(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4567754号
(P4567754)

(45)発行日　平成22年10月20日(2010.10.20)　(24)登録日　平成22年8月13日(2010.8.13)

(51)Int.Cl.　F I
G06F 17/30 (2006.01)　G06F 17/30 414A
G06F 17/30 140

請求項の数 15　(全 30 頁)

(21)出願番号　特願2008-7690(P2008-7690)
(22)出願日　平成20年1月17日(2008.1.17)
(65)公開番号　特開2009-169715(P2009-169715A)
(43)公開日　平成21年7月30日(2009.7.30)
審査請求日　平成22年6月25日(2010.6.25)

早期審査対象出願

(73)特許権者　506235616
株式会社エスグランツ
千葉県千葉市美浜区高洲三丁目５番３棟１２１０号
(74)代理人　100133570
弁理士　▲徳▼永　民雄
(72)発明者　新庄　敏男
千葉県千葉市美浜区高洲三丁目５番３棟１２１０号　株式会社エスグランツ内
(72)発明者　國分　光裕
千葉県千葉市美浜区高洲三丁目５番３棟１２１０号　株式会社エスグランツ内

審査官　吉田　誠

最終頁に続く

(54)【発明の名称】ビット列検索装置、検索方法及びプログラム

(57)【特許請求の範囲】
【請求項１】
コンピュータが実行する、
ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対、からなるビット列検索に用いるツリーであって、
前記ルートノードは、ツリーの始点を表すノードであって、該ツリーのノードが１つのときは前記リーフノード、ツリーのノードが２つ以上のときは前記ブランチノードであり、
前記ブランチノードは、リンク先のノード対の一方のノードである代表ノードの位置を示す第一の位置情報を含み、前記リーフノードは、検索対象のビット列からなるインデックスキーを格納した記憶領域の位置を示す第二の位置情報を含むカップルドノードツリーを用いたビット列検索方法において、
前記インデックスキーは３つ以上のキーの列であって該キー列の末尾のキーは重複のないユニークキーであり、前記ブランチノードは、ビット列検索を行う検索キー列中のキーの位置を識別するキー順序番号と、該キーの弁別ビット位置をさらに含み、
前記カップルドノードツリーの任意のノードを検索開始ノードとして、前記ブランチノードにおいて、前記検索キー列のうち該ブランチノードに含まれるキー順序番号で識別される位置のキーの前記弁別ビット位置のビット値に応じて、リンク先のノード対の代表ノードかあるいはそれと隣接した記憶領域に配置されたノードにリンクすることを順次前記

リーフノードに至るまで繰り返すことにより、前記リーフノードに含まれる前記第二の位置情報が示す記憶領域に格納されたインデックスキーを、前記検索開始ノードをルートノードとする前記カップルドノードツリーの任意の部分木の前記検索キー列による検索結果である検索結果キー列とすることを特徴とするビット列検索方法。

【請求項２】

前記カップルドノードツリーは配列に記憶され、前記第一の位置情報は、該第一の位置情報に対応する前記代表ノードが格納された前記配列の配列要素の配列番号であることを特徴とする請求項１記載のビット列検索方法。

【請求項３】

前記検索開始ノードの格納された配列要素の配列番号及び前記検索開始ノードから前記リーフノードに至るリンク先のノードの格納された配列要素の配列番号が、順次スタックに保持されていくことを特徴とする請求項２記載のビット列検索方法。

【請求項４】

コンピュータが実行する、

請求項１記載のビット列検索方法で用いるカップルドノードツリーに、新たなインデックスキーが格納された記憶領域の位置を示す前記第二の位置情報を含むリーフノードを挿入するリーフノード挿入方法において、

前記新たなインデックスキーを前記検索キー列とし、前記カップルドノードツリーのルートノードを検索開始ノードとして、ルートノードからリーフノードに至るリンク経路を記憶しながら請求項１記載のビット列検索方法により前記検索結果キー列を取得する検索結果キー列取得ステップと、

前記検索キー列と前記検索結果キー列のキーを先頭のキーから順次比較して最初に異なる値となるキーの位置を取得するキー順序番号取得ステップと、

前記検索キー列と前記検索結果キー列の前記キー順序番号取得ステップで取得したキーの位置にあるキーの間で大小比較とビット列比較を行う比較ステップと、

前記リンク経路上のブランチノードのキー順序番号と前記キー順序番号取得ステップで取得した前記キーの位置との相対的位置関係、及び前記比較ステップにおけるビット列比較で異なるビット値となる先頭のビット位置と前記リンク経路上のブランチノードの弁別ビット位置との相対的位置関係により、挿入される前記リーフノードともう一方のノードからなるノード対の挿入位置を決定する挿入位置決定ステップと、

前記比較ステップにおける前記大小比較の結果により、挿入される前記リーフノードを挿入される前記ノード対のどちらのノードとするかを決定するノード位置決定ステップと、

前記新たなインデックスキーを格納する記憶領域の位置を示す情報を前記第二の位置情報として前記リーフノードに格納するリーフノード生成ステップと、

を含むことを特徴とするリーフノード挿入方法。

【請求項５】

前記カップルドノードツリーは配列に記憶され、前記第一の位置情報は、該第一の位置情報に対応する前記代表ノードが格納された前記配列の配列要素の配列番号であることを特徴とする請求項４記載のリーフノード挿入方法。

【請求項６】

前記ルートノードの格納された配列要素の配列番号及び前記ルートノードから前記リーフノードに至るリンク先のノードの格納された配列要素の配列番号が、順次スタックに保持されていくことを特徴とする請求項５記載のリーフノード挿入方法。

【請求項７】

コンピュータが実行する、

請求項１記載のビット列検索方法で用いるカップルドノードツリーから、指定された前記インデックスキーが格納された記憶領域の位置を示す前記第二の位置情報を含むリーフノードを削除する、リーフノード削除方法において、

前記指定されたインデックスキーを検索キー列とし、前記カップルドノードツリーのル

ートノードを検索開始ノードとして請求項１記載のビット列検索方法により前記検索結果キー列を取得し、
前記検索結果キー列を格納する記憶領域の位置を示す情報を前記第二の位置情報として含むリーフノードと対をなすもう一方のノードを、該リーフノードのリンク元のブランチノードに格納することにより前記リーフノードを削除する、
ことを特徴とするリーフノード削除方法。
【請求項８】
前記カップルドノードツリーは配列に記憶され、前記第一の位置情報は、該第一の位置情報に対応する前記代表ノードが格納された前記配列の配列要素の配列番号であることを特徴とする請求項７記載のリーフノード削除方法。
【請求項９】
前記ルートノードの格納された配列要素の配列番号及び前記ルートノードから前記リーフノードに至るリンク先のノードの格納された配列要素の配列番号が、順次スタックに保持されていくことを特徴とする請求項８記載のリーフノード削除方法。
【請求項１０】
請求項１～９いずれか１項に記載の方法をコンピュータに実行させるためのプログラム。
【請求項１１】
ビット列検索に用いるツリー状のデータ構造であって、
ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対、からなり、
前記ルートノードは、ツリーの始点を表すノードであって、該ツリーのノードが１つのときは前記リーフノード、ツリーのノードが２つ以上のときは前記ブランチノードであり、
前記ブランチノードは、リンク先のノード対の一方のノードである代表ノードの位置を示す第一の位置情報を含み、
前記インデックスキーは３つ以上のキーの列であって該キー列の末尾のキーは重複のないユニークキーであり、前記ブランチノードは、ビット列検索を行う検索キー列中のキーの位置を識別するキー順序番号と、該キーの弁別ビット位置をさらに含み、
前記カップルドノードツリーの任意のノードを検索開始ノードとして、前記ブランチノードにおいて、前記検索キー列のうち、該ブランチノードに含まれるキー順序番号で識別される位置のキーの前記弁別ビット位置のビット値に応じてリンク先のノード対の代表ノードかあるいはそれと隣接した記憶領域に配置されたノードにリンクすることを順次前記リーフノードに至るまで繰り返すことにより、コンピュータが前記検索キー列による検索の実行を可能とすることを特徴とするデータ構造。
【請求項１２】
前記データ構造は配列に記憶され、前記第一の位置情報は、該第一の位置情報に対応する前記代表ノードが格納された前記配列の配列要素の配列番号であることを特徴とする請求項１１記載のデータ構造。
【請求項１３】
ビット列検索装置において、
ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対、からなるビット列検索に用いるツリーであって、
前記ルートノードは、ツリーの始点を表すノードであって、該ツリーのノードが１つのときは前記リーフノード、ツリーのノードが２つ以上のときは前記ブランチノードであり、
前記ブランチノードは、リンク先のノード対の一方のノードである代表ノードの位置を示す第一の位置情報を含み、前記リーフノードは、検索対象のビット列からなるインデックスキーを格納した記憶領域の位置を示す第二の位置情報を含むカップルドノードツリー

を備え、
前記インデックスキーは３つ以上のキーの列であって該キー列の末尾のキーは重複のないユニークキーであり、前記ブランチノードは、ビット列検索を行う検索キー列中のキーの位置を識別するキー順序番号と、該キーの弁別ビット位置をさらに含むものであり、
前記カップルドノードツリーの任意のノードを検索開始ノードとして、前記ブランチノードにおいて、前記検索キー列のうち該ブランチノードに含まれるキー順序番号で識別される位置のキーの前記弁別ビット位置のビット値に応じてリンク先のノード対の代表ノードかあるいはそれと隣接した記憶領域に配置されたノードにリンクすることを順次前記リーフノードに至るまで繰り返すことにより、前記リーフノードに含まれる前記第二の位置情報が示す記憶領域に格納されたインデックスキーを、前記検索開始ノードをルートノードとする前記カップルドノードツリーの任意の部分木の前記検索キー列による検索結果である検索結果キー列とすることを特徴とするビット列検索装置。
【請求項１４】
配列を備え、前記カップルドノードツリーは該配列に記憶され、前記第一の位置情報は、該第一の位置情報に対応する前記代表ノードが格納された前記配列の配列要素の配列番号であることを特徴とする請求項１３記載のビット列検索装置。
【請求項１５】
スタックを備え、前記検索開始ノードの格納された配列要素の配列番号及び前記検索開始ノードから前記リーフノードに至るリンク先のノードの格納された配列要素の配列番号が、順次前記スタックに保持されていくことを特徴とする請求項１４記載のビット列検索装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明はビット列の集合から所望のビット列を検索する検索装置、検索方法及びプログラムに関するものであり、特にビット列を記憶するデータ構造に工夫をして、検索速度等の向上を図る技術分野のものである。
【背景技術】
【０００２】
近年、社会の情報化が進展し、大規模なデータベースが各所で利用されるようになってきている。このような大規模なデータベースからレコードを検索するには、各レコードの記憶されたアドレスと対応づけられたレコード内の項目をインデックスキーとして検索をし、所望のレコードを探し出すことが通例である。また、全文検索における文字列も、文書のインデックスキーと見なすことができる。
【０００３】
そして、それらのインデックスキーはビット列で表現されることから、データベースの検索はビット列の検索に帰着されるということができる。
上記ビット列の検索を高速に行うために、ビット列を記憶するデータ構造を種々に工夫することが従来から行われている。このようなものの一つとして、パトリシアツリーという木構造が知られている。
【０００４】
図１は、上述の従来の検索処理に用いられているパトリシアツリーの一例を示すものである。パトリシアツリーのノードは、インデックスキー、検索キーの検査ビット位置、左右のリンクポインタを含んで構成される。明示はされていないが、ノードにはインデックスキーに対応するレコードにアクセスするための情報が含まれていることは勿論である。
【０００５】
図１の例では、インデックスキー“１０００１０”を保持するノード１７５０ａがルートノードとなっており、その検査ビット位置は０である。ノード１７５０ａの左リンク１７４０ａにはノード１７５０ｂが接続され、右リンク１７４１ａにはノード１７５０ｆが接続されている。

【０００６】
ノード１７５０ｂの保持するインデックスキーは“０１００１１”であり、検査ビット位置２０３０ｂは1である。ノード１７５０ｂの左リンク１７４０ｂにはノード１７５０ｃが、右リンク１７４１ｂにはノード１７５０ｄが接続されている。ノード１７５０ｃが保持するインデックスキーは“０００１１１”、検査ビット位置は3である。ノード１７５０ｄが保持するインデックスキーは“０１１０１０”、検査ビット位置は2である。
【０００７】
ノード１７５０ｃから実線で接続された部分はノード１７５０ｃの左右のリンクポインタを示すものであり、点線の接続されていない左ポインタ１７４０ｃは、その欄が空欄であることを示している。点線の接続された右ポインタ１７４１ｃの点線の接続先は、ポインタの示すアドレスを表しており、今の場合ノード１７５０ｃを右ポインタが指定していることを表している。
【０００８】
ノード１７５０ｄの右ポインタ１７４１ｄはノード１７５０ｄ自身を指しており、左リンク１７４０ｄにはノード１７５０ｅが接続されている。ノード１７５０ｅの保持するインデックスキーは“０１００１０”、検査ビット位置は5である。ノード１７５０ｅの左ポインタ１７４０ｅはノード１７５０ｂを、右ポインタ１７４１ｅはノード１７５０ｅを指している。
【０００９】
また、ノード１７５０ｆの保持するインデックスキーは“１０１０１１”であり、検査ビット位置１７３０ｆは2である。ノード１７５０ｆの左リンク１７４０ｆにはノード１７５０ｇが、右リンク１７４１ｆにはノード１７５０ｈが接続されている。
【００１０】
ノード１７５０ｇの保持するインデックスキーは“１０００１１”であり、検査ビット位置１７３０ｇは5である。ノード１７５０ｇの左ポインタ１７４０ｇはノード１７５０ａを、右ポインタ１７４１ｇはノード１７５０ｇを指している。
【００１１】
ノード１７５０ｈの保持するインデックスキーは“１０１１００”であり、検査ビット位置１７３０ｈは3である。ノード１７５０ｈの左ポインタ１７４０ｈはノード１７５０ｆを、右ポインタ１７４１ｈはノード１７５０ｈを指している。
【００１２】
図1の例では、ルートノード１７５０ａからツリーを降りるにしたがって、各ノードの検査ビット位置が大きくなるように構成されている。
ある検索キーで検索を行うとき、ルートノードから順次各ノードに保持される検索キーの検査ビット位置を検査していき、検査ビット位置のビット値が1であるか0であるか判定を行い、1であれば右リンクをたどり、0であれば左リンクをたどる。そして、リンク先のノードの検査ビット位置がリンク元のノードの検査ビット位置より大きくなければ、すなわち、リンク先が下方でなく上方に戻れば（図1において点線で示されたこの逆戻りのリンクをバックリンクという）、リンク先のノードのインデックスキーと検索キーの比較を行う。比較の結果、等しければ検索成功であり、等しくなければ検索失敗であることが保証されている。
【００１３】
上記のように、パトリシアツリーを用いた検索処理では、必要なビットの検査だけで検索できること、キー全体の比較は1回ですむことなどのメリットがあるが、各ノードからの2つのリンクが必ずあることにより記憶容量が増大することや、バックリンクの存在による判定処理の複雑化、バックリンクにより戻ることで初めてインデックスキーと比較することによる検索処理の遅延及び追加削除等データメンテナンスの困難性などの欠点がある。
【００１４】
これらのパトリシアツリーの欠点を解消しようとするものとして、例えば下記特許文献

１に開示された技術がある。下記特許文献１に記載されたパトリシアツリーにおいては、下位の左右のノードは連続した領域に記憶することによりポインタの記憶容量を削減するとともに、次のリンクがバックリンクであるか否かを示すビットを各ノードに設けることにより、バックリンクの判定処理を軽減している。

【００１５】

しかしながら、下記特許文献１に開示されたものにおいても、１つのノードは必ずインデックスキーの領域とポインタの領域を占めること、下位の左右のノードを連続した領域に記憶するようにしてポインタを１つとしたため、例えば図１に示したパトリシアツリーの最下段の部分である左ポインタ１７４０ｃ、右ポインタ１７４１ｈ等の部分にもノードと同じ容量の記憶領域を割り当てる必要があるなど、記憶容量の削減効果はあまり大きいものではない。また、バックリンクによる検索処理の遅延の問題や追加削除等の処理が困難であることも改善されていない。

【００１６】

また、データベースからレコードを検索する場合、データベースのレコードと１対１で対応する項目の値をインデックスキーとするだけでなく、レコードを構成する任意の項目の値を検索キーとして検索を行うことが通常行われている。この項目の値はレコードによってユニークとは限らないことから、複数のレコードにおいて重複するキーによる検索が行われている。このような重複キーの取り扱いの一例が下記特許文献２に記載されている。

【特許文献１】特開２００１－３５７０７０号公報

【特許文献２】特開平１１－９６０５８号公報

【発明の開示】

【発明が解決しようとする課題】

【００１７】

そこで本発明の解決しようとする課題は、重複キーによる検索が可能であって、かつ、必要とする記憶容量が小さく、検索速度が高速であり、データメンテナンスの容易なデータ構造を備えたビット列検索装置及び検索方法を提供することである。

【００１８】

上述の従来の検索手法における問題点を解決するものとして、本出願人は、特願２００６－１８７８２７において、ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対からなるビット列検索に用いるツリーであって、ルートノードはツリーの始点を表すノードであって、該ツリーのノードが１つのときはリーフノード、ツリーのノードが２つ以上のときは前記ブランチノードであり、前記ブランチノードは、ビット列検索を行う検索キーの弁別ビット位置とリンク先のノード対の一方のノードである代表ノードの位置を示す位置情報を含み、前記リーフノードは検索対象のビット列からなるインデックスキーを含むカップルドノードツリーを用いたビット列検索を提案した。

【００１９】

上記出願においては、与えられたインデックスキーの集合からカップルドノードツリーを生成する方法と、カップルドノードツリーから単一のインデックスキーを検索する手法等の、カップルドノードツリーを用いた基本的な検索手法が示されている。

【００２０】

また、ビット列の検索には、最小値、最大値を求める、ある範囲の値のものを求める等の各種の検索要求が存在する。そこで、本出願人は、特願２００６－２９３６１９において、カップルドノードツリーの任意の部分木に含まれるインデックスキーの最大値／最小値を求める手法等を提案した。

【００２１】

さらに本出願人は、特願２００７－１１４９１５において、インデックスキーをカップルドノードツリーとは別の領域に配置し、リーフノードにはインデックスキーに代えてインデックスキーが配置された記憶領域の位置を示す情報を格納したカップルドノードツリ

ーとそれを用いた各種検索処理を提案した。

【００２２】

しかし、上述のカップルドノードツリーの構造は、インデックスキー同士のビット値の異なる位置である差分ビット位置に基づくものであるため、そのままでは重複キーを取り扱うことができない。

【００２３】

本発明は、このカップルドノードツリーを応用した高速な検索手法において、重複キーの取り扱いを可能とすることを目的とする。

【課題を解決するための手段】

【００２４】

本発明の一つの態様によれば、複数のキーをキー列として組み合わせ、キー列の最後尾のキーを重複のないユニークなものとすることにより、複数キーのキー列からなるインデックスキーをユニークキーとして構成する。そして、上述のカップルドノードツリーのリーフノードにはそのユニークキーが配置された記憶領域の位置を示す第２の位置情報を格納する。検索キーは、複数のキーをキー列として組み合わせ、キー列の最後尾のキーがユニークな検索キー列とする。ブランチノードには、ビット列検索を行う検索キー列中の、ビット列比較をするキーの位置情報を示すキー順序番号とそのキーの弁別ビット位置を持たせる。また、ブランチノードは、リンク先のノード対の一方のノードである代表ノードの位置を示す第１の位置情報を含む。

【００２５】

以上のようなデータ構造を有するカップルドノードツリーを用いて、検索キー列から、ブランチノードのキー順序番号の指すキーを取り出し、該キーの弁別ビット位置のビット値に応じて、リンク先のノート対のうちのいずれかのノードにリンクすることをリーフノードに至るまで行い、リーフノードに格納された記憶領域の位置を示す情報により、該記憶領域に配置されたインデックスキーを取得することにより、上記検索キー列による検索を実施する。

【発明の効果】

【００２６】

本発明によれば、重複キーの取り扱いが可能であり、かつ、より高速なビット列データの検索を行うことが可能となる。しかもビット列データの追加削除も容易に実行することができる。

【発明を実施するための最良の形態】

【００２７】

以下、本発明を実施するための最良の形態として、カップルドノードツリーを配列に格納する例について説明する。ブランチノードが保持するリンク先の代表ノードの位置を示すデータとして、記憶装置のアドレス情報とすることもできるが、ブランチノードあるいはリーフノードのうち占有する領域の記憶容量の大きい方を格納可能な配列要素からなる配列を用いることにより、ノードの位置を配列番号で表すことができ、代表ノードの位置を示す位置情報の情報量を削減することができる。

【００２８】

図２Ａは、本発明の一実施形態における配列に格納されたカップルドノードツリーの構成例を説明する図である。

図２Ａを参照すると、ノード１０１が配列１００の配列番号１０の配列要素に配置されている。ノード１０１はノード種別１０２、キー順序番号１０３ａ、弁別ビット位置１０３及び代表ノード番号１０４で構成されている。ノード種別１０２は０であり、ノード１０１がブランチノードであることを示している。キー順序番号１０３ａには０が格納されており、検索キー列の０番目の位置のキーについてビット列比較を行うことを示している。弁別ビット位置１０３には１が格納されている。代表ノード番号１０４にはリンク先のノード対の代表ノードの配列番号２０が格納されている。なお、以下では表記の簡略化の

ため、代表ノード番号に格納された配列番号を代表ノード番号ということもある。また、代表ノード番号に格納された配列番号をそのノードに付した符号あるいはノード対に付した符号で表すこともある。さらに、キー順序番号あるいは弁別ビット位置に格納された値を、単にキー順序番号あるいは弁別ビット位置ということもある。

【００２９】

配列番号２０の配列要素には、ノード対１１１の代表ノードであるノード［０］１１２が格納されている。そして隣接する次の配列要素（配列番号２０＋１）に代表ノードと対になるノード［１］１１３が格納されている。ノード［０］１１２はノード１０１と同様にブランチノードである。ノード［０］１１２のノード種別１１４には０が、キー順序番号１１５ａには１が、弁別ビット位置１１５には３が、代表ノード番号１１６には３０が格納されている。またノード［１］１１３は、ノード種別１１７と参照ポインタ１１８ａで構成されている。ノード種別１１７には１が格納されており、ノード［１］１１３がリーフノードであることを示している。参照ポインタ１１８ａには、インデックスキーの記憶領域を参照するポインタが格納されている。参照ポインタ１１８ａに格納されたデータは、上記の第２の位置情報の具体例である。以下では表記の簡略化のため、参照ポインタに格納されたデータのことも参照ポインタということがある。

【００３０】

パトリシアツリーについて先に述べたと同様に、インデックスキーと対応するレコードにアクセスするためのアクセス先情報も当然必要である。インデックスキーとアクセス先情報との対応づけは、例えば、インデックスキーを記憶している記憶領域に隣接する記憶領域に、当該インデックスキーに対応するアクセス先情報を記憶することによって行ってもよい。以下ではアクセス先情報については省略して説明する。

【００３１】

なお、代表ノードをノード［０］で表し、それと対になるノードをノード［１］で表すことがある。また、ある配列番号の配列要素に格納されたノードを、その配列番号のノードということがあり、ノードの格納された配列要素の配列番号を、ノードの配列番号ということもある。

【００３２】

配列番号３０及び３１の配列要素に格納されたノード１２２とノード１２３からなるノード対１２１の内容は省略されている。

ノード［０］１１２、ノード［１］１１３、ノード１２２、及びノード１２３の格納された配列要素にそれぞれ付された０あるいは１は、検索キー列で検索を行う場合にノード対のどちらのノードにリンクするかを示すものである。検索キー列のうち前段のブランチノードのキー順序番号の指すキー（以下、検索キーということがある。）の弁別ビット位置にあるビット値である０か１を代表ノード番号に加えた配列番号のノードにリンクする。

【００３３】

したがって、前段のブランチノードの代表ノード番号に、前段のブランチノードのキー順序番号の指す検索キーの弁別ビット位置にあるビット値を加えることにより、リンク先のノードが格納された配列要素の配列番号を求めることができる。

【００３４】

なお、上記の例では代表ノード番号をノード対の配置された配列番号のうち小さい方を採用しているが、大きいほうを採用することも可能であることは明らかである。

図２Ｂは、本実施形態に係るカップルドノードツリーのツリー構造と検索キー列（以下、インデックスキーということがある。）の格納領域を概念的に示す図である。

【００３５】

図２Ｂの（１）に示すのはカップルドノードツリーのツリー構造である。符号２１０ａで示すのがルートノードである。図示の例では、ルートノード２１０ａは配列番号２２０に配置されたノード対２０１ａの代表ノードとしている。

【００３６】

ツリー構造としては、ルートノード２１０ａの下にノード対２０１ｂが、その下層にノード対２０１ｃとノード対２０１ｆが配置され、ノード対２０１ｆの下層にはノード対２０１ｈとノード対２０１ｇが配置されている。ノード対２０１ｃの下にはノード対２０１ｄが、さらにその下にはノード対２０１ｅが配置されている。

【００３７】

各ノードの前に付された０あるいは１の符号は、図２Ａにおいて説明した配列要素の前に付された符号と同じである。検索キーの弁別ビット位置のビット値に応じてツリーをたどり、検索対象のインデックスキーに対応するリーフノードを見つけることになる。

【００３８】

図示された例では、ルートノード２１０ａのノード種別２６０ａは０でブランチノードであることを示し、キー順序番号２４０ａは０、弁別ビット位置２３０ａは０を示している。代表ノード番号は２２０ａであり、それはノード対２０１ｂの代表ノード２１０ｂの格納された配列要素の配列番号である。

【００３９】

ノード対２０１ｂはノード２１０ｂと２１１ｂで構成され、それらのノード種別２６０ｂ、２６１ｂはともに０であり、ブランチノードであることを示している。ノード２１０ｂのキー順序番号２４０ｂには０が、弁別ビット位置２３０ｂには１が格納され、リンク先の代表ノード番号にはノード対２０１ｃの代表ノード２１０ｃの格納された配列要素の配列番号２２０ｂが格納されている。

【００４０】

ノード２１０ｃのノード種別２６０ｃには１が格納されているので、このノードはリーフノードであり、したがって、参照ポインタ２５０ｃを含んでいる。参照ポインタ２５０ｃには、図示の例では、第１のキー２９０ｃと第２のキー２９０ｃ’からなるインデックスキーが格納されている記憶領域を参照するポインタを格納する。参照ポインタ２５０ｃに格納されたデータのことも参照ポインタといい、符号２８０ｃにより表す。他のリーフノードでも同様に、参照ポインタと参照ポインタに格納されたデータを同じ参照ポインタという語で表す。

【００４１】

図２Ｂの（２）には、複数のインデックスキーの記憶領域が連続して設けられる例を示し、それら連続した記憶領域全体をインデックスキーの記憶領域３１１として示したが、インデックスキーは連続した領域に格納されなくてもよい。また、リーフノード同士のツリー構造上での関係と、インデックスキーの記憶領域３１１におけるインデックスキーの配置順は無関係であってもよい。

【００４２】

ノード対２０１ｃの説明に戻ると、代表ノード２１０ｃと対になるもう一方のノード２１１ｃのノード種別２６１ｃは０、キー順序番号２４１ｃは１、弁別ビット位置２３１ｃは０であり、代表ノード番号にはノード対２０１ｄの代表ノード２１０ｄの格納された配列要素の配列番号２２１ｃが格納されている。

【００４３】

ノード２１０ｄのノード種別２６０ｄは０、キー順序番号２４０ｄは１、弁別ビット位置２３０ｄは２であり、代表ノード番号にはノード対２０１ｅの代表ノード２１０ｅの格納された配列要素の配列番号２２０ｄが格納されている。ノード２１０ｄと対になるノード２１１ｄのノード種別２６１ｄは１であり、参照ポインタ２５１ｄには、“０１１０１０”、“１０００”というキー列２９１ｄ、２９１ｄ’を格納した記憶領域を示す参照ポインタ２８１ｄが格納されている。

【００４４】

ノード対２０１ｅのノード２１０ｅ、２１１ｅのノード種別２６０ｅ、２６１ｅはともに１であり双方ともリーフノードであることを示す。ノード２１０ｅ、２１１ｅの参照ポインタ２５０ｅ、２５１ｅにはそれぞれ、“０１１０１０”、“０１０１”というキー列２９０ｅ、２９０ｅ’と、“０１１０１０”、“０１１０”というキー列２９１ｅ、２９

１ｅ’を格納した記憶領域への参照ポインタ２８０ｅ、２８１ｅが格納されている。

【００４５】

ノード対２０１ｂのもう一方のノードであるノード２１１ｂのキー順序番号２４１ｂには０、弁別ビット位置２３１ｂには２が格納され、リンク先の代表ノード番号にはノード対２０１ｆの代表ノード２１０ｆの格納された配列要素の配列番号２２１ｂが格納されている。

【００４６】

ノード対２０１ｆのノード２１０ｆ、２１１ｆのノード種別２６０ｆ、２６１ｆはともに０であり双方ともブランチノードである。それぞれのキー順序番号２４０ｆ、２４１ｆには０、１が、弁別ビット位置２３０ｆ、２３１ｆには５、２が格納されている。ノード２１０ｆの代表ノード番号にはノード対２０１ｇの代表ノード２１０ｇの格納された配列要素の配列番号２２０ｆが格納され、ノード２１１ｆの代表ノード番号にはノード対２０１ｈの代表ノードであるノード［０］２１０ｈの格納された配列要素の配列番号２２１ｆが格納されている。

【００４７】

ノード対２０１ｇのノード２１０ｇ、２１１ｇのノード種別２６０ｇ、２６１ｇはともに１であり双方ともリーフノードであることを示す。ノード２１０ｇ、２１１ｇのそれぞれの参照ポインタ２５０ｇ、２５１ｇには“１０００１０”、“０１００”というキー列２９０ｇ、２９０ｇ’と“１０００１１”、“００１１”というキー列２９１ｇ、２９１ｇ’を格納した記憶領域への参照ポインタ２８０ｇ、２８１ｇが格納されている。

【００４８】

また同じくノード対２０１ｈの代表ノードであるノード［０］２１０ｈとそれと対をなすノード［１］２１１ｈのノード種別２６０ｈ、２６１ｈはともに１であり双方ともリーフノードであることを示す。ノード２１０ｈ、２１１ｈのそれぞれの参照ポインタ２５０ｈ、２５１ｈには、“１０１１００”、“０００１” というキー列２９０ｈ、２９０ｈ’と“１０１１００”、“００１０” というキー列２９１ｈ、２９１ｈ’を格納した記憶領域への参照ポインタ２８０ｈ、２８１ｈが格納されている。

【００４９】

以下、上述のツリーからインデックスキー“１０１１０００００１０”を検索する処理の流れを簡単に説明する。上記インデックスキーは第１のキー“１０１１００”と第２のキー“００１０”からなるキー列である。キー順序番号及び弁別ビット位置は、左から０、１、２、・・・とする。

【００５０】

まず、ビット列“１０１１０００００１０”を検索キー列としてルートノード２１０ａから処理をスタートする。ルートノード２１０ａのキー順序番号２４０ａは０であり、弁別ビット位置２３０ａは０であるので、検索キー列の第１のキー“１０１１００”の弁別ビット位置が０のビット値をみると１である。そこで代表ノード番号の格納された配列番号２２０ａに１を加えた配列番号の配列要素に格納されたノード２１１ｂにリンクする。ノード２１１ｂのキー順序番号２４１ｂには０、弁別ビット位置２３１ｂには２が格納されているので、第１のキー“１０１１００”の弁別ビット位置が２のビット値をみると１であるから、代表ノード番号の格納された配列番号２２１ｂの配列要素に格納されたノード２１１ｆにリンクする。

【００５１】

ノード２１１ｆのキー順序番号２４１ｆには１、弁別ビット位置２３１ｆには２が格納されているので、第２のキー“００１０”の弁別ビット位置が２のビット値をみると１であるから、代表ノード番号の格納された配列番号２２１ｆの配列要素に格納されたノード２１１ｈにリンクする。

【００５２】

ノード２１１ｈのノード種別２６１ｈは１でありリーフノードであることを示しているので、参照ポインタ２８１ｈにより示される記憶領域を参照し、そこに格納されたインデ

ックスキーであるキー列２９１ｈ、２９１ｈ’を読み出す。このようにしてカップルドノードツリーを用いた検索が行われる。読み出されたキー列を検索キーと比較すると、上記の例の場合は一致していることが分かる。

【００５３】

なお、上述の説明では、検索キー列中のキーの位置を識別するキー順序番号を、左から０、１、２、・・・のようにキーの並びの順番に応じたキーの位置番号としたが、これに限ることなく、例えば検索キー列全体の先頭ビットからのオフセット値としたり、０と１を交互に用いることによりキー位置が切り替わったことを示すことにより、キーの位置の識別を可能とすることができる。

【００５４】

次に、図２Ｂを参照してカップルドノードツリーの構成の意味について説明する。

カップルドノードツリーの構成はインデックスキーの集合により規定される。図２Ｂの例で、ルートノードのキー順序番号２４０ａが０であるのは、インデックスキーの第１のキーには異なるものがあること、すなわち全ての第１のキーが重複しているのではない、ということを反映している。ルートノード２１０ａの弁別ビット位置２３０ａが０であるのは、インデックスキーの先頭のキーである第１のキーに０ビット目が０のものと１のものがあるからである。第１のキーの０ビット目が０のインデックスキーのグループはノード２１０ｂの下に分類され、０ビット目が１のインデックスキーのグループはノード２１１ｂの下に分類されている。

【００５５】

ノード２１１ｂの弁別ビット位置２３１ｂが２であるのは、その下位のリーフノード２１１ｈ、２１０ｈ、２１１ｇ、２１０ｇに対応するインデックスキーの第１のキーの１ビット目がすべて０で等しく、２ビット目で初めて異なるものがあるという、インデックスキーの集合の性質を反映している。

【００５６】

ノード２１１ｂの直近下位のノード対２０１ｆのノード２１１ｆのキー順序番号が１であるのは、ノード２１１ｆの下位のリーフノード２１１ｈ、２１０ｈに対応するインデックスキーの第１のキー２９０ｈ、２９１ｈが重複していることを反映しており、次に第２のキー２９０ｈ’、２９１ｈ’のビット列によりカップルノノードツリー上のノードの位置が決定されることを示している。そして、ノード２１１ｆの弁別ビット位置２３１ｆが２であるのは、第２のキー２９０ｈ’、２９１ｈ’は先頭から２ビット目で異なる値となっているからであり、そのビット値を反映してそれぞれのインデックスキーに対応した位置にリーフノード２１１ｈ、２１０ｈが配置されている。

【００５７】

一方、第１のキーの２ビット目が０であるインデックスキーでは３ビット目も４ビット目も等しく５ビット目で異なるのでノード２１０ｆの弁別ビット位置２３０ｆには５が格納される。インデックスキーには第１のキーの５ビット目が１のものと０のものがそれぞれ１つしかないことから、ノード２１０ｆのリンク先のノード２１０ｇ、２１１ｇはリーフノードとなり、参照ポインタ２５０ｇと２５１ｇには、キー列２９０ｇ、２９０ｇ’とキー列２９１ｇ、２９１ｇ’を格納した記憶領域を指す参照ポインタ２８０ｇ、２８１ｇがそれぞれ格納されている。

【００５８】

仮にインデックスキーの集合に“１０１１０００００１”の代わりに“１０１１０１０００１”か“１０１１１０００１”が含まれていたとしても、第１のキーの３ビット目までと第２のキーは“１０１１０００００１”と等しいので、ノード２１０ｈの参照ポインタ２８０ｈにより示される記憶領域に格納されるインデックスキーの値が変わるだけで、ツリー構造自体は変わることはない。しかし、“１０１１０００００１”に加えて“１０１１０００００００”が含まれていると、ノード２１０ｈはブランチノードとなり、その弁別ビット位置は３になる。

【００５９】

以上説明したように、カップルドノードツリーの構造は、インデックスキーの集合に含まれる各インデックスキーの各ビット位置のビット値により決定される。

そしてさらにいえば、異なるビット値となるビット位置ごとにビット値が“１”のノードとビット値が“０”のノードとに分岐していることから、ノード［1］側とツリーの深さ方向を優先させてリーフノードをたどると、それらに格納されたインデックスキーは、ノード２１１ｈに対応するインデックスキー“１０１１０００００１０”、ノード２１０ｈに対応するインデックスキー“１０１１０００００１”、・・・、ノード２１０ｃに対応するインデックスキー“０００１１１０１１１”となり降順にソートされている。

【００６０】

すなわち、カップルドノードツリーにおいては、インデックスキーはソートされてツリー上に配置されている。

検索キー列で検索するときはインデックスキーがカップルドノードツリー上に配置されたルートをたどることになり、例えば検索キー列が“１０１１０００００１”であればノード２１０ｈに到達することができる。また、上記説明からも想像がつくように、“１０１１０００００００”を検索キー列とした場合でもノード２１０ｈにたどり着き、参照ポインタ２８０ｈにより示される記憶領域に格納されたインデックスキーが検索結果キー列として得られる。

【００６１】

また、例えば“１００１００１００１”で検索した場合でも、ノード２１０ａ、２１１ｂ、２１０ｆのリンク経路では検索キーの列の第１のキーの３ビット目と４ビット目は使われることがなく、第１のキーの“１００１００”の５ビット目が０なので、“１０００１００１００”で検索した場合と同様にノード２１０ｇに到達することになる。このように、カップルドノードツリーに格納されたインデックスキーのビット構成に応じたキー順序番号と弁別ビット位置を用いて分岐が行われる。

なお、図２Ｂに示す例は、インデックスキーが第１のキーと第２のキーからなるキー列あったが、３つ以上のキーからなるキー列であっても、キー順序番号と弁別ビット位置に応じて分岐が行われる検索が可能なカップルドノードツリーが得られることは、上記の説明から明らかである。

【００６２】

図３は、本発明を実施するためのハードウェア構成例を説明する図である。

本発明の検索装置による検索処理及びデータメンテナンスは中央処理装置３０２及びキャッシュメモリ３０３を少なくとも備えたデータ処理装置３０１によりデータ格納装置３０８を用いて実施される。カップルドノードツリーが配置される配列３０９と検索中にたどるノードが格納された配列要素の配列番号を記憶する探索経路スタック３１０とインデックスキーの記憶領域３１１を有するデータ格納装置３０８は、主記憶装置３０５または外部記憶装置３０６で実現することができ、あるいは通信装置３０７を介して接続された遠方に配置された装置を用いることも可能である。図２Ａの配列１００は配列３０９の一例である。また、図２Ｂと同様に、インデックスキーの記憶領域３１１は連続した領域のように図示されているが、不連続の領域でもよいことは当然である。なお、カップルドノードツリーは配列に配置されるとして説明するため、探索経路スタック３１０には検索中にたどるノードが格納された配列要素の配列番号を記憶すると説明したが、一般的には、ノードの格納された記憶領域のアドレス等のノードの位置を示す情報が記憶される。

【００６３】

図３の例示では、主記憶装置３０５、外部記憶装置３０６及び通信装置３０７が一本のバス３０４によりデータ処理装置３０１に接続されているが、接続方法はこれに限るものではない。また、主記憶装置３０５をデータ処理装置３０１内のものとすることもできるし、探索経路スタック３１０を中央処理装置３０２内のハードウェアとして実現することも可能である。あるいは、配列３０９は外部記憶装置３０６に、探索経路スタック３１０を主記憶装置３０５に持つなど、使用可能なハードウェア環境、インデックスキー集合の大きさ等に応じて適宜ハードウェア構成を選択できることは明らかである。

【００６４】
また、特に図示されてはいないが、処理の途中で得られた各種の値を後の処理で用いるためにそれぞれの処理に応じた一時記憶領域が用いられることは当然である。以下の説明では、先に述べたキー順序番号等の場合と同様に、一時記憶領域に格納されたあるいは設定された値を一時記憶領域の名前で呼ぶことがある。
【００６５】
図３に示したとおり、カップルドノードツリーのノードを格納した配列要素からなる配列３０９と、インデックスキーの記憶領域３１１とは別の領域である。したがって、リーフノードを格納した配列要素にインデックスキーが含まれる場合に比べて、図３の構成では、一般に１つの配列要素に必要な記憶領域の量が少ない。つまり、カップルドノードツリーを格納する配列３０９からインデックスキーの記憶領域３１１を分離することによって、キャッシュメモリ３０３へのカップルドノードツリーの読み込みにおいて１キャッシュブロックあたりに格納されるノード数を増やすことが可能となる。それにより、後述する検索処理等においてキャッシュミスの頻度が減って処理がより高速に行われるようになる。
【００６６】
次に、本発明の一実施態様に係るカップルドノードツリーを用いた基本的な操作である、検索、挿入、削除について順に詳しく説明する
図４は、一実施形態におけるビット列の検索処理を示すフローチャートである。
【００６７】
まず、ステップＳ４０１ａで、検索開始ノードの配列番号を取得する。取得された配列番号に対応する配列は、カップルドノードツリーを構成する任意のノードを格納したものである。検索開始ノードの指定は、後に説明する各種応用検索において行われる。
【００６８】
取得された検索開始ノードの配列番号は、図示しない検索開始ノード設定エリアに設定されるが、この検索開始ノード設定エリアは、先に述べた「処理の途中で得られた各種の値を後の処理で用いるためにそれぞれの処理に応じた一時記憶領域」の一つである。以下の説明では、「図示しない検索開始ノード設定エリアに設定する」のような表現に変えて、「検索開始ノードの配列番号を得る。」、「検索開始ノードとして設定する」あるいは単に「検索開始ノードに設定する」のように記述することもある。
【００６９】
次にステップＳ４０１ｂで、後にキー順序番号の値を退避する一時記憶領域である退避キー順序番号に初期値を設定する。インデックスキーの先頭のキー列のキー順序番号を０としているので、初期値にはマイナスの値を設定する。
【００７０】
次に、ステップＳ４０２で、探索経路スタックに取得された配列番号を格納し、ステップＳ４０３で、その配列番号に対応する配列要素を参照すべきノードとして読み出す。そして、ステップＳ４０４で、読み出したノードから、ノード種別を取り出し、ステップＳ４０５で、ノード種別がブランチノードであるか否かを判定する。
【００７１】
ステップＳ４０５の判定において、読み出したノードがブランチノードである場合は、ステップＳ４０６ａに進む。ステップＳ４０６ａでは、ノードからキー順序番号を取り出し、次のステップＳ４０６ｂで、ステップＳ４０６ａで取り出したキー順序番号が退避キー順序番号と一致するか判定する。
【００７２】
キー順序番号が退避キー順序番号と一致する場合はステップＳ４０６に移行し、一致しない場合はステップＳ４０６ｃに進んで、検索キー列から、ステップＳ４０６ａで取り出したキー順序番号が指すキーを取り出し、検索キーに設定する。
【００７３】
次にステップＳ４０６ｄで退避キー順序番号にステップＳ４０６ａで取り出したキー順

序番号を設定し、ステップＳ４０６に進む。
ステップＳ４０６では、ステップＳ４０３で読み出したノードから弁別ビット位置を取り出し、更に、ステップＳ４０７で、取り出した弁別ビット位置に対応するビット値を検索キーから取り出す。そして、ステップＳ４０８で、ステップＳ４０３で読み出したノードから代表ノード番号を取り出し、ステップＳ４０９で、検索キーから取り出したビット値と代表ノード番号とを加算し、新たな配列番号として、ステップＳ４０２に戻る。
【００７４】
以降、ステップＳ４０５の判定においてリーフノードと判定されてステップＳ４１０ａに進むまで、ステップＳ４０２からステップＳ４０９までの処理を繰り返す。ステップＳ４１０ａでは、リーフノードから参照ポインタを取り出し、検索を終了する。
【００７５】
次に、図５～図８Ａによりカップルドノードツリーにおけるノード挿入処理を説明する。図５～図７が通常の挿入処理を説明するものであり、図８Ａはルートノードの挿入処理を説明するものである。ルートノードの挿入処理と通常の挿入処理により、カップルドノードツリーが生成されることから、ノード挿入処理の説明はカップルドノードツリーの生成処理の説明でもある。
【００７６】
図５は挿入処理の前段である検索処理の処理フローを示す図であり、図４に示した検索処理において、挿入キー列を検索キー列とし、検索開始ノードをルートノードとしたものに相当する。
【００７７】
まず、ステップＳ５０１ａで検索開始ノードにルートノードの配列番号を設定し、ステップＳ５０１ｂで検索キー列に挿入キー列を設定する。挿入キー列は、挿入処理の前提条件として、予めインデックスキーの格納領域のポインタを取得して、該格納領域に格納されているものとする。挿入キー列の末尾のキーは、インデックスキー全体でユニークな値をとるものとする。
【００７８】
次にステップＳ５１０ａにおいて、検索キー列により検索開始ノードより図４に示す検索処理を行い、参照ポインタを取得し、ステップＳ５１０ｂにおいて該参照ポインタの指すキー列を取り出して比較キー列に設定する。
【００７９】
次にステップＳ５１０ｃにおいて、挿入キー列のキーと比較キー列のキーを順次比較する。比較の結果、比較キー列が挿入キー列に完全に一致するかしないかの情報を出力する。完全に一致しない場合には、最初に不一致となったキーの位置をキー順序番号として設定するとともに、挿入キー列における該キー順序番号のキーを挿入キーに設定し、比較キー列における該キー順序番号のキーを比較キーに設定する。ステップＳ５１０ｃの詳細は、後に図８Ｂを参照して説明する。
【００８０】
次にステップＳ５１１ａにおいて、ステップＳ５１０ｃでの比較の結果、挿入キー列のキーと比較キー列のキーが全て等しいか判定し、等しければ挿入キー列は既にカップルドノードツリーの参照ポインタが指す記憶領域に存在するのであるから、挿入は失敗となり、処理を終了する。等しくなければ次の処理、図６のステップＳ５１２以下の処理に進む。
【００８１】
図６は、挿入するノード対のための配列要素を準備する処理を説明する処理フロー図である。
ステップＳ５１２において、配列から空きのノード対を求め、そのノード対のうち代表ノードとなるべき配列要素の配列番号を取得する。
【００８２】
ステップＳ５１３ａに進み、ステップＳ５１０ｃで得た挿入キーと比較キーの大小を比

較し、挿入キーが大きいときは値１を小さいときは値０のブール値を得る。
ステップＳ５１４に進み、ステップＳ５１２で得た代表ノードの配列番号にステップＳ５１３で得たブール値を加算した配列番号を得る。

【００８３】
ステップＳ５１５に進み、ステップＳ５１２で得た代表ノードの配列番号にステップＳ５１３で得たブール値の論理否定値を加算した配列番号を得る。
ステップＳ５１４で得た配列番号は、挿入キー列をインデックスキーとして格納する記憶領域への参照ポインタを持つリーフノードが格納される配列要素の配列番号であり、ステップＳ５１５で得た配列番号は、そのリーフノードとノード対を成すノードが格納される配列要素のものである。

【００８４】
つまり、前段の検索処理で得られたリーフノードに対応するインデックスキーと挿入キー列の大小により、挿入されるノード対のうちどちらのノードに、挿入キー列への参照ポインタを保持するリーフノードが格納されるかが決定される。

【００８５】
例えば図２Ｂのカップルドノードツリーに挿入キー列“０１１０１１１００１”を挿入する場合、検索結果のインデックスキーは、ノード２１１ｄに対応する、キー列“０１１０１０１０００”になる。挿入キー列と検索結果のキー列のキーを順次比較すると第１のキーで不一致となり、第１のキー同士の大小比較によりブール値が求められ、今の例では挿入キー列の第１のキーの方が大きいのでブール値１が得られ、挿入されるノード対の代表ノード番号に１を加えた配列要素に挿入キー列への参照ポインタを保持するリーフノードが格納される。一方、インデックスキー“０１１０１０１０００”への参照ポインタ２８１ｄは、大小比較で得られたブール値を論理反転した値を代表ノード番号に加算した配列番号の配列要素に格納される。

【００８６】
その際、検索結果のインデックスキーの第１のキー２９１ｄ“０１１０１０”と挿入キー列の第１のキー“０１１０１１”とは５ビット目で異なることから、ノード２１１ｄは、弁別ビット位置を５とし、代表ノード番号を挿入されたノード対の代表ノードの配列番号とするブランチノードとなる。

【００８７】
また図２Ｂのカップルドノードツリーに“０１１００１１０１０”を挿入しようとする場合も、検索結果は、ノード２１１ｄに対応する、“０１１０１０１０００”という値のインデックスキーになる。この場合にも挿入キー列と検索結果のキー列の第１のキーは不一致であり、この場合には挿入キー列の第１のキーの方が小さいのでブール値０が得られ、挿入されるノード対の代表ノード番号に０を加えた配列要素に挿入キー列への参照ポインタを保持するリーフノードが格納される。そして、インデックスキーの第１のキー２９１ｄ“０１１０１０”と挿入キー列の第１のキー“０１１００１”とは４ビット目で異なることから、ノード２１１ｄは、弁別ビット位置を４とし、代表ノード番号を挿入されたノード対の代表ノードの配列番号とするブランチノードとなる。

【００８８】
次にステップＳ５１６で、ステップＳ５１０ｃで得た挿入キーと比較キーのビット列比較を例えば排他的論理和で行い、差分ビット列を得る。
ステップＳ５１７に進み、ステップＳ５１６で得た差分ビット列から、上位０ビット目から見た最初の不一致ビットのビット位置（以下、差分ビット位置ということがある。）を得る。この処理は、例えばプライオリティエンコーダを有するＣＰＵではそこに差分ビット列を入力し、不一致のビット位置を得ることができる。また、ソフト的にプライオリティエンコーダと同等の処理を行い最初の不一致ビットのビット位置を得ることも可能である。

【００８９】
ステップＳ５１７に引き続き、図７に示すステップＳ５１８以下の処理を行う。

図７は図６で準備された配列要素にノードを格納するとともにその挿入位置を求め、既存のノードの内容を変更して挿入処理を完成させる処理フローを示す図である。

【００９０】

ステップＳ５１８～ステップＳ５２３までの処理は、挿入するノード対のカップルドノードツリー上の位置を求める処理であり、ステップＳ５２４以下の処理は各ノードにデータを設定して挿入処理を完成させる処理である。

【００９１】

ステップＳ５１８において、探索経路スタックのスタックポインタがルートノードの配列番号を指しているか判定する。指していればステップＳ５２４に移行し、指していなければステップＳ５１９に進む。

【００９２】

ステップＳ５１９において、探索経路スタックのスタックポインタを１つ戻してそこにスタックされている配列番号を取り出す。

ステップＳ５２０に進み、ステップＳ５１９で取り出した配列番号の配列要素を配列からノードとして読み出す。

【００９３】

ステップＳ５２０ａに進み、ステップＳ５２０で読み出したノードから、キー順序番号を取り出し、ステップＳ５２０ｂにおいて、ステップＳ５２０ａで取り出したキー順序番号“Ａ”とステップＳ５１０ｃで得たキー順序番号“Ｂ”の大小を比較する。

【００９４】

Ａ＞Ｂであれば、ステップＳ５１８にもどり、Ａ＝Ｂであれば、ステップＳ５２１に進み、Ａ＜Ｂであれば、ステップＳ５２３に進む。

ステップＳ５２１では、ステップＳ５２０で読み出したノードから、弁別ビット位置を取り出し、ステップＳ５２２に進み、ステップＳ５２１で取り出した弁別ビット位置がステップＳ５１７で得た差分ビット位置より上位の位置関係か判定する。ここで上位の位置関係とは、ビット列のより左側の位置、すなわちビット位置の値が小さい位置であることとする。

【００９５】

ステップＳ５２２の判定結果が否定であれば、ステップＳ５１８に戻り、ステップＳ５１８での判定が肯定になるか、ステップＳ５２０ｂにおいてＡ＜Ｂと判定されるか、ステップＳ５２２での判定が肯定になるまで繰り返す。ステップＳ５２２での判定が肯定になると、ステップＳ５２３に進む。

【００９６】

ステップＳ５２３では、探索経路スタックのスタックポインタを１つ進め、ステップＳ５２４以下の処理に移行する。

上記ステップＳ５１８～ステップＳ５２３で説明した処理は、挿入するノード対の挿入位置を決定するために、探索経路スタックに格納されているブランチノードのキー順序番号（Ａ）とステップＳ５１０ｃで取得したキー順序番号（Ｂ）の相対的位置関係を調べ、Ａ＜Ｂであればブランチノードのリンク先を挿入するノード対の挿入位置とし、Ａ＝Ｂであれば、挿入するインデックスキー（挿入キー）と検索により取得されたインデックスキー（比較キー）の間のビット列比較で異なるビット値となる差分ビット位置と探索経路スタックに格納されているブランチノードの弁別ビット位置との相対的位置関係を調べ、弁別ビット位置が上位となるブランチノードの次のブランチノードのリンク先を挿入するノード対の挿入位置とするものである。

【００９７】

また、探索経路スタックを逆にたどりルートノードに至った場合は、ルートノードのリンク先が挿入位置となる。

例えば図２Ｂのカップルドノードツリーに“１１１００００００００”を挿入するとき、検索結果のインデックスキーは、ノード２１０ｈに対応する“１０１１０００００１”になる。この例の場合、キー順序番号（Ａ）は１（キー順序番号２４１ｆ）、キー順序番号

（Ｂ）は０であるからＡ＞Ｂとなり、リンク経路をノード２１１ｂに戻ると、キー順序番号２４１ｂは０なのでＡ＝Ｂとなる。今の例の場合、挿入キー“１１１０００”と比較キー“１０１１００”の差分ビット位置は１であり、弁別ビット位置２３１ｂは２なので、さらにルートノード２１０ａまでさかのぼる。

【００９８】

ルートノード２１０ａの弁別ビット位置２３０ａは０で差分ビット位置１より上位になるのでスタックポインタは２２０ａを指す。したがって、挿入位置はノード２１１ｂのリンク先である。ブランチノード２１１ｂの弁別ビット位置は、ノード対が挿入されると後に述べるように差分ビット位置の値１になる。

【００９９】

なお、Ａ＜Ｂとなるのは、例えば第１のキーに重複が無いインデックスキーに対して、重複する第１のキーを含む挿入キー列が挿入される場合であり、図２Ｂの例では、挿入キー列“１０００１０１００１”が与えられるとキー順序番号（Ａ）はキー順序番号２４０ｆであって値が０であり、キー順序番号（Ｂ）の値は１であって、Ａ＜Ｂが成立し、ノード２１０ｇが挿入位置になり、挿入されるノード対の直近上位のブランチノードになる。

【０１００】

次に、ステップＳ５２４以下の各ノードにデータを設定して挿入処理を完成させる処理について説明する。

ステップＳ５２４では探索経路スタックからスタックポインタの指す配列番号を取り出す。

【０１０１】

ステップＳ５２５ｄに進み、ステップＳ５１４で得た配列番号の指す配列要素の、ノード種別にリーフを、参照ポインタに挿入キー列のポインタを書き込む。

ステップＳ５２６に進み、配列からステップＳ５２４で得た配列番号の配列要素を読み出す。

【０１０２】

次にステップＳ５２７において、ステップＳ５１５で得た配列番号の配列要素にステップＳ５２６で読み出した内容を書き込む。

最後にステップＳ５２８ａにおいて、ステップＳ５２４で得た配列番号の指す配列要素のノード種別にブランチを、キー順序番号にステップＳ５１０ｃで得たキー順序番号を、弁別ビット位置にステップＳ５１７で得たビット位置を、代表ノード番号にステップＳ５１２で得た配列番号を書き込み、処理を終了する。

【０１０３】

上述の図２Ｂのカップルドノードツリーに挿入キー列“１１１００００００００”を挿入する例では、ステップＳ５２５ｄにおいて、取得された空ノード対のノード［１］を挿入キー列“１１１００００００００”への参照ポインタを保持するリーフノードとし、ステップＳ５２７において、ノード［０］にノード２１１ｂの内容を書き込む。そして、ステップＳ５２８ａにおいて、ノード２１１ｂのノード種別に０、キー順序番号に０、弁別ビット位置にビット列比較により得られた差分ビット位置１を格納し、代表ノード番号には取得されたノード対の代表ノードが格納される配列要素の配列番号が格納される。

【０１０４】

図８Ａは、本発明の一実施形態におけるルートノードの挿入処理を含むリーフノードの挿入処理全体の処理フローを説明する図である。

ステップＳ５５１において、取得することを求められたカップルドノードツリーのルートノードの配列番号が登録済みであるか判定される。登録済みであれば、図５～図７を用いて説明した通常の挿入処理が行われる。

【０１０５】

ステップＳ５５１での判定が登録済みでなければ、まったく新しいカップルドノードツリーの登録、生成が始まることになる。この場合にも、挿入キー列は、挿入処理の前提条件として、予めインデックスキーの格納領域のポインタを取得して、該格納領域に格納さ

れているものとする。

【０１０６】

まず、ステップＳ５５２において、配列から空きのノード対を求め、そのノード対のうち代表ノードとなるべき配列要素の配列番号を取得する。次にステップＳ５５３において、ステップＳ５５２で得た配列番号に０を加えた配列番号を求める。（実際には、ステップＳ５５２で取得した配列番号に等しい。）次にステップＳ５５４ｄにおいて、ステップＳ５５３で得た配列番号の配列要素すなわち挿入するルートノードに対応する配列要素の、ノード種別にリーフを、参照ポインタに挿入キー列のポインタを書き込む。そしてステップＳ５５６では、ステップＳ５５３で取得したルートノードの配列番号を登録して処理を終了する。

【０１０７】

先にも述べたように、インデックスキーの集合があるとき、そこから順次インデックスキーを取り出し、図８Ａ及び図５～図７の処理を繰り返すことにより、インデックスキーの集合に対応した本発明のカップルドノードツリーを構築することができることは明らかである。

【０１０８】

次に、図８Ｂを参照して、先に述べた図５に示すステップＳ５１０ｃで実行される、本発明の一実施形態におけるキー列の比較処理の処理フローを説明する。

図に示すように、ステップＳ１０１において、キー番号に初期値として値“０”を設定する。

【０１０９】

ステップＳ１０２において、列中の全てのキーは処理済みか判定し、処理済であれば完全一致の情報を出力して処理を終了し、処理済でなければステップＳ１０３に進む。

ステップＳ１０３では、挿入キー列から、キー番号の指すキーを取り出し、挿入キーとして設定する。

【０１１０】

ステップＳ１０４に進み、図５に示すステップＳ５１０ｂで設定した比較キー列から、キー番号の指すキーを取り出し、比較キーとして設定する。

次にステップＳ１０５に進み、挿入キーは比較キーと一致するか判定する。一致すればステップＳ１０６でキー番号に設定した値を更新してステップＳ１０２に戻り、一致しなければ、ステップＳ１０７に移行してキー番号をキー順序番号として設定するとともに、非完全一致を出力して処理を終了する。

【０１１１】

次に図９、図１０を参照して、本発明の一実施形態におけるカップルドノードツリーから特定のインデックスキーに対応するリーフノードを削除する処理フローを説明する。

図９は、削除処理の前段である検索処理の処理フローを示す図であり、図４に示した検索処理において、削除キー列を検索キー列とし、検索開始ノードをルートノードとしたものに相当する。

【０１１２】

まず、ステップＳ９０１ａで検索開始ノードにルートノードの配列番号を設定し、ステップＳ９０１ｂで検索キー列に削除キー列を設定する。

次にステップＳ９１０ａにおいて、検索キー列により検索開始ノードより図４に示す検索処理を行い、参照ポインタを取得し、ステップＳ９１０ｂにおいて該参照ポインタの指すキー列を取り出して比較キー列に設定する。

【０１１３】

次にステップＳ９１０ｃにおいて、削除キーとして、削除キー列の末尾のキーを設定し、ステップＳ９１０ｄにおいて、インデックスキーとして、比較キー列の末尾のキーを設定する。

【０１１４】

ステップＳ９１１においてステップＳ９１０ｃで設定した削除キーとステップＳ９１０

ｄで設定したインデックスキーを比較し、等しくなければ削除するインデックスキーはカップルドノードツリーに存在しないのであるから、削除は失敗となり、処理を終了する。等しければ次の処理、図１０のステップＳ９１２以下の処理に進む。なお、前述のキー列中の末尾のキーを比較するのは、末尾のキーが重複のないユニークキーであることによる。

【０１１５】

図１０は、削除処理の後段の処理フローを説明する図である。

まず、ステップＳ９１２で探索経路スタックに２つ以上の配列番号が格納されているか判定する。２つ以上の配列番号が格納されていないということは、言い換えれば１つだけで、その配列番号はルートノードの格納された配列要素のものである。その場合はステップＳ９１８に移行し、ステップＳ９０１ａで得たルートノードの配列番号に係るノード対を削除して、処理を終了する。

【０１１６】

ステップＳ９１２において探索経路スタックに２つ以上の配列番号が格納されていると判定されたときはステップＳ９１３ａに進み、ステップＳ９１０ａで実行した図４に示す検索処理のステップＳ４０８で得た代表ノード番号に、同じくステップＳ９１０ａで実行した図４に示す検索処理のステップＳ４０７で得たビット値を反転した値を加算した配列番号を得る。この処理は、削除対象のリーフノードと対をなすノードの配置された配列番号を求めるものである。

【０１１７】

次にステップＳ９１４において、ステップＳ９１３で得た配列番号の配列要素の内容を読み出し、ステップＳ９１５において探索経路スタックのスタックポインタを１つ戻して配列番号を取り出す。

【０１１８】

次にステップＳ９１６に進み、ステップＳ９１４で読み出した配列要素の内容をステップＳ９１５で得た配列番号の配列要素に上書きする。この処理は、削除対象のリーフノードへのリンク元であるブランチノードを上記リーフノードと対をなすノードに置き換えるものである。

【０１１９】

続くステップＳ９１７において、ステップＳ９１０ａで実行した図４に示す検索処理のステップＳ４０８で得た代表ノード番号に係るノード対を削除し、処理を終了する。

図１１Ａ及び図１１Ｂは、図２Ｂに例示したカップルドノードツリーにおいて“０１１０１０１０００”を削除キー列として削除処理を行う例を説明する図である。

【０１２０】

図１１Ａに示したカップルドノードツリーは、ノード対２０１ｆ以下のノードは記載を省略している。削除キー列“０１１０１０１０００”は、第１のキー“０１１０１０”と第２のキー“１０００”から構成されるキー列であり、一時記憶領域である削除キー２７０に格納されている。

【０１２１】

探索経路スタック３１０には配列番号が格納されており、そのスタックポインタは配列番号２２１ｃ＋１を指している。図中太枠で囲まれたノードが検索処理でたどられたノードであり、その配列番号がルートノード２１０ａのものからリーフノード２１１ｄのものまで探索経路スタック３１０に積まれている。

【０１２２】

削除キー列“０１１０１０１０００”による検索処理においては、まず始めにルートノード２１０ａの配列番号２２０を取得し、それを探索経路スタック３１０に格納する。ルートノード２１０ａのキー順序番号２４０ａが０であるので、削除キー列の第１のキー“０１１０１０”が取り出されて検索キーに設定され、弁別ビット位置２３０ａが０であり、検索キーのビット位置０のビット値が０であるので、代表ノード番号２２０ａにビット値０を加えた配列番号２２０ａが探索経路スタック３１０に格納される。

【０１２３】
次に配列番号２２０ａの指すノード２１０ｂが読み出され、ブランチノードであることが判定される。キー順序番号２４０ｂは０でキー順序番号２４０ａと変わりは無いので、先に設定した検索キーから、弁別ビット位置２３０ｂの値１に対応するそのビット位置のビット値１を取得し、代表ノード番号２２０ｂに加えて配列番号２２０ｂ＋１を得てそれを探索経路スタック３１０に格納する。

【０１２４】
次にノード２１１ｃが読み出され、キー順序番号２４１ｃが１であって、１つ更新されたので、削除キー列の第２のキー“１０００”が検索キーとして設定される。弁別ビット位置２３１ｃが０であり、検索キーのビット位置０のビット値が１であるので、代表ノード番号２２１ｃに１を加えた配列番号２２１ｃ＋１が図に示すとおり探索経路スタック３１０に格納されている。

【０１２５】
配列番号が２２１ｃ＋１の配列要素に格納されたノード２１１ｄのノード種別２６１ｄは１であり、リーフノードであることを示している。このリーフノードに対応するインデックスキー（キー列２９１ｄ、２９１ｄ’）は、参照ポインタ２８１ｄにより示される記憶領域に格納されている。その記憶領域はインデックスキーの記憶領域３１１の一部である。そこで参照ポインタ２８１ｄの参照するインデックスキーの末尾のキーである第２のキー２９１ｄ’を取り出すとその値は“１０００”であり、削除キー２７０に格納された第２のキーと一致している。

【０１２６】
図１１Ａに示した状態において、削除対象のノード２１１ｄと対をなすノード２１０ｄの内容が読み出され、その内容が、探索経路スタック３１０のスタックポインタを１つ戻したところに格納されている配列番号２２０ｂ＋１の配列要素（ノード２１１ｃ）に書き込まれる。その後ノード対２０１ｄを削除する。ノード対が削除された配列要素は空となり、再利用可能となる。

【０１２７】
図１１Ｂに示したカップルドノードツリーは、削除処理の終了後のものである。ノード２１１ｃのノード種別２６１ｃ、キー順序番号２４１ｃ、弁別ビット位置２３１ｃ、代表ノード番号２２１ｃには、括弧書きで示すように、ノード２１０ｄに格納されていた値がそのまま格納されている。また、探索経路スタック３１０のスタックポインタは配列番号２２０ｂ＋１を指している。

【０１２８】
次に、図１２Ａ及び図１２Ｂを参照して挿入処理の具体例を説明する。
図１２Ａに示すのは、ビット列“０１０００００１”、“０００１００１０”、“０００００１１”をインデックスキーとして参照する参照ポインタ１２８１ｂ、１２８１ｃ、１２８０ｃを持つカップルドノードツリーである。

【０１２９】
参照ポインタ１２８１ｂの指す記憶領域には、第１のキー１２９１ｂ“０１００”と第２のキー１２９１ｂ’“０００１”からなるキー列が格納されている。同様に、参照ポインタ１２８１ｃの指す記憶領域には、第１のキー１２９１ｃ“０００１”と第２のキー１２９１ｃ’“００１０”からなるキー列が格納され、参照ポインタ１２８０ｃの指す記憶領域には、第１のキー１２９０ｃ“００００”と第２のキー１２９０ｃ’“００１１”からなるキー列が格納されている。

【０１３０】
これから挿入しようとする挿入キー列は図示の例では“０００００１００”である。挿入キー列は、先に述べたように、インデックスキーの格納領域のポインタ１２８１ｄを取得して、ポインタ１２８１ｄの指す領域に格納されているものとし、挿入キー列の末尾のキーである第２のキー“０１００”はインデックスキー全体でユニークな値をとることを

前提としている。

【０１３１】

図示のツリーはノード対１２０１ａ、１２０１ｂ、１２０１ｃで構成されている。

ノード対１２０１ａの代表ノードはルートノード１２１０ａであり、キー順序番号には０、弁別ビット位置には１が保持されている。ノード対１２０１ａの下位のノード対１２０１ｂの代表ノード１２１０ｂはブランチノードであり、キー順序番号には０、弁別ビット位置には３が保持され、代表ノード１２１０ｂと対になるノード１２１１ｂはリーフノードであり、キー列１２９１ｂ、１２９１ｂ’への参照ポインタ１２８１ｂが保持されている。ブランチノードであるノード１２１０ｂはノード対１２０１ｃにリンクしている。

【０１３２】

ノード対１２０１ｃを構成するノード１２１０ｃと１２１１ｃはともにリーフノードであり、それぞれキー列１２９０ｃ、１２９０ｃ’とキー列１２９１ｃ、１２９１ｃ’への参照ポインタ１２８０ｃ、１２８１ｃが格納されている。

【０１３３】

挿入キー列の第１のキー１２９１ｄは、参照ポインタ１２８０ｃの指す記憶領域に格納されたキー列の第１のキーと重複している。したがって図示の例の場合、挿入キー列で検索をすると、参照ポインタ１２８０ｃの格納されたリーフノード１２１０ｃに至り、比較キーとして、第２のキー１２９０ｃ’が設定され、挿入キー列の第２のキー１２９０ｄ’を挿入キーとして大小関係の判定とビット列比較が行われる。すると、挿入キーが比較キーより大きく、最初の不一致ビットの位置は１となる。

【０１３４】

図１２Ｂは、挿入キー列“０００００１００”を挿入したカップルドノードツリーを示す図である。新たなノード対１２０１ｄがノード対１２０１ｃの下位に挿入されている。ノード対１２０１ｄのノード［１］１２１１ｄが挿入キー列のポインタを参照ポインタ１２８１ｄとして含むリーフノードとして生成され、ノード［０］１２１０ｄには、図１２Ａに示すノード１２１０の内容が書き込まれている。そして、図１２Ｂに示すブランチノード１２１０ｃのキー順序番号には第２のキーであることを示す１が、弁別ビット位置には挿入キーと比較キーの最初の不一致ビットの位置である１が格納され、代表ノード番号には、ノード対１２０１の代表ノード１２１０ｄの配置された配列要素の配列番号が格納されている。

【０１３５】

以上本発明を実施するための最良の形態について詳細に説明したが、本発明の実施の形態はそれに限ることなく種々の変形が可能であることは当業者に明らかである。例えばリーフノードが、インデックスキーを格納した記憶領域の位置を示す情報に代えてインデックスキー自体を含むようにすることが可能であることは、当業者に自明である。また、検索キー列中のキーの位置を識別するキー順序番号を、左から０、１、２、・・・のようにキーの並びの順番に応じたキーの位置番号としたが、これに限ることなく、例えば検索キー列全体の先頭ビットからのオフセット値としたり、０と１を交互に用いることによりキー位置が切り替わったことを示すことにより、キーの位置の識別が可能であることは、当業者に自明である。

【０１３６】

また、本発明のビット列検索方法を実行する装置が、カップルドノードツリーを格納する記憶手段と図４に示した処理をコンピュータに実行させるプログラムによりコンピュータ上に構築可能なことは明らかである。

【０１３７】

さらに、図５～図７、図８Ａ、図８Ｂに示した挿入処理とその均等物をコンピュータに実行させるプログラムにより、本発明の挿入方法が実現可能であり、図９及び図１０に示した削除処理とその均等物をコンピュータに実行させるプログラムにより、本発明の削除方法が実現可能であることも明らかである。そして、それらのプログラムにより、ブランチノードとリーフノードの識別手段、ブランチノードの弁別ビット位置に応じてリンク先

のノード対のどちらかにリンクする手段等がコンピュータ上に実現される。

【０１３８】

したがって、上記プログラム、及びプログラムを記憶したコンピュータ読み取り可能な記憶媒体は、本発明の実施の形態に含まれる。さらに、本発明のカップルドノードツリーのデータ構造も、本発明の実施の形態に含まれる。

【０１３９】

以上詳細に説明した、本発明が提供する新しいデータ構造であるカップルドノードツリーを用いることにより、重複キーを取り扱うことができるとともに、より高速なビット列データの検索を行うことが可能となる。しかもビット列データの追加削除も容易に実行することができる。

【図面の簡単な説明】

【０１４０】

【図１】従来の検索で用いられるパトリシアツリーの一例を示す図である。

【図２Ａ】配列に格納されたカップルドノードツリーの構成例を説明する図である。

【図２Ｂ】カップルドノードツリーのツリー構造を概念的に示す図である。

【図３】本発明を実施するためのハードウェア構成例を説明する図である。

【図４】本発明の一実施形態における検索処理を説明するフローチャートである。

【図５】本発明の一実施形態における挿入処理の前段である検索処理の処理フローを説明する図である。

【図６】本発明の一実施形態における挿入処理における挿入するノード対のための配列要素を準備する処理フローを説明する図である。

【図７】ノード対を挿入する位置を求め、ノード対の各ノードの内容を書き込んで挿入処理を完成させる処理フローを説明する図である。

【図８Ａ】本発明の一実施形態におけるルートノードの挿入処理を含むリーフノードの挿入処理全体の処理フローを説明する図である。

【図８Ｂ】本発明の一実施形態におけるキー列の比較処理の処理フローを説明する図である。

【図９】本発明の一実施形態における削除処理の前段である検索処理の処理フローを説明する図である。

【図１０】本発明の一実施形態における削除処理の後段の処理フローを説明する図である。

【図１１Ａ】削除処理前のカップルドノードツリーと削除キー列を例示して説明する図である。

【図１１Ｂ】削除処理後のカップルドノードツリーを説明する図である。

【図１２Ａ】挿入処理前のカップルドノードツリーと挿入キー列を例示して説明する図である。

【図１２Ｂ】挿入処理後のカップルドノードツリーを説明する図である。

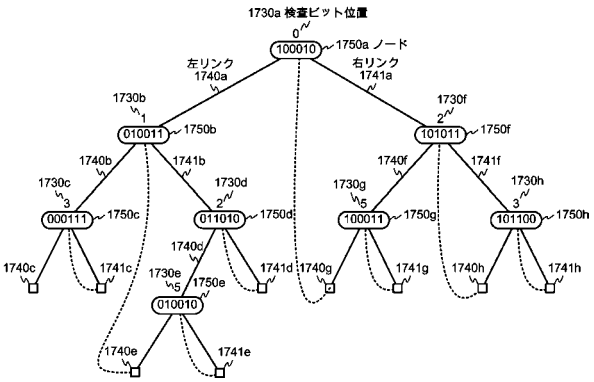
【符号の説明】

【０１４１】

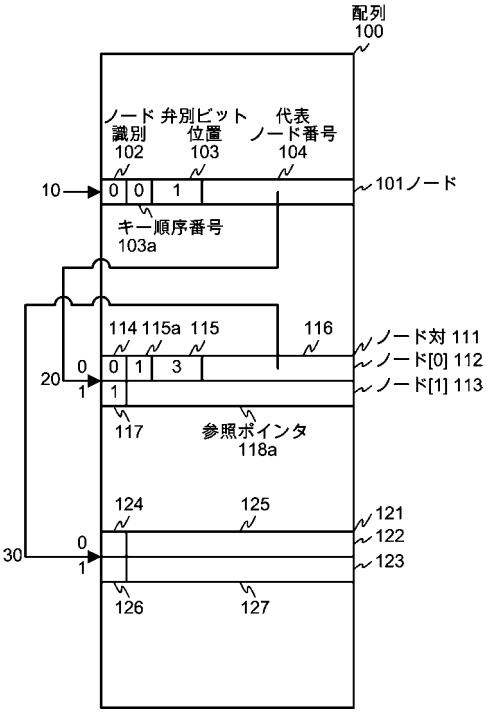
１０、２０、３０　配列番号
１００　配列
１０１　ノード
１０２　ノード種別
１０３ａ　キー順序番号
１０３　弁別ビット位置
１０４　代表ノード番号
１１１　ノード対
１１２　ノード［０］、代表ノード
１１３　ノード［１］、代表ノードと対をなすノード
１１８ａ　参照ポインタ

３０１　　データ処理装置
３０２　　中央処理装置
３０３　　キャッシュメモリ
３０４　　バス
３０５　　主記憶装置
３０６　　外部記憶装置
３０７　　通信装置
３０８　　データ格納装置
３０９　　配列
３１０　　探索経路スタック
３１１　　インデックスキーの格納領域

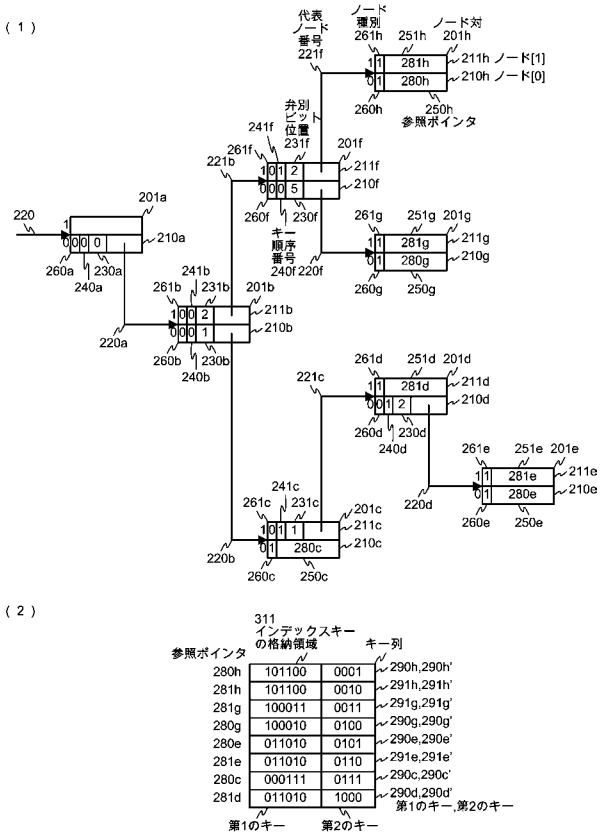
【図１】



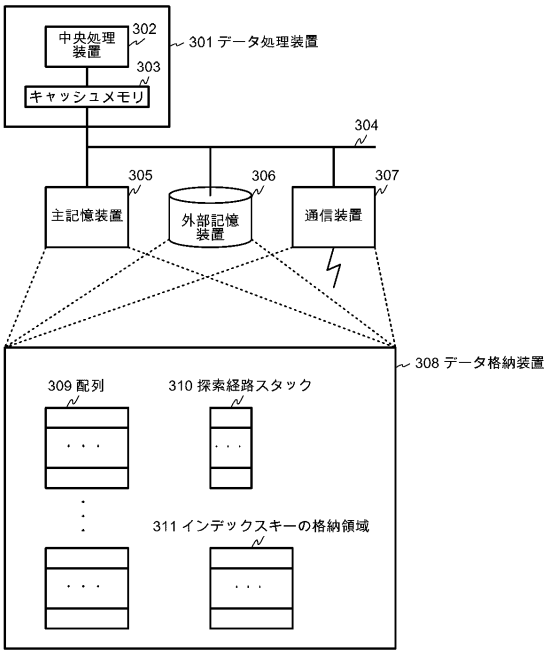
【図２Ａ】



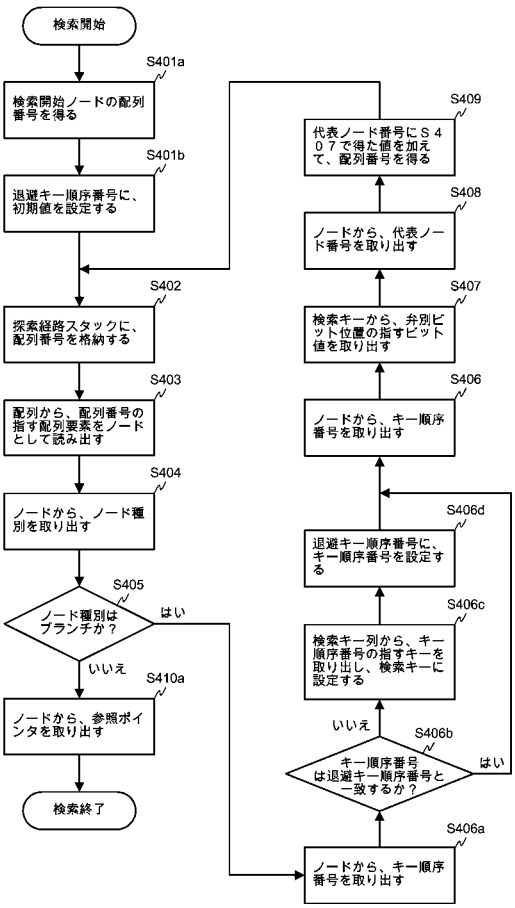
【図２Ｂ】


（１）
代表ノード番号
ノード種別
ノード対
221f
261h
251h
201h
281h
280h
211h ノード[1]
210h ノード[0]
260h
250h
参照ポインタ
弁別ビット位置
241f
231f
201f
261f
221b
211f
210f
260f
230f
キー順序番号
240f
220f
201a
220
210a
260a
230a
240a
241b
231b
201b
211b
210b
220a
260b
230b
240b
261g
251g
201g
281g
280g
211g
210g
260g
250g
221c
261d
251d
201d
281d
211d
210d
260d
230d
240d
261e
251e
201e
281e
280e
211e
210e
220d
260e
250e
241c
231c
261c
201c
211c
210c
220b
260c
250c
（２）
311
インデックスキーの格納領域
参照ポインタ
キー列
280h 101100 0001 290h,290h'
281h 101100 0010 291h,291h'
281g 100011 0011 291g,291g'
280g 100010 0100 290g,290g'
280e 011010 0101 290e,290e'
281e 011010 0110 291e,291e'
280c 000111 0111 290c,290c'
281d 011010 1000 290d,290d'
第1のキー
第2のキー
第1のキー,第2のキー

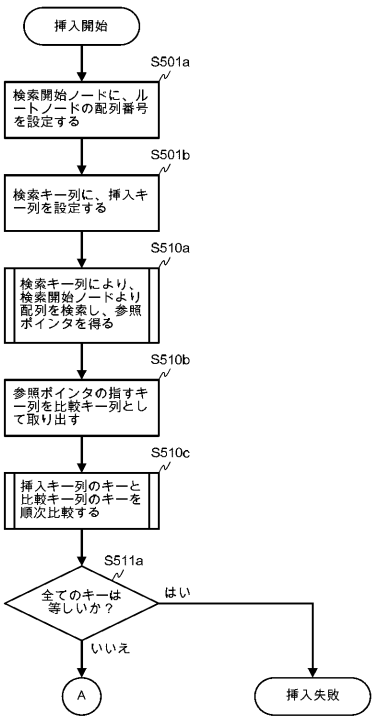
【図３】


中央処理装置
302
301 データ処理装置
303
キャッシュメモリ
304
305
306
307
主記憶装置
外部記憶装置
通信装置
308 データ格納装置
309 配列
310 探索経路スタック
311 インデックスキーの格納領域

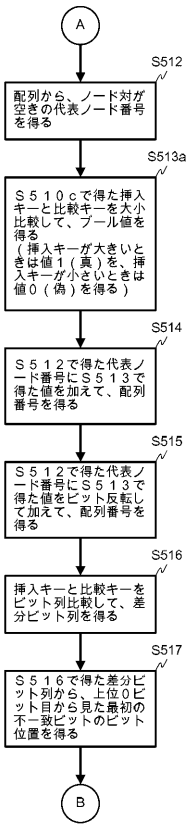
【図４】


検索開始
S401a
検索開始ノードの配列番号を得る
S401b
退避キー順序番号に、初期値を設定する
S402
探索経路スタックに、配列番号を格納する
S403
配列から、配列番号の指す配列要素をノードとして読み出す
S404
ノードから、ノード種別を取り出す
S405
ノード種別はブランチか？
はい
いいえ
S410a
ノードから、参照ポインタを取り出す
検索終了
S409
代表ノード番号にＳ４０７で得た値を加えて、配列番号を得る
S408
ノードから、代表ノード番号を取り出す
S407
検索キーから、弁別ビット位置の指すビット値を取り出す
S406
ノードから、キー順序番号を取り出す
S406d
退避キー順序番号に、キー順序番号を設定する
S406c
検索キー列から、キー順序番号の指すキーを取り出し、検索キーに設定する
いいえ
S406b
キー順序番号は退避キー順序番号と一致するか？
はい
S406a
ノードから、キー順序番号を取り出す

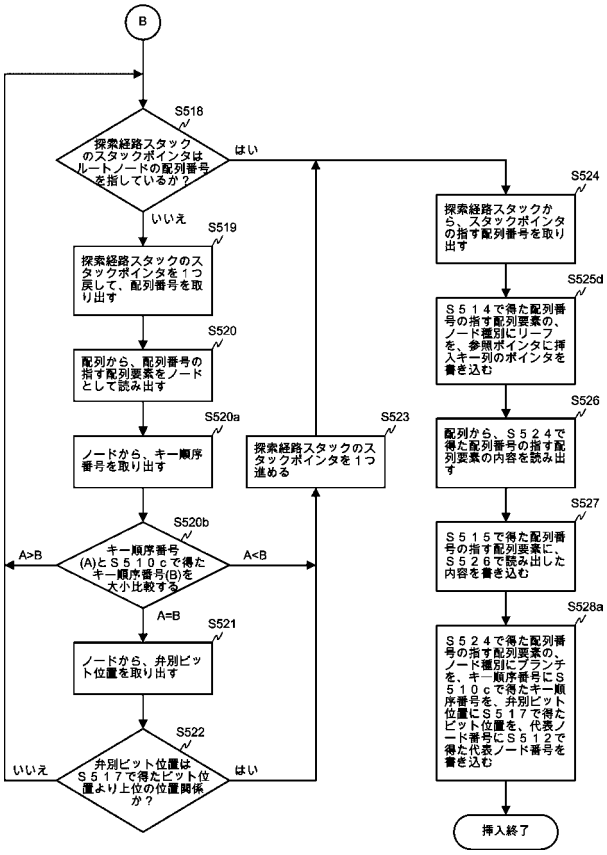
【図５】


挿入開始
S501a
検索開始ノードに、ルートノードの配列番号を設定する
S501b
検索キー列に、挿入キー列を設定する
S510a
検索キー列により、検索開始ノードより配列を検索し、参照ポインタを得る
S510b
参照ポインタの指すキー列を比較キー列として取り出す
S510c
挿入キー列のキーと比較キー列のキーを順次比較する
S511a
全てのキーは等しいか？
はい
いいえ
A
挿入失敗

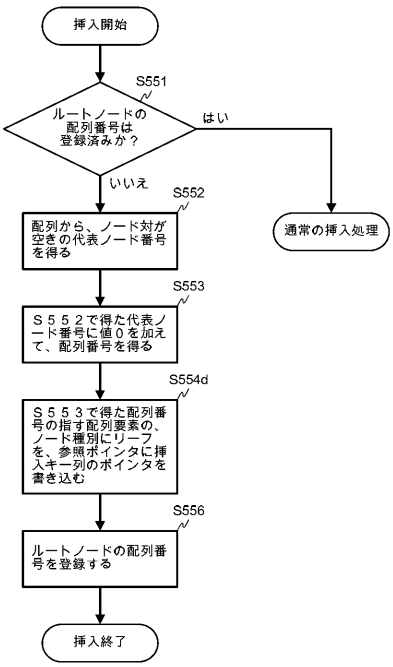
【図６】


A
S512
配列から、ノード対が空きの代表ノード番号を得る
S513a
Ｓ５１０ｃで得た挿入キーと比較キーを大小比較して、ブール値を得る（挿入キーが大きいときは値１（真）を、挿入キーが小さいときは値０（偽）を得る）
S514
Ｓ５１２で得た代表ノード番号にＳ５１３で得た値を加えて、配列番号を得る
S515
Ｓ５１２で得た代表ノード番号にＳ５１３で得た値をビット反転して加えて、配列番号を得る
S516
挿入キーと比較キーをビット列比較して、差分ビット列を得る
S517
Ｓ５１６で得た差分ビット列から、上位０ビット目から見た最初の不一致ビットのビット位置を得る
B

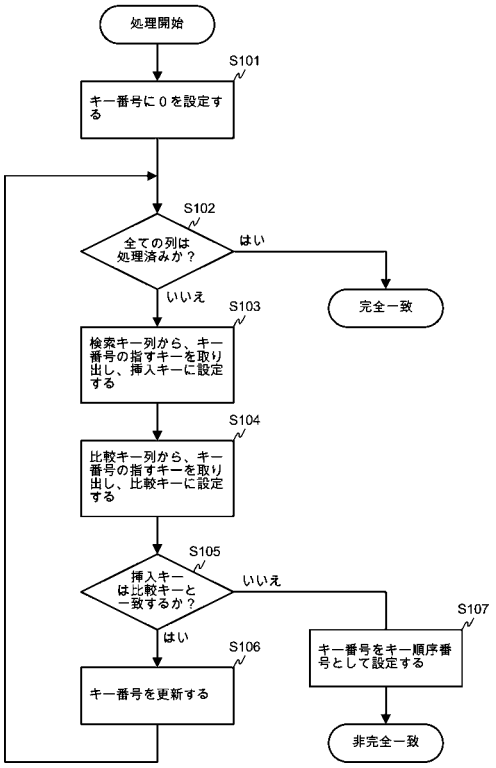
【図７】


B
S518
探索経路スタックのスタックポインタはルートノードの配列番号を指しているか？
はい
いいえ
S519
探索経路スタックのスタックポインタを１つ戻して、配列番号を取り出す
S520
配列から、配列番号の指す配列要素をノードとして読み出す
S520a
ノードから、キー順序番号を取り出す
S520b
キー順序番号(A)とＳ５１０ｃで得たキー順序番号(B)を大小比較する
A>B
A<B
A=B
S521
ノードから、弁別ビット位置を取り出す
S522
弁別ビット位置はＳ５１７で得たビット位置より上位の位置関係か？
いいえ
はい
S523
探索経路スタックのスタックポインタを１つ進める
S524
探索経路スタックから、スタックポインタの指す配列番号を取り出す
S525d
Ｓ５１４で得た配列番号の指す配列要素の、ノード種別にリーフを、参照ポインタに挿入キー列のポインタを書き込む
S526
配列から、Ｓ５２４で得た配列番号の指す配列要素の内容を読み出す
S527
Ｓ５１５で得た配列番号の指す配列要素に、Ｓ５２６で読み出した内容を書き込む
S528a
Ｓ５２４で得た配列番号の指す配列要素の、ノード種別にブランチを、キー順序番号にＳ５１０ｃで得たキー順序番号を、弁別ビット位置にＳ５１７で得たビット位置を、代表ノード番号にＳ５１２で得た代表ノード番号を書き込む
挿入終了

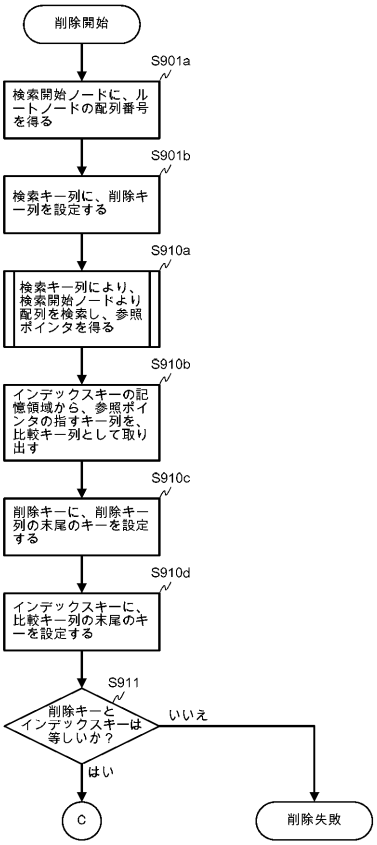
【図８Ａ】


挿入開始
S551
ルートノードの配列番号は登録済みか？
はい
いいえ
通常の挿入処理
S552
配列から、ノード対が空きの代表ノード番号を得る
S553
Ｓ５５２で得た代表ノード番号に値０を加えて、配列番号を得る
S554d
Ｓ５５３で得た配列番号の指す配列要素の、ノード種別にリーフを、参照ポインタに挿入キー列のポインタを書き込む
S556
ルートノードの配列番号を登録する
挿入終了

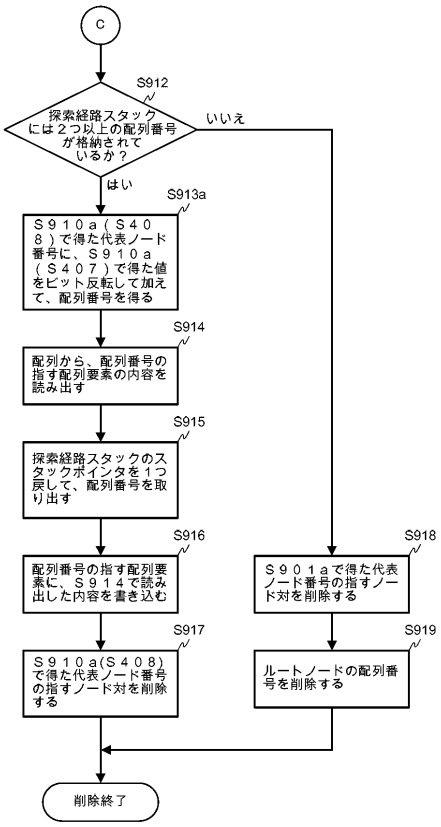
【図８Ｂ】


処理開始
S101
キー番号に０を設定する
S102
全ての列は処理済みか？
はい
いいえ
完全一致
S103
検索キー列から、キー番号の指すキーを取り出し、挿入キーに設定する
S104
比較キー列から、キー番号の指すキーを取り出し、比較キーに設定する
S105
挿入キーは比較キーと一致するか？
いいえ
はい
S106
キー番号を更新する
S107
キー番号をキー順序番号として設定する
非完全一致

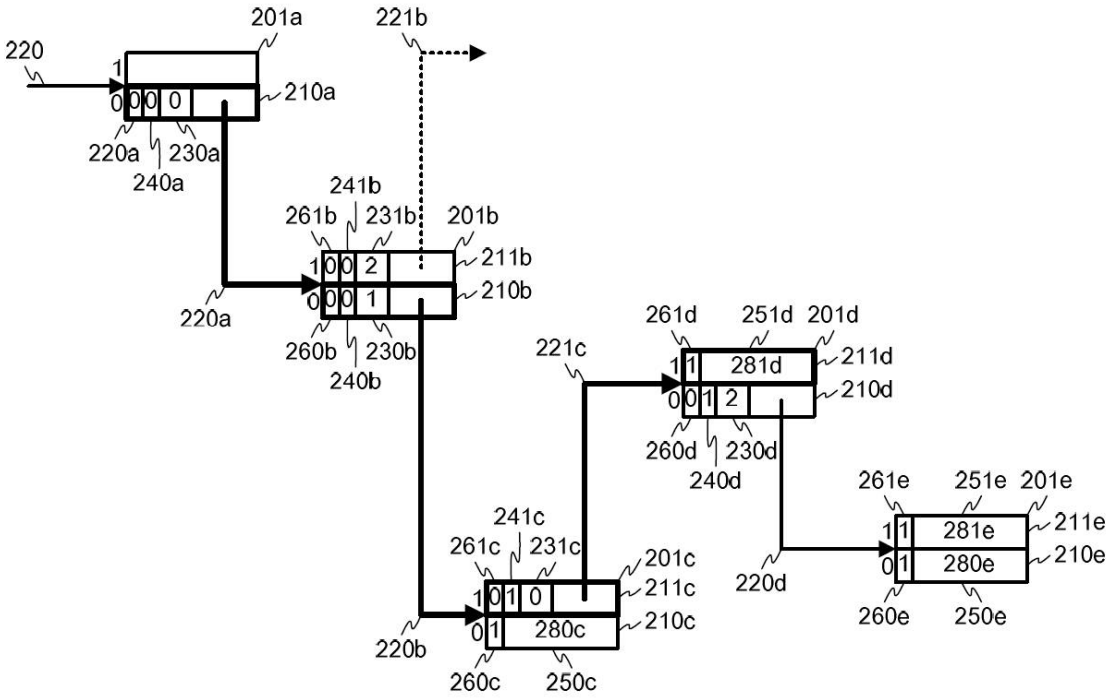
【図９】

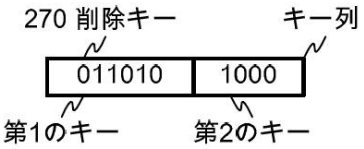

削除開始
S901a
検索開始ノードに、ルートノードの配列番号を得る
S901b
検索キー列に、削除キー列を設定する
S910a
検索キー列により、検索開始ノードより配列を検索し、参照ポインタを得る
S910b
インデックスキーの記憶領域から、参照ポインタの指すキー列を、比較キー列として取り出す
S910c
削除キーに、削除キー列の末尾のキーを設定する
S910d
インデックスキーに、比較キー列の末尾のキーを設定する
S911
削除キーとインデックスキーは等しいか？
いいえ
はい
C
削除失敗

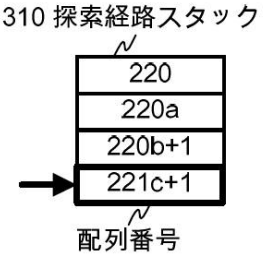
【図１０】

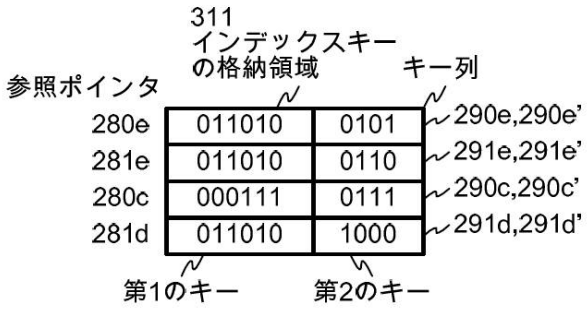

C
S912
探索経路スタックには２つ以上の配列番号が格納されているか？
いいえ
はい
S913a
Ｓ９１０ａ（Ｓ４０８）で得た代表ノード番号に、Ｓ９１０ａ（Ｓ４０７）で得た値をビット反転して加えて、配列番号を得る
S914
配列から、配列番号の指す配列要素の内容を読み出す
S915
探索経路スタックのスタックポインタを１つ戻して、配列番号を取り出す
S916
配列番号の指す配列要素に、Ｓ９１４で読み出した内容を書き込む
S917
Ｓ９１０ａ(Ｓ４０８)で得た代表ノード番号の指すノード対を削除する
S918
Ｓ９０１ａで得た代表ノード番号の指すノード対を削除する
S919
ルートノードの配列番号を削除する
削除終了

【図１１Ａ】

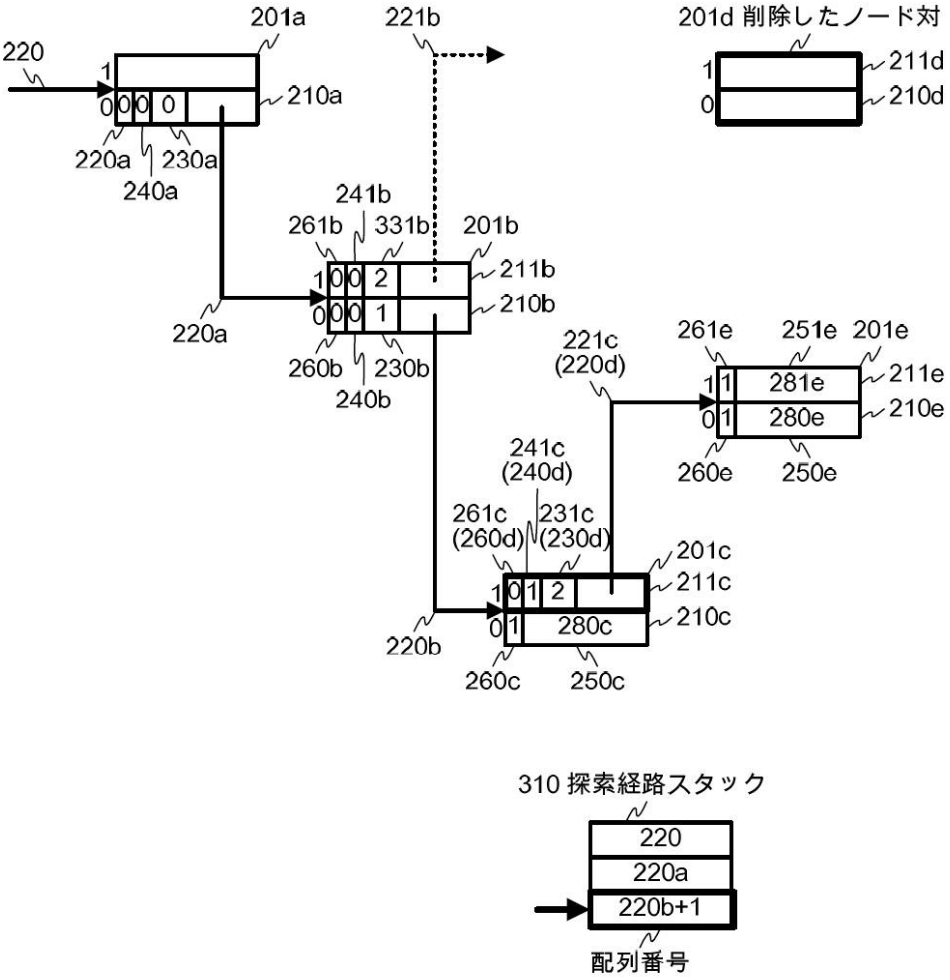

220
201a
221b
1
0 0 0 0
210a
220a
230a
240a
241b
261b
231b
201b
1 0 0 2
211b
0 0 0 1
210b
220a
260b
230b
240b
221c
261d
251d
201d
1 1 281d
211d
0 0 1 2
210d
260d
230d
240d
261e
251e
201e
1 1 281e
211e
0 1 280e
210e
220d
260e
250e
241c
261c
231c
201c
1 0 1 0
211c
0 1 280c
210c
220b
260c
250c


270 削除キー
キー列
011010
1000
第1のキー
第2のキー

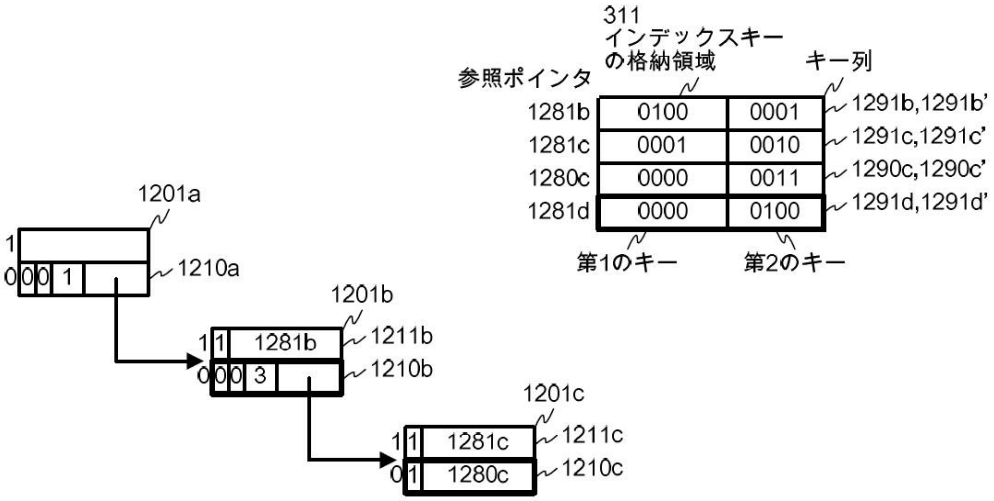

310 探索経路スタック
220
220a
220b+1
221c+1
配列番号


311
インデックスキー
の格納領域
キー列
参照ポインタ
280e
011010
0101
290e,290e'
281e
011010
0110
291e,291e'
280c
000111
0111
290c,290c'
281d
011010
1000
291d,291d'
第1のキー
第2のキー

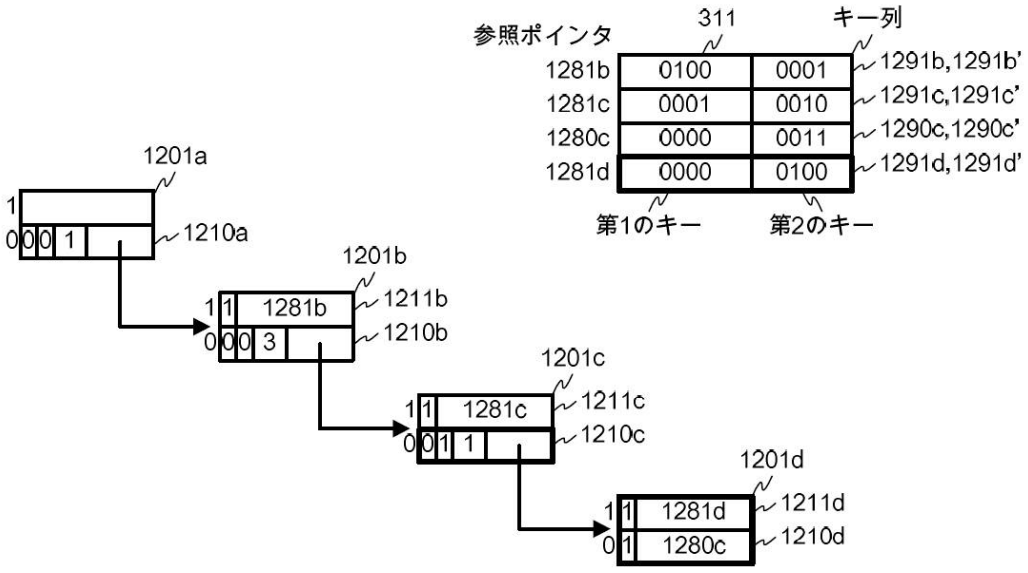
【図１１Ｂ】


220
201a
221b
201d 削除したノード対
1
0 0 0 0
210a
220a
230a
240a
211d
210d
241b
261b
331b
201b
1 0 0 2
0 0 0 1
211b
210b
220a
260b
230b
240b
221c
(220d)
261e
251e
201e
1 1 281e
0 1 280e
211e
210e
260e
250e
241c
(240d)
261c
(260d)
231c
(230d)
201c
1 0 1 2
0 1 280c
211c
210c
220b
260c
250c
310 探索経路スタック
220
220a
220b+1
配列番号

【図１２Ａ】


311
インデックスキー
の格納領域
参照ポインタ
キー列
1281b 0100 0001 1291b,1291b'
1281c 0001 0010 1291c,1291c'
1280c 0000 0011 1290c,1290c'
1281d 0000 0100 1291d,1291d'
第1のキー
第2のキー
1201a
1
000 1
1210a
1201b
1 1 1281b
000 3
1211b
1210b
1201c
1 1 1281c
0 1 1280c
1211c
1210c

【図１２Ｂ】


311
キー列
参照ポインタ
1281b 0100 0001 1291b,1291b'
1281c 0001 0010 1291c,1291c'
1280c 0000 0011 1290c,1290c'
1281d 0000 0100 1291d,1291d'
第1のキー
第2のキー
1201a
1
000 1
1210a
1201b
1 1 1281b
000 3
1211b
1210b
1201c
1 1 1281c
001 1
1211c
1210c
1201d
1 1 1281d
0 1 1280c
1211d
1210d

フロントページの続き

(56)参考文献 国際公開第２００８／００４３３５（WO，Ａ１）
特開平１－２３９６３２（ＪＰ，Ａ）
特開平５－２６５８２１（ＪＰ，Ａ）
特開平７－３６７５５（ＪＰ，Ａ）
特開平１１－９６０５８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１７／３０