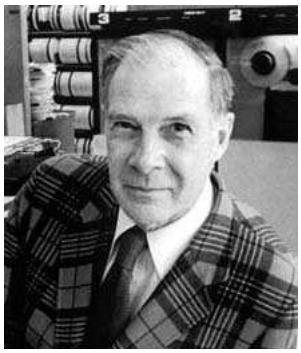
Théorie des codes - TP 3
ZZ3 F5 - Réseaux et Sécurité Informatique
Codes linéaires : Code de Hamming

Richard Hamming (1915-1998) introduit en 1950 [a] dans le cadre de son travail pour les laboratoires Bell le code de Hamming (7,4). C'est un code correcteur linéaire binaire, à travers un message de sept bits, il transfère quatre bits de données et trois bits de parité. Il permet la correction de toute erreur portant sur un unique bit.
Il travaillait, à l'époque, sur un modèle de calculateur à carte perforée de faible fiabilité qui, durant les périodes chômées comme la fin de semaine, s'arrêtait invariablement sur des bugs. La frustration d'Hamming le conduisit à inventer ce premier code correcteur véritablement efficace.


R. Hamming (1915-1998)

a. Richard Hamming, "Error-detecting and error-correcting codes", *Bell System Technical Journal*, 29(2) :147-160, 1950.

Code de Hamming (7,4)

Le code de Hamming permet la détection et la localisation (donc la correction) d'une erreur par bloc. Dans le cas du code de Hamming (7,4) on avons les matrices suivantes :

Matrice de vérification V : On peut écrire la matrice de vérification V en écrivant les entiers de 1 à 7 sous la forme binaire dans chaque colonne de V :

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

Si le syndrome $s = Vc_r^T$ où c_r est le code reçu, est non nul (présence d'erreurs), alors s est une colonne de la matrice V. Le numéro de la colonne donne la position de l'erreur. L'intérêt de la forme de cette matrice (justement sous forme non systématique) est que la $j^{\text{ième}}$ colonne de V est l'expression du nombre j en base 2. Donc le syndrome s est exactement la position de l'erreur exprimée en base 2.

Matrice génératrice G : La matrice génératrice associée à la matrice de contrôle V (vous pouvez essayer de le vérifier) est :

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

On peut écrire ces matrices sous forme systématique :

$$V = \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \quad G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right].$$

1. Réalisez le codage par le code de Hamming (7,4) du fichier à votre disposition (`test.txt`).
2. Réalisez également le décodage des données.
3. Simulez une transmission des données en rajoutant des erreurs.
4. Finalement, programmez une fonction permettant de calculer la distance de Hamming entre deux mots. Vous l'utiliserez pour implémenter le calcul de la distance minimale d'un code (vous pouvez vérifier que le code de Hamming (7,4) est de distance 3).

Remarque : Pour réaliser les différentes fonctions vous pouvez utiliser les opérations classiques bit à bit [1] ou les *bit field* [2] voire une libraire dédiée [3].

Code de Hamming

1. En utilisant l'algorithme général en annexe, réaliser une fonction permettant de créer le codage ou la matrice génératrice d'un code de Hamming ($2^m - 1$, $2^m - m - 1$).
2. Vérifiez que la distance minimale du code est bien 3 à l'aide de la fonction programmée à l'exercice précédent.
3. Réalisez le décodage et la correction des données reçues. Testez le codage sur le fichier `test.txt` et comparez les données envoyées et reçues.

Code de Hamming étendu

La dimension égale à $2m - 1$ n'est pas idéal, en terme pratique, il est préférable d'utiliser une dimension de la forme $2m$. De plus, le code de Hamming corrige une erreur, mais si deux erreurs se produisent, non seulement le code ne le détecte pas, mais en plus il en ajoute une troisième. En général on ajoute une dernière somme de contrôle validant la parité des $2m - 1$ premiers éléments du code. Une deuxième erreur est alors détectée, même si elle ne peut être corrigée sans nouvelle transmission.
Le **code de Hamming étendu** de paramètre ($2m$, $2m - m - 1$, 4) correspond à un code de Hamming classique ($2^m - 1$, $2^m - m - 1$,3) auquel a été ajouté un bit de parité portant sur les $2m - 1$ lettres du mot du code.

1. Mettez en place le codage et le décodage (détection et correction) en utilisant le codage de Hamming étendu.

1. `http://en.cppreference.com/w/cpp/utility/bitset`
2. `http://en.cppreference.com/w/cpp/language/bit_field`
3. `http://web.eecs.utk.edu/~plank/plank/papers/CS-07-593/`

Appendix General algorithm of code Hamming

The following general algorithm [4] generates a single-error correcting (SEC) code for any number of bits.

1. Number the bits starting from 1 : bit 1, 2, 3, 4, 5, etc.
2. Write the bit numbers in binary : 1, 10, 11, 100, 101, etc.
3. All bit positions that are powers of two (have only one 1 bit in the binary form of their position) are parity bits : 1, 2, 4, 8, etc. (1, 10, 100, 1000)
4. All other bit positions, with two or more 1 bits in the binary form of their position, are data bits.
5. Each data bit is included in a unique set of 2 or more parity bits, as determined by the binary form of its bit position.
   (a) Parity bit 1 covers all bit positions which have the least significant bit set : bit 1 (the parity bit itself), 3, 5, 7, 9, etc.
   (b) Parity bit 2 covers all bit positions which have the second least significant bit set : bit 2 (the parity bit itself), 3, 6, 7, 10, 11, etc.
   (c) Parity bit 4 covers all bit positions which have the third least significant bit set : bits 4-7, 12-15, 20-23, etc.
   (d) Parity bit 8 covers all bit positions which have the fourth least significant bit set : bits 8-15, 24-31, 40-47, etc.
   (e) In general each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

The form of the parity is irrelevant. Even parity is simpler from the perspective of theoretical mathematics, but there is no difference in practice.

This general rule can be shown visually :

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	×		×		×		×		×		×		×		×		×		×		...
	p2		×	×			×	×			×	×			×	×			×	×		
	p4				×	×	×	×					×	×	×	×					×	
	p8								×	×	×	×	×	×	×	×						
	p16																×	×	×	×	×	

Shown are only 20 encoded bits (5 parity, 15 data) but the pattern continues indefinitely. The key thing about Hamming Codes that can be seen from visual inspection is that any given bit is included in a unique set of parity bits. To check for errors, check all of the parity bits. The pattern of errors, called the error syndrome, identifies the bit in error. If all parity bits are correct, there is no error. Otherwise, the sum of the positions of the erroneous parity bits identifies the erroneous bit. For example, if the parity bits in positions 1, 2 and 8 indicate an error, then bit 1+2+8=11 is in error. If only one parity bit indicates an error, the parity bit itself is in error.

As you can see, if you have m parity bits, it can cover bits from 1 up to $2^m - 1$. If we subtract out the parity bits, we are left with $2^m - m - 1$ bits we can use for the data. As m varies, we get all the possible Hamming codes :

Parity bits	Total bits	Data bits	Name	Rate
2	3	1	Hamming(3,1) (Triple repetition code)	$\frac{1}{3} \approx 0.333$
3	7	4	Hamming(7,4)	$\frac{4}{7} \approx 0.571$
4	15	11	Hamming(15,11)	$\frac{11}{5} \approx 0.733$
5	31	26	Hamming(31,26)	$\frac{26}{31} \approx 0.839$
...				
m	$2^m - 1$	$2^m - m - 1$	Hamming($2^m - 1$,$2^m - m - 1$)	$\frac{2^m - m - 1}{2^m - 1}$

4. http://en.wikipedia.org/wiki/Hamming_code