プログラムのページ

78-04 パーソナルコンピュータにおける文字データの圧縮

堰　田　愛　子*

はじめに

小規模の図書室や研究者の文献整理など，パーソナルコンピュータ程度のものを手近に置いて，文字データを含む情報の処理を簡便に行いたい場合が日常よく生じているが，このようなときデータの量が増えてくるとメモリの容量と，これに比べて文字データのコードの冗長性が大変問題になる．

ここでは，図書名（洋書）などよりなる文字データの圧縮について考察し，実例によりいくつかの点が明らかになったので報告する．

このような文字データを普通の英文と考えることとすると，スペースを含めて各文字の出現率には大きなかたよりがあり，数個の文字で全体の約5割をしめている（**表-1** 参照）．

また，普通の図書名には文字の長さに大きな違いがあり，身近の例（大学の図書）でも 10 文字台から，まれに 200 文字台位まであるが，図書館関係の文献[3]によると最近のコンピュータ処理では，図書の判別上の要と処理速度の点から，これを 50 から 60 文字程度の固定長のレコードとして処理する場合が多いようである．

ここでは，入出力及び演算に文字データと 10 進数値のみがソフトで扱えるパーソナルコンピュータで，使用できる言語はインタープリタ形式の BASIC のみであるものについて考察した．

メインメモリは 8k ワード（1 ワード 16 ビット），周辺機器として 1 機 32k ワード程度の容量をもつカセットテープが複数個使用できるようなものである．全体的な処理は会話型であり，この場合に圧縮の処理速度はコンピュータと人との対話に自然な応答が妨げられない程度を一応の限界と考えた．

* 東京工業大学理学部情報科学科

1. 圧縮方法

文字要素にはアルファベット 26 文字とブランク，他に少しの特殊記号を含めて 29 個程度を含むとした．バイナリなコードは扱えないため，又ビット使用に無駄を生じないために，1 ワードの整数型変数 1 個を単位にしてこれに 8 進数 5 個を記憶させる方法で圧縮することを考え，そのためにコードとしては Huffman[4] によるコンパクトコードの 8 元のを用いることとした．これにより出現率の高い方から上位 6 個までの文字は 1 文字当り 3 ビットの使用ですむこととなる．

文字データのエントロピーは 4.03 ビットであり，Huffman 8 元コードでは平均符号長が 4.38 ビットとなる．特殊文字の出現率は分っていないのでこれを最後につけ加えることとしたため，実際の平均符号長は

表-1 文字の出現率[1),2)]と 8 元コード

順　位	コード	文　字	出現率
1	0	ブランク	0.1859
2	1	E	0.1031
3	2	T	0.0796
4	3	A	0.0642
5	4	O	0.0632
6	5	I	0.0575
7	60	N	0.0574
8	61	S	0.0514
9	62	R	0.0484
10	63	H	0.0467
11	64	L	0.0321
12	65	D	0.0317
13	66	U	0.0228
14	67	C	0.0218
15	70	F	0.0208
16	71	M	0.0198
17	72	W	0.0175
18	73	Y	0.0164
19	74	G	0.0152
20	75	P	0.0152
21	76	B	0.0127
22	770	V	0.0083
23	771	K	0.0049
24	772	X	0.0013
25	773	J	0.0008
26	774	Q	0.0008
27	775	Z	0.0005
28	776	.	
29	777	,	

注：参考文献の 1) と 2) では出現率が少し違っているところがあった．

もう少し長くなるものと思われる.

2. プログラム法について

符号化を行う場合普通に行われる方法は原データに対応する符号をテーブル（配列）を用いて，そこから引き出すことであるが，今回の実例によると，配列を用いるプログラムは予想以上に時間がかかることが分った．その例として，実験のはじめに行った処理時間のかかる圧縮のプログラムを一応 **図-1** に示す.

これは，カセットテープから処理データとしての文字列と符号化のためのテーブルを読み出し圧縮を行うものである．実験のときの時間測定は圧縮に要する時間のみを計ったが，このプログラムでの処理時間は約 20 秒（57 文字当り）を要した．プログラムの長さでは，後のものにくらべはるかに短く 200 ワード位となっている.

その後，処理時間の短縮のためにプログラムの変更を行って，最終的には符号テーブル（配列）は用いず（インデックス操作がなくなる），全くプログラム的に符号を与える方法をとった．これによりプログラムは

```
10 COM A$[60],BI[18],KI
20 DIM L$[29],MI,XI,EI,LI,OI
30 M=0
40 L=1
50 I=0
60 LOAD  DATA 3
80 L$=A$
85 DISP "FILE NO.";
86 INPUT O
90 LOAD  DATA O
100 GOSUB 400
120 M=M+1
130 GOTO X OF 140,144,147,150,160,180,220
135 A$[M]=" "
136 GOTO 100
140 A$[M]="E"
141 GOTO 100
144 A$[M]="T"
145 GOTO 100
147 A$[M]="A"
148 GOTO 100
150 A$[M]="O"
152 GOTO 100
160 A$[M]="I"
170 GOTO 100
180 GOSUB 400
190 X=X+1
200 A$[M]=L$[X,X]
210 GOTO 100
220 GOSUB 400
240 GOTO X OF 250,250,250,250,250,250,280
250 X=X+9
270 GOTO 200
280 GOSUB 400
290 X=X+16
300 GOTO 200
400 REM
410 GOTO L OF 420,460,480,480,530
420 I=I+1
425 IF I>K THEN 600
428 E=4096
430 Y=BI[I]
440 X=INT(Y/E)
450 Y=Y-E*X
460 L=L+1
470 RETURN
480 E=E*0.125
490 X=INT(Y/E)
500 Y=Y-E*X
510 L=L+1
520 RETURN
530 X=Y
540 L=1
550 RETURN
600 PRINT A$
670 END
```

変数の説明
A$ 書名
L$ 文字テーブル
RI 符号テーブル
N 文字列の長さ
X 8元コードの1桁
BI 圧縮後のデータ
M データを入れる位置
注：$, Iはデータの型を示す.

図-1 符号テーブルを用いる圧縮プログラム

```
10 COM A$[60],BI[18],KI
20 DIM II,MI,OI,LI,XI,YI,NI,JI
25 DISP "FILE NO.";
26 INPUT O
40 LOAD  DATA O
91 N=LEN(A$)
100 M=0
110 L=1
130 FOR I=1 TO N
131 D$=A$[I,I]
140 IF D$=" " THEN 400
141 IF D$="E" THEN 395
150 IF D$="T" THEN 390
160 IF D$="A" THEN 385
170 IF D$="O" THEN 380
175 IF D$="I" THEN 375
180 X=6
181 IF D$="N" THEN 398
182 IF D$="S" THEN 392
183 IF D$="R" THEN 388
185 IF D$="H" THEN 382
187 IF D$="L" THEN 378
189 IF D$="D" THEN 374
200 IF D$="U" THEN 365
201 IF D$="C" THEN 350
202 X=7
203 GOSUB 496
204 IF D$="F" THEN 400
205 IF D$="M" THEN 395
206 IF D$="W" THEN 390
208 IF D$="Y" THEN 385
210 IF D$="G" THEN 380
211 IF D$="P" THEN 375
220 IF D$="B" THEN 370
230 GOSUB 496
240 IF D$="V" THEN 400
250 IF D$="K" THEN 395
260 IF D$="X" THEN 390
270 IF D$="J" THEN 385
280 IF D$="Q" THEN 380
290 IF D$="Z" THEN 375
300 IF D$="." THEN 370
310 GOTO 460
350 GOSUB 496
355 X=7
360 GOTO 460
365 GOSUB 496
370 X=6
371 GOTO 460
374 GOSUB 496
375 X=5
376 GOTO 460
378 GOSUB 496
380 X=4
381 GOTO 460
382 GOSUB 496
385 X=3
386 GOTO 460
388 GOSUB 496
390 X=2
391 GOTO 460
392 GOSUB 496
395 X=1
396 GOTO 460
398 GOSUB 496
400 X=0
460 GOSUB 496
480 NEXT I
482 IF L=1 THEN 490
484 E=5-L+1
486 Y=Y*8↑E
487 M=M+1
488 B[M]=Y
490 K=M
492 STORE  DATA O
494 GOTO 670
496 REM
498 GOTO L OF 510,530,530,530,550
510 Y=X
511 L=L+1
520 RETURN
530 Y=Y*8+X
535 L=L+1
540 RETURN
550 M=M+1
560 B[M]=Y*8+X
570 L=1
580 RETURN
670 END
```

変数 A$, BI, N, X, M は図-1 と同じ
O ファイル NO.
Y 8進数貯える場所
K 圧縮後のデータ数

図-2 圧縮プログラム

```
10 COM A$[100]
20 DIM BI[70],L$[31],RI[43]
40 LOAD  DATA 1
91 N=LEN(A$)
92 LOAD  DATA 0,R
93 TRANSFER R[32] TO L$
100 M=0
110 L=0
130 FOR I=1 TO N
140 FOR J=1 TO 31
150 IF A$[I,I]=L$[J,J] THEN 170
160 NEXT J
170 IF J>6 THEN 210
180 X=R[J]
200 GOTO 460
210 I1=INT(R[J]/10)
220 R1=R[J]-I1*10
230 IF J<21 THEN 400
240 X=2
250 GOSUB 496
260 IF J<28 THEN 300
270 X=4
290 GOTO 410
300 X=3
320 GOTO 410
400 X=I1
410 GOSUB 496
450 X=R1
460 GOSUB 496
480 NEXT I
490 GOTO 670
496 L=L+1
498 GOTO L OF 500,530,530,530,550
500 M=M+1
510 BI[M]=X
520 RETURN
530 BI[M]=BI[M]*8+X
540 RETURN
550 BI[M]=BI[M]*8+X
570 L=0
580 RETURN
670 END
```

変数 A$, BI, K, L$, Y などは図-1, 2と同様
M 解読後の文字列の中の位置
X 文字テーブル内の位置

図-3 解読プログラム

長くなり約500ワード位になったが，処理速度は9秒(57文字当り)と半分位に短縮される結果となった．ここで用いた命令文は一致不一致の比較，無条件分岐など実行時間の速いものが多い．最終的に実行したプログラムを**図-2**（前頁参照）に示す．

ここでは，入力文字列の頭から順に出現率の高い文字との比較を行って行き，一致したところでそのコードを与えている．

又，圧縮されたデータから普通の文字に解読する逆変換のプログラムを**図-3** に示す．

3. 実　例

最近の2，3年間に身近で購入された電子計算機関係，オペレーションズ・リサーチ関係の図書について書名の文字数を調べた例では**表-2** のようであった．

専門の図書館の例にならって，入力データの最長を60文字までとし（以下切捨て）て扱うものとして，上記の図書より適宜に選んでカセットテープへの書名の

表-2 洋書名の文字数と出現頻度

文字数	29まで	30～39	40～49	50～59	60以上	計
頻　度	41	45	57	19	12	174
パーセント	23.6	25.9	32.7	10.9	6.8	100

文字数の最小：13，平均：38.5，最大：81

表-3 文字データの所要メモリ量と対話型処理における実行時間

(1)	(2)	(3)	(4)	(5)	(6)	(7)
文字数	これまでの方法の所要メモリ（ワード）	実際の使用メモリ（ワード）	圧縮後の使用メモリ（ワード）	人間のタイプイン時間（秒）	圧縮時間（秒）	解読時間（秒）
13	31	8	5	19	3	2.8
23	31	13	7	27.7	4	4.0
38	31	20	12	48	6.4	6.4
40	31	21	12	42.5	6.4	6.6
49	31	26	16	44	8.5	8.8
58	31	30	17	56	9.5	9.4
(計) 221	186	118	69			

(4)/(3)=58%　　(4)/(2)=37%

(注) 計算機のカセットテープへの入出力時間は作動開始から終了まで，50ワードで書込みに3秒，読出しに約1秒を要す．

書込み，読出しなどを実行した．この場合の人間とコンピュータとの対話の様子を，実行時間及び所要メモリで記してみると**表-3** のごとくになる．文字のコード化，圧縮と，逆にコードの解読に要する時間とはほぼ同じであった．

図書室等における業務で書名の使われ方をみると，はじめに一度これを記憶してしまえば，その後最も利頻用度が多いのは検索の時であるが，この時には文字列のはじめの15文字程度をキーとして用いるので，必要な解読の時間は，カセットテープへのレコードの書込みや読出しとほぼ同程度となることがいえる．他に全書名を書出しすることは頻度からいうとあまり多くない．この例で，各文字数13,……58のウエイトを等しいとして，メモリの使用量は圧縮後58%に縮少されていることが分る．従来の記憶方法に比し，圧縮して固定長レコードとして用いる場合で約6割，可変長で用いるなら4割の容量で済むことが分る．

おわりに

以上でパーソナルコンピュータでの英文としての文字データの圧縮について述べてきたが，処理速度，圧縮率など上記のように判明した．今の機種でこの方法が用いられたとして，応答に大きな不自然さはないと思われるが，データの量や使用可能メモリなど情況によっては大変有用になると考えられる．更にアセンブラの使える機種や，LSIなどでこれを組込める機種であるなら処理時間もはるかに減少し，メモリの節約に伴って，周辺機器の取扱いなど人手の介入による手間と時間の減少が考えられるので一層有効になろう．

プログラムの方法によっては処理時間に大きな差異が出てくることが判明したので，使用する命令とプログラムの組立て方にはよく配慮する必要がある．

参 考 文 献

1) アブラムソン：宮川訳：情報理論入門，p. 229，好学社，東京 (1975).
2) 奥野治他監修：情報処理ハンドブック，光琳書院，東京 (1965).
3) 井上，関：東京工業大学における図書館業務の電算化，大学図書館研究 VI, pp. 8～21 (1975. 1).
4) D. A. Huffman: A Method for the Construction of Minimum Redundancy Codes, Proc. I. R. E., Vol. 40, pp. 1098～1101 (1952).

(昭和52年9月29日受付)
(昭和53年3月14日再受付)