(4) Like pair R,R or S,S precedes unlike pair R,S or S,R; M,M or P,P precedes M,P or P,M; R,M or S,P precedes R,P or S,M; M,R or P,S precedes M,S or P,R; also r precedes s.

(5) R precedes S; M precedes P. Note: The above notations have the following correspondence with our notation: R ≡ /R/; S ≡ /S/; r ≡ /R#; s ≡ /S#. Also M and P denote left-handed and right-handed helices, respectively.

LITERATURE CITED

(1) Cahn, R. S., and Ingold, C. K., "Specification of Configuration about Quadricovalent Asymmetric Atoms," *J. Chem. Soc.*, 612 (1951).

(2) Cahn, R. S., Ingold, C. K., and Prelog, V., "Specification of Configuration in Organic Chemistry," *Experientia*, **12**, 81 (1956).

(3) Cahn, R. S., Ingold, C. K., and Prelog, V., "Specification of Molecular Chirality," *Angew. Chem. Int. Ed. Engl.*, **5**, 385 (1966).

(4) Cahn, R. S., "An Introduction to the Sequence Rule," *J. Chem. Educ.*, **41**, 116 (1964).

(5) Krishnamurthy, E. V., Sankar, P. V., and Krishnan, S., "ALWIN—Algorithmic Wiswesser Notation System for Organic Compounds," *J. Chem. Doc.*, **14**, 130 (1974).

(6) IUPAC "Tentative Rules for the Nomenclature of Organic Chemistry, Section E, Fundamental Stereochemistry," *J. Org. Chem.*, **35**, 2849 (1970).

(7) Blackwood, J. E., Gladys, C. L., Petrarca, A. E., Powell, W. H., and Rush, J. E., "Unique and Unambiguous Specification of Stereoisomerism about Double Bond in Nomenclature and Other Notation Systems," *J. Chem., Doc.*, **8**, 30 (1968).

Huffman Binary Coding of WLN Symbols for File-Compression†

K. SUBRAMANIAN,‡ S. KRISHNAN, and E. V. KRISHNAMURTHY*

Molecular Biophysics Unit, Indian Institute of Science, Bangalore 560012, India

Received December 3, 1973

Construction of Huffman binary codes for WLN symbols is described for the compression of a WLN file. Here, a parenthesized representation of the tree structure is used for computer encoding.

In this paper, we describe the construction of Huffman binary codes[1] for WLN symbols, to facilitate file-compression in storing WLN for chemical compounds. This construction has been made feasible by the data given in ref 2 which gives the statistics of occurrence of WLN symbols.

It is well known that the Huffman procedure yields an optimum set of uniquely decipherable and instantaneously decodable code words (meaning thereby that no other set has a smaller number of symbols per message); hence the code constructed is of value in economizing the chemical information storage.

HUFFMAN CODE

Given a set of N source symbols $S = \{s_1, s_2, \ldots, s_N\}$ and their probability of occurrences $P = \{p_1, p_2, \ldots, p_N\}$, the Huffman procedure constructs a set of instantaneously decodable (optimum length) code words using the code alphabet set $C = \{c_1, c_2, \ldots, c_M\}$ by the following steps.

Step 1. The N symbols are arranged in the order of their decreasing probabilities.

Step 2. Let k be the integer satisfying the two requirements

$$\begin{aligned} &\text{(i)} \quad \text{For } M = 2,\ k = 2 \\ &\text{(ii)} \quad \text{For } 3 \leq k < M \end{aligned} \tag{1}$$

$$(N - k)/(M - 1) = \text{positive integer} \tag{2}$$

Here we add β $(=(N - k) \text{ MOD } (M - 1))$ dummy symbols (to the given symbol set), each with a zero probability of occurrence.

Group together k least probable symbols and compute the total probability of this subset.

Step 3. Construct an auxiliary ensemble of symbols from the original ensemble, by regarding the subset of k symbols formed in step 2 as a single symbol, with probability equal to the probability of the whole subset. Rearrange the symbols of the auxiliary ensemble in the order of their decreasing probabilities.

Step 4. Again form a subset of the k least possible symbols of the auxiliary ensemble and compute its total probability.

Step 5. Construct a second auxiliary ensemble from the first auxiliary ensemble, by regarding the subset of k symbols formed in step 4 as a single symbol, with probability equal to the total probability of the whole subset. Rearrange the symbols of this second auxiliary ensemble in the order of decreasing probabilities.

Step 6. Form successive auxiliary ensembles by repeating step 4 and step 5 until the total number of symbols in the auxiliary ensemble is equal to the number of code alphabets M.

Step 7. The preceding steps, when formally carried out, yield a tree for which the specified messages are the terminal nodes. Code words can be constructed by assigning different symbols from the prescribed alphabet to the branches stemming from each intermediate node.

PARENTHESIZED FORM OF HUFFMAN PROCEDURE

For computer implementation, we use a parenthesized representation (PR) of the tree structure encountered in Huffman coding;[1] this facilitates easy computer parsing. This is illustrated by an example below. Let

† Contribution No. 48, Molecular Biophysics Unit, Indian Institute of Science, Bangalore, India.

‡ School of Automation, Indian Institute of Science, Bangalore-12, India.

* Author to whom correspondence should be addressed.

$$S = \{s_1, s_2, s_3, s_4\}$$

$$P = \{0.6, 0.1, 0.1, 0.2\}$$

$$C = \{0, 1\}$$

$N = 4$, $M = 2$, and $k = 2$

Symbol	Frequency of occurrence	Probability
s_1	6	0.6
s_2	1	0.1
s_3	1	0.1
s_4	2	0.2

Step 1. The probabilities are arranged in decreasing order. For the sake of computer implementation, we represent the index "i" of a given symbol s_i and its probability of occurrence p_i as an ordered pair within a set of parentheses as (i, p_i). For the above example we obtain

Symbol	Probability	PR
s_1	0.6	(1, 0.6)
s_4	0.2	(4, 0.2)
s_2	0.1	(2, 0.1)
s_3	0.1	(3, 0.1)

Step 2. The sum of the least two probabilities is now computed and the computations which resulted in this sum are enclosed in parentheses. Thus, in the above example we combine (2, 0.1) and (3, 0.1) to obtain ((2, 0.1)(3, 0.1)0.2). The new list thus obtained is shown below.

Symbol grouping (SG)	Probability	PR
$\{s_1\}$	0.6	(1, 0.6)
$\{s_4\}$	0.2	(4, 0.2)
$\{s_2, s_3\}$	0.2	((2, 0.1)(3, 0.1)0.2)

Step 3. Again the new symbol set is reordered in decreasing order of their probabilities by considering the probability of the previously combined symbols, as the probability of a new symbol. Then, the least two probabilities are combined as before.

Step 4. Step 3 is reiterated until the total number of symbols in the new symbol set is equal to the number of available code symbols M.

For the above example we stop after two steps (since $k = 2$).

Comments: As a result of steps 3 and 4, we have the following PR for the two-member SG.

SG	PR
$\{s_1\}$	(1, 0.6)
$\{s_4, \{s_2, s_3\}\}$	((4, 0.2)((2, 0.1)(3, 0.1)0.2)0.4)

These are stored in an array A, word by word, each word consisting of either non-numeric characters (such as a left or a right parenthesis or a comma) or a real number (probability) or a numeral (suffix of a symbol). Let $L(A)$ be the number of words in A (length of A).

Using the above parenthesized representation, it is now required to assign the Huffman code for each one of the combined probabilities and also find the code corresponding to each symbol s_i.

The following properties of PR are easily observed.

(a) The individual or combined probabilities always occur immediately preceding a right parenthesis.

(b) The suffix i of symbol s_i is always preceded by one or more left parentheses; the code which is assigned to that right parenthesis which pairs with the innermost left parenthesis gives the code of s_i.

The algorithmic steps to be described below are based on the above properties.

Step 5. Here, we first assign the levels of parentheses in PR.

Step 5(a). Define an array PARENT (parenthesis − count). Let J be the index of PARENT. Set $J = 1$.

While scanning the array A, from left to right, if we encounter a left parenthesis, we put "1" in PARENT and its index J is incremented by 1; if, however, a right parenthesis is encountered, "2" is put in PARENT and its index J is incremented by 1. If the encountered symbol is neither of these, scanning continues until all characters are exhausted in A. Note $(J - 1)$ gives the total number of entries T in PARENT. Go to step 5(b).

For the above example, the array PARENT is given by

PARENT

	J									
SG	1	2	3	4	5	6	7	8	9	10
$\{s_1\}$	1	2								
$\{s_4, \{s_2, s_3\}\}$	1	1	2	1	1	2	1	2	2	2

Step 5(b). Define by LEVEL, the level of each parenthesis; its value is placed in an array VALUE. To start with, LEVEL and J are set to 1. Set VALUE(J) = 1.

Let IEND be set to PARENT(J).
J is set to 2.

(i) If IEND = PARENT(J) = 1 (implies successive left parentheses), then LEVEL is incremented by 1 and set VALUE(J) = LEVEL, and go to (iv); otherwise go to (ii).

(ii) If IEND = PARENT(J) = 2 (implies successive right parentheses), then LEVEL is decremented by 1 and set VALUE(J) = LEVEL, and go to (iv); otherwise go to (iii).

(iii) The LEVEL is unchanged and set VALUE(J) = LEVEL and go to (iv).

(iv) Set IEND to PARENT(J) and increment J by 1. If $J \leq T$ then go to (i); otherwise, go to step 5(c).

For the given example, we have

VALUE

	J									
SG	1	2	3	4	5	6	7	8	9	10
$\{s_1\}$	1	1								
$\{s_4, \{s_2, s_3\}\}$	1	2	2	2	3	3	3	3	2	1

Step 5(c). Since we are interested in assigning a code corresponding to each right parenthesis, it is enough to consider the levels of only right parentheses. Hence, we set VALUE(J) = 0 for all PARENT(J) $\neq 2$ for $J = 1$ to T. The nonzero entries of the array VALUE are then transferred in the same order as successive entries in a new PARENT array. We now denote the number of nonzero entries of PARENT (corresponding to the number of right parentheses) by NUMBER ($=T/2$). Let LMAX be the maximal element in the PARENT array (maximum level of the parenthesis). Note LMAX also indicates the maximum length of the code.

For the given example

Table I

I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
A	(	1	,	0.6	)																
INDEX(CODE)					1(0)																
A	(	(	4	,	0.2	)	(	(	2	,	0.1	)	(	3	,	0.1	)	0.2	)	0.4	)
INDEX(CODE)						1(11)						2(101)					3(100)		4(10)		5(1)

Table II. Huffman Code for WLN Symbols[a]

No.	Symbol	Probability	Code	No.	Symbol	Probability	Code
1	J	0.05431997	1111	40	*B	0.04498521	1001
2	T	0.05118707	1101	41	*D	0.03967079	01000
3	1	0.04981699	1011	42	*C	0.03570051	00100
4	V	0.04396745	1000	43	*E	0.02603333	11001
5	O	0.04324289	01110	44	*A	0.02141090	011011
6	&	0.04220161	01100	45	*F	0.01689645	000110
7	6	0.04155247	01011	46	*&	0.01152291	101001
8	N	0.04032416	01001	47	*G	0.00946930	0011001
9	R	0.03925995	00111	48	*H	0.00903010	0010100
10	5	0.02932430	00001	49	*I	0.00874897	0001110
11	U	0.02751864	00000	50	*2	0.00749894	0001010
12	Y	0.02721097	11101	51	*J	0.00547911	1010001
13	2	0.02382407	10101	52	*O	0.00378265	00010110
14	Q	0.02195869	011111	53	*K	0.00341706	11100101
15	—	0.02167755	011110	54	*L	0.00334828	11100100
16	M	0.02129265	011010	55	*M	0.00304000	11000001
17	S	0.01959076	001101	56	*N	0.00238180	010100001
18	L	0.01823697	001011	57	*P	0.00198182	000101111
19	G	0.01317775	111000	58	*1	0.00195226	000100011
20	W	0.01081163	0101011	59	*3	0.00129769	110000001
21	X	0.01001106	0101001	60	*R	0.00123253	0101000001
22	H	0.00916222	0010101	61	*Q	0.00111971	0101000000
23	3	0.00898003	0001111	62	*S	0.00091701	0001011100
24	/	0.00749411	0001001	63	*4	0.00082772	1100011101
25	Z	0.00709051	1110011	64	*T	0.00065940	1100000001
26	4	0.00620186	1100010	65	*X	0.00054115	00010111011
27	E	0.00601182	1100001	66	*5	0.00050496	00010111010
28	C	0.00534699	1010000	67	*U	0.00039878	00010001010
29	I	0.00522030	01010101	68	*0	0.00036439	11000111001
30	F	0.00503448	01010100	69	*V	0.00026304	000100010111
31	7	0.00492650	01010001	70	*6	0.00017857	110001110001
32	P	0.00472077	00110001	71	*W	0.00016892	110001110000
33	A	0.00347497	00010000	72	*8	0.00006757	00010001011010
34	K	0.00313652	11000110	73	*7	0.00004645	00010001011000
35	B	0.00231424	001100001	74	*Z	0.00003982	000100010110111
36	0	0.00218754	001100000	75	*9	0.00003077	000100010110011
37	8	0.00165423	110001111	76	*Y	0.00002353	000100010110010
38	9	0.00085306	0001000100	77	*/	0.00002051	0001000101101101
39	D	0.00061657	1100000000	78	*—	0.00001267	0001000101101100

[a] An asterisk denotes blank preceded characters.

	PARENT				
	←——————J——————→				
SG	1	2	3	4	5
$\{s_1\}$	1				
$\{s_4, \{s_2, s_3\}\}$	2	3	3	2	1

we get for the SG

NUMBER = 1, LMAX = 1 } corresponding to $\{s_1\}$

and

NUMBER = 5, LMAX = 3 } corresponding to $\{s_4, \{s_2, s_3\}\}$

Step 6. This step assigns the Huffman code for each level of the right parenthesis. Corresponding to PARENT (NUMBER) we assign the code "0" to the first combination in SG and code "1" to the second combination in SG.

If the NUMBER is equal to 1, then go to step 7; otherwise, go to step 6(a).

Step 6(a). NUMBER is decremented by 1. The code corresponding to previous right parenthesis is now read from storage CODE (NUMBER + 1).

Step 6(b). If PARENT(NUMBER) is greater than PARENT(NUMBER + 1) (this implies that the level of parenthesis under consideration is greater than that of the previous one), a "0" is added to the previously assigned code; go to step 6(c).

If the PARENT(NUMBER) is equal to the PARENT(NUMBER + 1) (this implies that the successive levels of the parentheses are equal), the last bit of the previously assigned code is set to "1"; then go to step 6(c).

If the PARENT(NUMBER) is less than the PARENT(NUMBER + 1) (this implies that the parenthesis under consideration has a level less than that of the previous one), the difference in their levels is computed and stored in LEDIFF.

Now take from the previously formed code only the most significant bits equal to PARENT(NUMBER); the other bits (LEDIFF) are ignored. Scan these bits from the least significant end (right to left); if "1" is encountered, it is set to "0" and the scanning is continued until a "0" is encountered. The encountered 0th bit is set to "1"; go to step 6(c).

Step 6(c). The present code is then stored in CODE (NUMBER). The NUMBER is compared with 1, and if it is greater than or equal to 1 go to step 6(b); otherwise, go to step 7.

For the example under consideration, we have the following codes for the various parenthesis levels.

CODE

SG	NUMBER 1	2	3	4	5
$\{s_1\}$	0				
$\{s_4, \{s_2, s_3\}\}$	11	101	100	10	1

We now have the situation which is shown in Table I at the end of step 6.

Step 7. In this step, we remove the symbols and their corresponding Huffman codes from the parenthesis representation.

Let I be the index of the array A, and NUMBER be the index of CODE.

Set I and NUMBER to 1.

(i) If $I \geq L(A)$ go to (vii); otherwise, go to (ii).
(ii) If $A(I)$ is a left parenthesis go to (v); otherwise, go to (iii).
(iii) If $A(I + 1)$ is a right parenthesis, go to (vi); otherwise, go to (iv).
(iv) $A(I)$ gives the suffix i of the symbol s_i. The CODE (NUMBER) is read from the storage. NUMBER is incremented by 1; I is incremented by 4; go to (i).
(v) If $A(I + 1)$ is again a left parenthesis, increment I by 2; go to (i); otherwise, increment I by 1 and go to (iv).
(vi) I is incremented by 2 and NUMBER is incremented by 1; go to (i).
(vii) List Huffman code for all s_i.

For the above example, we obtain

Symbol	Probability	Huffman code
s_1	0.6	0
s_2	0.1	101
s_3	0.1	100
s_4	0.2	11

HUFFMAN CODE FOR WLN SYMBOLS

Using the above procedure, the Huffman code for WLN symbols was constructed using the statistics available in ref. 2. The computer output is given in Table II.

Note that the minimal length of the code is 4 bits while the maximal length is 16 bits.

ADVANTAGES OF HUFFMAN CODING.

(i) Compression Ratio. A statistical analysis on a large number of chemical compounds shows that the use of the Huffman code results in compressing the WLN-file storage by 20–30%. This will, however, require programs for encoding into and decoding from the Huffman code. Since the codes for WLN symbols have already been found, it is only necessary to write programs for decoding. The time required for decoding is very small, and it seems worthwhile compressing the WLN file for economizing on the storage.

(ii) Use of Huffman–Shannon–Fano Code. Recently, Connell[3] has suggested another coding scheme which reduces storage space, transmission time, translation table space, and encoding–decoding times.

It is possible to modify the above procedure to obtain the Huffman–Shannon–Fano code for WLN symbols. This should be very useful for file compression.

(iii) Error Protection Feature. The chemical compounds can be coded in WLN, and the error detection feature described in ref 4 can be easily applied and the total code can be compressed using the Huffman procedure.

(iv) Programs Available. Fortran IV programs for Huffman coding of WLN symbols and decoding are available from the authors.

ACKNOWLEDGMENT

The authors would like to thank C. E. Granito, *et al.*,[2] for making available to them the statistical data on the occurrence of WLN symbols. Two of the authors (K. S. and S. K.) express their gratitude to the Indian Institute of Science and NCERT for providing research fellowships.

LITERATURE CITED

(1) Huffman, D. A., "A Method for the Construction of Minimum Redundancy Codes," *Proc. IRE*, **40**, 1098 (1952).
(2) Granito, C. E., Becker, G. T., Roberts, S., Wiswesser, W. J., and Windlinx, K. J., "Computer-Generated Substructure Codes (Bit Screens)," *J. Chem. Doc.*, **11**, 106–110 (1971).
(3) Connell, J. B., "A Huffman–Shannon–Fano Code," *Proc. IEEE*, **61**, 1046–1047 (1973).
(4) Subramanian, K., and Sankar, P. V., "Error Checking Digit for Nonconventional Chemical Codes," *J. Chem. Doc.*, **13**, 39–41 (1973).