ＪＡＶＡによる圧縮・解凍ソフトの開発

栗原 隆浩

Takahiro KURIHARA

名古屋文理大学　情報文化学部　情報文化学科　はせがわ研究室

HASEGAWA Laboratory, Department of Information Culture, Nagoya Bunri University

平成１６年1月３０日　提出

要旨

現在、インターネットの利用者の増加と家庭への PC の普及により、データ圧縮・解凍技術が一般利用者にも浸透し、特に圧縮・解凍ソフトがインターネットを通して配信され、また HP 上の BBS 等は技術交換の場として広がった。

今回は、特にデータ圧縮技術に注目し、3 つの圧縮法（ランレングス法・ハフマン法・特定条件の圧縮法）による圧縮及び解凍ソフトを開発し、テキストデータにおける圧縮効果の比較を行った。また将来的に、ネットワークを介したソフトに発展させるため、Web 上で動作ができる Java 言語を利用した。

調査の結果、テキストデータにおいて０と１のランレングス法では、今回のプログラムではデータ量が逆に増加し圧縮は不可能になってしまった。ハフマン法では圧縮可能なのだが、文字の出現頻度に左右され低圧率になることもある。それらに対して、今回考案した特定条件の圧縮法は、仮名のみで構成されているテキストデータに適応すればテキストの内容に左右されず常にほぼ 50%の圧縮率を実現することがわかった。

１．はじめに

圧縮技術はいかに元のデータより小さくするかを目的とし、様々なアルゴリズムが考え出されてきた。そして、近年ＰＣ・インターネットの爆発的な広がりにより多くの人に圧縮技術のノウハウの一般化と共に広がり、一般の人たちも色々な圧縮方法による圧縮･解凍ソフトを利用するようになってきた。また、インターネットの利用者も年々増え、ネット上で利用しやすい圧縮･解凍ソフトの需要は大きい。

しかし、今現在多く使われている圧縮ソフトの多くは、使用されるＰＣに一つ一つインストールして使用さするものである。

私は、近年のインターネットのブロードバンド化による高速・大容量なデータのやり取りに着目し、圧縮ソフトをあるサイトに置いておき、それにアクセスすることで誰でも利用できるようにならないかと考えた。これにより、一台一台のＰＣに個別ソフトをインストールする必要がなくなり、ソフト自体のバージョンアップ等の管理がしやすいのではな

いかと思った。以上を考えて、開発するに当たりWeb上で動作するプログラムが作成しやすいJava言語を使用することに決めた。

本研究では、まず圧縮技術のみに注目し、圧縮効率について調べた。Javaを使って実際に圧縮・解凍ソフトを開発した。

２．圧縮プログラムについて

今回使用する圧縮方法として、以下の（１）～（３）に示す3つの方法を試した。なお、今回はテキストファイルのみを対象とする。

（１）ランレングス法

ランレングス法とは、例えばデータを「０」又は「１」という二通りしかない状態（バイナリ）で表し、もとのデータを「０」又は「１」がいくつ続いたかという情報に置き換えることにより圧縮する方法である。

圧縮前：
00001111000111011111
↓
圧縮後：
0 4 4 3 3 1 5

図１：0と1のランレングス法の考え方

最初の数値が「0」か「1」かを判定すれば、次に続くのは必ずその逆になる。後は連続した回数を順次保存していけば良くなる。

図１を例にあげれば、圧縮後のデータの最初の「0」は圧縮前のデータの最初の数値が「0」であることを示し、その後は「0」と「1」の連続した回数が続いている。

（２）ハフマン法[1)]

ハフマン法とは、例えば普通8ビットで表す文字データを、頻繁に現れる文字は4ビットで表し、あまり現れない文字は10ビットで表すなどすれば、ファイルの全体の大きさが小さくなるという考え方である。

（３）特定条件の圧縮法

この方法はランレングス圧縮を元に、文字データ2バイトの日本語を圧縮するために、今回、考案したものである。

考え方としては、2バイト文字を前半１バイトと後半１バイトに分け、前半１バイトが共通である文字がいくつ続いたかによって圧縮する方法である。

テキストの中身が「あいあ」の場合、

文字コードは、
あ＝（00110000 01000010）2
い＝（00110000 01000100）2
であり、
「あ」は前半（00110000）2＝４８と
後半（01000010）2＝６６に、
「い」は前半（00110000）2＝４８と
後半（01000100）2＝６８になる。

上のように「あ」と「い」の前半１バイトが共通のため、保存されるのは
03 48 66 68 66
となる。

これにより、元６バイトデータが５バイトに圧縮される。

図２：特定条件の圧縮法の考え方

図２に例を示すように、圧縮前のテキストの中身が「あいあ」の場合、「あ」と「い」の前半１バイト（8ビット）は「00110000」（10進数で４８）と共通している。この圧縮方法は、基本的にランレングスの考え方を前半１バイトのみに適用したものである。すなわち、前半１バイトの数値の同じ文字が何文字連続しているかに着目して圧縮する。「あいあ」の場合、前半１バイトに３回「４８」が続くので、図２のように、圧縮後のデータの初めは「03 48」となり、後は、後半１バイトのデータを順次処理していくと、図２にあるような「03 48 66 68 66」という形で保存される。

なお、今回のプログラムでは、漢字を使用した場合、前半１バイトの判定が複雑になる

ので、判定を簡単にするために、漢字のない仮名のみのテキストデータを扱うことにした。

３．実験方法

前項で説明したランレングス法・ハフマン法・特定条件の圧縮法の３つを用いて、それぞれの圧縮効果を調べる実験を行った。

卒業研究の報告書から任意に切り出した500バイト～700バイト程度のテキスト4種と1000バイト程度のテキストの、計５種類のテキスト(漢字仮名混じり文なので「漢字+仮名」と記す）を用意した。ただし、漢字がある文では前述のように特定条件の圧縮法のプログラムが使えないので、漢字が含まれる「漢字+仮名」テキストの漢字の部分を仮名に変換したもの（「仮名のみ」と記す）も用意した。これにより、「漢字+仮名」ではランレングス法とハフマン法の比較ができ、「仮名のみ」では3 つの圧縮法の比較ができることになる。また、これらとは別に特殊な条件のデータとして、テキストの内容が全て「あ」である500 バイトのテキストも用意した。これらを使って、各アルゴリズムの圧縮効果を調査した。

なお、これらのテキストデータは Windows 上では Shift-JIS コードなのだが、今回のプログラム上内部的には EUC に変換して処理している。

４．実験結果

前述のテキスト５種類と、全て「あ」であるテキストを３つの圧縮法で圧縮した結果は次の表１・表２のようなものになった。結果のうち平均圧縮率に関してグラフ化したものが図３である。

表１と２、図３に示すように、ランレングス法では３倍以上のデータ増加となっている。

また、ランレングス法とハフマン法は「漢字+仮名」の圧縮率に対して「仮名のみ」の圧縮率の方が高く、「仮名のみ」に比べ「あ」のみの方がさらに圧縮率が高かった。

さらに、ここでの注目点はハフマン法と特

表１：各圧縮法によるテキストデータの圧縮結果

テキスト	もとデータ サイズ (バイト)	ランレングス法 サイズ (バイト)	ランレングス法 圧縮率 (%)	ハフマン法 サイズ (バイト)	ハフマン法 圧縮率 (%)	特定条件 サイズ (バイト)	特定条件 圧縮率 (%)
1 (漢字+仮名)	1070	3790	354.2	924	86.4	—	—
(仮名のみ)	1340	4450	332.1	906	67.6	530	39.6
2 (漢字+仮名)	538	1960	364.3	512	95.2	—	—
(仮名のみ)	664	2230	335.8	521	78.5	427	64.3
3 (漢字+仮名)	692	2400	346.8	617	89.2	—	—
(仮名のみ)	856	2850	332.9	603	70.4	446	52.1
4 (漢字+仮名)	632	2220	351.3	572	90.5	—	—
(仮名のみ)	792	2560	323.2	551	69.6	406	51.3
5 (漢字+仮名)	681	2290	336.3	538	79.0	—	—
(仮名のみ)	837	2580	308.2	540	64.5	450	53.8
漢字+仮名	平均圧縮率		350.6±0.1		88.0±0.1		—
仮名のみ	平均圧縮率		326.5±0.1		70.1±0.1		52.2±0.1

表２：テキスト内容が「あ」のみの時の圧縮結果

テキスト	もとデータ サイズ (バイト)	ランレングス法 サイズ (バイト)	ランレングス法 圧縮率 (%)	ハフマン法 サイズ (バイト)	ハフマン法 圧縮率 (%)	特定条件 サイズ (バイト)	特定条件 圧縮率 (%)
全文字「あ」	500	1460	292	68	13.6	252	50.4

定条件の圧縮法の圧縮率である。まず、ハフマン法だが「漢字＋仮名」、「仮名のみ」に比べ「あ」のみの場合の圧縮率がかなり高かった。次に、特定条件の圧縮法であるが、この方法では、「仮名のみ」も「あ」のみも、どちらも圧縮率は約 50%程度となった。この方法では、漢字が含まれていなければどのようなテキストでも平均 50%程度の圧縮率が実現できると考えられる。

５．考察

調査の結果から、３つの圧縮法の効果についてそれぞれ考察する。

（1）ランレングス法

今回のプログラムでは、たとえば、ある一文字（２バイト）のデータが２進数で「0010 0000 0000 0111」と表現される場合、保存されるのは「０２１Ａ3」となり５バイトになってしまう（図１参照）。この原因は、保存の仕方によるものである。今回のファイル操作では、例えば「0」が 4 回連続した時、保存するのに 3 ビットで足りる所を、8 ビットで書きこんでいるため、必要以上のデータサイズを要することになる。これが、圧縮率が 300%以上になった原因である。

また、図３から「漢字＋仮名」、「仮名のみ」、

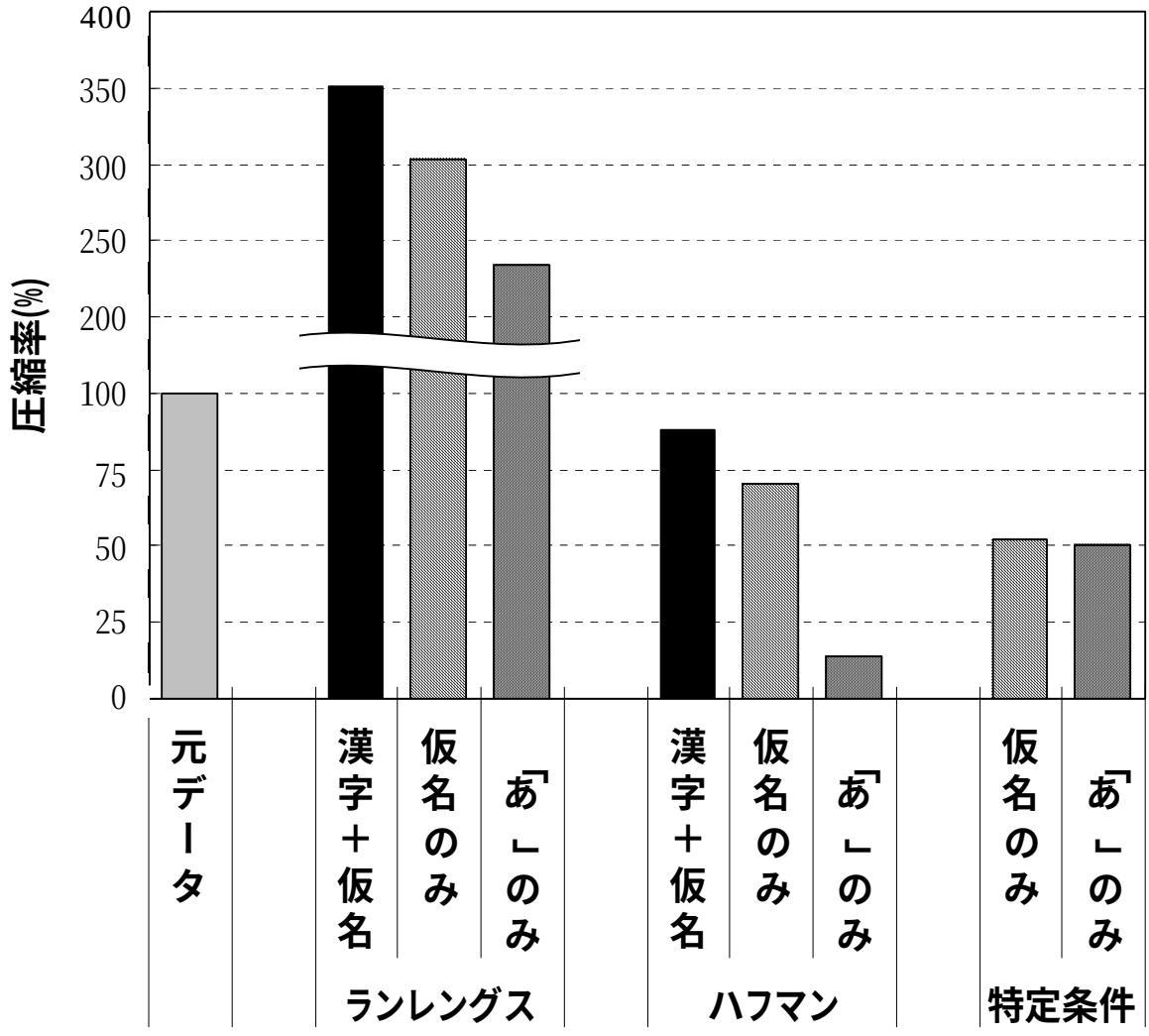

図３　各圧縮法によるテキストデータの圧縮率

「あ」のみの順に圧縮率が高くなっている。これは、漢字に比べ仮名の方が一文字当たりの「0」と「1」の連続する回数が多いことが原因だと思われる。さらに「あ」の連続は、他の仮名と比較して「0」の連続が多く（図４参照）、圧縮率が高くなったと思われる。

今回のプログラム上、テキストデータに０と１のランレングス法をそのまま適用すると、データが増加する性質を持っている。このような原因があるため、逆にデータ量を増やしてしまい、結果として処理後データが増加してしまった。ランレングス法は、元々画像などに適用される方法で、FAXなどのモノクロ画像に有効である[2)]ため、テキストデータには有効な方法ではないとわかった。

（２）ハフマン法

ハフマン法は、文字の出現頻度によって圧縮率が左右される圧縮法である。このことから、ファイルの中身の文字の種類が多ければ圧縮率は小さくなり、逆では圧縮率は高くなる。

今回の結果における「漢字＋仮名」と「仮名のみ」の圧縮率の差は、漢字を仮名に変換したことで、出現する文字の種類の数が少なくなり（出現頻度は高くなる）圧縮率が高くなったためと考えられる。また「あ」のみでは、他の２つに比べ圧縮率が極めて高い。これは、文字の種類が「あ」のみであるため出現頻度は最高になり、圧縮率が高くなったためと考えられる。

以上から、通常のテキストを圧縮する場合にハフマン法は有効ということが分かった。

（３）特定条件の圧縮法

五十音全ての文字コードは、前半１バイト（8 ビット）が共通している（図４と図５に50音の平仮名とカタカナの一部のEUCコードを示す）。また、特殊文字（句読点や？！など）は五十音とは別のコードで前半８ビットが共通している。変化しているのは、後半８ビットのみである。

図４：平仮名のコード表（EUC）

図５：カタカナのコード表（EUC）

共通する前半８ビットに対しランレングスのような考え方を利用することで、前半８ビットの数値と、それが連続した回数を保存する。このようにすれば、前半８ビットが共通する文字がｎ文字続けば、8＊(n−1) bit のデータが減るはずである。

しかし、後半８ビットはｎ文字分保存しなければならない。理論として２バイト文字の前半８ビットは共通する最初の文字の前半８ビットを除き、保存する必要はなくなるので、

圧縮率は平均 50%程度におさまることが考えられる。調査の結果からも同程度の数値に収まっているため、この考えは正しいといえるだろう。

６．今後の課題

今回、Javaを使って圧縮・解凍ソフトを実現し[3)]、評価した。Javaの利点は、Web上でアプレットなどで利用できることと、OSに依存せずWindowsでも、UNIXでも、利用できる 2 点が上げられる。

よって、さらにネットワークプログラムを研究していくことで、ネットワークを介した圧縮・解凍ソフトの利用が可能になると思われる。今後、ネットワーク上で使用できるようにしていきたい。

謝辞

今回の開発にあたり、研究室の皆様に多大な迷惑と協力をしてくれたことをこの場を借りて謝辞を表します。

参考文献

1) 奥村晴彦他：「Ｊａｖａによるアルゴリズム辞典」, 技術評論社, p.252-254, p.408-412, (2004)
2) プラスワンデジタル：「ファイルの圧縮・解凍辞典」, 翔泳社（2002）
3) Java2Platform,StandardEdition,1.4.0 API 仕様：http://java.sun.com/j2se/1.4/ja/docs/ja/api/