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

特許第4514771号
(P4514771)

(45)発行日　平成22年7月28日(2010.7.28)　(24)登録日　平成22年5月21日(2010.5.21)

(51)Int.Cl.　　F I
G06F 17/30 (2006.01)　　G06F 17/30 414A
G06F 12/00 (2006.01)　　G06F 17/30 419A
　　G06F 12/00 520A

請求項の数 12　(全 31 頁)

(21)出願番号　特願2007-132289 (P2007-132289)
(22)出願日　平成19年5月18日 (2007.5.18)
(65)公開番号　特開2008-287533 (P2008-287533A)
(43)公開日　平成20年11月27日 (2008.11.27)
審査請求日　平成22年1月20日 (2010.1.20)

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】カップルドノードツリーの最長一致／最短一致検索装置、検索方法及びプログラム

(57)【特許請求の範囲】
【請求項１】
ビット列からなる最長一致検索キーにより、検索対象であるビット列からなるインデックスキーが格納されたツリーのデータ構造に基づいて、前記インデックスキーのうち前記最長一致検索キーとの最長一致条件を満たすインデックスキーを検索する最長一致検索装置において、
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、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み、
前記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該ブランチノードに含まれる弁別ビット位置の検索キー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ツリーの任意の部分木の前記検索キーによる検索の結果である検索結果キーとするように構成されたカップルド

ノードツリーと、
前記カップルドノードツリーの前記ルートノードを前記検索開始ノードとし、指定された最長一致検索キーを前記検索キーとして、前記ブランチノードにおいて該ブランチノードに含まれる弁別ビット位置の検索キー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リーフノードに至るまで繰り返し、前記ルートノードから該リーフノードまでの経路を記憶しながら該リーフノードに格納されたインデックスキーを前記検索結果キーとして取得する検索手段と、
前記最長一致検索キーと前記検索結果キーのビット列を比較して、ビット値が一致しない不一致ビットの位置のうち最も上位の位置である差分ビット位置を取得する差分ビット位置取得手段と、
前記差分ビット位置がビット列の最上位以外の位置のとき、記憶された前記経路を参照して最長一致ノードを設定する最長一致ノード設定手段であって、
前記ルートノードが前記リーフノードの場合または前記ルートノードが前記ブランチノードであり該ルートノードの前記弁別ビット位置が前記差分ビット位置よりも下位の位置の場合は、前記ルートノードを前記最長一致ノードとして設定し、
その他の場合は、前記経路上のノードであって前記弁別ビット位置が前記差分ビット位置よりも上位にある前記ブランチノードのうち前記弁別ビット位置が最下位の前記ブランチノードの次に前記検索手段によって記憶された前記ブランチノードまたは前記リーフノードを前記最長一致ノードとして設定する最長一致ノード設定手段と、
を備えることを特徴とする最長一致検索装置。
【請求項２】
前記最長一致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部分木に含まれる前記リーフノードを選択し、選択した該リーフノードに含まれるインデックスキーを取得するインデックスキー取得手段をさらに備えることを特徴とする請求項１に記載の最長一致検索装置。
【請求項３】
コンピュータが、ビット列からなる最長一致検索キーにより、検索対象であるビット列からなるインデックスキーが格納されたツリーのデータ構造に基づいて、前記インデックスキーのうち前記最長一致検索キーとの最長一致条件を満たすインデックスキーを検索する最長一致検索方法であって、
前記ツリーは、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、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み、
前記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該ブランチノードに含まれる弁別ビット位置の検索キー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ツリーの任意の部分木の前記検索キーによる検索の結果である検索結果キーとするように構成されたカップルドノードツリーである最長一致検索方法において、
前記カップルドノードツリーの前記ルートノードを前記検索開始ノードとし、指定された最長一致検索キーを前記検索キーとして、前記ブランチノードにおいて該ブランチノードに含まれる弁別ビット位置の検索キー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リー

フノードに至るまで繰り返し、前記ルートノードから該リーフノードまでの経路を記憶しながら該リーフノードに格納されたインデックスキーを前記検索結果キーとして取得する検索ステップと、

前記最長一致検索キーと前記検索結果キーのビット列を比較して、ビット値が一致しない不一致ビットの位置のうち最も上位の位置である差分ビット位置を取得する差分ビット位置取得ステップと、

前記差分ビット位置がビット列の最上位以外の位置のとき、記憶された前記経路を参照して最長一致ノードを設定する最長一致ノード設定ステップであって、

前記ルートノードが前記リーフノードの場合または前記ルートノードが前記ブランチノードであり該ルートノードの前記弁別ビット位置が前記差分ビット位置よりも下位の位置の場合は、前記ルートノードを前記最長一致ノードとして設定し、

その他の場合は、前記経路上のノードであって前記弁別ビット位置が前記差分ビット位置よりも上位にある前記ブランチノードのうち前記弁別ビット位置が最下位の前記ブランチノードの次に前記検索ステップにおいて記憶された前記ブランチノードまたは前記リーフノードを前記最長一致ノードとして設定する最長一致ノード設定ステップと、

を備えることを特徴とする最長一致検索方法。

【請求項４】

前記最長一致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部分木に含まれる前記リーフノードを選択し、選択した該リーフノードに含まれるインデックスキーを取得するインデックスキー取得ステップをさらに備えることを特徴とする請求項３に記載の最長一致検索方法。

【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、前記位置情報は、該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り、前記検索ステップにおいて、前記経路上のノードの格納された前記配列要素の前記配列番号を順次スタックに保持していくことにより、前記経路を記憶することを特徴とする請求項３に記載の最長一致検索方法。

【請求項６】

ビット列からなる最短一致検索キーにより、検索対象であるビット列からなるインデックスキーが格納されたツリーのデータ構造に基づいて、前記インデックスキーのうち前記最短一致検索キーとの最短一致条件を満たすインデックスキーを検索する最短一致検索装置において、

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、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み、

前記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該ブランチノードに含まれる弁別ビット位置の検索キー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ツリーの任意の部分木の前記検索キーによる検索の結果である検索結果キーとするように構成されたカップルドノードツリーと、

前記カップルドノードツリーの前記ルートノードを前記検索開始ノードとし、指定された最長一致検索キーを前記検索キーとして、前記ブランチノードにおいて該ブランチノードに含まれる弁別ビット位置の検索キー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リーフノードに至るまで繰り返し、前記ルートノードから該リーフノードまでの経路を記憶しながら該リーフノードに格納されたインデックスキーを前記検索結果キーとして取得する検索手段と、

前記最短一致検索キーと前記検索結果キーのビット列を比較して、ビット値が一致しない不一致ビットの位置のうち最も上位の位置である差分ビット位置を取得する差分ビット位置取得手段と、

前記差分ビット位置がビット列の最上位以外の位置のとき、記憶された前記経路を参照して最短一致ノードを設定する最短一致ノード設定手段であって、

前記経路が、前記ルートノードと、該ルートノードのリンク先のノード対の一方のブランチノードとを含み、前記ルートノードの弁別ビット位置が前記差分ビット位置よりも上位の位置の場合は、前記ブランチノードの次に前記検索手段により記憶されたノードとノード対を構成するノードを前記最短一致ノードとして設定する最短一致ノード設定手段と、

を備えることを特徴とする最短一致検索装置。

【請求項７】

前記最短一致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部分木に含まれる前記リーフノードを選択し、選択した該リーフノードに含まれるインデックスキーを取得するインデックスキー取得手段をさらに備えることを特徴とする請求項６に記載の最短一致検索装置。

【請求項８】

コンピュータが、ビット列からなる最短一致検索キーにより、検索対象であるビット列からなるインデックスキーが格納されたツリーのデータ構造に基づいて、前記インデックスキーのうち前記最短一致検索キーとの最短一致条件を満たすインデックスキーを検索する最短一致検索方法であって、

前記ツリーは、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、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み、

前記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該ブランチノードに含まれる弁別ビット位置の検索キー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ツリーの任意の部分木の前記検索キーによる検索の結果である検索結果キーとするように構成されたカップルドノードツリーである最短一致検索方法において、

前記カップルドノードツリーの前記ルートノードを前記検索開始ノードとし、指定された最長一致検索キーを前記検索キーとして、前記ブランチノードにおいて該ブランチノードに含まれる弁別ビット位置の検索キー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リーフノードに至るまで繰り返し、前記ルートノードから該リーフノードまでの経路を記憶しながら該リーフノードに格納されたインデックスキーを前記検索結果キーとして取得する検索ステップと、

前記最短一致検索キーと前記検索結果キーのビット列を比較して、ビット値が一致しない不一致ビットの位置のうち最も上位の位置である差分ビット位置を取得する差分ビット位置取得ステップと、

前記差分ビット位置がビット列の最上位以外の位置のとき、記憶された前記経路を参照して最短一致ノードを設定する最短一致ノード設定ステップであって、
前記経路が、前記ルートノードと、該ルートノードのリンク先のノード対の一方のブランチノードとを含み、前記ルートノードの弁別ビット位置が前記差分ビット位置よりも上位の位置の場合は、前記ブランチノードの次に前記検索ステップにおいて記憶されたノードとノード対を構成するノードを前記最短一致ノードとして設定する最短一致ノード設定ステップと、
を備えることを特徴とする最短一致検索方法。
【請求項９】
前記最短一致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部分木に含まれる前記リーフノードを選択し、選択した該リーフノードに含まれるインデックスキーを取得するインデックスキー取得ステップをさらに備えることを特徴とする請求項８に記載の最短一致検索方法。
【請求項１０】
前記カップルドノードツリーは、配列に記憶され、前記位置情報は、該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り、前記検索ステップにおいて、前記経路上のノードの格納された前記配列要素の前記配列番号を順次スタックに保持していくことにより、前記経路を記憶することを特徴とする請求項８に記載の最短一致検索方法。
【請求項１１】
請求項３～請求項５のいずれか１項に記載の最長一致検索方法または請求項８～請求項１０のいずれか１項に記載の最短一致検索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１２】
請求項３～請求項５のいずれか１項に記載の最長一致検索方法または請求項８～請求項１０のいずれか１項に記載の最短一致検索方法をコンピュータに実行させるためのプログラムを記憶したことを特徴とするコンピュータ読み取り可能な記憶媒体。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ビット列を記憶するツリー状のデータ構造を用いてビット列の集合から所望のビット列を検索する検索方法に関するものであり、特に本出願人が特願２００６－１８７８２７において提案したカップルドノードツリーの最長一致検索／最短一致検索方法及びそのプログラムに関するものである。
【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１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１３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１３の例では、インデックスキー“１００００１０”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１７３０ａ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１７３０ｂ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１７４１ｃ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１７３０ｅ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１３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１３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１３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】
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。
【００１７】
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。
【００１８】
また、ビット列の検索には、最小値、最大値を求めたり、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。
【００１９】
さらに、ビット列の検索には、検索キーと完全に一致する値の検索の他に、検索キーと少なくとも一部が一致する値の検索がある。最上位のビットから１ビット以上が検索キーと一致し、その一致する範囲がなるべく大きい（あるいは小さい）インデックスキーを検索しようとする最長一致（あるいは最短一致）検索がその例である。
【００２０】
最長一致検索は、ルーティングテーブルの検索や、上位ビットが共通の複数のアドレス宛にデータを同報送信する場合の、同報アドレスの検索などに利用可能である。また、最短一致検索は、正規表現による文字列検索や検索対象の絞り込みなどに利用可能である。したがって、これらの検索を高速化し、様々な応用分野における処理効率を向上させるこ

とが望まれている。

【００２１】

特許文献２にはパトリシアツリーを用いた最長一致検索を高速化する技術が記載されているが、カップルドノードツリーは本出願人が発明した新規なデータ構造であるため、従来の最長一致検索手法を使うことはできない。

【特許文献１】特開２００１－３５７０７０号公報

【特許文献２】特開２００３－２２４５８１号公報

【発明の開示】

【発明が解決しようとする課題】

【００２２】

本発明の目的は、カップルドノードツリーを検索対象として最長一致／最短一致検索を行う方法およびプログラムを提供することである。

【課題を解決するための手段】

【００２３】

本発明が検索対象とするカップルドノードツリーは、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る。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る。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。

【００２４】

前記カップルドノードツリーは、前記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該ブランチノードに含まれる弁別ビット位置の検索キー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ツリーの任意の部分木の前記検索キーによる検索の結果である検索結果キーとするように構成されている。

【００２５】

本発明による前記カップルドノードツリーを用いた最長一致検索は、前記カップルドノードツリーの前記ルートノードを前記検索開始ノードとし、指定された最長一致検索キーを前記検索キーとして、前記ルートノードからの経路を記憶しながらの検索による前記検索結果キーの取得と、前記最長一致検索キーと前記検索結果キーのビット列比較による、ビット値が一致しない不一致ビットの位置のうち最も上位の位置である差分ビット位置の取得と、さらに、前記差分ビット位置がビット列の最上位以外の位置のとき、記憶された前記経路を参照し、前記ルートノードが前記リーフノードの場合または前記ルートノードが前記ブランチノードであり該ルートノードの前記弁別ビット位置が前記差分ビット位置よりも下位の位置の場合は、前記ルートノードを最長一致ノードとし、その他の場合は、前記経路上のノードであって前記弁別ビット位置が前記差分ビット位置よりも上位にある前記ブランチノードのうち前記弁別ビット位置が最下位の前記ブランチノードの次に記憶された前記ブランチノードまたは前記リーフノードを前記最長一致ノードとして設定する最長一致ノードの設定により実現される。

【００２６】

本発明による前記カップルドノードツリーを用いた最短一致検索は、前記最長一致検索キーのかわりに最短一致検索キーを用いる以外は上記と同様の検索および差分ビット位置の取得と、前記差分ビット位置がビット列の最上位以外の位置のときの、記憶された前記経路を参照した最短一致ノードの設定であって、前記経路が、前記ルートノードと、該ル

ートノードのリンク先の前記ノード対の一方の前記ブランチノードである第一のノードとを含み、前記ルートノードの前記弁別ビット位置が前記差分ビット位置よりも上位の位置の場合は、前記ブランチノードの次に記憶されたノードとノード対を構成するノードを前記最短一致ノードとして設定する最短一致ノードの設定により実現される。

【発明の効果】

【００２７】

本発明によれば、カップルドノードツリーを検索対象として最長一致／最短一致検索を行うことが可能となる。よって、最長一致／最短一致検索が必要な各種応用分野にカップルドノードツリーを用いることが可能となる。カップルドノードツリーは、検索を高速化するよう構成されているため、カップルドノードツリーを用いた最長一致／最短一致検索は高速である。よって、各種応用分野における最長一致／最短一致検索の処理の高速化が実現される。

【発明を実施するための最良の形態】

【００２８】

以下、本発明の実施形態について、図面を参照しながら詳細に説明する。

最初に、本出願人により先の上記出願において提案された、本発明の前提となるカップルドノードツリーについて、カップルドノードツリーを配列に格納する例を説明する。ブランチノードが保持するリンク先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、位置情報の情報量を削減することができる。

【００２９】

図１は、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１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、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。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。

【００３０】

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のノード種別１１４には０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のノード種別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。インデックスキー１１８には、“０００１”が格納されている。パトリシアツリーについて先に述べたと同様に、リーフノード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情報が含まれることは当然であるが、表記は省略している。

【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。前段のブランチノードの弁別ビット位置にある検索キーのビット値である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、検索キーの弁別ビット位置の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２は、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。図示の６ビットのインデックスキーは、図１３に例示されたパトリシアツリーのものと同じである。

【００３５】

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。

ツリー構造としては、ルートノード２１０ａの下にノート対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。

【００３６】

各ノードの前に付された０あるいは１の符号は、図１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、検索対象のリーフノードを見つけることになる。

【００３７】

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、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。

【００３８】

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。

【００３９】

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、インデックスキー２５０ｃを含んでいる。インデックスキー２５０ｃには“０００１１１”が格納されている。一方ノード２１１ｃのノード種別２６１ｃは０、弁別ビット位置２３１ｃは２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。

【００４０】

ノード２１０ｄのノード種別２６０ｄは０、弁別ビット位置２３０ｄは５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、インデックスキー２５１ｄには“０１１０１０”が格納されている。

【００４１】

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し、それぞれのインデックスキー２５０

ｅ、２５１ｅにはインデックスキーとして“０１００１０”と“０１００１１”が格納されている。

【００４２】

ノード対２０１ｂのもう一方のノードであるノード２１１ｂの弁別ビット位置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。

【００４３】

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。それぞれの弁別ビット位置２３０ｆ、２３１ｆには5、3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。

【００４４】

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し、それぞれのインデックスキー２５０ｇ、２５１ｇには“１０００１０”と“１０００１１”が格納されている。

【００４５】

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し、それぞれのインデックスキー２５０ｈ、２５１ｈには“１０１０１１”と“１０１１００”が格納されている。

【００４６】

以下、上述のツリーからインデックスキー“１０００１０”を検索する処理の流れを簡単に説明する。弁別ビット位置は、左から0、1、2、・・・とする。

まず、ビット列“１０００１０”を検索キー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の弁別ビット位置２３０ａは0であるので、検索キー“１０００１０”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の弁別ビット位置２３１ｂには2が格納されているので、検索キー“１０００１０”の弁別ビット位置が2のビット値をみると0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２１０ｆにリンクする。

【００４７】

ノード２１０ｆの弁別ビット位置２３０ｆには5が格納されているので、検索キー“１０００１０”の弁別ビット位置が5のビット値をみると0であるから、代表ノード番号の格納された配列番号２２０ｆの配列要素に格納されたノード２１０ｇにリンクする。

【００４８】

ノード２１０ｇのノード種別２６０ｇは1でありリーフノードであることを示しているので、インデックスキー２５０ｇを読み出して検索キーと比較すると両方とも“１０００１０”であって一致している。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。

【００４９】

次に、図2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2の例で、ルートノード２１０ａの弁別ビット位置が0であるのは、図2に例示されたインデックスキーに0ビット目が0のものと1のものがあるからである。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。

【００５０】

ノード２１１ｂの弁別ビット位置が２であるのは、ノード２１１ｈ、２１０ｈ、２１１ｇ、２１０ｇに格納された０ビット目が１のインデックスキーの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。

【００５１】

以下０ビット目の場合と同様に、２ビット目が１であるものはノード２１１ｆ側に分類され、２ビット目が０であるものはノード２１０ｆ側に分類される。

そして２ビット目が１であるインデックスキーは３ビット目の異なるものがあるのでノード２１１ｆの弁別ビット位置には３が格納され、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の弁別ビット位置には５が格納される。

【００５２】

ノード２１１ｆのリンク先においては、３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２１０ｈ、２１１ｈはリーフノードとなり、それぞれインデックスキー２５０ｈと２５１ｈに“１０１０１１”と“１０１１００”が格納されている。

【００５３】

仮にインデックスキーの集合に“１０１１００”の代わりに“１０１１０１”か“１０１１１０”が含まれていたとしても、３ビット目までは“１０１１００”と等しいので、ノード２１１ｈに格納されるインデックスキーが変わるだけで、ツリー構造自体は変わることはない。しかし、“１０１１００”に加えて“１０１１０１”が含まれていると、ノード２１１ｈはブランチノードとなり、その弁別ビット位置は５になる。追加されるインデックスキーが“１０１１１０”であれば、弁別ビット位置は４となる。

【００５４】

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のインデックスキー２５１ｈの“１０１１００”、ノード２１０ｈのインデックスキー２５０ｈの“１０１０１１”、・・・、ノード２１０ｃのインデックスキー２５０ｃの“０００１１１”となり降順にソートされている。

【００５５】

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、例えば検索キーが“１０１１００”であればノード２１１ｈ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、“１０１１０１”か“１０１１１０”を検索キーとした場合でもノード２１１ｈにたどり着き、インデックスキー２５１ｈと比較することにより検索が失敗したことが分かる。

【００５６】

また、例えば“１００１００”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３ビット目と４ビット目は使われることがなく、“１００１００”の５ビット目が０なので、“１０００１０”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弁別ビット位置を用いて分岐が行われる。

【００５７】

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。

本発明の最長一致／最短一致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。

【００５８】
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。
【００５９】
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装置が用いられることは当然である。
次に、上述の出願において、本出願人により提案されたカップルドノードツリーを用いた基本的な検索処理、カップルドノードツリーにおける挿入削除処理、及びカップルドノードツリーに含まれるインデックスキーの最大値／最小値を求める処理等の応用処理の一部について、図４～図５を参照することにより、本発明を理解するために必要な範囲で紹介する。
【００６０】
図４は、本出願人による出願である上記特願２００６－２９３６１９で提案されたビット列検索の基本動作を示したフローチャートである。
まず、ステップＳ４０１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要素は、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。
【００６１】
次に、ステップＳ４０２で、探索経路スタック３１０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。
【００６２】
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Ｓ４０６に進み、ノードから弁別ビット位置についての情報を取り出し、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、ノードから代表ノード番号を取り出して、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。
【００６３】
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Ｓ４１０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Ｓ４１０で、リーフノードからインデックスキーを取り出して、処理を終了する。
【００６４】
以上、本発明の基礎となるカップルドノードツリーについてとビット列検索の基本動作について詳細に説明した。カップルドノードツリーの生成処理については、特に図示はしていないが、本出願人による上記先の出願である特願２００６－１８７８２７において説明したように、ビット列の集合があるとき、そこから任意のビット列を順次取り出してインデックスキーとして挿入する処理を繰り返すことにより可能である。

【００６５】
先に述べたように、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値は、ノード［１］側とツリーの深さ方向をたどることにより降順にソートされてツリー上に配置されていることから、インデックスキーの挿入処理は、挿入するインデックスキーを検索キーとしてカップルドノードツリーから該当するリーフノードを検索するとともに、該リーフノードに至るまでたどったリンク経路のブランチノード及び該リーフノードが格納された配列要素の配列番号をスタックに順次格納し、検索キーと該当するリーフノードに含まれるインデックスキーの間で大小比較とビット列比較を行い、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上記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により挿入されるインデックスキーを含む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し、上記大小関係により挿入するインデックスキーを含むリーフノードを上記挿入されるノード対のどちらのノードとするかを決定することにより実施することができる。
【００６６】
図５は、本出願人による出願である上記特願２００６－２９３６１９で提案されたカップルドノードツリー（部分木を含む）に格納されたインデックスキーの最小値を求める処理を示したフローチャートである。先に述べたようなインデックスキーのツリー上の配置から、インデックスキーの最小値を求める処理は検索開始ノードからノード［０］をリーフノードに至るまでツリー上でたどることに相当する。
【００６７】
まず、ステップＳ５０１の検索開始ノードの配列番号の取得からステップＳ５０５のノード種別の判定までは、それぞれ上述の図４のステップＳ４０１からステップＳ４０５の処理と同様である。
【００６８】
ステップＳ５０５のノード種別の判定においてノード種別がブランチノードと判定されると、ステップＳ５０６に進み、ノードから配列の代表ノード番号を取り出し、ステップＳ５０７で、取り出した代表ノード番号に値「０」を加算してその結果を新たな配列番号とし、ステップＳ５０２に戻る。以降、ステップＳ５０５においてそのノードがリーフノードと判定されるまでステップＳ５０２からステップＳ５０７までの処理を繰り返し、ステップＳ５０８で、リーフノードからインデックスキーを取り出し、処理を終了する。
【００６９】
上記の図５に示す処理においては、ノード［０］をたどるため、代表ノード番号に一律「０」を加算している。すなわち、図５の処理によれば、リンク先のノードは、ノード対のうち必ずノード［０］とし、より小さい値のインデックスキーが格納されているノードのほうに分岐している。これにより、ツリー構造が先に述べたように順列構成であるカップルドノードツリーの最小のインデックスキーを取り出すことができる。
【００７０】
なお、図５と類似の方法で、カップルドノードツリー（部分木を含む）に格納されたインデックスキーの最大値を求めることができるのは当然である。具体的には、図５の処理を、ステップＳ５０７において「０」ではなく「１」を加算するように変更すれば、インデックスキーの最大値を求めることができる。この変更により、代表ノード番号が表すノード対のうち、ノード［１］に常にリンクすることとなり、リーフノードに至るまでツリー構造の階層を順次たどっていくことができるからである。
【００７１】
図４と図５に示すように、検索キーと一致するインデックスキーを検索する基本動作やインデックスキーの最小値／最大値の検索処理を実行する際には、探索経路スタック３１０に参照した配列の配列番号が順次格納されていく。
【００７２】
なお、上述の図５を参照したインデックスキーの最小値／最大値の検索処理では、カップルドノードツリーは配列に記憶されたものとして説明したが、カップルドノードツリー

が配列に記憶されることは必須ではなく、ノード対を構成する２つのノードのうちの代表ノードのみあるいは代表ノードと隣接した記憶領域に配置されたノードのみをリンクしてリーフノードに至ることによりインデックスキーの最小値／最大値の検索が可能であることは明らかである。

【００７３】

以上、カップルドノードツリーの最長一致／最短一致検索の前提となる技術を説明したが、必要とあれば、上述の特許出願の明細書及び図面の記載を参照されたい。

以下では、本発明によるカップルドノードツリーの最長一致／最短一致検索について説明する。まず、下記の実施形態で用いる用語と、下記の実施形態における前提条件を説明する。

【００７４】

（ａ）最長一致検索は、指定された検索キーである最長一致検索キーと部分一致するインデックスキーのうち、差分ビット位置が最も下位となるインデックスキーを検索する処理である。そのようなインデックスキーを「最長一致条件を満たすインデックスキー」と定義する。最長一致条件を満たすインデックスキーは存在しないこともあるし、１つまたは複数存在することもある。なお、差分ビット位置とは、２つのビット列を比較したときにビット値が一致しない不一致ビットのうち、最も上位の位置を表す。

【００７５】

（ｂ）最短一致検索は、指定された検索キーである最短一致検索キーと部分一致するインデックスキーのうち、差分ビット位置が上位のものを優先的に検索する処理である。最短一致の定義の細部については実施形態によって違いがあってもよい。以下の実施形態では、最短一致を次のように定義して最短一致検索方法について詳しく説明する。

【００７６】

ルートノードの弁別ビット位置までのビット値が最短一致検索キーと完全に一致するが、ルートノードの弁別ビット位置以降のビット値には最短一致検索キーと不一致のビットが存在するインデックスキーを「最短一致条件を満たすインデックスキー」と定義する。最短一致条件を満たすインデックスキーは存在しないこともあるし、１つまたは複数存在することもある。

【００７７】

次に、カップルドノードツリーの最長一致検索の具体的な方法について図６、８、１０、１１を参照して説明する。

図６は、本発明の一実施形態における最長一致検索処理のフローチャートである。図６の処理の概要は、ステップＳ６０１～Ｓ６０６で最長一致ノードを取得し、ステップＳ６０７～Ｓ６０８で最長一致ノードをルートノードとする部分木の中から一つのインデックスキーを取得する、というものである。なお、ある部分木に含まれるすべてのリーフノードのインデックスキーが最長一致条件を満たすとき、その部分木のルートノードを最長一致ノードと定義する。よって、ステップＳ６０７～Ｓ６０８により取得されたインデックスキーは最長一致条件を満たす。より詳細には、図６の処理は下記のとおりである。

【００７８】

ステップＳ６０１で指定された最長一致検索キーを検索キーに設定し、ステップＳ６０２で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に設定する。

次に、ステップＳ６０３において、ステップＳ６０１で設定された検索キーを用いて配列３０９を検索し、インデックスキーを取得する。この検索は、図４に示したとおりである。ステップＳ６０２で検索開始ノードにルートノードが設定されているので、検索対象はカップルドノードツリー全体である。ステップＳ６０３の検索で探索経路スタック３１０に格納された内容は、後にステップＳ６０６で利用される。

【００７９】

ステップＳ６０３でインデックスキーを取得した後、処理はステップＳ６０４に進み、ステップＳ６０１で設定された検索キー、すなわち最長一致検索キーと、ステップＳ６０３で取得されたインデックスキーとの差分ビット位置を求める。この処理の詳細は図１０

と合わせて後述する。

【００８０】

次に、ステップＳ６０５において、ステップＳ６０４で求めた差分ビット位置が０か否かを判定する。

差分ビット位置が０のとき、判定は「はい」となり、最長一致検索処理を終了する。この場合、最長一致検索キーと、ステップＳ６０３で取得されたインデックスキーは、先頭の０ビット目が不一致である。また、カップルドノードツリーの構造から、カップルドノードツリーに含まれる他のすべてのインデックスキーも、最長一致検索キーと先頭の０ビット目が不一致である。したがって、カップルドノードツリー内に部分一致条件を満たすインデックスキーを含むリーフノードが存在しないため、「最長一致条件を満たすインデックスキーは存在しない」という検索結果が得られる。

【００８１】

差分ビット位置が０ではないとき、ステップＳ６０５の判定が「いいえ」となり、処理はステップＳ６０６に進む。この場合、カップルドノードツリー内に最長一致条件を満たすインデックスキーが少なくとも１つ存在する。しかし、ステップＳ６０５の段階では、最長一致条件を満たすインデックスキーの存在が判明しただけなので、ステップＳ６０６では、具体的な最長一致ノードを取得する。

【００８２】

ステップＳ６０６では、ステップＳ６０３の検索によって配列３０９の配列番号が格納された探索経路スタック３１０を参照し、ステップＳ６０４で求めた差分ビット位置に基づいて、最長一致ノードを取得する。

【００８３】

最長一致ノードの取得後、次のステップＳ６０７において最長一致ノードを検索開始ノードに設定する。そして、その次のステップＳ６０８において、検索開始ノードより、図５に示した最小値検索処理を実行し、最長一致条件を満たすインデックスキーのうちの最小値を取得し、取得した最小値を最長一致検索の結果として出力して、最長一致検索処理を終了する。

【００８４】

なお、ステップＳ６０７とＳ６０８は、実施形態に応じて様々に変形可能である。本実施形態のように、最長一致条件を満たす１つ以上のインデックスキーの中から最小値を選択することは、処理の一例にすぎない。他の変形例については後述する。

【００８５】

次に、図６のステップＳ６０４の処理について図１０を参照して説明する。後述するように、図１０の処理は、最短一致検索処理からも呼び出される。

ステップＳ１０１において、検索キーを比較キー１に設定し、インデックスキーを比較キー２に設定する。最長一致検索処理の場合、ここでの「検索キー」はステップＳ６０１で設定された最長一致検索キーであり、ここでの「インデックスキー」はステップＳ６０３で取得されたインデックスキーである。

【００８６】

次に、ステップＳ１０２において、最上位の０ビット目から順に比較キー１と比較キー２のビット列を比較する。比較キー１と比較キー２で値が一致しない不一致ビットのうち最初に見つかったもののビット位置が、差分ビット位置として取得されて図１０の処理が終了する。上記の説明から明らかなとおり、図６のステップＳ６０４から図１０の処理が呼び出された場合、ステップＳ１０２では、最長一致検索キーとステップＳ６０３で取得されたインデックスキーとの差分ビット位置が得られる。

【００８７】

次に、図６のステップＳ６０６で行われる最長一致ノードを取得する処理について図８を参照して説明する。図８の処理を理解しやすくするために、説明は（Ａ１）～（Ａ３）の３つの場合に分けて行う。

【００８８】

（Ａ１）ルートノードがリーフノードの場合
この場合、図６のステップＳ６０３で取得されたインデックスキーはルートノードに格納されたインデックスキーである。よって、最長一致検索キーとインデックスキーとの差分ビット位置が０より大きければ、インデックスキーは最長一致条件を満たし、ルートノードは最長一致ノードに該当する。
【００８９】
また、図６のステップＳ６０５から明らかなとおり、図８の処理が行われるのは差分ビット位置が０より大きい場合のみなので、（Ａ１）の場合、図８の処理によって必ずルートノードが最長一致ノードに設定される。具体的には次のとおりである。
【００９０】
ステップＳ８０１で、探索経路スタック３１０のスタックポインタを１つ戻す。
また、（Ａ１）の場合、探索経路スタック３１０に格納されている配列３０９の配列番号はルートノードの配列番号のみである。よって、ステップＳ８０１の実行後、スタックポインタが指す内容は、ルートノードの配列番号である。この状態で処理はステップＳ８０２に移行する。
【００９１】
ステップＳ８０２で、探索経路スタック３１０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否かを判定する。（Ａ１）の場合は、上記のとおりであるから、判定の結果は「はい」となり、ステップＳ８０８に進む。
【００９２】
ステップＳ８０８で、探索経路スタック３１０から、スタックポインタの指す配列番号を取り出す。つまり、（Ａ１）の場合は、ルートノードの配列番号が取り出される。
そして、続くステップＳ８０９では、ステップＳ８０８で取得された配列番号、すなわちルートノードの配列番号を最長一致ノードの配列番号に設定し、図８の処理が終了する。
【００９３】
（Ａ２）ルートノードがブランチノードであり、ルートノードの弁別ビット位置が０の場合
この場合、０ビット目が０のインデックスキーと０ビット目が１のインデックスキーの双方を検索対象のカップルドノードツリーが含む。よって、少なくとも０ビット目が一致するインデックスキーが必ず存在する。これは、最長一致条件を満たすインデックスキーが必ず存在することと、図６のステップＳ６０４で取得された差分ビット位置が０より大きいことを意味する。つまり、（Ａ２）の場合、ステップＳ６０５の判定は必ず「いいえ」となり、必ず図８の処理が実行される。
【００９４】
ステップＳ８０１で、探索経路スタック３１０のスタックポインタを１つ戻す。その結果、スタックポインタが指す内容は、ステップＳ６０３で取得されたインデックスキーを格納しているリーフノードの配列番号となる。（Ａ２）の場合、このリーフノードはルートノードではない。ステップＳ８０１の実行後、ステップＳ８０２～Ｓ８０６のループが繰り返し実行される。このループは、探索経路スタック３１０を遡る操作である。
【００９５】
ステップＳ８０２では、探索経路スタック３１０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否かを判定する。（Ａ２）の場合、１回目にステップＳ８０２を実行した時点での判定は「いいえ」となり、ステップＳ８０３に移行する。
【００９６】
ステップＳ８０３では、探索経路スタック３１０のスタックポインタを１つ戻し、スタックポインタが指す配列番号を取り出す。続くステップＳ８０４では、配列３０９から、ステップＳ８０３で取得した配列番号の指す配列要素をノードとして読み出す。そして、次のステップＳ８０５では、ステップＳ８０４で読み出したノードから弁別ビット位置を取り出す。

【００９７】
なお、ステップＳ８０３～Ｓ８０６は、ステップＳ８０２の判定が「いいえ」の場合のみ実行される。よって、ステップＳ８０３でスタックポインタを１つ戻したときにアンダーフローを起こさないことが保証される。また、図４とステップＳ８０２の説明から分かるとおり、ステップＳ８０３～Ｓ８０６が実行されるのは、探索経路スタック３１０にルートノード以外のノードの配列番号が１つ以上格納されている場合である。よって、ステップＳ８０３の実行後にスタックポインタが指すのは、ルートノードであるブランチノードまたはそれ以外のブランチノードの配列番号である。
【００９８】
ステップＳ８０５の次のステップＳ８０６では、取り出された弁別ビット位置が、図１０のステップＳ１０２で得た差分ビット位置よりも上位か否かが判定される。弁別ビット位置が差分ビット位置よりも上位の場合、判定は「はい」となってステップＳ８０７に進み、下位の場合、判定は「いいえ」となってステップＳ８０２に戻る。
【００９９】
（Ａ２）の場合は必ず、ステップＳ８０２～Ｓ８０６のループを１回以上繰り返して実行した後、ステップＳ８０６の判定が「はい」となって、ステップＳ８０７、Ｓ８０８、Ｓ８０９と順に処理が進む。
【０１００】
ステップＳ８０７では、探索経路スタック３１０のスタックポインタを１つ進める。ステップＳ８０７が実行されるのはステップＳ８０３が１回以上実行された後なので、ステップＳ８０７の操作によりオーバーフローが生じることはない。
【０１０１】
続くステップＳ８０８で探索経路スタック３１０からスタックポインタの指す配列番号を取り出し、次のステップＳ８０９において、ステップＳ８０８で取得された配列番号を最長一致ノードの配列番号に設定し、図８の処理が終了する。
【０１０２】
上記の処理により（Ａ２）の場合に最長一致ノードとして設定されるノードは、次の２つのいずれかである。
ステップＳ６０３で取得されたインデックスキーを格納するリーフノードにリンクするブランチノードの弁別ビット位置が、差分ビット位置よりも上位の場合、上記リーフノードが最長一致ノードとして設定される。
【０１０３】
弁別ビット位置がｍであるブランチノードの配列番号と、そのブランチノードからリンクされ弁別ビット位置がｎであるブランチノードの配列番号が探索経路スタック３１０に順に格納されており、弁別ビット位置ｍと弁別ビット位置ｎと差分ビット位置ｄとの関係が、ｍ＜ｄ＜ｎの場合、弁別ビット位置がｎであるブランチノードが最長一致ノードとして設定される。
【０１０４】
（Ａ３）ルートノードがブランチノードであり、ルートノードの弁別ビット位置が０より大きい場合
この場合、図６のステップＳ６０５の判定が「はい」になることもある点が、（Ａ２）の場合と異なる。
【０１０５】
ステップＳ８０１で探索経路スタックのスタックポインタを１つ戻すことと、ステップＳ８０２～Ｓ８０６のループの１回目の実行時にはステップＳ８０２の判定が必ず「いいえ」になることと、続くステップＳ８０３～Ｓ８０６の動作は、（Ａ２）の場合と同様である。
【０１０６】
ステップＳ８０６で判定が「いいえ」となってステップＳ８０２に戻った場合、再度ステップＳ８０２の判定が行われる。（Ａ３）の場合は、ステップＳ８０２の２回目以降の

実行において判定が「はい」となることがある点で、（Ａ２）の場合と異なる。

【０１０７】

上記のとおり、ステップＳ８０２の１回目の実行時には判定が必ず「いいえ」になる。ステップＳ８０２の２回目以降の実行時には、直前にステップＳ８０６が実行され、ステップＳ８０６で「いいえ」と判定されている。つまり、ループを１回以上繰り返して実行した結果、スタックポインタがルートノードの配列番号を指すまで探索経路スタック３１０を遡った場合であって、ルートノードの弁別ビット位置が差分ビット位置よりも下位の場合にのみ、ステップＳ８０２で「はい」と判定される。

【０１０８】

ステップＳ８０２で「はい」と判定されると、ステップＳ８０８に進み、（Ａ１）の場合と同様にして、ルートノードの配列番号が最長一致ノードの配列番号に設定され、図８の処理が終了する。

【０１０９】

（Ａ３）の場合はルートノードの弁別ビット位置が０より大きいので、ルートノードの弁別ビット位置ｍと差分ビット位置ｄとの関係は、下記のとおりである。カップルドノードツリーに含まれるすべてのインデックスキーは、０～（ｍ－１）ビット目が共通である。また、（Ａ３）の場合でステップＳ８０２において「はい」と判定された場合は、０＜ｄ＜ｍである。よって、この場合は、カップルドノードツリーに含まれるすべてのインデックスキーは、０～（ｄ－１）ビット目が最長一致検索キーと一致している。したがって、この場合は上記のとおり、ルートノードの配列番号を最長一致ノードの配列番号に設定する。

【０１１０】

また、（Ａ２）の場合と同様に、（Ａ３）の場合でも、ステップＳ８０２～Ｓ８０６のループを１回以上繰り返して実行した後、ステップＳ８０６の判定が「はい」となって、ステップＳ８０７、Ｓ８０８、Ｓ８０９と順に処理が進むことがある。その場合の処理の内容は（Ａ２）の場合と同様である。

【０１１１】

次に、図１１を参照して、最長一致検索の具体例について説明する。図１１のカップルドノードツリーは、“０００１１１”、“０１００１０”、“０１００１１”、“０１１０１０”、“１０００１０”、“１０００１１”、“１０１１００”、“１０１１１１”なる８つのインデックスキーを含む。図１１と図２のカップルドノードツリーは同じ構造を有しているので、同じ参照符号を付した。ただし、ノード２１１ｆの弁別ビット位置２３１ｆの値と、ノード２１０ｈのインデックスキー２５０ｈの値と、ノード２１１ｈのインデックスキー２５１ｈの値は、図２とは異なる。

【０１１２】

以下では最長一致検索キーとして“１０１０１０”が指定された場合について説明する。

図６のステップＳ６０１で“１０１０１０”が検索キーに設定され、ステップＳ６０２でルートノードが検索開始ノードに設定される。ステップＳ６０３から図４の処理が呼び出されると、探索経路スタック３１０には、配列番号２２０、（２２０ａ＋１）、（２２１ｂ＋１）、（２２１ｆ＋１）が順に格納される。また、ステップＳ６０３で得られるインデックスキーは、配列番号（２２１ｆ＋１）の配列要素であるノード２１１ｈのインデックスキー２５１ｈに格納された“１０１１１１”である。図１１では、このことをノード２１１ｈの横の「初期検索」で示した。

【０１１３】

ステップＳ６０４で、“１０１１１１”なる値のインデックスキー２５１ｈと“１０１０１０”なる値の検索キーの差分ビット位置が求められ、その値３は０より大きいのでステップＳ６０５の判定は「いいえ」となる。そのため、ステップＳ６０６、すなわち図８の処理が実行される。ステップＳ８０２～Ｓ８０６の繰り返しにより探索経路スタック３１０を遡り、配列番号（２２０ａ＋１）が得られると、配列番号（２２０ａ＋１）の配列

要素に格納されたノード２１１ｂの弁別ビット位置２３１ｂが２であり、弁別ビット位置と差分ビット位置との関係は２＜３なので、ステップＳ８０６で「はい」と判定される。よって、ステップＳ８０７でスタックポインタが１つ進められ、ステップＳ８０８で配列番号（２２１ｂ＋１）が取り出され、配列番号（２２１ｂ＋１）が最長一致ノードの配列番号に設定される。つまり、配列番号（２２１ｂ＋１）の配列要素に格納されたノード２１１ｆが最長一致ノードに設定される。

【０１１４】

ここで処理は図６に戻り、ステップＳ６０７でノード２１１ｆが検索開始ノードに設定され、ステップＳ６０８の処理、すなわち図５の最小値検索が行われる。図１１のとおり、最長一致ノードに設定されたブランチノードであるノード２１１ｆの子孫のリーフノードは、ノード２１０ｈとノード２１１ｈの２つである。両ノードのインデックスキーの値は“１０１１００”と“１０１１１１”である。いずれも、最長一致検索キー“１０１０１０”とは０～２ビット目が一致し、３ビット目が不一致である。また、カップルドノードツリーに含まれる他の６つのインデックスキーは、いずれも、最長一致検索キー“１０１０１０”との差分ビット位置が３より小さい。ステップＳ６０８の処理により、“１０１１００”と“１０１１１１”のうちの最小値である“１０１１００”が選択され、図６の処理は終了する。

【０１１５】

次に、カップルドノードツリーの最短一致検索の具体的な方法について図７、９、１０、１２を参照して説明する。

図７は、本発明の一実施形態における最短一致検索処理のフローチャートである。図７は、図６における「最長一致」を「最短一致」に変えた図である。

【０１１６】

まず、ステップＳ７０１で最短一致検索キーを検索キーにし、ステップＳ７０２でルートノードを検索開始ノードに設定する。

次に、ステップＳ７０３において、ステップＳ７０１で設定された検索キーを用いて配列を検索し、インデックスキーを取得する。この検索は、図４に示したとおりである。ステップＳ７０２で検索開始ノードにルートノードが設定されているので、検索対象はカップルドノードツリー全体である。ステップＳ７０３の検索で探索経路スタック３１０に格納された内容は、後にステップＳ７０６で利用される。インデックスキーの取得後、処理はステップＳ７０４に進む。

【０１１７】

ステップＳ７０４では、ステップＳ７０１で設定された検索キー、すなわち最短一致検索キーと、ステップＳ７０３で取得されたインデックスキーとの差分ビット位置を求める。

【０１１８】

この処理の詳細は図１０と合わせて既に述べたとおりである。ただし、ステップＳ７０４から図１０の処理を呼び出す場合には、ステップＳ１０１で設定される比較キー１が最短一致検索キーであり、比較キー２がステップＳ７０３で取得されたインデックスキーである点が、最長一致検索の場合と異なる。

【０１１９】

ステップＳ７０４の実行後、ステップＳ７０５において、ステップＳ７０４で求めた差分ビット位置が０か否かを判定する。

差分ビット位置が０のとき、判定は「はい」となり、最短一致検索処理を終了する。この場合、最短一致検索キーと、ステップＳ７０３で取得されたインデックスキーは、先頭の０ビット目が不一致である。また、カップルドノードツリーの構造から、カップルドノードツリーに含まれる他のすべてのインデックスキーも、最短一致検索キーと先頭の０ビット目が不一致である。したがって、「最短一致条件を満たすインデックスキーは存在しない」という検索結果が得られる。

【０１２０】

差分ビット位置が０ではないとき、ステップＳ７０５の判定は「いいえ」となり、処理がステップＳ７０６に進む。この場合、カップルドノードツリー内に最短一致条件を満たすインデックスキーが少なくとも１つは存在する。しかし、ステップＳ７０５の段階では最短一致条件を満たすインデックスキーの存在が判明しただけなので、ステップＳ７０６では、具体的な最短一致ノードを取得する。なお、ある部分木に含まれるすべてのリーフノードのインデックスキーが最短一致条件を満たすとき、その部分木のルートノードを最短一致ノードと定義する。

【０１２１】

ステップＳ７０６では、ステップＳ７０３の検索によって配列３０９の配列番号が格納された探索経路スタック３１０を参照し、ステップＳ７０４で求めた差分ビット位置に基づいて、最短一致ノードを取得する。

【０１２２】

最短一致ノードの取得後、次のステップＳ７０７において最短一致ノードを検索開始ノードに設定する。そして、その次のステップＳ７０８において、検索開始ノードより、図５に示した最小値検索処理を実行し、最短一致条件を満たすインデックスキーのうちの最小値を取得し、取得した最小値を最短一致検索の結果として出力して、最短一致検索処理を終了する。

【０１２３】

なお、図６と同様に、ステップＳ７０７とＳ７０８は、実施形態に応じて様々に変形可能である。

次に、図７のステップＳ７０６で行われる最短一致ノードを取得する処理について図９を参照して説明する。図９の処理は、上記（ｂ）の「最短一致条件」の定義にしたがった処理だが、図９の処理を理解しやすくするために、説明は（Ｂ１）～（Ｂ４）の４つの場合に分けて行う。（Ｂ１）と（Ｂ２）は最短一致キーによる検索の結果、探索経路スタック上に少なくとも３つのノードの配列番号が格納されている場合であり、（Ｂ３）と（Ｂ４）はそれ以外の場合である。

【０１２４】

（Ｂ１）ルートノードがブランチノードであり、ステップＳ７０３で取得されたインデックスキーを格納するリーフノードへのルートノードからの経路上には、ルートノード以外のブランチノードが少なくとも１つ存在し、ルートノードの弁別ビット位置をｍとすると最短一致検索キーと０～ｍビット目までの範囲が一致するインデックスキーが存在する場合

この場合、上記経路上には少なくとも３つのノードが存在する。探索経路スタック３１０には、ルートノード、ルートノードの子ノードであるブランチノード、そのブランチノードの子ノードの順で、配列番号が格納されている。上記（ｂ）の定義より、この場合、ルートノードの子ノードであるブランチノードからリンクされ、探索経路スタック３１０に配列番号が格納されたノードと対をなすノードが最短一致ノードに該当する。（Ｂ１）の場合における図９の処理は、このノードを最短一致ノードに設定する処理である。

【０１２５】

まず、ステップＳ９０１で探索経路スタック３１０のスタックポインタを退避エリアに設定する。

続くステップＳ９０２～Ｓ９０３はループを形成しており、１回以上繰り返し実行される。このループは、スタックポインタがルートノードの配列番号を指すまで探索経路スタック３１０を遡ることを表す。まず、ステップＳ９０２で、探索経路スタック３１０のスタックポインタを１つ戻す。続くステップＳ９０３で、探索経路スタック３１０のスタックポインタがルートノードの配列番号を指すか否かが判定される。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場合は判定が「はい」となってステップＳ９０４に進み、それ以外の場合は判定が「いいえ」となってステップＳ９０２に戻る。

【０１２６】

ステップＳ９０４では、スタックポインタと退避エリアの値が比較される。ここで、ス

テップＳ９０４が実行されるのは、スタックポインタがルートノードの配列番号を指す場合のみである。つまり、ステップＳ９０４は、図９の処理の開始直前のスタックポインタがルートノードの配列番号を指すスタックポインタとどれほど離れているかを調べる処理であり、退避エリアの値とスタックポインタを比較し、その差に応じて処理を分岐させる処理である。

【０１２７】

（Ｂ１）の場合、上記のとおり探索経路スタック３１０には少なくとも３つのノードの配列番号が格納されているため、ステップＳ９０４の比較結果は必ず「差＞２」となる。よって、処理は必ずステップＳ９０５に進む。

【０１２８】

ステップＳ９０５において、探索経路スタック３１０からスタックポインタの指す配列番号、すなわちルートノードの配列番号を取り出す。続くステップＳ９０６において、ステップＳ９０５で取り出した配列番号の指す配列要素を配列３０９からノードとして読み出す。次のステップＳ９０７では、ステップＳ９０６で読み出されたノード、すなわちルートノードから、弁別ビット位置を取り出す。

【０１２９】

続くステップＳ９０８では、ステップＳ９０７で取り出された弁別ビット位置が、図１０のステップＳ１０２で得た差分ビット位置よりも上位か否かが判定される。弁別ビット位置が差分ビット位置よりも上位の場合、判定は「はい」となってステップＳ９０９に進み、下位の場合、判定は「いいえ」となってステップＳ９１４に進む。

【０１３０】

（Ｂ１）の場合、ステップＳ９０８の判定は必ず「はい」になる。理由は次のとおりである。

図９の処理が実行されるのは差分ビット位置が０より大きいときのみなので、ステップＳ９０７で取り出されたルートノードの弁別ビット位置が０のときは必ずステップＳ９０８の判定が「はい」となる。

【０１３１】

また、ステップＳ９０７で取り出されたルートノードの弁別ビット位置が０より大きい場合、ルートノードの弁別ビット位置をｍとすると、カップルドノードツリーに含まれるすべてのインデックスキーは０～（ｍ－１）ビット目の範囲が同一である。また、（Ｂ１）の場合、最短一致検索キーとｍビット目まで一致するインデックスキーが存在する。よって、カップルドノードツリーに含まれるすべてのインデックスキーは０～（ｍ－１）ビット目の範囲が最短一致検索キーと一致するので、弁別ビット位置ｍと差分ビット位置ｄとの関係は、ｍ≦ｄである。よって必ずステップＳ９０８の判定が「はい」となる。

【０１３２】

ステップＳ９０８で「はい」と判定すると、ステップＳ９０９では探索経路スタック３１０のスタックポインタを２つ進める。

続くステップＳ９１０で探索経路スタック３１０からスタックポインタの指す配列番号を取り出す。

【０１３３】

次のステップＳ９１１では、ステップＳ９１０で得た配列番号と対をなす配列番号を得る。

ステップＳ９１１の実行後、処理はステップＳ９１４に進み、ステップＳ９１１で得られた配列番号を最短一致ノードの配列番号に設定する。そして、図９の処理が終了する。

【０１３４】

（Ｂ２）ルートノードがブランチノードであり、ステップＳ７０３で取得されたインデックスキーを格納するリーフノードへのルートノードからの経路上には、ルートノード以外のブランチノードが少なくとも１つ存在し、ルートノードの弁別ビット位置をｍとすると最短一致検索キーと０～ｍビット目までの範囲が一致するインデックスキーが存在しない場合

この場合、（Ｂ１）と違うのは、最短一致検索キーと０～ｍビット目までの範囲が一致するインデックスキーが存在しない点のみである。よって、ステップＳ９０７までの処理は（Ｂ１）の場合とまったく同じである。

【０１３５】

ステップＳ９０７の次のステップＳ９０８において、（Ｂ２）の場合は「いいえ」と判定され、ステップＳ９１４に移行する。この場合、配列番号はステップＳ９０５で取り出された配列番号、すなわちルートノードの配列番号である。よって、ステップＳ９１４ではルートノードの配列番号を最短一致ノードの配列番号に設定し、処理を終了する。

【０１３６】

（Ｂ３）ルートノードがブランチノードであり、ステップＳ７０３で取得されたインデックスキーを格納するリーフノードがルートノードからリンクされている場合

この場合、上記経路上には２つのノードのみが存在する。つまり、探索経路スタック３１０は、ルートノードの配列番号とインデックスキーを格納するリーフノードの配列番号が格納された状態である。

【０１３７】

（Ｂ３）の場合も、ステップＳ９０１～Ｓ９０３は（Ｂ１）の場合と同様である。しかし、ステップＳ９０３の次のステップＳ９０４における比較結果が（Ｂ１）の場合とは異なる。（Ｂ３）の場合、ステップＳ９０１で退避エリアに設定された値とスタックポインタが比較され、「差＝２」という結果が得られる。よって、処理は必ずステップＳ９１２に進む。

【０１３８】

ステップＳ９１２では探索経路スタック３１０のスタックポインタを１つ進める。次のステップＳ９１３で探索経路スタック３１０からスタックポインタの指す配列番号を取り出す。つまり、リーフノードの配列番号を取り出す。そして処理はステップＳ９１４に進み、ステップＳ９１３で取り出されたリーフノードの配列番号を最短一致ノードの配列番号に設定し、処理を終了する。

【０１３９】

（Ｂ４）ルートノードがリーフノードの場合

この場合、図７のステップＳ７０３で取得されたインデックスキーはルートノードに格納されたインデックスキーである。よって、最短一致検索キーとインデックスキーとの差分ビット位置が０より大きければ、インデックスキーは最短一致条件を満たし、ルートノードは最短一致ノードに該当する。また、図７のステップＳ７０５から明らかなとおり、図９の処理が行われるのは差分ビット位置が０より大きい場合のみなので、（Ｂ４）の場合、図９の処理によって必ずルートノードが最短一致ノードに設定される。具体的には次のとおりである。

【０１４０】

ステップＳ９０１は（Ｂ１）の場合と同様である。

続くステップＳ９０２において、スタックポインタの値を１減らす。また、（Ｂ４）の場合、探索経路スタック３１０に格納されている配列３０９の配列番号はルートノードの配列番号のみである。したがって、ステップＳ９０２の実行後、スタックポインタが指す配列番号は、すなわち、ルートノードの配列番号である。この状態で処理はステップＳ９０３に移行する。

【０１４１】

ステップＳ９０３では、探索経路スタック３１０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否かが判定される。（Ｂ４）の場合は、上記のとおりであるから、判定の結果は「はい」となり、ステップＳ９０４に移行する。

【０１４２】

ステップＳ９０４では、スタックポインタと退避エリアの値を比較する。（Ｂ４）の場合、「差＝１」という結果が得られ、この結果にしたがって処理はステップＳ９１３に移行する。

【０１４３】
ステップＳ９１３では、探索経路スタック３１０からスタックポインタの指す配列番号、つまりルートノードの配列番号を取り出す。そして、続くステップＳ９１４では、ルートノードの配列番号を、最短一致ノードの配列番号に設定し、図９の処理が終了する。
【０１４４】
次に、図１２を参照して、最短一致検索の具体例について説明する。図１２のカップルドノードツリーは図１１のものと全く同一である。
以下では最短一致検索キーとして“１０１０１０”が指定された場合について説明する。
【０１４５】
図７のステップＳ７０１～Ｓ７０５の処理は、図１１の例と全く同様なので説明を省略する。ステップＳ７０５で「いいえ」と判定されてステップＳ７０６、すなわち図９の処理が実行される。まず、ステップＳ９０１でスタックポインタの値を退避エリアに設定する。次に、ステップＳ９０２～Ｓ９０３の繰り返しにより探索経路スタック３１０を遡り、配列番号２２０が得られたらステップＳ９０３の判定が「はい」となってステップＳ９０４に進む。ここでスタックポインタと退避エリアの値が比較され、その結果ステップＳ９０５に進み、ステップＳ９０５～Ｓ９０７によってルートノード２１０ａの弁別ビット位置２３０ａが取り出される。弁別ビット位置２３０ａの値は０なので、ステップＳ９０８の判定は「はい」となり、ステップＳ９０９でスタックポインタを２つ進める。その結果、ステップＳ９１０では配列番号（２２１ｂ＋１）が取り出され、ステップＳ９１１ではそれと対の配列番号２２１ｂが取得される。よって、ステップＳ９１４では配列番号２２１ｂが最短一致ノードの配列番号に設定される。
【０１４６】
ここで処理は図７に戻り、ステップＳ７０７で最短一致ノードであるノード２１０ｆが検索開始ノードに設定され、ステップＳ７０８の処理、すなわち図５の最小値検索が行われる。図１２のとおり、最短一致ノードに設定されたブランチノードであるノード２１０ｆの子孫のリーフノードは、ノード２１０ｇとノード２１１ｇである。両ノードのインデックスキーの値は“１０００１１”と“１０００１０”である。いずれも、最短一致検索キー“１０１０１０”とは０～１ビット目が一致し、２ビット目が一致しない。
【０１４７】
ルートノードからの経路上のブランチノードにおける弁別ビット位置と対比すると、ルートノード２１０ａの弁別ビット位置２３０ａは０、次のノード２１１ｂの弁別ビット位置２３１ｂは２である。よって、０～０ビット目の範囲ではこの２つのインデックスキーは最短一致検索キーと一致し、１～２ビット目の範囲では不一致のビットが存在する。
【０１４８】
一方、カップルドノードツリーに含まれる他のインデックスキーは、部分一致条件を満足しないインデックスキー２５０ｃ、２５１ｄ、２５０ｅ、２５１ｅと、０～２ビット目が一致し、３ビット目が一致しないインデックスキー２５０ｈと２５１ｈである。よって、最短一致検索により、最短一致検索キーと部分一致するインデックスキーのうち、一致部分の短いものが優先的に検索されることが分かる。
【０１４９】
上記実施形態では、“１０００１１”と“１０００１０”のうちの最小値である“１０００１０”がステップＳ７０８により選択される。
なお、本発明の実施形態は上述のものに限られず、様々な変形が可能である。
【０１５０】
上記の実施形態では、最左という物理的なビット位置が最上位という論理的なビット位置に対応していたが、別の実施形態では、別の対応関係であってもよく、例えば、最右ビットを最上位ビットとしてもよい。図８のステップＳ８０６や図９のステップＳ９０８における判定は、論理的なビット位置にしたがった判定である。
【０１５１】

また、上記の最長一致検索では、最長一致条件を満たすインデックスキーのうちの最小値を結果として取得している。しかし、様々な実施形態に応じた適切なものを検索結果として得るように、上記の最長一致検索処理を変形することが可能である。例えば、最長一致条件を満たすインデックスキーのうちの最大値を、最長一致検索処理の結果として取得してもよい。

【０１５２】

同様に、最短一致検索も、検索結果として何を取得するかは実施形態に応じて様々である。

図９のステップＳ９０２～Ｓ９０３のループでは探索経路スタック３１０を１つずつ遡っているが、例えば、スタックポインタにルートノードの配列番号を指すスタックポインタを代入するステップでこのループを置き換えてもよい。

【０１５３】

また、最短一致検索の定義の細部は実施形態によって様々に変形可能である。図７のステップＳ７０４で得られた差分ビット位置が、ルートノードとその直下のブランチノードの弁別ビット位置の間の値である場合、図９の処理を変形して、ルートノードの直下のブランチノードを最短一致ノードに設定してもよい。

【０１５４】

また、ルートノードの弁別ビット位置が、０より大きく、ステップＳ７０４で得られた差分ビット位置より小さい場合、別の変形も可能である。この場合、最短一致検索キーとの差分ビット位置が最も上位なのは、ルートノードからリンクされたノード対のうち、ステップＳ７０３の検索で探索経路スタック３１０に配列番号が格納されなかった方のノードをルートノードとする部分木に含まれるリーフノードに格納されたインデックスキーである。よって、ステップＳ９０８の判定が「はい」の場合、ルートノードの弁別ビット位置の値が０より大きければ、ステップＳ９０９でスタックポインタを１つだけ進めるように図９の処理を変形してもよい。

【０１５５】

また、ステップＳ７０４で得られた差分ビット位置がルートノードの弁別ビット位置の値よりも小さいとき、図９の処理を変形し、ルートノードを最短一致ノードに設定してもよい。

【０１５６】

また、ルートノードの弁別ビット位置の値が０より大きく、ステップＳ７０４で得た差分ビット位置の値よりも小さければ、ステップＳ９１３で取り出した配列番号と対になる配列番号を最短一致ノードの配列番号として設定してもよい。

【０１５７】

なお、当然ながら、上記で説明した最長一致／最短一致検索の方法は、プログラムによってコンピュータに実行させることができる。

【図面の簡単な説明】

【０１５８】

【図１】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２】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】ビット列検索の基本動作を示すフローチャートである。

【図５】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最小値を求める処理を示すフローチャートである。

【図６】最長一致検索処理を示すフローチャートである。

【図７】最短一致検索処理を示すフローチャートである。

【図８】最長一致ノードを取得する処理を示すフローチャートである。

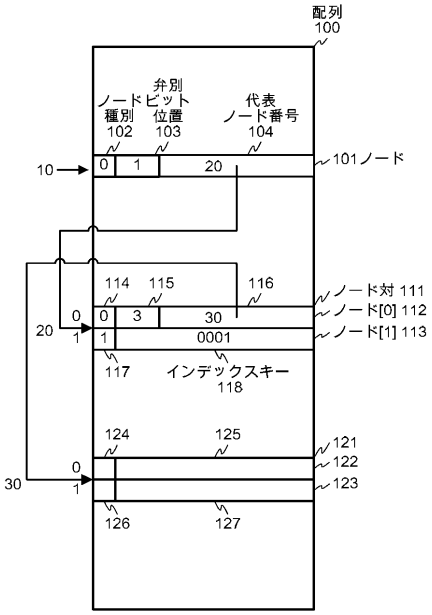
【図９】最短一致ノードを取得する処理を示すフローチャートである。

【図１０】差分ビット位置を取得する処理を示すフローチャートである。

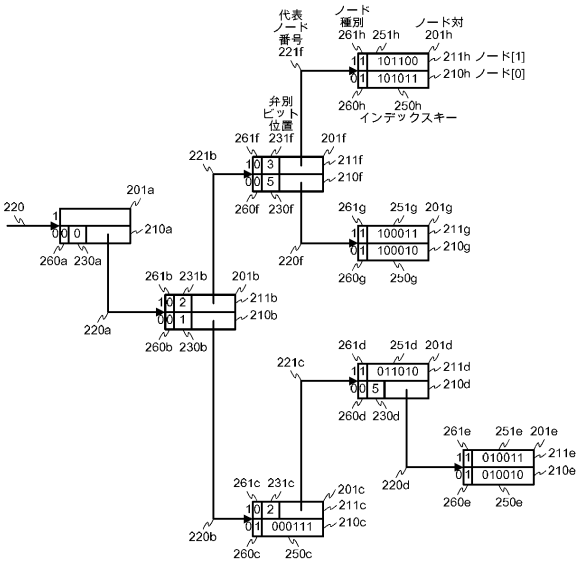
【図１１】最長一致検索の一例をカップルドノードツリーにより説明する図である。

【図１２】最短一致検索の一例をカップルドノードツリーにより説明する図である。
【図１３】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。
【符号の説明】
【０１５９】
１０、２０、３０　配列番号
１００　配列
１０１　ノード
１０２、１１４、１１７、１２４、１２６、２６０ａ～２６０ｈ、２６１ｂ～２６１ｈ　ノード種別
１０３、１１５、２３０ａ～２３０ｆ、２３１ｂ～２３１ｆ　弁別ビット位置
１０４、１１６、２２０、２２０ａ～２２０ｆ、２２１ｂ～２２１ｆ　代表ノード番号
１１１、１２１、２０１ａ～２０１ｈ　ノード対
１１２、１２２、２１０ａ～２１０ｈ　ノード［０］、代表ノード
１１３、１２３、２１１ｂ～２１１ｈ　ノード［１］、代表ノードと対をなすノード
１１８、１２５、１２６、２５０ｃ～２５０ｈ、２５１ｄ～２５１ｈ　インデックスキー
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
１７３０ａ～１７３０ｈ　検査ビット位置
１７４０ａ～１７４０ｆ　左リンク
１７４１ａ～１７４１ｆ　右リンク
１７５０ａ～１７５０ｈ　インデックスキー

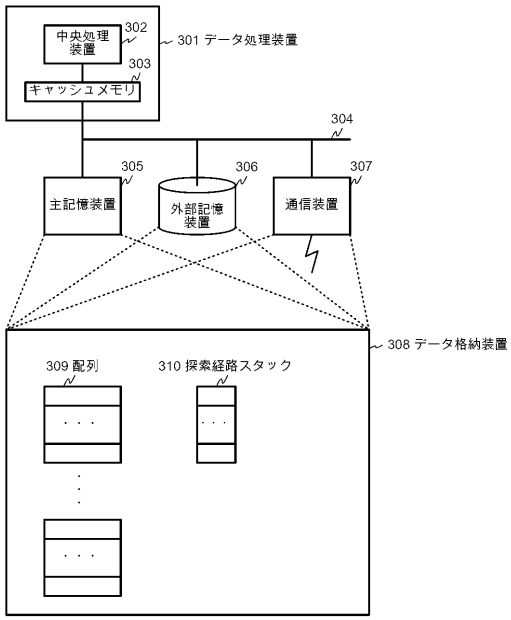
【図１】


配列
100
弁別
ノードビット
種別
位置
代表
ノード番号
102
103
104
101ノード
10
0
1
20
114
115
116
ノード対 111
ノード[0] 112
ノード[1] 113
0
3
30
20
1
0001
117
インデックスキー
118
124
125
121
122
123
30
126
127

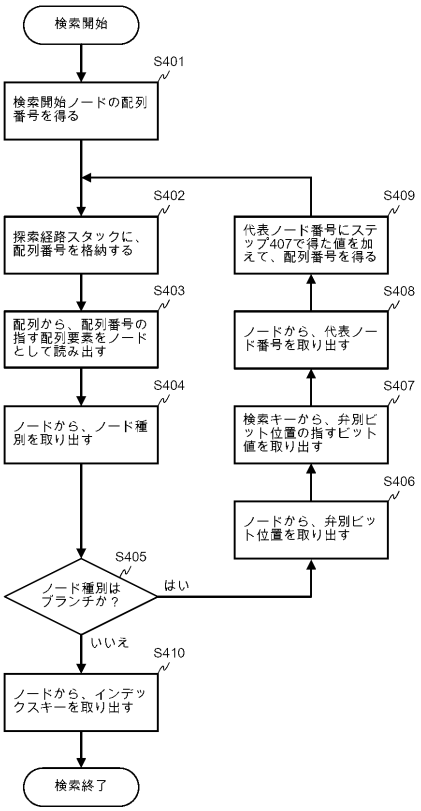
【図２】


代表
ノード
番号
221f
ノード
種別
261h
251h
ノード対
201h
101100
101011
211h ノード[1]
210h ノード[0]
260h
250h
インデックスキー
弁別
ビット
位置
261f
231f
201f
221b
211f
210f
260f
230f
201a
220
210a
260a
230a
261g
251g
201g
100011
100010
211g
210g
220f
260g
250g
261b
231b
201b
211b
210b
220a
260b
230b
261d
251d
201d
221c
011010
211d
210d
260d
230d
261e
251e
201e
010011
010010
211e
210e
220d
260e
250e
261c
231c
201c
211c
210c
000111
220b
260c
250c

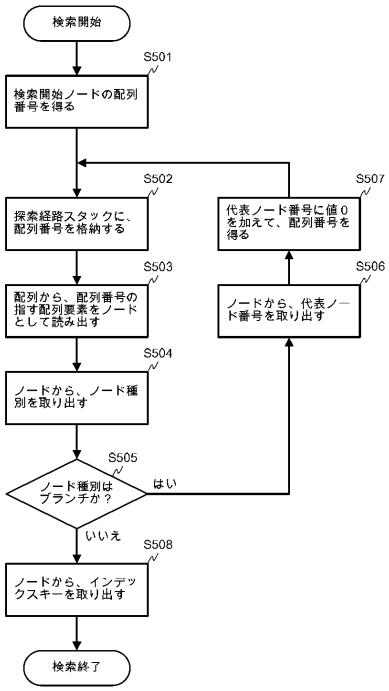
【図３】


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

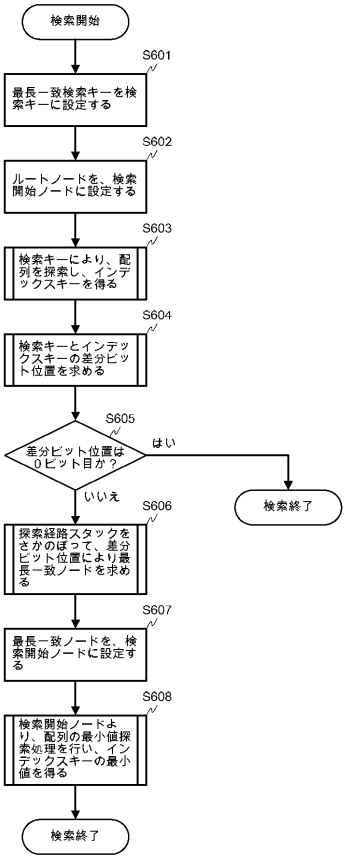
【図４】


検索開始
S401
検索開始ノードの配列
番号を得る
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
S406
ノードから、弁別ビッ
ト位置を取り出す
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
S408
ノードから、代表ノー
ド番号を取り出す
S409
代表ノード番号にステ
ップ407で得た値を加
えて、配列番号を得る
S410
ノードから、インデッ
クスキーを取り出す
検索終了

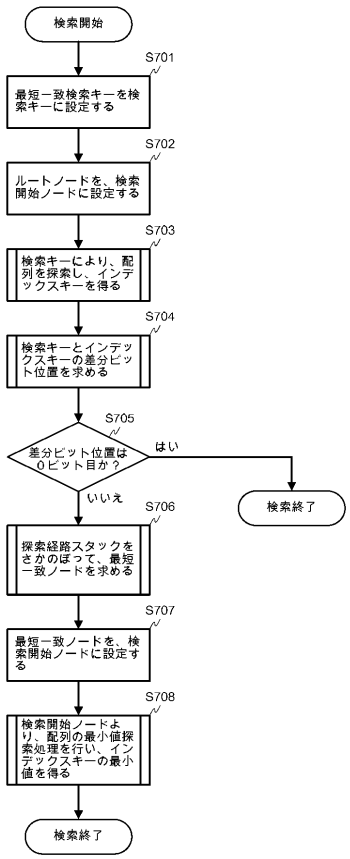
【図５】


検索開始
S501
検索開始ノードの配列番号を得る
S502
探索経路スタックに、配列番号を格納する
S503
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
S504
ノードから、ノード種別を取り出す
S505
ノード種別はブランチか？
はい
いいえ
S506
ノードから、代表ノード番号を取り出す
S507
代表ノード番号に値０を加えて、配列番号を得る
S508
ノードから、インデックスキーを取り出す
検索終了

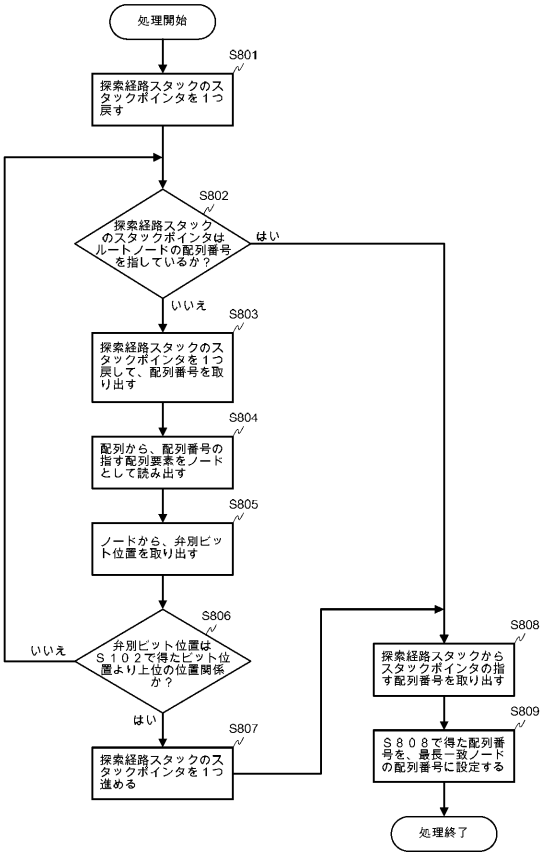
【図６】


検索開始
S601
最長一致検索キーを検索キーに設定する
S602
ルートノードを、検索開始ノードに設定する
S603
検索キーにより、配列を探索し、インデックスキーを得る
S604
検索キーとインデックスキーの差分ビット位置を求める
S605
差分ビット位置は０ビット目か？
はい
いいえ
検索終了
S606
探索経路スタックをさかのぼって、差分ビット位置により最長一致ノードを求める
S607
最長一致ノードを、検索開始ノードに設定する
S608
検索開始ノードより、配列の最小値探索処理を行い、インデックスキーの最小値を得る
検索終了

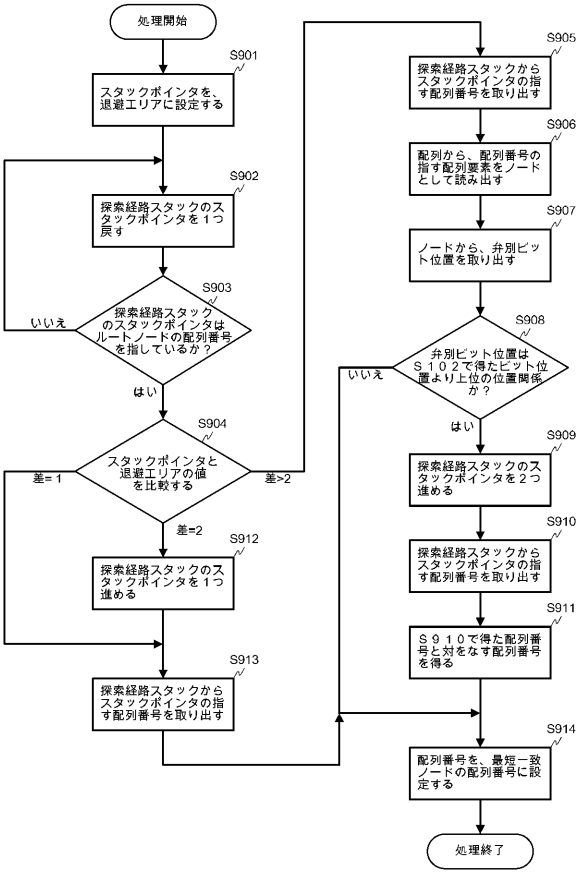
【図７】


検索開始
S701
最短一致検索キーを検索キーに設定する
S702
ルートノードを、検索開始ノードに設定する
S703
検索キーにより、配列を探索し、インデックスキーを得る
S704
検索キーとインデックスキーの差分ビット位置を求める
S705
差分ビット位置は０ビット目か？
はい
いいえ
検索終了
S706
探索経路スタックをさかのぼって、最短一致ノードを求める
S707
最短一致ノードを、検索開始ノードに設定する
S708
検索開始ノードより、配列の最小値探索処理を行い、インデックスキーの最小値を得る
検索終了

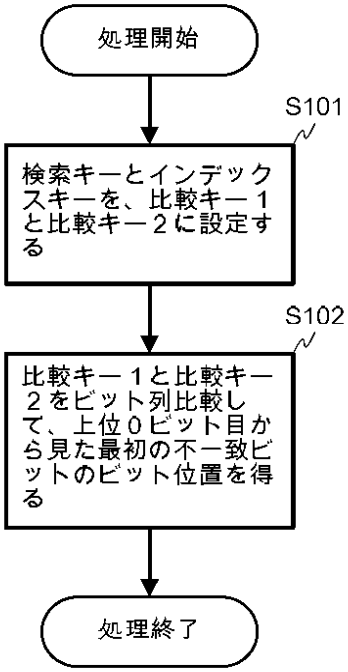
【図８】


処理開始
S801
探索経路スタックのスタックポインタを１つ戻す
S802
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
S803
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
S804
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
S805
ノードから、弁別ビット位置を取り出す
S806
弁別ビット位置はＳ１０２で得たビット位置より上位の位置関係か？
いいえ
はい
S807
探索経路スタックのスタックポインタを１つ進める
S808
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
S809
Ｓ８０８で得た配列番号を、最長一致ノードの配列番号に設定する
処理終了

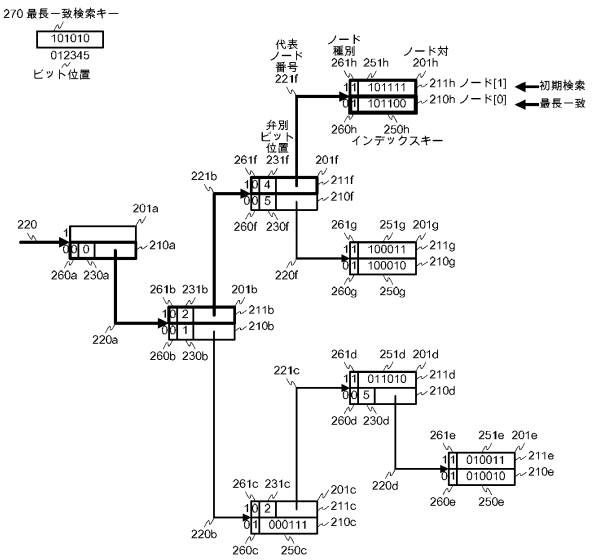
【図９】



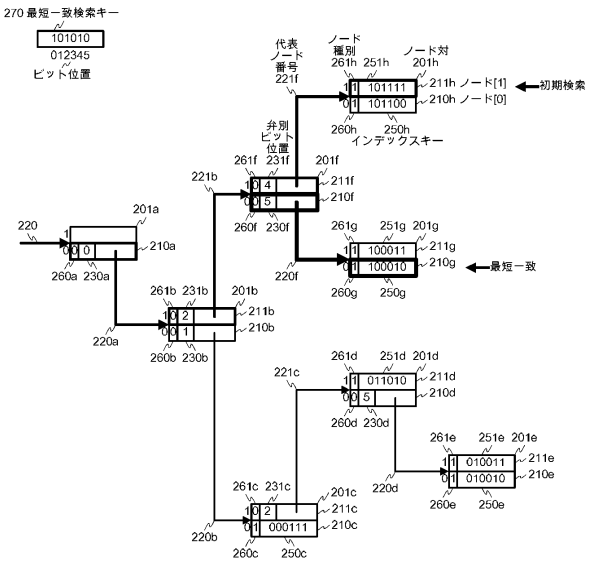
【図１０】



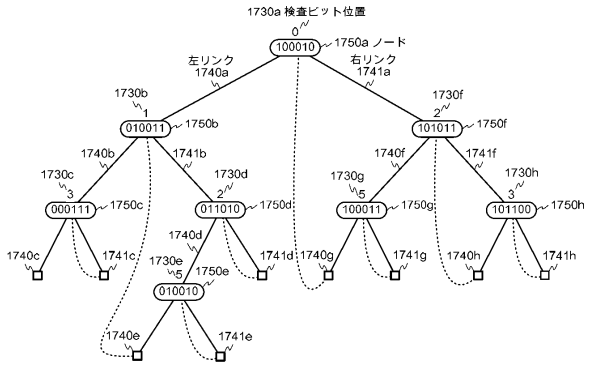
【図１１】



【図１２】



【図１３】


1730a 検査ビット位置
0
100010
1750a ノード
左リンク
1740a
右リンク
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
010010
1750e
1741d
1740g
1741g
1740h
1741h
1740e
1741e

フロントページの続き

(56)参考文献 特開２００１－３５７０７０（ＪＰ，Ａ）
後藤 大地，さわって学ぶデータ構造、つかって身につくアルゴリズム こちらＪａｖａ ＡＰＩ研究所！，ＪＡＶＡ ＰＲＥＳＳ Ｖｏｌ．３４，日本，（株）技術評論社，２００４年 ２月１５日，第34巻，２０２－２１０ページ
用語解説 パトリシアツリー，人工知能学会誌，日本，社団法人人工知能学会，１９９６年 ３月 １日，第１１巻 第2号，３３７－３３９ページ
セジウィック Ｒ．，アルゴリズム，株式会社近代科学社，１９９０年１０月１０日，第1巻，第1版，４９－５４ページ

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １７／３０
Ｇ０６Ｆ １２／００