Upgrading bred with multiple tables

David Wheeler

Jan 1997
revised April 97
printed May 16, 1997

Introduction

The Bred routine set have been changed and improved. The input filter to remove multiples has been removed and a different sorting algorithm used which deals well with simple repeats. To ensure workability in all cases, when the sorting algorithm becomes slow the block size is repeatedly halved, so that the block is still compressed, but at a lower efficiency, due to the smaller block size.

A set of tables can be used (up to eight in the current implementation) which gives adaption to changing statistics, and reduces the inefficiencies of Huffman coding.

Currently, the -m option followed by a decimal number specifies the number of groups 1-8 the size of the groups 25, 50, 100 or 200 and the maximum number of iterations allowed. The default is one group, for which no iterations take place.

The routine improves coding efficiency by scanning the data to be coded after the move to front and 1/2 transforms, and choosing the best table of the set for the next group of symbols. The selection is coded and mixed with the rest of the coded data. The extra cost of the tables and selection bits is more than compensated for by the improved coding efficiency, although small files default to a single table.

The method could be readily improved by additional searching, and the decoding not slowed any further. However, the increased small gain would be bought at a cost of cosiderable extra time.

The bulk space needed by bred3 is five times the -K parameter+256.000 bytes. For computers whose long is 8 bytes the primary space is nine times the -K parameter+512,000 bytes unless a macro is used to change long into int at the noted point in the program. The block size is reduced under certain rare conditions.

The bulk space used by bexp3 is four (or eight where long is eight bytes) times the -K parameter.

Using the routines

The routines are called by the routine name, a list of parameters and a list of files to be compressed or expanded.

The - parameters are as follows, where n is a decimal number immediately after the parameter, if null then n=0.

- Parameter	Effects
Kn	use a block size of 1000n space $(1000n + 65,000) sizeof long + n$ bytes default 200,000

kn	keep the original file if n=0 otherwise delete if no errors default n=1 i.e. delete
sn	n=0 no printing if n an odd multiple of 1 : print compression data and files, or files expanded 2 : print extra information prints when block size has to be reduced due to slow sorting prints when block size reduced to allow a long prefix for zero strings in ecess of 1000. prints millibits for,tables, selectors,number of iterations after first sweep, number of groups used, and size of groups. 4 : self followed by final selection frequencies. and GAIN followed by iteration gain in millibits, followed by the number of bits used for the character coding last time and this time. 8 : prints all the selections after the last iteration. Advisable only for small files default n=1
mn	bred only n=d1 d2 d3, three decimals. d1 0-7 gives extra tables used d2 0-3 gives size of groups, 25, 50, 100 or 200 d3 limits the number of repeat scans 0-9 if no convergence default n=0, i.e. single table.
pn	If the cipher option has been compiled, uses -pm as the key where m is any set of characters that is not special for parameters for example *,?,[,],' ', or newline. If m is absent no ciphering is done.
wn	bred only n is the work limit for the sorting routine. It measures the work done in in sort comparisons excess of DEPTH characters deep. The DEPTH macro may be adjusted before compilation default 40,000,000
en	n is the error limit. When errors are detected, the next file may be tried, until the limit is reached. default n=1

The file name is extended by . when compressed. The file to be expanded must end its name with a .

The use of extra tables slows the compression phase, perhaps 2 tables by about 1/3, 6 tables about 2/3. Different computers would give different figures. See Appendix.

The gain by raising the iteration limit much above 3 is quite small.

The speed of decode is almost unaffected by the use of extra tables.

The extra delay in ciphering is quite small.

Error messages

The actions taken are:
close source file
close destination

unlink destination file

all error messages have the format
bred3/bexp3 fails, file "name"
"error message"

Error Message	Probable Cause
"cannot open to write"	the destination file cannot be opened.
"cannot open to read"	the source file cannot be opened.
"not enough space from malloc"	there is not enough space, change option K if possible.
"end of file too early"	the source file encountered end of file too early.
"options wrong"	an unimplemented option else a wrong one. no more files processed.
"no . terminator"	the file to be expanded did not have an end . in its name.
"length start calculations wrong"	Use a smaller number of groups else just one. the computation of the initial lengths for searching is wrong.
"wrong first character"	the first character of the compressed file is wrong.
"wrong first character for cipher"	ciphered block has wrong first character
"Early end of file"	end of file in 8 character salt
"wrong usage table"	character usage table wrong-possibly wrong password
"decode tables wrong"	the tables used for decoding have not been read correctly. Possibly wrong password
"blocksize too small"	the decoded string overran the blocksize.
"not exactly one end of block"	either none or more than one end of file code has been received.

Macro options

There are number of options available for the compilation of the routines.

The macro DEPTH defines how deep sorting goes before adding to work. The default is 32. If the macro CIPHER is defined the cyphering options will exist. If CIPHER is not defined then the cyphering options will be omitted. The default is for it to be set. If the macro is not set the routines are shortened the by a few %.1-7%. The speed is increased by ——————

The macros GPNO and GPSZ may be changed for both bred and bexp. The working values may be 1-8 or the smallest size, with the m option selecting multiples 1, 2, 4, 8 of that size.

The default is 8 and 25. The program would have to be changed internally if GPNO is larger than 8.

Macros FAST and FASTMASK may be changed, and have values N 1 to 8 with the mask having corresponding values 2^N-1. They improve the decoding speed by using two tables of length 2^N to give a direct decode rather than a search. The default value is 6 using two tables of length 64 per decoding group.

Method outline

The symbols are read in, sorted, transformed by move to front and the 1,2 code, and stored.
A prefix character is sent giving the number of groups,and size of groups, and a check.
A table is made of which characters are present.
Using the next character sequence and move to front encoding, set up the data to be output and form the first frequency table.

For multiple tables
An initial set of code length tables is set up
The data is scanned in groups to select the best length table for each group and new frequency tables are formed from the selections.
The new tables are then used to generate new code length tables .
When the given number of iterations is reached, or the improvement is less than one millibit is gained, the search is stopped.
The selections are scanned, coded relatively (move to front), and 1/2 coding used for repeats to reduce the Huffman inefficiency for probabilities greater than 1/2 and utilise the correlations of successive selections.
The selection coding table is sent.
The length tables are used to form the coding tables and also sent to the decoder.

Then the actual values are coded and sent using the tables generated.
The selection codes are also sent once every group size, or less when the 1/2 coding skips groups.

The decoding is simpler.
Read all data for tables and make them
Read data, decode remove 1/2 coding, remove move to front, make a list each element containing a character and the number of those characters already encountered.
Sum the frequency table.
Chain using character and character pointer sending original text.

Sorting method

The sorting method has been changed and now uses one long plus one byte per character, together with a table of 65000 longs. It uses a quicksort type of method, uses a next character method to reduce the work by about one half. A preliminary two character sort is used. This may be a little slower for small files but is better for large files. There is a work limit so that files which are very slow to sort, are repeatedly halved in size until success is achieved. This reduces the compression a little. Files which have large parts replicated are likely to cause many deep comparisons for the sorting routine. Such files are not usually encountered. An improved sorting routine to cope, would be slower on average and need more space.

Data layout of Output

The compressed data is as follows.

The first character has value 224+ 4 (gpno-1) + gpszz
where gpno+1 is the number of main decode tables
gpszz is the group size indicator.
0 : 25, 1 : 50 2 : 100 3 : 200

Next there is a table giving the characters used. A dual 1/2 coding is used for nulls and characters.
00=1 null, 01=2 null, 10=1 character, 11=2 character,

This is usually much shorter than a marker bit encoding.

The coding has two main effects.
1 : The remaining tables are much shorter for ascii.
2 : The table cost is partly offset by the gaps in the alphabet which otherwise cause the end few characters to have a higher defining and coding cost.

Next there is a table defining for the main table selectors. This has gpno+1 entries of six bits each giving the length and code to be used for the selectors.

Next there are the lengths of the codes of the main tables. These enable the complete decoding tables to be generated. The lengths are encoded in a simple 2 bit differential code, which takes good advantage of the roughly monotonic nature of the codes after move to front.

Finally, there is the coding of the symbols interspersed with the table selection codes at intervals of multiples of gpsz.

Initial group selection

This starts after the sorting, using predicted character, doing the move to front and listing the result with a 1/2 coding for strings of repeats. The starting length selection tries to equalise sums of the frequencies of consecutive entries which make the groups. Due to the high frequency early codes, the group frequencies are not equal but their effective frequencies are made more equal by adjusting the starting code lengths.

The number of groups is reduced if the length of a block or subblock is too small to about 1+size/8K.

Ciphering

The compressed block can be encyphered and decyphered using the -p option. This will use time 0f day to generate a “salt” which is prefixed to the compressed file, after the “magic” first character. This ensures that no two files are likely to be encoded with the same effective password. This prevents some attacks.

The usual rules apply and suitable passwords should be chosen. The ciphering and deciphering take little extra time.

Decoding tables sent

There is a preliminary table and “magic character”. This gives the number of tables used and the size of groups. This is followed by bits which define the characters used.

This has two effects. First, when the table is not full, it reduces the number of bits to define the tables and also decreases the length of the code for the first use of a character. Both of these effects are beneficial.

Data formats

Performance data

The average gain using 2 or 6 tables and group size of 50 is given below.

```
 2 tables 1/3 extra time   1/30 extra compression
 6 tables 2/3 extra time   1/24 extra compression
```

There is still a loss due to Huffman coding, but the use of multiple tables reduces the Huffman loss and is essentially adaptive. It is difficult to make precise statements and it is true that in all cases the best adaptive is as good as the best using non adaptive tables.

However, the " look-a-head " selection can react on two levels. First there is the table selection, and then there is the use of a table for the selection elements. This is quite adaptive.

The loss induced (for the Calgary corpus) by using the previous context rather than the next context is reduced by the use of multiple tables.

Appendix

Typical sequences of selectors.

```
            start of bookI  group size 25
 7 7 3 4 0 7 0 4 4 3 2 0 5 7 5 5 5 7 5 5 0 0 5 0 5
 5 5 5 5 0 0 2 5 5 0 4 5 5 4 3 2 5 5 0 4 5 3 0 5 5
 0 3 4 5 0 0 1 2 0 2 3 2 0 5 7 7 0 7 7 5 5 7 5 5 6
             middle of book
 3 4 3 3 2 4 2 4 4 4 4 0 4 4 4 3 3 3 3 0 1 3 3 4 4
 2 3 4 4 3 2 0 0 4 4 3 4 2 1 3 3 3 4 4 4 3 0 4 4 3
 3 3 4 4 3 0 4 4 4 3 4 4 0 0 0 0 0 2 1 0 5 5 0 1 0
               end of book
 4 4 4 4 2 4 3 4 4 4 4 4 0 0 0 0 0 5 0 1 1 1 1 1 1
 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 5 0 4 5 4
      With block size 200   start
 0 2 2 2 2 2 2 4 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1
 1 0 0 0 0 0 1 1 1 0 0 0 1 0 1 0 1 1 1 1 1 1 0 1 0
 0 1 0 0 0 1 1 0 0 0 0 1 1 0 1 0 0 1 1 1 1 1 0 1 1
                   middle
 4 5 5 3 3 3 3 5 4 4 3 4 3 3 3 7 7 7 7 4 7 4 7 7 5
 4 4 3 7 7 4 4 7 7 6 7 6 7 7 7 7 7 5 6 4 6 5 5 6 5
 6 6 6 7 7 6 7 7 4 7 7 7 7 7 7 7 4 7 6 4 7 2 2 4 7
                     end
 5 5 5 4 5 4 5 4 4 7 7 7 7 7 7 3 4 7 4 4 7 7 5 6 4
 3 4 7 6 6 6 6 6 7 4 7 4 4 4 6 7 4 7 6 7 7 5 6 4 5
 5 6 5 5 6 5 4 4 5 4 6 4 7 7 5 4 7 7
```

Code length tables for bookI groupsize 200 number 8

```
               tables 1-4      first twenty of each
 3  5  3  4  4  4  4  5  5  5 5  5  5  5  5  6  6  6  6  6
 4  6  4  4  4  4  4  4  4  4 4  5  5  5  5  5  5  5  5  6
 3  4  3  3  3  4  4  4  4  5 5  5  6  6  6  7  7  7  8  8
 3  4  2  3  3  3  4  4  5  6 8  8  9 10 11 12 13 13 13 12
              tables 1-4    last 15 of each
14 15 16 16 16 14 15 15 14 14 15 16 16 16 16
 0 16 16  0 16 16 15  0  0  0  0  0  0  0 16
15  0  0  0 15 15 15  0 15  0  0  0 15 15 15
 0  0  0  0  0  0  0  0  0  0  0  0  0 14 14
```

Some results for Calgary corpus

```
               in       millibits
                       1    2    6  tables
average       224401  2462 2380 2359
 times user and system password nopassword for 1, 2 or 6 tables
 6.3 0.2   6.3 0.3   8.4 0.2  8.4 0.2  10.4 0.3  11.4 0.2
Note almost no time lost due to passwords.
 2 tables 1/3 extra time   1/30 extra compression
```

6 tables 2/3 extra time 1/24 extra compression

performance check on files and reversed files
file, reversed file 1,2 or 6 tables

millibits per character

tables	1	1	2	2	6	6
files	dir	rev	dir	rev	dir	rev
bib	2061	2031	2020	1990	2002	1964
bookI	2601	2572	2460	2439	2428	2418
bookII	2190	2172	2094	2088	2071	2058
geo	4872	4873	4470	4569	4332	4472
news	2621	2601	2555	2540	2529	2512
objI	3910	3940	3878	3928	3886	3925
objII	2575	2516	2527	2486	2514	2475
paperI	2574	2543	2503	2487	2501	2473
paperII	2573	2526	2472	2441	2461	2433
pic	843	832	810	802	809	800
progc	2562	2555	2522	2518	2519	2515
progl	1778	1776	1736	1738	1728	1735
progp	1754	1750	1723	1717	1727	1718
trans	1550	1530	1538	1520	1521	1507
avs	2461	2444	2379	2375	2359	2357
gain		17		4		2

using up to six table with group size 50
and iterations (its) having a maximum of 4
Ciphering increased file out by 8

							its	group	
file	in	%	out	mbits	tab	sel		no	sz
bib	111261	74	27834	2001	12	22	4	6	50
bookI	768771	69	233747	2432	1	23	3	6	50
bookII	610856	74	158141	2071	2	21	3	6	50
geo	102400	45	55440	4331	32	16	4	6	50
news	377109	68	119224	2529	3	25	3	6	50
objI	21504	51	10452	3888	87	19	2	3	50
objII	246814	68	77575	2514	15	24	2	6	50
paperI	53161	68	16622	2501	27	28	3	6	50
paperII	82199	69	25304	2462	18	25	3	6	50
pic	513216	89	51963	809	5	7	2	6	50
progc	39611	68	12480	2520	31	25	3	5	50
progl	71646	78	15489	1729	20	20	2	6	50

progp	49379	78	10675	1729	31	22	2	6	50
trans	93695	80	17830	1522	17	19	3	6	50
mean	224401	69	59484	2359	21	21	2	5	50