Arbres de codage

Algorithme de Huffman

Jean-Marc Vincent[1]

[1] Laboratoire LIG
Universités de Grenoble
Jean-Marc.Vincent@imag.fr

Codage

- Symboles $\mathcal{S} = \{s_1, \cdots, s_k\}$
- Codage

$$\begin{array}{rcl} C : \mathcal{S} & \longrightarrow & \{0,1\}^* \\ s_i & \longmapsto & c(s_i) \text{ longueur } l_i \end{array}$$

- Uniquement déchiffrable (CS propriété du préfixe) ;
- **Inégalité de Kraft** Pour un codage ayant la propriété du préfixe

$$\sum_{i=1}^{k} 2^{-l_i} \leq 1, \tag{1}$$

Réciproquement, si (1) il existe un codage avec la propriété du préfixe.

Complexité d'un code

Sources aléatoires : $p_1, \cdots, p_k$ fréquence de transmission ;
longueur moyenne du codage

$$L(c) = \sum_{i=1}^{k} p_i l_i;$$

$L_{inf} = \inf_c L(c)$; h ayant la propriété du préfixe

Théorème (Shannon 1948)

$$\mathcal{H}(p) \leq L_{inf} \leq \mathcal{H}(p) + 1.$$

Complexité d'un code

Sources aléatoires : $p_1, \cdots, p_k$ fréquence de transmission ;
longueur moyenne du codage

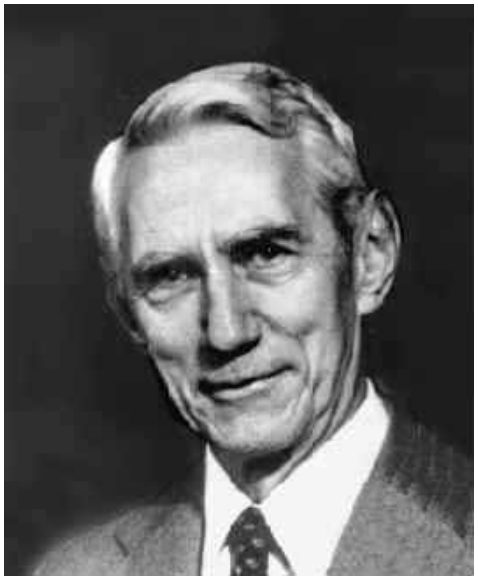
$$L(c) = \sum_{i=1}^{k} p_i l_i;$$

$L_{inf} = \inf_c L(c)$; h ayant la propriété du préfixe

Théorème (Shannon 1948)

$$\mathcal{H}(p) \leq L_{inf} \leq \mathcal{H}(p) + 1.$$

Claude Shannon (1916-2001)



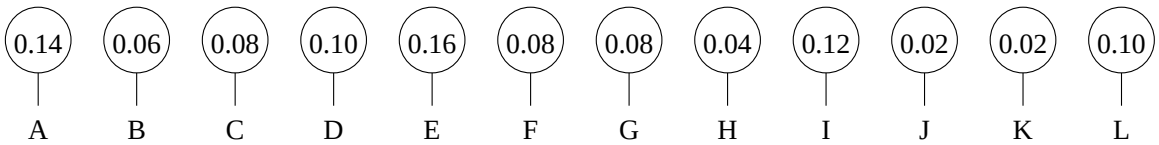
Claude Elwood Shannon (30 avril 1916 à Gaylord, Michigan - 24 février 2001), ingénieur électrique, est l'un des pères, si ce n'est le père fondateur, de la théorie de l'information. Son nom est attaché à un célèbre "schéma de Shannon" très utilisé en sciences humaines, qu'il a constamment désavoué.

Il étudia le génie électrique et les mathématiques à l'Université du Michigan en 1932. Il utilisa notamment l'algèbre booléenne pour sa maîtrise soutenue en 1938 au MIT. Il y expliqua comment construire des machines à relais en utilisant l'algèbre de Boole pour décrire l'état des relais (1 : fermé, 0 : ouvert).
Shannon travailla 20 ans au MIT, de 1958 à 1978. Parallèlement à ses activités académiques, il travailla aussi aux laboratoires Bell de 1941 à 1972.
Claude Shannon était connu non seulement pour ses travaux dans les télécommunications, mais aussi pour l'étendue et l'originalité de ses hobbies, comme la jonglerie, la pratique du monocycle et l'invention de machines farfelues : une souris mécanique sachant trouver son chemin dans un labyrinthe, un robot jongleur, un joueur d'échecs (roi tour contre roi)...
Souffrant de la maladie d'Alzheimer dans les dernières années de sa vie, Claude Shannon est mort à 84 ans le 24 février 2001.

Wikipedia

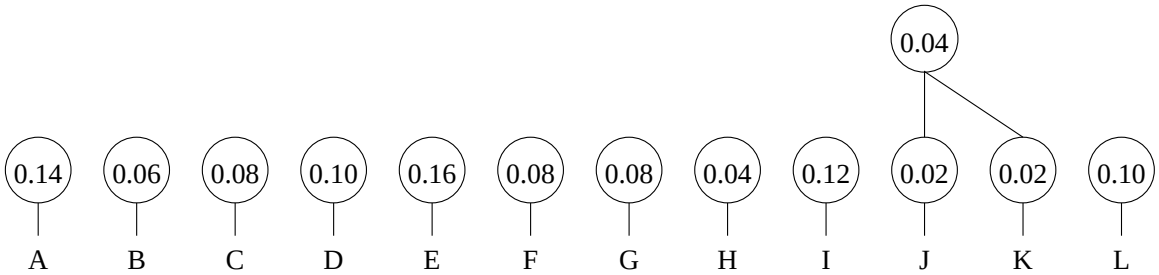
Algorithme de Huffman (1951)

A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10



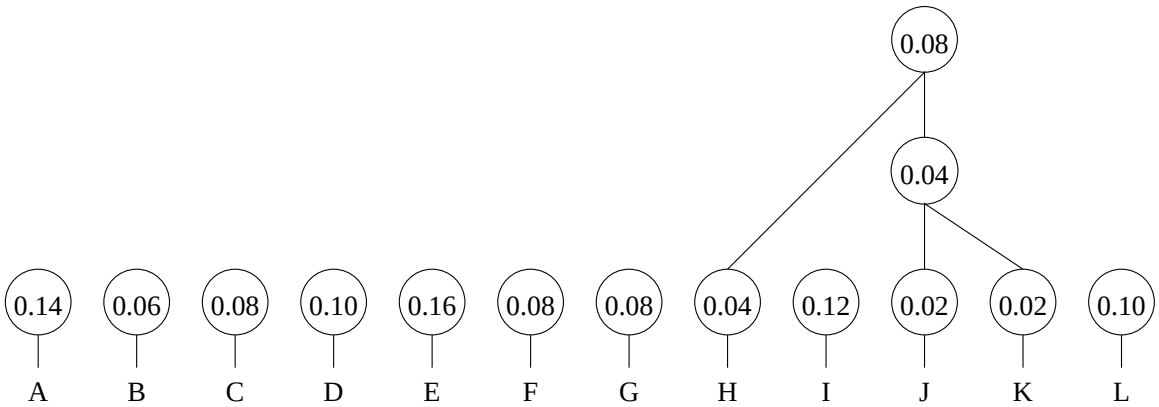
Algorithme de Huffman (1951)

A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10



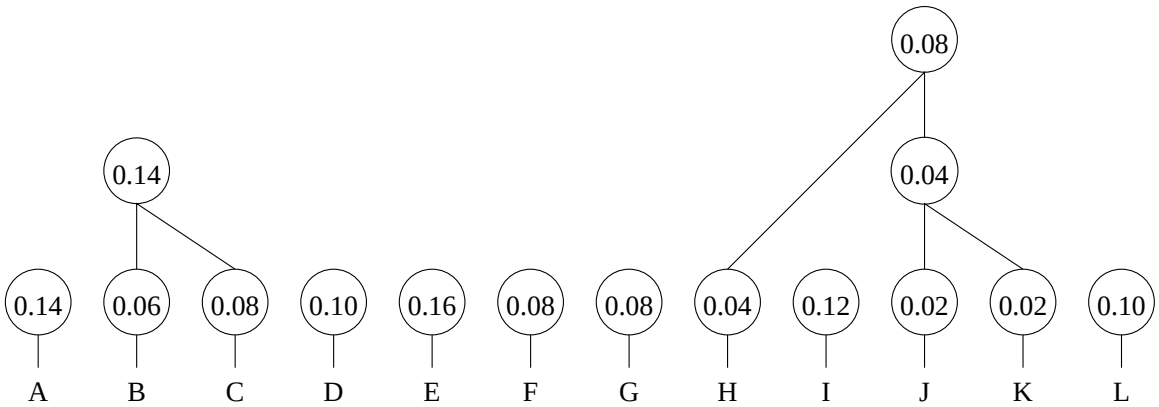
Algorithme de Huffman (1951)

A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10



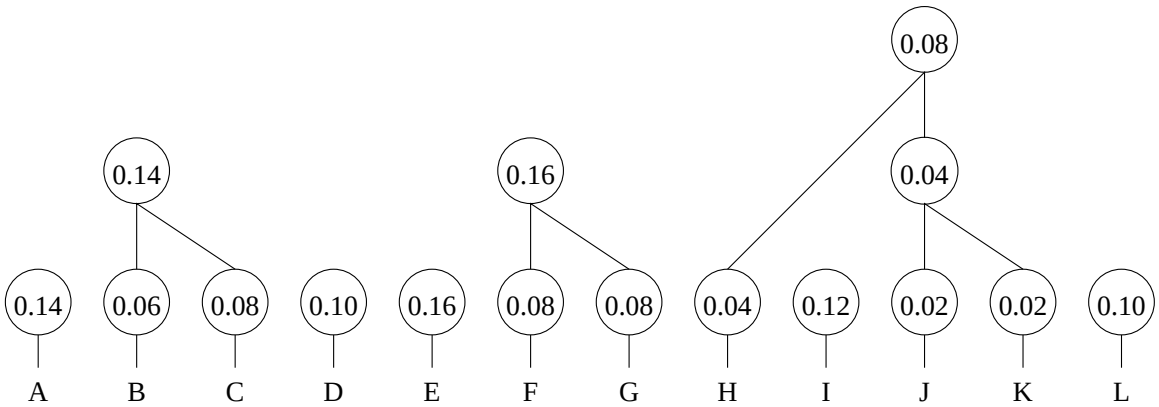
Algorithme de Huffman (1951)

A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10



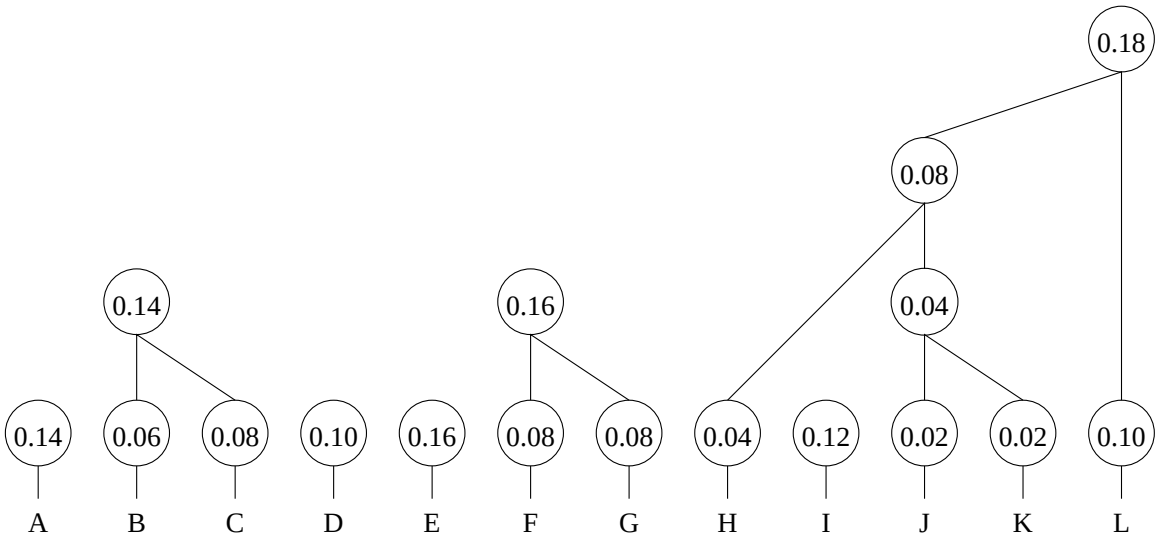
Algorithme de Huffman (1951)

A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10



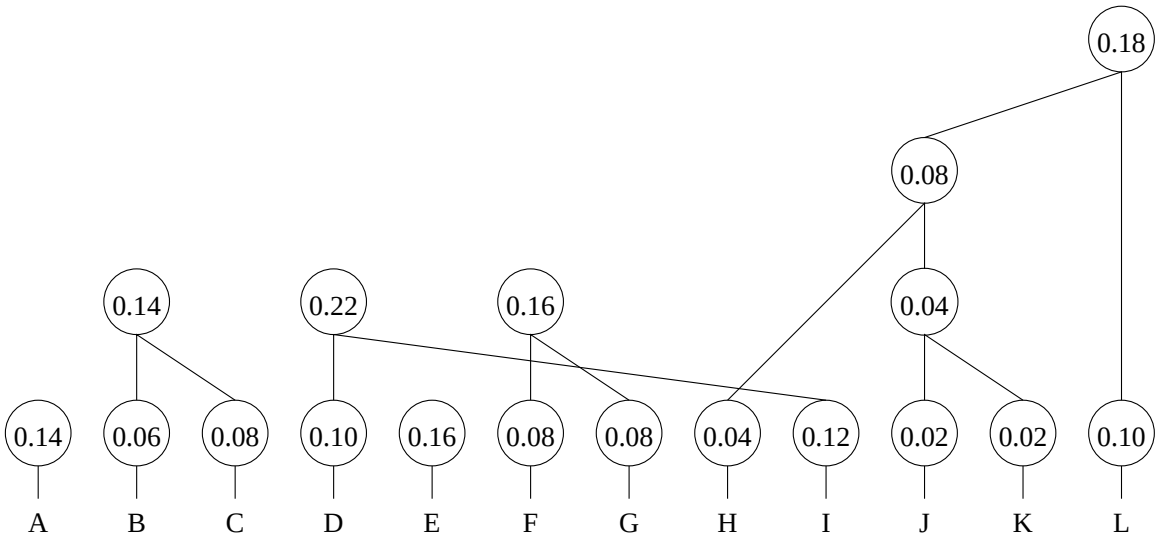
Algorithme de Huffman (1951)

A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10



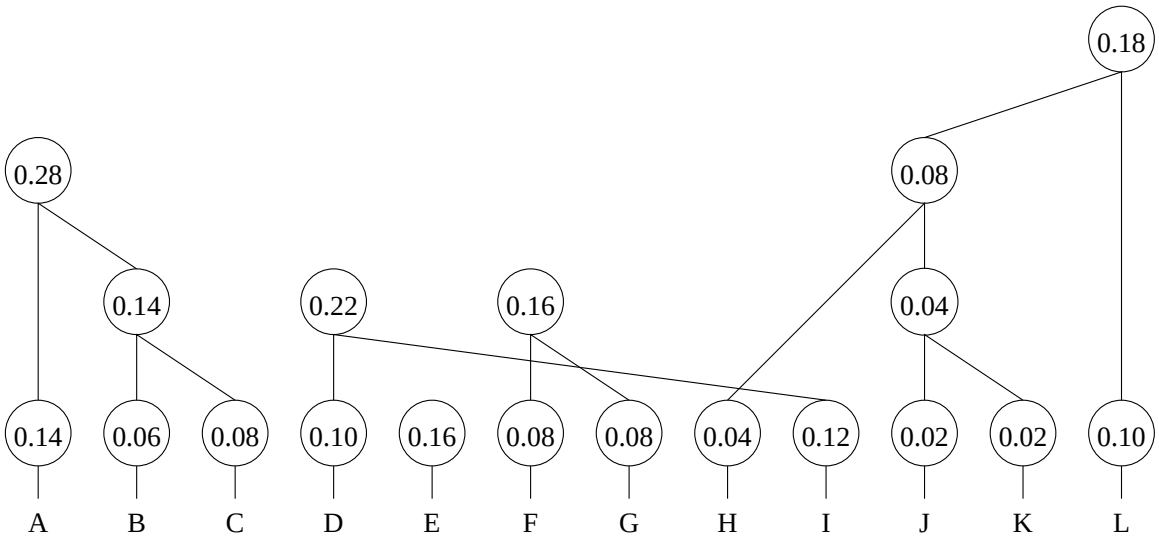
Algorithme de Huffman (1951)

A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10



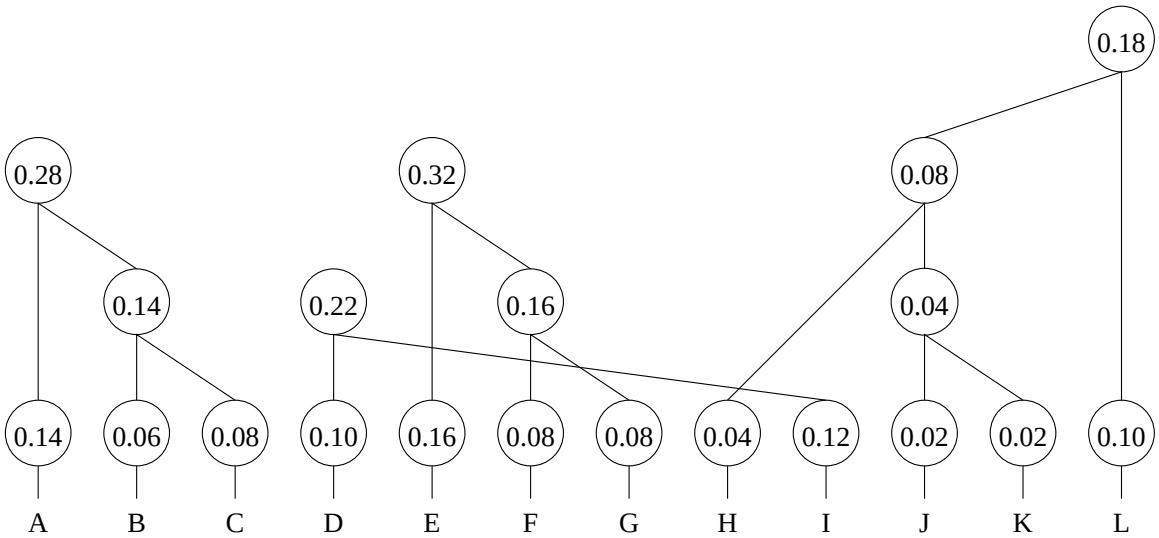
Algorithme de Huffman (1951)

A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10

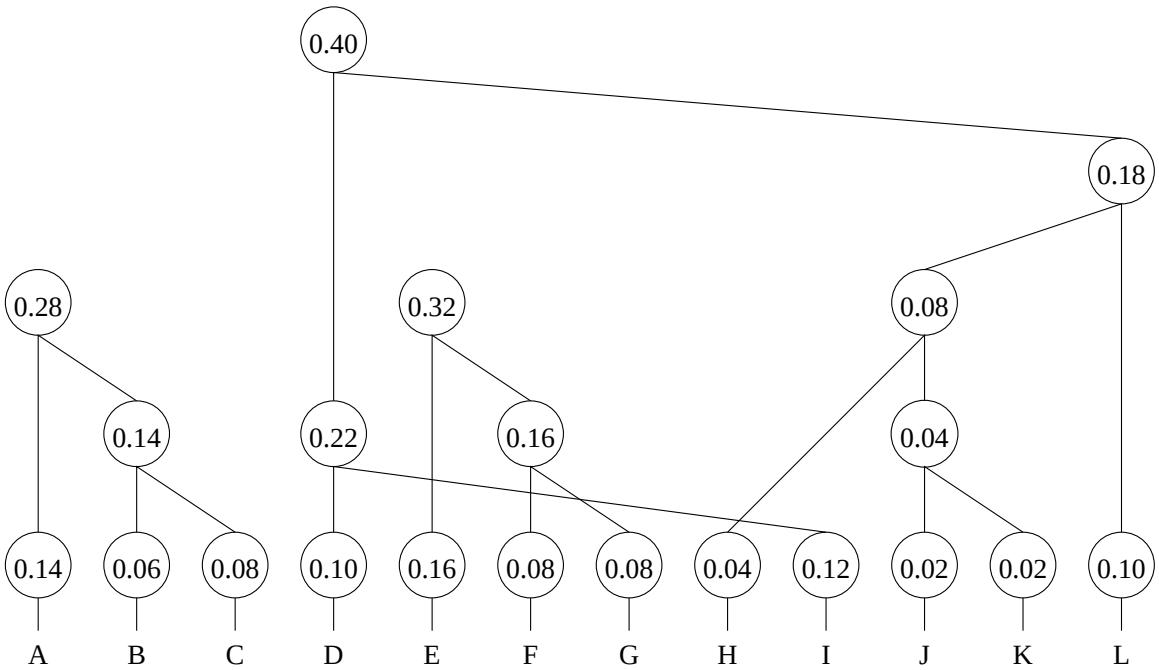


Algorithme de Huffman (1951)

A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10

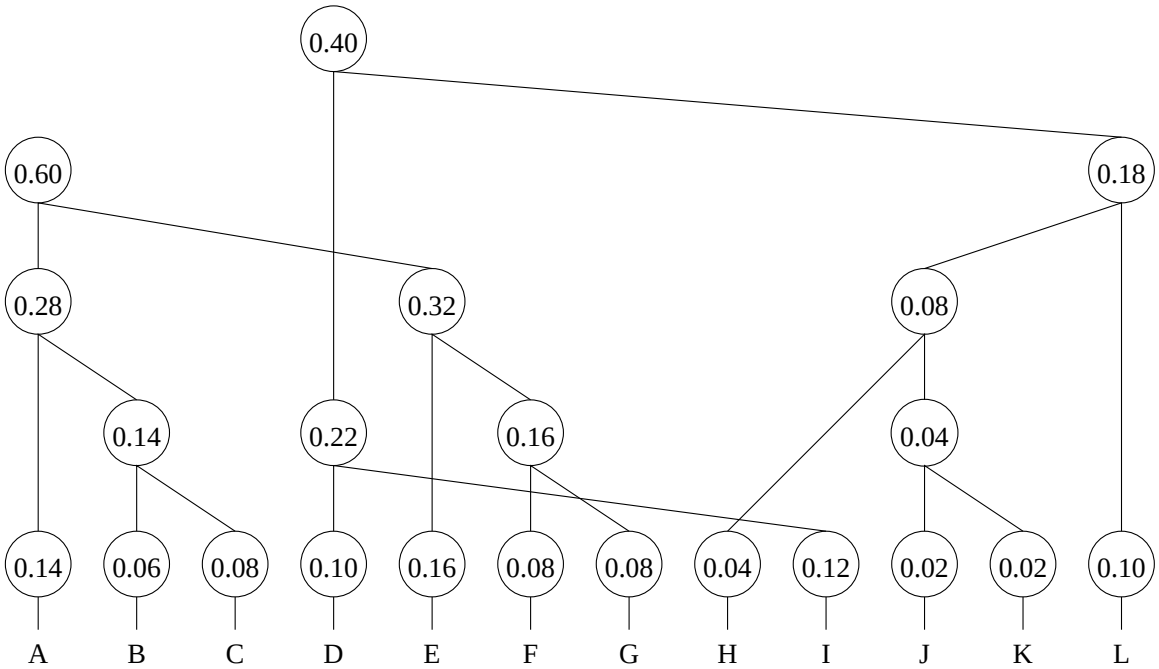


Algorithme de Huffman (1951)



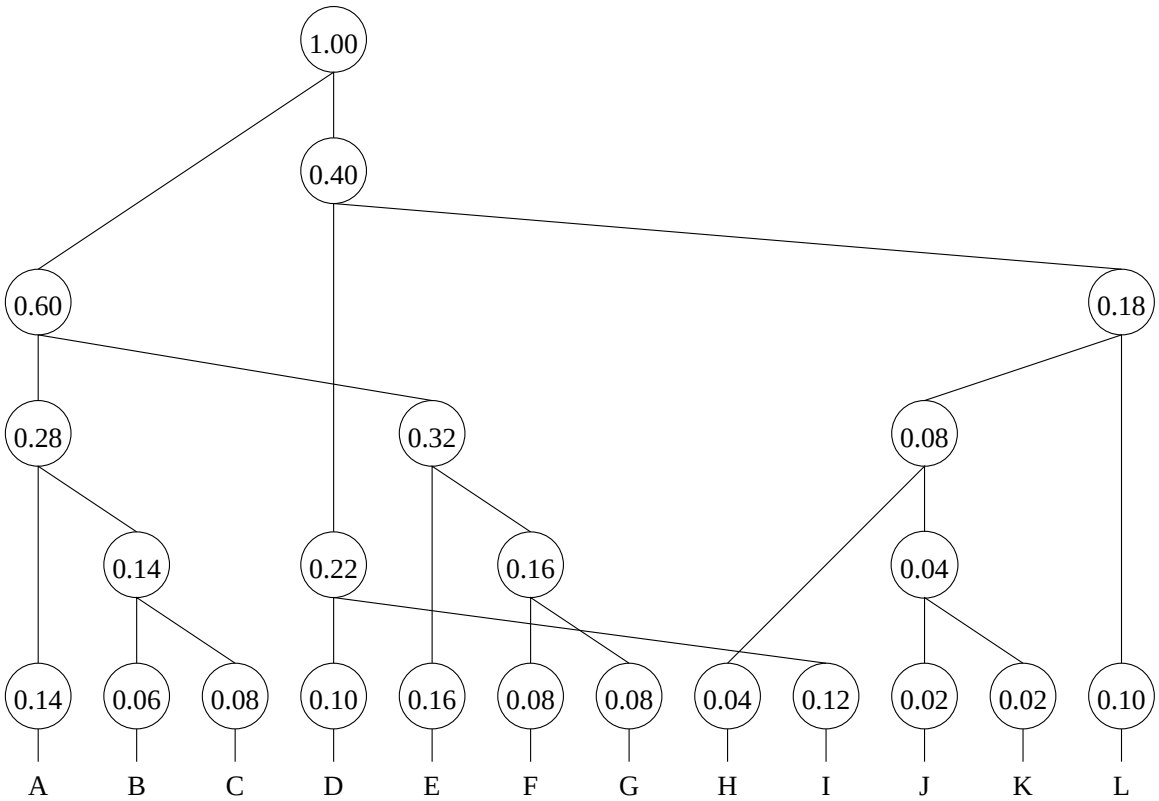
A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10

Algorithme de Huffman (1951)



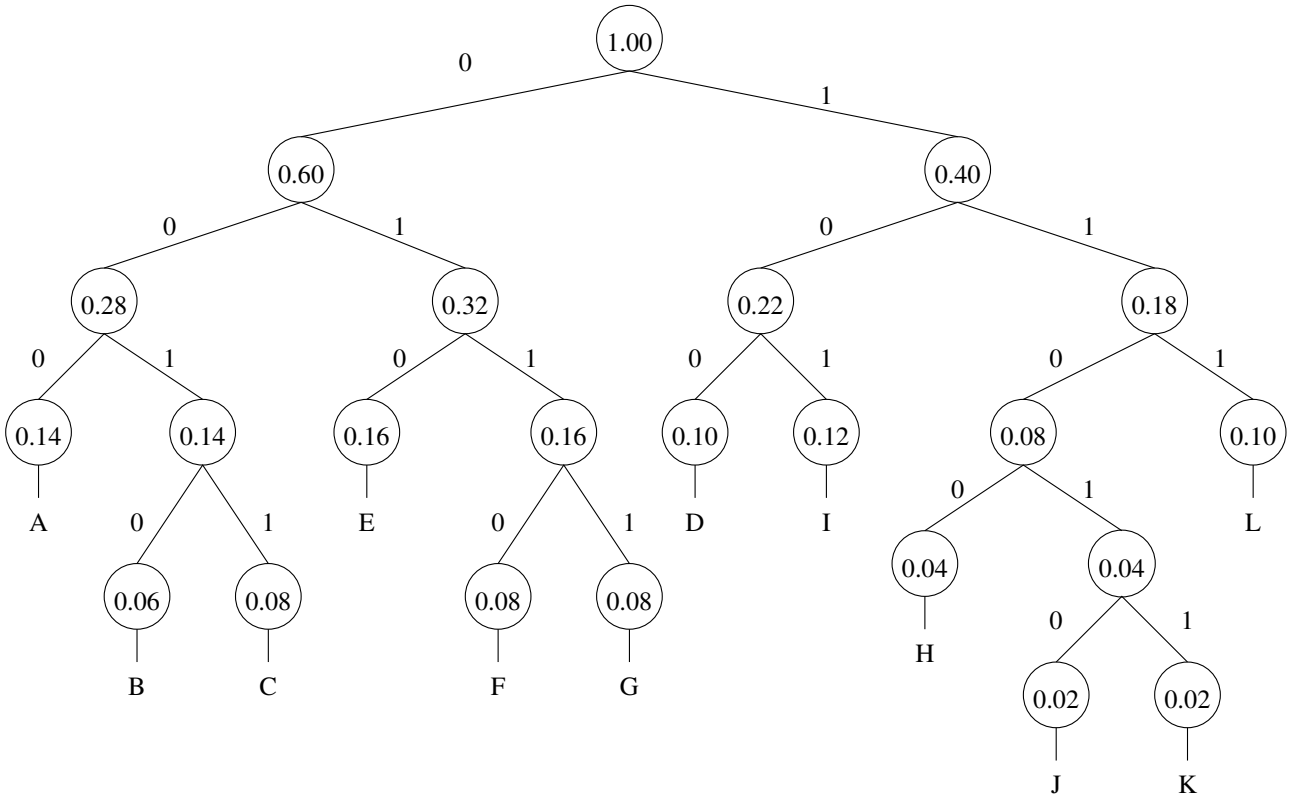
A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10

Algorithme de Huffman (1951)



A	0.14
B	0.06
C	0.08
D	0.10
E	0.16
F	0.08
G	0.08
H	0.04
I	0.12
J	0.02
K	0.02
L	0.10

Algorithme de Huffman (1951)



A	0.14	000
B	0.06	0010
C	0.08	0011
D	0.10	100
E	0.16	010
F	0.08	0110
G	0.08	0111
H	0.04	1100
I	0.12	101
J	0.02	11010
K	0.02	11011
L	0.10	111

Codage optimal : L-moy = 3.42, Entropie = 3.38

Profondeur = $-\log_2$(probabilité)

Généralisation Lempel-Ziv,...

Algorithme de Huffman (1951) : Implantation

{Alphabet à coder C_1, ... C_k : Caractères}
{Proportion des caractères $p_1, \cdots, p_k$}
file_priorité Q=$\emptyset$ {File à priorité de nœuds pondérés}
pour i=1 ; i $\leq$k ; i++ **faire**
  z=nouveau_nœud()
  z.etiquette=C_i ; z.poids=p_i
  z.gauche=NIL ; z.droit=NIL
  Insérer(Q,z)
fin pour
pour i=1 ; i $\leq$k-1 ; i++ **faire**
  z=nouveau_nœud()
  z.gauche= Extrait_min(Q) ; z.droit=Extrait_min(Q)
  z.poids=z.gauche.poids + z.droit.poids
  Insérer(Q,z)
fin pour
retourner Extrait_min(Q)

Algorithme de Huffman (1951) : Preuve

Optimalité

L'algorithme de Huffman produit un code ayant la propriété du préfixe de longueur moyenne optimale.

Algorithme glouton : tout choix est définitif
à tout moment de l'algorithme la forêt construite est une sous-forêt d'un arbre optimal.
Remarque : algorithme en $\mathcal{O}(k \log k)$

Algorithme de Huffman (1951) : Preuve

Lemme : fréquences faibles

Soit C un alphabet de k lettres. Soit x et y deux caractères de plus petite fréquence. Alors il existe un codage préfixé optimal pour C tel que les mots codes de x et de y ne diffèrent que par le dernier bit.

Idée : prendre un arbre optimal et le transformer de manière à vérifier la propriété.

Algorithme de Huffman (1951) : Preuve

Lemme : fréquences faibles

Soit C un alphabet de k lettres. Soit x et y deux caractères de plus petite fréquence. Alors il existe un codage préfixé optimal pour C tel que les mots codes de x et de y ne diffèrent que par le dernier bit.

Idée : prendre un arbre optimal et le transformer de manière à vérifier la propriété.

Algorithme de Huffman (1951) : Preuve

Lemme : propagation de l'optimalité

Soit T un arbre de codage optimal (complet) de C. Alors la fusion z de 2 feuilles sœurs x et y affectée de la somme des fréquences des feuilles $f(z) = f(x) + f(y)$ produit un arbre optimal pour l'alphabet C' dans lequel tous les caractères x et y ont été remplacés par z.

Idée : raisonner par l'absurde

Algorithme de Huffman (1951) : Preuve

Lemme : propagation de l'optimalité

Soit T un arbre de codage optimal (complet) de C. Alors la fusion z de 2 feuilles sœurs x et y affectée de la somme des fréquences des feuilles $f(z) = f(x) + f(y)$ produit un arbre optimal pour l'alphabet C' dans lequel tous les caractères x et y ont été remplacés par z.

Idée : raisonner par l'absurde

Algorithme de Huffman (1951) : Preuve

Invariant de la boucle :

La file à priorité contient une forêt incluse dans un arbre de codage optimal de l'alphabet C.

Cette précondition est vraie en début de l'algorithme.
Preuve partielle : Si la précondition est vraie à l'entrée de l'itération, il existe un arbre optimal contenant la forêt incluse dans la file à priorité. Soit x et y les nœuds extraits de la FAP, d'après le lemme 1 il existe un arbre optimal tq x et y soient 2 feuilles sœurs. Par le lemme 2 l'arbre optimal, lorsque l'on réalise la fusion de x et y reste optimal. CQFD.
Terminaison : l'algorithme, faisant un nombre fini d'itérations, se termine. On peut montrer que chaque itération diminue de 1 le nombre de nœuds dans la FAP. A la fin des itérations il ne reste qu'un nœud racine de l'arbre optimal.

Algorithme de Huffman (1951) : Preuve

Invariant de la boucle :

La file à priorité contient une forêt incluse dans un arbre de codage optimal de l'alphabet C.

Cette précondition est vraie en début de l'algorithme.

Preuve partielle : Si la précondition est vraie à l'entrée de l'itération, il existe un arbre optimal contenant la forêt incluse dans la file à priorité. Soit x et y les nœuds extraits de la FAP, d'après le lemme 1 il existe un arbre optimal tq x et y soient 2 feuilles sœurs. Par le lemme 2 l'arbre optimal, lorsque l'on réalise la fusion de x et y reste optimal. CQFD.

Terminaison : l'algorithme, faisant un nombre fini d'itérations, se termine. On peut montrer que chaque itération diminue de 1 le nombre de nœuds dans la FAP. A la fin des itérations il ne reste qu'un nœud racine de l'arbre optimal.

Algorithme de Huffman (1951) : Preuve

Invariant de la boucle :

La file à priorité contient une forêt incluse dans un arbre de codage optimal de l'alphabet C.

Cette précondition est vraie en début de l'algorithme.
Preuve partielle : Si la précondition est vraie à l'entrée de l'itération, il existe un arbre optimal contenant la forêt incluse dans la file à priorité. Soit x et y les nœuds extraits de la FAP, d'après le lemme 1 il existe un arbre optimal tq x et y soient 2 feuilles sœurs. Par le lemme 2 l'arbre optimal, lorsque l'on réalise la fusion de x et y reste optimal. CQFD.
Terminaison : l'algorithme, faisant un nombre fini d'itérations, se termine. On peut montrer que chaque itération diminue de 1 le nombre de nœuds dans la FAP. A la fin des itérations il ne reste qu'un nœud racine de l'arbre optimal.

Algorithme de Huffman (1951) : Preuve

Invariant de la boucle :

La file à priorité contient une forêt incluse dans un arbre de codage optimal de l'alphabet C.

Cette précondition est vraie en début de l'algorithme.
Preuve partielle : Si la précondition est vraie à l'entrée de l'itération, il existe un arbre optimal contenant la forêt incluse dans la file à priorité. Soit x et y les nœuds extraits de la FAP, d'après le lemme 1 il existe un arbre optimal tq x et y soient 2 feuilles sœurs. Par le lemme 2 l'arbre optimal, lorsque l'on réalise la fusion de x et y reste optimal. CQFD.
Terminaison : l'algorithme, faisant un nombre fini d'itérations, se termine. On peut montrer que chaque itération diminue de 1 le nombre de nœuds dans la FAP. A la fin des itérations il ne reste qu'un nœud racine de l'arbre optimal.

Codage et complexité

- $\mathcal{H}(p)$ est appelée quantité d'information
- Propriété du préfixe $\Rightarrow$ arbre (automate d'état fini)
- $\mathcal{H}(p)$ donne le taux de compression
- Borne supérieure de la complexité de Kolmogorov