(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2012 年 8 月 23 日(23.08.2012)

WIPO | PCT

(10) 国際公開番号
WO 2012/111078 A1

(51) 国際特許分類:
H03M 7/40 (2006.01) *G06F 5/00* (2006.01)

(21) 国際出願番号: PCT/JP2011/053059

(22) 国際出願日: 2011 年 2 月 14 日(14.02.2011)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(71) 出願人 (米国を除く全ての指定国について): 富士通株式会社(FUJITSU LIMITED) [JP/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号 Kanagawa (JP).

(72) 発明者; および

(75) 発明者/出願人 (米国についてのみ): 片岡　正弘 (KATAOKA, Masahiro) [JP/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内 Kanagawa (JP).

(74) 代理人: 酒井　昭徳(SAKAI, Akinori); 〒1006020 東京都千代田区霞が関３丁目２番５号　霞が関ビルディング２０階　酒井総合特許事務所 Tokyo (JP).

(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

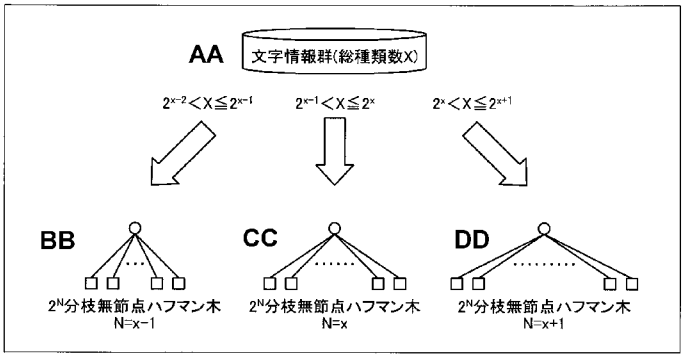
添付公開書類:

— 国際調査報告（条約第 21 条(3)）

(54) **Title**: GENERATED PROGRAM, GENERATING DEVICE AND GENERATING METHOD

(54) 発明の名称 : 生成プログラム、生成装置、および生成方法

[図1]



AA Character information group (total number of types X)

BB 2^N branch non-contact Huffman tree N = x-1

CC 2^N branch non-contact Huffman tree N = x

DD 2^N branch non-contact Huffman tree N = x + 1

(57) **Abstract**: According to the present invention, the size of an applied 2^N-branch non-contact Huffman tree is determined depending on where in a range the total number of types (X) of character information groups exists. The size of the 2^N-branch non-contact Huffman tree has the maximum number of branches, 2^N. The radicand N is an upper limit of the length of a compression code. Thus, when the size of the 2^N-branch non-contact Huffman tree is determined, the radicand (N) may be determined depending on the total number of types (X) of character information groups. Specifically, when the total number of types (X) of character information groups is $2^{x-2} < X \leq 2^{x-1}$, if the maximum number of branches (2^N) is at least 2^{x-1}, a Huffman tree can be established. To minimize the size, N = x-1 may be adopted. Further, when the total number of types (X) of character information groups is $2^{x-1} < X \leq 2^x$, if the maximum number of branches (2^N) is at least 2^x, a Huffman tree can be established. To minimize the size, N = x may be adopted.

(57) 要約:

[続葉有]

WO 2012/111078 A1

文字情報群の総種類数Ｘが、どの範囲にあるかで、適用される２^{N}分枝無節点ハフマン木のサイズが決定される。２^{N}分枝無節点ハフマン木のサイズとは、最大枝数２^{N}である。べき数Ｎは、圧縮符号長の上限となる。したがって、２^{N}分枝無節点ハフマン木のサイズを決定する場合は、べき数Ｎを文字情報群の総種類数Ｘに応じて決定すればよい。具体的には、文字情報群の総種類数Ｘが２$^{x-2}$＜Ｘ≦２$^{x-1}$である場合、少なくとも最大枝数２^{N}が２$^{x-1}$本あればハフマン木が構築できる。サイズを最小限にするには、Ｎ＝ｘ－１とすればよい。また、文字情報群の総種類数Ｘが２$^{x-1}$＜Ｘ≦２^{x}である場合、少なくとも最大枝数２^{N}が２^{x}本あればハフマン木が構築できる。サイズを最小限にするには、Ｎ＝ｘとすればよい。

明　細　書

発明の名称 ： 生成プログラム、生成装置、および生成方法

技術分野

[0001] 本発明は、ハフマン木を生成する生成プログラム、生成装置、および生成方法に関する。

背景技術

[0002] 従来、４，８、および１６分枝ハフマン木の生成元となる節点集合体から展開することにより、圧縮符号長が２，３，４の整数倍で規定された2^N分枝無節点ハフマン木が生成されている（たとえば、下記特許文献１を参照。）。

先行技術文献

特許文献

[0003] 特許文献1：特開２０１０－０９３４１４号公報

発明の概要

発明が解決しようとする課題

[0004] しかしながら、2^N分枝無節点ハフマン木では、圧縮符号長が２，３，４の整数倍でしか規定できなかったため、必然的に最大枝数2^Nのべき数Ｎが２，３，４の整数倍でしか規定できなかった。このため、最大枝数2^Nは４（＝2^2）、８（＝2^3）、１６（＝2^4）、６４（＝2^6）、２５６（＝2^8）、５１２（＝2^9）、１０２４（＝2^{10}）、４０９６（＝2^{12}）、１６３８４（＝2^{14}）となる。このため、最大枝数2^Nとして、２０４８（＝2^{11}）や８１９２（＝2^{13}）といった２，３，４の整数倍以外は規定できない。

[0005] 一方、テキストデータを構成する、単一文字、基礎単語、および予約語（以下、文字情報と呼ぶ）の種類数が2^m個（ただし、ｍは２，３，４の整数倍）より大きく2^{m+1}個（ただし、ｍ＋１は２，３，４の整数倍ではない）以下である場合、最大枝数2^NをＮ≧ｍ＋２としなければならない。たとえば、文字情報の種類数が2^{10}個より大きく2^{11}個以下である場合、最大枝数2^Nを2^{12}と

しなければならない。また、文字情報の種類数が2^{12}個より大きく2^{13}個以下である場合、最大枝数2^Nを2^{14}としなければならない。このため、2^N分枝無節点ハフマン木のサイズが大きくなり、さらなる改善の余地がある。

[0006] 本発明は、上述した従来技術による問題点を解消するため、文字情報の総種類数に最適なサイズで2^N分枝無節点ハフマン木を構築することができる生成プログラム、生成装置、および生成方法を提供することを目的とする。

課題を解決するための手段

[0007] 上述した課題を解決し、目的を達成するため、本発明の一観点として、対象ファイルに出現する文字情報群内の各文字情報の出現率に応じた生起確率で規定される圧縮符号長ごとに、前記文字情報の種類数を集計し、前記対象ファイルに出現する前記文字情報の総種類数に基づいて、最小圧縮符号長から最大圧縮符号長までの圧縮符号長群の中から、前記文字情報に割り当てられる圧縮符号長の上限長Ｎを決定し、集計された圧縮符号長ごとの前記文字情報の種類数のうち、決定された上限長Ｎを圧縮符号長とする前記文字情報の種類数を、前記上限長Ｎ以上の圧縮符号長での前記文字情報の種類数の総和に補正し、補正された補正後の前記圧縮符号長ごとの前記文字情報の種類数に基づいて、決定された上限長Ｎを最大枝数とし、前記文字情報の各々の生起確率に応じた圧縮符号長の圧縮符号を葉とする2^N分枝無節点ハフマン木を構築する生成プログラム、生成装置、および生成方法が提案される。

発明の効果

[0008] 本発明にかかる生成プログラム、生成装置、および生成方法によれば、文字情報の総種類数に最適なサイズで2^N分枝無節点ハフマン木を構築することができるという効果を奏する。

図面の簡単な説明

[0009] [図1]2^N分枝無節点ハフマン木のサイズ決定方法の一例を示す説明図である。

[図2]2^N分枝無節点ハフマン木の生成までの流れを示す説明図である。

[図3]図２の（１）出現回数の集計の詳細を示す説明図である。

[図4]図２の（２）圧縮符号長算出の詳細（Ｎ＝１１）を示す説明図である。

[図5]図２の（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１１）を示す説明図である。

[図6]文字情報ごとの補正結果を示す説明図である。

[図7]葉へのポインタ生成の詳細（Ｎ＝１１）を示す説明図（その１）である。

[図8]葉へのポインタ生成の詳細（Ｎ＝１１）を示す説明図（その２）である。

[図9]葉へのポインタ生成の詳細（Ｎ＝１１）を示す説明図（その３）である。

[図10]葉へのポインタ生成の詳細（Ｎ＝１１）を示す説明図（その４）である。

[図11]葉へのポインタ生成の詳細（Ｎ＝１１）を示す説明図（その５）である。

[図12]葉へのポインタ生成の詳細（Ｎ＝１１）を示す説明図（その６）である。

[図13]図２の（７）2^N分枝無節点ハフマン木の構築の詳細（Ｎ＝１１）を示す説明図である。

[図14]図２の（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１２）を示す説明図である。

[図15]葉へのポインタ生成の詳細（Ｎ＝１２）を示す説明図（その１）である。

[図16]葉へのポインタ生成の詳細（Ｎ＝１２）を示す説明図（その２）である。

[図17]葉へのポインタ生成の詳細（Ｎ＝１２）を示す説明図（その３）である。

[図18]葉へのポインタ生成の詳細（Ｎ＝１２）を示す説明図（その４）である。

[図19]葉へのポインタ生成の詳細（Ｎ＝１２）を示す説明図（その５）であ

る。

[図20]葉へのポインタ生成の詳細（Ｎ＝１２）を示す説明図（その６）である。

[図21]葉へのポインタ生成の詳細（Ｎ＝１２）を示す説明図（その７）である。

[図22]（７）２^{N}分枝無節点ハフマン木の構築の詳細（Ｎ＝１２）を示す説明図である。

[図23]図１の（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１３）を示す説明図である。

[図24]葉へのポインタ生成の詳細（Ｎ＝１３）を示す説明図（その１）である。

[図25]葉へのポインタ生成の詳細（Ｎ＝１３）を示す説明図（その２）である。

[図26]葉へのポインタ生成の詳細（Ｎ＝１３）を示す説明図（その３）である。

[図27]葉へのポインタ生成の詳細（Ｎ＝１３）を示す説明図（その４）である。

[図28]葉へのポインタ生成の詳細（Ｎ＝１３）を示す説明図（その５）である。

[図29]葉へのポインタ生成の詳細（Ｎ＝１３）を示す説明図（その６）である。

[図30]葉へのポインタ生成の詳細（Ｎ＝１３）を示す説明図（その７）である。

[図31]葉へのポインタ生成の詳細（Ｎ＝１３）を示す説明図（その８）である。

[図32]図２の（７）２^{N}分枝無節点ハフマン木の構築の詳細（Ｎ＝１３）を示す説明図である。

[図33]葉の構造体を示す説明図である。

[図34]高位文字コードの構造体を示す説明図である。

[図35]分割文字コードの構造体を示す説明図である。

[図36]特殊単語の構造体を示す説明図（その１）である。

[図37]特殊単語の構造体を示す説明図（その２）である。

[図38]実施の形態にかかる生成装置のハードウェア構成例を示すブロック図である。

[図39]生成装置の機能的構成例を示すブロック図である。

[図40]ＵＴＦ１６での文字情報の分類例を示す説明図である。

[図41]ＡＳＣＩＩコードでの文字情報の分類例を示す説明図である。

[図42]シフトＪＩＳコードでの文字情報の分類例を示す説明図である。

[図43]2^N分枝無節点ハフマン木の生成処理手順（前半）を示すフローチャートである。

[図44]2^N分枝無節点ハフマン木の生成処理手順（後半）を示すフローチャートである。

[図45]図４３に示した第１集計処理（ステップＳ４３０１）の詳細な処理手順を示すフローチャートである。

[図46]図４５に示した対象ファイルＦｉの集計処理（ステップＳ４５０３）の詳細な処理手順を示すフローチャートである。

[図47]図４６に示した特殊単語集計処理（ステップＳ４６０２）の詳細な処理手順を示すフローチャートである。

[図48]図４７に示した最長一致検索処理（ステップＳ４７０１）の詳細な処理手順を示すフローチャートである。

[図49]図４３に示した第２集計処理（ステップＳ４３０２）の詳細な処理手順を示すフローチャートである。

[図50]図４４に示した補正B^+処理（ステップＳ４４０３）の詳細な処理手順を示すフローチャートである。

[図51]図４４に示した補正B^-処理（ステップＳ４４０４）の詳細な処理手順を示すフローチャートである。

[図52]図５０および図５１に示した更新処理（ステップＳ５００１）の詳細な処理手順を示すフローチャートである。

[図53]図４４に示した枝数特定処理（ステップＳ４４０６）の詳細な処理手順を示すフローチャートである。

[図54]図４４に示した構築処理（ステップＳ４４０７）の詳細な処理手順を示すフローチャートである。

[図55]図５４に示した葉へのポインタ生成処理（ステップＳ５４０３）の詳細な処理手順を示すフローチャートである。

[図56]補正B^{+}処理（ステップＳ４４０３）の他の例の詳細な処理手順を示すフローチャートである。

[図57]補正B^{-}処理（ステップＳ４４０４）の他の例の詳細な処理手順を示すフローチャートである。

[図58]図５６に示した補正B^{+}処理の他の例を適用した場合における、図２の（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１２）を示す説明図である。

[図59]補正B^{-}処理の他の例を適用した場合における、図２の（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１１）を示す説明図である。

[図60]圧縮対象文字列の一例を示す説明図である。

[図61]図６０に示した圧縮対象文字列の圧縮符号を示す説明図である。

[図62]圧縮対象文字列の他の例を示す説明図である。

[図63]図６２に示した圧縮対象文字列の圧縮符号を示す説明図である。

[図64]2^{N}分枝無節点ハフマン木を用いた圧縮処理の具体例を示す説明図である。

[図65]コンピュータが自動実行する2^{N}分枝無節点ハフマン木を用いたファイル圧縮処理手順を示すフローチャートである。

[図66]図６５に示した圧縮処理（ステップＳ６５０３）の詳細な処理手順を示すフローチャート（その１）である。

[図67]図６５に示した圧縮処理（ステップＳ６５０３）の詳細な処理手順を

示すフローチャート（その２）である。

[図68]図６５に示した圧縮処理（ステップＳ６５０３）の詳細な処理手順を示すフローチャート（その３）である。

[図69]図６１に示した圧縮符号列の伸長処理例を示す説明図（その１）である。

[図70]図６１に示した圧縮符号列の伸長処理例を示す説明図（その２）である。

[図71]図６１に示した圧縮符号列の伸長処理例を示す説明図（その３）である。

[図72]図６１に示した圧縮符号列の伸長処理例を示す説明図（その４）である。

[図73]図６１に示した圧縮符号列の伸長処理例を示す説明図（その５）である。

[図74]図６３に示した圧縮符号列の伸長処理例を示す説明図（その１）である。

[図75]図６３に示した圧縮符号列の伸長処理例を示す説明図（その２）である。

[図76]図６３に示した圧縮符号列の伸長処理例を示す説明図（その３）である。

[図77]図６３に示した圧縮符号列の伸長処理例を示す説明図（その４）である。

[図78]図６３に示した圧縮符号列の伸長処理例を示す説明図（その５）である。

[図79] 2^N分枝無節点ハフマン木を用いた伸長処理手順を示すフローチャート（その１）である。

[図80] 2^N分枝無節点ハフマン木を用いた伸長処理手順を示すフローチャート（その２）である。

発明を実施するための形態

[0010] 以下に添付図面を参照して、本発明にかかる生成プログラム、生成装置、および生成方法の実施の形態を詳細に説明する。なお、本明細書において、「文字情報」とは、テキストデータを構成する単一文字、基礎単語、予約語などの情報である。単一文字とは１つの文字コードで表現される文字である。単一文字の文字コード長は、文字コード種により異なる。

[0011] たとえば、ＵＴＦ（Ｕｎｉｃｏｄｅ Ｔｒａｎｓｆｏｒｍａｔｉｏｎ Ｆｏｒｍａｔ）１６の場合は１６ビットコード、ＡＳＣＩＩ（Ａｍｅｒｉｃａｎ Ｓｔａｎｄａｒｄ Ｃｏｄｅ ｆｏｒ Ｉｎｆｏｒｍａｔｉｏｎ Ｉｎｔｅｒｃｈａｎｇｅ）コードの場合は８ビットコード、シフトＪＩＳ（Ｊａｐａｎｅｓｅ Ｉｎｄｕｓｔｒｉａｌ Ｓｔａｎｄａｒｄ）コードの場合は８ビットコードである。シフトＪＩＳコードで日本語の文字を表現する場合は、２個の８ビットコードを組み合わせることとなる。

[0012] また、単一文字以外に、たとえば、特定の文字列で表現される単語や予約語が挙げられる。単語としては、たとえば、児童や生徒が学校教育で学習すべき数百～数千の基礎単語が挙げられる。基礎単語は出現頻度が高い文字列である。予約語とは、予め決められた文字列であり、たとえば、ＨＴＭＬのタグ（たとえば、＜ｂｒ＞）が挙げられる。基礎単語および予約語を「特殊単語」と称す。なお、本実施の形態では、文字コードとしてＵＴＦ１６を例に挙げて説明する。

[0013] ＜2^N分枝無節点ハフマン木のサイズ決定方法＞

図１は、2^N分枝無節点ハフマン木のサイズ決定方法の一例を示す説明図である。2^N分枝無節点ハフマン木とは、根から分岐する枝が2^N本あり、１または複数本の枝で葉を直接ポイントするハフマン木である。節点（内部節点）はない。節点がなく直接葉にヒットするため、節点を有する通常のハフマン木に比べて、伸長速度の高速化を図ることができる。葉は、該当する文字情報とその圧縮符号を含む構造体である。葉の構造体とも呼ぶ。葉に割り当てられる枝数は、割当先の葉に存在する圧縮符号の圧縮符号長に依存する。これらの詳細については、後述する。

[0014] 文字情報群の総種類数Ｘが、どの範囲にあるかで、適用される2^N分枝無節点ハフマン木のサイズが決定される。2^N分枝無節点ハフマン木のサイズとは、最大枝数2^Nである。べき数Ｎは、圧縮符号長の上限となる。したがって、2^N分枝無節点ハフマン木のサイズを決定する場合は、べき数Ｎを文字情報群の総種類数Ｘに応じて決定すればよい。

[0015] 具体的には、文字情報群の総種類数Ｘが$2^{x-2}<X\leqq 2^{x-1}$である場合、少なくとも最大枝数2^Nが2^{x-1}本あればハフマン木が構築できる。サイズを最小限にするには、Ｎ＝ｘ－１とすればよい。また、文字情報群の総種類数Ｘが$2^{x-1}<X\leqq 2^x$である場合、少なくとも最大枝数2^Nが2^x本あればハフマン木が構築できる。サイズを最小限にするには、Ｎ＝ｘとすればよい。また、文字情報群の総種類数Ｘが$2^x<X\leqq 2^{x+1}$である場合、少なくとも最大枝数2^Nが2^{x+1}本あればハフマン木が構築できる。サイズを最小限にするには、Ｎ＝ｘ＋１とすればよい。

[0016] たとえば、文字情報群の総種類数Ｘ＝１３０５個である場合、$2^{10}<X\leqq 2^{11}$となるため、最大枝数2^Nのべき数Ｎは、Ｎ＝１１となる。したがって、2^{11}分枝無節点ハフマン木が生成され、最大でも１１ビットの圧縮符号長の圧縮符号で文字情報を圧縮することができる。

[0017] また、文字情報群の総種類数Ｘ＝３０４８個である場合、$2^{11}<X\leqq 2^{12}$となるため、最大枝数2^Nのべき数Ｎは、Ｎ＝１２となる。したがって、2^{12}分枝無節点ハフマン木が生成され、最大でも１２ビットの圧縮符号長の圧縮符号で文字情報を圧縮することができる。

[0018] さらに、文字情報群の総種類数Ｘ＝５４０１個である場合、$2^{12}<X\leqq 2^{13}$となるため、最大枝数2^Nのべき数Ｎは、Ｎ＝１３となる。したがって、2^{13}分枝無節点ハフマン木が生成され、最大でも１３ビットの圧縮符号長の圧縮符号で文字情報を圧縮することができる。

[0019] ＜2^N分枝無節点ハフマン木の生成までの流れ＞

図２は、2^N分枝無節点ハフマン木の生成までの流れを示す説明図である。

[0020] （１）出現回数の集計

まず、生成装置は、対象ファイル群に存在する文字情報の出現回数を計数する。計数対象となる対象ファイル群は、たとえば、文書ファイル、Ｗｅｂページなどの電子データであり、たとえば、テキスト形式、ＨＴＭＬ（ＨｙｐｅｒＴｅｘｔ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ）形式、ＸＭＬ（Ｅｘｔｅｎｓｉｂｌｅ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ）形式の電子データである。また、単一の対象ファイルを計数対象としてもよい。集計結果は、出現回数の降順にソートされ、出現回数の大きい方から昇順の順位がつけられる。なお、ここでは、文字情報の総種類数は、例として１３０５個（＜２０４８（＝2^{11}））とする。

[0021] （２）圧縮符号長算出

つぎに、（１）で得られた集計結果を基にして、生成装置は、文字情報ごとの圧縮符号長を算出する。具体的には、生成装置は、文字情報ごとに、出現率を算出する。出現率は、文字情報の出現回数を全文字情報の総出現回数で割ることで得られる。そして、生成装置は、出現率に対応する生起確率を求め、生起確率から圧縮符号長を導き出す。

[0022] 生起確率は、１／2^{x}で表現される。ｘはべき数である。圧縮符号長は、生起確率のべき数ｘとなる。具体的には、出現率が生起確率の以下のどの範囲であるかで圧縮符号長が決定される。ＡＲは出現率である。

１／2^{0}＞ＡＲ≧１／2^{1}・・・圧縮符号長は１ビット。

１／2^{1}＞ＡＲ≧１／2^{2}・・・圧縮符号長は２ビット。

１／2^{2}＞ＡＲ≧１／2^{3}・・・圧縮符号長は３ビット。

１／2^{3}＞ＡＲ≧１／2^{4}・・・圧縮符号長は４ビット。

・

・

・

１／2^{N-1}＞ＡＲ≧１／2^{N}・・・圧縮符号長はＮビット。

[0023] （３）葉数特定

つぎに、生成装置は、圧縮符号長ごとに葉数を集計することで圧縮符号長

ごとの葉数を特定する。図２では、最大圧縮符号長が１７ビットとする。また、葉数とは、文字情報の種類数である。したがって、圧縮符号長５ビットの葉数が２である場合、５ビットの圧縮符号が割り当てられる文字情報が２つ存在することを示している。

[0024] （４）葉数補正

つぎに、生成装置は、葉数を補正する。具体的には、生成装置は、枝数の上限2^Nのべき数Ｎが最大圧縮符号長となるように補正する。たとえば、べき数Ｎ＝１１の場合、圧縮符号長１１ビット～１７ビットまでの葉数の総和を、補正後の圧縮符号長１１ビットの葉数にする。そして、生成装置は、圧縮符号長ごとに葉当たりの枝数を割り当てる。具体的には、補正後の圧縮符号長に対し、その降順に、2^0、2^1、2^2、2^3、2^4、2^5、2^6、2^7として葉当たりの枝数を決定する。

[0025] たとえば、図２では、圧縮符号長１１ビットの圧縮符号が割り当てられる文字情報の総数（葉数）は１２１５個であるが、その葉当たりの枝数は１である。圧縮符号長１１ビットの圧縮符号が割り当てられる文字情報については、それぞれ１本の枝しか割り当てられないこととなる。一方、圧縮符号長６ビットの圧縮符号が割り当てられる文字情報の総数（葉数）は６個であるが、その葉当たりの枝数は３２である。圧縮符号長６ビットの圧縮符号が割り当てられる文字情報については、それぞれ３２本の枝が割り当てられることとなる。

[0026] （５）葉の構造体生成

つぎに、生成装置は、葉の構造体を生成する。葉の構造体とは、文字情報とその圧縮符号長とその圧縮符号長での圧縮符号が対応付けられたデータ構造体である。たとえば、出現順位が１位である文字「０」の圧縮符号長は６ビットであり、圧縮符号は「００００００」となる。図２の例では、文字情報の種類数（葉数）は１３０５個であるため、葉Ｌ１の構造体～葉Ｌ１３０５の構造体が生成されることとなる。

[0027] （６）葉へのポインタ生成

つぎに、生成装置は、葉の構造体ごとに葉へのポインタを生成する。葉へのポインタは、そのポイント先となる葉の構造体内の圧縮符号に、その葉当たりの枝数分の番号に相当するビット列を連結したビット列である。たとえば、葉Ｌ１である文字「０」に割り当てられた圧縮符号「０００００００」の圧縮符号長は６ビットであるため、葉Ｌ１当たりの枝数は３２本である。

[0028] したがって、葉Ｌ１へのポインタの先頭６ビットは、圧縮符号「０００００００」となる。後続ビット列は、葉Ｌ１当たりの枝数で表現される３２（＝2^5）種のビット列となる。すなわち、３２種の５ビットのビット列が圧縮符号「０００００００」の後続ビット列となる。したがって、葉Ｌ１へのポインタは、先頭６ビットが「０００００００」で固定された３２種の１１ビットのビット列となる。なお、葉当たりの枝数が１本の場合は、葉へのポインタは１個であり、圧縮符号とその葉へのポインタは同一ビット列となる。

[0029] （７）2^N分枝無節点ハフマン木の構築

最後に、生成装置は、2^N分枝無節点ハフマン木を構築する。具体的には、葉のポインタを根とすることで、葉の構造体を直接指定する2^N分枝無節点ハフマン木が構築される。圧縮符号列が、先頭６ビットが「０００００００」の１１ビットのビット列である場合、後続の３ビットが３２種のいずれのビット列であっても、2^N分枝無節点ハフマン木により文字「０」の葉Ｌ１の構造体をポイントすることができる。

[0030] ＜（１）出現回数の集計の詳細＞

図３は、図２の（１）出現回数の集計の詳細を示す説明図である。生成装置は、対象ファイル群Ｆｓ内の文字コードについて出現回数を集計して降順にソートする。そして、高位（たとえば、１～１０２４位まで）の文字コードと低位（１０２５位以下）の文字コードに分ける。

[0031] 高位の文字コードは出現頻度が高いため、１６ビットコードとして圧縮符号を割り当てる。低位の文字コードは、上位８ビットと下位８ビットに分割する。１６ビットコードでは最大で６５５３６（２５６×２５６）種の文字を扱えるが、低位の文字コードを、８ビットで分割することにより、６万種

以上ある低位の文字コードを２５６種の分割文字コードに抑えることができる。

[0032] たとえば、漢字の「兎」の１６ビットの文字コードは“０ｘ５１４Ｅ”であるため、上位８ビット“０ｘ５１”と下位８ビット“０ｘ４Ｅ”に分割する。同様に、漢字の「兆」の１６ビットの文字コードは“０ｘ５１４６”であるため、上位８ビット“０ｘ５１”と下位８ビット“０ｘ４６”に分割する。そして、分割された８ビットの分割文字コードの出現頻度を集計する。この「兎」および「兆」の例では、分割文字コード“０ｘ５１”の出現頻度は２回、分割文字コード“０ｘ４Ｅ”，“０ｘ４６”は１回となる。

[0033] また、＜ｐ＞や＜ｐ／＞といったタグなどを予約語とする。予約語は数十種（例として２５種）決めておく。また、必要に応じて基礎単語も決めておく。基礎単語は例として４０９６種としておく。そして、高位文字コード群、分割文字コード群および予約語群、基礎単語群を混在させて出現頻度を再集計し、出現頻度の降順に再ソートする。再ソート結果が図１の集計結果となる。

[0034] 基礎単語群を集計しない場合、出現頻度の降順にソートされた１３０５種の文字情報が得られる。文字情報の種類数をこの程度の数に抑えることで、１６ビットの文字コードを１２ビット以下の圧縮符号に圧縮し、圧縮符号を１６ビットの文字コードに伸長することが可能となる。

[0035] また、基礎単語群を集計する場合、出現頻度の降順にソートされた５４０１種の文字情報が得られる。４０９６種の基礎単語を集計すると、１６ビットの文字コードを１３ビット以下の圧縮符号に圧縮することができ、圧縮符号を１６ビットの文字コードに伸長することが可能となる。また、基礎単語は１６ビットコードが複数連結された文字列であるため、文字数がｎ個である場合、ｎ×１６ビットコードを１３ビット以下の圧縮符号に圧縮でき、圧縮率の向上を図ることができる。また、その圧縮符号をｎ×１６ビットコードの基礎単語に伸長でき、伸長速度を高速化することができる。

[0036] ＜（２）圧縮符号長算出の詳細（Ｎ＝１１）＞

図４は、図２の（２）圧縮符号長算出の詳細（Ｎ＝１１）を示す説明図である。図４の文字情報テーブルは、図３の集計結果を反映したテーブルであり、文字情報ごとに順位項目、伸長種別項目、コード項目、文字項目、出現回数項目、総回数項目、出現率項目、補正前での生起確率項目および圧縮符号長項目が設定されている。このうち、順位項目～総回数項目までが図３の再ソート結果で得られた情報である。

[0037] ここで、順位項目には、文字情報の出現回数の降順に順位（昇順）が書き込まれている。文字情報項目のうち伸長種別項目には、文字情報の種別が書き込まれる。「１６」は１６ビットコード（の単一文字）を示している。「８」は８ビットの分割文字コードを示している。「特」は特殊単語（基礎単語または予約語）を示している。

[0038] 文字情報項目のうちコード項目には、文字コードまたは分割文字コードが書き込まれている。特殊単語の場合は空欄とする。文字情報項目のうち文字項目には、文字や特殊単語が書き込まれている。分割文字コードの場合は空欄とする。出現回数項目には、対象ファイル群での文字情報の出現回数が書き込まれている。総回数項目には、全文字情報の総出現回数が書き込まれている。

[0039] 出現率項目には、出現回数を総回数で割り算した値が出現率として書き込まれている。補正前項目の生起確率項目には、出現率に対応する生起確率が書き込まれている。圧縮符号長項目には、生起確率に応じた圧縮符号長、すなわち、生起確率１／2^{y}のべき数ｙが圧縮符号長として書き込まれている。

[0040] ＜（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１１）＞

図５は、図２の（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１１）を示す説明図である。図４の文字情報テーブルを圧縮符号長単位で葉数（文字情報の総種類数）を集計した結果が、図５における補正前の葉数となる。ここで、補正Ａとは、圧縮符号長の上限長Ｎ（すなわち、2^{N}分枝無節点ハフマン木の最大枝数2^{N}のべき数Ｎ）以上の圧縮符号長に割り当てられた葉数を、圧縮符号長の上限長Ｎに集約する補正である。この場合、補正前での最

大圧縮符号長は１７ビットであるが、文字情報の総種類数が１３０５種であるため、圧縮符号長の上限長Ｎは、Ｎ＝１１となる。したがって、補正Ａでは、圧縮符号長１１ビットの葉数が、圧縮符号長が１１ビット～１７ビットの葉数の総和（１１９０個）となる。

[0041] そして、生成装置は、生起確率総和を求める。圧縮符号長ごとの生起確率は決められているため（５ビットなら１／2^5）、圧縮符号長ごとに生起確率を葉数で乗じることで、圧縮符号長ごとの乗算結果が得られる。たとえば、補正Ａにおける圧縮符号長５ビットの葉数は２である。圧縮符号長５ビットの生起確率は、１／2^5である。したがって、補正Ａにおける圧縮符号長５ビットの生起確率は、２×（１／2^5）＝１／2^4となる。圧縮符号長６ビット以降も同様に補正Ａにおける圧縮符号長生起確率を求める。そして、補正Ａ後における各圧縮符号長の生起確率を合計することで、補正Ａでの生起確率総和が得られる。

[0042] そして、生成装置は、生起確率総和が１以下であるか否かを判断する。しきい値ｔは０＜ｔ≦１である。しきい値ｔを設けたくない場合は、ｔ＝１とすればよい。しきい値ｔ未満であれば、補正Ｂに移行する。しきい値ｔ以上１以下である場合は、補正Ｂに移行せず、この時点での圧縮符号長ごとの葉数で確定する。

[0043] 補正Ｂは、補正Ａでの圧縮符号長群（５ビット～１２ビット）は変えずに、葉数を更新する補正である。具体的には、補正Ａでの生起確率総和が、しきい値ｔ以上１以下でない場合におこなわれる補正である。より具体的には、補正Ｂは２種類ある。

[0044] １つ目は、生起確率総和がしきい値ｔ未満である場合、生起確率総和が１以下の最大値が得られるまで、たとえば、最大漸近値に収束するまで、生起確率総和を増加させる補正（以下、補正B^+）である。もう１つは、生起確率総和が１より大きい場合、生起確率総和が１以下に割り込んでから１以下の最大値が得られるまで、たとえば、最大漸近値に収束するまで、生起確率総和を減少させる補正（以下、補正B^-）である。

[0045] 図５に示した例では、補正Ａでの生起確率総和が「１．１４６」であるため、補正Ｂ⁻をおこなうこととなる。なお、補正Ｂ⁺および補正Ｂ⁻のいずれの補正Ｂであっても、葉数を生起確率総和で割るという同じ補正をおこなう。

[0046] まず、補正Ｂ⁻の１回目（補正Ｂ⁻１）では、圧縮符号長ごとの補正Ａでの葉数を、前回の補正（この場合は補正Ａ）の生起確率の総和（１．１４６）で割ることで、葉数を更新する。なお、小数点以下は切り捨てでも、四捨五入でもよい。なお、補正Ａでの圧縮符号長の上限長Ｎ（Ｎ＝１１ビット）については、前回の補正（この場合は補正Ａ）の生起確率総和（１．１４６）で割ることはせず、葉の総数（１３０５個）から、補正Ｂ⁻１での圧縮符号長ごとの葉の総数（圧縮符号長の上限長Ｎの葉数除く）を引くことで、圧縮符号長の上限長Ｎの葉数を求める。この場合は、１２０８個である。

[0047] このあと、生成装置は、補正Ａの場合と同様の計算処理により、補正Ｂ⁻１での生起確率総和を求める。そして、生成装置は、補正Ｂ⁻１での生起確率総和が１以下の最大漸近値に収束したか否かを判断する。補正Ｂ⁻１での生起確率総和が１以下の最大漸近値に収束していなければ、補正Ｂ⁻の２回目（補正Ｂ⁻２）に移行する。最大漸近値に収束した場合は、補正Ｂ⁻２に移行せず、この時点での圧縮符号長ごとの葉数で確定する。補正Ｂ⁻１で更新された生起確率総和「１．０４２」は１より大きいため、最大漸近値に収束しておらず、補正Ｂ⁻２に移行する。

[0048] 補正Ｂ⁻２では、圧縮符号長ごとの補正Ｂ⁻１での葉数を、前回の補正（この場合は補正Ｂ⁻１）の生起確率総和（１．０４２）で割ることで、葉数を更新する。なお、小数点以下は切り捨てでも、四捨五入でもよい。なお、補正Ｂ⁻１での圧縮符号長の上限長Ｎ（Ｎ＝１１ビット）については、前回の補正（この場合は補正Ｂ⁻１）の生起確率総和（１．０４２）で割ることはせず、葉の総数（１３０５個）から、補正Ｂ⁻２での圧縮符号長ごとの葉の総数（圧縮符号長の上限長Ｎの葉数除く）を引くことで、圧縮符号長の上限長Ｎの葉数を求める。この場合は、１２１５個である。

[0049] このあと、生成装置は、補正Ｂ⁻１の場合と同様の計算処理により、補正Ｂ⁻

２での生起確率総和を求める。そして、生成装置は、補正Ｂ⁻２での生起確率総和が１以下の最大漸近値に収束したか否かを判断する。補正Ｂ⁻２での生起確率総和が１以下の最大漸近値に収束していなければ、補正Ｂ⁻の３回目（補正Ｂ⁻３）に移行する。最大漸近値に収束した場合は、補正Ｂ⁻３に移行せず、この時点での圧縮符号長ごとの葉数で確定する。補正Ｂ⁻２で更新された生起確率総和「０．９８２」は１以下であるが、最大漸近値に収束しているかが不明であるため、補正Ｂ⁻３に移行する。

[0050] 補正Ｂ⁻３では、圧縮符号長ごとの補正Ｂ⁻２での葉数を、前回の補正（この場合は補正Ｂ⁻２）の生起確率総和（０．９８２）で割ることで、葉数を更新する。なお、小数点以下は切り捨てでも、四捨五入でもよい。なお、補正Ｂ⁻２での圧縮符号長の上限長Ｎ（Ｎ＝１１ビット）については、前回の補正（この場合は補正Ｂ⁻２）の生起確率総和（０．９８２）で割ることはせず、葉の総数（１３０５個）から、補正Ｂ⁻３での圧縮符号長ごとの葉の総数（圧縮符号長の上限長Ｎの葉数除く）を引くことで、圧縮符号長の上限長Ｎの葉数を求める。この場合は、１２１５個である。

[0051] このあと、生成装置は、補正Ｂ⁻２の場合と同様の計算処理により、補正Ｂ⁻３での生起確率総和を求める。そして、生成装置は、補正Ｂ⁻３での生起確率総和が１以下の最大漸近値に収束したか否かを判断する。補正Ｂ⁻３での生起確率総和が１以下の最大漸近値に収束していなければ、補正Ｂ⁻の４回目（補正Ｂ⁻４）に移行する。最大漸近値に収束した場合は、補正Ｂ⁻４に移行せず、この時点での圧縮符号長ごとの葉数で確定する。補正Ｂ⁻３で更新された生起確率総和「０．９８２」は、補正Ｂ⁻２で更新された生起確率総和「０．９８２」と同じ値である。すなわち、補正Ｂ⁻３での各圧縮符号長の葉数と補正Ｂ⁻２での各圧縮符号長の葉数とは同じである。この場合、生成装置は、生起確率総和が最大漸近値に収束したと判断し、葉数が確定する。

[0052] このようにして、葉数が確定するまで補正Ｂ⁻を継続することとなる。図５の例では、補正Ｂ⁻３で圧縮符号長ごとの葉数が確定したこととなる。このあと、生成装置は、圧縮符号長ごとに、葉当たりの枝数を算出することとなる

。葉当たりの枝数の算出は、上述したように、圧縮符号長の上限長Ｎ（この場合はＮ＝１１ビット）から降順に、2^0、2^1、2^2、2^3、2^4、2^5、2^6として葉当たりの枝数を割り当てることとなる。なお、枝数の小計は、圧縮符号長ごとに、葉当たりの枝数に確定した葉数を乗じた乗算結果である。

[0053] 図６は、文字情報ごとの補正結果を示す説明図である。図６において、文字情報テーブルには、補正Ａ、補正Ｂ-１～補正Ｂ-２までの補正結果が追加されている。図５に示したように、補正により圧縮符号長別の葉数が更新されるため、順位項目の１位の文字情報から短い圧縮符号長が割り当てられることとなる。

[0054] たとえば、補正Ｂ-２で確定した場合、圧縮符号長６ビットでは葉数が６、圧縮符号長７ビットでは葉数が１８、…、圧縮符号長１１ビットでは葉数が１２１５となっている。したがって、順位が１位から６位までの文字情報（葉数６個分）については６ビットの圧縮符号長、順位が７位から２４位までの文字情報（葉数１８個分）については７ビットの圧縮符号長、…、順位が９１位から１３０５位までの文字情報（葉数１２１５個分）については１１ビットの圧縮符号長が割り当てられる。

[0055] そして、生成装置は、文字情報と文字情報に割り当てられた圧縮符号長と圧縮符号長ごとの葉数とに基づいて、文字情報ごとに圧縮符号を割り当て、葉の構造体を生成することとなる。たとえば、出現率１位の高位文字「０」は５ビットの圧縮符号長が割り当てられているため、圧縮符号が「０００００」となる。したがって、圧縮符号「０００００」、圧縮符号長「６」、文字情報「０」を含む葉Ｌ１の構造体が生成されることとなる。

[0056] ＜（６）葉へのポインタ生成の詳細（Ｎ＝１１）＞

図７～図１２は、葉へのポインタ生成の詳細（Ｎ＝１１）を示す説明図である。図７～図１２は、圧縮符号長の上限Ｎが１１ビットの場合の葉へのポインタを示している。図７において、圧縮符号長が６ビットの葉数は６個であるため、圧縮符号は「０００００」～「００１０１」が割り当てられる。また、圧縮符号長が６ビットの葉当たりの枝数は、３２本である。したが

って、圧縮符号長が６ビットの圧縮符号についての葉へのポインタは、３２（＝2^5）個生成される。具体的には、葉へのポインタの先頭６ビットが圧縮符号で後続５ビットが３２種のビット列となる。したがって、圧縮符号長が６ビットの圧縮符号の各々について、３２種の葉へのポインタが生成されることとなる。

[0057] 図８において、圧縮符号長が７ビットの葉数は１８個であるため、圧縮符号「０００１１００」～「００１１１１１」が割り当てられる。また、圧縮符号長が７ビットの葉当たりの枝数は、１６本である。したがって、圧縮符号長が７ビットの圧縮符号についての葉へのポインタは、１６（＝2^4）個生成される。具体的には、葉へのポインタの先頭７ビットが圧縮符号で後続４ビットが１６種のビット列となる。したがって、圧縮符号長が７ビットの圧縮符号の各々について、１６種の葉へのポインタが生成されることとなる。

[0058] 以下同様、図９において、圧縮符号長が８ビットの圧縮符号の各々について、８種の葉へのポインタが生成されることとなる。図１０において、圧縮符号長が９ビットの圧縮符号の各々について、４種の葉へのポインタが生成されることとなる。図１１において、したがって、圧縮符号長が１０ビットの圧縮符号の各々について、２種の葉へのポインタが生成されることとなる。図１２において、圧縮符号長が１１ビットの圧縮符号の各々について、１種の葉へのポインタが生成されることとなる。

[0059] ＜（７）2^N分枝無節点ハフマン木の構築の詳細（Ｎ＝１１）＞

図１３は、図２の（７）2^N分枝無節点ハフマン木の構築の詳細（Ｎ＝１１）を示す説明図である。図１３では、Ｎ＝１１とした場合の２０４８（＝2^{11}）分枝無節点ハフマン木を示している。根の構造体には、葉へのポインタが格納されている。葉へのポインタはポイント先の葉の構造体を指定することができる。

[0060] 具体的には、圧縮符号長が６ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図７に示したように３２個生成される。したがって、葉Ｌ１の構造体については、根の構造体に３２個の葉Ｌ１へのポイン

タＬ１Ｐ（１）～Ｌ１Ｐ（３２）が格納される。葉Ｌ２の構造体～葉Ｌ６の構造体についても同様である。

[0061] また、圧縮符号長が７ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図８に示したように１６個生成される。したがって、葉Ｌ７の構造体については、根の構造体に１６個の葉Ｌ７へのポインタＬ７Ｐ（１）～Ｌ７Ｐ（１６）が格納される。葉Ｌ８の構造体～葉Ｌ２４の構造体についても同様である。

[0062] また、圧縮符号長が８ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図９に示したように８個生成される。したがって、葉Ｌ２５の構造体については、根の構造体に８個の葉Ｌ２５へのポインタＬ２５Ｐ（１）～Ｌ２５Ｐ（８）が格納される。葉Ｌ２６の構造体～葉Ｌ４６の構造体についても同様である。

[0063] また、圧縮符号長が９ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図１０に示したように４個生成される。したがって、葉Ｌ４７の構造体については、根の構造体に４個の葉Ｌ４７へのポインタＬ４７Ｐ（１）～Ｌ４７Ｐ（４）が格納される。葉Ｌ４８の構造体～葉Ｌ６９の構造体についても同様である。

[0064] また、圧縮符号長が１０ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図１１に示したように２個生成される。したがって、葉Ｌ７０の構造体については、根の構造体に２個の葉Ｌ７０へのポインタＬ７０Ｐ（１）～Ｌ７０Ｐ（２）が格納される。葉Ｌ７１の構造体～葉Ｌ８９の構造体についても同様である。

[0065] また、圧縮符号長が１１ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図１２に示したように１個生成される。したがって、葉Ｌ９０の構造体については、根の構造体に１個の葉Ｌ９０へのポインタＬ９０Ｐが格納される。葉Ｌ９１の構造体～葉Ｌ１３０５の構造体についても同様である。

[0066] ＜（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１２）＞

図１４は、図２の（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１２）を示す説明図である。なお、Ｎ＝１２の場合は、図３の文字情報テーブルには、たとえば１０２４種の基礎単語のレコードも追加されることとなる。基礎単語は、１３０５種の文字コード、分割文字コード、予約語よりも、出現回数が低いこととする。したがって、１３０５種に１０２４種を追加した２３２９種の文字情報についての文字情報テーブルが生成されているものとする。また、基礎単語の追加により、図４の総回数がかわるため、２３２９種の文字情報の生起確率も図４に示した値とは異なることとなる。

[0067] そして、生成装置は、図５の場合と同様に、生起確率総和を求める。そして、生成装置は、生起確率総和がしきい値ｔ以上１以下であるか否かを判断する。しきい値ｔ未満であれば、補正Ｂに移行する。しきい値ｔ以上１以下である場合は、補正Ｂに移行せず、この時点での圧縮符号長ごとの葉数で確定する。

[0068] 図１４の例では、補正Ａでの生起確率総和「０．８２３」はしきい値ｔ未満であるとして、補正B^-をおこなうこととする。

[0069] まず、補正B^+の１回目（補正B^+1）では、圧縮符号長ごとの補正Ａの葉数を、前回の補正（この場合は補正Ａ）の生起確率総和（０．８２３）で割ることで、葉数を更新する。なお、小数点以下は切り捨てでも、四捨五入でもよい。なお、補正Ａでの圧縮符号長の上限長Ｎ（Ｎ＝１２ビット）については、前回の補正（この場合は補正Ａ）の生起確率総和（０．８２３）で割ることはせず、葉の総数（２３２９個）から、補正B^+1での圧縮符号長ごとの葉の総数（圧縮符号長の上限長Ｎの葉数除く）を引くことで、圧縮符号長の上限長Ｎの葉数を求める。この場合は、２１９２個である。

[0070] このあと、生成装置は、補正Ａの場合と同様の計算処理により、補正B^+1での生起確率総和を求める。そして、生成装置は、補正B^+1での生起確率総和が１以下の最大漸近値に収束したか否かを判断する。補正B^+1での生起確率総和が１以下の最大漸近値に収束していなければ、補正B^+の２回目（補正B^+2）に移行する。１以下の最大漸近値に収束した場合は、補正B^+2に移行

せず、この時点での圧縮符号長ごとの葉数で確定する。補正Ｂ+１で更新された生起確率総和「０．８６１」は１以下であるが、最大漸近値に収束しているかが不明であるため、補正Ｂ+２に移行する。

[0071] 補正Ｂ+２では、圧縮符号長ごとの補正Ｂ+１での葉数を、前回の補正（この場合は補正Ｂ+１）の生起確率総和（０．８６１）で割ることで、葉数を更新する。なお、小数点以下は切り捨てでも、四捨五入でもよい。なお、補正Ｂ+２での圧縮符号長の上限長Ｎ（Ｎ＝１２ビット）については、前回の補正（この場合は補正Ｂ+１）の生起確率総和（０．８６１）で割ることはせず、葉の総数（２３２９個）から、補正Ｂ+２での圧縮符号長ごとの葉の総数（圧縮符号長の上限長Ｎの葉数除く）を引くことで、圧縮符号長の上限長Ｎの葉数を求める。この場合は、２１７３個である。

[0072] このあと、生成装置は、補正Ｂ+１の場合と同様の計算処理により、補正Ｂ+２での生起確率総和を求める。そして、生成装置は、補正Ｂ+２での生起確率総和が１以下の最大漸近値に収束したか否かを判断する。補正Ｂ+２での生起確率総和が１以下の最大漸近値に収束していなければ、補正Ｂ+の３回目（補正Ｂ+３）に移行する。１以下の最大漸近値に収束した場合は、補正Ｂ+３に移行せず、この時点での圧縮符号長ごとの葉数で確定する。補正Ｂ+２で更新された生起確率総和「０．８９７」は１以下であるが、補正Ｂ+１で更新された生起確率総和「０．８６１」よりも大きい。すなわち、補正Ｂ-１での生起確率総和は最大漸近値ではなく、補正Ｂ+２での生起確率総和は最大漸近値に収束しているかが不明であるため、補正Ｂ+３に移行する。

[0073] このようにして、葉数が確定するまで補正Ｂ+を継続することとなる。図１４の例では、補正Ｂ+１０で更新された生起確率総和「０．９８４」は、補正Ｂ+１１で更新された生起確率総和「０．９８４」と同じ値である。すなわち、補正Ｂ+１０での各圧縮符号長の葉数と補正Ｂ+１１での各圧縮符号長の葉数とは同じである。この場合、生成装置は、補正Ｂ+１０において、生起確率総和が最大漸近値に収束したと判断し、葉数が確定する。

[0074] このあと、生成装置は、圧縮符号長ごとに、葉当たりの枝数を算出するこ

ととなる。葉当たりの枝数の算出は、上述したように、圧縮符号長の上限長Ｎ（この場合はＮ＝１２ビット）から降順に、2^0、2^1、2^2、2^3、2^4、2^5、2^6として葉当たりの枝数を割り当てることとなる。なお、枝数の小計は、圧縮符号長ごとに、葉当たりの枝数に確定した葉数を乗じた乗算結果である。

[0075] ＜（６）葉へのポインタ生成の詳細（Ｎ＝１２）＞

図１５～図２１は、葉へのポインタ生成の詳細（Ｎ＝１２）を示す説明図である。図１５～図２１は、圧縮符号長の上限Ｎが１２ビットの場合の葉へのポインタを示している。図１５において、圧縮符号長が６ビットの葉数は２個であるため、圧縮符号「０００００」，「０００００１」が割り当てられる。また、圧縮符号長が６ビットの葉当たりの枝数は、６４本である。したがって、圧縮符号長が６ビットの圧縮符号についての葉へのポインタは、６４（＝2^6）個生成される。具体的には、葉へのポインタの先頭６ビットが圧縮符号で後続６ビットが６４種のビット列となる。したがって、圧縮符号長が６ビットの圧縮符号の各々について、６４種の葉へのポインタが生成されることとなる。

[0076] 図１６において、圧縮符号長が７ビットの葉数は１２個であるため、圧縮符号「００００１００」～「０００１１１１」が割り当てられる。また、圧縮符号長が７ビットの葉当たりの枝数は、３２本である。したがって、圧縮符号長が７ビットの圧縮符号についての葉へのポインタは、３２（＝2^5）個生成される。具体的には、葉へのポインタの先頭７ビットが圧縮符号で後続５ビットが３２種のビット列となる。したがって、圧縮符号長が６ビットの圧縮符号の各々について、３２種の葉へのポインタが生成されることとなる。

[0077] 以下同様、図１７において、圧縮符号長が８ビットの圧縮符号の各々について、１６種の葉へのポインタが生成されることとなる。図１８において、圧縮符号長が９ビットの圧縮符号の各々について、８種の葉へのポインタが生成されることとなる。図１９において、圧縮符号長が１０ビットの圧縮符号の各々について、４種の葉へのポインタが生成されることとなる。図２０

において、圧縮符号長が１１ビットの圧縮符号の各々について、２種の葉へのポインタが生成されることとなる。図２１において、圧縮符号長が１２ビットの圧縮符号の各々について、１種の葉へのポインタが生成されることとなる。

[0078] ＜（７）2^N分枝無節点ハフマン木の構築の詳細（Ｎ＝１２）＞

図２２は、（７）2^N分枝無節点ハフマン木の構築の詳細（Ｎ＝１２）を示す説明図である。図２２では、Ｎ＝１２とした場合の４０９６（＝2^{12}）分枝無節点ハフマン木を示している。根の構造体には、葉へのポインタが格納されている。葉へのポインタはポイント先の葉の構造体を指定することができる。

[0079] 具体的には、圧縮符号長が６ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図１５に示したように６４個生成される。したがって、葉Ｌ１の構造体については、根の構造体に６４個の葉Ｌ１へのポインタＬ１Ｐ（１）～Ｌ１Ｐ（６４）が格納される。葉Ｌ２の構造体についても同様である。

[0080] また、圧縮符号長が７ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図１６に示したように３２個生成される。したがって、葉Ｌ３の構造体については、根の構造体に３２個の葉Ｌ３へのポインタＬ３Ｐ（１）～Ｌ３Ｐ（３２）が格納される。葉Ｌ４の構造体～葉Ｌ１４の構造体についても同様である。

[0081] また、圧縮符号長が８ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図１７に示したように１６個生成される。したがって、葉Ｌ１５の構造体については、根の構造体に１６個の葉Ｌ１５へのポインタＬ１５Ｐ（１）～Ｌ１５Ｐ（１６）が格納される。葉Ｌ１６の構造体～葉Ｌ５５の構造体についても同様である。

[0082] また、圧縮符号長が９ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図１８に示したように８個生成される。したがって、葉Ｌ５６の構造体については、根の構造体に８個の葉Ｌ５６へのポインタＬ

５６Ｐ（１）～Ｌ５６Ｐ（８）が格納される。葉Ｌ５７の構造体～葉Ｌ１０７の構造体についても同様である。

[0083] また、圧縮符号長が１０ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図１９に示したように４個生成される。したがって、葉Ｌ１０８の構造体については、根の構造体に４個の葉Ｌ１０８へのポインタＬ１０８Ｐ（１）～Ｌ１０８Ｐ（４）が格納される。葉Ｌ１０９の構造体～葉Ｌ１６５の構造体についても同様である。

[0084] また、圧縮符号長が１１ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図２０に示したように２個生成される。したがって、葉Ｌ１６６の構造体については、根の構造体に２個の葉Ｌ１６６へのポインタＬ１６６Ｐ（１）～Ｌ１６６Ｐ（２）が格納される。葉Ｌ１６７の構造体～葉Ｌ２１３の構造体についても同様である。

[0085] また、圧縮符号長が１２ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図２１に示したように１個生成される。したがって、葉Ｌ２１４の構造体については、根の構造体に１個の葉Ｌ２１４へのポインタＬ２１４Ｐが格納される。葉Ｌ２１５の構造体～葉Ｌ２３２９の構造体についても同様である。

[0086] ＜（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１３）＞

図２３は、図１の（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１３）を示す説明図である。なお、Ｎ＝１３の場合は、図３の文字情報テーブルには、たとえば４０９６種の基礎単語のレコードも追加されることとなる。基礎単語は、１３０５種の文字コード、分割文字コード、予約語よりも、出現回数が低いこととする。したがって、１３０５種に４０９６種を追加した５４０１種の文字情報についての文字情報テーブルが生成されているものとする。また、基礎単語の追加により、図４の総回数がかわるため、５４０１種の文字情報の生起確率も図４に示した値とは異なることとなる。図２３に示した例では、補正Ａでの生起確率総和が「０．８７７」であるため、補正Ｂ$^{+}$が行われることとなる。

[0087] ＜（６）葉へのポインタ生成の詳細（Ｎ＝１３）＞

図２４～図３１は、葉へのポインタ生成の詳細（Ｎ＝１３）を示す説明図である。図２４～図３１は、圧縮符号長の上限ＮがＮ＝１３ビットの場合の葉へのポインタを示している。図２４において、圧縮符号長が６ビットの葉数は２個であるため、圧縮符号「００００００」，「０００００１」が割り当てられる。また、圧縮符号長が６ビットの葉当たりの枝数は、１２８本である。したがって、圧縮符号長が６ビットの圧縮符号についての葉へのポインタは、１２８（＝2^7）個生成される。具体的には、葉へのポインタの先頭６ビットが圧縮符号で後続７ビットが１２８種のビット列となる。したがって、圧縮符号長が６ビットの圧縮符号の各々について、１２８種の葉へのポインタが生成されることとなる。

[0088] 図２５において、圧縮符号長が７ビットの葉数は１１個であるため、圧縮符号「００００１００」～「０００１１１０」が割り当てられる。また、圧縮符号長が７ビットの葉当たりの枝数は、６４本である。したがって、圧縮符号長が７ビットの圧縮符号についての葉へのポインタは、６４（＝2^6）個生成される。具体的には、葉へのポインタの先頭７ビットが圧縮符号で後続６ビットが６４種のビット列となる。したがって、圧縮符号長が７ビットの圧縮符号の各々について、６４種の葉へのポインタが生成されることとなる。

[0089] 以下同様、図２６において、圧縮符号長が８ビットの圧縮符号の各々について、３２種の葉へのポインタが生成されることとなる。図２７において、圧縮符号長が９ビットの圧縮符号の各々について、１６種の葉へのポインタが生成されることとなる。図２８において、圧縮符号長が１０ビットの圧縮符号の各々について、８種の葉へのポインタが生成されることとなる。図２９において、圧縮符号長が１１ビットの圧縮符号の各々について、４種の葉へのポインタが生成されることとなる。図３０において、圧縮符号長が１２ビットの圧縮符号の各々について、２種の葉へのポインタが生成されることとなる。図３１において、圧縮符号長が１３ビットの圧縮符号の各々につい

て、１種の葉へのポインタが生成されることとなる。

[0090] ＜（７）２Ｎ分枝無節点ハフマン木の構築の詳細（Ｎ＝１３）＞

図３２は、図２の（７）２Ｎ分枝無節点ハフマン木の構築の詳細（Ｎ＝１３）を示す説明図である。図３２では、Ｎ＝１３とした場合の８１９２（＝2^{13}）分枝無節点ハフマン木を示している。根の構造体には、葉へのポインタが格納されている。葉へのポインタはポイント先の葉の構造体を指定することができる。

[0091] 具体的には、圧縮符号長が６ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図２４に示したように１２８個生成される。したがって、葉Ｌ１の構造体については、根の構造体に１２８個の葉Ｌ１へのポインタＬ１Ｐ（１）～Ｌ１Ｐ（１２８）が格納される。葉Ｌ２の構造体についても同様である。

[0092] また、圧縮符号長が７ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図２５に示したように６４個生成される。したがって、葉Ｌ３の構造体については、根の構造体に６４個の葉Ｌ３へのポインタＬ３Ｐ（１）～Ｌ３Ｐ（６４）が格納される。葉Ｌ４の構造体～葉Ｌ１３の構造体についても同様である。

[0093] また、圧縮符号長が８ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図２６に示したように３２個生成される。したがって、葉Ｌ１４の構造体については、根の構造体に３２個の葉Ｌ１４へのポインタＬ１４Ｐ（１）～Ｌ１４Ｐ（３２）が格納される。葉Ｌ１５の構造体～葉Ｌ４０の構造体についても同様である。

[0094] また、圧縮符号長が９ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図２７に示したように１６個生成される。したがって、葉Ｌ４１の構造体については、根の構造体に１６個の葉Ｌ４１へのポインタＬ４１Ｐ（１）～Ｌ４１Ｐ（１６）が格納される。葉Ｌ４２の構造体～葉Ｌ７５の構造体についても同様である。

[0095] また、圧縮符号長が１０ビットの圧縮符号が格納される葉の構造体につい

ての葉へのポインタは、図２８に示したように８個生成される。したがって、葉Ｌ７６の構造体については、根の構造体に８個の葉Ｌ７６へのポインタＬ７６Ｐ（１）～Ｌ７６Ｐ（８）が格納される。葉Ｌ７７の構造体～葉Ｌ１１１の構造体についても同様である。

[0096] また、圧縮符号長が１１ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図２９に示したように４個生成される。したがって、葉Ｌ１１２の構造体については、根の構造体に４個の葉Ｌ１１２へのポインタＬ１１２Ｐ（１）～Ｌ１１２Ｐ（４）が格納される。葉Ｌ１１３の構造体～葉Ｌ１４２の構造体についても同様である。

[0097] また、圧縮符号長が１２ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図３０に示したように２個生成される。したがって、葉Ｌ１４３の構造体については、根の構造体に２個の葉Ｌ１４３へのポインタＬ１４３Ｐ（１）～Ｌ１４３Ｐ（２）が格納される。葉Ｌ１４４の構造体～葉Ｌ２４８の構造体についても同様である。

[0098] また、圧縮符号長が１３ビットの圧縮符号が格納される葉の構造体についての葉へのポインタは、図３１に示したように１個生成される。したがって、葉Ｌ２４９の構造体については、根の構造体に１個の葉Ｌ２４９へのポインタＬ２４９Ｐが格納される。葉Ｌ２５０の構造体～葉Ｌ５４０１の構造体についても同様である。

[0099] ＜葉の構造体＞

図３３は、葉の構造体を示す説明図である。葉の構造体は、第１領域～第４領域を有するデータ構造体である。葉の構造体は、第１領域には、圧縮符号およびその圧縮符号長が格納される。第２領域には、葉の標識と伸長種別（図３参照）が格納される。第３領域には、伸長種別に応じて高位１６ビット文字コード、低位の分割８ビット文字コード、または特殊単語へのポインタが格納される。

[0100] 第４領域には、符号種別と符号区分が格納される。符号種別とは、文字コードが数字、英字、特殊記号、カタカナ、ひらがな、漢字のいずれに該当す

るか、または特殊単語へのポインタであるかを識別する情報である。符号区分とは、文字コードが１６ビットであるか８ビットであるかを識別する情報である。１６ビットの文字コードである場合または予約語である場合、符号区分として“１”を割り当て、８ビットの分割文字コードの場合、符号区分として“０”を割り当てる。

[0101] ＜文字コードの構造体＞

図３４は、高位文字コードの構造体を示す説明図である。高位文字コードの構造体３４００は、高位文字コードｅ＃とその葉Ｌ＃へのポインタを格納するデータ構造体である。具体的には、たとえば、生成装置は、葉の構造体から伸長種別や符号区分を参照して文字コードｅ＃を抽出し、抽出した文字コードｅ＃をポイントする葉Ｌ＃のポインタを根の構造体から抽出する。葉Ｌ＃へのポインタが複数ある場合は、いずれのポインタでもよい。これにより、高位文字コードの構造体３４００が生成される。

[0102] 図３５は、分割文字コードの構造体を示す説明図である。分割文字コードの構造体３５００は、分割文字コードとその葉Ｌ＃へのポインタを格納する。具体的には、たとえば、生成装置は、葉の構造体から伸長種別や符号区分を参照して分割文字コードを抽出し、抽出した分割文字コードをポイントする葉Ｌ＃のポインタを根の構造体から抽出する。葉Ｌ＃へのポインタが複数ある場合は、いずれのポインタでもよい。これにより、分割文字コードの構造体３５００が生成される。

[0103] 図３６および図３７は、特殊単語の構造体を示す説明図である。図３６では、特殊単語が予約語のみであり、図３７では、特殊単語は予約語および基礎単語である。特殊単語の構造体３６００は、特殊単語とその葉Ｌ＃へのポインタを格納するデータ構造体である。具体的には、たとえば、生成装置は、葉の構造体から伸長種別や符号区分を参照して特殊単語を抽出し、抽出した特殊単語をポイントする葉Ｌ＃のポインタを根の構造体から抽出する。葉Ｌ＃へのポインタが複数ある場合は、いずれのポインタでもよい。これにより、特殊単語の構造体３６００が生成される。

[0104] ＜生成装置のハードウェア構成＞

図３８は、実施の形態にかかる生成装置のハードウェア構成例を示すブロック図である。図３８において、生成装置は、ＣＰＵ（Ｃｅｎｔｒａｌ Ｐｒｏｃｅｓｓｉｎｇ Ｕｎｉｔ）３８０１と、ＲＯＭ（Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）３８０２と、ＲＡＭ（Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）３８０３と、磁気ディスクドライブ３８０４と、磁気ディスク３８０５と、光ディスクドライブ３８０６と、光ディスク３８０７と、ディスプレイ３８０８と、Ｉ／Ｆ（Ｉｎｔｅｒｆａｃｅ）３８０９と、キーボード３８１０と、マウス３８１１と、スキャナ３８１２と、プリンタ３８１３と、を備えている。また、各構成部はバス３８００によってそれぞれ接続されている。

[0105] ここで、ＣＰＵ３８０１は、生成装置の全体の制御を司る。ＲＯＭ３８０２は、ブートプログラムなどのプログラムを記憶している。ＲＡＭ３８０３は、ＣＰＵ３８０１のワークエリアとして使用される。磁気ディスクドライブ３８０４は、ＣＰＵ３８０１の制御にしたがって磁気ディスク３８０５に対するデータのリード／ライトを制御する。磁気ディスク３８０５は、磁気ディスクドライブ３８０４の制御で書き込まれたデータを記憶する。

[0106] 光ディスクドライブ３８０６は、ＣＰＵ３８０１の制御にしたがって光ディスク３８０７に対するデータのリード／ライトを制御する。光ディスク３８０７は、光ディスクドライブ３８０６の制御で書き込まれたデータを記憶したり、光ディスク３８０７に記憶されたデータをコンピュータに読み取らせたりする。

[0107] ディスプレイ３８０８は、カーソル、アイコンあるいはツールボックスをはじめ、文書、画像、機能情報などのデータを表示する。このディスプレイ３８０８は、たとえば、ＣＲＴ、ＴＦＴ液晶ディスプレイ、プラズマディスプレイなどを採用することができる。

[0108] インターフェース（以下、「Ｉ／Ｆ」と略する。）３８０９は、通信回線を通じてＬＡＮ（Ｌｏｃａｌ Ａｒｅａ Ｎｅｔｗｏｒｋ）、ＷＡＮ（Ｗｉ

ｄｅ　Ａｒｅａ　Ｎｅｔｗｏｒｋ）、インターネットなどのネットワーク３８１４に接続され、このネットワーク３８１４を介して他の装置に接続される。そして、Ｉ／Ｆ３８０９は、ネットワーク３８１４と内部のインターフェースを司り、外部装置からのデータの入出力を制御する。Ｉ／Ｆ３８０９には、たとえばモデムやＬＡＮアダプタなどを採用することができる。

[0109] キーボード３８１０は、文字、数字、各種指示などの入力のためのキーを備え、データの入力をおこなう。また、タッチパネル式の入力パッドやテンキーなどであってもよい。マウス３８１１は、カーソルの移動や範囲選択、あるいはウィンドウの移動やサイズの変更などをおこなう。ポインティングデバイスとして同様に機能を備えるものであれば、トラックボールやジョイスティックなどであってもよい。

[0110] スキャナ３８１２は、画像を光学的に読み取り、生成装置内に画像データを取り込む。なお、スキャナ３８１２は、ＯＣＲ（Ｏｐｔｉｃａｌ　Ｃｈａｒａｃｔｅｒ　Ｒｅａｄｅｒ）機能を持たせてもよい。また、プリンタ３８１３は、画像データや文書データを印刷する。プリンタ３８１３には、たとえば、レーザプリンタやインクジェットプリンタを採用することができる。

[0111] なお、本実施の形態にかかる生成プログラムや、上述した2^N分枝無節点ハフマン木、高位文字コードの構造体３４００、分割文字コードの構造体３５００、特殊単語の構造体３６００は、上述したＲＡＭ３８０３や磁気ディスク３８０５などの記憶装置に記憶される。

[0112] ＜生成装置の機能的構成例＞

図３９は、生成装置の機能的構成例を示すブロック図である。生成装置３９００は、集計部３９０１と、決定部３９０２と、補正部３９０３と、算出部３９０４と、判断部３９０５と、更新部３９０６と、生成部３９０７と、特定部３９０８と、構築部３９０９と、を備える。集計部３９０１～構築部３９０９は、具体的には、たとえば、図３８に示したＲＯＭ３８０２、ＲＡＭ３８０３、磁気ディスク３８０５、光ディスク３８０７などの記憶装置に記憶されたプログラムをＣＰＵ３８０１に実行させることにより、その機能

を実現する。

[0113] 集計部３９０１は、データを集計する機能を有する。具体的には、対象ファイル群を読み込んで、単一文字の出現回数を集計する。たとえば、図２に示したように、集計部３９０１は、出現回数を降順にソートして、所定順位（たとえば、１０２４位）までを高位文字コードとする。所定順位よりも下位の単一文字は、上位８ビットと下位８ビットに分割して、それぞれ８ビットの分割文字コードとして再集計される。

[0114] また、集計部３９０１は、最長一致検索処理をおこなうことにより、特殊単語を特定し、特殊単語についても出現回数を集計する。そして、集計部３９０１は、高位文字コード、分割文字コード、特殊単語を混在させて出現回数を再ソートする。これにより、図４に示した順位項目、伸長種別項目、コード項目、文字項目、出現回数項目が得られる。

[0115] また、集計部３９０１は、文字情報ごとの出現回数が得られた場合、全文字情報の出現回数の総和を総回数として算出する。そして、集計部３９０１は、文字情報ごとに、出現回数を総回数で割ることで、出現率を算出する。集計部３９０１は、算出された出現率に応じた生起確率を、図２の「（２）圧縮符号長算出」の項で説明したように特定し、そのべき数を圧縮符号長とする。これにより、文字情報ごとに圧縮符号長が特定されたこととなる（図３を参照。）。そして、集計部３９０１は、図３の文字情報テーブルを参照して、圧縮符号長ごとに文字情報の種類数（図４に示した補正前の葉数）を集計する。

[0116] 決定部３９０２は、対象ファイルに出現する文字情報の総種類数に基づいて、最小圧縮符号長から最大圧縮符号長までの圧縮符号長群の中から、文字情報に割り当てられる圧縮符号長の上限長Ｎを決定する機能を有する。具体的には、たとえば、決定部３９０２は、文字情報の総種類数が2^{K-1}個より大きく2^{K}個以下である場合は、上限長ＮをＫビットに決定する。たとえば、文字情報の総種類数が2^{10}個より大きく2^{11}個以下である場合は、決定部３９０２は、上限長Ｎを１１ビットに決定することとなる。

[0117] また、文字情報の総種類数が2^{11}個より大きく2^{12}個以下である場合は、決定部３９０２は、上限長Ｎを１２ビットに決定することとなる。また、文字情報の総種類数が2^{12}個より大きく2^{13}個以下である場合は、決定部３９０２は、上限長Ｎを１３ビットに決定することとなる。ここで、文字コードの種別ごとの文字情報の分類について説明する。

[0118] 図４０は、ＵＴＦ１６での文字情報の分類例を示す説明図である。特殊単語には、２５種の予約語と最大４０９６種の基礎単語が含まれている。基礎単語を入れない場合は、予約語のみの２５種となる。ＵＴＦ１６において、基礎単語を含めない場合の文字情報の最大総数は、１３０５個である。したがって、2^{10}＜１３０５≦2^{11}となるため、上限長ＮはＮ＝１１ビットに決定される。また、基礎単語を２０４８個含める場合の文字情報の最大総数は、３３５３個である。したがって、2^{11}＜３３５３≦2^{12}となるため、上限長ＮはＮ＝１２ビットに決定される。また、基礎単語を４０９６個含める場合の文字情報の最大総数は、５４０１個である。したがって、2^{12}＜５４０１≦2^{13}となるため、上限長ＮはＮ＝１３ビットに決定される。

[0119] 図４１は、ＡＳＣＩＩコードでの文字情報の分類例を示す説明図である。特殊単語には、２５種の予約語と最大２０４８種の基礎単語が含まれている。基礎単語を１０２４個含める場合の文字情報の最大総数は、１３０５個である。したがって、2^{10}＜１３０５≦2^{11}となるため、上限長ＮはＮ＝１１ビットに決定される。基礎単語を２０４８個含める場合の文字情報の最大総数は、３３５３個である。したがって、2^{11}＜３３５３≦2^{12}となるため、上限長ＮはＮ＝１２ビットに決定される。

[0120] 図４２は、シフトＪＩＳコードでの文字情報の分類例を示す説明図である。特殊単語には、２５種の予約語と最大４０９６種の基礎単語が含まれている。また、シフトＪＩＳコードの場合、日本語の文字は２つの８ビットコードの組み合わせで表現されるため、偶数個の単語（文字列）とみなされる。基礎単語を２０４８個含める場合の文字情報の最大総数は、３３５３個である。したがって、2^{11}＜３３５３≦2^{12}となるため、上限長ＮはＮ＝１２ビッ

トに決定される。また、基礎単語を４０９６個含める場合の文字情報の最大総数は、５４０１個である。したがって、2^{12}＜５４０１≦2^{13}となるため、上限長ＮはＮ＝１３ビットに決定される。

[0121] また、図３９において、補正部３９０３は、集計部３９０１によって集計された圧縮符号長ごとの文字情報の種類数のうち、決定部３９０２によって決定された上限長Ｎを圧縮符号長とする文字情報の種類数を、上限長Ｎ以上の圧縮符号長での文字情報の種類数の総和に補正する機能を有する。具体的には、たとえば、補正部３９０３は、上述した補正Ａによる補正処理を実行する。

[0122] 算出部３９０４は、補正部３９０３による補正後の上限長Ｎまでの各圧縮符号長を規定する各生起確率の総和を算出する機能を有する。具体的には、たとえば、図５、図１４、図２３に示した生起確率総和を算出する。

[0123] 判断部３９０５は、算出部３９０４によって算出された生起確率総和がしきい値ｔ以上１以下であるか否かを判断する機能を有する。しきい値ｔは、０＜ｔ≦１の値であり、あらかじめ設定される。しきい値ｔを１としてもよい。しきい値ｔ以上１以下である場合は、補正部３９０３による補正後の圧縮符号長ごとの文字情報の種類数が、圧縮符号長ごとの葉数として確定させてもよい。

[0124] 一方、しきい値ｔ未満である場合、まだ、生起確率総和を増加させることが可能であるため、上述した補正B^{+}を実行することが可能となる。また、生起確率総和が１よりも大きくなった場合は、上述した補正B^{-}を実行することが可能となる。

[0125] 更新部３９０６は、判断部３９０５によってしきい値ｔ以上１以下でないと判断された場合、上限長Ｎまでの各圧縮符号長の文字情報の種類数を、生起確率総和で割ることにより、上限長Ｎまでの各圧縮符号長の文字情報の種類数を更新する機能を有する。具体的には、たとえば、更新部３９０６は、しきい値ｔ未満と判断された場合は補正B^{+}を開始し、１より大きいと判断された場合は補正B^{-}を開始する。

[0126] この場合、算出部３９０４は、更新後における上限長Ｎまでの各圧縮符号長の文字情報の種類数に基づいて、生起確率総和を再算出する。そして、判断部３９０５は、算出部３９０４によって再算出された生起確率総和については、１以下の最大値が得られたか、具体的には、たとえば、１以下の最大漸近値に収束したか否かを判断することとなる。

[0127] 生成部３９０７は、文字情報に圧縮符号を割り当てた葉の構造体を生成する機能を有する。具体的には、たとえば、生成部３９０７による生成の条件は、判断部３９０５によって補正Ａによりしきい値ｔ以上１以下であると判断された場合、または、補正Ｂにより１以下の最大値が得られたと判断された場合である。そして、当該条件を満たした場合、生成部３９０７は、具体的には、たとえば、上限長Ｎまでの圧縮符号長と、上限長Ｎまでの圧縮符号長ごとの文字情報の種類数と、文字情報の出現率と、に基づいて、文字情報ごとに圧縮符号を割り付ける。そして、生成部３９０７は、割り付けた圧縮符号、その圧縮符号長、文字情報を含む葉の構造体を生成する。

[0128] より具体的には、まず、出現率により順位付けされた文字情報群のうち第１位から昇順に文字情報を選択する。そして、生成部３９０７は、その選択文字情報の圧縮符号長を特定して、その圧縮符号長となる固有の圧縮符号を割り当てる。

[0129] たとえば、出現回数第１位の文字情報は、高位文字の「０」である。Ｎ＝１２の場合、圧縮符号長は６ビットであるため、固有の圧縮符号「０００００」が割り当てられることとなる。次に、出現回数第２位の文字情報は、高位文字の「＞」である。Ｎ＝１２の場合、圧縮符号長は６ビットであるため、固有の圧縮符号「０００００１」が割り当てられることとなる。

[0130] このようにして、出現回数の順位の昇順で圧縮符号を割り当てる。圧縮符号が決まると、生成部３９０７は、文字情報ごとに、その文字情報や圧縮符号長、伸長種別などをまとめて葉の構造体を生成することとなる。

[0131] 特定部３９０８は、上限長Ｎまでの圧縮符号長に基づいて、生成部３９０７によって生成された葉の構造体１つ当たりの枝数を、上限長Ｎまでの圧縮

符号長ごとに特定する機能を有する。具体的には、たとえば、特定部３９０８は、図５、図１４、図２３に示したように、葉当たりの枝数を、圧縮符号長ごとに特定する。

[0132] 構築部３９０９は、2^N分枝無節点ハフマン木を構築する機能を有する。具体的には、たとえば、構築部３９０９は、葉の構造体内の圧縮符号に特定部３９０８によって特定された枝数分の枝番号をあらわす各ビット列を連結した葉へのポインタ群を、葉の構造体ごとに生成する。たとえば、Ｎ＝１１の場合は、図７～図１２に示したように、葉へのポインタ群を葉の構造体ごとに生成する。また、Ｎ＝１２の場合は、図１５～図２１に示したように、葉へのポインタ群を葉の構造体ごとに生成する。また、Ｎ＝１３の場合は、図２４～図３１に示したように、葉へのポインタ群を葉の構造体ごとに生成する。

[0133] また、構築部３９０９は、各葉の構造体についての葉へのポインタ群を根とする2^N分枝無節点ハフマン木を構築する。具体的には、たとえば、Ｎ＝１１の場合は、図１３に示したように、2^{12}分枝無節点ハフマン木を構築する。Ｎ＝１２の場合は、図２２に示したように、2^{12}分枝無節点ハフマン木を構築する。Ｎ＝１３の場合は、図３２に示したように、2^{13}分枝無節点ハフマン木を構築する。

[0134] ＜2^N分枝無節点ハフマン木の生成処理手順＞

図４３は、2^N分枝無節点ハフマン木の生成処理手順（前半）を示すフローチャートである。まず、図４３において、生成装置３９００は、集計部３９０１により、第１集計処理（ステップＳ４３０１）と第２集計処理（ステップＳ４３０２）を実行する。第１集計処理（ステップＳ４３０１）では、各文字情報の出現回数を集計する。第２集計処理（ステップＳ４３０２）では、圧縮符号長ごとの文字情報の種類数を集計する。第１集計処理（ステップＳ４３０１）と第２集計処理（ステップＳ４３０２）の詳細については後述する。

[0135] つぎに、生成装置３９００は、決定部３９０２により、圧縮符号長の上限

長Ｎを決定する（ステップＳ４３０３）。このＮが後述する補正Ａでの最大圧縮符号長となる。そして、生成装置３９００は、全文字情報の圧縮符号長がＮビット以内であるか否かを判断する（ステップＳ４３０４）。

[0136] Ｎビット以内でない場合（ステップＳ４３０４：Ｎｏ）、生成装置３９００は、補正部３９０３により、補正Ａ処理を実行する（ステップＳ４３０５）。具体的には、生成装置３９００は、上限長Ｎが最大圧縮符号長となるように、Ｎビット以上の各圧縮符号長での文字情報の種類数（葉数）の総和を求め、この総和を、圧縮符号長Ｎビットでの文字情報の種類数に補正する。したがって、（Ｎ＋１）ビット以上の圧縮符号長での文字情報の種類数は０となり、最大圧縮符号長がＮビットとなる。このあと、ステップＳ４３０６に移行する。

[0137] 一方、Ｎビット以内である場合（ステップＳ４３０４：Ｙｅｓ）、生成装置３９００は、補正Ａ処理（ステップＳ４３０５）を実行せずに、ステップＳ４３０６に移行する。ステップＳ４３０６では、生成装置３９００は、算出部３９０４により、圧縮符号長ごとの生起確率を算出し（ステップＳ４３０６）、算出された圧縮符号長ごとの生起確率の総和（生起確率総和ＴＯＰ）を算出する（ステップＳ４３０７）。そして、図４４のステップＳ４４０１に移行する。

[0138] 図４４は、2^N分枝無節点ハフマン木の生成処理手順（後半）を示すフローチャートである。生成装置３９００は、判断部３９０５により、算出された生起確率総和ＴＯＰが、ｔ≦ＴＯＰ≦１であるか否かを判断する（ステップＳ４４０１）。ｔは上述したしきい値である。ｔ≦ＴＯＰ≦１でない場合（ステップＳ４４０１：Ｎｏ）、生成装置３９００は、判断部３９０５により、ｔ＞ＴＯＰであるか否かを判断する（ステップＳ４４０２）。ｔ＞ＴＯＰである場合（ステップＳ４４０２：Ｙｅｓ）、生成装置３９００は、更新部３９０６により、補正B^+処理を実行する（ステップＳ４４０３）。補正B^+処理（ステップＳ４４０３）の詳細については後述する。補正B^+処理のあとは、ステップＳ４４０５に移行する。

[0139] 一方、ステップＳ４４０２において、ｔ＞ＴＯＰでない場合（ステップＳ４４０２：Ｎｏ）、生成装置３９００は、更新部３９０６により、補正Ｂ-処理を実行する（ステップＳ４４０４）。補正Ｂ-処理（ステップＳ４４０４）の詳細については後述する。補正Ｂ-処理のあとは、ステップＳ４４０５に移行する。また、ステップＳ４４０１において、ｔ≦ＴＯＰ≦１である場合（ステップＳ４４０１：Ｙｅｓ）、ステップＳ４４０５に移行する。

[0140] また、ステップＳ４４０５において、生成装置３９００は、生成部３９０７により、文字情報ごとに葉の構造体を生成する（ステップＳ４４０５）。そして、生成装置３９００は、特定部３９０８により、枝数特定処理を実行する（ステップＳ４４０６）。枝数特定処理（ステップＳ４４０６）では、圧縮符号長ごとの葉当たりの枝数を特定する。枝数特定処理（ステップＳ４４０６）の詳細については後述する。

[0141] そして、生成装置３９００は、構築部３９０９により、構築処理を実行する（ステップＳ４４０７）。枝数特定処理（ステップＳ４４０６）により葉の構造体ごとの枝数が特定されるため、まず、構築部３９０９は、葉の構造体ごとに、枝数分の葉へのポインタ群を生成する。そして、生成された各葉の構造体についての葉へのポインタ群を集約して根の構造体とする。これにより、2^N分枝無節点ハフマン木が生成されることとなる。なお、生成された2^N分枝無節点ハフマン木は、生成装置３９００内の記憶装置（ＲＡＭ３８０３や磁気ディスク３８０５など）に格納される。

[0142] ＜第１集計処理（ステップＳ４３０１）＞

図４５は、図４３に示した第１集計処理（ステップＳ４３０１）の詳細な処理手順を示すフローチャートである。まず、生成装置３９００は、ファイル番号ｉをｉ＝１に設定し（ステップＳ４５０１）、対象ファイルＦｉを読み込む（ステップＳ４５０２）。そして、生成装置３９００は、対象ファイルＦｉの集計処理を実行する（ステップＳ４５０３）。このあと、生成装置３９００は、ファイル番号ｉがｉ＞ｎ（ｎは対象ファイルＦ１～Ｆｎの総数）であるか否かを判断する（ステップＳ４５０４）。

[0143] ｉ＞ｎでない場合（ステップＳ４５０４：Ｎｏ）、生成装置３９００は、ｉをインクリメントし（ステップＳ４５０５）、ステップＳ４５０２に戻る。一方、ｉ＞ｎである場合（ステップＳ４５０４：Ｙｅｓ）、生成装置３９００は、第２集計処理（ステップＳ４３０２）に移行して、第１集計処理（ステップＳ４３０１）を終了する。この第１集計処理（ステップＳ４３０１）によれば、対象ファイルＦｉごとに対象ファイルＦｉの集計処理（ステップＳ４５０３）を実行することができる。

[0144] ＜対象ファイルＦｉの集計処理（ステップＳ４５０３）＞

図４６は、図４５に示した対象ファイルＦｉの集計処理（ステップＳ４５０３）の詳細な処理手順を示すフローチャートである。まず、生成装置３９００は、対象文字を対象ファイルＦｉの先頭文字とし（ステップＳ４６０１）、特殊単語集計処理を実行する（ステップＳ４６０２）。このあと、生成装置３９００は、対象文字の出現回数を１増加する（ステップＳ４６０３）。そして、生成装置３９００は、対象文字が対象ファイルＦｉの末尾文字であるか否かを判断する（ステップＳ４６０４）。

[0145] 対象文字が対象ファイルＦｉの末尾文字でない場合（ステップＳ４６０４：Ｎｏ）、生成装置３９００は、対象文字を末尾方向へ１文字シフトし（ステップＳ４６０５）、ステップＳ４６０２に戻る。一方、対象文字が対象ファイルＦｉの末尾文字である場合（ステップＳ４６０４：Ｙｅｓ）、生成装置３９００は、ステップＳ４５０４に移行して、対象ファイルＦｉの集計処理（ステップＳ４５０３）を終了する。この対象ファイルＦｉの集計処理（ステップＳ４５０３）によれば、対象ファイル群Ｆに存在する特殊単語および単一文字の出現頻度を集計することができる。

[0146] ＜特殊単語集計処理（ステップＳ４６０２）＞

図４７は、図４６に示した特殊単語集計処理（ステップＳ４６０２）の詳細な処理手順を示すフローチャートである。まず、生成装置３９００は、最長一致検索処理を実行し（ステップＳ４７０１）、最長一致した特殊単語があったか否かを判断する（ステップＳ４７０２）。最長一致した特殊単語が

あった場合（ステップＳ４７０２：Ｙｅｓ）、生成装置３９００は、特殊単語出現頻度テーブルにおいて最長一致した特殊単語の出現回数を１増加し（ステップＳ４７０３）、ステップＳ４６０３に移行する。

[0147] 一方、最長一致した特殊単語がなかった場合（ステップＳ４７０２：Ｎｏ）、ステップＳ４６０３に移行する。これにより、特殊単語集計処理（ステップＳ４６０２）を終了する。この特殊単語集計処理（ステップＳ４６０２）によれば、最長一致検索処理（ステップＳ４７０１）により特殊単語を計数することができるため、文字列が長い特殊単語を優先的に計数することができる。

[0148] ＜最長一致検索処理（ステップＳ４７０１）＞

図４８は、図４７に示した最長一致検索処理（ステップＳ４７０１）の詳細な処理手順を示すフローチャートである。まず、生成装置３９００は、ｃ＝１とする（ステップＳ４８０１）。ｃは対象文字からの文字数（対象文字含む）である。ｃ＝１の場合は、対象文字だけである。つぎに、生成装置３９００は、対象文字からｃ文字目までの対象文字列と前方一致する特殊単語を、特殊単語構造体を検索する（ステップＳ４８０２）。そして、生成装置３９００は、検索により特殊単語があるか否かを判断する（ステップＳ４８０３）。２分探索により特殊単語がヒットしなかった場合（ステップＳ４８０３：Ｎｏ）、ステップＳ４８０６に移行する。

[0149] 一方、２分探索により特殊単語がヒットした場合（ステップＳ４８０３：Ｙｅｓ）、生成装置３９００は、ヒットした特殊単語と対象文字列とが完全一致するか否かを判断する（ステップＳ４８０４）。そして、完全一致しない場合（ステップＳ４８０４：Ｎｏ）、ステップＳ４８０６に移行する。一方、完全一致する場合（ステップＳ４８０４：Ｙｅｓ）、生成装置３９００は、最長一致候補として記憶装置に保持し（ステップＳ４８０５）、ステップＳ４８０６に移行する。

[0150] ステップＳ４８０６では、生成装置３９００は、対象文字列について２分探索が終了したか否かを判断する（ステップＳ４８０６）。具体的には、生

成装置３９００は、末尾の特殊単語まで２分探索したか否かを判断する。２分探索が終了していない場合（ステップＳ４８０６：Ｎｏ）、生成装置３９００は、ステップＳ４８０２に移行して、２分探索が終了するまで継続する。

[0151] 一方、対象文字列について２分探索が終了した場合（ステップＳ４８０６：Ｙｅｓ）、生成装置３９００は、ｃ文字目の文字が対象ファイルＦｉの末尾文字であるか否かを判断する（ステップＳ４８０７）。ｃ文字目の文字が対象ファイルＦｉの末尾文字である場合（ステップＳ４８０７：Ｙｅｓ）、ステップＳ４８１０に移行する。一方、ｃ文字目の文字が対象ファイルＦｉの末尾文字でない場合（ステップＳ４８０７：Ｎｏ）、生成装置３９００は、ｃ＞ｃｍａｘであるか否かを判断する（ステップＳ４８０８）。ｃｍａｘは予め設定された値であり、これにより対象文字列の上限文字数が設定される。

[0152] ｃ＞ｃｍａｘでない場合（ステップＳ４８０８：Ｎｏ）、生成装置３９００は、ｃをインクリメントして（ステップＳ４８０９）、ステップＳ４８０２に戻る。一方、ｃ＞ｃｍａｘである場合（ステップＳ４８０８：Ｙｅｓ）、生成装置３９００は、最長一致候補があるか否かを判断する（ステップＳ４８１０）。具体的には、生成装置３９００は、ステップＳ４８０５において１つでも最長一致候補がメモリに保持されているか否かを判断する。

[0153] 最長一致候補がある場合（ステップＳ４８１０：Ｙｅｓ）、生成装置３９００は、最長一致候補のうち最長文字列を、最長一致した特殊単語に決定する（ステップＳ４８１１）。そして、ステップＳ４７０２に移行する。一方、ステップＳ４８１０において、最長一致候補が１つもない場合（ステップＳ４８１０：Ｎｏ）、ステップＳ４７０２に移行する。これにより、最長一致検索処理（ステップＳ４７０１）を終了する。この最長一致検索処理（ステップＳ４７０１）によれば、特殊単語構造体にある特殊単語の中から、完全一致した文字列の中でかつ最長の文字列を特殊単語として検索することができる。

[0154] ＜第２集計処理（ステップＳ４３０２）＞

図４９は、図４３に示した第２集計処理（ステップＳ４３０２）の詳細な処理手順を示すフローチャートである。まず、生成装置３９００は、文字情報ごとに出現率を算出する（ステップＳ４９０１）。つぎに、生成装置３９００は、未選択文字情報があるか否かを判断する（ステップＳ４９０２）。未選択文字情報がある場合（ステップＳ４９０２：Ｙｅｓ）、生成装置３９００は、出現率が最上位の未選択文字情報を選択する（ステップＳ４９０３）。そして、生成装置３９００は、選択文字情報の出現率に応じて生起確率および圧縮符号長を特定し（ステップＳ４９０４）、ステップＳ４９０２に戻る。

[0155] そして、ステップＳ４９０２において、未選択文字情報がない場合（ステップＳ４９０２：Ｎｏ）、生成装置３９００は、圧縮符号長ごとに文字情報の種類数を計数する（ステップＳ４９０５）。これにより、圧縮符号長ごとの葉数（文字情報の種類数）が特定されることとなる。

[0156] ＜補正B^+処理（ステップＳ４４０３）＞

図５０は、図４４に示した補正B^+処理（ステップＳ４４０３）の詳細な処理手順を示すフローチャートである。まず、生成装置３９００は、更新処理を実行する（ステップＳ５００１）。更新処理（ステップＳ５００１）の詳細については後述する。つぎに、生成装置３９００は、更新処理（ステップＳ５００１）後において、更新前後で葉数の変動があるか否かを判断する（ステップＳ５００２）。変動がある場合（ステップＳ５００２：Ｙｅｓ）、まだ、生起確率総和ＴＯＰが１以下の最大漸近値に収束していないため、更新処理（ステップＳ５００１）に戻る。

[0157] 一方、変動がない場合（ステップＳ５００２：Ｎｏ）、これ以上更新処理（ステップＳ５００１）をしても葉数が変動しない。すなわち、前回の補正B^+で最大漸近値に収束したこととなり、生成装置３９００は、今回の更新処理（ステップＳ５００１）での圧縮符号長ごとの葉数で確定させ（ステップＳ５００３）、ステップＳ４４０５に移行する。これにより、生起確率総和

ＴＯＰを増加させて、１に漸近させることができ、圧縮効率の向上を図ることができる。

[0158] ＜補正Ｂ⁻処理（ステップＳ４４０４）＞

図５１は、図４４に示した補正Ｂ⁻処理（ステップＳ４４０４）の詳細な処理手順を示すフローチャートである。補正Ｂ⁻処理（ステップＳ４４０４）は、図５０に示した補正Ｂ⁺処理（ステップＳ４４０３）と同一内容であるため、同一処理には同一ステップ番号を付す。補正Ｂ⁻処理（ステップＳ４４０４）は、補正Ｂ⁺処理（ステップＳ４４０３）と同一内容であるが、扱う生起確率総和ＴＯＰは１より大きい値であり、更新処理（ステップＳ５００１）を繰り返すことで、ある時点の補正Ｂ⁻で生起確率総和ＴＯＰが１を下回ることとなる。

[0159] 生起確率総和ＴＯＰが１未満となると、補正Ｂ⁺の場合と同様、１以下の最大漸近値に収束するまで、更新処理（ステップＳ５００１）を繰り返すこととなる。これにより、１より大きい生起確率総和ＴＯＰを減少させて、１に漸近させることができ、圧縮効率の向上を図ることができる。

[0160] ＜更新処理（ステップＳ５００１）＞

図５２は、図５０および図５１に示した更新処理（ステップＳ５００１）の詳細な処理手順を示すフローチャートである。まず、生成装置３９００は、未選択の圧縮符号長があるか否かを判断する（ステップＳ５２０１）。未選択の圧縮符号長がある場合（ステップＳ５２０１：Ｙｅｓ）、生成装置３９００は、未選択の中で最短の圧縮符号長を１つ選択する（ステップＳ５２０２）。生成装置３９００は、選択圧縮符号長が上限長Ｎ（補正Ａでの最長圧縮符号長）であるか否かを判断する（ステップＳ５２０３）。

[0161] 選択圧縮符号長が上限長Ｎでない場合（ステップＳ５２０３：Ｎｏ）、生成装置３９００は、選択圧縮符号長の葉数（文字情報の種類数）を、生起確率総和ＴＯＰで割る（ステップＳ５２０４）。生成部３９０７は、選択圧縮符号長の葉数をこの割り算結果の値に更新する。割り算結果の値については、小数点以下は切り捨て、四捨五入、切り上げのいずれでもよい。そして、

ステップＳ５２０１に戻る。

[0162] また、ステップＳ５２０３において、選択圧縮符号長が上限長Ｎである場合（ステップＳ５２０３：Ｙｅｓ）、生成装置３９００は、葉数の総和から、ステップＳ５２０５で更新済みの葉数の総和を引くことにより、選択圧縮符号長（この場合は、上限長Ｎ）の葉数を引き算結果の値に更新する（ステップＳ５２０６）。そして、ステップＳ５２０１に戻る。

[0163] ステップＳ５２０１において、未選択の圧縮符号長（上限長Ｎ以下の圧縮符号長）がない場合（ステップＳ５２０１：Ｎｏ）、生成装置３９００は、葉数の更新後における生起確率総和ＴＯＰを算出し（ステップＳ５２０７）、図５０および図５１のステップＳ５００２に移行する。

[0164] ＜枝数特定処理（ステップＳ４４０６）＞

図５３は、図４４に示した枝数特定処理（ステップＳ４４０６）の詳細な処理手順を示すフローチャートである。まず、生成装置３９００は、特定部３９０８により、最大圧縮符号長ＣＬｍａｘ（＝Ｎ）と最小圧縮符号長ＣＬｍｉｎ（＝Ｍ）との差分Ｄ（＝Ｎ－Ｍ）を算出する（ステップＳ５３０１）。たとえば、Ｎ＝１１の場合、図５を参照すると、Ｍ＝６である。したがって、Ｄ＝５である。

[0165] つぎに、生成装置３９００は、２のべき数の変数ｊをｊ＝０とし、圧縮符号長の変数ＣＬをＣＬ＝Ｎとする（ステップＳ５３０２）。そして、生成装置３９００は、ｊ＞Ｄであるか否かを判断する（ステップＳ５３０３）。ｊ＞Ｄでない場合（ステップＳ５３０３：Ｎｏ）、生成装置３９００は、圧縮符号長ＣＬの葉当たりの枝数ｂ（ＣＬ）を算出する（ステップＳ５３０４）。圧縮符号長ＣＬの葉当たりの枝数ｂ（ＣＬ）は、ｂ（ＣＬ）＝2^{j}で算出される。たとえば、ｊ＝０のとき、圧縮符号長ＣＬ＝Ｎ＝１１であるため、圧縮符号長１１ビットでの葉当たりの枝数ｂ（１１）は、ｂ（１１）＝2^{j}＝2^{0}＝１となる。

[0166] つぎに、生成装置３９００は、圧縮符号長ＣＬの総枝数Ｂ（Ｌ）を算出する（ステップＳ５３０５）。圧縮符号長ＣＬの総枝数Ｂ（Ｌ）は、Ｂ（Ｌ）

＝Ｌ（ＣＬ）×ｂ（ＣＬ）で算出される。Ｌ（ＣＬ）は、圧縮符号長ＣＬでの葉数（文字情報の種類数）である。たとえば、ｊ＝０のとき、圧縮符号長ＣＬ＝Ｎ＝１１であるため、圧縮符号長１１ビットでの総枝数Ｂ（Ｌ）は、１２１６×2^0＝１２１６となる。

[0167] このあと、生成装置３９００は、ｊをインクリメントし、圧縮符号長ＣＬをデクリメントして（ステップＳ５３０６）、ステップＳ５３０３に戻り、インクリメント後のｊがｊ＞Ｄであるか否かが判断される。なお、Ｎ＝１１の場合は、ｊ＝Ｄになるとｊ＝Ｄ＝５となり、ＣＬ＝Ｍ＝６となる。したがって、ステップＳ５３０４では、圧縮符号長ＣＬ（５ビット）当たりの枝数ｂ（６）は、ｂ（６）＝2^6＝６４となる。同様に、総枝数Ｂ（Ｌ）は、Ｂ（６）＝０×2^6＝０となる。そして、ｊ＞Ｄである場合（ステップＳ５３０３：Ｙｅｓ）、構築処理（ステップＳ４４０７）に移行する。

[0168] ＜構築処理（ステップＳ４４０７）＞

図５４は、図４４に示した構築処理（ステップＳ４４０７）の詳細な処理手順を示すフローチャートである。まず、生成装置３９００は、圧縮符号長ＣＬをＣＬ＝ＣＬｍｉｎ＝Ｍとする（ステップＳ５４０１）。つぎに、生成装置３９００は、圧縮符号長ＣＬでの未選択の葉があるか否かを判断する（ステップＳ５４０２）。未選択の葉がある場合（ステップＳ５４０２：Ｙｅｓ）、生成装置３９００は、葉へのポインタ生成処理（ステップＳ５４０３）を実行して、ステップＳ５４０２に戻る。葉へのポインタ生成処理（ステップＳ５４０３）では、葉の構造体ごとに、圧縮符号長ＣＬに応じた枝数分の葉へのポインタ群を生成する。なお、葉へのポインタ生成処理（ステップＳ５４０３）の詳細については後述する。

[0169] 一方、ステップＳ５４０２において、未選択の葉がない場合（ステップＳ５４０２：Ｎｏ）、生成装置３９００は、ＣＬ＞Ｎであるか否かを判断する（ステップＳ５４０４）。ＣＬ＞Ｎでない場合（ステップＳ５４０４：Ｎｏ）、生成装置３９００は、ＣＬをインクリメントして（ステップＳ５４０５）、ステップＳ５４０２に戻る。一方、ＣＬ＞Ｎである場合（ステップＳ５

４０４：Ｙｅｓ）、2^N分枝無節点ハフマン木が構築されたこととなり、一連処理が終了することとなる。

[0170] ＜葉へのポインタ生成処理（ステップＳ５４０３）＞

図５５は、図５４に示した葉へのポインタ生成処理（ステップＳ５４０３）の詳細な処理手順を示すフローチャートである。まず、生成装置３９００は、未選択の葉Ｌを選択し（ステップＳ５５０１）、選択葉へのポインタ数ｋをｋ＝１に設定する（ステップＳ５５０２）。そして、生成装置３９００は、選択葉へのポインタＰＬ（ｋ）の先行ビット列を、選択葉の圧縮符号に設定する（ステップＳ５５０３）。たとえば、上限長Ｎ＝１１については、選択葉が、文字情報「０」の葉の構造体である場合、圧縮符号は「０００００」である。したがって、選択葉へのポインタＰＬ（ｋ）の先行ビット列も、図７に示したように、「０００００」となる。

[0171] つぎに、生成装置３９００は、選択葉へのポインタＰＬ（ｋ）の後続ビット列のビット長を、最大圧縮符号長Ｎから選択葉の圧縮符号長ＣＬを引いた差分とし、後続ビット列の初期値をオール０に設定する（ステップＳ５５０４）。たとえば、選択葉が、文字情報「０」の葉の構造体である場合、圧縮符号長ＣＬは６ビットであるため、後続ビット列のビット長は５ビット（＝１１－６）となる。ｋ＝１の場合は、後続ビット列はオール０となるため、後続ビット列は、５ビットの「０００００」となる。

[0172] そして、生成装置３９００は、選択葉へのポインタＰＬ（ｋ）を根の構造体に格納する（ステップＳ５５０５）。このあと、生成装置３９００は、ｋ＞ｂ（ＣＬ）であるか否かを判断する（ステップＳ５５０６）。ｂ（ＣＬ）は、選択葉の圧縮符号長ＣＬの葉当たりの枝数である。ｋ＞ｂ（ＣＬ）でない場合（ステップＳ５５０６：Ｎｏ）、選択葉に割り当てられたすべての枝について葉へのポインタが生成されていないため、生成装置３９００は、ｋをインクリメントする（ステップＳ５５０７）。

[0173] そして、生成装置３９００は、現在の後続ビット列をインクリメントし、インクリメント後の後続ビット列を先行ビット列の末端に連結することで、

あらたに、選択葉へのポインタＰＬ（ｋ）を生成する（ステップＳ５５０８）。そして、生成装置３９００は、選択葉へのポインタＰＬ（ｋ）を根の構造体に格納し（ステップＳ５５０９）、ステップＳ５５０６に戻る。ステップＳ５５０６～ステップＳ５５０９を繰り返すことで、葉当たりの枝数分の葉へのポインタ群が生成されることとなる。そして、ステップＳ５５０６において、ｋ＞ｂ（ＣＬ）である場合（ステップＳ５５０６：Ｙｅｓ）、ステップＳ５４０２に移行する。

[0174] このように、本実施の形態によれば、対象ファイル群に出現する文字情報の種類数に応じて、2^N分枝無節点ハフマン木の最大枝数2^Nを最適な本数に設定することができるため、2^N分枝無節点ハフマン木のサイズの適正化を図ることができる。また、本実施の形態によれば、上限長Ｎが２～４の整数倍でない場合（たとえば、上限長Ｎ＝１１，１３）であっても、圧縮効率のよい2^N分枝無節点ハフマン木を生成することができる。

[0175] ＜補正Ｂ処理の他の例＞

つぎに、補正Ｂ処理の他の例について説明する。上述した補正B^+処理および補正B^-処理では、圧縮符号長の葉数ごとに、各圧縮符号長の葉数を生起確率総和で割ることにより、各圧縮符号長の葉数を更新することとした。これに対し、本例では、最小圧縮符号長ＣＬｍｉｎから最大圧縮符号長ＣＬｍａｘ（すなわち、上限長Ｎ）に向かって、小さい圧縮符号長の葉数がそれより大きい圧縮符号長の葉数よりも増減するように、葉数を移動させる。

[0176] 具体的には、本例の補正B^+処理では、小さい圧縮符号長の葉数がそれより大きい圧縮符号長の葉数よりも減少するように、葉数を補正する。一方、本例の補正B^-処理では、小さい圧縮符号長の葉数がそれより大きい圧縮符号長の葉数よりも増加するように、葉数を移動させる。その後は、補正後の葉数を生起確率総和で割ることで、各圧縮符号長の葉数を更新する。

[0177] なお、以下に説明する例では、ある圧縮符号長ＣＬの葉数Ｌ（ＣＬ）を、圧縮符号長（ＣＬ＋１）の葉数Ｌ（ＣＬ＋１）よりも増加／減少するように補正をおこなう例について説明するが、圧縮符号長ＣＬよりも大きい圧縮符

号長（ＣＬ＋２），（ＣＬ＋３），…であってもよい。また、移動対象葉数の個数は、１個に限らず、２個以上でもよい。たとえば、移動元の圧縮符号長の葉数をすべて移動先に移動させてもよい。

[0178] 図５６は、補正Ｂ+処理（ステップＳ４４０３）の他の例の詳細な処理手順を示すフローチャートである。補正Ｂ+処理（ステップＳ４４０３）において、最小圧縮符号長ＣＬｍｉｎをＭとする。また、最大圧縮符号長ＣＬｍａｘは上限長Ｎとなる。図５６では、葉数を１個ずつ移動させる例を示している。

[0179] まず、生成装置３９００は、圧縮符号長ＣＬをＣＬ＝Ｍに設定する（ステップＳ５６０１）。つぎに、生成装置３９００は、圧縮符号長ＣＬがＣＬ＝Ｎであるか否かを判断する（ステップＳ５６０２）。ＣＬ＝Ｎでない場合（ステップＳ５６０２：Ｎｏ）、圧縮符号長ＣＬが最大圧縮符号長ＣＬｍａｘに到達していないこととなる。この場合、生成装置３９００は、葉数Ｌ（ＣＬ）がＬ（ＣＬ）＝０であるか否かを判断する（ステップＳ５６０３）。

[0180] Ｌ（ＣＬ）＝０の場合（ステップＳ５６０３：Ｙｅｓ）、移動させる葉数Ｌ（ＣＬ）がないため、生成装置３９００は、圧縮符号長ＣＬをインクリメントし（ステップＳ５６０４）、ステップＳ５６０２に戻る。そして、圧縮符号長ＣＬがＣＬ＝Ｎとなった場合（ステップＳ５６０２：Ｙｅｓ）、ステップＳ５６１３に移行し、生成装置３９００は、今回の補正における圧縮符号長ごとの葉数で確定させることとなる。

[0181] また、ステップＳ５６０３において、葉数Ｌ（ＣＬ）がＬ（ＣＬ）＝０でない場合（ステップＳ５６０３：Ｎｏ）、生成装置３９００は、現在対象となっている圧縮符号長ＣＬの葉数Ｌ（ＣＬ）をデクリメントし（ステップＳ５６０５）、移動先となる圧縮符号長（ＣＬ＋１）の葉数Ｌ（ＣＬ＋１）をインクリメントする（ステップＳ５６０６）。

[0182] そして、圧縮符号長ＣＬ，（ＣＬ＋１）については、ステップＳ５６０５，ステップＳ５６０６において葉数が増減したため、生成装置３９００は、圧縮符号長ＣＬについての生起確率と圧縮符号長（ＣＬ＋１）についての生

起確率とを再計算する（ステップＳ５６０７）。このあと、生成装置３９００は、各圧縮符号長ＣＬｍｉｎ～ＣＬｍａｘの最新の葉数での生起確率総和ＴＯＰを算出する（ステップＳ５６０８）。

[0183] そして、生成装置３９００は、算出されたＴＯＰがｔ＞ＴＯＰであるか否かを判断する（ステップＳ５６０９）。ｔ＞ＴＯＰである場合（ステップＳ５６０９：Ｙｅｓ）、まだ改善の余地があるため、ステップＳ５６０３に戻る。一方、ｔ＞ＴＯＰでない場合（ステップＳ５６０９：Ｎｏ）、生成装置３９００は、ｔ≦ＴＯＰ≦１であるか否かを判断する（ステップＳ５６１０）。ｔ≦ＴＯＰ≦１である場合（ステップＳ５６１０：Ｙｅｓ）、これ以上補正する必要がないため、ステップＳ５６１３に移行する。

[0184] 一方、ｔ≦ＴＯＰ≦１でない場合（ステップＳ５６１０：Ｎｏ）、生成装置３９００は、現在対象となっている圧縮符号長Ｌの葉数Ｌ（ＣＬ）をインクリメントし（ステップＳ５６１１）、移動先の圧縮符号長（ＣＬ＋１）の葉数Ｌ（ＣＬ＋１）をデクリメントする（ステップＳ５６１２）。すなわち、ｔ≦ＴＯＰ≦１でない場合（ステップＳ５６１０：Ｎｏ）は、ＴＯＰ＞１であるため、今回の補正自体が失敗となる。

[0185] したがって、ステップＳ５６０５およびステップＳ５６０６での葉数移動を、ステップＳ５６１１およびステップＳ５６１２で元に戻すこととなる。このあと、ステップＳ５６０４に戻る。ステップＳ５６０４では、圧縮符号長ＣＬがインクリメントされるため、葉数移動による生起確率の増加がインクリメント前の圧縮符号長ＣＬに比べて１／２に抑えられる。したがって、葉数ＣＬをインクリメントするほど、微調整が可能となり、より微小な補正幅でｔ≦ＴＯＰ≦１に収めることができる。

[0186] また、ステップＳ５６１３になると、生成装置３９００は、そのときの圧縮符号長ＣＬｍｉｎ～ＣＬｍａｘごとの葉数Ｌ（ＣＬｍｉｎ）～葉数Ｌ（ＣＬｍａｘ）で確定させることとなる（ステップＳ５６１３）。その後、ステップＳ４４０５に移行することで、補正Ｂ+処理（ステップＳ４４０３）を終了する。

[0187] 図５７は、補正Ｂ-処理（ステップＳ４４０４）の他の例の詳細な処理手順を示すフローチャートである。補正Ｂ-処理（ステップＳ４４０４）は、図５６に示した補正Ｂ+処理（ステップＳ４４０３）とほぼ同一内容であるため、同一処理には同一ステップ番号を付す。異なる点は、ステップＳ５７０５，ステップＳ５７０６，ステップＳ５７０９，ステップＳ５７１１，ステップＳ５７１２である。

[0188] まず、生成装置３９００は、圧縮符号長ＣＬをＣＬ＝Ｍに設定する（ステップＳ５６０１）。つぎに、生成装置３９００は、圧縮符号長ＣＬがＣＬ＝Ｎであるか否かを判断する（ステップＳ５６０２）。ＣＬ＝Ｎでない場合（ステップＳ５６０２：Ｎｏ）、圧縮符号長ＣＬが最大圧縮符号長ＣＬｍａｘに到達していないこととなる。この場合、生成装置３９００は、葉数Ｌ（ＣＬ）がＬ（ＣＬ）＝０であるか否かを判断する（ステップＳ５６０３）。

[0189] Ｌ（ＣＬ）＝０の場合（ステップＳ５６０３：Ｙｅｓ）、移動させる葉数Ｌ（ＣＬ）がないため、生成装置３９００は、圧縮符号長ＣＬをインクリメントし（ステップＳ５６０４）、ステップＳ５６０２に戻る。そして、圧縮符号長ＣＬがＣＬ＝Ｎとなった場合（ステップＳ５６０１：Ｙｅｓ）、ステップＳ５６１３に移行し、生成装置３９００は、今回の補正における圧縮符号長ごとの葉数で確定させることとなる。

[0190] また、ステップＳ５６０３において、葉数Ｌ（ＣＬ）がＬ（ＣＬ）＝０でない場合（ステップＳ５６０３：Ｎｏ）、生成装置３９００は、現在対象となっている圧縮符号長ＣＬの葉数Ｌ（ＣＬ）をインクリメントし（ステップＳ５７０５）、移動先となる圧縮符号長（ＣＬ＋１）の葉数Ｌ（ＣＬ＋１）をデクリメントする（ステップＳ５７０６）。

[0191] そして、圧縮符号長ＣＬ，（ＣＬ＋１）については、ステップＳ５７０５，ステップＳ５７０６において葉数が増減したため、生成装置３９００は、圧縮符号長ＣＬについての生起確率と圧縮符号長（ＣＬ＋１）についての生起確率とを再計算する（ステップＳ５６０７）。このあと、生成装置３９００は、各圧縮符号長ＣＬｍｉｎ～ＣＬｍａｘの最新の葉数での生起確率総和

ＴＯＰを算出する（ステップＳ５６０８）。

[0192] そして、生成装置３９００は、算出されたＴＯＰが１＜ＴＯＰであるか否かを判断する（ステップＳ５７０９）。１＜ＴＯＰである場合（ステップＳ５７０９：Ｙｅｓ）、まだ改善の余地があるため、ステップＳ５６０３に戻る。一方、１＜ＴＯＰでない場合（ステップＳ５７０９：Ｎｏ）、生成装置３９００は、ｔ≦ＴＯＰ≦１であるか否かを判断する（ステップＳ５６１０）。ｔ≦ＴＯＰ≦１である場合（ステップＳ５６１０：Ｙｅｓ）、これ以上補正する必要がないため、ステップＳ５６１３に移行する。

[0193] 一方、ｔ≦ＴＯＰ≦１でない場合（ステップＳ５６１０：Ｎｏ）、生成装置３９００は、現在対象となっている圧縮符号長Ｌの葉数Ｌ（ＣＬ）をデクリメントし（ステップＳ５７１１）、移動先の圧縮符号長（ＣＬ＋１）の葉数Ｌ（ＣＬ＋１）をインクリメントする（ステップＳ５７１２）。すなわち、ｔ≦ＴＯＰ≦１でない場合（ステップＳ５６１０：Ｎｏ）は、ｔ＞ＴＯＰであるため、今回の補正自体が失敗となる。

[0194] したがって、ステップＳ５７０５およびステップＳ５７０６での葉数移動を、ステップＳ５７１１およびステップＳ５７１２で元に戻すこととなる。このあと、ステップＳ５６０４に戻る。ステップＳ５６０４では、圧縮符号長ＣＬがインクリメントされるため、葉数移動による生起確率の減少がインクリメント前の圧縮符号長ＣＬに比べて１／２に抑えられる。したがって、葉数ＣＬをインクリメントするほど、微調整が可能となり、より微小な補正幅でｔ≦ＴＯＰ≦１に収めることができる。

[0195] また、ステップＳ５６１３になると、生成装置３９００は、そのときの圧縮符号長ＣＬｍｉｎ～ＣＬｍａｘごとの葉数Ｌ（ＣＬｍｉｎ）～葉数Ｌ（ＣＬｍａｘ）で確定させることとなる（ステップＳ５６１３）。その後、ステップＳ４４０５に移行することで、補正Ｂ⁻処理（ステップＳ４４０４）を終了する。

[0196] 図５８は、図５６に示した補正Ｂ⁺処理の他の例を適用した場合における、図２の（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１２）を示す説

明図である。図５８では、移動元の圧縮符号長の葉数を、１ビット小さい移動先の圧縮符号長の葉数に１個ずつシフトさせた例を示している。図５８では、図５６におけるしきい値ｔをｔ＝１とする。図５８では、補正Ａでの生起確率総和ＴＯＰが「０．８２３」であるため、補正B^+処理が適用される。まず、移動先を最小圧縮符号長である６ビットとし、移動元を１ビット大きい７ビットとする。補正B^+1では、圧縮符号長７ビットの葉数９を１つずつ、最小圧縮符号長６ビットに移動させる。

[0197] 生成装置３９００は、この移動ごとに、生起確率総和ＴＯＰを求めて、１以下の最大漸近値に収束するか否かを判断することとなる。本例において、補正B^+1では生起確率総和ＴＯＰが１を超えていない。そして、移動元の圧縮符号長７ビットの葉数が０になった場合、移動先の圧縮符号長６ビットの葉数は１１（＝９＋２）となる。このときの補正B^-1での生起確率総和ＴＯＰは「０．８９４」であるため、さらに補正B^+が必要である。

[0198] つぎに、補正B^+2では、移動先および移動元の圧縮符号長を１ビット大きくする。すなわち、移動先の圧縮符号長を６ビットから７ビットとし、移動元の圧縮符号長を７ビットから８ビットとする。

[0199] そして、移動元の圧縮符号長８ビットの葉数２２を１つずつ、補正B^+1で葉数が０になった移動先の圧縮符号長７ビットに移動させる。生成装置３９００は、この移動ごとに、生起確率総和ＴＯＰを求めて、１以下の最大漸近値に収束するか否かを判断することとなる。本例において、補正B^+2では生起確率総和ＴＯＰが１を超えていない。そして、移動元の圧縮符号長８ビットの葉数が０になった場合、移動先の圧縮符号長７ビットの葉数は２２（＝０＋２２）となる。このときの補正B^+2での生起確率総和ＴＯＰは「０．９７９」であるため、さらに補正B^+が必要である。

[0200] つぎに、補正B^+3では、移動先および移動元の圧縮符号長を１ビット大きくする。すなわち、移動先の圧縮符号長を７ビットから８ビットとし、移動元の圧縮符号長を８ビットから９ビットとする。

[0201] そして、移動元の圧縮符号長９ビットの葉数１９を１つずつ、補正B^+2で

葉数が０になった移動先の圧縮符号長８ビットに移動させる。生成装置３９００は、この移動ごとに、生起確率総和ＴＯＰを求めて、１以下の最大漸近値に収束するか否かを判断することとなる。本例において、移動元の圧縮符号長９ビットの葉数が２８から１９になり、移動先の圧縮符号長８ビットの葉数が０から１１になると、生起確率総和ＴＯＰが１を超える。したがって、それぞれ葉数を１つ戻し、移動元の圧縮符号長９ビットの葉数を１８、移動先の圧縮符号長８ビットの葉数を１０として、補正B^+3を終了する。このときの補正B^+3での生起確率総和ＴＯＰは「０．９９９」であるため、さらに補正B^+が必要である。

[0202] つぎに、補正B^+4では、移動先および移動元の圧縮符号長を１ビット大きくする。すなわち、移動先の圧縮符号長を８ビットから９ビットとし、移動元の圧縮符号長を９ビットから１０ビットとする。

[0203] そして、移動元の圧縮符号長１０ビットの葉数２９を１つずつ、補正B^+3で葉数が１８になった移動先の圧縮符号長９ビットに移動させる。生成装置３９００は、この移動ごとに、生起確率総和ＴＯＰを求めて、１以下の最大漸近値に収束するか否かを判断することとなる。本例において、移動元の圧縮符号長１０ビットの葉数が２９から２７になり、移動先の圧縮符号長９ビットの葉数が１８から２０になると、生起確率総和ＴＯＰが１を超える。したがって、それぞれ葉数を１つ戻し、移動元の圧縮符号長１０ビットの葉数を２８、移動先の圧縮符号長９ビットの葉数を１９として、補正B^+4を終了する。このときの補正B^+4での生起確率総和ＴＯＰは「１．０００」であるため、さらに補正B^+が可能である。

[0204] つぎに、補正B^+5では、移動先および移動元の圧縮符号長を１ビット大きくする。すなわち、移動先の圧縮符号長を９ビットから１０ビットとし、移動元の圧縮符号長を１０ビットから１１ビットとする。

[0205] そして、移動元の圧縮符号長１１ビットの葉数２５を１つずつ、補正B^+4で葉数が２８になった移動先の圧縮符号長１０ビットに移動させる。生成装置３９００は、この移動ごとに、生起確率総和ＴＯＰを求めて、１以下の最

大漸近値に収束するか否かを判断することとなる。本例において、移動元の圧縮符号長１１ビットの葉数が２５から２３になり、移動先の圧縮符号長１０ビットの葉数が２８から３０になると、生起確率総和ＴＯＰが１を超える。したがって、それぞれ葉数を１つ戻し、移動元の圧縮符号長１１ビットの葉数を２４、移動先の圧縮符号長１０ビットの葉数を２９として、補正$B^{+}5$を終了する。このときの補正$B^{+}5$での生起確率総和ＴＯＰは「１．０００」であるため、さらに補正B^{+}が可能である。

[0206] つぎに、補正$B^{+}6$では、移動先および移動元の圧縮符号長を１ビット大きくする。すなわち、移動先の圧縮符号長を１０ビットから１１ビットとし、移動元の圧縮符号長を１１ビットから１２ビットとする。

[0207] そして、移動元の圧縮符号長１２ビットの葉数２２１４を１つずつ、補正$B^{+}5$で葉数が２４になった移動先の圧縮符号長１１ビットに移動させる。生成装置３９００は、この移動ごとに、生起確率総和ＴＯＰを求めて、１以下の最大漸近値に収束するか否かを判断することとなる。本例において、移動元の圧縮符号長１２ビットの葉数が２２１４から２２１３になり、移動先の圧縮符号長１１ビットの葉数が２４から２５になると、生起確率総和ＴＯＰが１を超える。したがって、それぞれ葉数を１つ戻すこととなる。戻した葉数は、補正$B^{+}5$終了時点での葉数となる。補正$B^{+}5$終了時点での生起確率総和ＴＯＰは「１．０００」である。本例では、これ以上、移動元の圧縮符号長を大きくすることができないため、補正$B^{+}5$終了時点での葉数が確定することとなる。

[0208] 図５９は、上述した補正B^{-}処理の他の例を適用した場合における、図２の（３）葉数特定～（５）葉の構造体生成の詳細（Ｎ＝１１）を示す説明図である。図５９では、移動元の圧縮符号長の葉数をすべて、１ビット大きい移動先の圧縮符号長の葉数にシフトさせた例を示している。図５９では、図５７におけるしきい値ｔをｔ＝１とする。図５９では、補正Ａでの生起確率総和ＴＯＰが「１．１４６」であるため、補正B^{-}処理が適用される。まず、移動元を最小圧縮符号長である５ビットとし、移動先を１ビット大きい６ビッ

トとする。補正Ｂ⁻１では、移動元の圧縮符号長５ビットの葉数２を１つずつ、圧縮符号長６ビットに移動させる。

[0209] 生成装置３９００は、この移動ごとに、生起確率総和ＴＯＰを求めて、１以下の最大漸近値に収束するか否かを判断することとなる。本例において、補正Ｂ⁻１では生起確率総和ＴＯＰが１以下にならない。そして、移動元の圧縮符号長５ビットの葉数が０になった場合、移動先の圧縮符号長６ビットの葉数は１１（＝９＋２）となる。このときの補正Ｂ⁻１での生起確率総和ＴＯＰは「１．１１５」であるため、さらに補正Ｂ⁻が必要である。

[0210] つぎに、補正Ｂ⁻２では、移動元および移動先の圧縮符号長を１ビット大きくする。すなわち、移動元の圧縮符号長を５ビットから６ビットとし、移動先の圧縮符号長を６ビットから７ビットとする。

[0211] そして、移動元の圧縮符号長６ビットの葉数１１を１つずつ、移動先の圧縮符号長７ビットの葉数２２に移動させる。生成装置３９００は、この移動ごとに、生起確率総和ＴＯＰを求めて、１以下の最大漸近値に収束するか否かを判断することとなる。本例において、補正Ｂ⁻２では生起確率総和ＴＯＰが１以下にならない。そして、移動元の圧縮符号長６ビットの葉数が０になった場合、移動先の圧縮符号長７ビットの葉数は３３（＝２２＋１１）となる。このときの補正Ｂ⁻２での生起確率総和ＴＯＰは「１．０２９」であるため、さらに補正Ｂ⁻が必要である。

[0212] つぎに、補正Ｂ⁻３では、移動元および移動先の圧縮符号長を１ビット大きくする。すなわち、移動元の圧縮符号長を６ビットから７ビットとし、移動先の圧縮符号長を７ビットから８ビットとする。

[0213] そして、移動元の圧縮符号長７ビットの葉数３３を１つずつ、移動先の圧縮符号長８ビットの葉数２８に移動させる。生成装置３９００は、この移動ごとに、生起確率総和ＴＯＰを求めて、１以下の最大漸近値に収束するか否かを判断することとなる。本例において、補正Ｂ⁻３では、移動元である圧縮符号長７ビットの葉数が２５、移動先である圧縮符号長８ビットの葉数が３６に到達すると、生起確率総和ＴＯＰが１以下の「０．９９８」となる。こ

れ以上葉数移動をしても、生起確率総和ＴＯＰが低下するため、このときの各圧縮符号長の葉数で確定させることとなる。

[0214] ＜圧縮処理＞

つぎに、上述した2^N分枝無節点ハフマン木を用いた対象ファイル群の圧縮処理について説明する。圧縮処理は、生成装置３９００が実行してもよく、また、2^N分枝無節点ハフマン木が格納された情報処理装置が実行してもよい。すなわち、少なくとも2^N分枝無節点ハフマン木が格納されていればよい。以下、生成装置３９００および情報処理装置を総称してコンピュータとする。ここで、まず、圧縮対象文字列とその圧縮符号を例に挙げて説明する。

[0215] 図６０は、圧縮対象文字列の一例を示す説明図である。図６０では、「次の兎は」という文字列の１６ビットコード（１６進と２進）を示している。

[0216] 図６１は、図６０に示した圧縮対象文字列の圧縮符号を示す説明図である。図６１に示した圧縮符号は、図２２に示した2^N分枝無節点ハフマン木（Ｎ＝１２）を用いて圧縮した例を示している。図６１では、図６０と比較すると、高位文字「次」は、１６ビットから１２ビットに圧縮されている。また、高位文字「の」は、１６ビットから７ビットに圧縮されている。

[0217] また、「兎」の上位分割文字コード「０ｘ５１」は、８ビットから１１ビットに圧縮されている。「兎」の下位分割文字コード「０ｘ４Ｅ」は、８ビットから１１ビットに圧縮されている。高位文字「は」は、１６ビットから９ビットに圧縮されている。なお、分割文字コードについては、圧縮後のほうがビット長が長いが、「兎」のような単一文字はそもそも出現回数が少ないために分割されているため、対象ファイル群全体で見れば問題ない。

[0218] 図６２は、圧縮対象文字列の他の例を示す説明図である。図６２では、「兎は動物園の」という文字列の１６ビットコード（１６進と２進）を示している。

[0219] 図６３は、図６２に示した圧縮対象文字列の圧縮符号を示す説明図である。図６３に示した圧縮符号は、図３２に示した2^N分枝無節点ハフマン木（Ｎ＝１３）を用いて圧縮した例を示している。図６３では、図６２と比較する

と、「兎」の上位分割文字コード「０ｘ５１」は、８ビットから１２ビットに圧縮されている。「兎」の下位分割文字コード「０ｘ４Ｅ」は、８ビットから１２ビットに圧縮されている。高位文字「は」は、１６ビットから１０ビットに圧縮されている。

[0220] 「動物園」は基礎単語に該当するため、４８ビットから１３ビットに圧縮されている。また、高位文字「の」は、１６ビットから８ビットに圧縮されている。

[0221] なお、分割文字コードについては、圧縮後のほうがビット長が長いが、「兎」のような単一文字はそもそも出現回数が少ないために分割されているため、対象ファイル群全体で見れば問題ない。また、基礎単語は、そのビット列に比べて圧縮符号が非常に短くなるため（最大でも１３ビット）、２文字（高位文字）以上であれば十分圧縮効率が向上することとなる。

[0222] ＜圧縮処理の具体例＞

図６４は、2^N分枝無節点ハフマン木を用いた圧縮処理の具体例を示す説明図である。まず、コンピュータは、対象ファイル群Ｆｓから１文字目の圧縮対象文字コードを取得し、対象ファイル上の位置を保持しておく。そして、コンピュータは、特殊単語の構造体３６００に対して２分木探索をおこなう。特殊単語は２文字以上の文字コード列であるため、１文字目の圧縮対象文字コードがヒットした場合、２文字目の文字コードを圧縮対象文字コードとして取得する。

[0223] そして、２文字目の文字コードは、１文字目の圧縮対象文字コードがヒットした位置から探索する。３文字目以降も、不一致の圧縮対象文字コードが出現するまで繰り返し２分木探索をおこなう。一致する特殊単語ｒａ（ａは葉の番号）が探索された場合、特殊単語の構造体３６００において対応付けされている葉Ｌａへのポインタにより葉Ｌａの構造体にアクセスする。そして、コンピュータは、アクセス先の葉Ｌａの構造体に格納されている特殊単語ｒａの圧縮符号を探索して、圧縮バッファ６４００に格納する。

[0224] 一方、不一致の圧縮文字コードが出現した場合、特殊単語の構造体３６０

０に対する２分木探索を終了する（ＥＯＴ（Ｅｎｄ　Ｏｆ　Ｔｒａｎｓｍｉｓｓｉｏｎ）まで進む）。そして、コンピュータは、１文字目の圧縮対象文字コードを再度レジスタにセットして、高位文字コードの構造体３４００に対する２分木探索をおこなう。

[0225] 一致する文字コードｅｂ（ｂは葉の番号）が探索された場合、コンピュータは、その葉Ｌｂへのポインタにより葉Ｌｂの構造体にアクセスする。そして、コンピュータは、アクセス先の葉Ｌｂの構造体に格納されている文字コードｅｂの圧縮符号を探索して、圧縮バッファ６４００に格納する。

[0226] 一方、一致する文字コードが出現せず２分木探索を終了した場合、圧縮対象文字コードは高位文字コードではないため、コンピュータは、上位８ビットと下位８ビットに分割する。そして、コンピュータは、上位８ビットの分割文字コードについて、分割文字コードの構造体３５００に対する２分木探索をおこなう。一致する分割文字コードＤｃ１（ｃ１は葉の番号）が探索された場合、コンピュータは、その葉Ｌｃ１へのポインタにより葉Ｌｃ１の構造体にアクセスする。そして、コンピュータは、アクセス先の葉Ｌｃ１の構造体に格納されている分割文字コードＤｃ１の圧縮符号を探索して、圧縮バッファ６４００に格納する。

[0227] 引き続き、コンピュータは、下位８ビットの分割文字コードについて、分割文字コードの構造体３５００に対する２分木探索をおこなう。一致する分割文字コードＤｃ２（ｃ２は葉の番号）が探索された場合、コンピュータは、その葉Ｌｃ２へのポインタにより葉Ｌｃ２の構造体にアクセスする。そして、コンピュータは、アクセス先の葉Ｌｃ２の構造体に格納されている分割文字コードＤｃ２の圧縮符号を探索して、圧縮バッファ６４００に格納する。

[0228] ＜ファイル圧縮処理＞

図６５は、コンピュータが自動実行する2^N分枝無節点ハフマン木を用いたファイル圧縮処理手順を示すフローチャートである。まず、コンピュータは、ファイル番号：ｐをｐ＝１とし（ステップＳ６５０１）、対象ファイルＦ

ｐを読み込む（ステップＳ６５０２）。つぎに、コンピュータは、圧縮処理を実行して（ステップＳ６５０３）、ファイル番号：ｐをインクリメントする（ステップＳ６５０４）。そして、コンピュータは、ｐ＞αであるか否かを判断する（ステップＳ６５０５）。αは対象ファイル群Ｆｓの総数である。ｐ＞αでない場合（ステップＳ６５０５：Ｎｏ）、ステップＳ６５０２に戻る。一方、ｐ＞αである場合（ステップＳ６５０５：Ｙｅｓ）、一連のファイル圧縮処理を終了する。

[0229] 図６６は、図６５に示した圧縮処理（ステップＳ６５０３）の詳細な処理手順を示すフローチャート（その１）である。図６６において、まず、コンピュータは、対象ファイル群Ｆｓに圧縮対象文字コードがあるか否かを判断する（ステップＳ６６０１）。ある場合（ステップＳ６６０１：Ｙｅｓ）、コンピュータは、圧縮対象文字コードを取得してレジスタにセットする（ステップＳ６６０２）。そして、コンピュータは、先頭の圧縮対象文字コードか否かを判断する（ステップＳ６６０３）。

[0230] ここで、先頭の圧縮対象文字コードとは、未圧縮の１文字目の文字コードをいう。先頭である場合（ステップＳ６６０３：Ｙｅｓ）、コンピュータは、その圧縮対象文字コードの対象ファイル群Ｆｓ上の位置（先頭位置）となるポインタを取得し（ステップＳ６６０４）、ステップＳ６６０５に移行する。一方、先頭でない場合（ステップＳ６６０３：Ｎｏ）、先頭位置を取得せずにステップＳ６６０５に移行する。

[0231] そして、コンピュータは、特殊単語の構造体３６００に対して２分木探索をおこなう（ステップＳ６６０５）。圧縮対象文字コードが一致した場合（ステップＳ６６０６：Ｙｅｓ）、コンピュータは、連続して一致した文字コード列が特殊単語（の文字コード列）に該当するか否かを判断する（ステップＳ６６０７）。該当しない場合（ステップＳ６６０７：Ｎｏ）、コンピュータは、ステップＳ６６０２に戻って後続の文字コードを圧縮対象文字コードとして取得する。この場合、後続の文字コードは先頭ではないため、先頭位置は取得しないこととなる。

[0232] 一方、ステップＳ６６０７において、特殊単語に該当する場合（ステップＳ６６０７：Ｙｅｓ）、コンピュータは、該当する特殊単語の葉Ｌ＃へのポインタにより葉Ｌ＃の構造体にアクセスする（ステップＳ６６０８）。そして、コンピュータは、ポイントされた葉Ｌ＃の構造体に格納されている特殊単語の圧縮符号を抽出する（ステップＳ６６０９）。

[0233] この後、コンピュータは、抽出された圧縮符号を圧縮バッファ６４００に格納して（ステップＳ６６１０）、ステップＳ６６０１に戻る。このループが特殊単語の圧縮処理の流れとなる。ステップＳ６６０１において、圧縮対象文字コードがない場合（ステップＳ６６０１：Ｎｏ）、コンピュータは、対象ファイルＦｐから圧縮された圧縮ファイルｆｐを圧縮バッファ６４００からファイル出力して保存する（ステップＳ６６１１）。そして、ステップＳ６５０４に移行する。一方、ステップＳ６６０６において不一致となった場合（ステップＳ６６０６：Ｎｏ）、１６ビットの文字コードの圧縮処理のループに入る。

[0234] 図６７は、図６５に示した圧縮処理（ステップＳ６５０３）の詳細な処理手順を示すフローチャート（その２）である。図６７において、コンピュータは、ステップＳ６６０４で取得された先頭位置のポインタを参照して、対象ファイル群Ｆｓから圧縮対象文字コードを取得してレジスタにセットする（ステップＳ６７０１）。

[0235] つぎに、コンピュータは、圧縮対象文字コードについて、高位文字コードの構造体３４００に対して２分木探索をおこなう（ステップＳ６７０２）。一致した場合（ステップＳ６７０３：Ｙｅｓ）、コンピュータは、該当する文字の葉Ｌ＃へのポインタにより葉Ｌ＃の構造体にアクセスする（ステップＳ６７０４）。そして、コンピュータは、ポイントされた葉Ｌ＃の構造体に格納されている圧縮対象文字コードの圧縮符号を抽出する（ステップＳ６７０５）。

[0236] この後、コンピュータは、探索された圧縮符号を圧縮バッファ６４００に格納して（ステップＳ６７０６）、ステップＳ６６０１に戻る。このループ

が１６ビットの文字コードの圧縮処理の流れとなる。一方、ステップＳ６７０３において一致する文字コードが存在しなかった場合（ステップＳ６７０３：Ｎｏ）、分割文字コードの圧縮処理のループに入る。

[0237] 図６８は、図６５に示した圧縮処理（ステップＳ６５０３）の詳細な処理手順を示すフローチャート（その３）である。図６８において、まず、コンピュータは、圧縮対象文字コードを上位８ビットと下位８ビットとに分割し（ステップＳ６８０１）、上位８ビットの分割文字コードを抽出する（ステップＳ６８０２）。そして、コンピュータは、分割文字コードの構造体３５００に対して２分木探索をおこなう（ステップＳ６８０３）。

[0238] そして、コンピュータは、探索された分割文字コードの葉Ｌ＃へのポインタにより葉Ｌ＃の構造体にアクセスする（ステップＳ６８０４）。そして、コンピュータは、ポイントされた葉Ｌ＃の構造体に格納されている分割文字コードの圧縮符号を抽出する（ステップＳ６８０５）。この後、コンピュータは、探索された圧縮符号を圧縮バッファ６４００に格納する（ステップＳ６８０６）。

[0239] つぎに、コンピュータは、下位８ビットが探索済みか否かを判断し（ステップＳ６８０７）、探索済みでない場合（ステップＳ６８０７：Ｎｏ）、コンピュータは、下位８ビットの分割文字コードを抽出して（ステップＳ６８０８）、ステップＳ６８０３～Ｓ６８０６を実行する。一方、下位８ビットが探索済みである場合（ステップＳ６８０７：Ｙｅｓ）、ステップＳ６６０１に戻り、特殊単語の圧縮処理のループに入る。

[0240] このように、2^N分枝無節点ハフマン木を用いた圧縮処理では、内部節点がないため根に向かって探索する必要はなく、ポイントされた葉Ｌ＃の構造体に格納されている文字情報を抽出して、圧縮バッファ６４００に書き込むだけでよい。したがって、圧縮処理の高速化を図ることができる。

[0241] また、圧縮対象文字コードが格納されている葉Ｌ＃の構造体を、特殊単語の構造体３６００、高位文字コードの構造体３４００および分割文字コードの構造体３５００により即座に特定することができる。したがって、2^N分枝

無節点ハフマン木の葉を探索する必要がなく、圧縮処理の高速化を図ることができる。また、低位文字コードを上位ビットコードと下位ビットコードに分割することで、６万種以上ある低位の文字コードをたかだか２５６種の分割文字コードの圧縮符号に圧縮することができる。したがって、圧縮率の向上を図ることができる。

[0242] ＜伸長処理例＞

つぎに、2^N分枝無節点ハフマン木により圧縮された圧縮符号列を伸長する伸長処理例について説明する。

[0243] 図６９～図７３は、図６１に示した圧縮符号列の伸長処理例を示す説明図である。伸長処理では、コンピュータは、レジスタに圧縮符号列をセットし、マスクパターンにより圧縮符号を抽出する。コンピュータは、抽出した圧縮符号を、１パス（１枝分のアクセス）で2^N分枝無節点ハフマン木の根から探索する。そして、コンピュータは、アクセスした葉Ｌ＃の構造体に格納されている文字コードを読み出して伸長バッファ６９００に格納する。

[0244] 圧縮符号を抽出するため、コンピュータは、マスクパターンのマスク位置をオフセットする。また、マスクパターンの初期値を“０ｘＦＦＦ００００”とする。このマスクパターンは先頭１２ビットが“１”であり、後続の２０ビットが“０”のビット列である。

[0245] コンピュータは、ビットアドレスａｂｉとバイトオフセットｂｙｏｓとビットオフセットｂｉｏｓとを算出する。ビットアドレスａｂｉは、抽出された圧縮符号のビット位置を示す値であり、今回のビットアドレスａｂｉは、前回のビットアドレスａｂｉに前回抽出された圧縮符号の圧縮符号長ｌｅｇを加算した値となる。なお、初期状態では、ビットアドレスａｂｉはａｂｉ＝０とする。

[0246] バイトオフセットｂｙｏｓは、メモリに保持されている圧縮符号列のバイト境界を示す値であり、ビットアドレスａｂｉ／８の商で求められる。たとえば、バイトオフセットｂｙｏｓ＝０のときは、メモリに記憶されている先頭からの圧縮符号列をレジスタにセットし、バイトオフセットｂｙｏｓ＝１

のときは、メモリに記憶されている先頭１バイト目からの圧縮符号列をレジスタにセットする。

[0247] また、ビットオフセットｂｉｏｓは、マスクパターンのマスク位置（“ＦＦ”）をオフセットする値であり、ビットアドレスａｂｉ／８の余りである。たとえば、ビットオフセットｂｉｏｓ＝０のときは、マスク位置はシフトされないこととなり、マスクパターンは、“０ｘＦＦＦ０００００”となる。一方、ビットオフセットｂｉｏｓ＝４のときは、マスク位置は末尾方向に４ビットシフトすることとなり、マスクパターンは、“０ｘ０ＦＦＦ０００００”となる。

[0248] レジスタシフト数ｒｓは、マスクパターンとのＡＮＤ演算後のレジスタ内の圧縮符号列を末尾方向にシフトするビット数であり、ｒｓ＝３２－１２－ｂｉｏｓで求められる。このシフトにより、コンピュータは、シフト後のレジスタの末尾ｍビットのビット列を対象ビット列として抽出する。コンピュータは、対象ビット列の抽出後はレジスタをクリアする。

[0249] なお、図６９～図７３において、メモリには図６１に示した圧縮符号列が保持されているものとする。また、図６９～図７３のメモリ内のブロックは１バイトのビット列を示しており、内部の数字は、バイト境界となるバイト位置を示している。

[0250] 図６９は、初期状態（（Ａ）の状態）を示している。（Ａ）では、ビットアドレスａｂｉ＝０により、バイトオフセットｂｙｏｓ＝０、ビットオフセットｂｉｏｓ＝０となる。バイトオフセットｂｙｏｓ＝０により、コンピュータは、メモリに保持されている圧縮符号列のうち先頭から４バイト分（図中、網掛け）の圧縮符号列をレジスタにセットする。

[0251] また、ビットオフセットｂｉｏｓ＝０により、マスクパターンは、“０ｘＦＦＦ０００００”である。したがって、レジスタにセットされた圧縮符号列とマスクパターン“０ｘＦＦＦ０００００”を論理積（ＡＮＤ）演算することにより、ＡＮＤ結果が得られる。

[0252] また、ビットオフセットｂｉｏｓ＝０により、レジスタシフト数ｒｓは、

ｒｓ＝３２－ｍ―ｂｉｏｓ＝３２－１２－０＝２０となる。したがって、レジスタ内のＡＮＤ結果を末尾方向に２０ビット分シフトする。このシフトによりレジスタには、“１０１１１１１１１０１１”が残されるため、コンピュータは、末尾１２ビットを対象ビット列として抽出する。この場合は、“１０１１１１１１１０１１”が対象ビット列として抽出される。抽出後、レジスタはクリアされる。

[0253] 図２２に示したように、2^N分枝無節点ハフマン木の根の構造体には、葉Ｌ１～Ｌ２３２９へのポインタが格納されている。したがって、コンピュータは、2^N分枝無節点ハフマン木の根の構造体の中から、抽出された対象ビット列“１０１１１１１１１０１１”と一致する葉Ｌ＃へのポインタを探索する。この場合、葉Ｌ６９１へのポインタ群の中の１つと一致するため、コンピュータは、該当する葉Ｌ６９１へのポインタを読み出して、葉Ｌ６９１の構造体にアクセスする。

[0254] 葉Ｌ６９１の構造体には、文字コード“０ｘ２１６Ｂ”（高位文字：「次」に相当）が格納されているため、コンピュータは、当該文字コード“０ｘ２１６Ｂ”を抽出して伸長バッファ６９００に格納する。また、葉Ｌ６９１の構造体には、文字コード“０ｘ２１６Ｂ”の圧縮符号長ｌｅｇ（＝１２ビット）も格納されているため、コンピュータは、文字コード“０ｘ２１６Ｂ”の圧縮符号長ｌｅｇも抽出する。コンピュータは、この抽出された圧縮符号長ｌｅｇによりビットアドレスａｂｉを更新する。この場合、更新後のビットアドレスａｂｉはａｂｉ＝０＋１２＝１２となる。

[0255] 図７０は、図６９に示した（Ａ）の状態からレジスタをバイトオフセットｂｙｏｓ分シフトした場合の伸長処理（（Ｂ）の状態）を示している。前回である（Ａ）のビットアドレスａｂｉはａｂｉ＝０、圧縮符号長ｌｅｇは１２ビットであるため、（Ｂ）のビットアドレスａｂｉはａｂｉ＝１２ビットとなる。

[0256] また、このビットアドレスａｂｉ＝１２により、バイトオフセットｂｙｏｓ＝１、ビットオフセットｂｉｏｓ＝４となる。バイトオフセットｂｙｏｓ

＝１により、コンピュータは、メモリに保持されている圧縮符号列のうち先頭１バイト目から４バイト分（図中、網掛け）の圧縮符号列をレジスタにセットする。

[0257] また、ビットオフセットｂｉｏｓ＝４により、マスクパターンは、“０ｘ０ＦＦＦ０００Ｏ”である。したがって、レジスタにセットされた圧縮符号列とマスクパターン“０ｘ０ＦＦＦ０００Ｏ”を論理積（ＡＮＤ）演算することにより、ＡＮＤ結果が得られる。

[0258] また、ビットオフセットｂｉｏｓ＝４により、レジスタシフト数ｒｓは、ｒｓ＝３２－ｍ―ｂｉｏｓ＝３２－１２－４＝１６となる。したがって、コンピュータは、レジスタ内のＡＮＤ結果を末尾方向に１６ビット分シフトする。このシフトによりレジスタには、“０００００１０１０１１１００１１”が残されるため、コンピュータは、末尾１２ビットを対象ビット列として抽出する。この場合は、“０１０１０１１１００１１”が対象ビット列として抽出される。抽出後、レジスタはクリアされる。

[0259] 図２２に示したように、2^N分枝無節点ハフマン木の根の構造体には、葉Ｌ１～Ｌ２３２９へのポインタが格納されている。したがって、コンピュータは、2^N分枝無節点ハフマン木の根の構造体の中から、抽出された対象ビット列“０１０１０１１１００１１”と一致する葉Ｌ＃へのポインタを探索する。この場合、対象ビット列“０１０１０１１１００１１”が葉Ｌ２７へのポインタ群の中の１つと一致するため、コンピュータは、該当する葉Ｌ２７へのポインタを読み出して、葉Ｌ２７の構造体にアクセスする。

[0260] 葉Ｌ２７の構造体には、文字コード“０ｘ６Ｅ３０”（文字：「の」に相当）が格納されているため、コンピュータは、当該文字コード“０ｘ６Ｅ３０”を抽出して伸長バッファ６９００に格納する。また、葉Ｌ２７の構造体には、文字コード“０ｘ６Ｅ３０”の圧縮符号長ｌｅｇ（＝７ビット）も格納されているため、コンピュータは、文字コード“０ｘ６Ｅ３０”の圧縮符号長ｌｅｇも抽出する。コンピュータは、この抽出された圧縮符号長ｌｅｇによりビットアドレスａｂｉを更新する。この場合、更新後のビットアドレ

スａｂｉはａｂｉ＝１２＋７＝１９となる。

[0261] 図７１は、図７０に示した（Ｂ）の状態からレジスタをバイトオフセットｂｙｏｓ分シフトした場合の伸長処理（（Ｃ）の状態）を示している。前回である（Ｂ）のビットアドレスａｂｉはａｂｉ＝１２、圧縮符号長ｌｅｇは７ビットであるため、（Ｃ）のビットアドレスａｂｉはａｂｉ＝１９ビットとなる。

[0262] また、このビットアドレスａｂｉ＝１９により、バイトオフセットｂｙｏｓ＝２、ビットオフセットｂｉｏｓ＝３となる。バイトオフセットｂｙｏｓ＝２により、メモリに保持されている圧縮符号列のうち先頭２バイト目から４バイト分（図中、網掛け）の圧縮符号列をレジスタにセットする。

[0263] また、ビットオフセットｂｉｏｓ＝３により、マスクパターンは、“０ｘ１ＦＦＥ０００”である。したがって、レジスタにセットされた圧縮符号列とマスクパターン“０ｘ１ＦＦＥ０００”を論理積（ＡＮＤ）演算することにより、ＡＮＤ結果が得られる。

[0264] また、ビットオフセットｂｉｏｓ＝３により、レジスタシフト数ｒｓは、ｒｓ＝３２－ｍ―ｂｉｏｓ＝３２－１２－３＝１７となる。したがって、コンピュータは、レジスタ内のＡＮＤ結果を末尾方向に１７ビット分シフトする。このシフトによりレジスタには、“０００１００１１００１１１０１”が残されるため、末尾１２ビットを対象ビット列として抽出する。この場合は、“１００１１００１１１０１”が対象ビット列として抽出される。抽出後、レジスタはクリアされる。

[0265] 図２２に示したように、2^N分枝無節点ハフマン木の根の構造体には、葉Ｌ１～Ｌ２３２９へのポインタが格納されている。したがって、コンピュータは、2^N分枝無節点ハフマン木の根の構造体セルの中から、抽出された対象ビット列“１００１１００１１１０１”と一致する葉Ｌ＃へのポインタを探索する。この場合、対象ビット列“１００１１００１１１０１”が葉Ｌ１６７へのポインタ群の中の１つと一致するため、コンピュータは、該当する葉Ｌ１６７へのポインタを読み出して、葉Ｌ１６７の構造体にアクセスする。

[0266] 葉Ｌ１６７の構造体には、分割文字コード“０ｘ５１”が格納されているため、コンピュータは、当該文字コード“０ｘ５１”を抽出して伸長バッファ６９００に格納する。また、葉Ｌ１６７の構造体には、文字コード“０ｘ５１”の圧縮符号長ｌｅｇ（＝１１ビット）も格納されているため、コンピュータは、文字コード“０ｘ５１”の圧縮符号長ｌｅｇも抽出する。コンピュータは、この抽出された圧縮符号長ｌｅｇによりビットアドレスａｂｉを更新する。この場合、更新後のビットアドレスａｂｉはａｂｉ＝１９＋１１＝３０となる。

[0267] 図７２は、図７１に示した（Ｃ）の状態からレジスタをバイトオフセットｂｙｏｓ分シフトした場合の伸長処理（（Ｄ）の状態）を示している。前回である（Ｃ）のビットアドレスａｂｉはａｂｉ＝１９、圧縮符号長ｌｅｇは１１ビットであるため、（Ｄ）のビットアドレスａｂｉはａｂｉ＝３０ビットとなる。

[0268] また、このビットアドレスａｂｉ＝３０により、バイトオフセットｂｙｏｓ＝３、ビットオフセットｂｉｏｓ＝６となる。バイトオフセットｂｙｏｓ＝３により、コンピュータは、メモリに保持されている圧縮符号列のうち先頭３バイト目から４バイト分（図中、網掛け）の圧縮符号列をレジスタにセットする。

[0269] また、ビットオフセットｂｉｏｓ＝６により、マスクパターンは、“０ｘ０３ＦＦＣ０００”である。したがって、レジスタにセットされた圧縮符号列とマスクパターン“０ｘ０３ＦＦＣ０００”を論理積（ＡＮＤ）演算することにより、ＡＮＤ結果が得られる。

[0270] また、ビットオフセットｂｉｏｓ＝６により、レジスタシフト数ｒｓは、ｒｓ＝３２－ｍ―ｂｉｏｓ＝３２－１２－６＝１４となる。したがって、コンピュータは、レジスタ内のＡＮＤ結果を末尾方向に１４ビット分シフトする。このシフトによりレジスタには、“０００００１００１１００００１１”が残されるため、コンピュータは、末尾１２ビットを対象ビット列として抽出する。この場合は、“１００１１００００１１１”が対象ビット列

として抽出される。抽出後、レジスタはクリアされる。

[0271] 図２２に示したように、２Ｎ分枝無節点ハフマン木の根の構造体には、葉Ｌ１～Ｌ２３２９へのポインタが格納されている。したがって、コンピュータは、２Ｎ分枝無節点ハフマン木の根の構造体の中から、抽出された対象ビット列“１００１１００００１１１”と一致する葉Ｌ＃へのポインタを探索する。この場合、対象ビット列“１００１１００００１１１”が葉Ｌ１５６へのポインタ群の中の一つと一致するため、葉Ｌ１５６へのポインタを読み出して、葉Ｌ１５６の構造体にアクセスする。

[0272] 葉Ｌ１５６の構造体には、分割文字コード“０ｘ４Ｅ”が格納されているため、コンピュータは、当該文字コード“０ｘ４Ｅ”を抽出して伸長バッファ６９００に格納する。また、葉Ｌ１５６の構造体には、文字コード“０ｘ４Ｅ”の圧縮符号長ｌｅｇ（＝１１ビット）も格納されているため、コンピュータは、文字コード“０ｘ４Ｅ”の圧縮符号長ｌｅｇも抽出する。コンピュータは、この抽出された圧縮符号長ｌｅｇによりビットアドレスａｂｉを更新する。この場合、更新後のビットアドレスａｂｉはａｂｉ＝３０＋１１＝４１となる。

[0273] 図７３は、図７２に示した（Ｄ）の状態からレジスタをバイトオフセットｂｙｏｓ分シフトした場合の伸長処理（（Ｅ）の状態）を示している。前回である（Ｄ）のビットアドレスａｂｉはａｂｉ＝３０、圧縮符号長ｌｅｇは１１ビットであるため、（Ｅ）のビットアドレスａｂｉはａｂｉ＝４１ビットとなる。

[0274] また、このビットアドレスａｂｉ＝４１により、バイトオフセットｂｙｏｓ＝５、ビットオフセットｂｉｏｓ＝１となる。バイトオフセットｂｙｏｓ＝５により、コンピュータは、メモリに保持されている圧縮符号列のうち先頭５バイト目から４バイト分（図中、網掛け）の圧縮符号列をレジスタにセットする。

[0275] また、ビットオフセットｂｉｏｓ＝１により、マスクパターンは、“０ｘ７ＦＦ８０００”である。したがって、レジスタにセットされた圧縮符号

列とマスクパターン“０ｘ７ＦＦ８０００”を論理積（ＡＮＤ）演算することにより、ＡＮＤ結果が得られる。

[0276] また、ビットオフセットｂｉｏｓ＝１により、レジスタシフト数ｒｓは、ｒｓ＝３２－ｍ―ｂｉｏｓ＝３２－１２－１＝１９となる。したがって、コンピュータは、レジスタ内のＡＮＤ結果を末尾方向に１９ビット分シフトする。このシフトによりレジスタには、“０１０００１１０１＊＊＊”（＊は１または０）が残されるため、コンピュータは、末尾１２ビットを対象ビット列として抽出する。この場合は、“１０００１１０１＊＊＊”が対象ビット列として抽出される。抽出後、レジスタはクリアされる。

[0277] 図２２に示したように、2^N分枝無節点ハフマン木の根の構造体には、葉Ｌ１～Ｌ２３２９へのポインタが格納されている。したがって、コンピュータは、2^N分枝無節点ハフマン木の根の構造体の中から、抽出された対象ビット列“１０００１１０１＊＊＊”と一致する葉Ｌ＃へのポインタを探索する。この場合、対象ビット列“１０００１１０１＊＊＊”が葉Ｌ７９へのポインタと一致するため、コンピュータは、葉Ｌ７９へのポインタを読み出して、葉Ｌ７９の構造体にアクセスする。

[0278] 葉Ｌ７９の構造体には、文字コード“０ｘ６Ｆ３０”が格納されているため、コンピュータは、当該文字コード“０ｘ６Ｆ３０”を抽出して伸長バッファ６９００に格納する。また、葉Ｌ７９の構造体には、文字コード“０ｘ６Ｆ３０”の圧縮符号長ｌｅｇ（＝８ビット）も格納されているため、コンピュータは、文字コード“０ｘ６Ｆ３０”の圧縮符号長ｌｅｇも抽出する。この抽出された圧縮符号長ｌｅｇによりビットアドレスａｂｉを更新する。この場合、更新後のビットアドレスａｂｉはａｂｉ＝４１＋８＝４９となる。

[0279] このように、葉へのポインタには圧縮符号が含まれているため、いずれの葉へのポインタが抽出された場合であっても、圧縮符号を特定するまでもなく、直接葉の構造体にアクセスすることができる。したがって、伸長処理の高速化を実現することができる。

[0280] 図７４～図７８は、図６３に示した圧縮符号列の伸長処理例を示す説明図である。図７４は、初期状態（（Ａ）の状態）を示している。（Ａ）では、ビットアドレスａｂｉ＝０により、バイトオフセットｂｙｏｓ＝０、ビットオフセットｂｉｏｓ＝０となる。バイトオフセットｂｙｏｓ＝０により、コンピュータは、メモリに保持されている圧縮符号列のうち先頭から４バイト分（図中、網掛け）の圧縮符号列をレジスタにセットする。

[0281] また、ビットオフセットｂｉｏｓ＝０により、マスクパターンは、“０ｘＦＦＦ８０００”である。したがって、レジスタにセットされた圧縮符号列とマスクパターン“０ｘＦＦＦ８０００”を論理積（ＡＮＤ）演算することにより、ＡＮＤ結果が得られる。

[0282] また、ビットオフセットｂｉｏｓ＝０により、レジスタシフト数ｒｓは、ｒｓ＝３２－ｍ―ｂｉｏｓ＝３２－１３－０＝１９となる。したがって、コンピュータは、レジスタ内のＡＮＤ結果を末尾方向に１９ビット分シフトする。このシフトによりレジスタには、“１００１１００１１１０１０”が残されるため、コンピュータは、末尾１３ビットを対象ビット列として抽出する。この場合は、“１００１１００１１１０１０”が対象ビット列として抽出される。抽出後、レジスタはクリアされる。

[0283] 図３２に示したように、2^N分枝無節点ハフマン木の根の構造体には、葉Ｌ１～Ｌ５４０１へのポインタが格納されている。したがって、コンピュータは、2^N分枝無節点ハフマン木の根の構造体の中から、抽出された対象ビット列“１００１１００１１１０１０”と一致する葉Ｌ＃へのポインタを探索する。この場合、葉Ｌ１６７へのポインタ群の中の１つと一致するため、コンピュータは、該当する葉Ｌ１６７へのポインタを読み出して、葉Ｌ１６７の構造体にアクセスする。

[0284] 葉Ｌ１６７の構造体には、分割文字コード“０ｘ５１”が格納されているため、コンピュータは、当該文字コード“０ｘ５１”を抽出して伸長バッファ６９００に格納する。また、葉Ｌ１６７の構造体には、文字コード“０ｘ５１”の圧縮符号長ｌｅｇ（＝１１ビット）も格納されているため、コンピ

ュータは、文字コード“０ｘ５１”の圧縮符号長ｌｅｇも抽出する。コンピュータは、この抽出された圧縮符号長ｌｅｇによりビットアドレスａｂｉを更新する。この場合、更新後のビットアドレスａｂｉはａｂｉ＝０＋１１＝１１となる。

[0285] 図７５は、図７４に示した（Ａ）の状態からレジスタをバイトオフセットｂｙｏｓ分シフトした場合の伸長処理（（Ｂ）の状態）を示している。前回である（Ａ）のビットアドレスａｂｉはａｂｉ＝０、圧縮符号長ｌｅｇは１１ビットであるため、（Ｂ）のビットアドレスａｂｉはａｂｉ＝１１ビットとなる。

[0286] また、このビットアドレスａｂｉ＝１１により、バイトオフセットｂｙｏｓ＝１、ビットオフセットｂｉｏｓ＝３となる。バイトオフセットｂｙｏｓ＝１により、コンピュータは、メモリに保持されている圧縮符号列のうち先頭１バイト目から４バイト分（図中、網掛け）の圧縮符号列をレジスタにセットする。

[0287] また、ビットオフセットｂｉｏｓ＝３により、マスクパターンは、“０ｘ１ＦＦＦ０００”である。したがって、レジスタにセットされた圧縮符号列とマスクパターン“０ｘ１ＦＦＦ０００”を論理積（ＡＮＤ）演算することにより、ＡＮＤ結果が得られる。

[0288] また、ビットオフセットｂｉｏｓ＝３により、レジスタシフト数ｒｓは、ｒｓ＝３２－ｍ―ｂｉｏｓ＝３２－１３－３＝１６となる。したがって、コンピュータは、レジスタ内のＡＮＤ結果を末尾方向に１６ビット分シフトする。このシフトによりレジスタには、“０００１００１１００００１１１０”が残されるため、コンピュータは、末尾１３ビットを対象ビット列として抽出する。この場合は、“１００１１００００１１１０”が対象ビット列として抽出される。抽出後、レジスタはクリアされる。

[0289] 図３２に示したように、2^N分枝無節点ハフマン木の根の構造体には、葉Ｌ１～Ｌ５４０１へのポインタが格納されている。したがって、コンピュータは、2^N分枝無節点ハフマン木の根の構造体の中から、抽出された対象ビット

列“１００１１００００１１１０”と一致する葉Ｌ＃へのポインタを探索する。この場合、対象ビット列“１００１１００００１１１０”が葉Ｌ１５６へのポインタ群の中の１つと一致するため、コンピュータは、該当する葉Ｌ１５６へのポインタを読み出して、葉Ｌ１５６の構造体にアクセスする。

[0290] 葉Ｌ１５６の構造体には、分割文字コード“０ｘ４Ｅ”が格納されているため、コンピュータは、当該文字コード“０ｘ４Ｅ”を抽出して伸長バッファ６９００に格納する。また、葉Ｌ１５６の構造体には、文字コード“０ｘ４Ｅ”の圧縮符号長ｌｅｇ（＝１１ビット）も格納されているため、コンピュータは、文字コード“０ｘ４Ｅ”の圧縮符号長ｌｅｇも抽出する。この抽出された圧縮符号長ｌｅｇによりビットアドレスａｂｉを更新する。この場合、更新後のビットアドレスａｂｉはａｂｉ＝１１＋１１＝２２となる。

[0291] 図７６は、図７５に示した（Ｂ）の状態からレジスタをバイトオフセットｂｙｏｓ分シフトした場合の伸長処理（（Ｃ）の状態）を示している。前回である（Ｂ）のビットアドレスａｂｉはａｂｉ＝１１、圧縮符号長ｌｅｇは１１ビットであるため、（Ｃ）のビットアドレスａｂｉはａｂｉ＝２２ビットとなる。

[0292] また、このビットアドレスａｂｉ＝２２により、バイトオフセットｂｙｏｓ＝２、ビットオフセットｂｉｏｓ＝６となる。バイトオフセットｂｙｏｓ＝２により、コンピュータは、メモリに保持されている圧縮符号列のうち先頭２バイト目から４バイト分（図中、網掛け）の圧縮符号列をレジスタにセットする。

[0293] また、ビットオフセットｂｉｏｓ＝６により、マスクパターンは、“０ｘ０３ＦＦＥ０００”である。したがって、レジスタにセットされた圧縮符号列とマスクパターン“０ｘ０３ＦＦＥ０００”を論理積（ＡＮＤ）演算することにより、ＡＮＤ結果が得られる。

[0294] また、ビットオフセットｂｉｏｓ＝６により、レジスタシフト数ｒｓは、ｒｓ＝３２－ｍ―ｂｉｏｓ＝３２－１３－６＝１３となる。したがって、コンピュータは、レジスタ内のＡＮＤ結果を末尾方向に１３ビット分シフトす

る。このシフトによりレジスタには、“０００００１０００１１０１１１１１”が残されるため、コンピュータは、末尾１３ビットを対象ビット列として抽出する。この場合は、“１０００１１０１１１１１”が対象ビット列として抽出される。抽出後、レジスタはクリアされる。

[0295] 図３２に示したように、2^N分枝無節点ハフマン木の根の構造体には、葉Ｌ１～Ｌ５４０１へのポインタが格納されている。したがって、コンピュータは、2^N分枝無節点ハフマン木の根の構造体セルの中から、抽出された対象ビット列“１０００１１０１１１１１”と一致する葉Ｌ＃へのポインタを探索する。この場合、対象ビット列“１０００１１０１１１１１”が葉Ｌ７９へのポインタ群の中の１つと一致するため、コンピュータは、該当する葉Ｌ７９へのポインタを読み出して、葉Ｌ７９の構造体にアクセスする。

[0296] 葉Ｌ７９の構造体には、文字コード“０ｘ６Ｆ３０”が格納されているため、コンピュータは、当該文字コード“０ｘ６Ｆ３０”を抽出して伸長バッファ６９００に格納する。また、葉Ｌ７９の構造体には、文字コード“０ｘ６Ｆ３０”の圧縮符号長ｌｅｇ（＝９ビット）も格納されているため、コンピュータは、文字コード“０ｘ６Ｆ３０”の圧縮符号長ｌｅｇも抽出する。コンピュータは、この抽出された圧縮符号長ｌｅｇによりビットアドレスａｂｉを更新する。この場合、更新後のビットアドレスａｂｉはａｂｉ＝２２＋９＝３１となる。

[0297] 図７７は、図７６に示した（Ｃ）の状態からレジスタをバイトオフセットｂｙｏｓ分シフトした場合の伸長処理（（Ｄ）の状態）を示している。前回である（Ｃ）のビットアドレスａｂｉはａｂｉ＝２２、圧縮符号長ｌｅｇは９ビットであるため、（Ｄ）のビットアドレスａｂｉはａｂｉ＝３１ビットとなる。

[0298] また、このビットアドレスａｂｉ＝３１により、バイトオフセットｂｙｏｓ＝３、ビットオフセットｂｉｏｓ＝７となる。バイトオフセットｂｙｏｓ＝３により、コンピュータは、メモリに保持されている圧縮符号列のうち先頭３バイト目から４バイト分（図中、網掛け）の圧縮符号列をレジスタにセ

ットする。

[0299] また、ビットオフセットｂｉｏｓ＝７により、マスクパターンは、“０ｘ０１ＦＦＦ０００”である。したがって、レジスタにセットされた圧縮符号列とマスクパターン“０ｘ０１ＦＦＦ０００”を論理積（ＡＮＤ）演算することにより、ＡＮＤ結果が得られる。

[0300] また、ビットオフセットｂｉｏｓ＝７により、レジスタシフト数ｒｓは、ｒｓ＝３２－ｍ―ｂｉｏｓ＝３２－１３－７＝１２となる。したがって、コンピュータは、レジスタ内のＡＮＤ結果を末尾方向に１２ビット分シフトする。このシフトによりレジスタには、“０００００００１１１１００１１００００１”が残されるため、コンピュータは、末尾１３ビットを対象ビット列として抽出する。この場合は、“１１１１００１１００００１”が対象ビット列として抽出される。抽出後、レジスタはクリアされる。

[0301] 図３２に示したように、2^N分枝無節点ハフマン木の根の構造体には、葉Ｌ１～Ｌ５４０１へのポインタが格納されている。したがって、コンピュータは、2^N分枝無節点ハフマン木の根の構造体の中から、抽出された対象ビット列“１１１１００１１００００１”と一致する葉Ｌ＃へのポインタを探索する。この場合、対象ビット列“１１１１００１１００００１”が葉Ｌ５４０１へのポインタ群の中の一つと一致するため、コンピュータは、葉Ｌ５４０１へのポインタを読み出して、葉Ｌ５４０１の構造体にアクセスする。

[0302] 葉Ｌ５４０１の構造体には、基礎単語“動物園”が格納されているため、コンピュータは、当該基礎単語“動物園”を抽出して伸長バッファ６９００に格納する。また、葉Ｌ５４０１の構造体には、基礎単語“動物園”の圧縮符号長ｌｅｇ（＝１３ビット）も格納されているため、コンピュータは、基礎単語“動物園”の圧縮符号長ｌｅｇも抽出する。コンピュータは、この抽出された圧縮符号長ｌｅｇによりビットアドレスａｂｉを更新する。この場合、更新後のビットアドレスａｂｉはａｂｉ＝３１＋１３＝４４となる。

[0303] 図７８は、図７７に示した（Ｄ）の状態からレジスタをバイトオフセットｂｙｏｓ分シフトした場合の伸長処理（（Ｅ）の状態）を示している。前回

である（Ｄ）のビットアドレスａｂｉはａｂｉ＝３１、圧縮符号長ｌｅｇは１３ビットであるため、（Ｅ）のビットアドレスａｂｉはａｂｉ＝４４ビットとなる。

[0304] また、このビットアドレスａｂｉ＝４４により、バイトオフセットｂｙｏｓ＝５、ビットオフセットｂｉｏｓ＝４となる。バイトオフセットｂｙｏｓ＝５により、コンピュータは、メモリに保持されている圧縮符号列のうち先頭５バイト目から４バイト分（図中、網掛け）の圧縮符号列をレジスタにセットする。

[0305] また、ビットオフセットｂｉｏｓ＝４により、マスクパターンは、“０ｘ０ＦＦＥ０００００”である。したがって、レジスタにセットされた圧縮符号列とマスクパターン“０ｘ０ＦＦＥ０００００”を論理積（ＡＮＤ）演算することにより、ＡＮＤ結果が得られる。

[0306] また、ビットオフセットｂｉｏｓ＝４により、レジスタシフト数ｒｓは、ｒｓ＝３２－ｍ―ｂｉｏｓ＝３２－１３－４＝１５となる。したがって、コンピュータは、レジスタ内のＡＮＤ結果を末尾方向に１５ビット分シフトする。このシフトによりレジスタには、“０００００１０１００１＊＊＊＊＊”（＊は１または０）が残されるため、コンピュータは、末尾１３ビットを対象ビット列として抽出する。この場合は、“０１０１００１＊＊＊＊＊”が対象ビット列として抽出される。抽出後、レジスタはクリアされる。

[0307] 図３２に示したように、2^N分枝無節点ハフマン木の根の構造体には、葉Ｌ１～Ｌ５４０１へのポインタが格納されている。したがって、コンピュータは、2^N分枝無節点ハフマン木の根の構造体の中から、抽出された対象ビット列“０１０１００１＊＊＊＊＊”と一致する葉Ｌ＃へのポインタを探索する。この場合、対象ビット列“０１０１００１＊＊＊＊＊”が葉Ｌ２７へのポインタと一致するため、コンピュータは、葉Ｌ２７へのポインタを読み出して、葉Ｌ２７の構造体にアクセスする。

[0308] 葉Ｌ２７の構造体には、文字コード“０ｘ６Ｅ３０”が格納されているため、コンピュータは、当該文字コード“０ｘ６Ｅ３０”を抽出して伸長バッ

ファ６９００に格納する。また、葉Ｌ２７の構造体には、文字コード“０ｘ６Ｅ３０”の圧縮符号長ｌｅｇ（＝９ビット）も格納されているため、コンピュータは、文字コード“０ｘ６Ｅ３０”の圧縮符号長ｌｅｇも抽出する。コンピュータは、この抽出された圧縮符号長ｌｅｇによりビットアドレスａｂｉを更新する。この場合、更新後のビットアドレスａｂｉはａｂｉ＝４４＋９＝５３となる。

[0309] このように、葉へのポインタには圧縮符号が含まれているため、いずれの葉へのポインタが抽出された場合であっても、圧縮符号を特定するまでもなく、直接葉の構造体にアクセスすることができる。したがって、伸長処理の高速化を実現することができる。

[0310] ＜伸長処理手順＞

図７９は、2^N分枝無節点ハフマン木を用いた伸長処理手順を示すフローチャート（その１）である。図７９において、まず、コンピュータは、ビットアドレスａｂｉをａｂｉ＝０とし（ステップＳ７９０１）、バイトオフセットｂｙｏｓを算出し（ステップＳ７９０２）、ビットオフセットｂｉｏｓを算出する（ステップＳ７９０３）。そして、コンピュータは、バイトオフセットｂｙｏｓの位置からの圧縮符号列をレジスタにセットする（ステップＳ７９０４）。

[0311] つぎに、コンピュータは、マスクパターンをビットオフセットｂｉｏｓ分、末尾方向にシフトして（ステップＳ７９０５）、レジスタにセットされた圧縮符号列とのＡＮＤ演算をおこなう（ステップＳ７９０６）。このあと、コンピュータは、レジスタシフト数ｒｓを算出して（ステップＳ７９０７）、ＡＮＤ演算後のレジスタをレジスタシフト数ｒｓ分、末尾にシフトする（ステップＳ７９０８）。

[0312] 図８０は、2^N分枝無節点ハフマン木を用いた伸長処理手順を示すフローチャート（その２）である。ステップＳ７９０８のあと、図８０において、コンピュータは、シフト後のレジスタから末尾Ｎビットを対象ビット列として抽出する（ステップＳ８００１）。つぎに、コンピュータは、2^N分枝無節点

ハフマン木の根の構造体から葉Ｌ＃へのポインタを特定し（ステップＳ８００２）、ポイント先となる葉Ｌ＃の構造体に１パスでアクセスする（ステップＳ８００３）。そして、コンピュータは、アクセス先の葉Ｌ＃の構造体から文字情報を抽出し（ステップＳ８００４）、抽出された文字情報を伸長バッファ６９００に書き込む（ステップＳ８００５）。

[0313] また、コンピュータは、葉Ｌ＃の構造体から圧縮符号長ｌｅｇを抽出し（ステップＳ８００６）、ビットアドレスａｂｉを更新する（ステップＳ８００７）。このあと、コンピュータは、メモリに圧縮符号列があるか否か、具体的には、マスクパターンによるマスク処理が施されていない圧縮符号列があるか否かを判断する（ステップＳ８００８）。たとえば、バイトオフセットｂｙｏｓに該当するバイト位置があるか否かにより判断する。圧縮符号列がある場合（ステップＳ８００８：Ｙｅｓ）、図７９のステップＳ７９０２に戻る。一方、圧縮符号列がない場合（ステップＳ８００８：Ｎｏ）、一連の伸長処理を終了する。

[0314] このような伸長処理により、圧縮符号列からＮビット単位で圧縮符号を抽出することができ、さらに、2^N分枝無節点ハフマン木の根の構造体にアクセスすることで、該当する枝番号となる葉Ｌ＃へのポインタを特定することができる。そして、アクセス先となる葉Ｌ＃の構造体から文字コードを抽出することで、圧縮符号を伸長する。このように、2^N分枝無節点ハフマン木は、内部節点を有していないため、葉Ｌ＃へのポインタが特定されれば、１パスで葉Ｌ＃の構造体にアクセスすることができ、伸長速度の高速化を図ることができる。

[0315] 以上説明したように、本実施の形態によれば、文字情報の総種類数に応じて2^N分枝無節点ハフマン木での圧縮符号長の上限長Ｎを１ビット刻みで調節することができる。したがって、文字情報の総種類数に適したサイズで2^N分枝無節点ハフマン木を生成することができ、省メモリ化を図ることができる。

[0316] 具体的には、文字情報の総種類数が2^{K-1}個より大きく2^K個以下である場合

は、上限長ＮをＫビットに決定することで、上述の補正Ａで示したように、上限長Ｋ以上の圧縮符号長の葉数は上限長Ｋの圧縮符号長の葉数に集約される。したがって、2^K分枝無節点ハフマン木が生成されるため、上限符号長が（Ｋ＋１）以上とした場合の無節点ハフマン木よりもサイズを小さくすることができる。

[0317] たとえば、文字情報の総種類数が2^{10}個より大きく2^{11}個以下である場合は、上限長Ｎを１１ビットに決定することで、2^{11}分枝無節点ハフマン木を生成することができる。また、文字情報の総種類数が2^{11}個より大きく2^{12}個以下である場合は、上限長Ｎを１２ビットに決定することで、2^{12}分枝無節点ハフマン木を生成することができる。さらに、文字情報の総種類数が2^{12}個より大きく2^{13}個以下である場合は、上限長Ｎを１３ビットに決定することで、2^{13}分枝無節点ハフマン木を生成することができる。

[0318] また、補正Ａでの生起確率総和がしきい値ｔ以上１以下であれば、十分な圧縮効率を見込める。したがって、補正Ａでの各圧縮符号長の葉数で2^N分枝無節点ハフマン木を生成することで、生成速度の高速化を図ることができる。

[0319] また、補正Ａでの生起確率総和がしきい値ｔ以上１以下でなければ、補正Ｂを実行することで、圧縮効率の向上を図ることができる。たとえば、しきい値ｔ未満であれば、補正$^+$処理を実行することで、生起確率総和を１に漸近させることができ、圧縮効率の向上を最大限までおこなうことができる。同様に、１より大きい場合でも、補正$^-$処理を実行することで、生起確率総和を１に漸近させることができ、圧縮効率の向上を最大限までおこなうことができる。

[0320] また、圧縮符号長間で葉数のシフトをおこなうことでも、生起確率総和を１に近似させることが可能となる。いずれにしても、生起確率総和が１に近似するように各圧縮符号長の葉数を最適な葉数とすることで、圧縮効率の向上を図ることができる。

[0321] なお、本実施の形態で説明した方法は、予め用意されたプログラムをパー

ソナル・コンピュータやワークステーション等のコンピュータで実行することにより実現することができる。本生成プログラムは、ハードディスク、フレキシブルディスク、ＣＤ－ＲＯＭ、ＭＯ、ＤＶＤ等のコンピュータで読み取り可能な記録媒体に記録され、コンピュータによって記録媒体から読み出されることによって実行される。また本生成プログラムは、インターネット等のネットワークを介して配布してもよい。

符号の説明

[0322] ３９００　生成装置

３９０１　集計部

３９０２　決定部

３９０３　補正部

３９０４　算出部

３９０５　判断部

３９０６　更新部

３９０７　生成部

３９０８　特定部

３９０９　構築部

請求の範囲

[請求項1] 対象ファイルに出現する文字情報群内の各文字情報の出現率に応じた生起確率で規定される圧縮符号長ごとに、前記文字情報の種類数を集計する集計工程と、

前記対象ファイルに出現する前記文字情報の総種類数に基づいて、最小圧縮符号長から最大圧縮符号長までの圧縮符号長群の中から、前記文字情報に割り当てられる圧縮符号長の上限長Ｎを決定する決定工程と、

前記集計工程によって集計された圧縮符号長ごとの前記文字情報の種類数のうち、前記決定工程によって決定された上限長Ｎを圧縮符号長とする前記文字情報の種類数を、前記上限長Ｎ以上の圧縮符号長での前記文字情報の種類数の総和に補正する補正工程と、

前記補正工程によって補正された補正後の前記圧縮符号長ごとの前記文字情報の種類数に基づいて、前記上限長Ｎを最大枝数とし、前記文字情報の各々の生起確率に応じた圧縮符号長の圧縮符号を葉とする2^N分枝無節点ハフマン木を構築する構築工程と、

をコンピュータに実行させることを特徴とする生成プログラム。

[請求項2] 前記決定工程は、

前記文字情報の総種類数が2^{K-1}個より大きく2^K個以下である場合は、前記上限長ＮをＫビットに決定することを特徴とする請求項１に記載の生成プログラム。

[請求項3] 前記決定工程は、

前記文字情報の総種類数が2^{10}個より大きく2^{11}個以下である場合は、前記上限長Ｎを１１ビットに決定することを特徴とする請求項２に記載の生成プログラム。

[請求項4] 前記決定工程は、

前記文字情報の総種類数が2^{11}個より大きく2^{12}個以下である場合は、前記上限長Ｎを１２ビットに決定することを特徴とする請求項２

に記載の生成プログラム。

[請求項5] 前記決定工程は、

前記文字情報の総種類数が2^{12}個より大きく2^{13}個以下である場合は、前記上限長Ｎを１３ビットに決定することを特徴とする請求項２に記載の生成プログラム。

[請求項6] 前記補正工程による補正後の前記上限長Ｎまでの各圧縮符号長での前記文字情報の種類数に基づいて、前記上限長Ｎまでの各圧縮符号長を規定する各生起確率の総和を算出する算出工程と、

前記算出工程によって算出された総和がしきい値以上１以下であるか否かを判断する判断工程と、

前記判断工程によって前記しきい値以上１以下であると判断された場合、前記上限長Ｎまでの圧縮符号長と、前記上限長Ｎまでの圧縮符号長ごとの前記文字情報の種類数と、前記文字情報の出現率と、に基づいて、前記文字情報に圧縮符号を割り当てた葉の構造体を生成する生成工程と、

前記上限長Ｎまでの圧縮符号長に基づいて、前記生成工程によって生成された葉の構造体１つ当たりの枝数を、前記上限長Ｎまでの圧縮符号長ごとに特定する特定工程と、を前記コンピュータに実行させ、

前記構築工程は、

前記葉の構造体内の圧縮符号に前記特定工程によって特定された枝数分の枝番号をあらわす各ビット列を連結した葉へのポインタ群を、葉の構造体ごとに生成することにより、前記各葉の構造体についての前記葉へのポインタ群を根とする2^{N}分枝無節点ハフマン木を構築することを特徴とする請求項１～５のいずれか一つに記載の生成プログラム。

[請求項7] 前記判断工程によって前記しきい値以上１以下でないと判断された場合、前記上限長Ｎまでの各圧縮符号長の前記文字情報の種類数を、前記総和で割ることにより、前記上限長Ｎまでの各圧縮符号長の前記

文字情報の種類数を更新する更新工程を前記コンピュータに実行させ、

前記算出工程は、

前記更新工程による更新後における前記上限長Ｎまでの各圧縮符号長の前記文字情報の種類数に基づいて、前記総和を再算出し、

前記判断工程は、

前記算出工程によって再算出された総和が１以下の最大値であるか否かを判断し、

前記生成工程は、

前記判断工程によって前記再算出された総和が前記最大値であると判断された場合、前記上限長Ｎまでの圧縮符号長と、前記上限長Ｎまでの圧縮符号長ごとの更新後における前記文字情報の種類数と、前記文字情報の出現率と、に基づいて、前記文字情報に圧縮符号を割り当てた葉の構造体を生成することを特徴とする請求項６に記載の生成プログラム。

[請求項8] 前記判断工程によって前記しきい値以上１以下でないと判断された場合、前記上限長Ｎまでの各圧縮符号長のうち第１の圧縮符号長の前記文字情報の種類数を所定数減少させ、当該第１の圧縮符号長よりも大きい第２の圧縮符号長の前記文字情報の種類数を前記所定数増加させることにより、前記上限長Ｎまでの各圧縮符号長の前記文字情報の種類数を更新する更新工程を前記コンピュータに実行させ、

前記算出工程は、

前記更新工程による更新後における前記上限長Ｎまでの各圧縮符号長の前記文字情報の種類数に基づいて、前記総和を再算出し、

前記判断工程は、

前記算出工程によって再算出された総和が１以下の最大値であるか否かを判断し、

前記生成工程は、

前記判断工程によって前記再算出された総和が１以下の最大値であると判断された場合、前記上限長Ｎまでの圧縮符号長と、前記上限長Ｎまでの圧縮符号長ごとの更新後における前記文字情報の種類数と、前記文字情報の出現率と、に基づいて、前記文字情報に圧縮符号を割り当てた葉の構造体を生成することを特徴とする請求項６に記載の生成プログラム。

[請求項9] 前記再算出された総和が１以下の最大値となるまで、前記更新工程による更新、前記算出工程による更新後における前記上限長Ｎまでの各圧縮符号長の前記文字情報の種類数に基づく前記総和の再算出、および、前記判断工程による再算出された総和が１以下の最大値であるか否かの判断を、前記コンピュータに繰り返し実行させることを特徴とする請求項７または８に記載の生成プログラム。

[請求項10] 対象ファイルに出現する文字情報群内の各文字情報の出現率に応じた生起確率で規定される圧縮符号長ごとに、前記文字情報の種類数を集計する集計手段と、

前記対象ファイルに出現する前記文字情報の総種類数に基づいて、最小圧縮符号長から最大圧縮符号長までの圧縮符号長群の中から、前記文字情報に割り当てられる圧縮符号長の上限長Ｎを決定する決定手段と、

前記集計手段によって集計された圧縮符号長ごとの前記文字情報の種類数のうち、前記決定手段によって決定された上限長Ｎを圧縮符号長とする前記文字情報の種類数を、前記上限長Ｎ以上の圧縮符号長での前記文字情報の種類数の総和に補正する補正手段と、

前記補正手段によって補正された補正後の前記圧縮符号長ごとの前記文字情報の種類数に基づいて、前記上限長Ｎを最大枝数とし、前記文字情報の各々の生起確率に応じた圧縮符号長の圧縮符号を葉とする2^N分枝無節点ハフマン木を構築する構築手段と、

を備えることを特徴とする生成装置。

[請求項11] 対象ファイルを記憶する記憶装置を有するコンピュータが、

集計手段により、前記対象ファイルに出現する文字情報群内の各文字情報の出現率に応じた生起確率で規定される圧縮符号長ごとに、前記文字情報の種類数を集計する集計工程と、

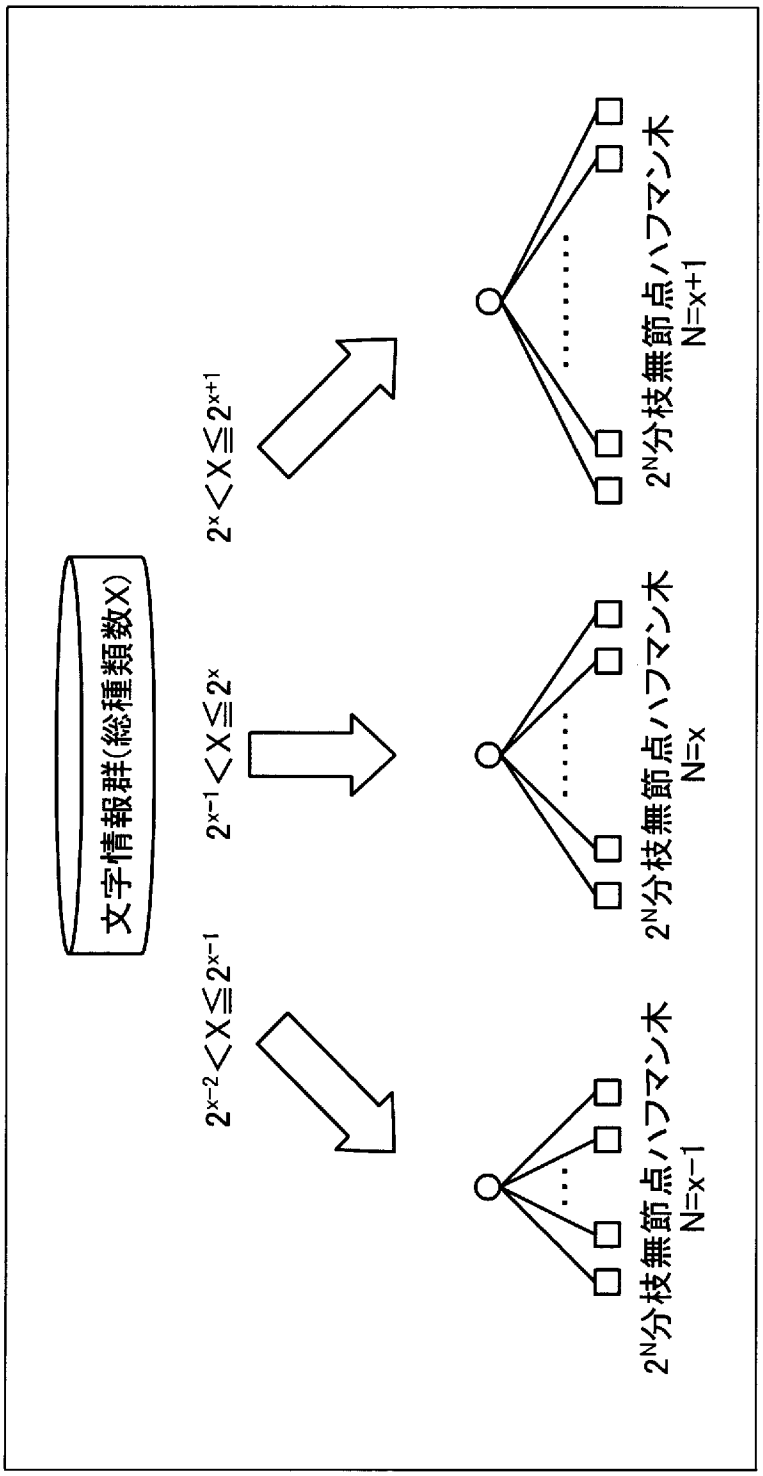
決定手段により、前記対象ファイルに出現する前記文字情報の総種類数に基づいて、最小圧縮符号長から最大圧縮符号長までの圧縮符号長群の中から、前記文字情報に割り当てられる圧縮符号長の上限長Ｎを決定する決定工程と、

補正手段により、前記集計工程によって集計された圧縮符号長ごとの前記文字情報の種類数のうち、前記決定工程によって決定された上限長Ｎを圧縮符号長とする前記文字情報の種類数を、前記上限長Ｎ以上の圧縮符号長での前記文字情報の種類数の総和に補正する補正工程と、

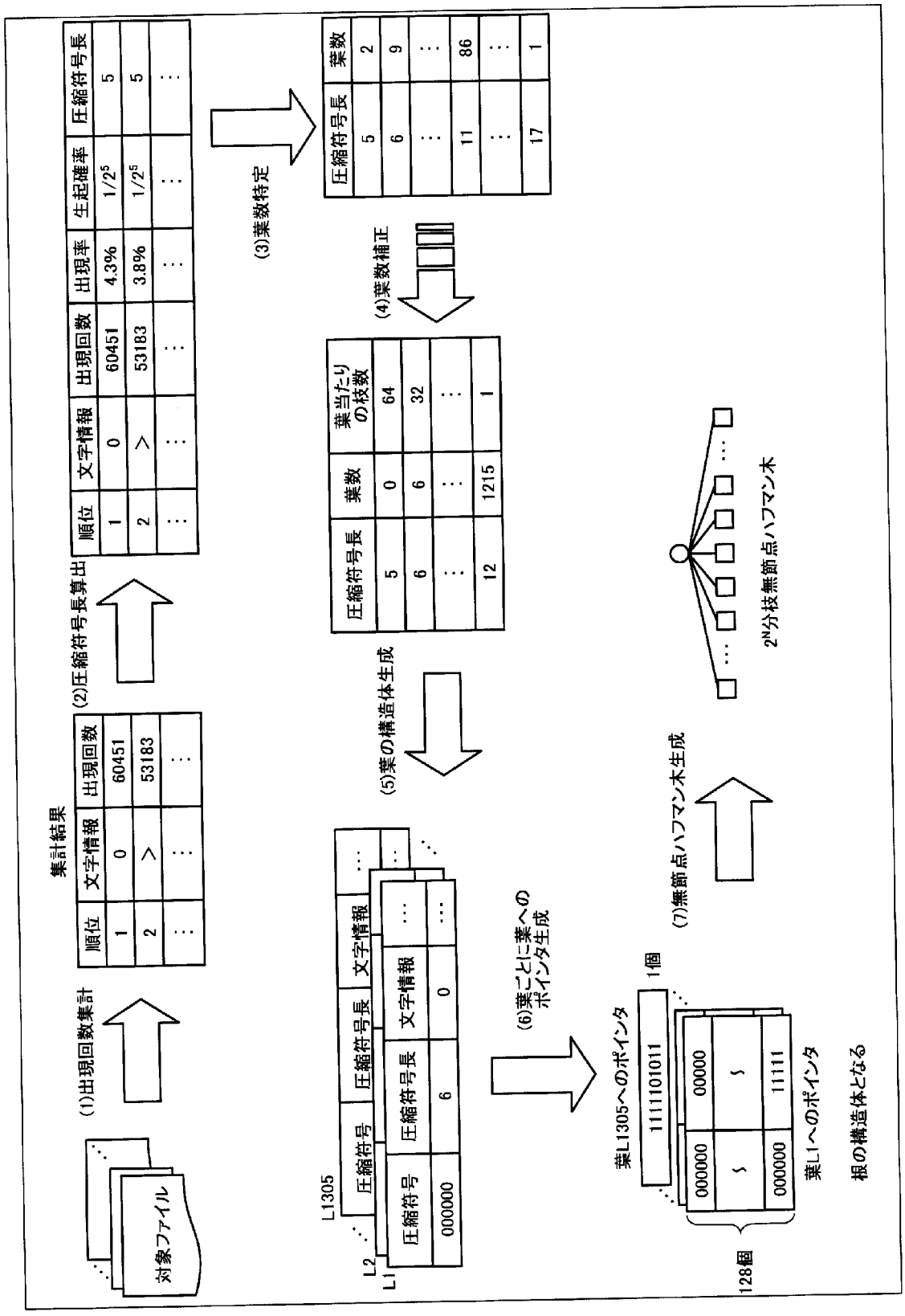
構築手段により、前記補正工程によって補正された補正後の前記圧縮符号長ごとの前記文字情報の種類数に基づいて、前記上限長Ｎを最大枝数とし、前記文字情報の各々の生起確率に応じた圧縮符号長の圧縮符号を葉とする２Ｎ分枝無節点ハフマン木を構築する構築工程と、

を実行することを特徴とする生成方法。

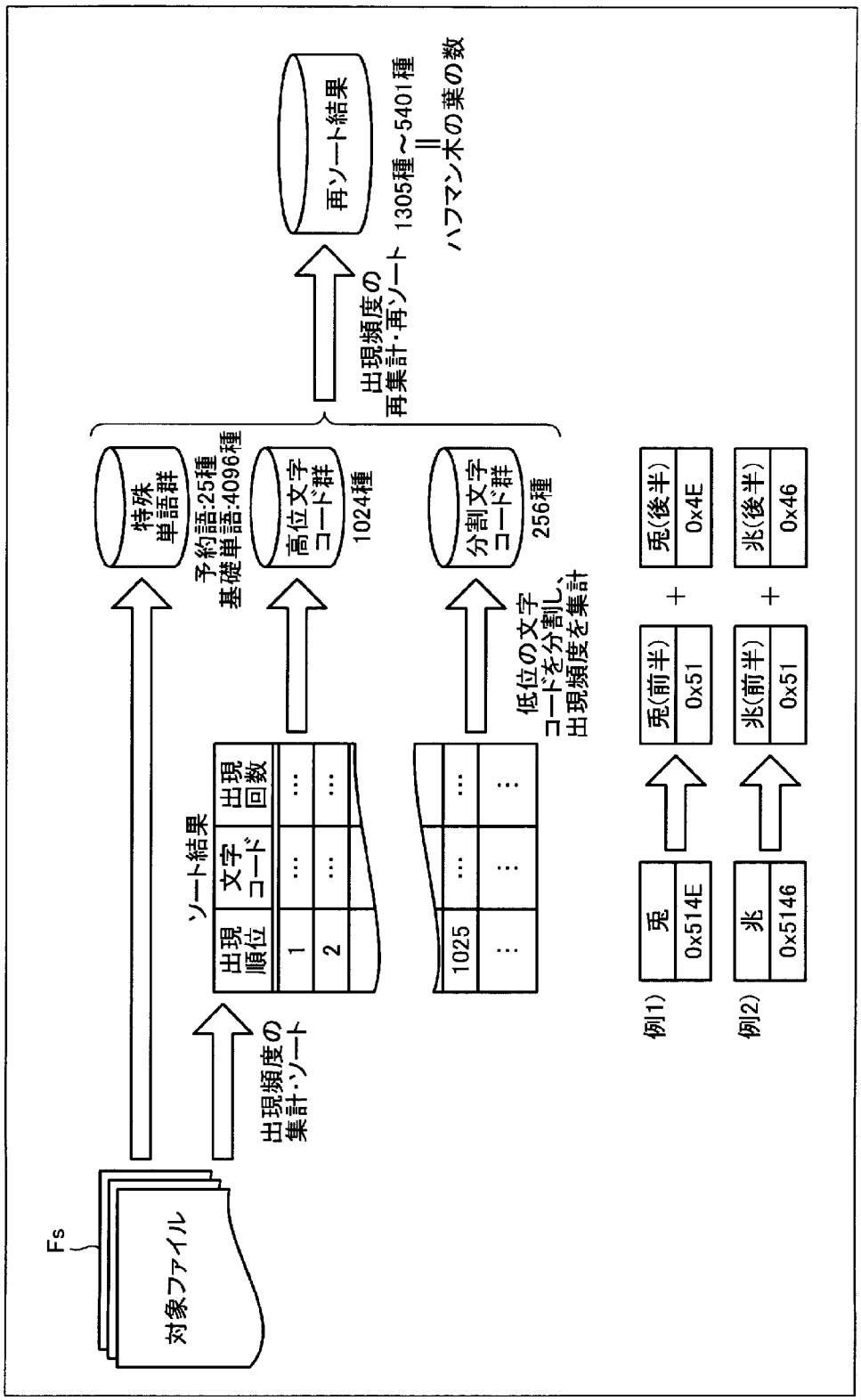
[図1]


文字情報群(総種類数X)
$2^{x-2}<X\leqq 2^{x-1}$
$2^{x-1}<X\leqq 2^{x}$
$2^{x}<X\leqq 2^{x+1}$
2^N分枝無節点ハフマン木
N=x-1
2^N分枝無節点ハフマン木
N=x
2^N分枝無節点ハフマン木
N=x+1

[図2]


対象ファイル
(1)出現回数集計
集計結果
順位 文字情報 出現回数
1 0 60451
2 > 53183
(2)圧縮符号長算出
順位 文字情報 出現回数 出現率 生起確率 圧縮符号長
1 0 60451 4.3% 1/2^5 5
2 > 53183 3.8% 1/2^5 5
(3)葉数特定
圧縮符号長 葉数
5 2
6 9
11 86
17 1
(4)葉数補正
圧縮符号長 葉数 葉当たりの枝数
5 0 64
6 6 32
12 1215 1
(5)葉の構造体生成
L1305
L2
L1
圧縮符号 圧縮符号長 文字情報
000000 6 0
(6)葉ごとに葉へのポインタ生成
葉L1305へのポインタ
1111101011
1個
128個
000000 00000
000000 11111
葉L1へのポインタ
(7)無節点ハフマン木生成
2^N分枝無節点ハフマン木
根の構造体となる

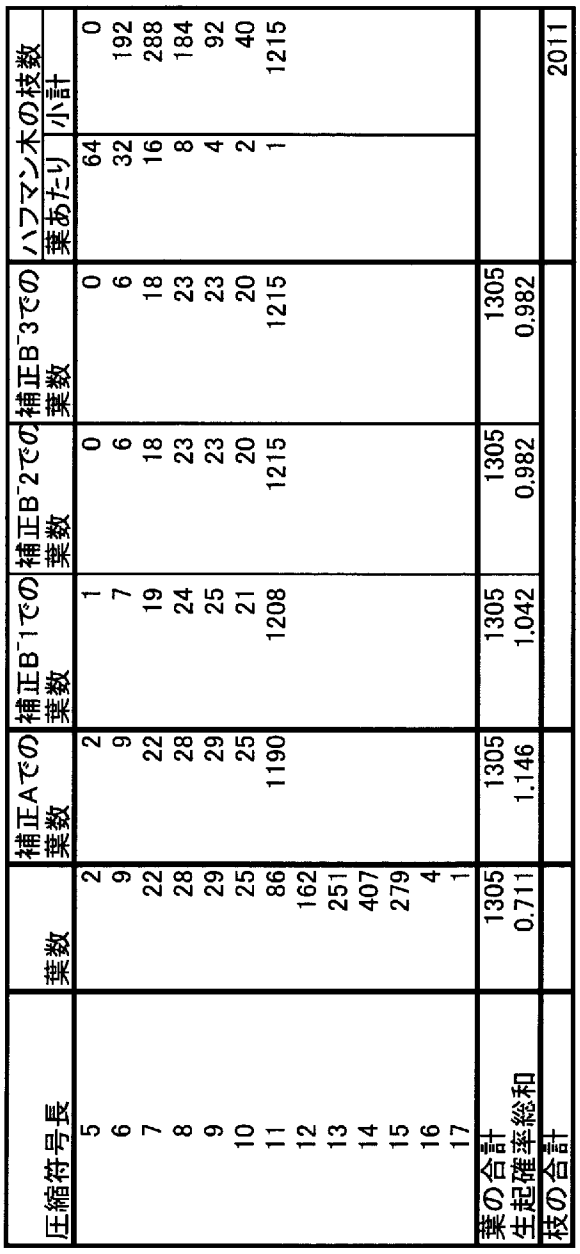
[図3]


Fs
対象ファイル
出現頻度の
集計・ソート
ソート結果
出現順位
文字コード
出現回数
1
2
1025
出現頻度の
再集計・再ソート
再ソート結果
1305種～5401種
=
ハフマン木の葉の数
特殊単語群
予約語:25種
基礎単語:4096種
高位文字コード群
1024種
分割文字コード群
256種
低位の文字
コードを分割し、
出現頻度を集計
例1)
兎
0x514E
兎(前半)
0x51
兎(後半)
0x4E
例2)
兆
0x5146
兆(前半)
0x51
兆(後半)
0x46

[図4]

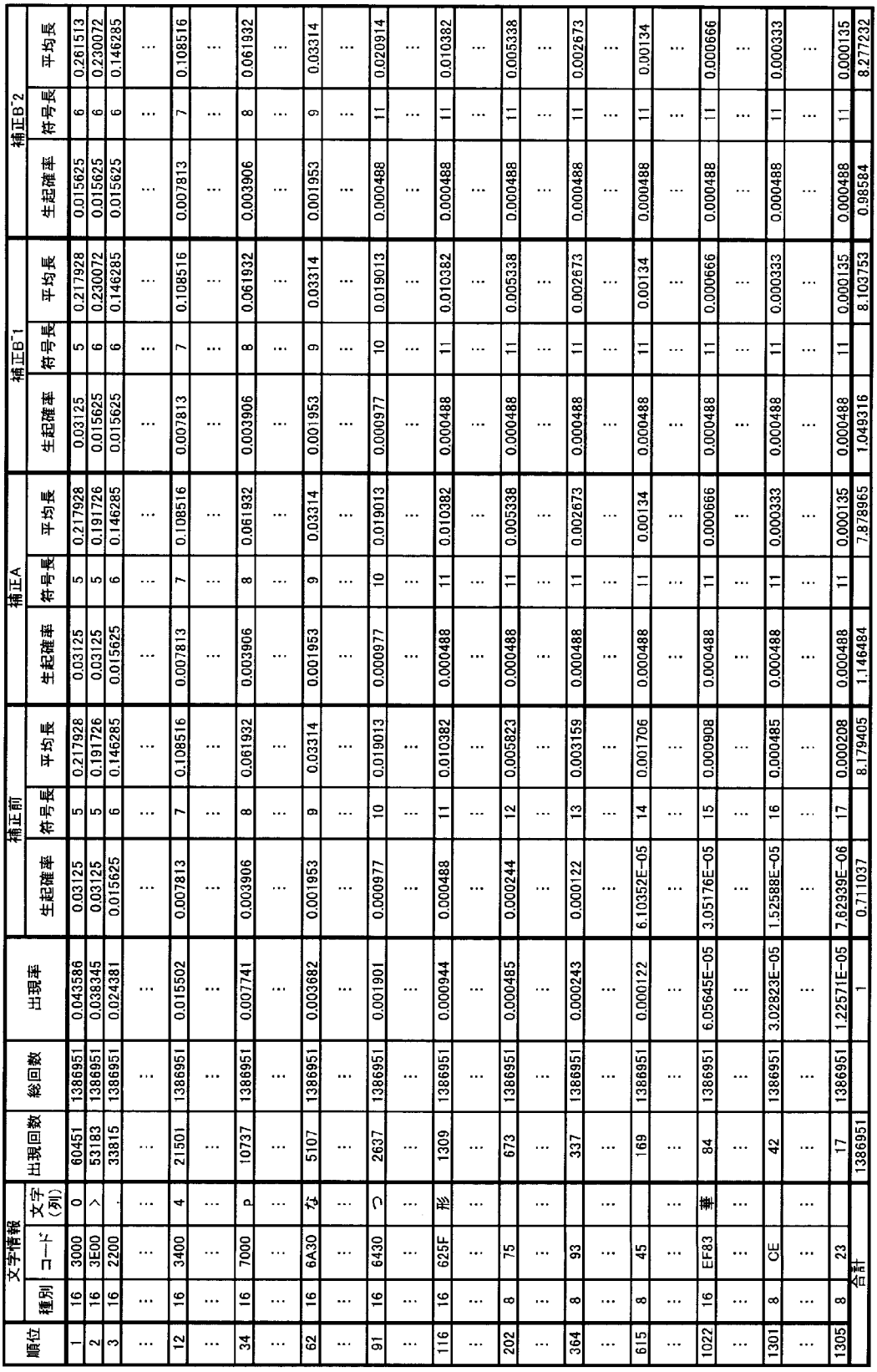
順位	文字情報			出現回数	総回数	出現率	補正前	
	種別	コード	文字(列)				生起確率	圧縮符号長
1	16	0x3000	0	60451	1386951	0.043585534	0.03125	5
2	16	0x3E00	>	53183	1386951	0.038345262	0.03125	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
6	特		 	28999	1386951	0.020908453	0.015625	6
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
24	16	6.00E+30	の	15577	1386951	0.011231111	0.0078125	7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
34	16	0x7000	p	10737	1386951	0.007741441	0.00390625	8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
79	16	6F30	は	3529	1386951	0.00254443	0.001953125	9
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
110	16	6972	物	1569	1386951	0.001131258	0.000976563	10
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
156	8	4E		879	1386951	0.000633764	0.000488281	11
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
167	8	51		835	1386951	0.00060204	0.000488281	11
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
286	16	D552	動	485	1386951	0.000349688	0.000244141	12
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
364	8	0x93		337	1386951	0.000242979	0.00012207	13
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
691	16	216B	次	145	1386951	0.000104546	6.10352E-05	14
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1022	16	0xEF83	華	84	1386951	6.05645E-05	3.05176E-05	15
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1301	8	0xCE		42	1386951	3.02823E-05	1.52588E-05	16
1302	8	0xFD		35	1386951	2.52352E-05	1.52588E-05	16
1303	8	0xEA		33	1386951	2.37932E-05	1.52588E-05	16
1304	8	0x24		23	1386951	1.65831E-05	1.52588E-05	16
1305	8	0x23		17	1386951	1.22571E-05	7.62939E-06	17
						生起確率総和	0.711037	

[図5]



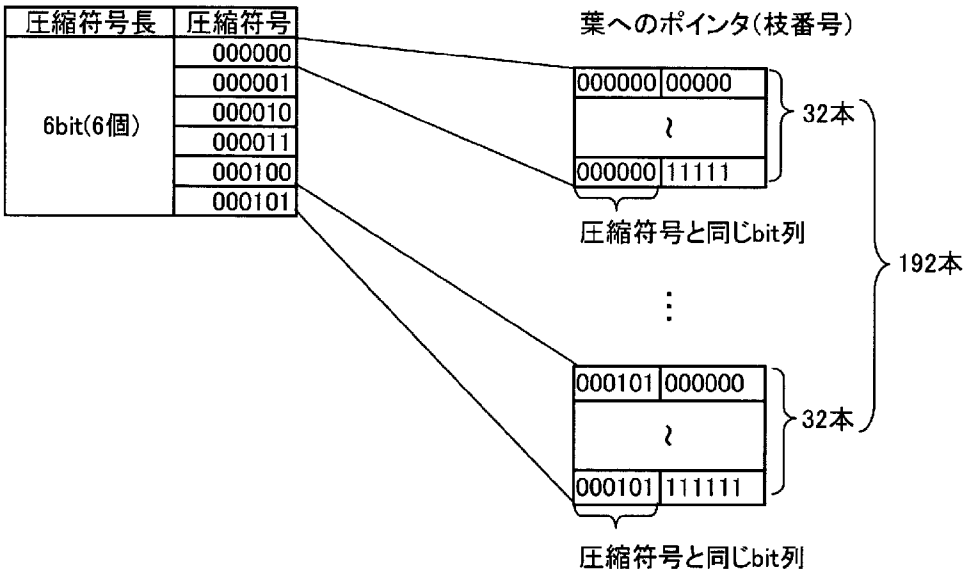
圧縮符号長	葉数	補正Aでの葉数	補正B-1での葉数	補正B-2での葉数	補正B-3での葉数	ハフマン木の枝数 葉あたり	小計
5	2	2	1	0	0	64	0
6	9	9	7	6	6	32	192
7	22	22	19	18	18	16	288
8	28	28	24	23	23	8	184
9	29	29	25	23	23	4	92
10	25	25	21	20	20	2	40
11	86	1190	1208	1215	1215	1	1215
12	162						
13	251						
14	407						
15	279						
16	4						
17	1						
葉の合計	1305	1305	1305	1305	1305		
生起確率総和	0.711	1.146	1.042	0.982	0.982		
枝の合計							2011

[図6]

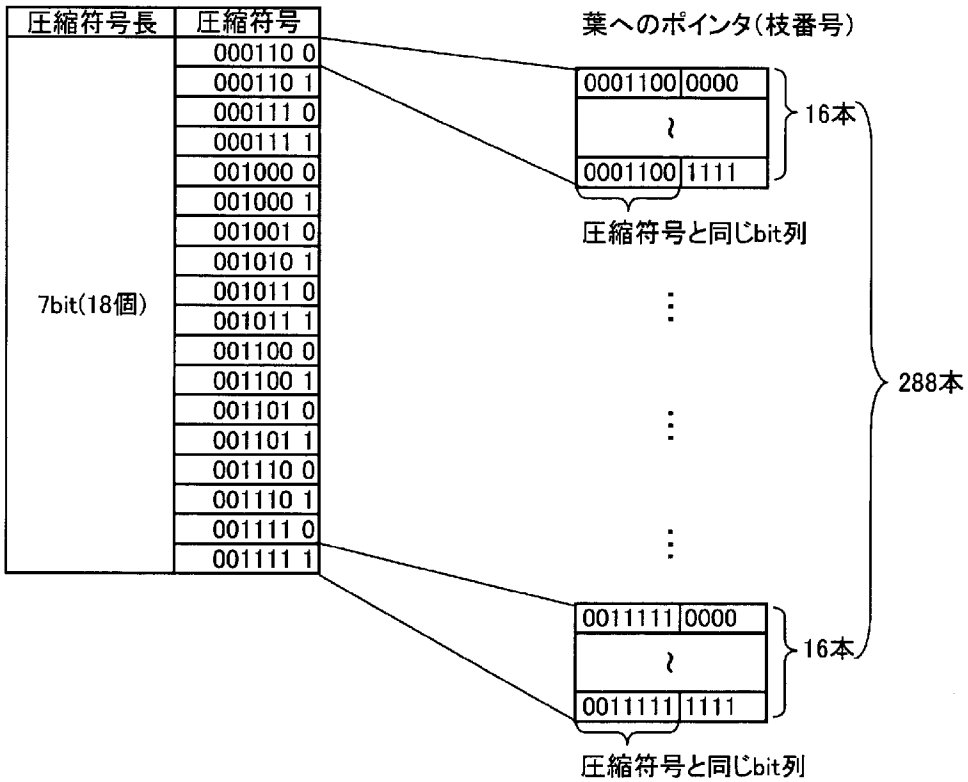


順位	文字情報			出現回数	総回数	出現率	補正前			補正A			補正B'1			補正B'2		
	種別	コード	文字(列)				生起確率	符号長	平均長	生起確率	符号長	平均長	生起確率	符号長	平均長	生起確率	符号長	平均長
1	16	3000	0	60451	1386951	0.043586	0.03125	5	0.217928	0.03125	5	0.217928	0.03125	5	0.217928	0.015625	6	0.261513
2	16	3E00	>	53183	1386951	0.038345	0.03125	5	0.191726	0.03125	5	0.191726	0.015625	6	0.230072	0.015625	6	0.230072
3	16	2200	、	33815	1386951	0.024381	0.015625	6	0.146285	0.015625	6	0.146285	0.015625	6	0.146285	0.015625	6	0.146285
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
12	16	3400	4	21501	1386951	0.015502	0.007813	7	0.108516	0.007813	7	0.108516	0.007813	7	0.108516	0.007813	7	0.108516
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
34	16	7000	p	10737	1386951	0.007741	0.003906	8	0.061932	0.003906	8	0.061932	0.003906	8	0.061932	0.003906	8	0.061932
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
62	16	6A30	々	5107	1386951	0.003682	0.001953	9	0.03314	0.001953	9	0.03314	0.001953	9	0.03314	0.001953	9	0.03314
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
91	16	6430	つ	2637	1386951	0.001901	0.000977	10	0.019013	0.000977	10	0.019013	0.000977	10	0.019013	0.000488	11	0.020914
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
116	16	625F	形	1309	1386951	0.000944	0.000488	11	0.010382	0.000488	11	0.010382	0.000488	11	0.010382	0.000488	11	0.010382
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
202	8	75		673	1386951	0.000485	0.000244	12	0.005823	0.000488	11	0.005338	0.000488	11	0.005338	0.000488	11	0.005338
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
364	8	93		337	1386951	0.000243	0.000122	13	0.003159	0.000488	11	0.002673	0.000488	11	0.002673	0.000488	11	0.002673
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
615	8	45		169	1386951	0.000122	6.10352E-05	14	0.001706	0.000488	11	0.00134	0.000488	11	0.00134	0.000488	11	0.00134
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1022	16	EF83	華	84	1386951	6.05645E-05	3.05176E-05	15	0.000908	0.000488	11	0.000666	0.000488	11	0.000666	0.000488	11	0.000666
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1301	8	CE		42	1386951	3.02823E-05	1.52588E-05	16	0.000485	0.000488	11	0.000333	0.000488	11	0.000333	0.000488	11	0.000333
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1305	8	23		17	1386951	1.22571E-05	7.62939E-06	17	0.000208	0.000488	11	0.000135	0.000488	11	0.000135	0.000488	11	0.000135
合計				1386951		1	0.711037		8.179405	1.146484		7.879965	1.049316		8.103753	0.98584		8.277232

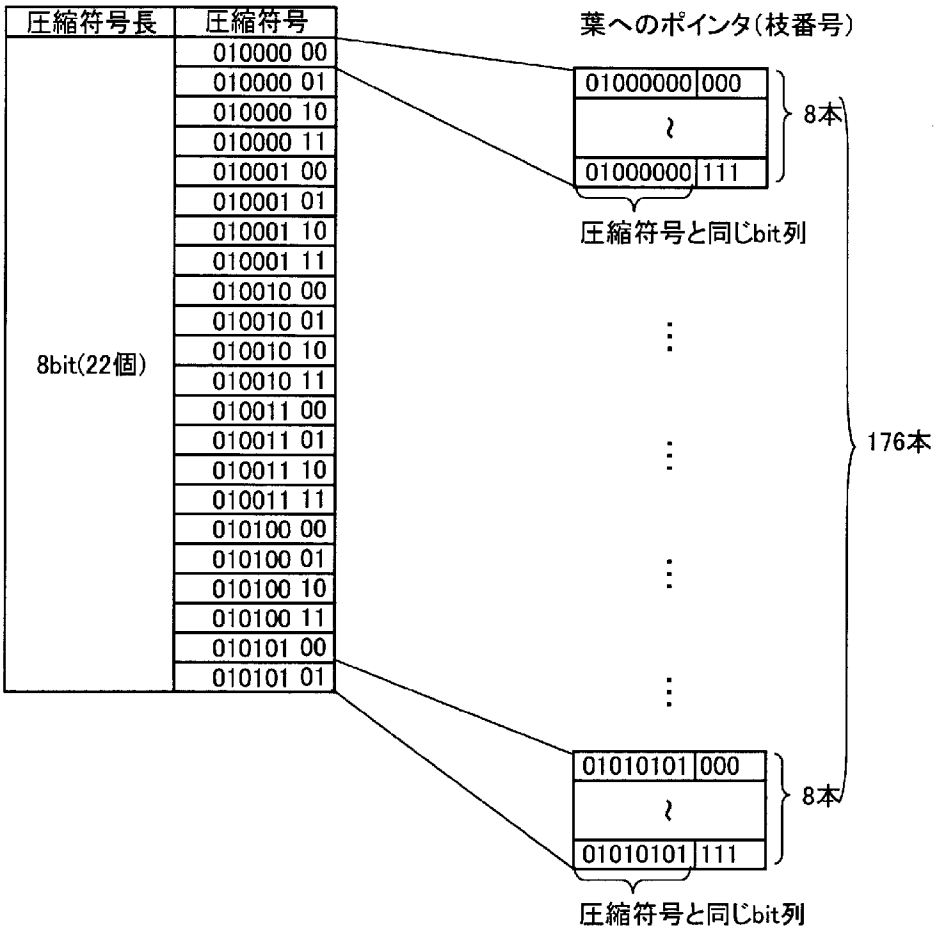
[図7]


圧縮符号長
圧縮符号
6bit(6個)
000000
000001
000010
000011
000100
000101
葉へのポインタ(枝番号)
000000 00000
～
000000 11111
32本
圧縮符号と同じbit列
192本
000101 000000
～
000101 111111
32本
圧縮符号と同じbit列

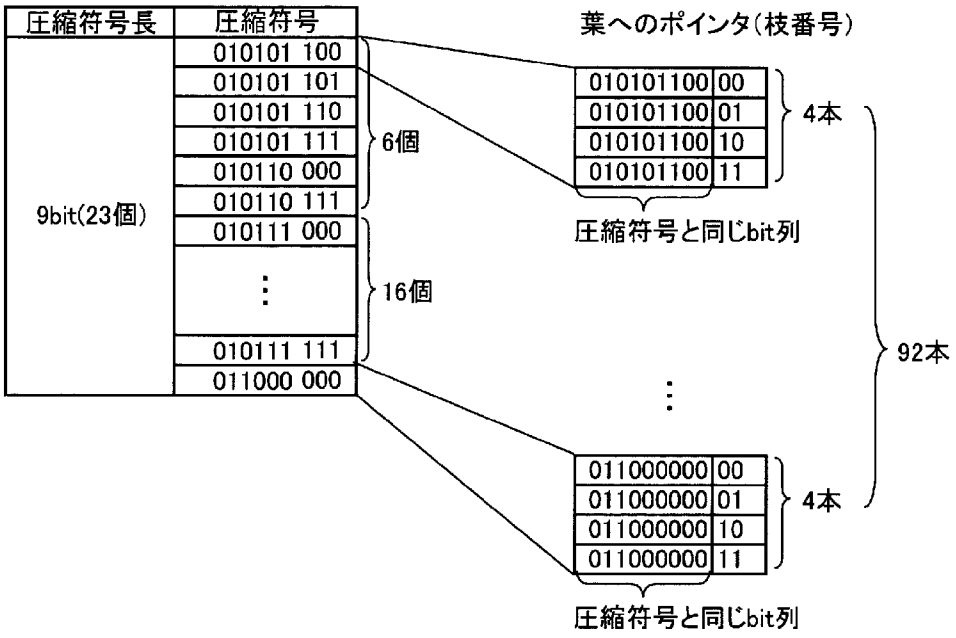
[図8]


圧縮符号長
圧縮符号
7bit(18個)
000110 0
000110 1
000111 0
000111 1
001000 0
001000 1
001001 0
001010 1
001011 0
001011 1
001100 0
001100 1
001101 0
001101 1
001110 0
001110 1
001111 0
001111 1
葉へのポインタ(枝番号)
0001100 0000
～
0001100 1111
16本
圧縮符号と同じbit列
288本
0011111 0000
～
0011111 1111
16本
圧縮符号と同じbit列

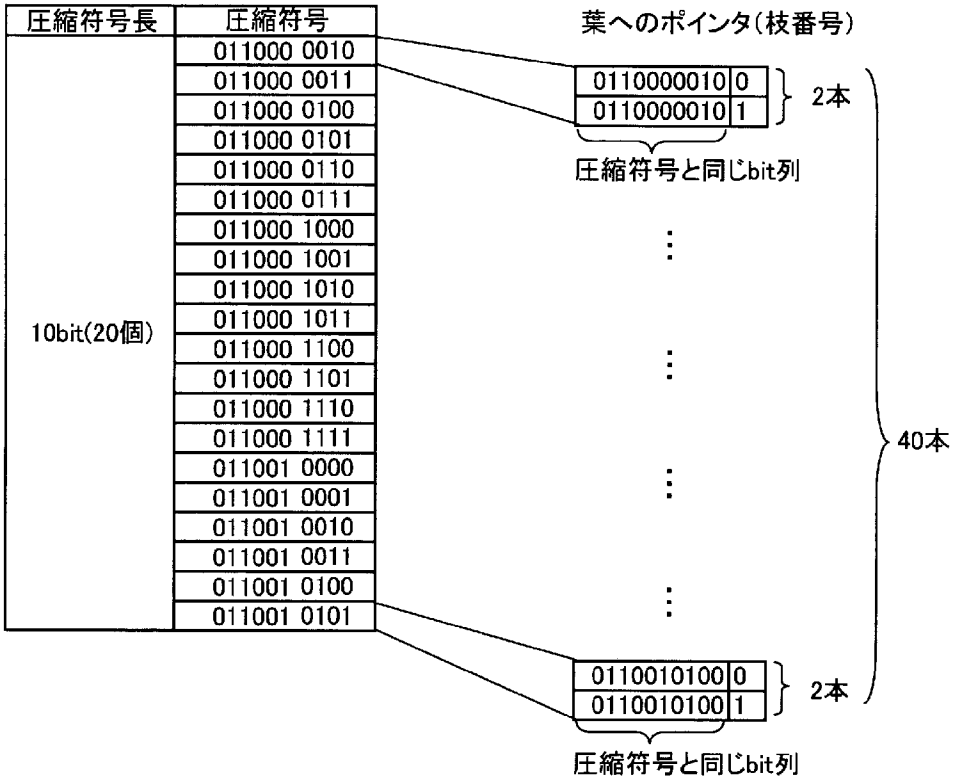
[図9]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
8bit(22個)
010000 00
010000 01
010000 10
010000 11
010001 00
010001 01
010001 10
010001 11
010010 00
010010 01
010010 10
010010 11
010011 00
010011 01
010011 10
010011 11
010100 00
010100 01
010100 10
010100 11
010101 00
010101 01
01000000 000
01000000 111
8本
圧縮符号と同じbit列
176本
01010101 000
01010101 111
8本
圧縮符号と同じbit列

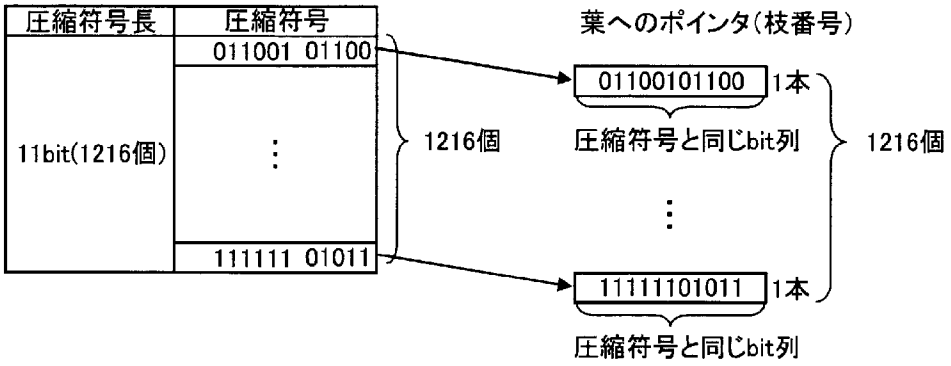
[図10]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
9bit(23個)
010101 100
010101 101
010101 110
010101 111
010110 000
010110 111
6個
010111 000
010111 111
16個
011000 000
010101100 00
010101100 01
010101100 10
010101100 11
4本
圧縮符号と同じbit列
92本
011000000 00
011000000 01
011000000 10
011000000 11
4本
圧縮符号と同じbit列

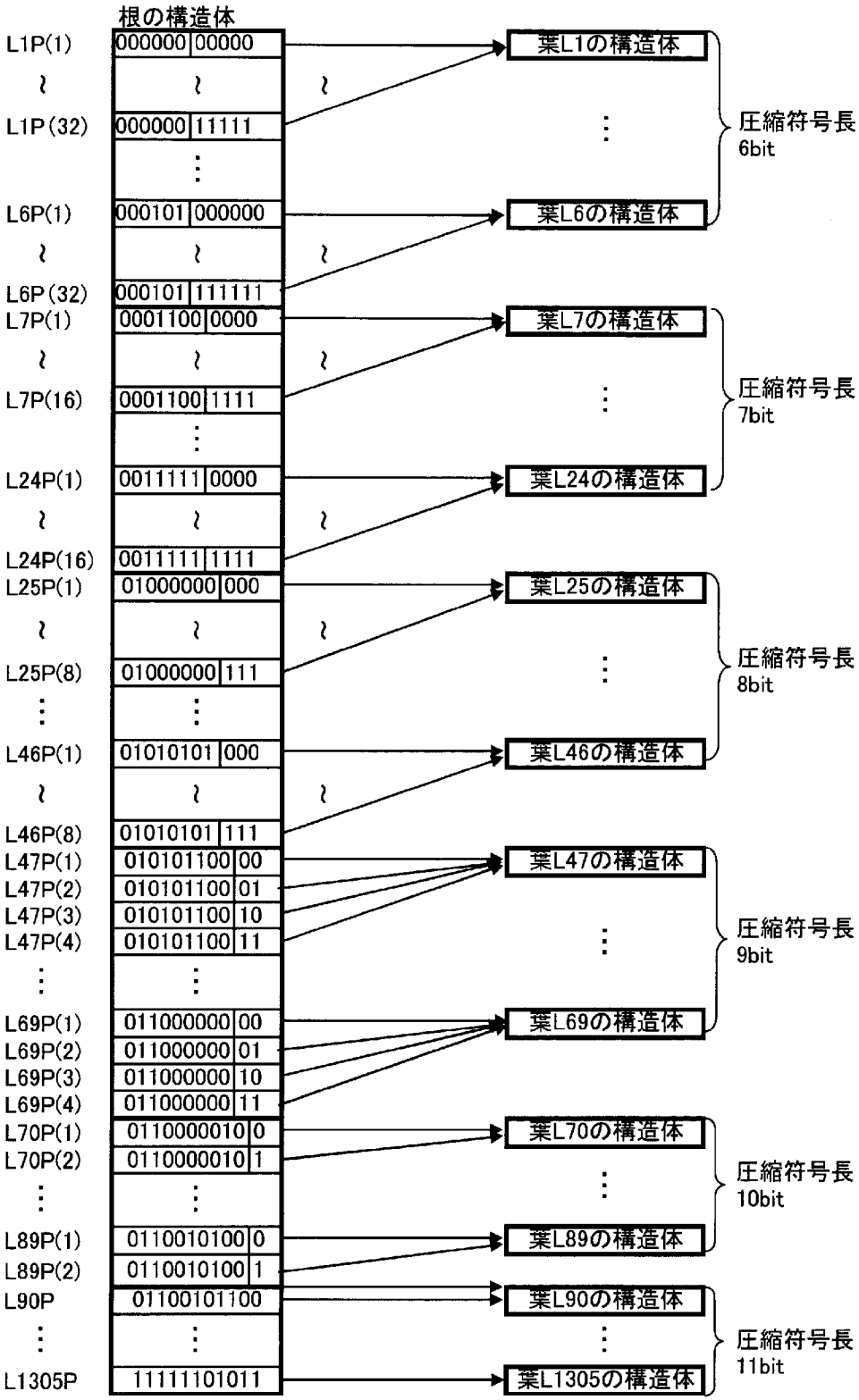
[図11]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
011000 0010
011000 0011
011000 0100
011000 0101
011000 0110
011000 0111
011000 1000
011000 1001
011000 1010
011000 1011
011000 1100
011000 1101
011000 1110
011000 1111
011001 0000
011001 0001
011001 0010
011001 0011
011001 0100
011001 0101
10bit(20個)
0110000010 0
0110000010 1
2本
圧縮符号と同じbit列
40本
0110010100 0
0110010100 1
2本
圧縮符号と同じbit列

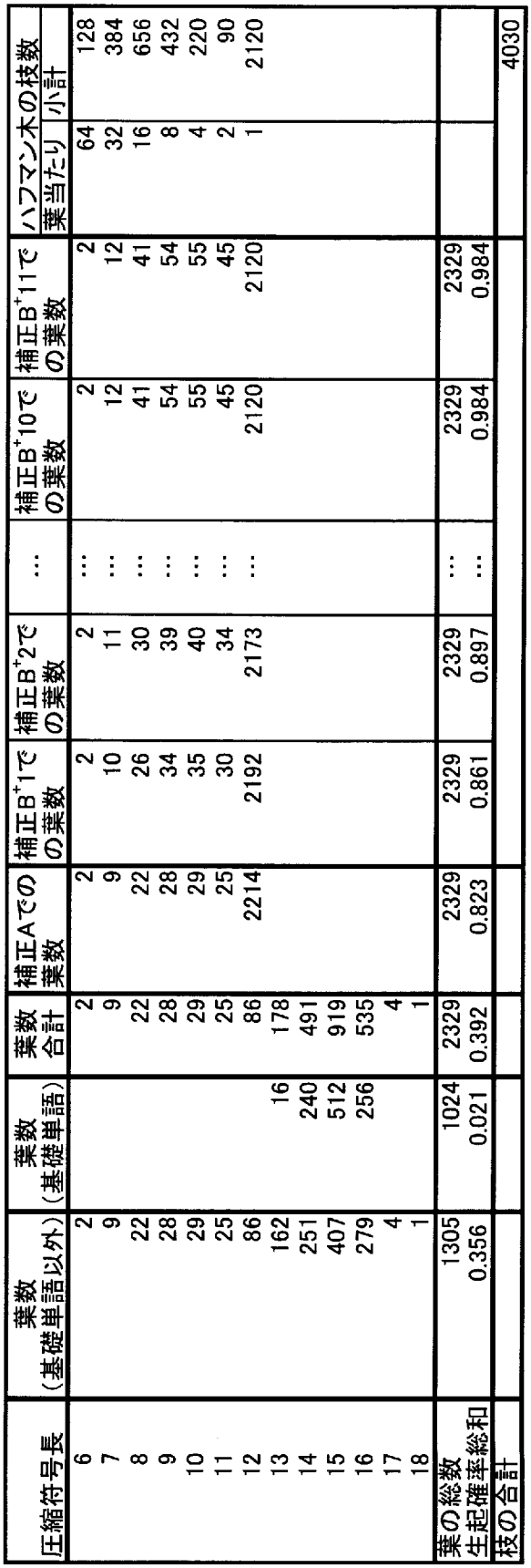
[図12]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
011001 01100
11bit(1216個)
1216個
01100101100
1本
圧縮符号と同じbit列
1216個
111111 01011
11111101011
1本
圧縮符号と同じbit列

[図13]

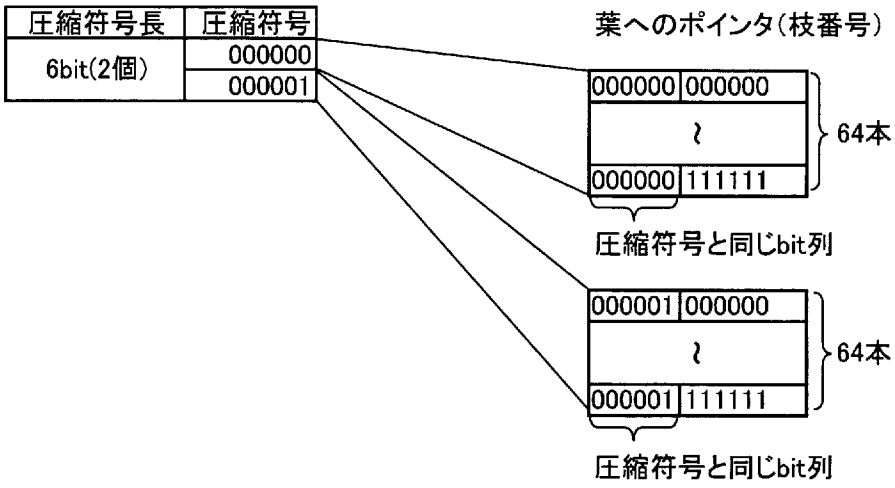

根の構造体
L1P(1)
000000 00000
L1P(32)
000000 11111
L6P(1)
000101 000000
L6P(32)
000101 111111
L7P(1)
0001100 0000
L7P(16)
0001100 1111
L24P(1)
0011111 0000
L24P(16)
0011111 1111
L25P(1)
01000000 000
L25P(8)
01000000 111
L46P(1)
01010101 000
L46P(8)
01010101 111
L47P(1)
010101100 00
L47P(2)
010101100 01
L47P(3)
010101100 10
L47P(4)
010101100 11
L69P(1)
011000000 00
L69P(2)
011000000 01
L69P(3)
011000000 10
L69P(4)
011000000 11
L70P(1)
0110000010 0
L70P(2)
0110000010 1
L89P(1)
0110010100 0
L89P(2)
0110010100 1
L90P
01100101100
L1305P
11111101011
葉L1の構造体
葉L6の構造体
葉L7の構造体
葉L24の構造体
葉L25の構造体
葉L46の構造体
葉L47の構造体
葉L69の構造体
葉L70の構造体
葉L89の構造体
葉L90の構造体
葉L1305の構造体
圧縮符号長
6bit
圧縮符号長
7bit
圧縮符号長
8bit
圧縮符号長
9bit
圧縮符号長
10bit
圧縮符号長
11bit

[図14]

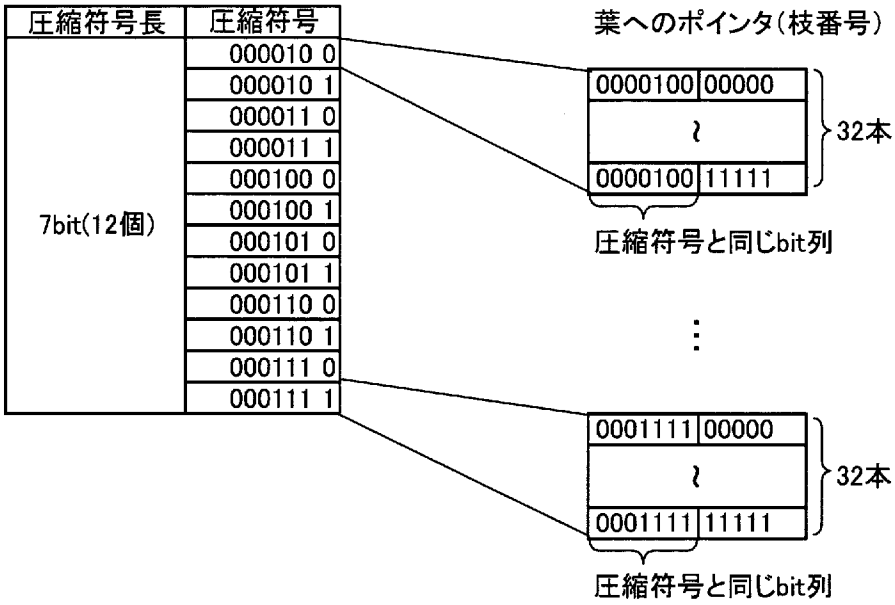


圧縮符号長	葉数（基礎単語以外）	葉数（基礎単語）	葉数合計	補正Aでの葉数	補正B+1での葉数	補正B+2での葉数	…	補正B+10での葉数	補正B+11での葉数	ハフマン木の枝数 葉当たり	ハフマン木の枝数 小計
6	2		2	2	2	2	…	2	2	64	128
7	9		9	9	10	11	…	12	12	32	384
8	22		22	22	26	30	…	41	41	16	656
9	28		28	28	34	39	…	54	54	8	432
10	29		29	29	35	40	…	55	55	4	220
11	25		25	25	30	34	…	45	45	2	90
12	86		86	2214	2192	2173	…	2120	2120	1	2120
13	162	16	178								
14	251	240	491								
15	407	512	919								
16	279	256	535								
17	4		4								
18	1		1								
葉の総数	1305	1024	2329	2329	2329	2329	…	2329	2329		
生起確率総和	0.356	0.021	0.392	0.823	0.861	0.897	…	0.984	0.984		
枝の合計											4030

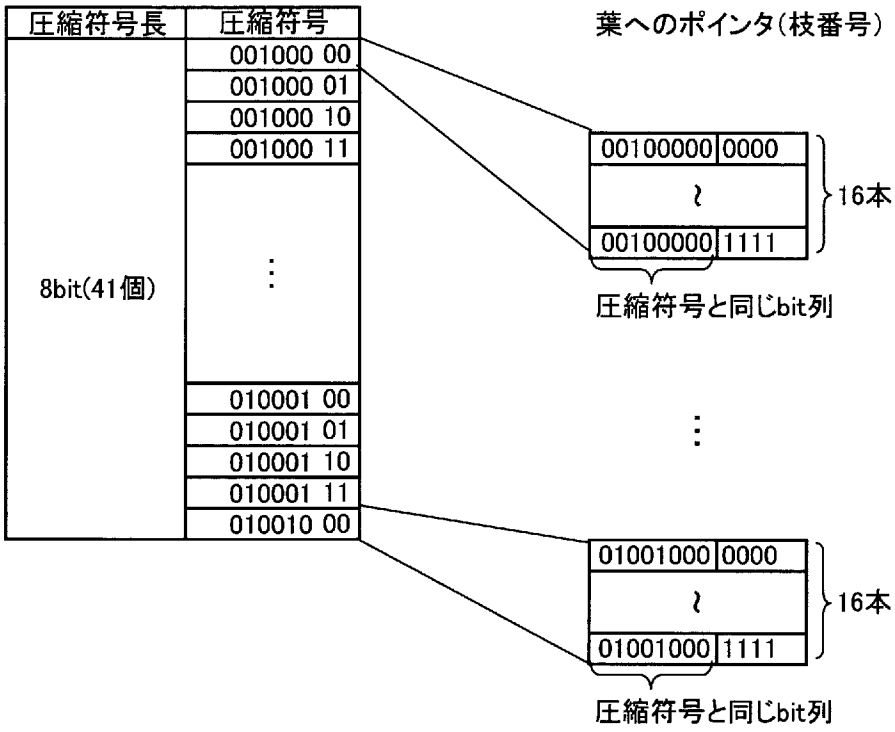
[図15]


圧縮符号長
圧縮符号
6bit(2個)
000000
000001
葉へのポインタ(枝番号)
000000 000000
~
000000 111111
64本
圧縮符号と同じbit列
000001 000000
~
000001 111111
64本
圧縮符号と同じbit列

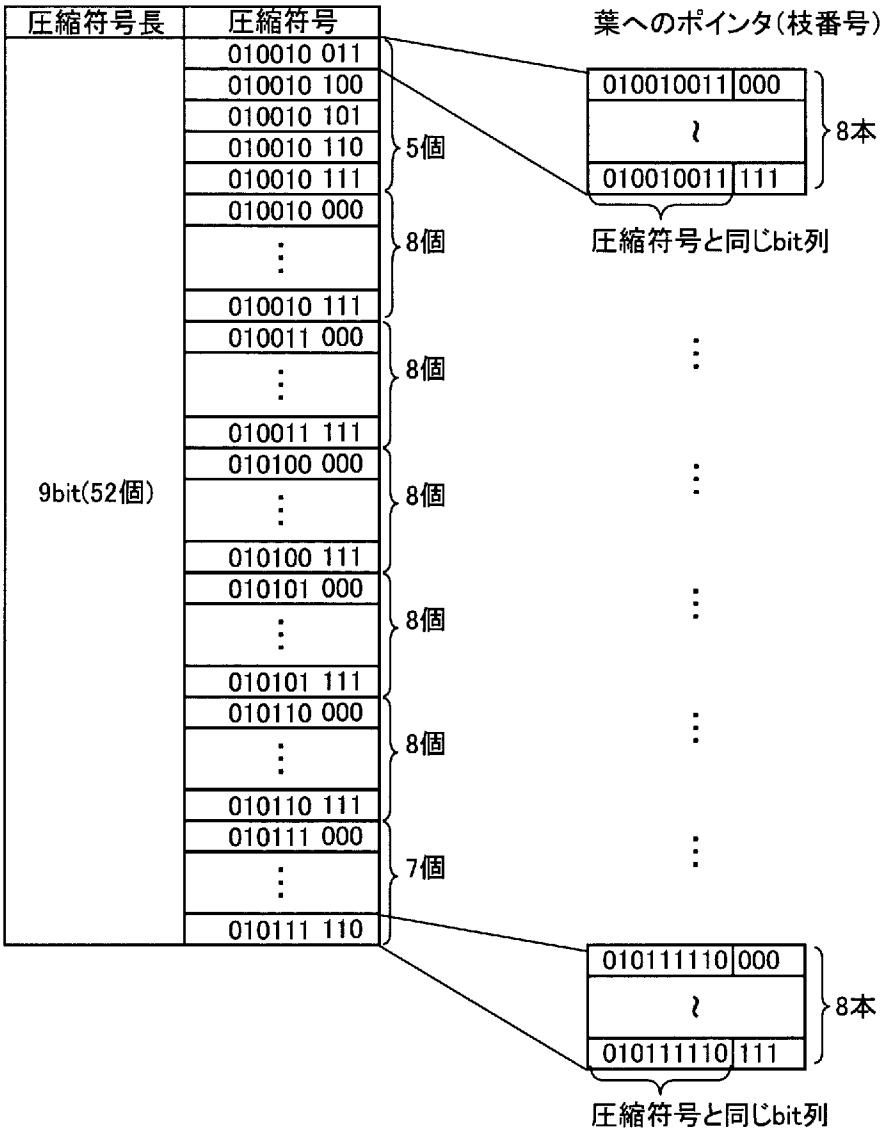
[図16]


圧縮符号長
圧縮符号
7bit(12個)
000010 0
000010 1
000011 0
000011 1
000100 0
000100 1
000101 0
000101 1
000110 0
000110 1
000111 0
000111 1
葉へのポインタ(枝番号)
0000100 00000
~
0000100 11111
32本
圧縮符号と同じbit列
0001111 00000
~
0001111 11111
32本
圧縮符号と同じbit列

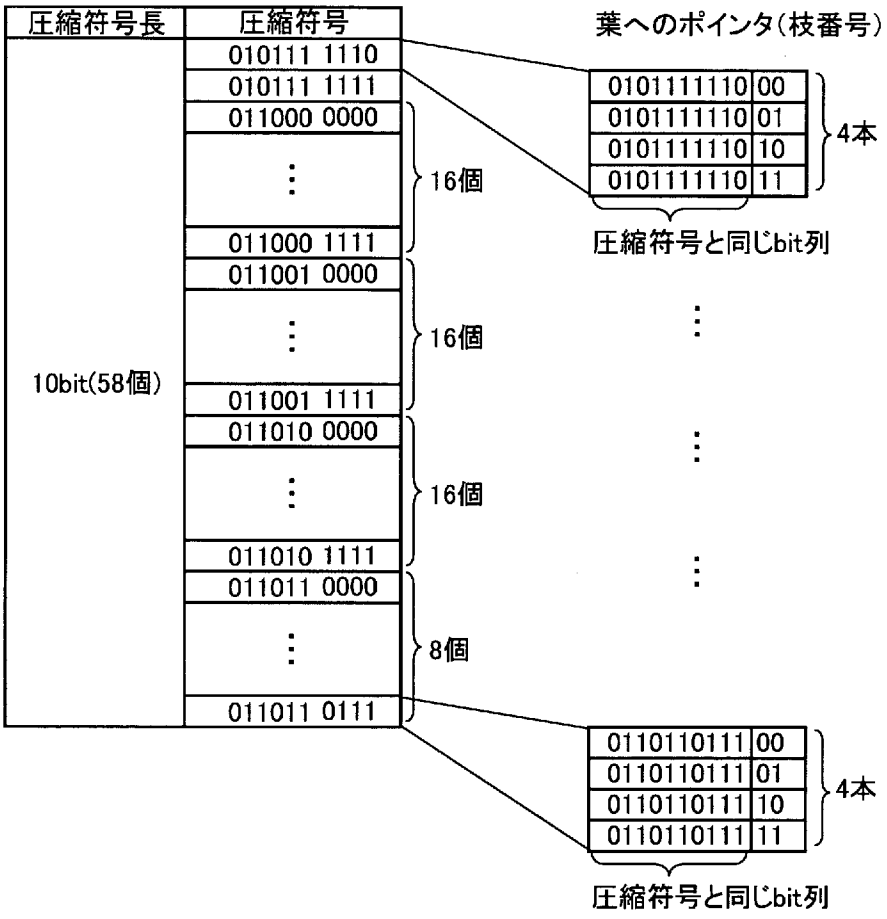
[図17]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
001000 00
001000 01
001000 10
001000 11
8bit(41個)
010001 00
010001 01
010001 10
010001 11
010010 00
00100000 0000
00100000 1111
16本
圧縮符号と同じbit列
01001000 0000
01001000 1111
16本
圧縮符号と同じbit列

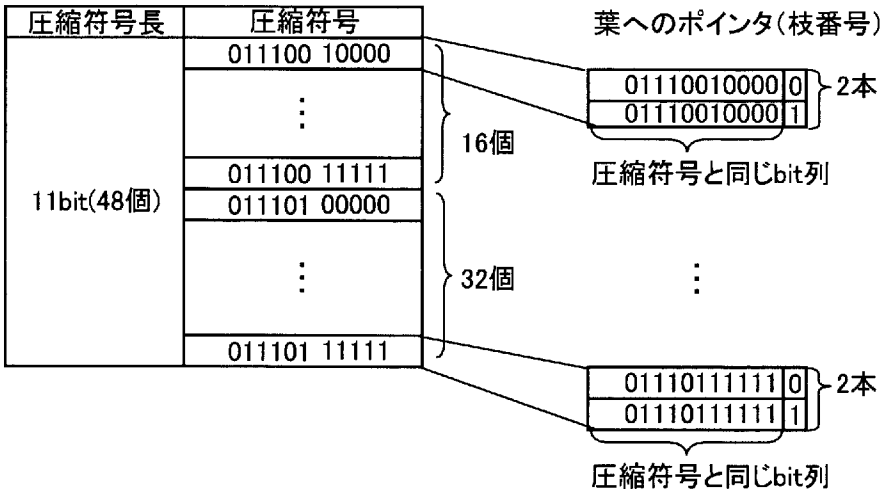
[図18]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
9bit(52個)
010010 011
010010 100
010010 101
010010 110
010010 111
010010 000
010010 111
010011 000
010011 111
010100 000
010100 111
010101 000
010101 111
010110 000
010110 111
010111 000
010111 110
5個
8個
8個
8個
8個
8個
7個
010010011 000
010010011 111
8本
圧縮符号と同じbit列
010111110 000
010111110 111
8本
圧縮符号と同じbit列

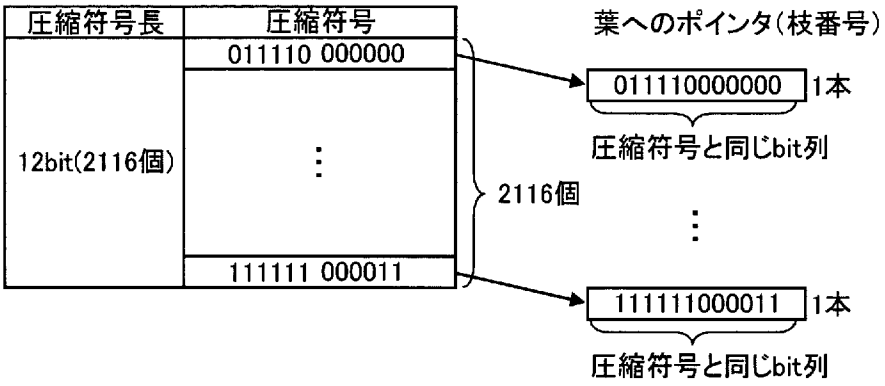
[図19]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
010111 1110
010111 1111
011000 0000
011000 1111
16個
011001 0000
011001 1111
16個
10bit(58個)
011010 0000
011010 1111
16個
011011 0000
011011 0111
8個
0101111110 00
0101111110 01
0101111110 10
0101111110 11
4本
圧縮符号と同じbit列
0110110111 00
0110110111 01
0110110111 10
0110110111 11
4本
圧縮符号と同じbit列

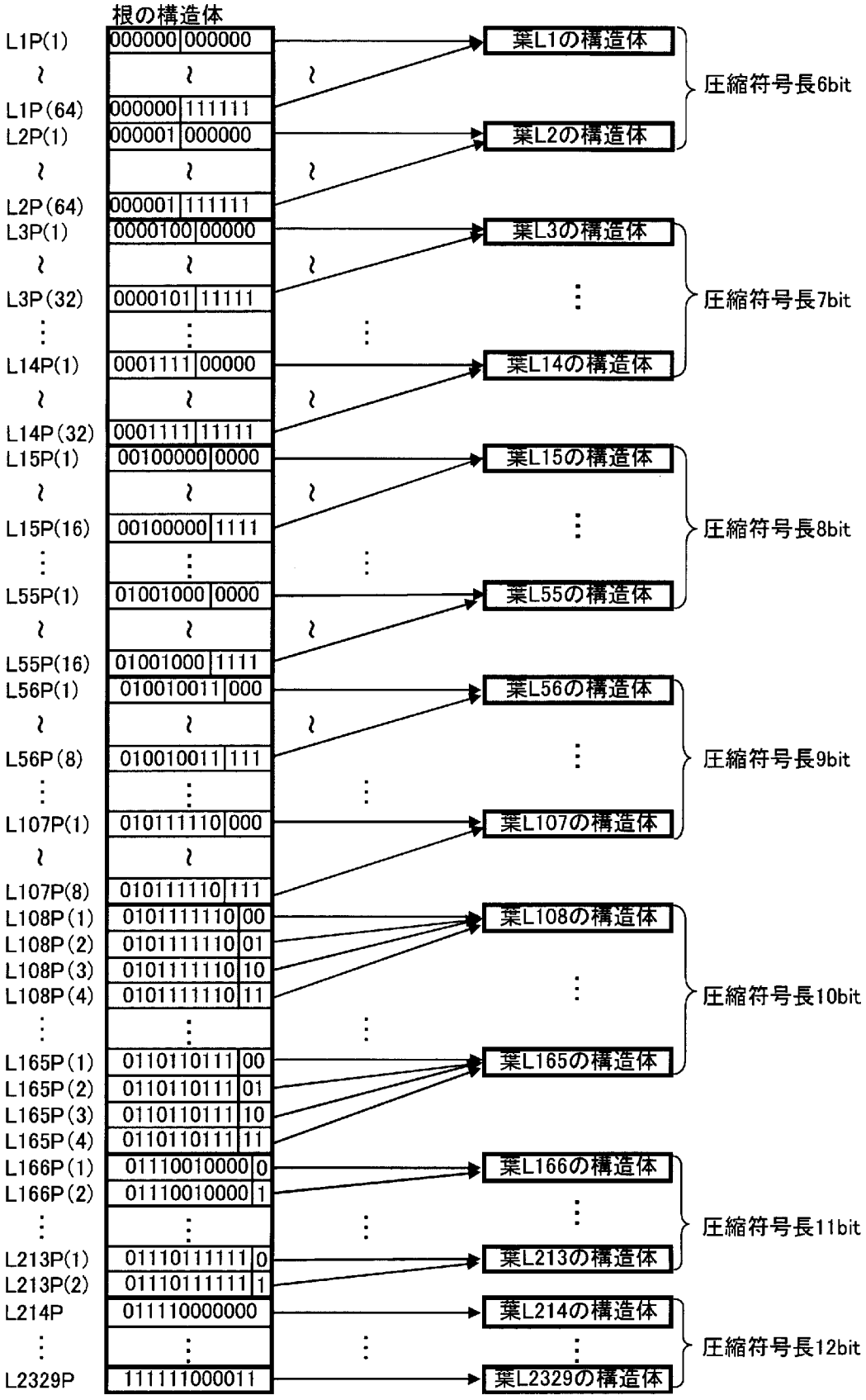
[図20]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
011100 10000
011100 11111
16個
011101 00000
11bit(48個)
011101 11111
32個
01110010000 0
01110010000 1
2本
圧縮符号と同じbit列
01110111111 0
01110111111 1
2本
圧縮符号と同じbit列

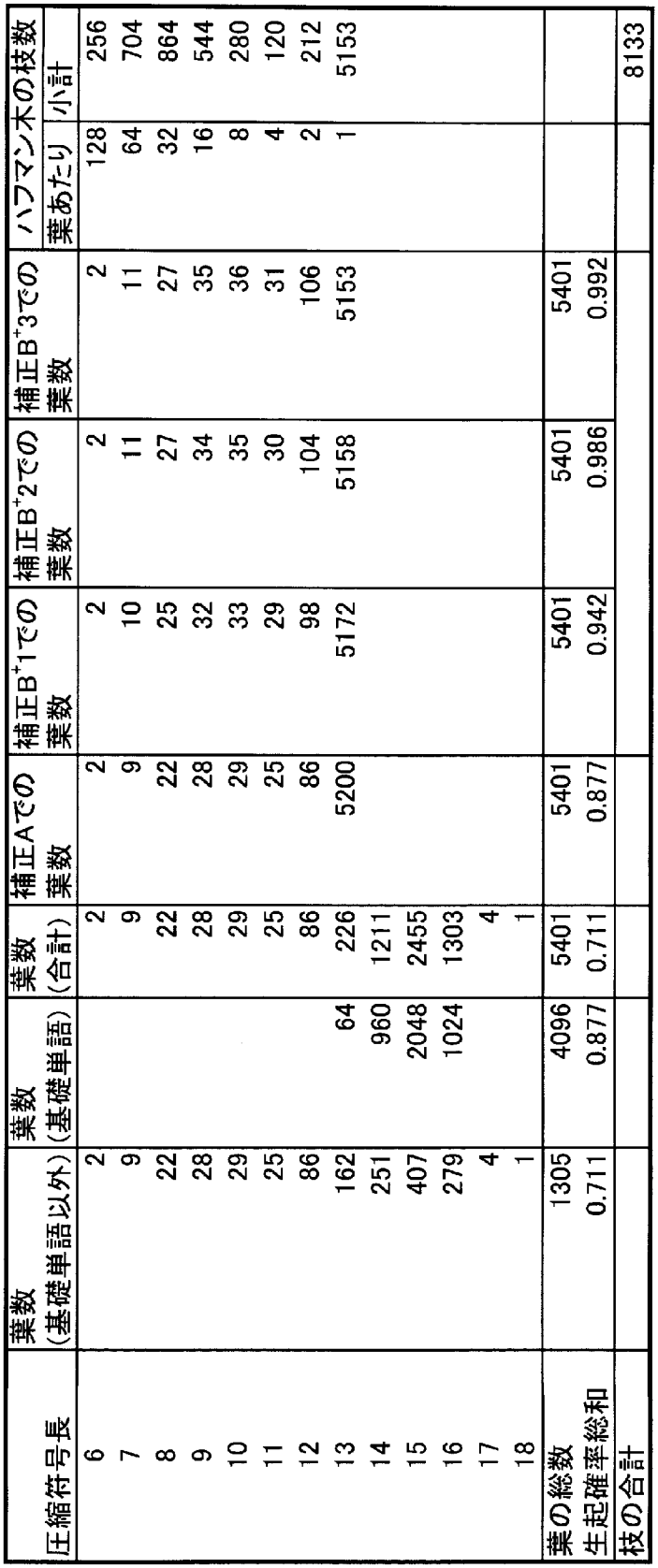
[図21]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
011110 000000
12bit(2116個)
2116個
111111 000011
011110000000
1本
圧縮符号と同じbit列
111111000011
1本
圧縮符号と同じbit列

[図22]

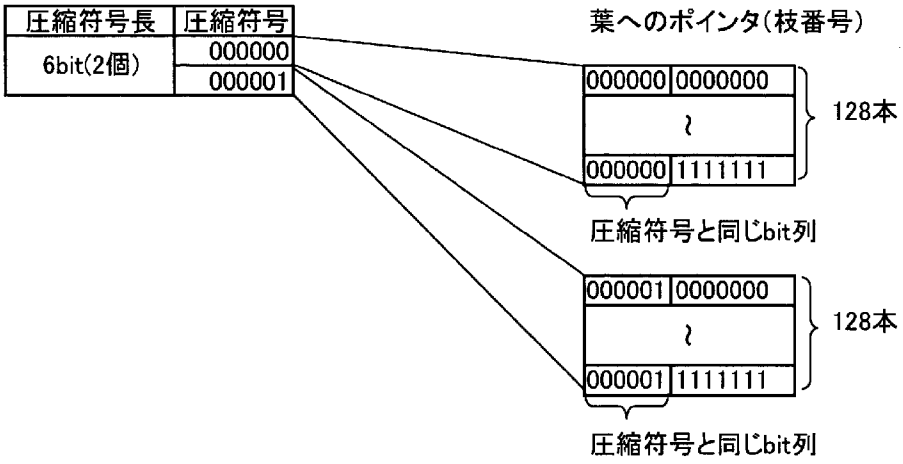

根の構造体
L1P(1)
000000 000000
L1P(64)
000000 111111
L2P(1)
000001 000000
L2P(64)
000001 111111
L3P(1)
0000100 00000
L3P(32)
0000101 11111
L14P(1)
0001111 00000
L14P(32)
0001111 11111
L15P(1)
00100000 0000
L15P(16)
00100000 1111
L55P(1)
01001000 0000
L55P(16)
01001000 1111
L56P(1)
010010011 000
L56P(8)
010010011 111
L107P(1)
010111110 000
L107P(8)
010111110 111
L108P(1)
0101111110 00
L108P(2)
0101111110 01
L108P(3)
0101111110 10
L108P(4)
0101111110 11
L165P(1)
0110110111 00
L165P(2)
0110110111 01
L165P(3)
0110110111 10
L165P(4)
0110110111 11
L166P(1)
01110010000 0
L166P(2)
01110010000 1
L213P(1)
01110111111 0
L213P(2)
01110111111 1
L214P
011110000000
L2329P
111111000011
葉L1の構造体
葉L2の構造体
葉L3の構造体
葉L14の構造体
葉L15の構造体
葉L55の構造体
葉L56の構造体
葉L107の構造体
葉L108の構造体
葉L165の構造体
葉L166の構造体
葉L213の構造体
葉L214の構造体
葉L2329の構造体
圧縮符号長6bit
圧縮符号長7bit
圧縮符号長8bit
圧縮符号長9bit
圧縮符号長10bit
圧縮符号長11bit
圧縮符号長12bit

[図23]

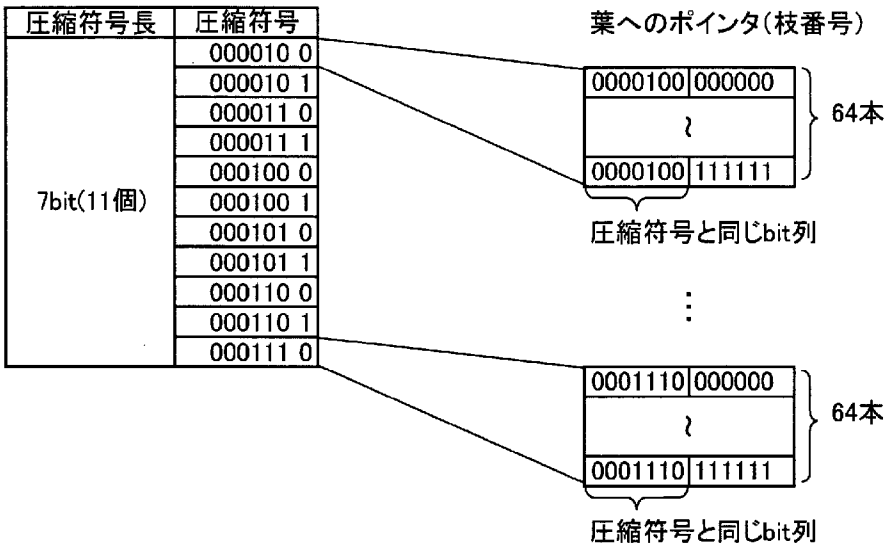


圧縮符号長	葉数（基礎単語以外）	葉数（基礎単語）	葉数（合計）	補正Aでの葉数	補正$B^{+}1$での葉数	補正$B^{+}2$での葉数	補正$B^{+}3$での葉数	ハフマン木の枝数 葉あたり	小計
6	2		2	2	2	2	2	128	256
7	9		9	9	10	11	11	64	704
8	22		22	22	25	27	27	32	864
9	28		28	28	32	34	35	16	544
10	29		29	29	33	35	36	8	280
11	25		25	25	29	30	31	4	120
12	86		86	86	98	104	106	2	212
13	162	64	226	5200	5172	5158	5153	1	5153
14	251	960	1211						
15	407	2048	2455						
16	279	1024	1303						
17	4		4						
18	1		1						
葉の総数	1305	4096	5401	5401	5401	5401	5401		
生起確率総和	0.711	0.877	0.711	0.877	0.942	0.986	0.992		
枝の合計									8133

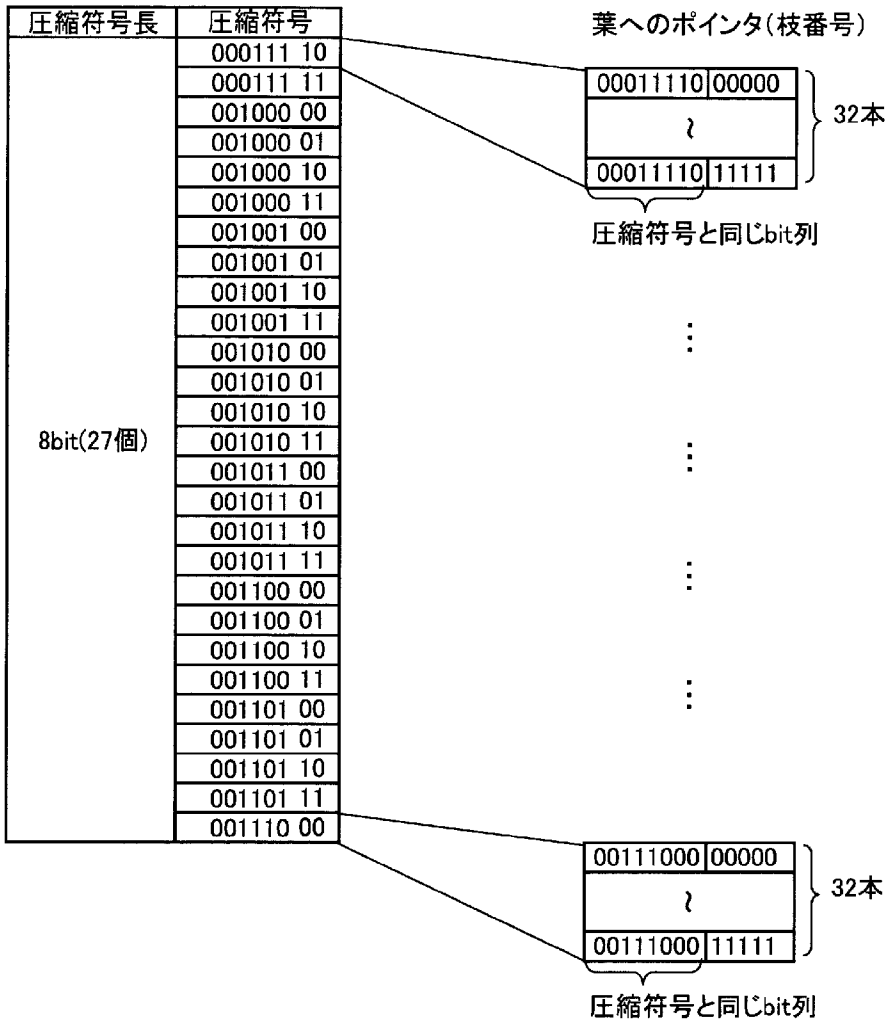
[図24]


圧縮符号長
圧縮符号
6bit(2個)
000000
000001
葉へのポインタ(枝番号)
000000 0000000
~
000000 1111111
128本
圧縮符号と同じbit列
000001 0000000
~
000001 1111111
128本
圧縮符号と同じbit列

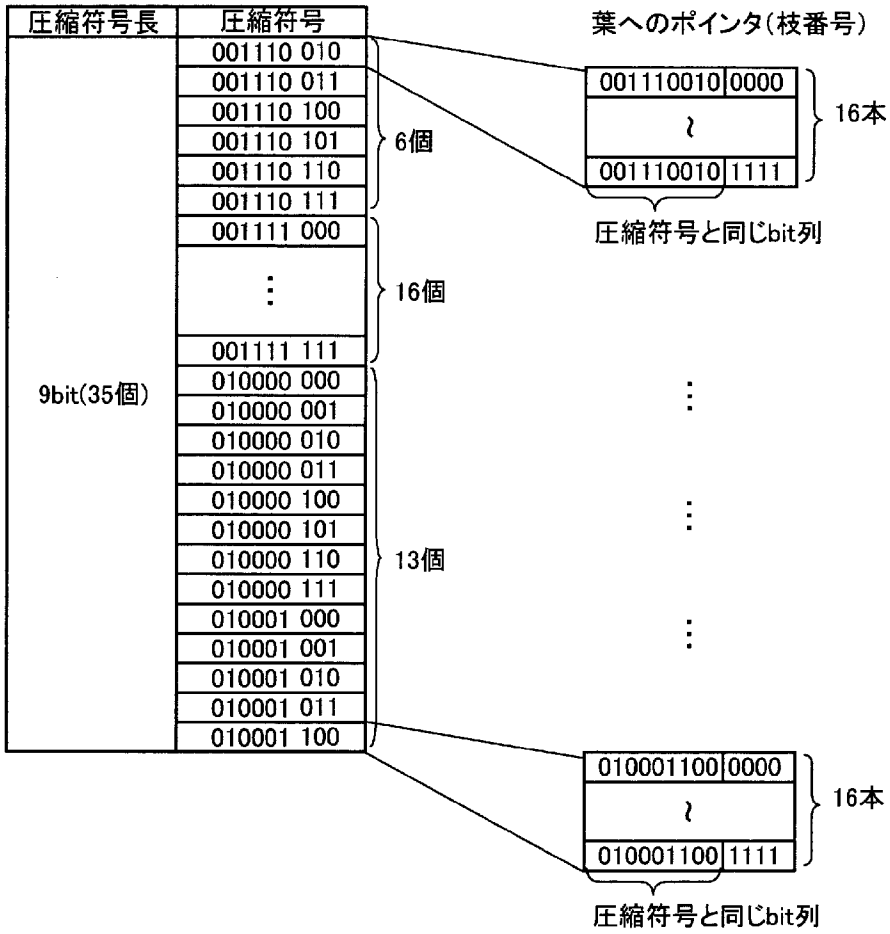
[図25]


圧縮符号長
圧縮符号
7bit(11個)
000010 0
000010 1
000011 0
000011 1
000100 0
000100 1
000101 0
000101 1
000110 0
000110 1
000111 0
葉へのポインタ(枝番号)
0000100 000000
~
0000100 111111
64本
圧縮符号と同じbit列
0001110 000000
~
0001110 111111
64本
圧縮符号と同じbit列

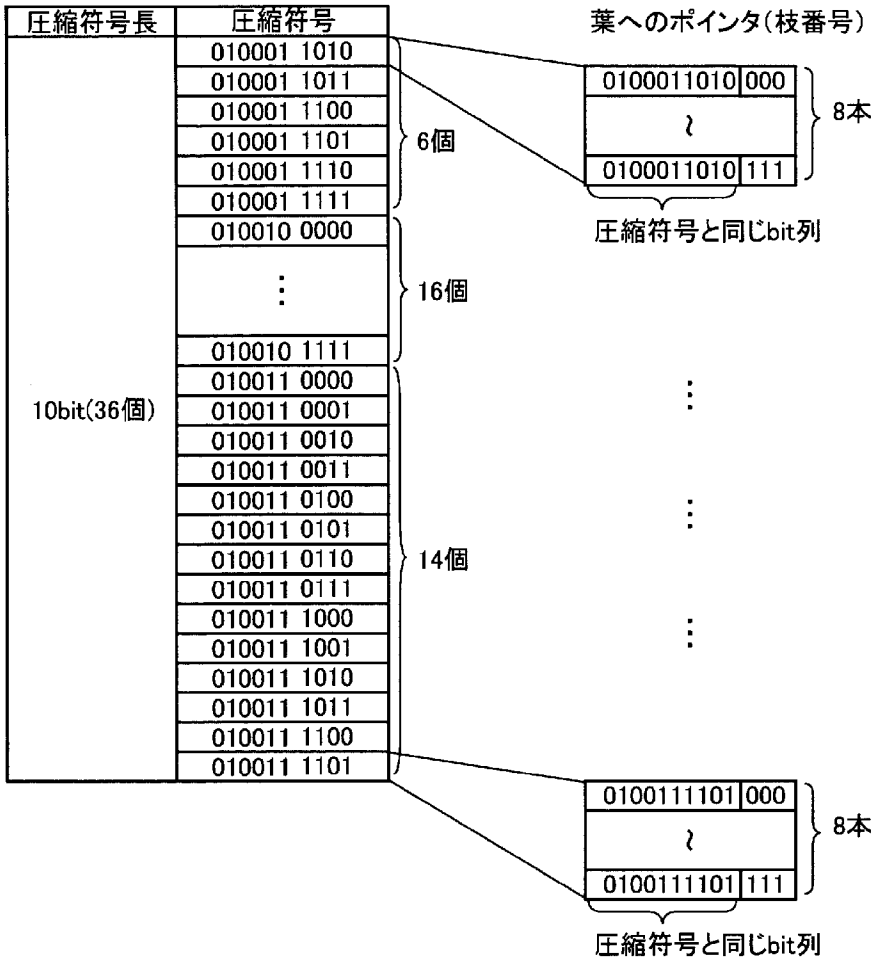
[図26]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
000111 10
000111 11
001000 00
001000 01
001000 10
001000 11
001001 00
001001 01
001001 10
001001 11
001010 00
001010 01
001010 10
001010 11
001011 00
001011 01
001011 10
001011 11
001100 00
001100 01
001100 10
001100 11
001101 00
001101 01
001101 10
001101 11
001110 00
8bit(27個)
00011110 00000
00011110 11111
32本
圧縮符号と同じbit列
00111000 00000
00111000 11111
32本
圧縮符号と同じbit列

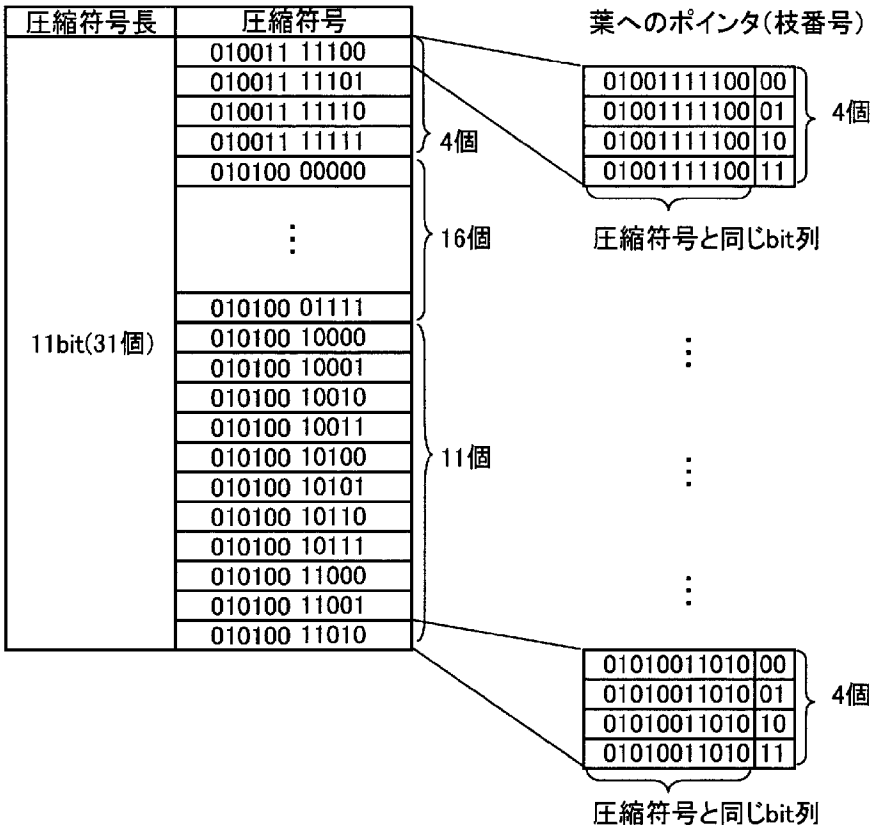
[図27]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
001110 010
001110 011
001110 100
001110 101
001110 110
001110 111
6個
001111 000
001111 111
16個
9bit(35個)
010000 000
010000 001
010000 010
010000 011
010000 100
010000 101
010000 110
010000 111
010001 000
010001 001
010001 010
010001 011
010001 100
13個
001110010 0000
001110010 1111
16本
圧縮符号と同じbit列
010001100 0000
010001100 1111
16本
圧縮符号と同じbit列

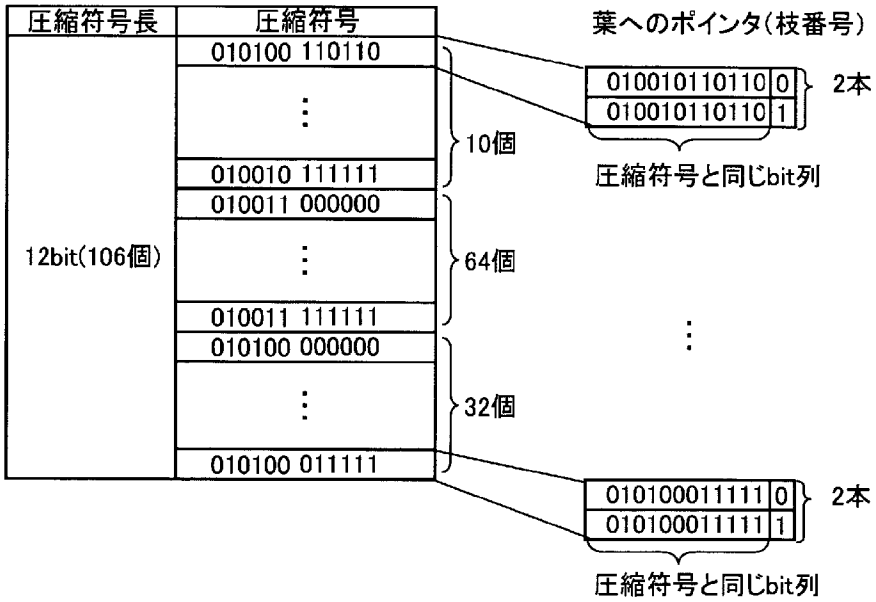
[図28]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
010001 1010
010001 1011
010001 1100
010001 1101
010001 1110
010001 1111
6個
0100011010 000
≀
0100011010 111
8本
圧縮符号と同じbit列
010010 0000
⋮
010010 1111
16個
010011 0000
010011 0001
010011 0010
010011 0011
010011 0100
010011 0101
010011 0110
010011 0111
010011 1000
010011 1001
010011 1010
010011 1011
010011 1100
010011 1101
14個
10bit(36個)
0100111101 000
≀
0100111101 111
8本
圧縮符号と同じbit列

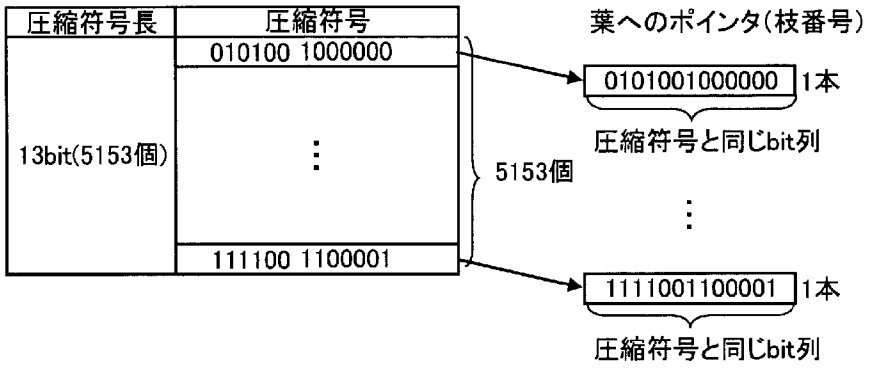
[図29]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
010011 11100
010011 11101
010011 11110
010011 11111
4個
010100 00000
16個
010100 01111
11bit(31個)
010100 10000
010100 10001
010100 10010
010100 10011
010100 10100
010100 10101
010100 10110
010100 10111
010100 11000
010100 11001
010100 11010
11個
01001111100 00
01001111100 01
01001111100 10
01001111100 11
4個
圧縮符号と同じbit列
01010011010 00
01010011010 01
01010011010 10
01010011010 11
4個
圧縮符号と同じbit列

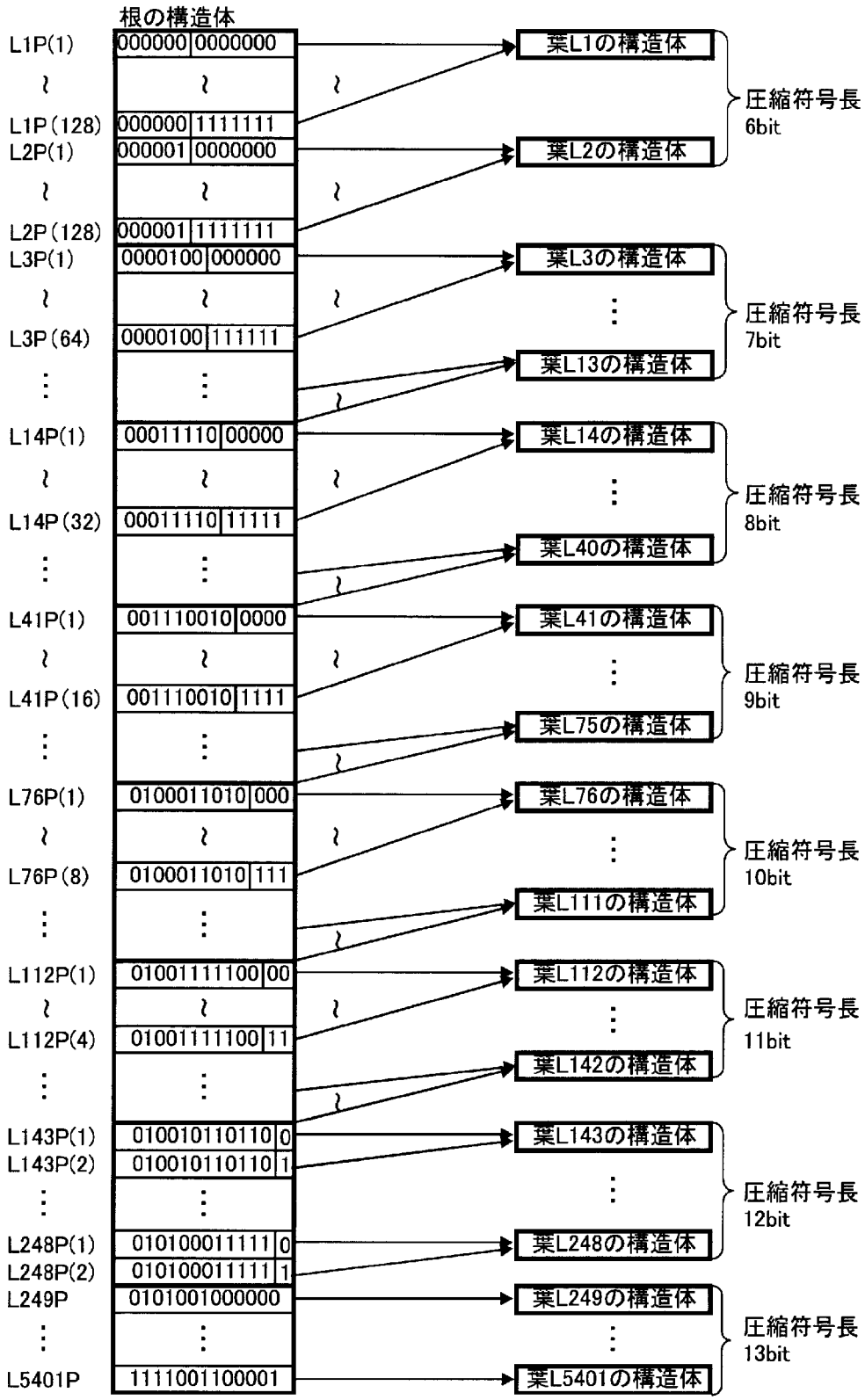
[図30]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
010100 110110
010010 111111
10個
010011 000000
010011 111111
64個
12bit(106個)
010100 000000
010100 011111
32個
0100101101100
0100101101101
2本
圧縮符号と同じbit列
0101000111110
0101000111111
2本
圧縮符号と同じbit列

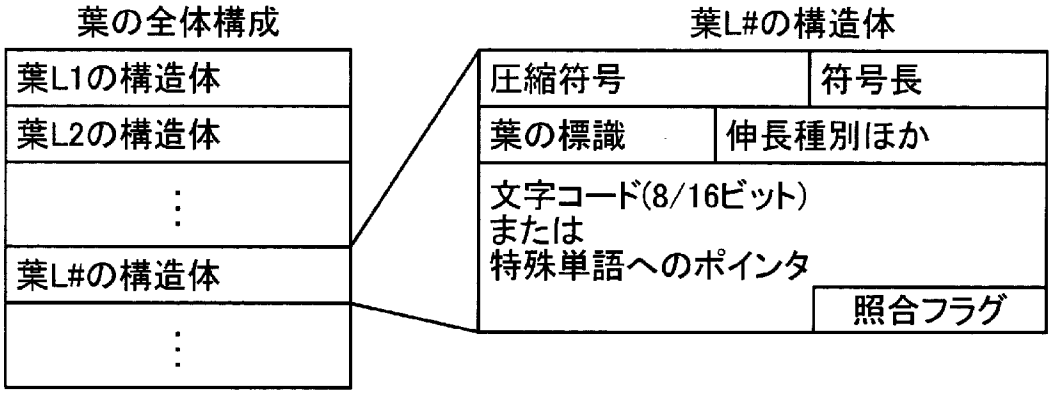
[図31]


圧縮符号長
圧縮符号
葉へのポインタ(枝番号)
010100 1000000
0101001000000
1本
13bit(5153個)
圧縮符号と同じbit列
5153個
111100 1100001
1111001100001
1本
圧縮符号と同じbit列

[図32]


根の構造体
L1P(1)
L1P(128)
L2P(1)
L2P(128)
L3P(1)
L3P(64)
L14P(1)
L14P(32)
L41P(1)
L41P(16)
L76P(1)
L76P(8)
L112P(1)
L112P(4)
L143P(1)
L143P(2)
L248P(1)
L248P(2)
L249P
L5401P
000000 0000000
000000 1111111
000001 0000000
000001 1111111
0000100 000000
0000100 111111
00011110 00000
00011110 11111
001110010 0000
001110010 1111
0100011010 000
0100011010 111
01001111100 00
01001111100 11
010010110110 0
010010110110 1
010100011111 0
010100011111 1
0101001000000
1111001100001
葉L1の構造体
葉L2の構造体
葉L3の構造体
葉L13の構造体
葉L14の構造体
葉L40の構造体
葉L41の構造体
葉L75の構造体
葉L76の構造体
葉L111の構造体
葉L112の構造体
葉L142の構造体
葉L143の構造体
葉L248の構造体
葉L249の構造体
葉L5401の構造体
圧縮符号長 6bit
圧縮符号長 7bit
圧縮符号長 8bit
圧縮符号長 9bit
圧縮符号長 10bit
圧縮符号長 11bit
圧縮符号長 12bit
圧縮符号長 13bit

[図33]



[図34]

3400

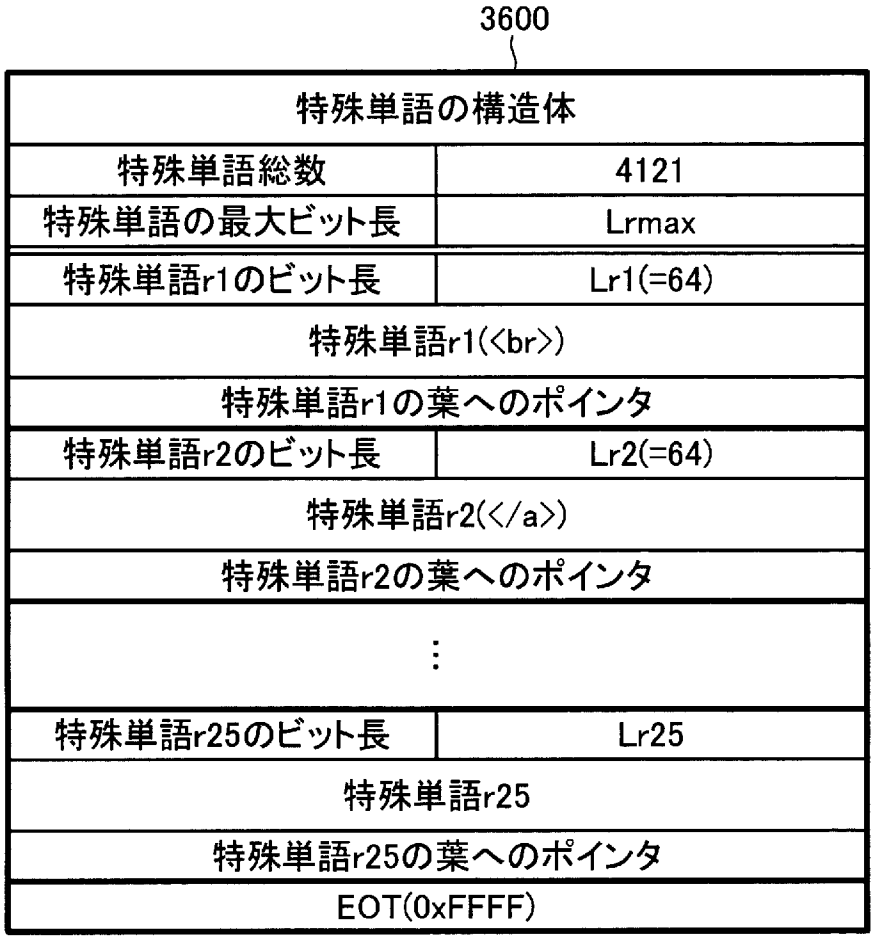
高位文字コードの構造体	
文字コード	葉へのポインタ
⋮	⋮
e691(0x216B: 次)	e691の葉へのポインタ
⋮	⋮
e24(0x6E30: の)	e24の葉へのポインタ
⋮	⋮
e79(0x6F30: は)	e79の葉へのポインタ
⋮	⋮

[図35]

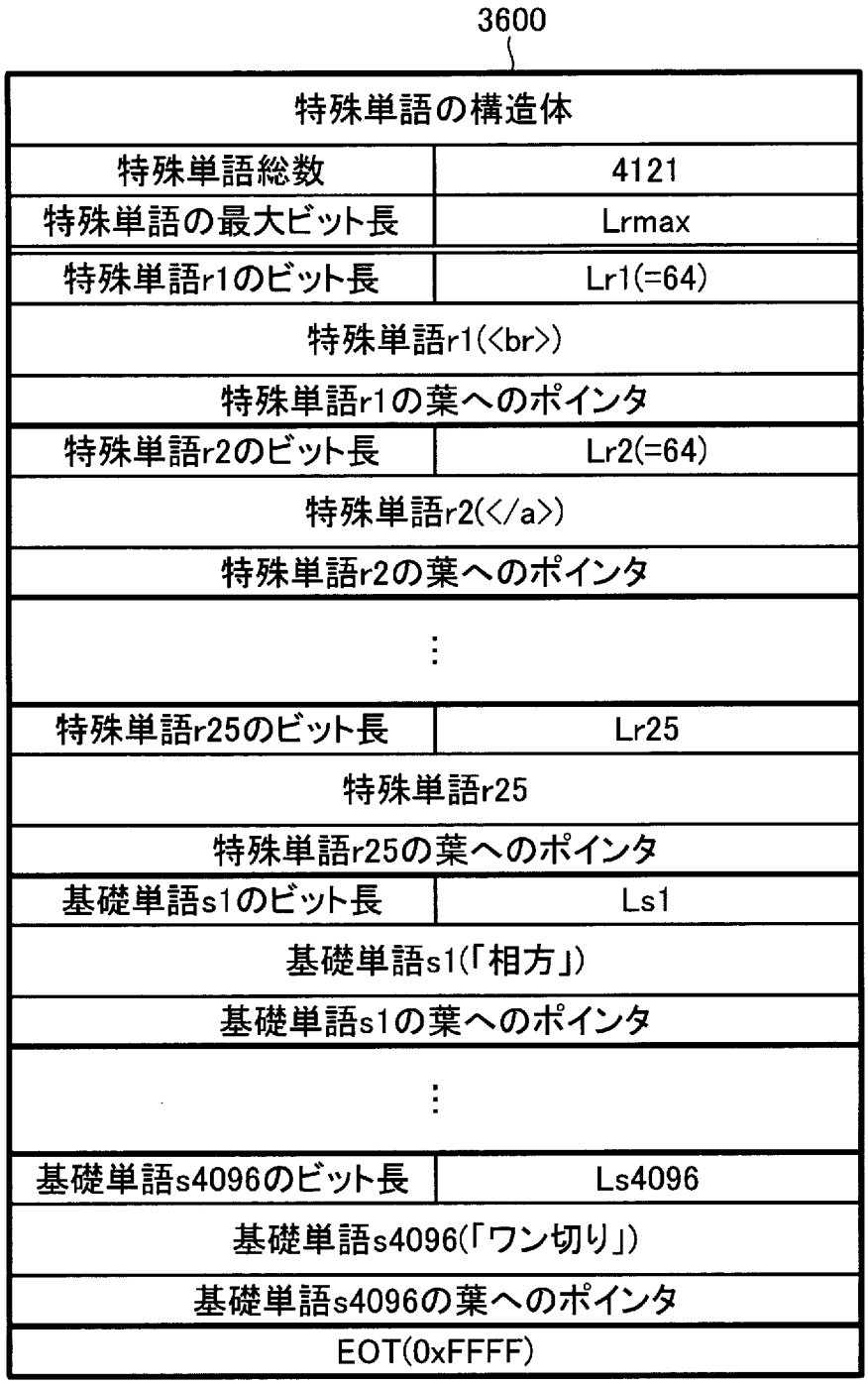
3500

分割文字コードの構造体	
分割文字コード	葉へのポインタ
0x00	「0x00」の葉へのポインタ
0x01	「0x01」の葉へのポインタ
⋮	⋮
0xFF	「0xFF」の葉へのポインタ

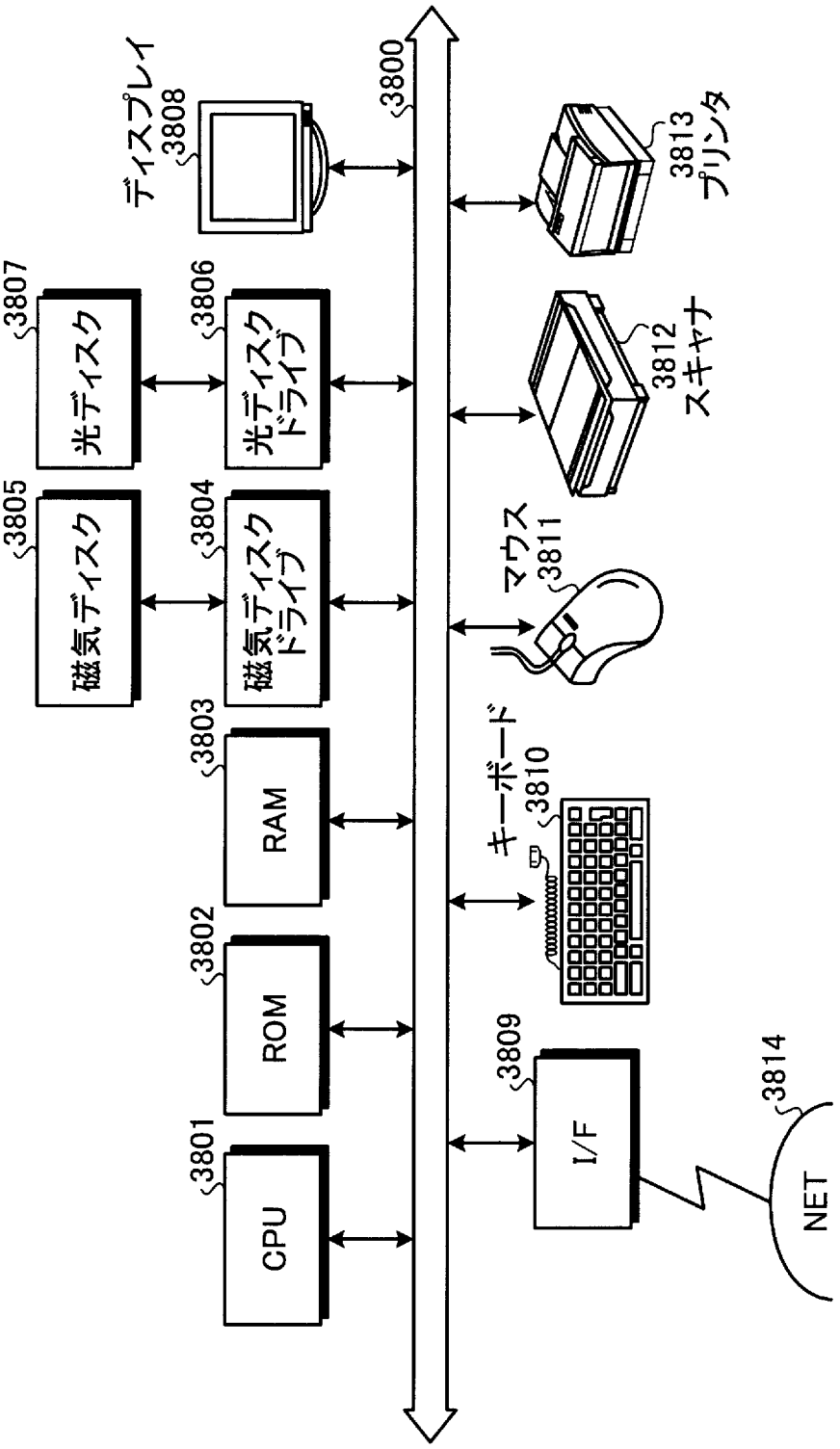
[図36]


3600
特殊単語の構造体
特殊単語総数
4121
特殊単語の最大ビット長
Lrmax
特殊単語r1のビット長
Lr1(=64)
特殊単語r1(<br>)
特殊単語r1の葉へのポインタ
特殊単語r2のビット長
Lr2(=64)
特殊単語r2()
特殊単語r2の葉へのポインタ
特殊単語r25のビット長
Lr25
特殊単語r25
特殊単語r25の葉へのポインタ
EOT(0xFFFF)

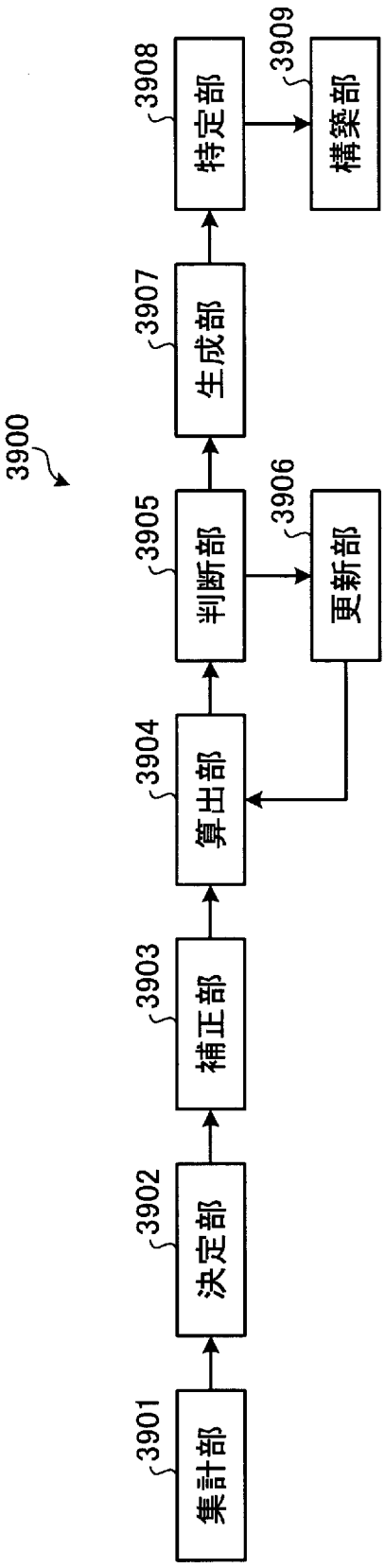
[図37]


3600
特殊単語の構造体
特殊単語総数
4121
特殊単語の最大ビット長
Lrmax
特殊単語r1のビット長
Lr1(=64)
特殊単語r1(<br>)
特殊単語r1の葉へのポインタ
特殊単語r2のビット長
Lr2(=64)
特殊単語r2()
特殊単語r2の葉へのポインタ
特殊単語r25のビット長
Lr25
特殊単語r25
特殊単語r25の葉へのポインタ
基礎単語s1のビット長
Ls1
基礎単語s1(「相方」)
基礎単語s1の葉へのポインタ
基礎単語s4096のビット長
Ls4096
基礎単語s4096(「ワン切り」)
基礎単語s4096の葉へのポインタ
EOT(0xFFFF)

[図38]


3800
ディスプレイ
3808
3813
プリンタ
3807
光ディスク
3806
光ディスク
ドライブ
3812
スキャナ
3805
磁気ディスク
3804
磁気ディスク
ドライブ
マウス
3811
3803
RAM
キーボード
3810
3802
ROM
3809
I/F
3814
NET
3801
CPU

[図39]


3900
3901
集計部
3902
決定部
3903
補正部
3904
算出部
3905
判断部
3906
更新部
3907
生成部
3908
特定部
3909
構築部

[図40]

	特殊単語(min25～max4121)	高頻度文字(max1024)	低頻度分割文字(256)
英語	英単語	英字、特殊記号	ー
日本語	日本語の単語	かな、カタカナ、漢字(教育漢字)	漢字(第2、3、4水準)ほか

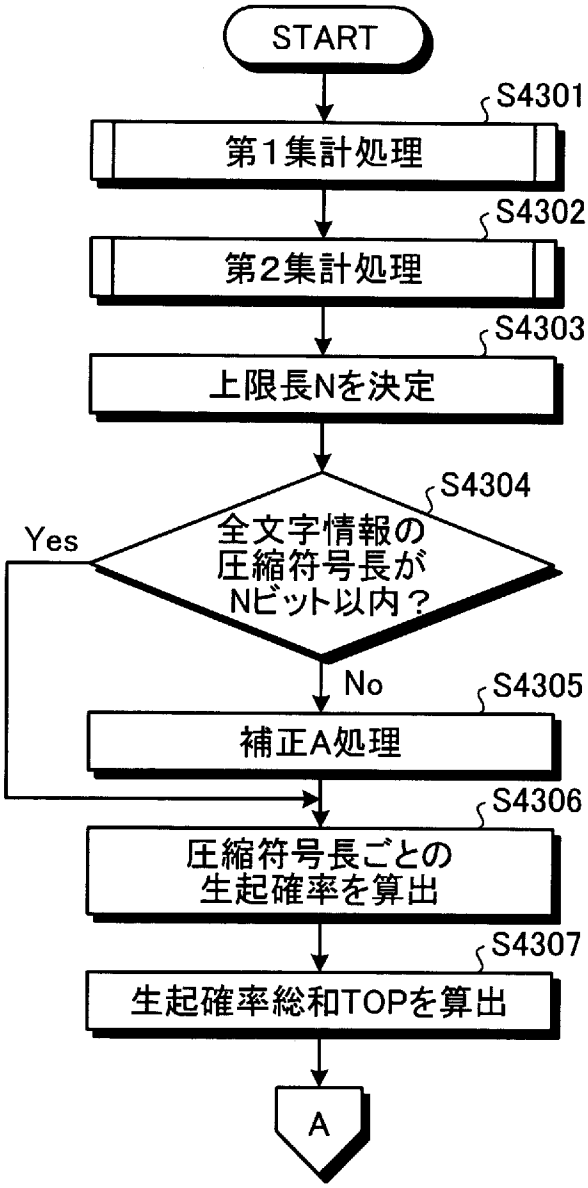
[図41]

	特殊単語(min25～max2073)	8ビット文字(256)
英語	英単語	英字、特殊記号

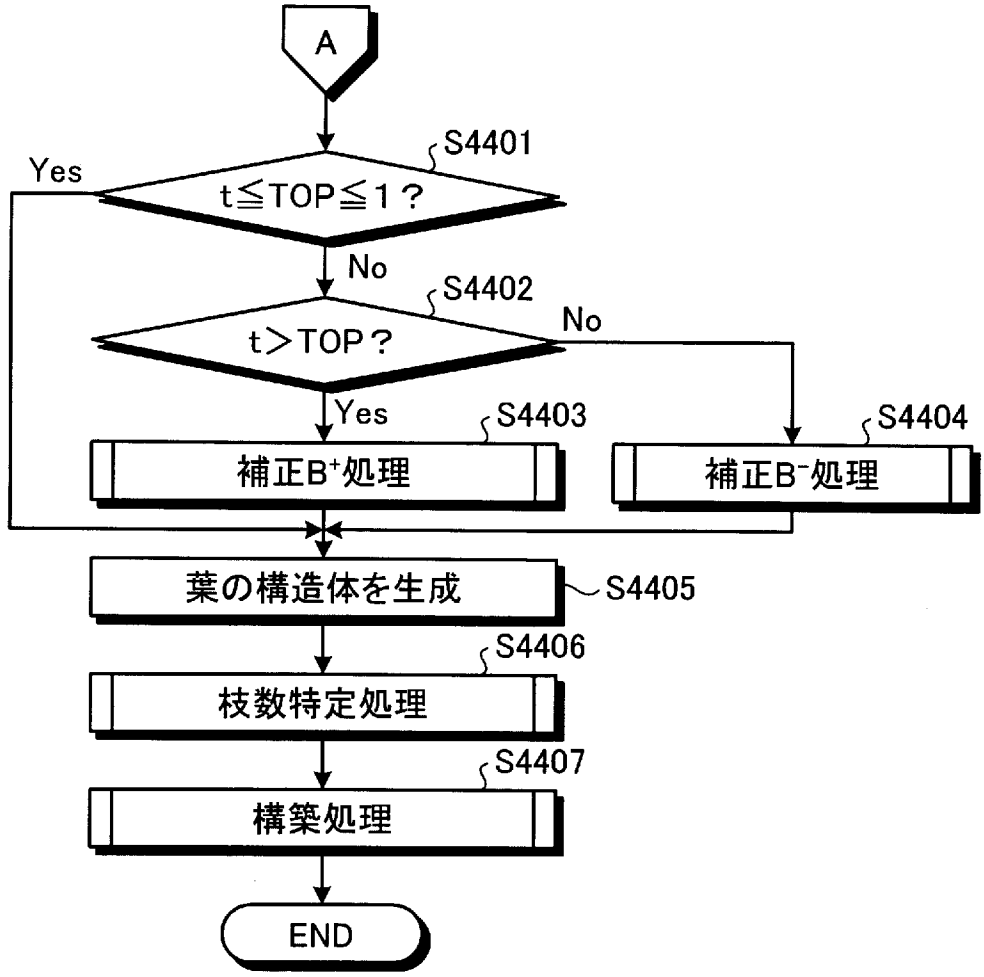
[図42]

	特殊単語(min25～max4121)と日本語の文字(256で組合せ)		8ビット文字(256)
英語	英単語	ー	英字、特殊記号
日本語	日本語の単語	かな、カタカナ、漢字	

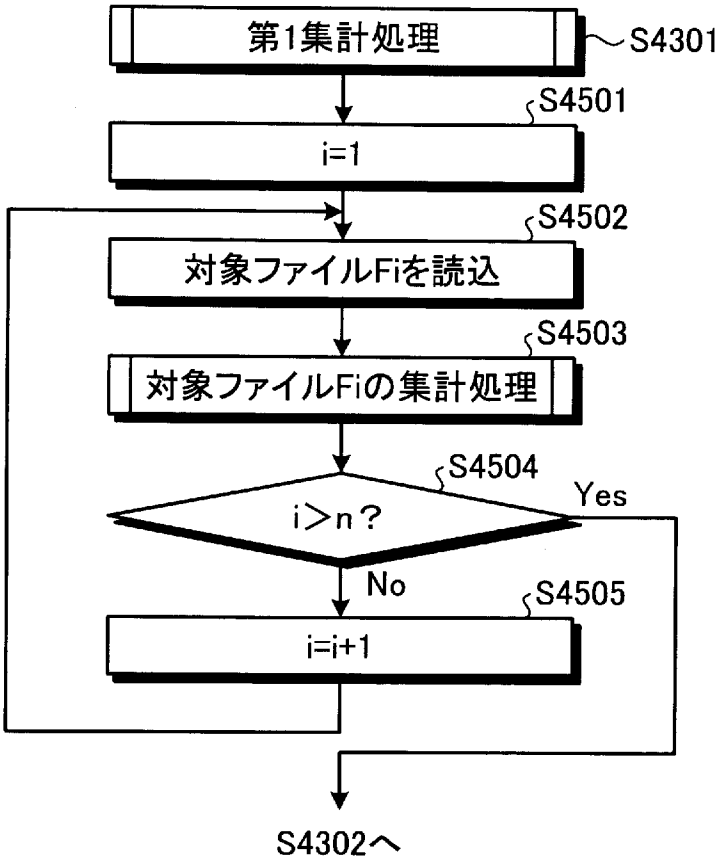
[図43]


START
S4301
第1集計処理
S4302
第2集計処理
S4303
上限長Nを決定
S4304
全文字情報の
圧縮符号長が
Nビット以内？
Yes
No
S4305
補正A処理
S4306
圧縮符号長ごとの
生起確率を算出
S4307
生起確率総和TOPを算出
A

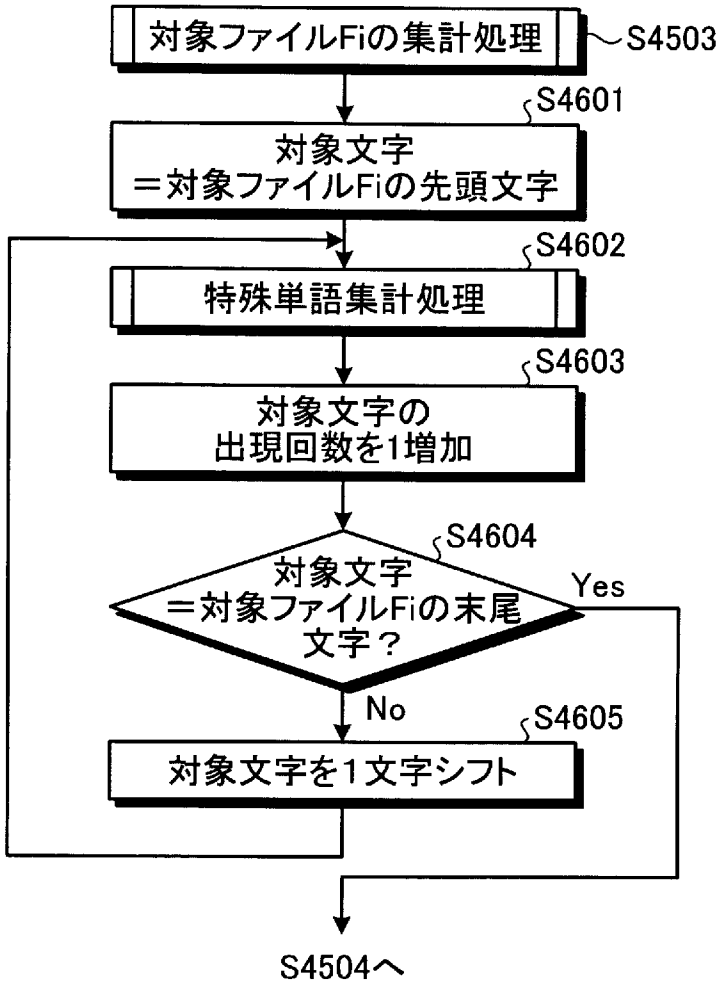
[図44]


A
S4401
Yes
t≦TOP≦1？
No
S4402
t＞TOP？
No
Yes
S4403
補正B+処理
S4404
補正B-処理
葉の構造体を生成
S4405
S4406
枝数特定処理
S4407
構築処理
END

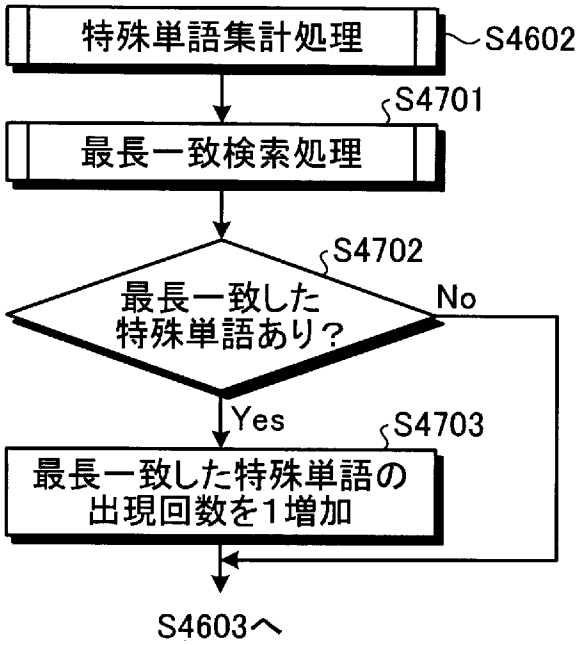
[図45]


第1集計処理
S4301
S4501
i=1
S4502
対象ファイルFiを読込
S4503
対象ファイルFiの集計処理
S4504
i>n？
Yes
No
S4505
i=i+1
S4302へ

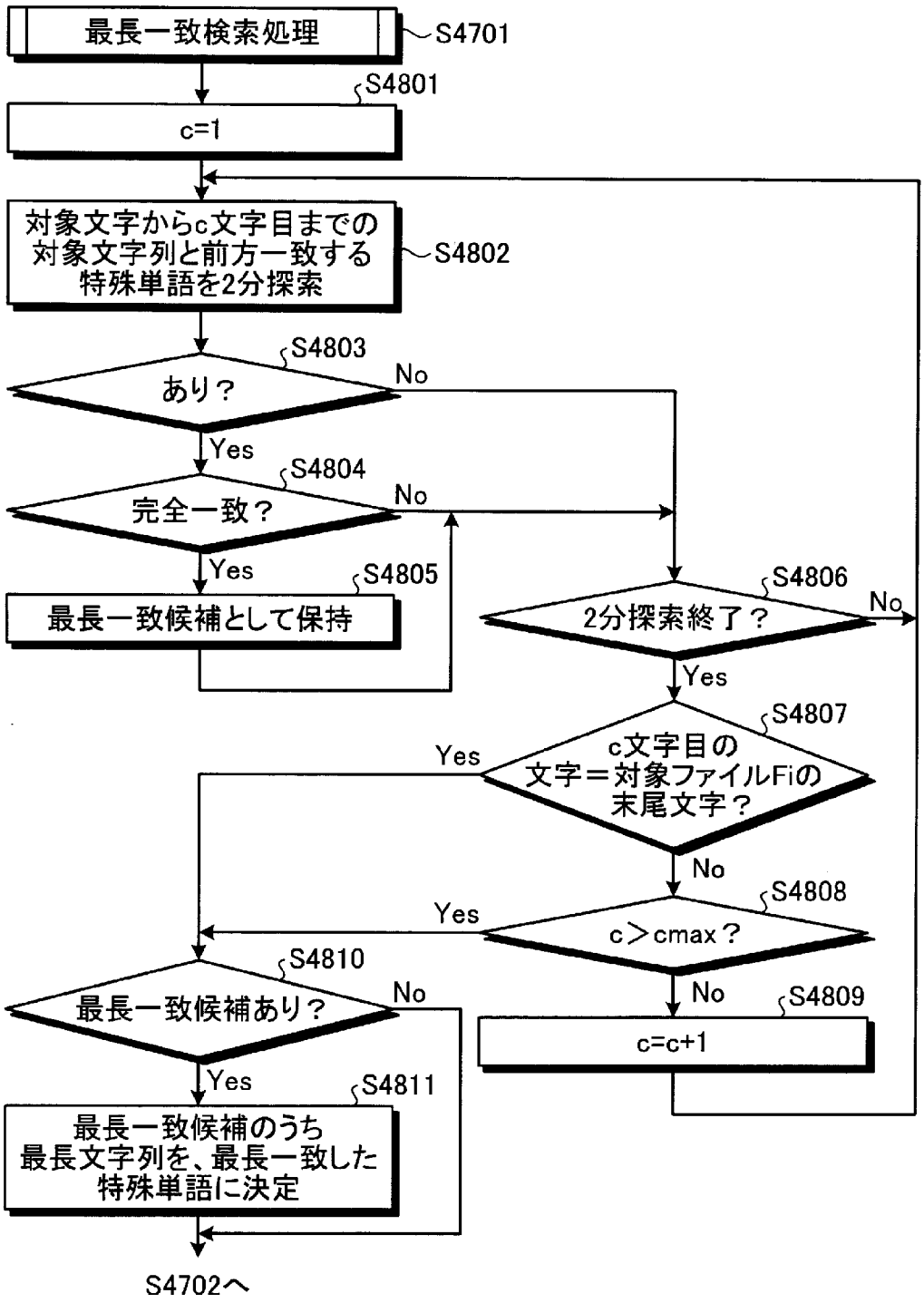
[図46]


対象ファイルFiの集計処理
S4503
S4601
対象文字
＝対象ファイルFiの先頭文字
S4602
特殊単語集計処理
S4603
対象文字の
出現回数を1増加
S4604
対象文字
＝対象ファイルFiの末尾
文字？
Yes
No
S4605
対象文字を1文字シフト
S4504へ

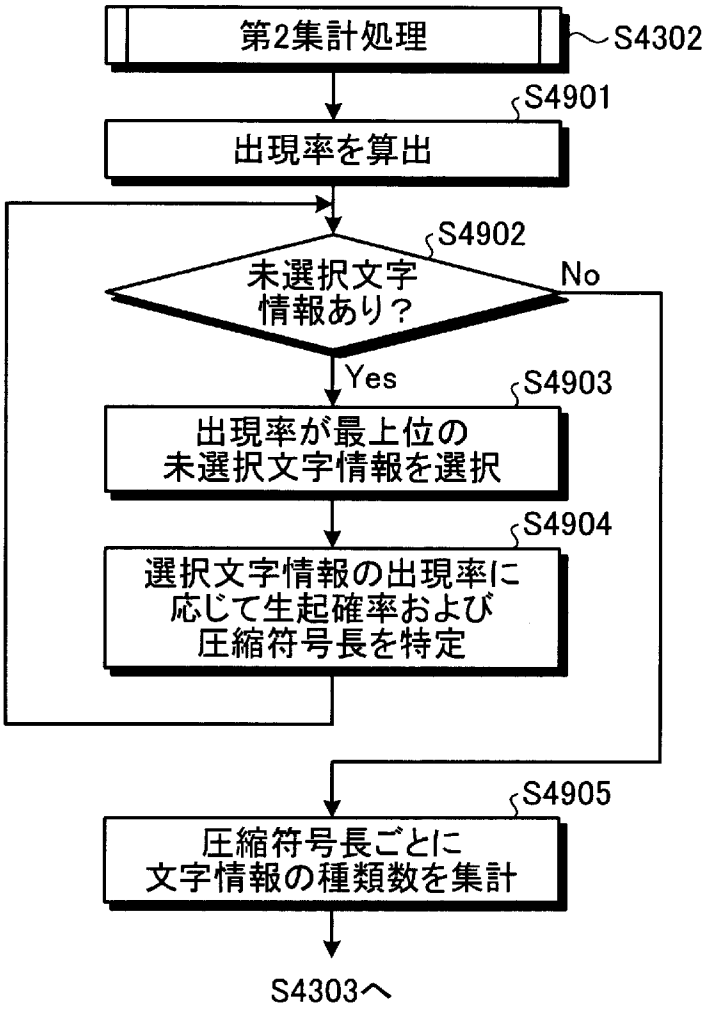
[図47]


特殊単語集計処理
S4602
S4701
最長一致検索処理
S4702
最長一致した
特殊単語あり？
No
Yes
S4703
最長一致した特殊単語の
出現回数を1増加
S4603へ

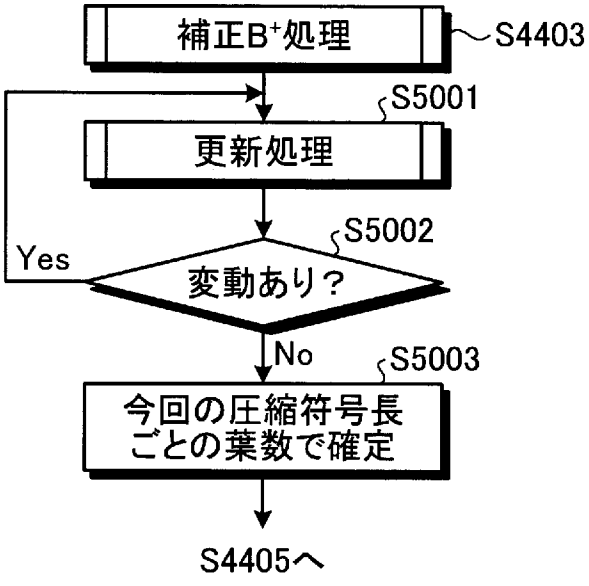
[図48]


最長一致検索処理
S4701
S4801
c=1
対象文字からc文字目までの
対象文字列と前方一致する
特殊単語を2分探索
S4802
S4803
あり？
No
Yes
S4804
完全一致？
No
Yes
S4805
最長一致候補として保持
S4806
2分探索終了？
No
Yes
S4807
c文字目の
文字＝対象ファイルFiの
末尾文字？
Yes
No
S4808
c＞cmax？
Yes
No
S4809
c=c+1
S4810
最長一致候補あり？
No
Yes
S4811
最長一致候補のうち
最長文字列を、最長一致した
特殊単語に決定
S4702へ

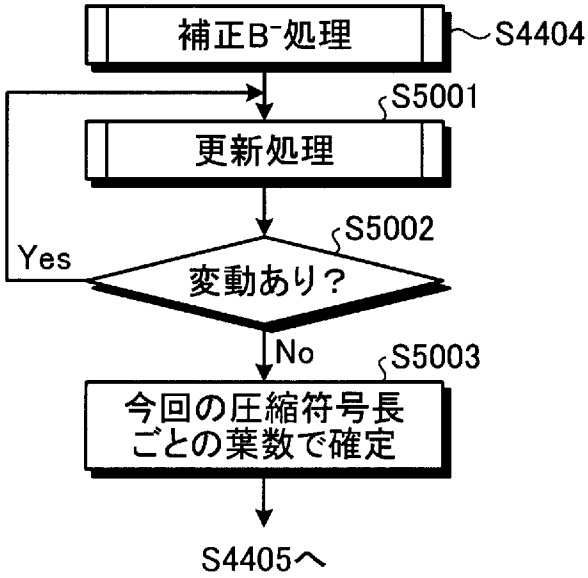
[図49]


第2集計処理
S4302
S4901
出現率を算出
S4902
未選択文字
情報あり？
No
Yes
S4903
出現率が最上位の
未選択文字情報を選択
S4904
選択文字情報の出現率に
応じて生起確率および
圧縮符号長を特定
S4905
圧縮符号長ごとに
文字情報の種類数を集計
S4303へ

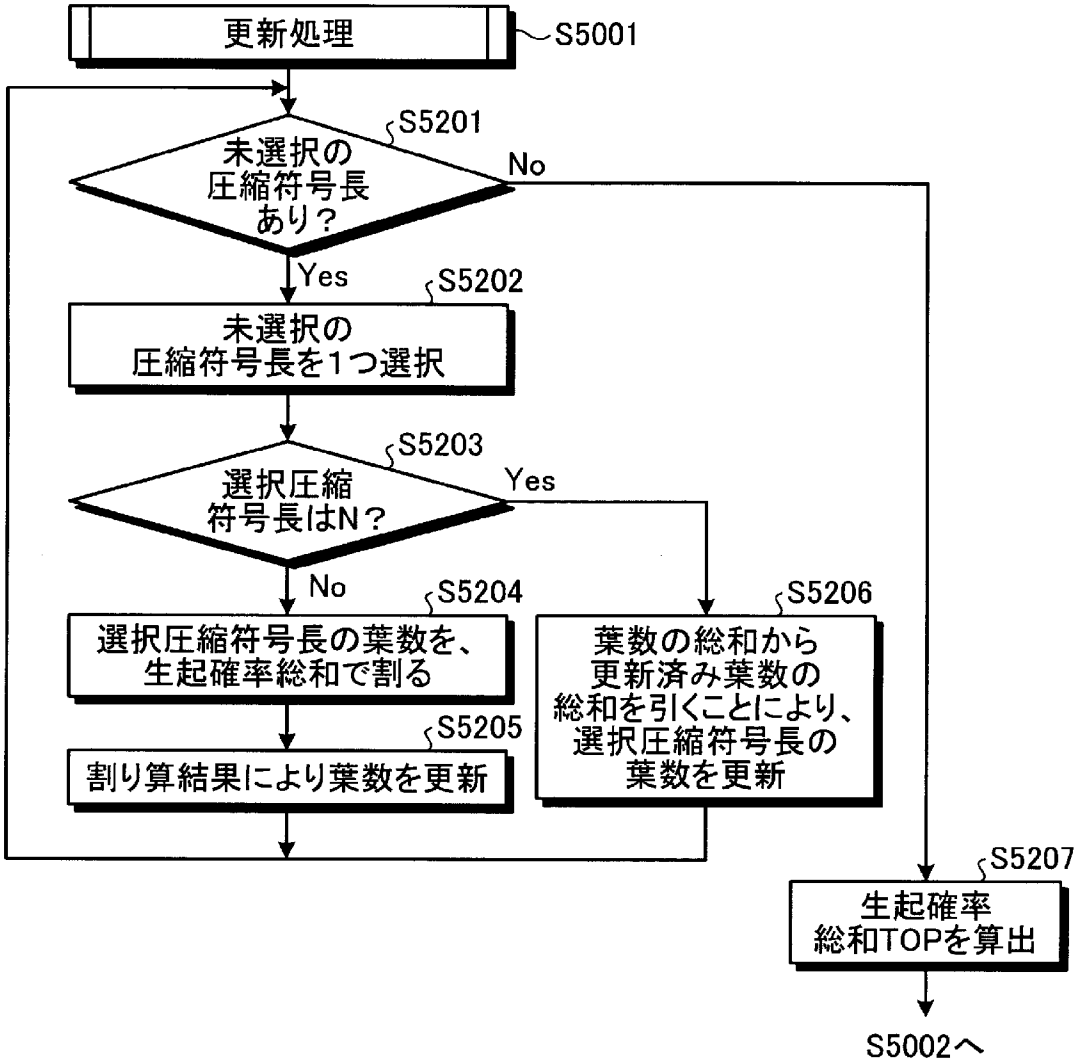
[図50]


補正B+処理
S4403
S5001
更新処理
S5002
Yes
変動あり？
No
S5003
今回の圧縮符号長
ごとの葉数で確定
S4405へ

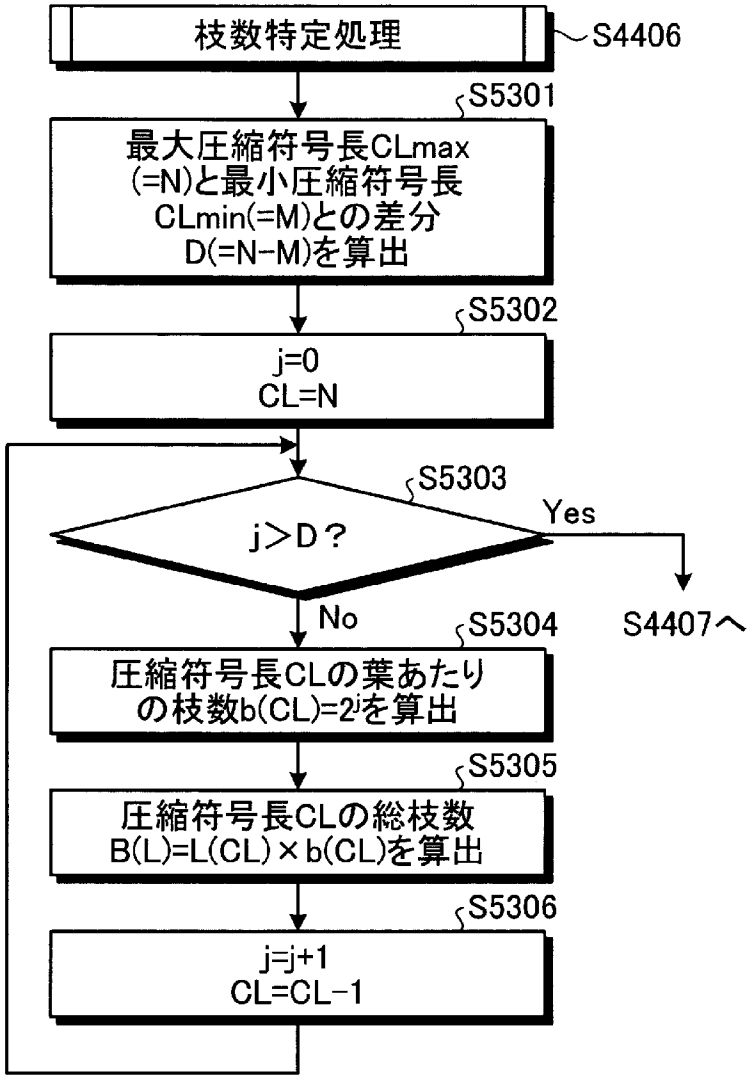
[図51]


補正B⁻処理
S4404
S5001
更新処理
S5002
Yes
変動あり？
No
S5003
今回の圧縮符号長
ごとの葉数で確定
S4405へ

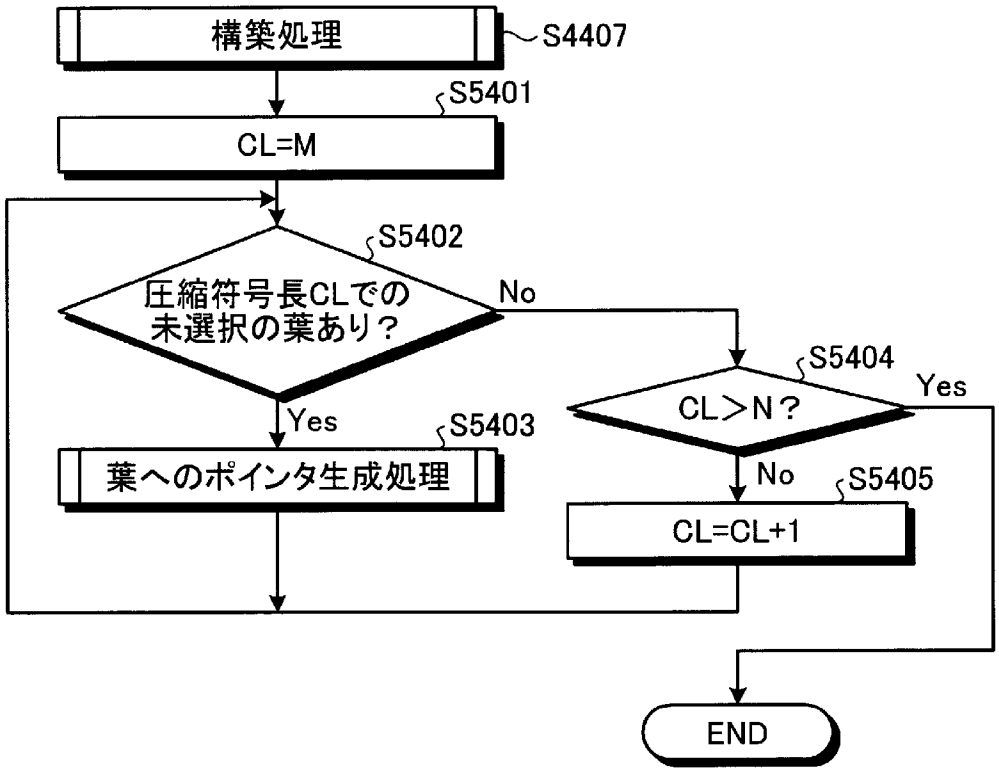
[図52]


更新処理
S5001
S5201
未選択の
圧縮符号長
あり？
No
Yes
S5202
未選択の
圧縮符号長を1つ選択
S5203
選択圧縮
符号長はN？
Yes
No
S5204
選択圧縮符号長の葉数を、
生起確率総和で割る
S5205
割り算結果により葉数を更新
S5206
葉数の総和から
更新済み葉数の
総和を引くことにより、
選択圧縮符号長の
葉数を更新
S5207
生起確率
総和TOPを算出
S5002へ

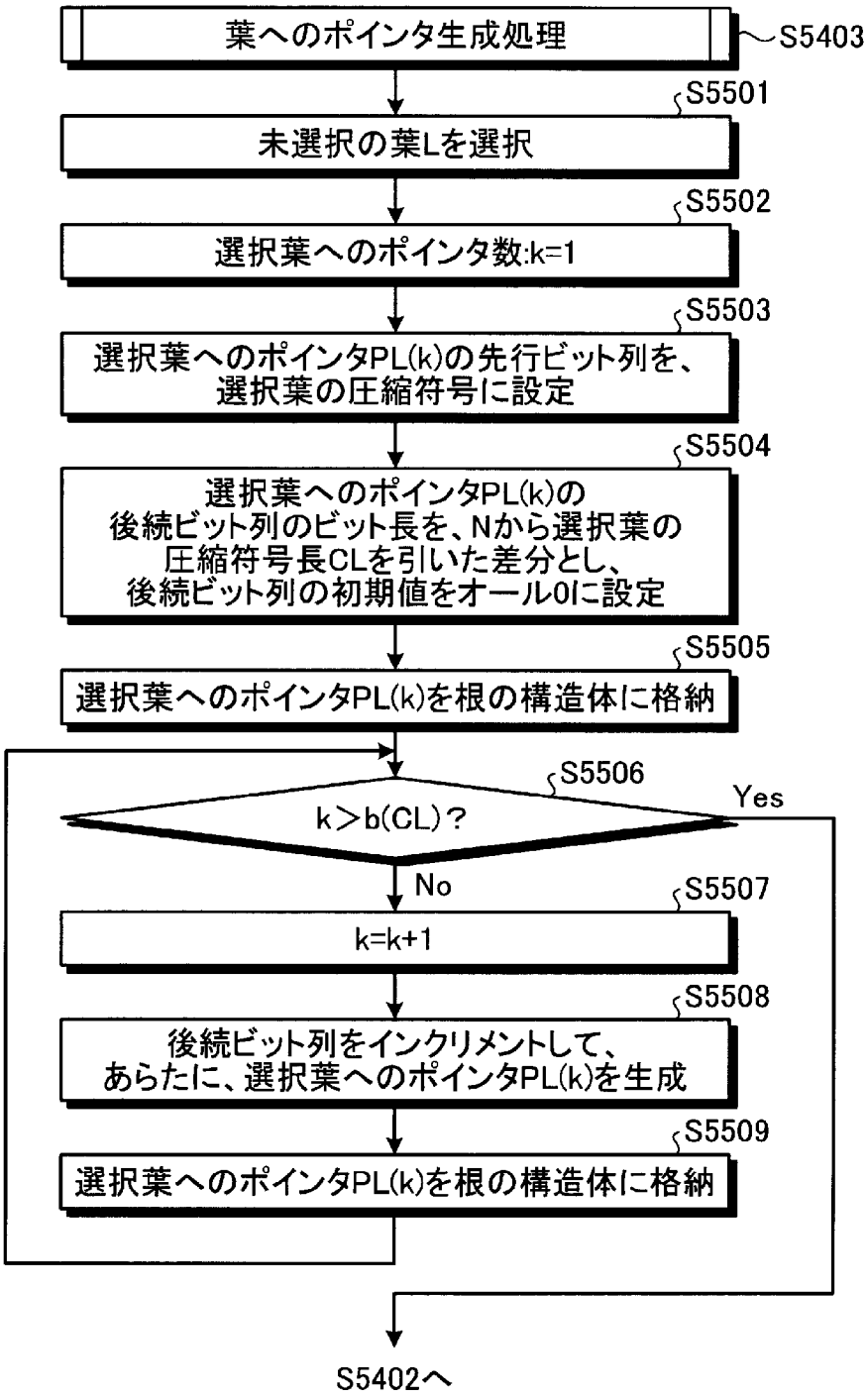
[図53]


枝数特定処理
S4406
S5301
最大圧縮符号長CLmax
(=N)と最小圧縮符号長
CLmin(=M)との差分
D(=N-M)を算出
S5302
j=0
CL=N
S5303
j>D?
Yes
S4407へ
No
S5304
圧縮符号長CLの葉あたり
の枝数b(CL)=2^jを算出
S5305
圧縮符号長CLの総枝数
B(L)=L(CL)×b(CL)を算出
S5306
j=j+1
CL=CL-1

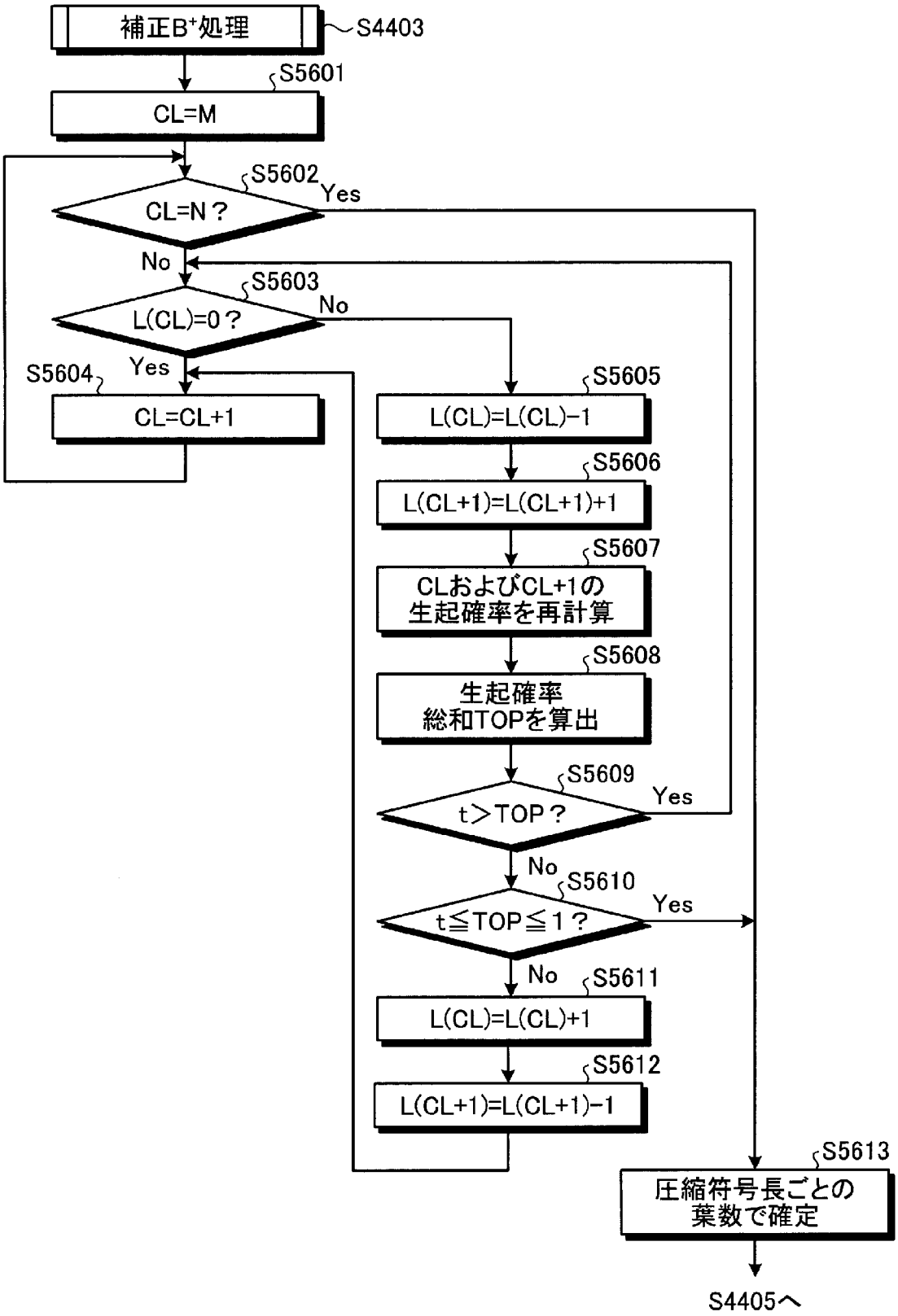
[図54]


構築処理
S4407
S5401
CL=M
S5402
圧縮符号長CLでの
未選択の葉あり？
No
Yes
S5403
葉へのポインタ生成処理
S5404
CL＞N？
Yes
No
S5405
CL=CL+1
END

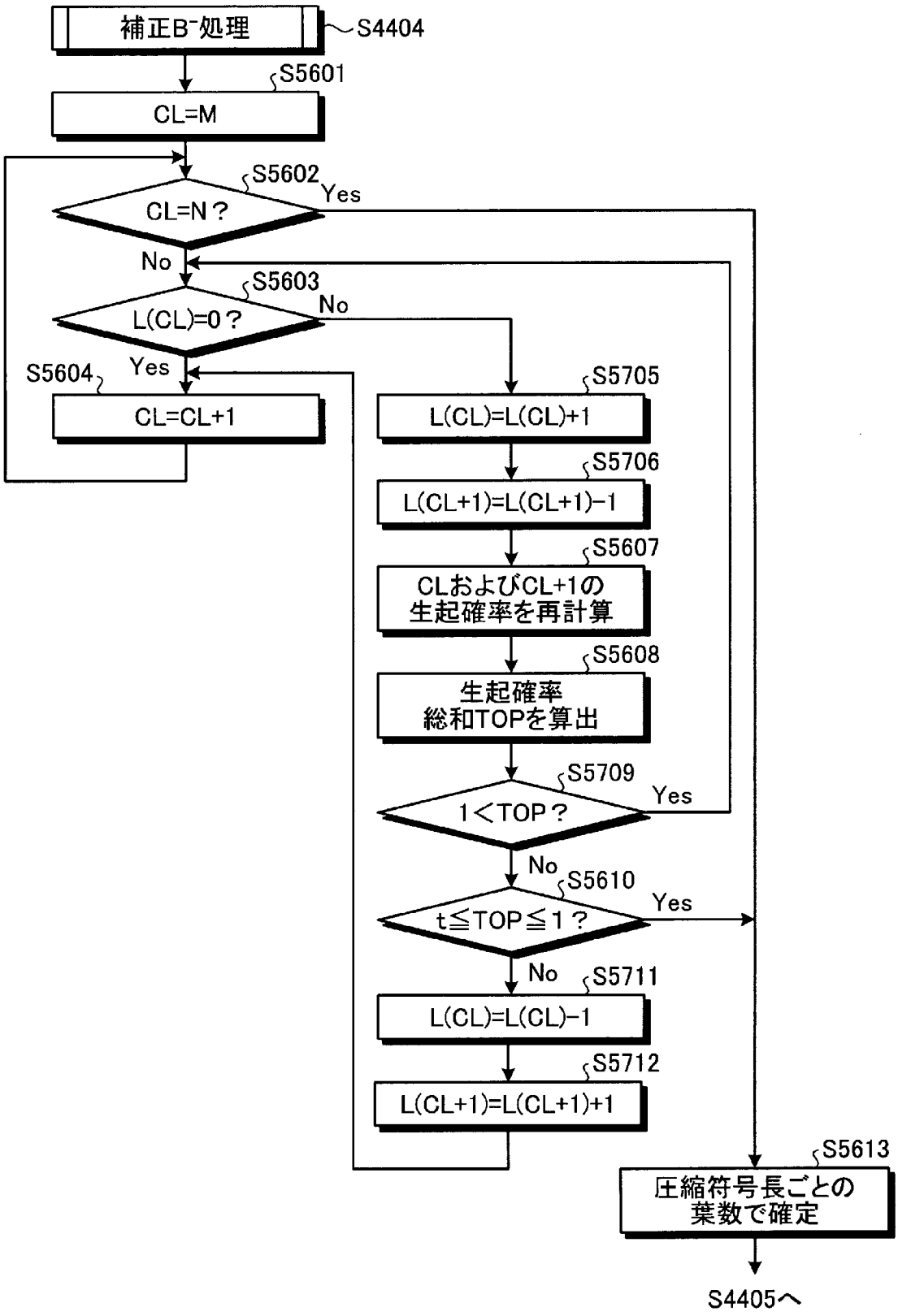
[図55]


葉へのポインタ生成処理
S5403
S5501
未選択の葉Lを選択
S5502
選択葉へのポインタ数:k=1
S5503
選択葉へのポインタPL(k)の先行ビット列を、
選択葉の圧縮符号に設定
S5504
選択葉へのポインタPL(k)の
後続ビット列のビット長を、Nから選択葉の
圧縮符号長CLを引いた差分とし、
後続ビット列の初期値をオール0に設定
S5505
選択葉へのポインタPL(k)を根の構造体に格納
S5506
k>b(CL)?
Yes
No
S5507
k=k+1
S5508
後続ビット列をインクリメントして、
あらたに、選択葉へのポインタPL(k)を生成
S5509
選択葉へのポインタPL(k)を根の構造体に格納
S5402へ

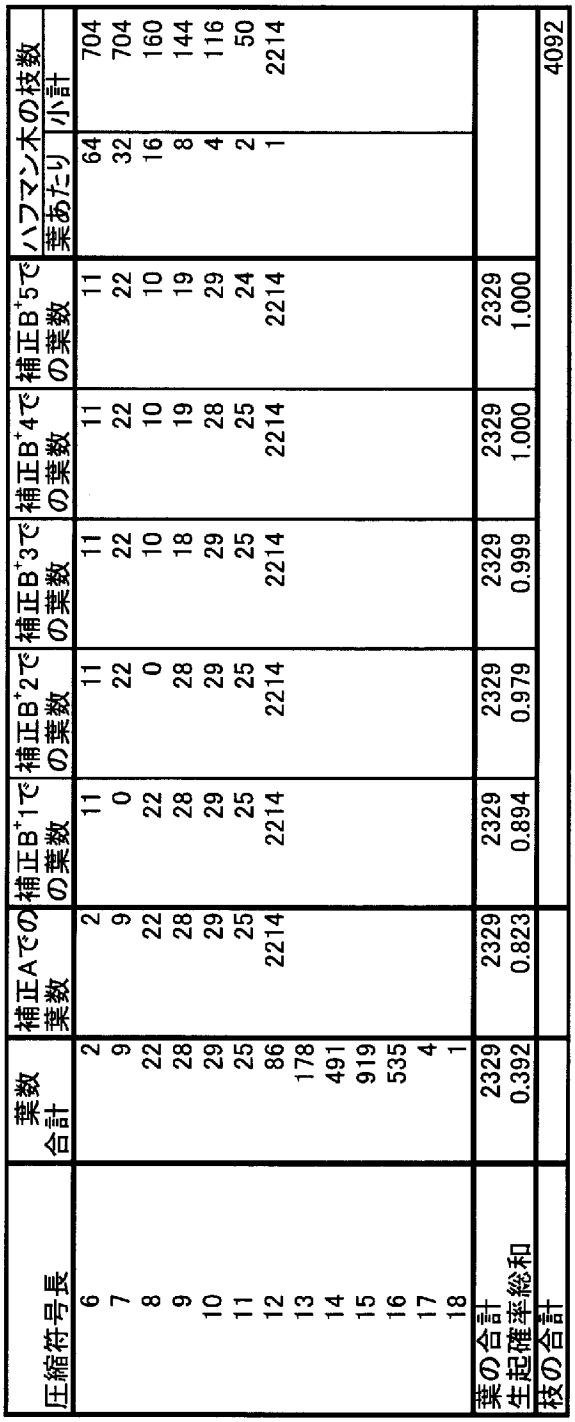
[図56]


補正B*処理
S4403
S5601
CL=M
S5602
CL=N？
Yes
No
S5603
L(CL)=0？
No
Yes
S5604
CL=CL+1
S5605
L(CL)=L(CL)−1
S5606
L(CL+1)=L(CL+1)+1
S5607
CLおよびCL+1の
生起確率を再計算
S5608
生起確率
総和TOPを算出
S5609
t＞TOP？
Yes
No
S5610
t≦TOP≦1？
Yes
No
S5611
L(CL)=L(CL)+1
S5612
L(CL+1)=L(CL+1)−1
S5613
圧縮符号長ごとの
葉数で確定
S4405へ

[図57]


補正B⁻処理
S4404
S5601
CL=M
S5602
CL=N？
Yes
No
S5603
L(CL)=0？
No
S5604
Yes
CL=CL+1
S5705
L(CL)=L(CL)+1
S5706
L(CL+1)=L(CL+1)−1
S5607
CLおよびCL+1の
生起確率を再計算
S5608
生起確率
総和TOPを算出
S5709
1＜TOP？
Yes
No
S5610
t≦TOP≦1？
Yes
No
S5711
L(CL)=L(CL)−1
S5712
L(CL+1)=L(CL+1)+1
S5613
圧縮符号長ごとの
葉数で確定
S4405へ

[図58]

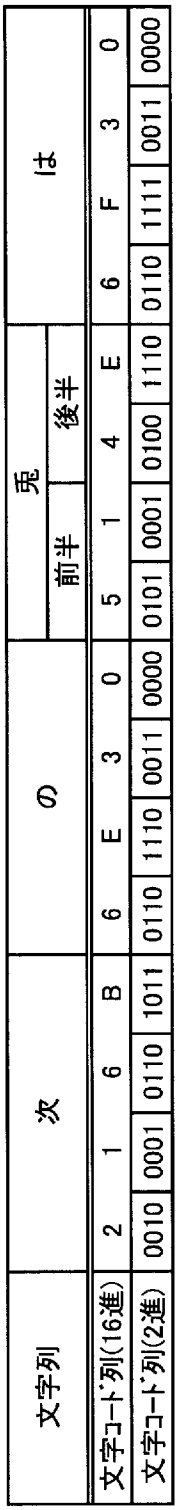


圧縮符号長	葉数合計	補正Aでの葉数	補正B+1での葉数	補正B+2での葉数	補正B+3での葉数	補正B+4での葉数	補正B+5での葉数	ハフマン木の枝数 葉あたり	小計
6	2	2	11	11	11	11	11	64	704
7	9	9	0	22	22	22	22	32	704
8	22	22	22	0	10	10	10	16	160
9	28	28	28	28	18	19	19	8	144
10	29	29	29	29	29	28	29	4	116
11	25	25	25	25	25	25	24	2	50
12	86	2214	2214	2214	2214	2214	2214	1	2214
13	178								
14	491								
15	919								
16	535								
17	4								
18	1								
葉の合計	2329	2329	2329	2329	2329	2329	2329		
生起確率総和	0.392	0.823	0.894	0.979	0.999	1.000	1.000		
枝の合計									4092

[図59]

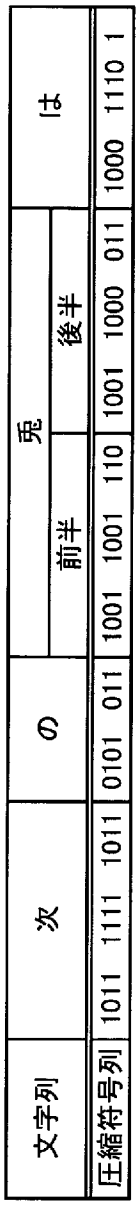
圧縮符号長	葉数	補正Aでの葉数	補正B¯1での葉数	補正B¯2での葉数	補正B¯3での葉数	ハフマン木の枝数	
						葉あたり	小計
5	2	2	0	0	0	64	0
6	9	9	11	0	0	32	0
7	22	22	22	33	25	16	400
8	28	28	28	28	36	8	288
9	29	29	29	29	29	4	116
10	25	25	25	25	25	2	50
11	86	1190	1190	1190	1190	1	1190
12	162						
13	251						
14	407						
15	279						
16	4						
17	1						
葉の合計	1305	1305	1305	1305	1305		
生起確率総和	0.711	1.146	1.115	1.029	0.998		
枝の合計							2044

[図60]



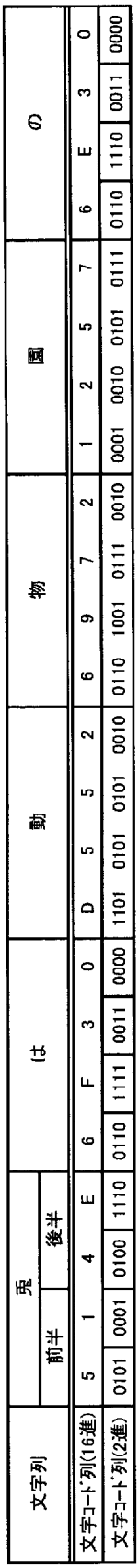
文字列	次				の					兎				は			
										前半		後半					
文字ｺｰﾄﾞ列(16進)	2	1	6	B	6	E	3	0		5	1	4	E	6	F	3	0
文字ｺｰﾄﾞ列(2進)	0010	0001	0110	1011	0110	1110	0011	0000		0101	0001	0100	1110	0110	1111	0011	0000

[図61]



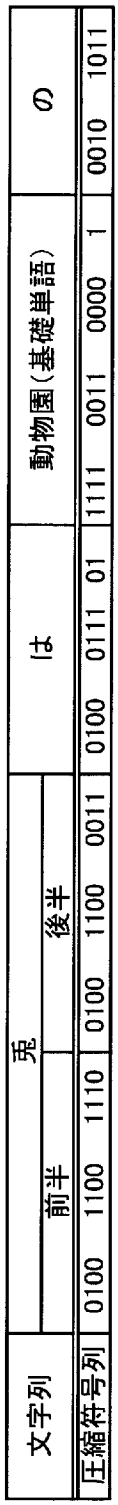
文字列	次				の		兎				は			
							前半		後半					
圧縮符号列	1011	1111	1011		0101	011	1001	1001	110	1001	1000	011	1000	1110 1

[図62]



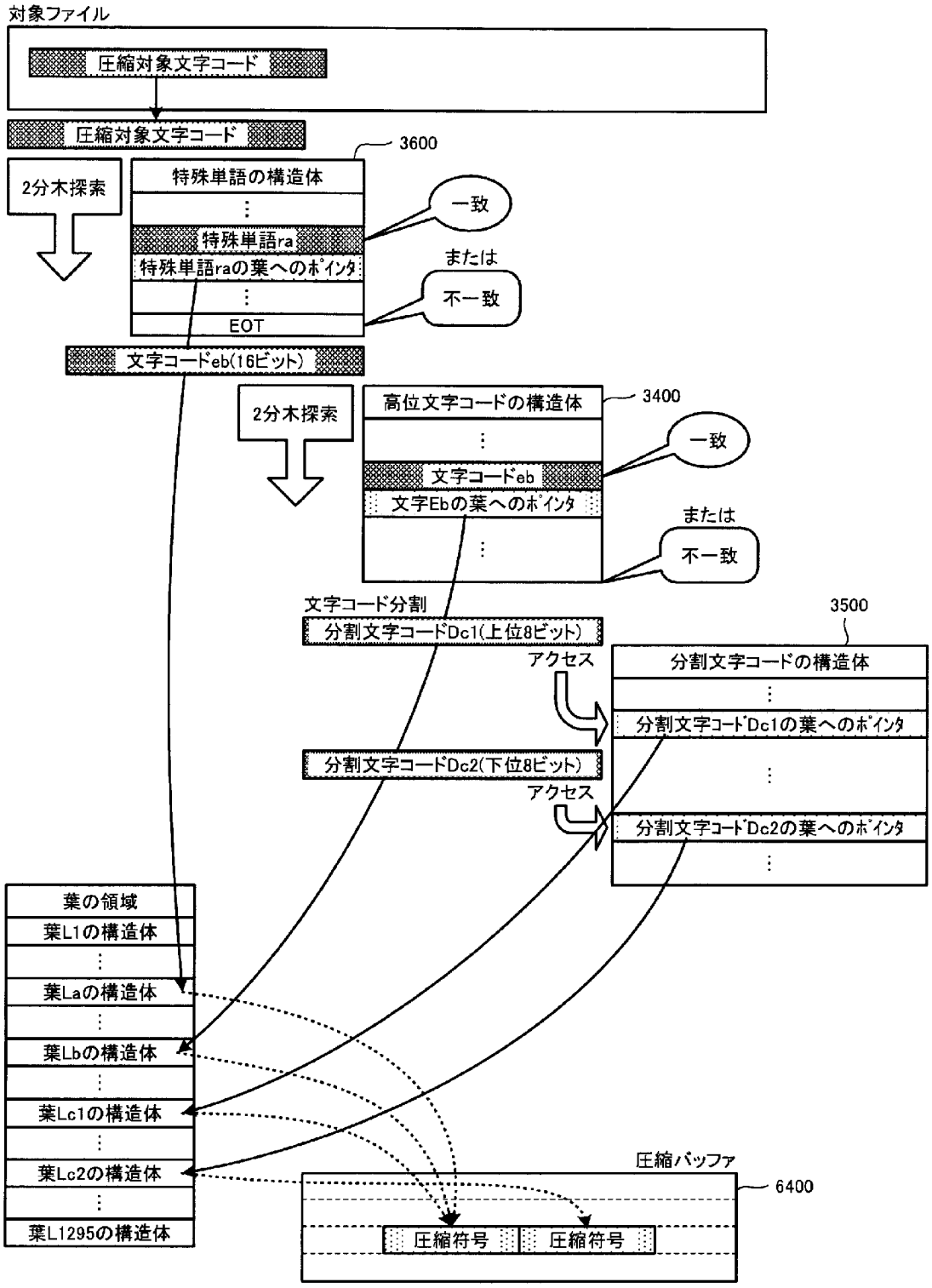
文字列	兎				は				動				物				園				の			
	前半				後半																			
文字ｺｰﾄﾞ列(16進)	5	1	4	E	6	F	3	0	D	5	5	2	6	9	7	2	1	2	5	7	6	E	3	0
文字ｺｰﾄﾞ列(2進)	0101	0001	0100	1110	0110	1111	0011	0000	1101	0101	0101	0010	0110	1001	0111	0010	0001	0010	0101	0111	0110	1110	0011	0000

[図63]

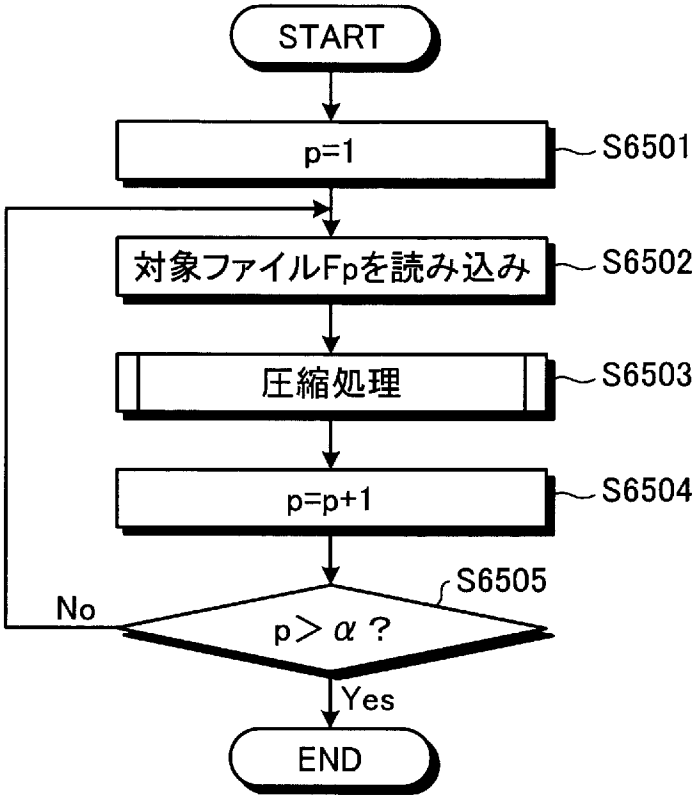


文字列	兎		は	動物園(基礎単語)		の
	前半	後半				
圧縮符号列	0100 1100 1110	0100 1100 0011	0100 0111 01	1111 0011 0000 1		0010 1011

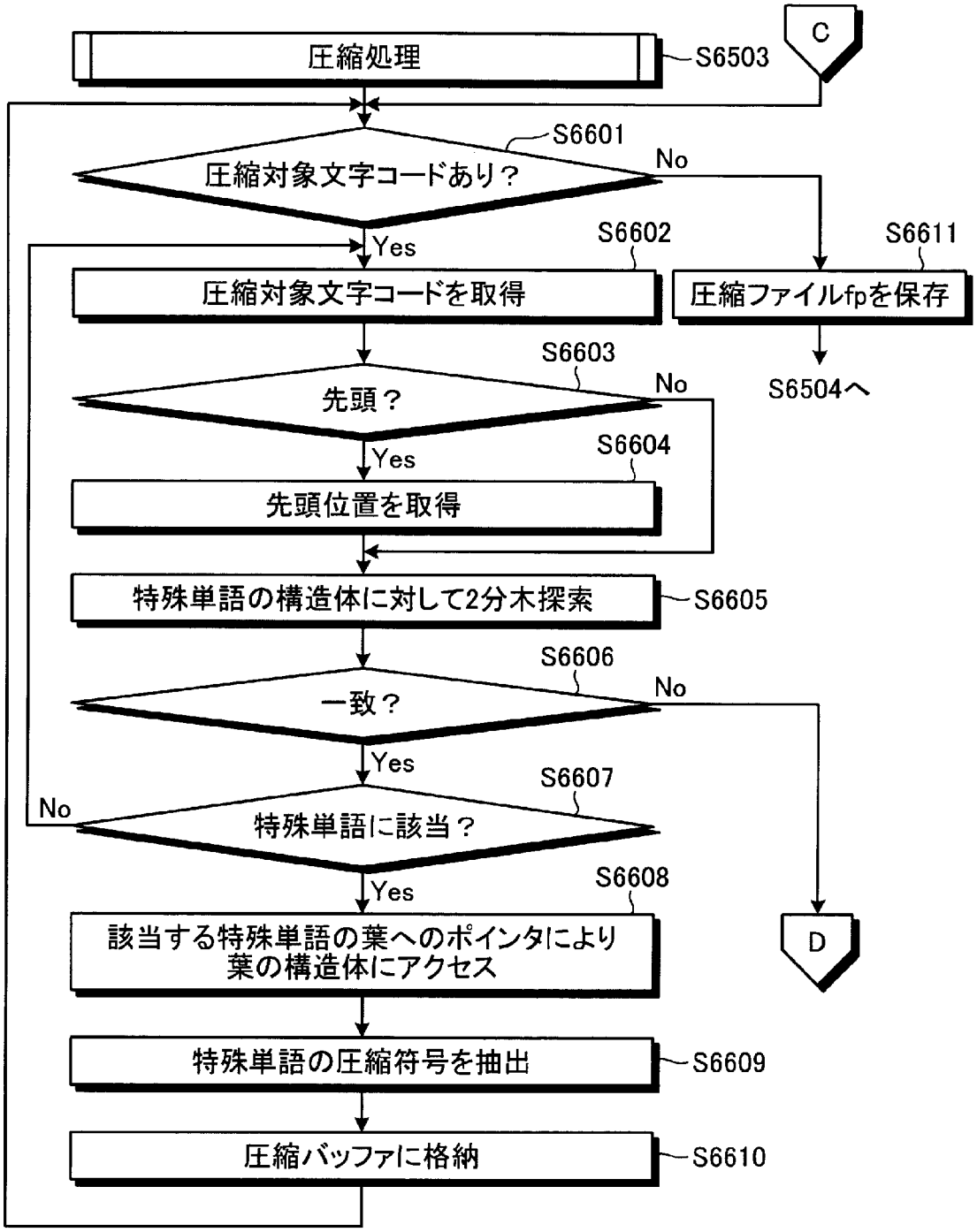
[図64]


対象ファイル
圧縮対象文字コード
圧縮対象文字コード
3600
2分木探索
特殊単語の構造体
一致
特殊単語ra
特殊単語raの葉へのポインタ
または
不一致
EOT
文字コードeb(16ビット)
2分木探索
高位文字コードの構造体
3400
一致
文字コードeb
文字Ebの葉へのポインタ
または
不一致
文字コード分割
分割文字コードDc1(上位8ビット)
3500
アクセス
分割文字コードの構造体
分割文字コードDc1の葉へのポインタ
分割文字コードDc2(下位8ビット)
アクセス
分割文字コードDc2の葉へのポインタ
葉の領域
葉L1の構造体
葉Laの構造体
葉Lbの構造体
葉Lc1の構造体
葉Lc2の構造体
葉L1295の構造体
圧縮バッファ
6400
圧縮符号
圧縮符号

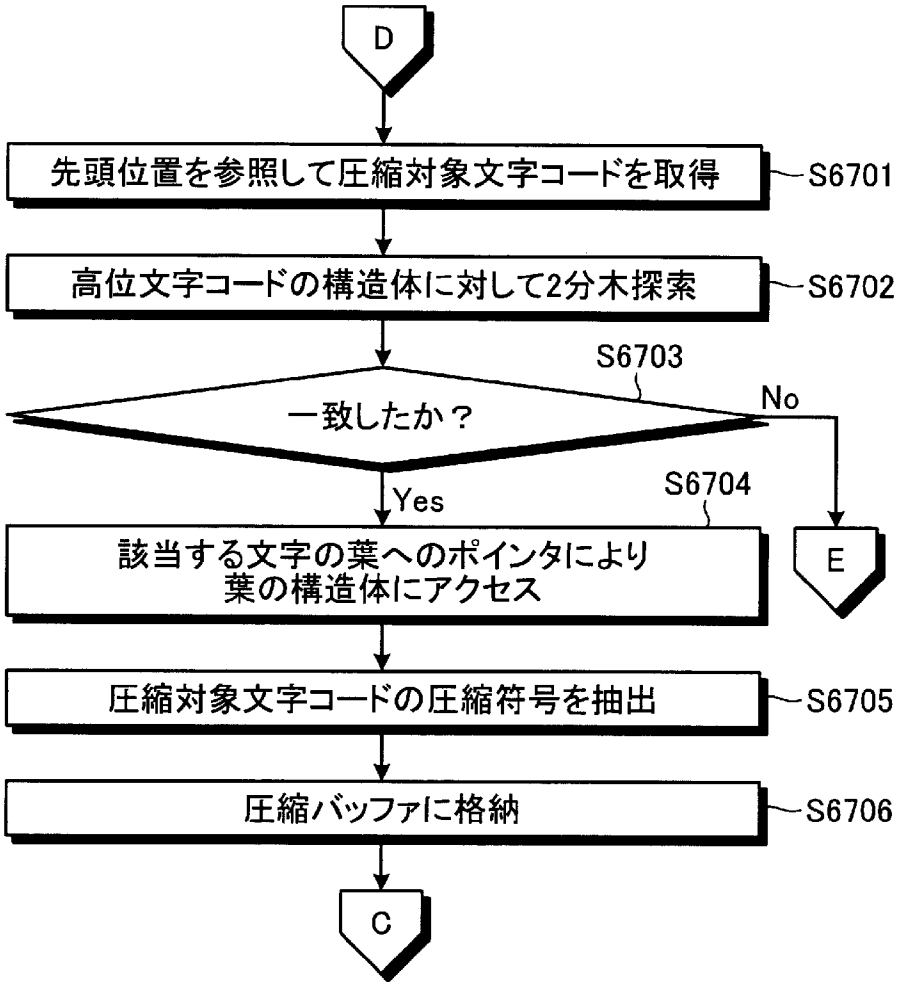
[図65]


START
p=1
S6501
対象ファイルFpを読み込み
S6502
圧縮処理
S6503
p=p+1
S6504
S6505
No
p＞α？
Yes
END

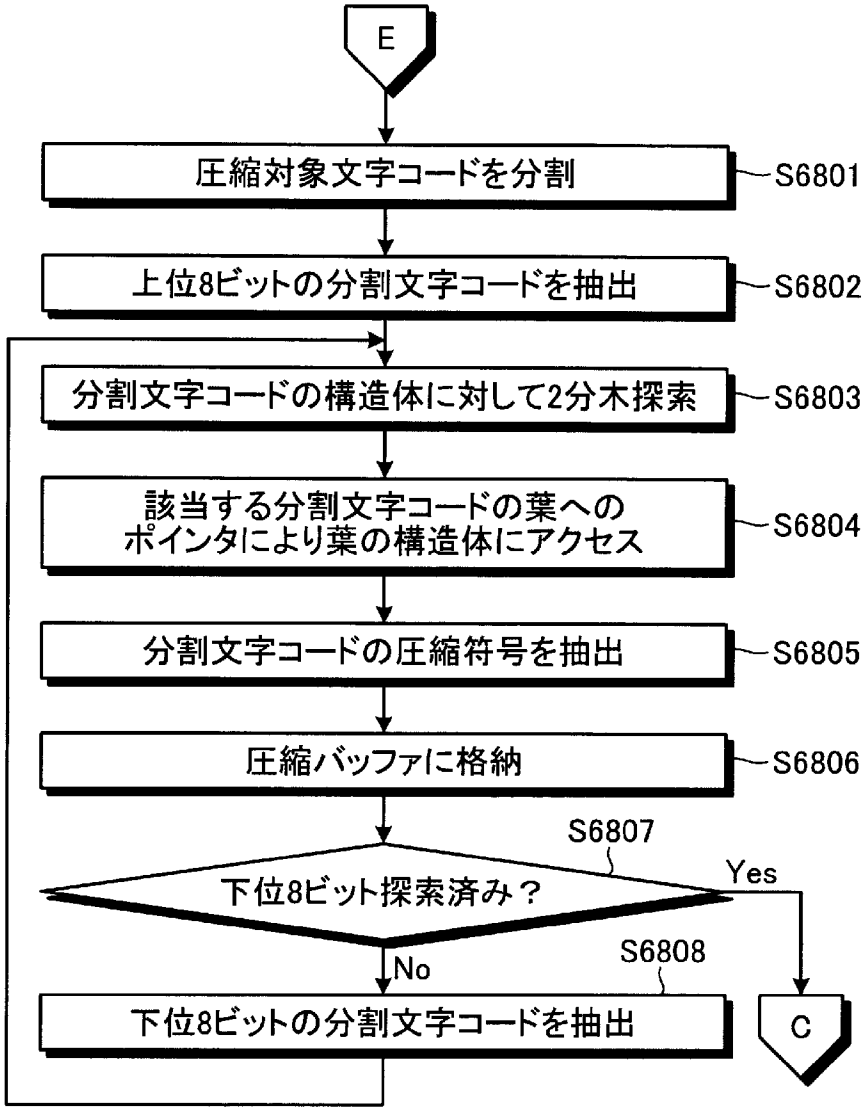
[図66]


圧縮処理
S6503
C
S6601
圧縮対象文字コードあり？
No
Yes
S6602
圧縮対象文字コードを取得
S6611
圧縮ファイルfpを保存
S6504へ
S6603
先頭？
No
Yes
S6604
先頭位置を取得
特殊単語の構造体に対して2分木探索
S6605
S6606
一致？
No
Yes
S6607
No
特殊単語に該当？
Yes
S6608
該当する特殊単語の葉へのポインタにより
葉の構造体にアクセス
D
特殊単語の圧縮符号を抽出
S6609
圧縮バッファに格納
S6610

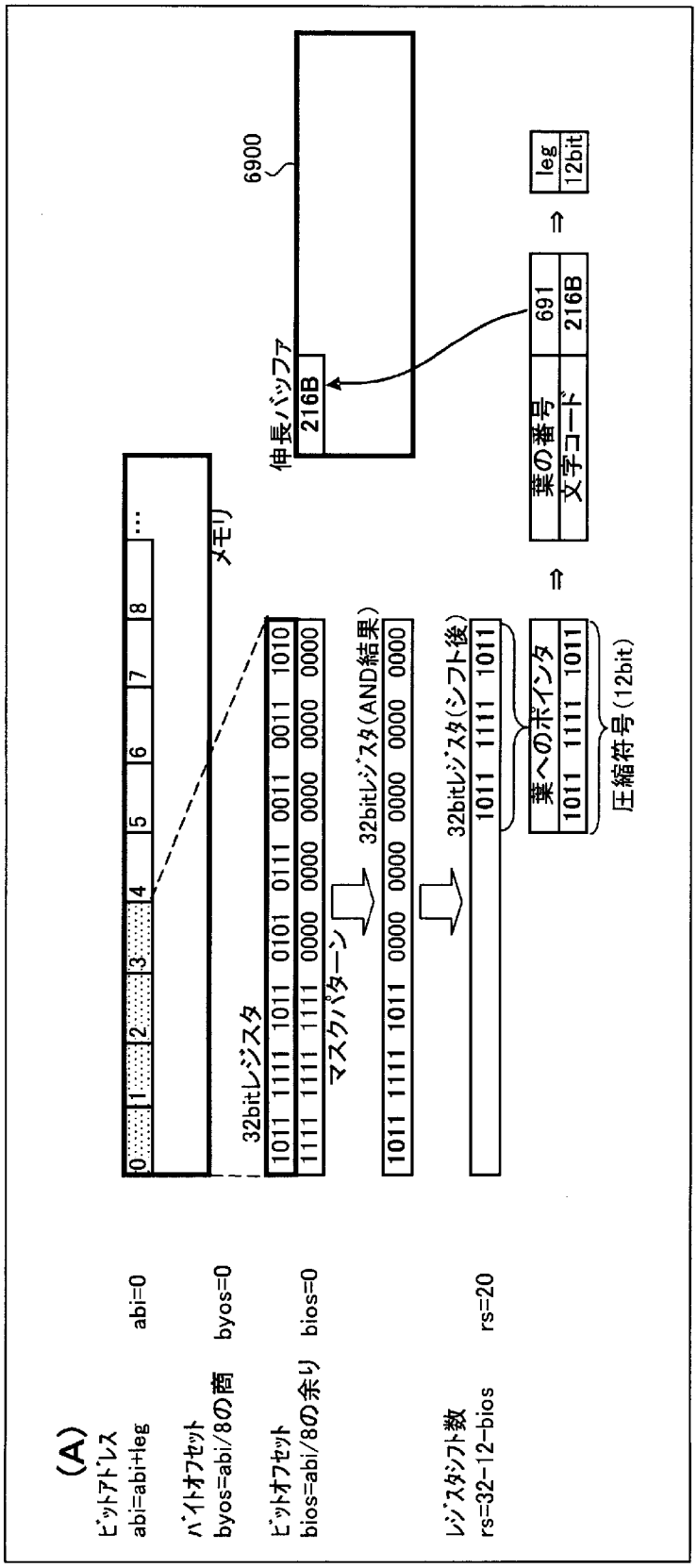
[図67]


D
先頭位置を参照して圧縮対象文字コードを取得
S6701
高位文字コードの構造体に対して2分木探索
S6702
S6703
一致したか？
No
Yes
S6704
該当する文字の葉へのポインタにより
葉の構造体にアクセス
E
圧縮対象文字コードの圧縮符号を抽出
S6705
圧縮バッファに格納
S6706
C

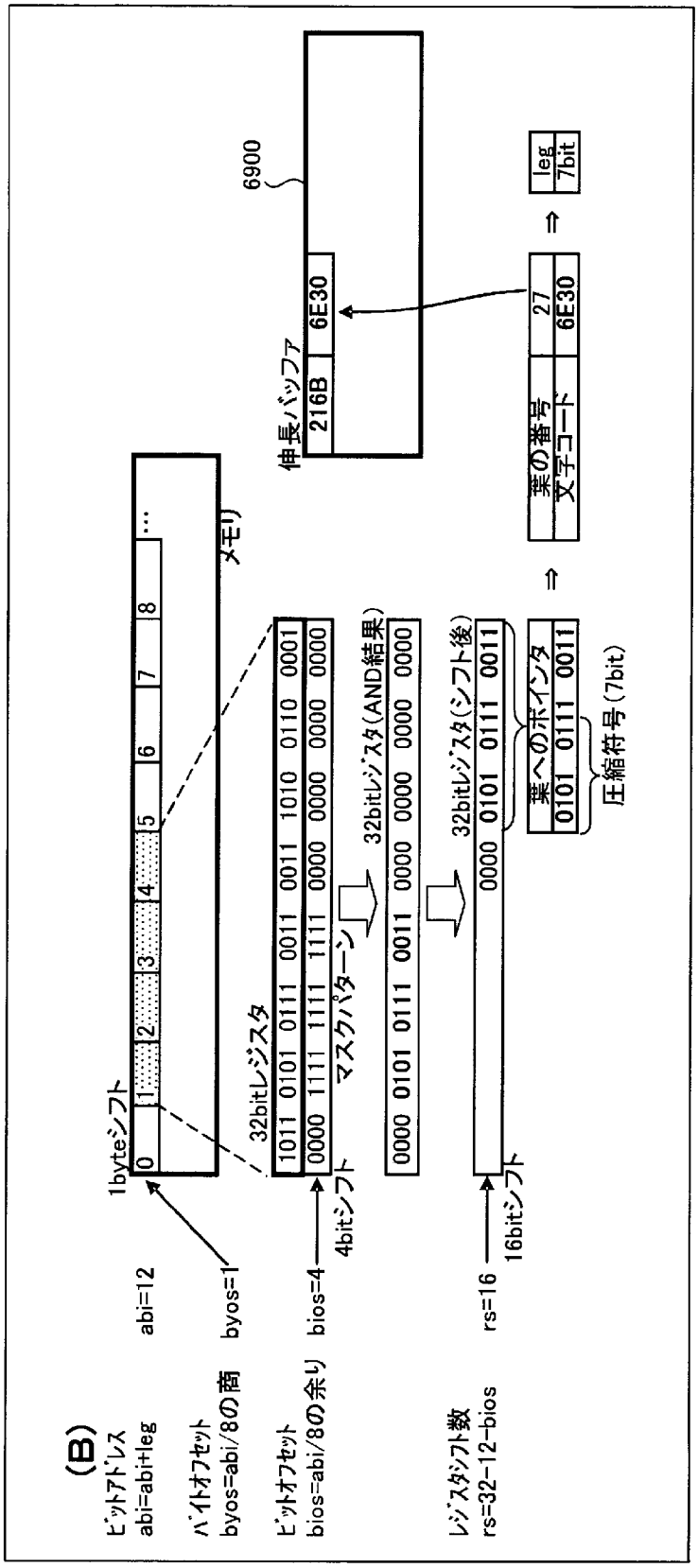
[図68]


E
圧縮対象文字コードを分割
S6801
上位8ビットの分割文字コードを抽出
S6802
分割文字コードの構造体に対して2分木探索
S6803
該当する分割文字コードの葉への
ポインタにより葉の構造体にアクセス
S6804
分割文字コードの圧縮符号を抽出
S6805
圧縮バッファに格納
S6806
S6807
下位8ビット探索済み？
Yes
No
S6808
下位8ビットの分割文字コードを抽出
C

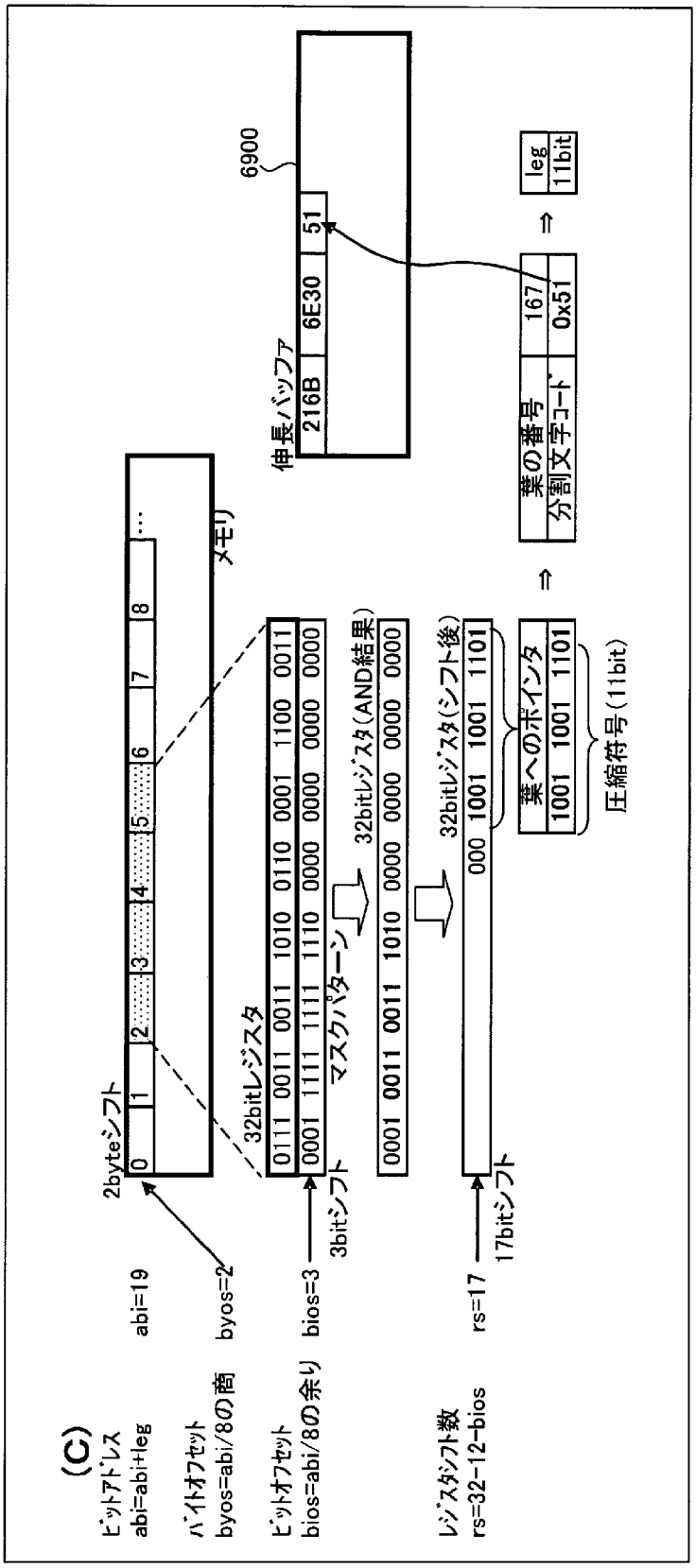
[図69]


(A)
ビットアドレス
abi=abi+leg
abi=0
バイトオフセット
byos=abi/8の商
byos=0
ビットオフセット
bios=abi/8の余り
bios=0
レジスタシフト数
rs=32-12-bios
rs=20
メモリ
0
1
2
3
4
5
6
7
8
…
32bitレジスタ
1011 1111 1011 0101 0111 0011 1010
1111 1111 1111 0000 0000 0000 0000
マスクパターン
32bitレジスタ(AND結果)
1011 1111 1011 0000 0000 0000 0000
32bitレジスタ(シフト後)
1011 1111 1011
葉へのポインタ
1011 1111 1011
圧縮符号(12bit)
6900
伸長バッファ
216B
葉の番号
文字コード
691
216B
leg
12bit

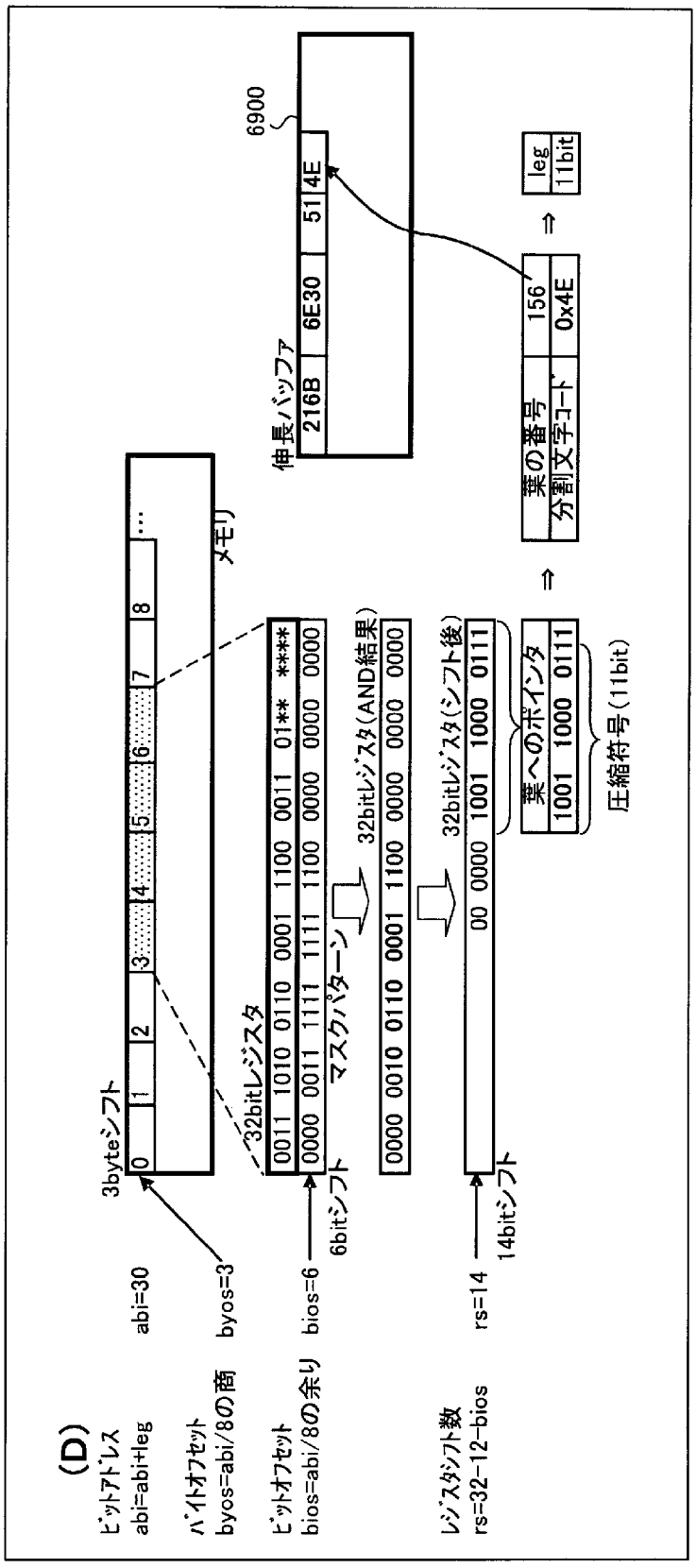
[図70]


(B)
ビットアドレス
abi=abi+leg
バイトオフセット
byos=abi/8の商
ビットオフセット
bios=abi/8の余り
レジスタシフト数
rs=32-12-bios
abi=12
byos=1
bios=4
rs=16
1byteシフト
メモリ
32bitレジスタ
4bitシフト
マスクパターン
32bitレジスタ(AND結果)
32bitレジスタ(シフト後)
16bitシフト
伸長バッファ
6900
葉へのポインタ
葉の番号
文字コード
圧縮符号(7bit)
leg
7bit

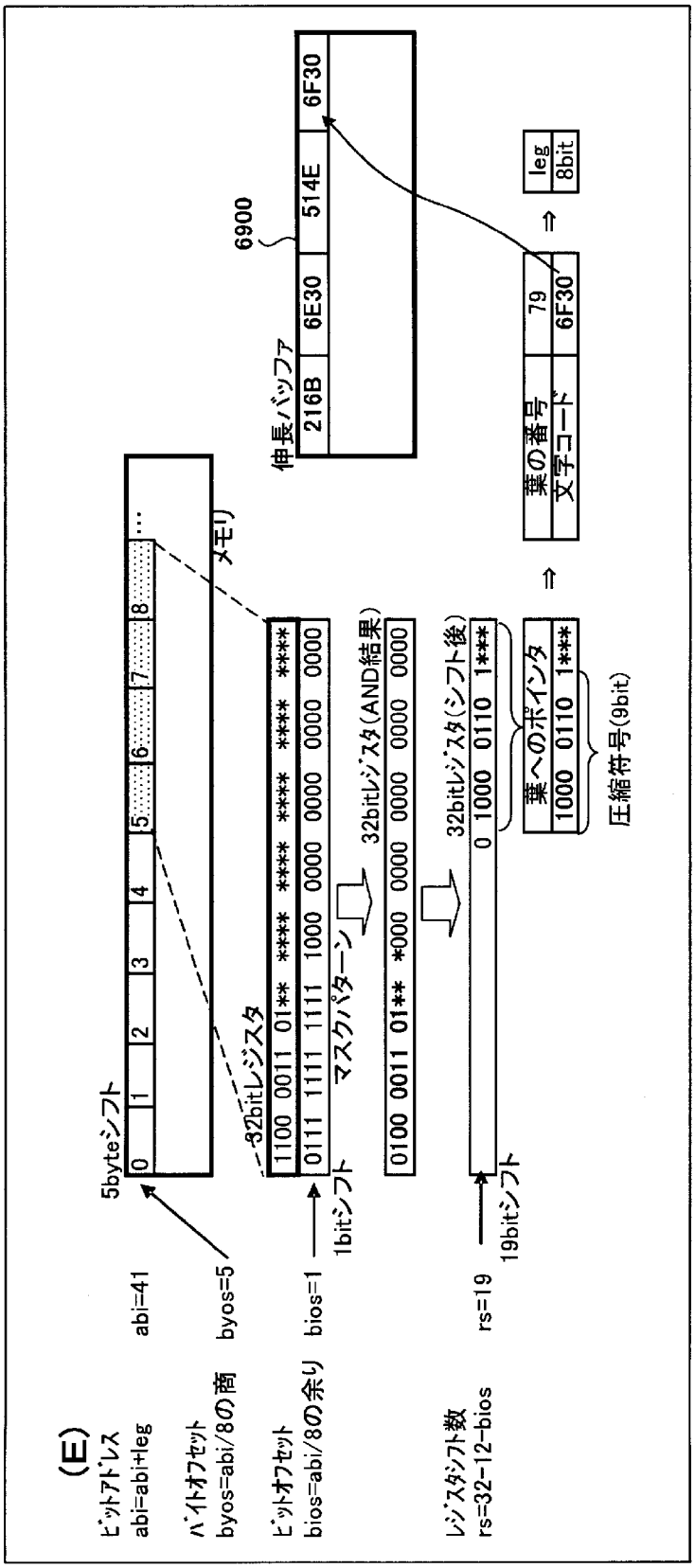
[図71]


(C)
ビットアドレス
abi=abi+leg
abi=19
バイトオフセット
byos=abi/8の商
byos=2
ビットオフセット
bios=abi/8の余り
bios=3
レジスタシフト数
rs=32-12-bios
rs=17
2byteシフト
メモリ
0
1
2
3
4
5
6
7
8
32bitレジスタ
0111 0011 0011 1010 0110 0001 1100 0011
0001 1111 1111 1110 0000 0000 0000 0000
3bitシフト
マスクパターン
32bitレジスタ(AND結果)
0001 0011 0011 1010 0000 0000 0000 0000
32bitレジスタ(シフト後)
000 1001 1001 1101
17bitシフト
葉へのポインタ
1001 1001 1101
圧縮符号(11bit)
伸長バッファ
6900
216B
6E30
51
葉の番号
167
分割文字コード
0x51
leg
11bit

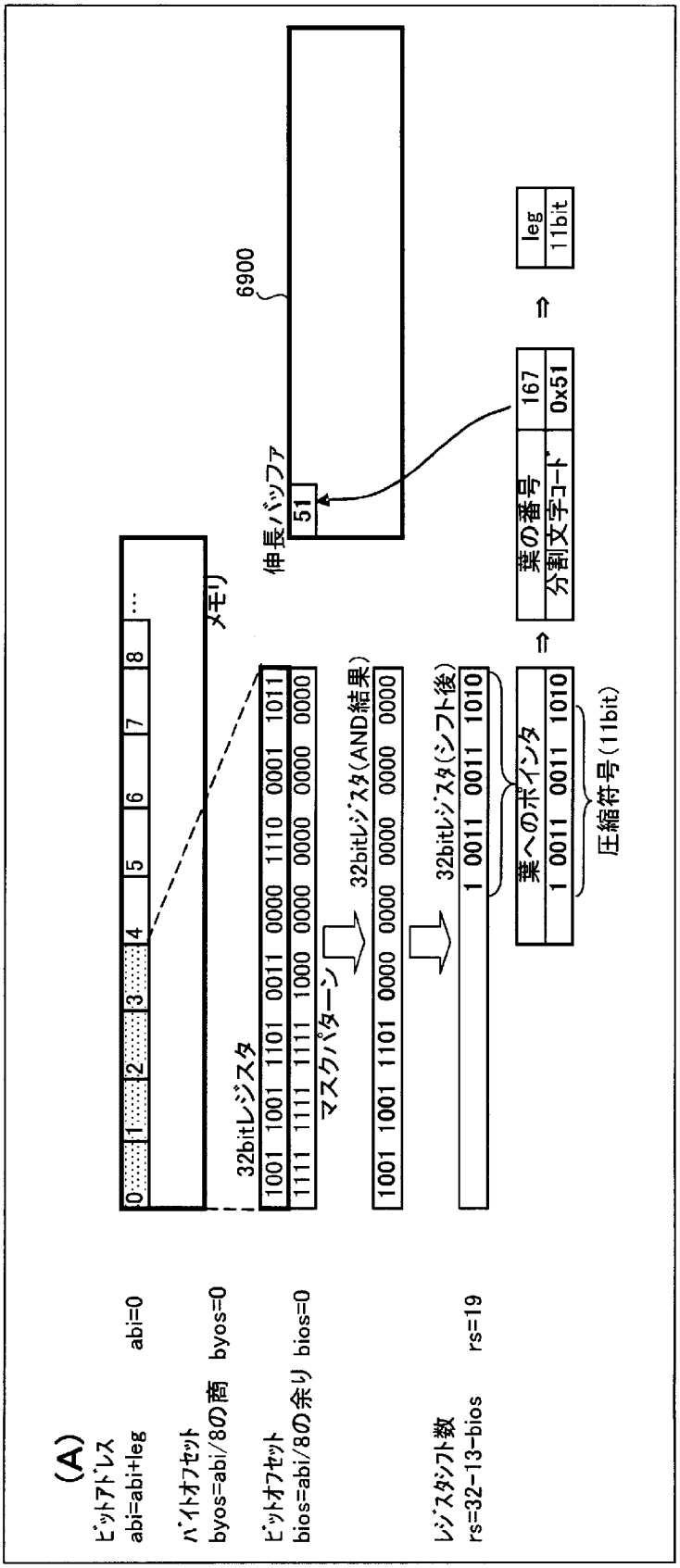
[図72]


(D)
ビットアドレス
abi=abi+leg
バイトオフセット
byos=abi/8の商
ビットオフセット
bios=abi/8の余り
レジスタ数
rs=32-12-bios
abi=30
byos=3
bios=6
rs=14
3byteシフト
メモリ
32bitレジスタ
マスクパターン
6bitシフト
32bitレジスタ(AND結果)
32bitレジスタ(シフト後)
14bitシフト
伸長バッファ
6900
216B
6E30
514E
葉の番号
156
分割文字コード
0x4E
葉へのポインタ
1001 1000 0111
圧縮符号(11bit)
leg
11bit

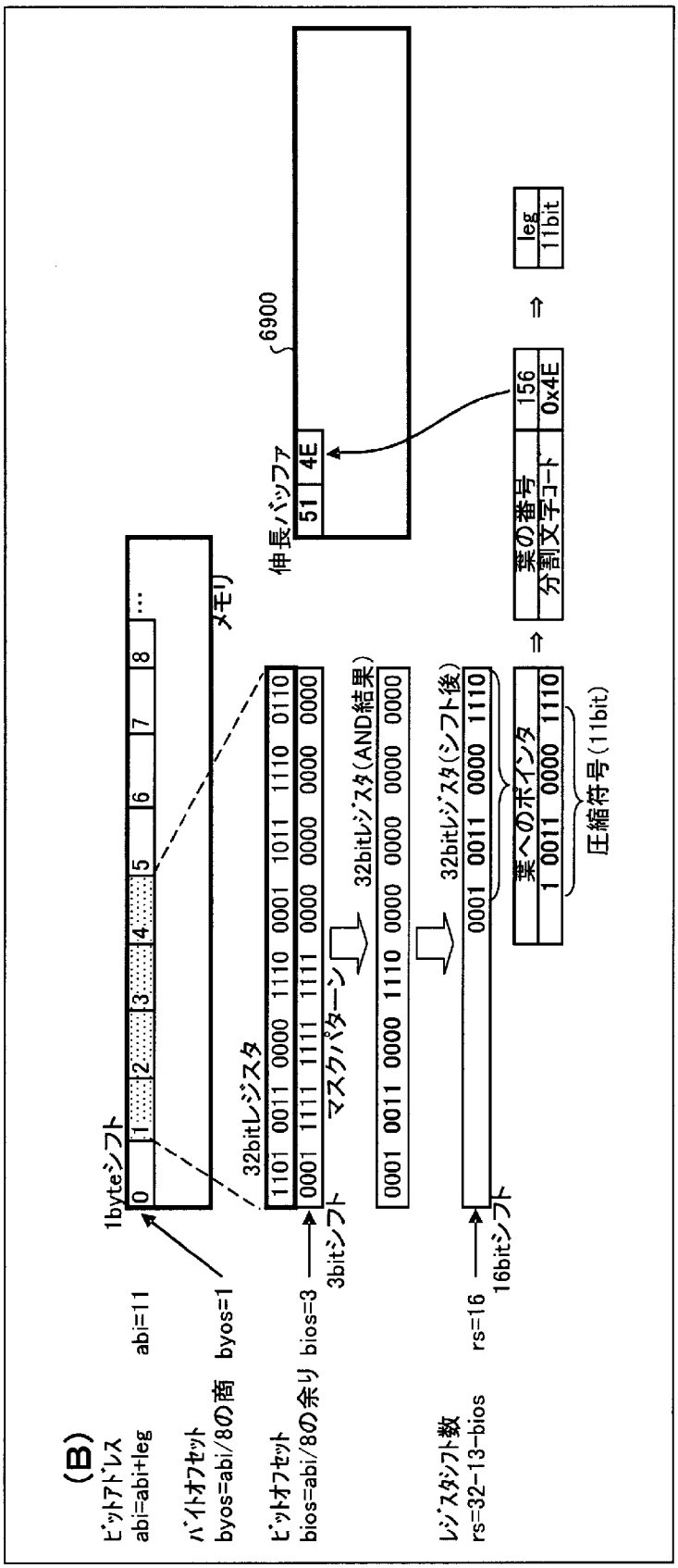
[図73]


(E)
ビットアドレス
abi=abi+leg
バイトオフセット
byos=abi/8の商
ビットオフセット
bios=abi/8の余り
レジスタシフト数
rs=32-12-bios
abi=41
byos=5
bios=1
rs=19
5byteシフト
メモリ
32bitレジスタ
1bitシフト
マスクパターン
32bitレジスタ(AND結果)
19bitシフト
32bitレジスタ(シフト後)
伸長バッファ
6900
葉へのポインタ
葉の番号
文字コード
leg
8bit
圧縮符号(9bit)

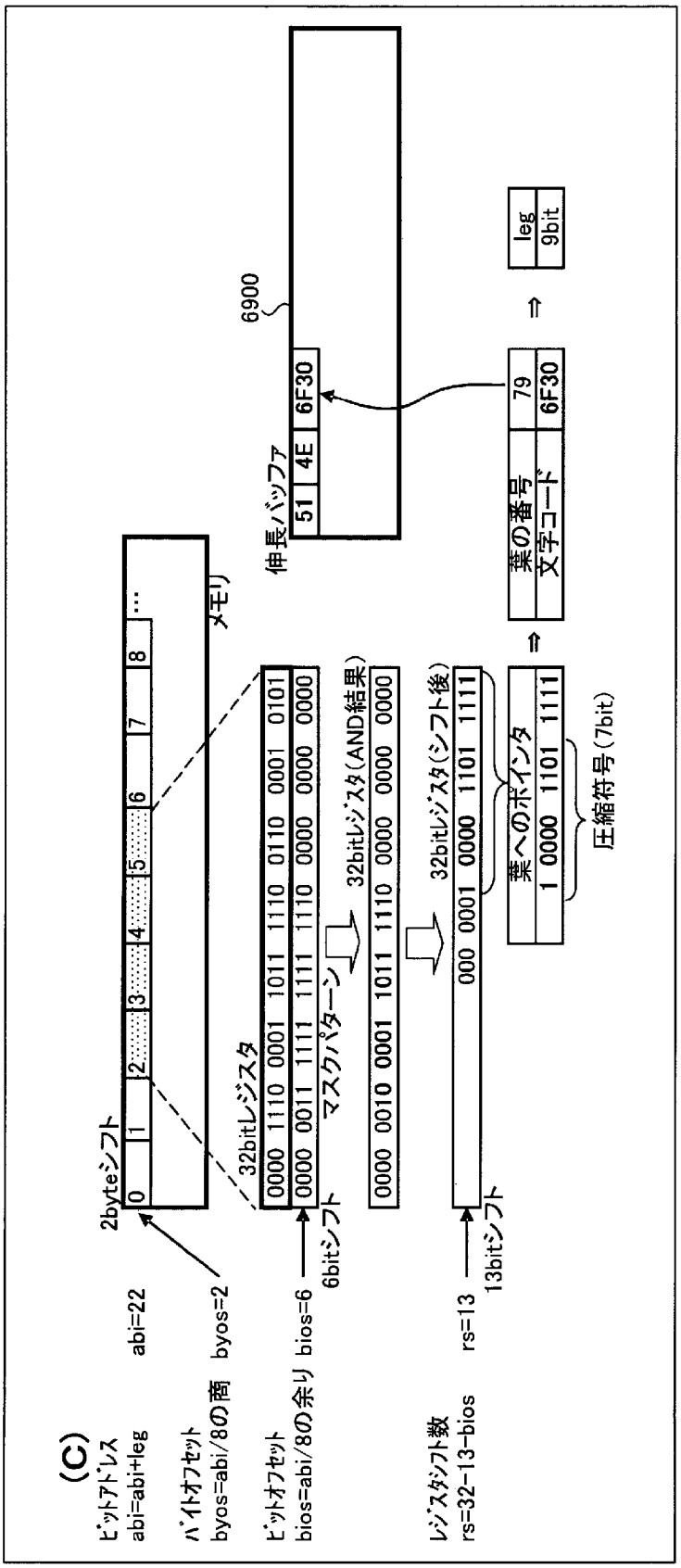
[図74]


(A)
ビットアドレス
abi=abi+leg
abi=0
バイトオフセット
byos=abi/8の商
byos=0
ビットオフセット
bios=abi/8の余り
bios=0
レジスタシフト数
rs=32-13-bios
rs=19
メモリ
0
1
2
3
4
5
6
7
8
32bitレジスタ
1001 1001 1101 0011 0000 1110 0001 1011
1111 1111 1111 1000 0000 0000 0000 0000
マスクパターン
32bitレジスタ(AND結果)
1001 1001 1101 0000 0000 0000 0000 0000
32bitレジスタ(シフト後)
1 0011 0011 1010
葉へのポインタ
1 0011 0011 1010
圧縮符号(11bit)
6900
伸長バッファ
51
葉の番号
167
分割文字コード
0x51
leg
11bit

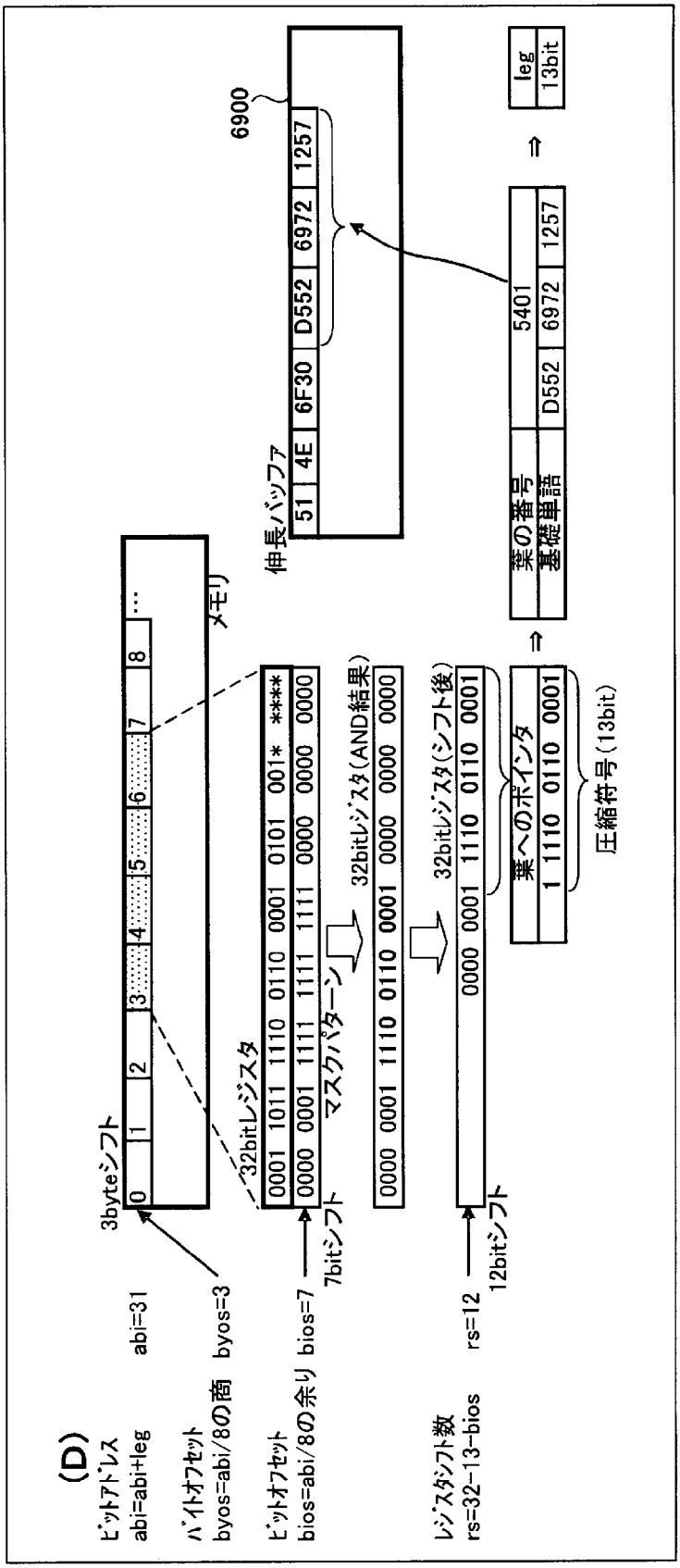
[図75]


(B)
ビットアドレス
abi=abi+leg
abi=11
バイトオフセット
byos=abi/8の商
byos=1
ビットオフセット
bios=abi/8の余り
bios=3
レジスタシフト数
rs=32-13-bios
rs=16
1byteシフト
メモリ
32bitレジスタ
1101 0011 0000 1110 0001 1011 1110 0110
0001 1111 1111 1111 0000 0000 0000 0000
3bitシフト
マスクパターン
32bitレジスタ(AND結果)
0001 0011 0000 1110 0000 0000 0000 0000
32bitレジスタ(シフト後)
0001 0011 0000 1110
16bitシフト
葉へのポインタ
1 0011 0000 1110
圧縮符号(11bit)
伸長バッファ
6900
51 4E
葉の番号
分割文字コード
156
0x4E
leg
11bit

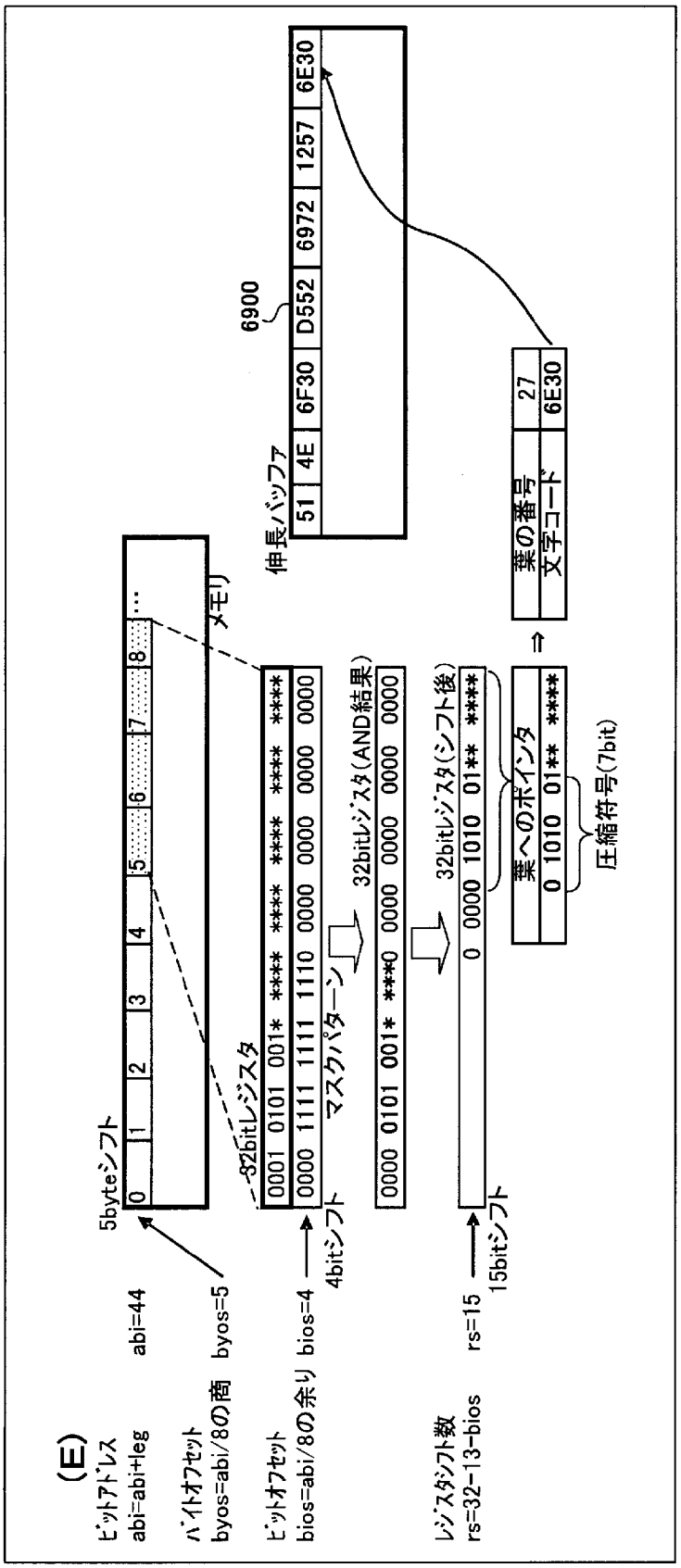
[図76]


(C)
ビットアドレス
abi=abi+leg
バイトオフセット
byos=abi/8の商
ビットオフセット
bios=abi/8の余り
レジスタシフト数
rs=32-13-bios
abi=22
byos=2
bios=6
rs=13
2byteシフト
メモリ
0
1
2
3
4
5
6
7
8
32bitレジスタ
0000 1110 0001 1011 1110 0110 0001 0101
マスクパターン
0000 0011 1111 1111 1110 0000 0000 0000
6bitシフト
32bitレジスタ(AND結果)
0000 0010 0001 1011 1110 0000 0000 0000
32bitレジスタ(シフト後)
000 0001 0000 1101 1111
13bitシフト
葉へのポインタ
1 0000 1101 1111
圧縮符号(7bit)
葉の番号
文字コード
79
6F30
leg
9bit
伸長バッファ
51
4E
6F30
6900

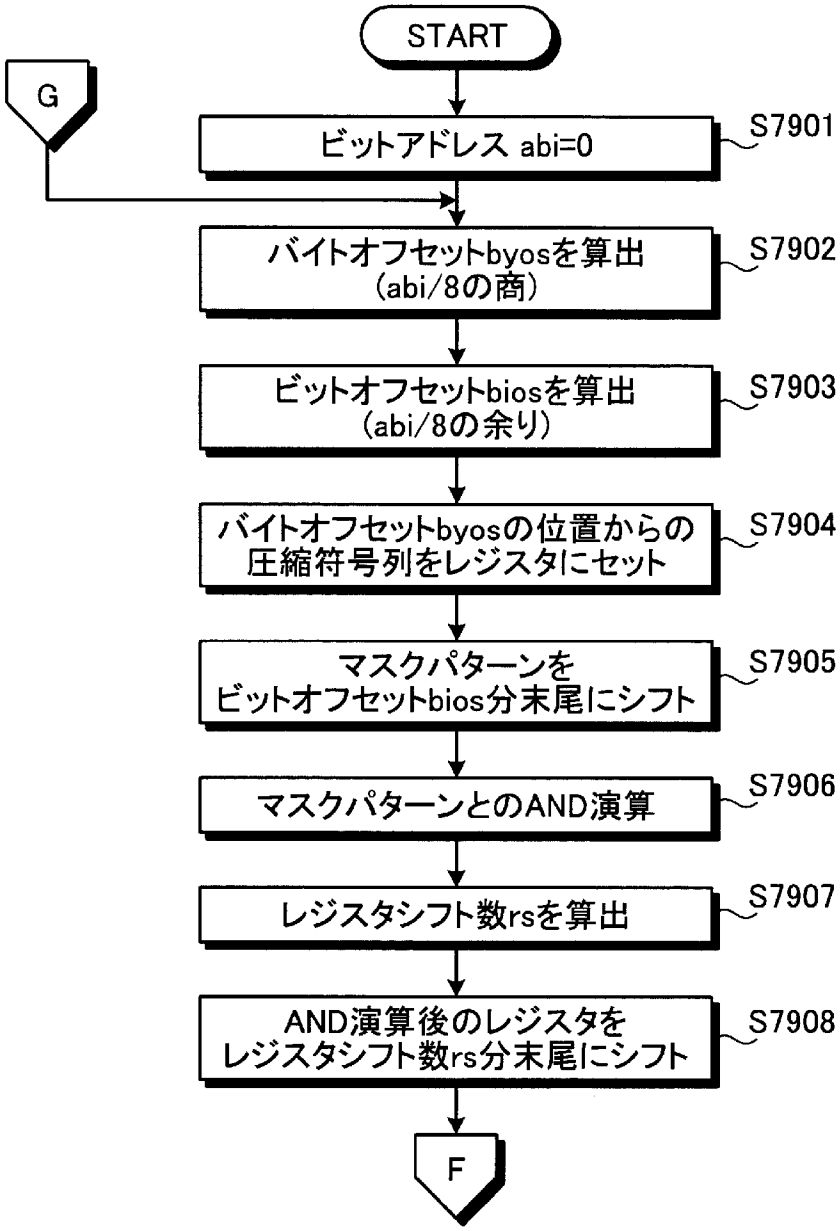
[図77]


(D)
ビットアドレス
abi=abi+leg
バイトオフセット
byos=abi/8の商
ビットオフセット
bios=abi/8の余り
レジスタシフト数
rs=32−13−bios
abi=31
byos=3
bios=7
rs=12
3byteシフト
メモリ
32bitレジスタ
0001 1011 1110 0110 0001 0101 001* ****
0000 0001 1111 1111 1111 0000 0000 0000
7bitシフト
マスクパターン
32bitレジスタ(AND結果)
0000 0001 1110 0110 0001 0000 0000 0000
32bitレジスタ(シフト後)
0000 0001 1110 0110 0001
12bitシフト
葉へのポインタ
1 1110 0110 0001
圧縮符号(13bit)
6900
伸長バッファ
51 4E 6F30 D552 6972 1257
葉の番号
基礎単語
5401
D552 6972 1257
leg
13bit

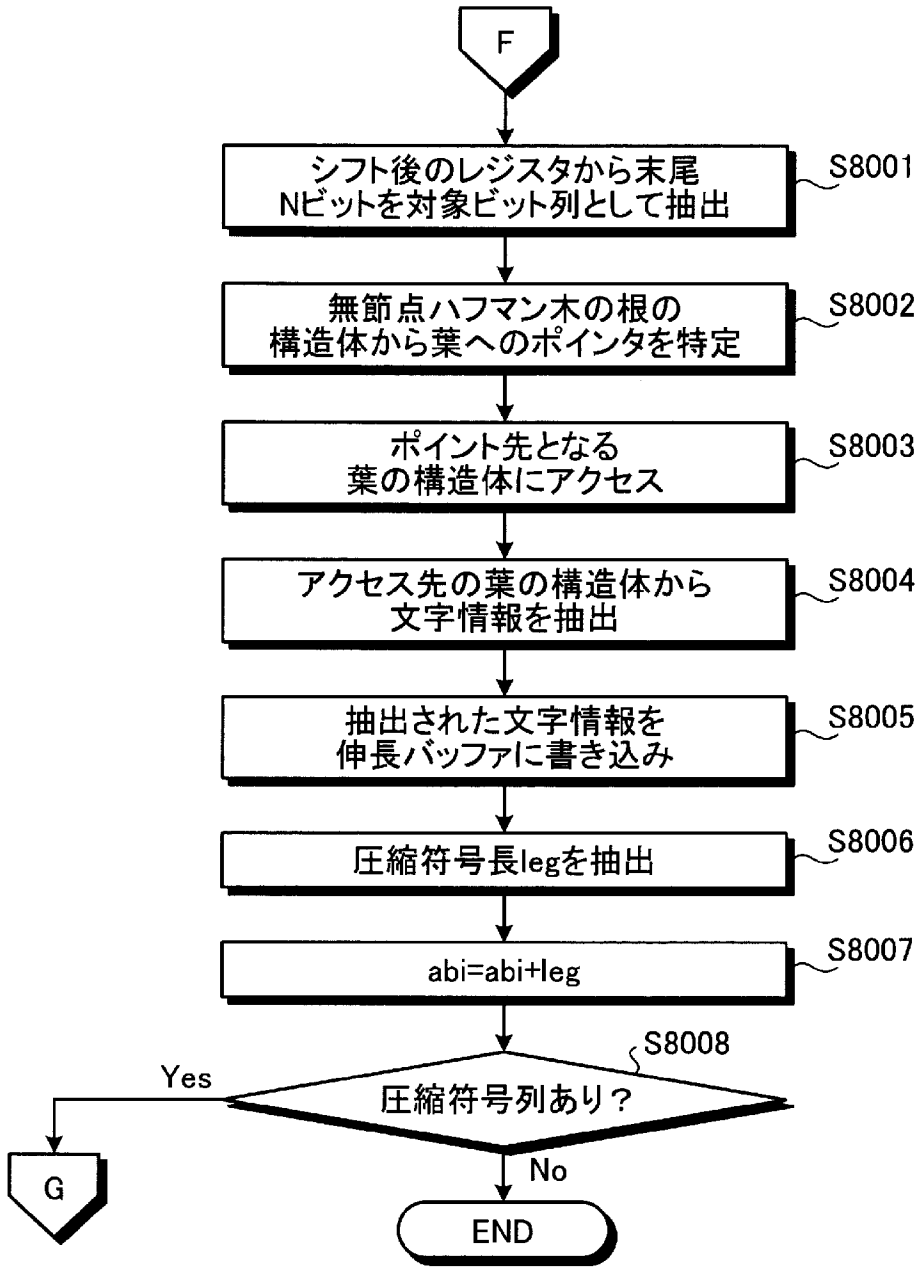
[図78]


(E)
ビットアドレス
abi=abi+leg
abi=44
バイトオフセット
byos=abi/8の商
byos=5
ビットオフセット
bios=abi/8の余り
bios=4
レジスタシフト数
rs=32−13−bios
rs=15
5byteシフト
メモリ
32bitレジスタ
0001 0101 001* **** **** **** ****
0000 1111 1111 1110 0000 0000 0000
マスクパターン
4bitシフト
32bitレジスタ(AND結果)
0000 0101 001* ***0 0000 0000 0000
15bitシフト
32bitレジスタ(シフト後)
0 0000 1010 01** ****
葉へのポインタ
0 1010 01** ****
圧縮符号(7bit)
⇒
葉の番号
文字コード
27
6E30
伸長バッファ
6900
51 4E 6F30 D552 6972 1257 6E30

[図79]


START
G
ビットアドレス abi=0
S7901
バイトオフセットbyosを算出
(abi/8の商)
S7902
ビットオフセットbiosを算出
(abi/8の余り)
S7903
バイトオフセットbyosの位置からの
圧縮符号列をレジスタにセット
S7904
マスクパターンを
ビットオフセットbios分末尾にシフト
S7905
マスクパターンとのAND演算
S7906
レジスタシフト数rsを算出
S7907
AND演算後のレジスタを
レジスタシフト数rs分末尾にシフト
S7908
F

[図80]


F
シフト後のレジスタから末尾
Nビットを対象ビット列として抽出
S8001
無節点ハフマン木の根の
構造体から葉へのポインタを特定
S8002
ポイント先となる
葉の構造体にアクセス
S8003
アクセス先の葉の構造体から
文字情報を抽出
S8004
抽出された文字情報を
伸長バッファに書き込み
S8005
圧縮符号長legを抽出
S8006
abi=abi+leg
S8007
S8008
圧縮符号列あり？
Yes
No
G
END

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2011/053059

A. CLASSIFICATION OF SUBJECT MATTER
H03M7/40(2006.01)i, *G06F5/00*(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H03M7/40, G06F5/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Jitsuyo Shinan Toroku Koho	1996-2011
Kokai Jitsuyo Shinan Koho	1971-2011	Toroku Jitsuyo Shinan Koho	1994-2011

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2010-93414 A (Fujitsu Ltd.), 22 April 2010 (22.04.2010), entire text; all drawings & US 2010/0085222 A1	1-11
A	JP 6-131151 A (Sony Corp.), 13 May 1994 (13.05.1994), entire text; fig. 1, 3 to 7 (Family: none)	1-11
A	Michael B.Baer, On Conditional Branches in Optimal Decision Trees, Information Theory, 2007. ISIT 2007.IEEE International Symposium on, 2007.06, pp.436-440	1-11

☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 April, 2011 (19.04.11)	10 May, 2011 (10.05.11)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2011/053059

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2002-278748 A (Sony Corp.), 27 September 2002 (27.09.2002), entire text; all drawings & US 6650256 B2	1-11
A	WO 2008/146756 A1 (Fujitsu Ltd.), 04 December 2008 (04.12.2008), entire text; all drawings & US 2010/0131476 A	1-11

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２０１１／０５３０５９

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl.　H03M7/40(2006.01)i, G06F5/00(2006.01)i

Ｂ．　調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Int.Cl.　H03M7/40, G06F5/00

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２０１１年
日本国実用新案登録公報	１９９６－２０１１年
日本国登録実用新案公報	１９９４－２０１１年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

Ｃ．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
A	JP 2010-93414 A (富士通株式会社) 2010.04.22, 全文, 全図 & US 2010/0085222 A1	1-11
A	JP 6-131151 A (ソニー株式会社) 1994.05.13, 全文, 図１, ３－７ (ファミリーなし)	1-11
A	Michael B.Baer, On Conditional Branches in Optimal Decision Trees, Information Theory,2007. ISIT 2007.IEEE International Symposium on, 2007.06, pp.436-440	1-11

☑　Ｃ欄の続きにも文献が列挙されている。　　☐　パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの
「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「Ｏ」口頭による開示、使用、展示等に言及する文献
「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「＆」同一パテントファミリー文献

国際調査を完了した日 １９．０４．２０１１	国際調査報告の発送日 １０．０５．２０１１
国際調査機関の名称及びあて先 日本国特許庁（ＩＳＡ／ＪＰ） 郵便番号１００－８９１５ 東京都千代田区霞が関三丁目４番３号	特許庁審査官（権限のある職員）　５Ｋ　９２９７ 北村　智彦 電話番号　０３－３５８１－１１０１　内線　３５５６

Ｃ（続き）．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
A	JP 2002-278748 A (ソニー株式会社) 2002.09.27, 全文, 全図 & US 6650256 B2	1-11
A	WO 2008/146756 A1 (富士通株式会社) 2008.12.04, 全文, 全図 & US 2010/0131476 A	1-11