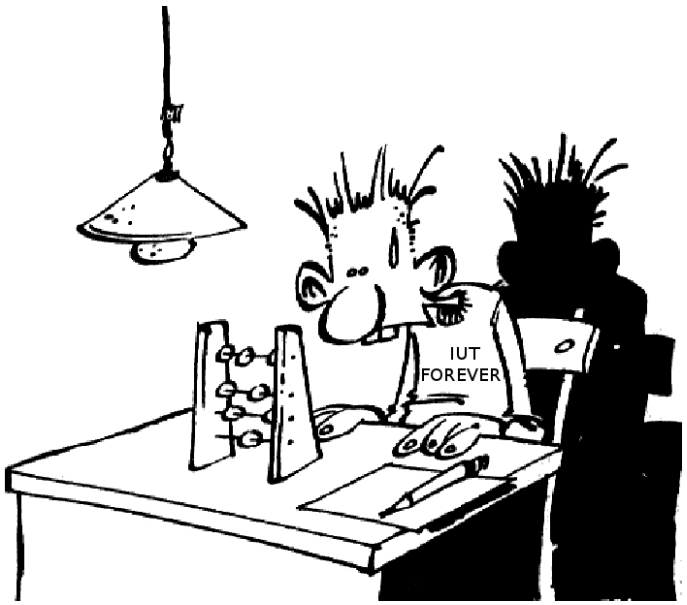
Mathématiques

IUT INFO 1 / 2012-2013

MAJ: 17 mars 2013

Mathématiques discrètes pour l'informatique (IV)



Guillaume CONNAN

IUT NANTES

UNIVERSITÉ DE NANTES

TABLE DES MATIÈRES

CHAPITRE

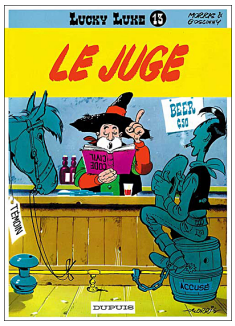
1

Arithmétique pour l'informatique

En 1995, le mathématicien Andrew WILES fit sensation en démontrant le grand théorème de FERMAT. Ce vieux problème d'arithmétique avait tenu les mathématiciens en haleine pendant trois siècles. Pour le vaincre, WILES travailla sur les courbes elliptiques dont l'étude mêle arithmétique, structures algébriques, géométrie, analyse. Ces travaux et leurs dérivés ont permis d'assurer la sécurité des cartes à puces, des échanges sur internet : la recherche fondamentale en mathématiques sur des nombres entiers fit faire un bon en avant à l'informatique. Quoi de plus normal après tout : un ordinateur n'est rien de plus qu'une machine calculant avec des zéros et des uns...

1 Structures mères

1 1 Lois de composition interne



La notion de loi de composition interne est primordiale, tant en mathématiques qu'en informatique : la notion de type en informatique impose un certain respect de la loi...

```
        Objective Caml version 3.12.0

# 1 + 1.5 ;;
      ^^^
"Error: This expression has type float but an expression was expected of type int"
# 1. +. 1.5 ;;
- : float = 2.5
# 1 + 2 ;;
- : int = 3
```

On n'additionne pas n'importe qui avec n'importe quoi : il ne faudra pas confondre la loi **+.** qui opère sur les flottants avec la loi **+** qui opère sur les entiers...
Formalisons la chose :

Définition 1 - 1

Loi de composition interne
Un loi de composition interne $\star$ définie sur un ensemble E est une application de $E \times E$ vers E. On dit alors que $(E, \star)$ est un **magma**.

Le plus souvent, si $\star$ est la loi, on note $x \star y$ au lieu de $\star(x, y)$.
Par exemple $(\mathbb{N}, +)$, $(\mathcal{P}(E), \cap)$, un langage L muni de la concaténation, sont trois magmas mais $(\mathbb{N}, -)$ n'est pas un magma.

Définition 1 - 2

Élément neutre
C'est un élément de E qui vérifie, pour tout élément x de E,

$$e \star x = x \star e = x$$

Quels sont les éléments neutres des magmas cités plus haut ? Si un magma admet un élément neutre, celui-ci est unique : pouvez-vous le démontrer ?

Définition 1 - 3

Monoïde
On appelle monoïde tout couple $(E, \star)$ tel que la loi $\star$ soit *associative*. Si en plus le monoïde admet un élément neutre, on dit que le monoïde est **libre** (ou **unitaire**).

Les magmas précédemment cités sont-ils des monoïdes ? Des monoïdes libres ?

Théorème 1 - 1

Sous-monoïde
Soit $(E, \star)$ un monoïde et $M \subseteq E$ tel que M soit *stable* par $\star$. Alors $(M, \star)$ est un monoïde : c'est un **sous-monoïde** de $(E, \star)$.

Enfin, le définition suivante introduit une notion fondamentale en arithmétique et en algèbre en général :

Définition 1 - 4

Élément inversible
Soit $(E, \star)$ un monoïde unitaire d'élément neutre e. Un élément x de E est inversible (ou symétrisable) par rapport à $\star$ s'il existe y dans E tel que

$$x \star y = y \star x = e$$

Montrez que si l'élément neutre existe a admet un symétrique, celui-ci est unique.

1 2 Les 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 les mathématiques et ses travaux ont eu une très grande influence sur le développement récent de la cryptographie depuis ces trente dernières années.
La notion de groupe a d'abord été liée à la recherche des solutions d'une équation : peut-on trouver un nombre entier naturel n tel que $n+2=0$?
Comme ce genre de problème a besoin d'être résolu (j'ai monté deux étages et je suis arrivé au rez-de-chaussée : de quel étage suis-je parti ?), la structure de groupe a été étudiée de près.
Vous me direz, ce genre de problème n'est pas vraiment révolutionnaire ni utile. Pourtant, nous verrons en TD que cette structure permet d'assurer la sécurité des cartes à puces et des échanges sur la toile...
Donnons-en tout d'abord une définition :

Définition 1 - 5

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

Les lois de groupe sont souvent notées $\times$ ou $+$.
On note alors :

$$\underbrace{x+x+\cdots+x}_{k \text{ fois}} = k\cdot x \qquad \text{ou} \qquad \underbrace{x\times x\times\cdots\times x}_{k \text{ fois}} = x^k$$

Que peut-on dire de $\mathbb{Z}$ muni de l'addition ?
Nous travaillerons en général avec des groupes ayant seulement un nombre fini d'éléments :

Définition 1 - 6

Groupe fini
Soit $(G,+)$ un groupe d'élément neutre 0_E ayant un nombre fini d'éléments. Le cardinal de G est appelé l'**ordre du groupe** G.
Soit x un élément de G. L'**ordre de l'élément** x est le plus petit entier k tel que $k\cdot x = 0$.

Étudions maintenant le groupe qui nous occupera le restant de notre chapitre...

1 3 Le groupe (Z/nZ,+)

1 3 1 Congruence

Définition 1 - 7

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

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

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

Cette notion de « modulo » est entrée dans le langage courant du geek :

> « *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

Vérifiez tout d'abord que la relation de congruence est une *relation d'équivalence.* On peut donc définir un ensemble quotient, c'est-à-dire les relations d'équivalence.
Prenons un exemple innocent : travaillons modulo 2. Combien y a-t-il de classes d'équivalence ? Quel est l'ensemble quotient ?
On a pris l'habitude de noter $\mathbb{Z}/_{n\mathbb{Z}}$ l'ensemble quotient de la relation de congruence modulo n. On notera dans ce cours $\overline{k}^n$ la classe de k modulo n.
Attention ! La plupart des langages possèdent une fonction `mod` ou quelque chose d'équivalent. Ainsi `a mod n` renvoie parfois l'élément de la classe d'équivalence de `a` compris entre 0 et $n-1$, d'autre fois entre $-n+1$ et $n-1$ selon le signe de a mais dans tous les cas le nombre qui vérifie `a = (a/n)*n + x mod n`.

```
        Objective Caml version 3.12.0

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

```
Python 2.7.2+ (default, Oct 4 2011, 20:06:09)
>>> 17 % 3
2
>>> -17 % 3
1
>>> -17/3
-6
>>> (-17 // 3)*3 + (-17 % 3)
-17
```

1 3 2 Division euclidienne

Pour éviter ce genre d'ambiguïté, nous allons nous mettre d'accord grâce au théorème suivant :

Théorème 1 - 2

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

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

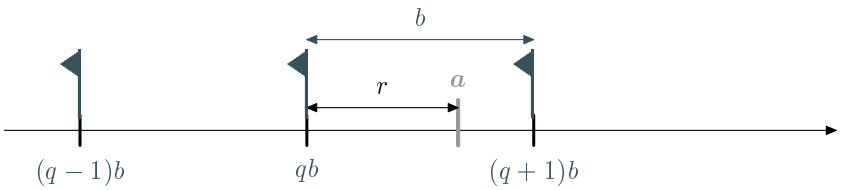
Déterminer q et r, c'est effectuer la division euclidienne de a par b.
On appelle a le dividende, b le diviseur, q le quotient et r le reste.

Remarque

Si l'on remplace la condition sur le reste par $0 \leqslant |r| < b$, il n'y a plus unicité (c'est d'ailleurs la formulation habituelle qui permet de travailler dans les mêmes conditions dans des structures plus vastes qu'on appelle *anneaux euclidiens*) ce qui explique que Python et Ocaml ne sont pas d'accord...

Papyrus d'une copie des éléments d'EUCLIDE datant du premier siècle après JC, quatre siècles après la mort du mathématicien grec.

Qui dit théorème dit démonstration. Le principe de celle qui va suivre est le schéma suivant



qui traduit qu'un entier a est soit un multiple de b, soit est encadré par deux multiples consécutifs de b.
L'énoncé contient deux termes importants : « il *existe* un *unique* couple ». Il va donc falloir prouver deux choses :
- qu'un tel couple existe
- qu'il est unique.

Traitons d'abord le cas particulier où $a \in \mathbb{N}$.

Existence Le monde des multiples de b se sépare en deux catégories : ceux qui sont inférieurs (ou égaux) à a et les autres.

En d'autres termes, appelons $\mathcal{M}_i$ l'ensemble des multiples de b inférieurs ou égaux à a. Cet ensemble $\mathcal{M}_i$ est non vide car il contient au moins 0. De plus $\mathcal{M}_i$ est majoré par a puisqu'on l'a défini comme ça.

L'ensemble $\mathcal{M}_i$ est donc une partie de $\mathbb{N}$ non vide et majorée : $\mathcal{M}_i$ admet donc un plus grand élément d'après une propriété que vous avez démontrée.

Appelons μ ce plus grand élément. C'est un multiple de b, donc il existe un entier q tel que $\mu = qb$.
Le multiple suivant est $\mu+b = qb+b = (q+1)b$ qui n'appartient pas à $\mathcal{M}_i$ car il est strictement supérieur au maximum μ.
On en déduit que

$$bq \leqslant a < bq + b$$

C'est ce que nous « montrait » le dessin. Il nous indique aussi que le reste correspond en fait à la différence $a - bq$.
Posons donc $r = a - bq$: on a alors $a = bq + r$ et donc $bq \leqslant bq + r < bq + b$, c'est à dire $0 \leqslant r < b$
Nous avons donc démontré l'existence de deux entiers q et r tels que $a = bq + r$ avec $0 \leqslant r < b$, *inspirés* que nous étions par un joli dessin.
Il reste à démontrer l'unicité d'un tel couple d'entiers.

unicité Une méthode habituelle pour démontrer une unicité est *supposer* l'existence d'un second couple.
Supposons donc qu'il existe deux couples d'entiers (b, q) et (b', q') vérifiant

$$a = bq + r \quad \text{avec} \quad 0 \leqslant r < b$$
$$a = bq' + r' \quad \text{avec} \quad 0 \leqslant r' < b$$

Effectuons la différence membre à membre de ces égalités. On obtient

$$0 = b(q - q') + r - r' \quad \text{avec} \quad -b < r - r' < b$$

Vous en déduisez comme moi que $r - r'$ est un multiple de b, et que ce multiple est strictement compris entre $-b$ et b.
Le seul multiple qui convient est 0, donc $r - r' = 0$ et par suite $q - q' = 0$, c'est à dire que $r = r'$ et $q = q'$.
Dès lors il n'existe qu'un seul couple solution.

Définition 1 - 8

Diviseur
Un entier non nul d divise (ou est un diviseur de) l'entier a si, et seulement si, le reste de la division euclidienne de a par d est nul.

Recherche

Pourquoi dit-on habituellement qu'on ne peut pas diviser par zéro ? D'ailleurs, dans de nombreux cas, le debugger de votre langage préféré renvoie des messages du type `Exception: Division_by_zero`...
Vous traiterez ensuite le cas où a est strictement négatif connaissant bien sûr le résultat pour $a \in \mathbb{N}$.

En général, on prend comme *représentant principal* de la classe de a modulo n le reste de la division de a par n. Par exemple, on notera :

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

Danger

Notez bien que $\overline{0}^8$, $\overline{1}^8$, etc. sont des ensembles d'entiers !...
Ici, $\overline{0}^8 = \{..., -16, -8, 0, 8, 16, 24, 32, ...\}$
Par **ABUS DE NOTATION**, on écrira le plus souvent k pour désigner $\overline{k}^n$ ce qui nous amènera à écrire par exemple que $8 = 0$, voire $3 \times 5 = 7 = -1$ comme nous le verrons plus loin...

Donnons quelques théorèmes importants que vous démontrerez en TD :

Théorème 1 - 3

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

Cette propriété nous permet en fait d'exploiter autrement les congruences. En effet, si par exemple $x \equiv 5 (\mathrm{mod}\, 32)$, cela signifie qu'il existe un entier k tel que $x - 5 = 32k$, soit encore que $x = 5 + 32k$.
Voyons une autre formulation utile :

Théorème 1 - 4

$$a \equiv b (\mathrm{mod}\, n) \iff \exists k \in \mathbb{Z} \mid a = b + kn$$

Ça ressemble à la division euclidienne de a par n mais ça peut ne pas l'être : n'oubliez pas que le reste doit obligatoirement être positif et strictement inférieur au diviseur.
Par exemple on a bien $33 \equiv 97 (\mathrm{mod}\, 32)$ mais 97 n'est certes pas le reste de la division de 33 par 32.
Il faudrait maintenant pouvoir définir des opérations sur les classes de congruences. On aurait envie de dire que la somme des classes de a et de b modulo n est en fait la classe de la somme $a + b$ modulo n et pareil pour la multiplication mais est-ce licite ?

1 3 3 Premières propriétés

Nous aurons besoin des propriétés suivantes que vous allez prouver en TD :

Théorème 1 - 5

1. $\text{Card}\left(\mathbb{Z}/_{n\mathbb{Z}}\right) = n$
2. $a \equiv a \pmod n$
3. Si $a \equiv b \pmod n$, alors $b \equiv a \pmod n$
4. Si $a \equiv b \pmod n$ et $b \equiv c \pmod n$, alors $a \equiv c \pmod n$
5. Si $a \equiv b \pmod n$ et $a' \equiv b' \pmod n$, alors
$$a + a' \equiv b + b' \pmod n \quad a - a' \equiv b - b' \pmod n \quad aa' \equiv bb' \pmod n$$
6. Si $a \equiv b \pmod n$, alors, pour tout $p \in \mathbb{N}$, $a^p \equiv b^p \pmod n$

En fait, il faut retenir qu'on peut additionner, soustraire, multiplier des congruences **de même module**.
Nous avons oublié la division et pour cause : c'est une opération à haut risque quand on travaille avec des entiers !
Par exemple $12 \equiv 0 \pmod 6$, mais $\frac{12}{3} \not\equiv \frac{0}{3} \pmod 6$.

1 3 4 Loi de groupe

La propriété 5 du théorème 1 - 5 nous permet de définir une loi d'addition et une loi de multiplication sur $\mathbb{Z}/_{n\mathbb{Z}}$:

Théorème 1 - 6

$$\forall n \in \mathbb{N}^*,\ \forall (x,y) \in \mathbb{Z}^2,\quad \overline{x}^n + \overline{y}^n = \overline{x+y}^n,\quad \overline{x}^n \cdot \overline{y}^n = \overline{x \cdot y}^n$$

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 $\left(\mathbb{Z}/_{n\mathbb{Z}}, +\right)$ et $\left(\mathbb{Z}/_{n\mathbb{Z}}, \cdot\right)$ sont des groupes ?
Dire que p est **inversible modulo** n signifie qu'il existe un entier p' tel que $p \cdot p' = 1 \pmod n$, c'est-à-dire qu'il existe un entier k tel que $pp' = 1 + kn$. Nous venons de voir pour certaines valeurs de n, cette recherche semble possible et qu'elle pose problème dans d'autres cas. Nous allons avoir besoin de quelques théorèmes de plus pour conclure à coup sûr, cette notion d'inverse modulaire étant centrale en cryptographie donc en sécurité informatique.

1 3 5 Calcul modulaire en Caml

On peut créer un type enregistrement pour travailler modulo un entier :

```
type classe  =
  {representant : int ; modulo : int};;
```

Pour plus de commodité, on va créer un opérateur infixe correspondant à notre $\cdots \equiv \cdots [\cdots]$:

```
let (%) a n = {representant = (a mod n); modulo = n};;
```

On obtient alors :

```
# 13 % 3;;
- : classe = {representant = 1; modulo = 3}
```

On peut ensuite créer des lois induites :

```
exception Classes_incompatibles;;

let loi_induite loi =  fun a  b ->
  {modulo =
      if a.modulo = b.modulo
      then a.modulo
      else raise Classes_incompatibles ;
   representant =
      (loi (a.representant) (b.representant)) mod (a.modulo)
  };;

let ( +% ) = loi_induite  ( + ) ;;
let ( *% ) = loi_induite  ( * ) ;;
```

Alors, pour obtenir $9 + 7$ modulo 5 :

```
# (9%5) *% (7%5);;
- : classe = {representant = 3; modulo = 5}
```

2 Divisibilité dans Z

2 1 Propriété fondamentale de N

Nous admettrons la propriété suivante qui nous sera très utile par la suite :

Théorème 1 - 7

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

Est-ce que cette propriété est encore vraie dans $\mathbb{Z}$? Dans $\mathbb{R}$?

2 2 PGCD

Considérons un entier relatif a. On notera $\mathcal{D}(a)$ l'ensemble de ses diviseurs dans $\mathbb{Z}$.
Par exemple, $\mathcal{D}(0) = \mathbb{Z}$, $\mathcal{D}(1) = \{-1, 1\}$, $\mathcal{D}(12) = \{-12, -6, -4, -3, -2, -1, 1, 2, 3, 4, 6, 12\}$ et plus généralement

$$\mathcal{D}(a) = \{-|a|, ..., -1, 1, ..., |a|\}$$

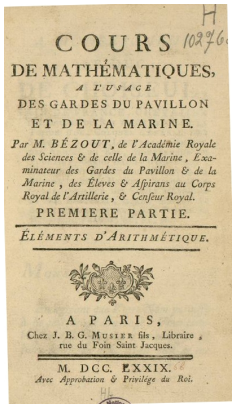
Définition 1 - 9

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.

Le PGCD de a et b est donc le plus grand élément de $\mathcal{D}(a) \cap \mathcal{D}(b)$: la notation le rappelle. Comme toute partie non vide et majorée de $\mathbb{N}$ admet un plus grand élément, cette définition en est bien une car $\mathcal{D}(a) \cap \mathcal{D}(b)$ contient au moins 1.

2 3 Égalité de Bézout


COURS
DE MATHÉMATIQUES,
A L'USAGE
DES GARDES DU PAVILLON
ET DE LA MARINE.
Par M. BÉZOUT, de l'Académie Royale des Sciences & de celle de la Marine, Examinateur des Gardes du Pavillon & de la Marine, des Élèves & Aspirans au Corps Royal de l'Artillerie, & Censeur Royal.
PREMIERE PARTIE.
ÉLÉMENTS D'ARITHMÉTIQUE.
A PARIS,
Chez J. B. G. Musier fils, Libraire, rue du Foin Saint Jacques.
M. DCC. LXXIX.
Avec Approbation & Privilège du Roi.

Première page du cours écrit par Étienne BÉZOUT (1730-1783)

Voici la clé de notre section : nous allons montrer que notre PGCD est en fait une combinaison linéaire de a et de b. Considérons d'abord le cas où a et b sont non nuls et notons

$$S = \{au + bv \mid (u, v) \in \mathbb{Z}^2 \text{ et } au + bv > 0\}$$

L'ensemble S est constitué d'entiers naturels et est non vide car il contient au moins $a^2 + b^2$: il admet donc un unique **plus petit élément** que nous noterons d qui s'écrit donc $d = au_0 + bv_0$.
Notre mission va bien sûr consister à prouver que d est en fait le PGCD de a et b.
Nous allons pour cela utiliser une ficelle classique : introduire le reste de la division euclidienne de a par d et utiliser le fait que d est le plus petit élément de S puis raisonner par l'absurde.
La division de a par d s'écrit :

$$a = dq + r \quad \text{avec} \quad 0 \leqslant r < d$$

Supposons que $r > 0$, alors

$$r = a - dq = a - q(au_0 + bv_0) = a(1 - qu_0) + b(-qv_0) > 0$$

et donc $r \in S$ or $r < d$. C'est impossible car d est le plus petit élément de S : **contradiction**.
Notre supposition de départ est donc fausse et $r = 0$. Nous en déduisons que d divise a.
On montre de manière similaire que d divise b. Ainsi

$$d \text{ est un diviseur commun de } a \text{ et } b$$

Il nous reste à montrer que c'est le plus grand. Soit δ un diviseur commun à a et b quelconque. On montre facilement qu'alors δ divise toute combinaison linéaire de a et de b donc en particulier δ divise $au_0 + bv_0$ donc d. Alors $c \leqslant |d| = d$, donc d est bien le plus grand des diviseurs.

Il faut encore vérifier que le PGCD est **unique**. La méthode habituelle est de supposer qu'il existe un deuxième PGCD, disons d'. Alors $d \leqslant d'$ car d' est le plus grand diviseur puis $d' \leqslant d$ car d aussi et finalement $d = d'$ ce qui assure l'unicité.

Comme $|a| = \pm 1 \times a + 0 \times 0$, l'égalité tient toujours si b (ou a) est nul, donc

Théorème 1 - 8

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 :

$$au + bv = d$$

2 4 Algorithme des différences

Voyons maintenant une propriété qui va être à la base de la construction algorithmique du PGCD :

Théorème 1 - 9

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

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

et en particulier

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

Notons $a \wedge b = d$ et $a \wedge (b + ka) = d'$. Alors

- d divisant a et b, d divise $(ka + b)$ et a donc divise d' d'après une propriété précédente.
- d'autre part, d' divise a et $b + ka$, donc divise $b + ka - ka$, donc b, donc d' est un diviseur commun à a et b donc d' divise d.

Comme $d|d'$ et $d'|d$, on a donc $d = d'$.

Cette propriété va nous permettre de fabriquer un algorithme de calcul de $a \wedge b$. En effet, si $a \geqslant b$, et comme $a \wedge b = b \wedge (a - b)$, on calcule le PGCD de deux entiers plus petits. Et on réitère. Par exemple :

$$15 \wedge 12 = 12 \wedge 3 = 9 \wedge 3 = 6 \wedge 3 = 3 \wedge 3 = 3 \wedge 0 = 3$$

Comment se persuader que cette descente aboutit ? Nous pouvons aisément nous convaincre sans se perdre dans trop de formalisme que :

Théorème 1 - 10

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

Et cette constante ne pouvant être par construction que 0, le PGCD de a et b sera la dernière valeur non nulle de cette suite car $k \wedge 0 = k$ pour tout entier k.
Ce procédé semble assez long : beaucoup d'opérations sont inutiles. Malgré tout il n'utilise que des sommes et des différences qu'un ordinateur traite extrêmement rapidement.

2 5 Algorithme d'Euclide I

C'est le même principe que ce que nous venons de faire, mais en plus rapide car nous remarquons que, en notant

$$a = bq_0 + r_0 \quad \text{avec } 0 \leqslant r_0 < b$$

la division euclidienne de a par b, on obtient que

$$a \wedge b = b \wedge (a - bq_0) = b \wedge r_0$$

Puis en notant

$$b = r_0 q_1 + r_1 \quad \text{avec } 0 \leqslant r_1 < r_0$$

on obtient de même que

$$a \wedge b = b \wedge r_0 = r_0 \wedge r_1$$

et on continue ainsi à fabriquer une suite strictement décroissante d'entiers naturels : elle est donc finie et son dernier terme $r_p = 0$.
Alors

$$a \wedge b = r_{p-1} \wedge r_p = r_{p-1} \wedge 0 = r_{p-1} = \text{ le dernier reste non nul}$$

Soit, à la mode spaghetti, en supposant déterminée une fonction **rem** qui renvoie le reste de deux entiers :

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

2 6 Algorithme d'Euclide II

L'algorithme d'Euclide nous assure que le PGCD de a et b est le dernier reste non nul de la suite (r_k) construite au paragraphe précédent. L'égalité de Bézout nous assure de l'existence de deux entiers u et v tels que $au+bv=a\wedge b$.
Nous allons essayer de combiner les deux en construisant deux suites (u_k) et (v_k) telles que

$$r_k = au_k + bv_k$$

Voici le départ

- $r_0 = a = 1\times a + 0\times b$
- $r_1 = b = 0\times a + 1\times b$
- $r_2 = \texttt{rem}\,(r_0, r_1)$

  $\vdots$
- $r_{i+1} = \texttt{rem}\,(r_{i-1}, r_i)$

  $\vdots$
- $r_{p-1} = a\wedge b$
- $r_p = 0$

Or par définition, comme $r_2 = \texttt{rem}\,(r_0, r_1)$, on a $r_0 = q_2 r_1 + r_2$, donc $r_2 = 1\times r_0 - q_2 \times r_1$, c'est à dire

$$u_2 = 1 \text{ et } v_2 = -q_2$$

On réitère le mécanisme. Supposons que $r_{k-1} = u_{k-1}\times a + v_{k-1}\times b$ et $r_k = u_k \times a + v_k \times b$
Comme $r_{k+1} = \texttt{rem}\,(r_k, r_{k-1})$, on a $r_{k-1} = q_{k+1} r_k + r_{k+1}$, donc $r_{k+1} = 1\times r_{k-1} - q_{k+1}\times r_k$, c'est à dire

$$r_{k+1} = 1\times(u_{k-1}\times a + v_{k-1}\times b) - q_{k+1}\times(u_k\times a + v_k\times b)$$

Finalement

$$u_{k+1} = u_{k-1} - u_k \times q_{k+1} \text{ et } v_{k+1} = v_{k-1} - v_k \times q_{k+1}$$

Ça a l'air un peu brut comme ça, au premier coup d'œil, mais en fait il y a une disposition pratique qui permet de mieux voir ce qui se passe. D'abord, dans algorithme d'Euclide étendu il y a algorithme d'Euclide, donc on commence par rechercher le PGCD de a et b par l'algorithme d'Euclide en n'oubliant pas cette fois de noter les quotients en remplissant le tableau suivant :

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			$r_p = 0$	

Et le secret tient dans le schéma

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

et pareil pour les v_k et les r_k.
Par exemple, pour 19 et 15 :

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		$L_5 \leftarrow L_3 - 3 \times L_4$

Le dernier reste non nul est 1 donc $19 \wedge 15 = 1$ et $4 \times 19 - 5 \times 15 = 1$
En version récursive, on peut procéder en deux étapes :

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

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

2.7 Nombres premiers entre eux

Définition 1 - 10

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

$$a \wedge b = 1$$

L'égalité de BÉZOUT vue précédemment nous permet donc dénoncer le théorème suivant :

Théorème 1 - 11

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

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

Nous avons donc en main tous les éléments pour étudier les éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$ mais avant, une petite digression...

3 À la recherche des nombres premiers

3.1 Définition

Quoi de plus simple qu'un nombre premier :

Définition 1 - 11

Un entier naturel est dit premier s'il est supérieur (i.e. supérieur ou égal) à 2 et n'est divisible que par 1 et lui-même.

et pourtant, ils renferment tant de mystères que les plus grands esprits depuis des siècles n'ont toujours pas réussi à en percer tous les secrets et ce malgré les énormes progrès technologiques et les investissements colossaux consentis par les pouvoirs tant civils que militaires pour assurer ou percer la confidentialité des transmissions de toutes natures qui continuent de dépendre d'une meilleure connaissance des nombres premiers, ce que nous verrons un peu plus loin. Qui sont-ils ? Combien sont-ils ? Où sont-ils ? À quoi servent-ils ? Nous essaierons de donner quelques éléments de réponses à ces questions.
Mais tout d'abord, pourquoi jouent-ils un rôle si important ? Fondamentalement, il existe deux manières d'engendrer $\mathbb{N}$:

– si on veut engendrer $\mathbb{N}$ en utilisant l'addition, on s'aperçoit que le nombre 1 nous suffit : on « fabrique » 2 en additionnant 1 avec lui-même ; 3 en additionnant 1 avec 2, etc.

– si on veut engendrer $\mathbb{N}$ en utilisant la multiplication, là, les choses se compliquent. Pour « fabriquer » 2, il faut le créer ; même problème pour 3. On fabrique 4 en multipliant 2 avec lui-même, mais il faut créer 5. On fabrique 6 en multipliant 3 avec 2. On crée 7. On fabrique 8 à partir de 2. On fabrique 9 à partir de 3. On fabrique 10 à partir de 2 et 5, etc.

Les nombres que l'on est obligés de créer sont les briques nécessaires à fabriquer tous les autres. C'est bien plus compliqué que l'addition me direz-vous, mais la multiplication est plus « puissante » et nous permet d'aller bien plus vite et plus loin.

Les nombres premiers sont donc ces éléments qui nous permettent de fabriquer tous les autres. Un des premiers problèmes étudiés à été de savoir s'ils peuvent tenir dans une boîte. Euclide à répondu à cette question il y a vingt-trois siècles et la réponse est non.
Pour le prouver, nous aurons besoin d'un résultat intermédiaire :

Théorème 1 - 12

Tout entier naturel admet au moins un diviseur premier.

Si ce nombre, appelons-le n, est premier, tout va bien.
Sinon, l'ensemble de ses diviseurs étant non vide (n est dedans) et borné (par 1 et lui-même), il admet un plus petit élément $p \neq 1$. Ce nombre n'est pas divisible par un autre, sinon ce nombre plus petit que p divisant p diviserait n et alors p ne serait plus le plus petit diviseur de n. Le nombre p est donc premier et voilà.

Théorème 1 - 13

Il y a une infinité de nombres premiers.

Raisonnons par l'absurde et supposons qu'il existe exactement n nombres premiers qu'on nommera p_1, $p_2, \cdots, p_n$ et appelons N le nombre

$$\mathrm{N} = p_1 p_2 \cdots p_n + 1$$

Il est plus grand que tous les p_i, donc il n'est pas premier d'après notre hypothèse, donc il admet un diviseur premier p qui est donc un des p_i, puisqu'il n'y a qu'eux. Soit i_0 tel que $p = p_{i_0}$. Alors p divise $p_1 p_2 \cdots p_n$. Or il divise N, donc il divise leur différence $\mathrm{N} - p_1 p_2 \cdots p_n$, c'est à dire 1, donc $p = 1$ ce qui est absurde puisqu'il est premier. Ainsi, il n'existe pas de plus grand nombre premier.

Il y en a donc une infinité et l'aventure ne fait que commencer.

3 2 Comment vérifier qu'un nombre est premier ?

Observons les diviseurs de 1321 par exemple : si aucune des 1319 divisions ne « tombe juste », alors on pourra dire que 1321 premier. Et puis d'abord, il est impair, il ne semble pas être divisible par 3, alors pourquoi pas...

diviseur	2	3	4	5	6	7	8	9	10	11	12	13
quotient	660,5	440,3	330,3	264,2	220,2	188,7	165,1	146,8	132,1	120,1	110,1	101,6
diviseur	14	15	16	17	18	19	20	21	22	23	24	25
quotient	94,36	88,07	82,56	77,71	73,39	69,53	66,05	62,90	60,05	57,43	55,04	52,84
diviseur	26	27	28	29	30	31	32	33	34	35	36	37
quotient	50,81	48,93	47,18	45,55	44,03	42,61	41,28	40,03	38,85	37,74	36,69	35,70

Le tableau est incomplet : il reste encore à essayer les quotients de 38 à 1320, mais cela aurait été mauvais pour la planète.
Tout d'abord, nous n'avons pas trouvé de quotient entier en ce début d'enquête. Nous observons de plus que si la suite des diviseurs est croissante, celle des quotients est décroissante. Vous avez sûrement remarqué qu'à partir de 37, les quotients sont inférieurs aux diviseurs donc nous pouvons nous arrêter là : si pour un diviseur supérieur à 37, on trouvait un quotient entier q, alors en divisant 1321 par q, qui est inférieur à 37, on devrait se trouver au début de notre liste de diviseurs et ce diviseur serait entier. Le problème, c'est qu'aucune division par un entier inférieur à 37 n'a donné de quotient entier donc on peut s'épargner les 1283 divisions restantes.

Mine de rien, nous venons de franchir un grand pas dans la théorie des nombres : pour tester si l'entier 1321 est premier, il nous a suffit d'effectuer 36 divisions [a]. Deux problèmes se posent maintenant : pourquoi 36 et pouvons-nous généraliser ce résultat aux autres entiers ?
Un petit Joker : $\sqrt{1321} \approx 36,3$...

a. Et encore, nous aurions pu faire mieux comme nous allons le voir tout de suite après.

Reprenons la propriété 1 - 12 page précédente. Tout naturel n admet un plus petit diviseur p qui est premier. Écartons le cas où n est premier et donnons un nom aux nombres qui restent :

Définition 1 - 12

Un entier naturel autre que 1 qui n'est pas premier est dit **composé**.

L'entier n étant composé, il s'écrit donc $n = pq$, avec q un entier supérieur à p (car p est le plus petit diviseur). Ainsi

$$p \leqslant q \text{ et donc } p^2 \leqslant pq = n \text{ c'est à dire } p \leqslant \sqrt{n}$$

Nous pouvons donc généraliser l'observation précédente

Théorème 1 - 14

Si un entier est composé, alors il admet un diviseur premier inférieur à sa racine carrée.

3 3 Tests et cribles

La théorie des nombres (i.e. la partie des mathématiques qui étudie les nombres premiers) est actuellement intimement liée à l'informatique et vice versa. Même à notre petit niveau, nous ne pouvons donc pas passer à côté...

3 3 1 Tests naïfs

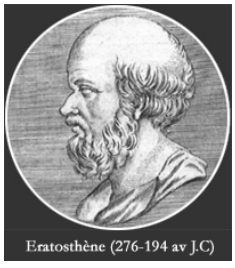
On teste tous les entiers de 2 à $\sqrt{n}$ en passant par une petite fonction locale récursive :

```
Fonction test1_tmp(t,n:entiers):booléen
Si t * t > n Alors
    Retourner Vrai
Sinon
    Si rem(n,t) = 0 Alors
        Retourner Faux
    Sinon
        Retourner test1_tmp(t+1,n)
    FinSi
FinSi
```

```
Fonction test1(n:entier):booléen
Retourner test1_local(2,n)
```

On pourrait déjà diviser facilement le nombre de tests par 2, voire plus...

3 3 2 Crible d'Ératosthène



Né 276 années avant JiCé, directeur de la Bibilothèque d'Alexandrie, on lui doit aussi une approximation du diamètre de la Terre. Devenu aveugle, il s'est laissé mourir de faim...

Autre problème crucial : comment obtenir une liste des entiers inférieurs à un petit nombre donné n ?
Le principe est ancien puisqu'il est attribué au grec ÉRATOSTHÈNE
On écrit les entiers de 2 à n puis on barre les multiples des nombres premiers inférieurs à $\sqrt{n}$. Les entiers restant sont premiers.
La programmation en Ocaml est un peu plus délicate donc nous allons la détailler cette fois ensemble.
On crée une fonction qui retire les multiples d'un entier dans une liste donnée. Pour cela, on utilise la fonction **`filter`** du module **`List`** dont voici la documentation :

```
val filter : ('a -> bool) -> 'a list -> 'a list

filter p l returns all the elements of the list l that satisfy the predicate p. The order of the
     elements in the input list is preserved.
```

Complétez alors

```
let retmultiples = fun liste n -> ...
```

Pour enlever les multiples de 3 compris entre 2 et 12 :

```
# retmultiples [2;3;4;5;6;7;8;9;10;11;12] 3 ;;
- : int list = [2; 4; 5; 7; 8; 10; 11]
```

On crée aussi une fonction qui renvoie la liste des entiers de **min** à **max** :

```
let liste_entiers = fun min max ->
  let rec aux = fun a acc ->
    if ...
    else ...
  in aux min [];;
```

Enfin, voici le cœur de la procédure :

```
let crible = fun m ->
  let rec crible_rec = fun n acc ->
      if n*n > m then
        acc
      else
        crible_rec (n+1) (retmultiples acc n)
  in crible_rec 2 (liste_entiers 2 m);;
```

3.4 Décomposition des entiers en produit de nombres premiers

Avant d'aller plus loin dans notre exploration, nous avons dit en introduction que les nombres premiers étaient les briques qui nous permettaient de construire tous les autres : il serait bon de le vérifier.

Théorème 1 - 15

Tout entier n supérieur à 2 se décompose en produit fini de nombres premiers.

La démonstration la plus simple (mais pas la plus intéressante) consiste à raisonner par récurrence sur n.
Nous savons que 2 est premier.

Supposons que tout entier inférieur ou égal à n se décompose en produit de facteurs premiers.
L'entier suivant $n+1$ admet au moins un diviseur premier p d'après la propriété 1 - 12 page 14.
Soit q le quotient de $n+1$ par p.
Si $q = 1$, alors $n+1 = p$ et donc $n+1$ est premier.
Si $q > 1$, nous appliquons l'hypothèse de récurrence à q : q se décompose en produit fini de nombres premiers, et par suite $n+1$ aussi, car $n+1 = p \times q$.

Par exemple, $50 = 2 \times 5^2$: les facteurs premiers ne sont pas forcément distincts. On a donc l'habitude d'écrire la décomposition sous la forme

$$n = p_1^{\alpha_1} \times p_2^{\alpha_2} \times \cdots \times p_r^{\alpha_r}$$

Une petite chose reste à vérifier : cette décomposition est-elle unique ? Elle a l'air de l'être, c'est pourquoi...nous l'admettrons :

Théorème 1 - 16

Tout entier n supérieur à 2 admet une et une seule (à l'ordre près des termes) décomposition en produit fini de nombres premiers

Bon, nous en savons assez pour revenir au problème de recherche de l'inverse modulaire d'un entier qui nous ouvrira les portes de la cryptographie.

4 Éléments inversibles de Z/nZ

4.1 Anneaux et corps

Définition 1 - 13

Anneau
Soit A un ensemble muni de deux lois $\cdot$ et $+$ qu'on nommera addition et multiplication.
On dit que $(A, +, \cdot)$ est un anneau si, et seulement si :

- $(A, +)$ est un groupe commutatif ;
- la multiplication est associative ;
- la multiplication est distributive sur l'addition.

Si la multiplication est commutative, alors l'anneau est dit *commutatif*.
Si la multiplication admet un élément neutre, l'anneau est dit *unitaire*.
Les éléments d'un anneau unitaire qui admettent un symétrique par la multiplication 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 ?
Vous démontrerez facilement en TD le théoremm suivant :

Théorème 1 - 17

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

Définition 1 - 14

Corps
Un corps est un anneau commutatif unitaire dans lequel tout élément non nul est inversible.

En anglais, le terme mathématique pour désigner les corps est *field*.
Voici un bon théorème que vous pourrez démontrer en TD :

Théorème 1 - 18

Deux théorèmes en un...

- 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 $\mathbb{F}_n$.

Définition 1 - 15

Fonction indicatrice d'EULER
On note $\varphi(n)$ le nombre d'éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$. C'est donc aussi le cardinal (on dit aussi l'*ordre*) du groupe $\left(\left(\mathbb{Z}/n\mathbb{Z}\right)^*, \cdot\right)$.

Cette fonction joue un rôle important en cryptographie, notamment dans le système RSA.

4 2 Comment calculer l'inverse modulaire d'un entier ?

C'est un problème important qui intervient souvent en cryptographie.
Avec les notations habituelles, le but est de trouver $y \in \{0,1,...,n-1\}$ tel que $x \cdot y \equiv 1 (\text{mod } n)$.

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

Pouvez-vous justifier cette série d'équivalences ? Dans quelle mesure nous donne-t-elle un moyen de déterminer y ? Nous reparlerons de tout cela en TD...
Nous démontrerons également le théorème suivant :

Théorème 1 - 19

Théorème d'EULER

$$a \wedge n = 1 \Longrightarrow a^{\varphi(n)} \equiv 1 (\text{mod } n)$$

En quoi peut-il nous aider à déterminer l'inverse modulaire d'un entier ? Le problème, c'est que le calcul de $\varphi(n)$ peut s'avérer compliqué...

Les propriétés suivantes vont nous aider :

Théorème 1 - 20

Propriétés de $\varphi(n)$

- Si p est premier, alors $\varphi(p) = p-1$;
- si $m \wedge n = 1$ alors $\varphi(mn) = \varphi(m)\varphi(n)$ (admis) ;
- Si p est premier, $\varphi(p^n) = p^n - p^{n-1} = p^n\left(1-\frac{1}{p}\right)$;
- $\varphi(n) = n\prod_{p\in\mathcal{P}_n}\left(1-\frac{1}{p}\right)$ avec $\mathcal{P}_n$ l'ensemble des diviseurs premiers de n.

C'est la dernière propriété qui nous permettra de calculer $\varphi(n)$ avec Ocaml.

4 3 Petit théorème de Fermat

Pierre de Fermat
(1601-1665)

Nous avons mis au point avec le modèle du crible d'ÉRATOSTHÈNE une méthode permettant de tester si un entier est premier ou non. Cela marche assez bien pour des petits nombres, mais cette méthode devient impraticable s'il s'agit de tester un entier d'une centaine de chiffres. Nous allons nous occuper dans cette section de deux problèmes imbriqués : d'une part trouver des méthodes permettant de vérifier rapidement si un nombre est premier et d'autre part réfléchir à d'éventuelles méthodes permettant de « fabriquer » des nombres premiers aussi grands que l'on veut.
Nous avons besoin pour cela d'un dernier résultat : le fameux « petit théorème de FERMAT ».S'il est qualifié de petit, c'est qu'une conjecture célèbre du même Fermat, restée indémontrée pendant des siècles, s'est accaparée le titre de *Grand Théorème de* FERMAT.

$$x^n + y^n = z^n \quad \text{n'a pas de solution dans } \mathbb{N}^3 \quad \text{pour } n > 2$$

Dans la marge d'un manuscrit, FERMAT prétendait en avoir trouvé la démonstration mais manquer de place pour l'écrire. Il a pourtant fallu attendre 1994 pour qu'Andrew WILES le démontre en utilisant des outils surpuissants : l'entêtement d'une multitude de chercheurs a abouti à la démonstration de ce théorème, mais surtout a permis de développer d'importants outils en cryptographie donc en sécurité informatique.
Voici celui qui nous sera utile qui n'est en fait qu'une conséquence du théorème d'EULER...

Théorème 1 - 21

Petit théorème de FERMAT

- Si p est premier et ne divise pas a alors $a^{p-1} \equiv 1 \pmod p$;
- Si p est premier, $a^p \equiv a \pmod p$.

5 EXERCICES

5 1 Structures mères

Exercice 1 - 1

Donnez des exemples de couples (E, $\star$) avec $\star$ une application définie sur E × E telle que (E, $\star$) ne soit pas un magma.

Exercice 1 - 2

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

$\star$	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 (A, $\star$) a une structure de groupe. Comparer ce groupe avec $\left(\mathbb{Z}/_{3\mathbb{Z}}, \overline{+}\right)$ et $\left(\{1, e^{2\pi/3}, e^{-2\pi/3}\}, \times\right)$.

Exercice 1 - 3

Soit A = $\{a_1, a_2, ..., a_p\}$ un alphabet. Nous avons vu au chapitre précédent que $A^\star$ muni de la concaténation des chaînes a une structure de monoïde (unitaire ?). Les ensembles suivants sont-ils des sous-monoïdes de $A^\star$ muni de la concaténation ? Unitaires ?

1. L'ensemble des chaînes de longueur paire ;
2. L'ensemble des chaînes de longueur impaire ;
3. $\{(a_1 a_2)^n \quad | \quad n \in \mathbb{N}\}$;
4. $\{a_1^n a_2^n \quad | \quad n \in \mathbb{N}\}$;
5. L'ensemble des chaînes ne contenant que a_1 et a_2, ceux-ci apparaissant un nombre égal de fois.

5 2 Division euclidienne

Exercice 1 - 4 Quotient et reste

On s'occupe de la division euclidienne d'un entier quelconque a par un entier non nul b.
On voudrait écrire un programme Caml le plus général possible, i.e. qui fonctionne quelque soit le signe de a ou de b.
Dans le cas où a et b sont tous les deux positifs, le petit dessin avec les drapeaux nous invite à écrire :

```
let rec quo_pos= fun a b ->
  if a < b then 0
  else 1 + quo_pos (a-b) b ;;
```

Modifiez ce programme pour qu'il traite aussi le cas $a < 0$ puis le cas général.
Faites de même avec la fonction **`rem a b`** qui renvoie le reste de la division de a par b.

Exercice 1 - 5 Nombre de chiffres

Déterminez une fonction **`nb_chiffres n`** qui renvoie le nombre de chiffres de l'écriture décimale de n.

Exercice 1 - 6 Décomposition en base quelconque

Déterminez une fonction **`base b n`** qui renvoie la liste des chiffres de l'écriture normalisée de n en base b.

```
# base 2 97 ;;
- : int list = [1; 1; 0; 0; 0; 0; 1]
```

Exercice 1 - 7 Persistance d'un entier

Déterminez une fonction récursive qui renvoie la liste des chiffres de l'écriture décimale d'un nombre.
Un entier naturel n étant donné, on calcule le produit **`prod n`** de ses chiffres dans son écriture en base 10 puis le produit des chiffres de **`prod n`** et on recommence ainsi l'application de **`prod`** jusqu'à obtenir un

chiffre entre 0 et 9. Le nombre minimal de fois où on applique **prod** pour transformer n en un chiffre entre 0 et 9 est appelé la *persistance* de n. Par exemple, la persistance de 9 est égale à 0, celle de 97 est égale à 3 car $\texttt{prod}(97) = 9 \cdot 7 = 63$, $\texttt{prod}(63) = 6 \cdot 3 = 18$, $\texttt{prod}(18) = 1 \cdot 8 = 8$. et celle de 9575 est égale à 5.
On voudrait connaître le plus petit entier naturel de persistance 5.

Exercice 1 - 8 Écriture littérale

Déterminez une fonction qui renvoie l'écriture littérale d'un nombre dont l'écriture décimale est comprise entre 1 et 999.

Exercice 1 - 9 Problème de calendrier

Connaissant une date, à quel jour de la semaine correspond-elle ? L'allemand ZELLER a proposé une formule en 1885.
On note m le numéro du mois à partir de janvier, s le numéro du siècle, a le numéro de l'année dans le siècle s, j le numéro du jour. Pour 1492, $s = 14$ et $a = 92$.
En fait, m doit être modifié selon le tableau suivant :

Mois	Jan.	Fév.	Mars	Avril	Mai	Juin	Juil.	Août	Sep.	Oct.	Nov.	Déc.
Rang	13*	14*	3	4	5	6	7	8	9	10	11	12

Les mois marqués d'une astérisque comptent pour l'année précédente.
Le nom du jour (0 correspondant à samedi) est alors déterminé en prenant le reste (positif !) dans la division par 7 de nj donné par la formule :

$$nj = j + a + \text{quo}(a,4) + \text{quo}(26(m+1),10) + \text{quo}(s,4) - 2s$$

Ceci est valable à partir du 15 octobre 1582, date à laquelle le pape Grégoire XIII a modifié le calendrier : on est passé du jeudi 4 octobre 1582 au vendredi 15 octobre 1582 (en Grande-Bretagne il a fallu attendre 1752, au Japon en 1873, en Russie 1918, en Grèce 1924).
L'année est alors de 365 jours, sauf quand elle est bissextile, i.e., divisible par 4, sauf les années séculaires (divisibles par 100), qui ne sont bissextiles que si divisibles par 400.
Avant le changement, il faut remplacer $\text{quo}(s,4) - 2s$ par $5 - s$.
Pour les démonstrations, voir l'article original de ZELLER disponible (en allemand...) à l'adresse :
http ://www.merlyn.demon.co.uk/zel-86px.htm
Quel jour est né le petit Jésus ? Quel jour a été prise la Bastille ?

Exercice 1 - 10 Critères de divisibilité

En travaillant modulo le bon entier, démontrer les critères de divisibilité usuels par 2, 3, 4, 5, 9, 10 et 11.
On rappelle que l'écriture en base 10 d'un nombre n est définie par la donnée de l'unique famille $(a_0, a_1, ..., a_k)$ vérifiant :

$$n = a_k 10^k + a_{k-1} 10^{k-1} + \cdots + a_2 10^2 + a_1 10 + a_0$$

avec $a_k \neq 0$ et $0 \leqslant a_1 < 10$ pour tout $i \in \{0, 1, .., k\}$.

Exercice 1 - 11 Nombre de diviseurs

Déterminer une fonction récursive qui calcule le nombre de diviseurs positifs d'un entier naturel donné. On pourra commencer par créer une fonction :

nb_div_naif_loc nb_a_diviser candidat

qui calcule le nombre de diviseurs de **nb_a_diviser** supérieurs à **candidat**.
On pourra ensuite tenter d'améliorer la fonction en remarquant que les diviseurs vont souvent deux par deux...

Exercice 1 - 12 Liste des diviseurs

Déterminer une fonction récursive qui calcule la liste des diviseurs positifs d'un entier naturel donné.

*

Exercice 1 - 13 Nombres parfaits

Un nombre parfait est un nombre entier n strictement supérieur à 1 qui est égal à la somme de ses diviseurs (sauf n bien sûr) ou, de manière équivalente, c'est un nombre tel que son double est égal à la somme de ses diviseurs, lui-même compris.

On pourra créer une fonction **`som_div n`** qui calcule la somme des diviseurs de n.
Il faut ensuite créer un test de « perfection » **`parfait n`**.
On terminera par dresser la liste des entiers parfaits inférieurs à 100 000.
On connaît une quarantaine de nombres parfaits actuellement, tous pairs. On ne sait toujours pas s'il en existe des impairs.
Cependant, EUCLIDE avait déjà découvert la propriété suivante :
« *Lorsque la somme d'une suite de nombres doubles les uns des autres est un nombre premier, il suffit de multiplier ce nombre par le dernier terme de cette somme pour obtenir un nombre parfait.* »
Qu'en pensez-vous ? Quel est la rapport avec $2^p - 1$ et $2^{p-1}(2^p - 1)$? Écrivez les nombres parfaits obtenus en binaire : joli, non ?

5 3 PGCD

Exercice 1 - 14 PGCD

Programmez sur Caml l'algorithme récursif d'EUCLIDE I.

Exercice 1 - 15 Nombres premiers entre eux

Déterminez une fonction qui teste si deux entiers sont premiers entre eux.

Exercice 1 - 16 Bézout

Programmez sur Caml l'algorithme récursif d'EUCLIDE II qui calcule des coefficients de BÉZOUT et le PGCD de deux nombres.

Exercice 1 - 17 Inverse modulaire

Déterminez une fonction qui calcule l'inverse d'un entier a modulo un entier n. Vous ferez attention aux exceptions.

5 4 Nombres premiers

Exercice 1 - 18 Test naïf

Programmez le test naïf de primalité vu en cours. Modifiez votre programme pour réduire le nombre de tests effectués sachant qu'un entier pair ou multiple de 3 autre que 2 et 3 n'est pas premier...

Exercice 1 - 19 Décomposition

Créez une fonction **`decompo n`** qui renvoie la liste des diviseurs premiers de n.

5 5 Complexité et relation de domination

Voici un autre standard du vocabulaire geek :

> *Brooks's Law [prov.]*
> *«Adding manpower to a late software project makes it later» – a result of the fact that the expected advantage from splitting work among N programmers is* O(N)*, but the complexity and communications cost associated with coordinating and then merging their work is* O(N^2)
>
> in « The New Hacker's Dictionnary »
> http ://outpost9.com/reference/jargon/jargon_17.html#SEC24

Les notations de LANDAU(1877-1938) ont en fait été crées par Paul BACHMANN(1837-1920) en 1894, mais bon, ce sont tous deux des mathématiciens allemands.
Par exemple, si l'on considère l'expression :

$$f(n) = n + 1 + \frac{1}{n} + \frac{75}{n^2} - \frac{765}{n^3} + \frac{\cos(12)}{n^{37}} - \frac{\sqrt{765\,481}}{n^{412}}$$

Quand n est « grand », disons 10 000, alors on obtient :

$$f(10\,000) = 10\,000 + 1 + 0,0001 + 0,00000000075 - 0,0000000000000765 + \text{ peanuts}$$

Tous les termes après n comptent pour du beurre quand n est « grand ». Donnons une définition pour plus de clarté :

Définition 1 - 16

« Grand O »
Soit f et g deux fonctions de $\mathbb{N}$ dans $\mathbb{R}$. On dit que f est un « grand O » de g et on note $f = O(g)$ ou $f(n) = O(g(n))$ si, et seulement si, il existe une constante strictement positive C telle que $|f(n)| \leqslant C|g(n)|$ pour tout $n \in \mathbb{N}$.

Dans l'exemple précédent, $\frac{1}{n} \leqslant \frac{1}{1} \times 1$ pour tout entier n supérieur à 1 donc $\frac{1}{n} = O(1)$.
De même, $\frac{75}{n^2} \leqslant \frac{75}{1^2} \times 1$ donc $\frac{75}{n^2} = O(1)$ mais on peut dire mieux : $\frac{75}{n^2} \leqslant \frac{75}{1} \times \frac{1}{n}$ et ainsi on prouve que $\frac{75}{n^2} = O\left(\frac{1}{n}\right)$.
En fait, un grand O de g est une fonction qui est au maximum majorée par un multiple de g.
Point de vue algorithmique, cela nous donne un renseignement sur la complexité au pire.
On peut cependant faire mieux si l'on a aussi une minoration.
C'est le moment d'introduire une nouvelle définition :

Définition 1 - 17

« Grand Oméga »
Soit f et g deux fonctions de $\mathbb{R}$ dans lui-même. On dit que f est un « grand Oméga » de g et on note $f = \Omega(g)$ ou $f(n) = \Omega(g(n))$ si, et seulement si, il existe une constante strictement positive C telle que $|f(n)| \geqslant C|g(n)|$ pour tout $n \in \mathbb{N}^*$.

Remarque

Comme Ω est une lettre grecque, on peut, par esprit d'unification, parler de « grand omicron » au lieu de « grand O »...

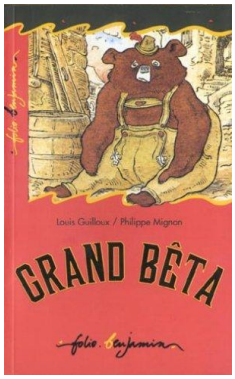
Remarque

$f = \Omega(g) \iff g = O(f)$...

Si l'on a à la fois une minoration et une majoration, c'est encore plus précis et nous incite à introduire une nouvelle définition :

Définition 1 - 18

« Grand Théta »
$f = \Theta(g) \iff f = O(g) \ \wedge \ f = \Omega(g)$



Voici maintenant une petite table pour illustrer les différentes classes de complexité rencontrées habituellement :

coût \ n	100	1000	10^6	10^9
$\log_2(n)$	≈ 7	≈ 10	≈ 20	≈ 30
$n\log_2(n)$	≈ 665	≈ 10000	$\approx 2 \cdot 10^7$	$\approx 3 \cdot 10^{10}$
n^2	10^4	10^6	10^{12}	10^{18}
n^3	10^6	10^9	10^{18}	10^{27}
2^n	$\approx 10^{30}$	$> 10^{300}$	$> 10^{10^5}$	$> 10^{10^8}$

Gardez en tête que l'âge de l'Univers est environ de 10^{18} secondes...

5 6 Exercices « papier-crayon »

Exercice 1 - 20 Complexité de l'algorithme d'Euclide et nombre d'or...

On suppose dans toute la suite que a et b sont deux entiers naturels tels que $a \geqslant b$.

1. Rappeler le principe de l'algorithme et de sa preuve.
2. Soit $r_n = a \wedge b$ avec les notations habituelles : $r_0 = b$ puis $r_1 = a \bmod b$, $r_2 = r_1 \bmod r_0$ et plus généralement :
$$r_{k+1} = r_k q_k + r_{k-1}, \quad 0 \leqslant r_{k-1} < r_k$$
Montrez que pour tout entier $k \in \{1, 2, ..., n\}$, on a $q_k \geqslant 1$ puis que $q_n \geqslant 2$.
3. Soit $\Phi = \frac{1+\sqrt{5}}{2}$ le fameux nombre d'or. Comparez Φ et $1 + \frac{1}{\Phi}$.
4. Montrez par récurrence que $r_k \geqslant \Phi^{n-k}$ pour tout entier $k \in \{0, 1, ..., n\}$.
5. Déduisez-en que $b \geqslant \Phi^n$ puis que le nombre d'appels récursifs de l'algorithme d'Euclide est au maximum de $\frac{\ln b}{\ln \Phi}$.

Exercice 1 - 21 Nombre de diviseurs

Déterminez le nombre de diviseurs de 2^n.

Exercice 1 - 22 Opérations sur les O

On suppose que f, g, F, G sont des fonctions de $\mathbb{N}$ dans $\mathbb{R}_+$ et que $f = O(F)$ et $g = O(G)$.
Rappelez la définition des O et trouvez φ et ψ telles que $f + g = O(\phi)$ et $fg = O(\psi)$.

Exercice 1 - 23 Bases

Écrire 355 en base 2 et en base 16.

Exercice 1 - 24 Bibinaire

Boby LAPOINTE, célèbre chanteur français, était aussi mathématicien à ses heures. Ayant trouvé le code binaire trop compliqué à utiliser, il inventa le code... bibinaire. Il suffit de remplacer les chiffres par des lettres. On commence par couper le nombre écrit en binaire en paquets de 2. S'il y a un nombre impair de chiffres, on rajoute un zéro à gauche, ce qui ne modifie pas la valeur de nombre. On commence par le premier groupe de deux chiffres le plus à droite. On remplace 00 par O, 01 par A, 10 par E, 11 par I.
Puis on prend le paquet de deux chiffres suivants en se déplaçant de droite à gauche. On remplace 00 par H, 01 par B, 10 par K, 11 par D.
Pour le paquet suivant, on recommence avec les voyelles. S'il y a encore un groupe, on remplace par une consonne, etc.
Écrivez les nombres de 0 à 25 en bibinaire. Pourquoi Boby a-t-il finement choisi le H ?
Écrivez la table de 3 en bibinaire.
Écrivez les nombres de 255 à 257 en bibinaire. Quel est la base du bibinaire ? Écrivez 355 en bibinaire.
Écrivez KEKIDI et HEHOBOBI en base 10.
Pourquoi est-il pratique d'écrire les nombres en base 2 avec un nombre de chiffres multiple de 4 avant de les traduire en bibinaire ?

Exercice 1 - 25 Groupe

Soit $A = \frac{1}{3}\begin{bmatrix} 0 & -2 & -2 \\ 2 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$.

1. Calculer A^2, A^3 et A^4.
2. Montrez que $\{A, A^2, A^3, A^4\}$ est un groupe pour le produit matriciel de $\mathcal{M}_3(\mathbb{R})$.
3. (INFO 2) On appelle $GL_3(\mathbb{R})$ l'ensemble des matrices de taille 3 à coefficients réels inversibles. Est-ce que $(GL_3(\mathbb{R}), \times)$ est un groupe ? Est-ce que le groupe étudié précédemment est un sous-groupe de $(GL_3(\mathbb{R}), \times)$?

Exercice 1 - 26 Puissance

Calculez à la main en un minimum d'opérations les deux derniers chiffres de l'écriture en base 10 de 7^{73} puis de 6^{73}

Exercice 1 - 27 Indicatrice d'Euler

Un vieux théorème publié en 1247 par le mathématicien chinois Qin Jiushao mais utilisé dès le troisième siècle après JC et appelé communément *théorème chinois* en référence au « Sunzi suanjing », permet de montrer un résultat primordial en cryptographie :

Si m et n sont premiers entre eux, alors $\varphi(mn) = \varphi(m)\varphi(n)$.

Nous admettrons ce résultat.

1. Soit p un nombre premier et n un entier naturel non nul.
   Prouvez que $\varphi\left(p^n\right) = p^n - p^{n-1}$ puis que, pour tout entier naturel non nul k,
   $$\varphi(k) = k\prod_{p|k}\left(1 - \frac{1}{p}\right)$$
2. Calculez $\varphi(200)$ et $\varphi(1000)$.
   Trouvez, DE TÊTE, les trois derniers chiffres de :
   $$789587546378637837837834545364653897897^{208582714591458551455135135135147454098258258001}$$
3. Comment utiliser le théorème d'EULER pour calculer plus simplement $m^e \pmod n$ lorsque m et n sont premiers entre eux ?
4. Cela est très important pour le RSA. Soit un couple (p, q) d'entiers premiers. On note $n = pq$ leur produit. Soit m un nombre plus petit que p et q.
   Comment calculer plus simplement $m^e \pmod n$? Quelle condition doit remplir e pour qu'il existe un entier d tel que $(m^e)^d \equiv m \pmod n$?

Exercice 1 - 28 Lemme de Gauß

Démontrez le lemme suivant :

Théorème 1 - 22

Lemme de Gauß

$$\forall (a,b,c) \in \mathbb{Z}^3, (a|bc \wedge \text{PGCD}(a,b) = 1) \Rightarrow a|c$$

Exercice 1 - 29 Restes chinois

Voici une ré-écriture du théorème chinois :

Théorème 1 - 23

Théorème du reste chinois
Soit $m_1, m_2, ..., m_n$ des entiers positifs premiers entre eux deux à deux. Le système :

$$\begin{array}{rcl} x & \equiv & a_1[x_1] \\ x & \equiv & a_2[x_2] \\ \vdots & \vdots & \vdots \\ x & \equiv & a_n[x_n] \end{array}$$

admet une unique solution unique x modulo $m = m_1 \times m_2 \times \cdots \times m_n$.

Occupons-nous de démontrer qu'il existe bien une solution x du problème modulo m. Il restera à démontrer son unicité.
Considérons les entiers $M_k = m/m_k$: montrez qu'il existe un entier y_k tel que $M_k y_k \equiv 1[m_k]$.
Montrez ensuite que $x = a_1 M_1 y_1 + \cdots + a_n M_n y_n$ est une solution du problème.

Exercice 1 - 30 Pirates chinois

15 pirates chinois se partagent un butin constitué de pièces d'or. Mais une fois le partage (équitable) effectué, il reste 3 pièces. Que va-t-on en faire ? La discussion s'anime. Bilan : 8 morts. Les 7 survivants recommencent le partage, et il reste cette fois-ci 2 pièces ! Nouvelle bagarre à l'issue de laquelle il ne reste que 4 pirates. Heureusement, ils peuvent cette fois-ci se partager les pièces sans qu'il n'en reste aucune. Sachant que 32 Tsing-Tao (bière chinoise) coûtent une pièce d'or, combien (au minimum) de Tsing-Tao pourra boire chaque survivant ?

Exercice 1 - 31 Caml chinois

Créez une fonction **`chinois`** de signature :

```
# chinois;;
- : classe list -> classe = <fun>
```

qui résout le problème des restes chinois.
Par exemple :

```
# chinois [3%15 ; 2%7 ; 0%4];;
- : classe = {representant = 408; modulo = 420}
```

Exercice 1 - 32 Opération sur les grands nombres version 1

Soit $(x_1, x_2, ..., x_n)$ une liste de n nombres entiers premiers entiers deux à deux et soit m leur produit.
Démontrez que tout entier a positif inférieur à m peut être représenté de manière unique par un n-uplet :

```
[a%m_1 ; a%m_2 ; ... ; a%m_n]
```

avec les notations Caml habituelles.
Soit $b = [99;98;97;95]$. Décomposez 123684 et 413456 dans cette « base » puis effectuez la somme des quadruplets obtenus. Que vous inspire cette méthode ?
Proposez des fonctions Caml effectuant ces calculs automatiquement.

Exercice 1 - 33 Ordre d'un élément dans un groupe fini

1. Comment justifier que pour tout élément g d'un groupe fini G, 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 $2^{120000000000300000000060000000200004000020} \pmod 7$.

Exercice 1 - 34 Permutations

Définition 1 - 19

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 - 20

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

Théorème 1 - 24

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 - 25

Nombre de permutations
Le groupe $\mathfrak{S}_n$ est d'ordre $n!$.

1. Démontrez le théorème 1 - 24.
2. Démontrez le théorème 1 - 25. 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 - 35 Chiffrement par blocs

Commençons par une définition :

Définition 1 - 21

Cryptosystème (ou système de chiffrement ou chiffre)
Un cryptosystème est un quintuplet $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ tel que :

- L'ensemble $\mathcal{P}$ est l'espace des messages en clair (*plaintext* en anglais) ;
- L'ensemble $\mathcal{C}$ est l'espace des messages chiffrés (*cyphertext* en anglais) ;
- L'ensemble $\mathcal{K}$ est l'espace des clés (*key*) ;
- $\mathcal{E}$ est la famille des fonctions de chiffrement, qui vont de $\mathcal{P}$ dans $\mathcal{C}$;
- $\mathcal{D}$ est la famille des fonctions de déchiffrement, qui vont de $\mathcal{C}$ dans $\mathcal{P}$;

Pour chaque élément $e \in \mathcal{K}$, il existe un élément $d \in \mathcal{K}$ tel que, pour tout message clair m de $\mathcal{P}$, il existe $D_d \in \mathcal{D}$ et $E_e \in \mathcal{E}$ telles que $D_d(E_e(m)) = m$. Les fonctions D_d et E_e sont des applications injectives. Il est entendu que d doit rester secret...

1. On travaille sur des mots formés des 26 minuscules non accentuées de notre alphabet latin et d'une espace qui sont modélisés par les entiers de $E = \{0,1,2,...,26\}$.
   Dans ce contexte, on remplace une « lettre » x par $kx \pmod{27}$ avec $k \in E$. Définit-on ainsi un cryptosystème ?
2. Un cryptosystème est un *chiffrement par blocs* si $\mathcal{P} = \mathcal{C} = A_n$ avec A_n l'ensemble des mots de l'alphabet A de longueur n. Est-ce que le chiffrement ROT-13 est un chiffrement par blocs ?
3. Pourquoi les fonctions de chiffrement d'un chiffrement par blocs sont des permutations ?

Depuis 2001, le chiffrement par blocs « officiel » est l'AES qui opère par blocs de 128 bits. Une présentation de l'algorithme de chiffrement associé, RIJNDAEL, et de sa résistance aux attaques est présentée ici : http ://www.cryptis.fr/assets/files/Canteaut-25ans-Cryptis.pdf.

Exercice 1 - 36 Mode ECB

ECB : *Electronic CodeBook*. C'est le plus simple...et le plus vulnérable des modes de chiffrement par blocs.
Voyons sur un exemple. On considère une chaîne de bits quelconque que l'on découpe en blocs de longueur fixe, par exemple 7 (plus pratique pour ensuite utiliser l'ASCII : pourquoi ?). On rajoute éventuellement des zéros en bout de chaîne.
Considérons $m = 1001001111011$. On rajoute un 0 en bout de chaîne :

$$m' = 1001001\ 1110110 = \beta_1\beta_2$$

avec $\beta_1 = 1001001$ et $\beta_2 = 1110110$.

On choisit une clé de chiffrement dans $\mathfrak{S}_7$ car $\mathcal{K} = \mathfrak{S}_7$: $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 1 & 4 & 2 & 6 & 5 & 7 \end{pmatrix}$

Alors $E_\pi(\beta_1) = 0110001$ et $E_\pi(\beta_2) = 1101110$.

1. Quelle est la principale faiblesse du mode ECB ?
2. Chiffrez « maman » avec ce cryptosystème en transformant la chaine de lettres en chaînes de bits constituée des codes ASCII de ses caractères. Le tableau ASCII standard (cf tableau page 34) est-il satisfaisant ? Comment y remédier ?
3. Quelle est la fonction de déchiffrement associée à l'exemple précédent ?
4. Déchiffrer 10110111001111 sachant que la clé de chiffrement est $k = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 1 & 7 & 6 & 5 & 4 \end{pmatrix}$
5. Déchiffrez « papa » sachant que c'est le cryptosystème de la question 2. qui a été employé.

Exercice 1 - 37 Mode CBC

CBC : *Cipher-Block Chaining*. Ce mode a été inventé par IBM en 1976. Pour éviter la faiblesse de l'ECB, les blocs sont maintenant chaînés : chaque bloc en clair est « XORé » ou « OUEXé » avec le bloc crypté précédent avant d'être crypté lui-même.
Pour le premier bloc, on utilise un *vecteur d'initialisation* public. Avec les notations habituelles, on a donc

$$c_i = E_k(m_i \oplus c_{i-1}), \quad c_0 = \overrightarrow{v_0}$$

1. Quel est le lien entre OUEX et $\mathbb{F}_2$?
2. Cryptez l' exemple traité avec ECB en prenant $\overrightarrow{v_0} = 1010101$.
3. Faites de même avec « Maman ».

4. Donnez une formule de déchiffrement. Déchiffrer le cryptogramme 101101110011111 sachant que la clé est $k = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$ et que $\overrightarrow{v_0} = 001$.
5. Déchiffrez « Papa » sachant que c'est le cryptosystème de la question 2. qui a été employé.

Ce mode est efficace mais nécessite l'utilisation de fonctions de chiffrement et de déchiffrement différentes ce qui peut ralentir la procédure.

Exercice 1 - 38 Mode CFB

CFB : *Cipher-FeedBack*. C'est un mode dérivé de CFB qui est utilisé par OpenPGP, format pour l'échange sécurisé de données (paiements sécurisés par exemple) largement utilisé actuellement mais est susceptible d'être attaqué, même s'il est très difficile de mettre en œuvre concrètement cette attaque :

http ://www.cert-ist.com/fra/ressources/Publications_ArticlesBulletins/Autres/FailledanslechiffrementCFBdOpenPGP/

Le mode CFB utilise un registre de décalage de taille r inférieure à celle de la clé.
On a besoin d'un vecteur d'initialisation $\overrightarrow{v_0}$. Le message clair est découpé en blocs de longueur r. Ensuite on procède comme suit, sachant que les m_j sont les blocs en clair de longueur r :

- $t_0 = \overrightarrow{v_0}$;
- $s_j = \mathrm{E}_k(t_j)$;
- g_j est la chaîne constituée des r bits les plus à gauche de s_j. Quelle est l'opération arithmétique associée ?
- $c_j = m_j \oplus g_j$;
- $t_j = (2^r t_{j-1} + c_{j-1}) \bmod 2^n$ (attention ! Il faut travailler en base 2).

Le déchiffrement fonctionne de manière identique en échangeant les rôles de m_j et c_j à la quatrième étape : démontrez-le !

1. Chiffrez « *Papa is cool* » sachant que la clé est $k = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 3 & 2 \end{pmatrix}$, que $\overrightarrow{v_0} = 0101$ et que $r = 3$.
2. Déchiffrer le cryptogramme 101101110011111 sachant que la clé est $k = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$, que $r = 2$ et que $\overrightarrow{v_0} = 001$.

5 7 Exponentiation rapide

Exercice 1 - 39 Complexité

Voici deux algorithmes :

```
Fonction algo1(a,n : entiers naturels) : entier naturel
Début
    Si n=0 Alors
        Retourner 1
    Sinon
        Retourner a*algo1(a,n-1)
    FinSi
Fin
```

```
Fonction algo2(a,n : entiers naturels) : entier naturel
Variable
    i,r : entiers
Début
    r ← 1
    Pour i variantDe 1 Jusque n Faire
        r ← a*r
    FinPour
    Retourner r
Fin
```

Que calculent-ils ? Quelle est leur complexité ?

En voici un troisième :

```
Fonction algo3(a,n : entiers naturels) : entier naturel
Début
    Si n=0 Alors
        Retourner 1
    Sinon
        Si n=1 Alors
            Retourner a
        Sinon
            Si n est pair Alors
                Retourner algo3(a*a,n/2)
            Sinon
                Retourner a*algo3(a*a,(n-1)/2)
            FinSi
        FinSi
    FinSi
Fin
```

Que calcule-t-il ? Quelle est sa complexité ?
Comparez le temps de calcul de algo3(1 000 000 000) et de algo1(1 000 000 000) sur un ordinateur équipé d'un processeur 3GHz effectuant une opération arithmétique élémentaire en 100 cycles.
Programmer **`algo3`** sur CAML. On l'appellera **`prap a n`**.
Calculer 2^{10}.
On peut améliorer l'efficacité du programme en essayant que le dernier appel de la fonction soit la dernière instruction à être évaluée.
Or ici, on a **`a*algo3(a*a,(n-1)/2)`** : la dernière instruction est une composition par la fonction addition qui fait intervenir l'appel récursif. Il va donc falloir empiler puis dépiler les résultats intermédiaires. On peut s'en passer en rajoutant un accumulateur qui contiendra le résultat intermédiaire. Caml ensuite pourra très simplement (en interne) transformer cette récursion terminale en itération et gagner en efficacité.

Essayez maintenant de calculer 2^{29}, 2^{30}, $2^{30}-1$, $2^{30}+1$.
Que donne :

```
# max_int;;
# 1 + max_int;;
```

Arghhhh... Trois solutions : étudiez le calcul polynomial, ce que nous ferons au prochain chapitre, ou bien utiliser le module **`BigInt`** de Caml pour calculer en multiprécision, ou bien utiliser un langage comme SAGE qui « masque » le problème informatique.
Nous sommes en formation initiale d'informatique donc nous ne devons rien masquer...
Pour info, avec **`Big_int`**, la syntaxe est un peu lourde...

```
# Big_int.string_of_big_int(Big_int.power_int_positive_int 2 100);;
- : string = "1267650600228229401496703205376"
```

Voici un moyen d'y remédier : on va redéfinir les opérations arithmétiques et les tests pour travailler avec des grands entiers.

```
#load "nums.cma";;
open Big_int;;

(* On définit des opérateurs infixes.
   Pour les distinguer, on les fait précéder d'une esperluette.
   On écrira a &+ b par exemple pour la somme des grands entiers a et b *)

let (&+) a b = Big_int.add_big_int a b;;
let (&*) a b = Big_int.mult_big_int a b;;
let (&%) a n = Big_int.mod_big_int a n;;
let (&/) a b = Big_int.div_big_int a b;;
let (&-) a b = Big_int.sub_big_int a b;;
let (&=) a b  =  Big_int.eq_big_int a b ;;

(* transforme un petit entier en grand entier *)
```

```
let big(n) =
  Big_int.big_int_of_int n;;

(* les petits entiers basiques *)

let zero = zero_big_int;;
let un = unit_big_int ;;
let deux = big(2) ;;

(* pour afficher un grand entier sous forme de chaîne *)

let montre(b) =
  Big_int.string_of_big_int b;;

(* Le pgcd pour plus tard  *)

let bgcd a b =
  Big_int.gcd_big_int a b ;;
```

Réécrire **prap** en **bprap** pour pouvoir travailler avec des grands entiers.
Attention ! On ne pourra plus utiliser le filtrage par motif : il faudra revenir à des conditionnelles **if**...**then**...**else**
Ensuite, on définira un opérateur infixe plus pratique :

```
let (&^) a n = bprap(a,n);;
```

Par exemple, on obtiendra 2^{512} avec :

```
# deux &^ big(512) ;;
- : Big_int.big_int = <abstr>
# montre(deux &^ big(512)) ;;
- : string =
"13407807929942597099574024998205846127479365820592393377723561443
72176403007354697680187429816690342769003185818648605085375388281 1
94656994643364900608409 6"
```

5 8 Petit théorème de Fermat et test de primalité

Exercice 1 - 40 Puissance modulaire et nombres pseudo-premiers

Pour nos calculs en cryptographie, nous aurons besoin de calculer $a^{p-1} \pmod p$ avec a ou p des « grands » nombres entiers.
Par exemple, que vaut $97^{p-1} \pmod p$ avec $p = 2^{500} - 1$?...
Cela risque de prendre pas mal de temps si nous calculons cette puissance naïvement.
On va donc redéfinir nos opérations arithmétiques mais cette fois modulo un entier p. Choisissons par exemple de créer des opérateurs infixes précédés cette fois de deux esperluettes pour les distinguer des précédents.
Pour gagner du temps, on peut créer une fonction polymorphe qui redéfinit l'opération op modulo p :

```
(* fonction polymorphe définissant une opération op modulo p*)
let modop = fun a b p op ->
  ((op) (a &% p)  (b &% p)) &% p;;

(* Version infixe de la multiplication modulaire
    a*b mod p  est obtenu par a &&* (b,p) *)
let (&&*) a (b,p) =
  modop a b p (&*);;
```

Déterminez ensuite une fonction **mprap a m p** qui calcule efficacement $a^m \pmod p$. Elle sera du type :

```
val mprap:Big_int.big_int*Big_int.big_int*Big_int.big_int -> Big_int.big_int = <fun>
```

Il ne restera plus qu'à créer l'opérateur infixe afférent :

```
let (&&^) a (n,p) = mprap a n p;;
```

Calculez maintenant $97^{p-1} \pmod p$ avec $p = 2^{500} - 1$.
Cela ne fait pas 1 ! Donc p n'est pas premier (pourquoi ?) et la réponse a été donnée immédiatement ! C'est quand même assez fort : on arrive, grâce au *petit* théorème de FERMAT, à démontrer qu'un entier de 113 chiffres (presque...) pris au hasard n'est pas premier.
Maintenant, que se passe-t-il si le résultat est effectivement 1 ? Cela fait-il de n un nombre premier ?
Par exemple, calculez $2^{341-1} \pmod{341}$. Est-ce que 341 est premier ? Comment expliquer ce résultat ?
On dit que 341 est **pseudo-premier** de base 2.
Créez une fonction **`pseudo(a,n)`** qui renvoie les entiers pseudo-premiers de base a inférieurs à n. On travaillera cette fois avec des petits entiers pour avoir à disposition la liste `crible(n)` des entiers premiers inférieurs à n.
On créera au préalable une fonction `appartient : 'a * 'a list -> bool` qui teste si une variable appartient à une liste d'éléments de même type.
Par exemple :

```
# pseudo(2,10000);;
- : int list =
[8911; 8481; 8321; 7957; 6601; 5461; 4681; 4371; 4369; 4033; 3277; 2821;
 2701; 2465; 2047; 1905; 1729; 1387; 1105; 645; 561; 341]
```

Testez **`pseudo(3,10000)`**, **`pseudo(5,10000)`**, etc.
Pour des nombres plus grands, il n'est pas facile de savoir si ceux-ci sont réellement premiers. Puisqu'on ne sait pas a priori s'ils le sont, on les appelle des nombres probablement premiers de base a. Certains sont probablement premiers d'une seule base. Par exemple :

```
# montre(mprap(2,big(340),big(341)));;
- : string = "1"
# montre(mprap(3,big(340),big(341)));;
- : string = "56"
```

Ainsi, nous pouvons accumuler les bases pour éliminer les candidats. Malheureusement, il existe encore une infinité de nombres qui sont probablement premiers dans toutes les bases, tout en étant composés :

Définition 1 - 22

Nombre de CARMICHAEL
Un entier composé n est un **nombre de CARMICHAEL** si $a^{n-1} \equiv 1 \pmod n$ pour tout entier a premier avec n

Robert CARMICHAEL (1879 - 1967)

Pour se donner un ordre de grandeur, il y a un peu plus d'un million de nombres premiers inférieurs à 25 milliards, mais seulement 21 853 pseudo-premiers de base quelconque et parmi eux 2 163 nombres de Carmichael. C'est peu, mais c'est toujours trop !
Le plus petit nombre de CARMICHAEL est 561. En 1994, ALFORT, GRANDVILLE et POMERANCE ont démontré qu'il y avait une infinité de nombres de CARMICHAEL.
Il a de plus été démontré qu'un nombre de CARMICHAEL n est le produit d'au moins trois facteurs premiers impairs p_i distincts et pour chacun des p_i, $n-1$ est divisible par $p_i - 1$.
Il nous faut des outils plus performants donc il nous faut creuser un peu plus profondément dans la mine mathématique...

5 9 Grands nombres premiers

En cryptographie à clé publique, on a besoin de très grands nombres premiers car on travaille avec des nombres de l'ordre de 1024 bits voire plus.
On adopte alors une approche probabiliste comme nous l'avons évoqué avec le test de FERMAT et les nombres *probablement premiers*.

Exercice 1 - 41 Algorithme de Rabin-Miller

Michael RABIN (1931)

Gary MILLER est professeur à l'université Carnegie Mellon de Pittsburgh. Il mit au point un premier test de primalité de complexité polynomiale en 1975 pendant qu'il était au MIT Cependant, ce test dépendait d'une hypothèse non encore démontrée (la fameuse hypothèse de RIEMANN dont la démonstration est mise à prix un million de dollars...). En visite au MIT au même moment, le prix TURING israélien Michael RABIN s'inspira du test de MILLER en évitant le recours à l'hypothèse de RIEMANN mais en le rendant probabiliste.
Nous allons le décortiquer et l'utiliser avec Ocaml.
Prenons un entier a strictement inférieur à un entier n. Si n est premier, nous avons vu que $a^{n-1} \equiv 1 \pmod n$ d'après le petit théorème de FERMAT mais tester cette opération sur de grands nombres peut être coûteuse.
Voici une idée pour éliminer plus rapidement certains candidats.
Si $n = 2$, on n'a pas besoin d'allumer un ordinateur. On peut donc considérer que n est impair donc que $n-1$ est ....

Recherche

1. Déterminez une fonction qui calcule s et t tel que $n-1=2^t s$.
2. Résolvez l'équation $x^2 \equiv 1 (\text{mod } n)$. Que se passe-t-il si n est premier ?
3. Montrez que si n est premier et impair, alors :
$$\begin{cases} a^s \equiv 1 (\text{mod } n) \\ \text{ou} \\ \exists u \in \{0,1,..,t-2\} \mid a^{2^u s} \equiv n-1 (\text{mod } n) \end{cases}$$
4. Comment en déduire un test de non-primalité sur n ?

Si n ne passe pas le test pour un certain a, on dit que a est un *témoin fort* pour la composition de n. Sinon, il est *fortement probablement premier*. Si n est fortement probablement premier tout en étant composé, on dit que a est un *menteur fort* et que n est *pseudo-premier fort*.

On peut montrer que pour chaque entier impair n composé, au moins 75% des bases a sont des témoins forts pour la composition de n.
Appelons E_a l'évènement « n est pseudo-premier fort pour une base a donnée ».
On choisit « au hasard » un nombre n et une base a.

Recherche

1. Majorez $\mathbb{P}(E_a)$.
2. On suppose qu'on choisit aléatoirement les bases a dans $\{0,1,...,n-1\}$ et que les évènement E_a et $E_{a'}$ sont **indépendants** pour tout couple (a,a') de bases. Que peut-on en déduire sur $\mathbb{P}(E_a \cap E_{a'})$?
3. Combien de bases va-t-on tester au maximum pour avoir une probabilité d'obtenir un nombre pseudo-premier fort inférieure à 2^{-128} ? (soit environ la probabilité de gagner cinq fois de suite le gros lot de l'euromillions en jouant une grille à chaque fois...)

Voici maintenant une fonction qui renvoie un grand entier aléatoire dans $[0,2^n[$:

```
exception Taille_negative;;

(* nombres aléatoires (d'après le module numerix de Michel Quercia) *)
(* land est le ET logique bit à bit : plus rapide que mod
   1 lsl p effectue un décalage de 1 bit : plus rapide que 2^p
   0xn est l'entier écrit n en base 16
   On travaille donc par paquet de 16^4
   La fonction calcule un nombre < 2^n et >= 2*)
let nrandom n =
  if n < 0 then raise Taille_negative
  else if n = 0 then big(0)
  else begin
    let p = if n land 15 = 0 then 16 else n land 15 in
    let mask = (1 lsl p) - 1 in
    let r = ref(big(((Random.int(0xfffe)) land (mask-2))+2)) in
    for i=1 to (n-p)/16 do
        r := add_int_big_int (Random.int(0x10000)) (mult_int_big_int 0x10000 !r)
    done;
    !r
  end
;;
```

Par exemple, on obtient un entier aléatoirement choisi inférieur à 2^{1024} :

```
# string_of_big_int (nrandom 1024);;
- : string =
"80656880405872425890276287356861322138632224126908132719794577622082089
57064839103288682721613147665208083709104463328596002726097261810537644 5
74791519158215952095887744683773234023226717158909447189732398803032360 1
72216871206672622305659922397261056701414963271372950110565303778531679 8
927182986135540715685"
```

Voici quelques fonctions mystérieuses :

```
let fonction_1(n) =
  let rec loop(s,t) =
```

```
    if (s &% deux) &= zero then
      loop( s &/ deux , t &+ un )
    else
      [|s;t|]
  in loop(n &- un,zero);;

let fonction_2(v,n,t) =
  if v &= un then
    true
  else
    let rec loop(v,u) =
      if v &= (n &- un)  then
        true
      else
        if u &= (t &- un) then
          false
        else
          loop( v &&^ (deux,n) , u &+ un )
    in loop(v,zero);;

let fonction_3(n) =
  int_of_float(log(float_of_big_int(n))/.log(2.));;
```

Recherche

Analysez les fonctions précédentes et utilisez-les dans une fonction `rabin_miller(n)` qui renverra `true` si le nombre est fortement probablement premier et FALSE sinon.
Une remarque : il existe une valeur maximale des nombres de type `float` donnée par `max_float` au-delà de laquelle toute variable de ce type est considérée comme ∞.
En quoi cela peut-il limiter notre calcul ? Comment y remédier ?
Testez avec **`rabin_miller((deux &^ big(607)) &- un)`**

Exercice 1 - 42 Test de primalité pour grands entiers

Créez une fonction `est_premier` qui vérifie si un grand entier est « fpp » à l'aide du test de RABIN-MILLER mais en commençant tout d'abord par tester s'il est divisible par des petits nombres premiers inférieurs à 1000.

Exercice 1 - 43 Génération de grands nombres premiers.

Nous avons besoin de déterminer une borne supérieure de la probabilité qu'un nombre testé et déclaré premier soit en fait composé.
Par exemple, on peut se demander quelle est la probabilité qu'un nombre n de 2001 bits de long soit premier. Il est compris entre 2^{2000} et 2^{2001} : il y a donc 2^{2000} tels nombres. Si l'on arrive à estimer $\pi(n)$, le nombre de nombres premiers inférieurs à n, nous allons voir que cela nous donnera une estimation de la probabilité de tomber sur un nombre premier en choisissant un nombre au hasard dans l'intervalle $[2^{2000}, 2^{2001}[$.

Recherche

On notera $\pi(n)$ le nombre de nombres premiers inférieurs à n. Comment le calculer avec nos fonctions précédemment créées ?

Existerait-il une loi régissant la fonction π ?
À la fin du XVIII^{e} siècle, GAUSS conjecture que, lorsque x tend vers l'infini, $\pi(x)$ est équivalent à $\mathrm{Li}(x) = \int_2^x \frac{\mathrm{d}t}{\ln t}$. À la même époque, LEGENDRE pense que $\pi(x)$ est équivalent à $\frac{x}{\ln x}$.
Notez bien que dire que $a(x)$ et $b(x)$ sont équivalents au voisinage de l'infini signifie, sous certaines conditions, que $\lim_{x\to+\infty} \frac{a(x)}{b(x)} = 1$.
Mais attention ! Cela ne veut pas dire que $a(x) - b(x)$ est petit.
Par exemple

$$\lim_{x\to+\infty} \frac{x^2+x}{x^2} = 1$$

mais leur différence $x^2 + x - x^2 = x$ tend vers l'infini.
Cela veut juste dire que $\ln x/x$ et $\mathrm{Li}(x)$ seraient des approximations de $\pi(x)$ au voisinage de $+\infty$. En fait, on peut employer n'importe quelle expression du type $\frac{x}{\ln x - a}$ avec a une constante arbitraire et on obtient le même résultat.

Le tableau suivant nous permet de nous faire une idée sur des petits (eh oui) nombres premiers :

x	$\pi(x)$	$\mathrm{Li}(x)$	$\frac{x}{\ln x}$	$\frac{x}{\ln x - 1}$
1 000	168	178	172	169
10 000	1 229	1 246	1 231	1 218
1 000 000	78 498	78 628	78 534	78 030
10 000 000 000	455 052 511	455 055 614	455 743 004	454 011 971

Jacques HADAMARD
(1865 - 1963)

Le problème, c'est que ces conjectures n'avaient pas été prouvées par leurs auteurs. Cinquante ans plus tard, le russe TCHEBYCHEV établit une estimation de $\pi(x)$:

$$(c_1 + \varepsilon_1(x))\frac{x}{\ln x} \leqslant \pi(x) \leqslant \frac{6}{5}(c_1 + \varepsilon_2(x))\frac{x}{\ln x}$$

avec $c_1 = \ln\left(\frac{\sqrt{2}\times\sqrt[3]{3}\times\sqrt[5]{5}}{\sqrt[30]{30}}\right) \approx 0,92129$ et $\lim\limits_{x\to+\infty}\varepsilon_1(x) = \lim\limits_{x\to+\infty}\varepsilon_2(x) = 0$

Il a fallu attendre 1896 pour que HADAMARD et LA VALLÉE POUSSIN prouvent, suite à d'importants travaux de RIEMANN, et indépendamment l'un de l'autre, le fameux théorème des nombres premiers.
Il fut amélioré en 1901 par VON KOCH sous la forme

$$\left|\pi(x) - \int_2^x \frac{dt}{\ln t}\right| \leqslant c\sqrt{x}\ln x$$

avec c une constante positive mais en supposant que la conjecture de RIEMANN (mise à prix un million de dollars !) soit vraie...
Ce théorème permet en particulier de monter que le n^e nombre premier p_n est « de l'ordre de » $n\ln(n)$ quand n est suffisamment grand.

À retenir

En tant qu'informaticien, il sera utile de retenir que, lorsque n est grand, $\pi(n) \approx \frac{n}{\ln(n)}$.

Recherche

Revenons à notre problème probabiliste. Il y a $\pi(n)$ nombres premiers entre 1 et n. Pour de grandes valeurs de n, par exemple dans l'intervalle $[2^{2000}, 2^{2001}[$, quelle est alors la densité approximative de nombre premiers ? Généralisez. Est-il licite de prendre comme approximation de la densité $\frac{1}{\ln(2^n)}$?

Voici une nouvelle fonction qui renvoie un nombre aléatoirement choisi dans l'intervalle $[2^{n-1}, 2^n[$:

```
let nrandom1 n =
  if n < 0 then raise Taille_negative
  else if n = 0 then zero_big_int
  else begin
    let p = if n land 15 = 0 then 16 else n land 15 in
    let mask = 1 lsl (p-1) in
    let r = ref(big_int_of_int((Random.int(0x10000) land (mask-1)) lor mask)) in
    for i=1 to (n-p)/16 do
        r := add_int_big_int (Random.int(0x10000)) (mult_int_big_int 0x10000 !r)
    done;
    !r
  end
;;
```

Recherche

Créez une fonction `genere_premier` qui calcule un nombre premier appartenant à l'intervalle $[2^{n-1}, 2^n[$. En allant souvent à la ligne, huit petites lignes suffiront...

5 10 Cryptographie

Exercice 1 - 44 Ave César

On veut créer une fonction `cesar cle message` qui code un message rentré sous forme d'une chaîne selon la méthode bien connue de César, en se donnant la possibilité de choisir sa clé. On travaillera avec un alphabet de 95 lettres décrit dans le tableau de la présente page.
La fonction **`int_of_char`** renvoie le code ASCII d'un caractère (non accentué :A comme american...).
Il y a 95 caractères qui commencent à 32 et finissent à 126 (les autres codes correspondent à des touches d'action comme le « retour chariot »).
Nous allons donc ajouter la clé au code des caractère modulo 95 mais il faudra avant enlever 32 puis le rajouter à la fin pour travailler avec des caractères et non pas des actions.

À retenir

char et string
Il faut bien distinguer **`'a'`**, **`"a"`** ! Mathématiquement, c'est la distinction que l'on fait entre les éléments de l'alphabet Σ et ceux du langage associé $\Sigma^\star$.

Créez une fonction **`code cle car`** qui satisfait à ces exigences.

```
# code 13 'A';;
- : char = 'N'
```

Danger

mod et rem
Testez avec **`code # (-15)`**. Que se passe-t-il ? Comment fonctionne le **`mod`** d'OCaML ? Comparez avec notre **`rem`**.

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

Enfin, créez la fonction principale récursive **`cesar cle mot`**.
Par exemple, si on code avec la clé 13 :

```
# cesar 13 "Je ne suis pas un numero ! Je suis un homme libre !!!";;
- : string = "Wr-{r-!#v!-}n!-#{-{#zr |-.-Wr-!#v!-#{-u|zzr-yvo r-..."
```

Comment décode-t-on ?

Remarque

La fonction **`tr`** (comme translation) du shell bash peut remplir ce rôle :
`$ echo "Je ne" | tr '\!-~' '\.-~\!-\-'`
`Wr {r`

Exercice 1 - 45 Chiffrement affine

Ce n'est pas trop dur de casser le code de CÉSAR. Nous allons essayer de faire mieux en utilisant nos outils arithmétiques.
On considère toujours les 95 caractères affichables et on les associe aux nombres 0, 1, 2,...,94.
On code ces nombres à l'aide d'une fonction f du type :

$$f : x \mapsto (ax+b) \bmod 95$$

où a et b sont des entiers. On aura remarqué que le chiffrement de César est en fait un cas particulier du chiffrement affine avec $a = 1$.
Si votre César a été bien conçu, il n'y a pas grand chose à modifier pour créer une fonction **chiffre_affine** :
Considérons par exemple $f(x) = (17x + 22) \bmod 95$ et le message « La maîtresse en maillot de bain ».

```
# chiffre_affine 17 22 "La maitresse en maillot de bain";;
- : string = "*r6!r<9vW((W6W26!r<ooC96FW6r<2"
```

Comment décrypter ce message ? On cherche g telle que :

$$\text{transmis} = f(\text{original}) \Leftrightarrow \text{original} g(\text{transmis}) \text{ dans } \mathbb{Z}_{95}$$

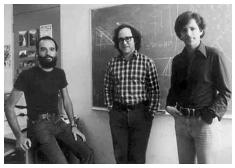
On a donc besoin de connaître l'inverse de a dans $\mathbb{Z}_{95}$...en s'assurant qu'il existe !

```
# inv_mod (17 % 95);;
- : classe = {representant = 28; modulo = 95}
```

puis on décode :

```
# dechiffre_affine 17 22  "*r6!r<9vW((W6W26!r<ooC96FW6r<2";;
- : string = "La maitresse en maillot de bain"
```

Exercice 1 - 46 Le système RSA



Ronald **R**IVEST, Adi **S**HAMIR et Leonard **A**DLEMAN mirent au point en 1977 un système de codage alors qu'ils essayaient au départ de démontrer que ce qu'ils allaient développer étaient une impossibilité logique : aah, la beauté de la recherche...
Et savez-vous quel est la base du système ? Le petit théorème de Fermat bien sûr ! Mais replaçons dans son contexte les résultats de R, S et A. Il n'y a pas si longtemps, la cryptographie était l'apanage des militaires et des diplomates. Depuis, les banquiers, les mega business men et women, les consommateurs internautes, les barons de la drogue ont de plus en plus recours aux messages codés. Oui, et alors ? Le problème, c'est que jusqu'ici, l'envoyeur et le destinataire partageaient une même clé secrète qu'il fallait faire voyager à l'insu de tous : les appareils d'état ont la valise diplomatique, mais les autres ?
C'est ici que Whitfield DIFFIE et Martin HELLMAN apparaissent un an auparavant, en 1976, avec leur idée de principe qu'on peut résumer ainsi : 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.
En termes plus mathématiques, cela donne : 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 π(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.
Tout ceci était très beau en théorie mais DIFFIE et HELLMAN n'arrivèrent pas à proposer de telles fonctions. Mais voici qu'arrivent Ronald, Adi et Leonard...
Nous utiliserons des fonctions **code** et **decode** dérivées de celles vues précédemment.
Avant d'envoyer un message, on doit connaître la *clé publique*, connue de tous et constituée d'un couple de nombres (n, e).
Et avant de connaître cette clé, il faut la fabriquer - secrètement - de la manière suivante :
On commence par fabriquer deux nombres premiers quelconques mais suffisamment grands, par exemple de l'ordre de 1000 bits (300 chiffres décimaux) :

```
let p = genere_premier 1000;;
let q = genere_premier 1000;;
```

Ces deux nombres ne sont connus que d'une personne : c'est la *clé privée*. Pour plus de clarté, nous appellerons la personne qui crée la clé Aliona et la personne qui va utiliser la clé publique pour envoyer le message Boris.
Aliona forme le produit de ces deux nombres.

```
let n = p &* q ;;
```

Ce nombre n est le module publique d'Aliona. Il reste à former le nombre e qui est la puissance de cryptage publique. Là encore, cela se fait par étapes.
Aliona commence par former le nombre **phi_n** (en référence à l'indicatrice d'EULER et en utilisant le théorème du même EULER : voir Exercice 1 - 27 page 23) de la manière suivante :

```
let phi_n = (p &- un) &* (q &- un) ;;
```

C'est là qu'il faut jouer avec la chance : Aliona doit choisir un nombre **e** premier avec **phi_n**. En pratique, elle choisit un grand nombre premier au hasard :

```
let e = genere_premier 1000;;
```

Il faut vérifier que **e** et **phi_n** sont premiers entre eux.

```
montre(bgcd e phi_n);;
```

Il reste à Aliona à calculer sa puissance de décryptage secrète : elle l'obtient en cherchant l'inverse de **e** modulo **phi_n**.
Aliona est maintenant prête à recevoir des messages cryptés selon le protocole RSA et Boris est prêt à lui en envoyer car Aliona a rendu publique sa clé **(e,n)**.
D'abord Boris numérise son message.
Il faut encore vérifier que notre nombre à crypter est plus petit que **p** et **q** (pourquoi ?). En pratique, on découpe le message en morceaux avec une méthode de chiffrement par blocs (cf Exercice 1 - 35 page 25).
Ensuite, Boris calcule ce nombre puissance **e** modulo **n** :
Aliona reçoit ce message et le décrypte grâce à sa clé secrète d qu'elle est la seule à connaître dans tout l'univers.

Recherche

Déterminez une fonction `bbezout` du type :

```
val inv_mod : Big_int.big_int * Big_int.big_int -> Big_int.big_int = <fun>
```

pour créer une fonction `binv_mod` du type :

```
val binv_mod : Big_int.big_int * Big_int.big_int -> Big_int.big_int = <fun>
```

qui calculera l'inverse de a modulo n puis coder et décoder un message.

CHAPITRE 2

Polynômes pour l'informatique

La notion de polynôme est primordiale en informatique : en effet, tout nombre entier peut être considéré comme un polynôme selon la base utilisée.
De plus, les codes détecteurs d'erreurs utilisé dans les réseaux (CRC) et les codes détecteurs et correcteurs d'erreurs utilisés par exemple dans les DVD, sont basés sur le calcul polynomial.

1 Généralités

Un polynôme à une indéterminée X et à coefficients dans un anneau $\mathbb{A}$ est en fait une suite *presque nulle* d'éléments de $\mathbb{A}$: « presque nulle » signifie que tous les termes de la suite sont nuls sauf un nombre fini d'entre eux...éventuellement nul (dans ce cas, on parle de *polynôme nul* qu'on notera par la suite Θ).
C'est le plus souvent ainsi qu'on les implémente informatiquement.
On note $\mathbb{A}[X]$ l'ensemble des polynômes à une indéterminée X et à coefficients dans un anneau $\mathbb{A}$.
Le **rang** du dernier coefficient non nul du polynôme est appelé le *degré* du polynôme. Le coefficient correspondant est appelé *coefficient dominant* du polynôme. Ainsi, $(1,0,5,2,0,0,...)$ a pour degré 3 et pour coefficient dominant 2.

Remarque

Par convention, le degré de Θ est $-\infty$.

L'ensemble des polynômes de $\mathbb{A}[X]$ de degré inférieur ou égal à n est noté $\mathbb{A}_n[X]$.
Pour nos yeux humains, il est souvent plus pratique de noter ces polynômes à l'aide d'une combinaison linéaire. Soit $P \in \mathbb{A}_n[X]$, alors il peut s'écrire :

$$P = (a_0, a_1, ..., a_n, 0, 0, 0, ...) = a_0X^0 + a_1X^1 + a_2X^2 + \cdots + a_nX^n$$

À retenir

Faites attention : les « X » et les « + » sont juste là pour faire joli. Enfin, disons plutôt que le « X » indique le rang du terme associé et les « + » séparent les différents termes.

Un polynôme ayant exactement un seul coefficient non nul est appelé un *monôme*.
On note le plus souvent le monôme $1X^k$ sous la forme X^k et le monôme a_0X^0 sous la forme a_0, quant à $0X^k$, on omet de l'écrire.

Remarque

Lorsqu'on travaille dans $\mathbb{A}_n[X]$, on peut mettre « en correspondance » un polynôme $a_0X^0 + a_1X^1 + a_2X^2 + \cdots + a_nX^n$ avec un vecteur de coordonnées $(a_0, a_1, ..., a_n)$ dans une base de $\mathbb{A}^{n+1}$.
Par la suite, nous confondrons $a_0X^0 + a_1X^1 + a_2X^2 + \cdots + a_nX^n$ avec $(a_0, a_1, ..., a_n)$.

Par exemple, $2X^3 + 3X + 1$, $X^2 + 3$, 37, sont des polynômes de $\mathbb{Z}_3[X]$, de degrés respectifs 3, 2 et 0.

2 Opérations élémentaires

2 1 Somme de polynômes

Considérons deux polynômes P et Q de $\mathbb{A}[X]$ de degrés respectifs n et m. Soit par exemple m le plus grand des deux. On peut donc restreindre l'ensemble d'étude à $\mathbb{A}_m[X]$.
Posons $P = (a_0, a_1, ..., a_m)$ et $Q = (b_0, b_1, ..., b_m)$, certains coefficients pouvant être éventuellement nuls. Alors la somme des polynômes P et Q qu'on notera $P + Q$ est définie par :

$$P + Q = (a_0 + b_0, a_1 + b_1, ..., a_m + b_m)$$

Par exemple, dans $\mathbb{Z}_3[X]$, $(2X^3 + 3X^2 + 7X + 1) + (5X^2 + 12X) = 2X^3 + 8X^2 + 19X + 1$.

2 2 Multiplication par un scalaire

Soit $P = (a_0, a_1, ..., a_m)$ un élément de $\mathbb{A}[X]$ et soit $\alpha \in \mathbb{A}$.
On définit le polynôme $\alpha P = (\alpha a_0, \alpha a_1, ..., \alpha a_m)$.
Par exemple, dans $\mathbb{Z}_3[X]$, $-3(2X^3 + 3X^2 + 7X + 1) = -6X^3 - 9X^2 - 21X - 3$.

2 3 Produit de polynômes

Considérons deux polynômes P et Q de $\mathbb{A}[X]$ de degrés respectifs n et m.
Posons $P = (a_0, a_1, ..., a_n)$ et $Q = (b_0, b_1, ..., b_m)$.
Alors le polynôme produit $P \cdot Q = (c_0, c_1, c_{n+m})$ avec

$$\forall k \in \{0, 1, 2, ..., n + m\}, \quad c_k = \sum_{i=0}^{k} a_i b_{k-i}$$

Par exemple, dans $\mathbb{Z}_3[X]$, avec $P = 2X^3 + 3X^2 + 7X + 1$ et $Q = 5X^2 + 12X$:

$$\begin{aligned} P\cdot Q &= (1\cdot 0)+(1\cdot 12+0\cdot 7)X+(1\cdot 5+0\cdot 3+7\cdot 12)X^2+(0\cdot 2+7\cdot 5+12\cdot 3)X^3+(2\cdot 12+3\cdot 5)X^4+(2\cdot 5)X^5 \\ &= 12X+89X^2+71X^3+39X^4+10X^5 \end{aligned}$$

3 Fonction polynôme associée

On peut associer au polynôme P, qui est un « être » formel, une fonction « concrète » :

$$\widetilde{P}:\begin{array}{ccc}\mathbb{A} & \to & \mathbb{A} \\ x & \mapsto & \widetilde{P}(x)=\sum_{i=0}^{n} a_i x^i\end{array}$$

Danger

Si dans $\mathbb{R}$, on confond souvent P et $\widetilde{P}$ comme vous l'avez fait au lycée, il est fondamental de bien voir la différence qui existe entre les deux dans d'autres anneaux, en particulier dans $\mathbb{F}_2$ qui sera notre corps fétiche.
Par exemple, $P=1+X+X^2+X^3$ et $Q=1+X^2+X^4+X^6$ sont des polynômes de $\mathbb{F}_2[X]$ différents : ils ne sont même pas de même degré.

Or $\widetilde{P}(x)=1+x+x^2+x^3=1+x+x+x=1+x$ donc $\widetilde{P}(x)=\begin{cases}1 & \text{si } x=0\\ 0 & \text{sinon}\end{cases}$

et $\widetilde{Q}(x)=1+x^2+x^4+x^6=1+x+x+x=1+x$ donc $\widetilde{Q}(x)=\begin{cases}1 & \text{si } x=0\\ 0 & \text{sinon}\end{cases}$

Ainsi $P\neq Q$ mais $\widetilde{P}=\widetilde{Q}$!
Pour plus de précisions, voir Exercice 2 - 5 page 42

4 Polynômes sur les corps

4 1 Division euclidienne

Nous allons à présent travailler sur $\mathbb{K}[X]$ avec $\mathbb{K}$ un corps. Cela va nous permettre d'introduire une division.

Théorème 2 - 1

1. L'anneau $\mathbb{K}[X]$ ne contient pas de diviseurs de zéro.
2. Si P et Q sont deux polynômes non nuls de $\mathbb{K}[X]$, alors $\deg(P\cdot Q)=\deg P+\deg Q$.

La preuve est demandée à l'Exercice 2 - 6 page 42.
On peut alors définir, comme avec les entiers, une division :

Théorème 2 - 2

Division euclidienne
Soient A et B deux polynômes de $\mathbb{K}[X]$ avec $B\neq\Theta$. Il existe un unique couple de polynômes (Q,R) dans $\mathbb{K}[X]\times\mathbb{K}[X]$ vérifiant :

$$A=B\cdot Q+R \quad \text{avec } R=\Theta \text{ ou } \deg(R)<\deg(B)$$

Nous allons étudier la démonstration car elle nous donnera une idée de l'algorithme de division.
On évacue les cas particuliers :

- si $A=\Theta$, alors $Q=R=\Theta$;
- sinon, si $\deg B>\deg A$, alors $Q=\Theta$ et $R=A$.

Nous supposerons donc maintenant que $A\neq\Theta$ et $\deg A\geqslant\deg B$. Nous allons raisonner par **récurrence** sur le degré de A.
Si $\deg A=0$, alors $\deg(B)\leqslant 0\Longrightarrow\deg(B)=0$. A et B sont donc des scalaires et on est ramené à la division dans le corps $\mathbb{K}$ où tous les éléments sont inversibles donc $Q=A/B$ et $R=\Theta$.
Supposons à présent que $\deg A=n\geqslant m=\deg B$ et $n>0$. On pose alors $A=(a_0,a_1,\ldots a_n)$ et $B=(b_0,b_1,\ldots,b_m)$ avec $b_m\neq 0_{\mathbb{K}}$ et $a_1\neq 0_{\mathbb{K}}$.

Posons $A_1 = A - (a_n/b_m)X^{n-m}B$. Alors soit $A_1 = \Theta$, soit $\deg A_1 < \deg A$: **pourquoi ? ?**
D'après l'hypothèse de récurrence, on en déduit qu'il existe deux polynômes Q_1 et R tels que $R = \Theta$ ou $\deg R < \deg B$ et $A_1 = BQ_1 + R$.
On obtient alors (**pourquoi ? ?**) :

$$A = \left((a_n/b_m)X^{n-m} + Q_1\right)B + R$$

et les polynômes $(a_n/b_m)X^{n-m} + Q_1$) et R conviennent.
Démontrez l'unicité du couple (Q,R).

4 2 Divisibilité - Polynômes irréductibles

Puisque nous avons défini une division, nous pouvons parler, comme avec les entiers, de divisibilité dans $\mathbb{K}[X]$:

Définition 2 - 1

Divisibilité dans K[X]
Dans $\mathbb{K}[X]$, un polynôme B divise un polynôme A si, et seulement si, il existe un polynôme Q tel que $A = B \cdot Q$.

Comme nous avions parlé de nombres entiers premiers, nous allons parler de polynômes irréductibles : si les premiers sont importants en cryptographie, les seconds le sont dans l'étude des réseaux.

Définition 2 - 2

Polynôme irréductible
Un polynôme P est irréductible si, et seulement si, il n'existe pas deux polynômes de degré supérieur ou égal à 1 tels que $P = P_1 \cdot P_2$.

Nous accepterons les résultats suivants qui peuvent être plus ou moins compliqués à démontrer :

Théorème 2 - 3

- Dans $\mathbb{R}[X]$, les seuls polynômes irréductibles sont les polynômes de degré inférieur à 1 ou les polynômes de degré 2 de discriminant strictement négatif.
- Dans $\mathbb{F}_2[X]$, pour tout entier naturel non nul, il existe au moins un polynôme de degré n irréductible.

Le dernier résultat bouleverse nos habitudes comme c'est souvent le cas dans $\mathbb{F}_2$ mais il sera important pour l'étude des réseaux.

Définition 2 - 3

Racine d'un polynôme
Soit $P \in \mathbb{K}[X]$ et $k \in \mathbb{K}$. On dit que k est une racine (ou un zéro) de P si, et seulement si, $\widetilde{P}(k) = 0_{\mathbb{K}}$.

Le théorème suivant sera utile :

Théorème 2 - 4

Le scalaire k est une racine de P si, et seulement si, $X - k$ divise P.

Une démonstration est demandée à l' Exercice 2 - 9 page 43.

Comme la caractérisation des nombres premiers, celle de polynômes irréductibles est un problème difficile. Un résultat est pourtant évident :

Théorème 2 - 5

Un polynôme irréductible **de degré supérieur à 1** n'a pas de racine.

Pourquoi ?
Ce qui nous intéresserait plus, c'est d'étudier la réciproque qui nous permettrait de caractériser les polynômes irréductibles. C'est l'objet de l' Exercice 2 - 10 page 43 où l'on démontre également le théorème suivant :

Théorème 2 - 6

Si P est un polynôme **de degré 2 ou 3** alors :

$$P \text{ est irréductible} \Leftrightarrow P \text{ n'a pas de racine}$$

4 3 Relation d'équivalence dans K[X]

Une nouvelle structure :

Définition 2 - 4

Idéal de $\mathbb{K}[X]$

Un idéal $(I,+)$ d'un anneau $(A,+,\cdot)$ est tel que :

- $I \subseteq A$;
- $(I,+)$ est un groupe ;
- $\forall (a,i) \in A \times I, \quad a \cdot i \in I$.

Soit $P \in \mathbb{K}[X]$ et $P\mathbb{K}[X] = \{P \cdot A \mid A \in \mathbb{K}[X]\}$.
On admettra le théorème suivant, dont une partie de la démonstration est demandée à l' Exercice 2 - 12 page 43 :

Théorème 2 - 7

Les idéaux de $\mathbb{K}[X]$ sont de la forme $P\mathbb{K}[X]$ (on dit que $\mathbb{K}[X]$ est un anneau principal).

On définit ensuite une relation d'équivalence dans $\mathbb{K}[X]$:

Théorème 2 - 8

Soit P un polynôme non nul de $\mathbb{K}[X]$. On définit la relation $\mathcal{R}$ dans $\mathbb{K}[X]$ par :

$$A \mathcal{R} B \Leftrightarrow A - B \in P\mathbb{K}[X]$$

C'est une relation d'équivalence. On la note le plus souvent à l'aide des mêmes notation qu'en arithmétique des entiers :

$$A \equiv B (\mathrm{mod}\, P) \quad \text{ou} \quad A \equiv B \bmod P$$

Comme en arithmétique des entiers, il est facile de relier cette relation à la division euclidienne :

Théorème 2 - 9

$A \equiv B \bmod P$ si, et seulement si, A et B ont le même reste dans la division euclidienne par P.

La démonstration de ce théorème constitue le sujet de l' Exercice 2 - 13 page 43.
Comme pour les entiers, on peut donc définir un ensemble quotient $\mathbb{K}[X]/p$: c'est l'ensemble des classes modulo P. On choisir comme représentants les restes dans la division par P...comme avec les entiers.
On peut définir des opérations sur les classes qui sont des prolongements des opérations dans $\mathbb{K}[X]$.
On peut alors vérifier que $(\mathbb{K}[X]/p,+,\cdot)$ a une structure d'anneau.
On pourrait alors définir le PGCD de deux polynômes, énoncer un théorème de BÉZOUT, donner une caractérisation des éléments inversibles de $\mathbb{K}[X]/p$ et démontrer le théorème suivant :

Théorème 2 - 10

$(\mathbb{K}[X]/p,+,\cdot)$ est un corps si, et seulement si, P est irréductible.

On pourrait parler d'extension de corps, de la théorie de GALOIS, de...mais ceci est une autre histoire, passionnante mais que nous n'aurons pas le temps de traiter...

5 Multiplication de polynômes

6 Exercices

Exercice 2 - 1 Complexité

Combien d'opérations arithmétiques élémentaires nécessitent la somme et le produit de deux polynômes ?

Exercice 2 - 2 Algorithme

Donnez des algorithmes impératifs et récursifs en pseudo-code de calcul de $P+Q$ et $P\cdot Q$. Une version CAML ?

Exercice 2 - 3 Structure

Quelle pourrait être la structure de $(\mathbb{A}[X],+,\cdot)$?

Exercice 2 - 4 Polynômes et bases : multiplication de grands entiers

Sur les machines de l'IUT à 32 bits, certains calculs posent vite des problèmes comme par exemple $123\,456\,789 \times 987\,654\,321$. Les polynômes vont nous permettre d'y remédier.

1. Écrivez 123 456 789 et 987 654 321 en base 1000.
2. Remplacez les « 1000 » par des « X » puis effectuez le produit des deux polynômes obtenus : nous avons un algorithme pour ça.
3. Remplacez les coefficients supérieurs à 1000 par un polynôme : par exemple, 12 345 sera remplacé par $12X+345$.
4. Simplifiez alors le polynôme produit puis lisez le résultat de la multiplication.

Exercice 2 - 5 Polynômes dans $\mathbb{F}_2[X]$

1. $A = X^5+X^2+X+1$ et $B = X^2+1$ sont deux polynômes de $\mathbb{F}_2[X]$. Effectuez les calculs suivants :
   (a) $A+B$.
   (b) $A-B$.
   (c) $A\times B$.
   (d) Les fonctions polynômes $\widetilde{A}$ et $\widetilde{B}$ sont-elles égales ?
2. (a) Simplifiez X^k+X^k.
   (b) Développez $(X-1)^2$, $(X+1)^2$.
   (c) Développez $(X^2+X+1)(X^3+X^2+X+1)$.
   (d) Effectuez la division euclidienne de X^3+X^2+X par $X+1$.
   (e) Soit $P_1 = X^2+X+1$ et $P_2 = X^4+X^2+1$. Comparez $\widetilde{P_1}$ et $\widetilde{P_2}$.

Exercice 2 - 6 Preuve du théorème 2 - 1 page 39

Démontrez le théorème sus-cité et profitez-en pour répondre aux « pourquoi » de la démonstration du théorème qui suit.

Exercice 2 - 7 Algorithme de division euclidienne

À partir de la démonstration du théorème 2 - 2 page 39, déterminez un algorithme déterminant quotient et reste de la division euclidienne de deux polynômes.

Exercice 2 - 8 Division euclidienne

1. Effectuez la division euclidienne de $2X^4-3X^2+5X-1$ par $X-2$ dans $\mathbb{R}[X]$.
2. Effectuer la division euclidienne de X^5+X+1 par $X+1$ dans $\mathbb{F}_2[X]$.
3. Effectuer la division euclidienne de X^n-1 par $X-1$ dans $\mathbb{F}_2[X]$.
4. Effectuer la division euclidienne de X^n+1 par $X+1$ dans $\mathbb{F}_2[X]$.
5. Effectuer la division euclidienne de $X^5+X^3+X^2+1$ par X^2+X+1 dans $\mathbb{F}_2[X]$.

Exercice 2 - 9 Preuve du théorème 2 - 4 page 40

1. Vérifiez que pour tout entier $i > 0$ on a
$$x^i - k^i = (x-k)(x^{i-1} + kx^{i-2} + k^2x^{i-3} + \cdots + k^{i-2}x + k^{i-1})$$
2. Calculez $\widetilde{P}(x) - \widetilde{P}(k)$ pour tout $x \in \mathbb{K}$ puis essayez de factoriser votre résultat.
3. Concluez.
4. Étudiez le problème réciproque.

Exercice 2 - 10 Polynômes irréductibles et racines

Travaillons dans $\mathbb{F}_2[X]$: il est plus simple d'y rechercher des racines car $\mathbb{F}_2$ n'a que deux éléments ! Vérifiez si les polynômes suivants ont des racines et s'ils sont irréductibles :

1. $X^2 + X + 1$
2. $X^3 + X + 1$
3. $X^4 + X^2 + 1$

Déduisez-en une preuve du théorème 2 - 6 page 40.

Exercice 2 - 11 Classes modulo un polynôme

P est le polynôme $X^2 + X + 1$ de $\mathbb{F}_2[X]$.

1. Démontrer que P est irréductible.
2. Déterminer tous les restes possibles de la division d'un polynôme de $\mathbb{F}_2[X]$ par P.
3. On assimile tout polynôme A de $\mathbb{F}_2[X]$ avec son reste dans la division par P. On dit alors que l'on fait les calculs modulo P ou dans l'anneau quotient $\mathbb{F}_2[X]/P$. On décide de noter

$$\begin{array}{rcl} a & = & (0,0) \text{ pour } \Theta = 0X^1 + 0X^0, \qquad b = (0,1) \text{ pour } 1 = 0X^1 + 1X^0 \\ c & = & (1,0) \text{ pour } X = 1X^1 + 0X^0, \qquad d = (1,1) \text{ pour } X+1 = 1X^1 + 0X^0 \end{array}$$

donc $\mathbb{F}_2[X]/P = \{a, b, c, d\}$ et, pour encore simplifier les notations, nous décidons de noter :

$$\begin{array}{rcl} a & = & 00 \text{ pour } \Theta, \qquad b = 01 \text{ pour } 1 \\ c & = & 10 \text{ pour } X, \qquad d = 11 \text{ pour } X+1 \end{array}$$

Calculez bb, dd, ad, bc, cc et cd.
Dressez la table de multiplication de $\mathbb{F}_2[X]/P = \{a, b, c, d\}$.

Exercice 2 - 12 Idéaux de K[X]

Soit $P \in \mathbb{K}[X]$. Vérifiez que $P\mathbb{K}[X] = \{P \cdot A \mid A \in \mathbb{K}[X]\}$ est un idéal de $\mathbb{K}[X]$.

Exercice 2 - 13 Démonstration du théorème 2 - 9 page 41

Tout est dans le titre...

Exercice 2 - 14 Décalage circulaire

Il s'avère utile en informatique de pouvoir « décaler » des chaînes de bits à l'aide d'une permutation circulaire.
Dans cet exercice, nous travaillerons avec des chaînes de 7 bits, comme en ASCII.
Soit c une chaîne quelconque, par exemple la chaîne 1000011 (qui correspond à « C » en ASCII :-)).
Nous allons la représenter par un polynôme de $\mathbb{F}_2[X]$:

$$1000011 \rightleftharpoons C = 1X^6 + 0X^5 + 0X^4 + 0X^3 + 0X^2 + 1X^1 + 1X^0 = X^6 + X + 1$$

On note $C_0 = C$ puis $C_{k+1} = (X \cdot C_k) \bmod (X^7 + 1)$ et c_k la chaîne de bits associée à C_k.
Calculez C_k pour $k \in \{0, 1, ..., 7\}$. Que remarquez-vous ? Une démonstration dans le cas général ?

Exercice 2 - 15 CRC

CRC : « Contrôle de redondance cyclique » en français ou « Cyclic Redundancy Check » en anglais.
Supposons que l'on veuille transmettre une chaîne de bits $b_n b_{n-1} ... b_1 b_0$ qui sera associée comme d'habitude au polynôme

$$C(x) = b_n X^n + b_{n-1} X^{n-1} + \cdots + b_1 X + b_0$$

Émetteur et récepteur disposent d'un même polynôme de contrôle G de degré $p < n$.
La chaîne émise sera :

$$C_e = X^p \cdot C + \left(X^p \cdot C \bmod G\right)$$

La chaîne reçue ne sera pas forcément la même, suite aux erreurs de transmission :

$$C_r = C_e + E$$

À quoi est égal C_r mod G ?
Le récepteur calcule le reste de C_r dans la division par G.
Si l'erreur est un multiple de G, elle ne sera pas détectée. Il faut donc choisir un bon générateur qui puisse détecter u maximum d'erreurs.
Avant d'en parler, voyons un exemple simple : on veut transmettre 10101101 et on dispose de G = 10111. Calculez la chaîne transmise.
On suppose que la chaîne reçue est 101011001110 : l'erreur est-elle détectée ?
Et si la chaîne reçue est 101011101110 ? 101000100010 ?
Il existe certaines règles dans le choix de G qui permettent de détecter certaines erreurs et dont la justification ne nous est pas toujours accessible :

Un seul bit erroné Alors E = 100...00. Il suffit de choisir G avec au moins deux bits valant 1 : pourquoi ?

Un nombre impair de bits erronés Il suffit de choisir G comme multiple de X + 1 ;

Une séquence de 1 le bit de poids faible doit être 1, ceci tant que la longueur de la séquence de 1 est inférieure à la taille de G. Pour de plus longues séquences, on n'a droit qu'à une probabilité de détection mais elle est égale à $0,5^r$.

Voici quelques polynômes générateurs usuels :

- CRC-16-IBM pour les ports USB : $X^{16} + X^{12} + x^2 + 1$;
- CRC-16-CCITT pour Bluetooth, PPP : $X^{16} + X^{12} + X^5 + 1$;
- CRC-32-IEEE802.3 pour Ethernet, WiFi : $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$.

CHAPITRE 3

Sommes, Suites et Séries pour l'informatique

L'addition est l'opération basique effectuée par le processeur : tout le reste n'est que fioriture...On rencontre les sommes en algorithmique (complexité), en réseau (décomposition d'un signal), etc.

1 Suites

2 $\sum$

« What is one and one and one and one and one and one and one and one ? »
« I don't know » said Alice. « I lost count ».
« She can't do Addition. »

Alice in wonderland - Lewis CAROLL

2 1 Notation

Si on écrit $1+2+3+4+\cdots+(n-1)+n$, on comprend que les $\cdots$ signifient qu'ils doivent être remplacés selon le motif induit par les termes les entourant.
On peut être amené à étudier des sommes plus générales : $a_0+a_1+\cdots+a_n$, chaque **terme** a_k de la somme étant défini d'une manière quelconque.
Si l'on écrit la somme avec des $\cdots$, il faut que le motif de construction soit présenté de manière suffisamment claire pour être induit :

$$1+4+9+22+53+128+309+...$$

n'est pas évident à comprendre mais :

$$1+4+(2\times4+1)+(2\times9+4)+\cdots+(2a_{n-1}+a_{n-2})$$

est plus explicite.

En 1820, le mathématicien Joseph FOURIER [a] introduit le fameux sigma majuscule. Citons-le :

Joseph FOURIER (1768-1830)

Le signe $\sum_{i=1}^{+\infty}$ indique que l'on doit donner au nombre entier i toutes ses valeurs 1, 2, 3,..., et prendre la somme des termes.

Notez bien, en tant qu'informaticien(ne), que le i joue ici le rôle d'une variable locale et aurait très bien pu porter un autre nom sans changer le résultat.
Depuis, on a généralisé l'emploi des bornes afin de définir les indices à partir de certaines propriétés. On peut par exemple écrire les sommes suivantes avec ou sans bornes explicitement spécifiées :

$$\sum_{\substack{1\leqslant k<100\\ k \text{ impair}}} k^2 = \sum_{k=0}^{49} (2k+1)^2$$

Cette formulation est particulièrement adaptée aux changements d'indices :

$$\sum_{1\leqslant k\leqslant n} a_k = \sum_{1\leqslant k+1\leqslant n} a_{k+1} = \sum_{k=1}^{n} a_k = \sum_{i=0}^{n-1} a_{i+1}$$

La dernière forme demande un peu de réflexion alors que la deuxième beaucoup moins.
L'informaticien canadien Kenneth IVERSON, prix TURING en 1979 et inventeur des langages APL et J, introduit une notation fort pratique mais peu utilisée en France, nommée **crochets d'IVERSON**. Soit P une certaine propriété :

$$[\text{P}] = \begin{cases} 1 & \text{si P est vraie} \\ 0 & \text{sinon} \end{cases}$$

Par exemple :

$$\sum_{\substack{1\leqslant k<100\\ k \text{ impair}}} k^2 = \sum_{k} k^2\,[1\leqslant k<100]\times[k \text{ impair}]$$

a. Nous aurions pu ne pas étuidier les sommes, ne pas compresser les images et le son car FOURIER échappa de peu à la guillotine durant la Terreur : http ://www.youtube.com/watch ?v=MwGD13BnH-o

2 2 Somme, boucle et récurrence

Une somme suggère naturellement l'emploi d'une boucle « pour ». En effet $\sum_{k=1}^{n} a_k$ se lit « somme des a_k **pour** k variant **de** 1 **à** n » ce qui, dans un langage quelconque, par exemple... OCAML donne, pour la somme des n premiers carrés d'entiers :

```
# let somme(n)=
    let s = ref 0 in
    for k=0 to n do
      s := !s + k*k
    done;
    !s;;
          val somme : int -> int = <fun>
# somme(3);;
- : int = 14
```

On peut également penser à une somme en terme de suite. Si on note $S_n = \sum_{k=0}^{n} k^2$, alors :

$$S_0 = 0; \quad \forall n > 0,\ S_n = S_{n-1} + n^2$$

Dans notre langage préféré :

```
# let rec s = function
  |0 -> 0
  |n -> s(n-1) + n*n;;
    val s : int -> int = <fun>
# s(3);;
- : int = 14
```

Cela nous rappelle d'ailleurs qu'une **suite** est une **fonction** de $\mathbb{N}$ dans tout autre ensemble (ici $\mathbb{N}$ lui-même).

2 3 Manipulation de sommes

La principale difficulté du calcul avec des sommes est d'obtenir par diverses manipulation une expression plus simple. Tout dérive le plus souvent des trois lois suivantes :

Distributivité $\sum_{k\in K} \lambda a_k = \lambda \sum_{k\in K} a_k$ (λ ne dépendant pas de k) ;

Associativité $\sum_{k\in K} (a_k + b_k) = \sum_{k\in K} a_k + \sum_{k\in K} b_k$;

Commutativité $\sum_{k\in K} a_k = \sum_{k\in K} a_{\pi(k)}$ ($\pi \in \mathfrak{S}_{\mathrm{Card}\,K}$).

Par exemple, nous allons prouver un résultat fort utile concernant les suites géométriques.

Nous voudrions calculer $S_n = \sum_{k=0}^{n} x^k$ pour tout réel $x \neq 1$.

Or

$$S_n + x^{n+1} = x^0 + \sum_{1\leqslant k\leqslant n+1} x^k = x^0 + \sum_{0\leqslant k-1\leqslant n} x^k = x^0 + \sum_{0\leqslant k-1\leqslant n} x^{(k-1)+1}$$

En posant $i = k-1$ on obtient :

$$S_n + x^{n+1} = x^0 + \sum_{0\leqslant i\leqslant n} x^{i+1} = x^0 + x \sum_{0\leqslant i\leqslant n} x^i = 1 + xS_n$$

d'où :

$$S_n(1-x) = 1 - x^{n+1}$$

Finalement, comme $x \neq 1$, on obtient :

$$\sum_{k=0}^{n} x^k = \frac{1-x^{n+1}}{1-x}$$

2 4 Sommes multiples

Nous voulons exprimer $S = a_1b_1 + a_1b_2 + a_1b_3 + a_2b_1 + a_2b_2 + a_2b_3 + a_3b_1 + a_3b_2 + a_3b_3$ avec le symbole $\sum$:

$$\begin{aligned}
S &= \sum_{i,j} a_i b_j \left[1 \leqslant i, j \leqslant 3\right] \\
&= \sum_{i,j} a_i b_j \left[1 \leqslant i \leqslant 3\right]\left[1 \leqslant j \leqslant 3\right] \\
&= \sum_{i} \left(\sum_{j} a_i b_j \left[1 \leqslant i \leqslant 3\right]\left[1 \leqslant j \leqslant 3\right] \right) \\
&= \sum_{i} a_i \left[1 \leqslant i \leqslant 3\right] \left(\sum_{j} b_j \left[1 \leqslant j \leqslant 3\right] \right) \\
&= \left(\sum_{i} a_i \left[1 \leqslant i \leqslant 3\right] \right) \left(\sum_{j} b_j \left[1 \leqslant j \leqslant 3\right] \right) \\
&= \left(\sum_{i=1}^{3} a_i \right) \left(\sum_{j=i}^{3} b_j \right)
\end{aligned}$$

Nous voulons cette fois exprimer $S' = a_1b_1 + a_1b_2 + a_1b_3 + a_2b_2 + a_2b_3 + a_3b_3$ avec le symbole $\sum$:

$$\begin{aligned}
S' &= \sum_{i,j} a_i b_j \left[1 \leqslant i \leqslant j \leqslant 3\right] \\
&= \sum_{i,j} a_i b_j \left[1 \leqslant i \leqslant 3\right]\left[i \leqslant j \leqslant 3\right] \\
&= \sum_{i} \left(\sum_{j} a_i b_j \left[1 \leqslant i \leqslant 3\right]\left[i \leqslant j \leqslant 3\right] \right) \\
&= \sum_{i} a_i \left[1 \leqslant i \leqslant 3\right] \left(\sum_{j} b_j \left[i \leqslant j \leqslant 3\right] \right) \\
&= \sum_{i=1}^{3} a_i \left(\sum_{j=i}^{3} b_j \right)
\end{aligned}$$

Nous verrons d'autres exemples en exercice.

3 Sommes infinies (séries)

3 1 Les dangers des ... mal maîtrisés

À quoi peut bien être égal $S = \frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} + \cdots$?
Multiplions les deux membres par 2, nous obtenons :
$2S = 2 + \frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} + \cdots = 2 + S$
donc $S = 2$. Waouh, magique !
On recommence avec :
$T = 2^0 + 2^1 + 2^2 + \cdots$
alors :
$2T = 2^1 + 2^2 + 2^3 + \cdots = T - 2^0 = T - 1$ donc $T = -1$.
Waouh !...Euh...Quoique, même réveillé brutalement en plein somme il peut sembler bizarre que la somme de nombres strictement positifs donne un nombre strictement négatif...
Il va falloir prendre des précautions.

3 2 Séries

Définition 3 - 1

Soit u une suite réelle définie sur $I = \{n \in \mathbb{N} \,|\, n \geq n_0\}$. On appelle série de terme général u_n la suite S définie par $S_n = \sum_{k=n_0}^{n} u_k$. Si la suite S converge, on dit que la série de terme général u_n converge et on note $\sum_{k=n_0}^{+\infty} u_k = \lim_{n\to+\infty} S_n$. Une série qui ne converge pas est dite divergente, dans ce cas l'écriture $\sum_{k=n_0}^{+\infty} u_k$ **n'a pas de sens**.
La série de terme général u_n est notée $(\sum u_n)$.

Si on nous demande d'étudier la série $\left(\sum \frac{1}{n^2}\right)$, on nous demande d'étudier la suite S définie par $S_n = \sum_{k=1}^{n} \frac{1}{k^2}$ avec évidemment $n \in \mathbb{N}^*$.

Remarque

Si la suite u est définie à partir de n_0, rien ne nous empêche de changer les notations pour travailler avec la même suite définie à partir du rang 0 ou du rang 1 ou ... En effet, condidérons la suite v définie par $v_n = u_{n+n_0}$, il est cair que la suite v est definie sur $\mathbb{N}$ et, qu'étudier la suite v, c'est étudier la suite u. La conséquence de ceci est que, dans ce qui suit, les suites qui interviendront seront, suivant les besoins, définie à partir de $n = 0$ ou $n = 1$ ...

Remarque

S_n est appelé une somme partielle.

Théorème 3 - 1

Si la série $(\sum u_n)$ converge alors on a $\lim_{n\to+\infty} u_n = 0$

La démonstration est très simple.
Considérons $S_n = \sum_{j=0}^{n} u_j$. Par hypothèse $\lim_{n\to+\infty} S_n = \lim_{k\to+\infty} S_k = \ell$. Si nous remplaçons k par $n-1$, nous obtenons

$$\lim_{n\to+\infty} S_n = \lim_{n-1\to+\infty} S_{n-1} = \ell$$

mais comme $S_n - S_{n-1} = u_n$ on en déduit

$$\lim_{n\to+\infty} u_n = \lim_{n\to+\infty} (S_n - S_{n-1}) = \ell - \ell = 0$$

Remarque

La réciproque est totalement fausse, ce n'est pas parce que le terme général d'une série a pour limite zéro que cette série converge.

3 3 Série harmonique

Nous allons démontrer que la série $\left(\sum \frac{1}{n}\right)$, qui porte le nom de série harmonique, diverge. Nous avons donc $u_n = \frac{1}{n}$ avec $n \in \mathbb{N}^*$ évidemment et

$$S_n = \sum_{j=1}^{n} u_j = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

La suite S est une suite de termes strictement positifs et comme $S_{n+1} - S_n = \frac{1}{n+1} > 0$ nous sommes assurés que la suite S est strictement croissante : ainsi, soit elle diverge vers $+\infty$, soit elle converge vers une limite réelle ℓ.
Calculons $S_{2n} - S_n$.

$$\begin{aligned} S_n &= \sum_{j=1}^{n} u_j = 1 + \frac{1}{2} + \cdots + \frac{1}{n} \\ S_{n=2n} &= \sum_{j=1}^{2n} u_j = 1 + \frac{1}{2} + \cdots + \frac{1}{n} + \frac{1}{n+1} + \frac{1}{n+2} + \cdots + \frac{1}{n+n} \\ S_{2n} - S_n &= \frac{1}{n+1} + \frac{1}{n+2} + \cdots + \frac{1}{n+n} \end{aligned}$$

La dernière somme comporte exactement n termes tous $\geqslant \frac{1}{2n}$. On obtient alors

$$S_{2n} - S_n \geqslant \frac{1}{2n} + \frac{1}{2n} + \cdots + \frac{1}{2n} = \frac{n}{2n} = \frac{1}{2}$$

Supposons que (S_n) converge vers ℓ, alors $\lim_{n\to+\infty} S_n = \lim_{n\to+\infty} S_{2n} = \ell$.
En reportant dans l'inégalité obtenue précédemment : $\ell - \ell \geqslant \frac{1}{2}$ ce qui est absurde donc (S_n) diverge vers $+\infty$, doucement, mais sûrement.

3.4 Séries télescopiques

Intéressons-nous à la convergence de la série de terme général $u_n = \dfrac{1}{n^2-1}$; il est évident que u_n n'est définie que pour $n \geq 2$. Nous allons calculer $\lim\limits_{n\to+\infty} S_n$ avec $S_n = \sum\limits_{k=2}^{n} u_k$ en utilisant la remarque indispensable suivante :

$$\frac{1}{x^2-1} = \frac{1}{2}\left(\frac{1}{x-1} - \frac{1}{x+1}\right)$$

Cela donne :

$$\begin{aligned} S_n &= \sum_{k=2}^{n} u_k = \sum_{k=2}^{n} \frac{1}{k^2-1} = \frac{1}{2}\sum_{k=2}^{n}\left(\frac{1}{k-1} - \frac{1}{k+1}\right) \\ S_n &= \frac{1}{2}\left(\sum_{k=2}^{n}\frac{1}{k-1} - \sum_{k=2}^{n}\frac{1}{k+1}\right) \end{aligned}$$

Dans la première sommation remplaçons k par $j+1$ et dans la deuxième k par $j-1$:

$$S_n = \frac{1}{2}\left(\sum_{j=1}^{n-1}\frac{1}{j} - \sum_{j=3}^{n+1}\frac{1}{j}\right) = \frac{1}{2}\left(1 + \frac{1}{2} - \frac{1}{n} - \frac{1}{n+1}\right)$$

On obtient alors

$$\lim_{k\to+\infty} S_n = \frac{3}{4}$$

Il est alors permis d'écrire que $\sum\limits_{n=2}^{+\infty} u_n = \sum\limits_{n=2}^{+\infty} \dfrac{1}{n^2-1}$ a un sens ou existe et vaut $\dfrac{3}{4}$. On dit aussi que la somme de la série est $\dfrac{3}{4}$.

Remarque

Il ne faut pas croire que l'on est capable de trouver la somme de toute série convergente, c'est plutôt rare, par contre, on est très souvent capable de dire si une série converge ou non mais nous ne présenterons pas les outils évolués qui le permettent.

3.5 Série géométrique

On appelle série géométrique toute série dont le terme général est de la forme $u_n = q^n$.
D'après le théorème précédent, il est manifeste (**pourquoi ? !**) qu'une série géométrique converge si, et seulement si,

$$|q| < 1$$

Calculons dans ce cas la somme de la série :

$$\begin{aligned} S_n &= \sum_{j=0}^{n} q^j = 1 + q + q^2 + q^3 + \cdots + q^n = \frac{1-q^{n+1}}{1-q} \text{ et} \\ \lim_{n\to+\infty} S_n &= \frac{1}{1-q} = \sum_{j=0}^{+\infty} q^j \end{aligned}$$

3.6 Série exponentielle

C'est la série dont le terme général est de la forme $u_n = \dfrac{x^n}{n!}$ où x est un réel quelconque. Nous admettons (la démonstration vous sera donnée peut-être donnée l'année prochaine) qu'une telle série converge et que

$$\sum_{k=0}^{+\infty} \frac{x^k}{k!} = \mathrm{e}^x, \text{ pour tout } x \text{ réel.}$$

4 Exercices

4 1 Le symbole $\sum$

Exercice 3 - 1

On travaille dans $\mathbb{R}$. Calculer les sommes :

1. $\sum_{i=3}^{6} ij$, $\sum_{i=0}^{4} i^2$, $\sum_{k=0}^{4} k^2$, $\sum_{j=0}^{4} i^2$, $\sum_{i=1}^{50} a_j$
2. $\sum_{i=1}^{n\geq 1} a$, $\sum_{i=h}^{n\geq h} a$, $\sum_{i=1}^{n\geq 1} 3$
3. $\sum_{1\leq i<j\leq 5} ij$, $\sum_{1\leq i<j\leq 5} (i+2j)$

Exercice 3 - 2

On travaille dans $\mathbb{R}$, calculer les sommes $\sum_{i=1}^{n} i$, $\sum_{i=50}^{100} j$, $\sum_{i=h}^{k} i$, $\sum_{i=h}^{k} (i+j)$, $\sum_{i=h}^{k} ij$, $\sum_{i=h}^{k} (\alpha i+j)$

Exercice 3 - 3

On considère un tableau de nombres ayant n lignes et p colonnes. Les lignes sont numérotées du haut en bas et les colonnes de la gauche vers la droite. Le nombre se trouvant sur la $i^{ème}$ ligne et $j^{ème}$ colonne est noté $a_{i,j}$ ou a_{ij}. Par exemple, si $n=4$ et $p=3$, le tableau se présente par

$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,1}$	$a_{3,2}$	$a_{3,3}$
$a_{4,1}$	$a_{4,2}$	$a_{4,3}$

On note L_i la somme des nombres de la ligne numéro i, C_j la somme des nombres de la colonne j et T la somme de tous les nombres du tableau.

1. Exprimer L_i et C_j à l'aide d'un symbole $\sum$.
2. Exprimer T de deux façons.

Exercice 3 - 4

On reprend les notations de l'exercice précédent avec $n=p=5$. Que calculent les sommes suivantes ? On pourra donner la solution en grisant les cases du tableau qui correspondent aux nombres intervenant dans les sommes.

1. $\sum_{i=1}^{5}\sum_{j=1}^{5} a_{ij}$
2. $\sum_{j=1}^{5}\sum_{i=1}^{5} a_{ij}$
3. $\sum_{i=1}^{5}\sum_{j=1}^{i} a_{ij}$
4. $\sum_{j=1}^{5}\sum_{i=1}^{j} a_{ij}$
5. $\sum_{i=1}^{5}\sum_{j\geq i} a_{ij}$
6. $\sum_{i+j=6} a_{ij}$
7. $\sum_{1\leq i<j\leq 5} a_{ij}$

Exercice 3 - 5

Un fichier séquentiel informatisé est schématiquement constitué ainsi, chaque fiche contient

- *L'adresse* S *de la fiche suivante si elle existe sinon* **nil** *pour indiquer la fin du fichier.*
- *L'adresse* P *de la fiche précédente si elle existe sinon* **nil** *pour indiquer qu'il n'y a rien en deçà.*
- *Une donnée* D_i *alphanumérique*
- *Une première adresse* **début** *qui est l'adresse de la première fiche.*

Nous supposerons que le fichier est trié suivant l'ordre croissant et pour simplifier l'étude nous poserons $D_i = i$ (la donnée est égale au numéro de la fiche). Le fichier contient N fiches. Nous appellerons temps d'accès à la fiche numéro i le nombre de fiches lues en partant de $début$ pour arriver à la fiche i y compris la fiche i. Ainsi le temps d'accès à la fiche 1 est 1.

Recherche séquentielle.

Pour trouver la fiche i ($1 \leq i \leq N$) on part de $début$, on lit les fiches dans l'ordre des numéros jusqu'à obtenir la fiche i.

Si on cherche successivement les fiches 1,2,3,$\cdots$,N en repartant de $début$ à chaque fois, quel est le temps moyen d'accès à une fiche ?

Recherche par saut.

- *Nous supposerons ici que* N *est le carré d'un entier,* $N = a^2$ *(*$a \in \mathbb{N}^*$*). Soit* k *un entier divisant* N *et supérieur à 1,* $N = kj$.
- *Nous créons un autre fichier dit de nœuds* $N_1, N_2, \ldots, N_j$.
- *Le départ, noté* **dép**, *contient l'adresse du premier nœud.*
- *Le premier nœud contient l'adresse de la fiche numéro* k, *la donnée de la fiche numéro* k *l'adresse du nœud suivant et l'adresse de la fiche* $k-1$.
- *Le deuxième nœud contient l'adresse de la fiche numéro* $2k$, *la donnée de la fiche numéro* $2k$ *l'adresse du nœud suivant et de la fiche numéro* $2k-1$.
- ……
- *Le dernier nœud (le* $j^{ème}$*) contient l'adresse de la fiche numéro* N, et celle de $N-1$, *la donnée de la fiche numéro* N *et* **nil**.

Pour chercher une fiche i on part de **dép**, on passe au premier nœud et on regarde la donnée α portée par le nœud. Si $\alpha < i$ on passe au nœud suivant, sinon on passe à la fiche indiquée par le nœud et, si nécessaire, on revient en arrière dans le fichier jusqu'à atteindre la fiche i. Un temps d'accès à une fiche sera le nombre de nœuds parcourus + le nombre de fiches parcourues y compris celle cherchée.
Si on cherche successivement les fiches $1,2,3,\cdots,N$ en repartant de **dép** à chaque fois, on se propose de déterminer le temps moyen d'accès à une fiche et trouver la valeur de k qui minimise ce temps moyen.

1. Déterminer le temps d'accès à la fiche 1.
2. Déterminer le temps d'accès à la fiche 2 (si $k > 2$).
3. Déterminer le temps d'accès à la fiche k.
4. Déterminer le temps d'accès à la fiche $k-1$.
5. Déterminer le temps d'accès à la fiche $k+1$.
6. Déterminer le temps d'accès à la fiche ik.
7. Déterminer le temps d'accès à la fiche N.
8. Déterminer le temps d'accès à la fiche j.
9. Déterminer le temps total d'accès aux fiches du $i^{ème}$ paquet.
10. Déterminer le temps total d'accès à toutes les fiches.
11. Déterminer le temps moyen d'accès à une fiche en fonction de k et de N.
12. Déterminer la valeur de k qui minimise ce temps moyen.
13. Comparer le temps moyen d'accès à une fiche par une recherche séquentielle et une recherche par saut en prenant $N = 10\,000$ et une unité de temps égale à $10^{-2}s$.

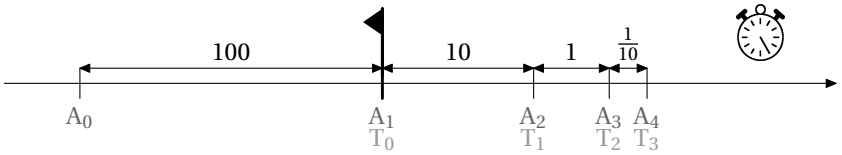
4 2 Séries

Exercice 3 - 6 Achille et la tortue

Le paradoxe suivant a été imaginé par ZÉNON D'ÉLÉE (490-430 Avant JC). Achille fait une course avec la tortue. Il part 100 mètres derrière la tortue, mais il va dix fois plus vite qu'elle. Quand Achille arrive au point de départ de la tortue, la tortue a parcouru 10 mètres. Pendant qu'Achille parcourt ces 10 mètres, la tortue a avancé d'un mètre. Pendant qu'Achille parcourt ce mètre, la tortue a avancé de 10cm... Puisqu'on peut réitérer ce raisonnement à l'infini, Zénon conclut qu'Achille ne peut pas dépasser la tortue...
Pour étudier ce problème, nous aurons besoin d'un résultat intermédiaire.
Il est assez simple de démontrer par récurrence, par exemple, que pour tout entier $n > 1$ et tout réel $x > -1$, on a :

$$(1+x)^n > 1 + nx \quad \text{(Inégalité de BERNOULLI)}$$

Qu'en déduisez-vous au sujet de la limite des suites de terme général q^n ?
Voici la situation :



Intéressons-nous d'abord à la distance Achille-Tortue. Notons d_n la distance :

$$d_n = T_n - A_n = T_n - T_{n-1} = \frac{1}{10}(T_{n-1} - A_{n-1}) = \frac{1}{10}d_{n-1}$$

La suite (d_n) est donc géométrique de raison $1/10$ et de premier terme 100. On en déduit que $d_n = 100 \times \left(\frac{1}{10}\right)^{n-1}$.
Or $|1/10| < 1$, donc $\lim\limits_{n\to+\infty} (1/10)^{n-1} = 0 = \lim\limits_{n\to+\infty} d_n$.
Ainsi Achille va rattrapper la tortue, mais au bout d'une infinité de trajets : pour le Grec Zénon, la notion d'infini étant « au-delà du réel » ; pour lui Achille ne rattrapera jamais la tortue, ce qui est absurde.
Intéressons-nous plutôt à la durée du trajet d'Achille pour atteindre la tortue, en supposant sa vitesse constante et égale à v.
Notons $t_1 = \frac{d_1}{v} = \frac{100}{v}$, $t_2 = \frac{d_2}{v} = \frac{1}{10}\frac{d_1}{v} = \frac{1}{10}t_1$, etc. La suite (t_n) est donc géométrique de raison $\frac{1}{10}$ de premier terme $t_1 = 100/v$, d'où $t_n = \frac{100}{v}\left(\frac{1}{10}\right)^{n-1}$. La durée du trajet est alors :

$$\begin{aligned}\tau_n &= t_1 + t_2 + \cdots + t_n \\ &= \frac{100}{v}\left(1 + \frac{1}{10} + \left(\frac{1}{10}\right)^2 + \cdots + \left(\frac{1}{10}\right)^{n-1}\right) \\ &= \frac{1000}{9v}\left(1 - \left(\frac{1}{10}\right)^n\right)\end{aligned}$$

Nous venons de voir qu'Achille atteindra la tortue quand n tend vers $+\infty$. Or nous savons calculer $\lim\limits_{n\to+\infty} \tau_n = \frac{1000}{9v}$ qui est un nombre fini. Achille va donc effectuer cette infinité de trajets en un temps fini égal à $1000/9v$.
Zénon pensait qu'une somme infinie de termes strictement positifs était nécessairement infinie, d'où le paradoxe à ses yeux. Il a fallu des siècles à l'esprit humain pour dépasser cette *limite*.
Vous pouvez donc prouver le petit théorème suivant :

Théorème 3 - 2

Série géométrique
La suite de terme général

$$S_n = \sum_{k=0}^{n} q^n$$

converge si, et seulement si, $|q| < 1$. Dans ce cas :

$$S = \lim_{n\to+\infty} S_n = \frac{1}{1-q}$$

Exercice 3 - 7 Tapis de Sierpinski

Monsieur SIERPINSKI avait ramené d'un voyage en Orient un tapis carré de 1 mètre de côté dont il était très content. Jusqu'au jour où les mites s'introduisirent chez lui.
En 24 heures, elles dévorèrent dans le tapis un carré de côté trois fois plus petit, situé exactement au centre du tapis. En constatant les dégats, Monsieur Sierpinski entra dans une colère noire ! Puis il se consola en se disant qu'il lui restait huit petits carrés de tapis, chacun de la taille du carré disparu. Malheureusement, dans les 12 heures qui suivirent, les mites avaient attaqué les huit petits carrés restants : dans chacun, elles avaient mangé un carré central encore trois fois plus petit. Et dans les 6 heures suivantes elles grignotèrent encore le carré central de chacun des tout petits carrés restants. Et l'histoire se répéta, encore et encore ; à chaque étape, qui se déroulait dans un intervalle de temps deux fois plus petit que l'étape précédente, les mites faisaient des trous de taille trois fois plus petite...

1. Faire des dessins pour bien comprendre la géométrie du tapis troué. Calculer le nombre total de trous dans le tapis de Monsieur Sierpinski après n étapes. Calculer la surface S_n de tapis qui n'a pas encore été mangée après n étapes. Trouver la limite de la suite $(S_n)_{n\geq 0}$. Que reste-t-il du tapis à la fin de l'histoire ?
2. Calculer la durée totale du festin « mitique »...

Exercice 3 - 8

$|x| < 1$ et on note $S_n(x) = \sum\limits_{k=0}^{n} x^k$, $\sigma_n(x) = \sum\limits_{k=0}^{n} kx^k = \sum\limits_{k=1}^{n} kx^k$.

1. Calculer $S_n'(x)$.
2. Calculer $\sum\limits_{k=0}^{+\infty} x^k$.
3. Calculer la limite de $S_n'(x)$ lorsque n tend vers $+\infty$.
4. Exprimer $\sigma_n(x)$ à l'aide de $S_n(x)$.
5. Calculer $\sum\limits_{k=0}^{+\infty} kx^k$.

Exercice 3 - 9

On note $\Pr([X=k]) = \dfrac{\lambda^k}{k!}e^{-\lambda}$. Calculer

1. $E(X) = \sum_{k=0}^{+\infty} k \Pr([X=k])$.
2. $E\left(X^2\right) = \sum_{k=0}^{+\infty} k^2 \Pr([X=k])$.
3. $E\left(X^2\right) - E(X)^2$.

Exercice 3 - 10

On note $\Pr([X=k]) = p(1-p)^{k-1}$ avec $p \in]0,1[$. Calculer $\sum_{k=1}^{+\infty} \Pr([X=k])$ et $\sum_{k=1}^{+\infty} k \Pr([X=k])$.