GRAU EN ENGINYERIA INFORMÀTICA EN MENCIÓ EN TECNOLOGIES DE LA INFORMACIÓ

Informe de progrés I

Ampliació d'una llibreria en MAGMA per a codis Z2Z4 - lineals

Cristina Diéguez

16/04/2016

Es presenten les modificacions realitzades en el treball de final de grau durant la seva execució. S'exposen els canvis fets a la metodologia i la planificació seguides, però es mantenen els objectius a assolir. Al mateix temps, també es fa una introducció en la matèria i es detalla l'estat actual en el qual es troba actualment la temàtica.

Índex

1. Introducció 2
2. Estat de l'art 5
3. Objectius 6
4. Planificació i metodologia 7
4.1. Metodologia 7
4.2. Planificació temporal 9
5. Bibliografia 13
6. Annex I 15

1. Introducció

Tot procés d'enviament i recepció de missatges es realitza a través d'un canal de comunicació. No obstant, el missatge rebut a la sortida del canal pot haver patit alguna alteració a causa del soroll i fa que aquest es converteixi en erroni. Aleshores, perquè el receptor pugui eliminar possibles errors es realitza el procés de codificació-descodificació.

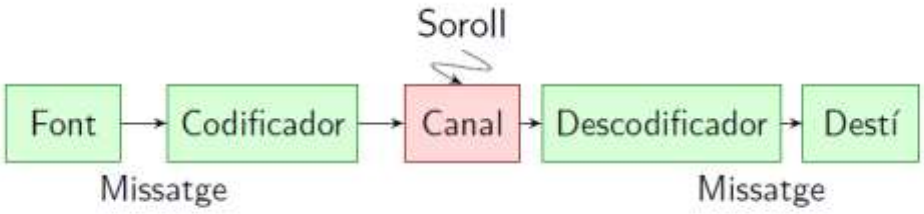

Figura 1. Procés de comunicació

En la codificació, s'assigna a cada símbol o conjunt de símbols de la font una paraula-codi afegint redundància. L'agrupació de totes aquestes paraules-codi formen un codi.

En el procés de descodificació, si s'ha produït algun error en la transmissió i el descodificador detecta l'error, es realitza la correcció aprofitant la redundància afegida. Aleshores el que es vol és poder seguir enviant missatges tenint la capacitat de detecció i correcció d'errors. El segon teorema de Shannon especifica aquest fet ja que declara que és possible transmetre informació a través d'un canal amb soroll si es transmet a una taxa de transmissió de la informació[1] inferior a la capacitat del canal. Aquest teorema dóna peu a la teoria de codificació per a la correcció d'errors que té per objectiu la construcció de codificadors i descodificadors que permetin enviar el màxim d'informació amb el mínim d'errors.

L'escenari ideal seria trobar codis on cadascun dels vectors tingués una única descodificació possible, però trobar codis amb aquestes propietats no és una tasca banal. Una manera de poder minimitzar els errors seria augmentant la redundància en el missatge per poder, després, corregir els errors. El desavantatge que presenta és que existeix una penalització de la velocitat de transmissió.

Dins de l'àmbit informàtic, el tipus de codis que presenten un gran interès per les seves característiques són els codis lineals sobre anells perquè experimenten les següents propietats:

[1] Taxa de transmissió de la informació $R_T=\frac{log2|C|}{n}$

1. Tancament per la suma: dues paraules codi sumades és igual a una altra paraula-codi.
2. Tancament per la multiplicació d'un escalar: si es multiplica una paraula-codi per un escalar, dóna una altra paraula-codi.

Gràcies a aquestes propietats, es pot trobar un conjunt mínim de paraules-codis linealment independents que són capaces de generar totes les paraules del codi. Aleshores, s'observa que un codi pot ser representat de forma matricial per les paraules-codi linealment independents. Es crea el que s'anomena matriu generadora que facilita el treball i la representació d'aquest tipus de codis. Amb aquest fet, es redueixen els costos de memòria i còmput a l'hora de treballar amb codis.

A part d'usar codis binaris que són un tipus de representació utilitzant l'anell d'enters mòdul 2, actualment es treballa amb codis Z_2Z_4-additius, que són codis sobre l'anell $Z_2^{\alpha} \times Z_4^{\beta}$. Aquests codis també posseeixen el concepte de matriu generadora G que està formada per:

- γ vectors d'ordre 2: aquells formats per coordenades a {0,2}.
- δ vectors d'ordre 4: aquells que contenen alguna coordenada a {1,3}.

Coneixent aquesta informació, es determina que el codi C obtingut a partir de la matriu generadora G és d'ordre $2^{\gamma} \cdot 4^{\delta}$. Aquesta expressió, a la vegada, proporciona el número de paraules-codi de C. A l'hora de generar-les, s'opera (x,y)·G on x pertany a Z_2^{γ} i y a Z_4^{δ} ja que si x correspongués a Z_4 es duplicarien les mateixes paraules-codi.

Amb la finalitat de poder treballar a Z_2, existeix una funció anomenada *Gray map* ϕ que proporciona les imatges de les paraules-codi en aquest conjunt; és a dir, realitza una bijecció:

$$\phi: \quad Z_4 \longrightarrow Z_2$$
$$0 \longrightarrow 00$$
$$1 \longrightarrow 01$$
$$2 \longrightarrow 11$$
$$3 \longrightarrow 10$$

Si $v=(v_1,v_2)$ que pertany a $Z_2^{\alpha} \times Z_4^{\beta}$, es defineix $\phi(v)=(v_1, \phi(v_2))$. Aquesta bijecció només afecta, realment, a les β coordenades quaternàries de les paraules-codi mentre que les α coordenades binàries es mantenen iguals. El codi binari ϕ(C) s'anomena Z_2Z_4 – lineal i, de forma general, no és lineal.

La distància de Hamming $d_H\ (u,v)$ entre dues paraules-codi u,v que pertanyen a Z_2 és el nombre de coordenades per les quals difereixen u i v. El pes de Hamming $w_H\ (u)$ és la distància de Hamming entre la paraula-codi u i la paraula-codi formada per un vector de tots 0. En codis quaternaris, que són un tipus de representació utilitzant l'anell d'enters mòdul 4, s'utilitza la mètrica de Lee. El pes de Lee $w_L(v)$ és la suma dels pesos de Lee de les seves coordenades. Els pesos d'aquestes coordenades estan definits de la següent manera:

- $W_L(0)=0$
- $W_L(1)=W_L(3)=1$
- $W_L(2)=2$

La funció Gray ϕ preserva les distàncies, transformant la distància de Lee a distància de Hamming. Gràcies a aquesta funció, s'observa que el pes de Lee d'un vector, v, sobre Z_4 coincideix amb el pes de Hamming de ϕ (v) sobre Z_2. Un exemple d'aquest succés seria:

$\phi(10230) = (0100111000)$

$w_L\ (10230) = w_L(1) + w_L(0) + w_L(2) + w_L(3) + w_L(0) = 4$

$w_H(0100111000)= 4$

La distància mínima d'un codi C és el mínim valor de la distància de totes les parelles de paraules-codi diferents que formen C. Així mateix, el pes mínim d'un codi C és el mínim valor dels pesos de totes les paraules-codi. En aquest projecte, es treballarà amb aquests conceptes a l'hora de plantejar i implementar funcions que ajudin en el procés de descodificació, el qual pot ser descodificació via síndrome o descodificació per mínima distància.

2. Estat de l'art

El grup CCSG és un equip format dins del Departament d'Enginyeria de la Informació i de les Comunicacions, dEIC, i dedicat, parcialment, a la investigació de codis Z_2Z_4- lineals amb la finalitat dur a terme els següents objectius, entre d'altres:

- Construir i caracteritzar nous codis Z_2Z_4- lineals, i computar els seus paràmetres estructurals.
- Caracteritzar i construir famílies de codis Reed-Muller Z_2Z_4- lineals, i computar-ne el rang i la dimensió del kernel.
- Desenvolupar els algorismes necessaris per establir noves opcions en l'ocultació de dades i en l'autentificació de documents, utilitzant codis Z_2Z_4- lineals.
- Expandir el software de MAGMA implementant nous paquets amb la finalitat de treballar eficientment amb codis Z_2Z_4- lineals, codis regulars i codis no lineals.

Un dels principals articles escrits sobre codis Z_2Z_4- lineals és [1] en què s'estudien els codis Z_2Z_4- additius a partir dels quals s'extreu la corresponent imatge binària i s'obtenen els codis Z_2Z_4- lineals. D'aquests nous codis, es detallen les seves propietats i les formes estàndard de les matrius generadores i de control. També esmentar l'article [5] ja que s'hi demostren les propietats necessàries pel càlcul de rangs i kernels dels codis Z_2Z_4- lineals.

Degut que en aquest projecte serà necessari treballar amb les distàncies de Hamming dels codis ja mencionats, una de les referències a destacar és [7] perquè defineix i implementa mètodes algorítmics per la computació de la distància mínima de Hamming per codis Z_2Z_4- lineals.

El llenguatge MAGMA porta incorporat llibreries sobre Z_2 i sobre Z_4 cosa que facilita el treball sobre aquests anells. Gràcies als membres d'aquest grup d'investigació CCSG, des de fa uns anys, es duu a terme el desenvolupament d'una nova llibreria en MAGMA per permetre treballar amb codis Z_2Z_4-lineals, fent ús d'algunes llibreries ja existents en aquest llenguatge. Es pretén que aquesta estigui totalment integrada amb les llibreries esmentades.

Com s'ha esmentat anteriorment, s'usarà el llenguatge MAGMA amb la finalitat de realitzar la part pràctica del projecte. Gràcies a l'estructura Z_2Z_4-lineals es pot treballar amb aquest llenguatge de manera lineal a $Z_2^{\alpha}xZ_4^{\beta}$ i també es poden relacionar aquests codis amb codis sobre Z_2 i sobre Z_4.

Les referències existents sobre l'ús d'aquest llenguatge en l'entorn desitjat són les proporcionades per MAGMA, [11] i [10]. Addicionalment, també es consultaran les guies que facilita el CCSG sobre MAGMA [9], per entendre els paquets de codis Z_2Z_4-lineals [2], i codis Z_2Z_4-additius [3].

El fet d'haver de dissenyar i implementar unes funcions que s'integraran dins de MAGMA implica que s'han de seguir els estàndards marcats per tal mantenir el format. Com a conseqüència, es consultarà la guia [8] ja que proporciona aquests estàndards.

3. Objectius

Els objectius que s'assoliran durant el transcurs del treball a realitzar es detallen a continuació:

- Realització d'un estudi dels codis Z_2Z_4- lineals tant a nivell conceptual com de les propietats que els defineixen.
- Aprenentatge d'un nou llenguatge i entorn de programació anomenat MAGMA.
- Disseny i desenvolupament de tests de prova i tests d'integració de les funcions realitzades prèviament.
- Creació de noves funcions de codis Z_2Z_4- lineals per realitzar els càlcul del pes mínim de Lee, la distància mínima de Lee, el nombre mínim de paraules-codi i la distribució de pesos .
- Ampliació de la llibreria de MAGMA existent sobre teoria de codis Z_2Z_4-lineals amb les noves funcions desenvolupades per tal de facilitar la descodificació via síndrome.

Aquests objectius mencionats anteriorment van ser establerts i especificats en [4] i no es preveu cap modificació al respecte. Es conservaran els existents ja que són amb els quals es treballa pel desenvolupament complet del treball de fi de grau.

4. Planificació i metodologia

4.1. Metodologia

El procés de desenvolupament del projecte segueix un paradigma de programació d'enginyeria de software anomenat *Test-Driven Development (TDD)*, tal i com es va mencionar en [4] de forma més detallada.

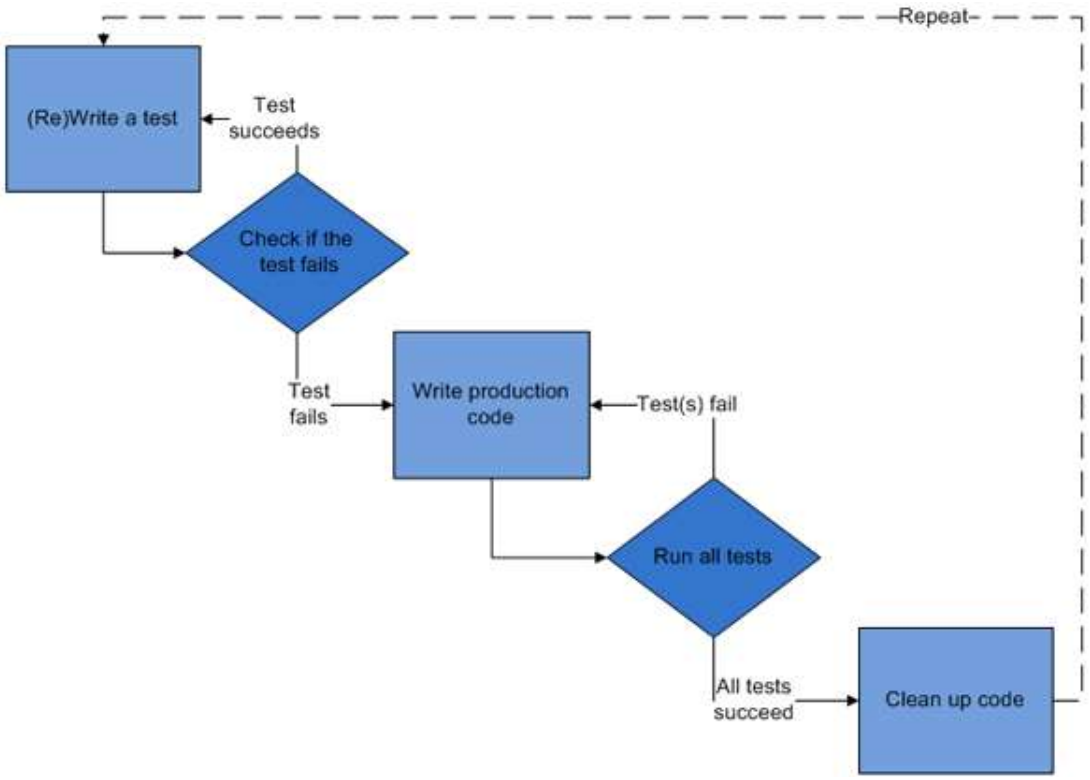

Figura 2: Cicle de vida de *Test-Driven Development*.

Les funcions previstes a desenvolupar durant tota l'execució del projecte són les següents, també esmentades amb anterioritat:

- Z2Z4MinimumLeeWeight
- Z2Z4MinimumLeeDistance
- Z2Z4MinimumWord
- Z2Z4MinimumWords
- WeightDistribution

En aquesta primera etapa, s'han acabat de realitzar unes proves de test proporcionades pel Departament d'Enginyeria de la Informació i de les Comunicacions, dEIC, amb la finalitat de tenir un primer contacte amb el llenguatge MAGMA i aprendre'n la codificació amb petits casos. Addicionalment, també es treballa amb el servidor on està situada la llibreria en desenvolupament de codis Z_2Z_4-lineals. D'aquesta forma, s'aspira a un aprenentatge gradual a mesura que avanci el projecte i, al mateix temps, s'aprèn l'estil de codificació concret que s'ha de seguir per tal de complir l'estàndard amb el qual s'estan desenvolupant els projectes de MAGMA.

Després d'un primer període de treball exclusiu amb les proves de test facilitades, s'ha començat a treballar, de forma paral·lela, amb el disseny de les proves de test per les funcions anteriors. L'estructura d'aquests nous tests és similar amb els ja treballats, cosa que implica realitzar aquests controls amb els següents tipus de codi:

- Trivial Z2Z4-additive zero code
- Trivial Z2Z4-additive universe code
- Repetiton Z2Z4-additive code
- A Z2Z4-additive code with alpha=0
- A Z2Z4-additive code with beta=0
- A Z2Z4-additive code with gamma=0
- A Z2Z4-additive code with delta=0
- A Z2Z4-additive code with kappa=0
- A Z2Z4-additive code with kappa=alpha
- A Z2Z4-additive code with kappa<>gamma

Conseqüentment, per cada tipus de codi, s'ha de seguir una metodologia per realitzar-ne la verificació de les funcions a testejar:

1. Generar el codi amb les característiques específiques.
2. Determinar els valors esperats dels paràmetres a provar.
3. Comprovar que aquests valors previstos coincideixin amb els resultats que genera cada funció testejada.

De manera paral·lela a les tasques anteriors, també s'ha treballat i s'està treballant amb la funció anomenada "WeightDistribution", la qual se li proporcionarà un codi C sobre l'anell $Z_2^{\alpha}xZ_4^{\beta}$ com a paràmetre. Partint d'aquest codi C, es generen totes les paraules-codi i es recorre una per una amb la finalitat de guardar-ne el pes de Lee $w_L()$. Al mateix temps, es

mantindrà un vector v de mida N, on N=α+2β+1, pel qual cada posició v[i] incrementarà en una unitat si el pes de Lee d'una paraula-codi, u, és igual a la posició, i, del vector v.

Per facilitar la comprensió de l'explicació anterior, es formulen les següents sentències més explícitament :

- Es genera una paraula-codi, u, de C.
- Es crea un vector, v, de N posicions.
- Es calcula el pes de Lee de u: $w_L(u)$.
- Si $w_L(u)=i$, aleshores v[i]=v[i] +1.

D'aquesta forma, s'obté tota la distribució de pesos de Lee, $w_L(u)$, d'un codi C sabent la quantitat de paraules-codi que tenen un pes de Lee concret. A partir d'aquesta informació, es podrà obtenir el pes mínim de Lee de tot C i, conseqüentment, el pes mínim de Hamming, $w_H()$, de C sobre Z_2.

El codi d'aquesta funció i de les seves pertinents proves de tests es detalla a l'Annex I per la seva visualització.

4.2. Planificació temporal

La planificació detallada a [4] ha sofert una sèrie de canvis en diverses tasques respecte a la seva durabilitat i la interdependència entre elles per adaptar-les a la situació real en què es troba el desenvolupament del projecte en l'actualitat.

El primer canvi és a la tasca "*Implementació final dels tests existents*" ja que s'ha allargat fins l'11 d'abril de 2016. Aquest canvi ha estat produït per una interrupció en l'execució del treball perquè també es va iniciar la realització de tasques de forma paral·lela. Conseqüentment, s'ha efectuat un desplaçament temporal de tota ella.

La següent tasca afectada és la "*Disseny i implementació dels tests per les funcions pensades*" perquè es va començar el 23 de març de 2016 amb data prevista de finalització el 22 d'abril de 2016 produint una elongació de 2 dies. En aquesta nova versió, també s'ha modificat la dependència amb la tasca predecessora fent que sigui d'inici a inici amb un desfalc de sis dies; habilitant la possibilitat del treball paral·lel amb altres comeses.

Seguint amb l'execució de tasques paral·lelament, l'anomenada "*Disseny i implementació de les funcions (determinades informe inicial)*" també ha patit modificacions ja que la seva

realització va començar el 4 d'abril de 2016 i durarà fins el 13 de maig de 2016. La seva predecessora és la tasca "*Implementació final dels tests existents*" amb dependència d'inici a inici amb un desfalc de disset dies.

D'aquesta forma s'aconsegueix avançar la realització de tasques paulatinament tenint més temps de previsió per la resolució de possibles problemes pràctics a l'hora de dur a terme la implementació de la codificació completa d'aquestes tasques esmentades amb anterioritat.

La tasca "*Creació i lliurament Informe progrés I*" s'ha reduït del 12 al 15 d'abril de 2017 fent que tingui com a predecessores "*Implementació final dels tests existents*", "*Disseny i implementació dels tests per les funcions pensades*" i "*Disseny i implementació de les funcions (determinades informe inicial)*" perquè en l'entrega d'aquest informe s'han de mostrar els avenços efectuats i els canvis de planificació soferts.

S'ha produït una reducció d'un dia de la tasca "*Memòria final*" amb la finalitat de quadrar les dates de la tasca de "*Lliurament final*" amb les establertes per la universitat, deixant un marge de dos dies per poder realitzar els últims retocs a la memòria i a l'article que prèviament s'hauran entregat a la tutora, Cristina Fernández, per l'acompliment de les correccions.

Finalment, també s'han efectuat modificacions de la tasques "*Creació pòster*", "*Lliurament pòster*" i "*Preparació de la presentació*". En la primera, s'han augmentat els dies amb l'objectiu de tenir més temps per la composició del pòster fent que la seva tasca predecessora sigui "*Memòria final*". Com a conseqüència, s'ha reduït l'entrega del pòster al dia 1 de juliol de 2016 fent que els canvis i correccions d'aquest ja estiguin fets amb anterioritat. Degut que la presentació és una part molt important per la valoració del treball realitzat, a la tasca "*Preparació de la presentació*" se li han atorgat tres dies extres fent que el període d'execució vagi del 23 al 30 de juny de 2016.

Tot i les modificacions introduïdes en la planificació, es manté la durabilitat de tot el projecte amb l'objectiu de respectar totes les dates d'entregues parcials així com la data de finalització del mateix.

Aquesta nova versió també s'ha realitzat mitjançant un diagrama de Gantt on s'han especificat els recursos del projecte i, a la vegada, la interdependència de tasques. Juntament amb aquest informe, s'afegeix el diagrama amb el format *pdf* per una millor apreciació de la planificació.

	Nombre	Duración	Inicio	Fin	Predecesoras	Recursos
1	Primera reunió de presentació de la proposta de TFG	1d	05/02/2016	05/02/2016		Cristina Diéguez
2	Reunió: plantejament del projecte i coneixements previs	1d	11/02/2016	12/02/2016	1	Cristina Diéguez
3	Creació i lliurament: Reunió inicial	2d?	12/02/2016	15/02/2016	2	Cristina Diéguez
4	Fonaments teòrics: codis Z4 i codis Z2Z4-lineals	10d	15/02/2016	26/02/2016	2	Cristina Diéguez
5	Creació i lliurament: Informe inicial	4d?	02/03/2016	07/03/2016	4	Cristina Diéguez
6	Implementació final dels tests existents	24.88d?	08/03/2016	11/04/2016	4	Cristina Diéguez
7	Disseny i implementació dels tests per les funcions pensades	22.75d?	23/03/2016	22/04/2016	6II 6d	Cristina Diéguez
8	Disseny i implementació de les funcions (determinades informe i	29.63d?	04/04/2016	13/05/2016	6II 17d	Cristina Diéguez
9	Creació i lliurament: Informe progrés I	3.88d?	12/04/2016	15/04/2016	6,7II,8II	Cristina Diéguez
10	Execució de les proves de test	20d?	25/04/2016	20/05/2016	7	Cristina Diéguez
11	Reconstrucció de les funcions i reexecució de les proves de test	16d?	09/05/2016	30/05/2016	8II,10II	Cristina Diéguez
12	Creació i Lliurament: Informe progrés II	3d?	18/05/2016	20/05/2016	6	Cristina Diéguez
13	Creació article	14d?	23/05/2016	09/06/2016	7,8,10	Cristina Diéguez
14	Lliurament proposta article	1d?	10/06/2016	10/06/2016	13	Cristina Diéguez
15	Memòria final	12.88d?	06/06/2016	22/06/2016	12	
16	Lliurament final	2d?	22/06/2016	24/06/2016	14,15	Cristina Diéguez
17	Creació pòster	6d?	22/06/2016	30/06/2016	15	Cristina Diéguez
18	Lliurament pòster	1d?	01/07/2016	01/07/2016	17	Cristina Diéguez
19	Preparació de la presentació	6d?	23/06/2016	30/06/2016	15	Cristina Diéguez
20	Presentació	4d?	01/07/2016	06/07/2016	19	Cristina Diéguez

Figura 3: Planificació de tasques

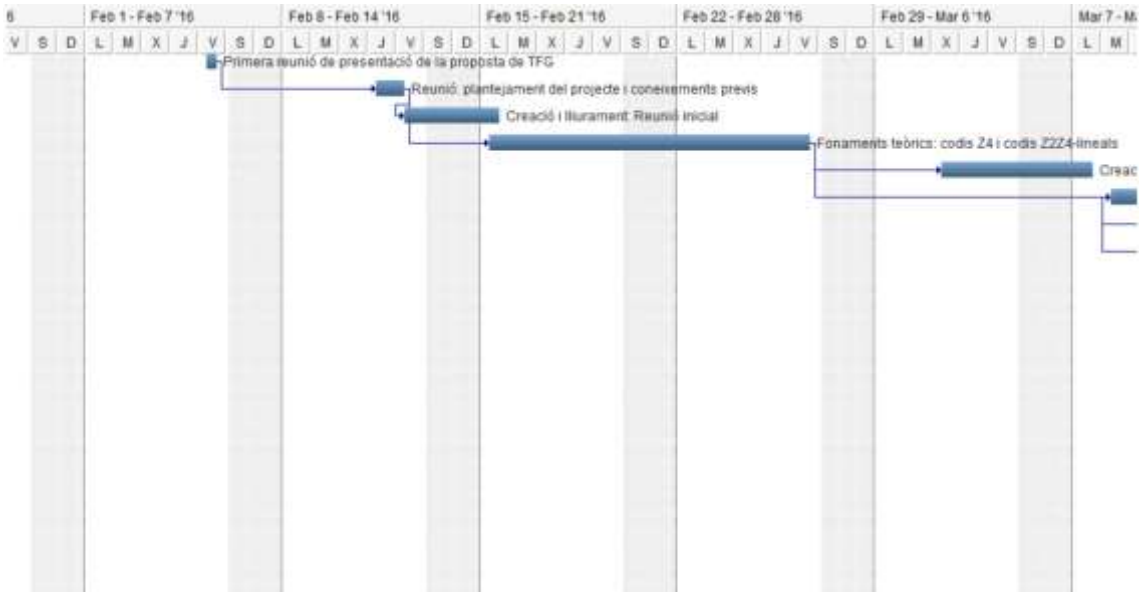

Figura 4: Diagrama temporal

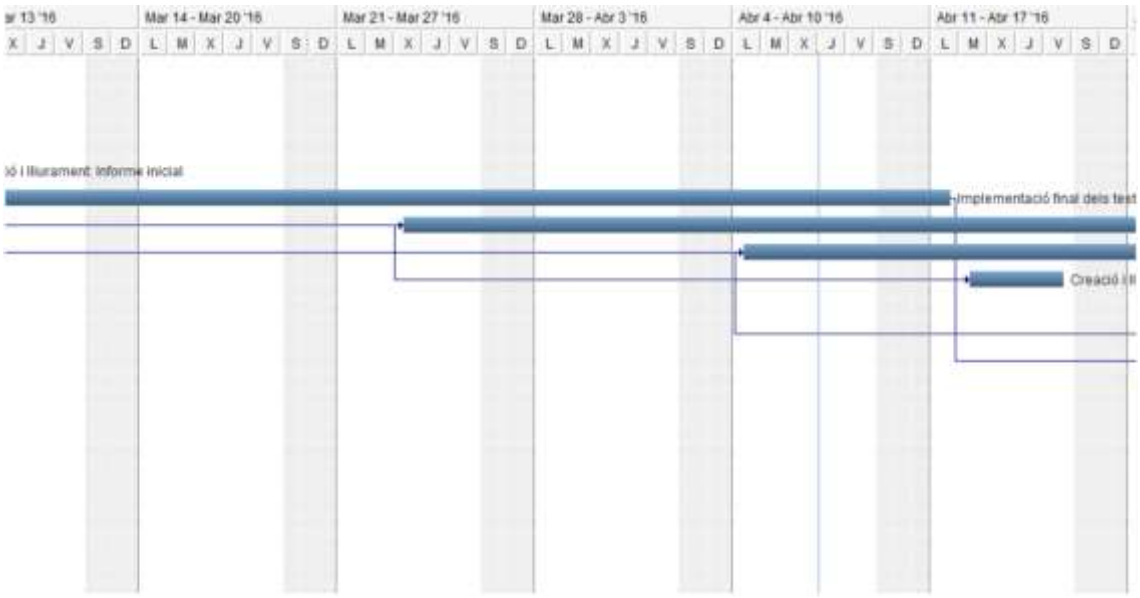

Figura 5: Diagrama temporal

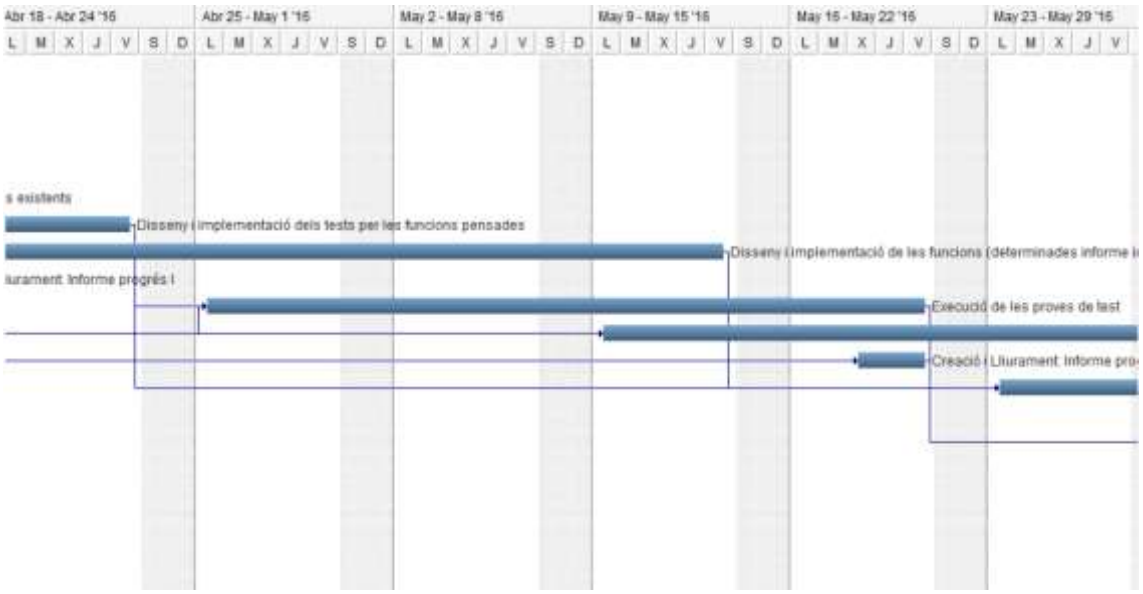

Figura 6: Diagrama temporal

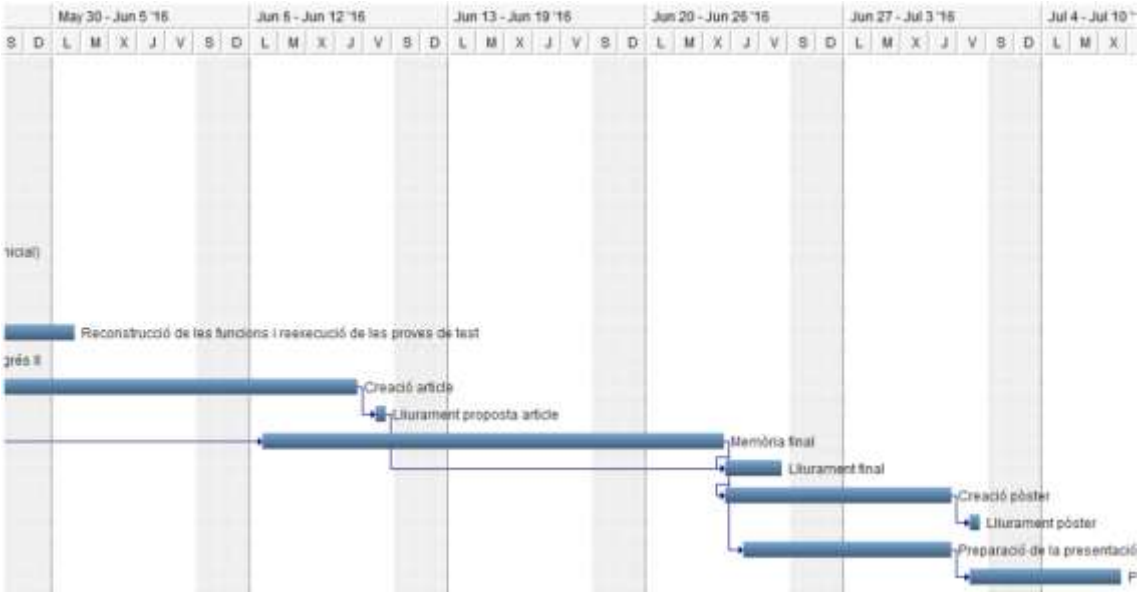

Figura 7: Diagrama temporal

5. Bibliografia

[1] J. Borges, C. Fernández-Córdoba, J. Pujol, J. Rifà and M. Villanueva, "Z_2Z_4-linear codes: generator matrices and duality," *Designs, Codes and Cryptography*, vol. 54, pp. 167-179, 2010.

[2] J. Borges, C. Fernández, J. Pujol, J. Rifà and M. Villanueva, "Z_2Z_4-linear codes. A Magma package," Universitat Autònoma de Barcelona, 2007.

[3] J.Borges, C. Fernández, B. Gastón, J. Pujol, J. Rifà and M. Villanueva, "Z_2Z_4-Additive Codes. A MAGMA Package", Univeristat Autònoma de Barcelona, 2009.

[4] C. Diéguez, "Informe inicial. Ampliació d'una llibreria en MAGMA per a codis Z2Z4-lineals", Informe inicial del projecte de final de carrera, Universitat Autònoma de Barcelona, 2016.

[5] C. Fernández-Córdoba, J. Pujol, and M. Villanueva, "Z_2Z_4-linear codes: rank and kernel," *Desings, Codes and Cryptography* (July 2010) vol. 56. no. 1, pp. 43-59, ISSN: 0925-1022. DOI: 10.1109/TIT.2011.2119465.

[6] B.Gastón, "Codis Z_2Z_4-Additius en MAGMA", Projecte de final de carrera, Universitat Autònoma de Barcelona, 2008.

[7] M. Pujol, "Computing the Minimum Hamming Distance for Z_2Z_4-linear codes", Projecte de final de màster, Universitat Autònoma de Barcelona, 2012.

[8] CCSG Development Group, “CCSG Style Guide”, Univeristat Autònoma de Barcelona, 2011.

[9] CCSG Development Group, “Breu Introducció al MAGMA”, Univeristat Autònoma de Barcelona, 2004.

[10] Computational Algebra Group, “Overview of MAGMA V2.19 Features”, University of Sydney, 2016, https://magma.maths.usyd.edu.au/magma/overview/pdf/overv219.pdf.

[11] “Handbook MAGMA”, https://magma.maths.usyd.edu.au/magma/handbook/

6. Annex I

- El codi de la funció implementada com a “WeightDistribution” és el següent:

```
/***************************************************************/
/*                                                             */
/* Package or project name: Z2Z4AdditiveCodes package          */
/* Test file name: weightDistribution.m                        */
/*                                                             */
/* Comments:                                                   */
/*                                                             */
/* Authors: C. Fernández and C. Diéguez                        */
/*                                                             */
/* Revision version and last date: 1.0, 2016/03/29             */
/*                                                             */
/***************************************************************/

intrinsic WeightDistribution (C)

/*Definir la longitud maxima de la llista*/
type_code=Z2Z4Type(C);
MAX_L:=type_code[1]+2*type_code[2]+1;

/*Inicialitzar la llista a 0*/
/*Llistes comencen a la posicio 1*/
listLeeWeight:=[* *];
for i:=0 to MAX_L do
   listLeeWeight[i+1]:=0;
end for;
```

```
/*Per cada paraula-codi de C, es busca el pes de Lee i es guarda a la llista */
/*la quantitat de paraules-codi codi amb pes X*/
for c in C do
   leeWeight:=Z2Z4LeeWeight(c, type_code[1]);
   listLeeWeight[leeWeight+1]:=listLeeWeight[leeWeight+1]+1;
end for;

/*Es crea una llista que conte el pes i la quantitat de paraules-codi d'aquell pes*/
listWeightDistribution:=[];
for i:=1 to MAX_L do
  if listLeeWeight[i] ne 0 then /*No mostrar els valors <0,0>*/
      a:=< i-1, listLeeWeight[i] >;
      listWeightDistribution:=Append( listWeightDistribution, a);
   end if;
end for;

return listWeightDistribution;

end intrinsic;
```

- El codi de les proves de test per la funció anomenada “WeightDistribution” és el que es mostra a continuació:

```
/*************************************************************/
/*                                                           */
/* Package or project name: Z2Z4AdditiveCodes package        */
/* Test file name: weightDistribution_test.m                 */
/*                                                           */
/* Comments: Black-box tests for the intrinsic functions     */
/*          WeightDistribution() included in the             */
/*          weightDistribution.m file                        */
/*                                                           */
/* Authors: C. Fernández and C. Diéguez                      */
/*                                                           */
/* Revision version and last date: 1.0, 2016/04/11           */
/*                                                           */
/*************************************************************/

//needs Z2Z4AdditiveCode package

SetAssertions(true);
Alarm(30*60);

/*************************************************************
        GLOBAL VARIABLES
*************************************************************/
Z4 := Integers(4);

/*************************************************************/
```

```
/*                                                                    */
/* Function name: Z2Z4AdditiveCode                                    */
/* Parameters: C                                                      */
/* Function description: Create all the codewords of code C           */
/*        and a vector v with N positions. For each codeword u,       */
/*        Lee weight, wL(u), is looked for and it is added one        */
/*        in v where position "i" is equal to Lee weight in each      */
/*        position adds one if Lee weight of u is equal to            */
/*        position "i".                                               */
/*                                                                    */
/* Input parameters description:                                      */
/*   - C: a Z2Z4-additive code                                        */
/* Output parameters description:                                     */
/*   - a vector  “weigthDistribution”  with weight distribution of code C */
/*                                                                    */
/* Signature:                                                         */
/*                                                                    */
/**********************************************************************/

print "WEIGHT DISTRIBUTION OF GENERAL Z2Z4-ADDITIVE CODES AND SOME TRIVIAL ONES";
print "test 1: Trivial Z2Z4-additive zero code
        alpha = 2, beta = 4, gamma = 0, delta = 0, kappa = 0, length = 6, #C = 1";
R := RSpace(Z4, 6);
L := [R!0];
C := Z2Z4AdditiveCode(L : Alpha := 2);

weigthDistribution := WeightDistribution(C);
```

```
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*********************************************************/
print "test 3: Trivial Z2Z4-additive universe code
            alpha = 4, beta = 8, gamma = 4, delta = 8, kappa = 4, length = 12, #C = 1048576";
C := Z2Z4AdditiveUniverseCode(4, 8);

weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
```

```
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);


   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*************************************************************/
print "test 4: Repetiton Z2Z4-additive code
            alpha = 4, beta = 8, gamma = 1, delta = 0, kappa = 1, length = 12, #C = 2";
C := Z2Z4AdditiveRepetitionCode(4, 8);

weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;
```

```
/*********************************************************/
print "test 5: A Z2Z4-additive code with alpha, beta even, p=Id.
          alpha = 10, beta = 20, gamma = 6, delta = 2, kappa = 1, length = 30, #C = 1024";
M := Matrix(Z4,[[2,0,2,0,0,0,2,0,2,2,0,0,0,1,0,0,0,3,2,1,2,0,1,2,0,1,2,1,0,3],
            [0,2,2,0,0,2,0,0,2,2,1,1,1,1,0,1,0,3,2,0,0,2,2,0,1,0,3,2,1,1],
            [0,0,0,0,2,0,2,2,2,0,0,0,0,1,0,0,0,1,2,3,0,0,1,0,2,3,0,3,2,3],
            [0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,2,2,0,0,2,0,2,2,0,0,2,2],
            [0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,2,0,2,2,2,2,2,0,0,2,2,2,0],
            [0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,2,0,2,0,0,0,2,2,0,2,2,2,2],
            [0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,2,0,2,0,0,2,0,0,2,0,2,0,2],
            [0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,2,0,2,0,0,2,2,0,2,0,0,2,2],
            [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,2,0,2,2,2,0,2,0,2,2,0,0],
            [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,0,0,2,2,0,2,2,2]]);
C := Z2Z4AdditiveCode(M : Alpha := 10);

weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
```

```
   print "No image Z2";
end if;



/*************************************************************/
print "test 6: A Z2Z4-additive code with alpha even, beta odd, p <> Id in delta quaternary part
          alpha = 10, beta = 19, gamma = 7, delta = 2, kappa = 3, length = 29, #C = 2048";
M := Matrix(Z4,[[2,0,2,0,0,0,2,0,2,2,0,0,0,1,0,0,0,3,2,1,2,0,1,2,0,1,2,1,0],
            [0,2,2,0,0,2,0,0,2,2,1,1,1,1,0,1,0,3,2,0,0,2,2,0,2,0,3,2,2],
            [0,0,0,0,2,0,2,2,2,0,0,0,0,1,0,0,0,1,2,3,0,0,1,0,2,3,0,3,2],
            [0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,2,2,0,0,2,0,2,2,0,0,2],
            [0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,2,0,2,2,2,2,2,0,0,2,2,2],
            [0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,2,0,2,0,0,0,2,2,0,2,2,2],
            [0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,2,0,2,0,0,2,0,0,2,0,2,0],
            [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,2,0,2,0,0,0,2,2,0,2,0,0,2],
            [0,2,0,2,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,2,0,2,2,2,0,2,0,2,2,0],
            [0,0,2,0,2,0,2,0,2,0,2,0,0,0,0,0,0,2,2,0,0,0,0,0,0,2,2,0,2,2]]);
C := Z2Z4AdditiveCode(M : Alpha := 10);

weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
```

```
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);


   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*********************************************************/

print "test 7: A Z2Z4-additive code with alpha even, beta odd, p <> Id in non delta quaternary
part.
          alpha = 10, beta = 19, gamma = 7, delta = 2, kappa = 3, length = 29, #C = 2048";
M := Matrix(Z4,[[2,0,0,0,0,0,2,2,2,2,0,0,2,2,0,0,2,0,2,0,2,0,0,0,0,0,0,0,0],
            [0,2,0,2,0,0,0,0,0,0,0,0,2,0,2,2,0,0,2,0,2,2,2,0,0,2,0,0,0],
            [0,0,2,0,2,0,2,0,2,0,0,0,0,2,0,0,2,0,2,0,0,0,2,0,0,0,0,0,0],
            [0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,2,0,2,2,2,0,0,0,2,0,0,0,0],
            [0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,2,2,0,0,0,2,0,0,0,0,2,0,0,0],
            [0,0,0,0,0,0,0,0,0,0,0,2,0,2,0,2,2,0,0,0,2,2,2,0,0,0,0,0,0],
            [0,0,0,0,0,0,0,0,0,0,0,2,2,2,0,0,0,0,0,2,2,2,2,2,0,0,0,2,0],
            [0,0,0,2,2,2,2,0,0,2,3,3,3,3,0,3,2,3,0,2,0,2,0,0,0,0,0,1,0],
            [0,0,0,0,2,0,2,2,2,0,2,0,0,3,0,0,0,3,0,3,0,0,3,0,0,1,0,0,1]])
C := Z2Z4AdditiveCode(M : Alpha := 10);


weigthDistribution := WeightDistribution(C1);
weigthDistribution;


/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
```

```
   grayMapImage:= HasLinearGrayMapImage(C);


   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);


   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;


/*************************************************************/
print "test 8: A Z2Z4-additive code with alpha even, beta odd, p <> Id in alpha part.
          alpha = 10, beta = 19, gamma = 7, delta = 2, kappa = 3, length = 29, #C = 2048";
M := Matrix(Z4,[[0,0,2,0,2,0,2,0,2,0,0,0,2,0,0,0,2,0,2,0,0,0,0,0,2,2,2,2,0],
            [2,0,0,0,0,0,2,2,2,2,0,0,0,0,0,0,2,0,2,0,2,0,2,0,2,2,2,2,0],
            [2,0,2,0,2,0,0,2,0,2,0,0,0,0,2,2,0,0,2,0,0,0,0,2,0,2,2,0,0],
            [0,0,2,0,2,0,2,0,2,0,0,2,0,0,0,0,2,0,2,0,2,0,0,2,2,2,0,2,0],
            [0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,2,2,0,0],
            [2,0,0,0,0,0,2,2,2,2,0,0,0,0,2,0,2,2,2,2,2,0,0,0,0,0,2,2,0],
            [2,0,2,2,2,0,0,2,0,2,0,0,0,0,2,0,0,0,0,0,2,0,0,0,0,0,2,0,2,2],
            [0,0,2,0,0,0,0,2,0,0,0,0,0,1,2,0,2,1,0,3,0,1,1,0,2,0,2,0,1],
            [0,2,0,2,0,2,0,2,2,2,1,1,1,0,0,1,2,0,2,3,0,1,1,2,0,2,2,3,1]]);
C := Z2Z4AdditiveCode(M : Alpha := 10);


weigthDistribution := WeightDistribution(C1);
weigthDistribution;
```

```
/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/************************************************************/
print "test 9: A Z2Z4-additive code with alpha=0
          alpha = 0, beta = 19, gamma = 7, delta = 2, kappa = 0, length = 19, #C = 2048";
M := Matrix(Z4,[[0,0,2,0,0,0,2,0,2,0,0,0,0,0,2,2,2,2,0],
            [0,0,0,0,0,0,2,0,2,0,2,0,2,0,2,2,2,2,0],
            [0,0,0,0,2,2,0,0,2,0,0,0,0,0,2,0,2,2,0,0],
            [0,2,0,0,0,0,2,0,2,0,2,0,0,2,2,2,0,2,0],
            [0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,2,2,0,0],
            [0,0,0,0,2,0,2,2,2,2,2,0,0,0,0,0,2,2,0],
            [0,0,0,0,2,0,0,0,0,0,2,0,0,0,0,2,0,2,2],
            [0,0,0,1,2,0,2,1,0,3,0,1,1,0,2,0,2,0,1],
            [1,1,1,0,0,1,2,0,2,3,0,1,1,2,0,2,2,3,1]]);
C := Z2Z4AdditiveCode(M : Alpha := 0);

weigthDistribution := WeightDistribution(C1);
```

```
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*************************************************************/
print "test 10: A Z2Z4-additive code with beta=0
            alpha = 10, beta = 0, gamma = 5, delta = 0, kappa = 5, length = 10, #C = 32";
M := Matrix(Z4,[[0,0,2,0,2,0,2,0,2,0],
            [2,0,0,0,0,0,2,2,2,2],
            [2,0,2,0,2,0,0,2,0,2],
            [0,0,2,0,2,0,2,0,2,0],
            [0,0,0,2,0,0,0,0,0,0],
            [2,0,0,0,0,0,2,2,2,2],
            [2,0,2,2,2,0,0,2,0,2],
            [0,0,2,0,0,0,0,2,0,0],
            [0,2,0,2,0,2,0,2,2,2]]);
C := Z2Z4AdditiveCode(M : Alpha:=10);
```

```
weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*************************************************************/
print "test 11: A Z2Z4-additive code with gamma=0
            alpha = 10, beta = 19, gamma = 0, delta = 7, kappa = 0, length = 19, #C = 16384";
M := Matrix(Z4,[[2,2,0,0,2,2,0,2,2,2,0,0,0,1,0,0,0,0,2,3,2,1,1,3,3,1,2,3,0],
            [0,2,0,0,0,0,2,0,0,0,0,0,0,0,0,2,0,1,0,3,0,2,2,1,3,1,1,0,1],
            [0,0,2,2,0,0,2,2,2,2,0,1,0,0,0,0,0,0,2,3,1,1,0,2,1,0,0,0,1],
            [2,0,0,0,2,0,2,0,0,0,0,0,0,0,0,0,0,1,0,3,3,1,3,1,0,0,3,1,0,2],
            [2,2,2,2,2,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,2,3,0,0,1,1,0,2,2,1],
            [2,0,2,2,2,0,2,2,0,2,0,0,1,0,0,3,0,0,0,1,2,3,1,3,0,2,1,3,2],
            [0,2,2,0,2,0,2,2,0,0,0,0,0,0,1,1,0,0,0,3,3,2,0,2,0,3,1,1,3]]);
C := Z2Z4AdditiveCode(M : Alpha := 10);
```

```
weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*************************************************************/
print "test 12: A Z2Z4-additive code with delta=0
            alpha = 10, beta = 19, gamma = 7, delta = 0, kappa = 4, length = 19, #C = 128";
M := Matrix(Z4,[[2,0,0,0,0,0,2,2,2,0,0,0,0,0,0,2,2,0,0,0,0,0,0,2,0,2,0,2,2],
            [0,2,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,2,2,2,2],
            [0,0,2,0,0,2,2,0,2,0,0,0,0,0,2,0,2,2,0,2,0,0,0,2,2,2,2,0,0],
            [0,0,0,2,0,2,0,0,2,0,0,0,0,0,2,0,0,0,2,0,2,2,0,0,2,0,2,0,2],
            [0,0,0,0,0,0,0,0,0,0,2,0,0,0,2,0,2,0,2,0,0,2,0,2,0,0,0,0,2],
            [0,0,0,0,0,0,0,0,0,0,0,2,0,2,0,2,2,2,0,2,0,2,0,2,2,2,2,0,2],
            [0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,2,2,0,0,0,0,0,2,2,0,2,2]]);
C := Z2Z4AdditiveCode(M : Alpha := 10);
```

```
weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*************************************************************/
print "test 13: A Z2Z4-additive code with kappa=0
          alpha = 10, beta = 19, gamma = 7, delta = 2, kappa = 0, length = 29, #C = 2048";
M := Matrix(Z4,[[0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,2,0,2,0,0,0,0,0,2,2,2,2,0],
          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,2,0,2,0,2,0,2,2,2,2,0],
          [0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,2,0,0,0,0,2,0,2,2,0,0],
          [0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,2,0,2,0,2,0,0,2,2,2,0,2,0],
          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,2,2,0,0],
          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,2,2,2,2,2,0,0,0,0,0,2,2,0],
          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,2,0,0,0,0,2,0,2,2],
          [0,0,2,0,0,0,2,0,0,0,0,0,0,1,2,0,2,1,0,3,0,1,1,0,2,0,2,0,1],
```

```
        [0,2,0,2,0,2,0,2,2,2,1,1,1,0,0,1,2,0,2,3,0,1,1,2,0,2,2,3,1]]);
C := Z2Z4AdditiveCode(M : Alpha := 10);

weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*************************************************************/
print "test 14: A Z2Z4-additive code with kappa=alpha
        alpha = 7, beta = 12, gamma = 8, delta = 2, kappa = 7, length = 19, #C = 4096";
M := Matrix(Z4,[[0,0,0,2,0,2,2,0,0,0,2,2,0,0,2,2,0,0,0],
        [0,0,0,2,0,0,0,0,0,0,2,0,0,0,0,2,2,0,0],
        [0,0,2,0,2,2,2,0,0,0,2,2,0,0,0,0,0,0,0],
        [0,0,0,2,0,0,0,0,0,2,2,2,0,0,0,2,0,0,0],
        [2,0,0,0,2,2,0,0,0,0,2,0,0,0,0,0,0,0,0],
        [0,2,0,0,2,2,0,0,0,0,0,2,0,2,0,0,0,0,0],
```

```
        [0,2,0,2,2,0,0,0,2,0,2,2,0,0,0,0,0,0,2],
        [0,2,0,2,0,0,0,0,0,0,2,2,0,0,0,0,0,2,2],
        [0,2,0,0,0,0,2,0,0,0,0,0,1,1,1,2,1,1,2],
        [0,2,0,2,0,0,2,1,1,1,2,0,0,1,0,0,1,0,3]]);
C := Z2Z4AdditiveCode(M : Alpha:=7);

weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*********************************************************/
print "test 15: A Z2Z4-additive code with kappa=gamma
        alpha = 9, beta = 12, gamma = 8, delta = 2, kappa = 8, length = 21, #C = 4096";
M := Matrix(Z4,[[0,0,0,2,0,2,2,2,2,0,0,2,0,0,0,0,2,0,2,0,0],
        [0,0,0,0,2,2,2,0,2,0,0,0,0,0,0,2,2,0,0,0,0],
        [2,2,2,0,0,0,2,2,2,0,0,0,0,0,2,0,0,0,0,0,0],
```

```
        [0,0,0,0,2,0,2,2,0,0,0,0,2,0,0,0,2,0,0,0,0],
        [0,0,0,2,0,0,0,2,0,0,0,0,0,0,0,0,0,0,2,2,0],
        [2,0,0,0,0,0,0,2,2,0,0,0,0,2,0,0,0,0,0,0,0],
        [2,0,2,2,2,0,2,0,2,0,0,0,0,0,0,0,0,0,2,0,0,0],
        [2,0,0,2,0,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,2],
        [2,2,0,2,0,2,0,0,2,0,1,0,0,1,1,0,2,1,2,1,1],
        [2,2,0,0,0,2,0,2,0,1,0,1,1,1,0,1,0,1,2,1,0]]);
C := Z2Z4AdditiveCode(M : Alpha := 9);

weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*********************************************************/
print "test 16: A Z2Z4-additive code with a zero column in alpha part.
        alpha = 9, beta = 12, gamma = 6, delta = 3, kappa = 4, length = 21, #C = 4096";
```

```
M := Matrix(Z4,[[2,0,0,0,2,2,2,0,2,0,0,0,0,2,0,0,0,0,0,0,2],
        [2,0,0,2,0,0,2,0,2,0,0,2,0,2,0,0,2,0,0,2,0],
        [0,0,2,2,0,2,2,0,0,0,0,0,0,2,2,0,0,0,2,2,0],
        [0,0,2,2,0,2,2,0,0,0,0,0,0,0,0,2,2,0,2,2,0],
        [0,0,0,0,2,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [2,0,0,2,0,0,2,0,2,0,0,0,0,0,0,0,2,0,2,0],
        [0,0,0,0,0,2,2,2,2,0,0,0,1,0,1,1,0,0,1,2,1],
        [0,0,2,0,2,2,0,0,2,1,0,1,0,2,0,1,0,1,0,3,0],
        [0,0,0,2,0,2,2,0,2,0,1,1,0,1,0,0,2,0,1,0,0]]);
C := Z2Z4AdditiveCode(M : Alpha := 9);

weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;

/*********************************************************/
```

```
print "test 17: A Z2Z4-additive code with a zero column in beta part.
        alpha = 9, beta = 12, gamma = 6, delta = 3, kappa = 4, length = 21, #C = 4096"
M := Matrix(Z4,[[2,0,0,0,2,2,2,0,2,0,0,0,0,2,0,0,0,0,0,2],
        [2,0,0,2,0,0,2,0,2,0,0,2,0,2,0,0,2,0,0,2,0],
        [0,0,2,2,0,2,2,0,0,0,0,0,0,2,2,0,0,0,2,2,0],
        [0,0,2,2,0,2,2,0,0,0,0,0,0,0,0,2,2,0,2,2,0],
        [0,0,0,0,2,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [2,0,0,2,0,0,2,0,2,0,0,0,0,0,0,0,0,0,0,2,0],
        [0,0,0,0,0,2,2,2,2,0,0,0,1,0,1,1,0,0,1,2,1],
        [0,0,2,0,2,2,0,0,2,1,0,1,0,2,0,1,0,0,0,3,0],
        [0,0,0,2,0,2,2,0,2,0,1,1,0,1,0,0,2,0,1,0,0]]);
C := Z2Z4AdditiveCode(M : Alpha := 9);

weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
   print "No image Z2";
end if;
```

```
/*************************************************************/

print "test 18: Another Z2Z4-additive code with two zero columns in beta part.
        alpha = 9, beta = 12, gamma = 6, delta = 3, kappa = 4, length = 21, #C = 4096";
M := Matrix(Z4,[[2,0,0,0,2,2,2,0,2,0,0,0,0,2,0,0,0,0,0,0,2],
        [2,0,0,2,0,0,2,0,2,0,0,2,0,2,0,0,2,0,0,0,0],
        [0,0,2,2,0,2,2,0,0,0,0,0,0,2,2,0,0,0,2,0,0],
        [0,0,2,2,0,2,2,0,0,0,0,0,0,0,0,2,2,0,2,0,0],
        [0,0,0,0,2,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [2,0,0,2,0,0,2,0,2,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,2,2,2,2,0,0,0,1,0,1,1,0,0,1,0,1],
        [0,0,2,0,2,2,0,0,2,1,0,1,0,2,0,1,0,0,0,0,0],
        [0,0,0,2,0,2,2,0,2,0,0,0,1,0,1,1,0,0,1,0,0]]);
C := Z2Z4AdditiveCode(M : Alpha := 9);

weigthDistribution := WeightDistribution(C1);
weigthDistribution;

/*Demostració a Z2*/
if HasZ2Z4LinearGrayMapImage(C) then
   grayMapImage:= HasLinearGrayMapImage(C);

   print "Distrubucio de pesos de Hamming:";
   expectedOutputWeightDsitribution := WeightDistribution(grayMapImage);

   assert weigthDistribution eq expectedOutputWeightDsitribution;
else
```

```
   print "No image Z2";
end if;
```