Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 11, Number 4 (2015), pp. 2419–2427

http://www.ripublication.com/gjpam.htm

b_i-Type perfect byte correcting binary codes

Vinod Tyagi

Department of Mathematics,
Shyamlal College (Evening),
University of Delhi, Delhi 110032, India.
E-mail: vinodtyagi@hotmail.com

Tarun Lata

Department of Mathematics,
University of Delhi, Delhi 110007, India.
E-mail: tarunlata1983@gmail.com

Abstract

Byte correcting codes have been studied by Chen [2], Hong and Patel [3], Tuvi [4, 5], Tyagi and Sethi [10]. Tuvi [4] has classified byte correcting perfect codes into five different categories depending upon the size of the byte and has also proved existence and non existence theorems. In this paper we study these byte correcting codes with the help of examples.

AMS subject classification: 94B25, 94B60.
Keywords: Perfect codes, Syndromes, b_i-Byte correcting codes, Byte oriented channels, Parity check matrix.

1. Introduction

The mathematical theory of error correcting codes has established itself as a very popular application of mathematics and as an important branch of modern communication theory. Burst is the most common error in almost all communication channels and burst error correcting codes are developed to deal with such errors. While many of these channels are normal, many others are byte oriented [1], [2]. In byte oriented channels bursts can occur only within the bytes. In such channels, if the bytes are of the same size say, β then the code length $n = m\beta$ when there are m bytes of size β and when they are of different size say, $\beta_1, \beta_2, \beta_3, \ldots, \beta_m$ then $\sum \beta_i = n$ for $i = 1, 2, \ldots, m$. If the codes are constructed for a byte-oriented memory, then, when we say that the code is b-byte correcting, meaning that it corrects a single burst of length b within the bytes of size β. So byte correcting codes are developed to correct errors within a byte.

For a code of length n and dimension k, the redundancy r is $n-k$. A code is called a b-byte correcting code if it can correct any single burst of length b or less with in a byte.

A burst of length b is a vector whose all the non zero components are confined to some b consecutive positions, the first and the last of which is non-zero.

For more information on byte-oriented burst-correcting codes and their applications, the reader is referred to [5], [6], [9].

Tuvi Etzion [4] has defined five types of Single-byte-correcting codes according to different sizes of the bytes viz.

Type 1: All bytes have the same size;

Type 2: One byte of size n_1 and other bytes of size n_2;

Type 3: Each byte is of either size n_1 or size n_2;

Type 4: The size of each byte is a power of 2;

Type 5: All the other cases.

and gave their existence and non existence theorems and construction. Tyagi and Sethi [10] on the other hand, considered a different situation and studied b_i-byte correcting perfect codes where $i = 1, 2, \ldots, m$.

This modification is based on an important observation and is possible in codes where it is known that a particular type of error may occur within a specified number of bytes and if one desires to increase the block length by adding some more bytes, it is natural to expect some more errors among the additional bytes. However, the errors, which are likely to occur in the additional bytes need not necessarily, be of the type that exists in earlier bytes. So there is a need to study b_i-type byte correcting perfect $(m\beta, m\beta - r)$ codes when all bytes are of same size or different size. In view of this observation, the byte correcting codes can now be classified with respect to the length of the burst also as follows:

Type 1.1: All bytes have the same size and same burst length.

Type 1.2: All bytes have the same size and different burst lengths b_i, $i = 1, \ldots, m$.

Type 1.3: All bytes have the same size but some have one burst length b_i for some i and others have other burst length b_j for some j; $b_i \neq b_j$.

Type 2.1: One byte of size n_1 and other bytes of size n_2 with same burst length b.

Type 2.2: One byte of size n_1 and all other bytes of size n_2 with different burst lengths.

Type 2.3: One byte of size n_1 with burst length b_1 and other bytes of size n_2 with burst length b_2.

Type 3.1: Each byte is of either size n_1 or size n_2 with same burst length.

Type 3.2: Each byte is of either size n_1 or size n_2 with different burst lengths.

Type 3.3: Each byte is of either size n_1 with burst length b_1 or size n_2 with burst length b_2.

Type 1: The size of each byte is a power of 2.

Type 1: All the other cases.

2. Necessary Condition

We state a theorem which gives a lower bound on the necessary number of parity check digits required for a code that corrects bursts of length b_1 or less in the first byte of size n_1, corrects bursts of length b_2 or less in the second byte of size n_2, and so on and corrects bursts of length b_m or less in the m^{th} byte of size n_m. The bound is based on the fact that the number of cosets is atleast as large as the number of error patterns to be corrected.

Theorem 2.1. The number of parity check digits in binary b_i byte-correcting perfect codes, correcting all bursts of length b_1 or less in the first byte of size n_1, correcting all bursts of length b_2 or less in the second byte of size n_2, and so on and correcting all bursts of length b_m or less in the m^{th} byte of size n_m is atleast

$$2^r - 1 \geq \sum_{i=1}^{m} [(n_i - b_i + 2)2^{b_i - 1} - 1] \tag{2.1}$$

This result is obtained with the help of well known Reiger bound [7] and lower bound proved by Tyagi and Sethi [10].

3. b_i-Byte Correcting Binary Perfect Codes

As we have mentioned earlier, we are interested in b_i-type perfect byte correcting binary codes for different types given by Tyagi and Sethi [10]. Clearly if the codes are constructed for a byte oriented memory then we say that the code is $(b_1, b_2, \ldots, b_m)_\beta = (b_i)_{\beta_i}$-byte correcting meaning that it corrects a single burst of length b_1 or less within the first byte, single burst of length b_2 or less within the second byte and so on where length of the byte is β and symbol is different for different sizes of the bytes i.e. $(b_1, b_2, \ldots, b_m)_\beta = (b_i)_{\beta_i}$-byte.

4. Discussion and Examples

We now present all possible different type of binary codes mentioned by Tuvi [4], [5] and Tyagi and Sethi [10] for different values of the parameters. We also show their existence with the help of examples.

Type 1.1. Where all bytes have the same size and same burst length. i.e., $n_1 = n_2 = n_3 = \cdots = n_m = n, b_1 = b_2 = b_3 = \cdots = b_m = b$.

Example 4.1. The following matrix is considered as a parity check matrix H_1 for (9, 5) code where, $n_1 = n_2 = n_3 = 3$, $b_1 = b_2 = b_3 = 2$ given by Tuvi [5].

$$H_1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Here it can be verified from the following error pattern-syndrome table that the code is a perfect code.

Error pattern	Syndrome
110 000 000	0101
011 000 000	1100
100 000 000	1110
010 000 000	1011
001 000 000	0111
000 110 000	0100
000 011 000	1111
000 100 000	1101
000 010 000	1001
000 001 000	0110
000 000 110	1010
000 000 011	0011
000 000 100	1000
000 000 010	0010
000 000 001	0001

We have shown in this example that the code is 2-perfect byte correcting code of Type 1.1.

Type 1.2. Where all bytes have the same size but different burst length. i.e., $n_1 = n_2 = n_3 = \cdots = n_m = n, b_1 \neq b_2 \neq b_3 \neq \cdots \neq b_m$.

Example 4.2. For, $n_1 = n_2 = 4$, $b_1 = 3$, $b_2 = 1$, if the following matrix is considered as a parity check matrix H_2 for (8, 4) code

$$H_2 = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix},$$

then it can be verified that the code is (8, 4) perfect code of Type 1.2.

Type 1.3. Where all bytes have the same size but some have one burst length b_i for some i and others have other burst length b_j for some j, where $b_i \neq b_j$. i.e., $n_1 = n_2 = n_3 = \cdots = n_m = n, b_1 \neq b_2 = b_3 = \cdots = b_m$.

Example 4.3. For, $n_1 = n_2 = n_3 = 2$, $b_1 = 2$, $b_2 = b_3 = 1$, if the following matrix is considered as a parity check matrix H_3 for (6, 3) code.

$$H_3 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix},$$

Then it can be verified that the code is a perfect byte correcting code of Type 1.3.

Type 2.1. Where one byte of size n_1 and other bytes of size n_2 with same burst length. i.e. , $n_1 \neq n_2 = n_3 = \cdots = n_m, b_1 = b_2 = b_3 = \cdots = b_m = b$.

Example 4.4. For, $n_1 = 3$, $n_2 = n_3 = n_4 = 4$, $b_1 = b_2 = b_3 = b_4 = 1$, if the given matrix is considered as a parity check matrix H_4 for (15, 11) code

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix},$$

then it can easily be verified that the (15, 11) code is a perfect code.

Type 2.2. Where one byte of size n_1 and other bytes of size n_2 but different burst length. i.e. , $n_1 \neq n_2 = n_3 = \cdots = n_m = n, b_1 \neq b_2 \neq b_3 \neq \cdots \neq b_m$.

Example 4.5. For, $n_1 = 7$, $n_2 = n_3 = 5$, $b_1 = 1$, $b_2 = 2$, $b_3 = 3$, if the following matrix is considered as a parity check matrix H_5

$$H_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

then the (17, 12) code that results from this matrix can be verified for its perfectness of Type 2.2.

Type 2.3. Where one byte of size n_1 and other bytes of size n_2 but some have one burst length b_i for some i and others have other burst length b_j for some j, where $b_i \neq b_j$. i.e. , $n_1 \neq n_2 = n_3 = \cdots = n_m = n, b_1 \neq b_2 = b_3 = \cdots = b_m$.

Example 4.6. For, $n_1 = 3, n_2 = n_3 = n_4 = n_5 = n_6 = 2, b_1 = 2, b_2 = b_3 = b_4 = b_5 = b_6 = 1$, if the following matrix is considered as a parity check matrix H_6 for (13, 9) code

$$H_6 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix},$$

then it can be verified that the (13, 9) code is a perfect code of Type 2.3.

Type 3.1. Where each byte is of either size n_1 or size n_2 with same burst length.

Example 4.7. For, $n_1 = n_2 = n_3 = 2, n_4 = n_5 = n_6 = 3, b_1 = b_2 = b_3 = b_4 = b_5 = b_6 = 1$, the following matrix is considered as a parity check matrix H_7 for (15, 11) code

$$H_7 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix},$$

then it can be easily verified the perfectness of code.

Type 3.2. Where each byte is of either size n_1 or size n_2 but different burst length.

Example 4.8. For, $n_1 = 7, n_2 = n_3 = 5, b_1 = 1, b_2 = 2, b_3 = 3$, the following matrix is considered as a parity check matrix H_8 for (17, 12) code

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Here it can be verified that the code is a (17, 12) perfect byte correcting code of Type 3.2.

Type 3.3. Where each byte is of either size n_1 or size n_2 but some have one burst length b_i for some i and others have other burst length b_j for some j, where $b_i \neq b_j$.

Example 4.9. For, $n_1 = n_2 = 2, n_3 = n_4 = n_5 = 3, b_1 = b_2 = 2, b_3 = b_4 = b_5 = 1$, the following matrix is considered as a parity check matrix H_9 for (13, 9) code

$$H_9 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix},$$

then it can be easily verified by the error pattern and syndrome table that the code is a Type 3.3 perfect code.

Type 4.1. Where size of each byte is equal and a power of 2 and same burst length.

Example 4.10. For, $n_1 = n_2 = n_3 = n_4 = n_5 = 2$, $b_1 = b_2 = b_3 = b_4 = b_5 = 2$, if the following matrix is considered as a parity check matrix H_{10}

$$H_{10} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Here it can be verified that the code is a perfect byte correcting code.

Type 4.2. Where size of each byte is equal and a power of 2 but different burst length.

Example 4.11. For, $n_1 = n_2 = 4$, $b_1 = 3$, $b_2 = 1$, if the following matrix is considered as a parity check matrix H_{11} for $(8, 4)$ code

$$H_{11} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix},$$

then it can be easily verified that the code is a perfect code.

Type 4.3. Where size of each byte is equal and a power of 2 but some have one burst length b_i for some i and others have other burst length b_j for some j, where $b_i \neq b_j$.

Example 4.12. For, $n_1 = n_2 = n_3 = 2$, $b_1 = 2$, $b_2 = b_3 = 1$, if the following matrix is considered as a parity check matrix H_{12} for $(6, 3)$ code

$$H_{12} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Then the code can easily be verified to be perfect.

Type 5.1. Where size of each byte is different but same burst length. i.e. , $n_1 \neq n_2 \neq n_3 \neq \cdots \neq n_m$, $b_1 = b_2 = b_3 = \cdots = b_m = b$.

Example 4.13. For, $n_1 = 4, n_2 = 5, n_3 = 6, b_1 = b_2 = b_3 = 1$ if the following matrix is considered as a parity check matrix H_{13} for (15, 11) code

$$H_{13} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix},$$

then its perfectness can be easily verified.

Type 5.2. Where size of each byte is different with different burst length. i.e. , $n_1 \neq n_2 \neq n_3 \neq \cdots \neq n_m, b_1 \neq b_2 \neq b_3 \neq \cdots \neq b_m$.

Example 4.14. For, $n_1 = 4, n_2 = 3, n_3 = 15, b_1 = 3, b_2 = 2, b_3 = 1$, if the following matrix is considered as a parity check matrix H_{13} for (22, 17) code

$$H_{14} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Then it can be seen that the code is a (22, 17) perfect byte correcting code of Type 5.2.

Type 5.3. Where size of each byte is different but some bytes have one burst length b_i for some i and others have other burst length b_j for some j, where $b_i \neq b_j$. i.e. , $n_1 \neq n_2 \neq n_3 \neq \cdots \neq n_m, b_1 \neq b_2 = b_3 = \cdots = b_m$.

Example 4.15. For, $n_1 = 3, n_2 = 4, n_3 = 6, b_1 = 2, b_2 = b_3 = 1$, if the matrix H_{15} is considered as a parity check matrix for (13, 9) code

$$H_{15} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix},$$

then it can be verified that the code is a perfect code of Type 5.3.

5. Conclusion and Open Problem

We have shown here in all the examples that b_i-byte correcting binary perfect codes for all types exist. It will be interesting to find the possibilities of all these byte oriented codes in non-binary case.

References

[1] C.L. Chen, Error correcting codes with byte error detection capabilities, *IEEE Trans. Comput.*, **C-32** (1983) 615–621.

[2] C.L. Chen, Byte oriented error correcting codes with semiconductor memory systems, *IEEE Trans. Comput.*, **C-35** (1986) 646–648.

[3] S.J. Hong and A.M. Patel, A general class of maximal codes for computer applications, *IEEE Trans. Comput.*, **C-21** (1972) 1322–1331.

[4] Tuvi Etzion, Perfect byte correcting codes, *IEEE Trans. Inform. Theory*, **44** (1998) 3140–3146.

[5] Tuvi Etzion, Construction for perfect 2-burst correcting codes, *IEEE Trans. Inform. Theory*, **47** (6) (2001) 2553–2555.

[6] T.R.N. Rao and E. Fujiwara, Error control coding for computer system, Englewood Cliff, NJ: Prentice Hall (1989).

[7] S.H. Reiger, Codes for the correction of clustered errors, *IRE Trans. Inform. Theory*, **IT-6** (1960) 16–21.

[8] Vinod Tyagi and Amita Sethi, $(n_{1b_1}, n_{2b_2}, n_{3b_3})$-Burst correcting linear codes, *Beykent University Journal of Pure and Applied Sciences*, **3** (2) (2007) 301–319.

[9] Vinod Tyagi and Amita Sethi, (b_1, b_2)-Optimal byte correcting codes, *Proceedings of XIIth workshop of ACCT*, pp. 280–285, September 5-11, Akademgorodok, Novosibirsk, Russia (2010).

[10] Vinod Tyagi and Amita Sethi, b_i-Byte correcting perfect codes, *Asian-European Journal of Mathematics*, **7** (1) (2014) 1–8.