

Rapport

TANG Yuke

Code d'étalement

Introduction

Réaliser d'un générateur de la matrice Hadamard. Utiliser la matrice pour réaliser la communication de multi-utilisateur.

Code sources : **hada.c**

Exécutable : **prog_hada**

Fonctions

void matCre(int nb,int mat[][N])

Création d'une matrice Hadamard à partir de nombre de utilisateur (nb) donnée. Résultat enregistré dans (mat).

void etale(int mat[][N],int donnee[][N],int nb,int res[],int nb_donne,int taille_donne)

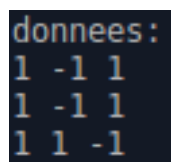
Étalement des données(donne) avec la matrice de Hadamard(mat). Résultat enregistré dans (res). (nb_donne) est le nombre des données. (taille_donne) est la longueur de chaque donné. (nb) le nombre de utilisateur maximale.

void deetale(int nb,int res[],int mat[][N],int nb_donne,int taille_donne)

Désetalement des données codés(res) avec la matrice Hadamard(mat). (nb_donne) est le nombre des données. (taille_donne) est la longueur de chaque donné. (nb) le nombre de utilisateur maximale.

Exécutions

Pour tester on a les 3 chaines de données :



```
donnees:
1 -1 1
1 -1 1
1 1 -1
```

1.Création de matrice Hadamard

Création d'une matrice avec le nombre maximale de utilisateur=16

```
matrice de hadamard :
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1
1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1
1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1
1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1
1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1
1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1
1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1
1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1
1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1
1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1
1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1
1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1
1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1
1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 1 1
```

2.Étalement des données

Faire le étalement avec matrice générée. Utilise les 3 premiers canal pour coder les données. Et faire l'addition des 3 canal.

```
transformation :  
1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1  
1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1  
1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1  
  
addition :  
3 1 1 -1 3 1 1 -1 3 1 1 -1 3 1 1 -1 -1 1 -3 -1 -1 1 -3 -1 -1 1 -3 -1 -1 1 -3 -1 1 -1 3 1 1 -1 3 1 1 -1 3 1
```

3.Desetalement des données

Faire le desétalement avec matrice générée.

```
desetalement:
1 -1 1
1 -1 1
1 1 -1
```

Générateur Pseudo Aléatoire

Introduction

Réalisation d'un codeur à Longueur Maximale. Réalisation d'un codeur de Gold, un codeur JPL. Générer les séquences avec ces codeurs. On lui donne des polynômes et la longueur de séquence.

Code sources : **aleatoire.c**

Exécutable : **prog_alea**

Fonctions

void codeur_lm(int n, poly_t poly, int res[N])

Le codeur à longueur maximale. Générer une séquence de longueur $2^n - 1$ avec le polynôme donné (poly). Le résultat est enregistré dans (res).

Void codeur_gold(int res [N], int n, poly_t poly1, poly_t poly2)

Le codeur de Gold. Générer une séquence de longueur $2^n - 1$ avec les deux polynômes donnés (poly1)(poly2). Le résultat est enregistré dans (res).

Void codeur_jpl(int res[N], int n, poly_t poly1, poly_t poly2, poly_t poly3)

Le codeur de JPL. Générer une séquence de longueur $2^n - 1$ avec les trois polynômes donnés (poly1)(poly2)(poly3). Le résultat est enregistré dans (res).

Exécutions

1.Codeur à longueur maximale

Générer d'une séquence avec le codeur à longueur maximale, nombre de séquence=6, donc le résultat est de longueur $2^6 - 1 = 63$. Le polynôme est [4,2] Résultat :

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

2.Codeur de gold

Générer d'une séquence avec le codeur de gold, nombre de séquence=6, donc le résultat est de longueur $2^6 - 1 = 63$. Le polynôme est [4,2] et [2,1] Résultat :

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

3.Codeur JPL

Générer d'une séquence avec le codeur JPL, nombre de séquence=6, donc le résultat est de longueur $2^6-1=63$. Le polynômes est [4,2] et [2,1] et [4,3,2,1] Résultat :

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

Agent life cycle

Introduction

Le but est de réaliser d'un système de multi-processus avec les deux programmes précédent. Pour réaliser cela, il faut faire la synchronisation entre deux processus.

Fonction

Une fonction pour exécuter les deux processus en même temps et réaliser l'attente et continuation.