ISN : A1-Codage Numérique

Présentation de l'option ISN, Représentation binaire, Opérations booléennes, Numérisation

Julien Lefevre, Edouard Thiel et Michel Van Caneghem

Janvier 2012

ISN : Informatique et Sciences du Numérique
La réforme du Lycée : s'applique en sept. 2012 pour les terminales.
Les spécialités de terminale S : les 4 actuelles + 1 nouvelle : ISN
- Enseignement : 2 heures hebdomadaires
- Programme : BO spécial n°8 du 13 octobre 2011
- Évaluation : BO spécial n°7 du 6 octobre 2011 épreuve de 20 mn (exposé 8 mn + questions 12 mn)
- Mise en place : BO n°36 du 6 octobre 2011
- Formation des enseignants volontaires réalisée au niveau des académies.

ISN : Contexte historique et objectifs
- L'Informatique a été enseignée précédemment au lycée (1981-91) et (1995-97).
- Elle est présente partout dans nos vies et modifie aussi nos habitudes voire nos capacités cognitives (Nicholas Carr, *Internet rend-il bête ?*).
- Elle est une grande pourvoyeuse d'emploi dans l'industrie, les services.
- Dans la recherche et l'enseignement supérieur également : 2 sections CNRS, un institut dédié (INRIA), plus de 3000 enseignants-chercheurs.
- L'informatique est une science ("Computer Science" en anglais).
- Science jeune, dont les origines remontent aux travaux de Turing et Von Neumann.

Il devient donc tout à fait légitime d'étudier l'Informatique avant le Bac en considérant celle-ci comme **une science à part entière** et non plus comme un outil technologique.

Nécessité de développer une véritable **culture informatique** pour comprendre les enjeux sociétaux.

ISN (1) -- extraits du BO du 13 Octobre 2011
Les sciences informatiques, et plus généralement les sciences du numérique, ont aujourd'hui envahi nos vies professionnelles et personnelles. Elles ont entraîné des mutations profondes dans nos sociétés (culture, sciences, économie, politique …). Pourtant, seule une faible partie de la population maîtrise les mécanismes fondamentaux qui régissent ces mutations et est en mesure d'apprécier les enjeux sociétaux qui en découlent. L'enseignement de l'informatique au lycée peut contribuer à réduire cette fracture.

Remarque 1 : En 1966, en France, l'usage officiel du mot informatique est consacré par l'Académie française pour désigner la science du traitement de l'information (et en particulier l'information numérique).

Remarque 2 : L'utilisation du concept de « Sciences du numérique » peut préter à confusion en laissant penser que c'est une partie des mathématiques : le calcul numérique. Le mot « numérique » est utilisé dans le sens information discrète : par exemple Télévision Numérique Terrestre

ISN (2) -- extraits du BO du 13 Octobre 2011

L'objectif de l'enseignement de spécialité ISN en classe terminale de la série S n'est pas de former des experts en informatique, mais plutôt de fournir aux élèves quelques notions fondamentales et de les sensibiliser aux questions de société induites.

Il s'agit d'un enseignement d'ouverture et de découverte des problématiques actuelles, adapté à la société d'aujourd'hui, qui valorise la créativité et contribue à l'orientation.

Remarque 1 : Attention à ne pas confondre l'informatique et l'utilisation de l'outil informatique (B2I et C2I à l'Université).

Remarque 2 : L'informatique est une science qui s'étudie à l'Université : Il faut trois ans pour obtenir une Licence d'Informatique et deux ans supplémentaires pour obtenir un Master d'Informatique.

Le programme ISN
Le programme pour les lycéens est pensé en quatre parties :

- ✗ Représentation de l'information
- ✗ Algorithmique
- ✗ Langages et programmation
- ✗ Architectures matérielles/Robotique

De façon très simpliste on peut dire : On a une information qui décrit des choses abstraites ou non. On cherche à manipuler cette information, en général à l'aide d'algorithme. Ces algorithmes ont besoin d'être traduits à l'aide d'un langage de programmation qui pourra les rendre effectifs sur une machine.

Pour la formation des futurs-enseignants de la spécialité, nous avons naturellement conservé ce découpage. Les 3 premières parties sont assurés par des enseignants-chercheurs d'informatique et mathématiques de Luminy (48H pour chaque groupe). La dernière par des automaticiens de Saint-Jérôme (13,5H + 6H de projet). Au total on aura donc 67,5H de formation.

La formation ISN à Marseille
Voici le programme :

- A1 - 06/01 - Codage numérique (Michel Van Caneghem)
- A4 - 06/01 - Information structurée 1 (Michel Van Caneghem)
- B1/C1 - 01/02 - Programmation 1/Algorithmique 1 (Fernand Didier)
- B2 - 01/02 - Programmation 2 (Fernand Didier)
- C2 - 15/03 - Algorithmique 2 (Laurent Régnier).
- C3 - 15/03 - Algorithmique 3 (Julien Lefèvre)
- A2 - 27/03 - Représentation numérique des données analogiques (Julien Lefèvre)
- B3 - 27/03 - Programmation 3 (Edouard Thiel)
- A5/B5 - 09/04 - Information structurée 2/Langages formels (Alexis Nasr)
- B4 - 09/04 - Programmation 4 (Edouard Thiel)
- C4 - 17/04 - Algorithmique avancée 1 (Lionel Vaux)
- A3 - 17/04 - Théorie de l'information (Lionel Vaux)
- B6 - 11/05 - Langages du Web (Michel Van Caneghem)
- C5 - 11/05 - Algorithmique avancée 2(Laurent Régnier)
- A6 - 23/05 - Contrôle de l'information (Michel Van Caneghem)
- A7 - 23/05 - Informatique et société (Emmanuel Beffara)

Lectures et documents
Il y a beaucoup d'informations sur internet, nous vous recommandons :

- Bien sûr il y a Wikipedia qui contient très souvent une bonne information en ce qui concerne l'informatique. Attention vous pouvez regardez l'information en différentes langues : ce ne sont pas des traductions
- En ce qui concerne l'option ISN : https ://wiki.inria.fr/sciencinfolycee/Accueil. Ce site contient beaucoup de références liées à ce cours.
- A compléter

En principe les copies de tout ces cours se trouveront sur le site de l'IREM de Marseille : www.irem.univ-mrs.fr

Codage et représentation
Sans parler d'ordinateurs comment a-t-on représenté les nombres ?
Quelques exemples historiques :
- les batons
- Systémes additifs : Numération égyptienne en base 10

Le nombre 14 s'écrira donc ou
- Systémes positionnels : Numération mésopotamienne en base 60

On voit déjà un premier niveau d'abstraction entre un dessin ou des sons et un nombre

Codage et représentation - la numération de position

Ecriture en base b

Tout entier positif x s'écrit dans n'importe quelle base $b \in \mathbb{N} \setminus \{0,1\}$:

$$x = \sum_{i=1}^{n} x_i b^i$$

avec $x_n \neq 0$, $0 \leq x_i < b$, $n = \lfloor \log_b(x) \rfloor + 1$. x est noté $x_n x_{n-1} ... x_0$

- ✍ Représentation pour facilité le calcul, mais est-ce que l'ordinateur en a besoin ?
- ✍ Les chiffres -- la forme graphique (dessin) des chiffres
- ✍ les nombres
- ✍ les touches et le clavier de l'ordinateur
- ✍ le codage des caractères
- ✍ le format interne et les opérations
- ✍ sortie en convertissant les nombres en chaîne de caractères
- ✍ dessin des caractères sur l'écran ou le papier

Le plan

- ✲ Représentation binaire : Un ordinateur est une machine qui manipule des valeurs numériques représentées sous forme binaire. Capacité : Manipuler à l'aide d'opérations élémentaires les trois unités de base : bit, octet, mot. Observations : On met en évidence, sous forme de questionnement, la présence du numérique dans la vie personnelle et professionnelle, au travers d'exemples.
- ✲ Opérations booléennes : Présentation des opérations booléennes de base (ET, OU, NON, OU-EXCLUSIF). Capacité : Exprimer des opérations logiques simples par combinaison d'opérateurs de base. Observations : On découvre les opérations logiques de base à l'aide d'exercices simples, et on met en évidence ces opérations dans les mécanismes de recherche. En parallèle avec les séances d'algorithmique, on peut expliquer le principe d'addition de deux octets.
- ✲ Numérisation : L'ordinateur manipule uniquement des valeurs numériques. Une étape de numérisation des objets du monde physique est donc indispensable. Capacité : Coder un nombre, un caractère au travers d'un code standard, un texte sous forme d'une liste de valeurs numériques.

Représentation binaire - Le bit

Quantité élémentaire d'information (voir le cours sur Théorie de l'information) : le bit (BInary digiT).
Deux valeurs : 0 ou 1. Selon le contexte, peuvent correspondre à :

- ✓ nombres 0 ou 1 (numérique),
- ✓ faux ou vrai (logique),
- ✓ ouvert ou fermé (interrupteur),
- ✓ nord ou sud (magnétique),
- ✓ noir ou blanc (optique),
- ✓ absence ou présence de trou (carte perforée),
- ✓ pile ou face
- ✓ pair ou impair

Remarque : La nature a choisi une autre base : la base 4 pour le codage de l'ADN (4 bases A T C G : que l'on peut donc représenter avec 2 bits)

Représentation binaire - Les nombres modulo 2

ou Z/2Z ou GF(2) [Corps de Galois]. On peut aussi parler des nombres pairs et impairs.

les opérations :

x	y	$x+y$	xy	$x+y+xy$
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	1

De belles propriétés :

$$x + x = 0 \qquad xx = x$$

Nous allons voir par la suite que l'addition correspond ou « ou-exclusif » en logique et la multiplication correspond au « et ».

Représentation binaire - Les octets

Pour des raisons pratiques, les bits sont regroupés par paquets adjacents pour représenter de l'information : Un octet (byte) est constitué de 8 bits (il peut représenter 256 valeurs). Historiquement cela correspondait au codage d'un caractère. L'octet est choisi comme unité pour représenter les capacités mémoire.

1	1	0	1	0	1	1	0
7	6	5	4	3	2	1	0

Si l'on numérote les bits de 0 à 7, le bit numéro 0 est le bit de poids faible, et le bit numéro 7 est le bit de poids fort.

Si l'on considère un octet comme un nombre écrit en base 2, sa valeur numérique est

$$\sum_{i=0}^{7} b_i \, 2^i$$

Ici, on a $11010110_2 = 2^7 + 2^6 + 2^4 + 2^2 + 2^1 = 214_{10}$

Base 16

La base 2 est peu pratique à écrire on utilise souvent la base 16 les nombres hexadécimaux. Les chiffres sont :

0	0000	0	4	0100	4	8	1000	8	C	1100	12
1	0001	1	5	0101	5	9	1001	9	D	1101	13
2	0010	2	6	0110	6	A	1010	10	E	1110	14
3	0011	3	7	0111	7	B	1011	11	F	1111	15

Pour convertir de base 2 en base 16, il suffit de faire des groupes de 4 bits.

1	1	0	1	0	1	1	0
D				6			

11010110_2 s'écrit donc D6 en hexadécimal.

Représentation binaire - Les unités

Standard SI		Unités binaires	
Unité	Valeur	Unité	Valeur
kilobit (kb)	10^3	kibibit (Kibit)	2^{10}
mégabit (Mb)	10^6	mébibit (Mibit)	2^{20}
gigabit (Gb)	10^9	gibibit (Gibit)	2^{30}
térabit (Tb)	10^{12}	tébibit (Tibit)	2^{40}
pétabit (Pb)	10^{15}	pébibit (Pibit)	2^{50}
kilooctet (ko)	10^3	kibioctet (Kio)	2^{10}
mégaoctet (Mo)	10^6	mébioctet (Mio)	2^{20}
gigaoctet (Go)	10^9	gibioctet (Gio)	2^{30}
téraoctet (To)	10^{12}	tébioctet (Tio)	2^{40}
pétaoctet (Po)	10^{15}	pébioctet (Pio)	2^{50}
exaoctet (Eo)	10^{18}	exbioctet (Eio)	2^{60}

Usage traditionnel $\neq$ notations officielles : 1 ko = 1024 octets ... (Calculez le pourcentage d'écart entre les deux unités)

Quelle taille pour une image

Imaginons une image A4 (600 points au pouces : pouvoir séparateur de l'œil).

- 1pouce = 2.54cm, donc 600 points au pouces = 558 points au mm2
- Une page A4 = 625cm2 = 625 x 55800 = 34 875 000 points
- Imaginons les couleurs codées sur 4 octets (32 bits), on obtient une taille de fichier de $139,5 \times 10^6$ octets = 133 Moctets

Avec compression (JPEG) une photo (10 Mpixel) ==> 4 Mo (?)

Quelle taille pour un disque audio
Prenons par exemple le format d'un CD audio :

- Stéréo, Echantillonage 44,1 Khz, Numérisation 16 bits
- 1 seconde de son = 2 x 44100 x 16 = 75 x 2352 octets = 176400 octets. (1 seconde = 75 secteurs)
- 1 minute de son = 176 400 x 60 = 10 584 000 octets = 10,3 Moctets
- 1h de son = 635×10^6 octets = 605 Moctets

Remarque : CD-ROM, il y a un codage supplémentaire : 1 secteur = 2048 Octets . Un CD-ROM de 80 Minutes = 80 x 60 x 75 x 2 Koctets = 720 000 Koctets = 703 Moctets

Avec compression (MP3) un CD ==> 60 Mo (?)

Quelle taille pour un film
Pour simplifier, si on utilise les standards de Télévision numérique (4/3) :

- 1 image = 720 x 576 x 16 = 0,8 Moctets
- 25 images par secondes : 1s = 0,8 x 25 = 20 Moctets/seconde
- 1 film de 90 minutes : 90 x 60 x 20 Moctets = 108 000 Moctets = 108 Goctets

La capacité d'un DVD est de : en fonction du nombre de faces et de couches de 4,7 a 17 Go de données soit l'équivalent de 6 CDroms. Un disque Blu-ray double couche contient 50 Go

Avec compression un film d'une heure occupe entre 500 Mo et 10 Go selon la qualité

Représentation binaire - Une question
La BNF pour son dépôt légal de musique, a une collection de 350 000 disques microsillions et 150 000 CD. On peut supposer que cela représente environ l'équivalent de 500 000 CD d'une heure.

1. Quel est le volume de donnée nécessaire pour stocker toute la musique éditée en France en utilisant la compression MP3. Vous exprimerez ce volume en Teraoctets : 1 To = 1000 Go. $500000 \times 60Mo = 30000000Mo = 30000Go = 30To$
2. Un ordinateur actuel a un disque dur de 1 To. On suppose que la capacité des disques durs double tous les 9 mois. Dans combien de temps pourrez vous avoir toute la musique éditée en France sur votre ordinateur ? [$2^5 = 32$]
3. Aujourd'hui 1To coute 60€ [actuellement 100€ ! !] (30 x 60 = 1800 €).

Représentation binaire - Les mots
Un mot est la quantité de bits standard manipulée par un microprocesseur (CPU). La taille du mot s'exprime en bits ou en octets.

Un microprocesseur est d'autant plus performant que ses mots sont longs, car les données qu'il traite à chaque cycle sont plus nombreuses. C'est pourquoi on classe les microprocesseurs par la taille de leur mot (16, 32, 64 bits : 2, 4 ou 8 octets).

Valeur max d'un mot :

$2^{16} - 1 = 65\,535 =$ 0xFFFF
$2^{32} - 1 = 4\,294\,967\,295 =$ 0xFFFF FFFF
$2^{64} - 1 = 18\,446\,744\,073\,709\,551\,615 =$ 0xFFFF FFFF FFFF FFFF

Opérations booléennes - Opérateurs booléens

• Algèbre de Boole (George Boole, milieu 19e siècle) :
- partie des mathématiques qui s'intéresse aux opérations et aux fonctions sur les variables logiques.
- En logique, domaine définissant les lois formelles du raisonnement, utilisée pour le calcul des propositions ($\wedge, \vee, \neg, \exists, \forall, \Rightarrow, =$) ;
- utilisée dans la conception des circuits électroniques.

Opérations booléennes - Opérateurs booléens
Opérateurs booléens : not (négation), and (et), or (ou), xor (ou exclusif).

Tables de vérité :

x	not x
false	true
true	false

x	y	x and y	x or y	x xor y
false	false	false	false	false
false	true	false	true	true
true	false	false	true	true
true	true	true	true	false

Numériquement, avec false = 0 et true = 1 :

$$\neg x = 1 - x$$
$$x \wedge y = x * y$$
$$x \oplus y = x + y \pmod 2$$
$$x \vee y = x + y + xy \pmod 2$$

Opérations booléennes - Propriétés
Lois de De Morgan (milieu 19e siècle) :

$$\text{not } (x \text{ or } y) = (\text{not } x) \text{ and } (\text{not } y)$$
$$\text{not } (x \text{ and } y) = (\text{not } x) \text{ or } (\text{not } y)$$

Négation des inégalités :

$$\text{not } (x < y) = x \geq y \quad ; \quad \text{not } (x \leq y) = x > y$$
$$\text{not } (x > y) = x \leq y \quad ; \quad \text{not } (x \geq y) = x < y$$

Exercices : simplifier $(x < 7)$ and $(y < 3)$ or not $((x \geq 7)$ or $(y < 3))$ not (x and (not x or y)) or y

Numérisation - Codage des entiers
Il s'agit simplement d'un codage en base 2. On a le choix de la taille n des entiers manipulés, souvent lié à la taille des mots machines (Attention ce n'est pas le cas en Python)

$$\sum_{i=0}^{n-1} b_i \, 2^i$$

On ne peut faire aucune hypoyhèse sur l'ordre des bits (poids fort à gauche ou a droite) cela dépend du processeur.

- 8 bits : $[0..2^8 - 1]$ = 0..255
- 16 bits : $[0..2^{16} - 1]$ = 0..65 535
- 32 bits : $[0..2^{32} - 1]$ = 0..4 294 967 295 (9 chiffres)
- 64 bits : $[0..2^{64} - 1]$ = 0..18 446 744 073 709 551 615 (19 chiffres]

Remarque : les opérations sur les entiers peuvent provoquer des débordements de capacité. Soit signalés par des erreurs, soit tout simplement en enlevant ce qui est en trop ! !

Les entiers signés
Il y a au moins deux possibilités :

- Un bit de signe (il y a alors 2 zéros : 0 positif et 0 négatif ! ! ! et nécessite une opération spéciale)
- Le complément à deux ($-x \Rightarrow 2^n - x$)

Un exemple sur 4 bits :

0000	0	0100	4	1000	-8	1100	-4
0001	1	0101	5	1001	-7	1101	-3
0010	2	0110	6	1010	-6	1110	-2
0011	3	0111	7	1011	-5	1111	-1

- Les nombres négatifs commencent par 1
- Il y a plus de nombres négatifs que de nombres positifs !

Les grands nombres
Pour calculer sur de grands nombres on utilise la représentation dans la base b, avec b souvent égal à la la taille du mot machine (Ex : $b = 2^{32}$: dans ce cas il y a : 4 294 967 296 chiffres ! ! !).

$$x = x_n b^n + x_{n-1} b^{n-1} \cdots + x_1 b + x_0 = \sum_{i=0}^{n} x_i b^i$$

On profite alors des opérations de la machine pour obtenir les tables d'addition et de multiplication dans la base b.

Il n'y a plus de problèmes de débordement, mais les performances deviennent alors très mauvaises. Même si on utilise d'autres méthodes que la méthode classique pour faire des multiplications.

Cela correspond maintenant au type int dans la version 3 du langage Python.

Utilisation des grands entiers
Exemples :
- Cryptographie : fabrication de grands nombres premiers (tests de primalité probabilistes de Solovay-Strassen et de Miller-Rabin).
- Cryptanalyse : factorisation de grands nombres premiers.
- Calcul des décimale de π.
- Calcul des décimales de $\sqrt{2}$.

le calcul de π

Babylonians	2000? BCE	1	3.125 = 3 + 1/8
Egyptians	2000? BCE	1	3.16045
China	1200? BCE	1	3
Bible (1 Kings 7 :23)	550? BCE	1	3
Archimedes	250? BCE	3	3.1418 (ave.)
Hon Han Shu	130 AD	1	3.1622 = sqrt(10)?
Ptolemy	150	3	3.14166
Tsu Ch'ung Chi	480?	7	3.1415926
Aryabhata	499	4	3.14156
Brahmagupta	640?	1	3.162277 = sqrt(10)
Al-Khowarizmi	800	4	3.1416
Fibonacci	1220	3	3.141818
Al-Kashi	1429	14	
Viete	1593	9	3.1415926536 (ave.)
Van Ceulen	1615	35	
Newton	1665	16	
Machin	1706	100	
De Lagny	1719	127	(112 correct)
Rutherford	1824	208	(152 correct)
Shanks	1874	707	(527 correct)

le calcul de π (2)

Ferguson	1946	620
Smith and Wrench	1949	1,120
Reitwiesner et al. (ENIAC)	1949	2,037
Nicholson and Jeenel	1954	3,092
Genuys	Jan. 1958	10,000
Shanks and Wrench	1961	100,265
Guilloud and Dichampt	1967	500,000
Guilloud and Bouyer	1973	1,001,250
Kanada, Yoshino and Tamura	1982	16,777,206
Ushiro and Kanada	Oct. 1983	10,013,395
Kanada, Tamura, Kubo et al	Jan. 1987	134,217,700
Chudnovskys	Aug. 1991	2,260,000,000
Takahashi and Kanada	Oct. 1995	6,442,450,938
Kanada	Dec. 2002	1,241,100,000,000

Deux calculs pendant près de 600 heures sur un HITACHI SR8000/MP doté de 1TB de stockage (1024Go), et basés sur deux formules indépendantes de type Machin, ont généré 1,241,100,000 décimales.

le calcul de π (3)
Une petite réflexion :

- En 1949 on obtient 2000 décimales en 70h (on aurait obtenu 6000 chiffre en 600 heures)
- En 2002 on obtient 1 241 100 000 000 décimales en 600h

En 53 ans on a fait un progrès de 200000000 décimales. Mais le calcul étant quadratique il faut plutôt voir un progrès de 10^{16} Pourquoi ?

- ➜ Le progrès des machines : $2 * 10^6 = 2^{21}$
- ➜ Le progrès des algorithmes de calcul = 100000000 ($n^2 \Rightarrow n \log n$)
- ➜ Le progrès des méthodes de calcul de π = 50 (???)

$\sqrt{2}$ avec 10 000 décimales
On va utiliser la formule classique :

$$x_{n+1} = \frac{1}{2}(x_n + \frac{2}{x_n})$$

Convergence rapide (quadratique). Si $x_n = \sqrt{2} + \epsilon_n$ alors :

$$\epsilon_{n+1} = \frac{1}{2}\frac{\epsilon_n^2}{\sqrt{2} + \epsilon_n}$$

En fait on calcule $u_n = 10^{10000} x_n$. La formule s'écrit alors :

$$u_{n+1} = \frac{1}{2}(u_n + \frac{2 \times 10^{20000}}{u_n})$$

On peut prendre $u_0 = 1,414213562373095 \times 10^{10000}$ et dans ce cas 14 itérations suffisent.

$\sqrt{2}$ avec 10 000 décimales (2)
Voici le programme en Python (les entiers n'ont pas de limite). Si on met '/' pour la division, alors il s'agit de la division entre flottants.

le programme en Python

```
from math import *

def rac2(ndec):
    u = 10**ndec
    deux = 2*10**(2*ndec)
    for i in range(14) :
        u = (u + deux//u)//2
    return u

print (rac2(10000))
14142135623730950488016887242096980785696718753769480731766797...
```

résultat immediat

Les rationnels
Codé comme un couple d'entier.

- ★ Problème des fractions irréductibles : à chaque calcul il faut simplifier à l'aide du pgcd.
- ★ Difficulté du calcul du à la croissance des nombres dans des opérations simples.

On peut représenter les rationnels, par la suite des chiffres décimaux.
Si le nombre est rationnel, alors cette suite est périodique. Suivant la base, la suite des chiffre décimaux est finie ou infinie.

- ★ 1/2 = 0,5
- ★ 1/7 = 0,142857 142857 142857 142857,....

Problème : 2 et 1,99999999999... sont identiques.

Un nombre étrange qui n'est pas rationnel : $0,1234567891011121314....$

Les réels et les flottants

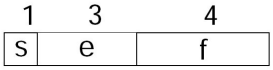
Bien sur on ne peut pas coder les réels. Il faut donc découper $\mathbb{R}$ en une suite finie d'intervalles. Par exemple :

$$[0],]0..1[, [1],]1.. + \infty[$$

et on va travailler avec ces intervalles. Pour les besoins du calcul numériques, il faut beaucoup de nombre autour de 0.

Les flottants sont un ensemble fini de rationnels (de la forme $a/2^n$), qui sont les bornes de ces intervalles. Ils sont choisis de telle manière qu'il y ait autant de flottants 0 et 1 qu'entre 1 et 2, qu'entre 2 et 4, etc :

Les flottants --- Un exemple



exposant e	mantisse f	valeur v	type de v
$0 < e < 7$		$v = (-1)^s \times 2^{(e-3)} \times (1.f)$	Normalisé
$e = 0$	$f \neq 0$	$v = (-1)^s \times 2^{(-2)} \times (0.f)$	Dénormalisé
$e = 0$	$f = 0$	$v = (-1)^s \times 0$	Zéro
$e = 7$	$f = 0$	$v = (-1)^s \times infini$	Infini
$e = 7$	$f \neq 0$	v n'est pas un nombre	NaN

	valeur exacte	valeur décimale
Nombre maximum :	$2^4 - 2^{-1}$	15.5
Nombre minimum (normalisé) :	2^{-2}	0.25
Nombre minimum (dénormalisé) :	2^{-6}	0.015625
Entier maximum :	$2^4 - 1$	15

Un exemple pédagogique(2)

f	$e = 0$	$e = 1$	$e = 2$	$e = 3$	$e = 4$	$e = 5$	$e = 6$	$e = 7$
0	0	16/64	16/32	1	2	4	8	$+\infty$
1	1/64	17/64	17/32	17/16	17/8	17/4	17/2	NaN
2	2/64	18/64	18/32	18/16	18/8	18/4	9	NaN
3	3/64	19/64	19/32	19/16	19/8	19/4	19/2	NaN
4	4/64	20/64	20/32	20/16	20/8	5	10	NaN
5	5/64	21/64	21/32	21/16	21/8	21/4	21/2	NaN
6	6/64	22/64	22/32	22/16	22/8	22/4	11	NaN
7	7/64	23/64	23/32	23/16	23/8	23/4	23/2	NaN
8	8/64	24/64	24/32	24/16	3	6	12	NaN
9	9/64	25/64	25/32	25/16	25/8	25/4	25/2	NaN
10	10/64	26/64	26/32	26/16	26/8	26/4	13	NaN
11	11/64	27/64	27/32	27/16	27/8	27/4	27/2	NaN
12	12/64	28/64	28/32	28/16	28/8	7	14	NaN
13	13/64	29/64	29/32	29/16	29/8	29/4	29/2	NaN
14	14/64	30/64	30/32	30/16	30/8	30/4	15	NaN
15	15/64	31/64	31/32	31/16	31/8	31/4	31/2	NaN

IEEE single

exposant e	mant. f	valeur v	type de v
$0 < e < 255$		$v = (-1)^s \times 2^{(e-127)} \times (1.f)$	Normalisé
$e = 0$	$f \neq 0$	$v = (-1)^s \times 2^{(-126)} \times (0.f)$	Dénormalisé
$e = 0$	$f = 0$	$v = (-1)^s \times 0$	Zéro
$e = 255$	$f = 0$	$v = (-1)^s \times infini$	Infini
$e = 255$	$f \neq 0$	v n'est pas un nombre	NaN

	valeur exacte	valeur décimale
Nb maximum :	$2^{128} - 2^{104}$	3.402823466E+38
Nb minimum (normalisé) :	2^{-126}	1.175494351E-38
Nb minimum (dénormalisé) :	2^{-149}	1.401298464E-45
Ent. maximum :	$2^{24} - 1$	16 777 215

Pour réfléchir
On veut calculer u_{100} avec :

$$u_n = 4u_{n-1} - 1 \qquad u_0 = 1/3$$

le programme en Python

```
def recurrence(n):
    u0 = 1/3
    u1 = 0
    for i in range(n) :
        u1 = 4*u0 - 1
        u0 = u1
    return u1

print (recurrence(100))
```

On trouve : -2.9734326931374163e+43 : Pourquoi ?

Le codage des caractères
- Le code ASCII/ISO 646
- Le code ISO 8859-1/ISO 8859-15 code actuel
- Le code Unicode
- Le code UTF-8/UTF-16 mieux ?

ASCII/ISO 646

- ASCII (American Standard Code for Information Interchange), Bob Bemer en 1961. C'est un code sur 7 bits, le premier bit étant 0 (Dans le temps on utilisait ce bit comme bit de parité).
- Les caractères de 0 à 31 ainsi que le 127 ne sont pas affichables, et correspondent à des directives de terminal. Le caractère 32 est l'espace blanc. Les autres correspondent aux chiffres, aux lettres majuscules et minuscules et à quelques symboles de ponctuation. Problème LF/CR.
- ASCII : norme américaine, la norme officielle en France ISO 646 (pas de variantes nationales).
- Problèmes : Pas d'accents et de nombreuses variantes pour utiliser le 8ème bit (Mac, EBDIC,...)

ISO 8859-1

- L'ISO 8859-1 [Ou Latin-1] (1992) recouvre les caractères utilisés par les langues européennes suivantes : albanais, allemand, anglais, basque, catalan, danois, gaélique écossais, espagnol, féringien, finnois, français. Code sur 8 bits.
- Code 191 caractères. Mais les caractères suivants ont été oubliés : œ, Œ et Ÿ. Et bien sûr il manquait le caractère € de l'euro ! !. Pour corriger cela il y a maintenant le code ISO 8859-15 : *c'est celui qui est utilisé de manière courante dans la plupart des ordinateurs actuels*.
- Mais il fallait tout une collection de codes pour chaque groupe de pays, par exemple ISO 8859-2 (latin-2 ou européen central), ISO 8859-7 (grec), ... Ne parlons pas du Japon et de la Chine ! !
- En 2004, le groupe de normalisation a décidé d'abandonner ce code au profit de l'Unicode et de l'UTF-8.

ISO 8859-15

ISO 8859-15																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
Ax	NBSP	¡	¢	£	€	¥	Š	§	š	©	ª	«	¬		®	¯
Bx	°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Unicode

- Unicode, dont la première publication remonte à 1991, a été développé dans le but de remplacer l'utilisation de pages de code nationales.
- Jeu de caractères abstraits : La couche la plus élevée est la définition du jeu de caractères. Par exemple, Latin-1 a un jeu de 256 caractères et Unicode normalise actuellement près de 100 000 caractères.
- Jeu de caractères codés : on ajoute à la table précédente un index numérique. Notons bien qu'il ne s'agit pas d'une représentation en mémoire, juste d'un nombre (de 4 à 6 caractères hexadécimaux).
- Exemples :
  - é : *latin small letter e with acute* (hexa = E9) &#233 ; or &eacute ; [ISO 8859-15 : E9]
  - œ : *latin small ligature oe* -- (hexa = 153) &#339 ; or &oelig ; [ISO 8859-15 : BD]

UTF-8

Codage des caractères Unicode sur plusieurs octets. (Il existe d'autres formats comme UTF-16).

0vvvvvvv	1 octet codant 1 à 7 bits
110vvvvv 10vvvvvv	2 octets codant 8 à 11 bits
1110vvvv 10vvvvvv 10vvvvvv	3 octets codant 12 à 16 bits

Quelques exemples :

é	E9	1110 1001	11000011 10101001
œ	153	1 0101 0011	11000101 10010011
€	20AC	10 0000 1010 1100	11100010 10000010 10101100

- On ne peut plus déduire la longueur d'une chaîne, en comptant le nombre d'octets.
- Il faut connaître le type de codage d'un texte.

Un exemple

Indication du codage : entête XML/HTML

```
<meta http-equiv="content-type" content="text-html ;
      charset=UTF-8">
<?xml version="1.0" encoding="UTF-8" ?>
```

Regardons différents codages du mot œuvrée :

HTML	&#339 ;uvr&eacute ;e
ISO 8859-15	BD 75 76 72 E9 65
UTF-8	C5 93 75 76 72 C3 A9 65
MIME	xZN1dnLDqWU=

MIME

Multipurpose Internet Mail Extensions (MIME) est un format de données permettant d'introduire dans les messages SMTP (courriels) différents types de fichiers multimédias (et en particulier des textes)

On peut utiliser base64 qui est un codage de l'information utilisant 64 caractères : (lettres majuscules, lettres minuscules,chiffres et + et /).

- ✗ On organise la suite d'octets par tranches de trois octets (24 bits)
- ✗ Chaque groupes de 3 octets est traduit par 4 caractères

C'est ainsi que l'on code par exemple les images dans les mails (attention augmentation de la taille par un facteur 4/3), mais on peut aussi coder des textes en ISO 8859 ou UTF-8. Cela permet d'éviter des transcodages dans les différentes manipulations des mails.

Conclusion : interprétations du même mot

A partir du mot :

11000101100100110111010101110110011100101100001110101001011001 01

On peut l'interpréter comme :

- hexadécimal : c593757672c3a965
- entier : 14236851998640286053
- entier signé : -4209892075069265563
- flottant : -1.5055547159432955 E27
- UTF8 : œuvrée
- ISO 8859-15 : Å¤uvrÃ©e

Conclusion : une question
On vient de recevoir le message suivant d'une race d'extra-terrestres :

00110000011000111111110110010011001001100101011100 01001000100100010010011001 10

Pourquoi peut-on supposer que ce message a été envoyé par des humanoïdes qui ont un bras deux fois plus long l'un que l'autre ?