IUT Littoral **2019 - 2020**
Info 2A **Math – C++**

TP 6 - Codes correcteurs

Les codes correcteurs d'erreurs ont leur source dans un problème très concret lié à la transmission de données. Dans la grande majorité des cas, la transmission de données se fait en utilisant une voyer de communication, la canal de communication, qui n'est pas entièrement fiable. Autrement, les données, lorsqu'elles circulent sur cette voie, sont susceptibles d'être altérée.

Exemple. de canal de communications.
- onde radio (FM, Wifi, Bluetooth, ...)
- fils conducteur (ADSL, Ethernet, ...)
- onde sonore (parole, ...)

Le même phénomène se produit lorsque l'on stocke de l'information sur un support. L'information lue peut être plus ou moins différentes que celles enregistrées.

Exemple. Mémoire vive, disque dur, CDROM, ...

Reprenons l'exemple de communication radio. La présence de parasites sur la ligne va perturber le son de la vois. Il y'a essentiellement deux approches possibles.
- augmenter la puissance de l'émission
- ajouter de la redondance à l'information

Augmenter la puissance a ses limites, pour des raisons diverses : consommation énergétique, nuisance, coût de l'émetteur, ... L'autre solution consiste à ajouter des données, ce qui donne lieu au code des aviateurs qui diront "Alpha Tango Charlie" dans le but de transmettre "ATC" à travers le radio. La séquence " Alpha Tango Charlie ", même avec friture sera plus reconnaissable pour l'oreille humaine qu'un "ATC' déformé.

Un code correcteur peut avoir plusieurs buts de manière non exclusive.
- détecter les erreurs
- corriger les erreurs

Dans le protocole de communication TCP seul la détection est assurée. En effet, la correction est réalisée par une nouvelle demande de transmission du message.
Dans d'autres situation on peut pas faire de nouvelle demande de transmission, c'est le cas notamment pour le stockage de données. Par exemples les codes correcteurs utilisés avec les CD sont conçus pour corrigée jusqu'à 4096 bits consécutifs, ce qui correspond à une rayure de plus d'un millimètre de large.
Les objectifs à atteindre ne sont donc pas du tout les mêmes en fonction de l'utilisation de l'information et du çanal de communication.

Notre modèle est le suivant :
- on considère que le message est une suite de bits
- regroupés par bloc de k bits ($k = 1$, ou 4, ou 8 ...)
- et que chaque bits à une probabilité non nulle d'être inversé.

Pour pouvoir corriger une éventuelle erreur dans un bloc de bits ont doit nécessairement ajouter une information supplémentaire.

Exemple. Code par adjonction d'un bit de parité $(8, 9)$. On découpe notre message initial en bloc de 8 bits. On transforme ensuite chaque bloc en un bloc de 9 bits en ajoutant un bit à la fin de chaque bloc de telle sorte que la somme des bits du nouveaux blocs soit toujours paire.

Exercice 1. Coder les blocs 01101011 et 00110101. Que ce passe-t-il si un bit est modifié ? et dans le cas de deux bits.

Si une erreur des produit, on peut la détecter, mais pas la localiser : on ne peut pas corriger notre bloc, il faut recommencer la transmission. Si d'avantage d'erreurs de produisent, on n'est même pas sûr de détecter le problème.

Qu'attend-on d'un bon code ?
- l'information ne doit pas être trop diluée,
- on doit pouvoir détecter et corriger un nombre raisonnable d'erreurs,
- l'algorithme de codage doit être rapide,
- l'algorithme de décodage aussi.

1 Paramètre d'un code

Un bloc de k bits sera indifféremment appelé bloc, mot ou vecteur. L'ensemble des mots de k bits sera noté $\{0,1\}^k$. On parlera indifféremment de bits ou de lettres.

Un mot de k bits sera noté $b_1b_2...b_k$ ou éventuellement $\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}$.

Définition 2. . Un code correcteur de paramètre (k,n) est application injective $\phi : \{0,1\}^k \to \{0,1\}^n$ appelé *encodage*. Le paramètre k est la *dimension* du code et n sa *longueur*.

Question 3. Pourquoi l'encodage doit-il être injectif ? Existe t-il un lien entre k et n ?

Reponse. Si le codage n'était pas injectif alors ils existeraient deux mots m et m' tels que $\phi(m) = \phi(m') = c$. Lors de la réception après transmission sans erreurs du mot c on ne saurait pas dire s'il est le codé de m ou de m'. Comme ϕ est injective on a que la cardinalité de $\{0,1\}^k$ est la même que $\phi(\{0,1\}^k)$ à savoir 2^k il doit donc y avoir au moins 2^k élément dans $\{0,1\}^n$ qui est de cardinalité 2^n, ce qui implique $k \leqslant n$.
Dans $\{0,1\} = \mathbb{Z}/2\mathbb{Z} = \mathbb{F}_2$, on a $1 + 1 = 0$.

Exercice 4. En vous inspirant du code vue précédemment, imaginé un code par adjonction d'un bit de parité de paramètres $(k, k+1)$ pour $k \geqslant 1$.

Définition 5. Soit ϕ un code de paramètre (k,n). L'ensemble $C = \{\phi(m),\ m \in \{0,1\}^k\}$ est appelé *image* du code ϕ. Les éléments de C sont les *mots de code* de ϕ.

Exercice 6. Considérons l'application $\phi : \{0,1\} \to \{0,1\}^3$ définie par $\phi(0) = 000$ et $\phi(1) = 111$. Précisée chacune des notions introduite pour ce code. Ce code sera appelé code de répétition pure $(1,3)$.

Exercice 7. Préciser ce que peuvent devenir les mots 000 et 111 après $0, 1$ et 2 erreurs. Parmi les mots trouvés repérées ceux qui sont des mots de code pour le code de répétition pure $(1,3)$. Combien d'erreurs ce code peut-il détecter ? Corriger ?

Si une, ou même deux erreurs se produisent, le mot reçu n'est pas un mot de code, l'erreur est donc détectée.
Comment corriger ? Si le mot reçu n'est pas un mot de code, la probabilité qu'il se soit produit une erreur est plus importante que celle qui se soit produit deux erreurs. Il est donc plus raisonnable de corriger par le mot de code le plus "proche". On peut alors corriger une erreur mais pas 2.

Définition 8. Soient m et m' deux mots de $\{0,1\}^k$. On appelle *distance de Hamming* entre m et m', et on note $d(m,m')$ le nombre de lettres distinctes de m et m'. On appelle *poids de Hamming* de m et on note $w(n)$ le nombre de lettres non nulles de m.

Exercice 9. Quel est le poids de Hamming de 01100111. Quelle est la distance de Hamming entre

— 0001001 et 0101001,
— 0000110 et 0001100.

Exercice 10. Montrer que pour tout m, m' de $\{0,1\}^k$, on a $d(m,m') = w(m+m')$. En déduire que pour tout m, m' et c de $\{0,1\}^k$ on a $d(m+c, m'+c) = d(m,m')$.

Définition 11. Soit ϕ un code d'image C. On appelle *capacité de détection* de ϕ et on note e_d le plus grand nombre d'erreurs que ϕ permet de détecter quelque soit le message. On appelle *capacité de correction* de ϕ et on note e_c le plus grand nombre d'erreurs que ϕ permet de corriger quelque soit le message. On appelle *distance minimale* de ϕ et on note d_ϕ la plus petite distance de Hamming non nulle entre deux mots de code.

Proposition 12. *On a* $e_d = d_\phi - 1$ *et* $e_c = \left\lfloor \frac{d_\phi - 1}{2} \right\rfloor$.

Exercice 13. Que vaut d_ϕ, e_d et e_c dans le cas du code de répétition pure $(1,3)$.

Exercice 14. Que vaut d_ϕ, e_d et e_c dans le cas du code de bit de parité $(8,9)$.

2 Codes linéaires

Définition 15. Un code ϕ de paramètre (n,k) est dit linéaire s'il existe une matrice $G \in M_{n,k}(\mathbb{F}_2)$ de rang k, telle que $\forall m \in \{0,1\}^k$, $\phi(m) = G \times m$. La matrice G est appelée matrice génératrice du code ϕ.

Exercice 16. Les codes répétition pure $(1,3)$ et bit de parité $(8,9)$ sont-ils linéaire ? Si oui, quelles sont leurs matrices génératrices.

Exercice 17. Etudier le code linéaire ayant

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

comme matrice génératrice. Ce code est de dimension 2 et de longueur 4. On a

$$G \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; G \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \; G \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \text{ et } G \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

On a donc $C = \{0000, 1011, 0101, 1110\}$. De

$$d(0000,1011) = 3, d(0000,0101) = 2, d(0000,1110) = 3,$$
$$d(1011,0101) = 3, d(1011,1110) = 2 \text{ et } d(0101,1110) = 3,$$

on obtient $d_\phi = 2$ et donc $e_d = 1$ et $e_c = 0$.

Exercice 18. Proposer une structure `C++` permettant de représenter un code linéaire.

Proposition 19. *Soit ϕ un code linéaire de paramètre (n, k) alors son image C est un sous-espace vectoriel de $\{0,1\}^n$.*

Exercice 20. Montrons que C est un sous-espace vectoriel de $\{0,1\}^n$ qui est un $\mathbb{F}_{\not=}$-espace vectoriel. Soit G la matrice génératirece de ϕ. Tout d'abord C est non vide car $0 = G \times 0$. Soient c et c' deux élément de C, alors il existent m et m' tels que $c = G \times m$ et $c' = G \times m'$. On a $c + c' = G \times (m + m')$ et donc $c + c'$ appartient à C. Les seuls éléments de $\{0,1\} = \mathbb{F}_2$ sont 0 et 1. Multiplié un vecteur u de C par 0 donne 0 qui est dans C et le multiplié par 1 donne u qui est aussi dans C. L'ensemble C est donc un sous-espace vectoriel de $\{0,1\}^n$.

Proposition 21. *Soit ϕ un code linéaire d'image C. Alors la distance minimale d_ϕ de ϕ est égale au plus petit poids non nul d'un mot de C.*

Démonstration. Notons p le plus petit poids non nul d'un mot de C. La distance minimale est la plus petite distance de Hamming entre deux mots de code. Soit c et c' dans C tel que $d_\phi = d(c, c')$. Comme ϕ est linéaire $c' - c$ est encore un mot de code. On a donc $d_\phi = d(0, c - c') = w(c - c')$. Ainsi $d_\phi \geqslant p$. Soit m un mot de C tel que $w(m) = p$. Alors $d(0, m) = p$. Les mots 0 et m étant des mots de code, on a $d_\phi \leqslant d(0, m) = p$. On a donc montrer $d_\phi = p$. □

Proposition 22. *Soit ϕ un code linéaire de paramètre (k, n) et d'image C. Alors on a $d_\phi \leqslant n - k + 1$. Un code pour lequel on a égalité est dit MDS (Maximum Distance Separable).*

Démonstration. D'après la proposition précédente, il est suffisant de montrer qu'il existe un mot de code de poids inférieur ou égale $n + 1 - k$. Un mot de $\{0,1\}^n$ dont les $k - 1$ dernière composante sont nulles à un poids inférieur ou égale à $n - k + 1$. Notons D l'espace vectoriel des mots dont les $k - 1$ dernières composantes sont nulles. La dimension de D est le nombre de composantes libres, à savoir $n - k + 1$. Par un résultat générale d'algèbre linéaire on a $n \geqslant \dim(C + D) = \dim(C) + \dim(D) - \dim(C \cap D)$. et donc $n \geqslant k + n - k + 1 - \dim(C \cap D)$, ce qui implique $\dim(C \cap D) > 0$. Il existe donc un mot non nul dans $C \cap D$. Ce qui signifie qu'il existe un mot de code avec ses k dernières composantes nulles. Le poids de ce mot étant inférieur à $n - k + 1$, on a $d_\phi \leqslant n - k + 1$. □

Exercice 23. Donner un minorant sur la longueur d'un code code linéaire de dimension k détectant d erreurs.

Exercice 24. Les codes bits parité $(8, 9)$, répétition pure $(1, 3)$ et celui donnée par matrice génératrice sont-ils MDS ?

Définition 25. Soit ϕ un code linéaire de matrice génératrice G. On appelle *matrice de contrôle* de ϕ toute matrice $H \in M_{n-k,n}(\mathbb{F}_2)$ telle que $H.m = \vec{0} \Leftrightarrow m \in C$.

Définition 26. Un code ϕ de paramètre (k, n) est dit systématique si pour tout $m \in \{0,1\}^k$, le mot m est un préfixe de $\phi(m)$.

Exercice 27. Montrer qu'un code de paramètre (k, n) est systématique si et seulement si sa matrice génératrice est de la forme $\begin{bmatrix} I_k \\ G' \end{bmatrix}$ où G' est une matrix de $M_{n-k,k}(\mathbb{F}_2)$.

Proposition 28. *Soit ϕ un code systématique de paramètre (k, n) et de matrice génératrice $\begin{bmatrix} I_k \\ G' \end{bmatrix}$ alors la matrice $H = \begin{bmatrix} G' & I_{n-k} \end{bmatrix}$ est une matrice de contrôle de ϕ.*

Exercice 29. Démontrer cette proposition.

Exercice 30. Donner la matrice de controle des code bit de parité $(8, 9)$, répétitions pure $(1, 3)$ et de celui donné par matrice génratrice.

Définition 31. Soit ϕ un code de paramètre (k, n), de matrice génératrice G et de contrôle H. On se fixe un mot de source x (de longueur k). Le mot de code correspondant sera $\phi(x) = y$. S'il y a eu des erreurs durant la transmission, on recoit z. On appelle *mot erreur* associé à z, le mot e tel que $z = y + e$. On appelle *syndrome* de z le mot Hz.

On vérifie immédiatement qu'on a $Hz = H(y + e) = He$. Le syndrome ne dépend que de la "maladie" (erreur) et non du "patient" (mot a transmettre).

Définition 32. L'ensemble des syndromes S_z de z est appellé *classe litérale* de z.

Principe de décodage Soit ϕ un code linéaire de paramètre (k, n), corrigeant e_c erreurs et de matrice de contrôle H.

1. On recoit le mot z transmis avec de possibles erreurs.
2. On calcul le syndrome s de z par $s = Hz$.
3. Si s vaut 0, on ne détecte pas d'erreur et on retourne $\phi^{-1}(z)$.
4. Sinon on recherche l'erreur e de plus petit poids possible telle que $He = s$.
5. Si le poid $w(e)$ est inférieur ou égale à e_c, on retourne $\phi^{-1}(z + e)$.
6. Sinon afficher "Impossible de corriger les erreurs".

Il nous reste à voir comment calculer $\phi^{-1}(c)$ pour un mot c de $\mathbb{F}_2^n$ (utiliser au 2 et 5) et comment trouver e (ligne 4). Si ϕ est un code systématique, alors m est le préfixe de longuer k de $\phi(m)$. Dans ce cas $\phi^{-1}(c)$ et le préfixe de longueur k de c.

3 Table de dacodage

Pour trouver e, nous allons construire une *table de décodage*. Soit ϕ un code linéaire de paramètre (k, n). Les matrices de contrôle associé à ϕ sont de taille $(n - k) \times n$. L'ensemble des syndromes possibles est donc $\mathbb{F}_2^{n-k}$.
Pour calculer la table de décodage de ϕ, on liste tous les syndrômes de ϕ. Puis on liste tous les mots z de $\mathbb{F}_2^n$ par poids croissant. Pour chaque z, on calculer Hz. Au syndrome s on associe alors le premier mot z apparu tel que $Hz = s$. On arrete ce procéder dés qu'on associer un mot à chaque syndrome. Le mot associé à un syndrome s est alors l'erreur de poid minimal donnant$s$.

Exercice 33. Calculer la table de décodage pour les codes bits de parité $(8, 9)$, répétition pure $(1, 3)$ et celui donner par matrice génératrice. Pour chacun des codes respectivement, décoder les mots recus 101010100, 101 et 1100.