Compte-Rendu TP3 : Implémentation des codes de Hamming

MACIA Mathys JRIRI Laïla

26 janvier 2026

Table des matières

1 Rappel du sujet

L'objectif de ce TP est de réaliser un programme en langage Java permettant de manipuler les codes de Hamming. Le programme doit remplir trois fonctions principales :

- Calculer et générer un mot de Hamming à partir d'une suite de bits donnée.
- Vérifier la validité d'un mot de Hamming et détecter d'éventuelles erreurs.
- Offrir une interface utilisateur simple en ligne de commande pour effectuer ces opérations.

2 Analyse du sujet

2.1 Structure des mots de Hamming

Un code de Hamming (x, y) est défini par :

- $x = 2^i - 1$: la longueur totale du mot.
- i : le nombre de bits de contrôle (parité).
- $y = x - i$: la longueur du message utile.

Les bits de contrôle sont placés aux positions correspondantes aux puissances de 2 $(1, 2, 4, 8, \dots)$.

2.2 Méthode de calcul et de vérification

Calcul : Pour chaque bit de contrôle à la position 2^k, on calcule la parité des bits dont la position possède le k-ième bit à 1 dans sa décomposition binaire.

Vérification : On recalcule les bits de parité. Si tous les bits de contrôle concordent, le message est valide. Sinon, la somme des indices des bits de parité erronés donne la position du bit corrompu (syndrome).

3 Choix techniques effectués

3.1 Algorithmes

Pour la gestion des positions, nous avons utilisé :

— Des **tableaux d'entiers** (ou chaînes de caractères) pour représenter les suites binaires.
— Une boucle vérifiant si un indice est une puissance de 2 via l'opération logique : $(n\&(n-1)) == 0$.

3.2 Méthodes principales

— `calculerHamming(String message)` : Insère des zéros aux positions 2^k, puis calcule la parité.
— `verifierHamming(String mot)` : Calcule le syndrome pour identifier l'erreur.

4 Résultats et tests

Pour un message d'entrée 1010 (Hamming 7,4) :
— Résultat obtenu : 1010010 (par exemple).

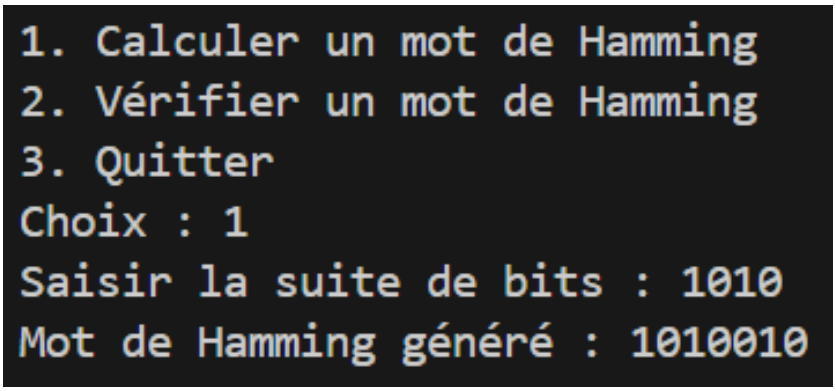

FIGURE 1 – Capture d'écran de l'exécution du calcul d'Hamming

— Test de vérification avec erreur : En modifiant un bit, le programme a correctement identifié l'indice erroné.

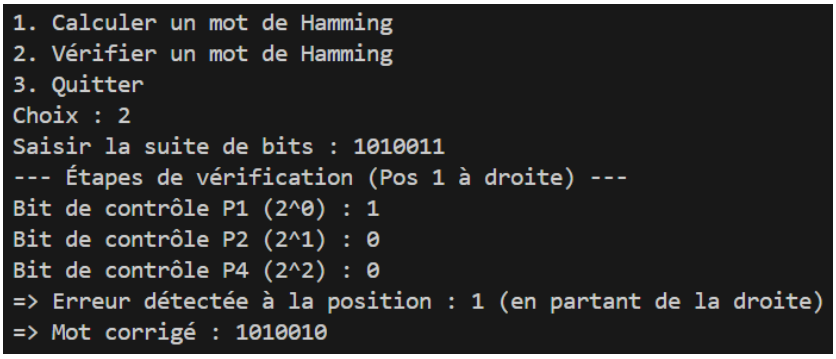

FIGURE 2 – Capture d'écran de l'exécution de la vérification d'Hamming

5 Conclusion

5.1 Difficultés rencontrées

La principale difficulté a été la gestion des indices : les codes de Hamming commencent traditionnellement à l'indice 1, alors que les tableaux Java commencent à l'indice 0. Cela a nécessité une attention particulière lors des calculs de puissances de 2.

5.2 Limites et perspectives

Le programme actuel gère la détection et la localisation d'une erreur simple. Une perspective d'évolution serait d'ajouter un bit de parité global (Hamming étendu) pour détecter les doubles erreurs sans les confondre avec une erreur simple.