Technical Correspondence

AGAIN: HUFFMAN CODING

I would like to clear up a couple of points about Huffman coding raised in the Technical Correspondence section of the February issue (*Commun. ACM* 29, 2(Feb. 1986), 149–152).

1. The dynamic Huffman coding algorithm distributed as the UNIX* BSD4.2 “compact” command, based on an article by Gallager [2], was independently discovered five years earlier by Faller [1].
2. It is misleading to say that this dynamic algorithm does not require the table to be transmitted. The table is implicitly transmitted as the characters are transmitted. There is an implicit cost associated with using a code which is initially not very well suited for transmitting the beginning of the message. Also, if the message statistics change, there will be an implicit cost associated with characters transmitted until the table statistics change to reflect the current statistics. Vitter [3] has an interesting discussion of this algorithm.
3. Byte-oriented Huffman coding is optimal only if the bytes of the message are statistically independent. Huffman coding is not obsolete if this assumption is satisfied, and word-oriented Huffman coding may still be of interest for text or source-code files.

Anselm Blumer
Department of Computer Science
Tufts University
Medford, MA 02155

REFERENCES

1. Faller, N. An adaptive system for data compression. *Record of the 7th Asilomar Conference on Circuits, Systems, and Computers* (1973), 593–597.
2. Gallager, R. G. Variations on a theme by Huffman. *IEEE Trans. Inform. Theory* IT-24, 6 (June 1978), 668–674.
3. Vitter, J. S. Design and analysis of dynamic Huffman coding. Tech. Rep. CS-85-13, Brown Univ., July 1985.

SYMBOLIC MATHEMATICAL COMPUTATION

As developers of a symbolic manipulation system, Maple, we had hoped that Wolfram’s article (*Commun. ACM* 28, 4(Apr. 1985), 390–394), would generally promote interest in computer algebra among our colleagues. However, we have perceived by com-

* UNIX is a trademark of Bell Laboratories.

ments from other colleagues and from network news, that this article has misguided people and may have given a false impression of work in the symbolic computation area.

Near the beginning of the article, the author asserts: “The most advanced and complete mathematics system at present available is probably SMP.” To test this claim we ran version 1.5.0 of SMP (the latest as of November 1985). The following examples of test runs cast serious doubt on this claim.

```
/* In the following examples, #I[...]
labels the input statements, while
#O[...] labels the lines printed by
the system in response to its input */
/* "B" is the function which specifies
that exact integers of arbitrary pre-
cision instead of floating point num-
bers will be used in the calculation */
#I[1]:: B[(x+1)^80]
#O[1]:* (x + (1))^80
#I[2]:: Ex[%]
/* The lengthy result from ex-
panding the above expression,
not reproduced here, contains
*negative* coefficients */
#I[1]:: p:B[(x+1)^100]
#O[1]:* (x + (1))^100
#I[2]:: Ex[p]
/* Appears to be in an infi-
nite loop (exceeded 2000 CPU
seconds) */
#I[1]:: Pgcd[(x+1)^2, (x+1)*(x-1), x]
#O[1]: (1 - x) (1 + x)
/* Wrong, the polynomial gcd
is obviously 1 + x */
#I[1]:: Fac[B[x^2-1]]
#O[1]: (-1 + x) (1 + x)
#I[2]:: Fac[B[x^2-y^2]]
#O[2]:* y^2 (-1) + x^2
/* SMP apparently thinks that
x^2-y^2 does not factor */
-----
/* A numerical Bessel function
computation */
```

```
#I[1]:: N[BesJ[0,2.7],40]
8878 Bus error
/* This is a Unix system error. It
means that SMP "crashed". */
/* The N function is used here to
compute a 40 digit floating point
approx. */
```

```
#I[1]:: x/0
#0[1]:  $\frac{x}{0}$ 

/* Although SMP allows such an expres-
sion to exist, this leads to bugs */
/* For example, taking the limit of
the above as x goes to 2 we get */
#I[2]:: Lim[% ,x,2]
#0[2]: 2
```

```
#I[1]:: 374^3*x - 374*374*374*x
#0[1]: -5.23136*^7x + 374^3 x
#I[2]:: B[%]
#0[2]: * x (-52313624) + x (52313624)
/* SMP did NOT do the coeffi-
cient arithmetic in a sum */
```

```
#I[1]:: d:x^2-1-(x+1)*(x-1)
#0[1]: -1 - (-1 + x) (1 + x) + x^2
/* In SMP, little attempt is made to
recognize zeroes. This leads to wrong
or meaningless answers being returned,
as well as system failures as the
following examples illustrate. Other
systems make use of a normal form for
polynomials and rational functions to
recognize zeroes. */
#I[2]:: Pgcd[d, x^2+x+1, x].
#0[2]: 1 + (1 - (1 - x) (1 + x))
(2 - (1 - x) (1 + x))
#I[3]:: Ex[%]
#0[3]: 1 + x^2 + x^4
/* Note that the degree of
both input polynomials is less
than 4 */
#I[4]:: Lim[1/d,x,0]
#0[4]: *  $\frac{1}{0 x^3}$ 
#I[5]:: Int[1/d,x]
SMP INTERNAL ERROR
Numerical overflow
```

```
#I[1]:: Coef[x,(x+y)*(x+z)]
#0[1]: 0
/* Mathematically, the coeffi-
cient in x is not 0. */
```

```
#I[1]:: Rand[10000000000]
#0[1]: 0.814888
#I[2]:: Rand[10000000000]
#0[2]: 0.309272
/* According to the manual,
this should produce random
numbers in the range
[0..10000000000) but in fact,
no numbers greater than 1 are
returned */
```

```
#I[1]:: Sol[{x+y=1, 2*x+2*y=2}, {x,y}]
#0[1]: {{{}}
/* There are, in fact, an in-
finite number of solutions to
this */
/* trivial system of linear
equations which we obtain by
doing */
#I[2]:: Sol[{x+y=1}, {x,y}]
#0[2]: {x -> 1 - y}
```

```
#I[1]:: Int[-1/x^2, {x,-1,1}]
#0[1]: 2
/* Wrong. The integrand is clearly
negative over the integration range */
/* SMP makes the error of assuming
continuity over the range of integra-
tion */
```

```
#I[1]:: Simtran[{{3,5,7},
{11,13,17},{19,23,29}}]
#0[1]:  $\begin{Bmatrix} 0 & 0 & 0 \\ \{-\}, \{-\}, \{-\} \\ 0 & 0 & 0 \end{Bmatrix}$ 
/* Simtran supposedly computes
the similarity transformation
matrix. */
/* The correct result has
three nonzero vector entries.
*/
```

Of course all systems have their share of shortcomings, bugs, and peculiarities. The results of all computer programs must be scrutinized and checked before being applied. Yet we feel that the *Communications* article makes unjustified claims about the power and reliability of SMP and does not fairly reflect the progress made in the field.

On the third page the author asserts: "[MACSYMA and REDUCE] are comparatively slow and able to handle comparatively small symbolic expressions, ...". We challenge this with examples that compare

TABLE I.

Problem	SMP 1.5.0		Maple 4.0		Reduce 3.1	Macsyma 308
1	15.6	5,012Kb	0.3	171Kb	0.7	0.15
1a	>60	(*)	0.6	195Kb	1.4	0.3
2	25.8	4,309Kb	1.1	203Kb	3.6	9.0
2a	>60	(*)	3.2	283Kb	9.3	22.5
3	28	3,501Kb	4.8	460Kb	5.4	6.6
3a	>71	(*)	11.4	717Kb	29.8	31.1
3b	>71	(*)	9.9	438Kb	7.2	4.5
3c	>71	(*)	20.0	606Kb	24.1	16.8
4	>119	(*)	1.9	218Kb	1.5	15.0
5	>75	(*)	2.1	285Kb	0.8	0.8
5a	>71	(*)	23.8	554Kb	8.1	1.7
5b	>300	(*)	1.6	221Kb	0.7	0.5
6	>71	(*)	0.7	201Kb	2.1	0.7
7	>90	(*)	2.2	228Kb	5.7	15.0

(*) SMP ran out of space (exceeded 7.2 megabytes of virtual storage)

TABLE II.

Problem	Description
1 Integer division	divide 3^{5000} by 3^{30}
1a	divide 3^{10000} by 3^{30}
2 Rational addition	$\text{sum}(1/k, k=1..200)$
2a	$\text{sum}(1/k, k=1..400)$
3 Polynomial multiplication	$s := \text{sum}(k \cdot x^k, k=1..64);$ $\text{expand}(s^2);$
3a	$s := \text{sum}(k \cdot x^k, k=1..128);$ $\text{expand}(s^2);$
3b Let $p := 1+x+y+z^2y+z.$	$\text{expand } p^5$ then expand that squared
3c	$\text{expand } p^6$ then expand that squared
4 Polynomial division	divide $\text{expand}(p^6)$ by $\text{expand}(p^3)$
5 Polynomial factorization	$\text{factor } \text{expand}((x^2+y^3)^3)$
5a	$\text{factor } x^6 - y^6$
5b	$\text{factor } x^4 - 3^{20}$
6 Polynomial gcd	Let $f=(107x+53)^7$ and $g=(109x+59)^7$ Compute the $\text{gcd}(\text{expand}(f), \text{expand}(g))$
7 Integrate	$x \cdot \exp(a \cdot x) \cdot \sin(b \cdot x)$ with respect to x

SMP with Maple, Reduce, and Macsyma. From these tests we concluded that SMP is weak at the fundamental level of arithmetic as can be deduced from Tables I and II.

Our sample problems were run on DEC VAX machines at Waterloo. SMP and Maple were tested on a VAX 11/780 with Berkeley UNIX 4.1. Macsyma and Reduce were tested on a VAX 11/785 with Berkeley

UNIX 4.2. To make a fair comparison, we reduced VAX 785 times by 33 percent to be equivalent to VAX 780. Memory requirements are given in kilobytes and include the system code and initial data space (1910 kilobytes for SMP, 163 kilobytes for Maple). Times are given in CPU seconds.

The problems in the test sample are almost trivial examples. Maple, Reduce, and Macsyma are capable of handling much larger problems without running out of space. We speculate that SMP's performance problems arise from two sources. First, some algorithms are clearly inefficient especially with respect to space. Second, SMP performed no garbage collection during these computations; the others did. SMP performed its garbage collection only after the computation had completed (or had run out of space). The apparent lack of garbage collection during the computation must severely limit the size of computations that can be completed successfully using SMP.

Michael Monagan

Gaston Gonnet

Bruce Char

Symbolic Computation Group

Department of Computer Science

University of Waterloo

Waterloo, Ontario, Canada

REFERENCE

1. Monagan, M.B., Gonnet, G.H., and Char, B.W. Measurements and comparisons of SMP. Tech. Rep. CS 85-47, Symbolic Computation Group, Dept. Computer Science, Univ. of Waterloo, Ontario, Canada, Nov. 1985.