Plan

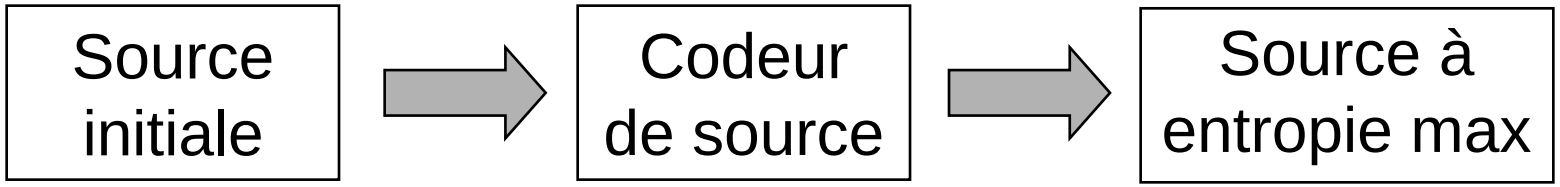
- 1. Introduction
- 2. Vue d’ensemble
- 3. Sources discrètes & Entropie
- 4. Canaux discrets & Capacité
- 5. Codage de source
- 6. Codage de canal
- 7. Cryptographie
- 8. Conclusion

5. Codage de source

✓**Adapter la source au canal** ⇨ l'alphabet
⇨ le débit

✓Utiliser la capacité du canal ⇨ maximiser I(X,Y)

- Hyp : Source stationnaire, canaux **sans perturbation**



Codeur de source ⇨ supprimer la **redondance**

- **Propriétés d'un codeur de source**

  ✓**Régularité** : messages ≠ ⇨ codes ≠

  ✓**Déchiffrabilité** : séparation des mots non ambiguë

- **Mot-code**

  $[S]=[s_1,s_2, \ldots, s_N]$ $[X]=[x_1,x_2, \ldots, x_D]$

  ⇘ $[C]=[c_1,c_2, \ldots, c_N]$

- **Exemple**

Symbole	Code A	Code B	Code C	Code D
s_1	00	0	0	0
s_2	01	10	01	10
s_3	10	110	011	110
s_4	11	1110	0111	111

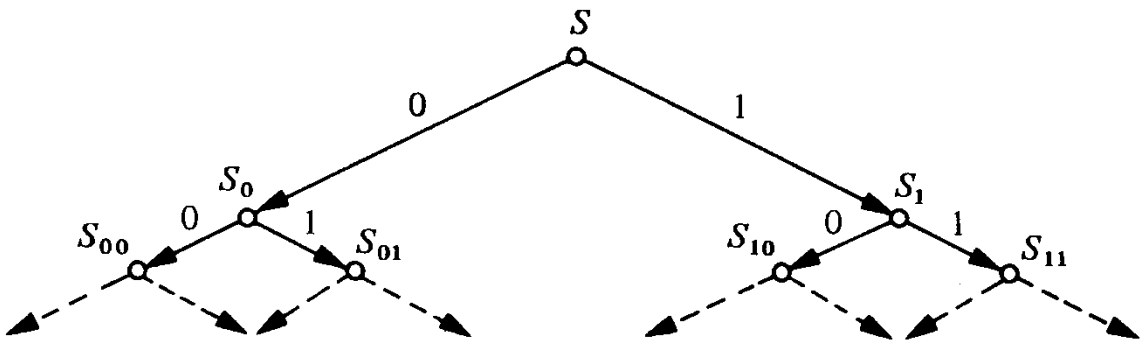
✓**Code à longueur variable / fixe**

✓**Code à décodage unique** : mot-code ⇔ symbole unique

✓**Code séparable** : pas de signe de démarcation entre les mots

✓**Code instantané** ou **irréductible** : on détermine les mots-codes à mesure que l'on reçoit les lettres de l'alphabet du code.
CNS : Aucun mot-code n'est le préfixe d'un autre !

• **Arbre & codes binaires instantanés**



- **Longueur moyenne d'un mot-code**

$$\bar{l} = \sum_{i=1}^{N} p(s_i).l_i$$

- **Limite de la longueur moyenne**

$$H(S) = H(C) = \bar{l}.H(X) \quad \Rightarrow \quad \bar{l} \geq \frac{H(S)}{\log D} = l_{\min}$$

- **Capacité - Efficacité - Redondance**

$$C = Max(H(X)) = \log D \qquad \eta = \frac{H(X)}{\log D} \qquad \rho = \frac{\log D - H(X)}{\log D}$$

$$\eta = \frac{H(S)}{\bar{l}.\log D}$$

Ex → code opt.

- **Codes optimaux absolus**

  Codes dont l'efficacité est maximale : $\eta = 1$

  ↳ $\bar{l} = l_{\min} = \dfrac{H(S)}{\log D}$

  ↳ $\boxed{\sum_{i=1}^{N} D^{-l_i} = 1}$ ⇨ Condition nécessaire pour les codes optimaux absolus

- **Théorème des canaux sans bruit (codage de source)**

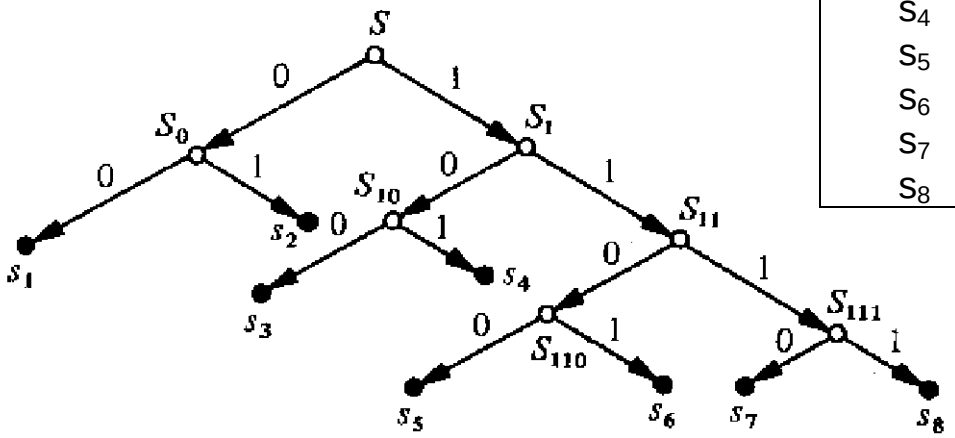
" Par un codage approprié (codage par groupe de n symboles de la source), l'information moyenne par lettre de l'alphabet du code peut être amenée aussi proche que l'on veut de la capacité du code, c'est-à-dire qu'**il existe toujours un codage optimal absolu** ."

Rq1 : à n fixé, le code qui donne $\eta_{max}<1$ est dit 'optimal'

Rq2 : en pratique, on travaillait à n=1

- **Codage de Shannon-Fano**

Algorithme de génération d'un **codage optimal absolu**, pour des sources divisibles récursivement (jusqu'à un symbole par ensemble) en deux sous-ensembles équiprobables.



Symboles s_k	Proba $p(s_k)$					Mots-codes c_k	Longueur l_k
s_1	0.25	0	0			00	2
s_2	0.25		1			01	2
s_3	0.125	1	0	0		100	3
s_4	0.125			1		101	3
s_5	0.0625		1	0	0	1100	4
s_6	0.0625				1	1101	4
s_7	0.0625			1	0	1110	4
s_8	0.0625				1	1111	4

- **Codage binaire de Huffman** (1952)

- Algorithme de génération d'un **codage optimal** symbole par symbole.

- Code à longueur variable ⇨ codes longs pour probas faibles

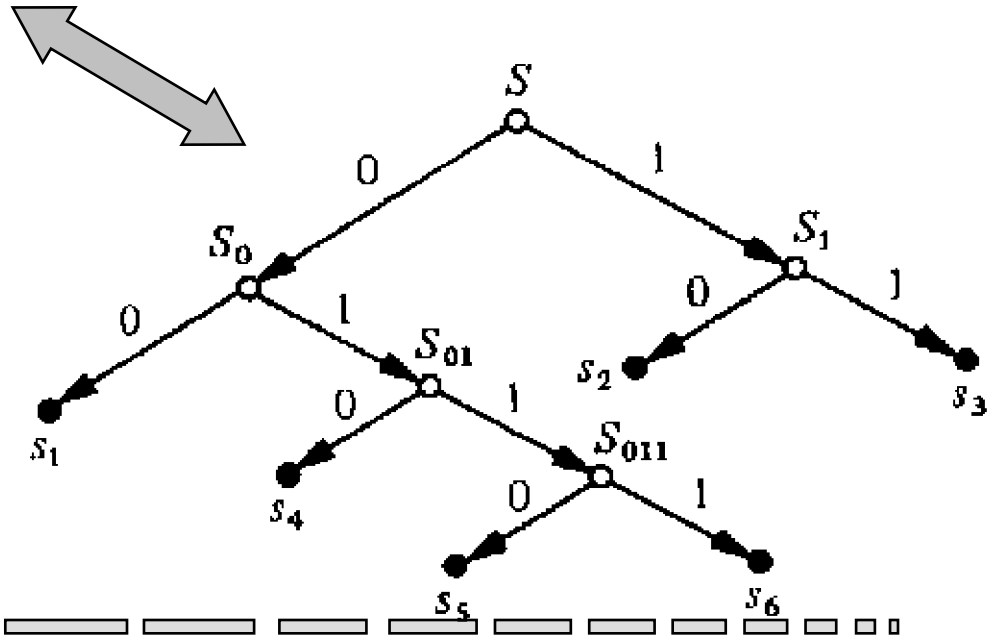
- Algorithme

① Extraction des probabilités
② Création de l'arbre
③ Création de la table d'Huffman
④ Codage

⇨On transmet la table + les codes en binaire ⇨

❶ Lecture de la table d'Huffman
❷ Création de l'arbre de décodage
❸ Lecture séquentielle et décodage

Messages s_k $[R_0]$	Probabilités $p(s_k)$	Mots-code c_k	Sources restreintes $[R_1]$		$[R_2]$		$[R_3]$		$[R_4]$	
									0,60	(0)
							0,40	(1)	0,40	(1)
s_1	0,30	00	0,30	(00)	0,30	(00)	0,30	(00)		
					0,30	(01)	0,30	(01)		
s_2	0,25	10	0,25	(10)	0,25	(10)				
s_3	0,15	11	0,15	(11)	0,15	(11)				
s_4	0,15	010	0,15	(010)						
			0,15	(011)						
s_5	0,10	0110								
s_6	0,05	0111								



Rq : code d'échappement
= Huffman + fixe

- **Codage Arithmétique** (1976)
  - Huffman ⇨ 1 symbole = 1 mot-code
  - Arithmétique ⇨ 1 flot de symboles = nbre en virgule flottante

❒ **Codeur**

m=0 ; M=1 ;
Tant que !(fin de fichier)
{
i = symbole suivant;
soit $[a_i ; b_i]$ associé à i ;
s = M-m ;
$M = m + s.b_i$;
$m = m + s.a_i$;
}
Renvoyer m, le compacté du fichier

❒ **Decodeur**

N = nombre codé ;
Faire
{
trouver i / $N \in [a_i ; b_i[$;
sortir i ;
$s = b_i - a_i$;
$N = (N - a_i) / s$;
}
Tant qu'il reste un symbole à lire

- Exemple

s_i	p_i	$[a_i ; b_i[$	$Huff_i$
␣	0.1	[0.0 ; 0.1[	111
A	0.1	[0.1 ; 0.2[	110
E	0.1	[0.2 ; 0.3[	101
I	0.1	[0.3 ; 0.4[	100
B	0.1	[0.4; 0.5[	0111
G	0.1	[0.5 ; 0.6[	0110
L	0.2	[0.6 ; 0.8[	00
S	0.1	[0.8; 0.9[	0100
T	0.1	[0.9 ; 1.0[	0101

0.4372207712 = ?

10111010 10100100 11011001 01
01111000 00011101 10110010 11010100

Arithmétique	≥	Huffman

☹ + de calcul

Proba très élevée →1 bit
Peu de symboles (↓)
⇩
Run Length

Codeurs statistiques

- Dépendants de la qualité de la statistique
- Statistique connue par le décodeur

- **Codage par longueur de plage** (Run length coding)

  ⇨Coder le nombre de symboles identiques

00000111110000000000000000	⇨	5w5b17w
00000000000111110000000000	⇨	11w5b11w
A B C C C C C C A B C A B C	⇨	A B !6C A B C A B C

- CCITT, Fax groupe III

  ↳ Huffman sur les plages de 0 précédant les 1

- JPEG

  ↳ Huffman sur les plages de 0 précédant les coeff. DCT

- Table d'Huffman FAX III

0	00110101
1	000111
2	0111
3	1000
4	1011
5	1100
6	1110
7	1111
8	10011
9	10100
10	00111
11	01000
12	001000
13	000011
14	110100
15	110101
16	101010
17	101011
18	0100111
19	0001100
20	0001000
21	0010111
22	0000011
23	0000100
24	0101000
25	0101011
26	0010011
27	0100100
28	0011000
29	00000010
30	00000011
31	00011010

32	00011011
33	00010010
34	00010011
35	00010100
36	00010101
37	00010110
38	00010111
39	00101000
40	00101001
41	00101010
42	00101011
43	00101100
44	00101101
45	00000100
46	00000101
47	00001010
48	00001011
49	01010010
50	01010011
51	01010100
52	01010101
53	00100100
54	00100101
55	01011000
56	01011001
57	01011010
58	01011011
59	01001010
60	01001011
61	00110010
62	00110011
63	00110100

64	11011
128	10010
192	010111
256	0110111
320	00110110
384	00110111
448	01100100
512	01100101
576	01101000
640	01100111
704	011001100
768	011001101
832	011010010
896	011010011
960	011010100
1024	011010101
1088	011010110
1152	011010111
1216	011011000
1280	011011001
1344	011011010
1408	011011011
1472	010011000
1536	010011001
1600	010011010
1664	011000
1728	010011011
EOL	000000000001

Table 6.5 - Mots de code spécifiant la longueur des séquences correspondant au blanc lors de la transmission de documents par télécopie.

- **Codage de type dictionnaire** (1977)

⇨ Coder une extension de la source de longueur variable

1977 : LZ (Lempel & Ziv) ⇨ 1984 : LZW (Welch)

✓**Dictionnaire** de symboles incrémenté dynamiquement
↳ **apprentissage**

✓ Fichier codé = suite des adresses des mots du dico

! Gérer l'incrément des bits d'adresse

PKZIP, ARJ ⇦ LZW + Huffman

❒ Codeur LZW

$\mathbb{D} = \{C_i, W_i\}$, $P = \varnothing$

Tant que (symboles à coder)
    C = symbole suivant
    Si $P \oplus C \in \mathbb{D}$
        $P = P \oplus C$
    Sinon
        sortir W_P
        $P \oplus C \rightarrow \mathbb{D}$
        P=C
    Fin si
Fin tant que

sortir W_P

A B B A B A B A C ….

❒ Décodeur LZW

$\mathbb{D} = \{C_i, W_i\}$
cW = 1^{er} code ; sortir s(cW)

Tant que (codes à lire)
    pW = cW
    cW = code suivant
    Si (s(cW) $\in \mathbb{D}$)
        sortir s(cW)
        P = s(pW)
        C = 1^{er} symbole de s(cW)
        $P \oplus C \rightarrow \mathbb{D}$
    Sinon
        P = s(pW)
        C = 1^{er} symbole de s(pW)
        sortir s(P⊕C)
        $P \oplus C \rightarrow \mathbb{D}$
    Fin si
Fin tant que

- **Conclusion sur le codage de source**

Utilisé en **compression audio & vidéo** (JPEG, MPEG ...) mais en étant associé à des algorithmes non réversibles (avec pertes)

Supprime la redondance

⇨ Sensibilité au bruit

⇨ Codage de canal