況時，由於係可從有意位元之數量來求取出隨意位元之數量，且亦可相反地從隨意位元之數量來求取出有意位元之數量，因此係可僅對其中一方作指定。

接下來，一面參考圖 8 之流程圖，一面對進行初期檢索之圖 6 的步驟 S603 之處理作詳細說明。

在步驟 S801 中，將探索經路堆疊之堆疊指標的值設定爲初期值。此初期值，係爲當在探索路徑堆疊中並未被儲存有任何東西時之值，而在後述之圖 11 的處理中亦會被使用。本實施形態之在圖 8 的處理中之堆疊指標，係代表接下來在步驟 S813 中應將配列號碼推入的探索路徑上之位置。於以下作說明。

接下來，在步驟 S802 中，取得檢索開始節點之配列號碼。圖 8 之處理所被實行的時間，由於係在圖 6 之步驟 S602 被實行之後，因此，在步驟 S802 中，具體而言，係得到根節點之配列號碼。

接著，在步驟 S803 中，係從儲存耦合節點樹之節點的配列中，將在步驟 S801 中所取得的配列號碼所指向之配列要素作爲節點而讀取出。

而後，在步驟 S804 中，從在步驟 S803 所讀出之節點，來取出節點種類別，在步驟 S805 中，判定節點種類別是否爲分支節點。

在步驟 S805 之判定中，當所讀出之節點係爲分支節點的情況時，前進至步驟 S806，並從節點取出關於辨別位元位置之資訊，進而，在步驟 S807 中，將對應於所取出之辨別位元位置的位元值從檢索鍵中取出。而後，在步驟 S808 中，從節點而取出代表節點號碼。

接下來，在步驟 S811 中，判定在步驟 S806 所取出之辨別位元位置是否爲偶數。

若依據本實施形態之編碼，則在編碼位元列中，存在於偶數之位元位置處的位元，係爲識別位元，而存在於奇數之位元位置處的位元，係爲資料位元。故而，當步驟 S811 之判定係爲偶數的情況時，係爲了判定後面是否連續有將有意位元編碼後之位元，而前進至步驟 S812，並判定在步驟 S807 中所取出之位元值是否爲 1。

當步驟 S812 之判定係爲 1 的情況時，則在本實施形態之編碼中，元檢索鍵之具備有意位元之值的資料位元，係仍殘留在檢索鍵中。故而，前進至步驟 S813 中，將在步驟 S803 中的讀取出了節點之配列號碼，儲存在探索路徑堆疊中。

接下來，在步驟 S814 中，在以步驟 S808 所取出之代表節點號碼處加上 1，而取得身爲在步驟 S803 所讀出之節點的子節點之節點〔1〕的配列號碼，並設定爲新的配列號碼。

而後，在步驟 S815 中，將以步驟 S814 所得到之子節點的配列號碼儲存在探索路徑堆疊中，並將堆疊指標之值增加 1 個，而回到步驟 S803。

另外，在此之「增加 1 個」的表現法，係爲配合於如同圖 5 之（c）一般而將探索路徑堆疊 310 分爲 2 列的圖示說明所作的表現，而並非爲對探索路徑堆疊 310 以及堆疊指標之具體的安裝方法作限定的意義。

在圖 5 之（c）的探索路徑堆疊 310 中，係將在步驟 S813 所儲存的配列號碼圖示於左列，並將在步驟 S815 所儲存之配列號碼圖示於右側。在步驟 S815 中所進行之堆疊指標的值之更新，係相當於將圖 5（c）中之堆疊指標所指向之行移動至下一行一事。

另一方面，當步驟 S811 之判定係爲奇數的情況時，係代表辨別位元位置係爲資料位元之位置。又，當步驟 S812 之判定係爲 0 的情況時，則在本實施形態之編碼中，係代表識別位元爲 0，亦即是代表已到達了元檢索鍵中之將隨意位元編碼後的位元位置處。

當在步驟 S811 中被判定爲奇數之情況、以及在步驟 S812 中被判定爲 0 之情況時，兩者均係前進至步驟 S809。

在步驟 S809 中，將在步驟 S807 中從檢索鍵所取出之位元值，與在步驟 S808 中所取出之代表節點號碼作加算，並將該加算之結果設定爲新的配列號碼。在步驟 S809 之實行後，處理係回到步驟 S803。

之後，反覆進行從步驟 S803 起直到步驟 S815 爲止之處理，直到在步驟 S805 之判定中被判定爲葉節點，而前進至步驟 S810 爲止。在反覆進行的過程中，在步驟 S809 又或是步驟 S814 中所設定之配列號碼，係在步驟 S803 中被使用。

在步驟 S810 中，從葉節點中取出索引鍵，並結束處理。

另外，由於隨意位元“＊”係被編碼爲“00”，因此，若是將 j 設爲 0 以上之整數，則若是編碼位元列之第 2j 位元係爲“0”，則編碼位元列之第（2j＋1）位元係必定爲“0”。

而，辨別位元位置爲 2j 之分支節點的子節點中之節點〔0〕，係必定爲儲存有將第 j 位元以後全部均爲隨意位元的元索引鍵作編碼後之索引鍵的葉節點。

故而，當處理從步驟 S811 而前進至步驟 S812、S809、S803、S804、S805 的情況時，在步驟 S805 中，係必定被判定爲葉節點，而必定會分歧至步驟 S810。

將此身爲節點〔0〕之葉節點，從身爲母節點之分支節點的角度來看，（以到達了有意位元之終端一事的意涵）而稱之爲「終端側節點」，並將與終端側節點成對之節點〔1〕，稱爲「非終端側節點」。後述之圖 11 的步驟

S1107 與步驟 S1108，係爲從非終端側節點而求取出終端側節點的處理。

接下來，一面參考圖 9 之流程圖，一面對求取差分位元位置之圖 6 的步驟 S604 之處理作詳細說明。

在步驟 S901 中，設定比較位元長度。在比較位元長度中所設定之値，係爲在檢索鍵中，從身爲最上位之第 0 位元起，直到將元檢索鍵之最上位的隨意位元作了編碼後之位元對的位置、和以圖 6 之步驟 S603 所得到之索引鍵中的將元索引鍵之最下位的有意位元編碼後之位元對的位置，此兩者中的較爲上位之位置爲止的範圍之長度。更正確而言，比較位元長係如同下述之（1）～（3）一般地來決定。

（1）當在元檢索鍵中並不包含有意位元的情況、亦即是元檢索鍵之第 0 位元係爲隨意位元的情況時，則將元檢索鍵之第 0 位元的隨意位元作了編碼後之位元對，係爲檢索鍵之第 0～1 位元。故而，將比較位元長度設定爲 2。

（2）當在元檢索鍵中包含有意位元，但在元索引鍵中並不包含有意位元的情況時，將比較位元長度設定爲 0。

（3）當在元檢索鍵與元索引鍵之雙方中均中包含有意位元的情況時，將比較位元長度如下述一般的作設定。

若是將元檢索鍵之最上位的隨意位元之位置設爲 a（a≧1），則將第 a 位元之隨意位元編碼後之位元對，係位置於檢索鍵之第 2a～（2a＋1）位元處。另外，當元檢索

鍵係爲僅由有意位元而成的情況時，則將元檢索鍵之長度設爲 n，而可視爲 a＝n。

又，若是將元索引鍵之最下位的有意位元之位置設爲 b（b≧0），則將第 b 位元之有意位元編碼後之位元對，係位置於索引鍵之第 2b～（2b＋1）位元處。

故而，當（2a＋1）≦（2b＋1）的情況時，亦即是當 a≦b 的情況時，比較位元長度，係被設定爲從第 0 位元直到第（2a＋1）位元爲止的長度，亦即是被設定爲（2a＋2）。另一方面，當（2a＋1）＞（2b＋1）的情況時，亦即是當 a＞b 的情況時，比較位元長度，係被設定爲從第 0 位元直到第（2b＋1）位元爲止的長度，亦即是被設定爲（2b＋2）。

如同上述（1）～（3）所述一般，在步驟 S901 中所設定之比較位元長度，由於係爲偶數，因此，在以下，爲了方便說明，係將比較位元長度表記爲 2c（c≧0）。

在步驟 S902 中，將檢索鍵與索引鍵，藉由比較位元長度所指向之位元列來作比較，而得到比較位元長度之差分位元列。亦即是，將檢索鍵與索引鍵，在第 0～（2c-1）位元的範圍內作比較，而得到長度爲 2c 之差分位元列。

差分位元列，係爲：在檢索鍵與索引鍵處值係爲一致的位置，該位置之位元的值係爲 0，而在不一致之位置，該位置之位元的值係爲 1，藉由此所成之位元列，並可藉由例如求取出排他性邏輯和之位元演算來得出。

接下來，在步驟 S903 中，從最上位之位置、亦即是第 0 位元起來檢視差分位元列，並將最初之不一致位元、亦即是最初之值爲 1 的位元之位元位置，設定爲差分位置，而結束處理。

當並不存在有不一致位元的情況時，由於之後係並不會有對差分位元位置作參考的情形，因此，在步驟 S903 中，爲了方便起見，亦可將差分位元位置設定爲例如 2c。又，當比較位元長度爲 0 時，爲了方便起見，係將差分位元位置設定爲負數。

接下來，參考圖 10，並針對差分位元位置之具體例作說明。

在圖 10 之例中，檢索鍵，係爲將元檢索鍵“10100”編碼後之編碼位元列“1110111010”。由於此元檢索鍵係僅由有意位元所成，因此，藉由上述（3）之說明，爲了方便起見，係可視爲 a＝5。

另一方面，藉由此檢索鍵而在步驟 S603 之檢索中所得到的索引鍵，係爲將包含有隨意位元之元索引鍵“1011＊”編碼後之索引鍵 251e 之“1110111100”。在此元索引鍵中，最下位之有意位元的位置，係爲 b＝3。

故而，由於 a＞b，因此，在圖 9 之步驟 S901 中，係將比較位元長度設定爲 8（＝2b＋2）。

而後，在步驟 S902 中，得到 8 位元之差分位元列，並在步驟 S903 中，將差分位元位置設定爲 7。藉由以上處理，而結束圖 9 之處理，亦即是結束圖 6 之步驟 S604

。

接下來，一面參考圖 11 之流程圖，一面對取得最長一致鍵之圖 6 的步驟 S606 之處理作詳細說明。

在步驟 S1101 中，將檢索鍵與藉由圖 6 之步驟 S603 所得到的索引鍵，在比較位元長度 2c 所指向之位元列處作比較，亦即是，在從最上位之第 0 位元起直到第（2c-1）位元爲止的範圍內作比較，並判斷兩者是否爲相等。

當在步驟 S1101 中判定此些係爲相等時，處理前進至步驟 S1111，將藉由步驟 S603 而得到之索引鍵，設定爲最長一致鍵，並結束處理。

當在步驟 S1101 中判定係並不相等時，則處理係前進至步驟 S1102。

在步驟 S1102 中，判斷探索路徑堆疊之堆疊指標的值是否爲初期值，若是爲初期值，則並不設定最長一致鍵，並結束圖 11 之處理，而若是並非爲初期值，則前進至步驟 S1103。

如同由圖 8 之步驟 S801 的說明而可得知一般，在本實施形態中，堆疊指標成爲初期值一事，係代表：堆疊指標係指向最初被推入至探索路徑堆疊中的配列號碼、或是探索路徑堆疊係爲空的。

在步驟 S1103 中，係將探索路徑堆疊之堆疊指標倒退一個，並從探索路徑堆疊而取出配列號碼。亦即是，步驟 S1103 之處理，係爲彈出動作。

另外，若是配合在圖 5（c）中之探索路徑堆疊 310 的

圖示來作說明，則步驟 S1103 之處理，係爲在 1 行上移動堆疊指標，並將變更後之堆疊指標所指向的行之左列起而取出配列號碼之處理。

接著，在步驟 S1104 中，係從儲存耦合節點樹之節點的配列中，將在步驟 S1103 中所取得的配列號碼所指向之配列要素作爲節點而讀取出。

而後，在步驟 S1105 中，從在步驟 S1104 所讀出之節點中，取出辨別位元位置。

於此，在步驟 S1103 中所取出之配列號碼，係必定爲在圖 8 之步驟 S813 中被推入至探索路徑堆疊中的配列號碼，而並非爲在圖 8 之步驟 S815 中被推入至探索路徑堆疊中之配列號碼。

接下來，在步驟 S1106 中，判定於步驟 S1105 中所取出之辨別位元位置，是否爲較在圖 9 之步驟 S903 中所得到之差分位元位置爲更上位之位置關係。

當辨別位元位置爲較差分位元位置更爲上位的情況時，則前進至步驟 S1107；當並非如此的情況時，則回到步驟 S1102。

在步驟 S1107 中，係從探索經路堆疊中，將探索路徑堆疊之堆疊指標所指向的子節點之配列號碼讀出。

例如，若是如圖 5（c）一般而將堆疊作圖示，則堆疊指標係代表行的位置，左列係代表偶數之辨別位元位置所包含的節點之配列號碼，右列係代表被寫入於相同行之左列的配列號碼之節點的子節點中，身爲非終端側節點的節

點〔1〕之配列號碼。

故而，若是依據圖 5 之（c）的圖示而對動作作說明，則在步驟 S1107 中，存在於堆疊指標所代表之行的右列處之配列號碼係被讀出。

接下來，在步驟 S1108 中，得到與在步驟 S1107 中所讀出之子節點的配列號碼成對之配列號碼。

在本實施形態中，由於構成節點對之 2 個的節點，係被儲存在相鄰接之配列號碼的配列要素之中，因此，構成節點對之 2 個的節點的配列號碼係亦彼此成對。例如，節點〔0〕係被儲存於偶數之配列號碼的配列要素中，而與該節點〔0〕成對之節點〔1〕，係被儲存於緊接在其之後的配列要素的配列號碼處。故而，偶數與奇數之配列號碼彼此亦係成對。

於此例之情況時，在步驟 S1108 中，藉由從在步驟 S1107 中所讀出之節點〔1〕的配列號碼減去 1，來得到與該節點〔1〕構成節點對的節點〔0〕之配列號碼。亦即是，在步驟 S1108 中，從非終端側節點之配列號碼，而取得身爲葉節點之終端側節點的配列號碼。

接著，在步驟 S1109 中，係從配列中，將在步驟 S1108 中所取得的配列號碼所指向之配列要素、亦即是將身爲終端側節點且身爲葉節點之上述節點〔0〕，作爲節點而讀取出。

接下來，在步驟 S1110 中，從在步驟 S1109 中所讀出的葉節點，而取出索引鍵，並在接下來的步驟 S1111 中，

將所取得之索引鍵設定爲最長一致節點，並結束圖 11 之處理。

接下來，對使用有圖 5 之耦合節點樹的圖 6～圖 9 以及圖 11 之處理的數個具體例作說明。

當元檢索鍵係爲圖 5 之（b）與圖 10 中所示的“10100”時，在圖 6 之步驟 S603 中，係得到在圖 5 中被顯示爲「初期檢索」的節點 211e 之索引鍵 251e。

於此情況，如同參考圖 10 而作了說明一般，比較位元長度係被設定爲 8 位元，而差分位元位置係爲 7。故而，圖 11 之步驟 S1102～步驟 S1106 係被實行，並從配列號碼 221c 之節點 210d 而取出辨別位元位置 230d。

由於辨別位元位置 230d 之值係爲 6，而爲較 7 更上位，因此，步驟 S1107～步驟 S1111 係被實行，於圖 5 中被展示爲「最長一致」之節點 210e 的索引鍵 250e，係被設定爲最長一致鍵。此係爲該當於最長一致鍵之定義（b）的情況。

當元檢索鍵係爲“1100＊”時，在圖 6 之步驟 S603 中，係得到索引鍵 250f。於此情況，元檢索鍵之最上位的隨意位元之位置係爲 4，元索引鍵之最下位的有意位元之位置係爲 3，因此比較位元長度係爲 8。

而，在圖 11 之步驟 S1101 中，係判定檢索鍵與索引鍵爲相等，並前進至步驟 S1111，索引鍵 250f 被設定爲最長一致鍵。於此例中，係爲該當於最長一致鍵之定義（a）的情況。

當元檢索鍵係爲“11001”時，在圖 6 之步驟 S603 中，係得到將元索引鍵“1100＊”編碼後之索引鍵 250f。

於此情況，元檢索鍵之最上位的隨意位元之位置，係可視爲 5，元索引鍵之最下位的有意位元之位置係爲 3，因此，在圖 9 之步驟 S901 中，比較位元長度係被設定爲 8。

故而，在圖 11 之步驟 S1101 中，係判定檢索鍵與索引鍵爲相等，並前進至步驟 S1111，索引鍵 250f 被設定爲最長一致鍵。於此例中，亦係爲該當於最長一致鍵之定義（b）的情況。

當元檢索鍵係爲“11＊＊＊”時，在圖 6 之步驟 S603 中，係得到將元索引鍵“1100＊”編碼後之索引鍵 250f。

於此情況，元檢索鍵之最上位的隨意位元之位置係爲 2，元索引鍵之最下位的有意位元之位置係爲 3，因此，在圖 9 之步驟 S901 中，比較位元長度係被設定爲 6。差分位元位置係爲 4。

故而，在圖 11 中，係從步驟 S1101 而前進至步驟 S1102。而後，步驟 S1102～步驟 S1106 係被實行，並從配列號碼（220a＋1）之節點 211b 而取出辨別位元位置 231b。辨別位元位置 231b 之值係爲 2，而爲較 4 更上位。

故而，步驟 S1107～步驟 S1111 係被實行，節點 210c 之將“1＊＊＊＊”編碼後的索引鍵 250c，係被設定爲最長一致鍵。於此例中，亦係爲該當於最長一致鍵之定義（b）的情況。

當元檢索鍵係爲“0＊＊＊＊＊”時，在圖 6 之步驟 S603 中，係得到將“1＊＊＊＊＊”編碼後之索引鍵 250c。於此情況，元檢索鍵之最上位的隨意位元之位置係爲 1，元索引鍵之最下位的有意位元之位置係爲 0，因此，在圖 9 之步驟 S901 中，比較位元長度係被設定爲 2，而差分位元位置係爲 1。

故而，圖 11 之步驟 S1102～步驟 S1106 係被實行，並從配列號碼 220 之根節點 210a 而取出辨別位元位置 230a。辨別位元位置 230a 之值係爲 0，而爲較 1 更上位。

故而，步驟 S1107～步驟 S1111 係被實行，節點 210b 之將“＊＊＊＊＊”編碼後的索引鍵 250b，係被設定爲最長一致鍵。於此例中，亦係爲該當於最長一致鍵之定義（b）的情況。

如以上所說明一般，若藉由上述之檢索方法，則就算是在元檢索鍵與元索引鍵之雙方中均存在有隨意位元，亦可進行檢索，又，不需要進行檢索與選擇之 2 階段的處理，便可取得被唯一性定義的最長一致鍵。

接下來，參考圖 12～圖 13C 之流程圖，針對在使用有編碼位元列之檢索用的耦合節點樹中，依據元插入鍵之指定，而插入葉節點的處理作說明。另外，由於藉由根節點之插入處理、以及將根節點以外之節點插入至既存的耦合節點樹中之通常的插入處理，耦合節點樹係被產生，因此，節點之插入處理的說明，係亦爲耦合節點樹之生成處理的說明。

在圖 12 之步驟 S1201 中，從所指定的元插入鍵，經由圖 7 之編碼處理而作成插入鍵。

接下來，在步驟 S1202 中，判定被要求作取得之耦合節點樹的根節點之配列號碼是否爲已登錄。在步驟 S1202 中之判定結果若是爲已完成登錄，則係移動至步驟 S1203。

在步驟 S1203 中，係將根節點設定爲檢索開始節點。

接下來，在步驟 S1204 中，從檢索開始節點而開始，並藉由插入鍵來對儲存耦合節點樹之配列進行檢索，再將插入鍵作爲索引鍵而插入（亦即是，將以插入鍵作爲索引鍵而包含的葉節點，插入至耦合節點樹中），而結束圖 12 之處理。步驟 S1204 之處理的詳細內容，係於後參考圖 13A～圖 13C 而作說明。

另一方面，在步驟 S1202 中之判定，若是係爲未登錄，則係成爲開始全新之耦合節點樹的登錄、產生。

於此情況，處理係前進至步驟 S1205。在步驟 S1205 中，從配列中求取空的節點對，並取得於該節點對中應成爲代表節點的配列要素之配列號碼。

接下來，在步驟 S1206 中，求取出在以步驟 S1205 所得到之配列號碼中加上 0 的配列號碼（在本實施形態中，由於係可在此得到與在步驟 S1205 中所取得的配列號碼相等之配列號碼，因此，步驟 S1206 係可作省略）。

進而，在步驟 S1207 中，作爲所插入之根節點，在以步驟 S1206 所得到之配列號碼的配列要素之節點種類別中

寫入1（葉節點），並在索引鍵中寫入插入鍵。

而後，在步驟S1208中，將在步驟S 1206中所取得之配列號碼，作爲根節點之配列號碼而登錄，並結束圖12之處理。

接下來，參考圖13A～圖13C，對圖12之步驟S1204的處理作詳細說明。圖13A，係爲展示身爲插入處理之前段的檢索處理之處理流程的圖，而大略相當於在圖4所示之檢索處理中將插入鍵作爲檢索鍵者。

步驟1301，係相當於在圖4之步驟S401處而將根節點設爲檢索開始節點者。又，步驟S1302～步驟S1310之處理，係完全對應於圖4之步驟S402～步驟S410。故而，係省略此些之步驟的說明。

如同從圖4與圖13A間之比較而可得知一般，不論索引鍵或是插入鍵是否被編碼，均經由相同之處理，而進行從檢索開始節點起直到葉節點爲止的檢索。

另外，請注意，在圖13A中，探索路徑堆疊之使用法，係與圖4相同，而與圖8或圖11者爲相異。在本實施形態之圖13的處理中，堆疊指標，係指向對在步驟S1302之推入動作中所應推入之配列號碼作儲存的探索路徑上之位置。

在圖13A之步驟S1311中，將插入鍵與索引鍵作比較，若是相等，則由於插入鍵係已存在於耦合節點樹中，因此插入係成爲失敗，而結束處理。而若是不相等，則進行接下來之處理，亦即是前進至圖13B之步驟S1312以下的

處理。

圖 13B，係爲用以說明對欲插入之節點對所需要的配列要素作準備之處理的處理流程圖。

在步驟 S1312 中，從配列中求取空的節點對，並取得於該節點對中應成爲代表節點的配列要素之配列號碼。

前進至步驟 S1313，將插入鍵與在步驟 S1310 中所得到之索引鍵的大小作比較，當插入鍵爲較大的情況時，得到值 1 之布林值，當較小時，則得到值 0 的布林值。

前進至步驟 S1314，在以步驟 S1312 所得到的代表節點之配列號碼上，加算上以步驟 S1313 所得到之布林值，而得到配列號碼。

前進至步驟 S1315，而得到在以步驟 S1312 所得到的代表節點之配列號碼上加算上以步驟 S1313 所得到之布林值的邏輯否定值後的配列號碼。

在步驟 S1314 所得之配列號碼，係爲儲存有將插入鍵作爲索引鍵而持有之葉節點的配列要素之配列號碼，在步驟 S1315 所得到之配列號碼，則係爲與該葉節點成對之節點所被儲存之配列要素者。

亦即是，藉由被儲存於以圖 13A 所示之前段之檢索處理所得到之葉節點中的索引鍵與插入鍵之大小，來決定在所插入之節點對中的何者之節點處，儲存保持有插入鍵之葉節點。此係藉由圖 13B 之處理來決定。

例如，當在圖 5（a）之耦合節點樹中，插入作爲索引鍵而包含有對元插入鍵“1101＊”作了編碼後的插入鍵

“1111101100”之葉節點的情況時，經由圖 13A 之處理，在探索路徑堆疊中，係被儲存有配列號碼 220、（220a＋1）、（221b＋1）、（220c＋1）、221d。又，在步驟 S1310 中，配列號碼 221d 之節點 210f 的內容爲“1111101000”之索引鍵 250f 係被取出。

故而，在步驟 S1313 中，插入鍵與索引鍵係被作比較，而由於插入鍵係爲較大，因此係得到布林值 1，並在將所插入之節點對的代表節點號碼上加上 1 後之配列要素中，儲存保持有插入鍵之葉節點。

另一方面，當在圖 5（a）之耦合節點樹中，插入作爲索引鍵而包含有對元插入鍵“0＊＊＊＊”作了編碼後的插入鍵“1000000000”之葉節點的情況時，經由圖 13A 之處理，在探索路徑堆疊中，係被儲存有配列號碼 220、（220a＋1）、221b。又，在步驟 S1310 中，節點 210c 之內容爲“1100000000”的索引鍵 250c 係被取出。

故而，在步驟 S1313 中，插入鍵與索引鍵係被作比較，而由於插入鍵係爲較小，因此係得到布林值 0，並在將所插入之節點對的代表節點號碼上加上 0 後之配列要素中，儲存保持有插入鍵之葉節點。

若是回到流程圖之說明，則接在圖 13B 之步驟 S1315 之後，圖 13C 之處理係被實行。圖 13C 係爲展示：在將節點儲存於在圖 13B 所準備之配列要素中的同時，求取其插入位置，並改變既存之節點的內容，而完成插入處理的處理流程之圖。

步驟 S1316～步驟 S1323 之處理，係爲求取出所插入之節點對的在耦合節點樹上之位置的處理，而步驟 S1324 以下之處理，係爲在各節點中設定資料並完成插入處理的處理。

在步驟 S1316 中，將插入鍵與在步驟 S1310 中所得到之索引鍵的位元列，以例如排他之邏輯和來進行比較，並得到差分位元列。

前進至步驟 S1317，從在步驟 S1316 中所得到的差分位元列，而得到從上位之第 0 位元起所看到之最初的不一致位元之位元位置。此處理，例如在具備有優先編碼器（priority encoder）的 CPU 中，係於該處輸入差分位元列，而可得到不一致之位元位置。又，亦可軟體性的來進行與優先編碼器同等之處理，來得到最初的不一致位元之位元位置。

接下來，前進至步驟 S1318，並判定探索經路堆疊之堆疊指標是否指向根節點之配列號碼。若是係指向該處，則移行至步驟 S1324，若是未指向該處，則前進至步驟 S1319。

在步驟 S1319 中，係將探索經路之堆疊指標倒退一個，並取出被堆疊於該處之配列號碼。

前進至步驟 S1320，將在步驟 S1319 中所取出的配列號碼之配列要素，從配列中作爲節點而讀取出。

前進至步驟 S1321，從在步驟 S1320 所讀取出之節點中，取出辨別位元位置。

接下來，前進至步驟 S1322，判定在步驟 S1321 中所取出之辨別位元位置是否爲較在步驟 S1317 中所得到之位元位置更上位之位置關係。於此，所謂的上位之位置關係，係指位元列之靠左側的位置，亦即是位元位置之値爲小的位置。

若是步驟 S1322 之判定結果係爲否定，則回到步驟 S1318，並反覆進行處理，直到在步驟 S1318 之判定成爲肯定，或是在步驟 S1322 之判定成爲肯定爲止。若是在步驟 S1322 之判定成爲肯定，則在步驟 S1323 中，使探索路徑堆疊之指標前進 1 個，並移行至步驟 S1324 以下之處理。

在上述步驟 S1316～步驟 S1323 中所說明之處理，係爲了決定所插入之節點對的插入位置，而對插入鍵與藉由圖 13A 之檢索所取得之索引鍵之間，進行位元列比較，並對在位元列比較中成爲相異之位元値的前端之（最上位之）位元位置與被儲存於探索路徑堆疊中之分支節點的辨別位元位置間之相對的位置關係作調查，而將辨別位元位置成爲上位之分支節點的下一個分支節點之連結目標，作爲所插入之節點對的插入位置。

例如，當元插入鍵係爲“1101＊”的上述之例的情況時，索引鍵“1111101000”與插入鍵“1111101100”，係在第 7 位元成爲相異。又，藉由第 1 次之步驟 S1319 的實行而取出的配列號碼（221c＋1）所指向的節點 211d 之辨別位元位置 231d，係爲 5，而爲較 7 更上位。

故而，藉由步驟 S1323，堆疊指標係前進 1 個，而堆疊指標，係成爲指向儲存節點 210f 之配列號碼 221d 的探索路徑堆疊上之場所。

又，當元插入鍵係爲“0＊＊＊＊”的上述之例的情況時，索引鍵“1100000000”與插入鍵“1000000000”，係在第 1 位元成爲相異。

故而，在此例中，係藉由步驟 S1318～步驟 S1322 的反覆進行，而對被儲存在堆積於探索路徑堆疊中之配列號碼的配列要素中之分支節點的辨別位元位置、和身爲「1」之位元位置，其兩者間之位置關係作檢查，同時，依序對探索路徑堆疊以逆方向作遍歷，直到使辨別位元位置成爲上位爲止。

其結果，在步驟 S1322 中，係成爲指向辨別位元位置 230a 之值係爲 0 的根節點 210a 之配列號碼 220 的狀態。而後，在步驟 S1323 中，堆疊指標係前進 1 個，而堆疊指標，係成爲指向儲存節點 211b 之配列號碼（220a＋1）的探索路徑堆疊上之場所。

又，若是就算是以反方向來對探索經路堆疊作遍歷並到達根節點，該根節點之辨別位元位置，亦並非爲較在先前所求取之位元列比較中成爲相異位元值之最上位的位元位置更上位之位元位置，則係表示：在該耦合節點樹之索引鍵的上位位元中，較根節點之辨別位元位置爲更上位的上位之位元的值係全部爲相等。而，係爲在插入鍵中第一次出現有在較根節點之辨別位元位置更上位的位元之值中

具有相異之位元值者。

故而，在此種情況中，所插入之節點對，係成爲根節點之直接的連結目標，根節點之辨別位元位置，係變更爲身爲與既存之索引鍵爲相異值之插入鍵的最上位位元之位置。

接下來，針對步驟 S1324 以下之在各節點中設定資料並完成插入處理的處理作說明。

在步驟 S1324 中，係從探索路徑堆疊中將指標所指向之配列號碼取出。

在步驟 S1325 中，在以步驟 S1314 所得到之配列號碼所指向的配列要素之節點種類別中寫入 1（葉節點），並在索引鍵中寫入插入鍵。

前進至步驟 S1326，將在步驟 S1324 中所取出的配列號碼之配列要素，從配列中讀取出。

接下來，在步驟 S1327 中，在以步驟 S1315 所得到之配列號碼的配列要素中，將藉由步驟 S1326 所讀出的內容寫入。

最後，在步驟 S1328 中，在以步驟 S1324 所得到之配列號碼所指向的配列要素之節點種類別中寫入 0（分支節點），並在辨別位元位置中寫入以步驟 S1317 所得到之位元位置，在代表節點號碼中寫入以步驟 S1312 所得到之配列號碼，而結束處理。

在元插入鍵係爲“1101＊”之上述例子的情況時，於步驟 S1325 中，將所取得之空節點對的節點〔1〕，設爲保

持插入鍵之葉節點。

另一方面，在節點〔0〕處，係於步驟 S1327 中，寫入節點 210f 之內容。而後，在步驟 S1328 中，在節點 210f 之辨別位元位置中儲存 7，而在代表節點號碼中，係儲存有所取得之節點對的代表節點所被儲存之配列要素的配列號碼。

又，在元插入鍵係爲"0＊＊＊＊"之上述例子的情況時，於步驟 S1325 中，將所取得之空節點對的節點〔0〕，設爲保持插入鍵之葉節點。

另一方面，在節點〔1〕處，係於步驟 S1327 中，寫入節點 211b 之內容。而後，在步驟 S1328 中，在節點 211b 之辨別位元位置中 231b 中儲存 1，而在代表節點號碼中，係儲存有所取得之節點對的代表節點所被儲存之配列要素的配列號碼。

如先前所述一般，明顯的，當存在有索引鍵之集合時，藉由從該處依序取出索引鍵，並反覆進行圖 12～圖 13C 之處理，而能夠建構對應於索引鍵之集合的本發明之耦合節點樹。

接下來，參考圖 14～圖 15B，針對從使用有編碼位元列之檢索用的耦合節點樹中，依據元削除鍵之指定，而將葉節點削除的處理作說明。

在圖 14 之步驟 S1401 中，從所指定的元削除鍵，經由圖 7 之編碼處理而作成削除鍵。接下來，在步驟 S1402 中，將耦合節點樹的根節點，設定爲檢索開始節點。而後

，在步驟 S1403 中，從檢索開始節點而開始，並藉由削除鍵來對儲存耦合節點樹之配列進行檢索，再將包含有與削除鍵相等之索引鍵的葉節點，從耦合節點樹中削除。

接下來，參考圖 15A 與圖 15B，對圖 14 之步驟 S1403 的處理作詳細說明。

圖 15A，係爲展示身爲削除處理之前段的檢索處理之處理流程的圖，而大略相當於在圖 4 所示之檢索處理中將削除鍵作爲檢索鍵者。

步驟 1501，係相當於在圖 4 之步驟 S401 處而將根節點設爲檢索開始節點者。又，步驟 S1502～步驟 S1510 之處理，係完全對應於圖 4 之步驟 S402～步驟 S410。故而，係省略此些之步驟的說明。

在圖 15A 之步驟 S1511 中，將削除鍵與索引鍵作比較，若是不相等，則由於欲削除之索引鍵鍵係不存在於耦合節點樹中，因此削除係成爲失敗，而結束處理。而若是相等，則進行接下來的處理，亦即是前進至圖 15B 之步驟 S1512 以下的處理。

圖 15B，係爲說明削除處理之後段的處理流程之圖。

首先，在步驟 S1512 中，判定在探索路徑堆疊中是否被儲存有 2 個以上的配列號碼。若是未被儲存有 2 個以上的配列號碼，換句話說，僅僅存在有 1 個，則該配列號碼係爲被儲存有根節點之配列要素。

於該情況，係移行至步驟 S1518，並將屬於以步驟 S1501 所得到之根節點的配列號碼之節點對削除。接下來

，前進至步驟 S1519，並將被登錄之根節點的配列號碼削除，而結束處理。

在步驟 S1512 中，若是被判定爲在探索路徑堆疊中係儲存有 2 個以上的配列號碼，則前進至步驟 S1513，並得到在以步驟 S1508 所得到之代表節點號碼中加算有將以步驟 S1507 所得到之位元值反轉後的值後之配列號碼。此處理，係爲求取與身爲削除對象之索引鍵所被儲存之葉節點成對之節點的所被配置之配列號碼者。

接下來，在步驟 S1514 中，讀取出以步驟 S1513 所得到之配列號碼的配列要素之內容，並在步驟 S1515 中，將探索路徑堆疊之堆疊指標倒退 1 個，而取出配列號碼。

接下來，前進至步驟 S1516，將在以步驟 S1514 所讀出之配列要素的內容，抹寫至以步驟 S1515 所得到的配列號碼之配列要素中。此處理，係爲將身爲對削除對象之索引鍵所被儲存之葉節點之連結源頭的分支節點，置換爲與上述葉節點成對之節點者。

最後，在步驟 S1517 中，將相關於以步驟 S1508 所得到之代表節點號碼的節點對作削除，並結束處理。

如以上所示一般，削除處理，係與插入處理的情況相同，不論索引鍵或是削除鍵是否被編碼，均經由與圖 4 相同之處理，而進行從檢索開始節點起直到葉節點爲止的檢索。另外，在削除處理中之探索路徑堆疊的使用法，係與圖 4 相同，而與圖 8 或圖 11 者爲相異。關於此點，圖 14～圖 15B 之削除處理，係與圖 12～圖 13C 之插入處理類

似。

於以下，以對於圖 5（a）之耦合節點樹，而指定元削除鍵“1011＊”並進行削除處理的情況爲例而作說明。

首先，在圖 14 之步驟 S1401 中，從元削除鍵“1011＊”來作成削除鍵“1110111100”，並在步驟 S1403 中，使用此削除鍵而進行圖 15A 與圖 15B 之處理。

在圖 15A 之處理中，最初係在步驟 S1501 中，得到根節點之配列號碼 220。

接下來，藉由圖 15A 之處理，在探索路徑堆疊中，係依序被推入有配列號碼 220、（220a＋1）、（221b＋1）、221c、（220d＋1）。而後，在圖 15A 之步驟 S1511 中，被儲存在配列號碼（220d＋1）中之節點 221e 的索引鍵 251e，係被與削除鍵作比較，而由於兩者係爲相等，故前進至圖 15B。

在圖 15B 之步驟 S1513 中，係得到配列號碼 220d，在步驟 S1514 中，係讀出被儲存在配列號碼 220d 中之節點 210e 的內容。

接下來，在步驟 S1515 中，配列號碼 221c 係被取出，在步驟 S1516 中，被儲存於配列號碼 221c 處的節點 210d 中，係被寫入有節點 210e 之內容，在步驟 S1517 中，藉由配列號碼 220d 所代表的節點對 201e 係被削除，並結束削除處理。

如同以上所說明一般，在上述之插入處理與削除處理中，係保持有耦合節點樹之長處，亦即是：受到影響的既

存之節點的範圍係被抑制在最小限度，而插入或削除所致之維護成本係爲低。又，藉由採用如同上述一般之編碼方法，在能夠保持有此長處的同時，亦使考慮有隨意位元之高速的最長一致檢索成爲可能。而，從在考慮有隨意位元之高速的最長一致檢索中所必須的成本之觀點來看，若是將上述之編碼方法與例如專利文獻 3 或是專利文獻 4 中所進行的繁雜之前處理相比較，則上述之編碼方法的成本係爲相當低。

以上。雖係對用以實施本發明之最佳實施形態作了說明，但是，本發明之實施形態係並不限定於此，而可作各種之變形。

在上述之實施形態中，係將對應於位元值 0 而被連結之節點〔0〕作爲代表節點，而將對應於位元值 1 而被連結之節點〔1〕作爲與代表節點成對之非代表節點。但是，亦可將節點〔1〕作爲代表節點，而將節點〔0〕作爲非代表節點。另外，關於可將根節點任意配置在代表節點之位置或者是非代表節點位置一事，參考圖 2 等之相關說明，便可明顯瞭解。

在上述之實施形態中，係在葉節點之索引鍵中，儲存有編碼位元列。但是，係能夠以將除此之外的形式之位元列作爲葉節點的索引鍵來儲存的方式，而對上述之實施形態作變形。

由於係存在有「在元位元列中，隨意位元係不會存在於較有意位元爲更前方」的前提條件，因此，例如，亦可

藉由「在元位元列中，僅有有意位元之部分不會被編碼」的形態，來作爲索引鍵而儲存。當元位元列係爲固定長度的情況時，則亦能夠從此種形勢之索引鍵中，來求取出隨意位元之位元數。

又，在此種形式之索引鍵的情況時，由於元位元列之長度係爲編碼位元列的一半以下，因此，係能夠將在葉節點中所需要的記憶容量削減至一半以下。換言之，係成爲能夠藉由相同之葉節點的記憶容量，來處理 2 倍以上之長度的位元列。不論何者，記憶區域之利用效率均會提昇。

另外，在利用此種形式之索引鍵的實施形態中，明顯的，係有必要對上述之實施形態作適當的變更，例如，在圖 6 之步驟 S604 中將索引鍵編碼，而後再與元檢索鍵作比較等。

或者是，亦可經由將在隨意位元中設定了任意之位元值的位元列、和代表有意位元之範圍的位元列（例如，遮罩位元列、或是代表數值之位元列等）作組合，而表現元位元列，並將該組合作爲葉節點之索引鍵來儲存。

又，在上述之實施形態中，未被編碼之元檢索鍵係作爲輸入而被給予，而檢索處理係包含有編碼處理。但是，當將預先被編碼了的檢索鍵作爲輸入而給予的情況時，則編碼處理係成爲不必要。在插入處理與削除處理中，亦爲相同。

另外，編碼處理，係亦可經由圖 7 所示之方法以外的方法來實現。例如，亦可採用：將輸入位元列以偏移暫存

器等來各作 1 位元之偏移，並一次 1 位元地從輸入位元列來取出，再將所取出之 1 位元與識別位元作連結的方法。

又，亦可採用除了上述之實施形態中所採用的編碼方法以外之編碼方法。在上述實施形態之編碼方法中，係將在元位元列中之 1 位元，編碼爲在編碼位元列中之 2 位元，但是，亦可採用此種具備有 1：2 之對應關係的編碼方法以外之編碼方法。

例如，亦可將在元位元列中之 1 位元，編碼爲在編碼位元列中之 3 位元。又，上述實施形態之編碼方法的識別位元，當值係爲“0”時，係代表隨意位元，而當值爲“1”時，係代表有意位元，但是，係亦可爲相反。

另外，檢索處理，係有必要配合於編碼方法而作適當的變更。例如，當採用識別位元之 0 與 1 的意義爲與上述實施形態相反之編碼方法的情況時，在圖 8 之步驟 S812 中，係有必要變更爲當位元值爲 0 時則前進至步驟 S813；當位元值爲 1 時則前進至步驟 S809，而在步驟 S814 中，係有必要將在代表節點號碼上所加上之值從 1 而變更爲 0。另一方面，插入處理與削除處理，係不隨編碼方法而改變，而可採用上述實施形態之方法。

又，在上述之圖 5 中，當將“0＊＊＊＊”作爲元檢索鍵時之最長一致鍵，係爲將“＊＊＊＊＊”編碼後之“0000000000”，但是，於此情況，最長一致鍵之元位元列“＊＊＊＊＊”所代表的集合，係包含有從“1”開始之位元列。但是，從“1”開始之位元列，與元檢索鍵“0＊＊＊＊”

，係在最上位之第 0 位元成爲相異。依存於檢索之目的，亦有欲將此種情況視爲檢索失敗來處理的情形。

於此，亦可考慮對其施加以下限制，亦即是：最長一致鍵之元位元列，至少第 0 位元係爲有意位元，而最長一致鍵之元位元列與元檢索鍵，係必須至少在第 0 位元的值成爲一致。爲了施加此種限制，亦可將圖 6 之處理如同下述一般的作變形。

亦即是，在步驟 S604 與步驟 S606 之間，追加判定所求取之差分位元位置是否爲第 0 位元的新的步驟 S605。

當差分位置爲第 0 位元時，則由於滿足上述之限制的最長一致鍵係並不存在於耦合節點樹中，因此檢索係失敗，而結束圖 6 之處理。另一方面，當在步驟 S605 中差分位元位置並非爲第 0 位元時，處理係前進至步驟 S606。

或者是，亦可將此變形例更進一步變形，在步驟 S605 中，若是差分位元位置爲 1 以下，則結束處理，而若是差分位元位置爲 2 以上，則處理係前進至步驟 S606。

又，當在檢索鍵之有意位元長度係保證恆常爲較索引鍵爲更長一般的應用領域中，利用本發明之檢索方法的情況時，係亦可省略圖 8 之步驟 S812，並在步驟 S811 中，當辨別位元位置爲偶數時，則前進至步驟 S813。

上述之檢索處理，係可因應於用途而作其他之各種變形。例如，在圖 6 中，雖係在步驟 S602 中，將根節點設定爲檢索開始節點，但是，亦可將任意之部分樹的根節點

設爲檢索開始節點。

又，在上述之各處理中，雖係利用有探索路徑堆疊，但是，探索路徑堆疊之實現方法，係爲任意。於圖 5 中，係爲了易於理解，而將探索路徑堆疊分爲 2 列來作表示，但是，此圖示，係並非爲用以對探索路徑堆疊之實現方法作限定者。

進而，在圖 8 之處理中，係爲了處理之效率化，而將辨別位元位置爲偶數之分支節點、與身爲其之子節點的節點〔1〕之配列號碼，儲存在探索路徑堆疊中。但是，亦可和插入處理或削除處理相同的，將從檢索開始節點起直到葉節點爲止的經過路徑上之所有節點的配列號碼儲存在探索路徑堆疊中。

於該情況，代替圖 11 之步驟 S1107 與步驟 S1108，亦可從探索路徑堆疊中讀出分支節點之配列號碼，並將該配列號碼之分支節點從配列中讀出，而取出代表節點號碼。所取出之代表節點號碼，由於係爲此分支節點的子節點中之身爲代表節點的節點〔0〕之配列號碼，因此，步驟 S1109 之後的流程，係可與圖 11 同樣地來實行。

或者是，在圖 8 中，雖係在步驟 S815 中，將以步驟 S814 所得到之身爲非終端側節點的節點〔1〕之配列號碼儲存在探索路徑堆疊中，但是，亦可將上述之實施形態變形爲：在步驟 S815 中，將身爲終端側節點的節點〔0〕之配列號碼儲存在探索路徑堆疊中。

但是，於此情況中，在步驟 S814 中得到了配列號碼

之節點〔1〕係在緊接於步驟 S815 之後的步驟 S803 處而被讀出，此點係與圖 8 相同。當進行此種變形的情況時，只要代替圖 11 之步驟 S1107 與步驟 S1108，而將節點〔0〕之配列號碼直接從探索路徑堆疊中讀出即可。

另外，明顯的，實施本發明之檢索方法、插入方法、又或是削除方法的裝置，係可藉由儲存耦合節點樹之記憶手段，與使電腦實行在各流程圖中所示之處理的程式，而構築在電腦上。故而，上述程式、以及記憶有程式而爲電腦可讀取之記憶媒體，係亦包含於本發明之實施形態內。

【圖式簡單說明】

〔圖 1〕說明被儲存於配列中之耦合節點樹的構成例之圖。

〔圖 2〕將耦合節點樹之樹狀構造作概念展示的圖。

〔圖 3〕說明用以實施本發明之硬體的構成例之圖。

〔圖 4〕展示在本發明之其中一種實施形態中的檢索處理之流程圖。

〔圖 5〕將利用有編碼位元列之耦合節點樹的樹狀構造作概念展示的圖。

〔圖 6〕展示在本發明之其中一種實施形態中的檢索處理之流程圖。

〔圖 7〕展示在本發明之其中一種實施形態中的編碼處理之流程圖。

〔圖 8〕在檢索處理中之初期檢索之流程圖。

〔圖 9〕求取差分位元位置之處理的流程圖。

〔圖 10〕展示差分位元位置之例的圖。

〔圖 11〕在檢索處理中求取最長一致鍵之處理的流程圖。

〔圖 12〕展示在本發明之其中一種實施形態中的插入處理之流程圖。

〔圖 13A〕身爲插入處理之前段的檢索處理之流程圖。

〔圖 13B〕說明將插入之節點對所需要的配列要素作準備之處理的流程圖。

〔圖 13C〕展示求取出插入節點對之位置，並將節點對之各節點的內容作寫入，而完成插入處理之流程圖。

〔圖 14〕展示在本發明之其中一種實施形態中的削除處理之流程圖。

〔圖 15A〕展示身爲削除處理之前段的檢索處理之流程圖。

〔圖 15B〕展示削除處理之後段的流程圖。

〔圖 16〕展示在先前之檢索中所使用的帕翠西雅樹之其中一例的圖。

【主要元件符號說明】

10、20、30：配列號碼

100：配列

101：節點

102、114、117、124、126、260a～260h、261b～

261h：節點種類別

103、115、230a～230f、231b～231f：辨別位元位置

104、116、220、220a～220f、221b～221f：代表節點號碼

111、121、201a～201h：節點對

112、122、210a～210h：節點〔0〕、代表節點

113、123、211b～211h：節點〔1〕、與代表節點成對之非代表節點

118、250c～250h、251d～251h：索引鍵

270：檢索鍵

301：資料處理裝置

302：中央處理裝置

303：快取記憶體

304：匯流排

305：主記憶裝置

306：外部記憶裝置

307：通訊裝置

308：資料儲存裝置

309：配列

310：探索經路堆疊

1730a～1730h：檢查位元位置

1740a～1740h：左連結、左指標

1741a～1741h：右連結、右指標

1750a～1750h：節點

五、中文發明摘要

發明之名稱：位元列檢索方法及程式產品

【課題】提供一種適合於處理隨意位元的檢索方法。

【解決手段】耦合節點樹，係由根節點，和由被配置於鄰接之記憶區域的分支節點與葉節點；又或是由分支節點彼此又或是由葉節點彼此所成的節點對所構成。分支節點，係包含有以對隨意位元與有意位元作識別的方式而被編碼之用以進行位元列檢索的檢索鍵之辨別位元位置、和代表身爲連結目標之節點對的其中一方之節點的代表節點之位置的位置資訊。葉節點，係包含有由被編碼後／未被編碼之狀態的位元列所成之索引鍵。在分支節點中，因應於辨別位元位置之檢索鍵的位元值，來對連結目標之節點對的其中一方之節點作連結，並反覆進行此，直到到達葉節點爲止，並因應於需要，而對到達葉節點爲止之經過路徑作回遡，藉由此，而進行考慮有隨意位元之檢索。

六、英文發明摘要

發明之名稱：

十、申請專利範圍

1. 一種位元列檢索方法，係爲使用有耦合節點樹之位元列檢索方法，該耦合節點樹，係由根節點（root node）、與由被配置於相鄰接之記憶區域中的分支節點（branch node）與葉節點（leaf node）、又或是分支節點彼此又或是葉節點彼此之節點對所成的使用於位元列檢索之樹（tree），並具有以下構成：

前述根節點，係爲表示樹之起點的節點，當該樹之節點係爲 1 個時，係身爲前述葉節點，而當該樹之節點係爲 2 個以上時，則係爲前述分支節點，

前述分支節點，係包含有用以進行位元列檢索之檢索鍵的辨別位元位置、以及代表身爲連接（link）目標之節點對的其中一方之節點的代表節點之位置之位置資訊，

前述葉節點，係包含有由檢索對象之位元列所成的索引鍵（index key），

該位元列檢索方法，其特徵爲：

前述索引鍵，係由將前方有意位元列作編碼後之編碼位元列所成，該前方有意位元列，係爲僅由有意位元所成之位元列、或是僅由任意位元值之隨意位元（don't care bit）所成之位元列、又或是在 1 位元以上之有意位元處連續有 1 位元以上之隨意位元的位元列中之任一者，

前述檢索鍵，係爲由將前方有意位元列編碼後之編碼位元列所成，

前述編碼位元列，係爲對應於構成前述前方有意位元

列之各位元的位元對，該位元對，係由表現該位元係爲隨意位元亦或是有意位元一事之識別位元；和當該位元係爲隨意位元時則成爲預先被決定之值、當該位元係爲有意位元時則爲代表該位元之值的資料位元所成，

前述位元列檢索方法，係具備有：

初期檢索步驟，係將前述耦合節點樹之任意的部分樹之根節點作爲檢索開始節點，在前述分支節點中，因應於被包含在該分支節點中之辨別位元位置之前述檢索鍵的位元值，而對連結目標之節點對的代表節點或者是與該代表節點成對之非代表節點作連結，並藉由在前述編碼位元列中之識別位元所存在的位置處至少儲存前述辨別位元位置所相當之分支節點的位址資訊，而一面對作了連結並遍歷（traverse）了的路徑作記憶，一面依序反覆進行前述連結，直到到達前述葉節點爲止；和

索引鍵取得步驟，係從藉由前述初期檢索步驟而到達了的前述葉節點處，而取得索引鍵；和

差分位元位置取得步驟，係在前述所取得之前述索引鍵與前述檢索鍵之間，對於從位元列之前端起，直到在前述所取得之前述索引鍵處的將末尾端之有意位元編碼後之位元對的位置、和在前述檢索鍵中，將前端之隨意位元作了編碼後之位元對的位置的兩者中，較接近於前述編碼位元列之前端的位元位置者爲止的範圍內，進行位元列比較，並將成爲相異之位元值的前端之位元位置，作爲差分位元位置而取得；和

第 1 最長一致鍵取得步驟，若是前述所取得之前述索引鍵與前述檢索鍵，在前述範圍中係爲一致，則將前述所取得之前述索引鍵，作爲最長一致鍵而取得；和

分支節點選擇步驟，係當前述比較之結果，前述所取得之前述索引鍵與前述檢索鍵並非爲一致的情況時，將辨別位元位置爲相當於在前述編碼位元列中的識別位元所存在之位置，且該辨別位元位置爲相較於前述差分位元位置而更接近於前述編碼位元列之前端的位元位置之前述路徑上之分支節點中，前述辨別位元位置爲最爲接近前述編碼位元列之尾端的位元位置之分支節點，作爲分支節點而選擇；和

第 2 最長一致鍵取得步驟，係將從前述所選擇之前述分支節點而連結之節點對中，身爲對應於代表前述隨意位元之前述識別位元之值而被連結的葉節點之終端側節點中所包含的索引鍵，作爲最長一致鍵而取得。

2. 如申請專利範圍第 1 項所記載之位元列檢索方法，其中，前述葉節點之前述索引鍵，係代替前述編碼位元列，而由編碼前之前述前方有意位元列所成，在前述差分位元位置取得步驟以及前述第 1 最長一致鍵取得步驟中，代替前述所取得之前述索引鍵，係將把前述所取得之前述索引鍵編碼後的編碼位元列與前述檢索鍵作比較。

3. 如申請專利範圍第 1 項所記載之位元列檢索方法，其中，在前述初期檢索步驟中，當將前述路徑作記憶時，除了前述分支節點之前述位置資訊以外，亦將代表從該

分支節點而被連結之節點對之中，身爲對應於代表前述有意位元之前述識別位元之值而被連結的節點之非終端側節點的位置之位置資訊作記憶。

4. 如申請專利範圍第 3 項所記載之位元列檢索方法，其中，在前述第 2 最長一致鍵取得步驟中，係從所記憶之前述非終端側節點的前述位置資訊，來求取出與該非終端側節點成爲節點對之前述終端側節點的位置資訊。

5. 如申請專利範圍第 1 項所記載之位元列檢索方法，其中，前述路徑，係被儲存在堆疊中。

6. 如申請專利範圍第 1 項所記載之位元列檢索方法，其中，前述耦合節點樹，係被記憶於配列中，被包含於前述分支節點中之前述位置資訊，係爲對應於該位置資訊之前述代表節點所被儲存之前述配列的配列要素之配列號碼。

7. 一種插入方法，係爲在耦合節點樹中，將包含有由所期望之位元列所成之索引鍵的葉節點作插入之插入方法，該耦合節點樹，係由根節點（root node）、與由被配置於相鄰接之記憶區域中的分支節點（branch node）與葉節點（leaf node）、又或是分支節點彼此又或是葉節點彼此而成之節點對所成的使用於位元列檢索之樹（tree），並具有以下構成：

前述根節點，係爲表示樹之起點的節點，當該樹之節點係爲 1 個時，係身爲前述葉節點，而當該樹之節點係爲 2 個以上時，則係爲前述分支節點，

前述分支節點，係包含有用以進行位元列檢索之檢索鍵的辨別位元位置、以及代表身爲連接（link）目標之節點對的其中一方之節點的代表節點之位置之位置資訊，

前述葉節點，係包含有由檢索對象之位元列所成的索引鍵（index key），

該插入方法，其特徵爲：

前述索引鍵，係由將前方有意位元列作編碼後之編碼位元列所成，該前方有意位元列，係爲僅由有意位元所成之位元列、或是僅由任意位元值之隨意位元（don't care bit）所成之位元列、又或是在1位元以上之有意位元處連續有1位元以上之隨意位元的位元列中之任一者，

前述檢索鍵，係爲由將前方有意位元列編碼後之編碼位元列所成，

前述編碼位元列，係爲對應於構成前述前方有意位元列之各位元的位元對，該位元對，係由表現該位元係爲隨意位元亦或是有意位元一事之識別位元；和當該位元係爲隨意位元時則成爲預先被決定之值、當該位元係爲有意位元時則爲代表該位元之值的資料位元所成，

身爲編碼位元列之前述所期望的位元列，係以編碼位元列又或是被編碼前之前方有意位元列的形式而被指定，

前述插入方法，係：

當經由前述編碼位元列而被指定有前述所期望之位元列的情況時，將被指定之該編碼位元列作爲檢索鍵；

當經由前述前方有意位元列而被指定有前述所期望之

位元列的情況時，將被指定之前述前方有意位元列作編碼，而取得編碼位元列，並將所取得之該編碼位元列作爲檢索鍵；

在前述分支節點中，因應於被包含在該分支節點中之辨別位元位置之前述檢索鍵的位元值，而對連結目標之節點對的代表節點或者是與該代表節點成對之非代表節點作連結，並從前述根節點起而一面將路徑作記憶一面依序反覆進行前述連結，直到到達前述葉節點爲止；

在前述葉節點之索引鍵與前述檢索鍵之間，進行大小比較與位元列比較；

藉由身爲在前述位元列比較中而成爲相異之位元值的前端之位元位置的差分位元位置，其與前述路徑上之分支節點的辨別位元位置間之相對的位置關係，來決定由被插入之前述葉節點與另外一方之節點所成的節點對之插入位置；

藉由前述大小關係，來決定將包含有由前述所期望之位元列所成的索引鍵之前述葉節點作爲被插入之前述節點對中的何者之節點。

8. 如申請專利範圍第 7 項所記載之插入方法，其中，前述葉節點之前述索引鍵，係代替前述編碼位元列，而由前述前方有意位元列所成，並代替在前述大小比較與前述位元列比較之前便將前述索引鍵編碼並取得編碼位元列，再在前述葉節點之索引鍵與前述檢索鍵之間進行前述大小比較與前述位元列比較，而在所取得之前述編碼位元列

與前述檢索鍵之間，進行前述大小比較與前述位元列比較。

9. 如申請專利範圍第 7 項所記載之插入方法，其中，前述耦合節點樹，係被記憶於配列中，被包含於前述分支節點中之前述位置資訊，係爲對應於該位置資訊之前述代表節點所被儲存之前述配列的配列要素之配列號碼。

10. 一種削除方法，係爲在耦合節點樹中，將包含有由所期望之位元列所成之索引鍵的葉節點作削除之削除方法，該耦合節點樹，係由根節點（root node）、與由被配置於相鄰接之記憶區域中的分支節點（branch node）與葉節點（leaf node）、又或是分支節點彼此又或是葉節點彼此而成之節點對所成的使用於位元列檢索之樹（tree），並具有以下構成：

前述根節點，係爲表示樹之起點的節點，當該樹之節點係爲 1 個時，係身爲前述葉節點，而當該樹之節點係爲 2 個以上時，則係爲前述分支節點，

前述分支節點，係包含有用以進行位元列檢索之檢索鍵的辨別位元位置、以及代表身爲連接（link）目標之節點對的其中一方之節點的代表節點之位置之位置資訊，

前述葉節點，係包含有由檢索對象之位元列所成的索引鍵（index key），

該削除方法，其特徵爲：

前述索引鍵，係由將前方有意位元列作編碼後之編碼位元列所成，該前方有意位元列，係爲僅由有意位元所成

之位元列、或是僅由任意位元值之隨意位元（don't care bit）所成之位元列、又或是在1位元以上之有意位元處連續有1位元以上之隨意位元的位元列中之任一者，

前述檢索鍵，係爲由將前方有意位元列編碼後之編碼位元列所成，

前述編碼位元列，係爲對應於構成前述前方有意位元列之各位元的位元對，該位元對，係由表現該位元係爲隨意位元亦或是有意位元一事之識別位元；和當該位元係爲隨意位元時則成爲預先被決定之值、當該位元係爲有意位元時則爲代表該位元之值的資料位元所成，

身爲編碼位元列之前述所期望的位元列，係以編碼位元列又或是被編碼前之前方有意位元列的形式而被指定，

前述削除方法，係：

當經由前述編碼位元列而被指定有前述所期望之位元列的情況時，將被指定之該編碼位元列作爲檢索鍵；

當經由前述前方有意位元列而被指定有前述所期望之位元列的情況時，將被指定之前述前方有意位元列作編碼，而取得編碼位元列，並將所取得之該編碼位元列作爲檢索鍵；

在前述分支節點中，因應於被包含在該分支節點中之辨別位元位置之前述檢索鍵的位元值，而對連結目標之節點對的代表節點或者是與該代表節點成對之非代表節點作連結，並從前述根節點起依序反覆進行前述連結，直到到達前述葉節點爲止；

將與前述葉節點成爲節點對之節點的內容，儲存在該當節點對之連結源頭的分支節點中；

將該當節點對削除。

11. 如申請專利範圍第 10 項所記載之削除方法，其中，前述葉節點之前述索引鍵，係代替前述編碼位元列，而由前述前方有意位元列所成。

12. 如申請專利範圍第 10 項所記載之削除方法，其中，前述耦合節點樹，係被記憶於配列中，被包含於前述分支節點中之前述位置資訊，係爲對應於該位置資訊之前述代表節點所被儲存之前述配列的配列要素之配列號碼。

13. 一種程式產品，其特徵爲：係用以在電腦中實行如申請專利範圍第 1～6 項中之任一項所記載之位元列檢索方法。

14. 一種程式產品，其特徵爲：係用以在電腦中實行如申請專利範圍第 7～9 項中之任一項所記載之插入方法。

15. 一種程式產品，其特徵爲：係用以在電腦中實行如申請專利範圍第 10～12 項中之任一項所記載之削除方法。

16. 一種資料構造，係爲使用於位元列檢索中之樹狀的資料構造，其特徵爲：

該資料構造，係爲由根節點（root node）、與由被配置於相鄰接之記憶區域中的分支節點（branch node）與葉節點（leaf node）、又或是分支節點彼此又或是葉節點彼

此而成之節點對所成的使用於位元列檢索之樹（tree），並具有以下構成：

前述根節點，係爲表示樹之起點的節點，當該樹之節點係爲 1 個時，係身爲前述葉節點，而當該樹之節點係爲 2 個以上時，則係爲前述分支節點，

前述分支節點，係包含有用以進行位元列檢索之檢索鍵的辨別位元位置、以及代表身爲連接（link）目標之節點對的其中一方之節點的代表節點之位置之位置資訊，

前述葉節點，係包含有由檢索對象之位元列所成的索引鍵（index key），

前述索引鍵，係由前方有意位元列或將該前方有意位元列作編碼後之編碼位元列所成，該前方有意位元列，係爲僅由有意位元所成之位元列、或是僅由任意位元值之隨意位元（don't care bit）所成之位元列、又或是在 1 位元以上之有意位元處連續有 1 位元以上之隨意位元的位元列中之任一者，

前述檢索鍵，係爲由將前方有意位元列編碼後之編碼位元列所成，

前述編碼位元列，係爲對應於構成前述前方有意位元列之各位元的位元對，該位元對，係由表現該位元係爲隨意位元亦或是有意位元一事之識別位元、和當該位元係爲隨意位元時則成爲預先被決定之值、當該位元係爲有意位元時則爲代表該位元之值的資料位元所成，

將任意之節點作爲檢索開始節點，並在前述分支節點

中，因應於被包含在該分支節點中之辨別位元位置之前述檢索鍵的位元值，而對連結目標之節點對的代表節點或者是與該代表節點成對之非代表節點作連結，並依序反覆進行前述連結，直到到達前述葉節點爲止，藉由此，而成爲能夠實行前述檢索鍵所致之檢索。

17. 如申請專利範圍第 16 項所記載之資料構造，其中，前述資料構造，係被記憶於配列中，被包含於前述分支節點中之前述位置資訊，係爲對應於該位置資訊之前述代表節點所被儲存之前述配列的配列要素之配列號碼。

200910127

97124937

771069

圖1

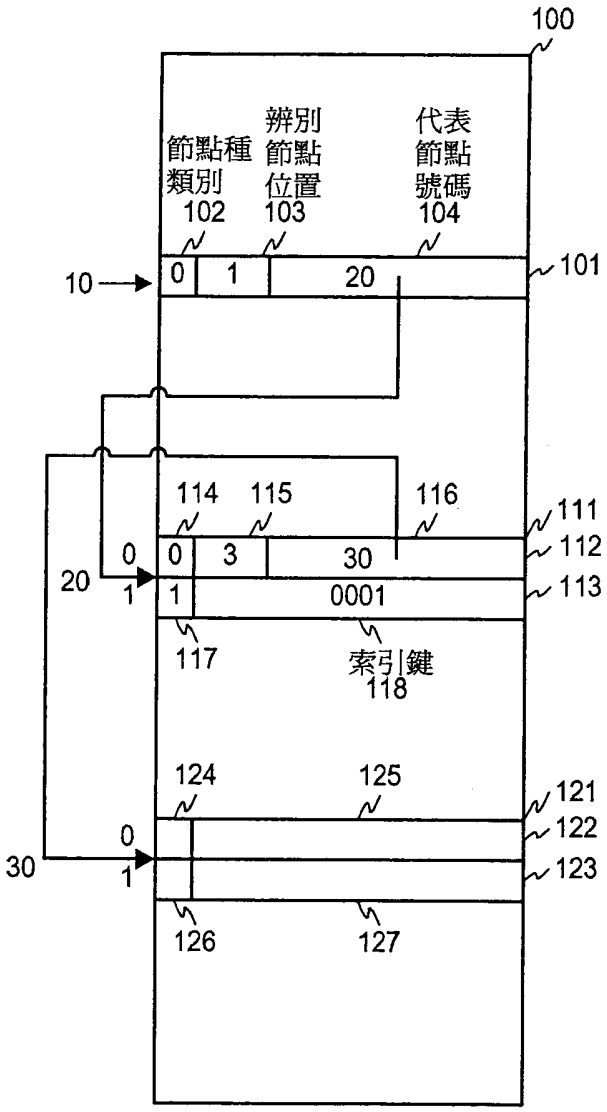


圖2

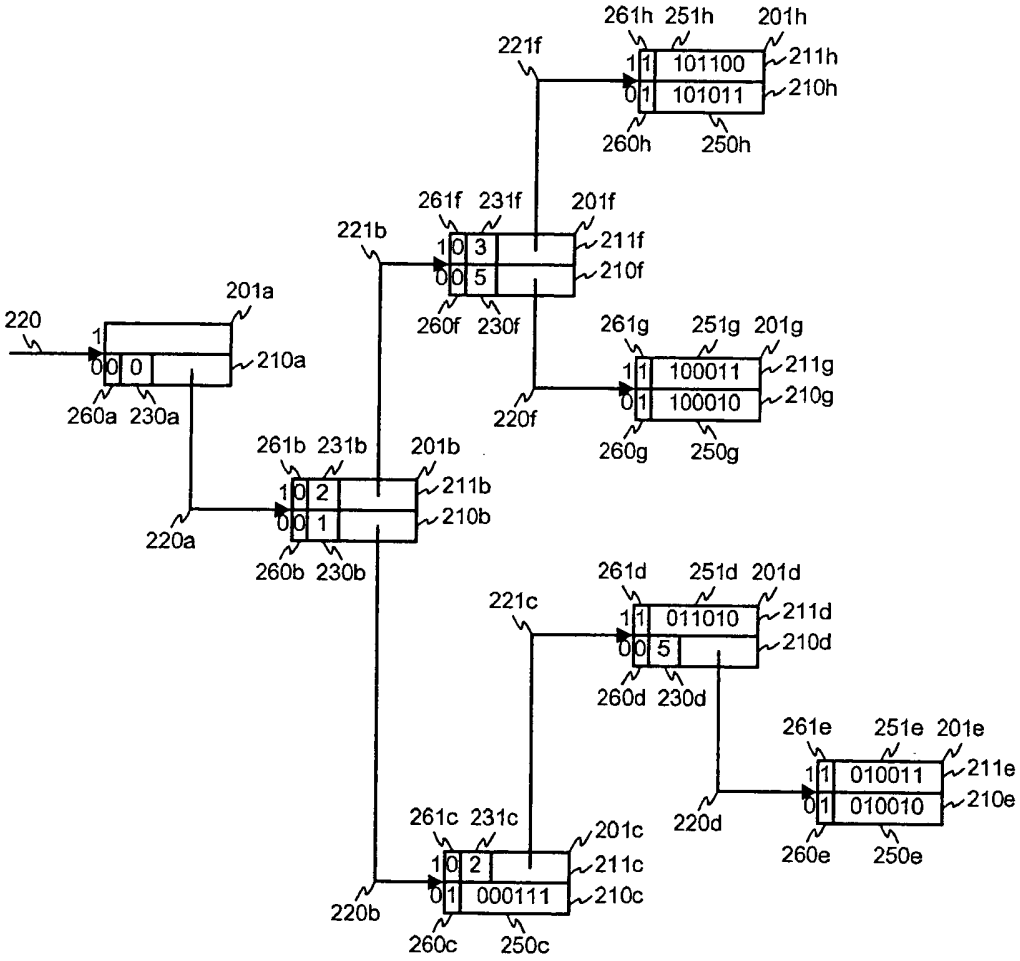

220
201a
1
00 0
210a
260a 230a
220a
261b 231b 201b
1 0 2
211b
0 0 1
210b
260b 230b
221b
261f 231f 201f
1 0 3
211f
0 0 5
210f
260f 230f
220f
261g 251g 201g
1 1 100011
211g
0 1 100010
210g
260g 250g
221f
261h 251h 201h
1 1 101100
211h
0 1 101011
210h
260h 250h
221c
261d 251d 201d
1 1 011010
211d
0 0 5
210d
260d 230d
220d
261e 251e 201e
1 1 010011
211e
0 1 010010
210e
260e 250e
220b
261c 231c 201c
1 0 2
211c
0 1 000111
210c
260c 250c

圖3

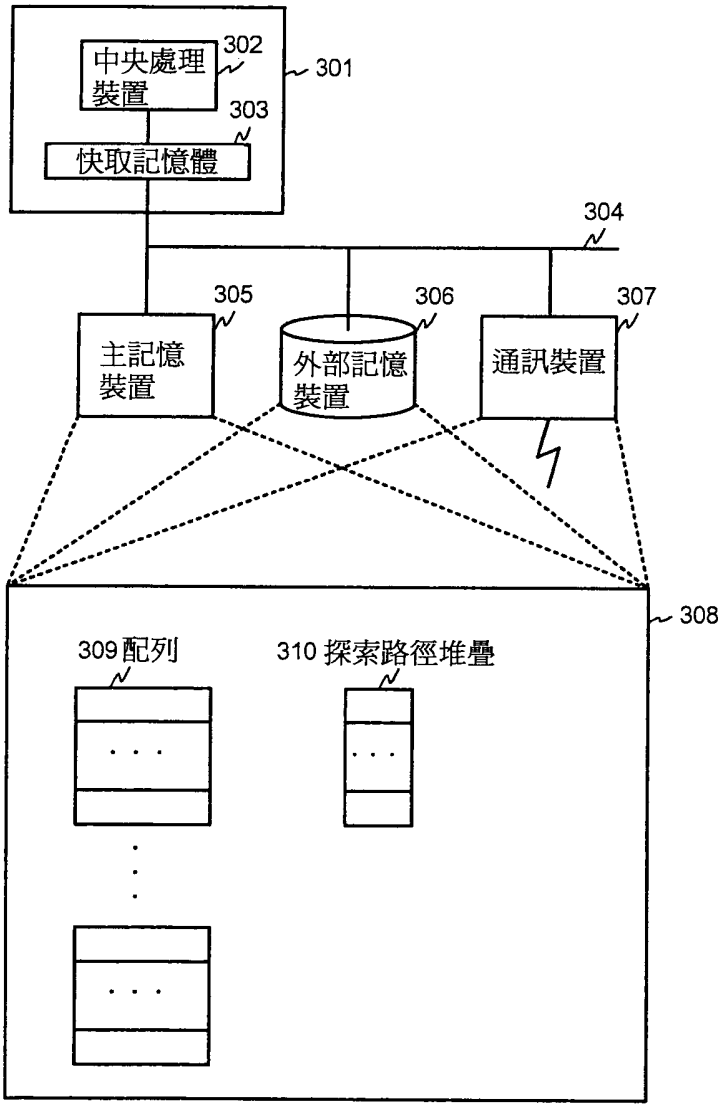

中央處理裝置
302
301
303
快取記憶體
304
305
306
307
主記憶裝置
外部記憶裝置
通訊裝置
308
309 配列
310 探索路徑堆疊

圖4

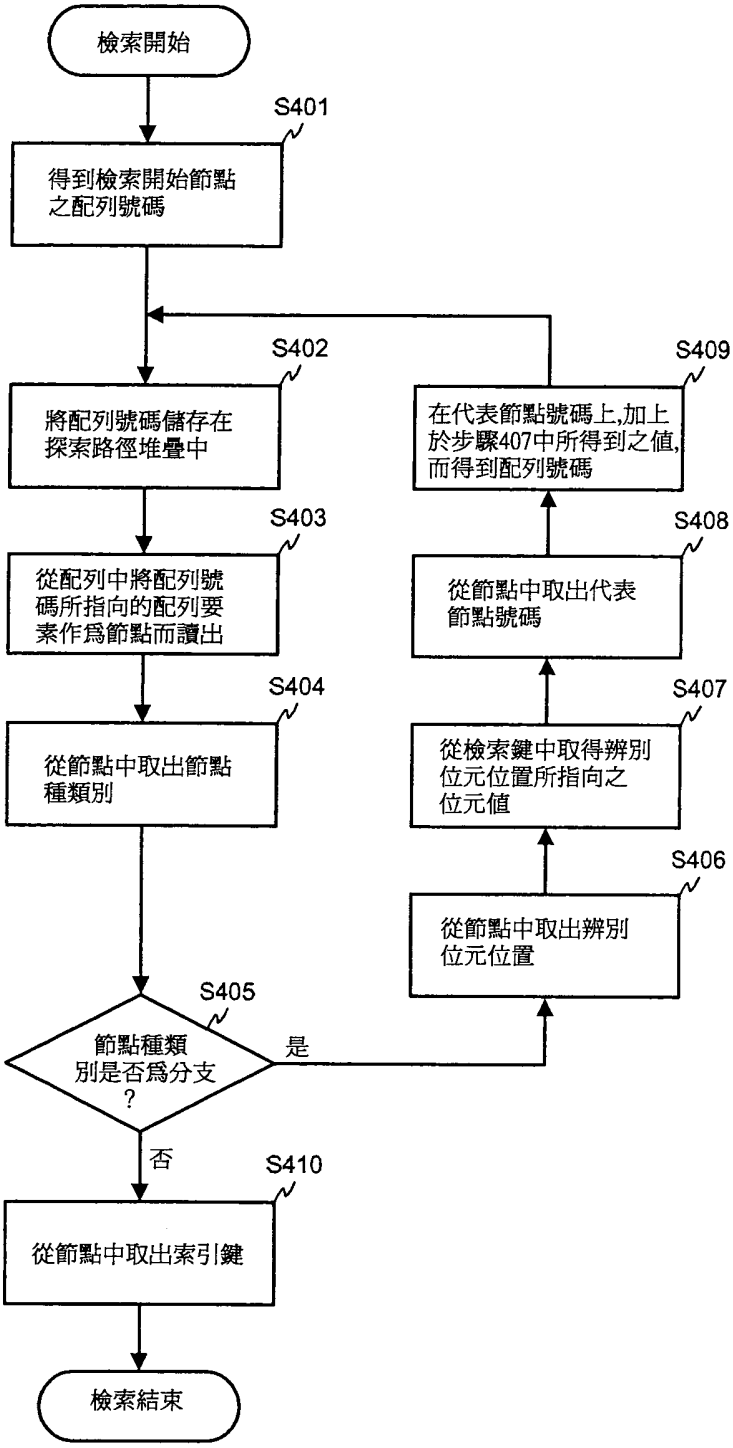

檢索開始
S401
得到檢索開始節點之配列號碼
S402
將配列號碼儲存在探索路徑堆疊中
S403
從配列中將配列號碼所指向的配列要素作爲節點而讀出
S404
從節點中取出節點種類別
S405
節點種類別是否爲分支？
是
否
S406
從節點中取出辨別位元位置
S407
從檢索鍵中取得辨別位元位置所指向之位元值
S408
從節點中取出代表節點號碼
S409
在代表節點號碼上,加上於步驟407中所得到之值,而得到配列號碼
S410
從節點中取出索引鍵
檢索結束

圖5

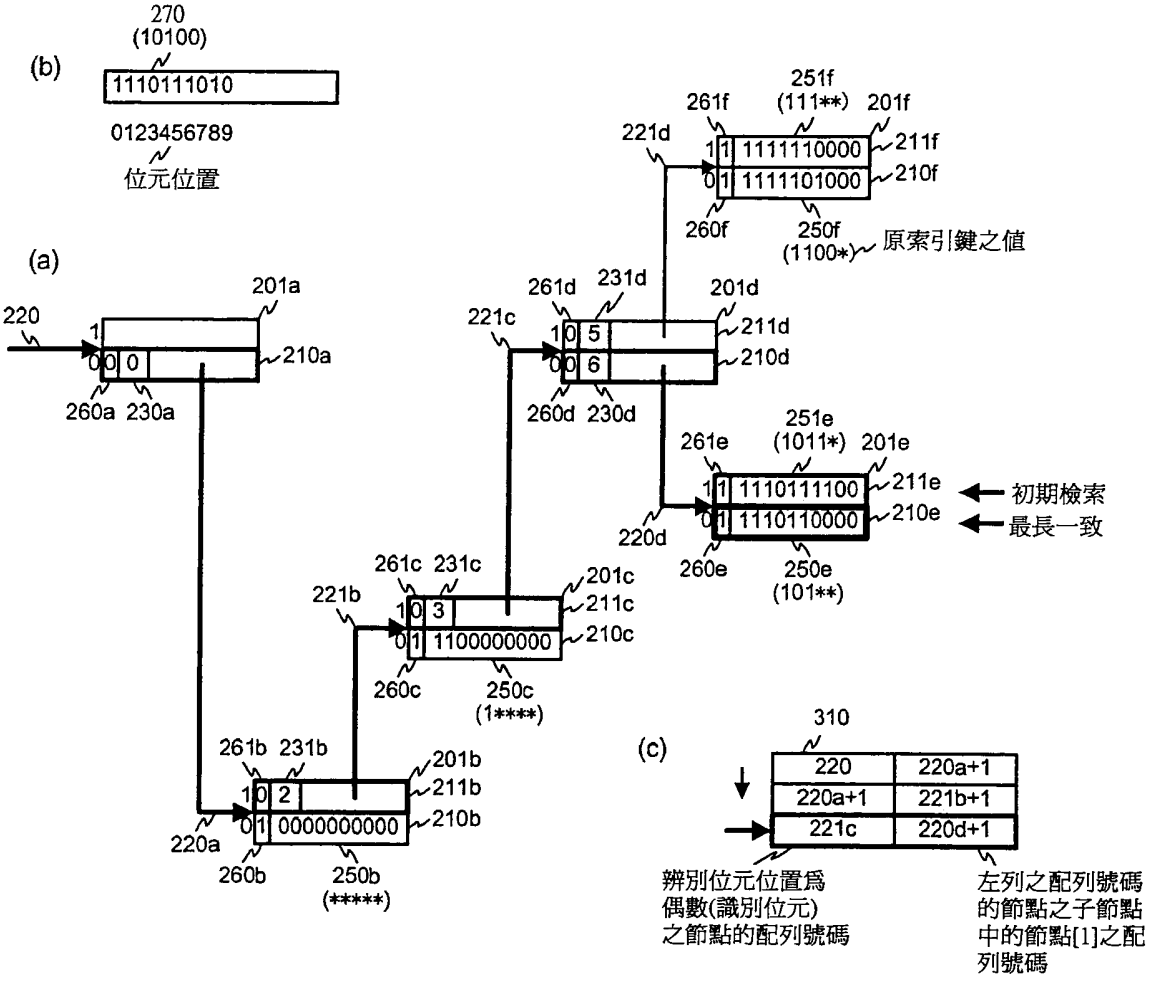

(b)
270
(10100)
1110111010
0123456789
位元位置
(a)
220
201a
210a
260a
230a
221d
261f
251f
(111**)
201f
1111110000
211f
1111101000
210f
260f
250f
(1100*)
原索引鍵之值
261d
231d
201d
211d
210d
260d
230d
221c
261e
251e
(1011*)
201e
1110111100
211e
初期檢索
1110110000
210e
最長一致
220d
260e
250e
(101**)
261c
231c
201c
211c
210c
221b
1100000000
260c
250c
(1****)
261b
231b
201b
211b
210b
220a
0000000000
260b
250b
(*****)
(c)
310
220
220a+1
220a+1
221b+1
221c
220d+1
辨別位元位置爲
偶數(識別位元)
之節點的配列號碼
左列之配列號碼
的節點之子節點
中的節點[1]之配
列號碼

圖6

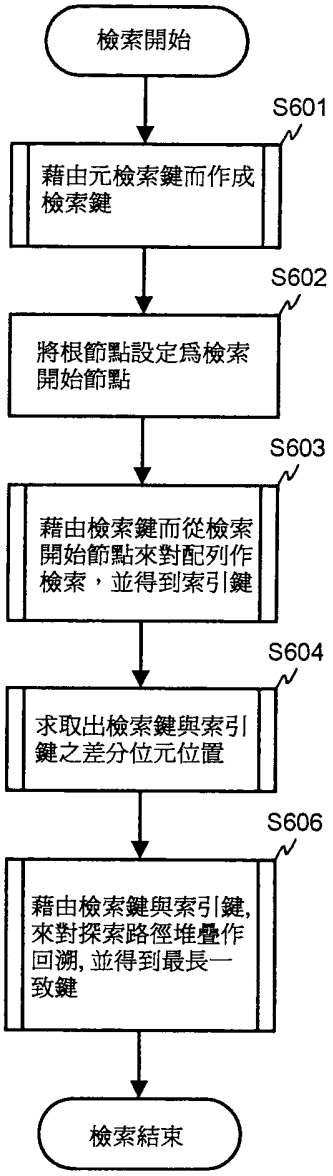

檢索開始
S601
藉由元檢索鍵而作成檢索鍵
S602
將根節點設定爲檢索開始節點
S603
藉由檢索鍵而從檢索開始節點來對配列作檢索，並得到索引鍵
S604
求取出檢索鍵與索引鍵之差分位元位置
S606
藉由檢索鍵與索引鍵，來對探索路徑堆疊作回溯，並得到最長一致鍵
檢索結束

圖7

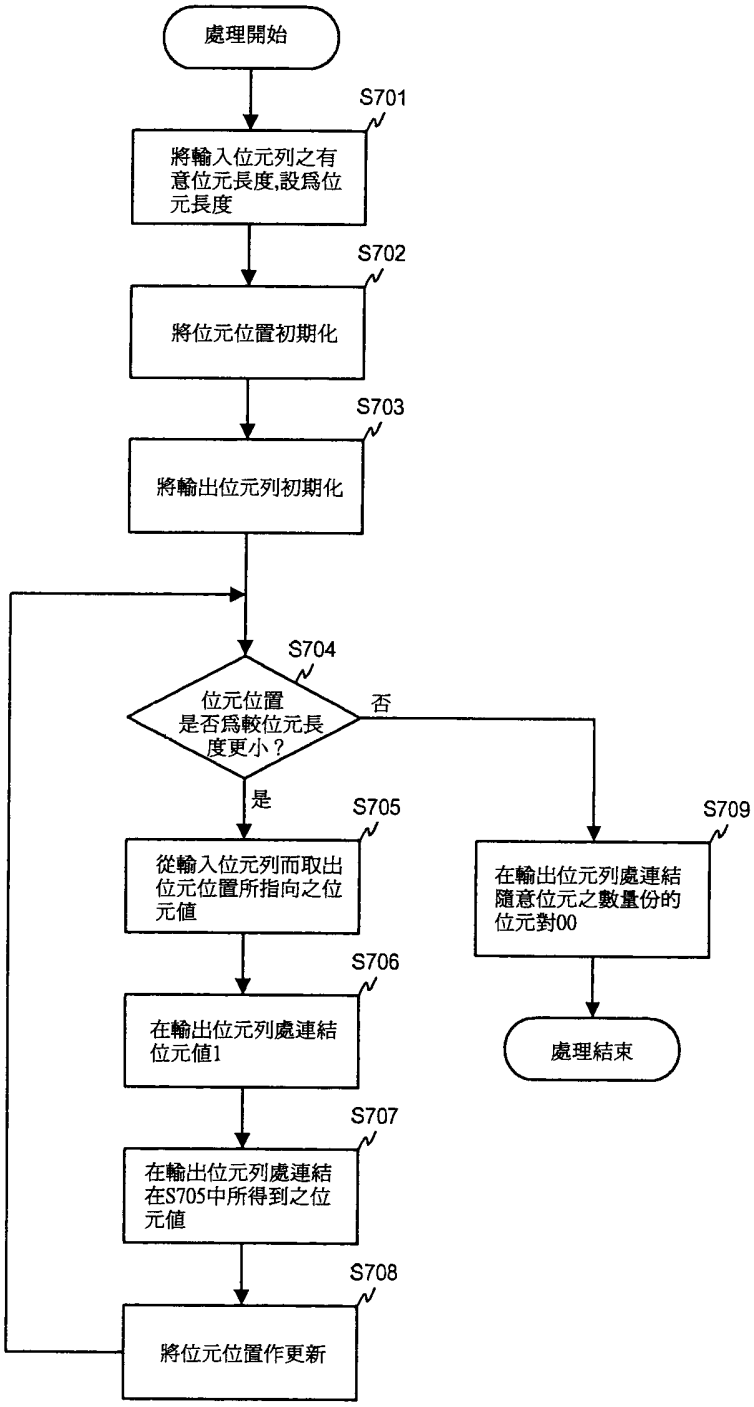

處理開始
S701
將輸入位元列之有意位元長度,設爲位元長度
S702
將位元位置初期化
S703
將輸出位元列初期化
S704
位元位置是否爲較位元長度更小？
否
是
S705
從輸入位元列而取出位元位置所指向之位元值
S706
在輸出位元列處連結位元值1
S707
在輸出位元列處連結在S705中所得到之位元值
S708
將位元位置作更新
S709
在輸出位元列處連結隨意位元之數量份的位元對00
處理結束

圖8

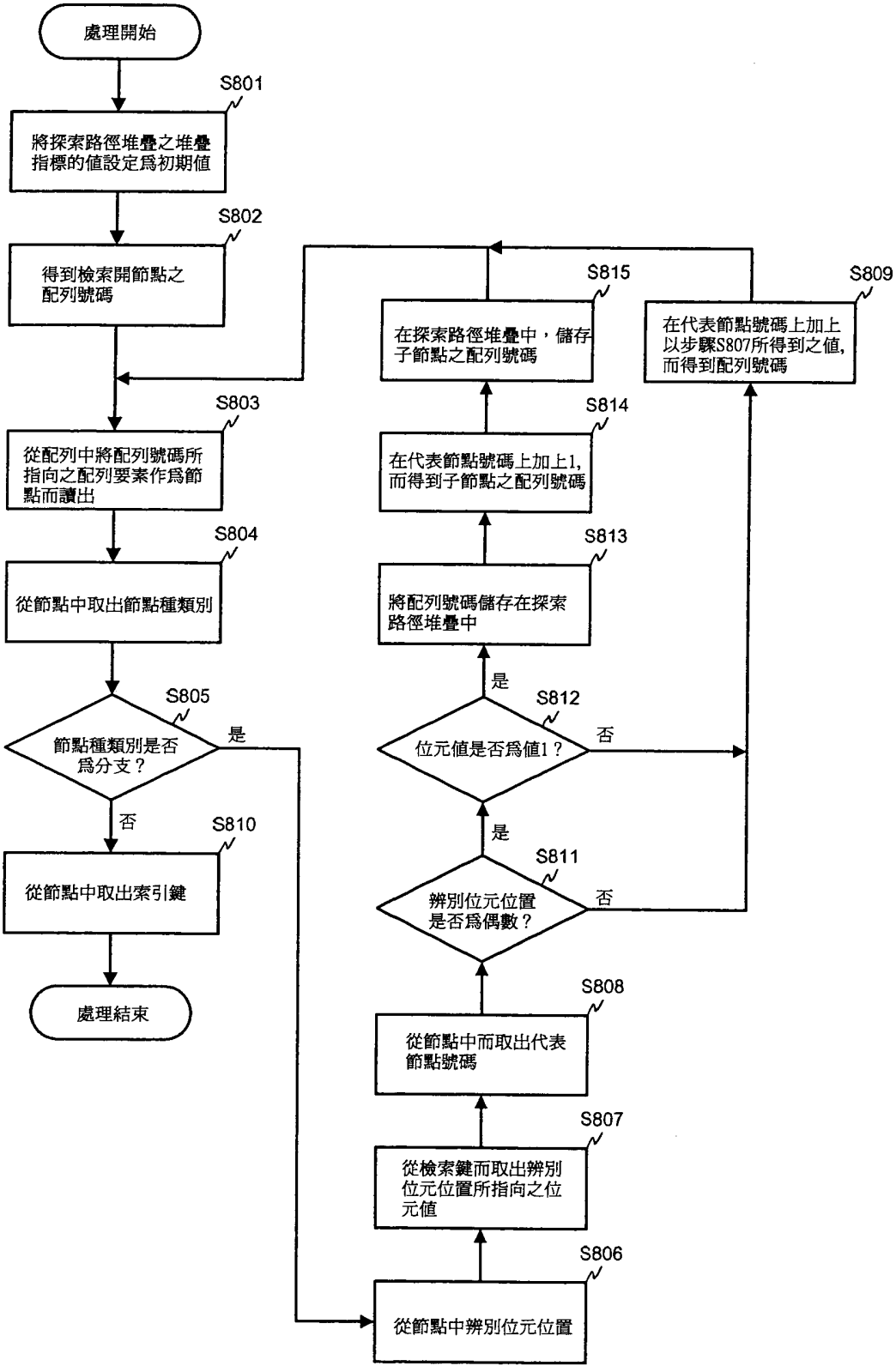

處理開始
S801
將探索路徑堆疊之堆疊指標的值設定爲初期值
S802
得到檢索開節點之配列號碼
S803
從配列中將配列號碼所指向之配列要素作爲節點而讀出
S804
從節點中取出節點種類別
S805
節點種類別是否爲分支？
是
否
S810
從節點中取出索引鍵
處理結束
S806
從節點中辨別位元位置
S807
從檢索鍵而取出辨別位元位置所指向之位元值
S808
從節點中而取出代表節點號碼
S811
辨別位元位置是否爲偶數？
是
否
S812
位元值是否爲值1？
是
否
S813
將配列號碼儲存在探索路徑堆疊中
S814
在代表節點號碼上加上1，而得到子節點之配列號碼
S815
在探索路徑堆疊中，儲存子節點之配列號碼
S809
在代表節點號碼上加上以步驟S807所得到之值，而得到配列號碼

圖9

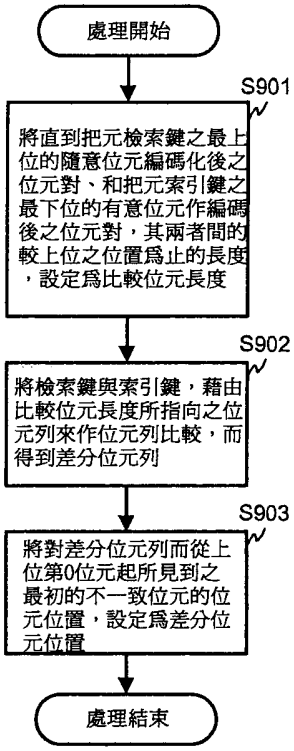

處理開始
S901
將直到把元檢索鍵之最上位的隨意位元編碼化後之位元對、和把元索引鍵之最下位的有意位元作編碼後之位元對，其兩者間的較上位之位置爲止的長度，設定爲比較位元長度
S902
將檢索鍵與索引鍵，藉由比較位元長度所指向之位元列來作位元列比較，而得到差分位元列
S903
將對差分位元列而從上位第0位元起所見到之最初的不一致位元的位元位置，設定爲差分位元位置
處理結束

圖10

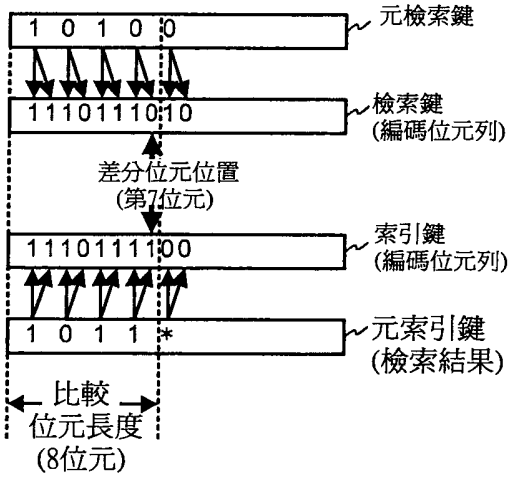

1 0 1 0 0
元檢索鍵
1 1 1 0 1 1 1 0 1 0
檢索鍵
(編碼位元列)
差分位元位置
(第7位元)
1 1 1 0 1 1 1 1 0 0
索引鍵
(編碼位元列)
1 0 1 1 *
元索引鍵
(檢索結果)
比較
位元長度
(8位元)

圖11

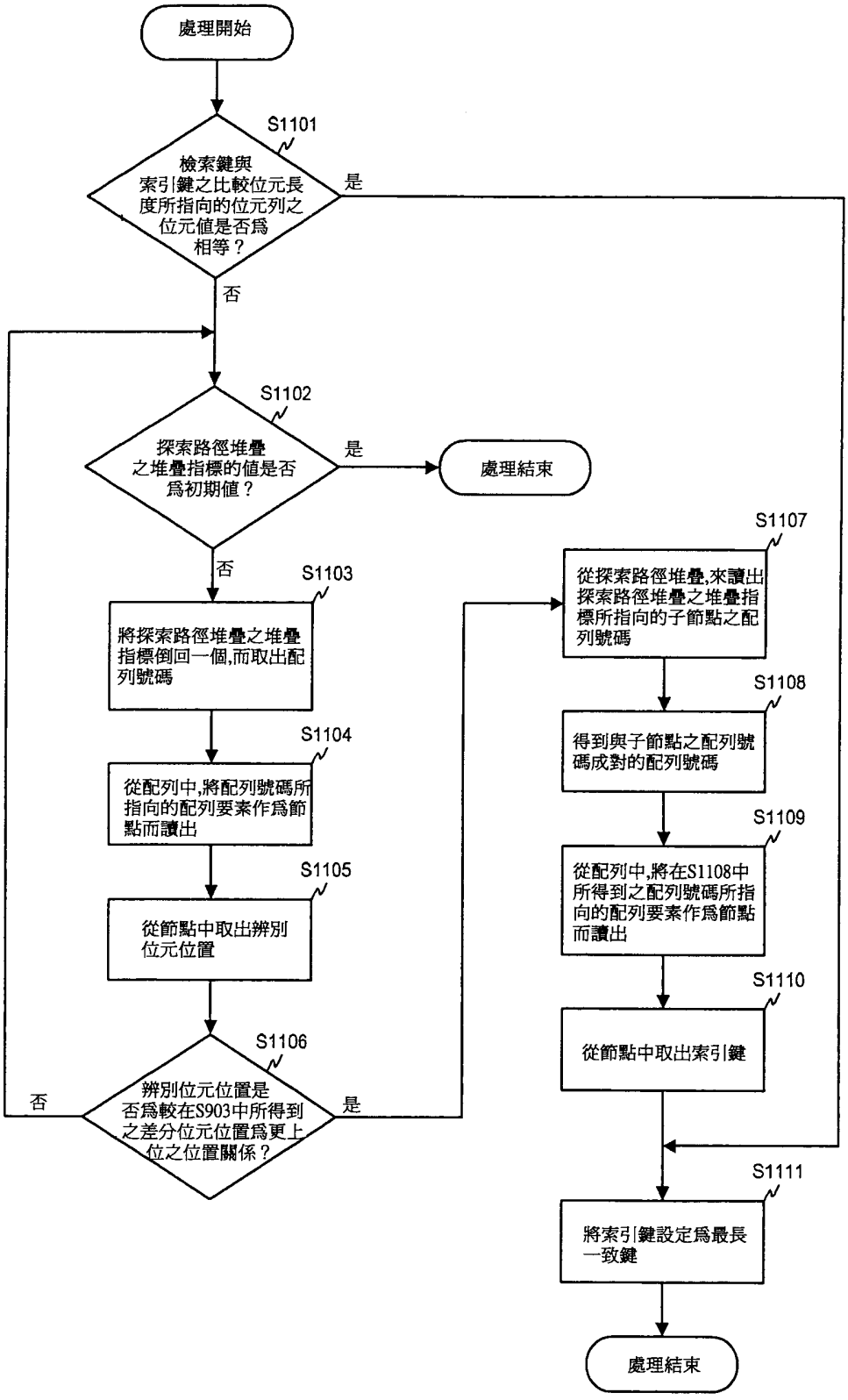

處理開始
S1101
檢索鍵與索引鍵之比較位元長度所指向的位元列之位元值是否爲相等？
是
否
S1102
探索路徑堆疊之堆疊指標的值是否爲初期值？
是
處理結束
否
S1103
將探索路徑堆疊之堆疊指標倒回一個,而取出配列號碼
S1104
從配列中,將配列號碼所指向的配列要素作爲節點而讀出
S1105
從節點中取出辨別位元位置
S1106
辨別位元位置是否爲較在S903中所得到之差分位元位置爲更上位之位置關係？
否
是
S1107
從探索路徑堆疊,來讀出探索路徑堆疊之堆疊指標所指向的子節點之配列號碼
S1108
得到與子節點之配列號碼成對的配列號碼
S1109
從配列中,將在S1108中所得到之配列號碼所指向的配列要素作爲節點而讀出
S1110
從節點中取出索引鍵
S1111
將索引鍵設定爲最長一致鍵
處理結束

圖12

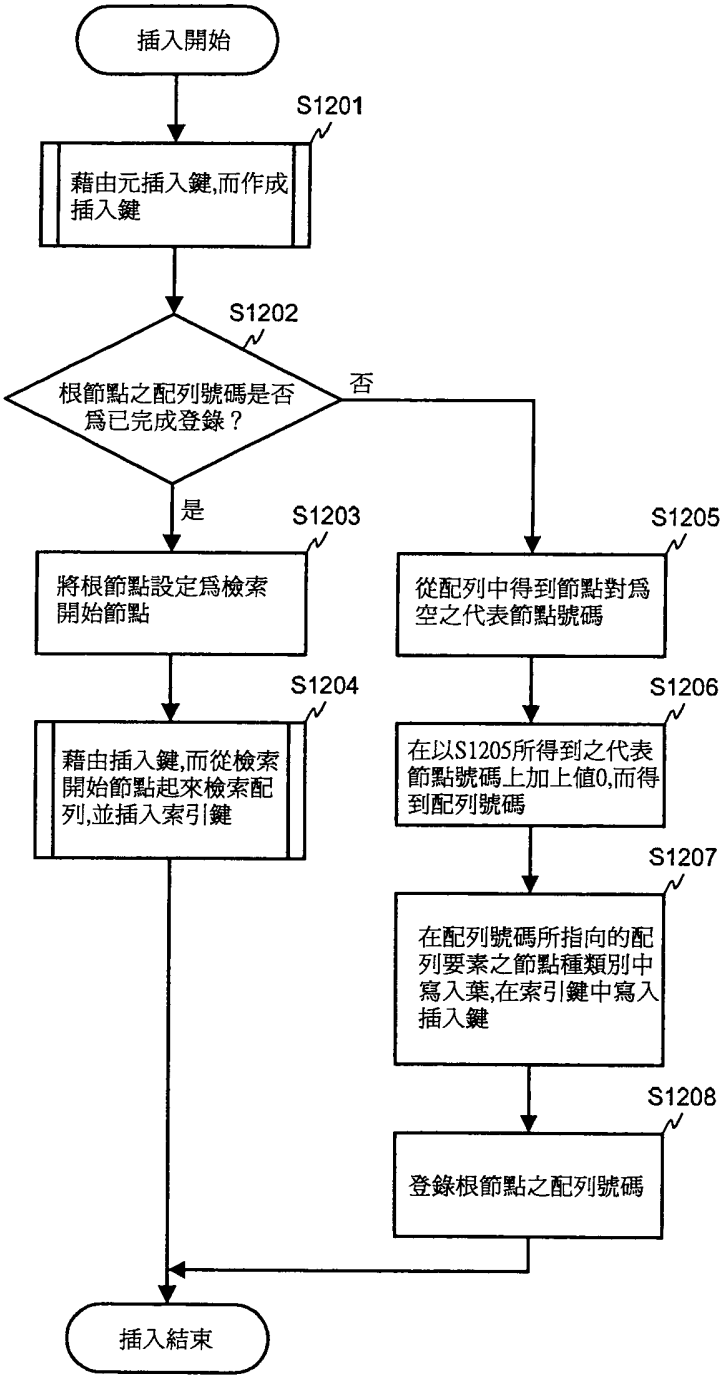

插入開始
S1201
藉由元插入鍵,而作成插入鍵
S1202
根節點之配列號碼是否爲已完成登錄？
否
是
S1203
將根節點設定爲檢索開始節點
S1204
藉由插入鍵,而從檢索開始節點起來檢索配列,並插入索引鍵
S1205
從配列中得到節點對爲空之代表節點號碼
S1206
在以S1205所得到之代表節點號碼上加上值0,而得到配列號碼
S1207
在配列號碼所指向的配列要素之節點種類別中寫入葉,在索引鍵中寫入插入鍵
S1208
登錄根節點之配列號碼
插入結束

圖13A

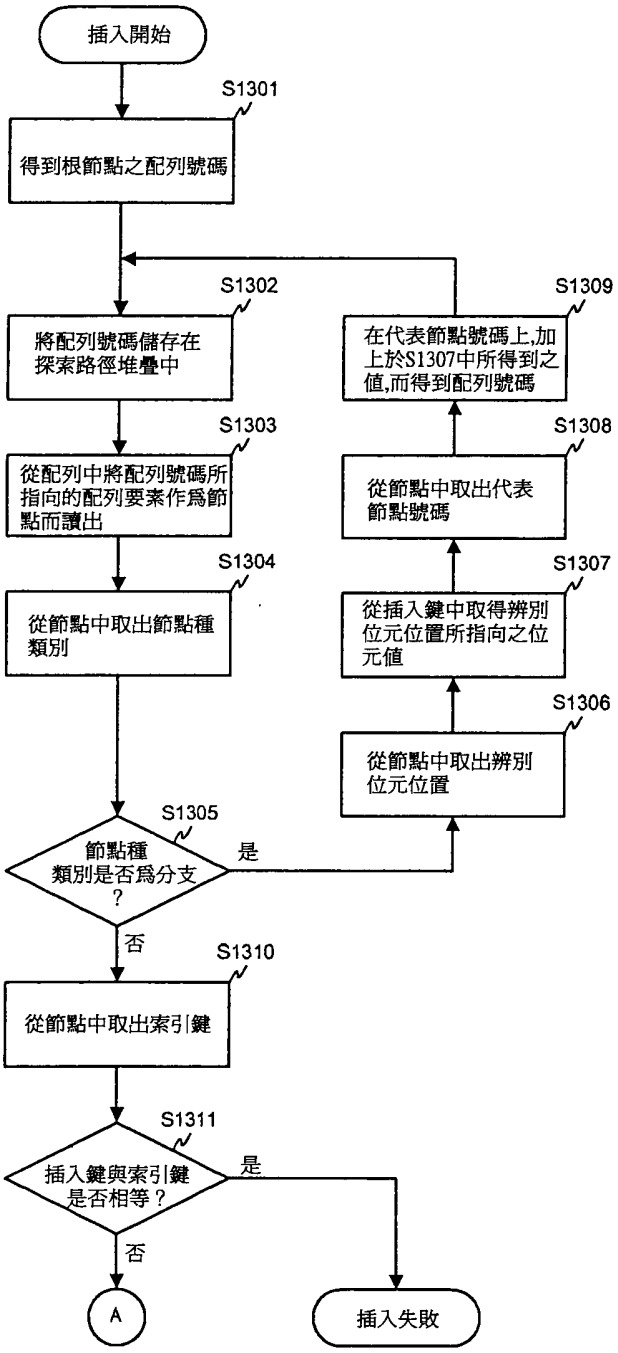

插入開始
S1301
得到根節點之配列號碼
S1302
將配列號碼儲存在探索路徑堆疊中
S1303
從配列中將配列號碼所指向的配列要素作爲節點而讀出
S1304
從節點中取出節點種類別
S1305
節點種類別是否爲分支？
是
否
S1306
從節點中取出辨別位元位置
S1307
從插入鍵中取得辨別位元位置所指向之位元值
S1308
從節點中取出代表節點號碼
S1309
在代表節點號碼上,加上於S1307中所得到之值,而得到配列號碼
S1310
從節點中取出索引鍵
S1311
插入鍵與索引鍵是否相等？
是
否
A
插入失敗

圖13B

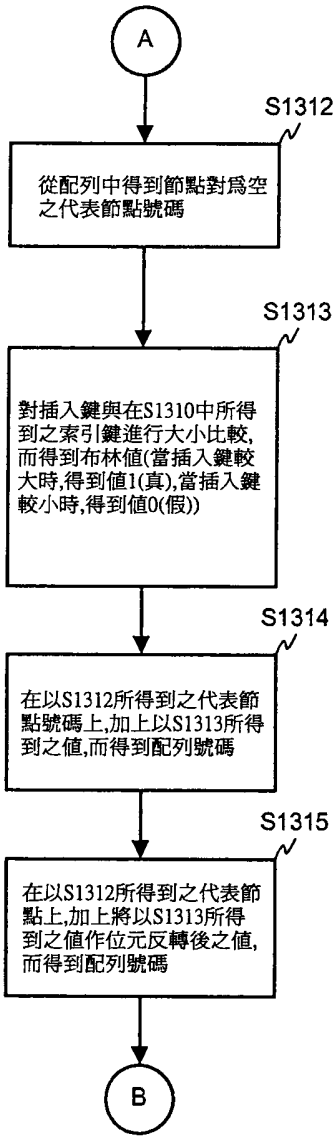

A
S1312
從配列中得到節點對爲空之代表節點號碼
S1313
對插入鍵與在S1310中所得到之索引鍵進行大小比較,而得到布林值(當插入鍵較大時,得到值1(真),當插入鍵較小時,得到值0(假))
S1314
在以S1312所得到之代表節點號碼上,加上以S1313所得到之值,而得到配列號碼
S1315
在以S1312所得到之代表節點上,加上將以S1313所得到之值作位元反轉後之值,而得到配列號碼
B

圖13C

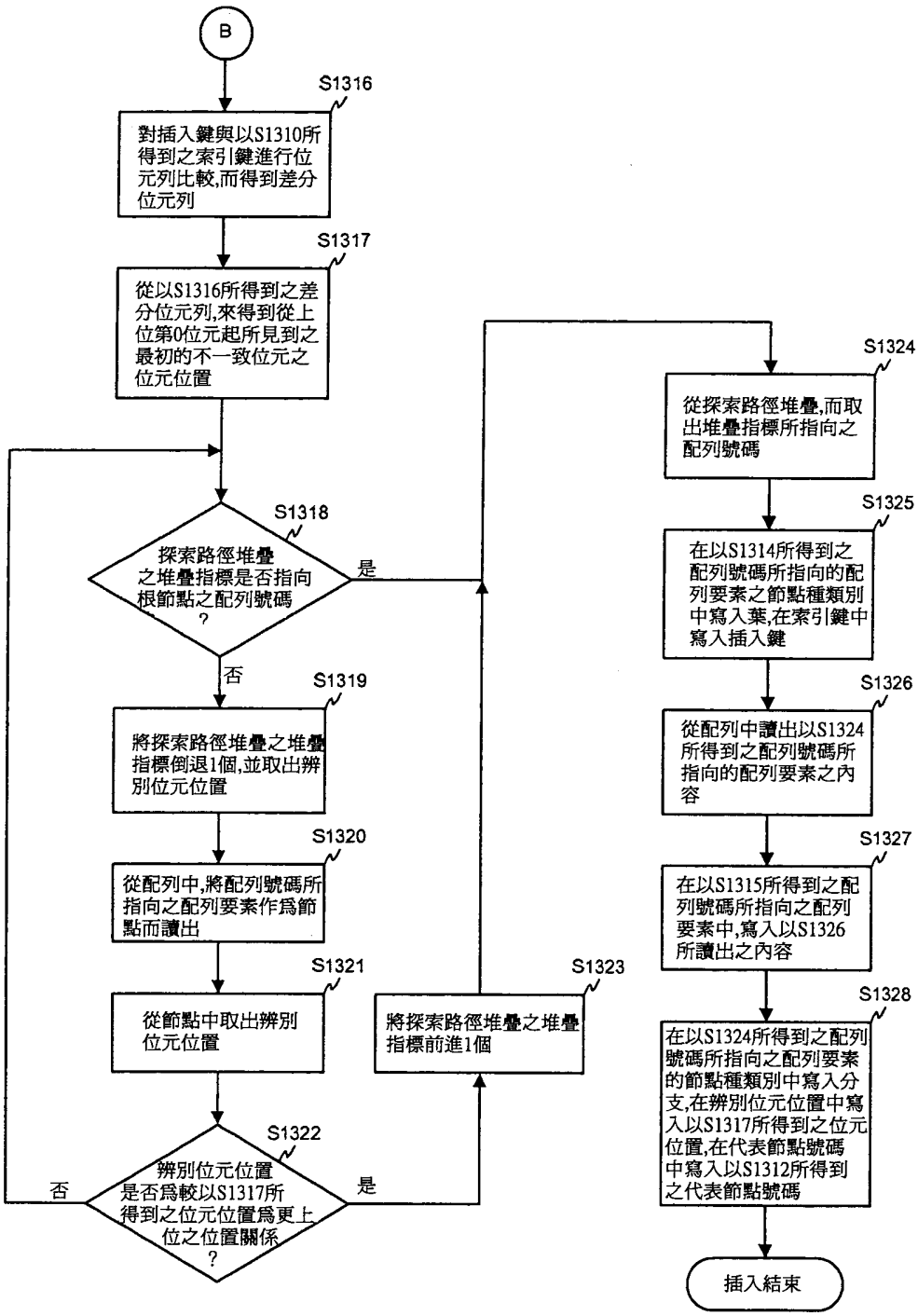

B
S1316
對插入鍵與以S1310所得到之索引鍵進行位元列比較,而得到差分位元列
S1317
從以S1316所得到之差分位元列,來得到從上位第0位元起所見到之最初的不一致位元之位元位置
S1318
探索路徑堆疊之堆疊指標是否指向根節點之配列號碼?
是
否
S1319
將探索路徑堆疊之堆疊指標倒退1個,並取出辨別位元位置
S1320
從配列中,將配列號碼所指向之配列要素作爲節點而讀出
S1321
從節點中取出辨別位元位置
S1322
辨別位元位置是否爲較以S1317所得到之位元位置爲更上位之位置關係?
否
是
S1323
將探索路徑堆疊之堆疊指標前進1個
S1324
從探索路徑堆疊,而取出堆疊指標所指向之配列號碼
S1325
在以S1314所得到之配列號碼所指向的配列要素之節點種類別中寫入葉,在索引鍵中寫入插入鍵
S1326
從配列中讀出以S1324所得到之配列號碼所指向的配列要素之內容
S1327
在以S1315所得到之配列號碼所指向之配列要素中,寫入以S1326所讀出之內容
S1328
在以S1324所得到之配列號碼所指向之配列要素的節點種類別中寫入分支,在辨別位元位置中寫入以S1317所得到之位元位置,在代表節點號碼中寫入以S1312所得到之代表節點號碼
插入結束

圖14

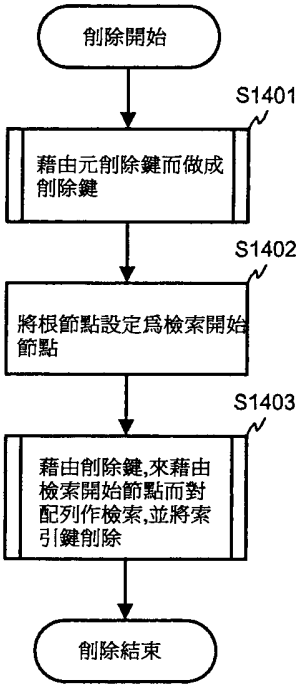

刪除開始
S1401
藉由元刪除鍵而做成刪除鍵
S1402
將根節點設定爲檢索開始節點
S1403
藉由刪除鍵,來藉由檢索開始節點而對配列作檢索,並將索引鍵刪除
刪除結束

圖15A

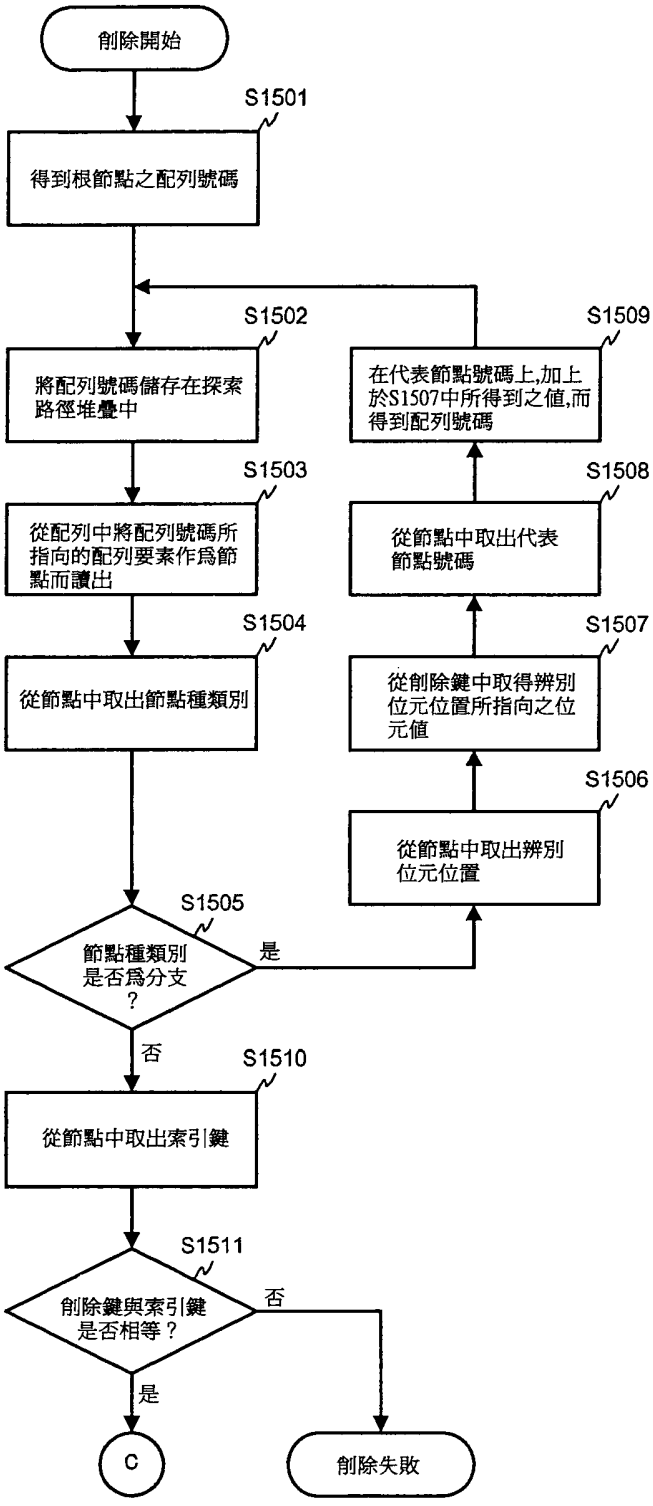

刪除開始
S1501
得到根節點之配列號碼
S1502
將配列號碼儲存在探索路徑堆疊中
S1503
從配列中將配列號碼所指向的配列要素作爲節點而讀出
S1504
從節點中取出節點種類別
S1505
節點種類別是否爲分支？
是
否
S1506
從節點中取出辨別位元位置
S1507
從刪除鍵中取得辨別位元位置所指向之位元值
S1508
從節點中取出代表節點號碼
S1509
在代表節點號碼上,加上於S1507中所得到之值,而得到配列號碼
S1510
從節點中取出索引鍵
S1511
刪除鍵與索引鍵是否相等？
否
是
C
刪除失敗

圖15B

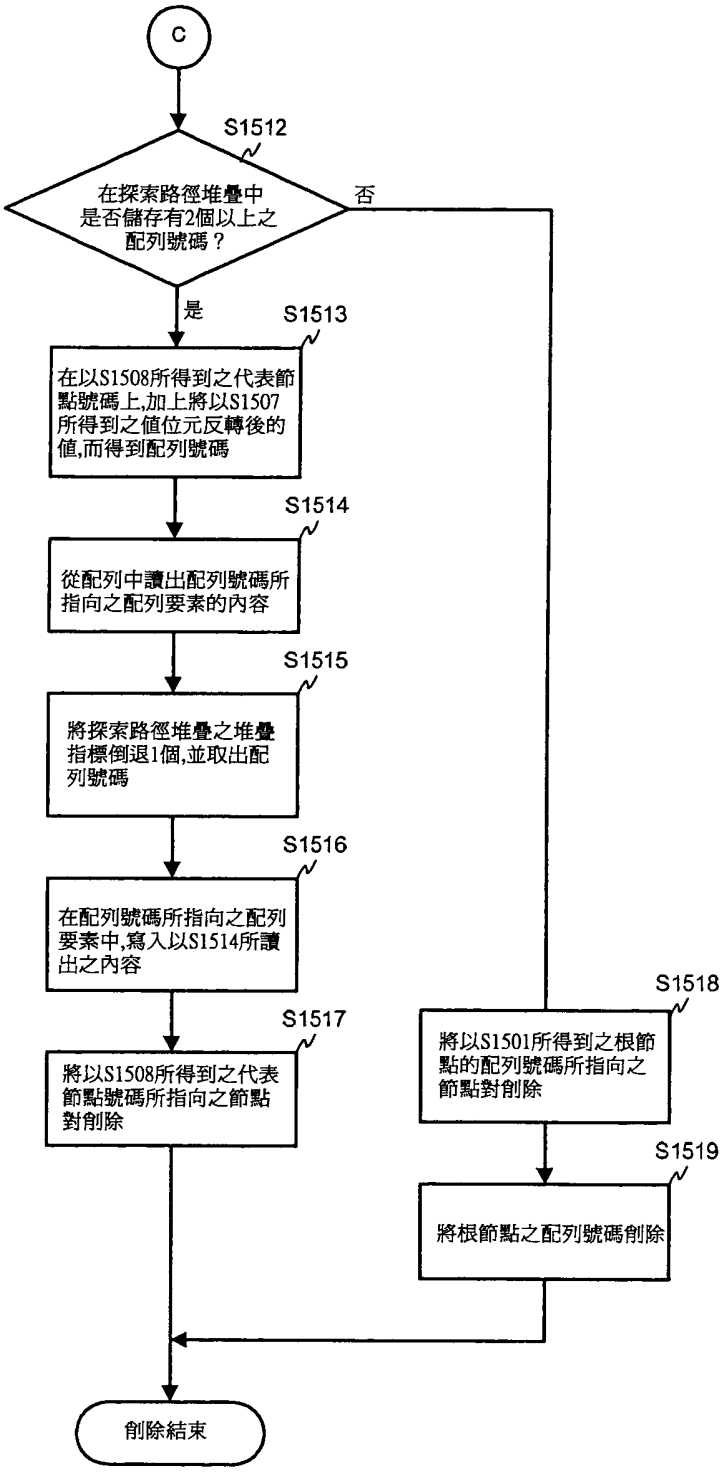
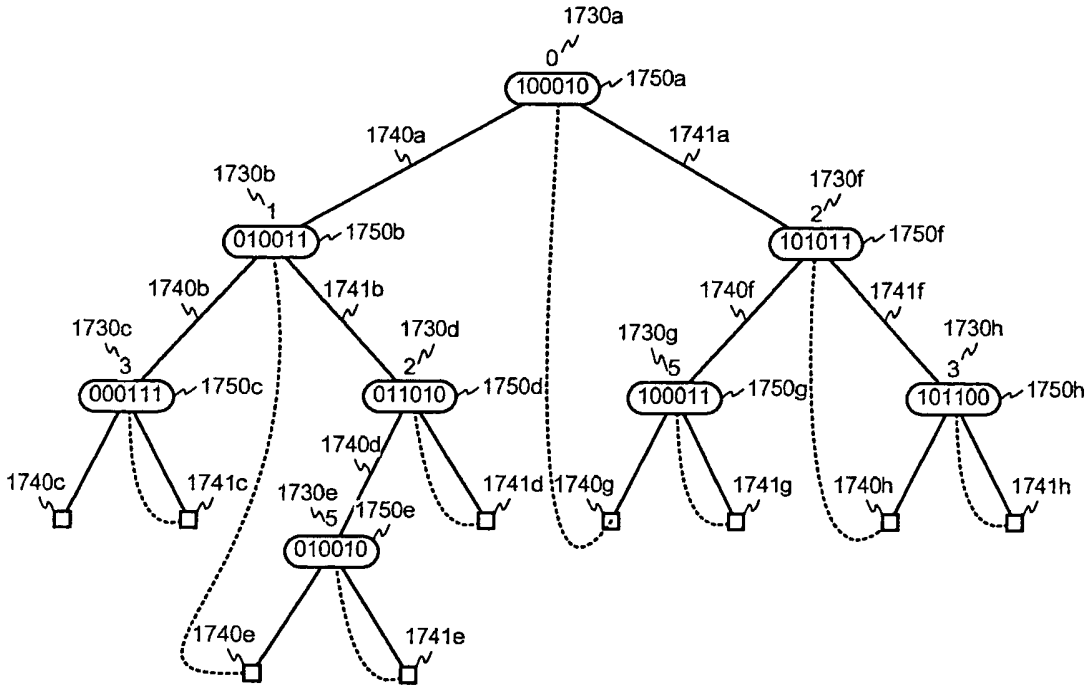

C
S1512
在探索路徑堆疊中是否儲存有2個以上之配列號碼？
否
是
S1513
在以S1508所得到之代表節點號碼上,加上將以S1507所得到之值位元反轉後的值,而得到配列號碼
S1514
從配列中讀出配列號碼所指向之配列要素的內容
S1515
將探索路徑堆疊之堆疊指標倒退1個,並取出配列號碼
S1516
在配列號碼所指向之配列要素中,寫入以S1514所讀出之內容
S1517
將以S1508所得到之代表節點號碼所指向之節點對削除
S1518
將以S1501所得到之根節點的配列號碼所指向之節點對削除
S1519
將根節點之配列號碼削除
削除結束

圖16


1730a
0
100010
1750a
1740a
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
1740c
1741c
1740d
1730e
5
1750e
1741d
1740g
1741g
1740h
1741h
010010
1740e
1741e

七、指定代表圖：

（一）、本案指定代表圖為：第（5）圖

（二）、本代表圖之元件代表符號簡單說明：

270:檢索鍵

310:探索路徑堆疊

250c~250h、251d~251h:索引鍵

260a~260h、261b~261h:節點種類別

230a~230f、231b~231f:辨別位元位置

220、220a~220f、221b~221f:代表節點號碼

201a~201h:節點對

210a~210h:節點[0]、代表節點

211b~211h:節點[1]、與代表節點成對之非代表節點

八、本案若有化學式時，請揭示最能顯示發明特徵的化學式：無