

Architecture des ordinateurs - TD 03

1 Position du bit à 1 le plus à gauche

1. Écrire une fonction C qui retourne la position du bit à 1 le plus à gauche dans un mot de 32 bits. Utiliser une boucle et l'opérateur `>>`. Quel est le nombre d'itérations maximum effectués ?

Solution:

```
char premier_bit_a_un(uint32_t M) {
    char pos = 0;
    while(M) {
        M = M >> 1;
        pos++;
    }
    return pos;
}
```

2. Nous souhaitons accélérer l'algorithme en utilisant une méthode de recherche dichotomique, dont l'algorithme est donné ci-dessous :
 - (a) Soit un mot M composé de n bits. On coupe le mot en deux parties : M_g qui contient les $n/2$ bits les plus à gauche et M_d qui contient les $n/2$ bits les plus à droite.
 - (b) Si $M_g = 0$ alors le bit à 1 le plus à gauche est contenu dans M_d et sa position est comprise entre 0 et $n/2$. On recommence ce processus en remplaçant M par M_d .
 - (c) Si $M_g > 0$ alors le bit à 1 le plus à gauche est contenu dans M_g et sa position est comprise entre $n/2$ et n . On recommence ce processus en remplaçant M par M_g .

Écrire cet algorithme en C. Pour sélectionner la partie gauche d'un mot de 32 bits vous pouvez utiliser l'opération `M & 0xFFFF0000` par exemple.

Solution:

```
uint32_t masque[5] = {0xFFFF0000, 0xFF00, 0xF0, 0b1100, 0b10};
uint32_t shift[5] = {16, 8, 4, 2, 1};

char premier_bit_a_un_dichotomique(uint32_t M) {
    char pos = 0;
    char i;

    for (i = 0; i <= 4; i++) {
        if (M & masque[i]) { // le bit a un est dans Mg
            M = M >> shift[i]; // on remplace M par Mg
            pos += shift[i];    // on réajuste la position
                                // (on aurait pu utiliser |=)
        }
    }
}
```

```

    return pos;
}

```

2 Code 2 parmi 5

Un code 2 parmi 5, représente chaque chiffre de 0 à 9 sur 5 bits. Chaque mot de 5 bits a exactement 2 bits à 1 (d'où le nom). Voici un exemple de code 2 parmi 5 :

0	01100
1	11000
2	10100
3	10010
4	01010
5	00110
6	10001
7	01001
8	00101
9	00011

Par la suite, on notera C_n le code pour le chiffre n .

1. Le code 2 parmi 5 peut-être utilisé pour encoder des codes barres. En utilisant la table ci-dessus, lire la valeur du code barre suivant :



Solution: 19889

2. La distance de Hamming $d(M, N)$ entre deux mots M et N représente le nombre de bits différents entre les mots. Ainsi, $d(1001, 1010) = 2$. Calculer $d(C_6, C_9)$? Calculer $d(C_5, C_6)$?

Solution: $d(C_6, C_9) = 2$
 $d(C_5, C_6) = 4$

3. La distance de Hamming d_C d'un code est définie comme la plus petite distance de Hamming possible entre deux mots distincts du code. Soit $d_C = \min\{\forall i \neq j, d_h(C_i, C_j)\}$. Quelle est la distance de Hamming pour le code 2 parmi 5.

Solution: Soit C_i et C_j deux mots du code distincts. On sait que C_i et C_j ont exactement deux bits à 1.

Soit $n_i < m_i$ les positions des bits à 1 de C_i et $n_j < m_j$ les position des bits à 1 de C_j .

Premier cas : $n_i = n_j$ et $m_i = m_j$ impossible car C_i et C_j distincts.

Deuxième cas : un bit à 1 partagé par C_i et C_j et un bit différent. Supposons ici sans perte de généralité $m_i \neq m_j$. Alors la distance de Hamming est 2, il faut pour passer de C_i à C_j éteindre m_i et allumer m_j .

Troisième cas : aucun bit à 1 partagé, la distance de Hamming est 4, il faut pour passer de C_i à C_j éteindre m_i et n_i puis allumer m_j et n_j .

Donc $d_C = \min(2, 4) = 2$.

4. La distance de Hamming d'un code permet de calculer combien d'erreurs peuvent être détectées e_d et combien d'erreurs peuvent être corrigées e_c . Ainsi, $e_d = d_C - 1$ et $e_c = \lfloor \frac{d_C-1}{2} \rfloor$. Combien d'erreurs peuvent être détectées par le code 2/5? Combien d'erreurs peuvent être corrigées?

Solution: $e_d = 2 - 1 = 1$ et $e_c = 0$.

3 Code avec bit de parité

On veut envoyer des mots $(a_3 \dots a_0)$ de 4 bits sur un canal. Le code de parité consiste à ajouter un bit p devant le mot $pa_3 \dots a_0$ qui signale la parité de la somme des bits du mot ($p = a_3 \oplus a_2 \oplus a_1 \oplus a_0$). Si la somme est paire, $p = 0$, sinon $p = 1$.

1. Encoder les mots 1010 et 1000 avec un bit de parité.

Solution: 01010
11000

2. Calculer la distance de Hamming du code avec bit de parité.

Solution:

- Prenons deux mots consécutifs du code : 00000 et 10001, $d_h(00000, 10001) = 2$, donc $d_c \leq 2$.
- Soit deux mots de 4 bits distincts x et y ,
 - Si $d_h(x, y) \geq 2$ alors $d_h(\text{parite}(x), \text{parite}(y)) \geq 2$.
 - Si $d_h(x, y) = 1$ alors $\text{parite}(x) \neq \text{parite}(y)$ donc $d_h(\text{parite}(x), \text{parite}(y)) = 2$
 - Dans tous les cas $d_h \geq 2$ donc $d_c = 2$.

3. Combien d'erreurs peuvent être détectées ? Combien peuvent être corrigées ?

Solution: On peut détecter $2 - 1 = 1$ erreur et on ne peut en corriger aucune.

4 Code à répétition

Le code à répétition 3, consiste à remplacer chaque 0 du mot initial par 000 et chaque 1 par 111.

1. Calculer la distance de Hamming du code à répétition 3.

Solution: Il n'y a que 2 mots dans l'image du code, 000 et 111, la distance est 3.

2. Combien d'erreurs peuvent être détectées ? Combien peuvent être corrigées ?

Solution: $3 - 1 = 2$ erreurs peuvent être détectées. $\frac{2}{2} = 1$ erreur peut être corrigée.

3. Vous recevez les messages suivants : indiquez si des erreurs se sont produites, le cas échéant corrigez les.

111000111

110000011011

Solution: premier message : pas d'erreur. deuxième message : 3 erreurs, 111000111111

4. Vous disposez d'un canal de communications dont la probabilité d'invertir un bit du message est de $p_e = 0.001$ (pour tout n et m distincts la probabilité d'avoir une erreur sur le bit n et une erreur sur le bit m sont considérées indépendantes). Vous utilisez un code à répétition 3 pour envoyer un message original de 100 bits. Quelle est la probabilité pour que le destinataire reçoive le bon message avant correction ? après correction ?

Solution: Soit N_n la loi donnant le nombre d'erreurs pour un message de longueur n . N_n suit une loi binomiale (car les erreurs sont des événements indépendants). Donc $P(N_n = k) = C_n^k \cdot p_e^k \cdot (1 - p_e)^{(n-k)}$.

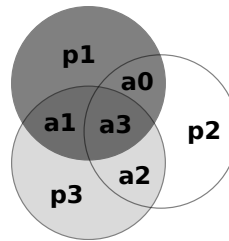
- La probabilité d'avoir un bon message avant correction, c'est $P(N_{300} = 0) = C_{300}^0 \cdot p_e^0 \cdot (1 - p_e)^{300} = (1 - p_e)^{300} > 0.73$
- Soit P_1 la probabilité d'avoir une ou zéro erreur par bloc de trois bits : $P_1 = P(N_3 = 1) + P(N_3 = 0) = 3 \cdot p_e \cdot (1 - p_e)^2 + (1 - p_e)^3$
Donc la probabilité de pouvoir corriger un bloc de trois bits est de : P_1 .
La probabilité de pouvoir corriger le message composé de 100 blocs est de $(P_1)^{100} > 0.9997$

5 Code de Hamming (7,4)

Le code de Hamming (7,4) transmet 4 bits d'informations en utilisant 3 bits de parité. Pour encoder le mot $a_3a_2a_1a_0$ on écrira $p_1p_2a_0p_3a_1a_2a_3$ avec $p_1 = a_0 \oplus a_1 \oplus a_3$, $p_2 = a_0 \oplus a_2 \oplus a_3$ et $p_3 = a_1 \oplus a_2 \oplus a_3$.

On peut résumer la couverture des bits de parité avec la table ou avec le diagramme de Venn suivants :

	a_0	a_1	a_2	a_3
p_1	1	1	0	1
p_2	1	0	1	1
p_3	0	1	1	1



1. En s'appuyant sur le diagramme de Venn, si on change un bit des données (a_0 à a_3) combien de bits de parité changent nécessairement ?

Solution: 2 ou 3 (pour a_3)

2. Quelle est donc la distance de Hamming du code ? Combien d'erreurs peuvent être détectées ? Combien peuvent être corrigées ?

Solution: La distance de Hamming du code est $2 + 1 = 3$. On peut donc détecter 2 erreurs et en corriger 1.

3. Vous recevez les messages suivants, indiquez si une erreur s'est produite, le cas échéant corrigez le mot reçu : 0001111 1110010 1101110 0011111.

Solution: 0001111 Message correct (vérifier que le diagramme de Venn est correct).

1110010 p_3 et p_2 sont incorrects, pour rendre le message correct en changeant un seul bit, il faut changer a_2 . On remarque que $3 + 2 \equiv 2[3]$.

1101110 p_3 est incorrect, erreur sur le bit de parité.

0011111 p_1 et p_2 incorrects, il faut changer a_0 . On remarque que $1 + 2 \equiv 0[3]$.