PF1 — Principes de Fonctionnement des machines binaires

Jean-Baptiste Yunès
Jean.Baptiste.Yunes@univ-paris-diderot.fr

Version 1.21

Numérisation

Numérisation

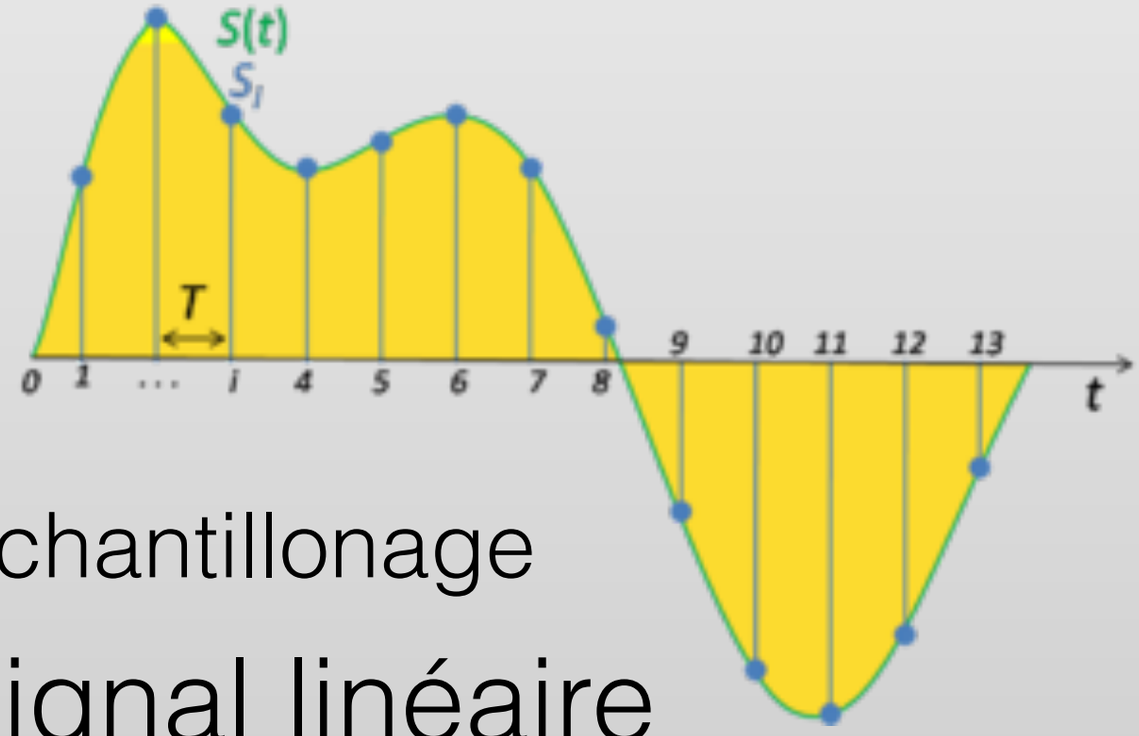
La **numérisation** désigne le processus consistant à convertir un signal analogique en données numériques

Il fait appel à l'**échantillonnage** qui consiste à sélectionner dans un ensemble un nombre fini d'éléments considérés comme représentatifs. L'échantillonnage est une approximation.

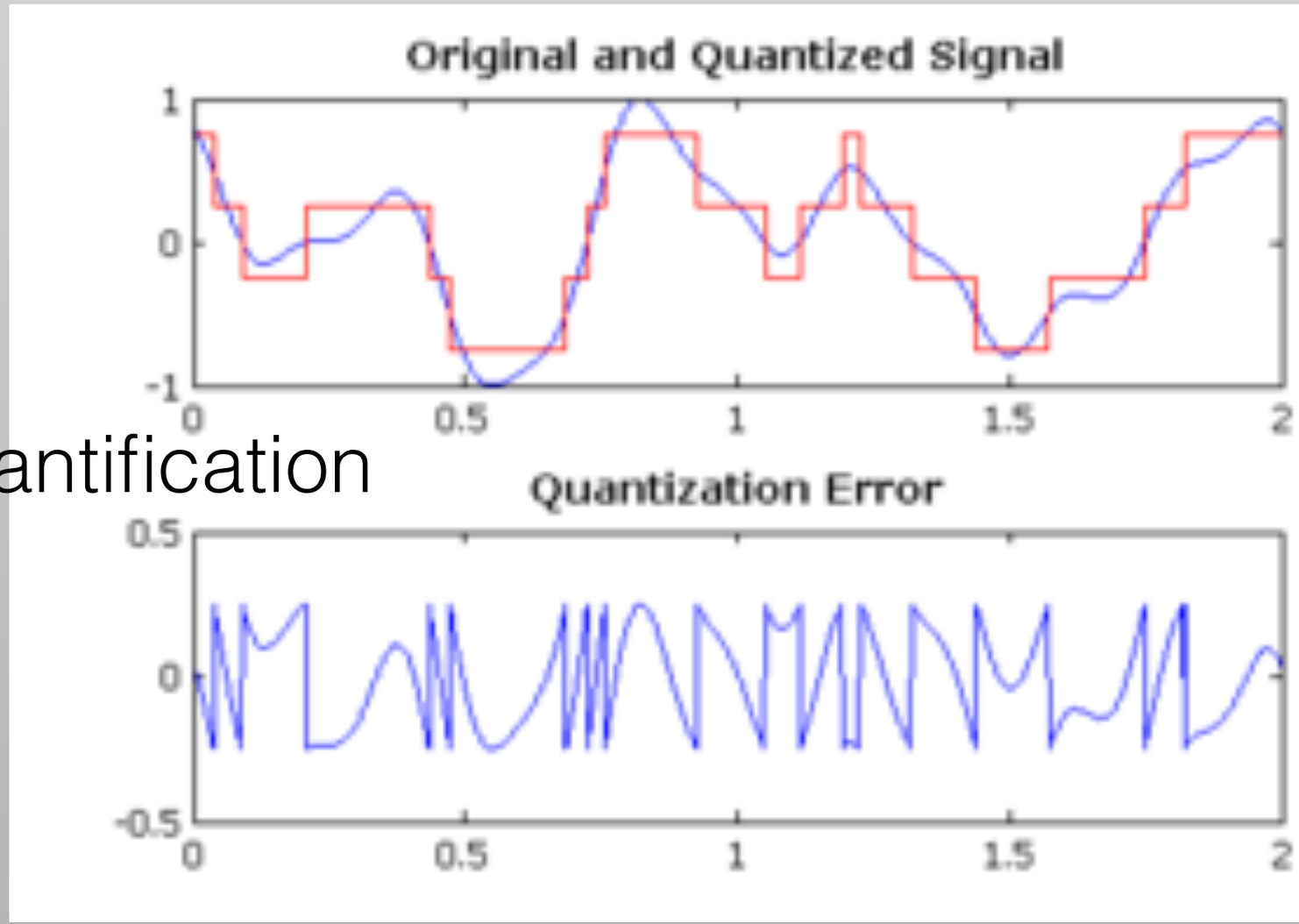
Il fait appel à la **quantification** qui consiste à associer à chaque élément prélevé une valeur numérique discrète.

Ces deux paramètres conditionnent la qualité l'approximation et donc de la reproduction du signal original.

Example d’un signal linéaire



échantillonage



quantification

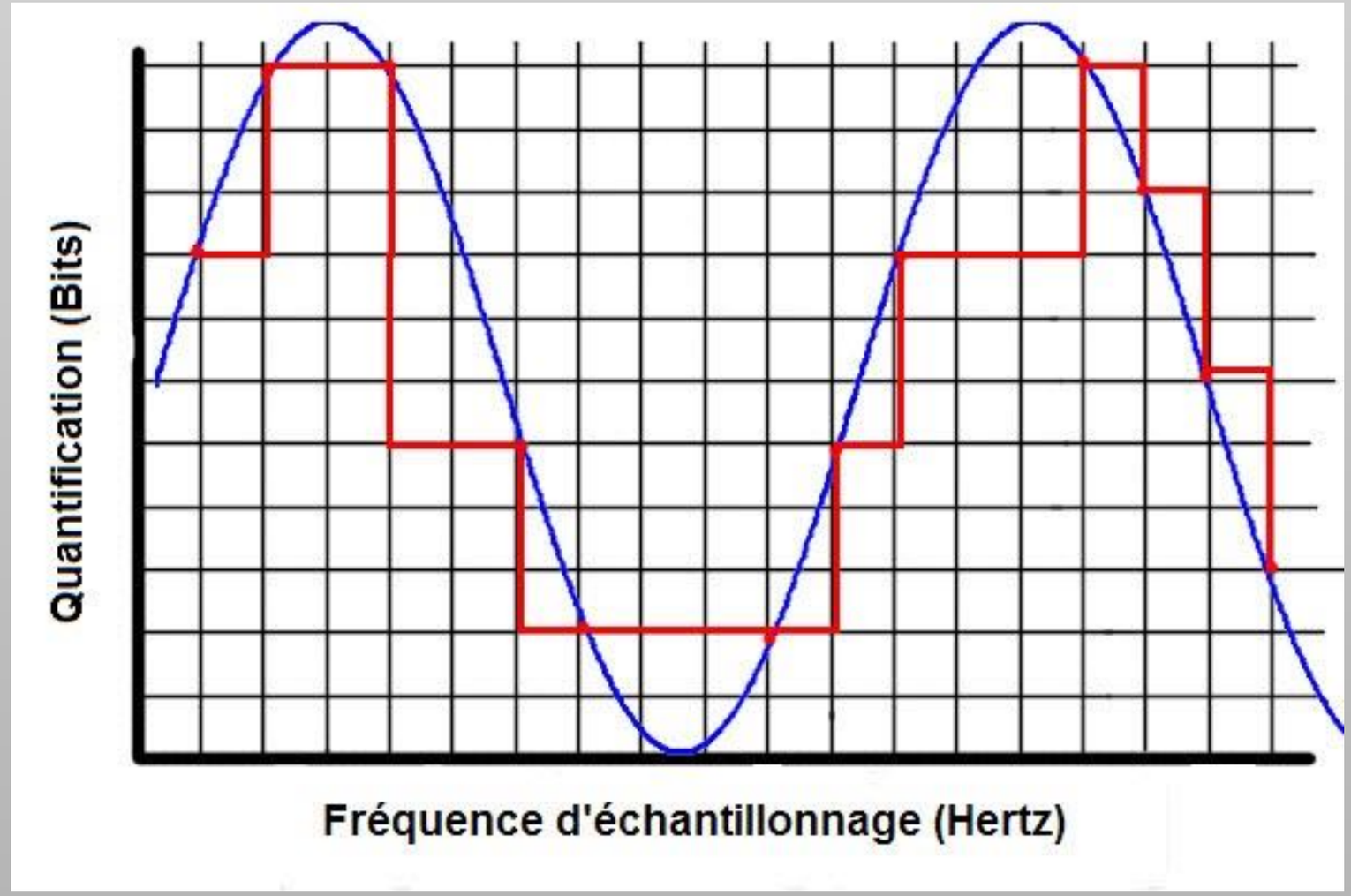
Reproduction du son :

avant 1980, reproduction de l'onde sonore par capture de sa forme analogique via un dispositif transformant la **pression acoustique** en **déplacement mécanique** et gravage dans un support (cire, plastique). Le dispositif étant facilement réversible (mouvement -> pression)

après 1980, numérisation de l'onde et enregistrement ou gravage de son codage

L’échantillonage est mesuré en Hertz (l’échantillonnage est très généralement effectué à intervalles réguliers - exception : VBR)

La finesse de la quantification est mesurée en bits



la reconstitution approximative d'un signal analogique depuis sa forme numérisée nécessite une fréquence d'échantillonnage au moins deux fois plus élevée que la largeur de la bande.
(Théorème de Nyquist-Shannon)

Par exemple, l'oreille humaine a une bande passante de 20Hz à 20kHz soit approximativement une largeur de bande de 20kHz, la reproduction d'un son destiné à l'oreille humaine nécessite alors une fréquence d'échantillonnage d'au moins 40kHz

La numérisation employée pour les CD est de ~44kHz avec une quantification sur 16 bits

le choix de 16 bits (14 suffisent) est justifié scientifiquement mais dépasse largement le cadre de ce cours (quantification logarithmique, rapport signal-bruit)

Compression

Code compresseur

il s'agit ici d'obtenir une représentation plus compacte

en vue de **transmission** (économie de bande passante)

en vue de **stockage** (économie d'espace)

Deux types de **compression** :

conservative ou **sans perte** : le message originel peut-être reconstruit à l'identique en inversant la fonction

ce type de compression est recherché avec du texte par exemple…

je peux vouloir compresser l'œuvre complète de Victor Hugo, mais je souhaite retrouver le texte d'origine!

Deux types de **compression** :

non conservative ou **avec perte** : le message originel n'est pas reconstruit à l'identique, mais un message similaire est obtenu à l'inversion

souvent le cas des images et des sons

il n'est pas nécessaire que des détails quasi-invisibles-sensibles soient conservés dans les photos prises par mon appareil…

cette compression permet d'obtenir de très bons taux de compression

Codage compressé d'un texte…

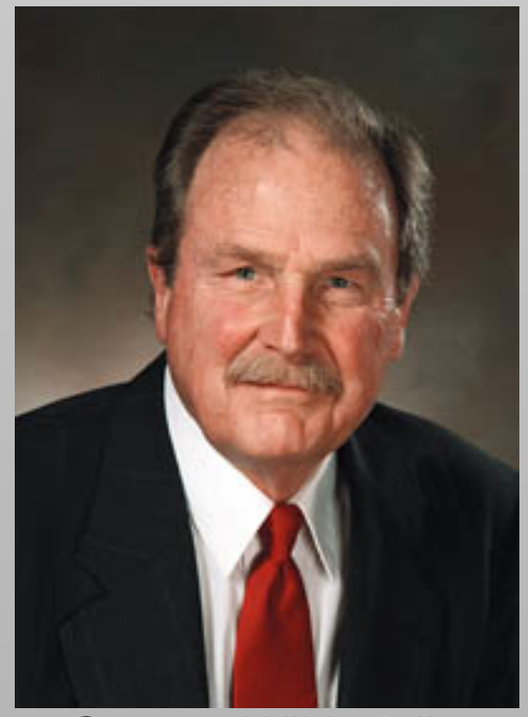
utilisation d'un code à longueur variable (fréquence des lettres), Huffman, Lempel-Ziv

création d'un code permettant d'encoder des groupes de lettres en fonction de la fréquence, calcul dynamique de la fréquence…

Code de Huffman

David Albert Huffman

1925—1999



Source Wikipédia

Idée :

- coder avec des mots de petite longueur les lettres les plus fréquentes,
- coder avec des mots de plus grande longueur les lettres les moins fréquentes

Ex. : trois lettres A, B, C avec A très fréquente, B moyennement et C rare

on peut utiliser quelque chose comme $\boldsymbol{\tau}$(A)=0, $\boldsymbol{\tau}$(B)=10 et $\boldsymbol{\tau}$(C)=11

soit 1 bit pour A, 2 pour B et C

Ainsi le texte AAABAAABBAAAAC serait codé par 00010000101000011 soit 18 bits.

Un codage ordinaire sur 2 bits/caractères aurait donné 14*2 soit 28 bits.

On obtient un taux de compression de 18/28 soit ~65%, 2/3…

À propos du **taux de compression**…deux façons habituelles de le définir :

- volume final / volume initial
- 1 - (volume final / volume initial)

Ex. : final : 80, initial : 100

- déf. 1 : 0.8 (taux faible, compression forte), ici il reste 80%
- déf. 2 : 0.2 (taux élevé, compression forte) ici on a gagné 20%

Il existe aussi le concept de **quotient de compression** :

volume initial / volume final

Ex. : 100/80, en le simplifiant on le note souvent comme 5:4, réduction de 5 à 4.

Comment obtenir un codage à partir des fréquences des lettres ?

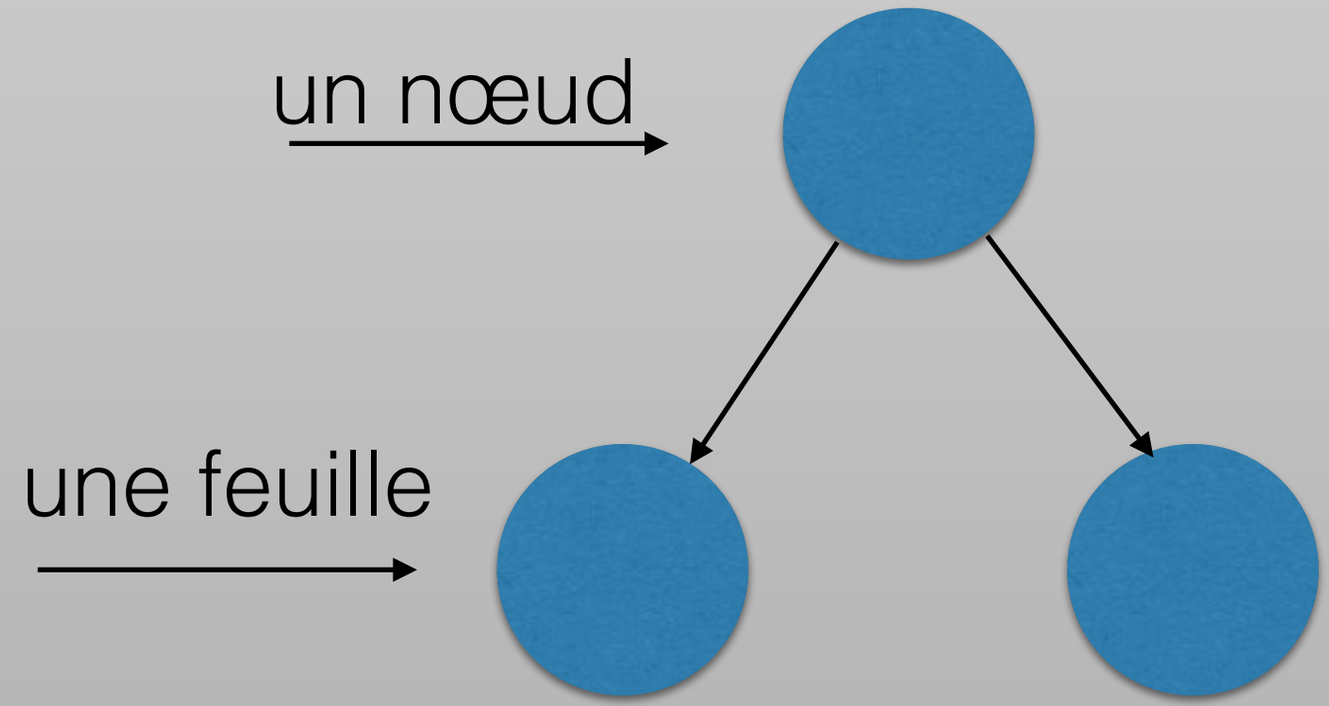
on va fabriquer un arbre

c'est une structure dans laquelle on trouve des nœuds

un **nœud** permet de désigner d'autres nœuds

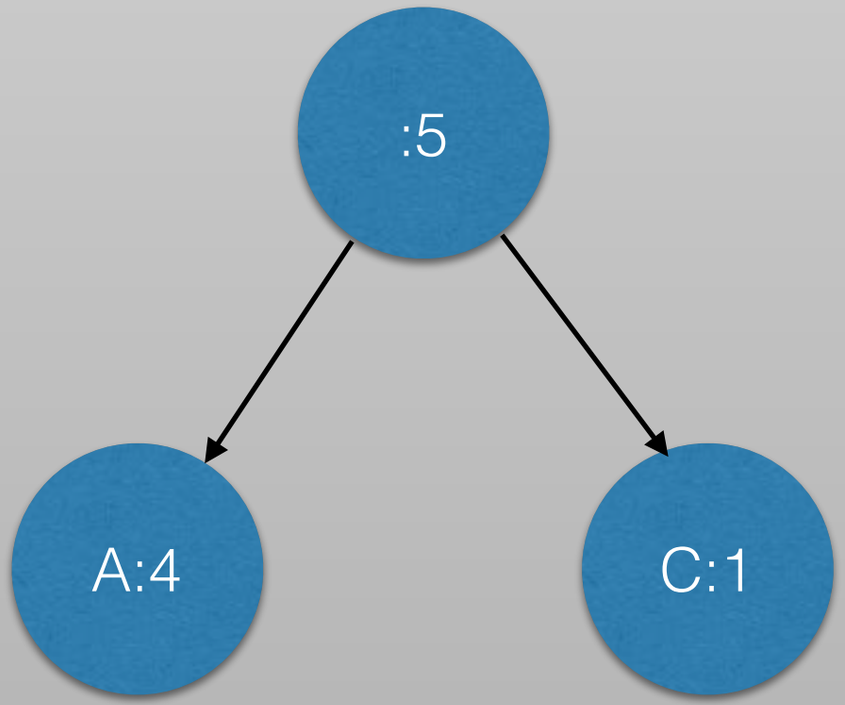
un nœud qui ne désigne rien est une **feuille**

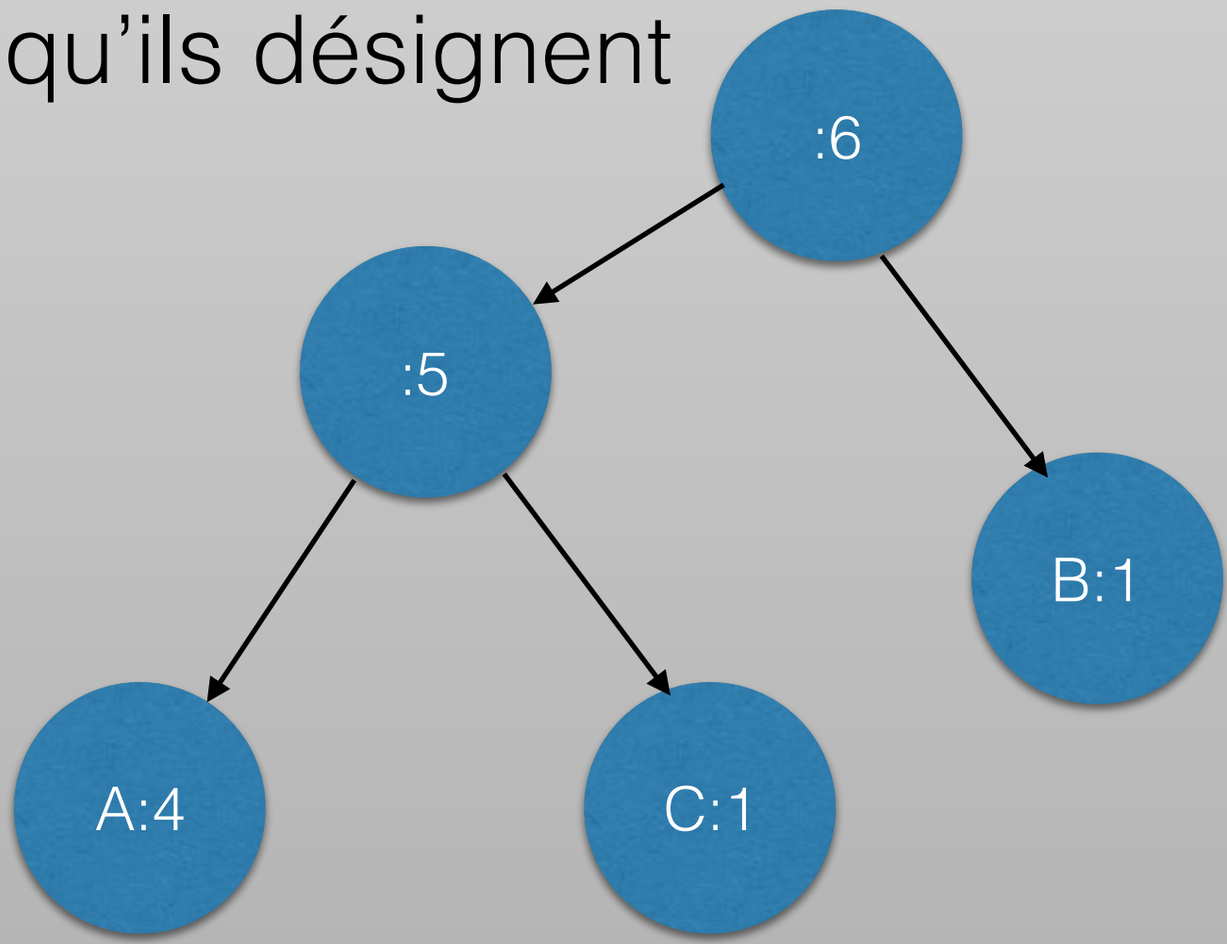
un nœud sans ascendant est appelé **racine**



on va fabriquer un **arbre binaire** (avec uniquement des nœuds à deux descendants) dans lequel

- les feuilles représenteront les lettres et leur fréquence associée (ou leur nombre d'occurence)
- les nœuds représenteront la somme des fréquences des lettres qu'ils désignent



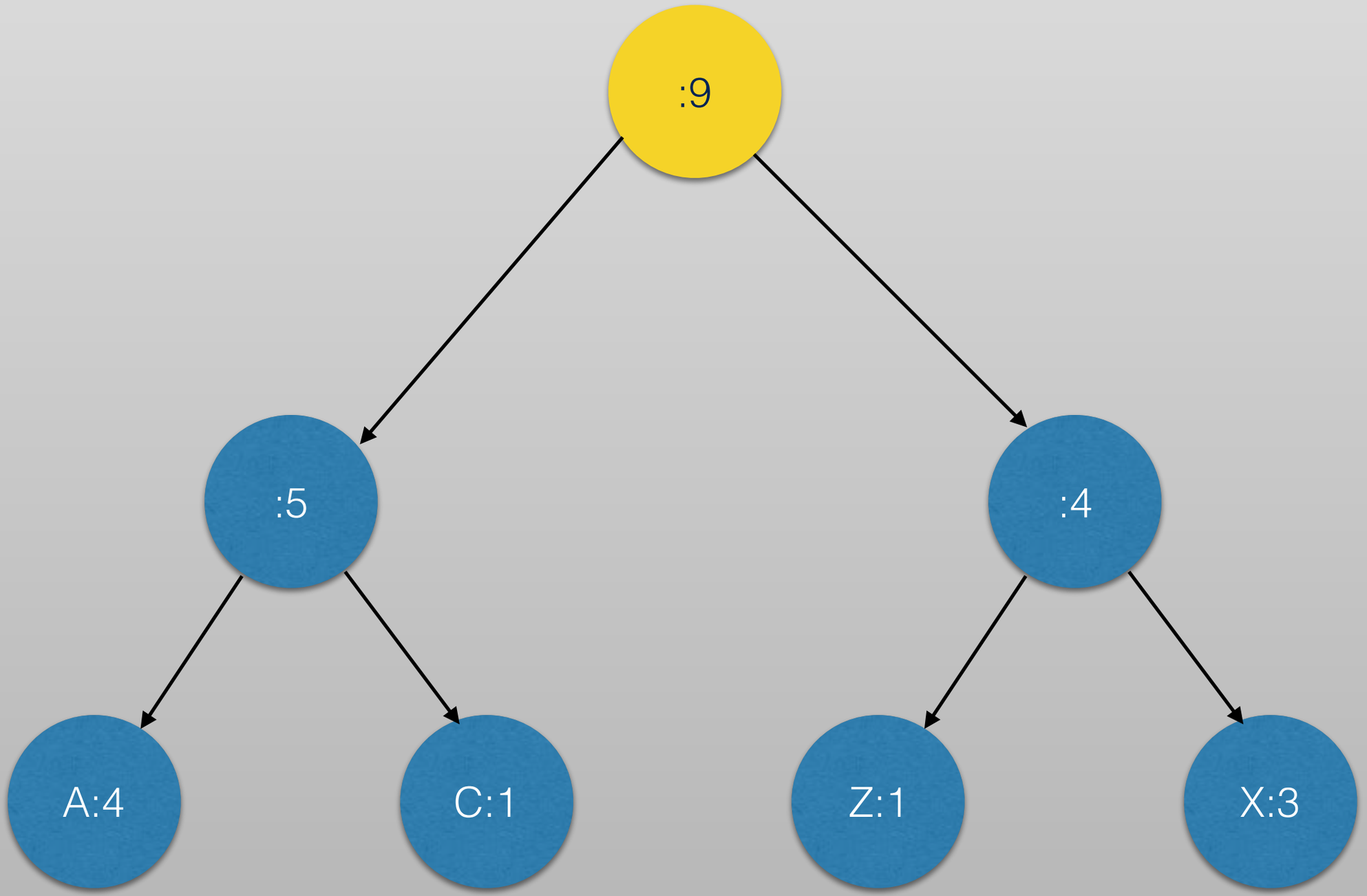


Attention : algorithme! (enfin presque)

Pour fabriquer cet arbre on part de l’ensemble des arbres réduits aux simples lettres pondérées par leur fréquence d’apparition dans le texte.

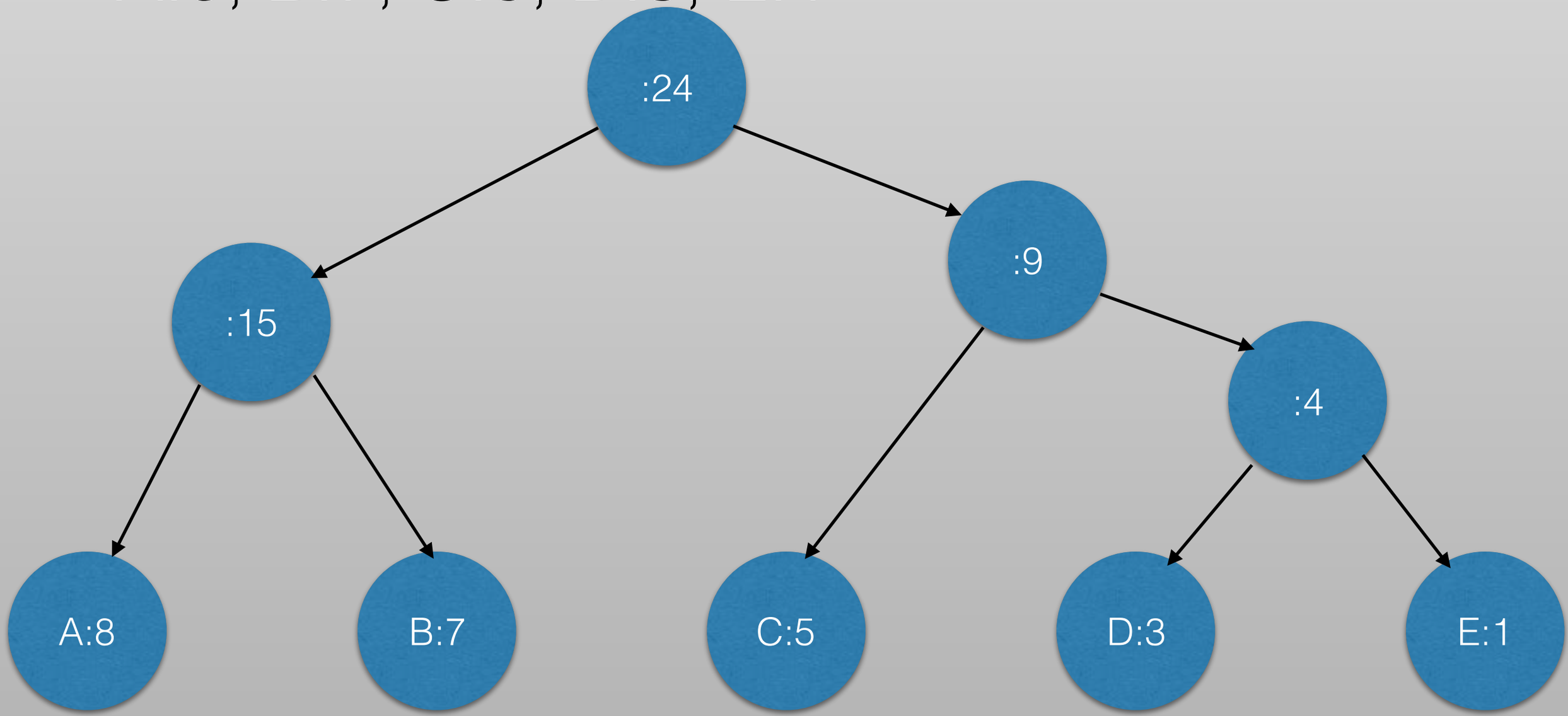
À chaque étape, on sélectionne deux arbres dont les fréquences des racines sont les plus petites et on fabrique un arbre dont le nœud racine pointera vers les deux arbres sélectionnés et dont la fréquence sera simplement la somme des fréquences/occurences

Exemple de la fusion de deux arbres…

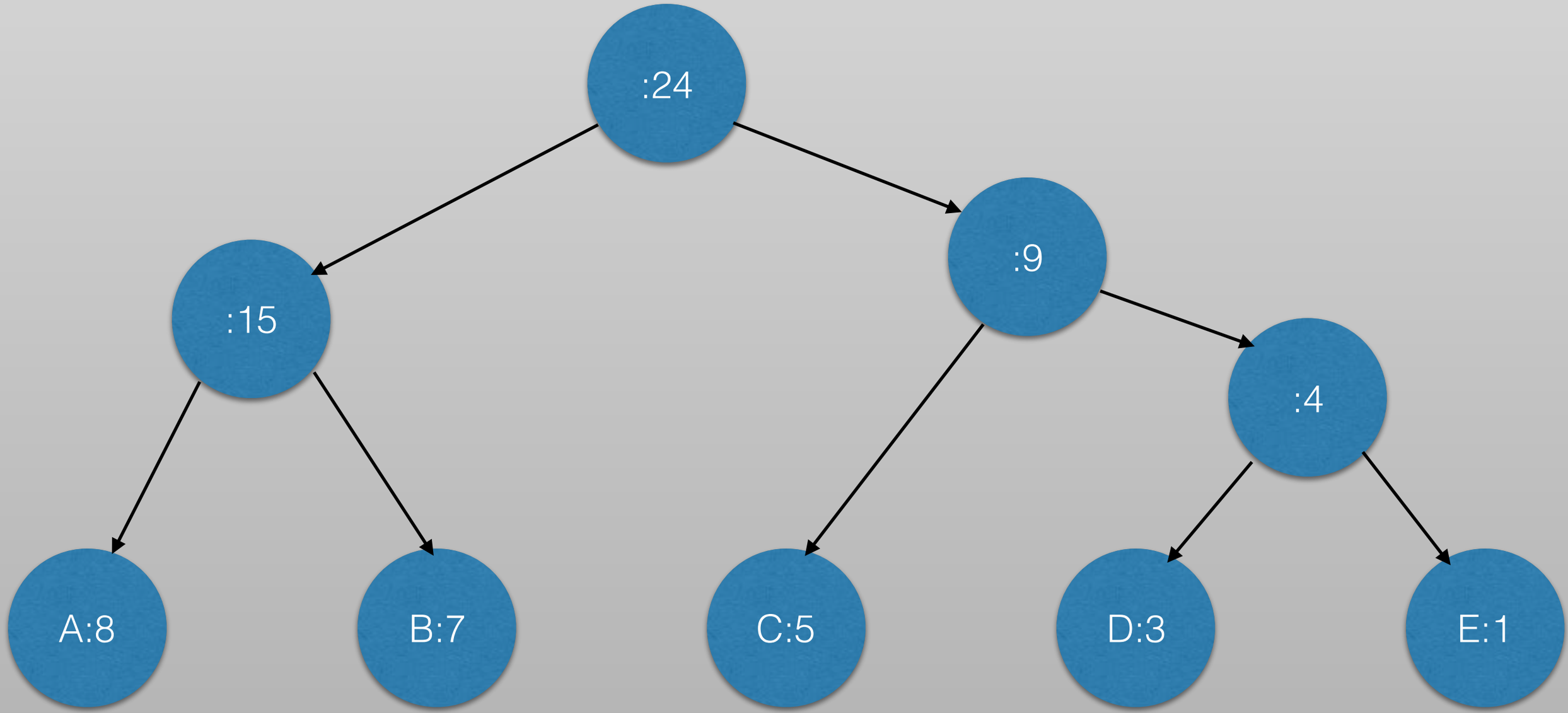


Prenons un exemple avec les lettres et leur nombre d'occurences suivantes :

A:8, B:7, C:5, D:3, E:1



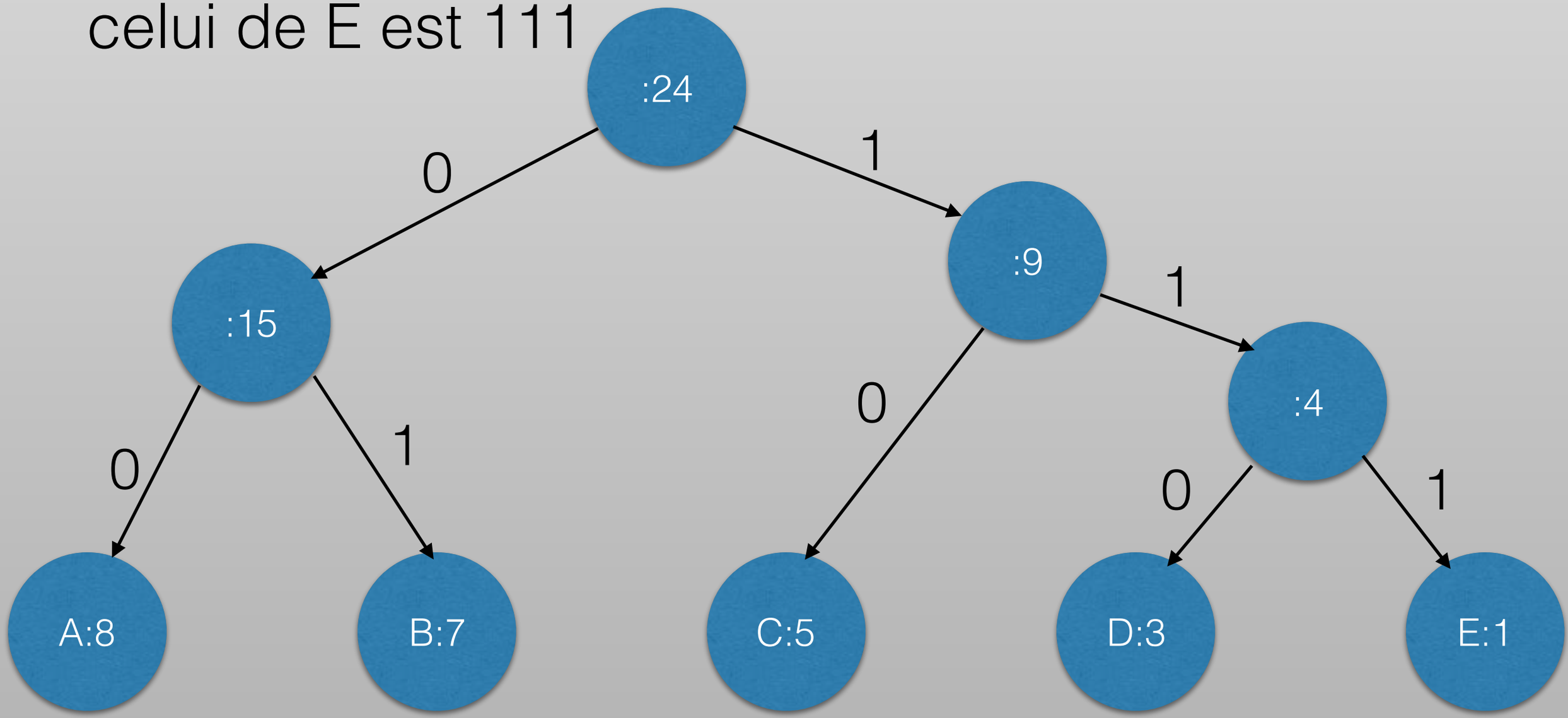
Ce que nous avons obtenu est un **arbre de Huffman**, on va l'utiliser pour coder les lettres!



Obtenir un codage à partir d'un arbre de Huffman consiste à étiquetter chaque paire de branche descendante l'une par 0 et l'autre par 1.

En général on choisit 0 pour étiquetter la branche descendante menant vers la plus haute fréquence; mais c'est arbitraire, et 1 pour l'autre

Les flèches qui descendent à gauche coderont 0
Les flèches qui descendent à droite coderont 1
Le codage de A est donc 00, le codage de B est 01, le codage de C est 10, celui de D est 110 et celui de E est 111



Un codage ordinaire (de longueur fixe) aurait conduit à utiliser 3 bits par lettre.

Donc pour coder E**A**B**D**A**C**A**B**B**C**A**B**A**B**D nous aurions utilisé 45 bits.

Le codage de Huffman de A est donc 00, le codage de B est 01, le codage de C est 10, celui de D est 110 et celui de E est 111

Ici les lettres les plus fréquentes sont codées sur 2 bits et les plus rares sur 3

Essayons de coder E**A**B**D**A**C**A**B**B**C**A**B**A**B**D

111**00**01**110**00**10**00**01**01**10**00**01**00**01**110

soit 33 bits.

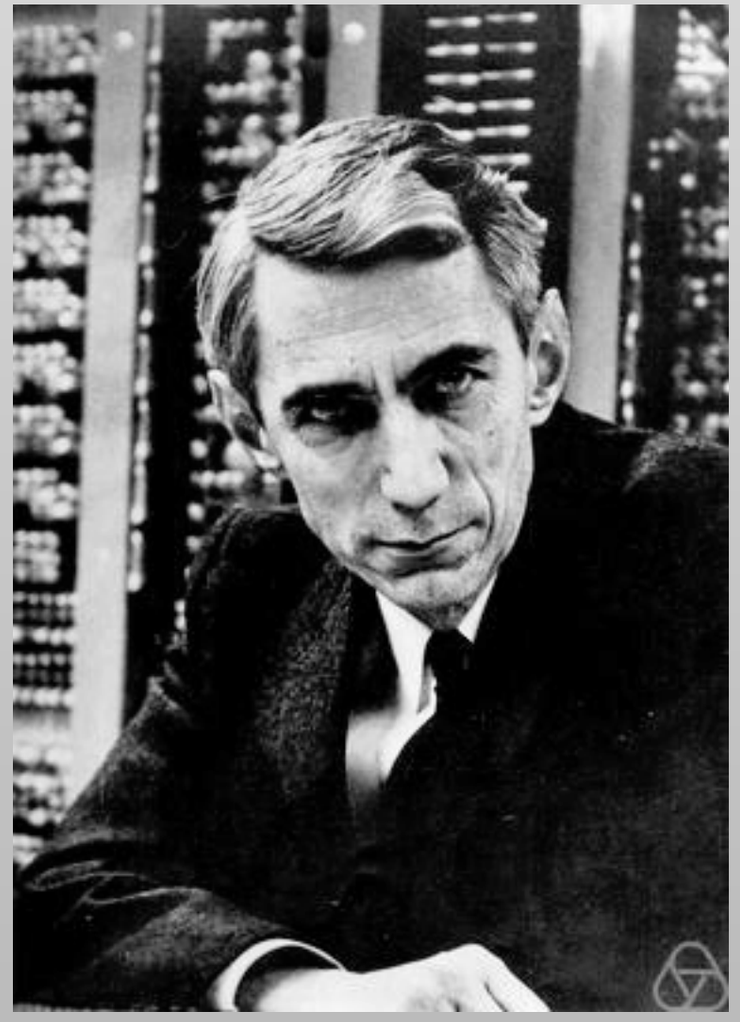
45 bits vs 33 bits! 33/45

Tx déf. 1 ~75%, Tx déf. 2 ~25%, Quot. 15:11

Ce codage est appelé **codage entropique**

L’**entropie de Shannon** correspond à la notion intuitive de quantité d’information contenue dans une source

Shannon est considéré comme le père de la **théorie de l’information**



Claude Shannon
1916 — 2001
Source Wikipédia

Fréquence des lettres en Français

(source Wikipédia)

	F		F
A	8,25 %	N	7,25 %
B	1,25 %	O	5,75 %
C	3,25 %	P	3,75 %
D	3,75 %	Q	1,25 %
E	17,75 %	R	7,25 %
F	1,25 %	S	8,25 %
G	1,25 %	T	7,25 %
H	1,25 %	U	6,25 %
I	7,25 %	V	1,75 %
J	0,75 %	W	0,00 %
K	0,00 %	X	0,00 %
L	5,75 %	Y	0,75 %
M	3,25 %	Z	0,00 %

Fréquence des lettres en Français

(source Wikipédia)

	F		F
E	17,75 %	M	3,25 %
A	8,25 %	V	1,75 %
S	8,25 %	B	1,25 %
I	7,25 %	F	1,25 %
N	7,25 %	G	1,25 %
R	7,25 %	H	1,25 %
T	7,25 %	Q	1,25 %
U	6,25 %	J	0,75 %
L	5,75 %	Y	0,75 %
O	5,75 %	K	0,00 %
D	3,75 %	W	0,00 %
P	3,75 %	X	0,00 %
C	3,25 %	Z	0,00 %

Il existe d'autre types de compressions sans perte (vous les étudierez plus tard) :

- par dictionnaire
- par contexte
- …

L'outil **zip** utilise par défaut l'algorithme **DEFLATE** qui est une variante du codage de Huffman

L'outil **bzip2** utilise l'algorithme de Burrows-Wheeler ainsi qu'un codage de Huffman

Les outils **compress** et **gzip** utilisent l'algorithme de Lempel-Ziv

L'outil **lzip** utilise l'algorithme Lempel-Zip-Markov chains

Compression d’image

La compression d’image recouvre de nombreuses techniques très différentes…

Les meilleures compressions sont avec perte

mais sur certaines images on peut vraiment gagner même sans perte…

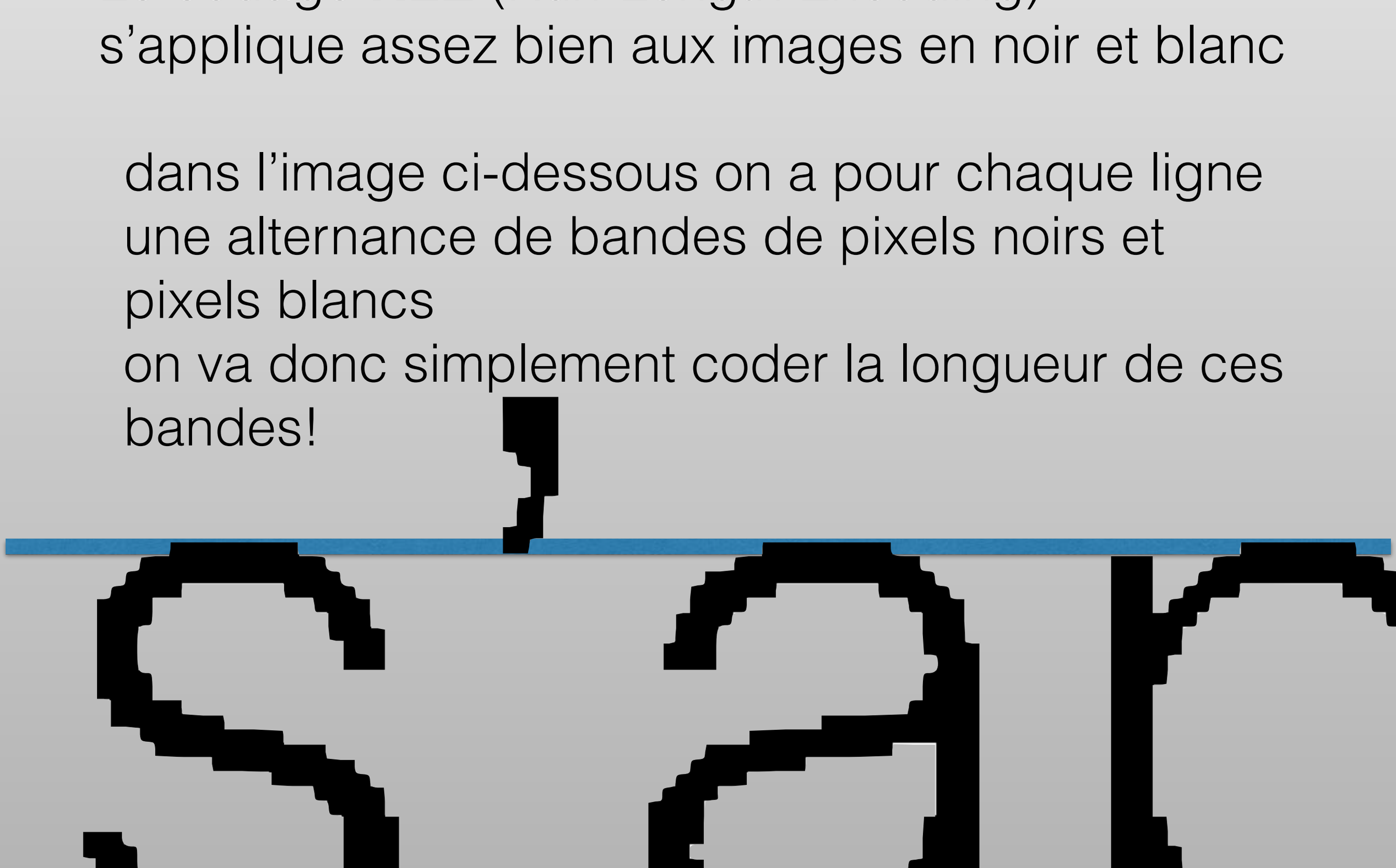
Prenons les images fabriquées avec un ordinateur (non numérisées)

beaucoup d'entre elles ont de larges bandes uniformes

on peut exploiter cette propriété

Le codage **RLE** (Run-Length Encoding) s'applique assez bien aux images en noir et blanc

dans l'image ci-dessous on a pour chaque ligne une alternance de bandes de pixels noirs et pixels blancs
on va donc simplement coder la longueur de ces bandes!



Au lieu de coder le mot de 40 bits directement
000000**111111**0000000000000**1111111**000000000

on va coder la suite des longueurs 6 **6** 12 **7** 9

pour cela il nous faut 4 bits par nombre et donc

0110 0110 1100 0111 1001

soit 20 bits seulement!

Tx déf 1 : 50%, Tx déf. 2 : 50%, Quot. 2:1

Ce codage est appelé **codage par répétition**

Cette technique est employée pour le fax

On y utilise d’abord le RLE afin d’obtenir une suite de nombres

Puis on utilise Huffman pour coder la suite de nombres!

Les autres types de compression d'images (avec ou sans perte) font appel à des techniques trop avancées pour être décrites ici…

ils reposent sur l'exploitation de particularités des images…

Attention : chaque algorithme de compression a son domaine d'emploi!

.gif, .png pour les images numériques

.jpeg pour les photographies

Cryptographie

écriture cachée selon les Grecs

des cas extrêmement simples

le chiffre de César, ROT13, Vigénère, le masque jetable de Vernam (brrr)

des cas plus compliqués

LFSR (linear feedback shift register - registre à décalage à rétroaction linéaire), DES, RSA

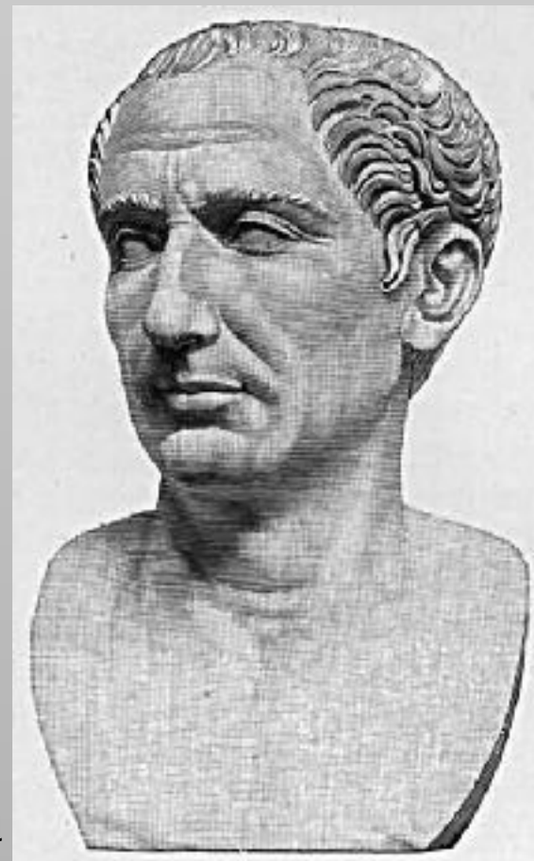
Idée : utiliser une fonction difficilement inversible pour coder un texte. Le secret est justement la fonction inverse…

Le chiffre de César

Attribué à l'empereur César (Caius Iulius Caesar IV -100 — -44)

il l'aurait utilisé pour masquer certaines correspondances (ce n'est pas le premier mécanisme de cryptographie)

c'est un chiffrement monoalphabétique par substitution



source Wikipédia

Il repose sur une permutation circulaire de l'alphabet

les lettres sont décalées de p (pour un p choisi) rangs dans l'ordre alphabétique

p=3, A→D, B→E, etc

On peut encore utiliser l'arithmétique modulaire (décidément) pour le définir…

Q: son inverse est aussi un chiffre de César. Lequel ?

Le codage ROT13 est le chiffre de César pour p=13

L’utilisation du chiffre de César ne peut pas bluffer quelqu’un très longtemps…

Force brute (seulement 25 chiffres possibles!)

vous seul avec un (petit) poil de courage

25 esclaves

un ordinateur

Analyse de fréquences pour casser le code…

Chiffre de Vigénère

Créé par Blaise de Vigénère (1523—1596)

Il a fallu attendre environ 300 ans avant de trouver la méthode permettant de casser ce code

Merci Friedrich Wilhem Kasiski

Blaise de Vigénère
1523 — 1596
source Wikipédia

Comment ça marche ?

On va utiliser différents chiffres de César pour les lettres du message

pour César la clé est p, le décalage

pour Vigénère on utilise n clés p_i, $0 \leq i < n$, en fait la clé est elle-même un texte (secret) en général plus court que le texte à encoder

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Clé secrète : `INFORMATIQUE`
Message : `JADORELINFORMATIQUE`

`INFORMATIQUEINFORMATIQUE`

`JADORELINFORMATIQUE`

Message codé :`RNICIQLBVVIVUNYWHGE`

Casser ce code n'est pas très difficile, si le message est assez long

On peut y repérer des répétitions et deviner la longueur de la clé, sinon on peut essayer diverses longueurs de clés

Puis faire des analyses fréquentielles

Un bon ordinateur (ou beaucoup de patience) et le tour est joué…

Q: Vous avez reçu le message
`RNNAVFRHXBYWUNY VJ`
êtes-vous d’accord ?

Chiffre de Vernam/Mauborgne ou masque jetable (one-time pad)

Un chiffre de Vigénère pour lequel :

la clé est aussi longue que le texte

la clé est obtenue par distribution aléatoire

la clé ne doit être employé qu'une seule fois

Si les conditions sont réunies, le chiffre est **inviolable**

il a probablement été utilisé pour sécuriser le « téléphone rouge »

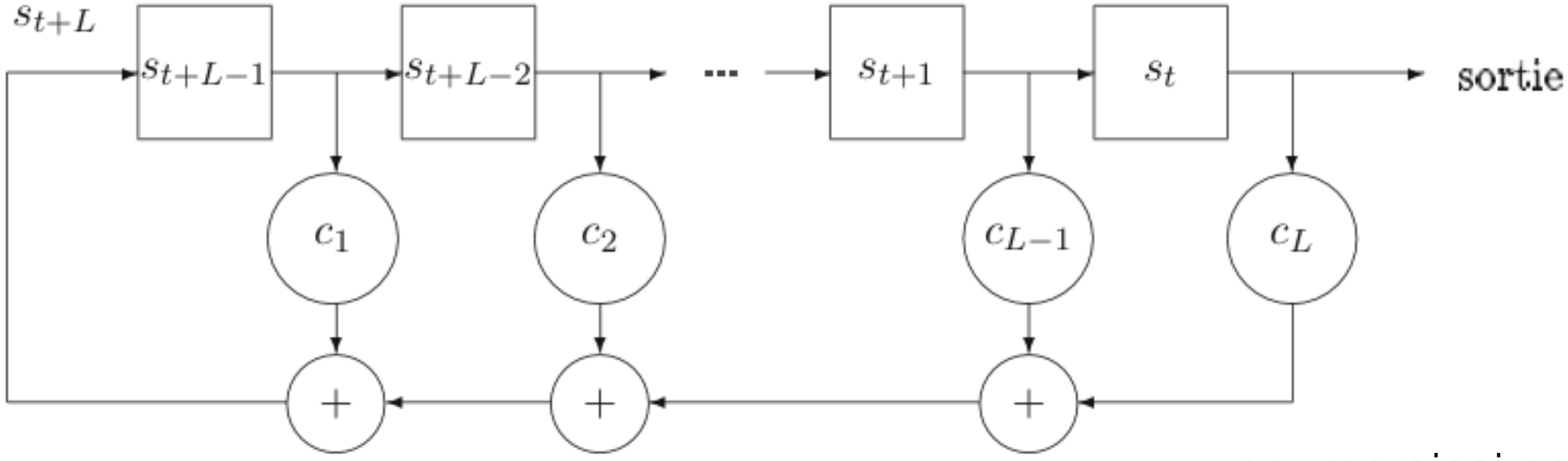
Aujourd’hui on utilise des systèmes plus solides…

- symétriques ou à clé secrète
- par blocs (par exemple le DES)
- ou par flot

les plus standards ne sont pas considérés comme très solides

- le DES peut être cassé en quelques jours/heures avec quelques dizaines d’ordinateurs
- mais c’est utile tout de même si vos messages n’ont pas un caractère vital ou si la durée de vie du contenu du message n’est pas très longue

LFSR

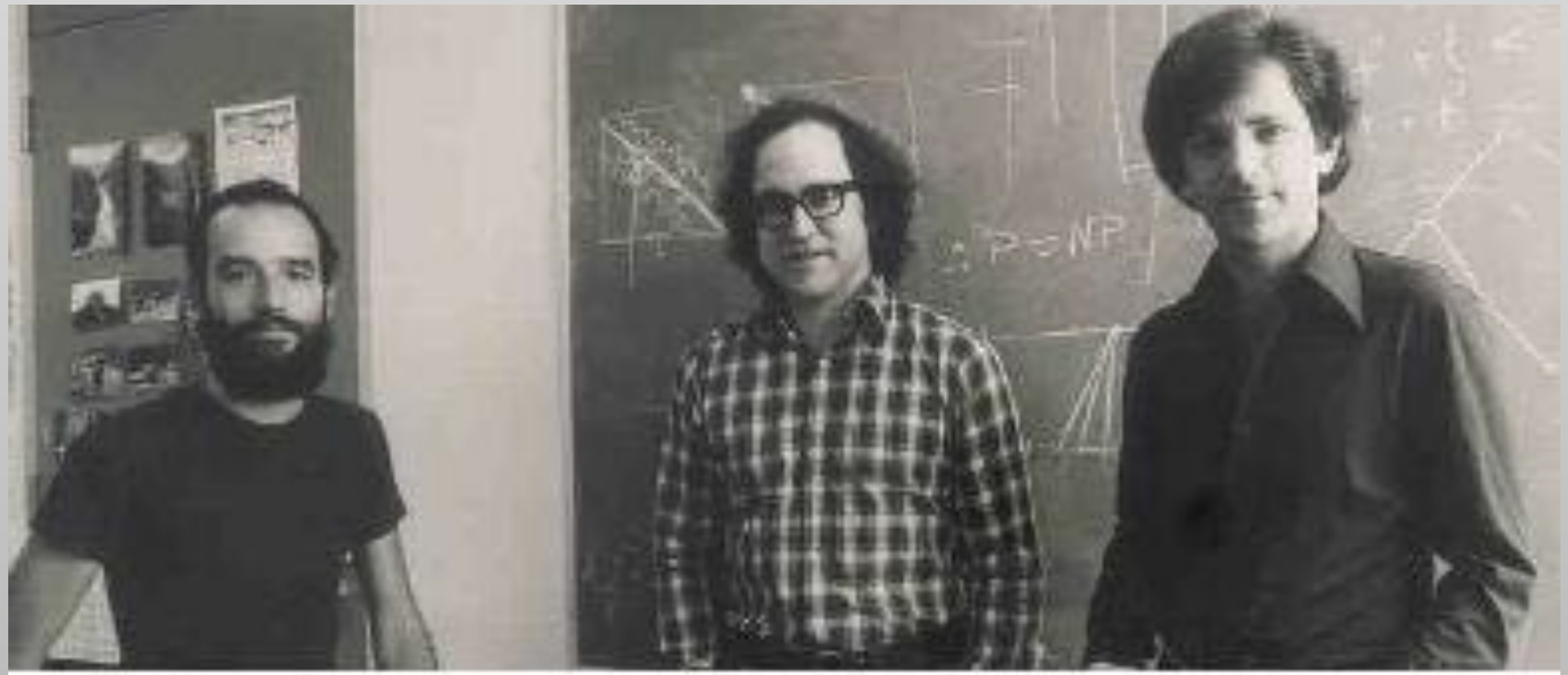


source picsi.org

permet d’obtenir une suite pseudo-aléatoire…

utile, par exemple, pour obtenir un masque jetable

RSA (1977)



Adi Shamir Ron Rivest Len Adleman

Aujourd'hui on utilise des systèmes plus solides que les chiffrement par flot pour les échanges très secrets…

asymétriques ou à clés publiques

le RSA (Rivest Shamir Adleman)

On utilise ces systèmes pour transmettre des clés secrètes de chiffrement par flots.

Comment ça marche ?

Encore des histoires de nombres et calculs modulaires

1. on prend 2 nombres p,q premiers
2. n = p.q, ϕ(n) = (p-1)(q-1)
3. on choisit e premier avec ϕ(n) et plus petit que ϕ(n)
4. on calcule d l'inverse de e modulo ϕ(n), c'est-à-dire tel que e.d ≡1 (mod ϕ(n))
5. (n,e) est la clé publique que tout le monde peut connaître et utiliser pour coder, (n,d) est la clé privée et qui sert à décoder

Pour chiffrer le message M :

on calcule $C \equiv M^e \pmod{n}$

Pour déchiffrer C :

on calcule $M \equiv C^d \pmod{n}$

La difficulté repose sur le fait que connaissant n et e il est très difficile de trouver d car il faut pour cela connaître p et q c'est-à-dire décomposer le nombre n en facteurs premiers…

Attention p,q,d,e sont normalement de très très très grand nombres!

Attention trouver de très grands nombres premiers est une tâche très ardue! C'est même considéré comme un véritable tour de force

En 1456 le plus grand nombre premier connu était 8191, soit de 4 chiffres…

En 1750 Euhler a produit le premier nombre premier de 10 chiffres!

En 1996 le plus grand nombre premier connu s'écrivait avec ~420.000 chiffres

En 2016 le plus grand s'écrit avec ~22.000.000 (dû à Cooper, Woltman, Kurowski et Blosser)

Consulter le **Great Internet Mersenne Prime Search** (GIMPS)

Exemple! Allons-y

on prend p=7 et q=11

n = 7*11 = 77

ϕ(n) = 6*10 = 60

prenons e=7 qui est premier avec 60

on calcule d tel que 7*d ≡ 1 (mod 60), d=43 fonctionne car 43*7=301=5*60+1

clé publique (7,77), clé privée (43,77)

prenons BONJOUR, soit en ascii 42 4F 4E 4A 4F 55 52, on prend la valeur décimale 18664546135659858, puis on découpe en une suite de nombres plus petits que n=77 donc 18 66 45 46 13 56 59 8 58

18 66 45 46 13 56 59 8 58

on chiffre

18^7 mod 77 = 39
66^7 mod 77 = 66
45^7 mod 77 = 45
46^7 mod 77 = 18
13^7 mod 77 = 62
56^7 mod 77 = 56
59^7 mod 77 = 38
8^7 mod 77 = 57
58^7 mod 77 = 9

Soit le message codé 39 66 45 18 62 56 38 57 9

39 66 45 18 62 56 38 57 9

on décode

39^43 mod 77 = 18

66^43 mod 77 = 66

45^43 mod 77 = 45

18^43 mod 77 = 46

62^43 mod 77 = 13

56^43 mod 77 = 56

38^43 mod 77 = 59

57^43 mod 77 = 8

9^43 mod 77 = 58

Soit 18 66 45 46 13 56 59 8 58 donc 186645461356598 58

et en hexa 424F4E4A4F5552

BONJOUR

Vous pouvez toujours vous demander comment calculer une puissance modulo avec des grands nombres ?

par exemple 3^{259} (mod 127) ?

Il n'est pas nécessaire de calculer 3^{259}

on sait que :

3^1 = 3 (mod 127)

3^2 = 9 (mod 127)

3^4 = 81 (mod 127)

3^8 = 84 (mod 127)

3^16 = 71 (mod 127)

3^32 = 88 (mod 127)

3^64 = 124 (mod 127)

3^128 = 9 (mod 127)

3^256 = 81 (mod 127)

or $3^{259} = 3^{256}.3^2.3$ donc $3^{259} \equiv 81.9.3 \equiv 28$ (mod 127)

Contrôle d’erreur

CRC

Hamming

Le principe est de fournir de la **redondance**

doubler/tripler le message

un autre exemple : l'alphabet radio international (encore très très employé!!! Y compris dans l'aviation civile!)

PF1 en radio international **Papa, Foxtrot, One**

on **rajoute de l'information** permettant d'assurer la bonne lisibilité du message en cas de bruit

Le CRC était employé dans la transmission du code ASCII, en ajoutant un 8-ième bit de contrôle

de sorte que la somme des huit bits soit toujours paire

permet de détecter si **une** erreur s'est produite, mais pas où

ne permet pas de détecter deux erreurs…

CRC du mot 0110001 = 1

CRC du mot 0110110 = 0

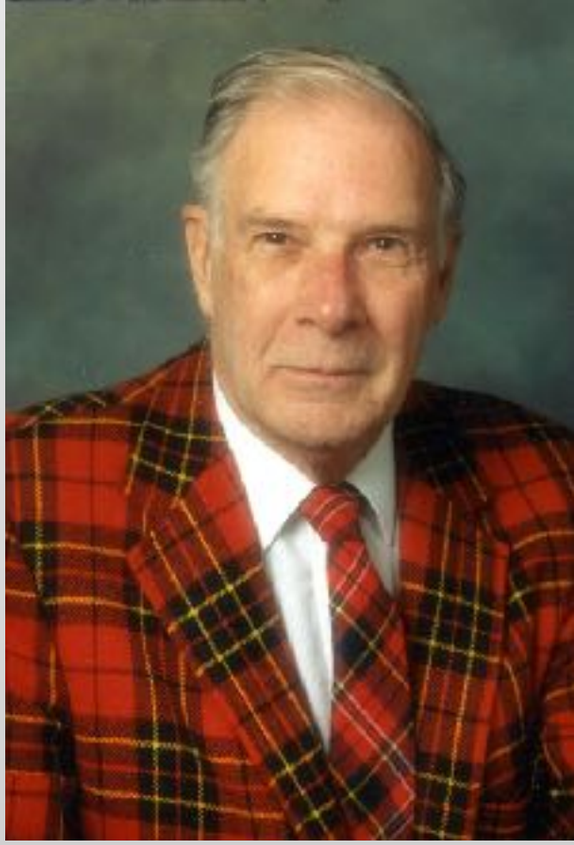
Donc si je reçois 8 bits dont le nombre de 1 est impair, je **sais** que la transmission est erronée. Dans les autres cas, je ne sais pas.

Le code de **(Richard) Hamming** (1915—1998)

une famille de codes

permet de

détecter et **corriger**

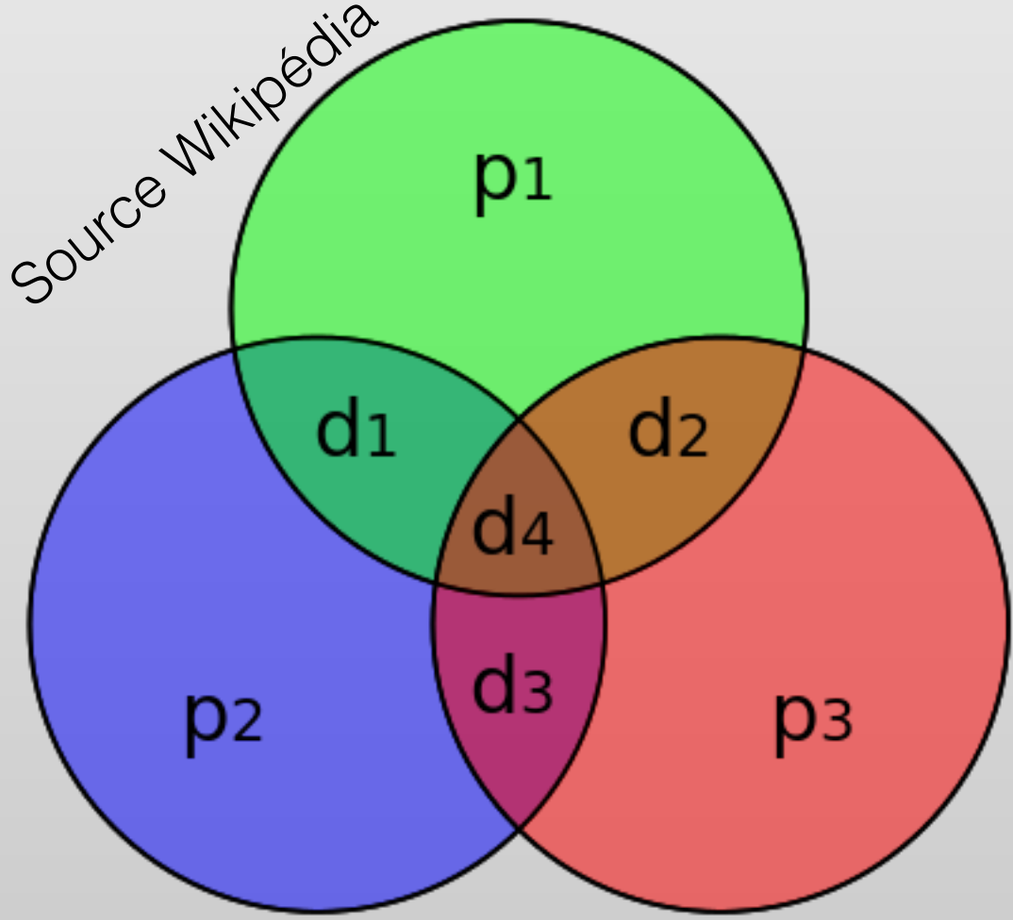


source acm.org

son principe est de calculer plusieurs parités sur différentes parties du mot de sorte qu'il soit possible de détecter si erreur il y a et où.

le code de Hamming le plus simple est le [7,4] (ou [7,4,3])

il code des mots de 4 bits sur 7 bits (c'est le prix à payer)



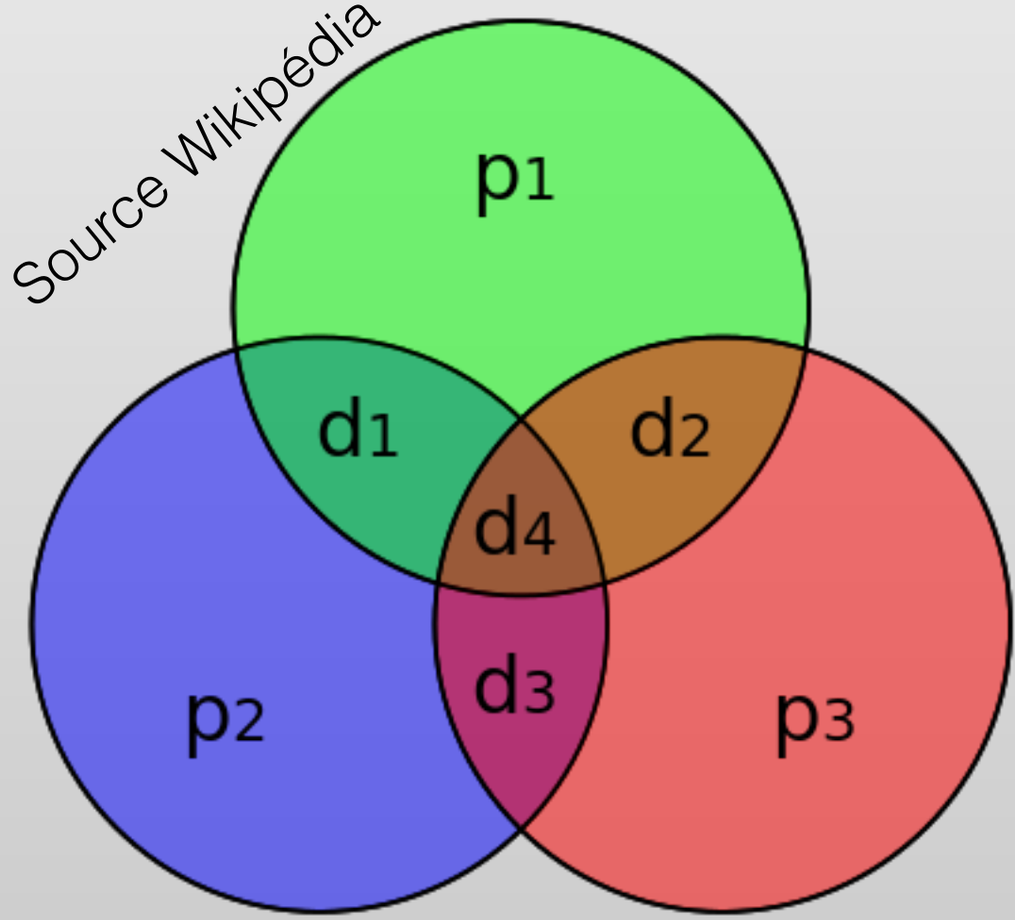
Le message est $d_1d_2d_3d_4$

On y rajoute 3 bits $p_1p_2p_3$ de sorte que

p_1 est la somme de contrôle $d_1+d_4+d_2$

p_2 la somme de contrôle $d_1+d_4+d_3$

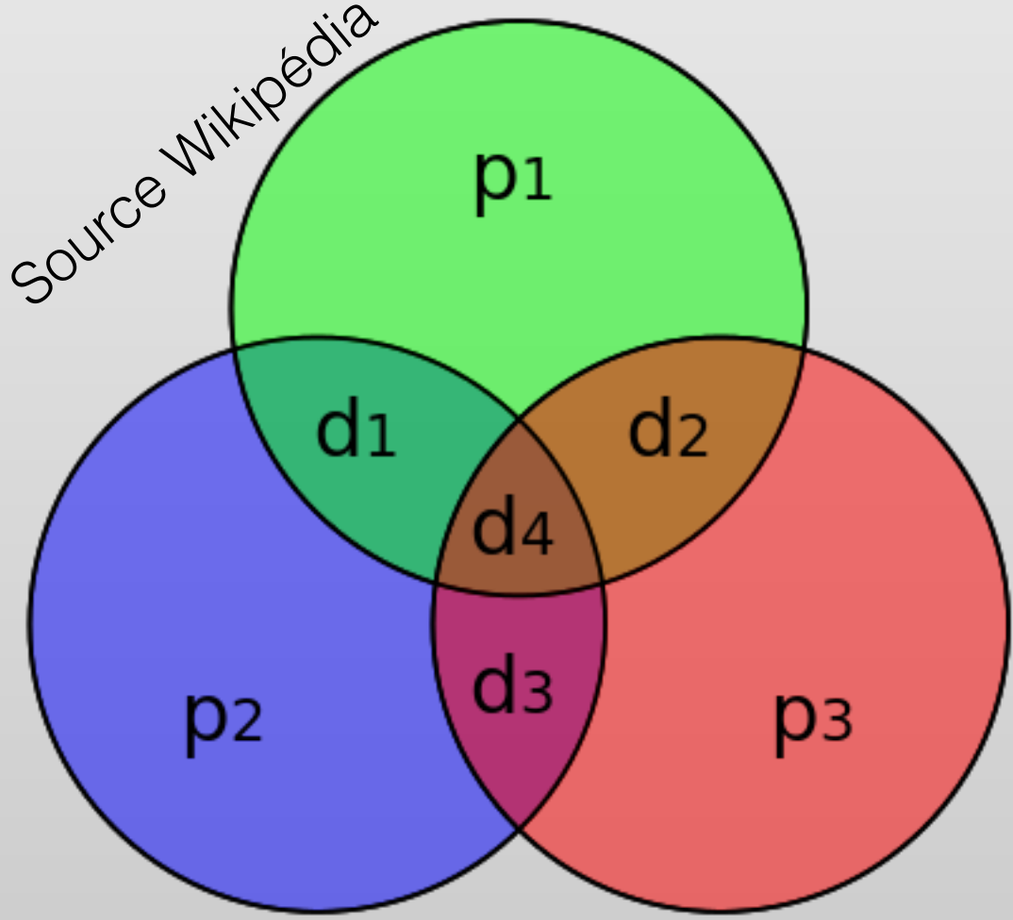
p_3 la somme de contrôle $d_2+d_4+d_3$



Le message est 0101

p_1=0, p_2=1, p_3=0

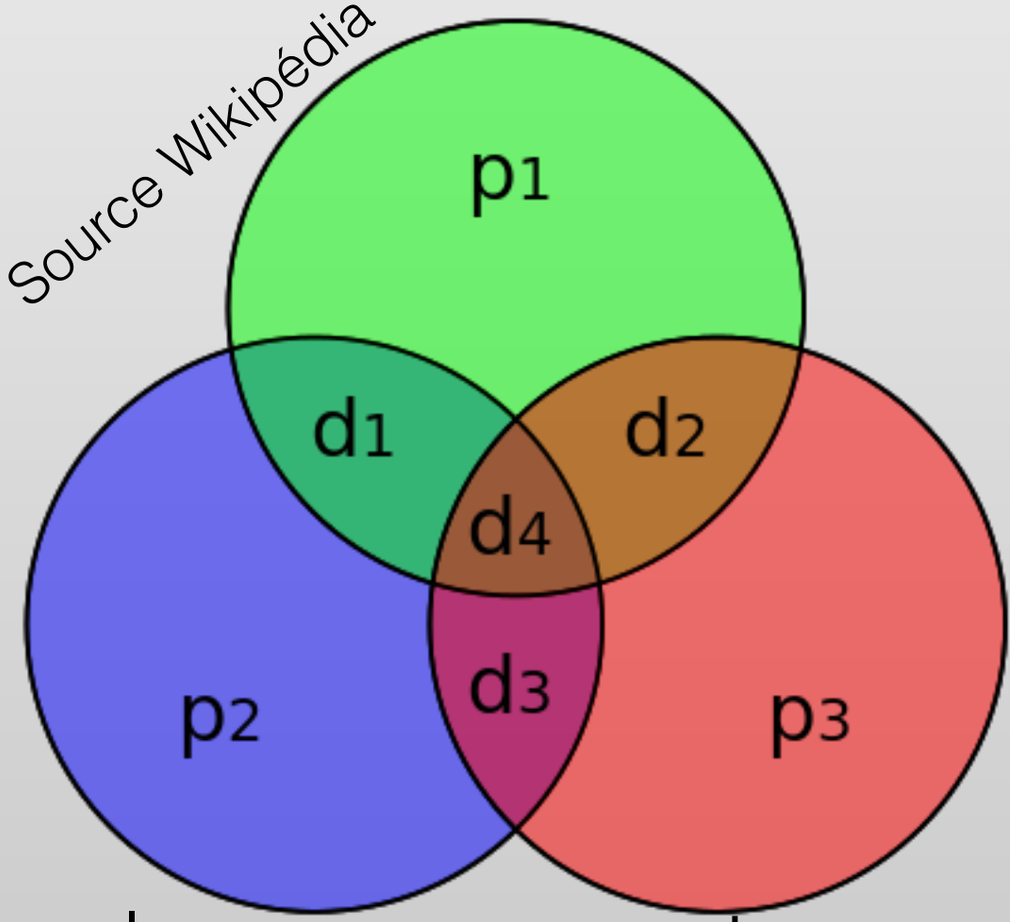
Le message codé est 010**0101**



Le message est 0101
p_1=0, p_2=1, p_3=0
Le message codé est 010**0101**

Si deux contrôles sont corrects c'est que le message est correct et le troisième contrôle faux

Si je reçois **1**100101, je recalcule
p'_1=0, p'_2=1, p'_3=0, je sais que p_1 (reçu) est faux, le message est correct **0101**



Le message est 0101
p_1=0, p_2=1, p_3=0
Le message codé est 010**0101**

Si un contrôle est correct, c'est que le message est faux et l'erreur est sur le bit qui n'est pas couvert pas le contrôle correct

Si je reçois 010**1**101, je recalcule
p'_1=1, p'_2=0, p'_3=0, je sais que d_1 (reçu) est faux, je corrige le message : **0101**.

Des codes plus compliqués existent

Par exemple le **BCH**

Bose, Ray-**C**haudhuri et **H**ocquenghem

utilisé pour les communications satellites, les SSD, et les codes-barres!