Protocoles réseaux
TD n° 1 : Codes détecteurs et correcteurs

Un **code** est une fonction C qui a tout mot de m bits associe un mot de n bits, $n > m$. On note $r = n - m$ la **redondance** du code. Pour qu'un code soit utile, il faut que deux mots différents soient toujours codés différement. Le **dictionnaire** du code est l'ensemble des 2^m mots de n bits codés.

Lors du décodage,
- Si le mot y de n bits reçu est dans le dictionnaire, alors on décode avec l'unique x à m bits tel que $C(x) = y$;
- Sinon, le code a **détecté** une erreur.

Si le mot envoyé est y et le mot reçu est y', le nombre d'**erreurs sur un bit** qui ont lieu est le nombre de positions où les bits de y et y' différent. Par exemple, 10110 et 00111 diffèrent en 2 positions.

Un code **détecte** k **erreurs** si, pour tout y' reçu avec au moins une erreur et au plus k par-rapport au y envoyé, le code signale une erreur.

Exercice 1 : bit de parité

Les anciens modems RTC utilisaient un système de contrôle d'erreurs basé sur le bit de parité : à la fin de chaque octet, on rajoute un 9e bit de sorte que le nombre de 1 dans l'octet soit pair.

1. L'utiliser pour coder 01100100 puis 0111010001111010
2. Donner l'algorithme de détection d'erreur.
3. Combien d'erreurs sur un bit ce code détecte-t-il ? Donner un exemple d'erreur non détectée.

Un code **corrige** k **erreurs** si, pour tout mot y' reçu avec au moins une erreur et au plus k par-rapport au mot y envoyé, y est le seul mot du dictionnaire tel que l'on peut obtenir y' en changeant au plus k bits dans un mot du dictionnaire. y' et k étant connus, le décodeur peut retrouver l'unique y et le produire, corrigeant ainsi l'erreur.

Exercice 2 : détection versus correction

1. Construire un code qui détecte 1 erreur. L'étendre pour détecter k erreurs (k donné).
2. Expliquez pourquoi tout code qui corrige k erreurs permet de détecter au moins k erreurs.
3. Montrez qu'un code qui détecte une erreur peut ne pas être capable de corriger une erreur.
4. On considère le code où chaque bit est répété $2k + 1$ fois. Expliquez comment détecter et corriger les erreurs avec ce code. Combien d'erreurs ce code peut-il détecter ? Combien d'erreurs peut-il corriger ?

Exercice 3 : matrice de parité

Une variante du bit de parité est la *matrice de parité*. On écrit 8 octets de données sur une matrice 8x8, puis on rajoute une 9e ligne et une 9ème colonne. Le bit de la ie colonne (resp. ligne) est choisi pour que le nombre de 1 dans cette colonne (resp. ligne) soit pair.

1. Codez la matrice ci-contre
2. Que vaut le bit en position (9,9) ?
3. Justifiez que ce code corrige une erreur.
4. Donnez un exemple de deux erreurs non corrigeable.
5. Combien d'erreurs ce code détecte-t-il ?
6. Donnez un exemple d'erreurs non détectées.
7. Expliquez comment étendre ce code à un message de taille $n \times n$. Comparez l'efficacité de ce code par rapport à celui de l'exercice 2.

1	0	0	0	1	1	0	1	
0	1	1	1	0	0	1	0	
1	0	0	0	0	0	0	1	
0	1	1	0	1	1	0	1	
0	1	0	0	0	0	0	0	
1	0	0	0	1	1	0	1	
0	1	0	1	0	1	0	0	
1	1	0	0	0	0	0	1	

Exercice 4 : complément à un de la somme de contrôle

Un entête IPv4 (couche 3) est émis accompagné d'une *somme de contrôle*. Le message est représenté par une suite $a_1, \ldots, a_n$ d'entiers codés sur 8 bits. On calcule d'abord la somme de contrôle $c = (a_1 + a_2 + \cdots + a_b) \mod 2^8 - 1$ puis on calcule l'entier b de 8 bits qui est le **complément à un** de c, c'est à dire qu'il doit vérifier $b + c = 0 \mod 2^8 - 1$. On envoie ensuite le message $a_1, \ldots, a_n, b$.

Lorsque l'on reçoit un message $m_0, m_1, \ldots, m_{n+1}$, on calcule $c' = (m_1 + \cdots + m_{n+1}) \mod 2^8 - 1$ et on vérifie que $c' = 0$: si ce n'est pas le cas c'est qu'il y a une erreur.

1. Codez le message $0, 0, 0$ et le message $42, 100, 200, 3$.

2. Décodez les messages de la question précédente et vérifiez que l'algorithme produit le bon résultat. Justifiez pourquoi cela marche dans le cas général.

3. Que se passe-t-il si une erreur pendant la transmission modifie un seul des a_i, par exemple a_0 ? *Indication : reprenez l'exemple du message* $0, 0, 0$ *en modifiant* a_0 *en* 160*, puis en le modifiant par* 255.

4. Montrez que le code ne peut pas détecter un échange entre a_0 et a_1. Déduisez-en un exemple d'erreur non détectée en modifiant seulement deux bits dans le message.