Université Blaise Pascal

UFR Sciences et Technologies

L1 : Découverte de l'informatique
1ère partie : de l'électronique à l'informatique

(3) Encodage de l'information

Codage ou encodage ?

Données sous une forme gérable par une machine → codage (représentation)
← ch (2)

Données conservées et transmises sans problème → encodage

1. Détecter, corriger les erreurs (intégrité des données)
2. Réduire la taille en machine (compression)
3. Sécuriser (chiffrement)

1. Codes détecteurs / correcteurs d'erreurs

m bits données + k bits de contrôle = n bit de code

1.1. Code "auto-vérificateur" : détection des erreurs

1.2. Code "auto-correcteur" : détection ET correction des erreurs

1.1 Code "auto-vérificateur"

1.1.1 Détection d'erreurs isolées

§ **Contrôle de parité** : exemple du code ASCII (n=7+1)

à coder	Code décimal	Code binaire (7 b)	Combien de 1	bit de parité	code sur 8b
...					
A	65	100 0001	deux	1	1100 0001
B	66	100 0010	deux	1	1100 0010
C	67	100 0011	trois	0	0100 0011
D	68	100 0100	deux	1	1100 0100
...					
s	115	111 0011	cinq	0	0111 0011
...					

Parité impaire (somme sans retenue / XOR**) ⇔ nombre impair de 1 en tout (contrôle inclus)**

→ détecte un nombre impair d'erreurs

✍ Parité paire : quelle détection ?

1.1.2 Détection d'erreurs groupées

§ Sommes de contrôle (exemple modulo 10)

► **Encodage :**
- Message M(x) en décimal
  Exp M=1880163251006
- Calcul de la somme de contrôle (checksum) : R=
  Exp R=1+8+8+0+1+6+3+2+5+1+0+0+6 mod 10 = 10 – (41 mod 10) = 9
- N= M.9 = 18801632510069

► **Vérification :**
- A la réception, pas d'erreur si N mod 10 = 0 ;

✍ Codes ISBN, EAN (Gencode), NIRPP : quelles sommes de contrôle ?

§ CRC (*Cyclic Redundancy Code, polynomial code checksum*)

Polynôme P(x) associé à un entier binaire b à m chiffres : $P(x)=\Sigma^{n}_{i=1}\ b_i x^i$ (théorie des corps)
exemple : b=l00ll0 => $P(x)=x^5+x^2+x$

► **Encodage :**
- Expéditeur et destinataire ont choisi un polynôme G(x) de degré g ;
- L'expéditeur prépare le message M(x) :
  - $L(x)=M(x)*x^g$ *// ajouter d zéros à droite du message*
  - $R(x)=N(x) \bmod_2 G(x)$ est* la "somme de contrôle" (checksum)
    * $\bmod_2$: *reste de la division modulo 2*
  - N(x)=L(x) - R(x) [mod 2] *// ajouter à la suite de M(x) les g bits de R(x)*
- L'expéditeur envoie T(x) ;

choix de G !

► **Vérification :**
- A la réception, pas d'erreur si $N(x) \bmod_2 G(x) = 0$;
- Le message initial est obtenu en supprimant les g derniers bits de N(x).

★ **Exemple 1 :**

- **Encodage :** — **en binaire :**

G(x)=x^3+x+1, de degré 3 — 1011

M(x)=$x^5+x^3+x^2+1$ — 101101

L(x)=M(x)*G(x) — 101101000

R(x)=$L(x) \bmod_2 G(x)$ — 011

$x^8+x^6+x^5+x^3$ | x^3+x+1
$x^8+x^6+x^5$ | x^5+1
$0+x^3$
$-x-1$

1 0 1 1 0 1 0 0 0 | 1 0 1 1
1 0 1 1 | 1 0 0 0 0 1
0 0 0 0 0 1 0 0 0
1 0 1 1
0 0 1 1

N(x)=L(x) – R(x) [mod 2] — 101101011

- **Vérification :**

1 0 1 1 0 1 0 1 1 | 1 0 1 1
1 0 1 1 | 1 0 0 0 0 1
0 0 0 0 0 1 0 1 1
1 0 1 1
0 0 0 0

★ **Exemple 2 :**

G(x)=1011 de degré 3
Message reçu : N(x)=11010101

Vérification :

1 1 0 1 0 1 0 1 | 1 0 1 1
1 0 1 1 | 1 0 0 0 0 1
0 1 1 0 0
1 0 1 1
0 1 1 1 1
1 0 1 1
0 1 0 0 0
1 0 1 1
0 0 1 1 1

Attention : division modulo 2 <=> XOR !

R(x)=111 : erreurs! => retransmettre le message

► **Choix du polynôme générateur :**

CRC-12 : 1 1000 0000 1111

CRC-16 : 1 0001 0000 0010 0001

CRC-CCITT : 1 0001 0000 0010 0001

- ***Recommandé pour les caractères codés sur 8b***
- ***Détecte toutes les erreurs groupées en bloc <=16b (et plupart au delà)***

1.2 Codes "auto-correcteurs"

§ Code de répétition (bégayer pour vérifier)

Bonjour! → BBBooonnnjjjooouuurrr!!!

N.B. la répétition se fait en réalité bit par bit

Vérification : vote majoritaire

BCBooonnniijooouuurrr!!! → Boniour!

§ Double parité

Données présentées en tableau → contrôle de parité en ligne et en colonne

Exemple d'un texte en ASCII : *Bonjour!*

Données envoyées

représentation binaire

Transversalement parité paire								
0	1	0	0	0	0	1	0	**B**
1	1	1	0	1	1	1	1	**o**
0	1	1	0	1	1	1	0	**n**
0	1	1	0	1	0	1	0	**j**
1	1	1	0	1	1	1	1	**o**
1	1	1	1	0	1	0	1	**u**
0	1	1	1	0	0	1	0	**r**
0	0	1	0	0	0	0	1	**!**
1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	

Longitudinalement parité paire

Données reçues

représentation binaire avec contrôle transversal pair

0	1	0	0	0	0	1	0	B
1	1	1	0	1	1	1	1	o
0	1	1	0	1	1	1	0	n
0	1	1	0	0	0	1	0	b
1	1	1	0	1	1	1	1	o
1	1	1	1	0	1	0	1	u
0	1	1	1	0	0	1	0	r
0	0	1	0	0	0	0	1	!
1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	

contrôle longitudinal pair

→ Erreur au 4ème bit du 4ème caractère

§ Code de Hamming (linéaire : calcul matriciel, parfait : minimise k/m)

Famille de codes basés sur les tests de parité

- Pour contrôler m bits, on utilise $k=\lceil \log_2(m) \rceil$ bits (k : nombre de chiffres nécessaires pour écrire m en binaire)
- Le bit de contrôle K_i est le $(2^{i-1})^{ème}$ bit des n bits du code (bits #1 à droite, bit #n à gauche)
- K_i contrôle (parité) chaque bit de N dont l'adresse α en binaire a son $i^{ème}$ bit à 1

►Exemple : code 4+3 (**3 bits de contrôle K_i pour 4 bits d'information M_i)**

N=	M4	M3	M2	K3	M1	K2	K1
	7	6	5	4	3	2	1 α
	III	II0	I0I	I00	0II	0I0	00I

K1 contrôle M1 (α_1=0II), M2 (α_2=I0I), M4 (α_4=III)
K2 contrôle M1 (α_1=0II), M3 (α_3=II0), M4 (α_4=III)
K3 contrôle M2 (α_2=I0I), M3 (α_3=II0), M4 (α_4=III)

M1 (α_1=0 I I) est contrôlé par K1 et K2
M2 (α_2= I 0 I) par K1 et K3
M3 (α_3=I I 0) par K2 et K3
M4 (α_4=I I I) par K1, K2 et K3

► Exemple : code 4+3 pour M=1010 avec parité paire

1	0	1	K3	0	K2	K1
7	6	5	4	3	2	1
III	II0	I0I	I00	0II	0I0	00I

★ **Encodage** :

M=1010 K1=0 car M1=0, M2=1, et M4=1
K2=1 car M1=0, M3=0, et M4=1
K3=0 car M2=1, M3=0, et M4=1

message envoyé : n=1010010

★ **Vérification d'un message reçu** : (parité paire)

N=I0II0I0 M1=0, M2=1 et M4=1 => OK car K1=0 => V_1=0
M1=0, M3=0 et M4=1 => OK car K2=1 => V_2=0
M2=1, M3=0 et M4=1 => Problème car K3=1 (V_3=1)

Problème sur K3 mais pas K1 ni K2 => L'erreur vient de K3 lui-même
N.B. l'erreur (unique!) est à l'adresse $V_3V_2V_1$ dans n

Message corrigé : N=1010010 donc M=1010

► Encodage et vérification rapides

★ **Exemple d'encodage** :

M=10101011001 avec parité paire

11 bits d'information => 4 bits de contrôle (11_{10}=I0II$_2$)

N=1010101?100?1??

i tq Ni=1	i en binaire				*une ligne par bit à 1 dans l'information M !*
15	1	1	1	1	
13	1	1	0	1	
11	1	0	1	1	
9	1	0	0	1	
7	0	1	1	1	
3	0	0	1	1	
somme	0	1	0	0	=> valeurs à attribuer aux Ki en parité paire
contrôle	K4	K3	K2	K1	
	1	*0*	*1*	*1*	*N.B. inverser pour une parité impaire*

=> N=10101010100 1100

★ **Exemple de vérification** :

Message reçu : N=1010001110001 11 avec parité paire (message 15=11+4 bits)

i tq Ni=1	i en binaire				*y compris les Ki !*
15	1	1	1	1	
13	1	1	0	1	
9	1	0	0	1	
8	1	0	0	0	
7	0	1	1	1	
3	0	0	1	1	
2	0	0	1	0	
1	0	0	0	1	
somme ⊕	0	1	0	0	
contrôle	V_4	V_3	V_2	V_1	

V=0100 => erreur en 0100_2 soit au 4ème bit (k3)

N.B. inverser pour une parité impaire : $\overline{V_4V_3V_2V_1}$=1011

=> message corrigé N=1010001110011 11 *(on suppose erreur unique)*
donc le message envoyé était : M=10100011001

Vérification du message corrigé N=1010001110011 111

i tq Ni=1	i en binaire			
15	1	1	1	1
13	1	1	0	1
9	1	0	0	1
8	1	0	0	0
7	0	1	1	1
4	0	1	0	0
3	0	0	1	1
2	0	0	1	0
1	0	0	0	1
somme	0	0	0	0
contrôle	V_4	V_3	V_2	V_1

V=0000 => OK

2. Compression

Diminuer le volume de données :

- Comprimer plus ou moins
  taux de compression (exp ZIP taux ~2:1, MP3 taux 10:1)
  ✍ **UTF-8 peut-il être considéré comme un algorithme de compression ?**

- Qualité de compression
  sans perte d'information (exp RLE, Huffman, LZW) ou avec (exp JPEG)

- Temps de compression
  (exp MP3 : lent)

→ **COMPROMIS**

2.1 Compression sans perte

§ **RLE** (*Run-Length Encoding,* Compression par plages)

► **Principe : regrouper les unités d'information ("*caractères*") successives identiques**

→ **Intéressant pour documents scannés N/B, les Fax**
→ **Utilisé dans le format BMP (taux 2:1)**

★ **Exemple 1 :** ***sans méta-caractère***

NNNNNBBNNNNNNNNNBBBBBNBBB → 5N2B9N5B1N3B

NBNBN → 1N1B1N1B1N

★ **Exemple 2 :** ***avec méta-caractère (un symbole éventuellement indisponible)***

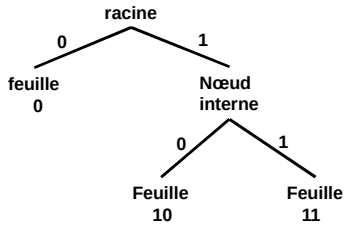
PAAAAAARFAITEMEEEEEEEEEEEEEENT → P\6ARFAITEM\15ENT

→ **Variantes (sans méta-car. & àpd k occurrences / méta-car doublé / …)**

§ Compression de Huffman

► **Principe : codes des caractères de taille variable (selon fréquence)**

1. **Déterminer les fréquences des caractères du texte** (triées par ordre décroissant)
2. **Construire une arborescence binaire**
   - **ses feuilles sont les caractères**
   - **ses arcs sont étiquetée par 0 (branches gauches) / 1 (branches droites)**
   - **ses sous-arborescences sont de poids (fréquence des caractères) équilibrés**
3. **Chemin racine-feuille → code du caractère considéré**

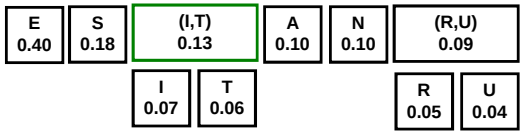


★ **Exemple :** ***méthodes ascendante des minimaux (regrouper les deux moins fréquents)***

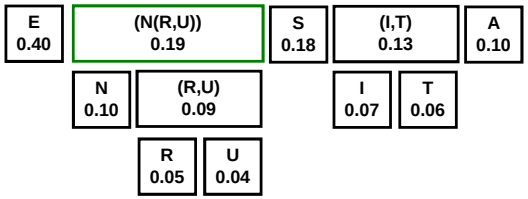
Caractère	E	S	A	N	I	T	R	U
fréquence	0.40	0.18	0.10	0.10	0.07	0.06	0.05	0.04

1°) R et U : 0.05+0.04=0.09

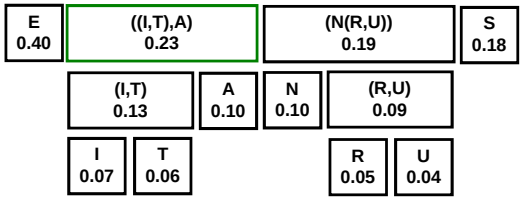
2°) I et T : 0.07+0.06=0.13



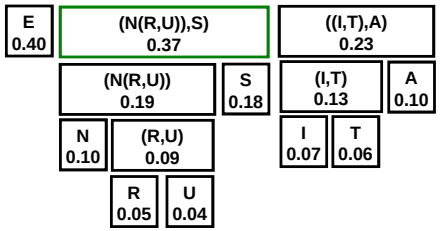
3°) N et (R,U) : 0.10+0.09=0.19



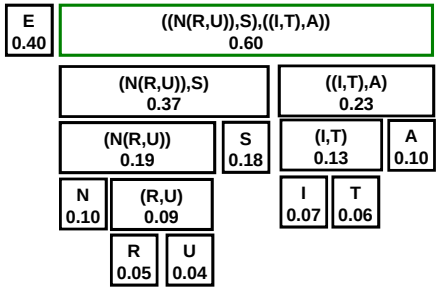
4°) (I,T) et A : 0.13+0.10=0.23



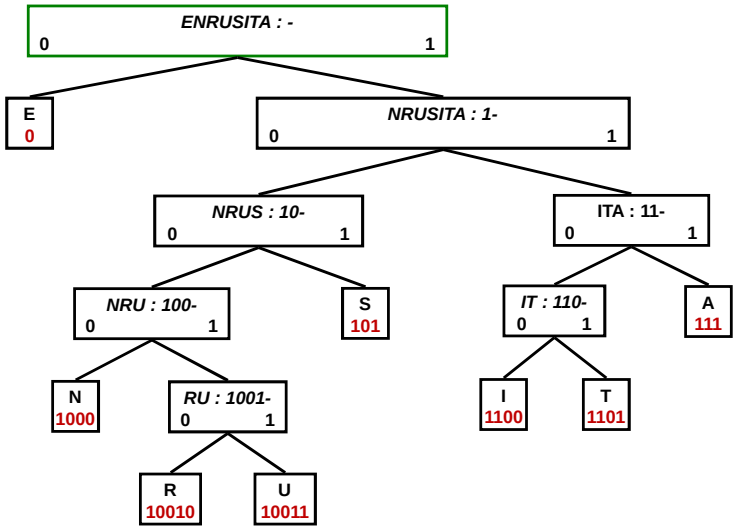
5°) (N(R,U)) et S : 0.19+0.18=0.37



6°) (N(R,U)) et ((I,T),A) : 0.37+0.23=0.60



7°) Arborescence finale (et codes des caractères)



8°) Code final :

Caractère	E	S	A	N	I	T	R	U
fréquence	0.40	0.18	0.10	0.10	0.07	0.06	0.05	0.04
code	0	101	111	1000	1100	1101	10010	10011

Compression prévisible de l'exemple :

Codage sans compression sur 3 b
Après compression : 0.40*1+(0.18+0.10)*3+(0.10+0.07+0.06)*4+(0.05+0.04)*5**=2.61 b**

→ Taux ~1.15:1 (*NB Taux variable selon textes à compresser*)

★ **Autres méhodes** : ***méthode descendante de la médiane (raffinement de partition)***

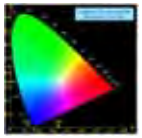
Caractère	E	S	A	N	I	T	R	U
fréquence	0.40	0.18	0.10	0.10	0.07	0.06	0.05	0.04

← 0.58 → ← 042 →

Compression prévisible de l'exemple → 3:2.64

2.2 Compression avec perte

L'exemple des images



1°) Encodage de l'information :
- **Par couleurs : RVB ou CJMN (jusqu'à 32 b/pixel)**
- **Par luminance-chrominance : YUV, YCbCr**

← physiologie humaine

2°) Compression
- **en limitant les couleurs (GIF : 2^8 parmi 2^{24})**
- **en privilégiant les contours (PNG)**
- **en privilégiant les images (JPG, taux variable)**

← combiner plusieurs méthodes

+ formats "conteneurs"

★ **Exemple du JPG**
- ***pixels groupés en matrices (8x8 ou 16x16)***
- ***encodage YCbCr***
- ***Sous-échantillonnage***
- ***Cosinus discret (DCT)***
- ***Quantification***
- ***Compression RLE puis Huffman***

3. Chiffrement

(*encryption* / "cryptage", encryptage)

- Concevoir des algorithmes de chiffrement pour communiquer en sécurité : cryptographie

utilisation de "clés" de chiffrement *(cf Hamming)* **:**
- chiffrement symétrique : même clé pour chiffrer et déchiffrer (clé privée)
exp DES
- chiffrement asymétrique : une clé pour chiffrer (publique), une pour déchiffrer (privée)
exp RSA, PGP

- Casser ces algorithmes : cryptanalyse

- difficulté proportionnelle à la taille de la clé (sym. : 40 à 128b / asym. : 256 à 2048b)
- difficultés propres au système de chiffrement

→ quelques exemples (simples) de chiffrement

3.1. Chiffrement par translation

§ **"Le" chiffre de César** (Rome, I°s. AC)

Translation de code : A→D (clé privée : +3) ***"vl ylv rdfhp sdud ehooxp"***

§ **Le baton de Plutarque** (Sparte, V°s. AC)

► **Casser les translations**

→ décalages successifs

3.2. Chiffrement par substitution

§ **Le chiffre de Polybe** (Grèce, II°s. AC)

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I	J
3	K	L	M	N	O
4	P	Q	R	S	T
5	U	V W	X	Y	Z

"3211432215344515444532153415432114153211225115434315"

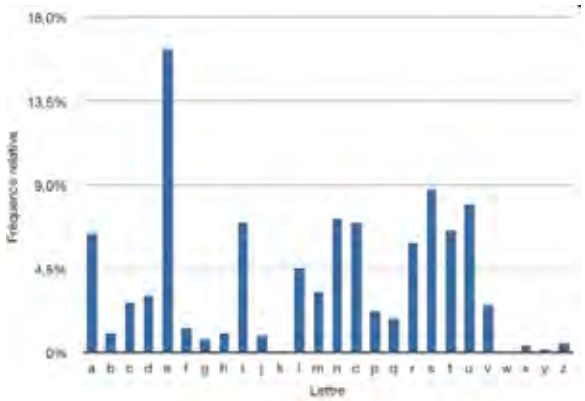
§ **Le chiffre de Vigénère** (XVI°s.)

Exemple :
mot-clé : abraca (1,2,18,1,3,1)
message : "secretesmanieresdecrire"
chiffrement : secret esmani eresde crire ← **table de Vigénère**
abraca abraca abraca abra
tgushuf.... **s+a=t**

→ **cassé au XIX°s. par C. Babbage**

► **Casser les substitutions simples**

→ **Utiliser les fréquences probables** (Français : ESANITRULODCMPF...)



Polybe → 32 11 43 22 E 34 45 E 44 45 32 E 34 E 43 21 14 E 32 11 22 51 E 43 43 E

§ **Substitution homophonique**

Pour éviter la vulnérabilité des fréquences :
Plusieurs substitutions possibles pour les caractères les plus fréquents

← **insuffisant si cryptanalyse par force brute**

§ **Enigma** (Allemagne 1939-45)

← **cassé par les Anglais**

3.3. Méthodes modernes

► **Chiffrer bit par bit**

- Utiliser le XOR entre message et clé binaire (cf Vigénère)
→ même algorithme pour crypter et décrypter

[convention Java : dans le message à crypter, \u0020 pour code ASCII #32 (espace)]

► **Utiliser les nombres premier**

- Exemple de l'algorithme RSA

► **Entre sécurité et ouverture**

- Exemple de l'algorithme PGP