Ecole Polytechnique Fédérale de Lausanne

Sciences de l'information

Série 12

Olivier Cloux

12.1

1. $a_1 = (s \mod 9) + 1 = 3 + 1 = \boxed{4}$
   $i = (s \mod 8) + 1 = 1 + 1 = 2$. a_2 est le second élément de $\{1, 2, 3, 5, 6, 7, 8, 9\}$, qui est $\boxed{2}$
   $j = (0s \mod 7) + 1 = 3 + 1 = 4$. a_3 est le quatrième élément de $\{1, 3, 5, 6, 7, 8, 9\}$, qui est $\boxed{6}$
   $b_1 = (a_1^2) \mod 13 = 4^2 \mod 13 = 16 \mod 13 = \boxed{3}$
   $b_2 = (a_2^2) \mod 13 = 2^2 \mod 13 = 4 \mod 13 = \boxed{4}$
   $b_3 = (a_3^2) \mod 13 = 6^2 \mod 13 = 36 \mod 13 = \boxed{10}$

2. Pour trouver H, nous devons d'abord passer G en forme systématique :

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 12 & 4 & 2 & 6 \\ 0 & 1 & 3 & 4 & 10 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 4 & 10 \\ 0 & 0 & 7 & 6 & 3 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 11 & 10 & 4 \\ 0 & 1 & 3 & 4 & 10 \\ 0 & 0 & 1 & 12 & 6 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 8 & 3 \\ 0 & 1 & 0 & 7 & 5 \\ 0 & 0 & 1 & 12 & 6 \end{pmatrix}$$

$$\rightarrow \begin{cases} x_1 = u_1 \\ x_2 = u_2 \\ x_3 = u_3 \\ x_4 = 8u_1 + 7u_2 + 12u_3 \\ x_5 = 3u_1 + 5u_2 + 6u_3 \end{cases} \rightarrow \begin{cases} 5x_1 + 6x_2 + x_3 + x_4 = 0 \\ 10x_1 + 8x_2 + 7x_5 + x_5 = 0 \end{cases} \rightarrow H = \begin{pmatrix} 5 & 6 & 1 & 1 & 0 \\ 10 & 8 & 7 & 0 & 1 \end{pmatrix}$$

Algorithm 1 Algorithme exo3.c

3.
```
G[5][3]
dmin ← 5
dtemp ← 0
motDeCode[3]
encode[5]
for (i = 0; i < 12; i++) do
    motDeCode[0] ← i
    for (j = 0; j < 12; j++) do
        motdeCode[1] ← j
        for (k = 0; k < 12; k++) do
            motDeCode[2] ← k
            multimat(encode, motDeCode, G)        → multimat est expliquée plus bas
            for l = 0, l < 5, l++ do
                if encode[l]! = 0 then
                    dtemp++
                end if
            end for
            if (dtemp < dmin) ∧ (dtemp! = 0) then
                dmin = dtemp
            end if
        end for
    end for
end for
return dmin
```

La fonction multimat prend en argument 3 tableaux (matrices), multiplie les second et troisième et met le résultat de la multiplication dans le premier.

4. Selon la théorie, notre code peut corriger tous les effacement de poids $p < d_{min}(C)$. Or, $d_{min}(C) = 3$, donc il est possible de corriger toutes les erreurs de poids 2 maximum

5. Nous savons qu'il existe un mot $(x\ y\ z)$ tel que $(x\ y\ z) \cdot G = (u\ 5\ 3\ v\ 9)$. En considérant ce que nous savons pour sûr (les colonnes 2,3 et 5), nous pouvons affirmer que notre mot doit satisfaire :

$$\begin{cases} x + 12y + z = 5 \\ x + 4y + 3z = 3 \\ x + 6y + 10z = 9 \end{cases} \rightarrow \left(\begin{array}{ccc|c} 1 & 12 & 1 & 5 \\ 1 & 4 & 3 & 3 \\ 1 & 6 & 10 & 9 \end{array}\right)$$

Nous réduisons cette matrice comme nous savons très bien le faire, car il ne s'agit que de la 80ème fois. Je me sens las. Nous obtenons donc

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 12 \end{array}\right)$$

Ce qui nous donne que le seul mot qui donne un mot encodé similaire est le mot (6 0 12). Nous le multiplions par G pour obtenir l'encodage complet, ce qui est (6 5 3 2 9) donc $u = 6$ et $v = 2$. Il est donc tout a fait possible de corriger cette erreur (car elle est de poids 2)

6. a) Nous n'avons qu'à multiplier nos vecteurs par H^T, pour obtenir un vecteur de taille 1×2 qui est le syndrome. Un vecteur est un mot de code si et seulement si le syndrome vaut $\vec{0}$.

Ainsi :

$$\begin{pmatrix}\vec{A}\\ \vec{B}\\ \vec{C}\end{pmatrix} \cdot H^T = \begin{pmatrix}0 & 7 & 11 & 12 & 10\\ 8 & 12 & 11 & 12 & 10\\ 0 & 7 & 8 & 12 & 10\end{pmatrix} \cdot \begin{pmatrix}5 & 10\\ 6 & 8\\ 1 & 7\\ 1 & 0\\ 0 & 1\end{pmatrix} = \begin{pmatrix}0 & 0\\ 5 & 3\\ 10 & 3\end{pmatrix} = \begin{pmatrix}\vec{s}_A\\ \vec{s}_B\\ \vec{s}_C\end{pmatrix}$$

Donc $\vec{A}$ est un mot de code, alors que $\vec{B}$ et $\vec{C}$ n'en sont pas

b) $\vec{e} \in F_{13}^5$ signifie toutes les vecteurs de dimension 5, dans la classe de congruence 13. De plus, $w(\vec{e}) = 1$ signifie que sa distance à 0 doit être de 1, donc avoir un et seulement un élément différent de 0. Soit l'ensemble des vecteurs

$$\{(0,0,0,0,1),(0,0,0,0,2),(0,0,0,0,3),\ldots,(0,0,0,0,12),(0,0,0,1,0),(0,0,0,2,0,),\ldots(12,0,0,0,0)$$

Nous voyons bien qu'il ne s'agit pas d'un espace vectoriel, car la somme de deux éléments n'est pas un élément de l'espace. Par exemple, (0,0,0,0,1) + (1,0,0,0,0) = (1,0,0,0,1), qui n'est pas un élément de ϵ_1.

Son cardinal est $12 \cdot 5 = 60$, car chaque coordonnée du vecteur peut prendre 12 formes (de 1 à 12), mais indépendamment : le premier peut prendre 12 formes et les autres sont à 0, puis le second prend 12 formes avec les autres à 0 et ainsi de suite. Ainsi : $12 \cdot 5 = 60$.

Le code C engendré par notre matrice génératrice est de distance minimale 3, ce qui signifie qu'aucun mot de code n'est de distance 1 ou 2 (et un seul mot, $\vec{0}$, est de distance 0) alors qu'au moins un mot de code est de poids 3.
Supposons, par l'absurde, qu'il existe deux mots distincts $\vec{a}$ et $\vec{b}$ tels que $\vec{a}H^T = \vec{b}H^T$. Par un simple calcul, nous voyons qu'il est nécessaire que $(\vec{a}-\vec{b})H^T = 0$, ce qui signifie que, par définition de la matrice de contrôle, que $(\vec{a}-\vec{b})$ est un mot de code valide. Comme les deux mots sont différents, $w(\vec{a}-\vec{b}) = 2$ (ou 1, par exemple avec $(0,0,5,0,0)-(0,0,3,0,0)$)Par définition de notre code, cela est strictement impossible. Il est donc obligatoire que $\vec{a} = \vec{b}$. Donc f est injectif. Par définition de l'injectivité, le cardinal de F_1 est le même que celui de ϵ_1, soit 60.
L'application n'est donc pas bijective, car $|F_{13}^2| = 13^2 = 169 \neq 60 = |F_1$

c)

$$\begin{aligned} & \vec{e} = \vec{y} - \vec{x} \\ \iff & \vec{y} = \vec{e} + \vec{x} \end{aligned}$$

Nous savons que $\vec{x}$ est un mot de code valide, donc que son syndrome vaut 0. Ainsi

$$\vec{s} = \vec{y}H^T = (\vec{e}+\vec{x})H^T = \vec{e}H^T + \vec{x}H^T = \vec{e}H^T$$

Puisque l'erreur est simple (de poids 1), nous pouvons conclure que $w(\vec{e}) = 1$ et donc (comme nous l'avons montré dans le point précédent), $\vec{e} \in \epsilon_1$. Ainsi

$$\vec{s} = \vec{e}H^T = f(\vec{e}) \in F_1$$

Comme nous venons aussi de le montrer.

Algorithm 2 Algorithme 6_*d*.c :

d)
```
S[2];
e[5];
H[2][5];
sol[2];
for (i = 0; i < 5; i++) do
    setNull(e);                          (▷) Met tous les éléments de e à 0
    for (j = 1; j < 13; j++) do
        sol = multimat(e, G)
        if ((S[0] == sol[0]) ∧ (S[1] == sol[1])) then
            return e
        end if
    end for
end for
return noSol;
```

Algorithm 3 Algorithme decodeur.c :

e)
```
x[5];
y[5];
S[2];
e[5];
eSol[5];
H[2][5];
S = multimat(y, H);
sol[5]
boolean corrigible = false;
if (S = 0) then
    return y;
else
    for (all e) do
        sol = multimat(e, H);
        if ((S[0] == sol[0]) ∧ (S[1] == sol[1])) then
            eSol = e
        end if
    end for
end if
if (corrigible) then
    x = y − eSol;
    return x;
else
    return nosol;
end if
```

En appliquant notre algorithme à A,B et C, nous trouvons que A est vrai, que B ne peut pas être corrigé, et que C peut l'être, et que sa version corrigée est (0,7,8,12,0).

f) Cela est tout a fait possible. Pour cela, nous n'avons qu'à considérer $\vec{x}$ comme étant la somme de 3 éléments de ϵ_1, $\vec{a}+\vec{b}+\vec{c}$. Donc $\vec{x}H^T = 0$ (par définition) $\iff$ $(\vec{a}+\vec{b}+\vec{c})H^T = 0 \iff -(\vec{a}+\vec{b})H^T = \vec{c}H^T = \vec{s}$ (nous définissons $\vec{s}$ comme tel). $\vec{s}$ est forcément dans F_1, car c est dans ϵ_1. Mais $w(-\vec{a}-\vec{b}) = 2$ (car deux mots différents dans ϵ_1, nous l'avons montré avant). Cela constitue une erreur double $\vec{e} = -\vec{a}-\vec{b}$ qui a pour syndrome un vecteur de F_1.

En effet, notre algorithme peut corriger des erreurs de poids simple (et nous pouvons même corriger des erreurs de poids 2). Mais en admettant qu'une erreur crée presque un mot valide, il est possible de confondre cette erreur pour une erreur simple, alors qu'il s'agissait d'une erreur de poids 4.