【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第７部門第３区分
【発行日】平成19年11月8日(2007.11.8)

【公表番号】特表2007-508753(P2007-508753A)
【公表日】平成19年4月5日(2007.4.5)
【年通号数】公開・登録公報2007-013
【出願番号】特願2006-534542(P2006-534542)
【国際特許分類】
Ｈ０３Ｍ　7/30　(2006.01)
Ｇ０６Ｆ　12/00　(2006.01)
【ＦＩ】
Ｈ０３Ｍ　7/30　Ｚ
Ｇ０６Ｆ　12/00　５１１Ａ

【手続補正書】
【提出日】平成19年9月21日(2007.9.21)
【手続補正１】
【補正対象書類名】特許請求の範囲
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
所定長以上の長さのバイトシーケンスを有するデータファイルの圧縮方法であって、
二次記憶装置からデータファイルを取り出すステップと、
前記データファイルをダイレクトアクセスメモリに保存するステップと、
前記データファイルの前記所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算するステップと、
前記サブシーケンス内で計算された前記一意なバイト値の頻度を表すデータ値を含むインデックスを前記サブシーケンスに対して作成するステップと、
所定の閾値未満である一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で該一意なバイト値の頻度を増加するために該サブシーケンスに対してデータ変換を適用し、該データ変換を表すデータ値を前記インデックスに追加するステップと、
所定の閾値を超える一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で１以上の一意な値の位置を表すデータ値を前記インデックスに追加するステップと、
ファイルタイプ識別子を持つ出力データファイルを作成するステップと、
前記出力データファイルに前記インデックスを追加するステップと、
を含むことを特徴とする方法。
【請求項２】
前記サブシーケンスに対してデータ変換を適用するステップは、
コンピュータメモリ内に複数の変換データセットを保持するステップであって、該変換データセットは一連のバイト値を持ち、且つ変換データセット識別子によって識別されるステップと、
コンピュータメモリから前記変換データセットの一つを読み出すステップであって、該読み出された変換データセットは前記データファイルのサブシーケンスの長さと実質的に等しい長さを持つステップと、
前記サブシーケンス内の各々のバイトの値に対して、前記読み出された変換データセットにおいて対応するバイトの値に基づいてデータ変換を適用するステップと、

を含む請求項１に記載の方法。
【請求項３】
前記読み出された変換データセットの少なくとも一つに基づいてデータ変換された前記サブシーケンスは、該データ変換前の前記サブシーケンスと実質的に同一である、請求項２に記載の方法。
【請求項４】
前記変換データセットの少なくとも一つは、ランダムに生成されたバイトレートのシーケンスを含む、請求項２に記載の方法。
【請求項５】
前記変換データセットの少なくとも一つは、バイトレートの所定のシーケンスを含む、請求項２に記載の方法。
【請求項６】
前記変換データセットの少なくとも一つは、前記データファイルのサブシーケンス以外の該データファイルの一部から得られたバイト値のシーケンスを含む、請求項２に記載の方法。
【請求項７】
前記サブシーケンスに対して適用された前記データの変換データセットの前記変換データセット識別子を前記インデックスに対して追加するステップをさらに含む、請求項２～６の何れか一項に記載の方法。
【請求項８】
所定長以上の長さのバイトシーケンスを有するデータファイルの圧縮方法であって、
二次記憶装置からデータファイルを取り出すステップと、
前記データファイルをダイレクトアクセスメモリに保存するステップと、
前記データファイルの前記所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算するステップと、
前記サブシーケンス内で前記１以上の一意なバイト値の位置を計算するステップと、
前記サブシーケンス内で計算された前記一意なバイト値の頻度を表すデータ値を含むインデックスを前記サブシーケンスに対して作成するステップと、
所定の閾値未満である一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で該一意なバイト値の頻度を増加するために該サブシーケンスに対してデータ変換を適用し、該データ変換を表すデータ値を前記インデックスに追加するステップと、
所定の閾値を超える一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で前記１以上の一意なバイト値の位置を表すデータ値を前記インデックスに追加するステップと、
ファイルタイプ識別子を持つ出力データファイルを作成するステップと、
前記出力データファイルに前記インデックスを追加するステップと、
を含むことを特徴とする方法。
【請求項９】
前記サブシーケンス内で前記１以上の一意な値の前記位置を計算するステップは、
コンピュータメモリ内にテンポラリポジションインデックスを作成するステップと、
前記サブシーケンスから連続したバイトの値を読み出すステップと、
各バイト値の読み出しにおいて、該読み出されたバイト値が一意なバイト値か繰り返された値かを決定するステップと、
一意なバイト値の検出において、二つのビットの値の一つを前記テンポラリポジションインデックスに追加し、さもなければ、該二つのビットの値の他方を前記テンポラリポジションインデックスに追加するステップと、
前記テンポラリポジションインデックスから、前記１以上の一意な値の前記位置を表すポジションインデックスを作成するステップと、
前記ポジションインデックスから、少なくとも部分的に前記１以上の一意な値の前記位置を表す前記データ値を計算するステップと、

を含む請求項８に記載の方法。
【請求項１０】
前記サブシーケンス内のバイト数が、前記テンポラリポジションインデックス内のビット数と実質的に等しい、請求項９に記載の方法。
【請求項１１】
前記ポジションインデックスのサイズは、前記テンポラリポジションインデックスのサイズよりも小さい、請求項９に記載の方法。
【請求項１２】
前記サブシーケンス内の一意なバイトの値の順序を表す順列インデックスを作成するステップと、
前記ポジションインデックス及び前記順列インデックスの両方から前記１以上の一意な値の前記位置を表す前記データ値を計算するステップと、
をさらに含む請求項９に記載の方法。
【請求項１３】
前記１以上の一意な値の前記位置を表す前記データ値を形成するために、前記ポジションインデックス及び前記順列インデックスを連結させるステップを含む、請求項１２に記載の方法。
【請求項１４】
所定長以上の長さのバイトシーケンスを有するデータファイルの圧縮システムであって、該システムは、
二次記憶装置からデータファイルを取り出し、
前記データファイルをダイレクトアクセスメモリに保存し、
前記データファイルの前記所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算し、
前記サブシーケンス内で計算された前記一意なバイト値の頻度を表すデータ値を含むインデックスを前記サブシーケンスに対して作成し、
所定の閾値未満である一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で該一意なバイト値の頻度を増加するために該サブシーケンスに対してデータ変換を適用し、該データ変換を表すデータ値を前記インデックスに追加し、
所定の閾値を超える一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で前記１以上の一意なバイト値の位置を表すデータ値を前記インデックスに追加し、
ファイルタイプ識別子を持つ出力データファイルを作成し、
前記出力データファイルに前記インデックスを追加する、
ように構成されたことを特徴とするシステム。
【請求項１５】
所定長以上の長さのバイトシーケンスを有するデータファイルの圧縮システムであって、該システムは、
二次記憶装置からデータファイルを取り出し、
前記データファイルをダイレクトアクセスメモリに保存し、
前記データファイルの前記所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算し、
前記サブシーケンス内で前記１以上の一意なバイト値の位置を計算し、
前記サブシーケンス内で計算された前記一意なバイト値の頻度を表すデータ値を含むインデックスを前記サブシーケンスに対して作成し、
所定の閾値未満である一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で該一意なバイト値の頻度を増加するために該サブシーケンスに対してデータ変換を適用し、該データ変換を表すデータ値を前記インデックスに追加し、
所定の閾値を超える一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で前記１以上の一意な値の位置を表すデータ値を前記インデックスに追加し、

ファイルタイプ識別子を持つ出力データファイルを作成し、
前記出力データファイルに前記インデックスを追加する、
ように構成されたことを特徴とするシステム。
【請求項１６】
所定長以上の長さのバイトシーケンスを有するデータファイルを圧縮する方法を実行する実行可能命令を有する、有形記録媒体に保存されたコンピュータプログラムであって、前記方法は、
二次記憶装置からデータファイルを取り出すステップと、
前記データファイルをダイレクトアクセスメモリに保存するステップと、
前記データファイルの前記所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算するステップと、
前記サブシーケンス内で計算された前記一意なバイト値の頻度を表すデータ値を含むインデックスを前記サブシーケンスに対して作成するステップと、
所定の閾値未満である一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で該一意なバイト値の頻度を増加するために該サブシーケンスに対してデータ変換を適用し、該データ変換を表すデータ値を前記インデックスに追加するステップと、
所定の閾値を超える一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で前記１以上の一意なバイト値の位置を表すデータ値を前記インデックスに追加するステップと、
ファイルタイプ識別子を持つ出力データファイルを作成するステップと、
前記出力データファイルに前記インデックスを追加するステップと、
を含むことを特徴とするコンピュータプログラム。
【請求項１７】
所定長以上の長さのバイトシーケンスを有するデータファイルを圧縮する方法を実行する実行可能命令を有する、有形記録媒体に保存されたコンピュータプログラムであって、前記方法は、
二次記憶装置からデータファイルを取り出すステップと、
前記データファイルをダイレクトアクセスメモリに保存するステップと、
前記データファイルの前記所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算するステップと、
前記サブシーケンス内で前記１以上の一意なバイト値の位置を計算するステップと、
前記サブシーケンス内で計算された前記一意なバイト値の頻度を表すデータ値を含むインデックスを前記サブシーケンスに対して作成するステップと、
所定の閾値未満である一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で該一意なバイト値の頻度を増加するために該サブシーケンスに対してデータ変換を適用し、該データ変換を表すデータ値を前記インデックスに追加するステップと、
所定の閾値を超える一意なバイト値の頻度を持つ前記サブシーケンス上で、該サブシーケンス内で前記１以上の一意なバイト値の位置を表すデータ値を前記インデックスに追加するステップと、
ファイルタイプ識別子を持つ出力データファイルを作成するステップと、
前記出力データファイルに前記インデックスを追加するステップと、
を含むことを特徴とするコンピュータプログラム。
【手続補正２】
【補正対象書類名】明細書
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【発明の詳細な説明】
【発明の名称】データ圧縮システム及び方法
【技術分野】

【０００１】
本発明は、データ圧縮の分野に関連し、特に階乗反復型のロスのない圧縮に基づくデータ圧縮システム及び方法に関する。
【背景技術】
【０００２】
電子バイナリファイルには、多くの様々な用途のために多く異なるフォーマットがある。これらのフォーマットには、画像、音、テキスト、データ、実行ファイルなどの保存に適した形式が含まれる。
データを含むバイナリファイルは、暗号化されていなければ、構造化されたフォーマットとなる傾向にある。通常、ヘッダ情報、テキスト、頻度及び他の構成要素間の配置がある。一般に、バイナリファイルの最初の数バイトは、ファイルタイプを表す指標を含むので、どのバイナリファイルを用いるアプリケーションも互換性を有する。実行ファイル若しくは何がしかの機能を実行するために使用されるファイルは、ほとんど構造化されたフォーマットを持たない。しかしながら、これらのファイルは機能を実行するためにオペレーティングシステムと情報をやりとりしなければならないか、オペレーションシステムの一部であるため、構造的要素がある。
【０００３】
圧縮され、暗号化されたファイルは、設計によってはファイル内で繰り返される値が除去されるので、最も構造を持たない。暗号化の場合、置換される値を定義するために鍵が使用される。圧縮については、“省略表現”が繰り返す構造に対して使用される。暗号化されたファイル又は圧縮ファイルの場合、そのファイルは内部構造を持たないばかりか、特に圧縮の場合、ファイルのサイズも変更される。
【０００４】
１，０４８，５７６バイト（１Ｍｂ）のサイズのバイナリファイルに対して、数学的に可能なバイトの配列について$256^{1048576}$通りのとり得る構造がある。実際の使用では、この数の一部のみが使用される。実際に使用される数は、異なるファイルタイプの数の推定、実行ファイル又はオペレーショナルファイルの機能性、及び利用可能な圧縮及び暗号化に基づいて近似するしかない。
【０００５】
データファイル上でデータ圧縮を実行する多くの技術が存在する。幾つかのデータ圧縮アルゴリズムは、インデックス化技術に基づいており、データファイル内の一意の値のインデックス化と計算を含む。最も圧縮されたデータファイルでは、２５６バイトの各コードセグメント内でデータ値の繰り返しが幾つか存在する。平均的なファイルでは、２５６バイトのコードセグメントごとに１６０から１７０個の繰り返しのない一意な値が存在する。階乗計算に基づいたデータ圧縮技術は、この値の数ではうまく動かない。
【発明の開示】
【０００６】
一つの側面において、本発明は、所定長以上の長さのバイトシーケンスを有するデータファイルの圧縮方法を提供する。その方法は、二次記憶装置からデータファイルを取り出すステップと、データファイルをダイレクトアクセスメモリに保存するステップと、データファイルの所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算するステップと、サブシーケンス内で計算された一意なバイト値の頻度を表すデータ値を含むインデックスを作成するステップと、所定の閾値未満である一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で一意なバイト値の頻度を増加するためにサブシーケンスに対してデータ変換を適用し、データ変換を表すデータ値をインデックスに追加するステップと、所定の閾値を超える一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で１以上の一意な値の位置を表すデータ値をインデックスに追加するステップと、ファイルタイプ識別子を持つ出力データファイルを作成するステップと、出力データファイルにインデックスを追加するステップとを含む。
【０００７】

本明細書及び請求項において使用される「含む」という用語は、「少なくとも一部は～からなる」という意味であり、すなわちこの用語を含む本明細書及び請求項の記述を解釈する場合、各記述においてその用語で前置きされた特徴は、全て存在することが必要とされ、また他の特徴が存在することもできる。「有する」、「含まれる」といった関連する用語も、同様に解釈される。

【０００８】

他の側面において、本発明は所定長以上の長さのバイトシーケンスを有するデータファイルの圧縮方法を提供する。その方法は、二次記憶装置からデータファイルを取り出すステップと、データファイルをダイレクトアクセスメモリに保存するステップと、データファイルの所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算するステップと、サブシーケンス内で１以上の一意なバイト値の位置を計算するステップと、サブシーケンス内で計算された一意なバイト値の頻度を表すデータ値を含むインデックスを作成するステップと、所定の閾値未満である一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で一意なバイト値の頻度を増加するためにサブシーケンスに対してデータ変換を適用し、そのデータ変換を表すデータ値をインデックスに追加するステップと、所定の閾値を超える一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で１以上の一意なバイト値の位置を表すデータ値をインデックスに追加するステップと、ファイルタイプ識別子を持つ出力データファイルを作成するステップと、出力データファイルにインデックスを追加するステップとを含む。

【０００９】

他の側面において、本発明は所定長以上の長さのバイトシーケンスを有するデータファイルの圧縮システムを提供する。そのシステムは、二次記憶装置からデータファイルを取り出し、データファイルをダイレクトアクセスメモリに保存し、データファイルの所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算し、サブシーケンス内で計算された一意なバイト値の頻度を表すデータ値を含むインデックスをサブシーケンスに対して作成し、所定の閾値未満である一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で一意なバイト値の頻度を増加するためにサブシーケンスに対してデータ変換を適用し、そのデータ変換を表すデータ値をインデックスに追加し、所定の閾値を超える一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で１以上の一意なバイト値の位置を表すデータ値をインデックスに追加し、ファイルタイプ識別子を持つ出力データファイルを作成し、出力データファイルにインデックスを追加する、ように構成される。

【００１０】

他の側面において、本発明は所定長以上の長さのバイトシーケンスを有するデータファイルの圧縮システムを提供する。そのシステムは、二次記憶装置からデータファイルを取り出し、データファイルをダイレクトアクセスメモリに保存し、データファイルの所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算し、サブシーケンス内で１以上の一意なバイト値の位置を計算し、サブシーケンス内で計算された一意なバイト値の頻度を表すデータ値を含むインデックスをサブシーケンスに対して作成し、所定の閾値未満である一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で一意なバイト値の頻度を増加するためにサブシーケンスに対してデータ変換を適用し、そのデータ変換を表すデータ値をインデックスに追加し、所定の閾値を超える一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で１以上の一意なバイト値の位置を表すデータ値をインデックスに追加し、ファイルタイプ識別子を持つ出力データファイルを作成し、出力データファイルにインデックスを追加する、ように構成される。

【００１１】

他の側面において、本発明は所定長以上の長さのバイトシーケンスを有するデータファイルを圧縮する方法を実行する実行可能命令を有する、有形記録媒体に保存されたコンピュータプログラムを提供する。その方法は、二次記憶装置からデータファイルを取り出すステップと、データファイルをダイレクトアクセスメモリに保存するステップと、データ

ファイルの所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算するステップと、サブシーケンス内で計算された一意なバイト値の頻度を表すデータ値を含むインデックスを作成するステップと、所定の閾値未満である一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で一意なバイト値の頻度を増加するためにサブシーケンスに対してデータ変換を適用し、そのデータ変換を表すデータ値をインデックスに追加するステップと、所定の閾値を超える一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で１以上の一意なバイト値の位置を表すデータ値をインデックスに追加するステップと、ファイルタイプ識別子を持つ出力データファイルを作成するステップと、出力データファイルにインデックスを追加するステップとを含む。

【００１２】

他の側面において、本発明は所定長以上の長さのバイトシーケンスを有するデータファイルを圧縮する方法を実行する実行可能命令を有する、有形記録媒体に保存されたコンピュータプログラムを提供する。その方法は、二次記憶装置からデータファイルを取り出すステップと、データファイルをダイレクトアクセスメモリに保存するステップと、データファイルの所定長を超えない長さを持つサブシーケンス内で一意なバイト値の頻度を計算するステップと、サブシーケンス内で１以上の一意なバイト値の位置を計算するステップと、サブシーケンス内で計算された一意なバイト値の頻度を表すデータ値を含むインデックスを作成するステップと、所定の閾値未満である一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で一意なバイト値の頻度を増加するためにサブシーケンスに対してデータ変換を適用し、そのデータ変換を表すデータ値をインデックスに追加するステップと、所定の閾値を超える一意なバイト値の頻度を持つサブシーケンス上で、サブシーケンス内で１以上の一意なバイト値の位置を表すデータ値をインデックスに追加するステップと、ファイルタイプ識別子を持つ出力データファイルを作成するステップと、出力データファイルにインデックスを追加するステップとを含む。

【発明を実施するための最良の形態】

【００１３】

本発明のデータ圧縮システム及び方法の好ましい形式を、ここで添付図面を参照しつつ説明する。

本発明は、データファイル５に対して適用しようとするデータ圧縮システム及び方法を提供する。データファイル５は、ＢＭＰ、ＷＡＶ、ＤＯＣ、ＸＬＳ、ＭＤＢ、ＺＩＰ、ＳＩＴ、ＡＲＪ、ＺＯＯ、ＴＩＦ、ＪＰＧ、ＧＩＦ、ＭＰ３、ＭＰ４などを含む適当なデータとすることができる。データファイル５を、コンピュータ装置１５の一部を形成するか、少なくとも接続される二次記憶装置１０に保存することができる。コンピュータ装置１５は、ダイレクトアクセスメモリ２５及びディスプレイ３０と接続されるプロセッサ２０を少なくとも含む。コンピュータ装置は、他のコンポーネント、例えばデータ入力装置（図示せず）及び出力装置（図示せず）を含むか、接続されてもよいことを理解されたい。

データファイル５は所定長以上の長さのバイトシーケンスを含むことが想定されている。本発明の一つの好ましい実施形式では、この所定長は３００バイトである。

【００１４】

動作の際、コンピュータ装置１５のプロセッサ２０は、二次記憶装置１０からデータファイル５の全て又は一部を読み出す。読み出されたデータファイル又はその一部は、ダイレクトアクセスメモリ２５に保存される。様々な操作がその保存されたデータファイル又はその一部に対して行われる。得られた出力データファイル３５は、ダイレクトアクセスメモリ２５に作成され、二次記憶装置１０又は他の記憶装置に保存される。多くの場合において、出力データファイル３５は、データファイル５よりも小さいサイズとなることが予想される。

データファイル５のサブシーケンスは最初に検査される。サブシーケンスの長さは、所定長の３００バイトを超えないことが好ましい。特定された一意な値の数が閾値を下回る場合、一連のデータ変換がそのサブシーケンスにおける一意なバイト値の頻度を増加するためにサブシーケンスに対して適用される。

【００１５】
複数のデータ変換コンポーネント４０はダイレクトアクセスメモリ２５又は二次記憶装置に保存される。データ変換コンポーネント４０は、ランダムに生成されたバイト値のシーケンス又は所定のバイト値のシーケンスを複数含んでもよい。そのシーケンスはマスク構造４５として保存される。あるいは、若しくは好ましくは、さらにデータ変換コンポーネントは、追加のマスク構造４５を生成するために使用することができる複数のマスク式５０も含む。データ変換コンポーネントのアプリケーションについては、後で説明する。
【００１６】
システムは、複数のインデックス化コンポーネント６０も含む。データファイル５のサブシーケンスの処理中、出力データファイル３５に同時に書き込まれるインデックス６５が作成される。インデックス化コンポーネント６０はテンポラリポジションインデックス７０、ポジションインデックス７５及び順列インデックス８０を含んでもよい。幾つかの場合においては、ポジションインデックス７５及び順列インデックス８０の内容がインデックス６５に追加される。様々なインデックス化コンポーネント６０の動作については、後で説明する。
【００１７】
システムは、ダイレクトアクセスメモリ２５又は二次記憶装置に保存されるデータアレイ９０を含んでもよい。データアレイ９０は、データアレイ９０の内容が出力データファイル３５に書き込まれる前に、様々なインデックス化コンポーネント６０及び圧縮されるデータファイル５のサブシーケンスの一部を保存するために使用することができる。
【００１８】
図２から図４は、本発明の好ましい形式の動作を示す。バイナリデータファイル５は、複数のデータグループに分割されることが好ましい。本発明の一つの好ましい実施形式では、各データグループは３００バイト以下であることが好ましい。しかし、圧縮されるデータグループのサイズは５ビットを超える如何なるサイズであってもよいことを理解されたい。最初に、データファイルは、データファイルの長さが所定長以上か否かを明らかにするためにチェックされる（ステップ２００）。本発明の一つの好ましい実施形式では、所定長の初期値は３００バイトである。一つの形式では、データファイル全体は二次記憶装置から読み出され、ダイレクトアクセスメモリ２５内のデータアレイ９０に保存される。あるいは、データファイル５の一部を、データストリームとして二次記憶装置１０から読み出してもよい。
【００１９】
データグループは、データグループ内の一意なデータ値の頻度を計算するためにカウントされる（ステップ２０５）。そして一意なデータ値の頻度は、所定の閾値と比較される（ステップ２１０）。一つの好ましい形式では、所定の閾値は２５６である。３００バイトのサブシーケンス内で一意な値が２５６未満の場合、サブシーケンス内の一意なバイト値の頻度を増加するために、１以上のデータ変換をそのサブシーケンスに適用してもよい。
一意なバイト値の頻度が、３００バイト内で所定の閾値の２５６未満の場合、そのサブシーケンスは、データ変換マスクをサブシーケンスに適用可能か否かを識別するためにテストされる（ステップ２１５）。本発明の一つ好ましい形式では、構造ライブラリが、コンピュータメモリ、例えばダイレクトアクセスメモリ２５に保持される。そのライブラリは、複数のランダムに生成されたデータセットを含むことが好ましい。これらのデータセットを、それぞれデータセット識別子によって識別することができ、そのデータセット識別子はコンピュータメモリに保存され、ランダムに生成されたデータセットのそれぞれと関連付けられる。
【００２０】
一つの形式では、ランダムに生成されたデータセットの少なくとも一つは、データファイルのサブシーケンスの長さと実質的に等しい長さを持つ。言い換えれば、サブシーケンスのバイト数は、変換データセット又はマスクのバイト数と同じである。そのようなマス

クは、サブシーケンス内のそれぞれのバイト値に対して、対応するバイト値及び読み出された変換データセットに基づいて、データ変換を適用することにより、サブシーケンスに適用できる。

データ変換の一つの例は、加算剰余（modulus addition）である。サブシーケンスの最初のバイト値とデータセットの最初のバイト値とが加算され、その合計について２５６を法として剰余が計算される。例えば、サブシーケンスの最初のバイナリ値が１６８であり、特定されたデータセットの最初のバイナリ値が２０３の場合、その合計は３７１である。変換値は、３７１ＭＯＤ２５６を計算することにより、１１５である。その後、シーケンスの２番目のバイトが、データセットの２番目のバイトにより同じ方法で変換される。その後、シーケンスの３番目のバイトが、データセットの３番目のバイトにより同じ方法で変換される、などとなる。

この方法で、マスクがサブシーケンスに適用される（ステップ２２０）。

【００２１】

一つの形式において、６５，５３６個のマスク構造をコンピュータメモリに保存しておくことができ、各マスクは0から６５，５３６までのインデックスナンバーという形式のデータセット識別子とともに提供される。そのインデックスは、関連するデータセット識別子を示す単なる１４ビットセグメントとすることができる。

データ変換コンポーネント４０は、マスク式を含むことができる。例えば以下の通りである。

・先の３００バイト以下のデータファイルのシーケンスの標準偏差。この式は前にサブシーケンスのない、データファイルの最初のシーケンスでは使用できないことを理解されたい。

・前回のサブシーケンス又は標準偏差に基づいたサブシーケンス内の値の反転

・サブシーケンスの構造に基づいて計算される適用可能な構造

・関連するサブシーケンスに対して加算又は減算され、ファイル構造に基づいてランダムに生成されるセグメント

【００２２】

上記の式は、一連のマスク構造を予め生成するために適用してもよい。あるいは、データ変換中に関連するバイト値を計算してもよい。一つの形式では、５１２個のランダムに生成された構造又はマスク構造は、ダイレクトアクセスメモリ２５に保存される。これらの構造は、３００バイトシーケンス内で２５６以上のヌル値を持つ可能性のあるデータファイルのサブシーケンスに対して適用される。これは、多くのソフトウェアアプリケーションのバイナリファイルのヘッダ構成部分において一般的である。これらのランダムに生成された構造を、高いレベルの反復をもつ他のフォーマットに適用することもできる。

【００２３】

サブシーケンスでのデータ変換に続けて、サブシーケンスは、３００バイトのシーケンス内で２５６個の一意な値があるか否かを識別するために再度テストされる（ステップ２１０）。２５６個の一意な値がなく、そのサブシーケンスに適用可能な別のマスクがなければ、３００バイトの閾値は下げられ、プロセスはより小さなサブシーケンスで繰り返される。一つの好ましい実施形態では、閾値は、３００未満の８ビット値（バイト）を検査するために、一時的に１５２個の７ビット値又は７７個の６ビット値へ低下させてもよい。それから、閾値は、次のサブシーケンスに対して３００バイトに引き上げられる。これについては以下に詳細を述べる。

【００２４】

ランダムファイルの追加が２５６バイトセグメント内で２５６個の一意な値を生成することは到底無理であり、約１０％が可能なところである。一度適切なランダムファイル構造が適用されると、３００バイトを超えないデータセグメント内で２５６個の一意な値があることが想定される。場合によっては、データ変換の意図はデータグループ内の一意なデータ値の頻度を増加することにある。

【００２５】

本発明は、データグループ内の３００個のデータ値のインデックスを計算する。

インデックスは、ダイレクトアクセスメモリ２５内のデータアレイ９０に保存されることが好ましい。３００個のデータ値のインデックスは、最初に2ビットで生成される。２５６個の一意なデータ値が３００バイトのデータグループ内で特定される場合、そのビット値“０１”がインデックスに書き込まれる（ステップ２２５）。

マスクがサブシーケンスに適用されると、マスク又はデータセット識別子がインデックスに書き込まれる（ステップ２３０）。このマスク識別子は、０から６５，５３６間のマスク値を識別する１６ビット値であることが好ましい。マスク識別子において値が０であるということは、サブシーケンスに対してマスクが適用されなかったか、ヌルマスクが適用されたことを表す。ヌルデータセットがサブシーケンスに適用される場合、データ変換後のサブシーケンスは、データ変換前のサブシーケンスと実質的に同一である。

【００２６】

本発明の方法において、次のステップは、テンポラリポジションインデックスを作成することである（ステップ２３５）。

テンポラリポジションインデックスの作成方法は、データグループの最初のバイトにおいて開始し、２５６個の一意な値が３００バイトのデータグループから抽出される場合、２５６個の一意な値が特定されるまで、データグループ内の次のバイトを検査する。検査される特定の値が、そのデータグループ又は以前のデータグループ内ではじめて現れる値である場合、“１”ビット値がテンポラリインデックスに加算される。一方、検査されるデータ値が既出のデータ値の繰り返しである場合、“０”ビット値がインデックスに書き込まれる。インデックス化の方法は、２５６個の“１”ビットがインデックスに書き込まれると、直ちに終了する。

【００２７】

テンポラリインデックスは、作成される圧縮データストリームにおいて、データグループ内の各データ値の配置と識別を容易にする。インデックスで値“１”の数は、使用されているビット数を示す。例えば、２５６個の“１”の値が、テンポラリインデックス内の２８３個のエントリの後のテンポラリインデックスで発生した場合、これはサブシーケンスの２８３バイト内に２５６個の一意なバイト値があることを示す。

３００バイトのデータグループ内に２５６個以上の値がある場合、インデックスの最初の2ビットは既に“０１”にセットされている。テンポラリインデックスはメインのインデックスに対して単に加算されるが、一方、この情報を格納する、より効率的な方法がある。サブシーケンスにおいて出現する“１”の値の数は既知である。それらが出現する順番を無視するならば、一意なバイト値の発生数を記録するだけで十分である。

【００２８】

テンポラリインデックスそれ自体を記録するよりも、ポジションインデックスを生成し、このポジションインデックスをメインのインデックスに書き込む方が好ましい。３００バイトのサブシーケンスに対して、テンポラリインデックスが、２５６個の“１”の値とそれに続く４４個の“０”の値を含む場合、これにポジションインデックス“０”を割り当ててもよい。４４個の“０”の値と２５６個の“１”の値が３００バイトのデータグループ内で取り得る配置の数は${}^{n}C_{r}$である。これは、２５６個の“１”の値と４４個の“０”の値がある場合、３００個の値内に1.34×10^{53}と等しい３００！／（２５６！・４４！）通りの組み合わせがあるということを意味する。

1.34×10^{53}個のこの最大ポジションインデックス値は、2^{177}よりも小さく、値を表現するために１７７ビットを必要とする。

これは、３００ビットの実際のテンポラリインデックスを保存するよりも、テンポラリインデックス内に少なくとも２５６個の“１”の値があるという事実を利用することによって、代わりにポジションインデックスは１７７ビット若しくは２２．１２５バイトで記録できるということを意味する。

【００２９】

圧縮だけでなく解凍も可能にするために、データグループ内のデータ値の順序も記録す

ることが重要である。これは順列インデックスを生成し、この順列インデックスをメインのインデックスに書き込むことにより達成される（ステップ２４５）。

順列インデックスの計算は、２５６個の一意な値を並べることができる方法の数、すなわち繰り返しのない２５６個の値の順列に基づく。最初の値に対しては、２５６通り有り、２番目の値に対しては２５５通り有り、３番目の値に対しては２５４通り有る、などである。これは２５６！として表され、“２５６の階乗”として参照される。そのため２５６個の一意な値の可能な順列の数は、８．５７×１０^{５０６}通りである。この値は、２^{１６８４}が８．５７×１０^{５０６}よりも大きい８．６×１０^{５０６}と等しいので、１６８４ビットで表すことができる。１６８４ビットは２１０．５バイトと等価である。

シーケンス０，１，２，３，４．．．，２５４，２５５は順列番号１で表し、シーケンス２５５，２５４，２５３．．．，３，２，１，０は順列番号８．５７×１０^{５０６}で表す。

【００３０】

順列インデックスは、メインのインデックスに書き込まれる。メインのインデックスは、これまでにサブシーケンス内の一意なビット値の計算された頻度を表すデータ値を含む。これは、ビット値“０１”、続いて適用されたマスクを表す１６ビット、その次にポジションインデックスを表す１７７ビット、その次に順列インデックスを表す１６８４ビットが続く。

【００３１】

十分な長さのサブシーケンスを得るにはデータファイル内の残りのビットが十分でないか、一意な値の残りが十分でないところに到達すると、インデックスは出力ファイルに書き込まれる（ステップ２５０）。

出力ファイルは、ファイルタイプを識別するために先頭３バイトを含むことが好ましい。さらにファイルタイプ識別子に続く２バイトは、最大６５，５３６回に対する特定のデータファイル全体にわたって実行された本発明の方法の試行数を示す。

これらの５バイトに続いて、データアレイ９０に保存されるインデックスが出力ファイルに加えられる。インデックスに続いて、インデックスで未処理な、すなわちデータファイル内に残るビットの値又は一意な値が十分でない値が加えられる。

たいていの場合、ヘッダの５バイトと、それに続く本体と、出力ファイルの終わりに未圧縮の形式でフルに書き込まれる６３以下のビットの値とが存在することが予想される。出力ファイルの本体は、ストリーミングの手法で抽出することを容易にするために連続して書き込まれるインデックスの集合であることが好ましい。

【００３２】

図２について上述してきたように、本発明の方法を複数回繰り返した後にデータファイルに残るバイトが３００もないか、２５６個の一意な値及び適用可能な別のマスクがない３００バイトのシーケンスがある場合も発生する。ステップ２６０で示されるように、一つの好ましい形式では、データファイルから読み出されるサブシーケンスのサイズを減らすことができる。

【００３３】

図３を参照すると、データファイルに少なくとも１３３バイトが残っているか否かを識別するために、データファイルはチェックされる（ステップ３０５）。

データファイルに残る１５２個の７ビット値を含む少なくとも１３３バイトが有る場合、１５２個の７ビット値内で一意な値の数をカウントする（ステップ３１０）。その後、一意な値の数が、閾値（例えば１２８）に対してチェックされる（ステップ３１５）。１３３バイトのサブシーケンスに十分な一意の数がなければ、図２のステップ２１５及び２２０と同様の方法で適用可能なマスクが特定され（ステップ３４０）、適用される（ステップ３４５）。

一度、一意な値の数の閾値がデータファイルの１５２個の７ビット値に対して確認されると、ビットシーケンス“１０”がインデックスに書き込まれ（ステップ３５０）、本方法は図２において２３０で示されるステップへ進む。

【００３４】
データファイル内に処理されるべき残りが１５２個の７ビット値もないか、１５２個の７ビットサブシーケンス内に１２８個の一意な値を見つけられず、適用可能な別のマスクもない場合、ステップ３５５に示されるように、本方法は図４に示されるものへ渡される。図４に示すように、検査下にあるデータファイルのビットグループの数は、７７個の６ビット値に減らされる。データファイルに７７個の６ビット値が残っている場合（ステップ４０５）、その７７個の６ビット値において一意な値の数がカウントされる（ステップ４１０）。
【００３５】
一意な値の数が閾値６４に対してチェックされる（ステップ４１５）。７７個の６ビット値において一意な値が６４よりも少なければ、本方法はマスクが適用可能か否かを明らかにする（ステップ４２０）。マスクが適用可能であれば、マスクが適用される（ステップ４２５）。これら最後の２ステップ（４２５、４３０）は、図２のステップ２１５と２２０、及び図３のステップ３４０と３４５と同様である。
７７個の６ビットサブシーケンスにおいて６４個の一意な値があれば、値“１１”がインデックスに書き込まれる（ステップ４３０）。そして制御は図２のステップ２３０へ戻される。
データファイル内に処理されるべき残りが７７個の６ビット値もないか、７７個の６ビットサブシーケンス内に６４個の一意な値を見つけられなければ、ビット値“００”がインデックスに書き込まれ（ステップ４３５）、インデックスは図２に示されるステップ２５０と同様の手法で出力ファイルに書き込まれ（ステップ４５０）、且つデータファイルの残りのバイトは出力ファイルに書き込まれる（ステップ４６０）。
【００３６】
検査下にあるバイト数に依存して、図２のステップ２４５で示される順列インデックスに対しても少々変更を必要とすることを理解されたい。１５２個の７ビットグループ内で１２８個の一意なデータ値が有る場合、ポジションインデックスは、１５２！／（１２８！・２４！）通りとなり、これは５．４８×10^{27}と等しい。これは、2^{93}＝９．９×10^{27}なので、９３ビットで表すことができる。
７７個の６ビットグループ全体にわたって６４個の一意な値が有る場合、インデックスは７７！／（６４！・１３！）通りとなる。これは、１．８４×10^{14}となる先の値よりも大きい２．８１×10^{14}＝2^{48}なので、４８ビットで表すことができる。
【００３７】
同様に、検査下にあるバイト数に依存して、図２のステップ２４５で示される順列インデックスに対しても少々変更が必要となる。１２８個の値に対する順列は、１２８！すなわち３．８６×10^{215}である。これは、2^{717}＝６．８９×10^{215}なので、表現するために７１７ビットを必要とする。
６４個の値に対する順列は、６４！すなわち１．２７×10^{89}である。これは、2^{296}＝１．２７×10^{89}なので、２９６ビットで表現することができる。
【００３８】
図５は、３７７バイト、３５０バイト、３２０バイト、３００バイト（８ビットグループ）、１５２（７ビットグループ）及び７７（６ビットグループ）のデータグループサイズにおける想定される結果の表を示す。この表に示されるのは、バリエーションに包含される効果の指標である。これについては後で説明する。
【００３９】
解凍は、上記の手順を単に反転するものである。インデックスの値は、最初から最後（２５６番目）までの各値の範囲を示す。範囲を備えることは関連する値を提供する。インデックスは、ヘッダとともに再構成に使用することができる。全てのコンポーネントは一緒にパックされるので、ストリーミングが使用されることが予想される。
【００４０】
繰り返される値の配置のインデックス化は、より効率的な方法が有る場合、セグメント

に対する“０”と“１”の値のストリングから変更してもよい。例えば、繰り返される値が１個か２個のみの場合、バイト数は２５７か２５８となる。２５７番目のビット又は２５８番目のビットを使用するよりも、最初と最後のバイトがそのセグメントに対して一意となることが知られている。そのため、２５７値の場合には、８ビットが単一の繰り返し値の位置を提供し、１６ビットが２５８バイトセグメントの場合における両方の繰り返し値の位置を提供する。

【００４１】

本方法は、全てのファイルタイプ及び構造に適用することができる。ＰＫＷ’ａｒｅのＺＩＰ製品のようなツールでかなりの量まで圧縮されたファイルタイプ又は構造に対して、本発明の方法は１回の試行では同じレベルに到達しないであろう。しかし、本方法は、同じファイルに対して繰り返し適用し、各回でそのサイズを減らすことができる。試行数すなわち繰り返す数は、処理するハードウェア及び／又はユーザが要求する時間に依存する。

【００４２】

全てのコンポーネントが既知であるため、解凍は極めて高速である。圧縮はランダムデータ構造とのマッチングを必要とするので、解凍は圧縮よりもさらに高速となり得る。

全てのインデックス化は実際のデータそれ自体の中に包含されるので、複数回の解凍ルーチンを同時に実行してもよい。

他のアプリケーションは、ソフトウェア圧縮、データ圧縮、ソニープレイステーション２（登録商標）、マイクロソフト（登録商標）Ｘ－Ｂｏｘなどのような対戦型オンラインゲーム、ボイスオーバーＩＰ、ビデオオンデマンドを含む。本発明は、データ又はバイナリ情報が保存され、変換され、若しくは如何なるフォーマットで使用される如何なるアプリケーションも含む。

【００４３】

上記の記述は、３００バイトのコードセグメント内の２５６個の一意な値又はそれよりも小さいものに基づいている。この選択された値は単に説明を目的とするものであることを理解されたい。５ビット又はそれ以上のデータグループ、又は０から３１の間の値はこの方法を用いて再構成することができる。ランダムに生成されたデータセット又はオーバーレイファイルの数を減らすことは、３及び４ビット値がよく使用されるということを意味する。

説明してきた８ビット（２５６値）よりも大きなビット値を用いて、より大きな削減を行うことができる。例えば、９ビット値で圧縮された場合、８ビットの圧縮により達成される圧縮をさらに超える圧縮ゲインが有る。

【００４４】

削減若しくは圧縮は、値に対して使用されるビット数とともに増大する。２５６値（３００バイトセグメント）は、５１２値（６００バイトセグメント）ほど圧縮しない。同様に、５１２値（６００バイトセグメント）は、１０２４値ほど圧縮しない。計算はファイルサイズに基づくはずなので、上限レベルはない。

上記の３００バイトの方法を用いて、これを３７７バイトグループに拡張することができる。これは、図５に示され、且つこの明細書で説明される好ましい実施形態に対して最適レベルである３００バイトを用いて、効果のある範囲が２５６から３７７バイトのグループであることを意味する。

【００４５】

３００個の８ビットグループ（バイト）のバリエーション、１５２個の７ビットグループ及び７７個の６ビットグループを、圧縮されたファイルのヘッダに示してもよい。そのバリエーションは、二つのパートから構成されてもよい。それらは、以下のものである。１．セグメントサイズに対する関連するビットグループの数の表示。８ビットグループに対するサイズの範囲は２５６から３７７であり、７ビットで表され得る。７ビットグループに対する範囲は５ビットで表され、６ビットグループに対する範囲は４ビットで表され得る。

２．ビットグループのそれぞれ内で変更があるか否かを示すために上記のそれぞれの終端に追加され得る追加ビット。“０”は変更無しを示し、“１”は変更有りを示すことができる。

ヘッダは、上記の値を示す追加の１９ビットを含んでもよい。

ヘッダごとに、変更値が許されるならば、グループごとを基礎として変更値をインデックスに書き込んでもよい。

例えば、８ビットグループの一つのグループのデフォルトは３００の値としてもよいが、各セグメントは、含まれる変更値によって示されるように、２５６から３７７の間で変化してもよい。

【００４６】

本発明の別の実施形態は、複数の繰り返しバイト圧縮エンハンスメントを含む。これは図６及び図７を参照して説明される。

機能的電子ファイルは、複数の異なるカテゴリのバイト構造に分類される。これらは単純な２色のビットマップから、これまでに利用可能なロスのない圧縮アルゴリズムを用いて圧縮されたファイルまで、様々である。

【００４７】

ヘッダ情報に続く、２色のビットマップについては、ビット値１は黒を意味し、他は白を意味する。多くの繰り返しが有るので、ロスのない手法でのこれらのファイルの圧縮は単純である。

２４ビットのビットマップに移ると、パターンの識別はより困難となり、そのためロスのない圧縮の圧縮率は、現在のアルゴリズムを用いて、より単純なビットマップ構造上での圧縮率と同程度に大きくはならない。

【００４８】

ここに説明するプロセスは、２４ビットのビットマップよりも単純なパターンを導入し、それは、標準的な写真タイプの画像に対して、圧縮量を飛躍的に増大させるように、現在利用可能なロスのない圧縮アルゴリズムを用いてロスのない圧縮を可能にする。

【００４９】

これを達成するために、図６の６１０に示されるように、オリジナルのイメージは３個の成分に分解され、結合されたもののサイズはオリジナルのイメージよりも非常に大きくなる。それから、６２０に示されるように、全ての３バイト(２４ビット）グループは、１０進数の昇順に並べられる。例えば、２３６，２１７，６７は、６７，２１７，２３６に並べ替えられる。バイト配置の変更は、ハフマン構造を用いてインデックスに記録される。

オリジナルの構造が６通りしかないので、これらは以下のビットインデックスを用いて記録される。

００＝１２３
０１＝１３２
１００＝２１３
１０１＝２３１
１１０＝３１２
１１１＝３２１

上記の数のそれぞれは、バイトの並び替えられた位置と比較したときのそのバイトのオリジナルの位置を表す。

【００５０】

６２５に示されるように、イメージが完全に走査されると、このインデックスはファイル（ファイルＡ）に書き込まれる。

６３０に示されるように、最小のものの全て、又は各グループから現在の最初のバイトの値が別個のファイル（ファイルＢ）に書き込まれる。

６３５に示されるように、バイトの値は順番に並んでいるので、２番目のバイトの値から最初のバイトの値を引いた値がファイル（ファイルＣ）に書き込まれ、直ぐに３番目の

バイトから２番目のバイトの値を引いた値が続く。
これで、３個のファイル、ファイルＡ、ファイルＢ、ファイルＣが生成される。ファイルＢとファイルＣを合わせた合計は、オリジナルの２４ビットのビットマップと同じである。ファイルＡはバイトのインデックスを表すので、ファイルＡはサイズにおいて余分のオーバーヘッドである。
【００５１】
その後、全ての３個のファイル（Ａ，Ｂ及びＣ）がロスのないアルゴリズム又はＷＩＮＺＩＰ６４０のような製品を用いて一つのファイル（６５０）に圧縮されると、得られたファイルは、未修正のイメージファイルにこれらのツールを単に適用することによって得られるものよりも平均で２５％小さい。
テストでは、最悪の場合のシナリオで２．５％の低下を示し、最高の場合、画質が２４ビットのトゥルーカラービットマップの８２％であった。同じゲインはＪＰＥＧのロスなしの圧縮モードを用いて得ることができる。
このプロセスは、データを保持する３バイトのグループ分けを用いて如何なるファイル構造にも適用できる。それはまた、ロスのない圧縮のレベルをより大きくしていくために、４、５、６、７、８等のバイト構造をカバーするように拡張してもよい。
【００５２】
ビットマップファイルが、イメージを表示するために使用されるように、Ｗａｖｅ（.ｗａｖ）ファイルは、音を出すために使用される。圧縮のエンハンスメントプロセスの別の例を、図７を参照しつつＷａｖｅフォーマットファイルに関してここで説明する。より多い色又は質を提供するそれぞれのビットマップファイル（２ビット、４ビット、８ビット、１０ビット、１２ビット、１６ビット、２４ビット、３０ビット）の異なるレベルがあるように、同じことがＷａｖｅファイルにも生じる。
Ｗａｖｅファイルは複数の成分を用いて作成され、その成分は平均サンプリングレート、サンプリングレート、オーディオサンプルサイズ及びチャンネル数である。
サンプリングレートが低いほど、ファイルサイズも小さく、質も劣化することを意味する。また、モノラルファイルはステレオファイルよりも小さい。
【００５３】
ここで取り上げるＷａｖｅフォーマットは、商用ＣＤに最高品質のステレオ音楽を保存するときに使用されるフォーマットである。このフォーマットはＷａｖｅフォーマットからＣＤフォーマットに変換される。
平均データレートが１７６．４Ｋｂ／秒、サンプリングレート４４．１ｋＨｚ、オーディオサンプルサイズ１６ビット、２チャンネル（ステレオ）のＷａｖｅファイルに対して、次のように適用される。
【００５４】
ファイル内の全てのバイトの値が、ｎをファイルの最後のバイトとして（通常のオーディオファイルに対して、これは５０，０００，０００のオーダとなるであろう）、１からｎの番号で表される場合、全ての偶数の位置のバイトの値が一つのファイル（ファイル１）に書き込まれ（７２５）、全ての奇数の位置のバイトの値が別のファイル（ファイル２）に書き込まれる（７３０）。例えば、以下の表の通りとなる。
【表１】

バイト値	255	167	33	0	0	24	24	167	167
順序値	1	2	3	4	5	6	7	8	9
ファイル１ (odd)	255	33	0	24	167				
ファイル２ (even)	167	0	24	167					

それから、両方のファイル（ファイル１及びファイル２）が、再度６４０で示されるよ

うに、ロスのないアルゴリズム又はＷＩＮＺＩＰのような製品を用いて一つのファイルに圧縮されると、得られたファイル６５０は、未修正のイメージファイルにこれらのツールを単に適用することによって得られるものよりも平均で２０％小さい。

テストでは、圧縮ファイルのサイズにおいて最悪の場合のシナリオで１０％のさらなる低下を示し、最高の場合、サイズにおいて４３％の低下であった。

抽出／解凍は単純であり、二つのファイルが関連するロスのないツールを用いて解凍された後に、ファイル２からのバイトがファイル１のバイトのそれぞれの間に挿入される。

【００５５】

上記は、その好ましい形式を含む本発明を説明する。当業者にとって明らかな変更や修正は、添付の特許請求の範囲で規定されるように、本発明の範囲に含まれることが意図されている。

【図面の簡単な説明】

【００５６】

【図１】本発明のシステムの好ましい形式を示すである。

【図２】本発明の好ましい形式のデータ圧縮プロセスのフローチャートである。

【図３】本発明の好ましい形式のデータ圧縮プロセスのフローチャートである。

【図４】本発明の好ましい形式のデータ圧縮プロセスのフローチャートである。

【図５】本発明の好ましい実施形態について想定されるデータ圧縮結果の表を示す図である。

【図６】複数の繰り返しバイト圧縮エンハンスメントに関する本発明のさらなる側面を表す図である。

【図７】複数の繰り返しバイト圧縮エンハンスメントに関する本発明のさらなる側面を表す図である。

【手続補正３】

【補正対象書類名】図面

【補正対象項目名】図７

【補正方法】変更

【補正の内容】

【図７】

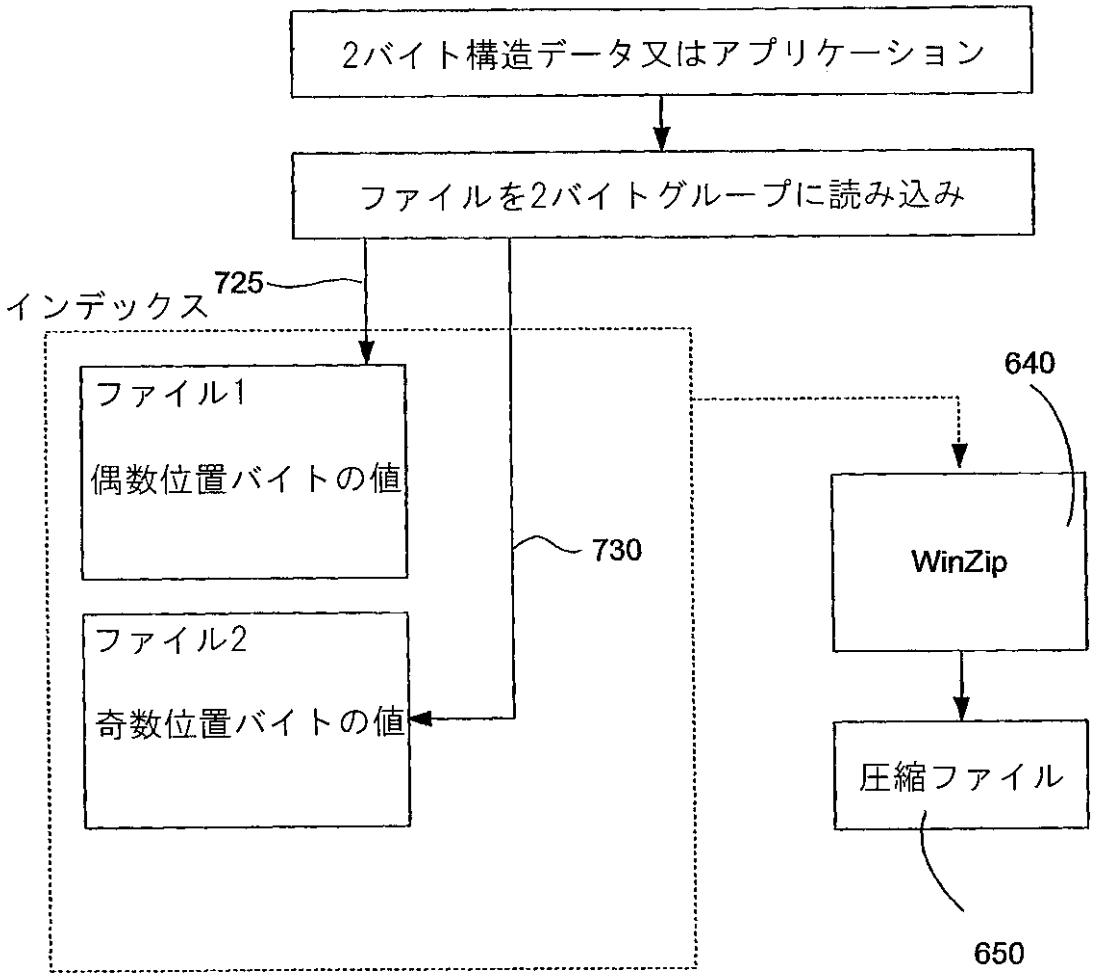

2バイト構造データ又はアプリケーション
ファイルを2バイトグループに読み込み
725
インデックス
ファイル1
偶数位置バイトの値
730
ファイル2
奇数位置バイトの値
640
WinZip
圧縮ファイル
650

FIGURE 7