PF1

numéra° posi°n en base b

$n_4 \cdot b^3 + n_3 \cdot b^2 + n_1 \cdot b + n_0$

méthodes de conversion

encadrem^t succ.

ex. $(329)_{10} \Rightarrow$ base 5

- $250 < 329 < 375$
  2.5^3 $\quad$ 3.5^3
- $329 - 250 = 79$
- $79 - 75 = 4$
  3.5^2
- $4 - 4 = 0$
  4.5^0

$\Rightarrow 2.5^3 + 3.5^2 + 4.5^0 \Rightarrow (2304)_5$

Puissances (colonnes) / Facteurs (lignes)

	0	1	2	3
1	1	5	25	125
2	2	10	50	250
3	3	15	75	375
4	4	20	100	

divisions successives (10 -> b)

- on divise n par la base d
- on collecte le reste r
- on recommence avec q
- on arrête qnd q est nul

$$\begin{array}{r|l} n & d \\ \hline r & q \end{array}$$

ex.
$$\begin{array}{r|l} 329 & 5 \\ 325 & 65 \\ \hline 4 & \end{array} \quad \begin{array}{r|l} 65 & 5 \\ 65 & 13 \\ \hline 0 & \end{array} \quad \begin{array}{r|l} 13 & 5 \\ 10 & 2 \\ \hline 3 & \end{array} \quad \begin{array}{r|l} 2 & 5 \\ 0 & 0 \\ \hline 2 & \end{array}$$

$\Rightarrow (329)_{10} = (2304)_5$

par recomposi° (b -> 10)

ex. $(2304)_5 \to$ base 10

$2.5^3 + 3.5^2 + 0.5^1 + 4$

$= ((2 \times 5 + 3) \times 5 + 0) \times 5 + 4$ <= Horner

10

13

65

65

325

329

dépliage - repliage ($b^n \to b^m$)

ex. $(21301)_4$ en base 8

- $4 = 2^2$ & $8 = 2^3$

dépliage

2	1	3	0	1
10	01	11	00	01

1001110001 =>

1	1	6	1
1	001	110	001

repliage

Rappel : compter en binaire

- 10 -> 2
- 100 -> 4
- 110 -> 6
- 1000 -> 8
- 1010 -> 10
- 1100 -> 12
- ...
- 10000 -> 16
- 100000 -> 32
- 1000 000 -> 64
- 10 000 000 -> 128
- 100 000 000 -> 256
- 1 000 000 000 -> 512

PF1

addi° a multiplica° en base b

· On s'appuie sur les tables en base b

ex. en base 16

$$\begin{array}{cccccc} & & {}^{1} & {}^{1} & & \\ & 1 & 8 & A & 4 \\ + & & & 7 & C \\ \hline & 1 & 9 & 2 & 0 \end{array} \qquad \begin{array}{cccccc} & & 1 & 8 & A & 4 \\ \times & & & & 7 & C \\ \hline & 1 & 2 & 7 & B & 0 \\ + & A & C & 7 & C & 0 \\ \hline & B & E & F & 7 & 0 \end{array}$$

Représenta° des nb décimaux

$n_p . b^p + \ldots + n_1 . b^1 + n_0 + n_{-1} . \frac{1}{b} + n_{-2} . \frac{1}{b^2}$

↳ permet de convertir en base 10

multiplica° successives

ex¹ : $(29,25)_{10}$

Partie entière : 29 => 11101

Partie fraction' : 0,25 × 2 = 0,5
0,5 × 2 = 1,0

=> 0,25 => 0,01
=> 29,25 => 11101,01

ex² $(7,7)_{10}$

Partie entière : 7 => 111

Partie fraction' : 0,7 × 2 = 1,4
0,4 × 2 = 0,8
0,8 × 2 = 1,6
0,6 × 2 = 1,2
0,2 × 2 = 0,4

boucle

=> $0,7 => 0,1(0110)^{\omega}$
=> $7,7 => 111,1(0110)^{\omega}$

numéra° en machine

ordre lexicographiq

ex. sur 8 bits . 0110 1101 => 109 (non signé)

-> débordem^t : dépasse le nb max

-> signé : 1^er bit à gauche => bit de signe
↳ 2 représenta° de 0
↳ pb d'opéra° algo

complém^t à deux

· on code la valeur absolue

· si négatif
↳ on inverse ts les bits à partir du 1^er 1 (non inclus) en partant de la droite

ex. 16 => 0001 0000   -16 => 1111 0000

· débordem^t si neg + neg = pos
pos + pos = neg

Représenta° à virgule fixe

· partie entière sur n bits

· partie fract°nelle sur p bits

· un bit pour le signe

ex. n = 3 p = 2

0 000 00 => 0
0 000 01 => 0,25
10 0,5
11 0,75
0 001 00 1

=> très limité

PF1

Représenta° à virgule flottante

· norme IEEE 754

1 bit signe	8 bits exposant	23 bits valeur

ex + étapes : $-118,625$

1. on convertit en binr

$-118,625 \Rightarrow -1110110,101$

2. bit de signe

pos $= 0$ neg $= 1$

3. mantisse $\Rightarrow$ écrire sous la forme $1,\text{val}.2^n$

$1110110,101 \Rightarrow 1,110110101.2^6$

4. exposant

$6 + 127 = 133 \Rightarrow 1000\ 0110$

ici en 8 bits : $2^{8-1} - 1 = 127$

$\Rightarrow$ $\underline{1}$ $\underline{1000\ 0110}$ $\underline{11011010100000000000000}$

cas spéciaux

· $e = 2^{p-1} - 1$ & $m = 0$ $\Rightarrow \pm\infty$

· $e = 2^{p-1} - 1$ & $m \neq 0$ $\Rightarrow$ NaN

arrondi

· 3 types : vers $\pm\infty$

vers 0

au plus près $\longrightarrow$ $M_{23}\ M_{24}\ M_{25}\ M_{26}$

· si $M_{24} = 0$, on tronque

· si $M_{23} = M_{24} = M_{25}$ & $M_{24} = 1$, on tronque

· sinon on arrondit

PF1

numérisa° & codage

code longueur fixe

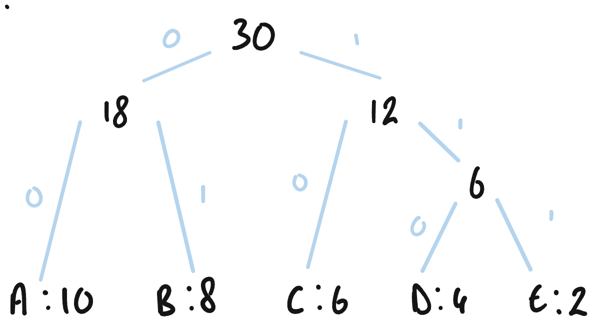
Pour coder en bin^r il faut $k \geqslant \log(|A|)$

nb de lettre

code longueur variable

Code de Huffman - par fréquence

ex.



=> A : 00 B : 01 C : 10 D : 110 E : 111

Compression

- Quotient Q : $\frac{v_i}{v_f}$

- Taux : $\frac{1}{Q} = \frac{v_f}{v_i}$

Codage RLE : coder la suite des longueurs

ex.

Cryptographie

code de Cesar

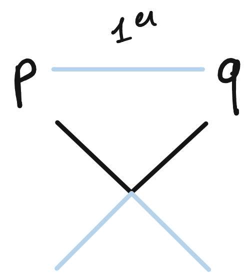
Permuta° circulaire (voir : chiffre de Vernam)

syst^m asymétriq

2 clefs : publiq & privée

coder (n,e) — décoder (n,d)

$M^e \% (n)$ — $M^d \% (n)$



$p \times q = n$ $(p-1)(q-1) = \phi(n)$

$e < \phi(n)$ & $PGCD(e, \phi(n)) = 1$

$d \Rightarrow (e \times d) \% \phi(n) = 1$

ex.

$42^7 \pmod{77}$

- $42 \% 77 = 42$
- $(42 \times 42) \% 77 = 70$
- $(42 \times 70) \% 77 = 14$
- ...
- $(42 \times 42) \% 77 = 70$

$70^{43} \pmod{77}$

- $70 \% 77 = 70$
- $(70 \times 70) \% 77 = 49$
- $(70)^4 \% 77 = (49 \times 49) \% 77$
- ...
- $70^{43} \% 77 = 70^{32} \times 70^8 \times (\ldots)$

PF1

contrôle des erreurs

CRC & bit de parité

- selon la parité des 1 sur 7 bits, on ajoute 0 (si pair) ou 1

distance de Hamming

- ≠ entre 2 mots bin^r (⊕)
- erreurs détectées : $e_d = d_H - 1$
  $d_H = e_d + 1$
- erreurs corrigées : $e_c = \frac{d_H - 1}{2}$
  $d_H = 2e_c + 1$

famille du code de Hamming

- bit de parité ds le code en posi° 2^n

ex. M = 0101

- 0101 => $P_1 \; P_2 \; 0 \; P_4 \; 1 \; 0 \; 1$

P_1 : prend en compte 1/2

P_2 : prend en compte 2/4

P_4 : prend en compte 4/8

P_1 : 011 => 0

P_2 : 001 => 1

P_3 : 101 => 0

cas d'erreurs :

on additionne les indices des bits de parité q ne sont pas bon

calcul proposi°nl & circuit numériq

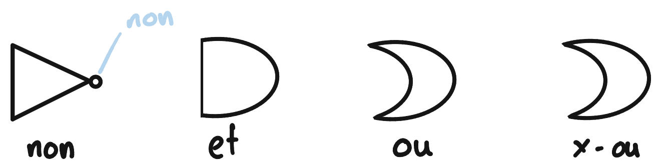
propriétés

- $p \Rightarrow r \Leftrightarrow \bar{p} \vee r$
- $(p \wedge q) \wedge \Leftrightarrow p \wedge (q \wedge r)$
- $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (q \wedge r)$ (& inv)
- $(p \vee q) \wedge p \Leftrightarrow p$
- $(p \wedge q) \vee p \Leftrightarrow p$ — absorp°
- $\overline{(p \wedge q)} \Leftrightarrow \bar{p} \vee \bar{q}$ de Morgan

Forme normale disjonctif · FND

↳ $\underbrace{(p \wedge q)}_{\text{↳ minterm}} \vee (r \wedge s)$

représenta° graphiq



code de Gray

ex.

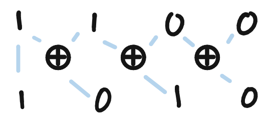


tableau de karnaugh

Permet d'exprimer les valeurs d'une fonc° booléenne & de les simplifier

ex. $A = (\bar{a} \wedge \bar{b}) \vee (a \wedge \bar{b}) \vee (a \wedge b)$

b \ a	0	1
0	1	1
1	0	1

$A = \bar{b} \vee a$

! simplifier avec des longueurs : 2^n