TP 5 : Codes détecteurs et correcteurs d'erreurs

RAPPEL : Code de Hamming $(7,4)$

Le code de Hamming $(7,4)$ [1] consiste à insérer de la redondance tous les 4 bits de sorte à détecter et corriger 1 erreur. Ainsi, une suite de 4 bits devient une suite de 7 bits avec des relations pour pouvoir détecter puis corriger. Si au plus l'un des 7 bits est erronés au cours de la transmission (un 1 devenu 0 ou inversement), alors l'algorithme le détecte et peut retrouver le message initial. Si plus d'erreurs se sont introduites, cette méthode ne permet pas de les détecter, ni de les corriger.

Plus précisément, si le mot binaire à transmettre est $b_1b_2b_3b_4$, il est alors encodé en $c_1c_2c_3c_4c_5c_6c_7$ de sorte que :

$$\begin{cases} c_1 = b_1 \\ c_2 = b_2 \\ c_3 = b_3 \\ c_4 = b_4 \\ c_5 = b_1 + b_2 + b_3 \mod 2 \\ c_6 = b_1 + b_2 + b_4 \mod 2 \\ c_7 = b_2 + b_3 + b_4 \mod 2 \end{cases} \tag{1}$$

Lorsque $c_1c_2c_3c_4c_5c_6c_7$ est reçu, il s'agit d'abord de vérifier s'il contient une erreur. Pour cela, chaque égalité est testée : est-ce que $c_5 = c_1+c_2+c_3 \mod 2$, $c_6 = c_1+c_2+c_4 \mod 2$ et $c_7 = c_2+c_3+c_4$? Si oui, alors le message ne comporte pas l'erreur. Si non, en admettant qu'une seule erreur s'est produite, il suffit d'analyser l'impact :

- si c_1 est erroné, alors les égalités pour c_5 et c_6 ne sont pas vérifiées ;
- si c_2 est erroné, alors les égalités pour c_5, c_6 et c_7 ne sont pas vérifiées ;
- si c_3 est erroné, alors les égalités pour c_5 et c_7 ne sont pas vérifiées ;
- si c_4 est erroné, alors les égalités pour c_6 et c_7 ne sont pas vérifiées ;
- si c_5 est erroné, alors l'égalité c_5 n'est pas vérifiée ;
- si c_6 est erroné, alors l'égalité c_6 n'est pas vérifiée ;
- si c_7 est erroné, alors l'égalité c_7 n'est pas vérifiée ;

Ainsi, il y a trois calculs à faire sur une suite de 7 bits pour savoir s'il y a une erreur et les égalités non vérifiées indiquent le bit erroné. Il est alors possible de corriger l'erreur retirer les bits de redondance pour retrouver la suite de 4 bits. À noter qu'une erreur sur les bits c_5, c_6 ou c_7 n'a pas d'impact sur le décodage.

1. https://web.archive.org/web/20060525060427/http://www.caip.rutgers.edu/~bushnell/dsdwebsite/hamming.pdf

Exercice 1 (Décodage ASCII).
Sur un champ de bataille, le message suivant est envoyé aux agents spéciaux pour confirmer ce qu'ils doivent faire : chaque caractère appartient à la table ASCII et est encodé sur 8 bits. De plus, les nombres sont écrits en binaire.

```
01000011011011110110110101101101100101011011100110011
01100101011110100010000001110000011000010111001000100000
01100011011000010111001101110011011001010111001000100000
01101100011001010111001100100000011000110110111101110100
01100101011100110010000001100100011001010010000001100011
01100101001000000110100001100001011000110110101101100101
01110010001000000111000001101111011101010111001000100000
01101111011000100111010001100101011011100110100101110010
00100000011001000110010101110011001000000110100101101110
01100110011011110111001001101101011000010111010001101001
01101111011011100111001100101110
```

Écrivez une fonction renvoyant la chaîne de caractères dont est issue une chaîne de bits donnée en paramètre. Vous aurez peut-être l'usage des fonctions `int(,2)` et `chr()` de `Python`.

Exercice 2 (Décodage Hamming).
En pratique, il est possible que des erreurs de transmission se produisent, ce qui introduit des erreurs dans le message reçu. Pour pallier cela, il existe des codes détecteurs et correcteurs d'erreurs. Le message ci-dessus n'est en réalité qu'une partie du message transmis sur le réseau. Le message complet ci-dessous applique le code de Hamming $(7, 4)$ tous les 4 bits afin de pouvoir détecter et corriger 1 erreur.

```
01001110011110011001011111111011001011010100110010110101 0
01100100101100011001011101000110010001111001100100101100
01110011010011001010100000000111001000000001100100001011
01110010010101001010100000000110010001111001100100001011
01110010011110011100100111100110010010110001110010010101
00101010000000011001011000010110010010110001110010011110
00101010000000011001000111100110010111111101110100100111
01100100101100011100100111100010101000000001100100100111
01100100101100001010100000000110010001111001100100101100
00101010000000011001010001100110010000101101100100011110
01100101011000011001001011000111001001010100101010000000
01110010000000011001011111111011100101011000111001001010 1
00101010000000011001011111111011001000101010111001010011 1
01100100101100011001011101000110010100110101110010010101
00101010000000011001001001111011001001011000111001001111 0
00101010000000011001010011010110010111010001100100110010
01100101111111011100100101010110010110101001100100001011
```

`01110010100111011001010011010110010111111101100101110100`
`0111001001111000101011110100`

1. Écrivez une fonction qui renvoie la chaîne de bits constituée des bits d'information (sans correction) d'une chaîne de bits reçue en paramètre, en supposant que cette dernière est encodée avec le code de Hamming $(7, 4)$.
2. Écrivez une fonction qui vérifie si une chaîne de bits (encodée avec le code de Hamming $(7, 4)$) reçue en paramètre contient des erreurs, les corrige et renvoie la chaîne de bits obtenue après correction.

Exercice 3 (Encodage ASCII).
Écrivez une fonction qui renvoie la chaîne de bits obtenue en collant bout à bout les codes ASCII des caractères composant une chaîne de caractères donnée en paramètre. Vous aurez peut-être l'usage des fonctions `bin()` et `ord()` de `Python`. Attention au nombre de bits des écritures binaires !

Exercice 4 (Encodage Hamming).
Écrivez une fonction qui effectue l'encodage d'une suite de bits passée en paramètre à l'aide du code de Hamming $(7, 4)$, et renvoie la suite de bits obtenue.

Exercice 5 (Insertion d'erreurs aléatoires).
Écrivez une fonction qui, sur la donnée d'une chaîne de bits et d'un entier positif `nb` passés en paramètres, renvoie une chaîne de bits comportant `nb` bits modifiés à des positions aléatoires.
Sur une chaîne de bits obtenue à l'aide du code de Hamming $(7, 4)$, les erreurs introduites ne sont pas toujours bien corrigées (faites plusieurs essais). Pourquoi ?