レンジ・コーダを用いて圧縮率を改善する LZ77符号によるファイルの圧縮とその改良

前編

広井 誠

ファイルを圧縮する場合，Windowsではzipや LHAが標準ツールとして一般に使われています．UNIXではgzipがよく使われますが，最近ではbzip2で圧縮されたファイル（拡張子が.bz2のファイル）もよく見かけるようになりました．一般的なファイルをbzip2で圧縮すると，LHA，zip，gzipよりも高い圧縮率になります．

bzip2は「ブロック・ソート（BlockSorting）」という方法を使って高い圧縮率を達成しています．これに対し，LHA，zip，gzipはLZ77符号とハフマン符号を組み合わせた方法です．LZ77符号はJ.ZipとA.Lempelが開発した「辞書に基づく符号化方式（辞書法）」の一つで，入力された記号列を辞書に登録し，その辞書を使って符号化を行うものです．辞書法は多くの圧縮ツールで用いられているオーソドックスな方法です．

一般に，辞書法の圧縮率は「ブロック・ソート法」や「統計型手法」よりも圧縮性能が劣るといわれています．実際，ブロック・ソートやPPM（Prediction by Partial Matching）を用いて圧縮ツール[(5)(6)(7)]を作成すると，LHAやzipよりも高い圧縮率を達成することができます．

それでも，辞書法が古びた圧縮アルゴリズムになったわけではありません．LZ77符号は辞書の探索に時間がかかるため，辞書のサイズを大きくすることは現実的ではありません．しかし近年になって，コンピュータの性能が著しく向上したことにより，大きな辞書を用いることが可能になりました．

LZ77符号の場合，辞書を大きくしたからといって簡単に圧縮率が向上するわけではありません．辞書を大きくすると，逆に圧縮率が低下する場合もあるからです．このため，辞書を大きくするにしても，高い圧縮率を達成するためのくふうが必要になります．

また，ハフマン符号の代わりにレンジ・コーダ（RangeCoder）という符号化法を用いると，圧縮率をさらに向上させることが可能です．そこで本稿では，LZ77符号の基本を解説し，圧縮率を改善するための方法について詳しく説明します．

LZ符号の基礎知識

● LZ符号は辞書を用いた圧縮

LZ符号は「辞書に基づく符号化（dictionary-based coding）」といい，J.ZipとA.Lempelが開発した圧縮アルゴリズムです．

なお，ハフマン符号は，出現頻度の高い記号には短い符号語を，低い記号には長い符号語を割り当てることでデータ圧縮を実現しています．

話をLZ符号に戻します．たとえば，「ALICE ’S ADVENTURES IN WONDERLAND」（不思議の国のアリス）を例に考えてみましょう．このテキストには2文字以上の英単語が約3000種類含まれています．これをASCIIコード順に並べた辞書を定義します．すると，単語の位置情報は12ビットで表すことができます．2文字の単語はASCIIコードで16ビットなので，単語を位置情報に変換すれば3/4に圧縮することができます．この方法は長い単語になるほど圧縮効果が高くなります．8文字の単語は64ビットなので，18.75%にまで圧縮することができます．

● 辞書を前もって用意するか，その場で作成するか

もしも，ファイルごとに専用の辞書を用意すれば，高い圧縮率を達成できるはずです．符号化に先だって辞書を編集しておいて，その辞書に基づいて符号化を行う方法を「静的辞書法（static dictionary method）」といいます．ところが，この方法では符号化と復号において同じ辞書を用意しなければなりません．復号するための辞書をファイルに添付する方法では，圧縮率の大幅な低下は避けられません．

この問題を解決する方法が，「適応型辞書法（adaptive dictionary method）」です．この方法は前もって辞書を用意せずに，ファイルを読み込みながら辞書を作成していきます．そして，辞書に登録されている記号列が現れたら，辞書の位置情報に変換することで圧縮を行います．最初は辞書が空の状態なので圧縮することはできませんが，ファイルを読み込むに従って十分な記号列が辞書に登録されるので，高い圧縮率を実現できます．

● LZ77符号とLZ78符号

そして，現在もっとも有名な適応型辞書法が「LZ符号」なのです．LZ符号には多数のバリエーションが存在しますが，LZ77符号とLZ78符号の二つに大別されます．LZ77符号は1977年に，LZ78符号は1978年に開発されました．

なお，LZ77符号とLZ78符号は，辞書の作成方法がまったく異なっているので，混同しないように注意してください．LZ77符号は「スライド辞書法」，LZ78符号は「動的辞書法」と呼ばれています．

KEYWORD ——LZ77，LZ78，レンジ・コーダ，LZSS符号，LZB符号

LZSS符号 ――スライド辞書と最長一致法を使用

● LZSS符号はLZ77符号の一つ

LZ77符号にはたくさんのバリエーションが存在しますが，その中でもっとも基本的で広く用いられている符号がLZSS符号です．参考文献(1)によると，「現在普及している圧縮プログラムでLZ77符号と呼ばれているもののほとんどは，じつはLZSS符号になっています」とのことです．そこで，本稿でもLZSS符号を用いて説明します．

● スライド窓とは

LZSS符号は，読み込んだ記号列をバッファに格納し，それを辞書として利用します．バッファの大きさは有限なので，新しく記号列を読み込んだ分だけ，古い記号列を捨てていかなければなりません．この動作はファイルを一つの記号列として考えると，その上をバッファがスライドしていくように見えることから，このバッファを「スライド窓」と呼びます．**図1**にスライド窓のイメージを示します．

スライド窓は参照部と符号化部からなり，符号化部が圧縮対象となる記号列です．LZSS符号は参照部の中から符号化部ともっとも長く一致する記号列(最長一致系列と呼ぶ)を探して，その位置情報と長さで符号化を行います．**図1**のスライド窓は，参照部の大きさが16で，符号化部の大きさが4に設定されています．

たとえば，スライド窓の参照部が8192で符号化部を16とすると，位置情報が13ビット，長さの情報が4ビットになるので，合計17ビットで符号語を表すことができます．この場合，2文字以下の記号列は圧縮できないので，3文字以上の記号列に対して符号化を行うことになります．

最長一致系列の長さが3文字未満の場合は，符号化部の先頭文字をASCIIコードで出力します．LZSS符号は位置と長さを表す符号語と記号を表す符号語を区別するため，**図2**に示すように1ビットのフラグを符号語の先頭に付加します．復号する場合は最初にフラグを1ビット読み込み，その値が1ならば長さと位置を復号し，0ならば記号を復号します．

● LZSS符号の符号化

それでは，具体的に符号化のようすを説明します．

図3に示すように，参照部にはすでに記号列が読み込まれているとします．この中から最長一致系列を探します．参照部は0から数えるとすると，**図3**(**a**)のように5番目の“ IS ”が見つかります．符号語は記号列の長さと位置を示すフラグ‘ 1 ’と，長さ3から1を引いた‘ 10 ’と，位置情報5を表す‘ 0101 ’の7ビットで表すことができます．

次に，最長一致系列の長さだけ新しい記号列を読み込みます．この場合，スライド窓を3文字分右へ動かします．今度は**図3**(**b**)のようにスライド窓の2番目の位置にある“ IS ”と一致するので，‘ 1_10_0010 ’と符号化します．そして，3文字分だけスライド窓を移動します．

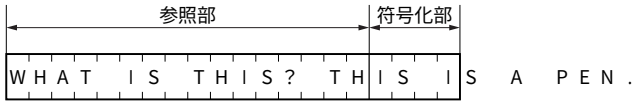
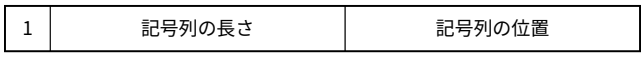
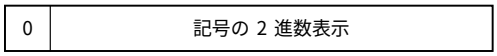


図1　スライド窓のイメージ



(**a**) 最長一致系列の長さが3以上の場合



(**b**) 最長一致系列の長さが3未満の場合

図2　LZSS符号の符号語

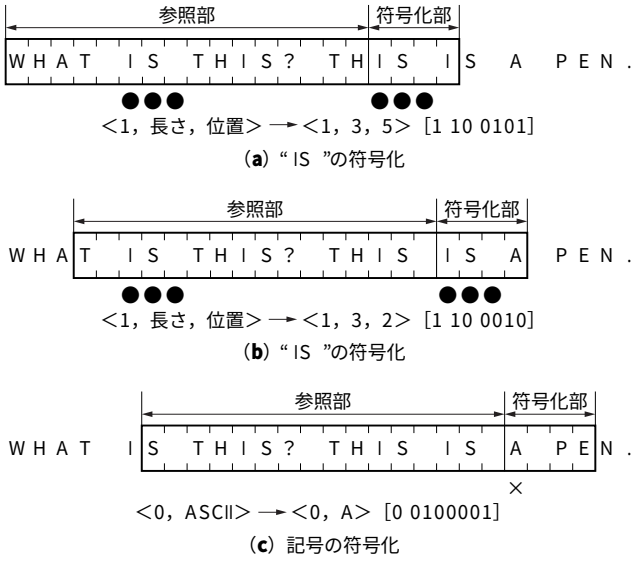


図3　LZSS符号の符号化

図3(**c**)の参照部には，符号化部の先頭記号Aが存在しないので，この場合は不一致となります．記号を示すフラグ0と記号AのASCIIコードを符号語として出力します．そのあと，1文字分だけスライド窓を移動します．あとは，符号化部に記号がなくなるまで，この処理を繰り返します．

● LZSS符号の復号

次に，復号のようすを説明します．**図4**を見てください．

復号する場合，スライド窓は参照部だけあればよく，符号化部は必要ありません．参照部には“ WHAT IS THIS? TH ”まで復号されているとします．

符号語の最初の1ビットは1なので，次の2ビットが長さを，その次の4ビットが位置情報を表しています．この場合は<3,5>なので，参照部の5番目から3文字“ IS ”と復号できます．これをファイルへ出力するとともに，参照部へ追加してバッファを更新します〔**図4**(**a**)〕．このとき，古い文字(左側の文字)から