KOREKSI KESALAHAN

a. KODE HAMMING

Kode Hamming merupakan kode non-trivial untuk koreksi kesalahan yang pertama kali diperkenalkan. Kode ini dan variasinya telah lama digunakan untuk control kesalahan pada system komunikasi digital. Kode Hamming ada dua macam, yaitu biner dan non-biner. Kode Hamming biner dapat direpresentasikan dalam bentuk persamaan berikut:

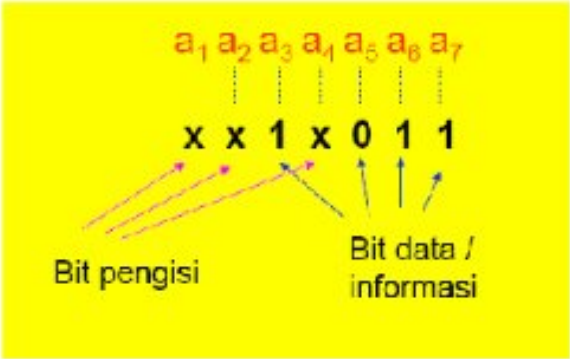
$(n,k) = (2^m-1,2^m-1-m)(1.7)$

Dimana k adalah jumlah bit informasi yang membentuk n bit kata sandi, dan m adalah bilangan bulat positif. Jumlah paritas bitnya adalah sejumlah m=n-k bit.

Sebagai contoh : Jika m3 dan n=7 dan k=4, kode hammingnya adalah C(7,4) dengan dmin=3. Tabel berikut menunjukkan data kata dan kode kata dari kode C(7,4):

Datawords	Codewords	Datawords	Codewords
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	101110
0100	0100011	1100	1100101
0101	0101110	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111

Cara pengisian bit tambahan pada bit – bit informasi x



Untuk bit data 4-bit, maka posisi bit informasi terletak pada a3, 35, a6 dan a7, sedangkan bit tambahan atau bit pengisi terletak pada a1, a2, dan a4. Hal ini dikarenakan letak bit tambahan harus berada pada indek data di a_{2^n}. dimana n = 0, 1,2,3,…

Jumlah bit informasi = 2^k-k-1, dimana k adalah jumlah bit ceknya.

Table jumlah bit cek dan bit informasi

Σ bit tambahan/pengisi	Σ bit data/informasi
2	1
3	4
4	11
5	26

Tabel hamming untuk informasi 4 bit

Data/bit	a_1	a_2	a_3	a_4	a_5	a_6	a_7
0000	0	0	0	0	0	0	0
0001	1	1	0	1	0	0	1
0010	0	1	0	1	0	1	0
0011	1	0	0	0	0	1	1
0100	1	0	0	1	1	0	0
0101	0	1	0	0	1	0	1
0110	1	1	0	0	1	1	0
0111	0	0	0	1	1	1	1
1000	1	1	1	0	0	0	0
1001	0	0	1	1	0	0	1
1010	1	0	1	1	0	1	0
1011	0	1	1	0	0	1	1
1100	0	1	1	1	1	0	0
1101	1	0	1	0	1	0	1
1110	0	0	1	0	1	1	0
1111	1	1	1	1	1	1	1

Nilai bit pengisi / cek: (untuk informasi 4-bit)

$$a_1 = a_3 \oplus a_5 \oplus a_7$$
$$a_2 = a_3 \oplus a_6 \oplus a_7$$
$$a_4 = a_5 \oplus a_6 \oplus a_7$$

Untuk informasi n-bit, nilai bit pengisi / cek adalah:

a_1= 3,5,7,9,11,13,15,...
a_2= 3,6,7,10,11,14,15,...
a_4= 5,6,7,12,13,14,15,20,21,22,23,...
a_8= 9-15,24-31,40-47,...
a_{16}= 17-31,48-63,80-95,...
a_{32}= 33-63,96-127,160-191,...
dst.

Bit-bit masing-masing posisi yang disertakan di Ex-OR kan

Contoh: Bagaimana bentuk data yang ditransmisikan dengan kode Hamming, jika diketahui bit data = 1010 ?

Jawab:

a1 = a3 + a5 + a7 → a1 = 1 + 0 + 0 = 1

a2 = a3 + a6 + a7 → a2 = 1 + 1 + 0 = 0

a4 = a5 + a6 + a7 → a3 = 0 + 1 + 0 = 1

Sehingga bentuk data yang ditransmisikan menjadi: 1011010

Cara penge-cekan disisi penerima: (untuk informasi 4-bit)

$$e_1 = a_1 \oplus a_3 \oplus a_5 \oplus a_7$$
$$e_2 = a_2 \oplus a_3 \oplus a_6 \oplus a_7$$
$$e_3 = a_4 \oplus a_5 \oplus a_6 \oplus a_7$$

Jika nilai **e = 0**, maka seluruh data yang diterima adalah benar

Untuk informasi n-bit, cara penge-cek an adalah:

1. Tanda semua posisi bit yang merupakan pangkat dua sebagai bit parity (posisi 1, 2, 4, 8, 16, 32, 64, ...)

2. Posisi yang lain digunakan sebagai bit data yang akan dikodekan (posisi 3, 5, 6, 7, 9, 10, 11, 12, 13, ...)

3. Masing-masing bit pengecek menghitung bit setiap posisi dengan cara menge-cek dan melewati, sebagai berikut:

   Posisi 1 : cek 1 bit, lewat 1 bit, cek 1 bit, lewat 1 bit dsb (1,3,5,7,9,11,13,15…)

   Posisi 2 : cek 2 bit, lewat 2 bit, cek 2 bit, lewat 2 bit dsb (2,3,6,7,10,11,14,15,…)

   Posisi 4 : cek 4 bit, lewat 4 bit, cek 4 bit, lewat 4 bit dsb (4,5,6,7,12,13,14,15,20,21,22,23, …)

   *Posisi 8 :*cek 8 bit, lewat 8 bit, cek 8 bit, lewat 8 bit dsb (8-15,24-31,40-47,...)

   Posisi 32 : cek 32 bit, lewat 32 bit, cek 32 bit, lewat 32 bit, dsb (32-63,96-127,160-191**,...)**

Bernilai bit parity = 1 jika total bit “1” diposisi yang dicek adalah ganjil (*Odd*) dan berinilai 0 jika total bit “1” adalah genap (*Even*)

Contoh:

Sebuah urutan data diterima: 0010011

Dengan:

e1 = 0

e2 = 1

e4 = 0

Tentukan bit diposisi mana yang salah?

Berapa nilai data asli (sebelum ditambah bit parity)?

Jawab:

e1 = a1 + a3 + a5 + a7 = 0 + 1 + 0 + 1 = 0 → benar
e2 = a2 + a3 + a6 + a7 = 0 + 1 + 1 + 1 = 1 → salah
e3 = a4 + a5 + a6 + a7 = 0 + 0 + 1 + 1 = 0 → benar

a1 = a3 + a5 + a7 = 1 + 0 + 1 = 0 → sama dengan yang dikirim
a2 = a3 + a6 + a7 = 1 + 1 + 1 = 1 → tidak sama dengan yang dikirim
a3 = a5 + a6 + a7 = 0 + 1 + 1 = 0 → sama dengan yang dikirim

Berarti bit diposisi 2 yang salah, seharusnya yang diterima adalah: 0110011

Nilai data asli= a3 a5 a6 a7 = 1011

b. FORWARD ERROR CORRECTION

Error-correcting codes dinyatakan sebagai penerusan koreksi kesalahan untuk mengindikasikan bahwa pesawat penerima sedangn mengoreksi kesalahan. Contohnya, pada komunikasi broadcast digunakan transmisi simplex. Metode transmisi ulang dinyatakan sebagai koreksi kesalahan balik karena pesawat penerima member informasi balik ke pesawat pengirim yang kemudian mentransmisi ulang data yang salah.

Kode pendeteksian kesalahan yang paling banyak digunakan merupakan kode hamming. Kode ini menggunakan bit-bit cek paritas yang diletakkan pada posisi tertentu pada setiap blok data. Bit-bit ini memungkinkan dilakukannya pemeriksaan paritas jamak pada terminal penerima.

Posisi bit-bit hamming dinyatakan dalam 2^n dengan n bilangan bulat sehingga bit-bit hamming akan berada pada posisi 1,2,4,8,16,dan seterusnya. Sebagai contoh:

Diasumsikan terdapat pesan 11000101100100. Setelah bit hamming disisipkan, pesan menjadi 110X0010110X010X0XX. Nilai X perlu diterapkan, 1 atau 0. Untuk menentukan setiap posisi bit 1, data ditandai dan nilai biner ditambahkan ke masing-masing posisi menggunakan aritmatika modulo 2. Pada modulo 2 penambahan bit 1 dengan cacah genap akan menghasilkan 0, dan dengan cacah ganjil akan menghasilkan 1 tanpa ada yang dibawa.

Pada contoh ini bit-bit 1 berada pada posisi 6,10,11,13,18,dan 19 penjumlahan bit ini akan memberikan hasil seperti berikut:

19	**1**	**0**	**0**	**1**	**1**
18	**1**	**0**	**0**	**1**	**0**
13	**0**	**1**	**1**	**0**	**1**
11	**0**	**1**	**0**	**1**	**1**
10	**0**	**1**	**0**	**1**	**0**
6	**0**	**0**	**1**	**1**	**0**
	0	**1**	**0**	**1**	**1**

Penyisipan bit-bit hamming dalam data akan menghasilkan 1100001011010100011. Pada computer penerima, nilai biner dari setiap posisi bit 1 ditambahkan pada untaian bit yang diterima. Dengan menggunakan aritmatika modulo-2, jika data diterima tanpa kesalahan, hasil penjumlahan sama dengan nol, seperti berikut:

19	**1**	**0**	**0**	**1**	**1**
18	1	0	0	1	0
13	0	1	1	0	1
11	0	1	0	1	1
10	0	1	0	1	0
8	0	1	0	0	0
6	0	0	1	1	0
2	0	0	0	1	0
1	0	0	0	0	1
	0	0	0	0	0

Jika terdapat kesalahan pada bit tunggal, kesalahan itu dapat ditunjukkan dengan melihat hasil penjumlahan modulo-2. Diasumsikan bahwa bit 11

dari data yang diterima berisi kesalahan, yaitu 0 dan bukan 1. Penjumlahan modulo-2 akan memberikan hasil seperti berikut:

19	**1**	**0**	**0**	**1**	**1**
18	1	0	0	1	0
13	0	1	1	0	1
10	0	1	0	1	0
8	0	1	0	0	0
6	0	0	1	1	0
2	0	0	0	1	0
1	0	0	0	0	1
	0	1	0	1	1

Hasil 01011 sama dengan nilai decimal 11 yang merupakan lokasi dari bit yang salah sehingga bit ke 11 dapat dikoreksi dengan membalik nilai bitnya.

c. FEEDBACK (BACKWARD) ERROR CONTROL

Dimana setiap karakter atau frame memiliki informasi yang cukup untuk memperoleh penerima mendeteksi bila menemukan kesalahan tetapi tidak menemukan lokasinya. Sebuah transmisi kontrol digunakan untuk meminta pengiriman ulang, menyalin informasi yang dikirimkan.

Kontrol kesalahan arus balik ini dibagi menjadi 2 bagian, yaitu:

1. Teknik yang digunakan untuk mendeteksi kesalahan
2. Kontrol algoritma yang telah disediakan untuk mengontrol transmisi ulang

Metode deteksi kesalahan:

1. Echo: metode sederhana dengan system interaktif. Operator memasukkan data melalui terminal dan mengirimkannya ke komputer. Komputer akan menampilkan kembali di terminal sehingga dapat diperiksa apakah data dikirimkan dengan benar.
2. Kesalahan otomatis/ Parity Check : penambahan bit paritas untuk akhir masing-masing kata dalam frame. Tetapi problem dari bit paritas adalah impulse noise yang cukup panjang yang merusak lebih dari satu bit pada kecepatan data yang tinggi.