info602 : Mathématiques pour l'informatique
TD 1 : codes correcteurs d'erreurs

Pierre Hyvernat
Laboratoire de mathématiques de l'université Savoie Mont Blanc
bâtiment Chablais, bureau 17, poste : 94 22
email : `Pierre.Hyvernat@univ-smb.fr`
www : `http://www.lama.univ-smb.fr/~hyvernat/`

Exercice 1 : Divers

Question 1. Programmez :
- la fonction `distance_hamming` qui calcule la distance de Hamming entre 2 mots (donnés comme des entiers 64 bits),
- la fonction `distance_code` qui calcule la distance d'un code (donné comme un tableau d'entiers 64 bits),
- la fonction `distance_code_lineaire` qui calcule la distance d'un code (donné comme un tableau d'entiers 64 bits).

Privilégiez les opérations bit à bit (`^` pour le XOR, `~` pour la négation, `>>` et `<<` pour les décalages, etc.)

Remarque : ces fonctions n'ont que peu d'utilité en pratique car les codes sont toujours donnés par des matrices génératrices (ou équivalents) plutôt que des listes de mots.

Question 2. Donnez tous les mots du code dont une matrice génératrice est

$$G = \left[\begin{array}{ccc|c} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{array}\right]$$

Donnez également la distance du code obtenu.

Question 3. En regardant comment générer tous les mots à partir d'une matrice génératrice, donnez une formule pour le nombre de mots dans un code linéaire lorsque sa matrice génératrice à k lignes et n colonnes.

Question 4. Programmez la fonction `genere_mots` pour générer tous les mots d'un code donné par sa matrice génératrice.

Exercice 2 : codes "3-répétitions"

Question 1. Donnez une matrice génératrice et une matrice de parité pour le code "3-répétitions" sur 3 bits, càd l'ensemble des mots de la forme $b_1b_2b_3b_1b_2b_3b_1b_2b_3$.

Combien de mots ce code contient il ? Quelle est sa distance ? Combien d'erreurs peut-il détecter / corriger ?

Question 2. On suppose qu'un bit unique est codé avec le code "3-répétitions" **sur 1 bit** est envoyé sur un canal binaire symétrique avec probabilité d'erreur $p = 0.1$ (10%) sur chaque bit transmis.

Quelle est la probabilité de transmettre le mot codé (3 bits)
- avec 0 erreur ?
- avec 1 erreur ?
- avec 2 erreurs ?
- avec 3 erreurs ?

Déduisez en la probabilité que le message décodé soit correct ?

Question 3. Même question, mais avec un code “5-répétitions”.

Exercice 3 : code de Hamming

Le code de Hamming permet de coder 4 bits de données “*dddd*” en ajoutant 3 bits de parité : “*ddd_d__*”.

On calcule les bits de parité sur ce mot de 7 bits de la manière suivante :

- le bit de parité de droite (bit numéro $1 = 2^0$) contient la parité des bits dont le numéro en binaire contient 2^0 (c'est à dire les bits 1,3,5 et 7) ;
- le bit de parité numéro $2 = 2^1$ contient la parité des bits dont le numéro en binaire contient un 2^1 (c'est à dire les bits 2,3,6 et 7) ;
- le bit de parité numéro $4 = 2^2$ contient la parité des bits dont le numéro en binaire contient un 2^2 (c'est à dire les bits 4,5,6 et 7).

Question 1. Codez les mots 1101 et 0100.

Question 2. Donnez une matrice génératrice de ce code.

Exercice 4 : code de Hamming, variante

Une matrice génératrice (en forme canonique) pour un code équivalent au code de Hamming est

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

Question 1. Combien de mots ce code possède-t'il ? Donnez la liste de ces mots.

Question 2. On veut coder la suite 01101010110. Comment procède-t'on ?

Question 3. Quelle est la distance de ce code ? Combien d'erreurs peut-on détecter ? Corriger ?

Question 4. On suppose que lors de la transmission, au plus 1 erreur a été commise. Pouvez-vous corriger les messages suivants :

- 1101111
- 0011111
- 0101010
- 1101011
- 0110110

Question 5. On suppose que lors de la transmission, au plus 2 erreurs ont été commises. Pouvez-vous corriger les messages suivants :

- 1101111
- 1111011
- 0000111

Question 6. Donnez la matrice de parité correspondant à ce code.

Question 7. En utilisant la multiplication des matrices appropriées, décidez si les mots suivants sont dans le code :

- 0011001
- 1101001
- 1110100
- 1001011

Question 8. On suppose qu'un message codé est envoyé sur un canal binaire symétrique avec probabilité d'erreur $p = 0.01$ (1%) sur chaque bit transmis.

- Quelle est la probabilité qu'un *mot non codé* (4 bits donc) soit incorrect si on l'envoie directement ?
- Quelle est la probabilité qu'un *mot* corrigé ne corresponde pas au mot envoyé ?

Exercice 5 : codes de Reed-Solomon, opérations scalaires

Nous allons regarder les codes correcteurs utilisés dans le QR codes dans une version plus simple : au lieu de considérer des octets (entre `0x00` et `0xff`), nous allons considérer des demi-octets (entre `0x0` et `0xf`).

Les opérations d'addition et de multiplication sont notées $\oplus$ et $\otimes$. L'addition $\oplus$ est simplement le XOR...

Question 1. Combien valent

- `0x3` $\oplus$ `0x3`
- `0x3` $\oplus$ `0x7`
- `0x3` $\oplus$ `0x4`
- `0x5` $\oplus$ `0xf`

Question 2. Il y a plusieurs manières de calculer (et programmer) la multiplication $\otimes$

- en considérant les demi-octets comme des polynômes à coefficients booléens, et en effectuant la multiplication *modulo* $X^4 \oplus X \oplus 1$,
- en codant en dur la table de multiplication,
- en utilisant le *logarithme*.

Quelle taille en mémoire fera la table de multiplication pour les demi-octets et pour les octets ?

Question 3. Le demi octet `0x5` (`0101` en binaire) représente le polynome $0X^3 \oplus 1X^2 \oplus 0X^1 \oplus 1X^0$, c'est à dire $X^2 \oplus 1$.

- calculez le polynôme obtenu en multipliant $X^2 \oplus 1$ par le polynome correspondant à `0x6`,
- divisez le résultat obtenu (division euclidienne) par $X^4 \oplus X \oplus 1$,
- déduisez-en le résultat de `0x5` $\otimes$ `0x6`.

Question 4. Pour multiplier un demi-octet par 2, càd X, on multiplie par X et "soustrait" (avec $\oplus$) le polynôme $X^4 \oplus X \oplus 1$ si nécessaire. En supposant que les demi octets sont codés dans les 4 bits de poids faibles d'un `unsigned char`, écrivez la fonction `double`.

Question 5. Le tableau suivant contient les puissances successive `0x2`^{n} pour n variant de 0 à 15. Complétez le.

```
          0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
EXP = [ 0x1 0x2 ___ ___ ___ ___ 0xc 0xb 0x5 ___ 0x7 0xe 0xf 0xd 0x9 ___ ]
```

Question 6. Donnez le tableau `LOG` des logarithmes correspondants et écrivez la fonction `init_EXP_LOG` qui permet de remplir les tableaux (variables globales) `EXP` et `LOG`.

Question 7. En utilisant les tables de logarithmes / puissances, calculez

- `0x5` $\otimes$ `0x6`
- `0x2` $\otimes$ `0x7`
- `0x9` $\otimes$ `0x3`

Quel est l'inverse de `0x9` ?

Question 8. Programmez les fonctions `mult`, `invert` et `divide` sur les demi octets en utilisant les tableaux `EXP` et `LOG`.

Question 9. Parmi les 3 méthodes pour calculer la multiplication, laquelle vous parait-elle la meilleure ?

Question 10. Les messages sont transformés en polynômes, codés en les multipliant par

$$(X \oplus 2^0) \times \cdots \times (X \oplus 2^{t-1})$$

puis envoyés. Le destinataire vérifie que le message reçu est correct en vérifiant que le polynôme correspondant s'annule bien sur les valeurs $2^0, \ldots, 2^{t-1}$.

En utilisant les fonctions de multiplication (`mult`) et d'addition (`^`), écrivez une fonction

```
eval_poly(unsigned char *P, size_t d, unsigned char w)
```

qui évalue le polynôme `P` de degré `d` sur le demi octet `w`.

Essayez de limiter le nombres de multiplications (`mult`) et d'additions (`^`) à un nombre *linéaire* d'opérations.

Question 11. Pensez vous pouvoir évaluer un polynôme de degré `d` avec uniquement `d` additions et `d` multiplications ?