Mathématique

DUT INFO 2 / 2013-2014

Mis à jour le 12 décembre 2013 à 23:40

Algèbre et informatique

Guillaume Connan

IUT Nantes Informatique

Université de Nantes

TABLE DES MATIÈRES

CHAPITRE

1

Opérations, lois et structures

Les langages de programmation modernes sont très algébriques : ils puisent dans ce domaine mathématique, développé surtout depuis la fin du XIX$^{\text{e}}$ siècle, la rigueur et les structures qui leur permettent de créer des programmes sûrs et vérifiables a priori. Les structures algébriques sont aussi les outils qui servent de base au codage de l'information, en cryptographie, elles sont aussi le point de départ des outils de transformation en infographie, en traitement d'image, elles offrent des outils indispensables à la théorie des langages, de la compilation, des types.
Bref, elles sont un passage obligé pour bien saisir l'informatique moderne..

1 Opération

Une opération est un concept à la fois simple (on l'utilise depuis le CP) et compliqué : en mathématique actuelle et en informatique théorique, ce terme recouvre des notions très abstraites. Comme nous n'avons guère vu que des ensembles, nous nous contenterons *pour l'instant* de dire ici qu'il s'agit d'une **fonction** dont les paramètres seront appelés *opérandes*. On utilise le plus souvent un symbole pour désigner cette fonction : c'est l'*opérateur*.

1 1 Opération unaire

C'est simplement un autre nom d'une fonction d'un ensemble dans un autre...
On dit aussi dans ce cas que l'opération est d'**arité** 1.
Par exemple, la négation d'une proposition sur l'ensemble des propositions, la fonction qui renvoie l'opposé d'un entier sur l'ensemble des entiers mais on peut imaginer tout ce qu'on veut...
Considérons la fonction **`vers_majuscule`** de l'ensemble des caractères dans lui-même qui à une lettre minuscule renvoie la lettre majuscule associée.

Aparté

> Pourquoi le 'A' est codé 65 et le 'a' 97 en ASCII ?
> Vous avez remarqué que $97-65=32=2^5$.
> Mais il vaut mieux voir les choses ainsi :
> 'A' -> [1,0,0,0,0,0,1]
> 'a' -> [1,1,0,0,0,0,1]
> 'Z' -> [1,0,1,1,0,1,0]
> 'z' -> [1,1,1,1,0,1,0]
> Qu'en pensez-vous ?

En Haskell :

Haskell
```haskell
import qualified Data.Char as Car

versMaj :: Char                          -> Char
versMaj    lettre
  | (lettre < 'a') || (lettre > 'z')  =  error "Ce n'est pas une minuscule !!!"
  | otherwise                         =  Car.chr (Car.ord lettre - 32)
```

On utilise deux fonctions fondamentales dont on aura besoin en cryptographie, **`chr`** et **`ord`** du module **`Data.Char`** qui font le lien entre un caractère et son code ASCII.

Haskell
```haskell
*Main> Car.ord 'a'
97
*Main> Car.ord 'A'
65
*Main> map Car.ord ['a'..'z']
[97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,
 118,119,120,121,122]
*Main> map Car.ord ['A'..'Z']
[65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90]

*Main> Car.chr 97
'a'
*Main> map Car.chr [32 .. 126]
" !\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]
  ^_`abcdefghijklmnopqrstuvwxyz{|}~"
```

Par exemple :

Haskell

```
*Main> versMaj 'a'
'A'
*Main> map versMaj "tralalahop"
"TRALALAHOP"
```

Bon, cette fonction existait déjà...

Haskell
```
*Main> map Car.toUpper ['a'..'z']
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

mais fonctionne autrement :

Haskell
```
*Main> map Car.toUpper "Ho ! Tu vas te mettre en majuscule ! #@$ $ %*$"
"HO ! TU VAS TE METTRE EN MAJUSCULE ! #@$ $ %*$"
```

Comment modifier notre **versMaj** pour qu'elle fonctionne comme **to_Upper** ?

1 2 Opération binaire

Une opération binaire est une fonction de signature $A \otimes B \to C$. On dit qu'elle est d'arité 2. Par exemple, que pensez-vous de :

Haskell
```
somCar :: (Char,Char) -> Char
somCar    (a    ,b)    =  Car.chr ((Car.ord a) + (Car.ord b))
```

dont voici des réalisations :

Haskell
```
*Main> somCar ('A','!')
'b'
*Main> somCar ('1','2')
'c'
```

On n'est pas obligé de travailler dans le même ensemble :

Haskell
```
estCode :: (Char,Int) -> Bool
estCode (car,nb) = (Car.ord car == nb)
```

Par exemple :

Haskell
```
*Main> estCode ('a', 97)
True
```

Mais la *curryfication* nous permet de transformer toute fonction de plusieurs variables en une fonction d'une variable :

Haskell
```
estCode_c :: Char -> Int -> Bool
estCode_c    car     nb   =  Car.ord car == nb
```

Cette fonction curryfiée, donc d'une seule variable, sera malgré tout considérée comme une opération binaire.
Ici, l'opérateur est placé avant ses arguments : on dit qu'il est dénoté par une notation **préfixée**. D'habitude, les opérateurs binaires sont dénotés par une notation **infixée**. C'est possible aussi en Haskell :

Haskell
```
infix @+

(@+) a b = somCar (a,b)
```

On peut maintenant utiliser une syntaxe plus habituelle :

Haskell
```
*Main> 'A' @+ '!'
'b'
```

On peut utiliser une notation **postfixée** comme pour la *Notation Polonaise Inversée* utilisée en assembleur, en postscript,...
Par exemple, 3 2 + désigne la somme de 3 et de 2.

1 3 Opération d'arité supérieure

On peut continuer ainsi.
Prenons par exemple l'opération ternaire définie sur $\mathbb{Z}$ par :

— Version non curryfiée :

Haskell
```
f :: (Int,Int,Int) -> Int
f    (a  ,b  ,c  ) =  a^2 + b^2 + c^2
```

Haskell
```
*Main> f (1,2,3)
14
```

— Version curryfiée :

Haskell
```
f :: Int -> Int -> Int -> Int
f    a      b      c    =  a^2 + b^2 + c^2
```

Haskell
```
*Main> f 1 2 3
14
```

On peut aussi la considérer comme opération d'arité 1 sur l'ensemble des triplets de $\mathbb{Z}^3$...

2 Loi de composition

2 1 Définitions

Une loi est une opération d'arité 2 qui est une fonction totale. On distingue deux cas :

Définition 1 - 1

Loi de composition interne (LCI)
Une loi de composition interne définie sur un ensemble E est une fonction totale (ou application) de $E \otimes E$ dans E. On dit alors que E est un **magma**.

Donnez vous-mêmes de nombreux exemples....

Définition 1 - 2

Stabilité
Un ensemble E est stable par une opération $\star$ si, et seulement si :

$$\Big(\forall\langle x, y\rangle\Big)\Big(\big(\langle x, y\rangle \in E \otimes E\big) \to \big(x \star y \in E\big)\Big)$$

Définition 1 - 3

Loi de composition externe (LCE)
Une loi de composition externe définie sur un ensemble E et à opérateurs dans un ensemble K est une fonction totale (ou application) de $K \otimes E$ dans E.

Donnez vous-mêmes de nombreux exemples....
Voici enfin une notion primordiale dont nous reparlerons souvent :

Définition 1 - 4

Morphisme
Soit $\langle E, \star \rangle$ et $\langle F, \dagger \rangle$ deux magmas et φ une fonction totale de E dans F. On dit que φ est un morphisme de E dans F si, et seulement si :

$$\big(\forall x\big)\big(\forall y\big)\big(\varphi(x \star y) = \varphi(x) \dagger \varphi(y)\big)$$

Pouvez-vous donner quelques exemples ?

Définition 1 - 5

Isomorphisme
Soit $\langle E, \star \rangle$ et $\langle F, \dagger \rangle$ deux magmas et φ une fonction totale de E dans F. On dit que φ est un morphisme de E dans F si, et seulement si, c'est un morphisme BIJECTIF de E dans F.

C'est une notion primordiale !!! Cela permet de simplifier (si si) énormément de situations qui paraissent différentes mais qui sont les mêmes *à isomorphisme près*...

2 2 Propriétés des lois (de composition interne)

Définition 1 - 6

Commutativité
Une LCI $\star$ sur E est commutative si, et seulement si

$$(\forall x)(\forall y)(x \star y = y \star x)$$

Un magma muni d'une loi commutative est dit *abélien* (en hommage au mathématicien norvégien Niels Henrik ABEL (1802 - 1829)).

Définition 1 - 7

Associativité
Une LCI $\star$ sur E est associative si, et seulement si

$$(\forall x)(\forall y)(\forall z)(x \star (y \star z) = (x \star y) \star z)$$

Un magma muni d'une loi associative est appelé... un magma associatif.

Aparté

Il y a une grosse bataille franco-anglo-saxonne au sujet de ce qu'est un semi-groupe ou un demi-groupe. Nous allons nous conformer à la dernière référence en date, même si elle n'est pas française : « Homotopy Type Theory » disponible à cette adresse http://homotopytypetheory.org/book/.1 Un magma associatif y est défini page 122 comme un semi-groupe.

Quand des lois sont associatives, on peut se passer de parenthèses et on peut noter plus simplement leurs itérations.
Par exemple, depuis le collège, vous notez a^n la multiplication itérée n fois de a par lui-même.
Certaines lois ne sont pas associatives mais on peut *décider*, pour se simplifier la vie au moment de les créer, qu'elles sont associatives à droite (ou à gauche).
Sur Haskell par exemple, l'opérateur `->` entre domaine et codomaine d'une fonction est associatif à droite. En effet, les concepteurs de Haskell ont décidé que `Char -> Int -> Bool` signifie en fait `Char -> (Int -> Bool)`.
On peut ainsi se passer de parenthèses mais il faudra bien garder à l'esprit que l'associativité n'est effective qu'à droite.
Par exemple, commentez ce qui passe ici :

Haskell
```haskell
estCode :: Char -> Int -> Bool
estCode car nb = Car.ord car == nb
```

Haskell

```
*Main> :t estCode 'a'
estCode 'a' :: Int -> Bool
*Main> :t Car.ord
Car.ord :: Char -> Int
*Main> :t estCode (Car.ord)

<interactive>:1:11:
    Couldn't match expected type `Char' with actual type `Char -> Int'
    In the first argument of `estCode', namely `(Car.ord)'
    In the expression: estCode (Car.ord)
```

Définition 1 - 8

Élément neutre

Un élément neutre e d'une LCI $\star$ sur E est un élément $e_\star$ qui vérifie :

$$(\forall x)(x \star e_\star = e_\star \star x = x)$$

Un magma associatif admettant un élément neutre est un **monoïde** (ou un magma associatif **unifère**).

Pouvez-vous démontrer que si l'élément neutre existe, il est unique ?

Définition 1 - 9

Élément symétrisable

Soit x un élément de E. Il est inversible (ou symétrisable) par $\star$ si, et seulement si,

$$(\exists y)(x \star y = y \star x = e_\star)$$

Pouvez-vous démontrer que si x admet un symétrique, alors ce symétrique est unique ?
On note souvent ce symétrique $x^{-1_\star}$ ou bien $-_\star x$.

Définition 1 - 10

Régularité

Un élément a de E est dit **régulier à gauche** (ou **simplifiable à gauche**) si, et seulement si :

$$(\forall x)\,(\forall y)\,\big((a \star x = a \star y) \to (x = y)\big)$$

On a une définition similaire de la régularité à droite.
Un élément à la fois régulier à gauche et à droite est dit **régulier**.
Une loi telle que tout élément soit régulier est dite régulière.

Un élément inversible est régulier (vérifiez-le). Un élément régulier est-il inversible ?

Définition 1 - 11

Distributivité

On dit que la loi $\star$ définie sur E est distributive sur $\dagger$ définie sur E si, et seulement si :

$$\Big(\forall x\Big)\Big(\forall y\Big)\Big(\forall z\Big)\Big(\big(x \star (y \dagger z) = (x \star y) \dagger (x \star z)\big) \wedge \big((y \dagger z) \star x = (y \star x) \dagger (z \star x)\big)\Big)$$

Remarque

Une loi de composition externe peut être distributive par rapport à une loi interne : cherchez un exemple...

Définition 1 - 12

Élément absorbant

Un élément absorbant d'une LCI $\star$ sur E est un élément $a_\star$ qui vérifie :

$$(\forall x)(x \star a_\star = a_\star \star x = a_\star)$$

Montrez que si un tel élément existe, il est unique.

Définition 1 - 13

Élément involutif
Soit $\star$ une loi sur un ensemble E admettant un élément neutre $e_\star$.
Alors x est **involutif** si, et seulement si, $x \star x = e_\star$.

Définition 1 - 14

Élément idempotent
Soit $\star$ une loi sur un ensemble E.
Alors x est **idempotent** si, et seulement si, $x \star x = x$.

3 Groupes

É.Galois (1811-1832)

Mort dans un duel à l'âge de vingt ans, Évariste Galois a tout de même eu le temps de révolutionner la mathématique et ses travaux ont eu une très grande influence sur le développement de nombreux domaines liés à l'informatique : la théorie de l'information, la spécification, la compilation,etc.
La notion de groupe puis d'anneau et de corps que nous allons voir dans ce chapitre permettent également d'organiser très efficacement notre programmation et de nous initier aux notions de polymorphisme.
Nous verrons que nous pourrons également aborder la notion d'*instantiation* de classes en Haskell et de *surcharge* d'opérateur.
Il existe également de nombreuses applications en infographie, en théorie des langages formels, de la compilation...

Définition 1 - 15

Groupe
Un groupe est un monoïde tel que tout élément est inversible.

Définition 1 - 16

Sous-groupe
$\langle H, \star\rangle$ est un sous-groupe de $\langle G, \star\rangle$ si, et seulement si, H est une partie de G et $\langle H, \star\rangle$ est un groupe.

Nous travaillerons souvent avec des groupes ayant seulement un nombre fini d'éléments :

Définition 1 - 17

Ordre d'un groupe fini
Soit G un groupe ayant un nombre fini d'éléments. Le cardinal de G est appelé l'**ordre du groupe** G.

4 Anneaux et corps

4 1 Anneaux

Définition 1 - 18

Anneau
Soit A un ensemble muni de deux lois $\boxdot$ et $\boxplus$.
On dit que $\langle A, \boxplus, \boxdot\rangle$ est un anneau si, et seulement si :

— $\langle A, \boxplus\rangle$ est un groupe commutatif ;
— $\boxdot$ est associative ;
— $\boxdot$ est distributive sur $\boxplus$.

Si $\boxdot$ est commutative, alors l'anneau est dit *commutatif*.
Si $\boxdot$ admet un élément neutre, l'anneau est dit *unitaire*.
Les éléments d'un anneau unitaire qui admettent un symétrique par $\boxdot$ sont dits *inversibles*.
On note A^* l'ensemble des éléments inversibles de A.

Avez-vous déjà rencontré un anneau dans ce cours ?
Dans un anneau, est-ce que tout élément est inversible ?

4 2 Corps

Définition 1 - 19

Corps
Soit $\mathbb{K}$ un ensemble muni de deux LCI $\boxplus$ et $\boxdot$. Le triplet $\langle \mathbb{K}, \boxplus, \boxdot \rangle$ possède une structure de **corps** si, et seulement si,

— $\langle \mathbb{K}, \boxplus, \boxdot \rangle$ a une structure d'anneau unitaire ;

— $\langle \mathbb{K} \setminus \{e_{\boxplus}\}, \boxdot \rangle$ a une structure de groupe.

Combien d'éléments a au minimum un corps ?
Quand $\langle \mathbb{K} \setminus \{e_{\boxplus}\}, \boxdot \rangle$ est un groupe abélien, le corps $\langle \mathbb{K}, \boxplus, \boxdot \rangle$ est dit commutatif.
En anglais, le terme mathématique pour désigner les corps est *field*.
Il existe un corps très important en théorie de l'information et en cryptographie : le corps $\mathbb{Z}/2\mathbb{Z}$ muni de l'addition et de la multiplication modulaire. On le note habituellement $\mathbb{F}_2$ en français et $GF(2)$ en anglais.
Vérifiez qu'il s'agit bien d'un corps muni des lois habituelles...

4 3 La classes numériques sur Haskell

La classe **Num** de **Prelude** regroupe les types numériques usuels (**Int**, **Integer**, **Float** et **Double**) mais avec des opérations basiques :

Haskell
```
class  (Eq a, Show a) => Num a  where
    (+), (-), (*)  :: a -> a -> a
    negate         :: a -> a
    abs, signum    :: a -> a
    fromInteger    :: Integer -> a
```

À quelle structure algébrique peut-on associer la classe **Num** ?
Il existe « au-dessus » une classe **Fractional** qui est définie avec les opérations suivantes :

Haskell
```
class  (Num a) => Fractional a  where
    (/)          :: a -> a -> a
    recip        :: a -> a
    fromRational :: Rational -> a
```

sachant qu'un **Rational** est, comme son nom l'indique un élément de $\mathbb{Q}$ c'est-à-dire le rapport entre deux entiers.

Haskell
```
Prelude> toRational 1.25
5 % 4
```

Mais nous n'insisterons pas sur ce type de données car nous allons...le recréer nous-mêmes !
À quelle structure algébrique peut-on associer la classe **Frational** ?
Voici enfin deux autres classes utiles :

Haskell
```
class  (Num a, Ord a) => Real a  where
    toRational ::  a -> Rational

class  (Real a, Enum a) => Integral a  where
    quot, rem, div, mod :: a -> a -> a
    quotRem, divMod     :: a -> a -> (a,a)
    toInteger :: a -> Integer

class  (Fractional a) => Floating a  where
    pi                  :: a
```

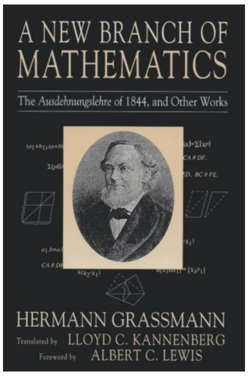
```
    exp, log, sqrt      :: a -> a
    (**), logBase       :: a -> a -> a
    sin, cos, tan       :: a -> a
    asin, acos, atan    :: a -> a
    sinh, cosh, tanh    :: a -> a
    asinh, acosh, atanh :: a -> a

class  (Real a, Fractional a) => RealFrac a  where
    properFraction   :: (Integral b) => a -> (b,a)
    truncate, round  :: (Integral b) => a -> b
    ceiling, floor   :: (Integral b) => a -> b
```

4 4 Un nouveau type numérique

Il est possible de créer un nouveau type numérique. Selon sa structure algébrique, on en fera une instance de **Num** ou de **Fractional**.

5 Structure d'espace vectoriel

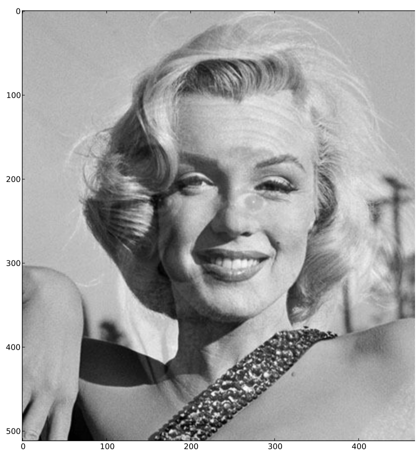
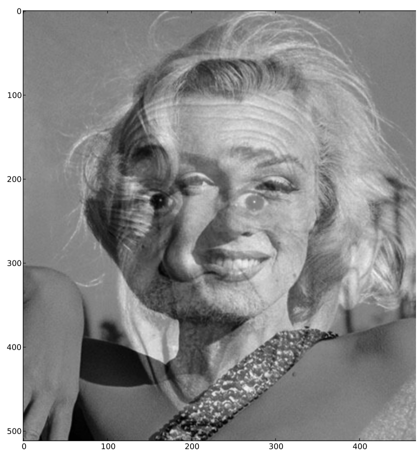
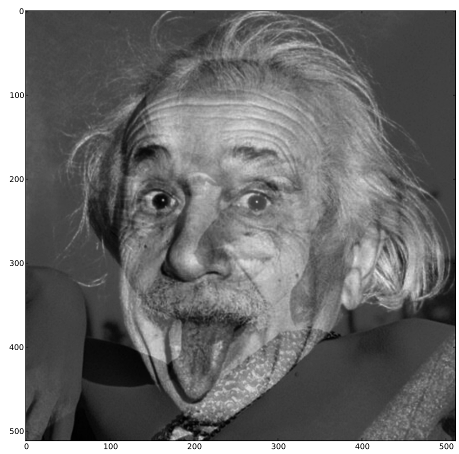
5 1 Un peu d'histoire

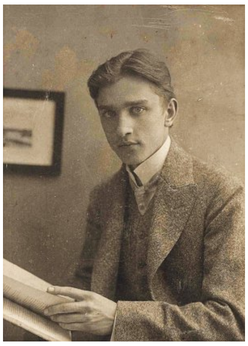


Hermann Grassmann (1809-1877) est un mathématicien allemand hors du commun. Il enseigna principalement au lycée des disciplines aussi diverses que les mathématiques, la physique, l'allemand, le latin et la théologie. Il commence à publier les fondements de ce qui deviendra la théorie des espaces vectoriels mais le rapporteur de ses travaux les lit d'un œil distrait et passe totalement à côté.
Grassmann persiste et publie en 1844 « *Die lineale Ausdehnungslehre, ein neuer Zweig der Mathematik* » (l'enseignement de l'extension linéaire, une nouvelle branche de la mathématique) qui passe encore inaperçu car Grassmann n'est pas un universitaire. Ses travaux ne seront reconnus qu'après sa mort lorsque l'italien Peano donne une présentation axiomatique des espaces vectoriels dérivée des travaux de Grassmann.
Enfin c'est un jeune polonais, Stefan Banach, qui, en 1920, propulsa les espaces vectoriels en dehors de la sphère purement géométrique et en fit un des outils majeurs de développement scientifique.

5 2 Vers une définition

Considérons le vecteur Marylin et le vecteur Albert. Voici trois nouveaux vecteurs obtenus par différentes combinaisons : $3 \times$ Marylin + Albert, Marylin + Albert et Marylin + $3 \times$ Albert



S. Banach (1892-1945)

On voit clairement apparaître la notion de produit externe : ici les vecteurs sont des images qui peuvent être additionnées et rester des images. De plus, on peut multiplier une image par un nombre et cela reste une image.
On est souvent confronté à cette situation : un groupe, un corps et un produit externe des éléments du corps sur les éléments du groupe.
On peut par exemple combiner des octets, des fonctions...et même des vecteurs du plan.
Mais il ne faut pas lier des opérations de manière trop hasardeuse : cela pourrait réserver de fâcheuses surprises. C'est pourquoi on impose des liens étroits entre les deux opérations, comme pour les corps.
Un **espace vectoriel** V **sur un corps** $\mathbb{K}$ est un groupe additif muni d'une loi de composition externe. Les éléments du corps sont appelés les **scalaires**, les éléments de V sont appelés les **vecteurs**.
Bon, soyons plus formels :

Définition 1 - 20

$\mathbb{K}$-espace vectoriel

Un $\mathbb{K}$-espace vectoriel V sur un corps commutatif $\langle \mathbb{K}, \oplus, \odot \rangle$ est un groupe abélien $\langle V, \dagger \rangle$ muni d'une loi de composition *externe* $\cdot$ de $\mathbb{K} \otimes V$ dans V vérifiant les quatre axiomes suivant, λ et μ désignant des scalaires quelconques, $\boldsymbol{u}$ et $\boldsymbol{v}$ des vecteurs quelconques et $1_\odot$ l'élément neutre de la loi $\odot$ sur $\mathbb{K}$:

$$(\lambda \oplus \mu) \cdot \boldsymbol{u} = \lambda \cdot \boldsymbol{u} \dagger \mu \cdot \boldsymbol{u} \qquad \text{(distributivité vectorielle)} ;$$
$$\lambda \cdot (\boldsymbol{u} \dagger \boldsymbol{v}) = \lambda \cdot \boldsymbol{u} \dagger \lambda \cdot \boldsymbol{v} \qquad \text{(distributivité scalaire)} ;$$
$$(\lambda \odot \mu) \cdot \boldsymbol{u} = \lambda \cdot (\mu \cdot \boldsymbol{u}) \qquad \text{(associativité)} ;$$
$$1_\odot \cdot \boldsymbol{u} = \boldsymbol{u} \qquad \text{(axiome d'identité)}$$

On dit que le corps $\mathbb{K}$ *opère* sur le groupe $\langle V, \dagger \rangle$.

On pourrait se demander pourquoi on n'a pas poussé V jusqu'à avoir une multiplication, tant qu'à faire....Mouais...mais s'il est naturel d'additionner des images en leur affectant des coefficients, il est moins naturel d'imaginer une multiplication d'images (et c'est même mathématiquement compromis).
Il est donc plus naturel de se contenter d'un groupe.
Souvent sur papier (mais pas sur machine :-)), par paresse, on omet les symboles de produit externe et de produit interne, mais cela peut prêter à confusion, surtout lors de notre phase d'apprentissage, donc nous n'en ferons rien.

Remarque

Pour obtenir les combinaisons d'images précédentes, nous avons utilisé **NumPy** et la syntaxe est très orientée espace vectoriel :

Python
```
imshow(3 * marilyn + einstein, cmap = cm.gray)
```

5 3 Un exemple de $\mathbf{F}_2$-espace vectoriel

On considère l'ensemble T des chaînes de 3 bits muni de l'addition canonique :

$$\langle b_1, b_2, b_3 \rangle \dagger \langle b'_1, b'_2, b'_3 \rangle = \langle b_1 \oplus b'_1, b_2 \oplus b'_2, b_3 \oplus b'_3 \rangle$$

avec $\oplus$ l'addition sur $\mathbb{F}_2$:

$$0 \oplus 0 = 1 \oplus 1 = 0 \quad \text{et} \quad 0 \oplus 1 = 1 \oplus 0 = 1$$

On muni $\mathbb{F}_2$ de sa multiplication canonique :

$$0 \odot 0 = 0 \odot 1 = 1 \odot 0 = 0 \quad \text{et} \quad 1 \odot 1 = 1$$

Ainsi $\langle \mathbb{F}_2, \oplus, \odot \rangle$ est un corps, $\langle T, \dagger \rangle$ est un groupe abélien.
Le produit externe est :

$$(\cdot) : \begin{array}{ccc} \mathbb{F}_2 \otimes T & \to & T \\ \langle \lambda, \langle b_1, b_2, b_3 \rangle \rangle & \mapsto & \langle \lambda \odot b_1, \lambda \odot b_2, \lambda \odot b_3 \rangle \end{array}$$

Il reste à vérifier les quatre axiomes (faites-le...)
Sur Haskell :

Haskell
```haskell
data BoolInt = O | I deriving(Show,Read,Eq)

type BitChaine = [BoolInt]

bPlus :: BoolInt -> BoolInt -> BoolInt
bPlus    b1         b2
  |b1 == b2                = O
  |otherwise               = I

bFois :: BoolInt -> BoolInt -> BoolInt
bFois    I          I       =  I
bFois    _          _       =  O

cPlus :: BitChaine -> BitChaine -> BitChaine
cPlus    c1           c2         =  zipWith bPlus c1 c2

prodExt :: BoolInt -> BitChaine -> BitChaine
prodExt    b          c          =  map (\x -> bFois b x) c
```

donne par exemple :

Haskell
```haskell
*Main> bPlus O I
I
*Main> cPlus [O,O,I] [I,O,I]
[I,O,O]
*Main> prodExt I [O,O,I]
[O,O,I]
*Main> prodExt O [O,O,I]
[O,O,O]
```

On peut alors tester l'axiome de distributivité scalaire par exemple :

Haskell
```haskell
lesBits = [O,I]

lesTrios = [[x,y,z] | x <- lesBits, y <- lesBits, z <- lesBits]

and [prodExt (bPlus b1 b2) t == cPlus (prodExt b1 t) (prodExt b2 t)
    | b1 <- lesBits, b2 <- lesBits, t <- lesTrios ]
```

5 4 Ensemble des combinaisons linéaires d'une famille de vecteurs

Dans un espace vectoriel, les opérations « linéaires » sont l'addition du groupe et le produit externe.

Définition 1 - 21

Combinaison linéaire de vecteurs
Soit $\langle \mathbb{K}, \oplus, \odot \rangle$ un corps opérant sur le groupe $\langle V, \dagger \rangle$ pour former un espace vectoriel de produit externe $\bullet$.
Soit $\langle \boldsymbol{u_0}, \boldsymbol{u_1}, \cdots, \boldsymbol{u_p} \rangle$ une famille de vecteurs de V.
Un vecteur $\boldsymbol{v}$ de V est une combinaison linéaire de la famille des $\langle \boldsymbol{u_i} \rangle_{0 \leqslant i \leqslant p}$ si, et seulement si, il existe une famille $\langle \lambda_0, \lambda_1, \cdots, \lambda_p \rangle$ de scalaires de $\mathbb{K}$ vérifiant :

$$\boldsymbol{v} = \mathop{\dagger}_{i=0}^{i=p} \lambda_i \bullet \boldsymbol{u_i} = \lambda_0 \bullet \boldsymbol{u_0} \dagger \lambda_1 \bullet \boldsymbol{u_1} \dagger \cdots \dagger \lambda_p \bullet \boldsymbol{u_p}$$

L'ensemble de ces combinaisons linéaires est noté :

$$\mathcal{V}\mathrm{ect}\{\boldsymbol{u_0}, \boldsymbol{u_1}, \cdots, \boldsymbol{u_p}\}$$

Par exemple, prenons le $\mathbb{R}$-espace vectoriel $\mathbb{R}^2$ muni de la somme et du produit externe canonique. On peut dire que $\boldsymbol{v} = \langle 4, 3\rangle$ est une combinaison linéaire de $\boldsymbol{u_1} = \langle 2, 1\rangle$ et $\boldsymbol{u_2} = \langle 0, 3\rangle$:

$$\boldsymbol{v} = 2\boldsymbol{u_1} + \frac{1}{3}\boldsymbol{u_2}$$

On peut alors légitimement se demander ce que représente l'ensemble des combinaisons linéaires d'une famille fixée de vecteurs.
Par exemple, travaillons dans $\mathbb{F}_2^3$ en tant que $\mathbb{F}_2$-espace vectoriel muni des opérations canoniques. Quelles sont toutes les combinaisons linéaires des vecteurs $\langle 0, 1, 1\rangle$ et $\langle 1, 0, 1\rangle$?
Vérifiez les résultats donnés par Haskell avec les fonctions introduites précédemment :

Haskell

```
u = [O,I,I]
v = [I,O,I]
vect = [cPlus (prodExt b1 u) (prodExt b2 v) | b1 <- lesBits, b2 <- lesBits]
```

Donne :

Haskell

```
*Main> vect
[[O,O,O],[I,O,I],[O,I,I],[I,I,O]]
```

5 5 Sous-espace vectoriel

Définition 1 - 22

Sev
Soit $\langle \mathbb{K}, \oplus, \odot\rangle$ un corps opérant sur le groupe $\langle V, \dagger\rangle$ pour former un espace vectoriel de produit externe $\bullet$.
Soit W un sous-ensemble de V.
Si, muni des mêmes opérations que V, W a une structure de $\mathbb{K}$-espace vectoriel, alors on dit que W est un **sous-espace vectoriel** de V.

Il serait bête de vérifier tous les critères pour un sous-espace. Démontrez les caractérisations suivantes :

Théorème 1 - 1

Caractérisation (1) des sous-espaces vectoriels
Soit $\langle \mathbb{K}, \oplus, \odot\rangle$ un corps opérant sur le groupe $\langle V, \dagger\rangle$ pour former un espace vectoriel de produit externe $\bullet$.
Soit W un sous-ensemble de V.
W est un sous-espace vectoriel (sev) de V si, et seulement si :

— $W \neq \emptyset$
— $(\forall \boldsymbol{u})(\forall \boldsymbol{v})(\langle \boldsymbol{u}, \boldsymbol{v}\rangle \in W^2 \to \boldsymbol{u} \dagger \boldsymbol{v} \in W)$
— $(\forall \lambda)(\forall \boldsymbol{u})(\langle \lambda, \boldsymbol{u}\rangle \in \mathbb{K} \otimes W \to \lambda \bullet \boldsymbol{u} \in W)$

Démontrez qu'on peut être plus synthétique :

Théorème 1 - 2

Caractérisation (2) des sous-espaces vectoriels
Soit $\langle \mathbb{K}, \oplus, \odot\rangle$ un corps opérant sur le groupe $\langle V, \dagger\rangle$ pour former un espace vectoriel de produit externe $\bullet$.
Soit W un sous-ensemble de V.
W est un sous-espace vectoriel (sev) de V si, et seulement si :

— $\mathbf{0} \in W$
— $(\forall \lambda)(\forall \boldsymbol{u})(\forall \boldsymbol{v})(\langle \lambda, \boldsymbol{u}, \boldsymbol{v}\rangle \in \mathbb{K} \otimes W \otimes W \to \boldsymbol{u} \dagger \lambda \bullet \boldsymbol{v} \in W)$

EXERCICES

LCI

Exercice 1 - 1 Have you got the power ?

On considère la LCI $\star$ définie sur $\mathbb{Z}$ par :

$$a \star b = a - b$$

L'écriture a^3 a-t-elle un sens ? nQuel est le sens de cette question ?

Exercice 1 - 2 Have you got the power now ?

On considère la LCI $\star$ définie sur $\mathbb{Z}$ par :

$$a \star b = \underbrace{a \times a \times \cdots \times a}_{b \text{ facteurs}}$$

L'écriture a^3 a-t-elle un sens ?

Exercice 1 - 3 And now ?

On considère la LCI définie sur $\{a, b, c, d\}$ par :

$\star$	a	b	c	d
a	b	c	d	a
b	d	a	b	c
c	a	b	c	d
d	c	d	a	b

L'écriture a^3 a-t-elle un sens ?

Exercice 1 - 4 Avec Haskell

On peut facilement construire une classe définissant les magmas : c'est un type (qui correspond pour nous à un ensemble) qu'on notera de manière générique **`t`**, muni d'une loi qu'on notera de manière générique **`<+>`**.

Haskell
```haskell
class Magma t where
  (<+>) :: t -> t -> t
```

Cela traduit qu'un ensemble de variables de type **`t`** est un magma quand il est muni d'une LCI (qui est donc de signature **`t -> t -> t`**)
On peut alors faire de **`Integer`** une instance de notre classe **`Magma`** en prenant comme loi l'addition des entiers : on remplace le type inconnu **`t`** par **`Integer`** et on définit la LCI générique **`<+>`** comme étant la classique addition des entiers :

Haskell
```haskell
instance Magma Integer where
  (<+>) = (+)
```

On peut de même faire de **`String`** une instance de **`Magma`** en prenant cette fois comme LCI la concaténation des chaînes qui est **`++`** sur Haskell.

Haskell
```haskell
instance Magma String where
  (<+>) = (++)
```

...mais on se fait insulter par le compilateur ! Il faut en effet passer une option au compilateur pour qu'il accepte les instances à partir de types comme **String**.
Il faut donc rajouter **sur la première ligne de votre fichier** la ligne suivante :

Haskell
```haskell
{-# LANGUAGE TypeSynonymInstances, FlexibleInstances #-}
```

Pour que notre ensemble muni de sa loi soit un magma associatif, il faut et il suffit que cette loi soit associative. On crée donc un test :

Haskell
```haskell
isMagmaAssociatif :: (Magma t, Eq t) => (t,t,t) -> Bool
isMagmaAssociatif                       (a,b,c) = (a <+> b) <+> c == a <+> (b <+> c)
```

Comment fait un mathématicien pour vérifier que **Integer** muni de la somme **+** a une structure de magma associatif ?
Il démontre que la loi est associative.
Que fait un informaticien qui a dormi en cours ? Il fait des tests.
Il existe sur Haskell un module **QuickChek** qui va nous aider ici :

Haskell
```haskell
import Test.QuickCheck
```

La documentation est là :
http://hackage.haskell.org/package/QuickCheck-2.1.0.1/docs/Test-QuickCheck.html
Écrivons notre test (**quickCheck** qui fait 100 tests sur des triplets aléatoirement choisis selon une loi uniforme) :

Haskell
```haskell
test1 = quickCheck (isMagmaAssociatif :: (Integer,Integer,Integer) -> Bool)
```

C'est bon pour 100, c'est bon pour tous dit un apprenti informaticien :

Haskell
```
*Main> test1
+++ OK, passed 100 tests.
```

Pour tester l'associativité du magma définit sur les **String**, on change la signature :

Haskell
```haskell
test2 = quickCheck (isMagmaAssociatif :: (String,String,String) -> Bool)
```

Mais, comme le disait Edsger Dijkstra (1930 - 2002), prix Turing 1972 :

> *Program testing can be used to show the presence of bugs, but never to show their absence !*
>
> in « Notes On Structured Programming, corollary at the end of section 3, On The Reliability of Mechanisms » (EWD249).

Recherche

Maintenant, faites un **quickCheck** sur les exercices Exercice 1 - 1 et Exercice 1 - 2 page ci-contre.

Pour l'exercice Exercice 1 - 3 page précédente, on ne va pas utiliser **quickCheck** car le type est inconnu.
On va donc construire les tests nous-mêmes.
Définissons notre loi et son ensemble :

Haskell
```haskell
data Truc = A | B | C | D deriving(Show,Eq)

lesTrucs = [A,B,C,D]

plus :: Truc -> Truc -> Truc
plus A A = A
plus A B = C
plus A C = D
```

```
plus A D = A
plus B A = D
plus B B = A
plus B C = B
plus B D = C
plus C A = A
plus C B = B
plus C C = C
plus C D = D
plus D A = C
plus D B = D
plus D C = A
plus D D = B

instance Magma Truc where
  (<+>) = plus
```

Construisons tous les triplets d'éléments de **Truc** à l'aide d'une liste par compréhension :

Haskell
```
lesTripletsTrucs = [(a,b,c) | a <- lesTrucs, b <- lesTrucs, c <- lesTrucs]
```

Maintenant, on peut tester toutes les possibilités :

Haskell
```
testa = all isMagmaAssociatif  lesTripletsTrucs
```

Argh :

Haskell
```
*Main> testa
False
```

Quel est le premier cas qui pose problème ? Quels sont les triplets qui ne posent pas de problème ?

Pour répondre à ces questions, nous allons construire une fonction **filtre** et une fonction **trouve**.

La première va filtrer une liste selon un prédicat. Sa signature sera donc :

Haskell
```
filtre :: (a -> Bool) -> [a] -> [a]
```

Recherche

Expliquez cette signature.
Expliquez ce que fait cette fonction et comment elle le fait :

Haskell
```
filtre    foncFiltre    []  =  []
filtre    foncFiltre   (t:q) = if (foncFiltre t)
                               then  t : (filtre foncFiltre q)
                               else      filtre foncFiltre q
```

Créez alors une fonction **trouve** qui renvoie le premier élément d'une liste qui vérifie un prédicat donné en argument.
Appliquez ces fonctions à nos **Trucs**.
Créez à présent un test **isMagmaSimplifiableGauche** qui teste si un triplet vérifie la propriété :

$$a \star b = a \star c \to b = c$$

Vous aurez sûrement du ET logique **&&**, du OU inclusif **||**, de la négation de l'égalité **/=**.
Créez de même un **isMagmaSimplifiableDroite** et un **isMagmaSimplifiable**.
Appliquez ce test aux **Trucs**.
Créez une fonction **isNeutre e liste** qui teste si **e** est neutre vis-à-vis de tous les éléments de la liste **liste**.
Créez enfin une liste **trouveNeutre liste** qui trouve un éventuel élément neutre dans la liste **liste**.

On peut maintenant créer un type correspondant au corps $\mathbb{F}_2$ qui est la base de travail en théorie de l'information et en faire une instance de **Magma** en lui associant une loi :

Haskell
```
data Z2 = O | I deriving(Eq,Show)

fois2 :: Z2 -> Z2 -> Z2
fois2 I I = I
fois2 _ _ = O
```

Recherche

Est-ce que ce magma est associatif ? Y a-t-il un élément neutre ? Y a-t-il des éléments simplifiables ? Symétrisables ?
Faites de même avec $\mathbb{Z}/_{4\mathbb{Z}}$.

Exercice 1 - 5

On considère l'opération définie sur $\mathbb{Z}$ par :

$$a \top b = a + 2b$$

Cette opération est-elle une LCI ? Est-elle commutative ? Associative ? Y a-t-il un élément neutre ? Y a-t-il des éléments simplifiables ? Symétrisables ?
Mêmes questions avec la loi définie sur $\mathbb{Q}$ par $a \dagger b = (a+b)/2$.

Exercice 1 - 6

On considère l'opération $\uparrow$ définie sur $\mathbb{N}_1$ par :

$$a \uparrow 1 = a, \quad a \uparrow b = a \times (a \uparrow (b-1))$$

Cette opération vous rappelle-t-elle quelque chose ? Est-ce une LCI ? Est-elle commutative ? Associative ? etc.
Cherchez un élément régulier à droite mais pas à gauche.

Exercice 1 - 7

On considère l'opération de concaténation de listes dont l'opérateur sur Haskell est **++**. Est-elle commutative ? Associative ?

Exercice 1 - 8

A, B et C étant des ensembles, que pensez-vous de $A \cap B \cup C$?

Exercice 1 - 9

Que pensez-vous de la soustraction des entiers ? Du produit cartésien des ensembles ? De la composition des relations ? De la multiplication d'un vecteur du plan par un réel ? etc.

Exercice 1 - 10

On définit une opération sur les quadruplets de nombres réels :

$$\langle a, b, c, d\rangle \odot \langle \alpha, \beta, \gamma, \delta\rangle = \langle a\alpha + b\gamma, a\beta + b\delta, c\alpha + d\gamma, c\beta + d\delta\rangle$$

Est-elle commutative ? Associative ?

Exercice 1 - 11

Donnez des exemples de couples $\langle E, \star\rangle$ avec $\star$ une application définie sur $E \otimes E$ telle que $\langle E, \star\rangle$ ne soit pas un magma.

Exercice 1 - 12

Soit $A = \{a_1, a_2, ..., a_p\}$ un ensemble de caractères. On note $A^\star$ l'ensemble des chaînes de caractères construites à partir des caractères de A.

1. $A^\star$ muni de la concaténation des chaînes a-t-il une structure de magma associatif ? de monoïde ?
2. Les ensembles suivants munis de la concaténation sont-ils des magmas associatifs ? Des monoïdes ?
   i. L'ensemble des chaînes de longueur paire ;
   ii. L'ensemble des chaînes de longueur impaire ;
   iii. $\{(a_1a_2)^n \quad | \quad n \in \mathbb{N}\}$;
   iv. $\{a_1^n a_2^n \quad | \quad n \in \mathbb{N}\}$;
   v. L'ensemble des chaînes ne contenant que a_1 et a_2, ceux-ci apparaissant un nombre égal de fois.

Exercice 1 - 13

Soit $\mathbb{B}_2 = \{0, 1\}$ l'ensemble des booléens. Est-ce que $\mathbb{B}_2$ muni de NAND est un monoïde ? Muni de NOR ?

Exercice 1 - 14

Soit l'opération $\top$ définie par $x \top y = \sqrt{x^2 + y^2}$. Est-ce que $\langle \mathbb{R}, \top\rangle$ est un magma associatif ? Un monoïde ? Un monoïde simplifiable ? Y a-t-il un élément absorbant ?

Exercice 1 - 15

Soit une fonction $\mathcal{F}(E)$ l'ensemble des fonctions de E dans E.
On dit que f est inversible à gauche s'il existe une fonction g telle que $g \circ f = \mathrm{Id}_E$.
Montrez que f inversible à gauche si, et seulement si, f est injective puis que f est inversible à droite si, et seulement si, f est surjective.

Exercice 1 - 16 **Curryfication, parenthèse et associativité**

On considère une fonction

$$f : \begin{array}{ccc} E \otimes F & \to & G \\ \langle x, y\rangle & \mapsto & f(\langle x, y\rangle) \end{array}$$

Pour chaque x de E, on peut donc définir une fonction f_x :

$$f_x : \begin{array}{ccc} F & \to & G \\ y & \mapsto & f(\langle x, y\rangle) \end{array}$$

Que pouvez-vous dire de la fonction : $\varphi : x \mapsto f_x$ (domaine, codomaine,lien avec f) ?
Pouvez-vous, inversement, associer une fonction f à une fonction du type φ ?
Notons C la fonction qui à une fonction f de $E \otimes F$ dans G associe la fonction φ selon le mécanisme précédent.
Quelle est la signature de C ? Quelle propriété intéressante possède C concernant notre utilisation de Haskell ?
On dit que $C(f)$ est la fonction currifiée de f
Ici, vous entamez un débat avec vos camarades et votre professeur au sujet de ces problèmes et vous les liez à votre cours :)
Notez pour finir que Haskell a tout prévu...

Haskell

```
Prelude> :t curry
curry :: ((a, b) -> c) -> a -> b -> c
Prelude> :t uncurry
uncurry :: (a -> b -> c) -> (a, b) -> c
```

Exercice 1 - 17 Notation polonaise

En 1920, le mathématicien-logicien-philosophe polonais Jan ŁUKASIEWICZ invente la notation préfixée alors qu'il est ministre de l'éducation (on a le droit de rêver...).
35 ans plus tard, le philosophe et informaticien (!) australien (! ! !) Charles HAMBLIN (1922 - 1985) s'en inspire.
Il a en effet en sa possession l'un des deux ordinateurs présents à l'époque en Australie. Il se rend compte que la saisie de calculs avec les opérateurs en notation infixée induit de nombreuses erreurs de saisie par oubli de parenthèses. Il pense aussi à la (très petite) mémoire des ordinateurs de l'époque.

Il trouve alors son inspiration dans le travail de ŁUKASIEWICZ pour éviter les parenthèses et il pense à mettre les opérateurs en position préfixée : ainsi il introduit la notion de *pile* (ou *stack* ou LIFO) qui économise le nombre d'adresses mémoire nécessaire.
C'est la naissance de la Notation Polonaise Inversée (NPI ou RPN). Elle économise également la taille des composants électroniques des portes logiques.
Son seul inconvénient : les mauvaises habitudes prises d'utiliser des notations infixées.
Un autre grand avantage : il faut comprendre le calcul avant de l'exécuter :)
Ouvrez emacs et tapez **`M-x calc RET`**. Vous pouvez alors faire n'importe quel calcul arithmétique en voyant chaque résultat intermédiaire et sans parenthèse.

Par exemple, la séquence **`2 3 4 + *`** donne : :

```
--- Emacs Calculator Mode ---
3:  2
2:  3
1:  4
--- Emacs Calculator Mode ---
2:  2
1:  7
--- Emacs Calculator Mode ---
1:  14
```

Mais **`2 3 + 4 *`** donne :

```
--- Emacs Calculator Mode ---
2:  2
1:  3
--- Emacs Calculator Mode ---
2:  5
1:  4
--- Emacs Calculator Mode ---
1:  20
    .
```

Que donne **`10 4 3 + 2 * -`** ?
Entraînez-vous à passer de la notation infixe à la NPI et vice-versa. Donnez vos exemples à votre voisin(ne) à titre d'exercice.
En Haskell, une machine à calculer fonctionnant en NPI se programme en 5 lignes comme vous pourrez le voir sur le tutoriel « apprendre Haskell vous fera le plus grand bien »...

Exercice 1 - 18 En-tête IPV4

Voici un en-tête IPV4 :
`4500 003c 1c46 4000 4006 b1e6 ac10 0a63 ac10 0a0c`
Voici un extrait de l'article de Wikipedia consacré au protocole IPV4 :

> *Pour reconstituer l'information, on utilise les champs suivants dans l'en-tête IPv4 :*
>
> **Version (4 bits)** *: version d'IP utilisée. Ici, 4.*
>
> **Longueur de l'en-tête ou IHL (pour Internet Header Length) (4 bits)** *: nombre de mots de 32 bits, soit 4 octets (ou nombre de lignes du schéma). La valeur est comprise entre 5 et 15, car il y a 20 octets minimum et on ne peut dépasser 40 octets d'option (soit en tout, 60 octets).*
>
> **Type de service ou ToS (pour Type of Service) (8 bits)** *: ce champ permet de distinguer différentes qualité de service différenciant la manière dont les paquets sont traités. Composé de 3 bits de priorité (donc 8 niveaux) et trois indicateurs permettant de différencier le débit, le délai ou la fiabilité.*
>
> **Longueur totale en octets ou Total Length (16 bits)** *: nombre total d'octets du datagramme, en-tête IP comprise. Donc, la valeur maximale est (216)-1 octets.*

Identification (16 bits) : *numéro permettant d'identifier les fragments d'un même paquet.*

Indicateurs ou Flags (3 bits) :

1. *actuellement inutilisé.*
2. *DF (Don't Fragment) : lorsque ce bit est positionné à 1, il indique que le paquet ne peut pas être fragmenté. Si le routeur ne peut acheminer ce paquet (taille du paquet supérieure à la MTU), il est alors rejeté.*
3. *MF (More Fragments) : quand ce bit est positionné à 1, on sait que ce paquet est un fragment de données et que d'autres doivent suivre. Quand il est à 0, soit le fragment est le dernier, soit le paquet n'a pas été fragmenté.*

Fragment offset (13 bits) : *position du fragment par rapport au paquet de départ, en nombre de mots de 8 octets.*

Durée de vie ou TTL (pour Time To Live) (8 bits) : *initialisé par l'émetteur, ce champ est décrémenté d'une unité généralement à chaque saut de routeur. Quand TTL = 0, le paquet est abandonné et un message ICMP est envoyé à l'émetteur pour information.*

Protocole (8 bits) : *numéro du protocole au-dessus de la couche réseau : TCP = 6, UDP = 17, ICMP = 1.*

Somme de contrôle de l'en-tête ou Header Checksum (16 bits) : *complément à un de la somme complémentée à un de tout le contenu de l'en-tête afin de détecter les erreurs de transfert. Si la somme de contrôle est invalide, le paquet est abandonné sans message d'erreur.*

Adresse source (32 bits) : *adresse IP de l'émetteur sur 32 bits.*

Adresse destination (32 bits) : *adresse IP du récepteur 32 bits.*

Options (0 à 40 octets par mots de 4 octets) : *facultatif.*

Remplissage ou Padding : *de taille variable comprise entre 0 et 7 bits. Il permet de combler le champ option afin d'obtenir un en-tête IP multiple de 32 bits. La valeur des bits de bourrage est 0.*

Créez sur Haskell une fonction qui vérifie la somme de contrôle.
Par exemple :

Haskell

```
Ok, modules loaded: IPV4.
*IPV4> verifIPheader ["4500","003c","1c46","4000","4006","b1e6","ac10","0a63","ac10","0a0c"]
True
```

Vous pourrez utiliser au choix les fonction **`intToDigit`** et **`digitToInt`** du module **`Data.Char`**.

Groupes

Exercice 1 - 19

1. $\mathbb{B}_2$ muni des opérations suivantes est-il un groupe :
   i. $\wedge$? ii. $\vee$? iii. $\oplus$?
2. Résoudre dans $\mathbb{B}_2$ l'équation $a \oplus x = b$.

Exercice 1 - 20

Soit $\langle G, \star\rangle$ un groupe et a et b deux éléments de G. Exprimez l'inverse de $a \star b$ en fonction des inverses de a et de b.

Exercice 1 - 21

Les tables ci-dessous définies sur $E = \{a, b, c\}$ définissent-elles des magmas associatifs ? des monoïdes ?

$\star$	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

$\perp$	a	b	c
a	a	c	c
b	b	c	a
c	c	a	a

Notons $A = \{a, b, c\}$. Montrez que $\langle A, \star\rangle$ a une structure de groupe. Comparer ce groupe avec

— $\langle \mathbb{Z}/3\mathbb{Z}, \mp\rangle$
— $\left\langle \{1, \mathrm{e}^{2i\pi/3}, \mathrm{e}^{-2i\pi/3}\}, \times \right\rangle$.

Exercice 1 - 22 Isomorphisme

Montrez que $\langle \mathbb{F}_2, +\rangle$ et $\langle \mathbb{Z}/_{4\mathbb{Z}}{}^{\star}, \cdot\rangle$ sont isomorphes.
Est-ce que $\langle \mathbb{Z}/_{4\mathbb{Z}}, +\rangle$ et $\langle \mathbb{Z}/_{5\mathbb{Z}}{}^{\star}, \cdot\rangle$ sont isomorphes ?
Est-ce que $\langle \mathbb{Z}/_{4\mathbb{Z}}, +\rangle$ et $\langle \mathbb{Z}/_{8\mathbb{Z}}{}^{\star}, \cdot\rangle$ sont isomorphes ?
Est-ce que $\langle \mathbb{F}_2 \otimes \mathbb{F}_2, +\rangle$ et $\langle \mathbb{Z}/_{8\mathbb{Z}}{}^{\star}, \cdot\rangle$ sont isomorphes ?

Exercice 1 - 23 Avec Haskell

Comme nous l'avons fait avec les magmas et les monoïdes, on peut créer une classe `Groupe` dérivant de `Monoide` et créer les tests qui vont bien : `Just` do it !

Exercice 1 - 24 Déplacement à partir du pavé numérique

À chaque touche du pavé numérique, on fait correspondre un déplacement élémentaire d'un point sur l'écran : 6 pour un déplacement unitaire à droite, 2 pour un déplacement unitaire vers le bas, etc.
On note D l'ensemble de ces dix déplacements et $D^{\star}$ l'ensemble des séquences de déplacements.
Définir D^{*} comme un monoïde. Montrez que c'est un groupe abélien.
Que se passe-t-il si au lieu de déplacements on considère la trace laissée par ces déplacements ?

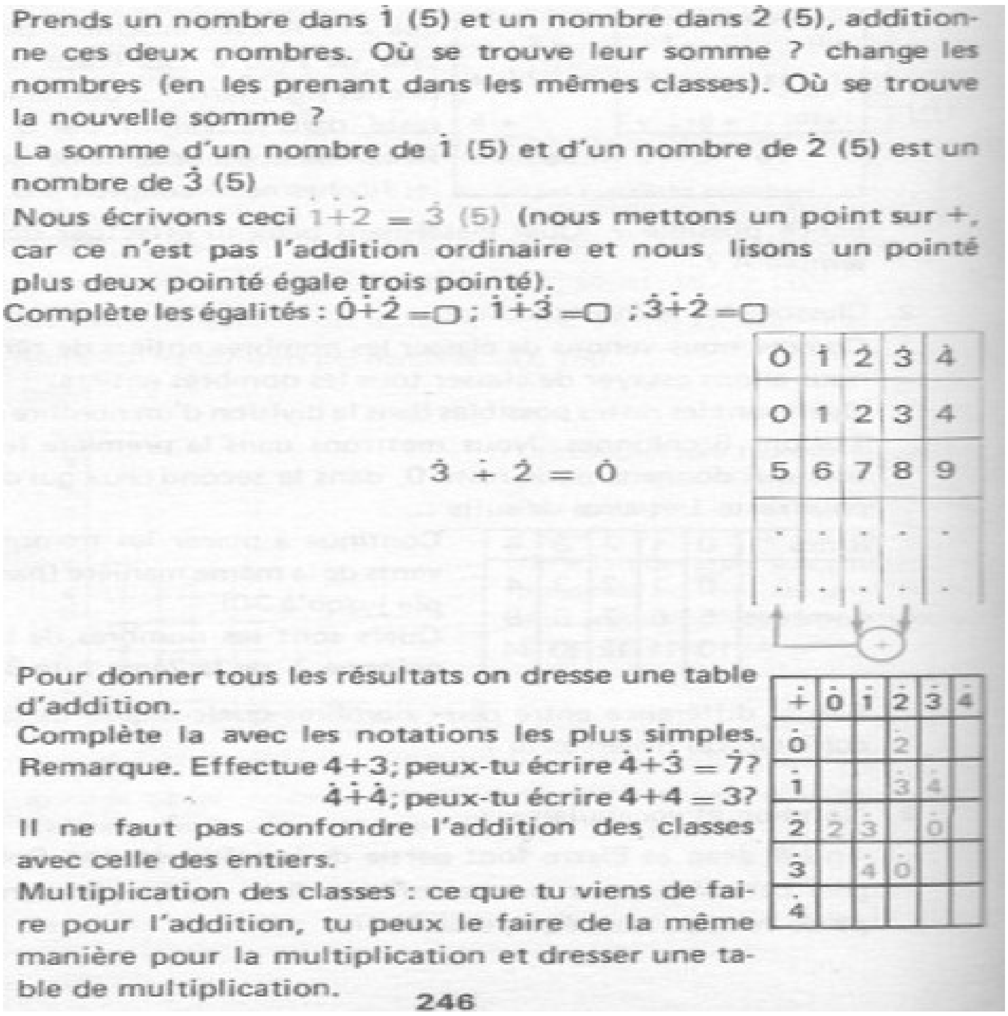
Exercice 1 - 25 Caractérisation d'un sous-groupe

La programmation, c'est l'efficacité... La définition d'un sous-groupe n'est pas efficace : pouvez-vous en donner une caractérisation plus synthétique ?

Exercice 1 - 26

Est-ce que l'ensemble $\left\{x + y\sqrt{2} \mid (x, y) \in \mathbb{Q}^2\right\}$ muni de la multiplication des réels a une structure de groupe ?

Exercice 1 - 27 Du CM1 à la 6e :-)



Prends un nombre dans $\dot{1}$ (5) et un nombre dans $\dot{2}$ (5), additionne ces deux nombres. Où se trouve leur somme ? change les nombres (en les prenant dans les mêmes classes). Où se trouve la nouvelle somme ?
La somme d'un nombre de $\dot{1}$ (5) et d'un nombre de $\dot{2}$ (5) est un nombre de $\dot{3}$ (5)
Nous écrivons ceci $\dot{1} \dot{+} \dot{2} = \dot{3}$ (5) (nous mettons un point sur +, car ce n'est pas l'addition ordinaire et nous lisons un pointé plus deux pointé égale trois pointé).
Complète les égalités : $\dot{0} \dot{+} \dot{2} = \square$; $\dot{1} \dot{+} \dot{3} = \square$; $\dot{3} \dot{+} \dot{2} = \square$

$$\dot{3} \dot{+} \dot{2} = \dot{0}$$

$\dot{0}$	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$
0	1	2	3	4
5	6	7	8	9
.	.	.	.	.
.	.	.	.	.

Pour donner tous les résultats on dresse une table d'addition.
Complète la avec les notations les plus simples.
Remarque. Effectue 4+3 ; peux-tu écrire $\dot{4} \dot{+} \dot{3} = \dot{7}$?
$\dot{4} \dot{+} \dot{4}$; peux-tu écrire $4 + 4 = 3$?
Il ne faut pas confondre l'addition des classes avec celle des entiers.
Multiplication des classes : ce que tu viens de faire pour l'addition, tu peux le faire de la même manière pour la multiplication et dresser une table de multiplication.

$\dot{+}$	$\dot{0}$	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$
$\dot{0}$			$\dot{2}$		
$\dot{1}$			$\dot{3}$	$\dot{4}$	
$\dot{2}$	$\dot{2}$	$\dot{3}$		$\dot{0}$	
$\dot{3}$		$\dot{4}$	$\dot{0}$		
$\dot{4}$					

246

Adaptez le texte précédent aux notations introduites lors de l'étude des classes d'équivalence..
Comment généraliser les résultats en terme de groupe ? (Attention, on passe maintenant au programme de 6e des années 1970...)
Nous en aurons besoin pour implémenter le calcul modulaire en Haskell....

Exercice 1 - 28 Ordre d'un élément dans un groupe fini

1. Comment justifier que pour tout élément g d'un groupe fini $(G,\cdot)$, il existe un plus petit entier n tel que $g^n = 1_G$? On appelle n l'ordre de g.
2. Déterminez l'ordre des éléments de $(\mathbb{Z}/_{9\mathbb{Z}})^*$.
3. Calculez $2^{12000000000030000000006000000020000040000020}[7]$.

Exercice 1 - 29 Permutations

Définition 1 - 23 **Permutation**
Soit E un ensemble. Une permutation de E est une application bijective de E sur E. On note $\mathfrak{S}(E)$ l'ensemble des permutations de E.

Généralement, on note les permutations sous forme d'une matrice où la première ligne correspond aux éléments de E et où la deuxième ligne correspond aux images des éléments de E.
Par exemple :

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 2 & 5 & 4 \end{pmatrix}$$

est une permutation de $E = \{1,2,3,4,5\}$.

Définition 1 - 24 Soit $n \in \mathbb{N}$, alors $\mathfrak{S}_n$ désigne l'ensemble des permutations de $\{1,2,3,\ldots,n\}$.

Théorème 1 - 3 **Groupe des permutations**
L'ensemble $\mathfrak{S}(E)$ des permutations sur un ensemble E, muni de la loi $\circ$ de composition des applications, a une structure de groupe.

Théorème 1 - 4 **Nombre de permutations**
Le groupe $\mathfrak{S}_n$ est d'ordre $n!$.

1. Démontrez le théorème 1 - 3.
2. Démontrez le théorème 1 - 4. Vous pourrez procéder par récurrence. Pour l'hérédité, distinguez les permutations qui envoient 1 sur un élément quelconque x.
3. Décrivez explicitement les groupes $\mathfrak{S}_2$ et $\mathfrak{S}_3$.
4. En cryptographie, on travaillera sur l'alphabet $E = \{0,1\}^n$ et on effectuera souvent des *permutations de bits* : seuls les positions des bits sont permutées.
   On formalise ces permutations ainsi :

$$f: \begin{array}{ccc} \{0,1\}^n & \to & \{0,1\}^n \\ b_1 b_2 \cdots b_n & \mapsto & b_{\pi(1)} b_{\pi(2)} \cdots b_{\pi(n)} \end{array}$$

   avec $\pi \in \mathfrak{S}_n$.
   Une *permutation circulaire gauche* « décale » les éléments d'un nombre fixé de « rangs ». Par exemple, voici une permutation circulaire gauche :

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 1 & 2 \end{pmatrix}$$

Décrire $\pi(k)$ dans le cas d'une permutation circulaire de bits de i rangs vers la gauche.

On définit de même des permutations circulaires droite.

Combien y a-t-il de permutations circulaires dans $\mathfrak{S}_n$?

Exercice 1 - 30 Hamilton et les couples

Le mathématicien irlandais William Rowan HAMILTON (1805-1865) publie en 1837 *Theory of Conjugate Functions, or Algebraic Couples* : (http://www.maths.tcd.ie/pub/HistMath/People/Hamilton/PureTime/PureTime.pdf).
En voici un extrait :

On the Addition, Substraction, Multiplication, and Division, of Number-Couples, as combined with each other.
6. Proceeding to operations upon number-couples, considered in combination with each other, it is easy now to see the reasonableness of the following definitions, and even their necessity, if we would preserve in the simplest way, the analogy of the theory of couples to the theory of singles :

$$\langle b_1, b_2\rangle + \langle a_1, a_2\rangle = \langle b_1 + a_1, b_2 + a_2\rangle; \tag{52.}$$

$$\langle b_1, b_2\rangle - \langle a_1, a_2\rangle = \langle b_1 - a_1, b_2 - a_2\rangle; \tag{53.}$$

$$\langle b_1, b_2\rangle\langle a_1, a_2\rangle = \langle b_1, b_2\rangle \cdot \langle a_1, a_2\rangle = \langle b_1a_1 - b_2a_2, b_2a_1 + b_1a_2\rangle; \tag{54.}$$

$$\frac{\langle b_1, b_2\rangle}{\langle a_1, a_2\rangle} = \left\langle \frac{b_1a_1 + b_2a_2}{a_1^2 + a_2^2}, \frac{b_2a_1 - b_1a_2}{a_1^2 + a_2^2} \right\rangle. \tag{55.}$$

Were these definitions even altogether arbitrary, they would at least not contradict each other, nor the earlier principles of Algebra, and it would be possible to draw legitimate conclusions, by rigorous mathematical reasoning, from premises thus arbitrarily assumed : but the persons who have read with attention the foregoing remarks of this theory, and have compared them with the Preliminary Essay, will see that these definitions are really not arbitrarily chosen, and that though others might have been assumed, no others would be equally proper.
With these definitions, addition and subtraction of number-couples are mutually inverse operations, and so are multiplication and division ; and we have the relations,

$$\langle b_1, b_2\rangle + \langle a_1, a_2\rangle = \langle a_1, a_2\rangle + \langle b_1, b_2\rangle, \tag{56.}$$

$$\langle b_1, b_2\rangle \cdot \langle a_1, a_2\rangle = \langle a_1, a_2\rangle \cdot \langle b_1, b_2\rangle, \tag{57.}$$

$$\langle b_1, b_2\rangle(\langle a_1', a_2'\rangle + \langle a_1, a_2\rangle) = \langle b_1, b_2\rangle\langle a_1', a_2'\rangle + \langle b_1, b_2\rangle\langle a_1, a_2\rangle : \tag{58.}$$

we may, therefore, extend to number-couples all those results respecting numbers, which have been deduced from principles corresponding to these last relations. For example,

$$(\langle b_1, b_2\rangle + \langle a_1, a_2\rangle) \cdot (\langle b_1, b_2\rangle + \langle a_1, a_2\rangle) = \langle b_1, b_2\rangle\langle b_1, b_2\rangle + 2\langle b_1, b_2\rangle\langle a_1, a_2\rangle + \langle a_1, a_2\rangle\langle a_1, a_2\rangle, \tag{59.}$$

in which

$$2\langle b_1, b_2\rangle\langle a_1, a_2\rangle = \langle 2, 0\rangle\langle b_1, b_2\rangle\langle a_1, a_2\rangle = \langle b_1, b_2\rangle\langle a_1, a_2\rangle + \langle b_1, b_2\rangle\langle a_1, a_2\rangle; \tag{60.}$$

for, in general, we may mix the signs of numbers with those of number-couples, if we consider every single number a as equivalent to a pure primary number-couple,

$$a = \langle a, 0\rangle. \tag{61.}$$

When the pure primary couple $\langle 1, 0\rangle$ is thus considered as equivalent to the number 1, it may be called, for shortness, the primary unit ; and the pure secondary couple $\langle 0, 1\rangle$ may be called in like manner the secondary unit.
We may also agree to write, by analogy to notations already explained,

$$\begin{aligned} \langle 0, 0\rangle + \langle a_1, a_2\rangle &= +\langle a_1, a_2\rangle, \\ \langle 0, 0\rangle - \langle a_1, a_2\rangle &= -\langle a_1, a_2\rangle; \quad (62.) \end{aligned}$$

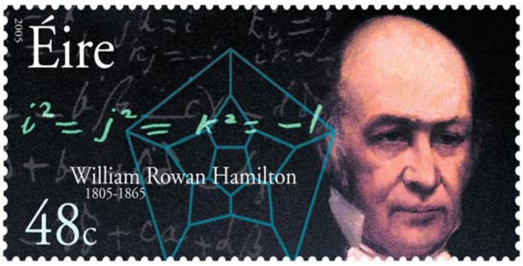
and then $+\langle a_1, a_2\rangle$ will be another symbol for the number-couple $\langle a_1, a_2\rangle$ itself, and $-\langle a_1, a_2\rangle$ will be a symbol for the opposite number-couple $\langle -a_1, -a_2\rangle$. The reciprocal of a number-couple $\langle a_1, a_2\rangle$ is this other number-couple,

$$\frac{1}{\langle a_1, a_2\rangle} = \frac{\langle 1, 0\rangle}{\langle a_1, a_2\rangle} = \left\langle \frac{a_1}{a_1^2 + a_2^2}, \frac{-a_2}{a_1^2 + a_2^2} \right\rangle = \frac{\langle a_1, -a_2\rangle}{a_1^2 + a_2^2}. \tag{63.}$$

It need scarcely be mentioned that the insertion of the sign of coincidence = between any two number-couples implies that those two couples coincide, number with number, primary with primary, and secondary with secondary ; so that an equation between number-couples is equivalent to a couple of equations between numbers.

Vous ferez de ce texte un commentaire composé :)

Exercice 1 - 31 Hamilton is back : ze quaternions



Après avoir travaillé, comme nous l'avons vu, sur les nombres complexes, Sir William (il a été anobli en 1835) voulut étendre ses travaux à la dimension 3. Il n'arrivait cependant pas à imaginer une multiplication qui puisse satisfaire des critères intéressants, notamment la conservation des normes.
Un jour qu'il se promenait avec son épouse dans un parc dublinois, il eut soudain un éclair qu'il s'empressa de graver sur la pierre du pont le plus proche :

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1$$

Remplissez alors la table de multiplication suivante :

$\times$	1	i	j	k
1				
i				
j				
k				

Hamilton définit alors un *quaternion* par la donnée de quatre réels a, b, c et d car un quaternion peut s'écrire de manière unique comme combinaison des quatre quaternions élémentaires :

$$q = a1 + b\mathrm{i} + c\mathrm{j} + d\mathrm{k}$$

Le nombre a est la partie scalaire et le triplet (b, c, d) est la partie vectorielle (c'est Hamilton qui a en effet introduit le terme de *vecteur* en mathématique).
Quelles propriétés possède ou ne possède pas la multiplication des quaternions ?
Il y a énormément de choses à dire sur les quaternions mais nous manquons un peu de recul mathématique... Point de vue informatique, il faut cependant noter qu'ils sont largement utilisés en infographie, en traitement du signal et dans tout domaine utilisant des rotations et entraînant de nombreux calculs.
En physique, ces calculs sont traditionnellement traités par du calcul matriciel mais nous verrons bientôt que celui-ci induit de nombreuses erreurs d'arrondis qui peuvent être atténuées en passant par les quaternions.
Un site à visiter : http://www.alcys.com/. Les quaternions apparaissent sous vos yeux ébaubis....

Anneaux et corps

Exercice 1 - 32 C'est fini

Soit $E = \{1, 2, 3, 5, 6, 10, 15, 30\}$.
Pour tout couple $\langle a, b\rangle$ d'éléments de E, on note $a \wedge b$ le PGCD de a et b et $a \vee b$ le PPCM de a et b.
Quelle est la structure de $\langle E, \wedge, \vee\rangle$? De $\langle E, \vee, \wedge\rangle$?

Exercice 1 - 33 Les Bool

Quelle structure possède $\langle\{\texttt{True}, \texttt{False}\}, \wedge, \vee\rangle$? Et $\langle\{\texttt{True}, \texttt{False}\}, \vee, \wedge\rangle$

Exercice 1 - 34 Les parties

Soit E un ensemble. Quelle structure possède $\langle\mathcal{P}(E), \cap, \cup\rangle$? $\langle\mathcal{P}(E), \cup, \cap\rangle$?

Exercice 1 - 35 Un type Rationnel sur Haskell

Qu'est-ce qu'un rationnel ? *Ratio* est un mot latin, qui désignait un calcul mais qui a donné en français le mot *raison*. Le mot *ratio* est passé en anglais pour désigner un indice en comptabilité : le rapport entre deux quantités de même nature. Ce mot a été importé de l'anglais de spécialité par les financiers.
En mathématique, un *nombre rationnel* est « construit » à partir d'un couple d'entiers relatifs, le deuxième étant non nul. Le premier est appelé *numérateur*, le second *dénominateur*.
Oui mais comment s'effectue cette « construction » ? On a une idée intuitive de la réponse : on veut qu'un *ratio*nnel représente une proportion. Par exemple, si Daniel a 5 billes bleues parmi ses 8 billes et si Fabienne a 10 billes bleues

parmi ses 16 billes, les deux enfants ont la même proportion de billes bleues. On voudrait alors que cette proportion soit désignée par le même nombre rationnel.
On voudrait donc mettre en *relation* tous les couples ⟨ nombre de billes bleues, nombre total de billes ⟩ qui « représentent » la même proportion de billes bleues.
On va donc modéliser cela en considérant la relation :

$$\Big(\forall\langle a,b\rangle \in \mathbb{Z}\otimes(\mathbb{Z}\setminus\{0\})\Big)\Big(\forall\langle c,d\rangle \in \mathbb{Z}\otimes(\mathbb{Z}\setminus\{0\})\Big)\Big(\langle a,b\rangle\,\mathcal{R}\,\langle c,d\rangle \leftrightarrow ad = bc\Big)$$

Il est facile de vérifier que $\mathcal{R}$ est une relation d'équivalence.
Une proportion est alors une classe modulo cette relation d'équivalence.
On désigne par $\mathbb{Q}$ l'ensemble quotient de $\mathbb{Z}\otimes(\mathbb{Z}\setminus\{0\})$ modulo $\mathcal{R}$.
Cette notation a été introduite par le mathématicien italien Guiseppe PEANO en 1895 car c'est l'initiale du mot italien « *quoziente* ».

$$\mathbb{Q} = \mathbb{Z}\otimes(\mathbb{Z}\setminus\{0\})/\mathcal{R} = \left\{\left[\langle n,d\rangle\right]_{\mathcal{R}} \mid \langle n,d\rangle \in \mathbb{Z}\otimes(\mathbb{Z}\setminus\{0\})\right\}$$

Un nombre rationnel est donc une classe d'équivalence, c'est-à-dire un ensemble !!!
Sur Haskell, on pourrait garder telle quelle cette idée et construire les rationnels à partir d'ensembles infinis, mais c'est informatiquement un peu dangereux.
On va plutôt adapter cette idée en commençant par *construire* un rationnel à partir d'un couple ⟨ **`Integer`**, **`integer`** ⟩.
Pour ne pas trop perturber nos habitudes, nous allons utiliser un constructeur infixe :

Haskell
```haskell
data Rationnel =  Integer :/ Integer
```

Ici, **`:/`** est le *constructeur* du type **`Rationnel`** : c'est en fait une fonction (bien sûr !!) :

Haskell
```haskell
Prelude> :t (:/)
(:/) :: Integer -> Integer -> Rationnel
```

On aura souvent besoin d'utiliser le numérateur ou le dénominateur d'un rationnel.
Remarquons tout d'abord que dans chaque classe, il y a un élément plus intéressant que les autres : lequel ? Comment l'obtenir ? Comment pensez-vous définir le signe d'un rationnel ?
On va donc créer deux fonctions **`num`** et **`den`** qui, pour un rationnel donné, vont renvoyer le numérateur et le dénominateur simplifiés.
Ainsi, on voudrait que le numérateur de **`10 :/ 8`** soit 5 et son numérateur soit... ?
On dispose de quelques fonctions utiles définies dans la classe **`Integral`** dont voici un extrait de la documentation :

Haskell
```haskell
Integral numbers, supporting integer division.

Methods

quot :: a -> a -> a
  integer division truncated toward zero

rem :: a -> a -> a
  integer remainder, satisfying (x `quot` y)*y + (x `rem` y) == x

div :: a -> a -> a
  integer division truncated toward negative infinity

mod :: a -> a -> a
  integer modulus, satisfying (x `div` y)*y + (x `mod` y) == x

quotRem :: a -> a -> (a, a)
  simultaneous quot and rem

divMod :: a -> a -> (a, a)
  simultaneous div and mod
```

```
toInteger :: a -> Integer
  conversion to Integer
```

Prelude dispose également de la fonction suivante :

Haskell

```
gcd              :: (Integral a) => a -> a -> a
gcd x y          =  gcd' (abs x) (abs y)
                    where gcd' a 0  =  a
                          gcd' a b  =  gcd' b (a `rem` b)
```

Complétez alors les pointillés :

Haskell

```
num :: Rationnel -> Integer
num    (x :/ 0)  = error "divide by zero"
num    (x :/ y)  = (...) * (...)


den :: Rationnel -> Integer
den    (x :/ 0)  = error "divide by zero"
den    (x :/ y)  = ... (...)
```

Par exemple, désignons par **x** le rationnel $\langle 5, 3, \rangle$:

Haskell

```
Prelude> let x = 10 :/ 8
Prelude> num x
5
Prelude> den x
4
```

Cependant, on est un peu embêté si l'on veut savoir si deux rationnels sont égaux, c'est-à-dire deux classes d'équivalence sont égales :

Haskell

```
Prelude> let x = 10 :/ 8
Prelude> let y = 20 :/ 16
Prelude> x == y

<interactive>:31:3:
    No instance for (Eq Rationnel)
      arising from a use of `=='
    Possible fix: add an instance declaration for (Eq Rationnel)
    In the expression: x == y
    In an equation for `it': it = x == y
```

Le débugueur de Haskell a été conçu par des gens tellement intelligents et gentils qu'il nous explique comment faire : on va faire du type **Rationnel** une *instance* de la classe de type **Eq** qui contient tous les types qui sont des classes d'équivalence.

Haskell

```
Prelude> :t (==)
(==) :: Eq a => a -> a -> Bool
```

Pour cela, on utilise la fonction **instance** :

Haskell

```
instance Eq Rationnel where
   f1 == f2 = (num f1) * (den f2) == (den f1) * (num f2)
```

Maintenant, on peut tester si deux rationnels sont égaux ou non :

Haskell

```
Prelude> 10 :/ 8 == 20 :/ 16
True
```

Pour améliorer notre confort d'utilisation de GHCi, nous allons créer une fonction d'affichage des rationnels (un « *pretty print* ») et instancier **Rationnel** dans la classe **Show** des types « montrables »...

Haskell

```
instance Show Rationnel where
  show f
     |den f == 0 = error "divide by zero"
     |den f == 1 = show (num f)
     |otherwise  = (show (num f)) ++ "/" ++ (show (den f))
```

Par exemple :

Haskell

```
*Main> 10 :/ (-8)
-5/4
*Main> 10 :/ (-5)
-2
```

Il ne reste plus qu'à donner une structure algébrique à **Rationnel** : est-ce que $\langle \mathbb{Q}, +, \times \rangle$ a une structure d'anneau ? De corps ?
Commençons par la structure d'anneau.
Il faut définir une addition et une multiplication : complétez les pointillés...

Haskell

```
r_plus :: Rationnel -> Rationnel -> Rationnel
r_plus    f1           f2         =  ... :/ ...

r_mult :: Rationnel -> Rationnel -> Rationnel
r_mult    f1           f2         =  ... :/ ...
```

Nous pouvons à présent faire de **Rationnel** une instance de **Num**
Comme évoqué en cours, il nous faut donc définir le sept fonctions de la classe **Num**

Haskell

```
class  (Eq a, Show a) => Num a  where
    (+), (-), (*)  :: a -> a -> a
    negate         :: a -> a
    abs, signum    :: a -> a
    fromInteger    :: Integer -> a
```

avec **a** notre type **Rationnel** pour que celui-ci en fasse partie.
Complétez alors les pointillés :

Haskell

```
instance Num Rationnel where
  (+) = ...
  (*) = ...
  negate f =  ... :/ ...
  fromInteger k = ... :/ ...
  abs f = ... :/ ...
  signum f = ... :/ ...
```

Alors, par exemple :

Haskell

```
*Main> 5 :/ 3 + 1 :/ 2
13/6
*Main> 5 :/ 3 - 1 :/ 2
7/6
*Main> 5 :/ 3 * 1 :/ 2
5/6
```

Bon, que pensez-vous de l'inverse d'un **Rationnel** ?

Haskell
```
r_inv :: Rationnel -> Rationnel
r_inv f
  |num f == 0 = ...
  |otherwise  = ...
```

On peut donc instancier **Fractional** avec notre **Rationnel** :

Haskell
```
instance Fractional Rationnel where
  (/) f1 f2 = ...
  fromRational r = (Ratio.numerator r) :/ (Ratio.denominator r)
```

La dernière ligne est un peu bizarre : en fait, il existe déjà dans **Prelude** un type pour les rationnels... Cette ligne permet donc de passer d'un **Rational** de **Prelude** à un élément de type **Rationnel**.

Haskell
```
*Main> (5 :/ 3) / (7 :/ 6)
10/7
```

Reprenons maintenant nos classes **Magma**, **Monoide**, **Groupe** : qu'elles nous inspirent pour créer des classes **Anneau** et **Corps** avec les tests qui vont avec. Est-ce qu'on peut faire de **Rationnel** une instance de **Anneau** ? De **Corps** ?

Exercice 1 - 36 Anneau des booléens

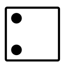
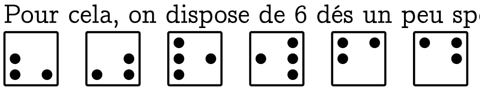
Pouvez-vous faire de **Bool** une instance de **Num** ? de **Rational** ?

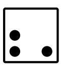
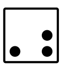
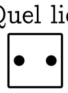
Espaces vectoriels

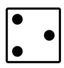
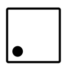
Exercice 1 - 37 RVB/CJMN

Comment modéliseriez-vous les systèmes RVB et CJMN en terme d'espace vectoriel ?
Comment passer en niveau de gris ? En codage 24 bits ?

Exercice 1 - 38 A bit of dice

On voudrait dessiner ce dé : 
Pour cela, on dispose de 6 dés un peu spéciaux :

On peut les additionner selon la règle suivante : on superpose les dés et un point apparaît si, et seulement si, un et un seul point est présent par position.
En fait, le jeu est constitué de petites lumières. Quand on appuie sur une de ces lumières, cela change l'état (allumé ou éteint) des lumières voisines. Quel lien y a-t-il entre la situation réelle et la modélisation proposée ?

Par exemple, 1  + 1  = 

Quel est le rapport avec notre cours ? Dessinez le dé demandé. Et celui-ci  ? Et celui-là  ?
Pour aller plus loin, aller jouer au *Lights Out* : http://www.jaapsch.net/puzzles/lights.htm
Pourra-t-on résoudre toutes les énigmes ? Est-ce qu'il est vraiment utile d'appuyer sur certaines touches ?

Exercice 1 - 39 Les canons

Montrez que $\mathbb{K}^n$ a une structure de $\mathbb{K}-$espace vectoriel.

Exercice 1 - 40 Espaces vectoriels pour les nuls

Avec les notations usuelles, que pensez-vous des égalités suivantes :

1. $0_{\oplus} \cdot \boldsymbol{u} = \mathbf{0}$
2. $\lambda \cdot \mathbf{0} = 0$

3. $(\lambda \cdot \boldsymbol{u} = \mathbf{0}) \rightarrow (\lambda = 0 \vee \boldsymbol{u} = \mathbf{0})$

Exercice 1 - 41

Démontrez le théorème 1 - 2 page 15

Exercice 1 - 42 **R-espaces vectoriels ?**

Déterminez si les ensembles suivants munis des opérations indiquées ont une structure de $\mathbb{R}$-espace vectoriel :

1. Soit $V = \{\langle a, b, c, 0\rangle \mid \langle a, b, c\rangle \in \mathbb{R}^3\}$ muni de l'addition canonique des quadruplets et du produit canonique d'un quadruplet par un réel.
2. Soit $V = \{\langle a, b, c, 2\rangle \mid \langle a, b, c\rangle \in \mathbb{R}^3\}$ muni de l'addition canonique des quadruplets et du produit canonique d'un quadruplet par un réel.
3. On munit $V = \mathbb{R}_+^\star$ de l'addition définie par $a \oplus b = a \times b$ et $\mathbb{R}$ opère sur le groupe $\langle V, \oplus\rangle$ par le produit externe défini par $\lambda \otimes \boldsymbol{x} = \boldsymbol{x}^\lambda$.
4. $V = \{\langle a, b, c\rangle \mid \langle a, b, c\rangle \in \mathbb{R}^3 \wedge a \cdot b \cdot c = 0\}$ avec les opérations canoniques.
5. V est l'ensemble des fonctions dérivables sur $\mathbb{R}$.
6. V est l'ensemble des suites arithmétiques réelles.

Exercice 1 - 43 **K-espaces vectoriels ?**

Les exemples suivants permettent-ils de définir des $\mathbb{K}$-espaces vectoriels ?

1. $\mathbb{K} = \mathbb{C}$, l'ensemble des nombres complexes, $V = \mathbb{C}$ muni de l'addition habituelle des complexes, et le produit externe est défini par :

$$(\star) : \begin{array}{ccc} \mathbb{C} \otimes \mathbb{C} & \rightarrow & \mathbb{C} \\ \langle \lambda, \boldsymbol{z}\rangle & \mapsto & \lambda^2 \cdot \boldsymbol{z} \end{array}$$

2. Soit $\mathbb{K}$ un corps quelconque et $V = \mathbb{K}^2$ muni de l'addition canonique des couples. On définit le produit externe ainsi :

$$(\star) : \begin{array}{ccc} \mathbb{K} \otimes \mathbb{K}^2 & \rightarrow & \mathbb{K}^2 \\ \langle \lambda, \langle x, y\rangle\rangle & \mapsto & \langle \lambda x, 0\rangle \end{array}$$

3. $\mathbb{K} = \mathbb{Q}$ et $V = \mathbb{R}$ munis des opérations canoniques.

Exercice 1 - 44

Soit V un espace vectoriel muni de tout ce qu'il faut.
Soit $\langle \boldsymbol{u_i}\rangle_{0 \leqslant i \leqslant p}$ une famille de vecteurs de V. Montrez que $\mathcal{V}\mathrm{ect}\{\boldsymbol{u_0} \cdots \boldsymbol{u_p}\}$ a une structure d'espace vectoriel.
Voilà pourquoi on appellera maintenant $\mathcal{V}\mathrm{ect}\{\boldsymbol{u_0} \cdots \boldsymbol{u_p}\}$ le **sous-espace vectoriel engendré** par la famille $\langle \boldsymbol{u_i}\rangle_{0 \leqslant i \leqslant p}$.
On dit que $\langle \boldsymbol{u_i}\rangle_{0 \leqslant i \leqslant p}$ est une **famille génératrice** de $\mathcal{V}\mathrm{ect}\{\boldsymbol{u_0} \cdots \boldsymbol{u_p}\}$.

Exercice 1 - 45

On travaille dans l'ev $\mathbb{R}^2$:

1. $u = \langle 6, -9\rangle$ et $v = \langle -10, 15\rangle$. Donner des CL de la famille $\langle u\rangle$, de la famille $\langle u, v\rangle$.
2. Que représente $\mathcal{V}\mathrm{ect}\{\mathcal{F}\}$ avec $\mathcal{F} = \langle u, v\rangle$?
3. Démontrer que $w = \langle 2, -3\rangle \in \mathcal{V}\mathrm{ect}\{\mathcal{F}\}$.
4. u et v sont-ils colinéaires ?
5. Démontrer que $\mathcal{V}\mathrm{ect}\{\mathcal{F}\} = \mathcal{V}\mathrm{ect}\{\langle w\rangle\}$.
6. Démontrer que $\langle 1, 2\rangle \notin \mathcal{V}\mathrm{ect}\{\mathcal{F}\}$.

Exercice 1 - 46

On note $H = \{\langle x, y, z\rangle \in \mathbb{R}^3 \mid x + 2y + 3z = 0\}$, démontrer que H est un sev de $\mathbb{R}^3$ en déterminant u et v, deux vecteurs de $\mathbb{R}^3$, de sorte que $H = \mathcal{V}\mathrm{ect}\{\langle u, v\rangle\}$.
On note $w = (1, 2, 3)$. Est-ce que $w \in H$?

Exercice 1 - 47

1. Démontrer que $H = \{\langle x, y, z\rangle \in \mathbb{R}^3 \mid x - z = 0\}$ est un sev de $\mathbb{R}^3$.
2. Démontrer que $H = \{\langle x, y, z\rangle \in \mathbb{R}^3 \mid x = 0\}$ est un sev de $\mathbb{R}^3$.

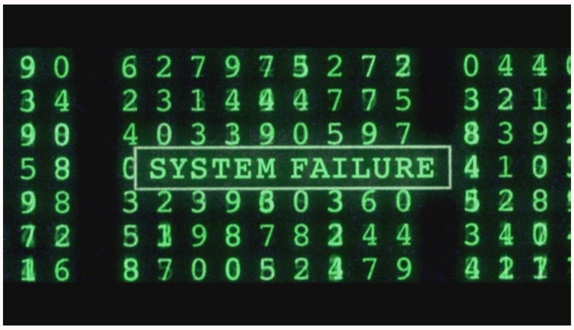
3. $H = \left\{\langle x, y, z\rangle \in \mathbb{R}^3 \mid ax + by + cz = 0\right\}$ où a, b et c sont des réels fixés. Démontrer que H est un sev de $\mathbb{R}^3$.

Exercice 1 - 48

On travaille dans $\mathbb{R}^3$ et on note $u = \langle 1, 2, 2\rangle$ et $v = \langle -2, 1, 2\rangle$.
Déterminer des réels a, b et c pour que $\mathcal{V}\mathrm{ect}\{\langle u, v\rangle\} = \left\{\langle x, y, z\rangle \in \mathbb{R}^3 \mid ax + by + cz = 0\right\}$.

CHAPITRE 2

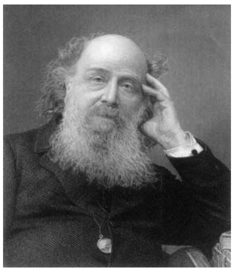
Les matrices



Une matrice, c'est juste un tableau de nombres, tout simplement...mais cela n'empêche pas la matrice d'être partout : dans la transmission d'information, dans une image, dans un robot, dans un complexe, dans un code secret, dans une relation, dans un GPS!
Entrons donc dans la matrice.

1 La matrice

1 1 Un peu d'histoire



J.SYLVESTER (1814-1897)

Vers 160 avant le petit Jésus, parut en Chine le Fang Tcheng ou, si vous préférez, « la disposition rectangulaire » qui désigne en mandarin actuel « équation mathématique ». C'est le huitième des *Neuf chapitres sur l'art mathématique*. Dès cette époque, les chinois regroupaient les coefficients des systèmes d'équations linéaires dans un tableau rectangulaire et transformait celui-ci pas à pas pour le résoudre. Cette méthode sera reprise vingt siècles plus tard par les Allemands Carl Friedrich GAUSS (1777 - 1855) et Wilhelm JORDAN (1842 - 1899)(à ne pas confondre avec le mathématicien français Camille JORDAN) et c'est elle que nous programmerons en CAML.
Le terme « matrice » pour désigner ces tableaux a été introduit en 1850 par James SYLVESTER en 1850. Son ami Arthur CAYLEY (1821 - 1855) introduisit le produit de matrices et la notion d'inverse cinq années plus tard. Le terme « matrice » est dérivé du latin *mater* qui signifie « mère »...

1 2 Qu'est-ce que c'est ?

1 2 1 Vecteur

Nous appellerons *vecteur* une liste d'éléments d'un ensemble $\mathbb{A}$. Cet ensemble $\mathbb{A}$ est muni de deux opérations, $\boxplus$ et $\boxdot$, qui lui confèrent une structure d'**anneau** (le plus souvent commutatif) :

— $(\mathbb{A}, \boxplus)$ a une structure de groupe commutatif ;
— $(\mathbb{A}, \boxdot)$ a une structure de monoïde ;
— $\boxdot$ est distributive sur $\boxplus$.

On désigne souvent par $0_{\mathbb{A}}$ l'élément neutre de $\boxplus$ et par $1_{\mathbb{A}}$ l'élément neutre de $\boxdot$.
$(\mathbb{A}, \boxdot)$ n'étant pas forcément un groupe, tout le monde n'admet pas forcément un inverse par $\boxdot$.

Recherche

Quels sont les éléments inversibles de $(\mathbb{Z}, \cdot)$?

On notera $\mathbb{A}^p$ l'ensemble des vecteurs à coefficients dans $\mathbb{A}$ de *taille* p.
Notons $u = [a_1, a_2, \dots, a_p]$ et $v = [b_1, b_2, \dots, b_p]$ de vecteurs de même taille. On peut alors en faire la *somme* :

$$u \boxplus v = [a_1 \boxplus b_1, a_2 \boxplus b_2, \dots, a_n \boxplus b_n]$$

On peut multiplier un vecteur par un *scalaire*, c'est-à-dire un élément de $\mathbb{A}$:

$$ku = [k \boxdot a_1, k \boxdot a_2, \dots, k \boxdot a_p]$$

On notera en particulier $-u = (-1_{\mathbb{A}})u$.
On peut multiplier deux vecteurs composante par composante :

$$u \boxdot v = [a_1 \boxdot b_1, a_2 \boxdot b_2, \dots, a_p \boxdot b_p]$$

1 2 2 Matrice

Une matrice à n lignes et p colonnes est un tableau d'éléments appartenant à un ensemble $\mathbb{A}$. On note alors $\mathbb{A}^{n \times p}$ l'ensemble des matrices de n lignes et p colonnes à coefficients dans $\mathbb{A}$.

$$\begin{array}{cc} & \begin{array}{cccccc} \hphantom{a_{1,1}} & \hphantom{a_{1,2}} & \hphantom{\cdots} & j & \hphantom{\cdots} & \hphantom{a_{1,p}} \end{array} \\ \begin{array}{c} \\ \\ i \\ \\ \\ \end{array} & \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,j} & \cdots & a_{1,p} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i,1} & a_{i,2} & \cdots & a_{i,j} & \cdots & a_{i,p} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,j} & \cdots & a_{n,p} \end{pmatrix} \end{array}$$

On note souvent a_{ij} le coefficients $a_{i,j}$ et M_{ij} le coefficient de M situé sur la ligne i et la colonne j.
En fait, une matrice est un vecteur dont les coefficients sont eux-mêmes des vecteurs ce qui facilitera notre tâche en programmation.

1 3 Matrices particulières

— On peut définir des **matrices par blocs** :

$$\begin{pmatrix} \boxed{a_{11}} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

pourra s'écrire :

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$$

avec $A \in \mathbb{A}^{1\times 1}$, $B \in \mathbb{A}^{1\times 2}$, $C \in \mathbb{A}^{2\times 1}$ et $D \in \mathbb{A}^{2\times 2}$

— Les **matrices diagonales** : $(a_{ij} = 0)$ pour $i \neq j$. On les notera :

$$A = \operatorname{diag}\left[a_{11}, a_{22}, \ldots, a_{mm}\right]$$

Avec $A \in \mathbb{A}^{n\times p}$ et $m = \min(n, p)$.

— Les **matrices tridiagonales** telles que $a_{ij} = 0$ pour $|i - j| > 1$.

— Les **matrices triangulaires supérieures** telles que $a_{ij} = 0$ pour $i > j$.

— Les **matrices triangulaires inférieures** telles que $a_{ij} = 0$ pour $i < j$.

— Les **matrices de Hessenberg supérieures** telles que $a_{ij} = 0$ pour $i > j + 1$.

— Les **matrices de Hessenberg inférieures** telles que $a_{ij} = 0$ pour $j > i + 1$.

— Une **matrice colonne** est une matrice de $\mathbb{A}^{n\times 1}$.

— Une **matrice ligne** est une matrice de $\mathbb{A}^{1\times n}$.

— Les **matrices Attila** sont les matrices remplies de 1.

— Les **matrices nulles** sont les matrices remplies de 0.

— Algorithmiquement parlant, il peut être intéressant de traiter à part les matrices contenant proportionnellement beaucoup de 0 : on les appelle les **matrices creuses**.

Donnez des exemples de matrices de chacun de ces types.

1 4 Opérations sur les matrices

1 4 1 Égalité

Deux matrices sont égales si, et seulement si, elles ont les mêmes dimensions et des coefficients égaux.

1 4 2 Opération terme à terme

Notons $(S_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant p}}$ la somme terme à terme des deux matrices $(A_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant p}}$ et $(B_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant p}}$ alors chaque coefficient de S est obtenu par :

$$S_{ij} = A_{ij} \boxplus N_{ij}$$

On peut définir de la même manière un produit terme à terme qu'il ne faudra pas confondre avec le produit de matrices ou le produit par un scalaire...

1 4 3 Multiplication par un scalaire

Soit $k \in \mathbb{A}$ et $M \in \mathbb{A}^{n\times p}$ alors la matrice $P = kM$ est définie par :

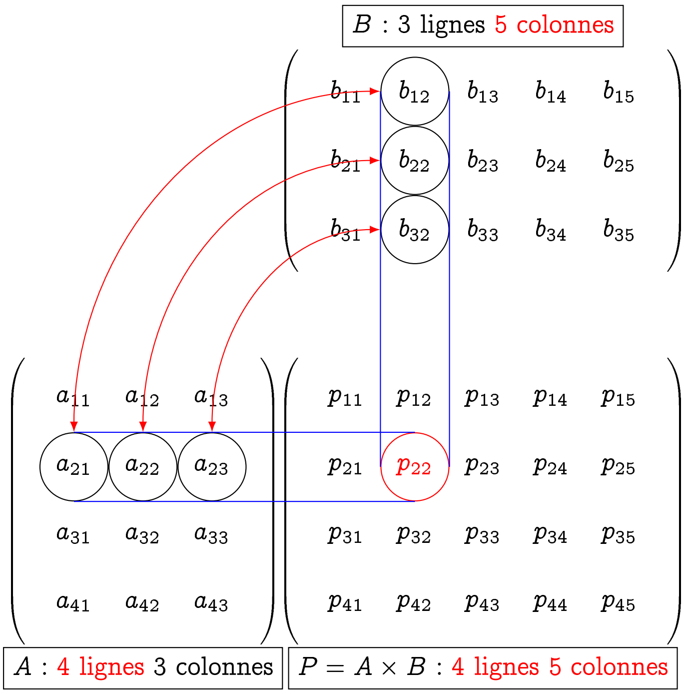
$$P_{ij} = k \boxdot M_{ij}$$

1 4 4 Produit de matrices

Un petit rappel...

Notons $(p_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant p}}$ le produit des deux matrices $(a_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant m}}$ et $(b_{ij})_{\substack{1\leqslant i\leqslant m\\1\leqslant j\leqslant p}}$ alors

$$p_{ij} = \bigboxplus_{k=1}^{m} a_{ik} \boxdot b_{kj}$$



Nous mettrons au point des fonctions nous permettant de calculer le produit de deux matrices.

Recherche

Montrez que $(\mathbb{A}^{n\times n}, +, \times)$ est un anneau. Est-il commutatif ?
On notera $\mathbb{I}_n$ l'élément neutre de $(\mathbb{A}^{n\times n}, \times)$: quelle est sa tête ?

1 4 5 Transposée d'une matrice

Soit $M \in \mathbb{A}^{n\times p}$. Sa transposée est la matrice T de $\mathbb{A}^{p\times n}$ définie par

$$T_{ij} = A_{ji}$$

On note le plus souvent $T = {}^{t}A$.
En pratique, on échange lignes et colonnes.

Recherche

Démontrez que ${}^{t}(A \times B) = {}^{t}B \times {}^{t}A$.

Définition 2 - 1

Une matrice telle que ${}^{t}A = A$ est dite **symétrique**.
Une matrice telles que ${}^{t}A = -A$ est dite **antisymétrique**

1 5 Matrice carrée inversible

Soit $M \in \mathbb{A}^{n\times n}$. M est inversible (régulière) si, et seulement si, il existe une matrice N dans $\mathbb{A}^{n\times n}$ telle que :

$$M \times N = N \times M = \mathbb{I}_n$$

On note alors $N = M^{-1}$.
On remarque (n'est-ce pas) que, d'après cette définition, $(M^{-1})^{-1} = M$.
Lorsque nous aurons étudier les déterminants, nous pourrons démontrer que si A et B sont deux matrices carrées de taille n vérifiant $A \times B = \mathbb{I}_n$, alors elles sont régulières et $A = B^{-1}$.

Recherche

Si A et B sont régulières et de taille n, alors comment calculer $(A \times B)^{-1}$ à partir des inverses de A te B ?

1 6 Opérations sur les lignes

1 6 1 Matrices élémentaires

Nous désignerons par E_n^{ij} la matrice carrée de $\mathbb{A}^{n\times n}$ dont tous les coefficients sont nuls sauf le coefficient (i,j) qui vaut $1_{\mathbb{A}}$. Par exemple, $E_3^{12} = \begin{pmatrix} 0&1&0\\0&0&0\\0&0&0 \end{pmatrix}$ dans $\mathbb{Z}^{3\times 3}$.

Leoplod Kronecker (1823-1891)

Leopold Kronecker est un mathématicien prussien né dans l'actuelle Pologne. On lui doit la célèbre citation : « *Die ganzen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk* ».
En son honneur, on a donné son nom à la fonction suivante :

$$\delta : \begin{array}{ccc} \mathbb{N}\times\mathbb{N} & \to & \{0;1\} \\ (i,j) & \mapsto & 1 \text{ si } i=j,\ 0 \text{ sinon} \end{array}$$

On condense souvent la notation en δ_{ij} et on parle alors de **symbole de Kronecker**. Par exemple, la matrice identité peut être définie par :

$$\mathbb{I}_n = (\delta_{ij})_{\substack{1\leqslant i\leqslant n\\ 1\leqslant j\leqslant n}}$$

Recherche

Étudiez le produit $E_n^{ij} \times E_n^{\ell k}$ et exprimez-le à l'aide du symbole de Kronecker.
Simplifiez ensuite le produit $(\mathbb{I}_n + \lambda E_n^{ij}) \times (\mathbb{I}_n - \lambda E_n^{ij})$: qu'en concluez-vous ?

1 6 2 Transvections de lignes

On s'intéresse à la fonction :

$$T_\lambda^{ij} : \begin{array}{ccc} \mathbb{A}^{n\times p} & \to & \mathbb{A}^{n\times p} \\ M & \mapsto & (\mathbb{I}_n + \lambda E_n^{ij}) \times M \end{array}$$

Calculez par exemple l'image par T_λ^{23} de $\begin{pmatrix} a_1&b_1&c_1&d_1\\a_2&b_2&c_2&d_2\\a_3&b_3&c_3&d_3 \end{pmatrix}$

Ainsi, $T_\lambda^{ij}(M)$ permet d'obtenir la matrice construite à partir de M en remplaçant la ligne i par elle-même plus λ fois la ligne j.
On note plus commodément cette transformation $L_i \leftarrow L_i \boxplus (\lambda \boxdot L_j)$.
Vous aurez bien noté que $(\mathbb{I}_n + \lambda E_n^{ij})^{-1} = \mathbb{I}_n - \lambda E_n^{ij}$.

1 6 3 Dilatations de lignes

On veut effectuer l'opération $L_i \leftarrow \lambda \boxdot L_i$. On note

$$\Delta_n^{i,\lambda} = \mathbb{I}_n + (\lambda \boxplus (-1_{\mathbb{A}}))E_n^{ii}$$

Calculez par exemple le produit de $\Delta_3^{2,\lambda}$ par $\begin{pmatrix} a_1&b_1&c_1&d_1\\a_2&b_2&c_2&d_2\\a_3&b_3&c_3&d_3 \end{pmatrix}$

Le produit par $\Delta_n^{i,\lambda}$ est une dilatation de ligne.

1 6 4 Échange de lignes

On considère la matrice $S_n^{ij} = \Delta_n^{j,-1_{\mathbb{A}}} \times (\mathbb{I}_n + E_n^{ij}) \times (\mathbb{I}_n - E_n^{ji}) \times (\mathbb{I}_n + E_n^{ij})$.

Développez ce produit. Que vaut le produit de S_3^{23} par $\begin{pmatrix} a_1&b_1&c_1&d_1\\a_2&b_2&c_2&d_2\\a_3&b_3&c_3&d_3 \end{pmatrix}$?

On note cette opération $L_i \leftrightarrow L_j$.

1 6 5 Opérations sur les lignes

De manière plus générale, une fonction φ de $\mathbb{A}^{n\times p}$ dans lui-même est une **opération sur les lignes** si c'est la composée finie de transvections et de dilatations de lignes.
L'image d'une matrice par φ est donc le produit à gauche par des matrices de transvections ou de dilatations :

$$\varphi : M \mapsto (F_k \times F_{k-1} \cdots \times F_1) \times M$$

avec chaque F_i inversible. La fonction φ est donc elle-même totale bijective et sa réciproque est :

$$\varphi^{-1} : N \mapsto (F_1^{-1} \times F_2^{-1} \cdots \times F_k^{-1}) \times N$$

On en déduit en particulier que l'inverse de $\varphi(\mathbb{I}_n)$ existe et que c'est $\varphi^{-1}(\mathbb{I}_n)$.

1 6 6 Lien avec les matrices régulières

Voici un théorème important qui nous sera très utile pour calculer l'inverse d'une matrice régulière.

Théorème 2 - 1

φ étant une opération élémentaire sur les lignes,

$$M \text{ inversible} \leftrightarrow \varphi(M) \text{ inversible}$$

En effet nous savons que $\varphi(M) = \varphi(\mathbb{I}_n) \times M$. Si M est inversible, $\varphi(M)$ s'exprime comme le produit de deux matrices inversibles et elle est donc inversible. Supposons maintenant $\varphi(M)$ inversible, comme $\varphi(\mathbb{I}_n)$ est inversible on obtient :

$$\varphi(\mathbb{I}_n)^{-1} \times \varphi(M) = \varphi(\mathbb{I}_n)^{-1} \times (\varphi(\mathbb{I}_n) \times M)$$

qui se transforme en

$$\varphi(\mathbb{I}_n)^{-1} \times \varphi(M) = \left(\varphi(\mathbb{I}_n)^{-1} \times \varphi(\mathbb{I}_n)\right) \times M = \mathbb{I}_n \times M$$

et en définitive

$$M = \varphi(\mathbb{I}_n)^{-1} \times \varphi(M) = \varphi^{-1}(\mathbb{I}_n) \times \varphi(M)$$

M s'exprimant comme le produit de deux matrices inversibles est inversible.

Théorème 2 - 2

Si $\varphi_1, \varphi_2, ..., \varphi_k$ est une suite d'opérations sur les lignes de M qui transforme M en $\mathbb{I}_n$ alors M est inversible et

$$M^{-1} = \varphi_k(\mathbb{I}_n) \times \varphi_{k-1}(\mathbb{I}_n) \times .... \times \varphi_1(\mathbb{I}_n) = \varphi_k \circ \varphi_{k-1} \circ ... \circ \varphi_1(\mathbb{I}_n)$$

En effet, si nous avons $\varphi_k \circ \varphi_{k-1} \circ ... \circ \varphi_1(M) = \mathbb{I}_n$ alors

$$\varphi_k \circ \varphi_{k-1} \circ ... \circ \varphi_1(M) = \mathbb{I}_n = (\varphi_k(\mathbb{I}_n) \times \varphi_{k-1}(\mathbb{I}_n) \times .... \times \varphi_1(\mathbb{I}_n)) \times M$$

Nous avons trouvé une matrice carrée qui multipliant M donne $\mathbb{I}_n$, c'est son inverse.
Le théorème précédent nous indique une méthode pour trouver l'inverse de M (s'il existe), nous allons étudier plus en détail cette technique et nous démontrerons plus loin que s'il est impossible de transformer M en $\mathbb{I}_n$ en utilisant les opérations élémentaires sur les lignes, alors M n'est pas inversible.

2 Rang d'une matrice

Dans tout ce qui suit nous allons utiliser les opérations élémentaires sur les lignes d'une matrice et on travaille dans $\mathbb{A}^{n\times p}$.

2 1 Matrices ligne-équivalentes

Nous dirons que deux matrices M et N sont **ligne-équivalentes** si, et seulement si, il existe une suite finie $(\varphi_i)_{i\in\mathbb{N}_k}$ d'opérations élémentaires sur les lignes de sorte que

$$N = \varphi_k \circ \varphi_{k-1} \circ \ldots \circ \varphi_1(M)$$

Nous écrirons alors $M \overset{\ell}{\equiv} N$. La relation $\overset{\ell}{\equiv}$ est manifestement une relation d'équivalence sur $\mathbb{A}^{n\times p}$, on démontre sans peine que

$$\begin{array}{l} M \overset{\ell}{\equiv} M \\ M \overset{\ell}{\equiv} N \to N \overset{\ell}{\equiv} M \\ \left(M \overset{\ell}{\equiv} N \text{ et } N \overset{\ell}{\equiv} C\right) \to M \overset{\ell}{\equiv} C \end{array}$$

Nous savons que $\varphi_k \circ \varphi_{k-1} \circ \ldots \circ \varphi_1(M) = \varphi_k \circ \varphi_{k-1} \circ \ldots \circ \varphi_1(\mathbb{I}_n) \times M$, par conséquent nous aurons $M \overset{\ell}{\equiv} N$ s'il existe une matrice R inversible vérifiant

$$N = R \times M$$

R étant le résultat du produit de matrices du type $\varphi(\mathbb{I}_n)$. Pour obtenir cette matrice R il suffit d'appliquer successivement en parallèle les opérations élémentaires qui transforment M en N sur $\mathbb{I}_n$. Pour cela on utilise un tableau où l'on juxtapose M et $\mathbb{I}_n$, M étant la partie gauche et $\mathbb{I}_n$ la partie droite de ce tableau. Toute opération élémentaire sur les lignes effectuée sur la partie gauche est simultanément effectuée sur la partie droite.

opérations φ	Partie gauche	Partie droite	Remarques
	M	$\mathbb{I}_n$	initialisation du tableau
φ_1	M_1	R_1	$M_1 = \varphi_1(M)$, $R_1 = \varphi_1(\mathbb{I}_n)$
φ_2	M_2	R_2	$M_2 = \varphi_2(M_1)$, $R_2 = \varphi_2(R_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
φ_i	M_i	R_i	$M_i = R_i \times M$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
φ_k	N	R	$N = R \times M$

2 2 L réduite échelonnée

La matrice $M = (m_{ij}) \in \mathbb{A}^{n\times p}(\mathbb{K})$ est dite ***ℓ*-réduite** (*ℓ* pour « ligne ») si, et seulement si, elle satisfait aux conditions suivantes :

1. Toutes les lignes nulles (une ligne est nulle si elle ne comporte que des zéros) sont au-dessous des lignes non nulles.
2. Dans chaque ligne non nulle le premier élément non nul est $1_{\mathbb{A}}$ (on lit une ligne de la gauche vers la droite), ce $1_{\mathbb{A}}$ est appelé **pivot** ou élément pivot. La colonne où se trouve ce $1_{\mathbb{A}}$ est appelée colonne pivot et c'est le seul élément non nul de cette colonne.
3. Si, **de plus**, les pivots apparaissent en ordre croissant par numéro de ligne **et** numéro de colonne, on dit que M est ***ℓ*-réduite échelonnée** (en abrégé lré ou LRé).

1

Zone du prochain pivot

Donnons quelques exemples de matrices entières :

$$\begin{pmatrix} 0 & \boxed{1} & 2 & 0 \\ \boxed{1} & 0 & 3 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ est } \ell\text{-réduite non échelonnée}$$

$$\begin{pmatrix} 0 & \boxed{1} & 0 \\ 0 & 2 & 0 \\ 0 & 0 & \boxed{1} \end{pmatrix} \text{ n'est pas } \ell\text{-réduite}$$

$$\begin{pmatrix} \boxed{1} & -2 & 0 \\ 0 & 0 & \boxed{1} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ est } \ell\text{-réduite échelonnée}$$

Théorème 2 - 3

Pour toute matrice $M \in \mathbb{A}^{n \times p}$, il existe une unique matrice ℓ-réduite échelonnée $N \in \mathbb{A}^{n \times p}$ telle que $M \overset{\ell}{\equiv} N$, N est appelée la ℓ-réduite échelonnée de M que l'on peut noter par $N = \text{lré}(M)$.

La démonstration de ce théorème se fait par récurrence sur n et elle est un peu difficile.
Le **rang** de M, noté $\text{rang}(M)$, est le nombre de lignes non nulles de sa ℓ-réduite échelonnée, c'est donc aussi le nombre de pivots da sa lré. M est dite de plein rang si son rang est égal à son nombre de lignes.
Nous énoncons les évidences (conséquences directes de la définition) :

1. Si $M \in \mathbb{A}^{n \times p}$ le rang de M est inférieur ou égal à n **et** à p.
2. Si $M \in \mathbb{A}^{n \times n}$ et si $\text{rang}(M) = n$ alors sa ℓ-réduite échelonnée est $\mathbb{I}_n$ et nous avons précédemment démontré que dans ce cas M est inversible.
3. Deux matrices ligne-équivalentes ont la même ℓ-réduite échelonnée et ont donc même rang.
4. Si M' est une matrice extraite de M, on a $\text{rang}(M') \leq \text{rang}(M)$.

3 Algorithme Fang-Tcheng

Wilhelm Jordan
(1842-1899)

Notre eurocentrisme préfère nommer cet algorithme Gauss-Jordan...
Nous allons le présenter sur un exemple. Soit à chercher la ℓ-réduite échelonnée de la matrice

$$A = \begin{pmatrix} 2 & 5 & -2 & 3 \\ 3 & 6 & 3 & 6 \\ 1 & 2 & -1 & 2 \end{pmatrix}$$

Nous allons faire apparaître les pivots ordonnés par numéro de ligne et numéro de colonne, cela veut dire que si un pivot (qui sera toujours égal à 1) est obtenu à la ligne i et colonne j, le pivot suivant sera au moins en ligne $i+1$ et en colonne $j+1$ et on s'imposera de trouver la solution minimale en numéro de ligne et numéro de colonne.

— **Première étape.** On repère l'élément de la première colonne qui est le plus grand en valeur absolue (il peut y avoir plusieurs choix). Si le résultat est nul (la première colonne ne contient que des zéros), la première colonne ne peut être une colonne pivot et on passe à la colonne suivante. Ici c'est 3 qui se trouve sur la deuxième ligne première colonne. On permute alors la première ligne avec la deuxième ligne et on obtient

$$A_1 = \begin{pmatrix} 3 & 6 & 3 & 6 \\ 2 & 5 & -2 & 3 \\ 1 & 2 & -1 & 2 \end{pmatrix}$$

On divise tous les éléments de la première ligne par 3 :

$$A_2 = \begin{pmatrix} \boxed{1} & 2 & 1 & 2 \\ 2 & 5 & -2 & 3 \\ 1 & 2 & -1 & 2 \end{pmatrix}$$

On fait apparaître ensuite des zéros sous le premier 1 de la première colonne en utilisant les opérations $L_2 \leftarrow L_2 - 2L_1$ et $L_3 \leftarrow L_3 - 1L_1$:

$$A_3 = \begin{pmatrix} \boxed{1} & 2 & 1 & 2 \\ 0 & 1 & -4 & -1 \\ 0 & 0 & -2 & 0 \end{pmatrix}$$

La première colonne est une colonne pivot et elle ne devra pas être modifiée par la suite.

— **Deuxième étape.** On repère dans la deuxième colonne (en fait la colonne qui suit la dernière colonne pivot obtenue), à partir de la deuxième ligne (en fait à partir du numéro de ligne qui suit le numéro de la ligne qui a donné le dernier pivot), le plus grand élément en valeur absolue (si on obtient zéro on passe à la colonne suivante, etc$\cdots$). Ici on obtient 1 qui se trouve sur la deuxième ligne et deuxième colonne et il n'y a pas de permutation de lignes à faire. Maintenant il faut faire apparaître des zéros dans la colonne pivot en ligne 1 et en ligne 3. On utilise les opérations $L_1 \leftarrow L_1 - 2L_2$ et $L_3 \leftarrow L_3 - 0L_2$ (qui ne sert à rien) ; on obtient :

$$A_4 = \begin{pmatrix} \boxed{1} & 0 & 9 & 4 \\ 0 & \boxed{1} & -4 & -1 \\ 0 & 0 & -2 & 0 \end{pmatrix}$$

La deuxième colonne est une colonne pivot, elle ne devra pas être modifiée dans la suite.

— **Troisième étape.** On repère dans la colonne qui suit la dernière colonne pivot obtenue et à partir de la ligne qui suit la ligne qui a donné le dernier pivot le plus grand élément en valeur asolue. Ici c'est -2 qui se trouve sur la troisième ligne et troisième colonne, la troisième colonne est alors la troisième colonne pivot. Il n'y a pas de permutation de lignes à faire, nous n'avons qu'à faire apparaître un 1 à la place de -2 puis faire apparaître des zéros dans la colonne pivot en conservant évidemment le pivot 1. Appliquons à A_4 l'opération $L_3 \leftarrow -\frac{1}{2}L_3$

$$A_5 = \begin{pmatrix} \boxed{1} & 0 & 9 & 4 \\ 0 & \boxed{1} & -4 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

puis appliquons $L_1 \leftarrow L_1 - 9L_3$ et $L_2 \leftarrow L_2 + 4L_3$

$$A_6 = \begin{pmatrix} \boxed{1} & 0 & 0 & 4 \\ 0 & \boxed{1} & 0 & -1 \\ 0 & 0 & \boxed{1} & 0 \end{pmatrix}$$

Nous venons d'obtenir la ℓ-réduite échelonnée de A, A est de rang 3.

Nous avons utilisé, ici, la méthode dite du pivot partiel en cherchant dans chaque colonne le plus grand élément en valeur absolue. Si nous avions choisi de prendre le premier élément rencontré non nul, en vertu de l'unicité de la ℓ-réduite échelonnée, nous aurions obtenu le même résultat final. Il est conseillé, en programmation, d'utiliser la méthode dite pivot partiel pour éviter une trop grande propagation des erreurs lors des divisions par des petits nombres. Si on fait les calculs à la main il vaut mieux se contenter de choisir, tant que c'est possible, des nombres sympathiques.

Théorème 2 - 4 A est une matrice carrée d'ordre n inversible si, et seulement si, le rang de A est égal à n.

Nous avons en fait déjà démontré une partie de ce théorème mais, vu son importance, nous allons reprendre ce qui a été dit. Supposons donc que le rang de A est égal à n, dans ces conditions la ℓ-réduite échelonnée de A est $\mathbb{I}_n$, cela signifie que $A \overset{\ell}{\equiv} \mathbb{I}_n$ et donc qu'il existe A' vérifiant $A' \times A = \mathbb{I}_n$, A' est l'inverse de A. Supposons maintenant que la rang de A est strictement inférieur à n, il est alors sûr que la dernière ligne de la ℓ-réduite échelonnée de A est une ligne nulle. Notons B cette ℓ-réduite échelonnée, nous savons que A inversible équivaut à écrire que B est inversible puisque $A \overset{\ell}{\equiv} B$. Supposons B inversible, il existe alors B' vérifiant $B \times B' = \mathbb{I}_n$ or cette égalité est impossible car la dernière ligne de B étant nulle, la dernière ligne du produit $B \times B'$ sera aussi nulle et donc distincte de la dernière ligne de $\mathbb{I}_n$. Nous venons de démontrer

$$\mathrm{rg}\,(A) < n \to A \text{ non inversible}$$

et donc A inversible $\to \mathrm{rg}\,(A) = n$.

Pratique : pour rechercher la matrice inverse de A, si elle existe, on applique simultanément sur $\mathbb{I}_n$ les opérations faites pour déterminer la ℓ-réduite échelonnée de A. Pour cela on considère le tableau

$$[A\,|\mathbb{I}_n\,]$$

on applique l'algorithme Fang Tcheng tableau, et chaque opération faite sur la partie gauche est reproduite sur la partie droite. Si la ℓ-réduite de A est la matrice $\mathbb{I}_n$ (le résultat de la partie gauche est $\mathbb{I}_n$) alors A est inversible et sa matrice inverse est donnée par la partie droite. Si la partie gauche ne donne pas $\mathbb{I}_n$, A n'est pas inversible, son rang est strictement inférieur à n.

4 Déterminant d'une matrice carrée

4 1 Déterminant d'une matrice de taille 2

Godfried W. Leibniz (1646-1716)

En fait, vous savez depuis la Seconde que deux vecteurs du plan de coordonnées (a, b) et (a', b') sont colinéaires si, et seulement si, leurs coordonnées sont proportionnelles. Il existe donc un réel λ tel que $a = \lambda a'$ et $b = \lambda b'$ et alors

$$ab' - a'b = \lambda ab - \lambda ab = 0$$

On appelle déterminant de la matrice $A = \begin{pmatrix} a & a' \\ b & b' \end{pmatrix}$ le nombre $ab' - a'b$ et on note :

$$\det(A) = \begin{vmatrix} a & a' \\ b & b' \end{vmatrix} = ab' - a'b$$

4 2 Propriétés des déterminants

Dans la suite, $A = (a_{ij})$, $B = (b_{ij})$ sont des matrices carrées de taille n, λ est un scalaire. L_i est une ligne quelconque de A. On note Δ_{ij} le déterminant de la matrice obtenue à partir de A en « barrant » la ligne i et la colonne j.

Propriété 2 - 1

Soient i et j deux entiers compris entre 1 et n.

$$\det A = \boxplus_{k=1}^{n} a_{1k} \square (-1)^{1+k} \square \Delta_{1k} = \boxplus_{k=1}^{n} a_{ik} \square (-1)^{i+k} \square \Delta_{ik} = \boxplus_{k=1}^{n} a_{kj} \square (-1)^{k+j} \square \Delta_{kj}$$

On admettra cette propriété. Voyons ce que cela donne pour une matrice de taille 3 :

$$(-1)^{(1+1)} \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} + (-1)^{(2+1)} \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} + (-1)^{(3+1)} \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

C'est-à-dire :

$$\det(A) = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{21}\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} + a_{31}\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix}$$

Par exemple, calculons le déterminant de $A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & -2 \\ -2 & 3 & 1 \end{pmatrix}$

$$(-1)^{(1+1)}\begin{vmatrix} \boxed{1} & \cancel{2} & \cancel{3} \\ \cancel{2} & \boxed{1} & \boxed{-2} \\ \cancel{-2} & \boxed{3} & \boxed{1} \end{vmatrix} +(-1)^{(2+1)}\begin{vmatrix} \cancel{1} & \boxed{2} & \boxed{3} \\ \boxed{2} & \cancel{1} & \cancel{-2} \\ \cancel{-2} & \boxed{3} & \boxed{1} \end{vmatrix} +(-1)^{(3+1)}\begin{vmatrix} \cancel{1} & \boxed{2} & \boxed{3} \\ \cancel{2} & \boxed{1} & \boxed{-2} \\ \boxed{-2} & \cancel{3} & \cancel{1} \end{vmatrix}$$

C'est-à-dire :

$$\begin{aligned} \det(A) &= 1 \times \begin{vmatrix} 1 & -2 \\ 3 & 1 \end{vmatrix} - 2 \times \begin{vmatrix} 2 & 3 \\ 3 & 1 \end{vmatrix} + (-2) \times \begin{vmatrix} 2 & 3 \\ 1 & -2 \end{vmatrix} \\ &= (1+6) - 2(2-9) - 2(-4-3) \\ &= 7 + 14 + 14 \\ &= 35 \end{aligned}$$

```
# det entier [[1;2;3];[2;1;-2];[-2;3;1]];;
- : int = 35
```

Recherche

Appliquez la même méthode pour calculer de tête le déterminant de la matrice suivante :

$$M = \begin{pmatrix} 1 & 17 & 32 & 49 \\ 0 & 5 & 0 & 3 \\ 0 & 39 & 2 & -49 \\ 0 & 7 & 0 & 1 \end{pmatrix}$$

Propriété 2 - 2

Le déterminant d'une matrice triangulaire est égal au produit des éléments de sa diagonale principale.

Sans utiliser cette propriété, calculez de tête le déterminant de la matrice suivante :

$$M = \begin{pmatrix} 2 & 32 & 45 & 87 & 987 \\ 0 & -1 & 568 & -542 & 712 \\ 0 & 0 & 5 & 741 & -12 \\ 0 & 0 & 0 & 1 & -789 \\ 0 & 0 & 0 & 0 & -10 \end{pmatrix}$$

Prouvez alors cette propriété dans le cas général.

Propriété 2 - 3

Si la matrice B résulte de l'échange de deux lignes ou de deux colonnes d'une matrice A alors $\det B = -\det A$

On admettra cette propriété.
Calculez de tête le déterminant de la matrice suivante :

$$M = \begin{pmatrix} 45 & 32 & 2 & 987 & 87 \\ 568 & -1 & 0 & 712 & -542 \\ 5 & 0 & 0 & -12 & 741 \\ 0 & 0 & 0 & -789 & 1 \\ 0 & 0 & 0 & -10 & 0 \end{pmatrix}$$

Propriété 2 - 4

$$\det\left({}^{t}A\right) = \det A$$

Démontrez cette propriété.

Propriété 2 - 5

Le déterminant d'une matrice comportant une ligne (ou une colonne) de 0 est nul.

Démontrez cette propriété.

Propriété 2 - 6

Soit B la matrice obtenue à partir de A en effectuant $L_i \leftarrow \lambda L_i$, alors $\det B = \lambda \det A$.

Démontrez cette propriété.

Propriété 2 - 7

$$\det(\lambda A) = \lambda^n \det A$$

Démontrez cette propriété.
Si A est une matrice carrée d'ordre 5 dont le déterminant vaut 4, que vaut $\det(-2A)$?
Démontrez que le déterminant d'une matrice antisymétrique (i.e. ${}^{t}A = -A$) d'ordre 37 est nul.

Propriété 2 - 8

Le déterminant d'une matrice comportant deux lignes (ou colonnes) identiques est nul.

Démontrez cette propriété en utilisant la propriété 2 - 3 page précédente.

Propriété 2 - 9

Le déterminant d'une matrice qui comporte deux lignes (ou colonnes) dont l'une est multiple de l'autre est nul.

Démontrez cette propriété.

Propriété 2 - 10

Soit B la matrice obtenue à partir de A en effectuant $L_i \leftarrow L_i + \lambda L_j$, alors $\det B = \det A$.

Démontrez cette propriété. Utilisez-la pour calculer le déterminant de la matrice suivante :

$$M = \begin{pmatrix} 1 & 3 & 2 & -1 \\ -1 & -2 & 2 & 6 \\ 2 & 4 & -3 & 2 \\ -2 & -7 & 4 & 1 \end{pmatrix}$$

Estimez le nombre d'opérations nécessaires pour effectuer le calcul du déterminant d'une matrice de taille n en utilisant uniquement la définition.
Faites de même lorsqu'on utilise l'algorithme Fang Tcheng
En juin 2012, un ordinateur américain de la firme IBM a établi un nouveau record de vitesse de calcul de 16,32 péta-flops, soit $16,32 \times 10^{16}$ opérations mathématiques à virgule flottante par seconde. En comparaison, un ordinateur individuel standard a une puissance d'environ 100 giga-flops. L'ordinateur utilisé compte 1 572 864 cœurs. Combien de temps lui faudra-t-il environ pour effectuer le calcul d'un déterminant de taille 30 à l'aide de la définition ? Avec l'algorithme Fang Tcheng ?

Il est à noter que le test de performance servant à classer les superordinateurs est le LINPACK : il mesure le temps mis pour résoudre un système dense (sans zéros) de n équations à n inconnues en utilisant l'algorithme Fang Tcheng

Propriété 2 - 11

Soit A et B deux matrices carrées de même taille alors

$$\det(A \times B) = \det A \cdot \det B$$

Nous admettrons cette propriété. Démontrez alors les propriétés suivantes :

Propriétés 2 - 12

- Si A est inversible alors $\det A \neq 0$;
- Si A est singulière (non-inversible) alors $\det A = 0$;
- A est inversible si, et seulement si, $\det A \neq 0$.

5 Résolution de systèmes

5 1 Généralités

Gabriel CRAMER
(1704-1752)

On appelle **système de n équations linéaires à p inconnues** dans $\mathbb{A}$ tout système de la forme :

$$(S) : \begin{cases} \sum\limits_{j=1}^{p} a_{i,j} x_j = b_i \\ i \in \{1, 2, \cdots, n\} \\ a_{i,j} \text{ et } b_i \in \mathbb{K} \end{cases}$$

Soit aussi

$$(S) : \begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,p}x_p = b_1 \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,p}x_p = b_p \end{cases}$$

$x_1, x_2, \cdots, x_p$ sont les p inconnues, les $a_{i,j}$ sont appelés les coefficients du système (S) et les b_i sont appelés les seconds membres (ce sont aussi des coefficients du système (S)).
Notons $A = (a_{i,j}) \in \mathbb{A}^{n \times p}$ (A est appelée la matrice du système).
$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$ est la matrice colonne des inconnues et $B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$ est la matrice colonne des seconds membres, le système (S) s'écrit matriciellement :

$$\begin{pmatrix} a_{1,1} & a_{2,2} & \cdot & \cdot & \cdot & a_{1,p} \\ a_{2,1} & a_{2,2} & \cdot & \cdot & \cdot & a_{2,p} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n,1} & a_{n,2} & \cdot & \cdot & \cdot & a_{n,p} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

soit aussi

$$A \times X = B \text{ ou } {}^tX \times^t A =^t B$$

Résoudre le système (S) c'est chercher tous les p-uplets $(x_1, x_2, \cdots, x_p)$ de $\mathbb{A}^p$ qui vérifient simultanément les n équations. C'est aussi chercher toutes les matrices $X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix} \in \mathbb{A}^{p \times 1}$

vérifiant l'égalité matricielle $A \times X = B$. C'est pourquoi, dans ce qui suit, nous serons souvent amenés à confondre le p-uplet $(x_1, x_2, \cdots, x_p)$ avec la matrice colonne $\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$. Attention, en informatique et plus particulièrement en réseaux, le p-uplet $(x, x_2, \cdots, x_p)$ est aussi noté matriciellement par

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_p \end{pmatrix}$$

qui n'est autre que la transposée de X.
Le système est dit **homogène** si, et seulement si, tous les seconds membres (les coefficients b_i) sont nuls. Dans ce cas le système admet forcément au moins une solution : la solution nulle ou solution banale qui est le p-uplet

$$(0, 0, \cdots, 0) \in \mathbb{A}^p$$

ou bien la matrice nulle $0_{p,1}$ si on note matriciellement cette solution.
Le système $A \times X = B$ est dit **régulier** ou de **Cramer** si, et seulement si, il a autant d'équations que d'inconnues et si A est inversible (le système ayant autant d'équations que d'inconnues on a $n = p$ et A est une matrice carrée). A étant inversible, A^{-1} existe et

$$\begin{aligned} A \times X &= B \text{ se transforme en} \\ A^{-1} \times (A \times X) &= A^{-1} \times B, \text{ c'est-à-dire} \\ X &= A^{-1} \times B \end{aligned}$$

Le système admet donc au moins une solution qui est $A^{-1} \times B$. Nous allons prouver qu'il n'y en a pas d'autre. Pour cela supposons l'existence d'une autre solution X', nous avons alors simultanément

$$A \times X = B \text{ et } A \times X' = B$$

En retranchant membre à membre on obtient :

$$A \times (X - X') = 0_{n,1}$$

Multiplions maintenant les deux membres par A^{-1}

$$\begin{aligned} A^{-1} \times (A \times (X - X')) &= A^{-1} \times 0_{n,1} = 0_{n,1} \\ (A^{-1} \times A) \times (X - X') &= 0_{n,1} \\ \mathbb{I}_n \times (X - X') &= 0_{n,1} = X - X' \text{ et pour finir} \\ X &= X' \end{aligned}$$

Nous retiendrons que si le système $A \times X = B$ est tel que A est inversible (il est nécessaire que A soit carrée et donc que le système possède autant d'équations que d'inconnues) alors il admet une unique solution qui est

$$X = A^{-1} \times B$$

5 2 Systèmes équivalents et résolution

Deux systèmes linéaires (S) et (S') sont **équivalents** si, et seulement si, ils possèdent le même ensemble solution.
Nous allons définir trois opérations élémentaires :

1. Multiplication d'une équation par un scalaire (un élément de $\mathbb{A}$) non nul. Si E_i est la i^{e} équation, le résultat de la multiplication de E_i par $\alpha \in \mathbb{A}$ est noté αE_i et nous écrirons $E_i \leftarrow \alpha E_i$.

$$\begin{aligned} E_i &: \sum_{j=1}^{p} a_{i,j} x_j = b_i \\ \alpha E_i &: \sum_{j=1}^{p} \alpha a_{i,j} x_j = \alpha b_i \end{aligned}$$

Pour retrouver l'équation initiale il suffit de multiplier l'équation obtenue par $\frac{1}{\alpha}$ et par conséquent le système obtenu par cette opération est équivalent au système initial.

2. Permutation de deux équations, nous noterons $E_i \longleftrightarrow E_j$ l'opération qui consiste à permuter l'équation numéro i avec l'équation numéro j. Pour retrouver le système initial il suffit d'appliquer de nouveau cette opération, cela nous permet d'affirmer que cette opération transforme un système en un système équivalent.
3. Ajout à une équation une autre équation multipliée par un scalaire : si on ajoute à l'équation E_i l'équation $E_{h\neq i}$ multipliée par β nous noterons $E_i \leftarrow E_i + \beta E_h$

$$\begin{cases} E_i : \sum\limits_{j=1}^{p} a_{i,j}x_j = b_i \\ E_h : \sum\limits_{j=1}^{p} a_{h,j}x_j = b_h \\ E_i \leftarrow E_i + \beta E_h : \sum\limits_{j=1}^{p} (a_{i,j} + \beta a_{h,j})x_j = b_i + \beta b_h \end{cases}$$

Pour retrouver l'équation de départ il suffit d'appliquer au résultat l'opération $E_i \leftarrow E_i - \beta E_h$ et cette opération, comme les précédentes, transforme un système en un système équivalent.

Nous allons maintenant passer à la résolution : nous pouvons écrire le système

$$(S) : \begin{cases} \sum\limits_{j=1}^{p} a_{i,j}x_j = b_i \\ i \in \{1, 2, \cdots, n\} \\ a_{i,j} \text{ et } b_i \in \mathbb{A} \end{cases}$$

sous la forme

$$[A \mid B]$$

où A est la matrice du système et B la matrice colonne des seconds membres. Nous notons T cette matrice (ou ce tableau), le nombre de lignes de T est égal au nombre de lignes de A soit aussi le nombre d'équations de (S), le nombre de colonnes de T est égal au nombre de colonnes de $A+1$ soit aussi le nombre d'inconnues $+1$. Les opérations élémentaires sur les équations sont alors des opérations élémentaires sur les lignes de ce tableau, les opérations se faisant simultanément sur la partie gauche et sur la partie droite, c'est-à-dire sur le tableau T. Appliquons à ce tableau l'algorithme de Gauss-Jordan pour obtenir la ℓ-réduite échelonnée de A (on ne cherchera pas pour le moment à faire apparaître un pivot dans la dernière colonne du tableau) et notons R cette ℓ-réduite échelonnée. Nous notons le résultat

$$T' = [R \mid H], R = \text{lré}(A)$$

Le système est alors devenu $R \times X = H$. Comme on a $A \overset{\ell}{\equiv} R$ et $B \overset{\ell}{\equiv} H$, il existe P (rappel, P est inversible) telle que

$$\begin{aligned} R &= P \times A \\ H &= P \times B \end{aligned}$$

Si nous notons U une solution du système (S), U vérifie

$$A \times U = B$$

et en multipliant les deux membres à gauhe par P on obtient

$$\begin{aligned} P \times A \times U &= P \times B \text{ soit} \\ R \times U &= H \end{aligned}$$

De même toute solution de $R \times X = H$ est solution de $A \times X = B$, en effet notons de même U une solution de $R \times X = H$, on a

$$\begin{aligned} R \times U &= H \text{ donc} \\ P \times A \times U &= P \times B \end{aligned}$$

et en multipliant les deux membres à gauche par P^{-1} on obtient

$$A \times U = B$$

On a bien prouvé que les deux systèmes $A \times X = B$ et $R \times X = H$ sont équivalents. Il nous reste à résoudre le système $R \times X = H$.
Continuons, si besoin, le calcul de la ℓ-réduite échelonnée de T et notons T'' le résultat :

$$T' = [R \mid H], T'' = [R \mid H']$$

Si on arrive à faire apparaître un pivot dans la dernière colonne, les opérations sur les lignes ne modifieront pas la partie gauche car la partie gauche de cette ligne pivot est constituée de zéros.

— Si le système est homogène (tous les seconds membres sont nuls) on a forcément $T' = T''$ et le système admet, rappelons-le, au moins la solution nulle.
— Si le rang de T est supérieur au rang de R (c'est donc que le rang de T est d'une unité supérieure au rang de A, on ne peut créer qu'un seul nouveau pivot au maximum) cela signifie qu'il y a plus de lignes nulles dans R que dans T'' et par conséquent que le système $R \times X = H'$ contient l'équation du type

$$0x_1 + 0x_2 + \cdots + 0x_p = 1$$

ou que le système $R \times X = H$ contient au moins une équation du type

$$0x_1 + 0x_2 + \cdots + 0x_p = h \text{ avec } h \neq 0$$

ce qui est impossible et le système n'admet pas de solution.
— Si $\text{rang}(T) = \text{rang}(R) = r$, c'est que $T' = T''$ et la résolution de $A \times X = B$ équivaut à la résolution des r équations non nulles de $R \times X = H$. Deux cas peuvent se présenter :

1. $r = p$, le système à résoudre est alors de la forme :

$$\begin{cases} x_1 & & & = h_1 \\ & x_2 & & = h_2 \\ & & \vdots & \vdots \\ & & x_p & = h_p \end{cases}$$

et il est résolu, il y a unicité de la solution.

2. $r < p$. Nous appelons inconnues pivots ou inconnues principales (IP) les r inconnues $x_{j_1}, x_{j_2}, \cdots, x_{j_r}$ correspondant aux r colonnes pivots $j_1, j_2, \cdots, j_r$ de R. Les $p - r$ autres inconnues sont appelées inconnues non principales (INP), certains les appellent aussi des inconnues ou des variables libres ou encore des paramètres. Pour résoudre le système la méthode consiste à faire passer dans les seconds membres les inconnues non principales, le système devient alors un sytème de Cramer résolu dont les solutions dépendent des $p - r$ inconnues non principales (ces inconnues non principales sont, à ce stade, considérées comme des paramètres), **il y a donc une infinité de solutions** ; si l'on désire une solution particulière, il suffit de donner des valeurs arbitraires aux inconnues non principales.

$$\begin{cases} x_{j_1} & & & = h_{j_1} + e_{j_1} \\ & x_{j_2} & & = h_{j_2} + e_{j_2} \\ & & \vdots & \vdots \\ & & x_{j_r} & = h_{j_r} + e_{j_r} \end{cases}$$

où les e_{j_i} sont des expressions linéaires des $(p - r)$ INP.

Il faut remarquer que si l'on modifie l'ordre d'écriture des inconnues nous n'obtiendrons pas forcément les mêmes inconnues principales et non principales mais l'ensemble solution sera évidemment le même.

EXERCICES

Exercice 2 - 1

L'ensemble des matrices à coefficients dans un corps $\mathbb{K}$ de la forme :

$$\begin{pmatrix} a & a+b \\ a+b & b \end{pmatrix}$$

muni de l'addition matricielle ainsi que de la multiplication par un scalaire est-il un espace vectoriel ?

Même question avec les matrices de la forme $\begin{pmatrix} a & 0_{\mathbb{K}} \\ 1_{\mathbb{K}} & b \end{pmatrix}$ et celles de la forme $\begin{pmatrix} 0_{\mathbb{K}} & a \\ b & 0_{\mathbb{K}} \end{pmatrix}$

Exercice 2 - 2

On considère les matrices suivantes à coefficients réels :

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 2 & -1 & 2 \\ 0 & 1 & 3 \end{pmatrix}, B = \begin{pmatrix} 2 & 0 \\ 1 & 5 \\ -1 & -3 \end{pmatrix}, C = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$$

$$D = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}, E = \begin{pmatrix} 1 & 2 & 4 & -3 \\ 1 & -2 & 0 & 5 \\ 0 & 3 & 1 & 6 \end{pmatrix}, F = \begin{pmatrix} 7 & 0 \\ 5 & -6 \\ -3 & 0 \\ -1 & 1 \end{pmatrix}, G = \begin{pmatrix} 1 & 2 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 1 \end{pmatrix}$$

1. Calculer : $A - 2G$, $3A + 2G$, ${}^tA$, ${}^tG$.

2. Calculer $B \times A$, $A \times B$, $E \times F$, $D \times C$, $C \times D$, A^2, $A \times G$, $G \times A$.

Exercice 2 - 3

Soit $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ et $B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ deux matrices booléennes. Calculez $A \times B$, $B \times A$, A^2.

Exercice 2 - 4 Matrices et relations

Considérons par exemple la relation $\mathcal{R}$ définie par son dictionnaire d'association :

De	Vers
a	γ
b	β,δ
c	γ
d	δ

Donnez sa représentation matricielle.
Donnez la représentation matricielle d'une relation incluse dans $\mathbb{R}$.
À votre avis, pourquoi le nom « transposée » a-t-il été bien choisi pour désigner la relation réciproque ?
Donnez la représentation matricielle de se transposée. Généralisez.
Quelle est la représentation matricielle de la négation de $\mathcal{R}$?

Soit la relation $\mathcal{R}'$:

De	Vers
a	α, γ
b	α, β, δ
c	β, δ
d	α, β, γ

Quelle est la représentation matricielle de $\mathcal{R} \cap \mathcal{R}'$? De $\mathcal{R} \cup \mathcal{R}'$?
Quelle est le rapport entre composition de relations et calcul matriciel ?
Quelle est la représentation matricielle de $\mathcal{R} \circ {}^t\mathcal{R}'$?

Exercice 2 - 5

1. Donner la représentation matricielle des relations suivantes définies sur $\{1, 2, 3, 4\}$:

i. $\{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\}$;

ii. $\{(1, 1), (1, 4), (2, 2), (3, 3), (4, 1)\}$;

iii. $\{(1, 2), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4), (3, 1), (3, 2), (3, 4), (4, 1), (4, 2), (4, 3)\}$;

iv. $\{(2, 4), (3, 1), (3, 2), (3, 4)\}$

2. Quelles sont, parmi ces relations, celles qui sont réflexives ? irrefléxives ? symétriques ? antisymétriques ? transitives ? Donner des algorithmes qui répondent à ces questions en prenant les relations en arguments et en effectuant des calculs sur les matrices.

Exercice 2 - 6

Soit $\mathcal{E} = \{1, 2, 3, 4, 5\}$ et $\mathcal{R}$ une relation sur $\mathcal{E}$ qui contient les couples $(1, 3)$, $(2, 4)$, $(3, 1)$, $(3, 5)$, $(4, 3)$, $(5, 1)$, $(5, 2)$ et $(5, 4)$. Déterminez :

1. $\mathcal{R}^2$

2. $\mathcal{R}^3$

3. $\mathcal{R}^4$

4. $\mathcal{R}^5$

5. $\mathcal{R}^6$

Utilisez le calcul matriciel.

Exercice 2 - 7

A et B sont deux matrices carrées d'ordre n, développer

1. $(A + B)^2$

2. $(A + B)^3$

Exercice 2 - 8

$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$, calculer A^n.

Exercice 2 - 9

$A \in \mathbb{A}^{3\times 2}$, peut-on trouver une matrice B de sorte que $A \times B = B \times A$?

Exercice 2 - 10

$A = (a_{ij}) \in \mathbb{R}^{n\times n}$. Expliciter la matrice A dans les cas suivants :

1. $a_{ij} = i + j$ si $i > j$ et $a_{ij} = 0$ sinon.

2. $a_{ij} = j$ si $i \leq j$ et $a_{ij} = i$ si $i > j$.

3. $a_{ij} = |i - j - 1|$

Exercice 2 - 11

$A = (a_{i,j}) \in \mathbb{A}^{n\times n}$ et $B = {}^t A \times A$.

1. Démontrer que B est symétrique.

2. $B = (b_{i,j})$, calculer $b_{i,j}$ en fonction des $a_{k,l}$.

Exercice 2 - 12

$A = (a_{i,j})$ est une matrice carrée d'ordre n à coefficients réels. $D = (d_{i,j})$ est une matrice carrée diagonale d'ordre n à coefficients réels ayant ses éléments diagonaux tous distincts. Démontrer :

$$A \times D = D \times A \rightarrow A \text{ est diagonale.}$$

Exercice 2 - 13

$A = (a_{ij}) \in \mathbb{R}^{n \times n}$, on appelle trace de A la somme de ses éléments diagonaux que l'on note $\operatorname{tr}(A)$ ce nombre : $\boldsymbol{tr}(A) = \sum_{i=1}^{n} a_{ii}$. Démontrer :

$$A = (a_{ij}) \in \mathbb{R}^{n \times n} \text{ et } B = (a_{ij}) \in \mathbb{R}^{n \times n} \rightarrow \boldsymbol{tr}(A \times B) = \boldsymbol{tr}(B \times A)$$

Exercice 2 - 14

$A = (a_{ij}) \in \mathbb{R}^{n \times n}$, calculer la trace de ${}^tA \times A$.

Exercice 2 - 15

$$A = \begin{pmatrix} 0 & 1 & -1 \\ -3 & 4 & -3 \\ -1 & 1 & 0 \end{pmatrix}$$

1. Calculer A^2.

2. Calculer $A^2 - 3A + 2I_3$.

3. En déduire que A est inversible et déterminer A^{-1}.

Exercice 2 - 16

Déterminer les ℓ-réduites échelonnées (lré) des matrices suivantes et donner leur rang :

1. $A = \begin{pmatrix} 1 & 2 & 0 & 1 \\ -1 & 2 & 1 & 1 \\ 3 & 1 & 0 & -2 \end{pmatrix}$

2. $A = \begin{pmatrix} 1 & 1 & 2 & 2 & 2 & 2 \\ 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 2 & 1 & -1 & 0 & -1 & 2 \end{pmatrix}$

3. $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

4. $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 5 & 10 \end{pmatrix}$

Exercice 2 - 17

Les matrices suivantes sont-elles inversibles ? On cherchera, pour chacune d'elles, l'inverse par la méthode de Gauss.

1. $A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$

2. $B = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 2 \\ -1 & 1 & 4 \end{pmatrix}$

3. $C = \begin{pmatrix} 1 & 2 & -1 \\ -1 & -1 & 2 \\ 2 & 1 & 1 \end{pmatrix}$

4. $D = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

Exercice 2 - 18

Soit $D = \lceil d_{1,1}, d_{2,2}, \dots, d_{n,n} \rfloor$ une matrice diagonale, démontrer que D est inversible si, et seulement si, $\prod_{i=1}^{n} d_{i,i} \neq 0$.

Exercice 2 - 19

a) Soit $U = \begin{pmatrix} 0 & 0 & 5 \\ 1 & 0 & 2 \\ 0 & 1 & 0 \end{pmatrix}$. Exprimer de façon simple la matrice U^3 en fonction de U^2, U et I_3. En déduire que U est inversible et donner U^{-1}.

b) Soit $U = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$. Déterminer $(a, b) \in \mathbb{R}^2$ tel que $(U - aI_3)(U - bI_3) = O_3$. En déduire que U est inversible et donner U^{-1}.

c) Inverser les matrices suivantes :

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 3 \end{pmatrix} \quad C = \begin{pmatrix} i & 2 & -3 \\ 0 & i & 2 \\ 0 & 0 & i \end{pmatrix} \quad D = \begin{pmatrix} 2 & 2 & 3 \\ 1 & -1 & 0 \\ -1 & 2 & 1 \end{pmatrix}$$

Exercice 2 - 20

Ecrire matriciellement de deux façons les systèmes suivants :

1. $\begin{cases} x - 2y + 3z = 2 \\ 2x - 5y - 4z = -1 \\ 4y - 3z = 8 \end{cases}$

2. $\begin{cases} 3x - z = 6 \\ 5y - 4z = -3 \\ 4x - 3y = 7 \end{cases}$

3. $\begin{cases} 5x + 2y + 3z = 2 \\ 2x - 5y - 6z = -2 \\ 4y - 3z = 8 \\ 4x + 3y + 2z = 0 \end{cases}$

Exercice 2 - 21

Résoudre le système :

$$\begin{cases} x - 2y + z = a \\ 2x - 3y - 2z = b \\ x - y + z = c \end{cases}$$

obligatoirement par la méthode Gauss-Jordan où x, y et z sont les inconnues. En déduire que la matrice $A = \begin{pmatrix} 1 & -2 & 1 \\ 2 & -3 & -2 \\ 1 & -1 & 1 \end{pmatrix}$ est inversible et donner son inverse.

Exercice 2 - 22

Résoudre les systèmes suivants par la méthode de Gauss-Jordan :

1. $\begin{cases} x + 2y + 3z = 1 \\ 4x + 5y + 6z = 2 \\ 7x + 8y + 9z = 3 \end{cases}$

2. $\begin{cases} x + 2y + 3z = 1 \\ 4x + 5y + 6z = 2 \\ 5x + 7y + 9z = 3 \end{cases}$

3. $\begin{cases} x + 2y + 3z + 4t = 1 \\ 4x + 5y + 6z + 2t = 2 \\ 7x + 8y + 9z - t = 3 \end{cases}$

4. $\begin{cases} x + y + z = 1 \\ x - y - 2z = 3 \\ 2x + 3y + 5z = 0 \\ 4x + 3y + 4z = 4 \end{cases}$

5. $\begin{cases} x + y + z = 1 \\ x - y - 2z = 3 \\ 2x + 3y + 5z = 0 \\ 4x + 3y + 4z = 1 \end{cases}$

6. $\begin{cases} x + y + z = 1 \\ x - y - 2z = 3 \\ 2x - z = 4 \\ 3x + y = 5 \end{cases}$

Exercice 2 - 23

Déterminer le rang des systèmes linéaires d'inconnues réelles suivants, en fonction du paramètre $m \in \mathbb{R}$. On donnera l'ensemble des solutions.

$$a) \begin{cases} mx + y + z = m \\ x + my + z = m \\ x + y + mz = m \end{cases} \quad b) \begin{cases} (m+1)x + my = 2m \\ mx + (m+1)y = 1 \end{cases} \quad c) \begin{cases} x - my + m^2z = 2m \\ mx - m^2y + mz = 2m \\ mx + y - m^2z = 1 - m \end{cases}$$

Exercice 2 - 24

Soit la matrice $A = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$.

a) Déterminer A^n pour tout $n \in \mathbb{N}$.

b) On considère les deux suites $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ définies par la donnée de u_0 et v_0 et les relations de récurrence

$$\begin{cases} u_{n+1} &= u_n - v_n \\ v_{n+1} &= -u_n + v_n \end{cases}$$

(i) On pose $W_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}$. Etablir une relation entre W_{n+1}, A et W_n.

(ii) En déduire une expression de u_n et v_n en fonction de u_0, v_0 et n pour tout $n \in \mathbb{N}$.

Exercice 2 - 25

Soit $A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$, $B_1 = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$ et $B_2 = \begin{pmatrix} 32,1 \\ 22,9 \\ 33,1 \\ 30,9 \end{pmatrix}$

Résolvez les systèmes $A \times X = B_1$ et $A \times X = B_2$: commentaires ?

Exercice 2 - 26

Soit $A = \begin{pmatrix} 10^{-16} & 1 \\ 1 & 1 \end{pmatrix}$ et $B = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$. Supposons que les nombres flottants soient représentés avec 10 chiffres significatifs.

Résolvez le système $A \times X = B$ en choisissant 10^{-16} comme premier pivot. Recommencez en commençant par permuter les deux lignes puis en prenant 1 comme pivot.

Généralisez en remplaçant 10^{-16} par un nombre strictement positif quelconque ε : des commentaires ?

Exercice 2 - 27

Dans toute la suite U désigne une suite de 4 bits qui peut être représentée par la matrice

$$U = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix}$$

et la transposée de U est notée X :

$${}^tU = X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

et il est bien entendu que les éléments x_i sont considérés comme des éléments du corps $\mathbb{F}_2$ en ce sens que $1+1=0$.
Pour améliorer la sécurité de la transmission des chaînes de 4 bits on décide de transformer le message U en

$$V = U \times B$$

où B est une matrice (judicieusement choisie) à coefficients dans $\mathbb{F}_2$ ou, si l'on préfère, on transforme le message X en $Y = A \times X$

1. Quelle relation existe-t-il entre Y et V ? Entre A et B ?
2. Si on veut que $V = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_2 \end{pmatrix}$, donner la matrice B.
3. Si on veut que $Y = \begin{pmatrix} x_1 \\ x_1 + x_2 \\ x_3 \\ x_{1+}x_4 \\ x_2 \\ x_3 \end{pmatrix}$, donner la matrice A.
4. On veut rajouter au message U un bit de parité paire (pour un bit de parité impaire, ce qui suit est impossible), décrire alors V et donner alors B ?
5. On veut que $V = \begin{pmatrix} x_4 & x_3 & x_2 & x_1 & x_1 & x_2 & x_3 & x_4 \end{pmatrix}$, déterminer B.

6. Quelqu'un décide de choisir

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

i. Calculer les images des éléments suivants :

U	V
1111	
1100	
0011	
0110	
0101	
0010	
1010	
0100	

ii. Qelle remarque faites-vous sur le choix de B ?

iii. Donner une matrice $B^{'}$ ayant la même taille que B et qui ne possède pas à coup sûr le défaut précédent.

iv. Donner une matrice B'' ayant la même taille que B et ayant le défaut (même pire si vous le voulez) de B.

Exercice 2 - 28

Soit $\mathcal{B} = (i, j)$ une base du plan ou $\mathcal{B} = (i, j, k)$ une base de l'espace, nous savons que tout vecteur se décompose de façon unique dans la base $\mathcal{B}$. Nous fixons un point O appelé origine, pour tout point M de notre plan ou espace, le vecteur $\overrightarrow{OM}$ se décompose de façon unique dans la base $\mathcal{B}$ et nous obtenons les coordonnées du vecteur $\overrightarrow{OM}$ qui correspondent au coordonnées du point M dans le repère $\mathcal{R} = (O, i, j)$ ou $\mathcal{R} = (O, i, j, k)$.
Ces coordonnées se notent, respectivement, par

$$M \begin{pmatrix} x \\ y \end{pmatrix}_{\mathcal{R}}, \ M \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\mathcal{R}}$$

Un changement de repère peut concerner un changement d'origine ou un changement de base ou les deux. Soit donc le repère $\mathcal{R} = (O, i, j, j)$ avec $\mathcal{B} = (i, j, k)$ la base associée et $\mathcal{R}' = (O', i', j', k')$ un autre repère avec $\mathcal{B}' = (i', j', k')$ la base associée. Nous notons

$$M \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\mathcal{R}}, M \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\mathcal{R}'}, O' \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}_{\mathcal{R}} \text{ et } M \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}_{\mathcal{R}_1 = (O, i', j', k')}$$

et $P = (p_{i,j})$ la matrice de passage de $\mathcal{B}$ à $\mathcal{B}'$, c'est-à-dire la matrice des coordonnées des vecteurs de $\mathcal{B}'$ dans la base $\mathcal{B}$.

1. Quelle sont les coordonnées de O dans le repère $\mathcal{R}$?

2. Démontrer que P est inversible en cherchant les coordonnées des vecteurs de $\mathcal{B}$ dans la base $\mathcal{B}'$.

3. On note $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, X' = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}, X_1 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$. Démontrer que $X_1 = P \times X$.

4. Déterminer X' en fonction de X et de P.

Exercice 2 - 29

Les déterminants peuvent servir à calculer certaines surfaces. On peut montrer par exemple que l'aire du triangle dont les sommets sont les points $A(x_1, y_1)$, $B(x_2, y_2)$ et $C(x_3, y_3)$ est égal à la valeur absolue de

$$\frac{1}{2} \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

Quelle est l'aire du triangle dont les sommets ont pour coordonnées (2,3), (1,1) et (5,-1) ? (3,4), (2,1) et (4,7) ? Qu'en concluez-vous dans ce dernier cas ?

Exercice 2 - 30

Soit $A = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}$ et $B = \begin{bmatrix} 0 & 0 \\ 3 & 4 \end{bmatrix}$

Calculez $\det(A)$, $\det(B)$ et $\det(A+B)$. Conclusion ?

Exercice 2 - 31

Résolvez les équations suivantes dans $\mathbb{R}$:

1. $\begin{vmatrix} 3 & x \\ x+2 & 1-x \end{vmatrix} = -3$
2. $\begin{vmatrix} x-2 & 3 \\ x & x+1 \end{vmatrix} = 3$

Exercice 2 - 32

Pour quelles valeurs de x les matrices suivantes sont-elles singulières ?

1. $M_1 = \begin{bmatrix} 1 & x-1 \\ x & 3x \end{bmatrix}$
2. $M_2 = \begin{bmatrix} 2x & x+1 \\ x-2 & 3x \end{bmatrix}$
3. $M_3 = \begin{bmatrix} 1 & x & x \\ 1 & 2 & 3 \\ x & x & 1 \end{bmatrix}$
4. $M_4 = \begin{bmatrix} 1 & x & x^2 \\ 1 & 1 & 1 \\ 1 & 3 & 9 \end{bmatrix}$

Exercice 2 - 33

$A_{ij} = \begin{cases} 2 \text{ si } i \leqslant j \\ 0 \text{ si } i > j \end{cases}$. Calculer $\det(A)$. Faites de même si $A_{ij} = \begin{cases} i \text{ si } i = j \\ 0 \text{ si } i \neq j \end{cases}$.

Exercice 2 - 34

Soit $M = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$ et $\det(M) = -4$. Calculez le déterminant des matrices suivantes :

$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ $\quad$ $C = \begin{bmatrix} 3a & d & -g \\ 3b & e & -h \\ 3c & f & -i \end{bmatrix}$

$B = \begin{bmatrix} g & a & d \\ h & b & e \\ i & c & f \end{bmatrix}$ $\quad$ $D = \begin{bmatrix} a & d & g \\ 4b+3a & 4e+3d & 4h+3g \\ c & f & i \end{bmatrix}$

Exercice 2 - 35

Soient A, B et P trois matrices de même ordre telles que $B = P^{-1}AP$. Calculez le déterminant de B en fonction de celui de A.

Exercice 2 - 36

Une matrice est dite **orthogonale** si, et seulement si, sa transposée est égale à son inverse.

1. La matrice I_n est-elle orthogonale ?
2. La matrice $A = \begin{pmatrix} \sin(t) & \cos(t) \\ -\cos(t) & \sin(t) \end{pmatrix}$ est-elle orthogonale pour tout $t \in \mathbb{R}$?
3. Quelles valeurs peut prendre le déterminant d'une matrice orthogonale ?
4. Déterminez la troisième ligne de la matrice orthogonale : $\begin{pmatrix} 1/3 & 2/3 & 2/3 \\ 2/3 & -2/3 & 1/3 \\ & & \end{pmatrix}$

Matrices et Haskell

Comment modéliser ?

Au moment de choisir une modélisation pour nos matrices, l'informaticien(ne) se pose des questions : vais-je choisir un type **Array** mutable ou un type **List** immuable ?
Ça se discute. Le premier peut paraître plus maniable a priori : on a accès à tout le tableau construit statiquement ce qui ne nous gène pas car on connaît la taille de la matrice a priori.
D'un autre côté, un type mutable est dangereux : il laisse la porte ouverte aux effets de bord, source de nombreux bugs car souvent mal maîtrisés. De plus, la machine ne gère pas très efficacement les tableaux statiques.
Tournons-nous vers les listes : un langage muni d'un bon ramasse-miettes les traitera très efficacement. Elles permettent de créer dynamiquement les matrices ce qui engendre plus de souplesse et cela sans effets de bord ce qui assure plus de sécurité dans la programmation. On cherchera de plus à être le plus polymorphe possible. Nous perdrons en efficacité pour le produit mais nous serons très efficace pour l'algorithme de Gau ß, la résolution de systèmes, le calcul de déterminant.
Nous choisirons donc de programmer sans effet de bord : du fonctionnel aussi pur que possible. Cela entraîne un changement des habitudes : au lieu de raisonner en termes de grosses boucles imbriquées et pas toujours contrôlées, nous utiliserons des petits modules fonctionnels. Cela pourra avoir quelques désavantages dans certains cas en termes d'efficacité mais cela nous fournira un bon entraînement pour nous habituer au paradigme fonctionnel.
Nous essaierons dans une certaine mesure de créer des fonctions polymorphes : cela pourra passer pour une contrainte forte au départ mais cela vous facilitera la vie pour votre mini-projet sur le chiffrement de HILL : vous n'aurez qu'à changer d'anneau mais toutes les fonctions sur les matrices resteront valables sans aucun changement !
Nous ne délaisserons pas la voie plus impérative que nous explorerons avec Sage mais dans le prochain module.

Constructeur

Nous utiliserons un constructeur pour définir notre type :

Haskell
```haskell
data Matrice a = Mat [[a]]
```

Une matrice sera donc déterminée par un type **a** qui correspond aux éléments de l'anneau sur lequel sont définies les matrices. La structure sera celle d'une liste de lignes ou plutôt un « vecteur constitué de vecteurs ».

Haskell
```haskell
*Main> let m = Mat [[1,2,3],[4,5,6]]
*Main> :t m
m :: Matrice Integer
*Main> let m' = Mat [[True,True,False],[True,False,False]]
*Main> :t m'
m' :: Matrice Bool
```

Joli affichage

Si l'on veut voir la matrice via GHCi :

Haskell
```haskell
<interactive>:19:1:
    No instance for (Show (Matrice Integer))
      arising from a use of `print'
    Possible fix:
      add an instance declaration for (Show (Matrice Integer))
    In a stmt of an interactive GHCi command: print it
```

On nous explique que le compilateur ne sait pas afficher les objets de type **Matrice**.
Comme c'est de la bidouille d'affichage, je vous donne une solution. Étudiez-la malgré tout car elle nous donnera une méthode de programmation sur nos matrices.
On rajoute dans notre en-tête :

Haskell
```haskell
import qualified Text.Printf as Pp
```

On commence par donner une fonction qui affiche les lignes :

Haskell
```
montre []    = " | "
montre (t:q) = " " ++ (Pp.printf "%5s" (show t)) ++ " " ++ (montre q)
```

Haskell
```
*Main> montre [1,2,3]
"     1      2      3  | "
```

On fait de **Matrice** une instance de la classe **Show** :

Haskell
```
instance (Show a) => Show (Matrice a) where
  show (Mat ([]))  = "  "
  show (Mat (t:q)) = " | " ++ (montre t) ++ "\n" ++ (show (Mat q))
```

Maintenant, on peut voir nos matrices :

Haskell
```
*Main> let m = Mat [[1,2,3],[4,5,6]]
*Main> m
 |      1      2      3  |
 |      4      5      6  |
```

Les opérations de base

Il nous faut construire une somme et un produit de matrice et être assez efficace (un produit de matrices de taille 100 sera quasi instantané, un produit de matrices de taille 500 nous prendra 1 seconde mais avec des matrices de taille 1000, ce sera de l'ordre de 30 secondes).
L'avantage cependant, c'est que le code sera très rapide à écrire : nous entrons dans le monde des fameux « *one liners* ».
Vous utiliserez donc au mieux les fonctions **zipWith**, **sum** et **transpose** pour écrire une somme et un produit de matrices en une ligne à chaque fois.

Haskell
```
-- produit de deux matrices
(&*) :: (Num a) => Matrice a -> Matrice a -> Matrice a
(&*) (Mat m) (Mat n) = Mat ...?

-- somme terme à terme
(&+) :: (Num a) => Matrice a -> Matrice a -> Matrice a
(&+) (Mat m) (Mat n) = Mat ...?
```

Découverte d'un foncteur

Vous connaissez maintenant la fonction **map** qui applique une fonction à tous les éléments d'une liste. C'est en fait un cas particulier de foncteur, une sorte de super-fonction qui s'applique à des types de variables complexes.
Nous voudrions nous aussi avoir un foncteur qui applique une fonction à tous les éléments d'une matrice.
Par exemple :

Haskell
```
*Main> fmap (\x -> x*x) (Mat [[1,2,3],[4,5,6]])
 |        1        4        9   |
 |       16       25       36   |
```

Créez cette fonction **fmap** dont nous ferons une instance de la classe **Functor** :

Haskell
```
instance Functor Matrice where
  fmap f (Mat m) =
    Mat ... ?
```

Pour aller plus loin

On peut aussi rendre nos matrices « pliables » en en faisant une instance de **Foldable**. C'est un peu plus abstrait à appréhender. Vous pouvez étudier ce problème pendant vos temps libres.
Ici, cela donnerait :

Haskell
```haskell
import qualified Data.Foldable as F

instance F.Foldable Matrice where
  foldMap f (Mat [])     = M.mempty
  foldMap f (Mat (t:q)) = (F.foldMap f (Mat q)) `M.mappend` (F.foldMap f t)
```

Est-ce un groupe ? Est-ce un anneau ?

Pouvez-vous faire de **Matrice** une instance de **Groupe** ? **Anneau Corps** ? **Num** ? **Fractional** ?

Liste par compréhension

Construisez des fonctions qui fabriquent des matrices identité, nulle, et attila.

Haskell
```haskell
-- matrice unité
unite :: (Num a) => Int -> Matrice a
unite n = ...?

-- matrice carrée nulle
carree_nulle :: (Num a)  => Int -> Matrice a
carree_nulle n =  ...?

-- matrice remplie de 1
attila :: (Num a) => Int -> Int -> Matrice a
attila row col = ...?
```

?

What is this, as we say in english of speciality...

Haskell
```haskell
f1 :: (Eq a, Num a) => Matrice a -> Matrice a
f1 (Mat (r:rs))
  | ((head r) /= 0) || (all (== 0) (head (List.transpose (r:rs)))) =
    Mat (r:rs)
  | otherwise       =
        f1 (Mat (rs ++  [map (* ((-1) ^ (length rs))) r]))

f2 (Mat []) = error "Ceci est une erreur"
f2 (Mat [[a]]) = a
f2 m =
  if c == 0 then
    0
  else
    c * (f2 (Mat rs'))
  where
    Mat ((c:l):rs) = f1 m
    rs' = map f rs
    f (t:q)
      | t == 0      = q
      | otherwise   = fmap (/ c) (tail $ zipWith (-) (map (*c) (t:q)) (map (* t) (c:l)))
```

Inverse

C'est le gros morceau... Inspirez-vous de ce qui vient d'être fait pour calculer l'inverse d'une matrice.