

明 細 書

発明の名称：データ圧縮装置、方法およびプログラム

技術分野

[0001] 本発明は、データ圧縮装置、方法およびプログラムに関するものである。

背景技術

[0002] データ圧縮アルゴリズムとして、1977年にAbraham Lemp

el氏とJacob Ziv氏が発表したLZ77方式が知られている。L

Z77方式は辞書式圧縮法の一つで、スライド辞書法とも呼ばれ、データを

先頭から順番に符号化していく。スライド辞書法は、現在注目している位置

から始まるシンボル列が、それ以前に出現していたかを探す。もし出現して

いたならば、シンボル列をその出現位置と長さのポインタに置き換える。ス

ライド辞書法の名前は、シンボル列を探す範囲をスライド窓と呼び、これを

辞書として使用することに由来する。

[0003] LZ77方式では、繰返し出現するデータ列を過去に出現したシンボル列

の長さとして出現位置に置き換えることでデータ量を削減する。復元の際には、

過去の位置から一致長分のシンボル列をコピーすることで元のデータに復元

できる。また、繰返しが見つからなかったときに、見つからなかった長さと

見つからなかったシンボル列を符号化する方法が示されている。

[0004] また、中央処理装置 (Central Processing Unit

、CPU) のオペレーティングシステム (OS) によっては、1命令で複数

データを扱うSingle Instruction Multiple

Data (SIMD) 命令が用意されており、16バイト、32バイト単位

でのメモリ間のデータコピーを高速に行うことができる。SIMD命令を復

元時のコピー処理において用いると復元処理の高速化を実現することができる。

先行技術文献

非特許文献

- [0005] 非特許文献1: F i a l a, E. and G r e e n e, D. (1989)
“Data compression with finite windows” Communications of the ACM、32 (4)
pp. 490-505

発明の概要

発明が解決しようとする課題

- [0006] LZ77方式では、コピー元の文字列の最後尾のアドレスがコピー先の文字列の先頭のアドレスより後方になることがある。このことを以下では、単に「コピー元とコピー先が重なる」という。たとえば、同じ文字が「aaaa…」と続くケースだと、コピー元の文字列はアドレス「1,2,3,4…」で、コピー先の文字列はアドレス「2,3,4,…」となり、アドレス「2,3,4,…」において重なりを持つ。このようなケースでは、復元処理において、ステップ1で、第1バイトを第2バイトにコピーし、ステップ2で第2バイトを第3バイトにコピーする、という手順でそれ以前に出現した文字列をコピーする。この場合、ステップ1の処理が終わっていないとステップ2の処理を行うことができない。つまり、LZ77方式は、繰返し出現するデータ列を過去に出現したデータ列の長さとお出現位置に置き換えているため、単純なコピー処理で復元できるにも関わらず、コピー元とコピー先が重なっているときは、1バイトずつのコピー処理が必要となることがある。このため、復元時に高速な多バイトコピーを使用するには、まずコピー元とコピー先が重なるか否か確認する必要があり、また、重なっている場合は、1バイトずつのコピーを行う煩雑な処理が必要であるという問題がある。

- [0007] よって、本発明は、多バイトコピーを用いて復元を行う上で不都合な重なりを制限した一致列を選択して圧縮することで、高速に復元できる圧縮データを生成するデータ圧縮装置、方法およびプログラム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

- [0008] データ圧縮装置が提供される。データ圧縮装置は、圧縮前のデータの文字

列中の第 1 の文字列の後に、前記第 1 の文字列と同一である第 2 の文字列を一致列として抽出し、前記一致列の長さ、前記第 1 の文字列が前記第 2 の文字列の幾つ前のアドレスの位置から開始されるかを示す相対位置を決定する一致列検索部と、前記一致列検索部で抽出された前記第 2 の文字列から、長さが前記相対位置より小さい第 3 の文字列を抽出する一致列分割部と、前記第 2 または第 3 の文字列の長さを符号化する一致長符号化部と、前記相対位置を符号化する一致位置符号化部と、を含むことを特徴とする。

発明の効果

[0009] 多バイト単位で圧縮データの復元を実現するために、元データ中で繰り返し出現する一致列を再分割してから圧縮することで、高速に復元できる圧縮データを生成することができる。

図面の簡単な説明

- [0010] [図1] L Z 7 7 方式の概略を説明する図である。
- [図2] 比較例におけるデータ圧縮装置の機能ブロック図の例を示す図である。
- [図3] 比較例におけるデータ圧縮装置での圧縮処理の流れの例を示す図である。
- [図4] 比較例における元データと圧縮データの例を示す図である。
- [図5] 比較例における復元処理の流れの例を示す図である。
- [図6A] 比較例におけるデータ圧縮装置での圧縮処理の例を示す図である。
- [図6B] 比較例における復元処理の例を示す図である。
- [図7] 実施形態におけるデータ圧縮装置の機能ブロック図の例を示す図である。
- [図8] 実施形態におけるデータ圧縮装置での圧縮処理の流れの例を示す図である。
- [図9] 実施形態における復元処理の流れの例を示す図である。
- [図10] 実施例におけるデータ圧縮装置での圧縮処理の流れの例を示す図である。
- [図11] 実施例における先頭位置調整処理の流れの例を示す図である。

[図12A]実施例における一致列分割処理の流れの例を示す図である。

[図12B]実施例における一致列分割処理の流れの別の例を示す図である。

[図13]実施例における一致長調整処理の流れの例を示す図である。

[図14] $P(1)$ 、 $P(3)$ 、 $\dots$ 、 $P(2^m - 1)$ で分割する例を示す図である。

[図15] $P(n \div 2)$ 、 $P(n \div 4)$ 、 $\dots$ 、 $P(n \div 2^m)$ で分割する例を示す図である。

[図16] $P(1)$ 、 $P(3)$ 、 $\dots$ 、 $P(2^m - 1)$ で分割する別の例を示す図である。

[図17] $P(1)$ 、 $P(3)$ 、 $\dots$ 、 $P(2^m - 1)$ で分割するさらに別の例を示す図である。

[図18]コンピュータの構成の例を示す図である。

発明を実施するための形態

[0011] 以下、図面を参照して、まず、比較例について説明し、その後、実施形態について説明する。

[0012] また、以下ではLZ77方式を用いたデータ圧縮装置、方法およびプログラムについて説明するが、もちろん、データ圧縮方法はLZ77方式には限定されず、繰返し出現するデータ列を過去に出現したデータ列の長さとの出現位置に置き換えることでデータ量を削減するような圧縮方法であれば適用可能である。

[0013] LZ77方式では、繰返し現われる文字列を検索し、2回目以降にその文字列が現われるとき、その文字列を、出現位置と長さに関するデータに置き換えることにより、圧縮する。ここで、出現位置とは、ある文字列が複数回目に現われるとき、複数回目に現れたときの先頭位置と、以前に現れたときの先頭位置のアドレスの差であり得る。

[0014] 以下では、次のような表記規則を用いる。

(規則1) 一致列は、(一致長、一致列の開始位置) で表す、

(規則2) 非一致列は、{非一致長、非一致文字列} で表す。

ここで、一致列の開始位置は、幾つ前のアドレスに同一の文字列があるか、を示す。一致列の開始位置は、一致位置、相対位置とも呼ぶ。一致列の開始位置（相対位置）は、ある文字列が複数回目に現われるとき、複数回目に現れたときの先頭位置と、以前に現れたときの先頭位置のアドレスの差で、すなわち、相対距離の大きさであり得る。たとえば、一致列の開始位置が、“14”である場合、複数回目に現れたときの先頭位置は、以前に現れたときの先頭位置より14アドレス分だけ後方に位置することを意味しても良い。元データの文字列中の一致列を（一致長、一致列の開始位置）のような形式の圧縮データに符号化するとも言う。同様に、元データの文字列中の非一致列を{非一致長、非一致文字列}のような形式の圧縮データに符号化するとも言う。

[0015] <比較例>

図1～6Bを参照して、比較例について説明する。以下の比較例の説明は、LZ77方式の概略の説明でもある。

[0016] 図1は、LZ77方式の概略を説明する図である。

例えば、図1に示されている第1～34番目のアドレスの位置に格納されている圧縮前のデータ（以下、単に元データと呼ぶことがある）では、第1～14番目のアドレスの位置に“compression_de”という文字列が格納されており、第15～22番目のアドレスの位置に“compress”という文字列が格納されており、第23番目のアドレスの位置には“_”（アンダーバー）が、第24～34番目のアドレスの位置には“compression”という文字列が格納されている。よって、圧縮データは、まず、第1～14番目のアドレスの位置に“compression_de”は非一致列であり、非一致長は14であるので、圧縮データでは、“{14, compression_de}”と表される。元データの第15～22番目のアドレスの位置に“compress”とあるが、これは第1～8番目のアドレスの位置の文字列と同一である。“compress”は8文字であり、一致文字列が開始されるアドレスの位置は14アドレス前で

あるので、一致列の開始位置は14である。よって、元データの第15～2

2番目のアドレスのデータは、圧縮データでは“(8、14)”と符号化さ

れる。元データの第23番目のアドレスの位置に格納されている“(1

ンダーバー)は、それ以前には出現していないので、圧縮データでは“(1

、)”と符号化される。元データの第24～34番目のアドレスの位置に

“compression”とあるが、これは第1～8番目のアドレスの位

置の文字列と同一である。“compression”は8文字であり、一致文字

列が開始されるアドレスの位置は23アドレス前であるので、一致列の開始

位置は23である。よって、元データの第24～34番目のアドレスのデー

タは、圧縮データでは“(11、23)”と符号化される。

[0017] このように、図1に示されている元データの文字列は、圧縮データでは、

“{14, compression_de}(8, 14){1, }11、

23)”のように表される。

[0018] 図2は、LZ77方式を用いて元データを圧縮し、圧縮データを生成する

圧縮装置の機能ブロック図の例である。

[0019] 圧縮装置10は、入力バッファ部11、一致列検索部12、一致長符号化

部13、一致位置符号化部14、非一致長符号化部15、非一致列符号化部

16、および出力バッファ部17を含む。

[0020] 入力バッファ部11は、例えば、図1に示されている第1～34番目のア

ドレスの位置に格納されているような圧縮前のデータ(元データ)を外部か

ら受信し、必要に応じて、一時的に格納する。図1に示されている例では、

元データは、“compression_decompression”

compression.”である。

[0021] 一致列検索部12は、入力バッファ部11に格納された元データの文字列

に、繰り返し出現する文字列(一致列)が存在するかどうかを検索、抽出す

る。また、一致列検索部12は、一致列が存在する場合、その一致列を決

定する。例えば、図1に示されている例では、“compression”なる文

字列や、“compression”なる文字列は複数回出現するので、一

致列である。

[0022] 一致長符号化部 1 3 は、一致列検索部 1 2 で検索され、抽出された一致列の長さを抽出し、規則 1 に従って、一致列を（一致長、一致列の開始位置）と表すときの、第 1 要素を決定する。例えば、図 1 に示されている例では、元データの第 1 5 ～ 2 2 番目のアドレスの位置にある “c o m p r e s s ” なる文字列は、第 1 ～ 8 番目のアドレスの位置の文字列と同一であり、8 文字である。よって、一致長符号化部 1 3 によって、一致長は 8 と符号化される。

[0023] 一致位置符号化部 1 4 は、一致列検索部 1 2 で検索され、抽出された一致列が幾つ前のアドレスの位置から開始されるかを検出し、規則 1 に従って、一致列を（一致長、一致列の開始位置）と表すときの、第 2 要素を決定する。例えば、図 1 に示されている例では、元データの第 1 5 ～ 2 2 番目のアドレスの位置にある “c o m p r e s s ” なる文字列は、第 1 ～ 8 番目のアドレスの位置の文字列と同一であり、一致文字列が開始されるアドレスの位置は 1 4 アドレス前である。よって、一致長符号化部 1 3 によって、一致列の開始位置は 1 4 と符号化される。

[0024] 非一致長符号化部 1 5 は、一致列検索部 1 2 では一致列として抽出されなかった非一致列の長さを抽出し、規則 2 に従って、非一致列を {非一致長、非一致文字列} と表すときの、第 1 要素を決定する。例えば、図 1 に示されている例では、第 1 ～ 1 4 番目のアドレスの位置に “c o m p r e s s i o n _ _ d e ” は非一致列であり、1 4 文字である。よって、非一致長符号化部 1 5 によって、非一致長は 1 4 と符号化される。

[0025] 非一致列符号化部 1 6 は、一致列検索部 1 2 では一致列として抽出されなかった非一致列を抽出し、規則 2 に従って、非一致列を {非一致長、非一致文字列} と表すときの、第 2 要素を決定する。例えば、図 1 に示されている例では、第 1 ～ 1 4 番目のアドレスの位置に “c o m p r e s s i o n _ _ d e ” は非一致列である。よって、非一致長符号化部 1 5 によって、非一致文字列は “c o m p r e s s i o n _ _ d e ” と符号化される。

[0026] 出力バッファ部 17 は、一致長符号化部 13、一致位置符号化部 14、非一致長符号化部 15、および非一致列符号化部 16 の結果を用いて、元データを符号化、圧縮した結果である圧縮データを生成し、必要に応じて、一時的に格納する。たとえば、図 1 に示されている例では、元データである文字列 “compression_decompression_compression.” は、“{14、compression_de} (8、14) {1、_} (11、23)” と符号化される。

[0027] 図 3 は、比較例におけるデータ圧縮装置での圧縮処理の流れの例を示す図である。

処理が開始されると S100 で一致列検索部 12 は、元データの文字列に、繰り返し出現する文字列である一致列が存在するかどうかを検索する。本ステップの処理が終了すると、処理は S102 に進む。元データの文字列は、入力バッファ部 11 に格納されているものであっても良い。

[0028] S102 で一致列検索部 12 は、一致列が見つかったかどうかを判定する。もしこの判定の結果が “YES”、すなわち、一致列が見つかった場合には、処理は S104 に進む。また、もしこの判定の結果が “NO”、すなわち、一致列が見つからなかった場合には、処理は S112 に進む。

[0029] S104 で一致列検索部 12 は、S102 で見つかった一致列の最初の文字のアドレスの位置の直前の文字列が非一致列かどうかを判定する。もしこの判定の結果が “YES”、すなわち、直前の文字列が非一致列である場合には、処理は S106 に進む。また、もしこの判定の結果が “NO”、すなわち、直前の文字列が非一致列ではない場合には、処理は S108 に進む。

[0030] S106 で非一致長符号化部 15 と非一致列符号化部 16 は、S102 で見つかった一致列の直前の非一致列を符号化する。符号化の例は、図 1 に示されている。本ステップの処理が終了すると、処理は S108 に進む。

[0031] S108 で一致長符号化部 13 と一致位置符号化部 14 は、S102 で見つかった一致列を符号化する。符号化の例は、図 1 に示されている。本ステップの処理が終了すると、処理は S110 に進む。

[0032] S 1 1 0では出力バッファ部17は、圧縮すべき全ての元データを処理したかどうかを判定する。もしこの判定の結果が“YES”、すなわち、全ての元データを処理した場合には、圧縮処理を終了する。また、もしこの判定の結果が“NO”、すなわち、全ての元データを処理していない場合には、処理はS 1 0 0に戻る。

[0033] S 1 0 2で一致列が見つからなかった場合は、S 1 1 2に進む。

S 1 1 2で非一致長符号化部15と非一致列符号化部16はそれぞれ、一致が見つからなかった非一致列の長さと言文字列を抽出する。そして、非一致列の長さおよび言文字列に関する情報をそれぞれ、非一致長符号化部15と非一致列符号化部16内に一時的に記憶する。この情報は、S 1 0 6で用いられる。本ステップの処理が終了すると、処理はS 1 0 0に戻る。

[0034] 図4は、比較例における元データと圧縮データの例を示す図である。

図4では、元データが循環言文字列である場合を取り上げている。循環言文字列の長さをcarton（カートン）と呼ぶ。図6においても、上記規則1、2を用いる。

[0035] また、図4において、P（n）（nは整数）は、cartonの区切りを示す。

carton=1の場合の例として、図4には、元データが“aaaaa
aaaaab”の場合が示されている。

[0036] この場合、第1番目のアドレスの位置の“a”が非一致列であり、第2～10番目のアドレスの位置の“aaaaaaa”が一致列である。

[0037] 元データが“aaaaaaaab”の場合、圧縮データは“{1、a}、（9、1）、{1、b}”となる。ここで、圧縮データ中の一致列を示す部分“（9、1）”の第2要素の1は、carton（カートン）に一致する。

[0038] carton=2の場合の例として、図4には、元データが“abababababc”の場合が示されている。この場合、第1～2番目のアドレスの位置の“ab”が非一致列であり、第3～10番目のアドレスの位置の“

a b a b a b a b” が一致列である。第 11 番目のアドレスの位置の文字 “c” は非一致列である。循環文字列の長さが 2 であるので、第 2～10 番目のアドレスは 2 文字ずつ分割され、第 4 番目のアドレスの位置が P (1)、第 6 番目のアドレスの位置が P (2) のように分割される。

[0039] 元データが “a b a b a b a b a b c” の場合、圧縮データは “{2、a b}、(10、2)、{1、c}” となる。

[0040] c a r t o n = 3 の場合の例として、図 4 には、元データが “a b c a b c a b c a b c d” の場合が示されている。この場合、第 1～3 番目のアドレスの位置の “a b c” が非一致列であり、第 4～12 番目のアドレスの位置の “a b c a b c a b c” が一致列である。第 13 番目のアドレスの位置の文字 “d” は非一致列である。

[0041] 元データが “a b c a b c a b c a b c d” の場合、圧縮データは “{3、a b c}、(9、3)、{1、d}” となる。

[0042] c a r t o n = 4 の場合の例として、図 4 には、元データが “a b c d a b c d a b c d a b c d e” の場合が示されている。この場合、第 1～4 番目のアドレスの位置の “a b c d” が非一致列であり、第 5～16 番目のアドレスの位置の “a b c d a b c d a b c d” が一致列である。第 17 番目のアドレスの位置の文字 “e” は非一致列である。

[0043] 元データが “a b c d a b c d a b c d a b c d e” の場合、圧縮データは “{4、a b c d}、(12、4)、{1、e}” となる。

[0044] c a r t o n = 7 の場合の例として、図 4 には、元データが “a b c d e f g a b c d e f g a b c d e f g h” の場合が示されている。この場合、第 1～7 番目のアドレスの位置の “a b c d e f g” が非一致列であり、第 8～21 番目のアドレスの位置の “a b c d e f g a b c d e f g” が一致列である。第 22 番目のアドレスの位置の文字 “h” は非一致列である。

[0045] 元データが “a b c d e f g a b c d e f g a b c d e f g h” の場合、圧縮データは “{7、a b c d e f g}、(14、7)、{1、h}” となる。

- [0046] c a r t o n = 8 の場合の例として、図 4 には、元データが “a b c d e f g h a b c d e f g h a b c d e f g h i” の場合が示されている。この場合、第 1 ～ 8 番目のアドレスの位置の “a b c d e f g h” が非一致列であり、第 9 ～ 24 番目のアドレスの位置の “a b c d e f g h a b c d e f g h” が一致列である。第 25 番目のアドレスの位置の文字 “i” は非一致列である。
- [0047] 元データが “a b c d e f g h a b c d e f g h a b c d e f g h i” の場合、圧縮データは “ { 8、a b c d e f g h }、(16、8)、{ 1、i } ” となる。
- [0048] ここで、図 3 のような処理によって圧縮されたデータの復元処理について図 5 を参照して説明する。
- [0049] 図 5 の処理は、たとえば、図 15 のような構成を有するコンピュータによって処理され得る。図 15 に示されているコンピュータ 100 は、中央処理装置 (C e n t r a l P r o c e s s i n g U n i t (C P U)) 102 と、メモリ、Read Only Memory (R O M) 104、及び R a n d o m A c c e s s M e m o r y (R A M) 106 を備えている。
- [0050] 処理が開始される前に、圧縮データが R A M 106 に格納されているものとする。処理は、圧縮データの列、たとえば図 1 の例では、圧縮データ “ { 14、c o m p r e s s i o n _ d e } (8、14) { 1、_ } (11、23) ” を括弧単位で順次、復元処理を行う。
- [0051] 処理が開始されると S 200 で C P U 102 は、現在処理しているデータが一致列であるかどうかを判定する。もしこの判定の結果が “ Y E S ”、すなわち、現在処理しているデータが一致列である場合には、処理は S 202 に進む。また、もしこの判定の結果が “ N O ”、すなわち、現在処理しているデータが一致列ではない場合には、処理は S 210 に進む。
- [0052] S 202 で C P U 102 は、規則 1 に従って、一致列の圧縮データから一致長と一致列の開始位置を読み取る。本ステップの処理が終了すると、処理は S 204 に進む。

[0053] S 2 0 4 で C P U 1 0 2 は、コピー元とコピー先に重なりがあるかどうかを判定する。

ここで、図 6 A、6 B を参照して、コピー元とコピー先に重なりがある例について説明する。

[0054] 図 6 A に示されているように、元データの文字列は、文字 “a” が 1 0 個並んだ文字列、“a a a a a a a a a a” で開始される文字列であるとする。この元データの文字列を、上記規則 1、2 に従って圧縮すると、次のようになる。

[0055] すなわち、第 1 番目のアドレスに位置する文字列 “a aa” は、それ以前には文字がないので非一致列である。よって、仮の非一致列として “{1, a}” が抽出される。

[0056] 第 2 ～ 9 番目のアドレスに位置する文字列 “a a a a a a a a a a” の処理では、この文字列は、第 1 番目のアドレスに位置する文字列 “a a a a a a a a a a” と一致するので、一致列であると判定される（図 3 の S 1 0 2 を参照）。さらに仮の非一致列を保有しているかどうかを判定し（図 3 の S 1 0 4 を参照）、保有しているので、非一致列 {1, a} の符号化が行われる（S 1 0 6）。また、第 2 ～ 9 番目のアドレスに位置する文字列「a a a a a a a a a a」の符号化が行われる。上記規則 1 に従うと、一致列は、（一致長、一致列の開始位置）で表され、今の場合、一致長は第 2 番目から第 9 番目のアドレスまでの 9、一致列の開始位置は第 1 番目のアドレスであり、第 2 番目のアドレスの 1 つ前であるので、一致列の開始位置は 1 である。よって、元データは、図 6 A に示されているように、“{1, a} (9, 1)” と符号化される。よって、圧縮データは、“{1, a} (9, 1)” となる。

[0057] 図 6 B は、圧縮データ “{1, a} (9, 1)” の復元処理の概略を説明する図である。

[0058] 圧縮データ “{1, a} (9, 1)” の復元では、まず、“{1, a}” の部分を復元する。結果は、第 1 番目のアドレスの位置に文字 “a” が格納される。

[0059] 次に、“(9、1)”の部分の復元では、各アドレスにおいて、1つ前の

アドレスに格納された文字を自身のアドレスにコピーする操作を行う。つまり、第3番目のアドレスの復元は、第2番目のアドレスに格納された文字を第3番目のアドレスの位置に格納する操作を含む。逆に言うと、第3番目のアドレスを復元するためには、第1番目のアドレスに格納された文字を第2番目のアドレスにコピーする操作の終了を待つ必要がある。

[0060] また、図1に示されている例では、圧縮データ中の“(8、14)”の部分の復元では、14アドレス前から8文字分コピーをして、第15～22番目のアドレスの位置にコピーをするが、この場合は、コピー元とコピー先の重なりはない。

[0061] このようにS204でCPU102は、たとえば、図5Bに示されているように、復元処理において、コピー元とコピー先に重なりがあるかどうかを判定する。もしこの判定の結果が“YES”、すなわち、コピー元とコピー先に重なりがある場合には、処理はS206に進む。また、もしこの判定の結果が“NO”、すなわち、コピー元とコピー先に重なりがない場合には、処理はS208に進む。

[0062] S206でCPU102は、一致列の開始位置から一致長分をコピーする。本スレッドの処理が終了すると、処理はS214に進む。

[0063] S208でCPU102は、重ならない範囲で、一致列の開始位置から順に一致長分をコピーする。図6Bに示されている例では、一致列の開始位置から1文字分の「a」をコピーする。

[0064] S200で現在処理しているデータが一致列ではないと判定された場合には、S210に進む。

[0065] S210でCPU102は、規則2に従って非一致列の圧縮データから非一致長を読み取り復元する。本スレッドの処理が終了すると、処理はS212に進む。

[0066] S212でCPU102は、規則2に従って非一致列の圧縮データから非一致文字列を読み取り復元する。

[0067] たとえば、圧縮データ “ { 1 4、 c o m p r e s s i o n _ _ d e } (8、 1 4) { 1、 _ _ } 1 1、 2 3) ” 中の “ { 1 4、 c o m p r e s s i o n _ _ d e } ” の部分から、非一致長 1 4、非一致文字列 “ c o m p r e s s i o n _ _ d e ” と読み取ることができる。そして、第 1 ～ 1 4 番目のアドレスの位置に “ c o m p r e s s i o n _ _ d e ” と格納する。本ステップの処理が終了すると、処理は S 2 1 4 に進む。

[0068] S 2 1 4 で C P U 1 0 2 は、復元すべき全ての元データを処理したかどうかを判定する。もしこの判定の結果が “ Y E S ”、すなわち、全ての圧縮データを処理した場合には、復元処理を終了する。また、もしこの判定の結果が “ N O ”、すなわち、全ての圧縮データを処理していない場合には、処理は S 2 0 0 に戻る。

[0069] このように、比較例では、たとえば図 6 B に示すようにコピー元とコピー先が重なることがある。図 6 B の例では、＜ステップ 1：第 1 バイトを第 2 バイトにコピー＞、＜ステップ 2：第 2 バイトを第 3 バイトにコピー＞の手順でコピーすることになり、ステップ 1 の処理が終わっていないとステップ 2 の処理が行えず、カートン以下の長さ単位での、たとえば 1 バイトずつのコピー処理が必要となる。このように、復元時に高速な多バイトコピーを使用するには、まずコピー元とコピー先が重なるか否か確認する必要があり、また、重なっている場合は、カートン以下の長さ単位での、たとえば 1 バイトずつでコピーを行う煩雑な処理が必要である。

[0070] ＜実施形態＞

以下で説明する実施形態では、たとえば L Z 7 7 方式において、相対位置を `offset`、一致長を `length` としたときに、相対位置が一致長より大きくなる、すなわち `offset > length` となるように一致列を分割して符号化する。このように圧縮することによって、一致列の分割によりコピー元とコピー先が重ならなくなるため、復元処理の際に条件分岐がなく、高速な復元処理を実現することができる圧縮データを生成することができる。すなわち、多バイト単位で圧縮データの復元を実現するために、元デ

ータ中で繰り返し出現する一致列を、復元時にコピー元とコピー先が重ならないように再分割してから圧縮することで、高速に復元できる圧縮データを生成することができる。

ここで、「重なり」とは、コピー元の文字列の最後尾のアドレスがコピー先の文字列の先頭のアドレスより後方になることを意味し得る。

[0071] 図7は、元データを圧縮し、圧縮データを生成する圧縮装置の機能ブロック図の例である。

[0072] 圧縮装置20は、入力バッファ部21、一致列検索部22、一致長符号化部23、一致位置符号化部24、非一致長符号化部25、非一致列符号化部26、出力バッファ部27、および一致列分割部28を含む。比較例の圧縮装置10とは、一致列分割部28が含まれている点で異なっている。

[0073] 入力バッファ部21は、例えば、圧縮前のデータ（元データ）を外部から受信し、必要に応じて、一時的に格納する。

[0074] 一致列検索部22は、入力バッファ部11に格納された元データの文字列に、繰り返し出現する文字列（一致列）が存在するかどうかを検索、抽出する。また、一致列検索部22は、一致列が存在する場合は、その一致列を決定する。例えば、図1に示されている例では、“compress”なる文字列や、“compression”なる文字列は複数回出現するので、一致列である。

[0075] このように一致列検索部22は、圧縮前のデータの文字列中の第1の文字列の後に、第1の文字列と同一である第2の文字列を一致列として抽出し、一致列の長さと、第1の文字列が第2の文字列の幾つ前のアドレスの位置から開始されるかを示す相対位置を決定する。

そして一致列分割部28は、相対位置が一致長以上、すなわち $offset \geq length$ となるように一致列を分割する。

[0076] このように一致列分割部28は、一致列検索部で抽出された第2の文字列から、長さが相対位置より小さい第3の文字列を抽出する。

[0077] たとえば、元データが“aaaaaaaaaab”である例をあげる。こ

の例では、第1～10番目のアドレスの位置に文字“a”が格納され、第11番目のアドレスの位置に文字“b”が格納されている。

[0078] このとき、たとえば、図14のように、元データの文字列“a a a a a a a a a a b”の第2～10番目のアドレスの位置の文字列“a a a a a a a a a”（9個の“a”）を、“a”+“a a”+“a a a a”+“a a”と分割することができる。つまり、図4のP(1)、P(3)、P(7)の位置、つまり $P(2^m - 1)$ の位置で分割する。すると、元データ“a a a a a a a a a a b”の第2番目のアドレスの一致列“a”に対しては、一致長lengthが1、相対位置offsetが1なので、 $offset \geq length$ となる。第3～4番目の一致列“a a”については、一致長lengthが2、相対位置offsetが2なので $offset \geq length$ となる。第5～8番目の一致列“a a a a”については、一致長lengthが4、相対位置offsetが4なので $offset \geq length$ となる。

[0079] また、たとえば、図14に示されているように、元データの文字列“a a a a a a a a a a b”の第2～10番目のアドレスの位置の文字列“a a a a a a a a a”（9個の“a”）を“a”+“a”+“a a”+“a a a a a a”と分割することができる。つまり、図4のP(1)、P(2)、P(4)の位置、つまり、 $n = (\text{一致長列} + \text{carton} - 1) \div \text{carton}$ とし、 $P(n \div 2)$ 、 $P(n \div 4)$ 、…、 $P(n \div 2^m)$ の位置で分割する。すると、元データ“a a a a a a a a a a b”の第2番目のアドレスの一致列“a”に対しては、一致長lengthが1、相対位置offsetが1なので、 $offset \geq length$ となる。第3番目の一致列“a”については、一致長lengthが1、相対位置offsetが2なので $offset \geq length$ となる。第3～5番目の一致列“a a”については、一致長lengthが2、相対位置offsetが3なので $offset \geq length$ となる。第6～10番目の一致列“a a a a a”については、一致長lengthが5、相対位置offsetが5なので $offset \geq length$ となる。

g t hとなる。

[0080] 一致長符号化部23および一致位置符号化部24は、一致列分割部28で分割された一致列に対し、規則1に従って、一致列を（一致長、一致列の開始位置）と表すときの、第1要素および第2要素をそれぞれ符号化する。

[0081] たとえば、図14のように、元データの文字列“a a a a a a a a a b”の第2～10番目のアドレスの位置の一致列“a a a a a a a a a”（9個の“a”）を、“a”+“a a”+“a a a a”+“a a”と分割すると、一致列は、“（1、1）（2、2）（4、4）（2、8）”と符号化される。

[0082] また、たとえば、図15に示されているように、元データの文字列“a a a a a a a a a b”の第2～10番目のアドレスの位置の一致列“a a a a a a a a a”（9個の“a”）を“a”+“a”+“a a”+“a a a a a a a”と分割すると、一致列は、“（1、1）（1、2）（2、3）（5、5）”と符号化される。

[0083] 非一致長符号化部25は、一致列検索部22では一致列として抽出されなかった非一致列の長さを抽出し、規則2に従って、非一致列を{非一致長、非一致文字列}と表すときの、第1要素を符号化する。

[0084] 非一致列符号化部26は、一致列検索部22では一致列として抽出されなかった非一致列を抽出し、規則2に従って、非一致列を{非一致長、非一致文字列}と表すときの、第2要素を符号化する。

[0085] 出力バッファ部27は、一致長符号化部23、一致位置符号化部24、非一致長符号化部25、および非一致列符号化部26の結果を用いて、元データを符号化、圧縮した結果である圧縮データを生成し、必要に応じて、一時的に格納する。

[0086] 図15は実施形態のデータ圧縮装置200の構成の例を示す図である。

このコンピュータ100は、Central Processing Unit（CPU）102、Read Only Memory（ROM）104、及びRandom Access Memory（RAM）106を

備えている。コンピュータ500は、さらに、ハードディスク装置108、入力装置110、表示装置112、インタフェース装置114、及び記録媒体駆動装置116を備えている。なお、これらの構成要素はバスライン120を介して接続されており、CPU102の管理の下で各種のデータを相互に授受することができる。

[0087] Central Processing Unit (CPU) 102は、このコンピュータ100全体の動作を制御する演算処理装置であり、コンピュータ100の制御処理部として機能する。

[0088] Read Only Memory (ROM) 104は、所定の基本制御プログラムが予め記録されている読み出し専用半導体メモリである。CPU 102は、この基本制御プログラムをコンピュータ100の起動時に読み出して実行することにより、このコンピュータ100の各構成要素の動作制御が可能になる。

[0089] Random Access Memory (RAM) 106は、CPU 102が各種の制御プログラムを実行する際に、必要に応じて作業用記憶領域として使用する、随時書き込み読み出し可能な半導体メモリである。

[0090] ハードディスク装置108は、CPU 102によって実行される各種の制御プログラムや各種のデータを記憶しておく記憶装置である。MPU 502は、ハードディスク装置108に記憶されている所定の制御プログラムを読み出して実行することにより、後述する各種の制御処理を行えるようになる。

[0091] 入力装置110は、例えばマウス装置やキーボード装置であり、情報処理装置のユーザにより操作されると、その操作内容に対応付けられている各種情報の入力を取得し、取得した入力情報をCPU 102に送付する。

[0092] 表示装置512は例えば液晶ディスプレイであり、MPU 502から送付される表示データに応じて各種のテキストや画像を表示する。

[0093] インタフェース装置114は、このコンピュータ100に接続される各種機器との間での各種情報の授受の管理を行う。

[0094] 記録媒体駆動装置 116 は、可搬型記録媒体 118 に記録されている各種の制御プログラムやデータの読み出しを行う装置である。CPU 102 は、可搬型記録媒体 118 に記録されている所定の制御プログラムを、記録媒体駆動装置 116 を介して読み出して実行することによって、後述する各種の制御処理を行うようにすることもできる。なお、可搬型記録媒体 118 としては、例えば USB (Universal Serial Bus) 規格のコネクタが備えられているフラッシュメモリ、CD-ROM (Compact Disc Read Only Memory)、DVD-ROM (Digital Versatile Disc Read Only Memory) などがある。

[0095] このようなコンピュータ 100 を用いて情報処理装置を構成するには、例えば、上述の各処理部における処理を CPU 102 に行わせるための制御プログラムを作成する。作成された制御プログラムはハードディスク装置 108 若しくは可搬型記録媒体 118 に予め格納しておく。そして、CPU 102 に所定の指示を与えてこの制御プログラムを読み出させて実行させる。こうすることで、情報処理装置が備えている機能が CPU 102 により提供される。

[0096] 図 8 は、実施形態におけるデータ圧縮装置 20 での圧縮処理の流れの例を示す図である。

[0097] また、データ圧縮装置 20 が図 15 に示されているような汎用コンピュータ 100 である場合には、下記の説明は、そのような処理を行う制御プログラムを定義する。すなわち、以下では、下記に説明する処理を汎用コンピュータに行わせる制御プログラムの説明でもある。

[0098] 処理が開始されると S300 で一致列検索部 22 は、元データの文字列に、繰り返し出現する文字列である一致列が存在するかどうかを検索する。本ステップの処理が終了すると、処理は S302 に進む。元データの文字列は、入力バッファ部 21 に格納されているものであっても良い。

[0099] S302 で一致列検索部 22 は、一致列が見つかったかどうかを判定する

。もしこの判定の結果が“YES”、すなわち、一致列が見つかった場合には、処理はS304に進む。また、もしこの判定の結果が“NO”、すなわち、一致列が見つからなかった場合には、処理はS306に進む。

[0100] S304で非一致長符号化部25と非一致列符号化部26はそれぞれ、一致が見つからなかった非一致列の長さおよび文字列を抽出する。そして、非一致列の長さおよび文字列に関する情報をそれぞれ、非一致長符号化部25と非一致列符号化部26内に一時的に記憶する。この情報は、S106で用いられる。本ステップの処理が終了すると、処理はS100に戻る。

[0101] S302で一致列が見つからなかった場合には、処理はS306に進む。

S306で一致列分割部28は、一致列の長さ、つまり一致長と、その一致列が幾つ前のアドレスの位置から開始されるか、つまり相対位置を検出する。そしてS306で一致列分割部28は、相対位置が一致長以上、すなわち相対位置offset $\geq$ 一致長lengthとなるように一致列を分割する。

[0102] 相対位置が一致長以上である、すなわち相対位置offset $\geq$ 一致長lengthを満たす場合には、このステップでの処理は行われない。

[0103] S308で一致列分割部28は、分割した一致列の最初の文字のアドレスの位置の直前の文字列が非一致列かどうかを判定する。もしこの判定の結果が“YES”、すなわち、直前の文字列が非一致列である場合には、処理はS310に進む。また、もしこの判定の結果が“NO”、すなわち、直前の文字列が非一致列ではない場合には、処理はS312に進む。

[0104] S310で非一致長符号化部25と非一致列符号化部26は、S306で必要に応じて分割された一致列の直前の非一致列を符号化する。本ステップの処理が終了すると、処理はS312に進む。

[0105] S312で一致長符号化部23と一致位置符号化部24は、S306で必要に応じて分割された一致列を符号化する。本ステップの処理が終了すると、処理はS314に進む。

[0106] S314で出力バッファ27は圧縮すべき全ての元データを処理したかど

うかを判定する。もしこの判定の結果が“YES”、すなわち、全ての元データを処理した場合には、圧縮処理を終了する。また、もしこの判定の結果が“NO”、すなわち、全ての元データを処理していない場合には、処理はS300に戻る。

[0107] ここで、図8のような処理によって圧縮されたデータの復元処理について図9を参照して説明する。

[0108] 図9の処理は、たとえば、図15のような構成を有するコンピュータによって処理され得る。

[0109] 処理が開始される前に、圧縮データがRAM106に格納されているものとする。

処理が開始されるとS400でCPU102は、現在処理しているデータが一致列であるかどうかを判定する。もしこの判定の結果が“YES”、すなわち、現在処理しているデータが一致列である場合には、処理はS402に進む。また、もしこの判定の結果が“NO”、すなわち、現在処理しているデータが一致列ではない場合には、処理はS406に進む。

[0110] S402でCPU102は、規則1に従って、一致列の圧縮データから一致長と一致列の開始位置を読み取る。本ステップの処理が終了すると、処理はS404に進む。

[0111] S404でCPU102は、一致列の開始位置から一致長分をコピーする。本ステップの処理が終了すると、処理はS410に進む。

[0112] S400で現在処理しているデータが一致列ではないと判定された場合には、S406に進む。

[0113] S406でCPU102は、規則2に従って非一致列の圧縮データから非一致長を読み取り復元する。本ステップの処理が終了すると、処理はS408に進む。

[0114] S408でCPU102は、規則2に従って非一致列の圧縮データから非一致文字列を読み取り復元する。

[0115] 本ステップの処理が終了すると、処理はS410に進む。

S 4 1 0でCPU 1 0 2は、復元すべき全ての元データを処理したかどうかを判定する。もしこの判定の結果が“YES”、すなわち、全ての圧縮データを処理した場合には、復元処理を終了する。また、もしこの判定の結果が“NO”、すなわち、全ての圧縮データを処理していない場合には、処理はS 4 0 0に戻る。

[0116] このように、図8のような処理によって圧縮されたデータの復元処理は、図4に示されている復元処理に比べると単純化されており、よって、高速に処理することができる。

[0117] <実施例>

コピー元とコピー先の重なりは、同じ1文字、2文字列、3文字列、・・・のいずれかが続けて繰返す循環文字列において発生する。LZ77方式において、一致列を検索して相対位置offset、一致長lengthが得られたとする。offset<lengthのとき、コピー元とコピー先が重なるため一致列を分割して符号化する。また、offset<lengthであれば常に分割を行うのではなく、重なりを最小距離を予めXと定め、offset<Xのときのみ分割を行うようにしてもよい。

[0118] 分割の仕方には、いくつか考えられる。以下では、次のような場合について説明する。

(Ex 1) $P(1)$ 、 $P(3)$ 、 $P(7)$ 、・・・、 $P(2^m - 1)$ (ただし $(2^m - 1) < n$) で分割する、

(Ex 2) $n = (\text{一致長} + \text{carton} - 1) \div \text{carton}$ とし、 $P(n \div 2)$ 、 $P(n \div 4)$ 、 $P(n \div 8)$ 、・・・、 $P(n \div 2^m)$ (ただし $n \div 2^m > 1$) で分割する、

(Ex 3) 重なりを最小距離X、一致長の最小値Nを定めた上で、 $P(1)$ 、 $P(1 + 2)$ 、 $P(1 + 2 + 4)$ 、・・・、 $P(2^m - 1)$ (ただし $(2^m - 1) < n$) で分割する、

(Ex 4) 一致長の最小値をNとし、循環の1セットがN以上になるように調節した上で、上記(Ex 1)～(Ex 3)の分割をする。

[0119] 以下では上記（E x 1）～（E x 4）について、説明する。

図 10 は、実施例におけるデータ圧縮装置での圧縮処理の流れの例を示す図である。

[0120] また、データ圧縮装置 20 が図 15 に示されているような汎用コンピュータ 100 である場合には、下記の説明は、そのような処理を行う制御プログラムを定義する。すなわち、以下では、下記に説明する処理を汎用コンピュータに行わせる制御プログラムの説明でもある。

[0121] 処理が開始されると S 500 で入力バッファ部 21 は、現在のアドレスの位置 `p__cur` と、非一致乗長 `n__len` をリセットする。たとえば、`p__cur = 0`、`n__len = 0` とする。本ステップの処理が終了すると、処理は S 502 に進む。

[0122] 次の S 502 で一致列検索部 22 は、一致列を検索する。具体的には、一致長 `len` と相対位置 `offset` を求める。本ステップの処理が終了すると、処理は S 504 に進む。

[0123] S 504 で一致列検索部 22 は、一致列が見つかったかどうかを判定する。もしこの判定の結果が“YES”、すなわち、一致列が見つかった場合には、処理は S 506 に進む。また、もしこの判定の結果が“NO”、すなわち、一致列が見つからなかった場合には、処理は S 532 に進む。

[0124] S 506 で一致列検索部 22 は、現在のアドレスの位置 `p__cur` と、非一致乗長 `n__len` を更新する。たとえば、現在のアドレスの位置 `p__cur` と、非一致乗長 `n__len` をそれぞれ、1 だけ増加させる。

[0125] S 506 では、先頭位置調整処理が行われる。

図 11 を参照して、先頭位置調整処理について説明する。この処理では、一致長の最小値 `N` が定められているとする。

[0126] 処理が開始されると S 600 で一致列検索部 22 は、相対位置 `offset` を変数 `i` に代入する。本ステップの処理が終了すると、処理は S 602 に進む。

[0127] S 602 で一致列検索部 22 は、変数 `i` が一致長の最小値 `N` 以上であるか

を判定する。もしこの判定の結果が“YES”、すなわち、変数 i が一致長の最小値 N 以上である場合には、処理は $S606$ に進む。また、もしこの判定の結果が“NO”、すなわち、変数 i が一致長の最小値 N 以上ではない場合には、処理は $S604$ に進む。

[0128] $S606$ で一致列検索部 22 は、非一致長 n_len に (i_offset) の値を加え、現在のアドレスの位置 p_cur に (i_offset) の値を加える。さらに、一致列の一致長 len から (i_offset) の値を引く。本ステップの処理が終了すると、先頭位置調整処理は終了する。

[0129] $S604$ で一致列検索部 22 は、変数 i に $offset$ の値を加える。

図 10 に戻って、 $S506$ の次の $S508$ で一致列分割部 28 は、一致列分割処理を行う。

[0130] 図 12A、12B を参照して、一致列分割処理について説明する。

図 12A は、上記 $E \times 1$ のように、 $P(1)$ 、 $P(3)$ 、 $P(7)$ 、 $\dots$ 、 $P(2^m - 1)$ (ただし $(2^m - 1) < n$) で分割する場合の処理を示す図である。

[0131] 処理を開始すると、 $S700$ で一致列分割部 28 は、分割した一致列の後尾番号 e_num の値をリセットし、一致列の一致長の変数 $len2$ に一致列の一致長 len の値を代入する。本ステップの処理が終了すると、処理は $S702$ に進む。

[0132] $S702$ で一致列分割部 28 は、 $offset > X$ または $offset > len2$ であるかを判定する。もしこの判定の結果が“YES”、すなわち、 $offset > X$ または $offset > len2$ である場合には、処理は $S704$ に進む。また、もしこの判定の結果が“NO”、すなわち、 $offset \leq X$ かつ $offset \leq len2$ の場合には、処理は $S706$ に進む。

[0133] $S704$ で一致列分割部 28 は、分割した一致列の一致位置 $A_offset[e_num]$ に $offset$ の値を代入し、分割した一致列の一致長 $A_len[e_num]$ に $len2$ の値を代入する。本ステップの処理が

終了すると、処理はS 7 0 6に進む。

[0134] S 7 0 6で一致列分割部28は、分割した一致列の一致位置A__offset[e__num]にoffsetの値を代入し、分割した一致列の一致長A__len[e__num]にlen2の値を代入する。さらに、e__numの値を更新する。たとえば、e__numの値を1だけ増やす。len2の値からoffsetの値を引き、offsetの値を更新する。offsetの値の更新は、たとえば、値を2倍にする。本ステップの処理が終了すると、処理はS 7 0 2に戻る。

[0135] 図12Bは、上記Ex2のように、 $n = (\text{一致長} + \text{carton} - 1) \div \text{相対位置}$ とし、 $P(n \div 2)$ 、 $P(n \div 4)$ 、 $P(n \div 8)$ 、 $\dots$ 、 $P(n \div 2^m)$ （ただし $n \div 2^m > 1$ ）で分割する場合の処理を示す図である。

[0136] 処理を開始すると、S 8 0 0で一致列分割部28は、一致長をlen、相対位置をoffsetとして、 $n = (\text{len} + \text{carton} - 1) \div \text{offset}$ で算出する。さらに、ダミー変数iをリセットし、 $i = 0$ とする。本ステップの処理が終了すると、処理はS 8 0 2に進む。

[0137] S 8 0 2で一致列分割部28は、S 8 0 0で算出されたnの値が1以下であるかどうかを判定する。もしこの判定の結果が“YES”、すなわち、nの値が1以下である場合には、処理はS 8 0 6に進む。また、もしこの判定の結果が“NO”、すなわち、nの値が1より大きい場合には、処理はS 8 0 4に進む。

[0138] S 8 0 4で一致列分割部28は、nの値を更新する。たとえば、値を半分にする。さらに、分割位置P(i)にnの値を代入し、さらにダミー変数iの値を1増やす。本ステップの処理が終了すると、処理はS 8 0 2に戻る。

[0139] S 8 0 6で一致列分割部28は、一致列の一致長の変数len2と、分割した一致列の後尾番号e__numの値をリセットする。 $\text{len2} = 0$ 、 $\text{e_num} = 0$ とする。本ステップの処理が終了すると、処理はS 8 0 8に進む。

[0140] S 8 0 8で一致列分割部28は、ダミー変数iの値が1より小さいかどうか

かを判定する。もしこの判定の結果が“YES”、すなわち、ダミー変数 i の値が1より小さい場合には、処理はS810に進む。また、もしこの判定の結果が“NO”、すなわち、ダミー変数 i の値が1以上の場合には、処理はS812に進む。

[0141] S810で一致列分割部28は、分割した一致列の一致長 $A_len[e_num]$ に $(len - len2)$ の値を代入する。また分割した一致列の一致位置 $A_offset[e_num]$ に $(P(0) + 1) \times offset$ の値を代入する。本ステップの処理が終了すると、一致列分割処理を終了する。

[0142] S812で一致列分割部28は、ダミー変数 i の値を1減らす。また、ダミー変数 tmp に分割位置 $P(i)$ と相対位置 $offset$ の積、 $P(i) \times offset$ を代入し、このダミー変数 tmp を分割した一致列の一致長 $A_len[e_num]$ と、分割した一致列の一致位置 $A_offset[e_num]$ に代入する。 $len2$ の値に tmp の値を加え、 e_num の値を1増やす。本ステップの処理が終了すると、処理はS808に戻る。

[0143] 図10に戻り、S510で一致列分割部28は、分割した一致列の先導番号 f_num と分割した一致列の非一致長 n_len2 をリセットする。 $f_num = 0$ 、 $n_len2 = 0$ とする。本ステップの処理が終了すると、S512に進む。

[0144] S512で一致列分割部28は、一致長調整処理を行う。

一致長調整処理について、図13を参照して説明する。

[0145] 処理を開始するとS900で一致列分割部28は、分割した一致列の一致位置 $A_offset[f_num]$ が一致長の最小値 N 以上であるかどうかを判定する。もしこの判定の結果が“YES”、すなわち、分割した一致列の一致位置 $A_offset[f_num]$ が一致長の最小値 N 以上である場合には、処理はS904に進む。また、もしこの判定の結果が“NO”、すなわち、分割した一致列の一致位置 $A_offset[f_num]$ が一致長の最小値 N より小さい場合には、処理はS902に進む。

[0146] S 9 0 2 で一致列分割部 2 8 は、分割した一致列の非一致長 n_len の値に $A_offset[f_num]$ の値を加える。また、 f_num の値を 1 増やす。本ステップの処理が終了すると、処理は S 9 0 0 に戻る。

[0147] S 9 0 0 および S 9 0 2 の処理で、先頭の分割列の調整を行う。

S 9 0 4 で一致列分割部 2 8 は、分割した一致列の一致長 $A_len[e_num-1]$ が一致長の最小値 N 以上であるか、または e_num の値が f_num の値より小さいかどうかを判定する。もしこの判定の結果が “YES”、すなわち、分割した一致列の一致長 $A_len[e_num-1]$ が一致長の最小値 N 以上であるか、または e_num の値が f_num の値より小さい場合には、一致長調整処理を終了する。また、もしこの判定の結果が “NO”、すなわち、分割した一致列の一致位置 $A_offset[e_num-1]$ が一致長の最小値 N より小さく、かつ e_num の値が f_num の値以上である場合には、処理は S 9 0 6 に進む。

[0148] S 9 0 6 で一致列分割部 2 8 は、分割した一致列の非一致長 n_len2 の値に分割した一致列の一致長 $A_len[e_num-1]$ の値を加え、 e_num の値を 1 減らす。

[0149] 図 1 0 に戻って、S 5 1 4 で一致列分割部 2 8 は、非一致長 n_len が 0 かどうかを判定する。もしこの判定の結果が “YES”、すなわち、非一致長 n_len が 0 である場合には、処理は S 5 1 8 に進む。また、もしこの判定の結果が “NO”、すなわち、非一致長 n_len が 0 でない場合には、処理は S 5 1 6 に進む。

[0150] S 5 1 6 で非一致長符号化部 2 5 と非一致列符号化部 2 6 は、非一致列の符号化を行う。つまり、非一致長 n_len と、非圧縮データ $InBuf[(p_cur-n_len), \dots, (p_cur-1)]$ の符号化を行う。本ステップの処理が終了すると、処理は S 5 1 8 に進む。

[0151] S 5 1 8 で一致列分割部 2 8 は、分割した一致列の先頭のアドレス番号 f_num が分割した一致列の後尾のアドレス番号 e_num の値より大きいかを判定する。もしこの判定の結果が “YES”、すなわち、 $f_num >$

e__numである場合には、処理はS 5 2 0に進む。また、もしこの判定の結果が“NO”、すなわち、 $f_num \leq e_num$ である場合には、処理はS 5 2 4に進む。

[0152] S 5 2 4で一致長符号化部2 3と一致位置符号化部2 4は、一致列の符号化を行う。つまり、一致列の一致長A__len[f__num]およびA__offset[f__num]の符号化を行う。本ステップの処理が終了すると、処理はS 5 2 6に進む。

[0153] S 5 2 6で一致長符号化部2 3と一致位置符号化部2 4は、分割した一致列の先頭のアドレス番号f__numの値を1増やす。本ステップの処理が終了すると、処理はS 5 1 8に戻る。

[0154] S 5 2 0で一致列分割部2 8は、非一致長n__lenに分割した一致列の非一致長n__len2の値を代入し、現在位置p__curに一致長lenの値を加える。本ステップの処理が終了すると、処理はS 5 2 2に進む。

[0155] S 5 2 2で出力バッファ2 7は圧縮すべき全ての元データを処理したかどうかを判定する。もしこの判定の結果が“YES”、すなわち、全ての元データを処理した場合には、S 5 2 4に進む。また、もしこの判定の結果が“NO”、すなわち、全ての元データを処理していない場合には、処理はS 5 0 2に戻る。

[0156] S 5 2 4で一致列分割部2 8は、非一致長n__lenが0かどうかを判定する。もしこの判定の結果が“YES”、すなわち、非一致長n__lenが0である場合には、処理を終了する。また、もしこの判定の結果が“NO”、すなわち、非一致長n__lenが0でない場合には、処理はS 5 2 6に進む。

[0157] S 5 2 6で非一致長符号化部2 5と非一致列符号化部2 6は、非一致列の符号化を行う。つまり、非一致長n__lenと、非圧縮データInBuf[(p__cur-n__len)、…、(p__cur-1)]の符号化を行う。本ステップの処理が終了すると、処理は終了する。

[0158] 圧縮装置2 0および上記の圧縮方法では、LZ77方式において、ある一

致列の相対位置 $offset$ と一致長 $length$ であるとき、 $offset < length$ のとき、コピー元とコピー先が重なるため一致列を分割して符号化する。

[0159] また、重なりの最小距離を予め X と定め、 $offset < X$ のときのみ分割を行うようにしてもよい。

[0160] また、コピー元とコピー先の重なりは、同じ 1 文字、2 文字列、3 文字列、 $\dots$ のいずれかが続けて繰返す循環文字列において発生する。循環の 1 セットの長さはコピー元とコピー先の相対位置（相対距離）と一致する。初回の 1 セットは別符号で符号化され、2 回目以降の循環において重なりが発生する。

[0161] そこで、循環の 1 セットの長さを $carton$ （カートン）とし、一致文字列の先頭から $P(1) = carton$ 、 $P(2) = carton \times 2$ 、 $\dots$ 、 $P(n) = carton \times n$ （ただし $P(n)$ は $length$ より小さい）の位置の全てまたはいずれかで分割することで、コピー元とコピー先が重ならないようにする。

[0162] 図 14 に、 $P(1)$ 、 $P(3)$ 、 $P(7)$ 、 $\dots$ 、 $P(2^m - 1)$ （ただし $(2^m - 1) < n$ ）で分割する例を、図 15 に、 $n = (一致長 + carton - 1) \div carton$ とし、 $P(n \div 2)$ 、 $P(n \div 4)$ 、 $P(n \div 8)$ 、 $\dots$ 、 $P(n \div 2^m)$ （ただし $n \div 2^m > 1$ ）で分割する例を示した。また、一致長の最小値を N を予め定め、一致文字列を分割することで一致長が N 以下となる部分は非一致列として扱ってもよい。

[0163] 図 16 では、重なりの最小距離 X 、一致長の最小値 N を定めた上で、 $P(1)$ 、 $P(1 + 2)$ 、 $P(1 + 2 + 4)$ 、 $\dots$ 、 $P(2^m - 1)$ （ただし $(2^m - 1) < n$ ）で分割する例において、重なりの最小距離 $X = 8$ 、一致長の最小値 $N = 3$ の場合が示されている。この場合、一致文字列を分割することで一致長が N 以下となる部分は非一致列として扱う。

[0164] また、一致長の最小値を N としているとき、循環の 1 セットが N 以上になるように調節した上で、上記分割を行っても良い。具体的には、相対位置 o

$offset < N$ のとき、 $N \leq offset \times i$ (ただし、 i は整数値) となる最大の「 $offset \times i$ 」の長さの分、一致列の先頭位置を後ろへずらしても良い。

[0165] 例えば、図 17 には、一致長の最小値 $N = 3$ として、循環の 1 セットが N 以上になるように調整した上で、 $P(1)$ 、 $P(1+2)$ 、 $P(1+2+4)$ 、 $\dots$ 、 $P(2^m-1)$ (ただし $(2^m-1) < n$) の分割を行った例が示されている。

[0166] また、 $P(1)$ 、 $P(2)$ 、 $\dots$ 、 $P(n)$ の全て、および図 14～17 に示す分割から最も符号量が小さくなるものを選択してもよい。

[0167] <<第 1 実施例 (Ex 1)>>

図 14 は、 $P(1)$ 、 $P(3)$ 、 $P(7)$ 、 $\dots$ 、 $P(2^m-1)$ (ただし $(2^m-1) < n$) で分割する例を示す図である。

[0168] $carton = 1$ の場合の例として、図 14 には、元データが “a a a a a a a a a b” の場合が示されている。

[0169] この場合、第 1 番目のアドレスの位置の “a” が非一致列であり、第 2～10 番目のアドレスの位置の “a a a a a a a a a a” が一致列である。第 11 番目のアドレスの位置の文字 “b” は非一致列である。第 2～10 番目のアドレスの位置の一致列 “a a a a a a a a a a” を 1 文字目、3 文字目、7 文字目で分割する。すなわち、一致列 “a a a a a a a a a a” を、「“a” + “a a” + “a a a a” + 残り」と分割する。

[0170] 元データが “a a a a a a a a a b” の場合、圧縮データは “{1, a} (1, 1) (2, 2) (4, 4) (2, 8) {1, b}” となる。

[0171] $carton = 2$ の場合の例として、図 14 には、元データが “a b a b a b a b a c” の場合が示されている。この場合、第 1～2 番目のアドレスの位置の “a b” が非一致列であり、第 3～10 番目のアドレスの位置の “a b a b a b a b a a” が一致列である。第 11 番目のアドレスの位置の文字 “c” は非一致列である。第 3～9 番目のアドレスの位置の “a b a b a b a b a a” を 1 カートン目、3 カートン目で分割する。すなわち、一致列 “a b a b a

b a b”を、「“a b” + “a b a b” + 残り」と分割する。

[0172] 元データが“a b a b a b a b a c”の場合、圧縮データは “{2, a b }、(2, 2) (4, 4) (3, 8) {1, c}”となる。

[0173] c a r t o n = 3 の場合の例として、図 1 4 には、元データが“a b c a b c a b c a b d”の場合が示されている。この場合、第 1 ~ 3 番目のアドレスの位置の“a b c”が非一致列であり、第 4 ~ 1 1 番目のアドレスの位置の“a b c a b c a b”が一致列である。第 1 2 番目のアドレスの位置の文字“d”は非一致列である。第 4 ~ 1 1 番目のアドレスの位置の“a b c a b c a b c a b”を 1 カートン目で分割する。すなわち、一致列“a b c a b c a b c”を、「“a b c” + 残り」と分割する。

[0174] 元データが“a b c a b c a b c a b d”の場合、圧縮データは “{3, a b c} (3, 3) (5, 6) {1, d}”となる。

[0175] c a r t o n = 4 の場合の例として、図 1 4 には、元データが“a b c d a b c d a b c d a b c e”の場合が示されている。この場合、第 1 ~ 4 番目のアドレスの位置の“a b c d”が非一致列であり、第 5 ~ 1 5 番目のアドレスの位置の“a b c d a b c d a b c d”が一致列である。第 1 7 番目のアドレスの位置の文字“e”は非一致列である。第 5 ~ 1 5 番目のアドレスの位置の“a b c d a b c d a b c d a b c”を 1 カートン目で分割する。すなわち、一致列“a b c d a b c d a b c d a b c”を、「“a b c d” + 残り」と分割する。

[0176] 元データが“a b c d a b c d a b c d a b c d e”の場合、圧縮データは “{4, a b c d} (4, 4) (7, 8) {1, e}”となる。

[0177] c a r t o n = 7 の場合の例として、図 1 4 には、元データが“a b c d e f g a b c d e f g a b c d e f h”の場合が示されている。この場合、第 1 ~ 7 番目のアドレスの位置の“a b c d e f g”が非一致列であり、第 8 ~ 2 0 番目のアドレスの位置の“a b c d e f g a b c d e f”が一致列である。第 2 1 番目のアドレスの位置の文字“h”は非一致列である。第 8 ~ 2 0 番目のアドレスの位置の“a b c d e f g a b c d e f”を 1 カート

ン目で分割する。すなわち、一致列 “a b c d e f g a b c d e f” を、「 “a b c d e f g” + 残り」と分割する。

[0178] 元データが “a b c d e f g a b c d e f g a b c d e f h” の場合、圧縮データは “ { 7、a b c d e f g } (7、7) (6、1 4) { 1、h } ” となる。

[0179] c a r t o n = 8 の場合の例として、図 1 4 には、元データが “a b c d e f g h a b c d e f g h a b c d e f g i” の場合が示されている。この場合、第 1 ～ 8 番目のアドレスの位置の “a b c d e f g h” が非一致列であり、第 9 ～ 2 4 番目のアドレスの位置の “a b c d e f g h a b c d e f g” が一致列である。第 2 5 番目のアドレスの位置の文字 “i” は非一致列である。第 9 ～ 2 3 番目のアドレスの位置の “a b c d e f g h a b c d e f g” を 1 カートン目で分割する。すなわち、一致列 “a b c d e f g h a b c d e f g” を、「 “a b c d e f g h” + 残り」と分割する。

[0180] 元データが “a b c d e f g h a b c d e f g h a b c d e f g i” の場合、圧縮データは “ { 8、a b c d e f g h } (8、8) (7、1 6) { 1、i } ” となる。

[0181] <<第 2 実施例 (E x 2) >>

図 1 5 は、 $n = (\text{一致長} + \text{c a r t o n} - 1) \div \text{c a r t o n}$ とし、 $P(n \div 2)$ 、 $P(n \div 4)$ 、 $P(n \div 8)$ 、 $\dots$ 、 $P(n \div 2^m)$ (ただし $n \div 2^m > 1$) で分割する例を示す図である。

[0182] c a r t o n = 1 の場合の例として、図 1 5 には、元データが “a a a a a a a a a b” の場合が示されている。

[0183] 第 2 ～ 1 0 番目のアドレスの位置の一致列 “a a a a a a a a a” を分割するが、この場合 $n = (9 + 1 - 1) \div 1 = 9$ であるので、 $P(4)$ 、 $P(2)$ 、 $P(1)$ で分割する。すなわち、一致列 “a a a a a a a a a” を「 “a” + “a” + “a a” + 残り」と分割する。

[0184] 元データが “a a a a a a a a a b” の場合、圧縮データは “ { 1、a } (1、1) (1、2) (2、3) (5、5) { 1、b } ” となる。

- [0185] $carton=2$ の場合の例として、図 15 には、元データが “a b a b a b a b a b c” の場合が示されている。この場合、第 1～2 番目のアドレスの位置の “a b” が非一致列であり、第 3～11 番目のアドレスの位置の “a b a b a b a b a” が一致列である。第 12 番目のアドレスの位置の文字 “c” は非一致列である。この場合、 $n = (9 + 2 - 1) \div 2 = 5$ であるので、 $P(2)$ 、 $P(1)$ で分割する。すなわち、一致列 “a b a b a b a b a b” を、「“a b” + “a b” + 残り」と分割する。
- [0186] 元データが “a b a b a b a b a b a b c” の場合、圧縮データは “{2、a b}、(2、2) (2、4) (5、4) {1、c}” となる。
- [0187] $carton=3$ の場合の例として、図 15 には、元データが “a b c a b c a b c a b d” の場合が示されている。この場合、第 1～3 番目のアドレスの位置の “a b c” が非一致列であり、第 4～11 番目のアドレスの位置の “a b c a b c a b c a b” が一致列である。第 12 番目のアドレスの位置の文字 “d” は非一致列である。この場合、 $n = (8 + 3 - 1) \div 3 = 3$ であるので、 $P(1)$ で分割する。すなわち、一致列 “a b c a b c a b c a b” を、「“a b c” + 残り」と分割する。
- [0188] 元データが “a b c a b c a b c a b c a b d” の場合、圧縮データは “{3、a b c} (3、3) (5、6) {1、d}” となる。
- [0189] $carton=4$ の場合の例として、図 15 には、元データが “a b c d a b c d a b c d a b c e” の場合が示されている。この場合、第 1～4 番目のアドレスの位置の “a b c d” が非一致列であり、第 5～15 番目のアドレスの位置の “a b c d a b c d a b c d a b c” が一致列である。第 16 番目のアドレスの位置の文字 “e” は非一致列である。この場合、 $n = (11 + 4 - 1) \div 4 = 3$ であるので、 $P(1)$ で分割する。すなわち、一致列 “a b c d a b c d a b c d a b c” を「“a b c d” + 残り」と分割する。
- [0190] 元データが “a b c d a b c d a b c d a b c d a b c e” の場合、圧縮データは “{4、a b c d} (4、4) (7、8) {1、e}” となる。
- [0191] $carton=7$ の場合の例として、図 15 には、元データが “a b c d

e f g a b c d e f g a b c d e f h” の場合が示されている。この場合、 $n = (11 + 7 - 1) \div 7 = 2$ であるので、一致列 “a b c d e f g a b c d e f” を、「“a b c d e f g” + 残り」と分割する。

[0192] 元データが “a b c d e f g a b c d e f g a b c d e f h” の場合、圧縮データは “{7、a b c d e f g} (7、7) (6、14) {1、h}” となる。

[0193] c a r t o n = 8 の場合の例として、図 15 には、元データが “a b c d e f g h a b c d e f g h a b c d e f g i” の場合が示されている。この場合、第 1～8 番目のアドレスの位置の “a b c d e f g h” が非一致列であり、第 9～23 番目のアドレスの位置の “a b c d e f g h a b c d e f g” が一致列である。第 24 番目のアドレスの位置の文字 “i” は非一致列である。この場合、 $n = (15 + 8 - 1) \div 8 \sim 2$ であるので、一致列 “a b c d e f g h a b c d e f g h” を、「“a b c d e f g h” + 残り」と分割する。

[0194] 元データが “a b c d e f g h a b c d e f g h a b c d e f g i” の場合、圧縮データは “{8、a b c d e f g h} (8、8) (7、16) {1、i}” となる。

[0195] <<第 3 実施例 (E x 3)>>

図 16 は、重なりの最小距離 X 、一致長の最小値 N を定めた上で、 $P(1)$ 、 $P(1+2)$ 、 $P(1+2+4)$ 、 $\dots$ 、 $P(2^m-1)$ (ただし $(2^m-1) < n$) で分割する例を示す図である。図 16 では、重なりの最小距離 $X=8$ 、一致長の最小値 $N=3$ の場合が示されている。この場合、一致文字列を分割することで一致長が N 以下となる部分は非一致列として扱う。

[0196] c a r t o n = 1 の場合の例として、図 16 には、元データが “a a a a a a a a a b” の場合が示されている。

[0197] この場合、第 1 番目のアドレスの位置の “a” が非一致列であり、第 2～10 番目のアドレスの位置の “a a a a a a a a a a” が一致列である。第 11 番目のアドレスの位置の文字 “b” は非一致列である。第 2～10 番目の

アドレスの位置の一致列 “a a a a a a a a a” を 1 文字目、3 文字目、7 文字目で分割するが、一致長の最小値 $N = 3$ とするためには、元データ “a a a a a a a a a b” を 「 “a a a a” + “a a a a” + “a a b” 」 と分割する。

[0198] よって、圧縮データは “ {4、a a a a} (4、4) {3、a a b} ” となる。

c a r t o n = 2 の場合の例として、図 1 4 には、元データが “a b a b a b a b a b c” の場合が示されている。この場合、第 1 ～ 2 番目のアドレスの位置の “a b” が非一致列であり、第 3 ～ 1 0 番目のアドレスの位置の “a b a b a b a b” が一致列である。第 1 1 番目のアドレスの位置の文字 “c” は非一致列である。第 3 ～ 1 0 番目のアドレスの位置の “a b a b a b a b” を 1 カートン目、3 カートン目で分割するが、一致長の最小値 $N = 3$ とするためには、元データ “a b a b a b a b a b c” を、 「 “a b a b” + “a b a b” + “a b a b” + “c” 」 と分割する。

[0199] よって圧縮データは、 “ {4、a b a b} (4、4) (4、8) {1、c} ” となる。

c a r t o n = 3 の場合の例として、図 1 6 には、元データが “a b c a b c a b c a b c d” の場合が示されている。この場合、第 1 ～ 3 番目のアドレスの位置の “a b c” が非一致列であり、第 4 ～ 1 2 番目のアドレスの位置の “a b c a b c a b c” が一致列である。第 1 3 番目のアドレスの位置の文字 “d” は非一致列である。第 4 ～ 1 2 番目のアドレスの位置の “a b c a b c a b c” を 1 カートン目で分割するが、一致長の最小値 $N = 3$ とするためには、元データ “a b c a b c a b c a b c d” を、 「 “a b c” + “a b c” + “a b c” + “d” 」 と分割する。

[0200] よって、圧縮データは、 “ {3、a b c} (3、3) (6、6) {1、d} ” となる。

c a r t o n = 4 の場合の例として、図 1 6 には、元データが “a b c d a b c d a b c d e” の場合が示されている。この場合、第 1 ～ 4

番目のアドレスの位置の “a b c d” が非一致列であり、第 5 ～ 16 番目のアドレスの位置の “a b c d a b c d a b c d” が一致列である。第 17 番目のアドレスの位置の文字 “e” は非一致列である。第 5 ～ 16 番目のアドレスの位置の “a b c d a b c d a b c d” を 1 カートン目で分割するが、一致長の最小値 $N = 3$ とするためには、元データ “a b c d a b c d a b c d a b c d e” を、「 “a b c d” + “a b c d” + “a b c d a b c d” + “e” 」と分割する。よって、圧縮データは “ {4、a b c d} (4、4) (8、8) {1、e} ” となる。

[0201] c a r t o n = 7 の場合の例として、図 16 には、元データが “a b c d e f g a b c d e f g a b c d e f h” の場合が示されている。この場合、第 1 ～ 7 番目のアドレスの位置の “a b c d e f g” が非一致列であり、第 8 ～ 21 番目のアドレスの位置の “a b c d e f g a b c d e f g” が一致列である。第 22 番目のアドレスの位置の文字 “h” は非一致列である。第 8 ～ 21 番目のアドレスの位置の “a b c d e f g a b c d e f” を 1 カートン目で分割するが、一致長の最小値 $N = 3$ とするためには、元データ “a b c d e f g a b c d e f g a b c d e f g h” を、「 “a b c d e f g” + “a b c d e f g” + “a b c d e f” + “h” 」と分割する。

[0202] よって、圧縮データは “ {7、a b c d e f g} (7、7) (6、14) {1、h} ” となる。

[0203] c a r t o n = 8 の場合の例として、図 16 には、元データが “a b c d e f g h a b c d e f g h a b c d e f g i” の場合が示されている。この場合、第 1 ～ 8 番目のアドレスの位置の “a b c d e f g h” が非一致列であり、第 9 ～ 23 番目のアドレスの位置の “a b c d e f g h a b c d e f g” が一致列である。第 24 番目のアドレスの位置の文字 “i” は非一致列である。この場合、c a r t o n = 8 は重なり以上の最小距離以上のため分割しない。

[0204] よって、圧縮データは “ {8、a b c d e f g h} (15、8) {1、i} ” となる。

[0205] <<第4実施例 (Ex 4)>>

図17は、循環の1セットが一致長の最小値 $N=3$ 以上になるように先頭を調整する例を示す図である。

[0206] $carton=1$ の場合は、 $carton=3$ となるように一致列の先頭のアドレスの位置を調整する。

[0207] 元データ “a a a a a a a a a b” を「 “a a a” + “a a a” + “a a a a” + “b” 」と分割する。

[0208] よって、圧縮データは、 “ {3, a a a} (3, 3) (4, 6) {1, b} ” となる。

$carton=2$ の場合は、 $carton=4$ となるように一致列の先頭のアドレスの位置を調整する。

[0209] 元データが “a b a b a b a b a c” の場合、「 “a b a b” + “a b a b” + “a b a” + “c” 」と分割する。

[0210] よって、圧縮データは、 “ {4, a b a b} (4, 4) (3, 8) {1, c} ” となる。

[0211] $carton=3$ の場合は、 $carton=3$ は一致長の最小値以上であるので、政党距離は調整しない。

[0212] 元データが “a b c a b c a b c a b d” の場合、圧縮データは、 “ {3, a b c} (3, 3) (5, 6) {1, d} ” となる。

符号の説明

- [0213]
- | | |
|-------|-----------|
| 10、20 | データ圧縮装置 |
| 11、21 | 入力バッファ部 |
| 12、22 | 一致列検索部 |
| 13、23 | 一致長符号化部 |
| 14、24 | 一致位置符号化部 |
| 15、25 | 非一致長符号化部 |
| 16、26 | 非一致位置符号化部 |
| 17、27 | 出力バッファ部 |

2 8 一致列分割部

請求の範囲

- [請求項1] 圧縮前のデータの文字列中の第1の文字列の後に、前記第1の文字列と同一である第2の文字列を一致列として抽出し、前記一致列の長さ、前記第1の文字列が前記第2の文字列の幾つ前のアドレスの位置から開始されるかを示す相対位置を決定する一致列検索部と、
- 前記一致列検索部で抽出された前記第2の文字列から、長さが前記相対位置より小さい第3の文字列を抽出する一致列分割部と、
- 前記第3の文字列の長さを符号化する一致長符号化部と、
- 前記相対位置を符号化する一致位置符号化部と、
- を含む、データ圧縮装置。
- [請求項2] 前記相対距離が所定の値より小さいとき、前記第2の文字列から前記第3の文字列を抽出する、請求項1に記載のデータ圧縮装置。
- [請求項3] さらに、前記一致列検索部は、圧縮前のデータの文字列中の第1の文字列の後に、前記第1の文字列と同一である第2の文字列が存在しない、または第2の文字列が所定の長さより短い文字列を非一致列として抽出し、
- 前記一致列分割部は、前記第3の文字列の長さが所定の値より小さいとき、前記第3の文字列の長さの倍数であって、前記所定の値を超えない最大の値を前記非一致列またはその一部として抽出する、請求項1または2に記載のデータ圧縮装置。
- [請求項4] 前記一致列分割部は、前記一致列内の循環文字列における循環の最小単位をカートンと定義し、前記第2の文字列の先頭から、カートンの整数倍の位置の全てまたは一部で前記第2の文字列を分割する、請求項1～3のいずれか一項に記載のデータ圧縮装置。
- [請求項5] 前記一致列分割部は、前記第2の文字列の先頭から前記カートンを単位として、 $2^m - 1$ （ m を整数）の位置の全てまたは一部で前記第2の文字列を分割する、請求項4に記載のデータ圧縮装置。
- [請求項6] 前記一致列分割部は、`length`を前記一致長、`carton`を

前記カートンの長さとして $n = (\text{length} + \text{carton} - 1) \div \text{carton}$ としたとき、前記第2の文字列の先頭から前記カートンを単位として、 $n \div 2^m$ (m を整数)の位置の全てまたは一部で前記第2の文字列を分割する、請求項4に記載のデータ圧縮装置。

[請求項7]

コンピュータで処理されるデータ圧縮方法であって、

圧縮前のデータの文字列中の第1の文字列の後に、前記第1の文字列と同一である第2の文字列を一致列として抽出し、前記一致列の長さと、前記第1の文字列が前記第2の文字列の幾つ前のアドレスの位置から開始されるかを示す相対位置を決定することと、

前記一致列検索部で抽出された前記第2の文字列から、長さが前記相対位置より小さい第3の文字列を抽出することと、

前記第3の文字列の長さを符号化することと、

前記相対位置を符号化することと、

を含む、データ圧縮方法。

[請求項8]

圧縮前のデータの文字列中の第1の文字列の後に、前記第1の文字列と同一である第2の文字列を一致列として抽出し、前記一致列の長さと、前記第1の文字列が前記第2の文字列の幾つ前のアドレスの位置から開始されるかを示す相対位置を決定させと、

前記一致列検索部で抽出された前記第2の文字列から、長さが前記相対位置より小さい第3の文字列を抽出させと、

前記第3の文字列の長さを検出し、符号化させと、

前記相対位置符号化させる、

処理をコンピュータに実行させることを特徴とするプログラム。