REPRESENTATION DES NOMBRES

I) Principe de la numération

Numération

Les nombres entiers naturels que nous connaissons aujourd'hui sont écrits en base 10. Cela signifie, par exemple, que le nombre 25021 vaut : $2 \times 10^4 + 5 \times 10^3 + 0 \times 10^2 + 2 \times 10^1 + 1 \times 10^0$. C'est le principe de la numération de position (ici en base 10) : les chiffres 2 du nombre 25021 ne représentent pas la même valeur (20000 pour le premier, 20 pour le second). Cette façon d'écrire les nombres vient des indiens via les mathématiciens arabes…

En fait le travail peut se faire dans une autre base que 10. Si on se donne une base b, c'est-à-dire un entier naturel supérieur ou égal à 2 on a le résultat (mathématique…)

Théorème : Soit $b \in \mathbb{N}^*$,$b>1$. $\forall n \in \mathbb{N}^*$, $\exists ! p \in \mathbb{N}$, $\exists ! (a_p, a_{p-1}, .. , a_0) \in [\![0, b-1]\!]^{p+1}$ tel que:

$n = a_p b^p + a_{p-1} b^{p-1} + ... + a_1 b^1 + a_0$ et $a_p \neq 0$

Notation: On note $n = \overline{a_p a_{p-1} ... a_1 a_0}^{(b)}$ et on dit que c'est l'écriture de n en base b

Dem: Pour info… Soit $b \in \mathbb{N}$, $b \geq 2$.

Montrons d'abord l'existence de l'écriture.

Soit P_q la propriété suivante : " $\forall k \in [\![b^q, b^{q+1}-1]\!]$, k s'écrit sous la forme : $k = a_q b^q + a_{q-1} b^{q-1} + ... + a_1 b^1 + a_0$ avec $(a_q, a_{q-1}, .. , a_0) \in [\![0, b-1]\!]^{q+1}$ et $a_q \neq 0$ "

- P_0 est vraie car si $k \in [\![1, b-1]\!]$, k s'écrit sous la forme $k = k$ avec $k \in [\![0, b-1]\!]$, $k \neq 0$
- Si $\forall j \in \{0,...,q\}$, P_j vraie: Montrons que P_{q+1} est vraie. Soit $k \in [\![b^{q+1}, b^{q+2}-1]\!]$, $x = \frac{k}{b^{q+1}} \in [1, b[$.

Donc, si on pose $a_{q+1} = E(x)$, on a $a_{q+1} \in [\![0, b-1]\!]$ et $a_{q+1} \neq 0$. Mais alors : $0 \leq k - a_{q+1} b^{q+1} < b^{q+1}$, donc comme toutes les P_j sont vraies pour $j \leq q$, soit $k - a_{q+1} b^{q+1} = 0$ soit il est de la forme $k - a_{q+1} b^{q+1} = a_p b^p + ... + a_1 b^1 + a_0 = a_q b^q + a_{q-1} b^{q-1} + ... + a_1 b^1 + a_0$ en posant, le cas échéant , $a_q = 0$, $a_{q-1} = 0$ ... si ces termes ne sont pas définis Ainsi : $k = a_{q+1} b^{q+1} + a_q b^q + ... + a_1 b^1 + a_0$ avec $a_{q+1} \neq 0$ et $\forall i \leq q+1$, $a_i \in [\![0, b-1]\!]$ Donc P_{q+1} est vraie.

- On a donc $\forall q \in \mathbb{N}$, P_q vraie .

Unicité

Si $n \in \mathbb{N}^*$ possède deux écritures sous la forme $n = a_q b^q + a_{q-1} b^{q-1} + ... + a_1 b^1 + a_0$ et $n = c_p b^p + ... + c_1 b^1 + c_0$ avec $\forall i$, $0 \leq a_i \leq b-1$, $\forall j$, $0 \leq c_j \leq b-1$, $a_q \geq 1$ et $c_p \geq 1$.

La première écriture entraîne $b^q \leq n \leq b^{q+1}-1$ et la seconde entraîne $b^p \leq n \leq b^{p+1}-1$ Ainsi $p = q$.

On suppose que les deux écritures sont distinctes.

Soit alors r le plus grand indice tel que $a_r \neq c_r$ On a : $(a_r - c_r) b^r = (c_{r-1} - a_{r-1}) b^{r-1} + ... + (c_1 - a_1) b + (c_0 - a_0)$

D'où : $|a_r - c_r| b^r \leq |c_{r-1} - a_{r-1}| b^{r-1} + ... + |c_1 - a_1| b + |c_0 - a_0| \leq (b-1)(b^{r-1} + .. + b + 1)$

Ainsi $|a_r - c_r| b^r \leq b^r - 1 < b^r$ ce qui est impossible car $|a_r - c_r| \geq 1$. D'où l'unicité de l'écriture.

Pratique: Ecrire 1998 en base 2, 8 et 16

Méthode 1: on utilise l'algorithme issu de la démonstration précédente, à savoir : on recherche la plus grande puissance de b qui soit inférieure à n puis le plus grand multiple de cette puissance qui soit inférieur à n puis on réitère le procédé avec le reste de la division euclidienne de n par ce multiple, en ajoutant des 0 pour les puissances absentes :

$1998 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^3 + 2^2 + 2^1 = = 3.8^3 + 7.8^2 + 1.8^1 + 6 = 7.16^2 + 12.16^1 + 14$

D'où : $1998 = \overline{11111001110}^{(2)} = \overline{3716}^{(8)} = \overline{7CE}^{(16)}$ (en base 16, A représente le "chiffre "dix" ", B vaut onze, C douze, D treize, E quatorze et F quinze)

Méthode 2: on effectue les divisions successives de n (puis des quotients) par b : l'écriture de n en base b est la juxtaposition de tous les restes obtenus mais dans l'ordre inverse de celui obtenu :

$1998 = 124 \times 16 + 14$, $124 = 7 \times 16 + 12$, $7 = 0 \times 16 + 7$:

D'où : $1998 = 7 \times 16^2 + 12 \times 16 + 14 = \overline{7CE}^{(16)}$

Algorithmes de l'addition et de la multiplication

On a $n = a_q b^q + a_{q-1} b^{q-1} + ... + a_1 b^1 + a_0$ et $m = c_p b^p + ... + c_1 b^1 + c_0$

Addition

Pour effectuer la somme de n et de m, on écrit les nombres en base b l'un au dessus de l'autre en alignant sur le chiffre des unités.
On ajoute les chiffres des unités : on écrit dans le chiffre des unités du total, le reste de la division euclidienne de cette somme par b et on "retient" le quotient de $a_0 + c_0$ par b.
Pour le chiffre suivant on effectue la somme de a_1 et de c_1 et de la retenue précédente, puis on inscrit le reste de la division euclidienne de cette somme par b et on met en retenue le quotient de cette division euclidienne. On réitère le procédé sachant qu'une absence de chiffre à une place équivaut à un 0

Exemple: Calculer $\overline{110110}^{(2)} + \overline{1110111}^{(2)}$

Retenues		1	1	1	0	1	1	0	
n				1	1	0	1	1	0
m	+		1	1	1	0	1	1	1
Somme	=	1	0	1	0	1	1	0	1

D'où $\overline{110110}^{(2)} + \overline{1110111}^{(2)} = \overline{10101101}^{(2)}$

De même on peut adapter l'algorithme de soustraction avec le système de retenues, dans une autre base que 10.

Multiplication

Pour le produit de n et de m, on écrit les nombres n et m comme pour la somme.
On commence par effectuer le produit de n par le chiffre c_0 des unités de m :

> On écrit alors dans le chiffre des unités du total, le reste de la division euclidienne de $a_0 \times c_0$ par b et on "retient" le quotient de $a_0 \times c_0$ par b.
> Pour le chiffre suivant on effectue le produit de $a_1 \times c_0$ et on ajoute la retenue précédente, puis on inscrit le reste de la division euclidienne de ce nombre par b et on met en retenue le quotient de cette division euclidienne. On réitère le procédé pour le produit de n et du premier chiffre de m

Puis on passe au second chiffre c_1 de m : on écrit sous le résultat précédent le résultat du produit de n par c_1 mais décalé d'un chiffre vers la gauche (car en fait on multiplie n par $b \times c_1$). Puis on réitère les calculs des produits de n par tous les chiffres de m (en décalant à chaque changement d'un chiffre vers la gauche) et on ajoute les résultats obtenus

Exemple: Calculer $\overline{40201}^{(5)} \times \overline{10321}^{(5)} = \overline{431020021}^{(5)}$

retenue c_2					2	0	1	0	0	
retenue c_1					1	0	0	0	0	
retenue c_0						0	0	0	0	
n						4	0	2	0	1
m	×					1	0	3	2	1
retenues de la somme		0	1	1	1	1	1	0	0	
$n \times c_0$						4	0	2	0	1
$n \times c_1$				1	3	0	4	0	2	
$n \times c_2$			2	2	1	1	0	3		
$n \times c_3$			0	0	0	0	0			
$n \times c_4$		4	0	2	0	1				
Produit	=	4	3	1	0	2	0	0	2	1

D'où $\overline{40201}^{(5)} \times \overline{10321}^{(5)} = \overline{431020021}^{(5)}$

Remarque: On pourra omettre d'écrire les retenues nulles ainsi que les multiplications par les chiffres 0 en prenant garde de décaler suffisamment les produits des chiffres suivants

Application aux opérations sur les grands nombres

Pour effectuer la somme ou le produit de deux "grands" entiers, on pourra écrire ces nombres dans une base puissance de 10 (par exemple 1000 ou 1000000) sur laquelle on maîtrise mieux les calculs habituels ou pour laquelle le nombre des chiffres limité des calculatrices n'est pas contraignant

Exemple: Calculer 742 361 423 × 405 512 741 On travaille en base 1000

retenue c_2				300	146	171	
retenue c_1				380	185	216	
retenue c_0				550	267	313	
n					742	361	423
m	×				405	512	741
retenues de la somme		001	001	000	001		
$n \times c_0$				550	089	814	443
$n \times c_1$			380	089	048	576	
$n \times c_2$		300	656	376	315		
Produit	=	301	037	016	453	390	443

D'où 742 361 423 × 405 512 741 = 301 037 016 453 390 443

II) Principe de la représentation des nombres entiers en machine

Dans un ordinateur, les nombres entiers, sauf sur certains logiciels spécifiques, sont codés sur un nombre prédéfini d'octets. Typiquement, un processeur 32 bits codent les entiers sur 32 bits donc sur 4 octets. Un bit est réservé pour le signe (0 pour + et 1 pour -). En fait cette écriture donnerait 2 représentations pour 0. On utilise donc une notation appelée "en complément à 2", qui conserve la notation précédente pour les entiers n positifs et qui code l'entier strictement négatif n de la même façon que l'entier naturel $n + 2^{31}$ à part le bit de signe.
Par exemple, en travaillant avec 8 bits, le nombre – 125 est représenté par 1000 0011 en complément à 2 alors que sa représentation "usuelle" est : 1111 1101. De même sur 16 bits, le nombre – 20214 est représenté par 1011 0001 0000 1010 en complément à 2 et 1100 1110 1111 0110 en représentation usuelle.

Ainsi les entiers représentables en machine avec n bits sont ceux compris entre -2^{n-1} et $2^n - 1$. Par exemple avec 31 bits, on obtient $-(2^{31})$ et $2^{31} - 1$ donc -2147483648 et 2147483647.
Avec ces contraintes, on peut obtenir des résultats aberrants du type $2147483647 + 1 = -2147483648$
Evidemment avec un processeur 64 bits, on peut gérer les entiers entre $-(2^{63})$ et $2^{63} - 1$.
Certains logiciels gèrent des entiers plus grands mais avec un nombre prédéfini de chiffres (par exemple Excel gère les entiers entre -10^{307} et 10^{307} . Il s'agit du même problème pour les calculatrices qui travaillent avec un nombre fixé de chiffres….
Certains logiciels arrivent à gérer des entiers plus grands en ne fixant pas au départ un nombre d'octets pour représenter un entier… mais encore faut-il conserver une place mémoire indiquant le nombre d'octets utilisés par ce grand entier…

III) Principe de la représentation des nombres réels en machine

Limité par le nombre fini de "décimales" accessibles sur un ordinateur, nous ne pouvons espérer pouvoir représenter tous les nombres réels (même en se limitant à l'intervalle [0, 1] par exemple).
En base b, on peut écrire tout nombre à l'aide d'un développement en base b, éventuellement infini. En effet, on a le résultat suivant :

Théorème : Soit $b \in \mathbb{N}^*$, $b>1$. $\forall r \in \mathbb{R}^+$, $\exists ! p \in \mathbb{N}$, $\exists ! (a_p, a_{p-1}, .., a_0) \in [\![0, b-1]\!]^{p+1}$ et il existe une suite $(c_1, c_2, \ldots, c_n, \ldots)$ d'éléments de $[\![0, b-1]\!]$ non tous égaux à $b-1$ à partir d'un certain rang tels que: $r = a_p b^p + a_{p-1} b^{p-1} + ... + a_1 b^1 + a_0 + c_1 b^{-1} + \ldots + c_n b^{-n} + \ldots$ et $a_p \neq 0$

Remarque : on note alors, en base b, r sous la forme $a_p a_{p-1} \ldots a_0 , c_1 c_2 \ldots c_n \ldots$

Les nombres réels, non nuls, représentables en base b seront écrits sous la forme $r = s \times m \times b^e$ avec s le signe de r, m la mantisse, qui est un nombre "ayant un nombre fini de chiffres après la virgule" vérifiant $1 \leq m < b$ et e un entier relatif. Sous cette forme on a unicité de l'écriture des réels "représentables".
Le nombre de chiffres après la virgule de la mantisse est le nombre de chiffres significatifs.
Cette forme est appelée l'écriture en virgule flottante normalisée.

Si on travaille en base 2, la mantisse est nécessairement de la forme m = 1,….. donc il suffit de connaitre la suite finie des chiffres après la virgule pour connaitre entièrement la mantisse .

Le standard IEEE 754 de 1985 définit 4 formats de nombres à virgule flottante : le moins gourmand en mémoire (flottant à simple précision) est codé avec 32 bits : 1 bit de signe, 8 bits d'exposant (E de 0 à 255 où E = e + biais avec e est l'exposant "réel" et le biais est $2^7 - 1 = 127$) et 23 bits de mantisse (avec un bit 1 implicite) : on ne peut avec cette forme, pas avoir accès aux nombres réels compris entre 0 et 2^{-128} soit un peu plus que 3×10^{-39} ni à ceux plus grands que 2^{128}
La forme du standard IEEE 754 la plus gourmande utilise 80 bits… mais, même si l'étendue des nombres représentables est plus grande, le principe d'inaccessibilité des grands nombres ou des petits reste présent.

Ces limites engendrent des phénomènes de dépassement de capacité ("overflow") et donc des résultats faux ou des erreurs d'arrondis. De même apparaissent des phénomènes d'underflow car la machine ne pourra par exemple pas faire la différence entre un "petit nombre" (10^{-40} par exemple) et 0

IV) Exercice

Numération

1) Ecrire en binaire et en octal (base 8), les nombres qui s'écrivent en base 10 sous la forme 375 et 1237
2) Ecrire en décimal les nombres $1011011_{(2)}$, $375_{(8)}$ et $1237_{(16)}$
3) Calculer les somme, produit et différence des nombres binaires 1011011 et 1010. Que se passe-t-il si les nombres 1011011 et 1010 sont écrits en décimal ?

Représentation des entiers

4) On suppose que les entiers sont codés sur 8 bits, le premier bit correspondant au signe (0 pour + et 1 pour –). Quels sont les résultats des additions : 00101101 + 01101111, 11111111 + 11111111, 00000001 + 11111111, 11110111 + 11101111.
5) Sur votre calculatrice (non formelle), calculer k! et (k+1)!/(k+1) pour toutes les valeurs de k comprises entre 5 et 20.
6) Sur votre calculatrice et sur Open-office Calc, calculer (15!)!

Représentation des réels.

7) Ecrire en binaire, les nombres décimaux : 32,625 et 128,75
8) Quel est le nombre dont l'écriture en virgule flottante (en base 2) est de signe +, de mantisse 1,1000101 et d'exposant 101 ? De même avec la mantisse 1,101 et l'exposant – 11 ?
9) On suppose créé le standard MPSI28 pour l'écriture des nombres flottants sous la forme (S, M, E) avec S un signe (0 si +, 1 pour –), une mantisse M commençant par 1 et possèdant 7 autres bits et E un entier commençant par un bit de signe et 4 bits donnant l'entier E. Tout cela étant écrit en base 2. Le nombre n valant $S * M * 2^E$ sera alors représenté sous 13 bits.
Ecrire sous ce standard les nombres 2,125, 42,535, - 6,22 et - 0,000325
10) Sur votre calculatrice et/ou sur Open Office Calc, calculer $\exp(\pi\sqrt{58}) - 24\,591\,257\,752$. De même avec $\exp(\pi\sqrt{163}) - 262\,537\,412\,640\,768\,744$
11) Calculer sur votre calculatrice et/ou sur Open Office Calc, $\cos(2500000\,\pi)$
12) Calculer $a = 3 - 2\sqrt{2}$ et $b = 3 + 2\sqrt{2}$. Calculer $A = a^{10}$ et $B = b^{10}$. Enfin calculer $A^2 - 45239074\,A$ et $B^2 - 45239074\,B$. Qu'en déduit-on ?