Structures algébriques et programmation

INFO1 - Semaines 45 à 3

Guillaume CONNAN

IUT de Nantes - Dpt d'informatique

Dernière mise à jour : 16 décembre 2014 à 17:40

Sommaire

Sommaire


كتاب الجبر

ما ذكره المصنف الشيخ الأجل أبو عبد الله
محمد بن موسى الخوارزمي رضي الله عنه وأثابه رحمته

وبيده لاستمداد نوبه وخطاياه العبد المعترف
الى الله المعين به خطاب بن محمد بن علي
ابن حبيب بن علي بن محمد بن علي بن احمد بن
جعفر بن الحسين بن يحيى بن ابرهيم بن محمد بن
ابرهيم بن احمد بن المغيرة بن عمران بن عاصم بن
الوليد بن هشيم بن رسعد بن عدسس بن
عبد مناف

سعد الله بالعلم والعمل
الصالحين

وحسبنا الله ونعم الوكيل

الجبر

كتاب المختصر في حساب الجبر والمقابلة

```
> versMin 'A'
'a'
.
.
```

```
> versMin 'A'
'a'
> map versMin "TRALALAHOP"
"tralalahop"
```

```
import Data.Char
import Data.Bits

.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
.
.
.
.
.
.
.
.
.
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
.
.
.
.
.
.
.
.
.
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
.
.
.
.
.
.
.
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
> chr 65
.
.
.
.
.
.
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
> chr 65
'A'
.
.
.
.
.
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
> chr 65
'A'
> chr (65 + 32)
.
.
.
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
> chr 65
'A'
> chr (65 + 32)
'a'
.
.
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
> chr 65
'A'
> chr (65 + 32)
'a'
> 65 .|. 32
.
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
> chr 65
'A'
> chr (65 + 32)
'a'
> 65 .|. 32
97
.
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
> chr 65
'A'
> chr (65 + 32)
'a'
> 65 .|. 32
97
> setBit 65 5
.
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
> chr 65
'A'
> chr (65 + 32)
'a'
> 65 .|. 32
97
> setBit 65 5
97
.
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
> chr 65
'A'
> chr (65 + 32)
'a'
> 65 .|. 32
97
> setBit 65 5
97
> 97 .&. (complement 32)
.
```

```
import Data.Char
import Data.Bits

> ord 'A'
65
> ord 'a'
97
> chr 65
'A'
> chr (65 + 32)
'a'
> 65 .|. 32
97
> setBit 65 5
97
> 97 .&. (complement 32)
65
```

```haskell
versMin :: Char                          -> Char
versMin    c
  | (c < 'A') || (c > 'Z') = error "Pas majuscule !"
  | otherwise              = chr (setBit (ord c) 5)
```

Définition 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 **muni de cette loi** est un **magma**.

Définition 2 (Stabilité)

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

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

```haskell
somCar :: (Char,Char) -> Char
somCar     (a    ,b)     =  chr ((ord a) + (ord b))
```

```
> somCar ('A',' ')
'a'
> somCar ('3','2')
'e'
```

```haskell
somCar :: (Char,Char) -> Char
somCar     (a   ,b)     =  chr ((ord a) + (ord b))
```

```
> somCar ('A',' ')
'a'
> somCar ('3','2')
'e'
```

```
infix @+

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

```
> 'A' @+ ' '
'a'
```

```
infix @+

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

```
> 'A' @+ ' '
'a'
```

Définition 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.

```
data Vect2D = Vect(Double,Double) deriving Show
.
.
.
```

```haskell
data Vect2D = Vect(Double,Double) deriving Show

(.@) :: Double -> Vect2D        -> Vect2D
(.@)    k       (Vect(x,y)) = Vect(k*x,k*y)
```

```
> let v = Vect(2,3)
> 5 .@ v
Vect (10.0,15.0)
```

Définition 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 :

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

```
> :t (++)
.
.
.
.
.
.
.
```

```
> :t (++)
(++) :: [a] -> [a] -> [a]
.
.
.
.
.
.
.
```

```
> :t (++)
(++) :: [a] -> [a] -> [a]
> :t (+)
.
.
.
.
.
```

```
> :t (++)
(++) :: [a] -> [a] -> [a]
> :t (+)
(+) :: Int -> Int -> Int
.
.
.
.
```

```
> :t (++)
(++) :: [a] -> [a] -> [a]
> :t (+)
(+) :: Int -> Int -> Int
> :t length
.
.
.
```

```
> :t (++)
(++) :: [a] -> [a] -> [a]
> :t (+)
(+) :: Int -> Int -> Int
> :t length
length :: [a] -> Int
.
.
```

```
> :t (++)
(++) :: [a] -> [a] -> [a]
> :t (+)
(+) :: Int -> Int -> Int
> :t length
length :: [a] -> Int
> length ("Ti" ++ "Poney") == length "Ti" + length "Poney"
.
```

```
> :t (++)
(++) :: [a] -> [a] -> [a]
> :t (+)
(+) :: Int -> Int -> Int
> :t length
length :: [a] -> Int
> length ("Ti" ++ "Poney") == length "Ti" + length "Poney"
True
```

Définition 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 isomorphisme de E dans F si, et seulement si, c'est un morphisme BIJECTIF de E dans F.

```haskell
somL :: [Double] -> [Double] -> [Double]
somL    l1          l2       =  zipWith (+) l1 l2
.
.
.
.
.
.
```

```haskell
somL :: [Double] -> [Double] -> [Double]
somL    l1          l2       =  zipWith (+) l1 l2

somV :: Vect2D      -> Vect2D        -> Vect2D
somV    (Vect (x,y)) (Vect (x',y')) = Vect (x + x', y + y')
.
.
.
```

```haskell
somL :: [Double] -> [Double] -> [Double]
somL    l1          l2       =  zipWith (+) l1 l2

somV :: Vect2D      -> Vect2D        -> Vect2D
somV    (Vect (x,y)) (Vect (x',y')) = Vect (x + x', y + y')

vectToList :: Vect2D       -> [Double]
vectToList    (Vect (x,y)) =  [x,y]
```

```
> let v1 = Vect (1,2)
> let v2 = Vect (10,15)
> vectToList (v1 `somV` v2) == vectToList v1 `somL` vectToList
  ↪  v2
True
```

Définition 6 (Commutativité)

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

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

Définition 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.

Définition 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 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 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 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 ?

Définition 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 ?

Définition 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 ?

Définition 10 (Régularité)

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

$$\big(\forall x\big)\big(\forall y\big)\big((a \star x = a \star y) \rightarrow (x = y)\big)$$

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

Définition 10 (Régularité)

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

$$\big(\forall x\big)\big(\forall y\big)\big((a \star x = a \star y) \rightarrow (x = y)\big)$$

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

Définition 10 (Régularité)

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

$$\big(\forall x\big)\big(\forall y\big)\big((a \star x = a \star y) \rightarrow (x = y)\big)$$

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

Définition 11 (Distributivité)

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

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

Définition 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 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 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 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 14 (Élément idempotent)

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

Définition 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 14 (Élément idempotent)

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

Définition 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 14 (Élément idempotent)

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

Sommaire

É.Galois (1811-1832)

Définition 15 (Groupe)

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

Définition de (Sous-groupe)

$(H, *)$ est un sous-groupe de $(G, *)$ si, et seulement si, H est une partie de G et $(H, *)$ est un groupe.

Définition 15 (Groupe)

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

Définition 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.

Définition 15 (Groupe)

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

Définition 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.

Sommaire

Définition 17 (Congruence des entiers)

Soit n un entier naturel non nul.
Deux éléments a et b du groupe $(\mathbb{Z}, +)$ sont **congrus modulo** n si, et seulement si, ils ont le même reste dans la division par n. On note

$$a \equiv b \pmod{n}$$

ou

$$a \equiv_n b$$

et on lit « *a est congru à b modulo n* ».

« *Well, LISP seems to work okay now, modulo that GC bug.* »

Définition 17 (Congruence des entiers)

Soit n un entier naturel non nul.
Deux éléments a et b du groupe $(\mathbb{Z}, +)$ sont **congrus modulo** n si, et seulement si, ils ont le même reste dans la division par n. On note

$$a \equiv b \pmod{n}$$

ou

$$a \equiv_n b$$

et on lit « *a est congru à b modulo n* ».

« *Well, LISP seems to work okay now, modulo that GC bug.* »
« *Le LISP semble marcher maintenant, modulo le bogue du GC.* »

Définition 17 (Congruence des entiers)

Soit n un entier naturel non nul.
Deux éléments a et b du groupe $(\mathbb{Z}, +)$ sont **congrus modulo** n si, et seulement si, ils ont le même reste dans la division par n. On note

$$a \equiv b \pmod{n}$$

ou

$$a \equiv_n b$$

et on lit « *a est congru à b modulo n* ».

« Well, LISP seems to work okay now, modulo that GC bug. »
« I feel fine today modulo a slight headache. »

in « The New Hacker's Dictionnary »
http://outpost9.com/reference/jargon/jargon_28.html#SEC35

Définition 17 (Congruence des entiers)

Soit n un entier naturel non nul.
Deux éléments a et b du groupe $(\mathbb{Z}, +)$ sont **congrus modulo** n si, et seulement si, ils ont le même reste dans la division par n. On note

$$a \equiv b \pmod{n}$$

ou

$$a \equiv_n b$$

et on lit « *a est congru à b modulo n* ».

« Well, LISP seems to work okay now, modulo that GC bug. »
« I feel fine today modulo a slight headache. »

in « The New Hacker's Dictionnary »
http://outpost9.com/reference/jargon/jargon_28.html#SEC35

Définition 17 (Congruence des entiers)

Soit n un entier naturel non nul.
Deux éléments a et b du groupe $(\mathbb{Z}, +)$ sont **congrus modulo** n si, et seulement si, ils ont le même reste dans la division par n. On note

$$a \equiv b \pmod{n}$$

ou

$$a \equiv_n b$$

et on lit « *a est congru à b modulo n* ».

« Well, LISP seems to work okay now, modulo that GC bug. »
« I feel fine today modulo a slight headache. »

in « The New Hacker's Dictionnary »
http://outpost9.com/reference/jargon/jargon_28.html#SEC35

Définition 17 (Congruence des entiers)

Soit n un entier naturel non nul.
Deux éléments a et b du groupe $(\mathbb{Z}, +)$ sont **congrus modulo** n si, et seulement si, ils ont le même reste dans la division par n. On note

$$a \equiv b \pmod{n}$$

ou

$$a \equiv_n b$$

et on lit « *a est congru à b modulo n* ».

« Well, LISP seems to work okay now, modulo that GC bug. »
« I feel fine today modulo a slight headache. »

in « The New Hacker's Dictionnary »
http://outpost9.com/reference/jargon/jargon_28.html#SEC35

OCaml

```
# 17 mod 3 ;;
- : int = 2
# -17 mod 3 ;;
- : int = -2
# -17 / 3;;
- : int = -5
# (-17 / 3)*3 + (-17 mod 3) ;;
- : int = -17
```

Python

```
>>> 17 % 3
2
>>> -17 % 3
1
>>> -17/3
-6
>>> (-17 // 3)*3 + (-17 % 3)
-17
```

OCaml

```
# 17 mod 3 ;;
- : int = 2
# -17 mod 3 ;;
- : int = -2
# -17 / 3;;
- : int = -5
# (-17 / 3)*3 + (-17 mod 3) ;;
- : int = -17
```

Python

```
>>> 17 % 3
2
>>> -17 % 3
1
>>> -17/3
-6
>>> (-17 // 3)*3 + (-17 % 3)
-17
```

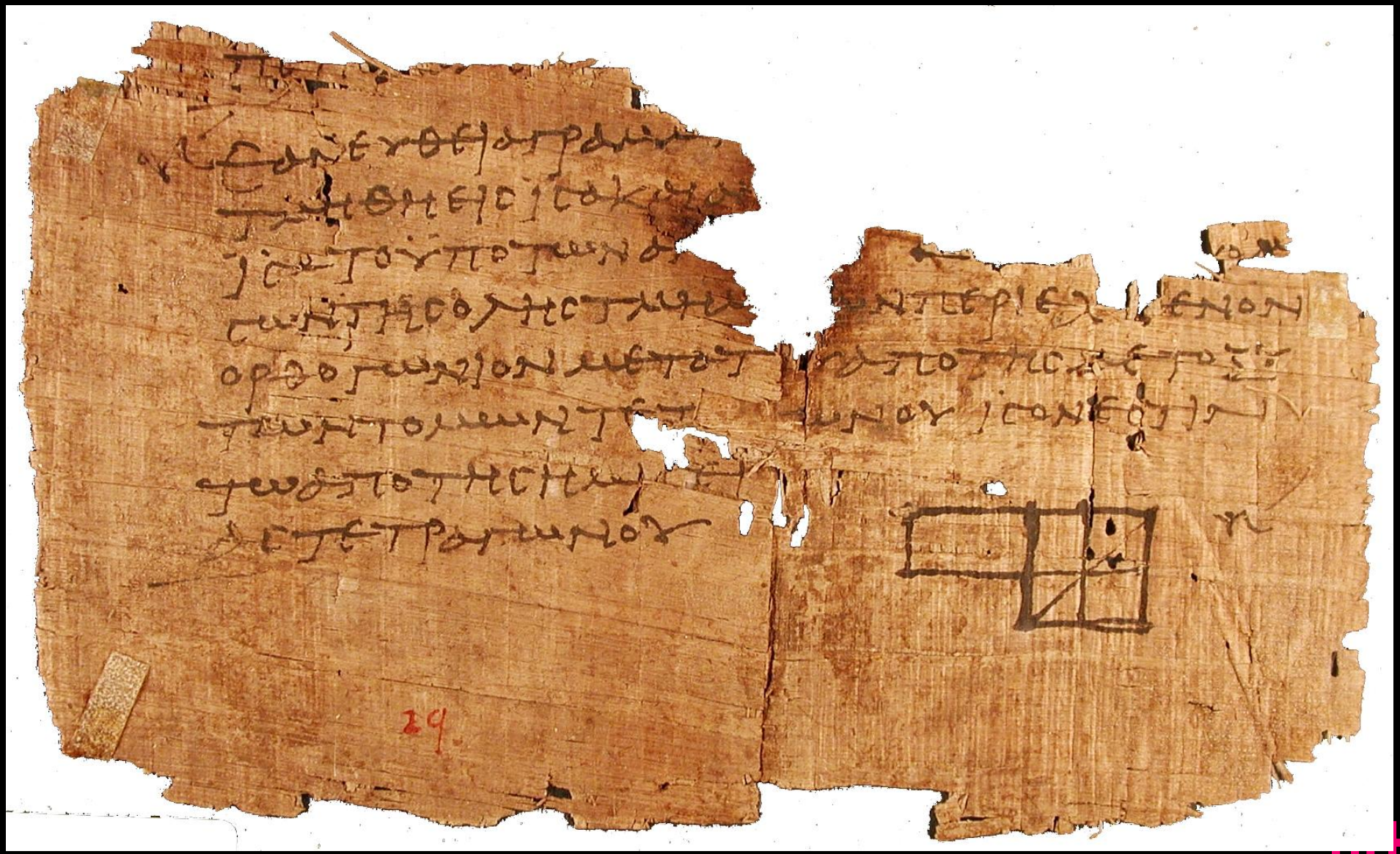
```
> mod 17 3
2
> mod (-17) 3
1
> div (-17) 3
-6
> 3 * (div (-17) 3) + (mod (-17) 3)
-17
> rem (-17) 3
-2
> quot (-17) 3
-5
> 3 * (quot (-17) 3) + (rem (-17) 3)
-17
> 3 * (quot (-17) 3) + (mod (-17) 3)
-14
> 3 * (div (-17) 3) + (rem (-17) 3)
-20
```

Théorème 18 (Division euclidienne)

Soit a un entier relatif et b un entier naturel non nul.
Il existe un unique couple d'entiers (q, r) tels que

$$a = bq + r \quad avec \quad 0 \leqslant r < b$$

Déterminer q et r, c'est effectuer la division euclidienne de a par b.
On appelle a le dividende, b le diviseur, q le quotient et r le reste.



Théorème 19

Soit a et b deux entiers et n un entier naturel.
a est congru à b modulo n si et seulement si a – b est un multiple de n

Théorème 20

$$a \equiv_n b \Longleftrightarrow \exists k \in \mathbb{Z} \mid a - b = kn$$

Théorème 19

Soit a et b deux entiers et n un entier naturel.
a est congru à b modulo n si et seulement si a – b est un multiple de n

Théorème 20

$$a \equiv_n b \Longleftrightarrow \exists k \in \mathbb{Z} \mid a = b + kn$$

Théorème 19

Soit a et b deux entiers et n un entier naturel.
a est congru à b modulo n si et seulement si a – b est un multiple de n

Théorème 20

$$a \equiv_n b \iff \exists k \in \mathbb{Z} \mid a = b + kn$$

Théorème 21

1. $\text{Card}\left(\mathbb{Z}/n\mathbb{Z}\right) = n$
2. $a \equiv_n a$
3. Si $a \equiv_n b$, alors $b \equiv_n a$
4. Si $a \equiv_n b$ et $b \equiv_n c$, alors $a \equiv_n c$
5. Si $a \equiv_n b$ et $a' \equiv_n b'$, alors

$$a + a' \equiv_n b + b' \quad a - a' \equiv_n b - b' \quad aa' \equiv_n bb'$$

6. Si $a \equiv_n b$, alors, pour tout $p \in \mathbb{N}$, $a^p \equiv_n b^p$

Théorème 21

1. $\text{Card}\,(\mathbb{Z}/n\mathbb{Z}) = n$
2. $a \equiv_n a$
3. *Si* $a \equiv_n b$, *alors* $b \equiv_n a$
4. *Si* $a \equiv_n b$ *et* $b \equiv_n c$, *alors* $a \equiv_n c$
5. *Si* $a \equiv_n b$ *et* $a' \equiv_n b'$, *alors*

$$a + a' \equiv_n b + b' \quad a - a' \equiv_n b - b' \quad aa' \equiv_n bb'$$

6. *Si* $a \equiv_n b$, *alors, pour tout* $p \in \mathbb{N}$, $a^p \equiv_n b^p$

Théorème 21

1. $\text{Card}\,(\mathbb{Z}/n\mathbb{Z}) = n$
2. $a \equiv_n a$
3. *Si* $a \equiv_n b$, *alors* $b \equiv_n a$
4. *Si* $a \equiv_n b$ *et* $b \equiv_n c$, *alors* $a \equiv_n c$
5. *Si* $a \equiv_n b$ *et* $a' \equiv_n b'$, *alors*

$$a + a' \equiv_n b + b' \quad a - a' \equiv_n b - b' \quad aa' \equiv_n bb'$$

6. *Si* $a \equiv_n b$, *alors, pour tout* $p \in \mathbb{N}$, $a^p \equiv_n b^p$

Théorème 21

1. $\text{Card}\,(\mathbb{Z}/n\mathbb{Z}) = n$
2. $a \equiv_n a$
3. *Si* $a \equiv_n b$, *alors* $b \equiv_n a$
4. *Si* $a \equiv_n b$ *et* $b \equiv_n c$, *alors* $a \equiv_n c$
5. *Si* $a \equiv_n b$ *et* $a' \equiv_n b'$, *alors*

$$a + a' \equiv_n b + b' \quad a - a' \equiv_n b - b' \quad aa' \equiv_n bb'$$

6. *Si* $a \equiv_n b$, *alors, pour tout* $p \in \mathbb{N}$, $a^p \equiv_n b^p$

Théorème 21

1. $\text{Card}\,(\mathbb{Z}/n\mathbb{Z}) = n$
2. $a \equiv_n a$
3. *Si* $a \equiv_n b$, *alors* $b \equiv_n a$
4. *Si* $a \equiv_n b$ *et* $b \equiv_n c$, *alors* $a \equiv_n c$
5. *Si* $a \equiv_n b$ *et* $a' \equiv_n b'$, *alors*

$$a + a' \equiv_n b + b' \quad a - a' \equiv_n b - b' \quad aa' \equiv_n bb'$$

6. *Si* $a \equiv_n b$, *alors, pour tout* $p \in \mathbb{N}$, $a^p \equiv_n b^p$

Théorème 21

1. $\text{Card}\,(\mathbb{Z}/n\mathbb{Z}) = n$
2. $a \equiv_n a$
3. *Si* $a \equiv_n b$, *alors* $b \equiv_n a$
4. *Si* $a \equiv_n b$ *et* $b \equiv_n c$, *alors* $a \equiv_n c$
5. *Si* $a \equiv_n b$ *et* $a' \equiv_n b'$, *alors*

$$a + a' \equiv_n b + b' \quad a - a' \equiv_n b - b' \quad aa' \equiv_n bb'$$

6. *Si* $a \equiv_n b$, *alors, pour tout* $p \in \mathbb{N}$, $a^p \equiv_n b^p$

Théorème 21

1. $\text{Card}\left(\mathbb{Z}/_{n\mathbb{Z}}\right) = n$
2. $a \equiv_n a$
3. *Si* $a \equiv_n b$, *alors* $b \equiv_n a$
4. *Si* $a \equiv_n b$ *et* $b \equiv_n c$, *alors* $a \equiv_n c$
5. *Si* $a \equiv_n b$ *et* $a' \equiv_n b'$, *alors*

$$a + a' \equiv_n b + b' \quad a - a' \equiv_n b - b' \quad aa' \equiv_n bb'$$

6. *Si* $a \equiv_n b$, *alors, pour tout* $p \in \mathbb{N}$, $a^p \equiv_n b^p$

Attention à la division !

$12 \equiv_6 0$, mais $\frac{12}{3} \not\equiv_6 \frac{0}{3}$!

Attention à la division !

$12 \equiv_6 0$, mais $\frac{12}{3} \not\equiv_6 \frac{0}{3}$!

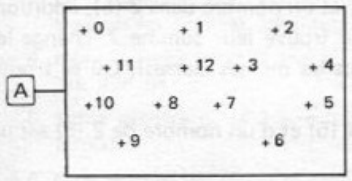
Attention à la division !

$12 \equiv_6 0$, mais $\frac{12}{3} \not\equiv_6 \frac{0}{3}$!

87 CLASSEMENT PAR 5

I Restes dans la division par 5

1. Effectue les divisions de 21 par 5 et 46 par 5.
Tu trouves le même reste : 1
Nous dirons que 46 donne dans la division par 5, le même reste que 21. A est l'ensemble des nombres de zéro à 12. Trace les traits fléchés signifiant "... donne le même reste, dans la division par 5 que ..." Recommence le schéma de façon que les flèches ne se coupent pas.



Est-ce possible ? Quel classement peux-tu effectuer dans l'ensemble A ?

2. Classons les nombres entiers.
Comme nous venons de classer les nombres entiers de zéro à 12, nous allons essayer de classer tous les nombres entiers.
Quels sont les restes possibles dans la division d'un nombre par 5 ?
Traçons 5 colonnes. Nous mettrons dans la première les nombres qui donnent pour reste 0, dans le second ceux qui donnent pour reste 1 et ainsi de suite ...

restes	0	1	2	3	4
nombres	0	1	2	3	4
	5	6	7	8	9
	10	11	12	13	14
	..	..	..	..	..

Continue à placer les nombres suivants de la même manière (par exemple jusqu'à 30).
Quels sont les nombres de la 1 ère colonne ? de la 2ème ? la 3 ème ?

Fais la différence entre deux nombres quelconques de la même colonne. Qu'observes-tu ?

3. Notation et vocabulaire.

André, Jean et Pierre font partie de la même équipe. Pour désigner cette équipe on peut aussi bien dire : équipe d'André, équipe de Jean ou bien équipe de Pierre.

245

L'ensemble des nombres de la 1ère colonne sera appelé la classe de zéro par 5 et noté $\dot{0}$ (5), ou bien classe de 5 par 5 et noté $\dot{5}$(5), ou bien classe de tout autre nombre de la colonne $\dot{10}$ (5), $\dot{15}$(5) ...) ;
Egalité : $\dot{0}$ (5), $\dot{5}$ (5) désignent le même ensemble, tu peux écrire : $\dot{0}$ (5) = $\dot{5}$ (5) ou bien, plus simplement $\dot{0} = \dot{5}$ (5).
De la même manière les nombres de la deuxième colonne constituent : $\dot{1}$ (5) ou bien $\dot{6}$ (5), ... et $\dot{1} = \dot{6} = \dot{11}$ (5).

4. Comment trouver la classe d'un nombre sans effectuer la division ?
Observe le chiffre des unités des nombres du tableau pour découvrir une règle.
Complète le tableau avec les notations les plus simples ($\dot{0}, \dot{1}, \dot{2}, \dot{3}, \dot{4}$).

Chiffres des unités	0	1	2	3	4	5	6	7	8	9
Classe par 5			$\dot{2}$				$\dot{1}$			

II Addition des classes

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 : $\dot{0} \dot{+} \dot{2} = \square$; $\dot{1} \dot{+} \dot{3} = \square$; $\dot{3} \dot{+} \dot{2} = \square$

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

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

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

Exercice 1

Écrivez par exemple les lois d'addition et de multiplication de $\mathbb{Z}/7\mathbb{Z}$ et $\mathbb{Z}/8\mathbb{Z}$: des remarques ?

Est-ce que $(\mathbb{Z}/n\mathbb{Z}, +)$ et $(\mathbb{Z}/n\mathbb{Z}, \cdot)$ sont des groupes ?

Exercice 1

Écrivez par exemple les lois d'addition et de multiplication de $\mathbb{Z}/7\mathbb{Z}$ et $\mathbb{Z}/8\mathbb{Z}$: des remarques ?

Est-ce que $(\mathbb{Z}/n\mathbb{Z}, +)$ et $(\mathbb{Z}/n\mathbb{Z}, \cdot)$ sont des groupes ?

Exercice 1

Écrivez par exemple les lois d'addition et de multiplication de $\mathbb{Z}/7\mathbb{Z}$ et $\mathbb{Z}/8\mathbb{Z}$: des remarques ?
Est-ce que $(\mathbb{Z}/n\mathbb{Z}, +)$ et $(\mathbb{Z}/n\mathbb{Z}, \cdot)$ sont des groupes ?

Par exemple, on notera :

$$\mathbb{Z}/8\mathbb{Z} = \left\{\overline{0}^{8}, \overline{1}^{8}, \overline{2}^{8}, \overline{3}^{8}, \overline{4}^{8}, \overline{5}^{8}, \overline{6}^{8}, \overline{7}^{8}\right\}$$

Attention !

Notez bien que $\overline{0}^8$, $\overline{1}^8$, etc. sont des ensembles d'entiers !...

Ainsi $\overline{0}^8 = \{\ldots, -16, -8, 0, 8, 16, 24, 32, \ldots\}$

Attention !

Notez bien que $\overline{0}^8$, $\overline{1}^8$, etc. sont des ensembles d'entiers !...

Ainsi $\overline{0}^8 = \{..., -16, -8, 0, 8, 16, 24, 32, ...\}$

Attention !

Notez bien que $\overline{0}^8$, $\overline{1}^8$, etc. sont des ensembles d'entiers !...

Ainsi $\overline{0}^8 = \{\ldots, -16, -8, 0, 8, 16, 24, 32, \ldots\}$

Théorème 22

Toute partie non vide de $\mathbb{N}$ possède un plus petit élément.

Théorème 23

Toute partie non vide MAJOREE de $\mathbb{N}$ possède un plus GRAND élément.

Théorème 22

Toute partie non vide de $\mathbb{N}$ possède un plus petit élément.

Théorème 23

Toute partie non vide MAJORÉE de $\mathbb{N}$ possède un plus GRAND élément.

Théorème 22

Toute partie non vide de $\mathbb{N}$ possède un plus petit élément.

Théorème 23

Toute partie non vide MAJORÉE de $\mathbb{N}$ possède un plus GRAND élément.

Théorème 24

Si d divise a et b alors d divise toute COMBINAISON LINÉAIRE de a et b.

Théorème 22

Toute partie non vide de $\mathbb{N}$ possède un plus petit élément.

Théorème 23

Toute partie non vide MAJORÉE de $\mathbb{N}$ possède un plus GRAND élément.

Théorème 24

Si d divise a et b alors d divise toute COMBINAISON LINÉAIRE de a et b.

Théorème 22

Toute partie non vide de $\mathbb{N}$ possède un plus petit élément.

Théorème 23

Toute partie non vide MAJORÉE de $\mathbb{N}$ possède un plus GRAND élément.

Théorème 24

Si d divise a et b alors d divise toute COMBINAISON LINÉAIRE de a et b.

Théorème 25

Si d divise a alors $|d| \leqslant |a|$.

Théorème 26

Si d divise d' et d' divise d alors $d' = d$.

Théorème 25

Si d divise a alors $|d| \leqslant |a|$.

Théorème 26

Si d divise d' et d' divise d alors $d' = d$.

Théorème 25

Si d divise a alors $|d| \leqslant |a|$.

Théorème 26

Si d divise d' et d' divise d alors $d' = d$.

Théorème 27 (suite strictement décroissante d'entiers naturels)

Toute suite strictement décroissante d'entiers naturels est constante à partir d'un certain rang.

Théorème 25

Si d divise a alors $|d| \leqslant |a|$.

Théorème 26

Si d divise d' et d' divise d alors $d' = d$.

Théorème 27 (Suite strictement décroissante d'entiers naturels)

Toute suite strictement décroissante d'entiers naturels est constante à partir d'un certain rang.

Théorème 25

Si d divise a alors $|d| \leqslant |a|$.

Théorème 26

Si d divise d' et d' divise d alors $d' = d$.

Théorème 27 (Suite strictement décroissante d'entiers naturels)

Toute suite strictement décroissante d'entiers naturels est constante à partir d'un certain rang.

PGCD

Définition 28 (PGCD)

Soit a et b deux entiers. On appelle PGCD de a et b et on note $a \wedge b$ l'entier défini de la manière suivante :

- $0 \wedge 0 = 0$
- Si a et b ne sont pas simultanément nuls, $a \wedge b$ est le plus grand entier naturel qui divise simultanément a et b.

$$D(a) \cap D(b)$$

PGCD

Définition 28 (PGCD)

Soit a et b deux entiers. On appelle PGCD de a et b et on note $a \wedge b$ l'entier défini de la manière suivante :

- $0 \wedge 0 = 0$
- Si a et b ne sont pas simultanément nuls, $a \wedge b$ est le plus grand entier naturel qui divise simultanément a et b.

$$\mathcal{D}(a) \cap \mathcal{D}(b)$$

PGCD

Définition 28 (PGCD)

Soit a et b deux entiers. On appelle PGCD de a et b et on note $a \wedge b$ l'entier défini de la manière suivante :

- $0 \wedge 0 = 0$
- Si a et b ne sont pas simultanément nuls, $a \wedge b$ est le plus grand entier naturel qui divise simultanément a et b.

$$\mathcal{D}(a) \cap \mathcal{D}(b)$$

PGCD

Définition 28 (PGCD)

Soit a et b deux entiers. On appelle PGCD de a et b et on note $a \wedge b$ l'entier défini de la manière suivante :

- $0 \wedge 0 = 0$
- Si a et b ne sont pas simultanément nuls, $a \wedge b$ est le plus grand entier naturel qui divise simultanément a et b.

$$\mathcal{D}(a) \cap \mathcal{D}(b)$$

PGCD

Définition 28 (PGCD)

Soit a et b deux entiers. On appelle PGCD de a et b et on note $a \wedge b$ l'entier défini de la manière suivante :

- $0 \wedge 0 = 0$
- Si a et b ne sont pas simultanément nuls, $a \wedge b$ est le plus grand entier naturel qui divise simultanément a et b.

$$\mathcal{D}(a) \cap \mathcal{D}(b)$$

Théorème 29 (Égalité de Bézout)

Soit a et b deux entiers relatifs. Si $d = a \wedge b$, alors il existe deux entiers u et v tels que :

$$au + bv = d$$

Théorème 30

Si $ab \neq 0$ et k un entier quelconque

$$a \wedge (b + ka) = a \wedge b$$

et en particulier

$$a \wedge b = a \wedge (b - a) = a \wedge (a - b)$$

Algorithme des différences

Théorème 31

Si $ab \neq 0$ *et* k *un entier quelconque*

$$a \wedge (b + ka) = a \wedge b$$

et en particulier

$$a \wedge b = a \wedge (b - a) = a \wedge (a - b)$$

Algorithme d'Euclide I

```
Fonction euclide( a, b : entiers ) : entier
Si b = 0 Alors
    Retourner a
Sinon
    Retourner euclide(b,rem(a,b))
FinSi
```

Algorithme d'Euclide II

k	u_k	v_k	r_k	q_k
0	1	0	$r_0 = a$	/
1	0	1	$r_1 = b$	q_1
2	$u_0 - u_1 q_1$	$v_0 - v_1 q_1$	$r_2 = r_0 - r_1 q_1$	q_2
3	$u_1 - u_2 q_2$	$v_1 - v_2 q_2$	$r_3 = r_1 - r_2 q_2$	q_3
$\vdots$			$\vdots$	$\vdots$
$p-1$			$r_{p-1} = a \wedge b$	q_{p-1}
p			$r_p = 0$	

$$u_k - (u_{k+1} \times q_{k+1}) = u_{k+2}$$

k	u_k	v_k	r_k	q_k	
0	1	0	19	/	L_0
1	0	1	15	1	L_1
2	1	-1	4	3	$L_2 \leftarrow L_0 - 1 \times L_1$
3	-3	4	3	1	$L_3 \leftarrow L_1 - 3 \times L_2$
4	4	-5	1	3	$L_4 \leftarrow L_2 - 1 \times L_3$
5			0		$L_5 \leftarrow L_3 - 3 \times L_4$

```
Fonction euc(u,v,r,u',v',r' : entiers) : liste d'entiers
Si r' = 0 Alors
    Retourner [u,v,r]
Sinon
    q ← quo(r,r')
    Retourner euc(u',v',r',u-q*u',v-q*v',r-q*r')
FinSi
```

```
Fonction EUC(a,b : entiers) : liste d'entiers
Retourner euc(1,0,a,0,1,b)
```

Nombres premiers entre eux

Définition 32

Deux entiers a et b sont premiers entre eux si et seulement si

$$a \wedge b = 1$$

Théorème 33 (Théorème de Bezout)

Les entiers a et b sont premiers entre eux si, et seulement si, il existe deux entiers u et v tels que $au + bv = 1$

$$a \wedge b = 1 \Longleftrightarrow \text{il existe } (u, v) \in \mathbb{Z}^2 \text{ tel que } au + bv = 1$$

Nombres premiers entre eux

Définition 32

Deux entiers a et b sont premiers entre eux si et seulement si

$$a \wedge b = 1$$

Théorème 33 (Théorème de Bézout)

Les entiers a et b sont premiers entre eux si, et seulement si, il existe deux entiers u et v tels que $au + bv = 1$

$$a \wedge b = 1 \iff \text{il existe } (u, v) \in \mathbb{Z}^2 \text{ tel que } au + bv = 1$$

Nombres premiers entre eux

Définition 32

Deux entiers a et b sont premiers entre eux si et seulement si

$$a \wedge b = 1$$

Théorème 33 (Théorème de Bézout)

Les entiers a et b sont premiers entre eux si, et seulement si, il existe deux entiers u et v tels que $au + bv = 1$

$$a \wedge b = 1 \iff \text{il existe } (u, v) \in \mathbb{Z}^2 \text{ tel que } au + bv = 1$$

Sommaire

Définition 34 (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.

Définition 34 (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.

Définition 34 (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.

Définition 34 (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.

Définition 34 (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.

Définition 34 (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.

Définition 34 (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.

Définition 34 (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.

Définition 35 (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 ?

Définition 35 (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 ?

Définition 35 (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 ?

Définition 35 (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 ?

Définition 35 (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 ?

```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` ?

```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` ?

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

À quelle structure algébrique peut-on associer la classe `Frational` ?

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

À quelle structure algébrique peut-on associer la classe `Frational` ?

Sommaire

```haskell
data Classe =
  Zero_c
  |Un_c
  |Classe {modulo :: Int, representant :: Int}
```

```haskell
instance Eq Classe where
  a == Zero_c = (mod (representant a) (modulo a) == 0)
  a == Un_c   = (mod (representant a) (modulo a) == 1)
  a == b      = (mod (representant a) (modulo a) == mod
    ↪  (representant b) (modulo b))
                && ((modulo a) == (modulo b))
```

```haskell
instance Show Classe where
  show Zero_c = "0%"
  show Un_c = "1%"
  show a = (show (representant a)) ++ "%" ++ (show (modulo a))
```

```haskell
(%) :: Int -> Int -> Classe
(%) a n = Classe {representant = (mod a n), modulo = n}
```

```
> 13 % 3
1%3
```

```haskell
(%) :: Int -> Int -> Classe
(%) a n = Classe {representant = (mod a n), modulo = n}
```

```
> 13 % 3
1%3
```

```haskell
-- somme induite
(%+) :: Classe -> Classe -> Classe
(%+) Zero_c a = a
(%+) a Zero_c = a
(%+) Un_c   a = Classe {representant = (representant a) + 1,
  ↪  modulo = modulo a}
(%+) a Un_c   = Classe {representant = (representant a) + 1,
  ↪  modulo = modulo a}
(%+) a b      =
  Classe{
    modulo =
       if (modulo a) == (modulo b)
       then modulo a
       else error "Classes incompatibles" ,
    representant = mod ( (representant a) + (representant b))
      ↪  (modulo a)
    }
```

```haskell
-- produit induit
(%*) :: Classe -> Classe -> Classe
(%*) Zero_c Un_c = Zero_c
(%*) Zero_c a    = 0%(modulo a)
(%*) a Zero_c    = 0%(modulo a)
(%*) Un_c a      = a
(%*) a Un_c      = a
(%*) a b         =
  Classe{
    modulo =
       if (modulo a) == (modulo b)
       then modulo a
       else error "Classes incompatibles" ,
    representant = mod ( (representant a) * (representant b))
      ↪  (modulo a)
    }
```

```haskell
instance Num Classe where
  (+)           = (%+)
  (*)           = (%*)
  negate Zero_c = Zero_c
  negate b      = Classe{representant = mod (- (representant
    ↪  b)) (modulo b), modulo = (modulo b)}
  fromInteger 0 = Zero_c
  fromInteger 1 = Un_c
  abs a         = a
  signum a      = 1
```

```
> 9%5 + 7%5
1%5
```

$\overline{x}^n$ est inversible dans $\mathbb{Z}/n\mathbb{Z}$

$$
\begin{aligned}
&\Leftrightarrow \quad \exists \overline{y}^n \in \mathbb{Z}/n\mathbb{Z} \quad | \quad \overline{x}^n \cdot \overline{y}^n = \overline{1}^n \\
&\Leftrightarrow \quad \exists y \in \{0, 1, ..., n-1\} \quad | \quad x \cdot y \equiv 1 (\mathrm{mod}\, n) \\
&\Leftrightarrow \quad \exists (y, k) \in \{0, 1, ..., n-1\} \times \mathbb{Z} \quad | \quad xy = 1 + \\
&\Leftrightarrow \quad \exists (y, k) \in \{0, 1, ..., n-1\} \times \mathbb{Z} \quad | \quad xy - kn = \\
&\Leftrightarrow \quad x \wedge n = 1
\end{aligned}
$$

Théorème 36

$\left((\mathbb{Z}/n\mathbb{Z})^*, \cdot\right)$ *est un groupe.*

Théorème 37

- *Les seuls éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$ sont les éléments premiers avec n.*
- *Si n est premier, $\mathbb{Z}/n\mathbb{Z}$ est un corps et on le note dans ce cas $\mathbb{F}_n$.*

Théorème 37

- *Les seuls éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$ sont les éléments premiers avec n.*
- *Si n est premier, $\mathbb{Z}/n\mathbb{Z}$ est un corps et on le note dans ce cas $\mathbb{F}_n$.*

Théorème 37

- *Les seuls éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$ sont les éléments premiers avec n.*
- *Si n est premier, $\mathbb{Z}/n\mathbb{Z}$ est un corps et on le note dans ce cas $\mathbb{F}_n$.*

Sommaire


VA A ROME PORTER UN MESSAGE À CÉSAR. TU LUI DIRAS : " TOUTE LA GAULE EST OCCUPÉE ", IL TE DEMANDERA : " TOUTE ? ", TU LUI RÉPONDRAS : " TOUTE ! " IL COMPRENDRA.
Uderzo & Goscinny

Comme le disait Suétone (70-127) dans La vie des 12 Césars :

> *Extant et ad Ciceronem, item ad familiares domesticis de rebus, in quibus, si qua occultius perferenda erant, per notas scripsit, id est sic structo litterarum ordine, ut nullum uerbum effici posset : quae si qui inuestigare et persequi uelit, quartam elementorum litteram, id est D pro A et perinde reliquas commutet.*

Ce que César aurait peut-être écrit sous cette forme :

> *Rkgnag rg nq Pvprebarz, vgrz nq snzvyvnerf qbzrfgvpvf qr erohf, va dhvohf, fv dhn bpphygvhf cresreraqn renag, cre abgnf fpevcfvg, vq rfg fvp fgehpgb yvggrenehz beqvar, hg ahyyhz hreohz rssvpv cbffrg : dhnr fv dhv vahrfgvtner rg crefrdhv hryvg, dhnegnz ryrzragbehz yvggrenz, vq rfg Q ceb N rg crevaqr eryvdhnf pbzzhgrg.*

Ou sous celle-ci si Rome avait été colonisé par des informaticiens américains :

"H{wdqw#hw#dg#Flfhurqhp/#lwhp#dg#idplolduhv#grphvwlflv#gh#uhexv/#lq#txlexv/#vl#txd#rffxowlxv#shuihuhqgd#hudqw/#shu#qrwdv#vfulsvlw/#lg#hvw#vlf#vwuxfwr#olwwhuduxp#ruglqh/#xw#qxooxp#xhuexp#hiilfl#srvvhw=#txdh#vl#txl#lqxhvwljduh#hw#shuvhtxl#xholw/#txduwdp#hohphqwruxp#olwwhudp/#lg#hvw#G#sur#D#hw#shulqgh#uholtxdv#frppxwhw1"

Ceux qui ne font pas de latin de spécialité seront sûrement plus à l'aise avec cette nouvelle transcription du même texte :

> *On possède enfin de César des lettres à Cicéron, et sa correspondance avec ses amis sur ses affaires domestiques. Il y employait, pour les choses tout à fait secrètes, une espèce de chiffre qui en rendait le sens inintelligible (les lettres étant disposées de manière à ne pouvoir jamais former un mot), et qui consistait, je le dis pour ceux qui voudront les déchiffrer, à changer le rang des lettres dans l'alphabet, en écrivant la quatrième pour la première, c'est-à-dire le D pour l'A, et ainsi de suite.*

Vous êtes César

Vous voulez transmettre cet important message :

Les sanglots longs des violons de l'automne

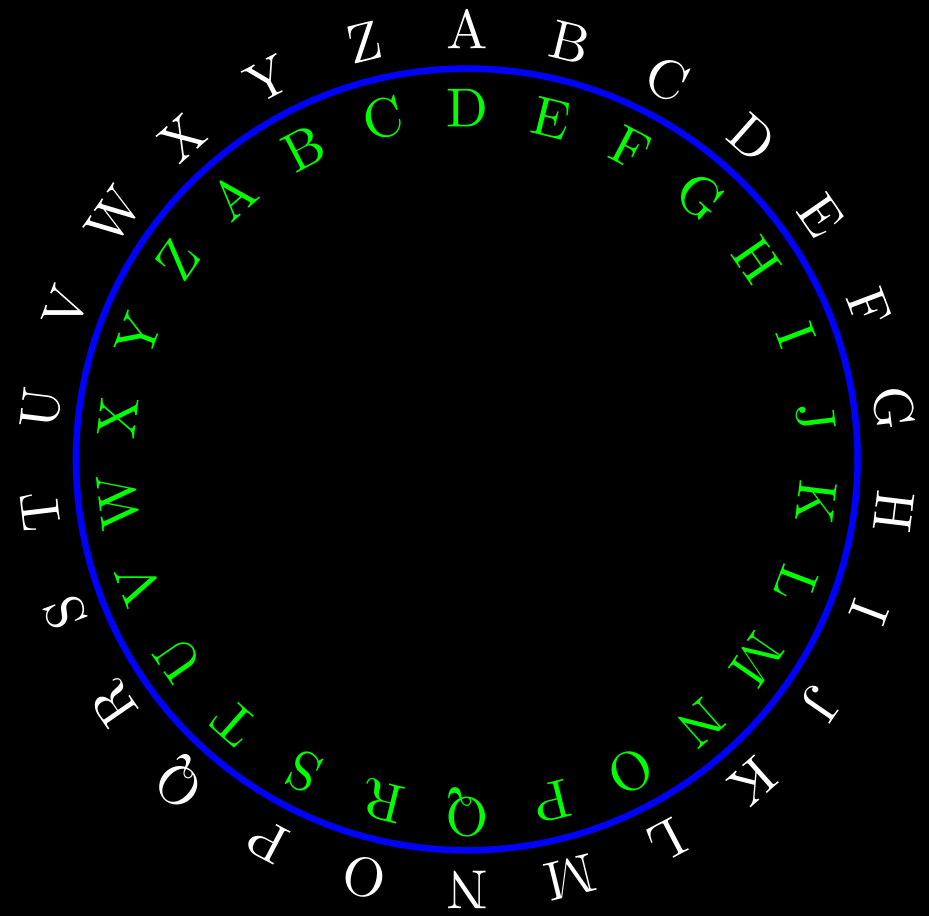
Vous êtes Vercingétorix

Vous voulez traduire ce message intercepté par vos espions :

*eohvvhqw#prq#frhxu#g*xqh#odqjxhxu#prqrwrqh*

La règle ou la roue ?

Que vous inspire ce dessin :



```
> map ord ['A'..'H']
[65,66,67,68,69,70,71,72]
```

```
> map chr [65..90]
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

```
> map ord ['A'..'H']
[65,66,67,68,69,70,71,72]
```

```
> map chr [65..90]
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

Caractère		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/	0	1	2
Code	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
Caractère	3	4	5	6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E
Code	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69
Caractère	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Code	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88
Caractère	Y	Z	[	\	]	^	_	‘	a	b	c	d	e	f	g	h	i	j	k
Code	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107
Caractère	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~
Code	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126

Table des caractères ASCII affichables

```
> cesar 3 "Les sanglots longs des violons de l'automne"
"Ohv#vdqjorwv#orqjv#ghv#ylrorqv#gh#o*dxwrpqh"
```

et pour déchiffrer

« *eohvvhqw#prq#frhxu#g*xqh#odqjxhxu#prqrwrqh* » ?

```
> cesar (-3) "eohvvhqw#prq#frhxu#g*xqh#odqjxhxu#prqrwrqh"
"blessent mon coeur d'une langueur monotone"
```

```
> cesar 3 "Les sanglots longs des violons de l'automne"
"Ohv#vdqjorwv#orqjv#ghv#ylrorqv#gh#o*dxwrpqh"
```

et pour déchiffrer

« *eohvvhqw#prq#frhxu#g*xqh#odqjxhxu#prqrwrqh* » ?

```
> cesar (-3) "eohvvhqw#prq#frhxu#g*xqh#odqjxhxu#prqrwrqh"
"blessent mon coeur d'une langueur monotone"
```

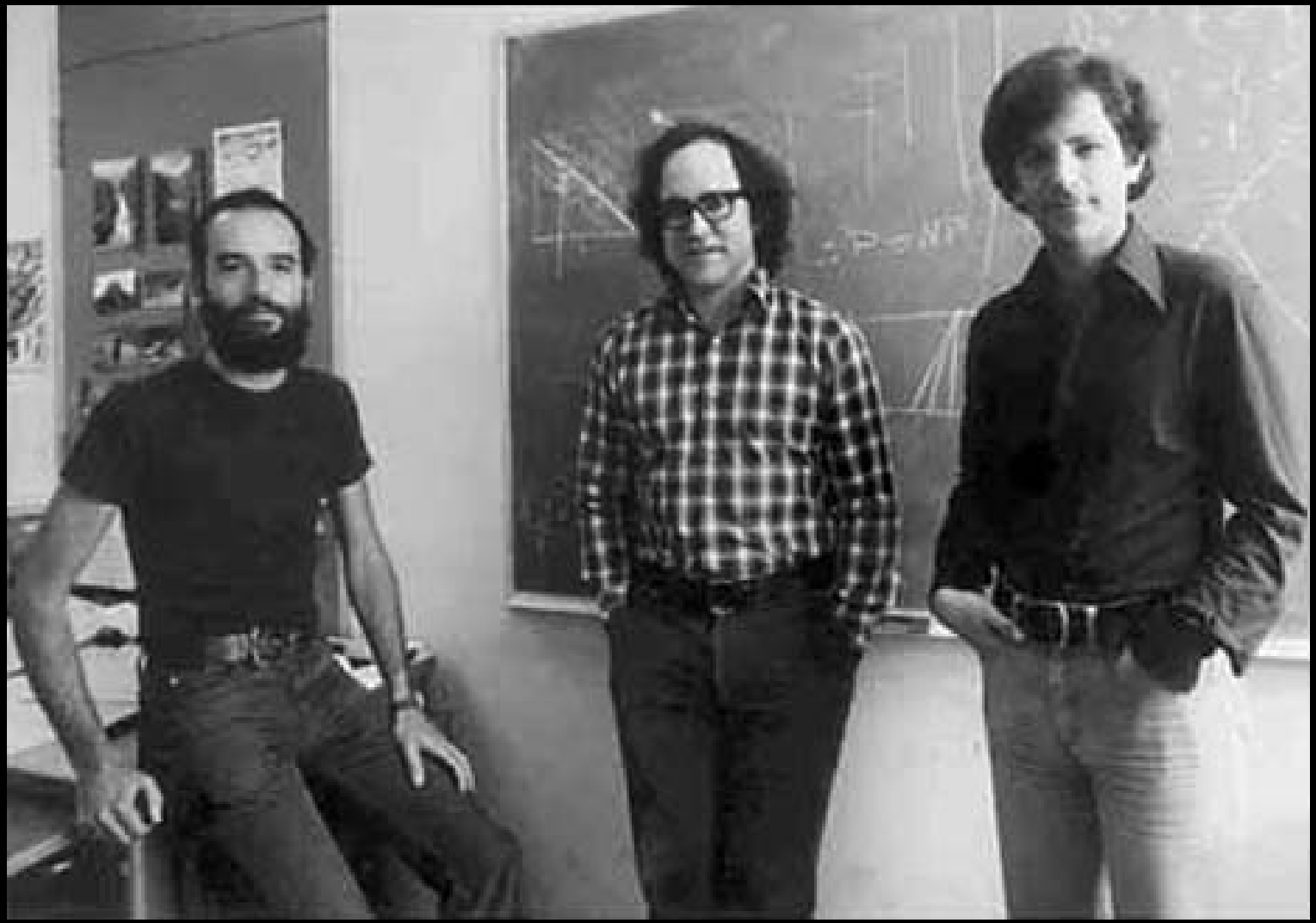
```
> cesar 3 "Les sanglots longs des violons de l'automne"
"Ohv#vdqjorwv#orqjv#ghv#ylrorqv#gh#o*dxwrpqh"
```

et pour déchiffrer

« *eohvvhqw#prq#frhxu#g*xqh#odqjxhxu#prqrwrqh* » ?

```
> cesar (-3) "eohvvhqw#prq#frhxu#g*xqh#odqjxhxu#prqrwrqh"
"blessent mon coeur d'une langueur monotone"
```


rSa
Revenu de
solidarité
active



- Votre amie Josette veut recevoir de vous une lettre d'amour mais elle a peur que le facteur l'intercepte et la lise.
- Elle fabrique donc dans son petit atelier une clé, un cadenas et une boîte vide .
- Elle vous envoie le cadenas, ouvert mais sans la clé, par la poste, donc à la merci du facteur : le cadenas est appelé *clé publique*.
- Vous recevez le cadenas et l'utilisez pour fermer une boîte contenant votre lettre : le facteur ne peut pas l'ouvrir car il n'a pas la clé.
- Josette reçoit donc la boîte fermée : elle utilise sa clé, qu'elle seule possède, pour ouvrir la boîte et se pâmer devant vos élans épistolaires.

- Votre amie Josette veut recevoir de vous une lettre d'amour mais elle a peur que le facteur l'intercepte et la lise.
- Elle fabrique donc dans son petit atelier une clé, un cadenas et une boîte vide .
- Elle vous envoie le cadenas, ouvert mais sans la clé, par la poste, donc à la merci du facteur : le cadenas est appelé *clé publique*.
- Vous recevez le cadenas et l'utilisez pour fermer une boîte contenant votre lettre : le facteur ne peut pas l'ouvrir car il n'a pas la clé.
- Josette reçoit donc la boîte fermée : elle utilise sa clé, qu'elle seule possède, pour ouvrir la boîte et se pâmer devant vos élans épistolaires.

- Votre amie Josette veut recevoir de vous une lettre d'amour mais elle a peur que le facteur l'intercepte et la lise.
- Elle fabrique donc dans son petit atelier une clé, un cadenas et une boîte vide .
- Elle vous envoie le cadenas, ouvert mais sans la clé, par la poste, donc à la merci du facteur : le cadenas est appelé *clé publique*.
- Vous recevez le cadenas et l'utilisez pour fermer une boîte contenant votre lettre : le facteur ne peut pas l'ouvrir car il n'a pas la clé.
- Josette reçoit donc la boîte fermée : elle utilise sa clé, qu'elle seule possède, pour ouvrir la boîte et se pâmer devant vos élans épistolaires.

- Votre amie Josette veut recevoir de vous une lettre d'amour mais elle a peur que le facteur l'intercepte et la lise.
- Elle fabrique donc dans son petit atelier une clé, un cadenas et une boîte vide .
- Elle vous envoie le cadenas, ouvert mais sans la clé, par la poste, donc à la merci du facteur : le cadenas est appelé *clé publique*.
- Vous recevez le cadenas et l'utilisez pour fermer une boîte contenant votre lettre : le facteur ne peut pas l'ouvrir car il n'a pas la clé.
- Josette reçoit donc la boîte fermée : elle utilise sa clé, qu'elle seule possède, pour ouvrir la boîte et se pâmer devant vos élans épistolaires.

- Votre amie Josette veut recevoir de vous une lettre d'amour mais elle a peur que le facteur l'intercepte et la lise.
- Elle fabrique donc dans son petit atelier une clé, un cadenas et une boîte vide .
- Elle vous envoie le cadenas, ouvert mais sans la clé, par la poste, donc à la merci du facteur : le cadenas est appelé *clé publique*.
- Vous recevez le cadenas et l'utilisez pour fermer une boîte contenant votre lettre : le facteur ne peut pas l'ouvrir car il n'a pas la clé.
- Josette reçoit donc la boîte fermée : elle utilise sa clé, qu'elle seule possède, pour ouvrir la boîte et se pâmer devant vos élans épistolaires.

Le principe se ramène à :

- je fabrique une fonction π définie sur $\mathbb{N}$ qui possède une réciproque σ.
- On suppose qu'on peut fabriquer de telles fonctions mais que si l'on ne connaît que π, il est (quasiment) impossible de retrouver σ.
- La fonction π est donc la clé publique : vous envoyez $\pi(\text{message})$ à Josette.
- Celle-ci calcule $\sigma\big(\pi(\text{message})\big) = \text{message}$.
- Josette est la seule à pouvoir décoder car elle seule connaît σ.

Le principe se ramène à :

- je fabrique une fonction π définie sur $\mathbb{N}$ qui possède une réciproque σ.
- On suppose qu'on peut fabriquer de telles fonctions mais que si l'on ne connaît que π, il est (quasiment) impossible de retrouver σ.
- La fonction π est donc la clé publique : vous envoyez $\pi(\text{message})$ à Josette.
- Celle-ci calcule $\sigma\big(\pi(\text{message})\big) = \text{message}$.
- Josette est la seule à pouvoir décoder car elle seule connaît σ.

Le principe se ramène à :

- je fabrique une fonction π définie sur $\mathbb{N}$ qui possède une réciproque σ.
- On suppose qu'on peut fabriquer de telles fonctions mais que si l'on ne connaît que π, il est (quasiment) impossible de retrouver σ.
- La fonction π est donc la clé publique : vous envoyez $\pi(\text{message})$ à Josette.
- Celle-ci calcule $\sigma\big(\pi(\text{message})\big) = \text{message}$.
- Josette est la seule à pouvoir décoder car elle seule connaît σ.

Le principe se ramène à :

- je fabrique une fonction π définie sur $\mathbb{N}$ qui possède une réciproque σ.
- On suppose qu'on peut fabriquer de telles fonctions mais que si l'on ne connaît que π, il est (quasiment) impossible de retrouver σ.
- La fonction π est donc la clé publique : vous envoyez $\pi(\text{message})$ à Josette.
- Celle-ci calcule $\sigma\big(\pi(\text{message})\big) = \text{message}$.
- Josette est la seule à pouvoir décoder car elle seule connaît σ.

Le principe se ramène à :

- je fabrique une fonction π définie sur $\mathbb{N}$ qui possède une réciproque σ.
- On suppose qu'on peut fabriquer de telles fonctions mais que si l'on ne connaît que π, il est (quasiment) impossible de retrouver σ.
- La fonction π est donc la clé publique : vous envoyez $\pi(\text{message})$ à Josette.
- Celle-ci calcule $\sigma\big(\pi(\text{message})\big) = \text{message}$.
- Josette est la seule à pouvoir décoder car elle seule connaît σ.

Comment faire cela informatiquement ?...

- On commence par fabriquer deux nombres premiers quelconques mais suffisamment grands, par exemple de l'ordre de 1000 bits (300 chiffres décimaux).
- Ces deux nombres p et q ne sont connus que d'une personne : c'est la *clé privée*.
- Josette forme le produit de ces deux nombres : $n = p \times q$.
- Ce nombre n est le module publique de Josette.
- Il reste à former le nombre e qui est la puissance de cryptage publique.
- Josette commence par former le nombre $\varphi_n = (p-1)(q-1)$.
- Josette doit choisir un nombre e premier avec φ_n. En pratique, elle choisit un grand nombre premier au hasard.
- Il reste à Josette à calculer sa puissance de décryptage secrète : elle l'obtient en cherchant l'inverse d de e modulo φ_n.

- On commence par fabriquer deux nombres premiers quelconques mais suffisamment grands, par exemple de l'ordre de 1000 bits (300 chiffres décimaux).
- Ces deux nombres p et q ne sont connus que d'une personne : c'est la *clé privée*.
- Josette forme le produit de ces deux nombres : $n = p \times q$.
- Ce nombre n est le module publique de Josette.
- Il reste à former le nombre e qui est la puissance de cryptage publique.
- Josette commence par former le nombre $\varphi_n = (p-1)(q-1)$.
- Josette doit choisir un nombre e premier avec φ_n. En pratique, elle choisit un grand nombre premier au hasard.
- Il reste à Josette à calculer sa puissance de décryptage secrète : elle l'obtient en cherchant l'inverse d de e modulo φ_n.

- On commence par fabriquer deux nombres premiers quelconques mais suffisamment grands, par exemple de l'ordre de 1000 bits (300 chiffres décimaux).
- Ces deux nombres p et q ne sont connus que d'une personne : c'est la *clé privée*.
- Josette forme le produit de ces deux nombres : $n = p \times q$.
- Ce nombre n est le module publique de Josette.
- Il reste à former le nombre e qui est la puissance de cryptage publique.
- Josette commence par former le nombre $\varphi_n = (p-1)(q-1)$.
- Josette doit choisir un nombre e premier avec φ_n. En pratique, elle choisit un grand nombre premier au hasard.
- Il reste à Josette à calculer sa puissance de décryptage secrète : elle l'obtient en cherchant l'inverse d de e modulo φ_n.

- On commence par fabriquer deux nombres premiers quelconques mais suffisamment grands, par exemple de l'ordre de 1000 bits (300 chiffres décimaux).
- Ces deux nombres p et q ne sont connus que d'une personne : c'est la *clé privée*.
- Josette forme le produit de ces deux nombres : $n = p \times q$.
- Ce nombre n est le module publique de Josette.
- Il reste à former le nombre e qui est la puissance de cryptage publique.
- Josette commence par former le nombre $\varphi_n = (p-1)(q-1)$.
- Josette doit choisir un nombre e premier avec φ_n. En pratique, elle choisit un grand nombre premier au hasard.
- Il reste à Josette à calculer sa puissance de décryptage secrète : elle l'obtient en cherchant l'inverse d de e modulo φ_n.

- On commence par fabriquer deux nombres premiers quelconques mais suffisamment grands, par exemple de l'ordre de 1000 bits (300 chiffres décimaux).
- Ces deux nombres p et q ne sont connus que d'une personne : c'est la *clé privée*.
- Josette forme le produit de ces deux nombres : $n = p \times q$.
- Ce nombre n est le module publique de Josette.
- Il reste à former le nombre e qui est la puissance de cryptage publique.
- Josette commence par former le nombre $\varphi_n = (p-1)(q-1)$.
- Josette doit choisir un nombre e premier avec φ_n. En pratique, elle choisit un grand nombre premier au hasard.
- Il reste à Josette à calculer sa puissance de décryptage secrète : elle l'obtient en cherchant l'inverse d de e modulo φ_n.

- On commence par fabriquer deux nombres premiers quelconques mais suffisamment grands, par exemple de l'ordre de 1000 bits (300 chiffres décimaux).
- Ces deux nombres p et q ne sont connus que d'une personne : c'est la *clé privée*.
- Josette forme le produit de ces deux nombres : $n = p \times q$.
- Ce nombre n est le module publique de Josette.
- Il reste à former le nombre e qui est la puissance de cryptage publique.
- Josette commence par former le nombre $\varphi_n = (p-1)(q-1)$.
- Josette doit choisir un nombre e premier avec φ_n. En pratique, elle choisit un grand nombre premier au hasard.
- Il reste à Josette à calculer sa puissance de décryptage secrète : elle l'obtient en cherchant l'inverse d de e modulo φ_n.

- On commence par fabriquer deux nombres premiers quelconques mais suffisamment grands, par exemple de l'ordre de 1000 bits (300 chiffres décimaux).
- Ces deux nombres p et q ne sont connus que d'une personne : c'est la *clé privée*.
- Josette forme le produit de ces deux nombres : $n = p \times q$.
- Ce nombre n est le module publique de Josette.
- Il reste à former le nombre e qui est la puissance de cryptage publique.
- Josette commence par former le nombre $\varphi_n = (p-1)(q-1)$.
- Josette doit choisir un nombre e premier avec φ_n. En pratique, elle choisit un grand nombre premier au hasard.
- Il reste à Josette à calculer sa puissance de décryptage secrète : elle l'obtient en cherchant l'inverse d de e modulo φ_n.

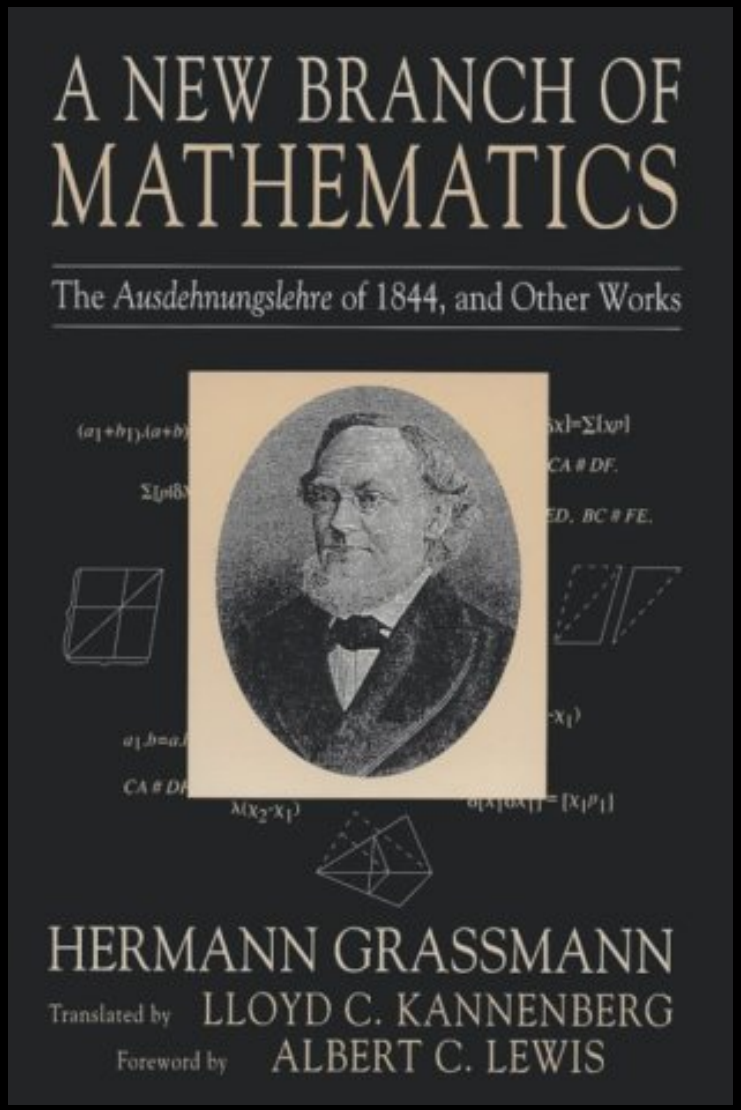
- On commence par fabriquer deux nombres premiers quelconques mais suffisamment grands, par exemple de l'ordre de 1000 bits (300 chiffres décimaux).
- Ces deux nombres p et q ne sont connus que d'une personne : c'est la *clé privée*.
- Josette forme le produit de ces deux nombres : $n = p \times q$.
- Ce nombre n est le module publique de Josette.
- Il reste à former le nombre e qui est la puissance de cryptage publique.
- Josette commence par former le nombre $\varphi_n = (p-1)(q-1)$.
- Josette doit choisir un nombre e premier avec φ_n. En pratique, elle choisit un grand nombre premier au hasard.
- Il reste à Josette à calculer sa puissance de décryptage secrète : elle l'obtient en cherchant l'inverse d de e modulo φ_n.

- Vous avez donc reçu (e, n), la clé publique de Josette. Vous découpez votre message en petits morceaux à l'aide d'un chiffrement par blocs.
- Vous calculez chaque bloc à la puissance e modulo n.
- Josette reçoit ce message et le décrypte grâce à sa clé secrète d qu'elle est la seule à connaître dans tout l'univers...

- Vous avez donc reçu (e, n), la clé publique de Josette. Vous découpez votre message en petits morceaux à l'aide d'un chiffrement par blocs.
- Vous calculez chaque bloc à la puissance e modulo n.
- Josette reçoit ce message et le décrypte grâce à sa clé secrète d qu'elle est la seule à connaître dans tout l'univers...

- Vous avez donc reçu (e, n), la clé publique de Josette. Vous découpez votre message en petits morceaux à l'aide d'un chiffrement par blocs.
- Vous calculez chaque bloc à la puissance e modulo n.
- Josette reçoit ce message et le décrypte grâce à sa clé secrète d qu'elle est la seule à connaître dans tout l'univers...

Sommaire



Hermann Graßmann (1809-1877)

Stefan Banach (1892-1945)


0
100
200
300
400
0
50
100
150
200
250
300


0
100
200
300
400
0
50
100
150
200
250
300


0
100
200
300
400
0
50
100
150
200
250
300

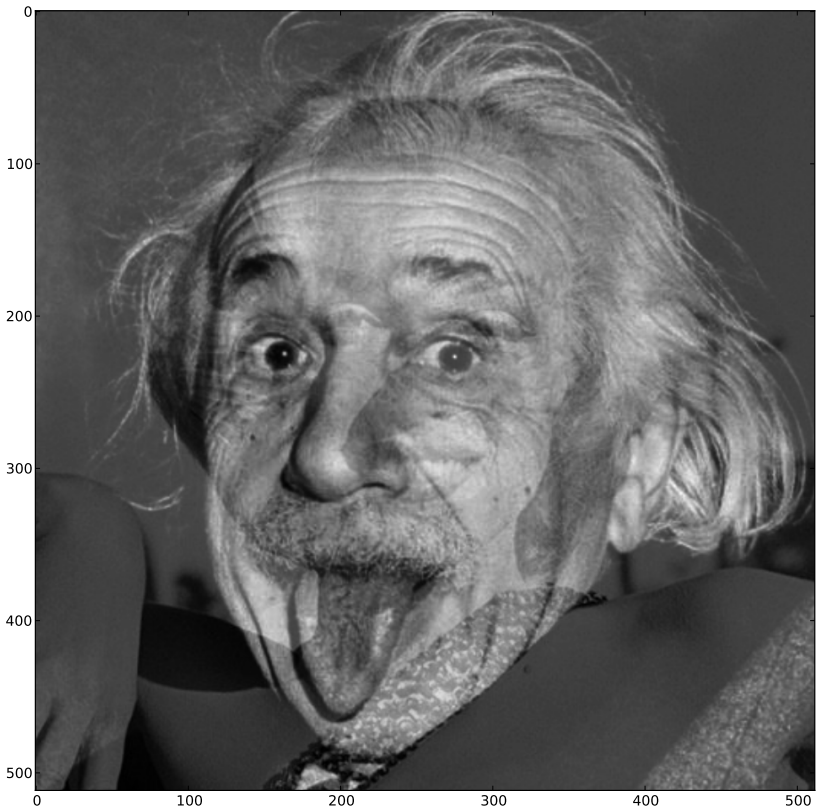

0
100
200
300
400
0
50
100
150
200
250
300

$3 \times$ Marylin $+$ Albert

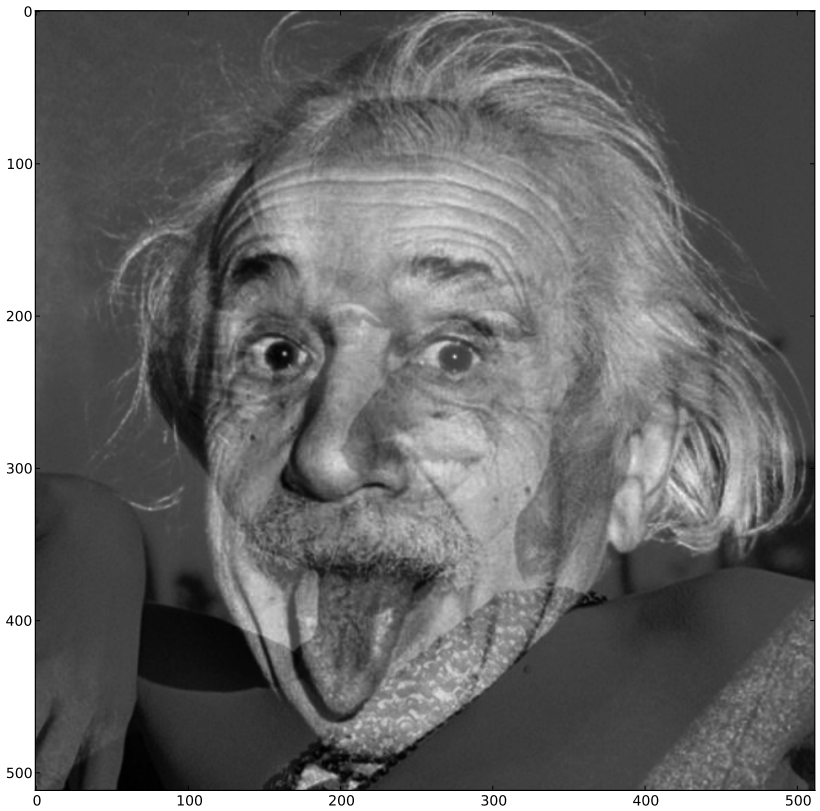
$3 \times$ Marylin + Albert

Marylin + Albert

Marylin + Albert



Marylin + 3 × Albert



Marylin + 3 × Albert

Un complexe ? La donnée de deux scalaires...

```
module ALcomplexes where

    type Scalaire = Int
    type Re = Scalaire
    type Im = Scalaire

    data Complexe = Re ::: Im
```

```
> let z = 3:::2
> z
3 + 2i
```

Un complexe ? La donnée de deux scalaires...

```haskell
module ALcomplexes where

    type Scalaire = Int
    type Re = Scalaire
    type Im = Scalaire

    data Complexe = Re ::: Im
```

```
> let z = 3:::2
> z
3 + 2i
```

Un complexe ? La donnée de deux scalaires...

```
module ALcomplexes where

    type Scalaire = Int
    type Re = Scalaire
    type Im = Scalaire

    data Complexe = Re ::: Im
```

```
> let z = 3:::2
> z
3 + 2i
```

```haskell
instance Show Complexe where
      show (r:::i) = (show r) ++ " + " ++ (show i) ++ "i"
```

Somme ?

```
(@+) :: Complexe -> Complexe  -> Complexe
(@+)    (r:::i)      (r':::i') =  (r + r'):::(i + i')
```

```
> z
3 + 2i
> z @+ z
6 + 4i
```

Somme ?

```
(@+) :: Complexe -> Complexe  -> Complexe
(@+)    (r:::i)      (r':::i') =  (r + r'):::(i + i')
```

```
> z
3 + 2i
> z @+ z
6 + 4i
```

Somme ?

```
(@+) :: Complexe -> Complexe  -> Complexe
(@+)    (r:::i)      (r':::i') =  (r + r'):::(i + i')
```

```
> z
3 + 2i
> z @+ z
6 + 4i
```

$3 * z$? Types ?

```
(@.) :: Scalaire -> Complexe  -> Complexe
(@.)    k           (r:::i)   =  (k*r):::(k*i)
```

```
> z
3 + 2i
> 3 @. z
9 + 6i
> z @+ ((-2) @. z)
-3 + -2i
```

$3 * z$? Types ?

```
(@.) :: Scalaire -> Complexe  -> Complexe
(@.)    k           (r:::i)   =  (k*r):::(k*i)
```

```
> z
3 + 2i
> 3 @. z
9 + 6i
> z @+ ((-2) @. z)
-3 + -2i
```

$3 * z$? Types ?

```
(@.) :: Scalaire -> Complexe  -> Complexe
(@.)    k           (r:::i)   =  (k*r):::(k*i)
```

```
> z
3 + 2i
> 3 @. z
9 + 6i
> z @+ ((-2) @. z)
-3 + -2i
```

Définition 38 ($\mathbb{K}$-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 $\mathbb{K}$:

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

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

Python

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

Définition 38 ($\mathbb{K}$-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 $\mathbb{K}$:

$(\lambda \oplus \mu) \cdot \boldsymbol{u} = \lambda \cdot \boldsymbol{u} \dagger \mu \cdot \boldsymbol{u}$ (distributivité vectorielle) ;

$\lambda \cdot (\boldsymbol{u} \dagger \boldsymbol{v}) = \lambda \cdot \boldsymbol{u} \dagger \lambda \cdot \boldsymbol{v}$ (distributivité scalaire) ;

$(\lambda \odot \mu) \cdot \boldsymbol{u} = \lambda \cdot (\mu \cdot \boldsymbol{u})$ (associativité) ;

$1_\odot \cdot \boldsymbol{u} = \boldsymbol{u}$ (axiome d'identité)

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

Python

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

Matrice ?

```
type Indice = Int
type Taille = (Indice,Indice)
type Coeffs = Int
```

```
data  Matrice  =
       Mat (Taille, (Indice,Indice) -> Coeffs)
```

```
Mat((5,3), \ (i,j) -> 3*i + j)
```

??

Matrice ?

```
type Indice = Int
type Taille = (Indice,Indice)
type Coeffs = Int
```

```
data  Matrice  =
        Mat (Taille, (Indice,Indice) -> Coeffs)
```

```
Mat((5,3), \ (i,j) -> 3*i + j)
```

??

Matrice ?

```
type Indice = Int
type Taille = (Indice,Indice)
type Coeffs = Int
```

```
data  Matrice  =
       Mat (Taille, (Indice,Indice) -> Coeffs)
```

```
Mat((5,3), \ (i,j) -> 3*i + j)
```

??

Matrice ?

```
type Indice = Int
type Taille = (Indice,Indice)
type Coeffs = Int
```

```
data  Matrice  =
       Mat (Taille, (Indice,Indice) -> Coeffs)
```

```
Mat((5,3), \ (i,j) -> 3*i + j)
```

??

Matrice ?

```haskell
type Indice = Int
type Taille = (Indice,Indice)
type Coeffs = Int
```

```haskell
data  Matrice  =
        Mat (Taille, (Indice,Indice) -> Coeffs)
```

```haskell
Mat((5,3), \ (i,j) -> 3*i + j)
```

??

```
> let ma = Mat((5,3), \ (i,j) -> 3*i + j)
> ma
 0   1   2
 3   4   5
 6   7   8
 9  10  11
12  13  14
```

Somme

Comment additionner deux matrices ?

```
(@+) :: Matrice          -> Matrice          -> Matrice
(@+)    (Mat (t,f))    (Mat (t',f'))
          | t == t'    = Mat(t, \ (i,j) -> f(i,j) + f'(i,j))
          | otherwise = error "Somme : tailles incompatibles"
```

Somme

Comment additionner deux matrices ?

```
(@+) :: Matrice          -> Matrice              -> Matrice
(@+)     (Mat (t,f))    (Mat (t',f'))
          | t == t'    = Mat(t, \ (i,j) -> f(i,j) + f'(i,j))
          | otherwise = error "Somme : tailles incompatibles"
```

Somme

```
> ma
 0   1   2
 3   4   5
 6   7   8
 9  10  11
12  13  14

> ma @+ ma
 0   2   4
 6   8  10
12  14  16
18  20  22
24  26  28
```

Produit par un scalaire

Multiplier une matrice par un scalaire ?

```
(@.) :: Coeffs -> Matrice    -> Matrice
(@.)    k          (Mat(t,f)) = Mat(t,\ (i,j) -> k * f(i,j))
```

Produit par un scalaire

Multiplier une matrice par un scalaire ?

```
(@.) :: Coeffs -> Matrice   -> Matrice
(@.)    k          (Mat(t,f)) = Mat(t,\ (i,j) -> k * f(i,j))
```

Produit par un scalaire

```
> ma
 0   1   2
 3   4   5
 6   7   8
 9  10  11
12  13  14

> (-4) @. ma
  0   -4   -8
-12  -16  -20
-24  -28  -32
-36  -40  -44
-48  -52  -56
```

Définition 39 (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 :

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

L'ensemble de ces combinaisons linéaires est noté :

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


0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140

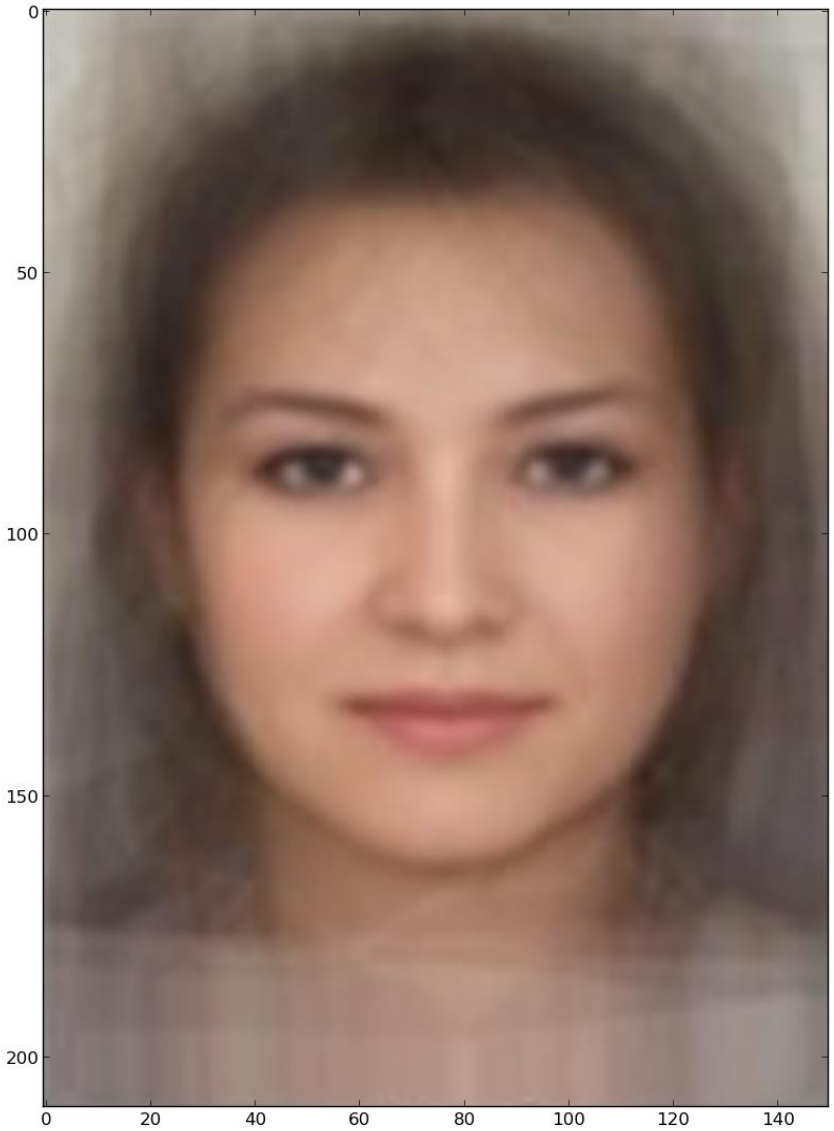

0
50
100
150
200
0
20
40
60
80
100
120
140

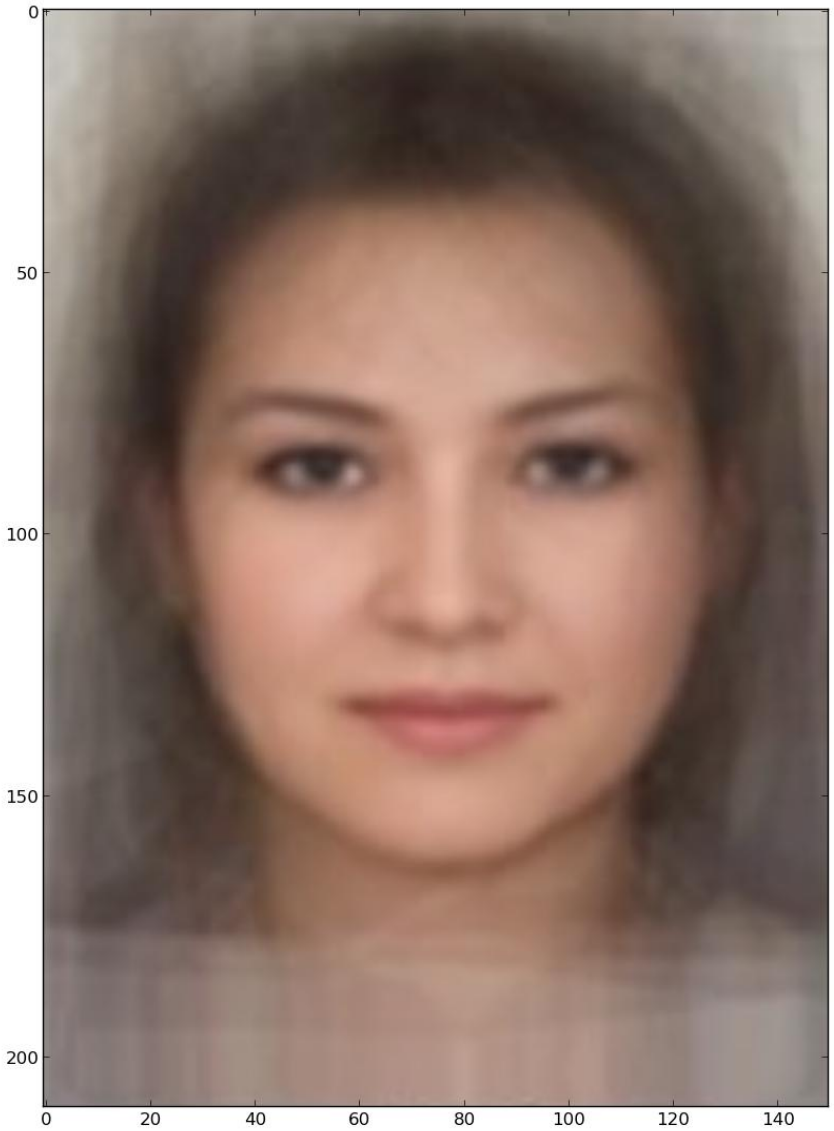

0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140

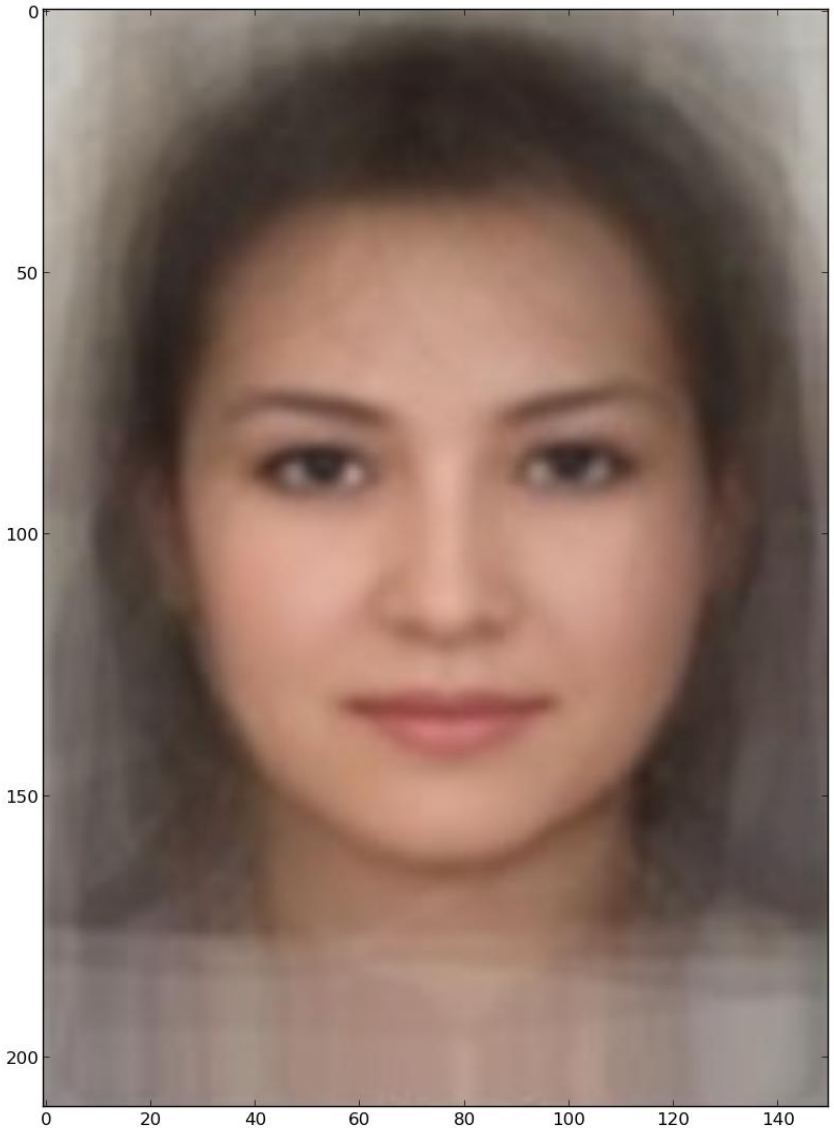

0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140

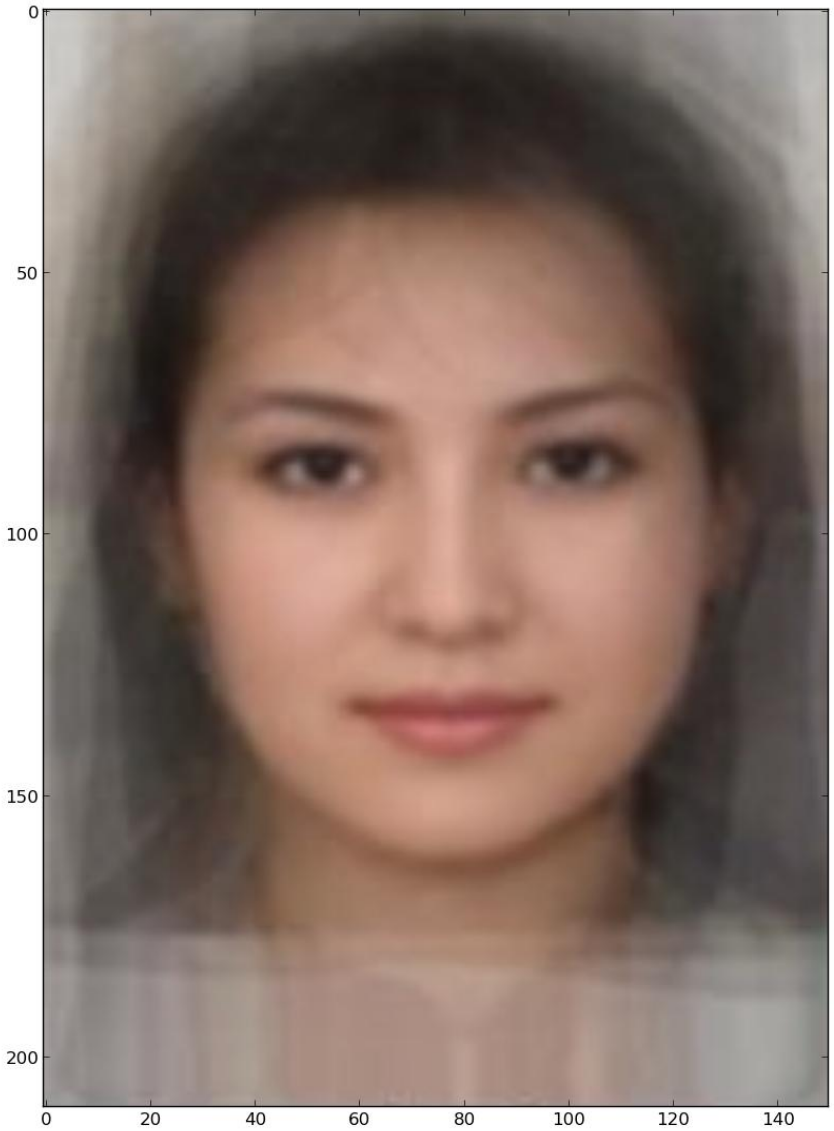

0
50
100
150
200
0
20
40
60
80
100
120
140

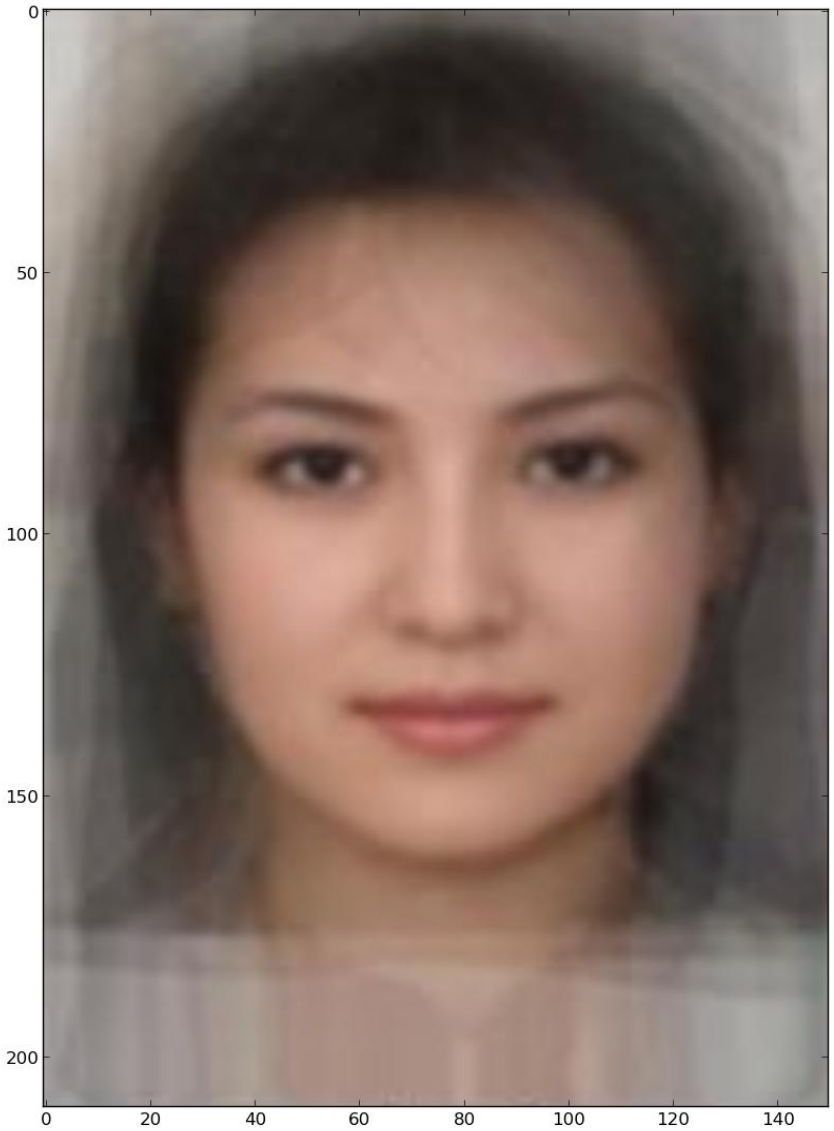

0
50
100
150
200
0
20
40
60
80
100
120
140

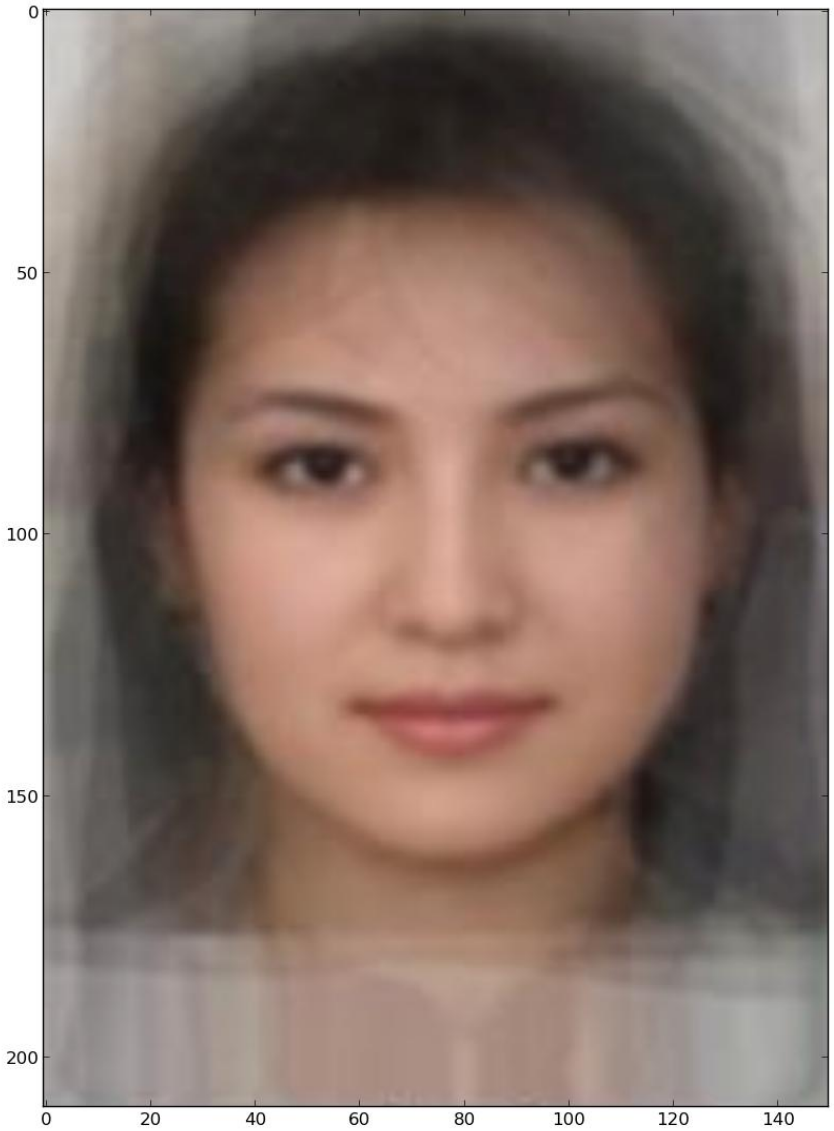

0
50
100
150
200
0
20
40
60
80
100
120
140

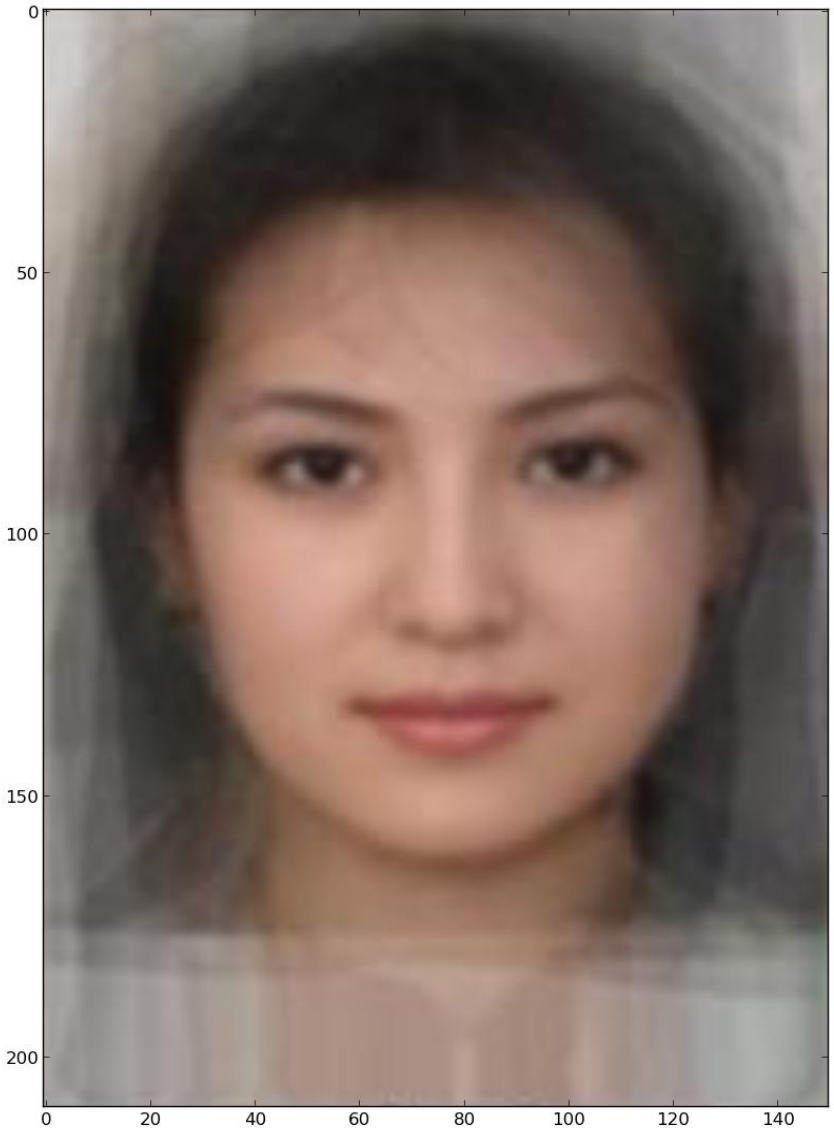

0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140


0
50
100
150
200
0
20
40
60
80
100
120
140

Python

```python
o = imread('ouzbek.png')

w = imread('welsh.png')

def fondu(n):
    for k in range(n+1):
        t = k / float(n)
        imshow(t*o + (1-t)*w)
        savefig("ow" + str(k) + ".jpg")
```

LaTeX

```latex
\multido{\n=0+1}{21}{
\begin{frame}
\begin{center}
  \includegraphics[height=\textheight]{ow\n}
\end{center}
\end{frame}}
```

Python

```python
o = imread('ouzbek.png')

w = imread('welsh.png')

def fondu(n):
    for k in range(n+1):
        t = k / float(n)
        imshow(t*o + (1-t)*w)
        savefig("ow" + str(k) + ".jpg")
```

LaTeX

```latex
\multido{\n=0+1}{21}{
\begin{frame}
\begin{center}
  \includegraphics[height=\textheight]{ow\n}
\end{center}
\end{frame}}
```

Définition 40 (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.

Théorème 41 (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 :*

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

Théorème 41 (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 :*

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

Théorème 41 (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 :*

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

Théorème 41 (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 :*

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

Théorème 42 (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 :*

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

Théorème 42 (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 :*

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

Théorème 42 (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 :*

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