DIU-EIL/Bloc1

Numériser, coder, chercher

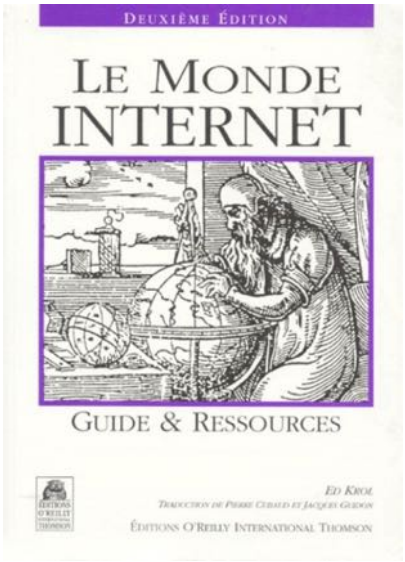
Pierre Cubaud
cubaud @ cnam.fr

juin 2019

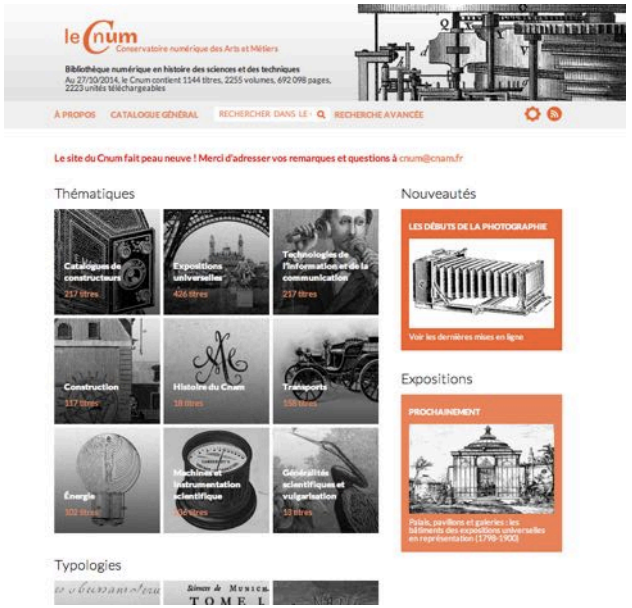
le cnam

Qui suis-je ?

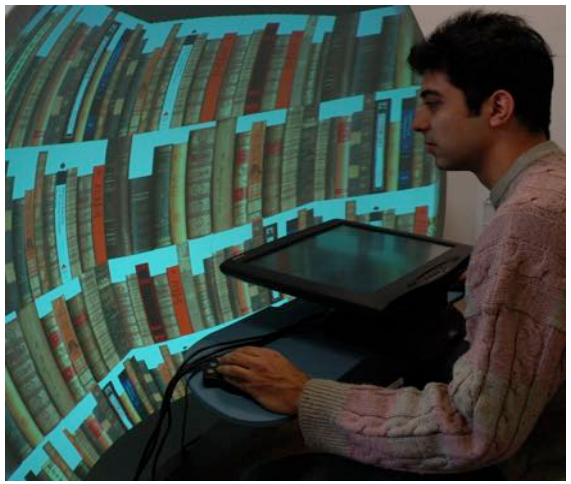
site ABU, 1993



1995



site CNUM, 2000



Interaction 3D

Rouleau de lecture

MOOC médias interactifs, 2014

Prof. du contingent : informatique au Lycée militaire d'Autun (1987-88)

TurboPascal, MSDOS 3, Leanord, Science & Vie Micro, etc.

```
PROGRAM JEU;

VAR NOMBRE_MYSTERIEUX,NOMBRE_PROPOSE,NOMBRE_ESSAIS:INTEGER;

BEGIN
     RANDOMIZE;    (* pour initialiser la generation *)
                   (* de nombres aleatoires          *)

     NOMBRE_MYSTERIEUX:=RANDOM(1000);

     WRITELN('-------- JEU DU NOMBRE MYSTERIEUX -------');
     WRITELN(' Vous devez trouver quel est le nombre ');
     WRITELN(' compris entre 0 et 1000 qui a été choisit ');
     WRITELN(' Vous avez droit à 10 essais....');
     WRITELN;

     NOMBRE_ESSAIS:=1;

     REPEAT
           WRITE('..... ESSAI N° ',NOMBRE_ESSAIS,' .....');
           WRITE('Votre choix ? : ');
           READLN(NOMBRE_PROPOSE);
           IF NOMBRE_PROPOSE<NOMBRE_MYSTERIEUX THEN WRITELN(' ++++ plus ++++ ');
           IF NOMBRE_PROPOSE>NOMBRE_MYSTERIEUX THEN WRITELN(' ---- moins --- ');
           NOMBRE_ESSAIS:=NOMBRE_ESSAIS+1
     UNTIL (NOMBRE_ESSAIS>10) OR (NOMBRE_PROPOSE=NOMBRE_MYSTERIEUX);

     IF NOMBRE_PROPOSE=NOMBRE_MYSTERIEUX THEN WRITELN(' !!!! GAGNE !!!! ')
                                         ELSE WRITELN(' ???? PERDU ???? ')
END.
```

Enseignement complémentaire d'informatique en lycées

R.L.R. : 524-5 *

Note de service n° 88-084 du 1er avril 1988

(Education nationale : bureaux DLC 3 et DLC 15)

Texte adressé aux recteurs.

L'enseignement optionnel complémentaire d'informatique, introduit réglementairement en classe de seconde à la rentrée de l'année scolaire 1985/1986, en classe de première à la rentrée de l'année scolaire 1986/1987 et en classe de terminale à la rentrée de l'année scolaire 1987/1988, sera sanctionné pour la première fois par une épreuve au baccalauréat dès la session de 1988.

Les objectifs généraux de cet enseignement ont été publiés dans l'annexe I de l'arrêté du 31 mai 1985 paru

Relire le programme de 1985 :
https://edutice.archives-ouvertes.fr/file/index/docid/30732/filename/b46p032.pdf

Organisation de la séance

- Exposés => 2h max

- la numérisation de l'information
- codes élémentaires
- formats de fichiers
- la compression
- chercher / trier

des moments d'échanges ?

- Travaux pratiques => 4h min

  nombre d'exercices à faire selon niveau stagiaire

  m'en rendre deux à la fin via le moodle

  + si besoin/envie :
  continuer en ligne la semaine prochaine

énoncés sur feuille séparée

Place dans le programme du DIU-bloc1

Bloc 1 : Représentation des données et programmation

Objectifs de formation

L'étude des représentations de l'information abordée conjointement avec celle des concepts fondamentaux des langages de programmation, a pour objectif d'outiller l'enseignant dans son choix des activités de programmation à proposer aux élèves, en lui donnant une bonne maîtrise des types de données et des méthodes de programmation. Cela permet d'aborder aussi la didactique de la programmation.

Connaissances préalables

Les thèmes abordés dans ce bloc sont largement présents dans le programme de la spécialité ISN de terminale S. Les notions de base sur la représentation de l'information – codage des nombres entiers, unités –, la programmation impérative – structures de contrôle et types de base – ainsi que les savoir-faire associés à la programmation – usage d'un éditeur, exécution d'un programme, utilisation d'un interpréteur, tests – sont supposés acquis.

Contenu de la formation

Représentation de l'information

- Codage des nombres flottants
- Fichiers et formats usuels, compression et archivage

matin

Langages et programmation

- Types structurés, p-uplets, tableaux et dictionnaires
- Traitement de données en tables (recherche, tris, fusion)
- Modularité, bibliothèques
- Diversité des langages de programmation
- Langages de description de pages web : HTML, CSS
- Programmation web côté client : Javascript
- Gestion des événements dans une interface web
- Spécification, prototypage et tests

après-midi

Didactique de l'informatique

- Pensée informatique et compétences associées
- Approche instrumentale, approche ergonomique, psychologie de la programmation
- Liens avec les didactiques des mathématiques : théorie des situations, transposition

un peu

Alignement sur le programme NSI de 1ère

- Représentation des données : types et valeurs de base, types construits
- Traitement de données en tables
- Langages et programmation
- Interactions entre l'homme et la machine sur le Web

Dans le programme SNI de première

Représentation des données : types et valeurs de base

Toute machine informatique manipule une représentation des données dont l'unité minimale est le bit 0/1, ce qui permet d'unifier logique et calcul. Les données de base sont représentées selon un codage dépendant de leur nature : entiers, flottants, caractères et chaînes de caractères. Le codage conditionne la taille des différentes valeurs en mémoire.

Contenus	Capacités attendues	Commentaires
Écriture d'un entier positif dans une base $b \geqslant 2$	Passer de la représentation d'une base dans une autre.	Les bases 2, 10 et 16 sont privilégiées.
Représentation binaire d'un entier relatif	Évaluer le nombre de bits nécessaires à l'écriture en base 2 d'un entier, de la somme ou du produit de deux nombres entiers. Utiliser le complément à 2.	Il s'agit de décrire les tailles courantes des entiers (8, 16, 32 ou 64 bits). Il est possible d'évoquer la représentation des entiers de taille arbitraire de Python.
Représentation approximative des nombres réels : notion de nombre flottant	Calculer sur quelques exemples la représentation de nombres réels : 0.1, 0.25 ou 1/3.	0.2 + 0.1 n'est pas égal à 0.3. Il faut éviter de tester l'égalité de deux flottants. Aucune connaissance précise de la norme IEEE-754 n'est exigible.
Valeurs booléennes : 0, 1. Opérateurs booléens : and, or, not. Expressions booléennes	Dresser la table d'une expression booléenne.	Le ou exclusif (xor) est évoqué. Quelques applications directes comme l'addition binaire sont présentées. L'attention des élèves est attirée sur le caractère séquentiel de certains opérateurs booléens.
Représentation d'un texte en machine. Exemples des encodages ASCII, ISO-8859-1, Unicode	Identifier l'intérêt des différents systèmes d'encodage. Convertir un fichier texte dans différents formats d'encodage.	Aucune connaissance précise des normes d'encodage n'est exigible.

Traitement de données en tables

Les données organisées en table correspondent à une liste de p-uplets nommés qui partagent les mêmes descripteurs. La mobilisation de ce type de structure de données permet de préparer les élèves à aborder la notion de base de données qui ne sera présentée qu'en classe terminale. Il s'agit d'utiliser un tableau doublement indexé ou un tableau de p-uplets, dans un langage de programmation ordinaire et non dans un système de gestion de bases de données.

Contenus	Capacités attendues	Commentaires
Indexation de tables	Importer une table depuis un fichier texte tabulé ou un fichier CSV.	Est utilisé un tableau doublement indexé ou un tableau de p-uplets qui partagent les mêmes descripteurs.
Recherche dans une table	Rechercher les lignes d'une table vérifiant des critères exprimés en logique propositionnelle.	La recherche de doublons, les tests de cohérence d'une table sont présentés.
Tri d'une table	Trier une table suivant une colonne.	Une fonction de tri intégrée au système ou à une bibliothèque peut être utilisée.
Fusion de tables	Construire une nouvelle table en combinant les données de deux tables.	La notion de domaine de valeurs est mise en évidence.

Par rapport à l'ex option ISN de Terminale S :

Les capacités de traitement et de stockage des ordinateurs croissent de façon continue depuis leur apparition. Il est donc crucial d'organiser ces flux d'informations en local sur une machine ou de façon distribuée sur un réseau.

L'intégration croissante du numérique dans les activités humaines et la numérisation de l'information suscitent des transformations culturelles, socio-économiques, juridiques et politiques profondes qui font apparaître de nouvelles opportunités, de nouveaux risques et de nouvelles contraintes qu'il convient d'étudier.

SAVOIRS	CAPACITÉS	OBSERVATIONS
Numérisation L'ordinateur manipule uniquement des valeurs numériques. L'élève comprend qu'une étape de numérisation des paramètres associés aux objets du monde physique est donc indispensable.	**Coder** un nombre, un caractère au travers d'un code standard, un texte sous forme d'une liste de valeurs numériques.	Il est ici utile de faire référence à des notions technologiques introduites à propos des architectures matérielles. Les images et les sons peuvent être choisis comme contexte applicatif et sont manipulés via des logiciels de traitement ou de synthèse.
Représentation binaire Un ordinateur est une machine qui manipule des valeurs numériques représentées sous forme binaire.	**Manipuler** à l'aide d'opérations élémentaires les trois unités de base : bit, octet, mot.	On met en évidence, sous forme de questionnement, la présence du numérique dans la vie personnelle et professionnelle, au travers d'exemples.
Formats Les données numériques sont agencées de manière à en faciliter le stockage et le traitement. L'organisation des données numériques respecte des formats qui sont soit des standards de fait, soit des normes.	**Identifier** quelques formats de documents, d'images, de données sonores. **Choisir un format** approprié par rapport à un usage ou un besoin, à une qualité, à des limites.	Le choix d'un format approprié pose le problème de l'interopérabilité qui est le fait d'assurer un usage sans restriction des mêmes données sur un système différent. Le choix de l'algorithme de traitement des données dépend en particulier du format de ces données, et vice versa.
Taille de l'information Les données numériques occupent de la place. Il faut évaluer leur taille en vue de leur stockage, de leur traitement, de leur transmission.	**Estimer** la taille des données. **Connaître les ordres de grandeur** courants (périphériques usuels de stockage, bibliothèque, corpus littéraire, débits des connexions...)	L'apparition de l'informatique a permis de traiter des données de taille beaucoup plus importante que ce qui pouvait être fait auparavant.

5.2. Algorithmique

Un algorithme se définit comme une méthode opérationnelle permettant de résoudre, en un nombre fini d'étapes clairement spécifiées, toutes les instances d'un problème donné. Cette méthode peut être exécutée par une machine ou par une personne.

Les élèves ont déjà appris au collège à écrire, mettre au point et exécuter un programme. Les programmes de mathématiques des classes de seconde et première développent une pratique de l'algorithmique sur laquelle il convient également de s'appuyer.

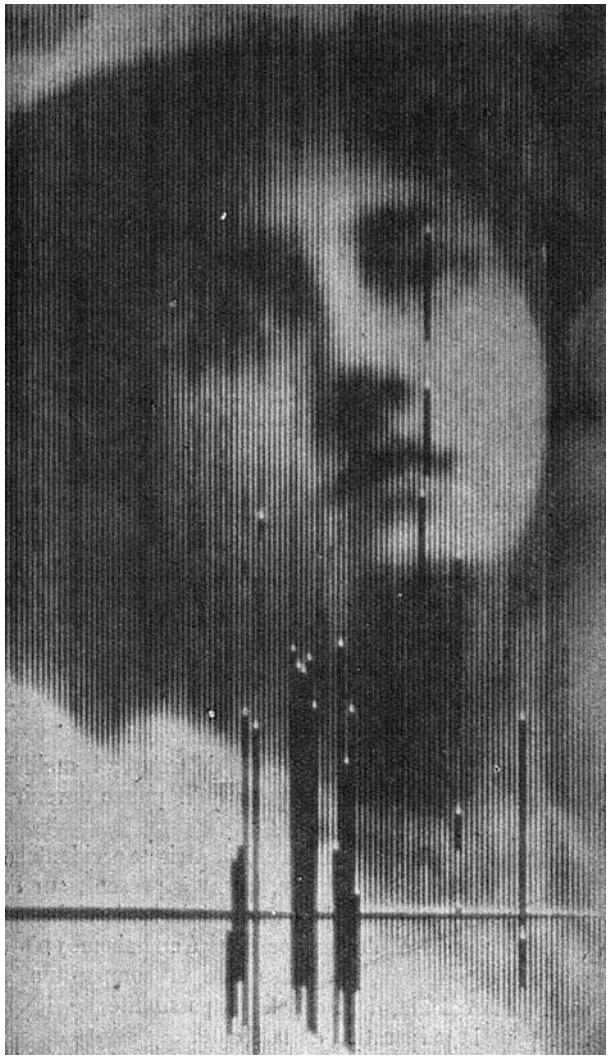
A partir du développement d'algorithmes, l'élève s'initie à la notion de complexité algorithmique. Ces algorithmes sont exprimés dans un langage de programmation et exécutés sur une machine ou bien définis de manière informelle.

SAVOIRS	CAPACITÉS	OBSERVATIONS
Algorithmes de référence – recherche dichotomique ; – addition de deux entiers exprimés en binaire ; – tri par sélection ; – tri par fusion ; – recherche d'un chemin dans un graphe par un parcours en largeur ou en profondeur.	**Comprendre** un algorithme et **expliquer** ce qu'il fait. **Modifier** un algorithme existant pour obtenir un résultat différent. **Concevoir** un algorithme. **Programmer** un algorithme. **S'interroger** sur l'efficacité d'un algorithme.	On présente simultanément les notions d'algorithme et de programme, puis on les distingue. L'objectif est une compréhension de ces algorithmes et la capacité à les mettre en œuvre. Les situations produisant une erreur (division par zéro, dépassement de capacité) sont mises en évidence. On présente les complexités logarithmique, linéaire et quadratique sur les exemples de la recherche dichotomique, de l'addition de deux entiers et du tri par sélection.
Traitement d'image Programmation d'algorithmes simples sur les images bitmap.	Modifier format, taille, contraste ou luminance d'images numériques. Détecter des informations spécifiques.	L'objectif est d'appliquer effectivement des programmes simples à des images. On peut aussi étudier le floutage, la rotation, la recherche de contour, etc.

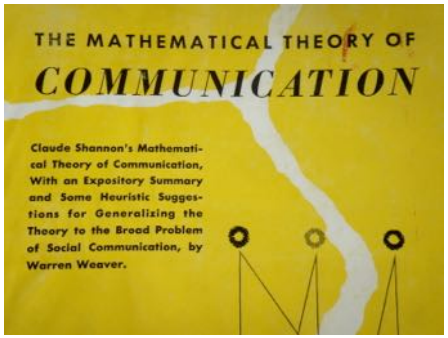
1- La numérisation

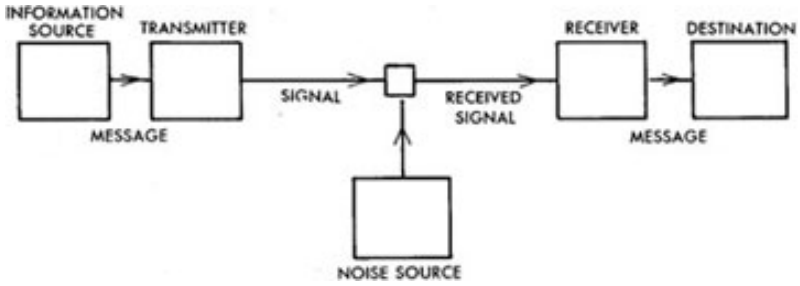
La numérisation de l'information : échantillonage et quantification

1907 : le téléphotographe de Korn



1948 : modèle de Shannon

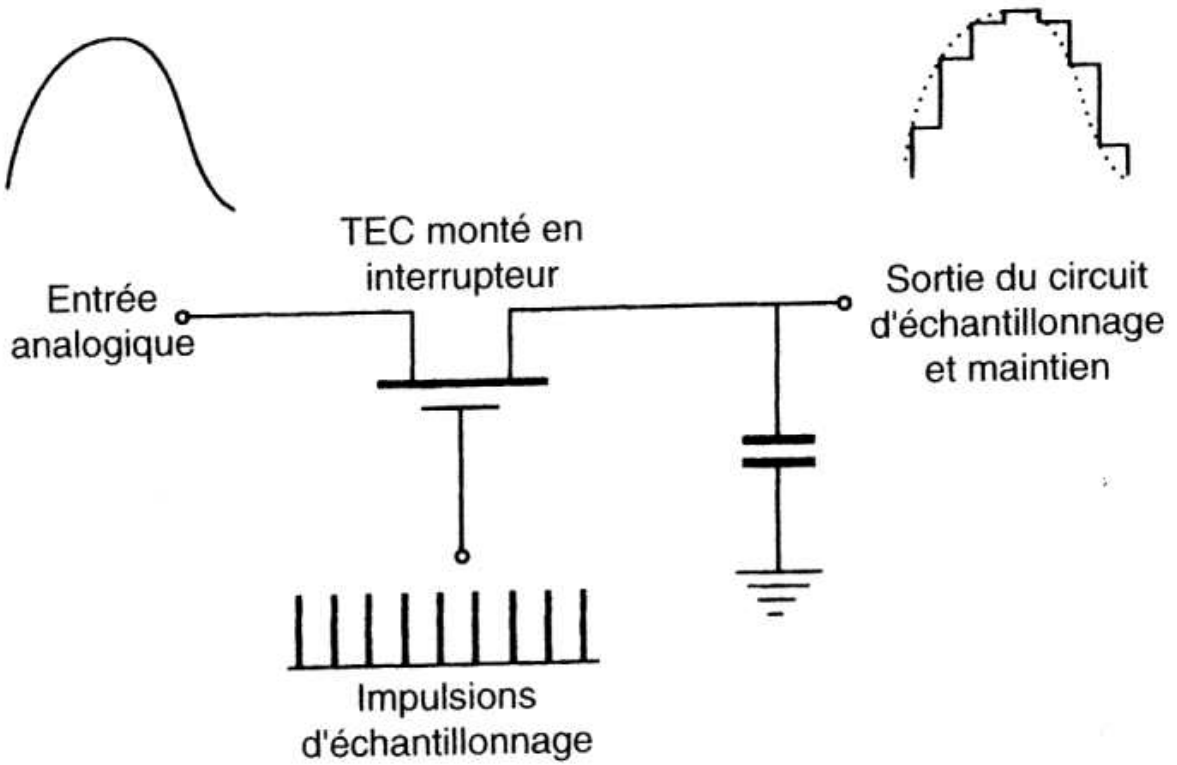




débit max d'un canal (bit/s) :

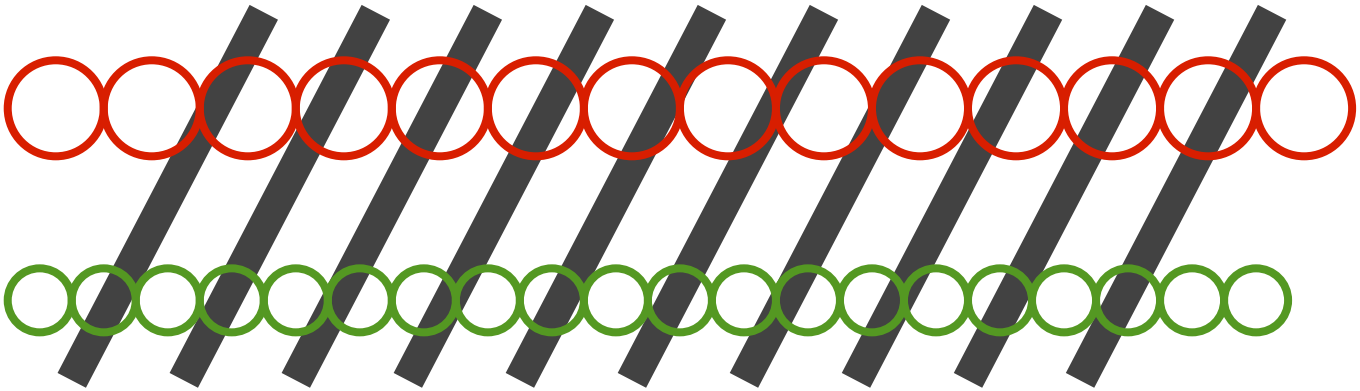
$$D_{\max} = B\log(1+\frac{S}{N})$$

Numérisation : principe



à quelle fréquence ???

Formule de Nyquist (version naïve)



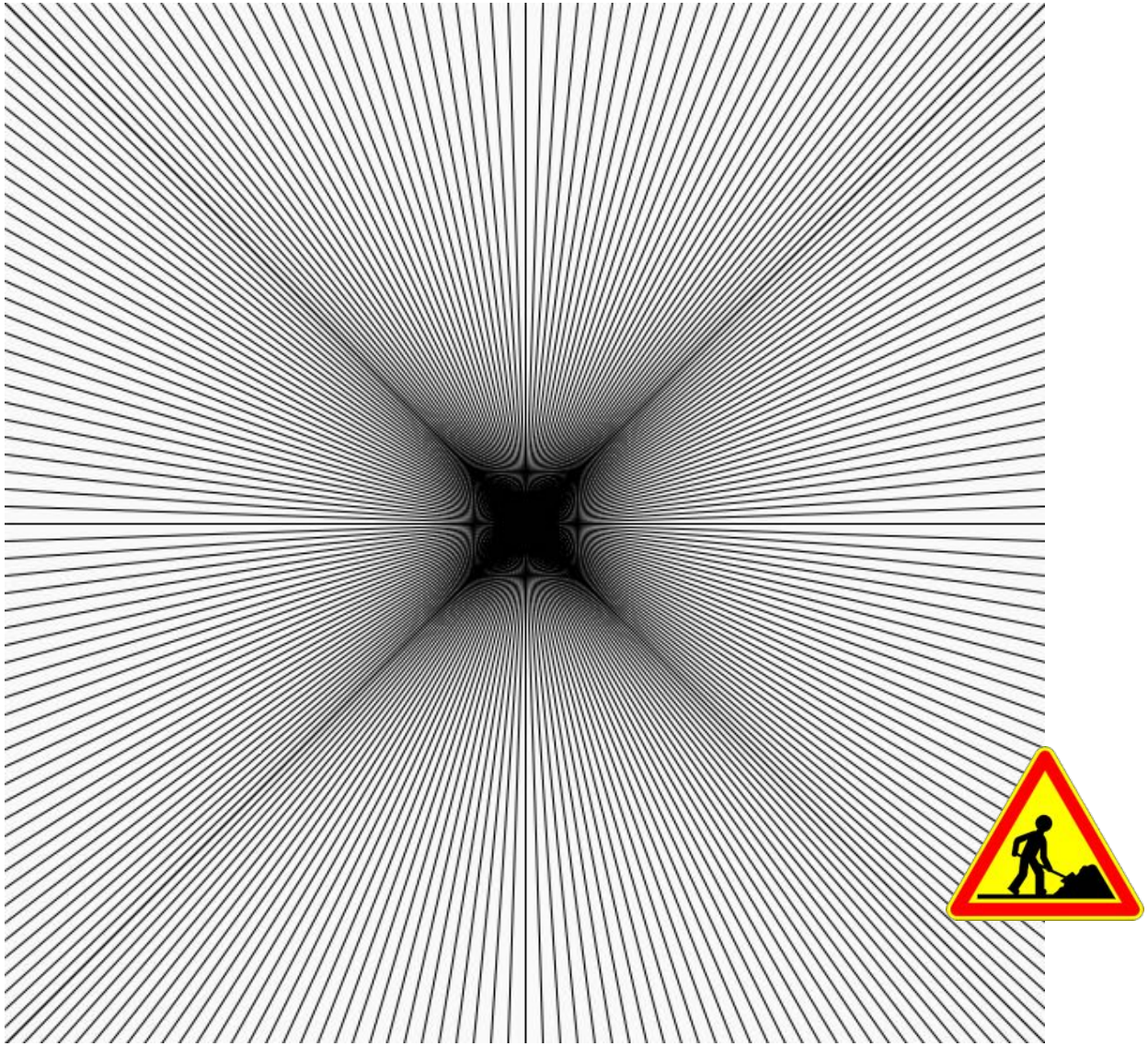
Fech > 2 F

Sous-échantillonage

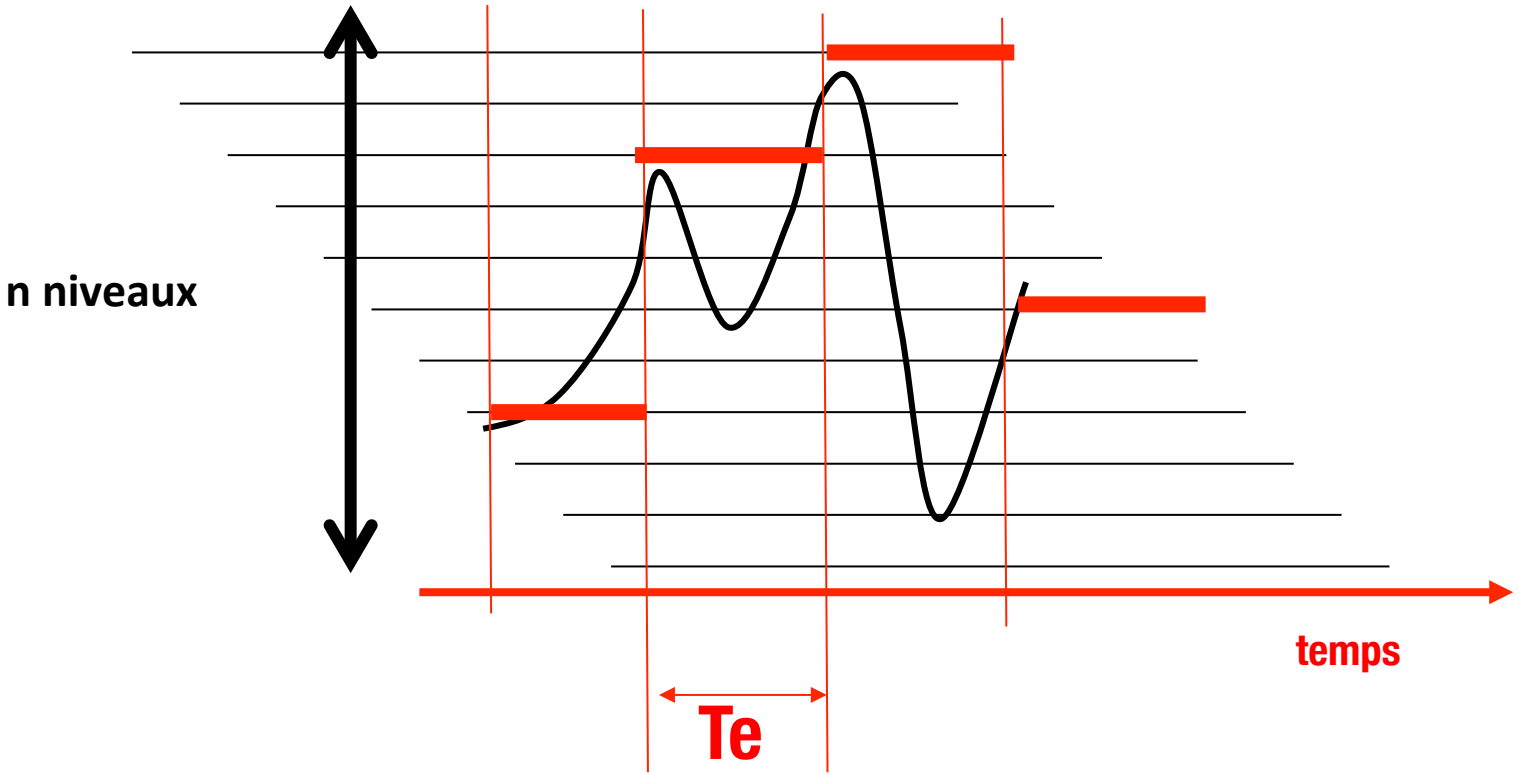
moiré

aliasing

une variante : phénomène de moiré



Quantification



n = 2, 4, 8, 16 etc.
= 2^K pour K bits par échantillon

Quantification : cas d'une image niveau de gris

1 bit/pixel

2 bits

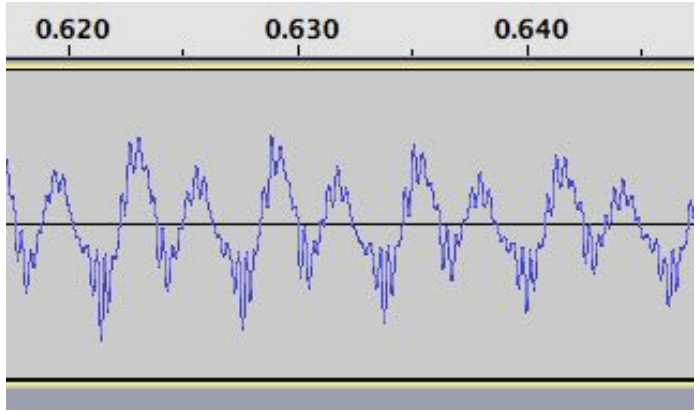
(sur Lenna : www.lenna.org)

4 bits

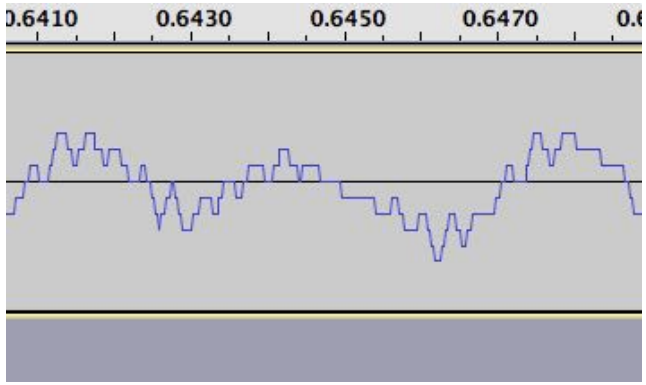
8 bits

Exemples sonores

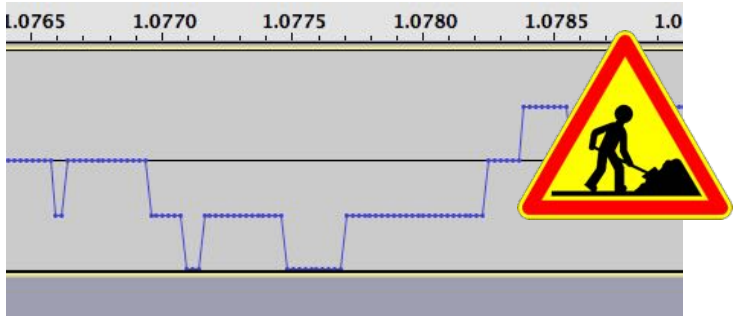
- quantification 16 bits



- quantification 4 bits

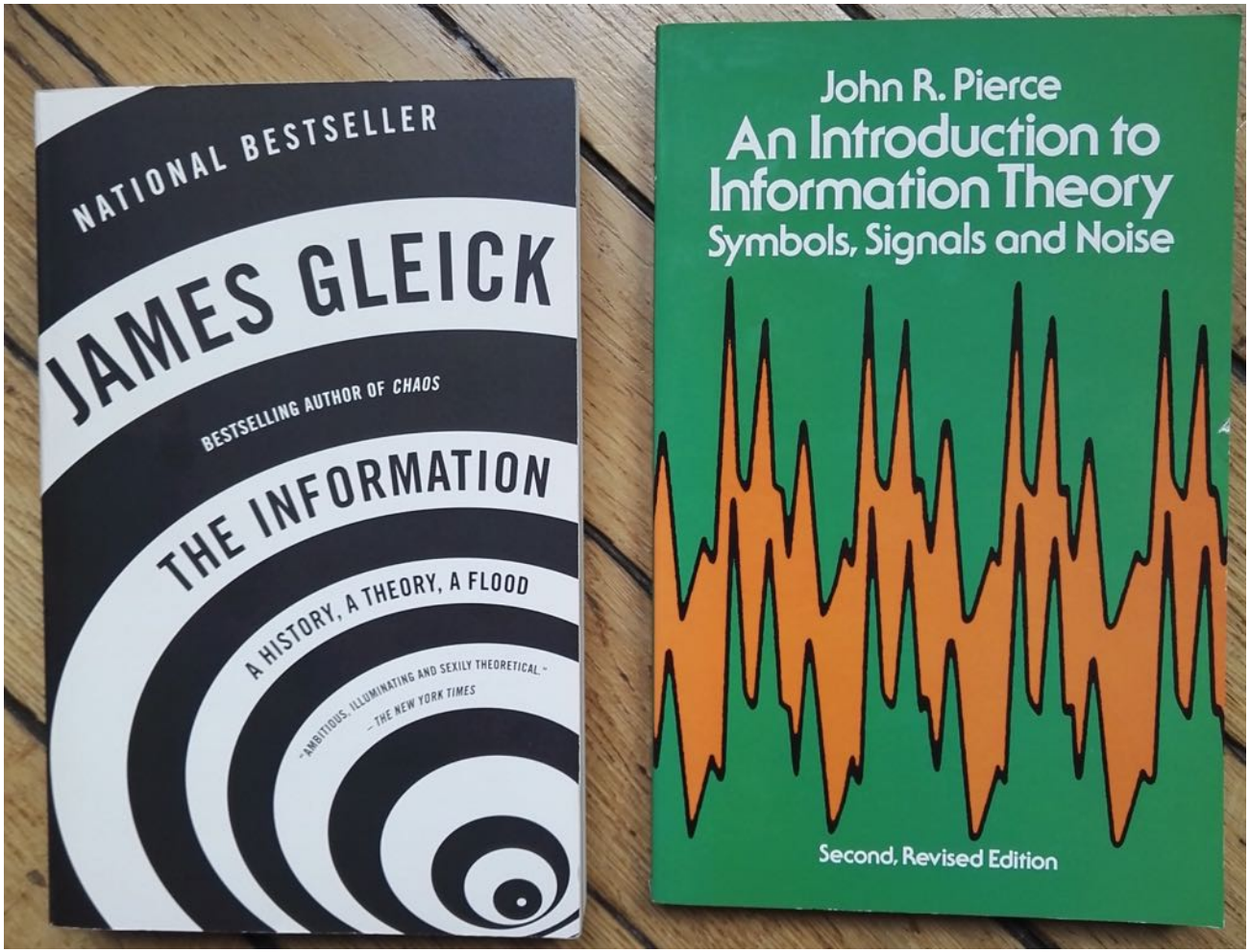


- quantification 2 bits !!



<= voir les sons
avec le logiciel audacity

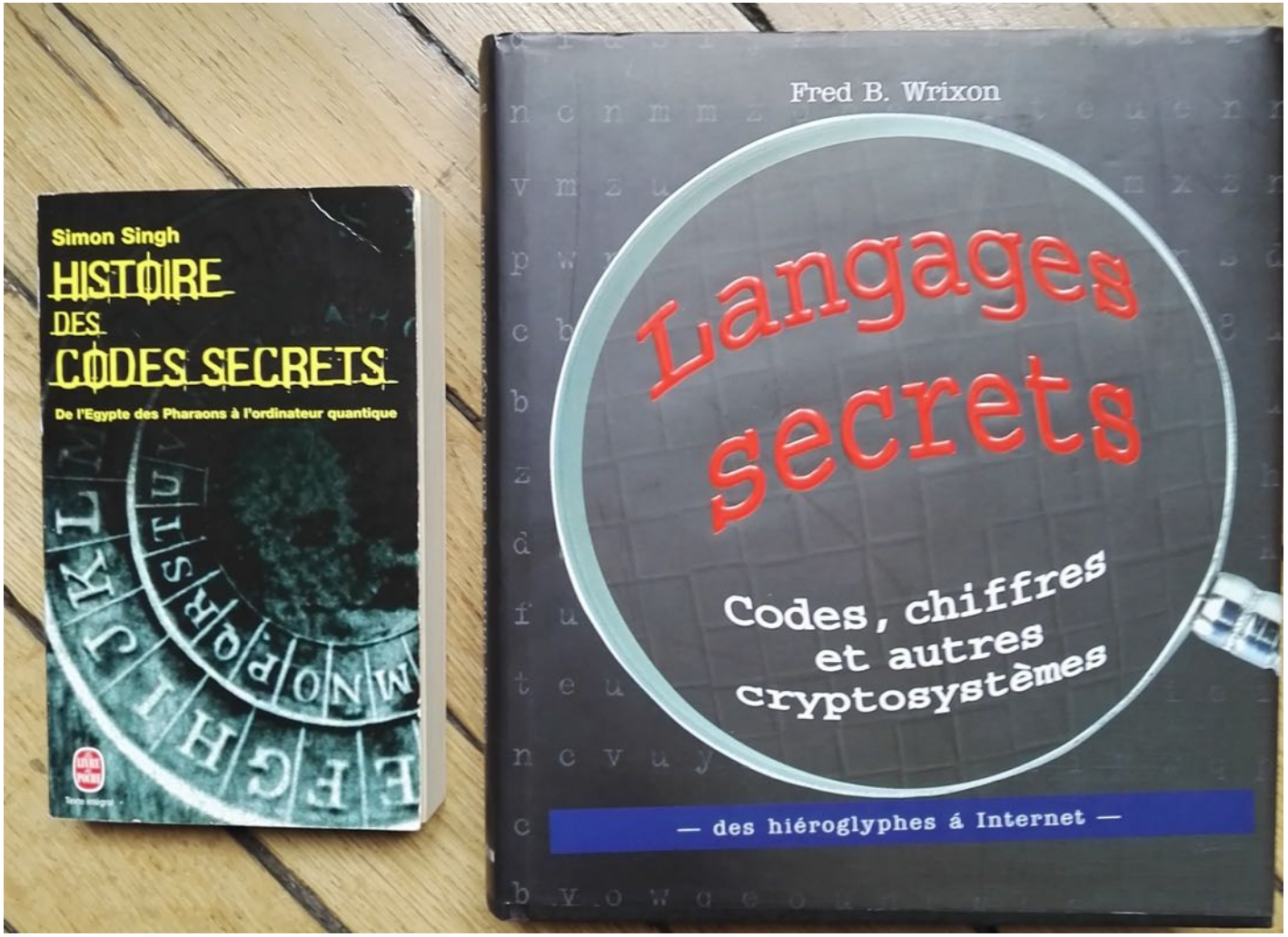
A lire ?



Livre de Gleick traduite en français (chez Cassini)
Tous les livres de Pierce sont bien (musique etc.)

2- Les codes

Au passage : codes et la cryptographie



Wrixon : chez Koneman (plus ludique)

Codage de l'information ?

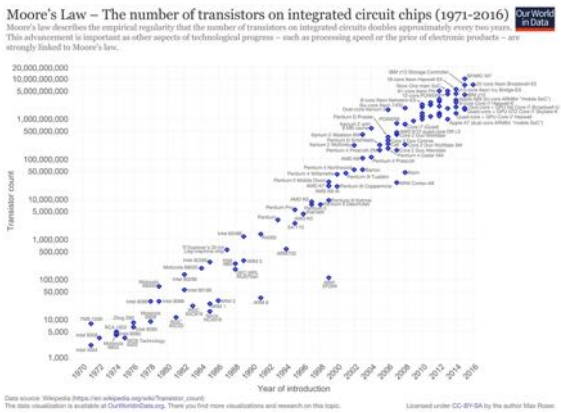
essentiellement une question liée à la technologie de la machine qui la manipule

binaire : plus simple pour les machines électroniques (≠ humain)

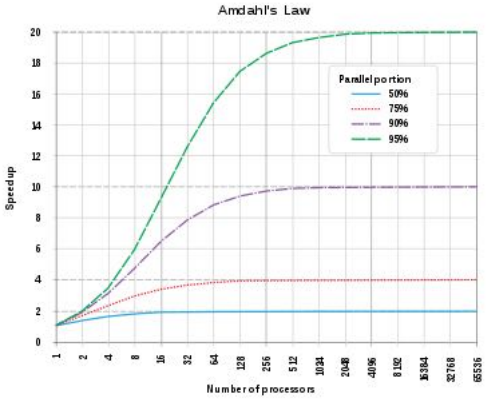
⇒ faut-il connaitre à minima l'architecture de l'ordinateur qu'on programme ?

⇒ un débutant en programmation doit-il la connaitre avant de programmer ?

sans doute de moins en moins vrai :



Loi de Moore sur les circuits intégrés



Loi d'Amdahl sur le temps d'execution d'un programme parallele

Préfixes du Système international d'unités — Wikipédia

1000^m	10^n	Préfixe français	Symbole	Depuis note 1	Nombre décimal	Désignation note 2
1000^8	10^{24}	yotta	Y	1991	1 000 000 000 000 000 000 000 000	Quadrillion
1000^7	10^{21}	zetta	Z	1991	1 000 000 000 000 000 000 000	Trilliard
1000^6	10^{18}	exa	E	1975	1 000 000 000 000 000 000	Trillion
1000^5	10^{15}	péta	P	1975	1 000 000 000 000 000	Billiard
1000^4	10^{12}	téra	T	1960	1 000 000 000 000	Billion
1000^3	10^9	giga	G	1960	1 000 000 000	Milliard
1000^2	10^6	méga	M	1960	1 000 000	Million
1000^1	10^3	kilo	k	1795	1 000	Millier
$1000^{2/3}$	10^2	hecto	h	1795	100	Centaine
$1000^{1/3}$	10^1	déca	da	1795	10	Dizaine
1000^0	10^0	(aucun)	—	—	1	Unité
$1000^{-1/3}$	10^{-1}	déci	d	1795	0,1	Dixième
$1000^{-2/3}$	10^{-2}	centi	c	1795	0,01	Centième
1000^{-1}	10^{-3}	milli	m	1795	0,001	Millième
1000^{-2}	10^{-6}	micro	μ	1960 note 3	0,000 001	Millionième
1000^{-3}	10^{-9}	nano	n	1960	0,000 000 001	Milliardième
1000^{-4}	10^{-12}	pico	p	1960	0,000 000 000 001	Billionième
1000^{-5}	10^{-15}	femto	f	1964	0,000 000 000 000 001	Billiardième
1000^{-6}	10^{-18}	atto	a	1964	0,000 000 000 000 000 001	Trillionième
1000^{-7}	10^{-21}	zepto	z	1991	0,000 000 000 000 000 000 001	Trilliardième
1000^{-8}	10^{-24}	yocto	y	1991	0,000 000 000 000 000 000 000 001	Quadrillionième

Quelques ordres de grandeurs

Capacité mémoire : en bits

1 octet = 8 bits = 1 byte (pas toujours)
1 o = 1 caractère (table d'encodage ISOLatin1)
3 o = un pixel sur un écran "million de couleur"
4 o = 1 nombre réel dans la norme IEEE-754

2 Ko (kilo) = 10^3 o = une page de texte

1Mo (mega) = 10^6 o = Un roman en mode texte
20 Mo = Un roman en fac-similé = Un logiciel
500 Mo = Texte de l'Encyclopedia Universalis (CDROM)

1 Go (giga) = 10^9 o
= Une bibliothèque personnelle en mode texte
10 Go = Un film compressé (DVD)
600 Go = Une librairie en fac-similé (300 Kvol.)

1 To (tera) = 10^12 o = Une cinémathèque personnelle
20 To = plus grand assemblage de disques en 1996 (LNB)
= la "library of congress" en mode texte (50 Mvol)

1 Po (peta) = 10^15 o
= Une bibliothèque nationale en mode image
15 Po = production mondiale de disques en 1995

Temps d'accès (≠ tps de cycle) : en secondes

1 ms (milli) = 10^-3 s
20 ms : tps accès maximum pour un relais
1 μs (micro) = 10^-6 s = tps accès maximum pour un tore
1 ns (nano) = 10^-9s
100 ns = temps d'accès DRAM
5 ns = temps d'accès bascule élementaire
1 ps (pico) = 10^-12s
temps pour qu'un signal lumineux ds le vide parcoure 3 cm

Controverse du kilo à 1024

Tableaux des préfixes binaires et décimaux [modifier | modifier le code]

Voici à gauche le tableau des préfixes binaires et à droite celui des préfixes décimaux, pour comparaison.

Préfixes binaires (préfixes CEI)

Nom	Symbole	2^{10a} = facteur	a
kibi	Ki	2^{10} = 1 024	1
mébi	Mi	2^{20} = 1 048 576	2
gibi	Gi	2^{30} = 1 073 741 824	3
tébi	Ti	2^{40} = 1 099 511 627 776	4
pébi	Pi	2^{50} = 1 125 899 906 842 624	5
exbi	Ei	2^{60} = 1 152 921 504 606 846 976	6
zébi	Zi	2^{70} = 1 180 591 620 717 411 303 424	7
yobi	Yi	2^{80} = 1 208 925 819 614 629 174 706 176	8

Préfixes décimaux (préfixes SI)

Nom	Symbole	10^{3a} = facteur	a	Erreur	Erreur inverse
kilo	k	10^{3} = 1 000	1	2 %	-2,3%
méga	M	10^{6} = 1 000 000	2	5 %	-4,6%
giga	G	10^{9} = 1 000 000 000	3	7 %	-6,9%
téra	T	10^{12} = 1 000 000 000 000	4	10 %	-9%
péta	P	10^{15} = 1 000 000 000 000 000	5	13 %	-11%
exa	E	10^{18} = 1 000 000 000 000 000 000	6	15 %	-13%
zetta	Z	10^{21} = 1 000 000 000 000 000 000 000	7	18 %	-15%
yotta	Y	10^{24} = 1 000 000 000 000 000 000 000 000	8	21 %	-17%

Dans ce deuxième tableau, l'erreur indiquée dans l'avant dernière colonne est celle effectuée quand on utilise un préfixe décimal à la place d'un préfixe binaire (et non le contraire). Si cette erreur n'est que de 2 % pour kilo au lieu de kibi, ce qui est parfois supportable, elle atteint 7 % pour giga/gibi, et même presque 10 % pour téra/tébi.

Ce rapprochement entre préfixes décimaux et binaires provient d'une coïncidence arithmétique qui fait que 1 024 = 2^{10} est proche de 1 000 = 10^{3}, à 2,4 % près.

Une autre 

Boutisme

[masquer]

En informatique, certaines données telles que les nombres entiers peuvent être représentées sur plusieurs octets. L'ordre dans lequel ces octets sont organisés en mémoire ou dans une communication est appelé **boutisme**, ***endianness*** ou plus rarement **endianisme**.

De la même manière que certaines langues s'écrivent de gauche à droite, et d'autres s'écrivent de droite à gauche[1], il existe une alternative majeure à l'organisation des octets représentant une donnée : l'orientation **gros-boutiste** et l'orientation **petit-boutiste** (ou gros-boutienne et petit-boutienne). Les expressions ***byte order***, **ordre des octets**, ou ***byte sex***, sont également utilisées (bien qu'ordre des octets fasse référence à l'unité d'une base numérale précise sur 8 bits, que les autres termes plus généraux ne traduisent pas).

Le boutisme qualifie aussi bien un fichier ou un protocole (dans lesquels ce sont les octets qui sont ordonnés différemment) qu'un processeur (dans lequel la gestion des bits a aussi un ordre).

Sommaire [masquer]

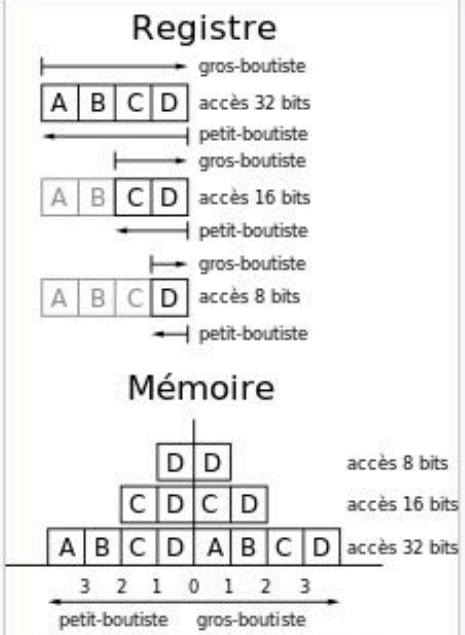


Schéma récapitulatif des modes de représentation.

Codage des caractères

Qautre problèmes :
- caractère ≠ glyphe
- coder ≠ classer
- norme ≠ standard
- gérer l'existant

Aussi vieux que le télégraphe… et toujours pas bien résolus.

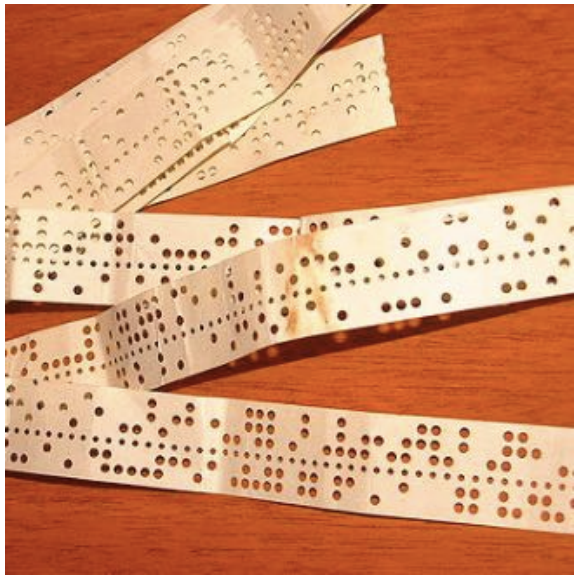
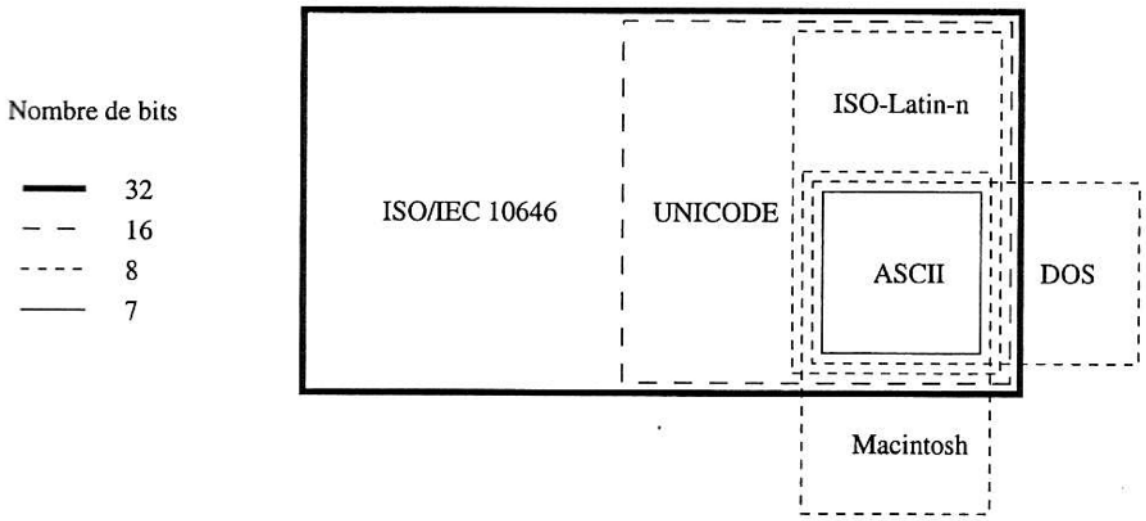


Tableau 1 – Relation entre le nombre de moments et le nombre de caractères pour les principaux codages

Nom du codage	Nombre de bits ou moments	Nombre de caractères
Telex	6	64
Ascii	7	128
ISO Latin-1	8	256
Unicode	16	65 536
ISO/IEC 10646	32	> 2 milliards



7 bits : American Standard Code for Information Interchange
(ASCII, 1967 - puis ISO646 en 1983)

Tableau 6 – Codage Ascii dans sa version finale de 1983 (ISO 646)

	000	001	002	003	004	005	006	007
0	NUL	DLE	SP	0	@	P	`	p
1	STX	DC1	!	1	A	Q	a	q
2	SOT	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Les codes sont donnés en hexadécimal (par exemple Q a pour code 0051_{16}).

code	*acronyme*	*nom anglais*	*nom français*
0	<NUL>	NULl	Nul
1	<SOH>	Start Of Heading	Début d'en-tête
2	<STX>	Start of TeXt	Début de texte
3	<ETX>	End of TeXt	Fin de texte
4	<EOT>	End Of Transmission	Fin de transmission
5	<ENQ>	ENQuiry	Requête
6	<ACK>	ACKnowledgement	Accusé de réception
7	<BEL>	BELl	Sonnerie
8	<BS>	BackSpace	Espace arrière
9	<HT>	Horizontal Tabulation	Tabulation horizontale
10	<LF>	Line Feed	Interligne
11	<VT>	Vertical Tabulation	Tabulation verticale
12	<FF>	Form Feed	Présentation de feuille
13	<CR>	Carriage Return	Retour chariot
14	<SO>	Shift Out	Hors-code
15	<SI>	Shift In	En-code
16	<DLE>	DataLink Escape	Échappement à la transmission
17	<DC1>	Device Control 1	Contrôle de périphérique 1
18	<DC2>	Device Control 2	Contrôle de périphérique 2
19	<DC3>	Device Control 3	Contrôle de périphérique 3
20	<DC4>	Device Control 4	Contrôle de périphérique 4
21	<NAK>	Negative AcKnowledge	Accusé de reception négatif
22	<SYN>	SYNchronous idle	Synchronisation
23	<ETB>	End of Transmission Block	Fin de bloc de transmission
24	<CAN>	CANcel	Annulation
25	<EM>	End of Medium	Fin de support
26	<SUB>	SUBstitute	Substitution
27	<ESC>	ESCape	Échappement
28	<FS>	File Separator	Séparateur de fichiers
29	<GS>	Group Separator	Séparateur de groupes de données
30	<RS>	Record Separator	Séparateur d'enregistrement
31	<US>	Unit Separator	Séparateur d'unités

TAB. 3.2 – *Codes ASCII inférieurs à 32.*

Au delà : Unicode, UCS, UTF-8, etc.

2^16 = 65536 trop petit pour toutes les langues humaines
2^32 = 4294967296 très confortable

⇒ norme IS0 10646 (universal character set, UCS) et, en parallèle, consortium Unicode

UTF-8 and Unicode FAQ for Unix/Linux

by Markus Kuhn

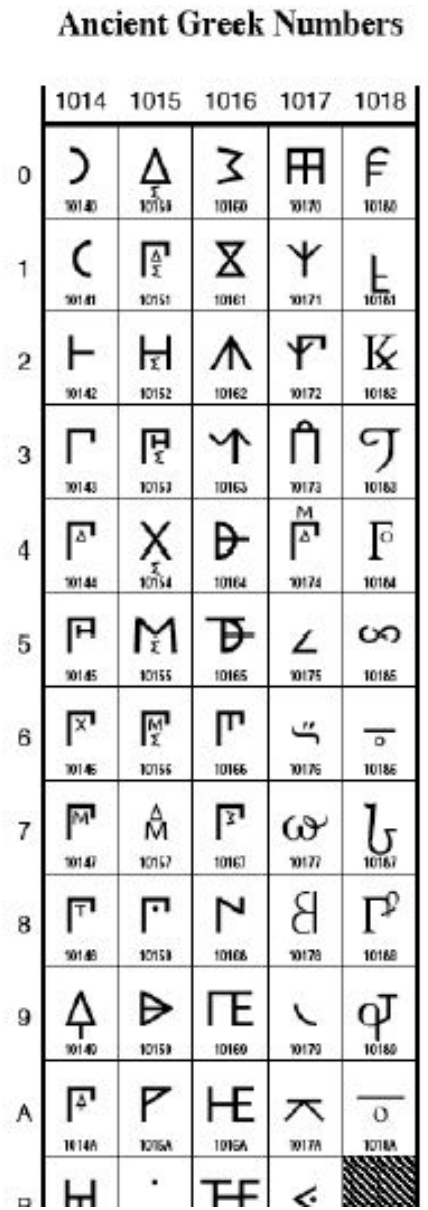
This text is a very comprehensive one-stop information resource on how you can use Unicode/UTF-8 on POSIX systems (Linux, Unix). You will find here both introductory information for every user, as well as detailed references for the experienced developer.

Unicode now replaces ASCII, ISO 8859 and EUC at all levels. It enables users to handle not only practically any script and language used on this planet, it also supports a comprehensive set of mathematical and technical symbols to simplify scientific information exchange.

With the UTF-8 encoding, Unicode can be used in a convenient and backwards compatible way in environments that were designed entirely around ASCII, like Unix. UTF-8 is the way in which Unicode is used under Unix, Linux, and similar systems. Make sure that you are well familiar with it and that your software supports UTF-8 smoothly.

Contents

https://www.cl.cam.ac.uk/~mgk25/unicode.html



http://www.unicode.org

Codage UTF-8 : 2^21 caractères au max (comme l'UCS actuel)

Définition du nombre d'octets utilisés dans le codage (uniquement les séquences valides)

Caractères codés	Représentation binaire UTF-8	Premier octet valide (hexadécimal)	Signification
U+0000 à U+007F	0xxxxxxx	00 à 7F	1 octet, codant 7 bits
U+0080 à U+07FF	110xxxxx 10xxxxxx	C2 à DF	2 octets, codant 11 bits
U+0800 à U+0FFF	1110 0000 10 1xxxxx 10xxxxxx	E0 (le 2e octet est restreint de A0 à BF)	3 octets, codant 16 bits
U+1000 à U+1FFF	1110 0001 10xxxxxx 10xxxxxx	E1	
U+2000 à U+3FFF	1110 001x 10xxxxxx 10xxxxxx	E2 à E3	
U+4000 à U+7FFF	1110 01xx 10xxxxxx 10xxxxxx	E4 à E7	
U+8000 à U+BFFF	1110 10xx 10xxxxxx 10xxxxxx	E8 à EB	
U+C000 à U+CFFF	1110 1100 10xxxxxx 10xxxxxx	EC	
U+D000 à U+D7FF	1110 1101 10 0xxxxx 10xxxxxx	ED (le 2e octet est restreint de 80 à 9F)	
U+E000 à U+FFFF	1110 111x 10xxxxxx 10xxxxxx	EE à EF	
U+10000 à U+1FFFF	11110 000 10 01xxxx 10xxxxxx 10xxxxxx	F0 (le 2e octet est restreint de 90 à BF)	4 octets, codant 21 bits
U+20000 à U+3FFFF	11110 000 10 1xxxxx 10xxxxxx 10xxxxxx		
U+40000 à U+7FFFF	11110 001 10xxxxxx 10xxxxxx 10xxxxxx	F1	
U+80000 à U+FFFFF	11110 01x 10xxxxxx 10xxxxxx 10xxxxxx	F2 à F3	
U+100000 à U+10FFFF	11110 100 10 00xxxx 10xxxxxx 10xxxxxx	F4 (le 2e octet est restreint de 80 à 8F)	

+ les variantes UTF-16 (Windows, Java...) et UTF-32 (un jour ?)

et un outil Unix : iconv

```
[numer:~] pcubaud% iconv -l
ANSI_X3.4-1968 ANSI_X3.4-1986 ASCII CP367 IBM367 ISO-IR-6 ISO646-US ISO_646.IRV:1991 US US-ASCII CSASCII
UTF-8
ISO-10646-UCS-2 UCS-2 CSUNICODE
UCS-2BE UNICODE-1-1 UNICODEBIG CSUNICODE11
UCS-2LE UNICODELITTLE
ISO-10646-UCS-4 UCS-4 CSUCS4
UCS-4BE
UCS-4LE
UTF-16
UTF-16BE
UTF-16LE
UTF-32
UTF-32BE
UTF-32LE
UNICODE-1-1-UTF-7 UTF-7 CSUNICODE11UTF7
UCS-2-INTERNAL
UCS-2-SWAPPED
UCS-4-INTERNAL
UCS-4-SWAPPED
C99
JAVA
CP819 IBM819 ISO-8859-1 ISO-IR-100 ISO8859-1 ISO_8859-1 ISO_8859-1:1987 L1 LATIN1 CSISOLATIN1
ISO-8859-2 ISO-IR-101 ISO8859-2 ISO_8859-2 ISO_8859-2:1987 L2 LATIN2 CSISOLATIN2
ISO-8859-3 ISO-IR-109 ISO8859-3 ISO_8859-3 ISO_8859-3:1988 L3 LATIN3 CSISOLATIN3
ISO-8859-4 ISO-IR-110 ISO8859-4 ISO_8859-4 ISO_8859-4:1988 L4 LATIN4 CSISOLATIN4
CYRILLIC ISO-8859-5 ISO-IR-144 ISO8859-5 ISO_8859-5 ISO_8859-5:1988 CSISOLATINCYRILLIC
ARABIC ASMO-708 ECMA-114 ISO-8859-6 ISO-IR-127 ISO8859-6 ISO_8859-6 ISO_8859-6:1987 CSISOLATINARABIC
ECMA-118 ELOT_928 GREEK GREEK8 ISO-8859-7 ISO-IR-126 ISO8859-7 ISO_8859-7 ISO_8859-7:1987 CSISOLATINGREEK
HEBREW ISO-8859-8 ISO-IR-138 ISO8859-8 ISO_8859-8 ISO_8859-8:1988 CSISOLATINHEBREW
ISO-8859-9 ISO-IR-148 ISO8859-9 ISO_8859-9 ISO_8859-9:1989 L5 LATIN5 CSISOLATIN5
ISO-8859-10 ISO-IR-157 ISO8859-10 ISO_8859-10 ISO_8859-10:1992 L6 LATIN6 CSISOLATIN6
ISO-8859-13 ISO-IR-179 ISO8859-13 ISO_8859-13 L7 LATIN7
ISO-8859-14 ISO-CELTIC ISO-IR-199 ISO8859-14 ISO_8859-14 ISO_8859-14:1998 L8 LATIN8
ISO-8859-15 ISO-IR-203 ISO8859-15 ISO_8859-15 ISO_8859-15:1998
ISO-8859-16 ISO-IR-226 ISO8859-16 ISO_8859-16 ISO_8859-16:2000
KOI8-R CSKOI8R
KOI8-U
KOI8-RU
CP1250 MS-EE WINDOWS-1250
CP1251 MS-CYRL WINDOWS-1251
CP1252 MS-ANSI WINDOWS-1252
CP1253 MS-GREEK WINDOWS-1253
CP1254 MS-TURK WINDOWS-1254
CP1255 MS-HEBR WINDOWS-1255
CP1256 MS-ARAB WINDOWS-1256
CP1257 WINBALTRIM WINDOWS-1257
CP1258 WINDOWS-1258
850 CP850 IBM850 CSPC850MULTILINGUAL
862 CP862 IBM862 CSPC862LATINHEBREW
866 CP866 IBM866 CSIBM866
MAC MACINTOSH MACROMAN CSMACINTOSH
MACCENTRALEUROPE
MACICELAND
MACCROATIAN
MACROMANIA
MACCYRILLIC
MACUKRAINE
MACGREEK
MACTURKISH
MACHEBREW
MACARABIC
MACTHAI
HP-ROMAN8 R8 ROMAN8 CSHPROMAN8
NEXTSTEP
ARMSCII-8
GEORGIAN-ACADEMY
GEORGIAN-PS
KOI8-T
MULELAO-1
CP1133 IBM-CP1133
ISO-IR-166 TIS-620 TIS620 TIS620-0 TIS620.2529-1 TIS620.2533-0 TIS620.2533-1
CP874 WINDOWS-874
VISCII VISCII1.1-1 CSVISCII
TCVN TCVN-5712 TCVN5712-1 TCVN5712-1:1993
ISO-IR-14 ISO646-JP JIS_C6220-1969-RO JP CSISO14JISC6220RO
JISX0201-1976 JIS_X0201 X0201 CSHALFWIDTHKATAKANA
ISO-IR-87 JIS0208 JIS_C6226-1983 JIS_X0208 JIS_X0208-1983 JIS_X0208-1990 X0208 CSISO87JISX0208
ISO-IR-159 JIS_X0212 JIS_X0212-1990 JIS_X0212.1990-0 X0212 CSISO159JISX02121990
CN GB_1988-80 ISO-IR-57 ISO646-CN CSISO57GB1988
CHINESE GB_2312-80 ISO-IR-58 CSISO58GB231280
CN-GB-ISOIR165 ISO-IR-165
ISO-IR-149 KOREAN KSC_5601 KS_C_5601-1987 KS_C_5601-1989 CSKSC56011987
EUC-JP EUCJP EXTENDED_UNIX_CODE_PACKED_FORMAT_FOR_JAPANESE CSEUCPKDFMTJAPANESE
MS_KANJI SHIFT-JIS SHIFT_JIS SJIS CSSHIFTJIS
CP932
ISO-2022-JP CSISO2022JP
ISO-2022-JP-1
ISO-2022-JP-2 CSISO2022JP2
CN-GB EUC-CN EUCCN GB2312 CSGB2312
CP936 GBK
GB18030
ISO-2022-CN CSISO2022CN
ISO-2022-CN-EXT
HZ HZ-GB-2312
EUC-TW EUCTW CSEUCTW
BIG-5 BIG-FIVE BIG5 BIGFIVE CN-BIG5 CSBIG5
CP950
BIG5-HKSCS BIG5HKSCS
EUC-KR EUCKR CSEUCKR
CP949 UHC
CP1361 JOHAB
ISO-2022-KR CSISO2022KR
[numer:~] pcubaud%
```

Codage des nombres entiers

- pb évident de quantification :
- mémoire et organes de calculs en nombre fini
- remplacer l'infinité d'espace par l'infinité de temps (Babbage)

- pudeur dommageable des langages de prog vs les architectures
  ex : « long » vs « int32 » ou « int64 »

- quel confort d'avoir un calculateur 64bits pour les entiers !
  2^64 ~ 1.8*10^18 valeurs
  exemple du générateur aléatoire 32bits de Park et Miller
  *u = 16807*u % 2147483647*
- entiers de longueur arbitraire en Python (entre autres)
- montrer calcul en base 2, base 16 : vraiment indispensable en 2019 ?
- astuce du complément à deux pour les soustractions binaires
- codage des couleurs <RVB> en hexa pour le Web

Codage des réels

- partie entière et partie fractionnaire = deux entiers ? 10^18 plage trop limitée ?
  nombre avogadro ~10^23 constante Planck ~10^-27
- plus malin : codage virgule flottante

completer

- partie fractionnaire : la conversion en binaire se passe souvent mal

  0.1 => **0**/2 + **0**/4 + **0**/8 + **1**/16 + **1**/32 + **0**/64 + **0**/128 + **0**/256 + **1**/512 + **1**/1024 + **0**/2048 + …
- conséquences :

  0.1 + 0.2 = 0.30000000000000004

  (a+b)+c ≠ a+(b+c) plus associatif

- la norme IEEE 754 (revue en 2008)
- En 2019, FPU 64bits pour tous !

En Python (et Ada, Java etc) : on peut faire du calcul en base 10 :

`decimal` — Arithmétique décimale en virgule fixe et flottante

Code source : Lib/decimal.py

Le module `decimal` fournit une arithmétique en virgule flottante rapide et produisant des arrondis mathématiquement corrects. Il possède plusieurs avantages en comparaison au type `float` :

- Le module `decimal` « est basé sur un modèle en virgule flottante conçu pour les humains, qui suit ce principe directeur : l'ordinateur doit fournir un modèle de calcul qui fonctionne de la même manière que le calcul qu'on apprend à l'école » -- extrait (traduit) de la spécification de l'arithmétique décimale.

- Les nombres décimaux peuvent être représentés exactement en base décimale flottante. En revanche, des nombres tels que `1.1` ou `1.2` n'ont pas de représentation exacte en base binaire flottante. L'utilisateur final ne s'attend typiquement pas à obtenir `3.3000000000000003` lorsqu'il saisit `1.1 + 2.2`, ce qui se passe en arithmétique binaire à virgule flottante.

- Ces inexactitudes ont des conséquences en arithmétique. En base décimale à virgule flottante, `0.1 + 0.1 + 0.1 - 0.3` est exactement égal à zéro. En virgule flottante binaire, l'ordinateur l'évalue à `5.5511151231257827e-017`. Bien que très proche de zéro, cette différence induit des erreurs lors des tests d'égalité, erreurs qui peuvent s'accumuler. Pour ces raisons `decimal` est le module utilisé pour des applications comptables ayant des contraintes strictes de fiabilité.

- Le module `decimal` incorpore la notion de chiffres significatifs, tels que `1.30 + 1.20` est égal à `2.50`. Le dernier zéro n'est conservé que pour respecter le nombre de chiffres significatifs. C'est

butive properties of addition:

```
# Examples from Seminumerical Algorithms, Section 4.2.2.
>>> from decimal import Decimal, getcontext
>>> getcontext().prec = 8

>>> u, v, w = Decimal(11111113), Decimal(-11111111), Decimal('7.51111111')
>>> (u + v) + w
Decimal('9.5111111')
>>> u + (v + w)
Decimal('10')

>>> u, v, w = Decimal(20000), Decimal(-6), Decimal('6.0000003')
>>> (u*v) + (u*w)
Decimal('0.01')
>>> u * (v+w)
Decimal('0.0060000')
```

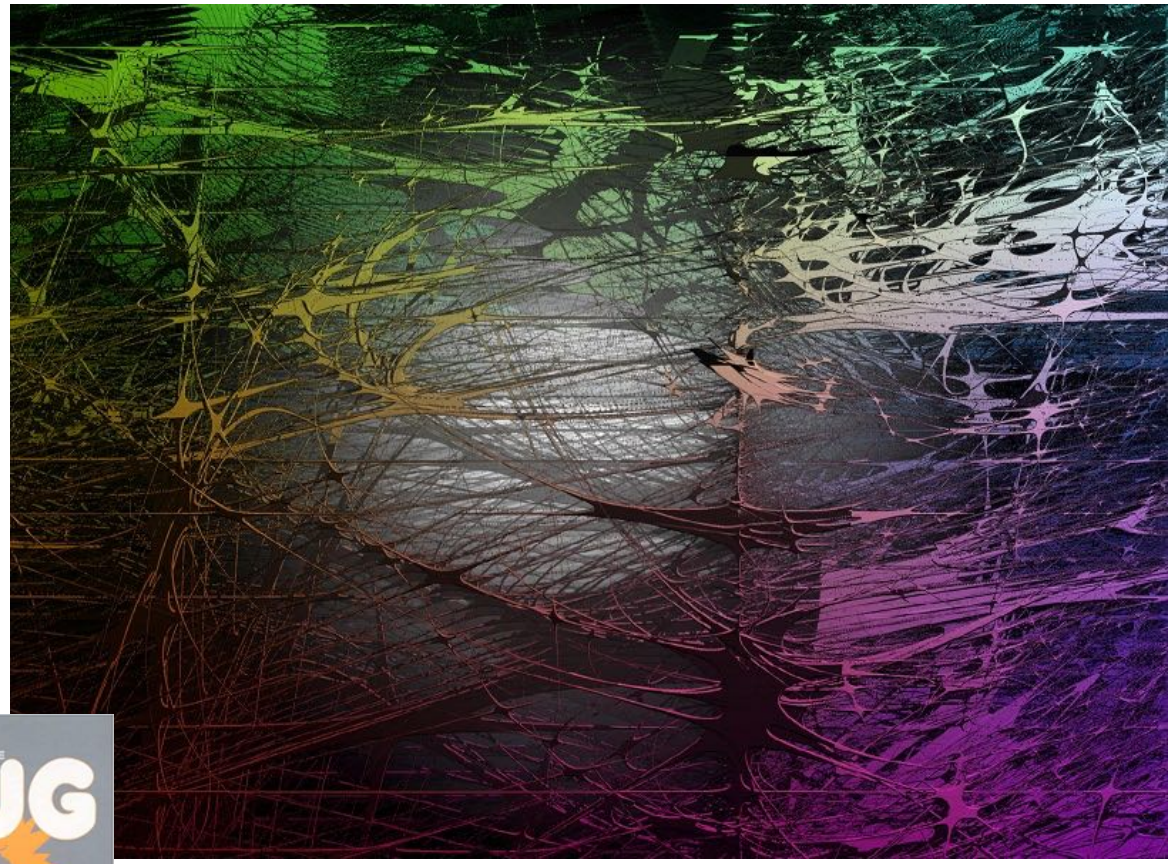
The `decimal` module makes it possible to restore the identities by expanding the precision sufficiently to avoid loss of significance:

```
>>> getcontext().prec = 20
>>> u, v, w = Decimal(11111113), Decimal(-11111111), Decimal('7.51111111')
>>> (u + v) + w
Decimal('9.51111111')
>>> u + (v + w)
Decimal('9.51111111')
>>>
>>> u, v, w = Decimal(20000), Decimal(-6), Decimal('6.0000003')
>>> (u*v) + (u*w)
Decimal('0.0060000')
>>> u * (v+w)
Decimal('0.0060000')
```

Mais ça peut ne pas suffire : cas des grandes itérations de calcul scientifique

Exemple de J.F. Colonna sur la dynamique de Verhulst

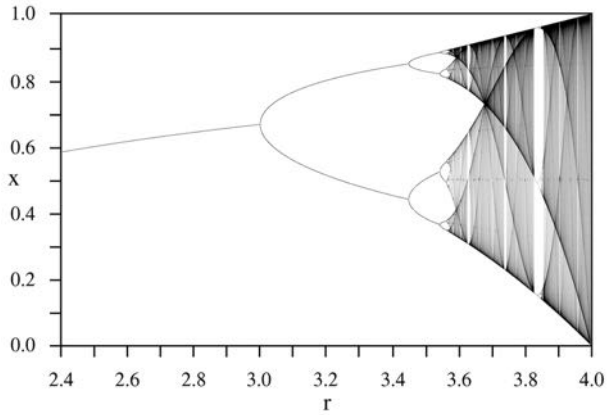
http://www.lactamme.polytechnique.fr/



<= lire aussi son bon petit livre (Flammarion, 6€)

Suite de Verhulst discrète (suite « logistique ») :

$$U(n+1) = R*U(n)*(1-U(n))$$



(wikipedia)

Avec U(0) dans [0;1[et R dans [0;4]

si R > 3.57 la suite est chaotique

Si R=3 la suite converge très lentement vers 2/3

Colonna montre dans ce cas l ’effet de la perte de l’associativité dans l’itération sur U(n)
Exemple en C avec flottants 64 bits :

IBM RS6000 :

	(R+1)X-R(XX)	(R+1)X-(RX)X	((R+1)-(RX))X	RX+(1-(RX))X	X+R(X-(XX))
X(00) =	0.500000	0.500000	0.500000	0.500000	0.500000
X(10) =	0.384631	0.384631	0.384631	0.384631	0.384631
X(20) =	0.418895	0.418895	0.418895	0.418895	0.418895
X(30) =	0.046399	0.046399	0.046399	0.046399	0.046399
X(40) =	0.320177	0.320184	0.320188	0.320190	0.320189
X(50) =	0.067567	0.063747	0.061859	0.060822	0.061486
X(60) =	0.001145	0.271115	0.616781	0.298613	1.307350
X(70) =	1.296775	1.328462	0.486629	0.938605	1.054669
X(80) =	0.553038	0.817163	1.277151	1.325437	0.617058
X(90) =	0.094852	0.154184	1.174162	0.148151	0.237355

http://www.lactamme.polytechnique.fr/Mosaic/descripteurs/FloatingPointNumbers.01.Fra.html

3- Les formats de fichiers

• fichier = stockage d'information => lié au système d'exploitation

- en 2019, on peut transférer en mémoire l'intégralité de beaucoup de fichiers de la vie courante !
- les mémoires SSD comme support de stockage permanents

• codage binaire ou textuel
• metadonnées ou non
• structure ouverte ou non (open source / secret industriel)

Pour l'enseignement :

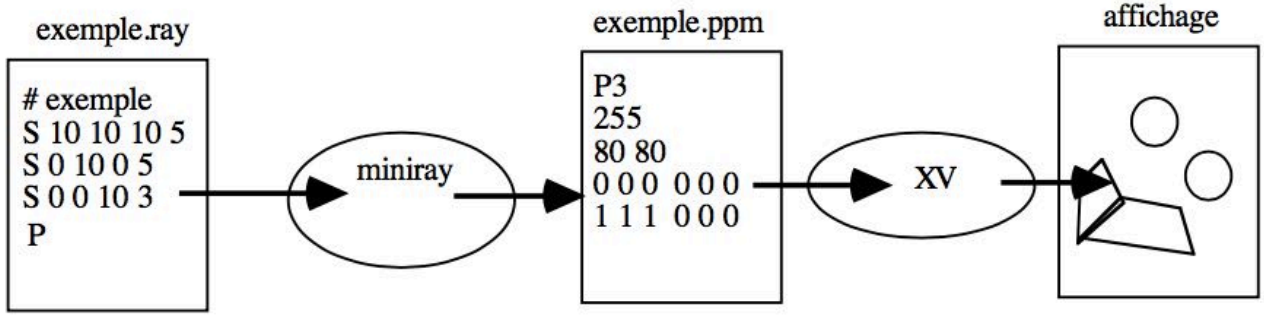
- les formats textuels à structure triviale : CSV, PPM pour les images, OBJ pour la 3D
- les formats textuels à la « XML » : SVG, VRML, JSON
- les formats binaire « plats » avec librairie associée : ex de WAV pour le son

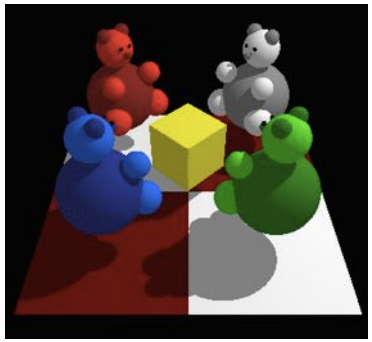
- en Python (et Java etc.) les fichiers sont accessibles via des objets/classes
- on peut utiliser les redirections Unix pour éviter de coder avec les fonctions d'E/S

Le format PPM (portable pix-map) et dérivés (PBM, PGM)

```
P3
#exemple.ppm
4 4
255
0 0 0          0 0 0          0 0 0      100      0
100
0 0 0          0 100 50       0 0 0      0 0 0
0 0 0          0 0 0          0 100 50   0 0 0
100 0 100      0 0 0          0 0 0      0 0 0
```

Exemple de projet au Cnam (NFP135) : miniray

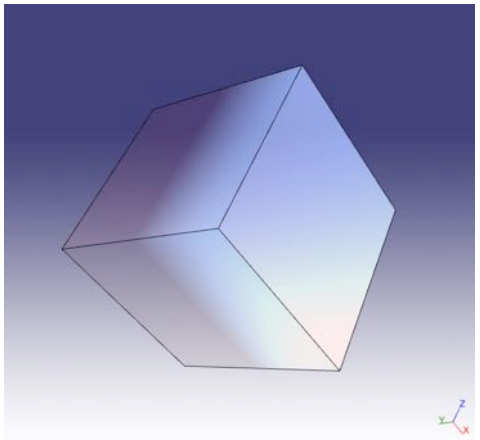




http://jasmin.cnam.fr:8081/TPVARI/01-02/

- Exemple de OBJ

```
v 0 0 0
v 1 0 0
v 1 1 0
v 0 1 0
v 0 1 1
v 1 1 1
v 1 0 1
v 0 0 1
f 8 7 6 5
f 2 1 4 3
f 7 2 3 6
f 1 8 5 4
f 3 4 5 6
f 1 2 7 8
```



- Exemple de SVG

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/
1.1/DTD/svg11.dtd">
<svg width="600" height="600" xmlns="http://www.w3.org/2000/svg" version="1.1">
<rect x="50" y="200" width="50" height="40"  style=" fill:#26CD22; "/>
<rect x="150" y="200" width="50" height="40"  style=" fill:#26CD22; "/>
<rect x="250" y="200" width="50" height="40"  style=" fill:#26CD22; "/>
</svg>
```

Les fichiers CSV (« comma separated values »)

communes-simple-metro.csv

	A	B	C	D	E	F	G	H	I
1	1	1	ozan	618	6.6	4.91667	46.3833	170	205
2	2	1	cormoranche	1058	9.85	4.83333	46.2333	168	211
3	3	1	plagne	129	6.2	5.73333	46.1833	560	922
4	4	1	tossiat	1406	10.17	5.31667	46.1333	244	501
5	5	1	pouillat	88	6.23	5.43333	46.3333	333	770
6	6	1	torcieu	698	10.72	5.4	45.9167	257	782
7	7	1	replonges	3500	16.6	4.88333	46.3	169	207
8	8	1	corcelles	243	14.16	5.58333	46.0333	780	1081
9	9	1	peron	2143	26.01	5.93333	46.2	411	1501
10	10	1	relevant	466	12.38	4.95	46.0833	235	282
11	11	1	chaveyriat	927	16.87	5.06667	46.1833	188	260
12	12	1	vaux en bug	1169	8.22	5.35	45.9167	252	681
13	13	1	maillat	668	11.31	5.55	46.1333	497	825
14	14	1	faramans	681	11.22	5.11667	45.9	255	306
15	15	1	beon	373	10.3	5.75	45.8333	228	1412
16	16	1	saint bernard	1375	3.15	4.73723	45.945	167	198
17	17	1	rossillon	148	8.07	5.6	45.8333	324	1022
18	18	1	pont d ain	2627	11.22	5.33333	46.05	232	314
19	19	1	nantua	3713	12.79	5.61667	46.15	427	1125
20	20	1	chavannes su	680	16.55	5	46.4333	175	218
21	21	1	neuville les d	1504	26.59	5	46.1667	210	271
22	22	1	flaxieu	63	2.79	5.73333	45.8167	225	385
23	23	1	hotonnes	306	28.4	5.68333	46	636	1338
24	24	1	saint sorlin e	1061	9.07	5.36667	45.8833	196	700
25	25	1	songieu	130	20.58	5.7	45.9667	567	1275
26	26	1	virieu le petit	309	16.36	5.71667	45.9	419	1524
27	27	1	saint denis e	2178	2.61	5.33333	45.95	234	338
28	28	1	charnoz sur s	911	6.62	5.21667	45.8667	203	244

- très ancien
- en français, c'est compliqué de séparer les champs avec la virgule...

Metadonnées ?

- Le format TIFF (Tagged Image File Format)
  - format initial spécifié par Aldus en 1986; évolution vers TIFF 6.0, établi en 1992 (Adobe)
  -format générique de représentation d'images, de 1 à 24 bits
  - intègre les méthodes de compression CCITT, JPEG et LZW
  - format à la fois puissant, très flexible et extensible
  - mais complexe avec des implémentations souvent incomplètes ou même erronées :

```
% /usr/local/bin/tiffinfo 4KY15.18.341.N.tif
TIFF Directory at offset 0x2d54
  Image Width: 2772 Image Length: 4235
  Resolution: 96, 96
  Compression Scheme: CCITT Group 4
  Orientation: row 0 top, col 0 lhs
  Planar Configuration: single image plane
```

```
% /usr/local/bin/tiffinfo 8PY9.1/0178.W328
TIFF Directory at offset 0x8
  Subfile Type: (0 = 0x0)
  Image Width: 6592 Image Length: 4672
  Resolution: 400, 400 pixels/inch
  Bits/Sample: 1
  Compression Scheme: CCITT Group 4
  Photometric Interpretation: min-is-white
  Thresholding: 64 (0x40)
  FillOrder: msb-to-lsb
  Date & Time: "1998:11:17 13:16:23"
  Software: "PMCompress version 3.00"
  Samples/Pixel: 1
  Rows/Strip: 4672
  Planar Configuration: single image plane
```

Doc et APIs : http://www.libtiff.org

Spec. de TIFF 6.0 : http://partners.adobe.com/asn/developer/pdfs/tn/TIFF6.pdf

4- La compression

- compression sans perte = réversible

  = supprimer la redondance dans les données

=> ça marche pour tous les types de données?

- compression avec perte = irréversible

  = supprimer ce que l'utilisateur peut tolérer

⇒ ça marche pour tous les types de données?

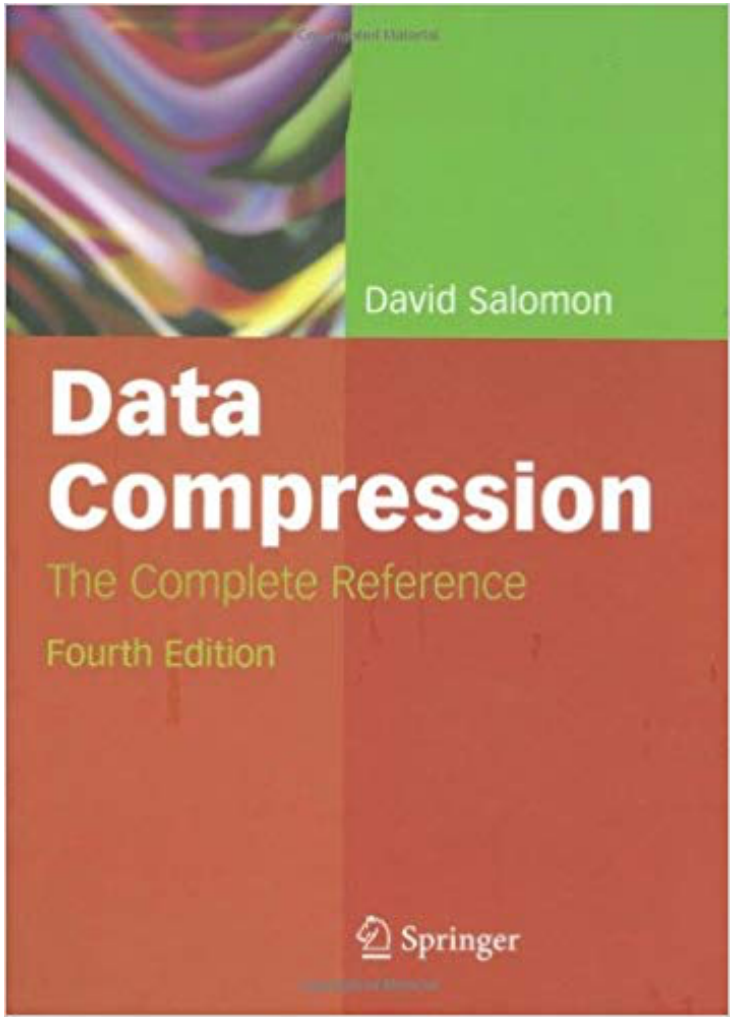
- compression (stenos) et transmission (telos) ?

600 x 600 pixels de 256 niveau de gris (q = 8bits) aléatoire => 508 Ko en PNG

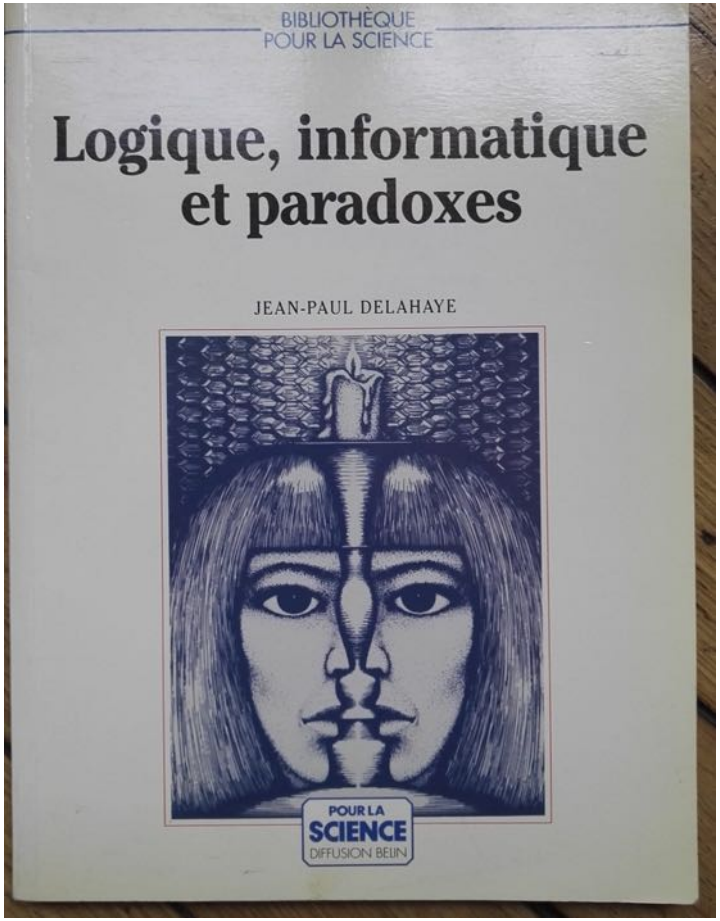
au lieu de => COMBIEN ?

LE HASARD N'EST PAS COMPRESSIBLE (c'est sa définition, cf Kolmogorov-Chaitin)

Livre de référence



A lire sur le hasard (chapitre 4)



+ ses autres livres (sur π, premiers, jeux)

Compression sans perte : ex. de CCITT groupe 3 (1980)

FAX : une barette de 1728 pixels (A4:204 dpi) qui scanne
en noir et blanc le document verticalement à 100 ou 200 dpi

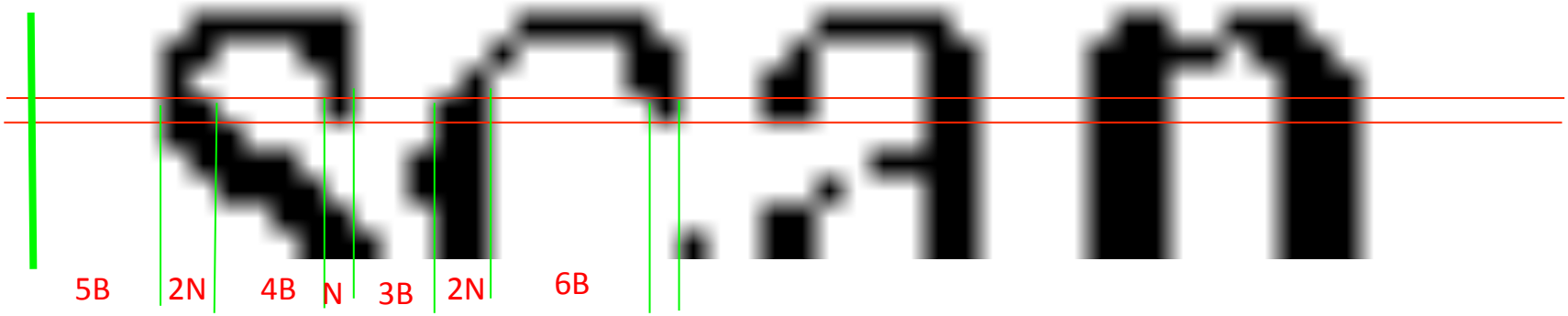
1 A4 = 1728 * 1188 = 2,05 Mpix = 2,05 Mb
débit du FAX : 4800 b/s => temps de transmission = 7 minutes !

avec la compression, on ramène (souvent) ce temps à 1 minute

⇒exploitation de la **cohérence de l'image** : entre points successifs
d'une même ligne + entre 2 lignes successives (groupe 4)

algo. limité par les contraintes sur l'appareil : petit, pas cher
+ gestion des erreurs de transmission + décompression progressive

Parcour de la ligne de balayage ("scanline") :

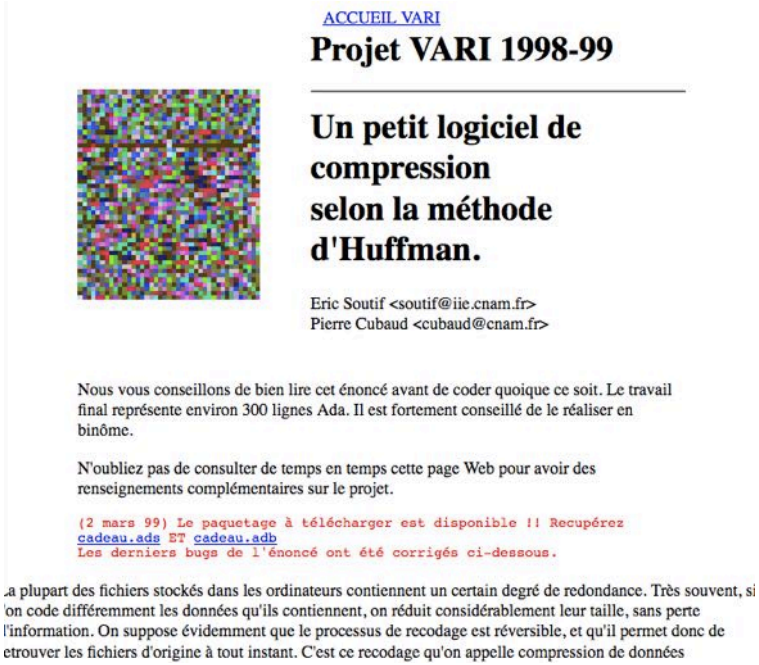


A chaque train ("run") de pixel (long ≤ 63), on associe un code d'Huffman pré-établi (de 2 à 12 bits). Pour les trains >12, on donne un code spécifique.

Long. train	blanc	noir
0	00110101	0000110111
1	000111	010
2	0111	11
3	1000	10
4	1011	011
5	1100	0011
6	1110	0010

etc.

Algorithme d'Huffman => voir Wikipedia et :



ACCUEIL VARI

Projet VARI 1998-99

Un petit logiciel de compression selon la méthode d'Huffman.

Eric Soutif <soutif@iie.cnam.fr>
Pierre Cubaud <cubaud@cnam.fr>

Nous vous conseillons de bien lire cet énoncé avant de coder quoique ce soit. Le travail final représente environ 300 lignes Ada. Il est fortement conseillé de le réaliser en binôme.

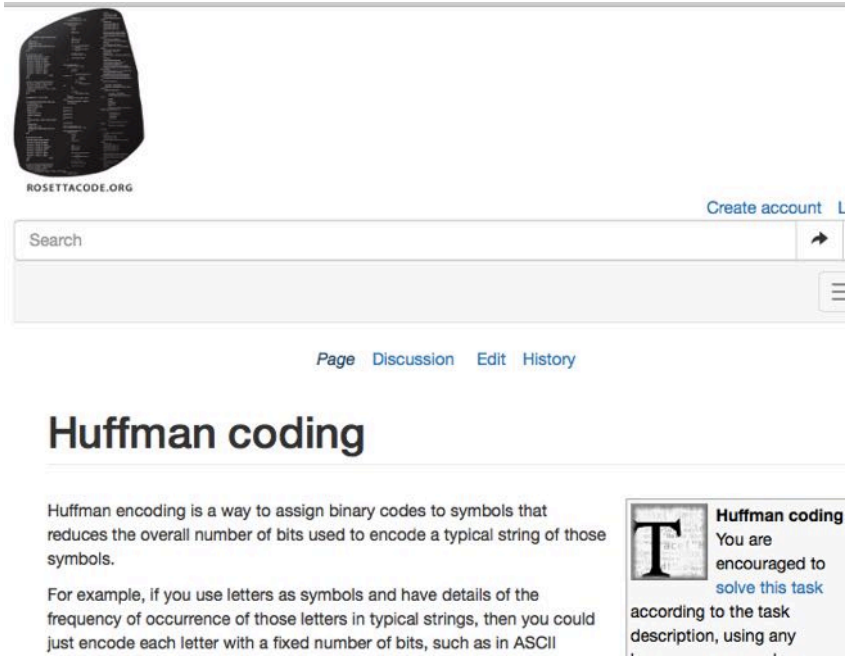
N'oubliez pas de consulter de temps en temps cette page Web pour avoir des renseignements complémentaires sur le projet.

(2 mars 99) Le paquetage à télécharger est disponible !! Recupérez cadeau.ads ET cadeau.adb
Les derniers bugs de l'énoncé ont été corrigés ci-dessous.

.a plupart des fichiers stockés dans les ordinateurs contiennent un certain degré de redondance. Très souvent, si 'on code différemment les données qu'ils contiennent, on réduit considérablement leur taille, sans perte l'information. On suppose évidemment que le processus de recodage est réversible, et qu'il permet donc de etrouver les fichiers d'origine à tout instant. C'est ce recodage qu'on appelle compression de données

http://jasmin.cnam.fr:8081/TPVARI/98-99/

Implémentation dans ±50 langages !



ROSETTACODE.ORG

Create account L

Search

Page Discussion Edit History

Huffman coding

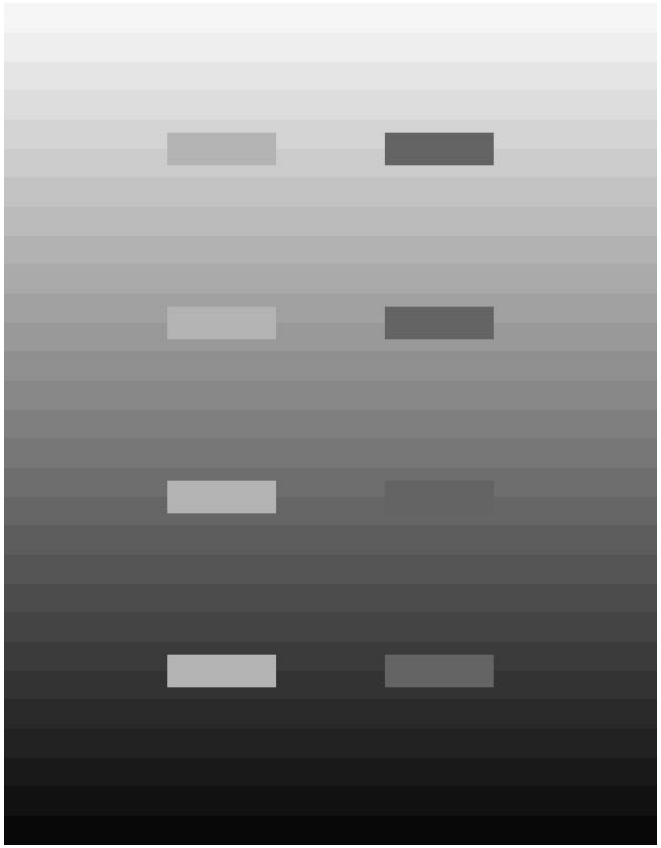
Huffman encoding is a way to assign binary codes to symbols that reduces the overall number of bits used to encode a typical string of those symbols.

For example, if you use letters as symbols and have details of the frequency of occurrence of those letters in typical strings, then you could just encode each letter with a fixed number of bits, such as in ASCII

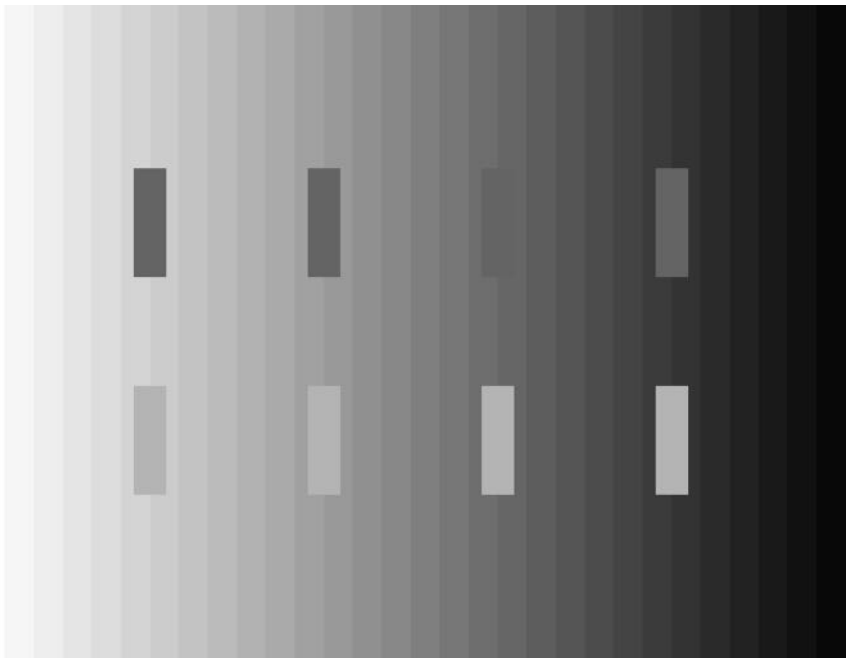
Huffman coding
You are encouraged to solve this task according to the task description, using any

https://rosettacode.org/wiki/Huffman_coding

Le RLE (run length encoding) sur une drôle d'image

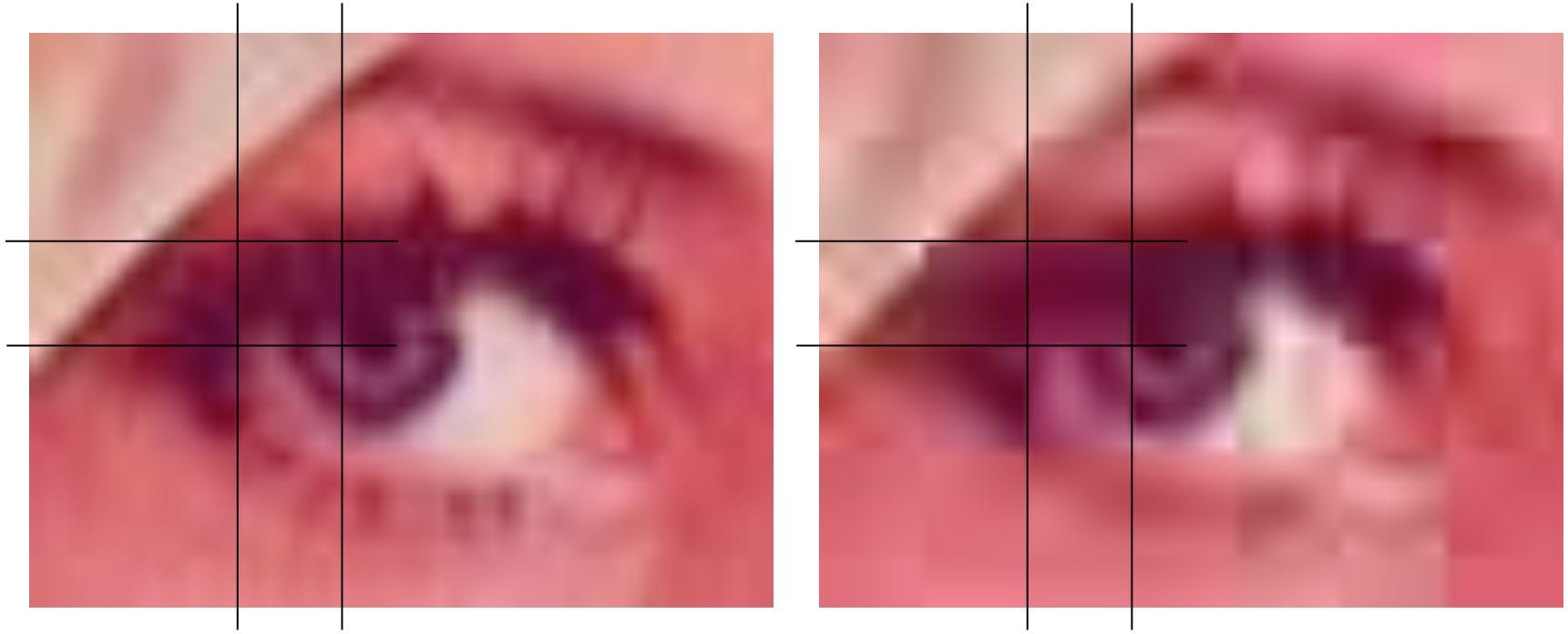


845x1124 pixels
en GIF : 15Ko
en PNG : 23 Ko



1124x845 pixels
en GIF : 51 Ko
en PNG : 21 Ko

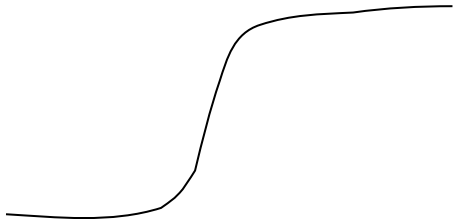
Compression avec perte : ex. de JPEG

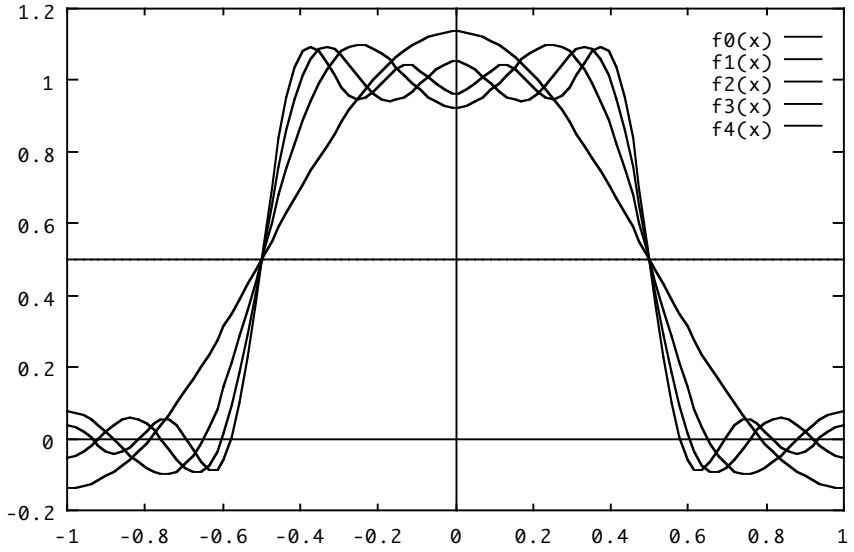


- Première passe de réduction de la chrominance
- Découpage de l'image en blocs de 8*8 pixels
- Calcul de la DCT : 64 coefficients pour décrire le bloc
- Selon le taux de compression,

on choisit de retenir les N≤64 plus importants

profil de l'image
au contour :

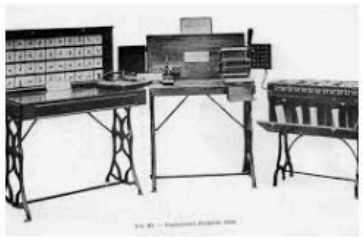




5- Chercher / trier

Trieuse Documentaire de cartes perforées,...
histoireinform.com

LES MACHINES A CARTE...
feb-patrimoine.com



LES MACHINES A CARTES PERFORÉ...
feb-patrimoine.com

Musée virtuel de l'informatique | Fig...
aconit.inria.fr

LES MACHINES A CARTES PERF...
feb-patrimoine.com

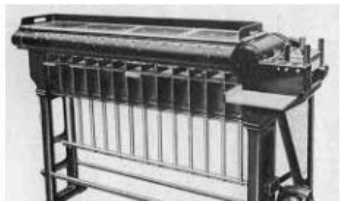
La machine d'Hollerith (40 ans de l'INRIA,...
flickr.com

LES MACHINES A CARTE...
feb-patrimoine.com

Herman Hollerith Photos & Herman H...
alamyimages.fr

LES MACHINES A CARTES PER...
feb-patrimoine.com

Hollerith Photos & Hollerith Images - Ala...
alamyimages.fr

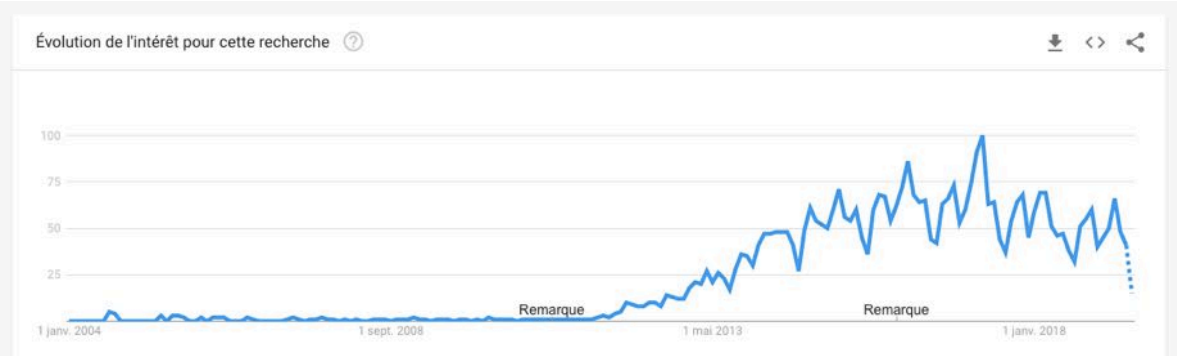


La trieuse d'Hollerith pour le recensement de aux USA

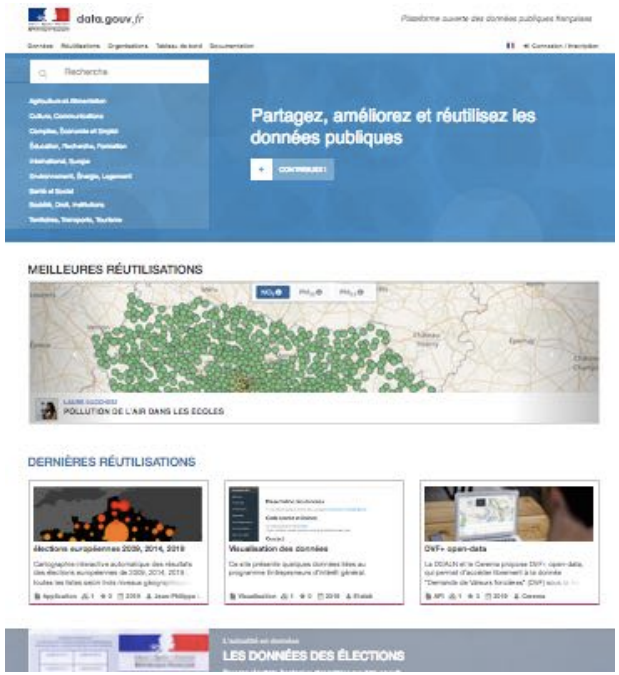
=> visite à planifier au Musée des Arts & Métiers

• explosion des données (« big data ») : l'usage de la donnée crée de la donnée ...

trends.google.fr

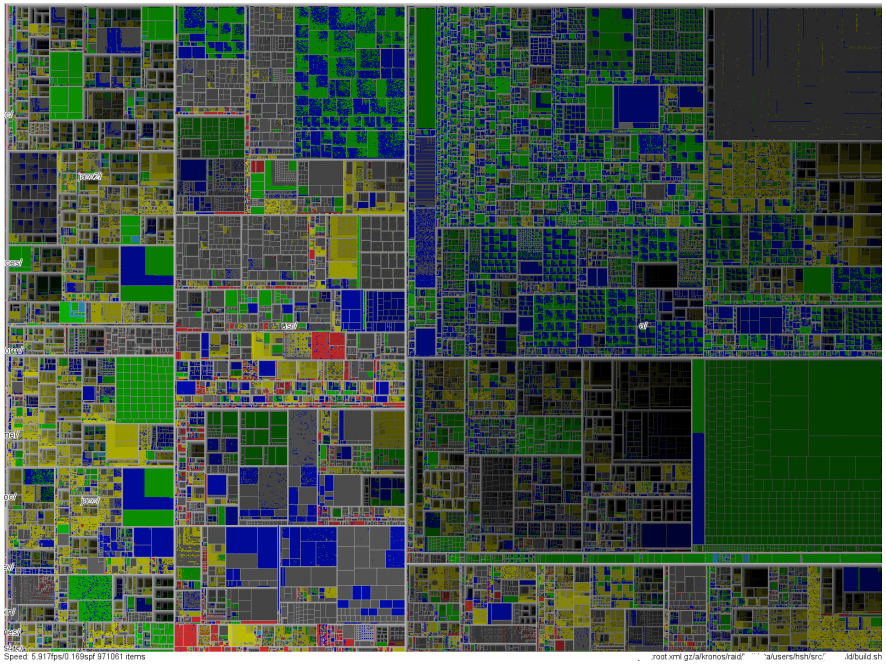


• « open data » exemple du site data.gouv.fr



plus anciens en sciences dures (astronomie, etc) et « digital humanities »

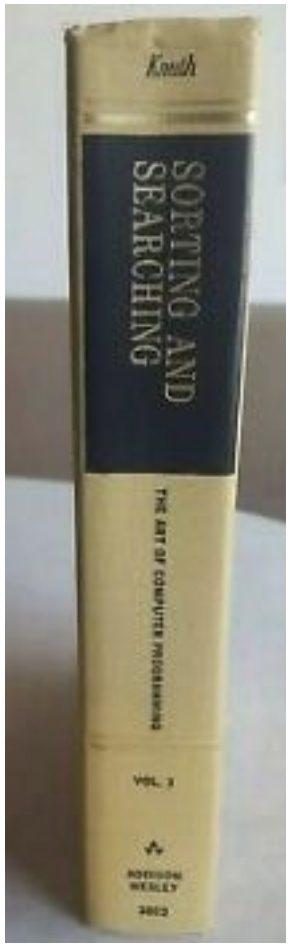
- la visualisation de données (dataviz, infoviz)



code source et infos :
http://www.cs.umd.edu/hcil/millionvis/

- la sérendipité ?

D. Knuth « The art of computer programming, vol 3 : Sorting & searching » 1973



the other basic structural ideas.

The title "Sorting and Searching" may sound as if this book is only for those systems programmers who are concerned with the preparation of general-purpose sorting routines or applications to information retrieval. But in fact the area of sorting and searching provides an ideal framework for discussing a wide variety of important general issues:

- How are good algorithms discovered?
- How can given algorithms and programs be improved?
- How can the efficiency of algorithms be analyzed mathematically?
- How can a person choose rationally between different algorithms for the same task?
- In what senses can algorithms be proved "best possible"?
- How does the theory of computing interact with practical considerations?
- How can external memories like tapes, drums, or disks be used efficiently with large databases?

Indeed, I believe that virtually *every* important aspect of programming arises somewhere in the context of sorting or searching!

un secret: existe en pdf sur http://www.softouch.on.ca/kb/data/

• rechercher un item dans une liste L de N items => o(N) ?

• si la liste est déja triée : rechercher par dichotomie
o(log N) au lieu de o(N)

• effectuer un tri ?
- tri par **insertion** (ou bulle, etc.) : o(N^2)

```
D = 0
repeter
    chercher le min dans L(D...N-1)  => case M avec 0<=M<=N
    permuter L(M) et L(D)
    D = D+1
jusqu'a D=N
```

- tri par fusion de liste : o(log N)

L'algorithme est naturellement décrit de façon récursive.

1. Si le tableau n'a qu'un élément, il est déjà trié.
2. Sinon, séparer le tableau en deux parties à peu près égales.
3. Trier récursivement les deux parties avec l'algorithme du tri fusion.
4. Fusionner les deux tableaux triés en un seul tableau trié.

Très nombreuses variantes de tri ! cf wikipedia, traité Cormen & Rivest etc.

plexités doivent être interprétées à l'aide d'un grand O de Landau. Il est supposé que les opérations élémentaires comme les comparaisons et les échang ent être effectués en temps constant.

Tableau comparatif des tris procédant par comparaisons

Nom	Cas optimal	Cas moyen	Pire des cas	Complexité spatiale	Stable
Tri rapide	$n \log n$	$n \log n$	n^2	$\log n$ en moyenne, n dans le pire des cas ; variante de Sedgewick : $\log n$ dans le pire des cas	Non
Tri fusion	$n \log n$	$n \log n$	$n \log n$	n	Oui
Tri par tas	$n \log n$	$n \log n$	$n \log n$	1	Non
Tri par insertion	n	n^2	n^2	1	Oui
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$	Non
Tri par sélection	n^2	n^2	n^2	1	Non
Timsort	n	$n \log n$	$n \log n$	n	Oui
Tri de Shell	n	$n \log^2 n$ ou $n^{3/2}$	$n \log^2 n$pour la meilleure suite d'espacements connue	1	Non
Tri à bulles	n	n^2	n^2	1	Oui
Tri arborescent	$n \log n$	$n \log n$	$n \log n$ (arbre équilibré)	n	Oui
Smoothsort	n	$n \log n$	$n \log n$	1	Non
Tri cocktail	n	n^2	n^2	1	Oui
Tri à peigne	n	$n \log n$	n^2	1	Non
Tri pair-impair	n	n^2	n^2	1	Oui

implementation dans tous les langages actuels

As of Perl 5.8, merge sort is its default sorting algorithm (it was quicksort in previous versions of Perl). In Java, the Arrays.sort() methods use merge sort or a tuned quicksort depending on the datatypes and for implementation efficiency switch to insertion sort when fewer than seven array elements are being sorted.[17] The Linux kernel uses merge sort for its linked lists.[18] Python uses Timsort, another tuned hybrid of merge sort and insertion sort, that has become the standard sort algorithm in Java SE 7 (for arrays of non-primitive types),[19] on the Android platform,[20] and in GNU Octave.[21]

⇒plus aucun programmeur ne code ce genre de traitement !
mais comprendre le fonctionnement reste essentiel

TRI PAR INSERTION

vérification de
tps exécution = α (nombre d'échanges)^2

tps (s) \ n	10	100	500	1000	1500	2000
essai 1	<0.1	<0.1	4	19	41	74
2	//	//	4	19	41	74
3	//	//	5	17	40	74
4	//	//	5	17	41	74
moyenne	<0.1	<0.1	4.5	18	40.75	74
n^2	100	10.000	250000 / 18	1000000 / 18.2	1200000 / 18.	4 000 00 / 18.5

$\frac{tps}{n^2} = \alpha$

$\Rightarrow$ $tps = 18\ n^2\ (\mu s)$

Sur mon Leanord en 1987-8

Et en Python sur mon mac en 2019 ?

unix : grep et sort

```
07/30/97 15:36:23 DONNE cocuage2 POUR pento.esil.univ-mrs.fr (139.124.4.66) AVEC OmniWeb/2.6-beta-4 OWF/1.0
07/30/97 15:47:22 DONNE jacques1 POUR prithivi (163.173.136.11) AVEC Mozilla/3.01 (X11; I; SunOS 5.5 sun4c)
07/30/97 15:52:04 DONNE commerce1 POUR dial066.ppp.lrz-muenchen.de (129.187.24.66) AVEC OmniWeb/2.5-beta-6 OWF/1.0
07/30/97 16:07:35 DONNE conscrit2 POUR ns.cci.dk (194.239.27.66) AVEC Mozilla/2.0 (compatible; MSIE 3.0;
   Windows 95) via Harvest Cache version 1.4pl3
07/30/97 16:09:00 DONNE rldased3 POUR rkawv247.kau.roche.com (145.245.98.98) AVEC Mozilla/3.01 (Win16; I)
07/30/97 16:11:04 DONNE DICO/prenoms POUR ifp.ifp.fr (156.118.212.2) AVEC Mozilla/3.01 (X11; I; SunOS 5.5.1 sun4u)
07/30/97 16:13:09 DONNE DICO/pays POUR ifp.ifp.fr (156.118.212.2) AVEC Mozilla/3.01 (X11; I; SunOS 5.5.1 sun4u)
```

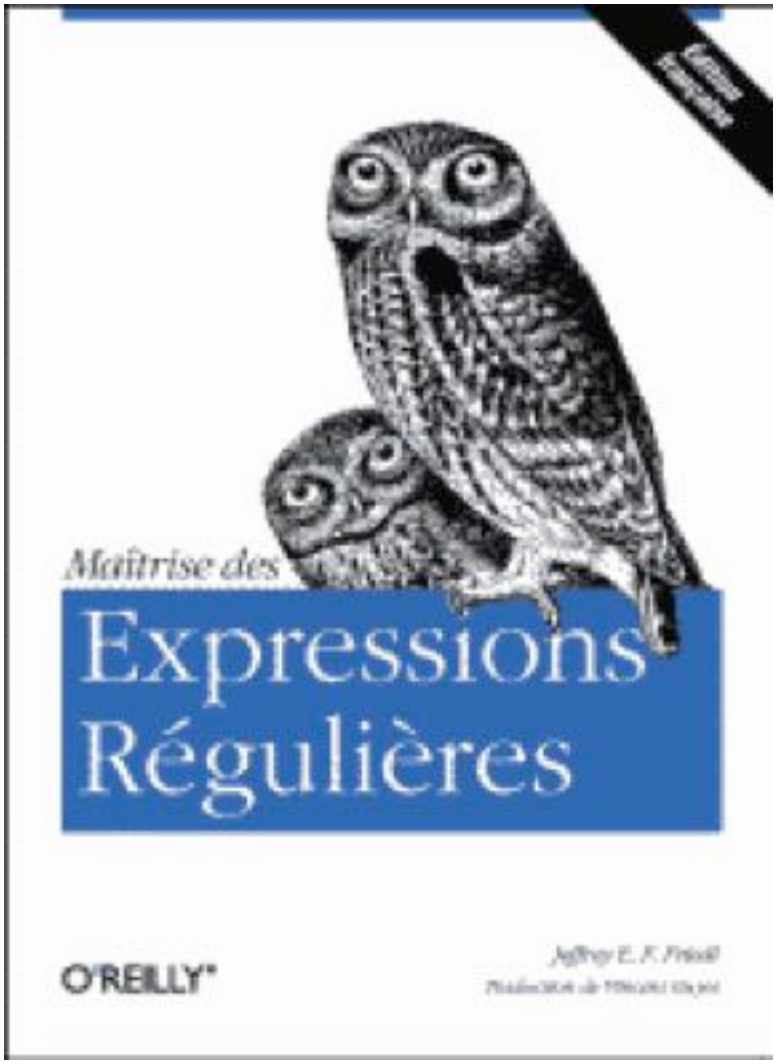
...

Enchaîner les traitements : l'art du « pipe »

```
awk '{print $6}' logdon | awk -F. '{print $7,$6,$5,$4,$3}' |
  awk '{print $1}' | grep -v '^[0123456789]' | sort | uniq -c | sort -nr
```

```
209479 fr
63549 com
61941 net
26769 ca
21707 be
16277 ch
13267 it
10855 de
8088 jp
7863 es
6803 edu
3743 uk
3197 br
3085 ru
```

Expressions régulières : utilisées au départ dans les commandes Unix



[abc...]
matches any of the characters *abc*...
[^abc...]
matches any character except *abc*...
r1|r2
matches either *r1* or *r2* (alternation).
r1r2
matches *r1*, and then *r2* (concatenation).
r+
matches one or more *r*'s.
r*
matches zero or more *r*'s.
r?
matches zero or one *r*'s.
(r)
matches *r* (grouping).
r{n,m}
matches at least *n*, *n* to any number, or *n* to *m* occurrences of *r* (interval expressions).
\b
matches the empty string at either the beginning or the end of a word.

etc ...