Codes correcteurs d’erreurs

N’oubliez pas de charger en mémoire la bibliothèque d’algèbre linéaire.

```
> with(linalg):
```

Codes de Hamming

En dimension 7: le code de Hamming H_7

Par définition, le code de Hamming H_7 est un sous-espace vectoriel de dimension 4 de l’espace $Z/2Z^7$ Pour simplifier le décodage, il est commode de définir H_7 par un système de 3 équations indépendantes. Par construction, on choisit les 3 équations de sorte que si l’on met leurs coefficients en ligne dans une matrice 3x7 V7 les 7 vecteurs colonnes de V7 sont les écritures en base 2 des nombres de 1 à 7.

Il faut commencer par créer cette matrice avec Maple.

On donne la fonction b2 suivante transforme un nombre $x<2^n$ en un vecteur de longueur n dont les composantes donnent la décomposition en base 2 de x.

```
> b2:=proc(x,n)
  local r;
  r:=x mod 2;
  if n=1 then [r];
  else
  [op(b2(iquo(x,2),n-1)),r]
  fi;
  end:
```

Par exemple, pour avoir la décomposition de 6=4+2:

```
> b2(6,3);
```

1) Créez une matrice 3x7 dont les colonnes sont (dans l’ordre) les écritures en base 2 des nombres entre 1 et 7.

```
> V7:=
```

2)Vérifiez que les trois équations sont effectivement indépendantes.

Attention: Pour demander à Maple de travailler sur le corps (Z/2Z), vous pouvez utiliser les fonctions suivantes (Noter les majuscules):

```
> ?Gausselim
> ?Nullspace
```

Pour réduire un vecteur ou un matrice modulo 2, vous pouvez utiliser la fonction reduit suivante.

```
> reduit:=proc(M)
  map(modp,evalm(M),2)
  end:
```

Par définition, le code H_7 est le noyau de votre matrice V7.

3) Calculez une base de H_7 et explicitez les 16 mots du code H_7.

Par définition, le **poids** d’un mot (= un vecteur) est le nombre de composantes non nulles.

Par exemple, le mot

```
> v:=vector([0,1,0,1,0,0,1]);
```

$$v := [0, 1, 0, 1, 0, 0, 1]$$

```
>
```

$$[0, 1, 0, 1, 0, 0, 1]$$

est de poids 3.

4) Ecrivez une fonction poids qui prend en argument un vecteur et calcule son poids.

```
> ?vectdim
> poids:=proc(v)

  end:
```

Par définition, la **distance** d'un code est le minimum des poids de tous les mots du code.

5) Calculez la distance du code H_7. En déduire que le code H_7 permet de corriger au plus une erreur.

```
> ?min
>
```

6) Si c est un mot du code H_7, que vaut le produit matriciel V7 &* c?

Si v est un mot du code qui a subi une erreur au cours de la transmission, on peut l'écrire v=c+e, où e est un mot de poids 1. Que vaut le produit matriciel V7 &* v?

En déduire un algorithme pour décoder un mot reçu ayant au plus une erreur et programmez le.

```
> decodeH7:=proc(v)

  end:
```

Testez votre fonction decodeH7 grâce à la fonction test suivante. Si vous obtenez toujours true c'est bon; si vous obtenez false c'est que votre fonction ne marche pas bien.

```
>
> test:=proc()
  local c,e,K,modif,i,dec;
  K:=augment(op(Nullspace(V7)mod 2));
  c:=map(modp,evalm(augment(op(Nullspace(V7)mod 2))&*
  randvector(4,entries=rand(0..1))),2);
  e:=rand(1..7)();
  modif:=copy(c);
  modif[e]:=1-modif[e];
  dec:=decodeH7(modif);
  for i from 1 to 7 do
  if (dec[i]<>c[i]) then RETURN(false) fi;
  od;
  true;
  end:
>
> test();
```

7) En faisant des opérations sur les lignes de V7, montrer que les 16 mots du code H_7 sont obtenus à partir des 16 mots de longueur 4 en leur rajoutant 3 bits de parité à des sous-mots.

```
> ?addrow
> N:=reduit(addrow(V7,,,));
```

En déduire une procédure encode qui associe à tout mot m de longueur 4 un mot du code H_7 commençant par m.

```
> encode:=proc(m)

  end;
```

Variante cyclique de H_7

La matrice V7bis suivante a 7 colonnes qui sont les écritures en base 2 des nombres de 1 à 7 mais dans un ordre différent. Son noyau définit donc un code H_7bis qui est comme H_7 1-correcteur parfait.

```
> V7bis:=matrix([[1, 0, 0, 1, 0, 1, 1], [0, 1, 0, 1, 1, 1,
  0], [0, 0, 1, 0, 1, 1, 1]]);
```

Vérifier que ce code H_7bis est **cyclique**, c'est à dire qu'il contient un vecteur v=($v_1, v_2, v_3, v_4, v_5, v_6, v_7$) si et seulement s'il contient son permuté circulaire ($v_2, v_3, v_4, v_5, v_6, v_7, v_1$).

En dimension 8: le code de Hamming étendu H_8

Par définition, le code de Hamming étendu H_8 est le code obtenu à partir du code H_7 précédent en rajoutant aux mots de H_7 un bit de parité.

Ecrire une matrice V8 dont les lignes sont les équations définissant H_8. Donner une base de H_8 et calculez sa distance.
Combien d'erreurs peut-on corriger avec ce code?

En dimension 15: le code de Hamming H_{15}

En mimant la construction de H7, construire un code H_{15} de dimension 11 et de longueur 15 et qui est 1-correcteur parfait.

Une variante de H_{15} pour corriger 2 erreurs

Voici comment on peut utiliser une variante du code H_{15} pour corriger 2 erreurs.
Les matrices V1 et V2 ci-dessous ont pour colonnes les écritures en base 2 des nombres de 1 à 15, mais dans un ordre différent. Soit C le code linéaire noyau de la matrice V obtenue en superposant V1 et V2.
Vérifier que C permet de corriger 2 erreurs.

```
> V1:=matrix([[1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1],
  [0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0], [0, 0, 1,
  0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0], [0, 0, 0, 1, 0, 0, 1,
  1, 0, 1, 0, 1, 1, 1, 1]]);
> V2:=matrix([[1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1],
  [0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1], [0, 0, 1,
  0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1], [0, 1, 1, 1, 1, 0, 1,
  1, 1, 1, 0, 1, 1, 1, 1]]);
> V:=(stack(V1,V2));
```

Programmez un algorithme naif de décodage (on peut faire plus rapide avec un peu plus de mathématiques).

```
>
```

Codes tirés au hasard

Tirer quelques codes au hasard (pour n=15 et k=11 par exemple) et calculez leur distance.
Peut-on espérer tirer un bon code au hasard?

```
> V:=randmatrix(4,15,entries=rand(0..1));
>
```

Le code de Golay G_{23}

On présente ici un autre code parfait, découvert par le physicien suisse Marcel Golay en 1949. On peut montrer que c'est le seul code correcteur binaire parfait permettant de corriger au moins deux erreurs. Ce code a eu une grande importance dans un domaine plus abstrait des mathématiques: la classification des groupes finis simples.

Le code de Golay G_{23} et par définition donné par les équations suivantes:

```
> V23:=matrix([[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
  0, 0, 1, 0, 0, 1, 0], [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
  1, 1, 1, 1, 0, 0, 1, 0, 0, 1], [0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
  0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0], [0, 0, 0, 1, 0, 0, 0,
  0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1], [0, 0, 0, 0,
  1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1], [0,
  0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1,
  0, 1], [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1,
  1, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
  1, 0, 1, 1, 1, 1, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
  0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0,
  1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1], [0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1]]);
```

Calculer la dimension du code G_{23} ainsi que sa distance. Combien d'erreurs peut-il corriger?
Vérifier que ce code est parfait.

```
>
```