A PerformanceAnalysis of Delta and Huffman Compression Algorithms

[a]Oke A. O. , [b]Fakolujo O. A, and[c]Emuoyibofarhe O.J

[a,c]Department of Computer Science and Engineering
LadokeAkintola University of Technology, Ogbomoso, Nigeria
[b]Department of Electrical Engineering
University of Ibadan.
Email address: alitemi2006@yahoo.com

ABSTRACT

With the recent trend in Information and Communication Technology, Storage and Transfer of data and Information are two vital issues which have Cost and Speed implication respectively. Large volume of data (text or image) is constantly being processed on the internet or on a Personal Computer, which has led to the Upgrade of current System. Hence the need for compression, which reduces storage capacity and effect Speed of transfer. Data Compression is the act of reducing the size of a file by minimizing redundant data. In a text file, redundant data can be frequently occurring characters or common vowels.

This research involves a comparative performance analysis of Huffman and Delta Compression schemes. A compression program is used to convert data from an easy-to-use format (ASCII) to one optimized for compactness. Huffman and Delta algorithms were implemented using C#. Result was also presented on the efficiency of the former based on three parameters: the number of bit, compression ratio and percentage of compression.

It was discovered that Huffman algorithm for data compression performs better, since it can store / transmit the least number of bits. The average compression percentage for Huffman and Delta algorithm was found to be 39% and 45% respectively. Which simply implies that for a large text file, Huffman algorithm will achieve a 39% reduction in the file size and as such increase the capacity of the storage medium.

Keywords: Data compression, Huffman algorithm, Delta algorithm

1.0 INTRODUCTION

Data compression is the act of forcing data together to occupy less space or pressing data together. Data compression reduces the size of a file by minimizing repetitive data. In a text file, repetitive data can be frequently occurring characters, such as the space character, or common vowels, such as the letters e and a, it can also be frequently occurring character strings. Data compressing creates a compressed version of a file by minimizing this repetitive data and as such produces faster transmission [5].

Most human communication is inherently redundant,this does not imply waste; rather human beings use that redundancy as a continual crosscheck on what information is really being sent and meant. For example, in face-to-face conversation much more information is being exchanged than just the words. Facial expressions, tones of voice, limb positions and movement, and other less obvious cues all contribute to the information stream flowing between two people having a conversation. Even though much of the information is duplicated but compression tends to remove duplication which is referred to as the redundancy.

In other words, data compression could be said to deal with how to take a large collection of binary bits, the life blood of digital technologies, and replaces this collection with a small compressed version of the original bits [4].

In data communications, compression is a technique applied either in advance to information to be transmitted or dynamically to an information stream being transmitted. The underlying technology is essentially the same in both cases: removal of repetitive information or expression of the information in a more compact form is used to reduce the total number of bytes that must pass over a communication medium in order to reduce the time the medium is occupied by a given transmission to a minimum [6].

A simple characterization of data compression is that it involves transforming a string of characters in some representation (such as ASCII) into a new string (of bits, for instance) which contains the same information but whose length is as small as possible. Data compression has important application in the areas of data transmission and data storage. Many data processing applications require storage of large volumes of data, and the proliferation of computer communication networks is resulting in massive transfer of data over communication links. Compressing data to be stored or transmitted reduces the costs involved. When the amount of data to be transmitted is reduced, the effect is that of increasing the capacity of the communication channel. Similarly, compressing a file to half of its original size is equivalent to doubling the capacity of the storage medium. It may then become feasible to store the data at a higher, thus faster, level of the storage hierarchy and reduce the load on the input / output channels of the computer system.

2.0 CODING

A code is a mapping of source messages (words from the source alphabet) into code words (words of the code alphabet). The source messages are the basic units into which the string to be represented is partitioned. These basic units may be single symbols from the source alphabet, or they may be strings of symbols. Coding is a very general term encompassing any special representation of data, which satisfy a given need.

In communications systems, coding can be said to be the altering of the characteristics of a signal to make the signal more suitable for an intended application, such as optimizing the signal for transmission, improving transmission quality and fidelity, modifying the signal spectrum, increasing the information content and providing error detection and/or correction.

3.0 DATA COMPRESSION

Data and programs are almost invariably entered in alphanumeric form, and the internal operation of computers, makes extensive use of alphanumeric codes. Alphabetic characters must be coded in binary form in order to store in computer memories (binary form because computer memories process it data in binary). These characters are letters such as A, a, B, b and so on.

A wide spread code for alphabetic characters is the American Standard code for information interchange (ASCII). ASCII [1] contains all the uppercase and lowercase English letters, the ten numeric digits and symbols. Another popular code for representing characters is the Extended Binary Coded Decimal Interchange Code (EBCDIC). ASCII has become an international standard published by the American National Standards Institute. It is an 8-bit code. 7 bits define one of 128 characters and the 8th bit is an optional parity bit.

Review of Related works

Early compression work mainly focused on disk space savingsand I/O performance [9,10] and 15]. However. later works acknowledgesthat compression can also lead to better CPU performance,as long as decompression costs are low [12]

Dictionary based domain compression, a lightweight compressionmethod where data values are mapped to fixedlength codes, is used in the implementation of many applications andhas been a subject of extensive research [8]. Entropycoding techniques, including Huffman encoding [14],are considered heavy-weight techniques. Both Huffman and arithmetic encoding have beenstudied and modified [12, 9, 11,13].

The Huffman algorithm and its variants using the Canonical Huffman coding scheme have been reported in [11] to be effective and efficient in data compression. The canonical Huffman coding is a reordering of the nodes of a Huffman tree so that shorter codes have smaller code values than longer codes and codes with the small length. In [16], C-Store was used to delta code data for column-wise storage and compression.

3.1 HUFFMAN CODING

Huffman coding assigns each column value a variable length code chosen so that more frequent values within the column are assigned shorter codes.
The basic idea here is assigning short code words to those input blocks with high probabilities and long code words to those with low probabilities. Huffman code can be categorized under the lossless fixed to variable code. The algorithm for constructing a Huffman code to represent items with given relative frequencies of occurrence proceeds in two phases [3]. First one constructs a Huffman tree based on the relative probabilities. Second, using the Huffman tree, one constructs the code words that go with the given relative frequencies. A simple example will illustrate the idea involved as shown in fig 3.1 and table 3.1.
A code is needed to represent 10 types of information (a - j) with relative frequencies.

Table 3.1.Information to be coded.

Information	a	b	c	d	e	f	g	h	i	j
Frequency	17	3	6	9	27	5	4	13	14	2

In all there are 100 items to represent. One begins the tree by assigning a vertex of a tree for each item, labeled so that the relative frequencies increase from left to right. We construct a tree by adding a new vertex to the previously existing construction by selecting the two vertices (say v and w) in the current structure with the smallest labels (in a case of a tie, any two is picked) and adding a new vertex to the structure with a label which is the sum of the labels at v and w. The new vertex is placed above and between the two vertices v and w. The tree is now used to construct the code words associated with the ten number of information as explained below.
The top vertex of the tree (the root) is labeled with a blank string. As we move down the tree, the next vertex one gets to is labeled with a binary string by appending 0 or 1 to the right according to whether one moves to the next vertex by taking a right branch or a left branch. (e.g. "a" vertex has the string"010" associated with it). This process winds up assigning short codeword to characters which occur frequently and long codeword to characters that rarely occur.

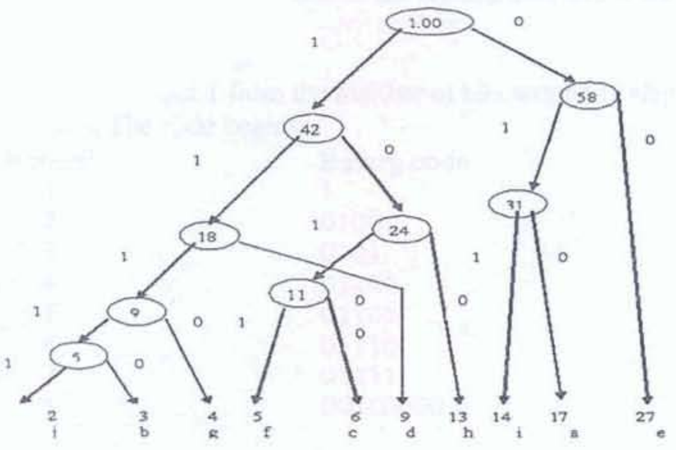


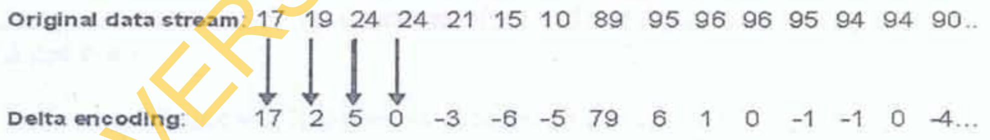
Fig. 3.1 Huffman Tree

Table 3.2. Code table

X	a	b	c	d	e	f	g	h	i	j
CODE	010	11110	1010	110	00	1011	1110	100	011	11111

3.2 DELTA ENCODING

Recall the use of the Greek letter delta (δ) science, engineering, and mathematics to denote the change in a variable, the delta encoding also refers to several techniques that store data as the difference between successive samples (or characters), rather than directly storing the samples themselves. Fig. 3.2. Shows an example of how this is done. The first value in the delta-encoded file is the same as the first value in the original data. The subsequent values in the encoded file are equal to the difference (delta) between the corresponding value in the input file, and the previous value in the input file.



Original data stream: 17 19 24 24 21 15 10 89 95 96 96 95 94 94 90..

Delta encoding: 17 2 5 0 -3 -6 -5 79 6 1 0 -1 -1 0 -4...

Delta coding is a branch of universal code. With universal codes it is not necessary to know the exact probabilities with which the source messages appear; but it is sufficient to know the probabilities distribution only to the extent that the source messages can be ranked in probability order. By mapping messages in order of decreasing probability to code words in order of increasing length. Delta encoding is one of the universal coding schemes, which map the set of positive integers onto the set of binary code words [2].

To code a number, the following algorithm holds:

- Write it in binary.

- Count the bits, remove the leading one, and write that number in binary preceding the
Previous bit string.
- Subtract 1 from the number of bits written in step 2 and prepend (affix) that many zeros. The code begins

Integer	Binary code
1	1
2	0100
3	0101
4	01100
5	01101
6	01110
7	01111
8	00100000

Table 3.3 Elias code generation.

Source message	Frequency	Rank	Codeword
g	8	1	delta (1) = 1
f	7	2	delta (2) = 0100
e	6	3	delta (3) = 0101
d	5	4	delta (4) = 01100
space	5	5	delta (5) = 01101
c	4	6	delta (6) = 01110
b	3	7	delta (7) = 01111
a	2	8	delta (8) = 00100000

Source message is the alphanumeric characters, frequency is the number of occurrence of the character in the message, and rank is the order of most frequency character: the most frequency character is ranked one (1), followed by the next most frequent and etc. codeword is the code generated.

4.0 RESULT

The message "go go gophers" was implemented using Huffman and Delta algorithm. The code generated from the two algorithms along with the ASCII equivalent is as shown in table 4.1 and analyzed below:

Table 4.1. Bits generated for the different codes

Character	ASCII Code	HUFFMAN Code	DELTA Code
G	11100111	10	1
0	01101111	01	0100
Space	00100000	111	0101
P	11110000	000	01100
H	11101000	0011	01101
E	01100101	0010	01110
R	01110010	1101	01111
S	11110011	1100	00100000

Standard code (ASCII) will store the message as

```
11100111   01101111   00100000   11100111   01101111   00100000
g             o        Space        g          o         Space
11100111   01101111   11110000   11101000   01100101   01110010
g             o          p          h          e           r
11110011
s
```

Huffman code will store the message as

```
10   01   111    10   01   111    10   01   000   0011   0010
g    o    space  g    o    space  g    o    p      h      e
1101 1100
r    s
```

Delta code stores message as

```
1   0100    0101   1   0100    0101   1   0100    01100   01101   01110
g      o    space  g      o    space  g   o         p       h        e
01111   00100000
r          s
```

Since data Compression is of great importance in the area of data storage and transmission, Discussion will centre on the two areas.

4.1 DATA STORAGE

The first parameter for which the compression algorithms were evaluated is the number of bit stored / transmitted: from fig. 4.1, it is discovered that Huffman algorithm for data Compression performs better, since it stores the least number of bit. Also when a text file has a number of characters occurring frequently, there is every tendency of achieving a better compression than one with characters that do not occur frequently. This can be seen from the difference that occurs in the number of bit from Table 4.2. for samples S1 and S2, 104 being the same number of bit for both samples in ASCII (uncompressed form) but 37 and 40 for sample S1 and sample S2 in Huffman and Delta (compressed form) respectively.

THE LENGTH OF BIT IN STORAGE MEDIUM

Standard Code (ASCII)
Number of characters: 13
Number of bit per character: 8
Word length: 13 * 8
Total bits (in storage): 104 bits

Huffman Code
Since it represents character with variable length:
Total bits = the sum of the number of bit represented by each character.
Total bits (in storage): 37 bits

Delta Code
It also represents its character with variable length:
Hence Total bits (in storage): 43 bits

COMPRESSION CHART

This chart provides a graphical representation of the result analyzed based on the compression algorithms. The samples tested are as follows:

S1 = go go gophers
S2 = this is a boy
S3 = go go gophers gum gum gophers
S4 = go go gophers gum gum gophers go go gum
S5 = peter piper picked a peck of pickle pepper
S6 = go go gophers gum gum gophers go go gum gumgum gophers
S7 = peter piper picked a peck of pickle pepper a peck of pickle pepper peter piper picked, if peter piper picked a peck of pickle pepper where is the peck of pickle pepper peter piper picked

Table 4.2. Number of bits stored / transmitted

TEXT	ASCII	HUFFMAN	DELTA
S1	104	37	43
S2	104	40	52
S3	232	93	110
S4	312	120	131
S5	344	139	150
S6	440	173	192
S7	1488	633	673

The first parameter of evaluation is the number of bits stored / transmitted using uncompressed ASCII, compressed HUFFMAN and DELTA bits against character text as shown on Table 4.2. The same information is represented in Fig. 4.1.

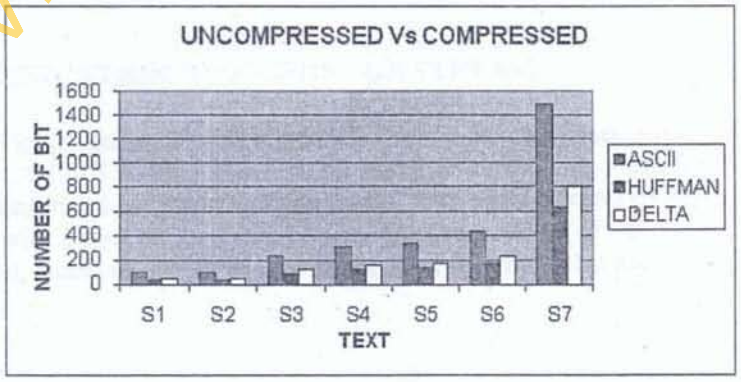


Fig.4.1.Uncompressed and Compressed data

The second parameter is the compression ratio of both algorithms and fig. 4.2. Shows Huffman having a higher Compression ratio.

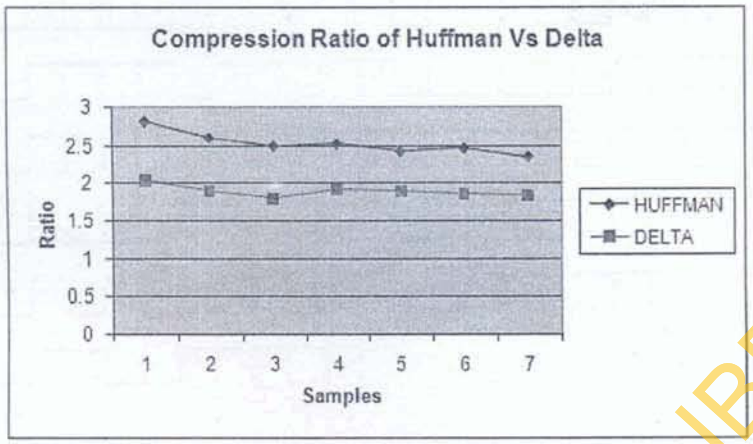


Fig. 4.2. Compression ratio of Huffman versus Delta

The third parameter of evaluation is percentage of compression. From Fig. 4.3, Huffman algorithm has the least percentage of compression.

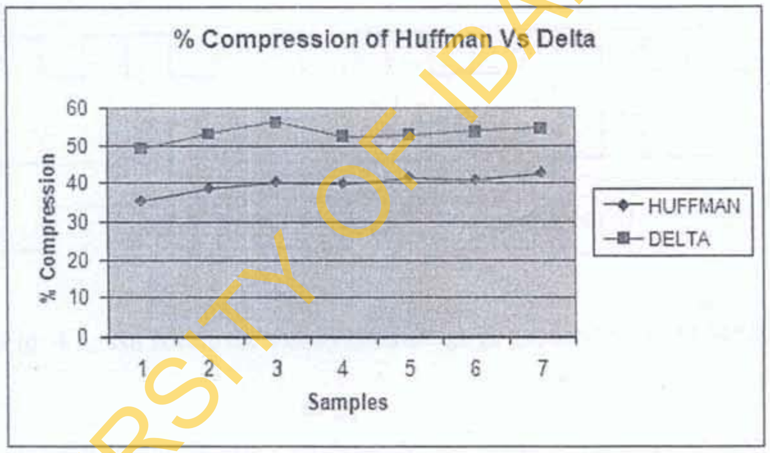


Fig. 4.3 Compression percentage of Huffman and Delta

4.2 COMMUNICATION TRANSMISSION STREAM

The sample "go go gophers" was transmitted over a communication line, using a serial data transfer, figs. 4.4, 4.5 and 4.6 show the bit stream of the message sent for the normal ASCII format, the generated Huffman and delta code. **In serial transfer of data, a character is transmitted with a start bit, two stop bits and the character bit of the different format. In the order start bit, character bit: starting with the least significant bit and stop bit.

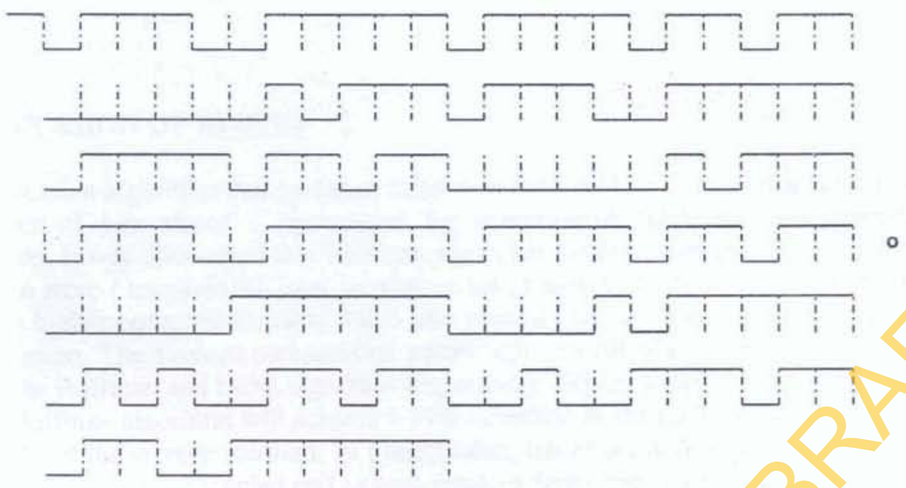


Fig. 4.4. Bit stream for transmission of "go go gophers" in ASCII

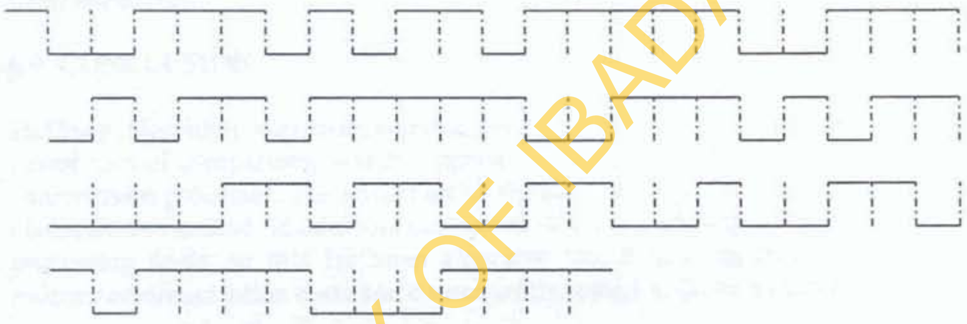


Fig. 4.5. Bit stream for transmission of "go go gophers" in HUFFMAN

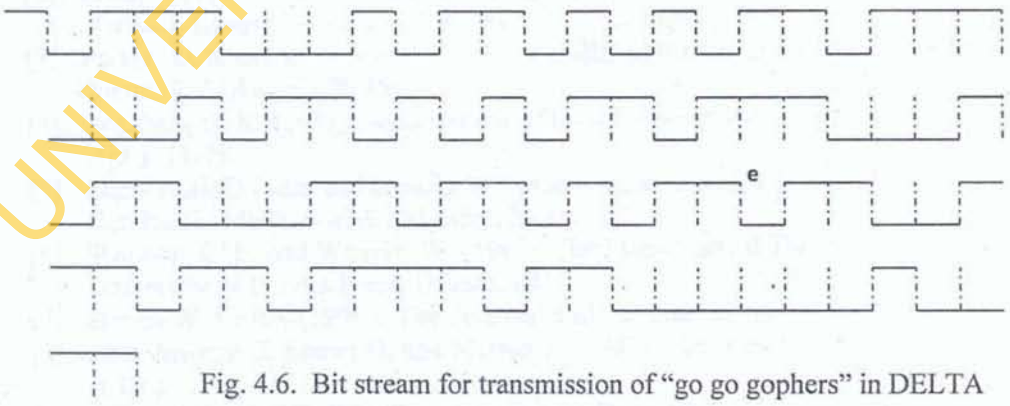


Fig. 4.6. Bit stream for transmission of "go go gophers" in DELTA

5.0 DISCUSSION OF RESULT

The compression algorithm that performs better was evaluated on three parameters. These are: the number of bits stored / transmitted the compression ratio and the percentage of compression. It was discovered that Huffman algorithm for data compression performs better, since it can store / transmit the least number of bit as shown in fig.6. Fig.7 shows Huffman having the higher compression ratio. Fig.8 also reveals Huffman having the least percentage of compression. The average compression percentage was calculated for all samples as 39% and 45% for Huffman and Delta algorithm respectively. Which simply implies that for a large text file, Huffman algorithm will achieve a 39% reduction in the file size and as such increase the capacity of the storage medium. In transmission, the effect will be to reduce the time the transmission channel is occupied and as such result in faster transmission as shown in fig.4.

It is also observed when we have a text file having a number of characters occurring frequently, there is every tendency of achieving a better compression than one with characters that rarely occurs taking a larger percentage. This can be seen from the difference that occurs in the number of bit in Table 4.2 for sample S1 and S2. 104 being the same number of bit for both samples in ASCII(uncompressed form) but 37 and 43 for sample S1; and 40 and 52 for sample S2 in Huffman and Delta(compressed form) respectively.

6.0 CONCLUSION

Huffman algorithm was discovered as better of the two schemes on the premise of the parameters of comparison, which supports (accounted) for the use of this algorithm in many compression processes. The advantage of Huffman coding is in the average number of bits per character transmitted. In addition, communication costs are beginning to dominate storage and processing costs, so that Huffman algorithm which is a variable "length-coding scheme" reduces communication costs and consequently results to faster transmission.

REFERENCES

[1]. Alan Clement (2000). Principles of Computer Hardware, Oxford Press, United Kingdom.
Third edition.
[2]. Elias, P. (1987). Interval and Recency Rank Source Coding: Two On-line Adaptive Variable-Length Schemes. IEEE Trans. Inform. Theory 33, 1
[3]. Parker, D. S. (1980). Conditions for optimality of the Huffman Algorithm. SIAM J. Comut. 9, 3 (Aug.), 470-489.
[4]. Reghbati, H. K. (1981). An overview of Data Compression Techniques. Computer 14, 4 (Apr.), 71-75.
[5]. Regis J. BUD Bates and Donald W. Gregory.(2000). Voice & Data communications Handbook. McGraw-Hill Publisher, New Delhi.
[6]. Shannon, C. E. and Weaver, W. (1949), The Mathematical Theory of Communication, University of Illinois Press, Urbana, III.
[7]. Steven W. Smith. (1999). The Scientist and Engineer Guide to digital signal Processing.
[8]. Antoshenkov G, Lomet D, and Murray J.(1996) Order preserving string compression. In ICDE,
[9]. Cormack G. (1985), Data Compresson on a Database System. CACM, 28(12).

[10] Goldstein, J., Ramakrishnan, R. and Shaft U. (1998), Compressing relations and indexes. In ICDE.
[11] Raman V. and Swart G.(2006) Entropy compression of relations and querying of compressed relations. In VLDB.
[12] Ray G, Haritsa J, and Seshadri S.(1995) Database compression: A performance enhancement tool. In COMAD.
[13] Zandi A, Iyer B, and Langdon G.(1993) Sort order preserving data compression for extended alphabets. In Data Compression Conference.
[14] Huffman D.(1952) A method for the construction of minimum-redundancy codes. In Proceedings of the I.R.E., pages 1098–1102.
[15] Iyer B. R and Wilhite D.(1994) Data compression support in databases. In VLDB.
[16] Stonebraker M,Wong E, Kreps P, and Held G.(1976) The implementation and design of INGRES. ACM Transactions on Database Systems, 1(3):189–222.