Département d'informatique et de génie logiciel

Compression de données

IFT-4003/IFT-7023

Notes de cours
Mathématiques pour la compression sans perte

Édition Hiver 2012

Mohamed Haj Taieb

Local: PLT 2113

Courriel: mohamed.haj-taieb.1@ulaval.ca

Faculté des sciences et de génie
Département de génie électrique et de génie informatique

UNIVERSITÉ LAVAL

Plan

- Mathématiques pour la compression sans pertes:
  - Quelques notions de la théorie d'information
  - Modèles de Markov
  - Codes uniquement décodables
  - Code préfixes

Information propre (1)

- Soit Un événement A ayant une probabilité P(A). L'information propose associée à A est donnée par:

$$i(A) = \log_b \frac{1}{P(A)} = -\log_b P(A)$$

- Si la probabilité de l'événement A est faible (i.e. P(A)=0.1):

$$i(A) = -\log_2 0.1 = 3.3219$$

- Si la probabilité de l'événement A est élevée (i.e. P(A)=0.9):

$$i(A) = -\log_2 0.9 = 0.152$$

- Intuitivement, quant on apprend la réalisation d'un événement peu probable, on reçoit beaucoup d'information et vice versa.

Information propre (2)

- Le choix de la base logarithmique b détermine les unités de la mesure de la quantité d'information :
- b = 2 Sh (shannons) ou bit (binary digit)
- b = e logons ou nats (natural units)
- b = 10 hartleys (R.V.L. Hartley, pionnier de la théorie des communications)

- Dans la calculatrice, on a pas la fonction log dans la base 2. Il y a seulement le log dans la base e et 10. Comment faire alors pour trouver l'information en bit.

$$\log_2 x = a$$

$$2^{\log_2 x} = 2^a$$

$$x = 2^a$$

$$\ln x = \ln 2^a$$

$$\ln x = a \ln 2$$

$$a = \frac{\ln x}{\ln 2} \Rightarrow \log_2 x = \frac{\ln x}{\ln 2} = \frac{\log_{10} x}{\log_{10} 2}$$

Information propre (3)

- Quelque soit la base choisie b = 2, e ou 10, la quantité d'information demeure la même. Les facteurs de conversion entre les différentes bases sont :

$$1\,\text{logon} = \frac{1}{\ln 2} = \log_2 e = 1.443\, bit$$

$$1\,\text{hartley} = \frac{1}{\log_{10} 2} = \log_2 10 = 3.322\, bit$$

$$1\,\text{hartley} = \frac{1}{\log_{10} e} = \ln 10 = 2.303\,\text{logons}$$

Entropie du premier ordre

- Soit A_1 ,…,A_n le résultat d'une expérimentation donnée de probabilité $P(A_1)$,…,$P(A_n)$. L'entropie associé a cette expérience est donnée par:

$$H(A) = \sum_{i=1}^{n} P(A_i)\, i(A_i) = -\sum_{i=1}^{n} P(A_i) \log_b(A_i)$$

- L'entropie c'est le nombre minimal moyen de symboles binaires (b=2) nécessaire pour représenter la source.

- L'entropie c'est alors la limite théorique de compression sans perte qu'on peut atteindre: Shannon.

Exemple de calcul d'entropie

- Considérons la séquence suivante:

$$1\ 2\ 3\ 2\ 3\ 4\ 5\ 4\ 5\ 6\ 7\ 8\ 9\ 8\ 9\ 10$$

- Si l'on suppose que cette séquence représente bien les probabilités d'apparition de chaque éléments, alors on a:

$$P(1) = P(6) = P(7) = P(10) = \frac{1}{16}$$

$$P(2) = P(3) = P(4) = P(5) = P(8) = P(9) = \frac{2}{16}$$

- L'entropie est donnée par

$$H = -\sum_{i=1}^{n} P(i) \log_2 P(i) = -4\frac{1}{16}\log_2\frac{1}{16} - 6\frac{2}{16}\log_2\frac{2}{16} = 3.25 \text{ bits}$$

- [Shannon] : Le meilleur schéma de compression sans pertes pour représenter cette séquence doit utiliser en moyenne 3.25 bits/échantillon.

Exploitation de la corrélation

- Si l'on prenne en considération la corrélation entre les échantillons consécutifs, on peut coder la différence entre les symboles voisins et la source devient

1 2 3 2 3 4 5 4 5 6 7 8 9 8 9 10 =>1 1 1 -1 1 1 1 1 -1 1 1 1 1 1 -1 1 1

- Cette séquence comporte deux valeurs de probabilité:

$$P(1) = \frac{13}{16} \quad P(-1) = \frac{3}{16}$$

- L'entropie est donnée par

$$H = -\sum_{i=1}^{n} P(i)\log_2 P(i) = -\frac{13}{16}\log_2\frac{13}{16} - \frac{3}{16}\log_2\frac{3}{16} = 0.7 \text{ bits}$$

- Le modèle doit être connu au récepteur lors de la reconstruction:

$$x_n = x_{n-1} + r_n$$

Exploitation de la structure des données

- Considérons la séquence suivante:

$$1\ 2\ 1\ 2\ 3\ 3\ 3\ 3\ 1\ 2\ 3\ 3\ 3\ 3\ 1\ 2\ 3\ 3\ 1\ 2$$

- Si l'on considère chaque symbole à part on:

$$P(1) = P(2) = \frac{1}{4}, \quad P(3) = \frac{1}{2}$$

- L'entropie est de 1.5 bits/symbole.
- Nombre total des bits: 1.5x20=30 bits
- Cependant il existe une certaine structure dans la séquence. En effet il n y que deux symboles: (1 2) et (3 3).

$$P(12) = \frac{1}{2}, \quad P(33) = \frac{1}{2}$$

- L'entropie est de 1 bits/symbole.
- Nombre total des bits: 1x10=10bits

Modélisation des données

- L'utilisation d'un modèle pertinent pour les données peut réduire énormément l'entropie estimée de la source.
- Modèle physique:
  - Connaissance du processus de génération des données.
  - Exemple: compression de la voix.
- Modèle d'ignorance:
  - Indépendance entre les symboles.
  - Probabilité égale pour tout les symboles.
- Modèle de probabilité
  - Alphabet: $\mathrm{A} = \; a_1, a_2, \ldots, a_n$
  - Probabilité: $\mathrm{P} = \; p(a_1), p(a_2), \ldots, p(a_n)$
  - Indépendance entre les symboles.

Modèle de Markov

- Un symbole données dépend des k symboles précédents.
- Soit une séquence $\{x_n\}$. Cette séquence forme une chaîne de Markov discrète d'ordre k (DMC) si:

$$p(x_n | x_{n-1}, \ldots, x_{n-k}) = p(x_n | x_{n-1}, \ldots, x_{n-k}, \ldots, x_1)$$

- Le modèle de Markov le plus communément utilisé est le modèle du premier ordre:

$$p(x_n | x_{n-1}) = p(x_n | x_{n-1}, \ldots, x_{n-k}, \ldots, x_1)$$

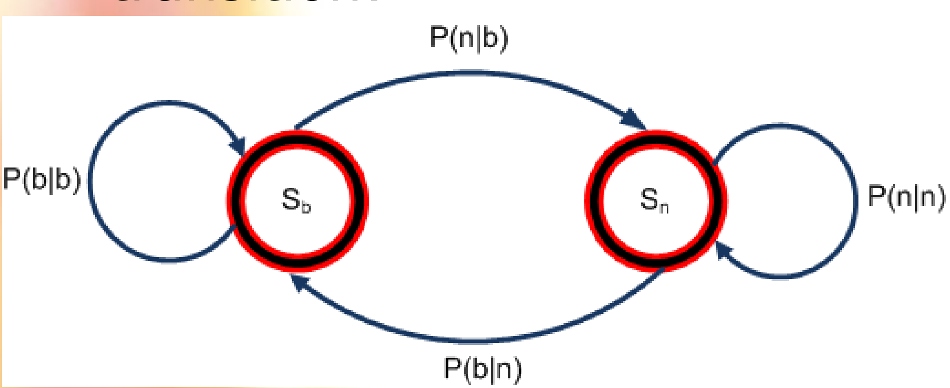
- Modèle de dépendance linéaire:
  - Modèle: $x_n = \rho x_{n-1} + \varepsilon_n$
  - ε_n: Bruit blanc
  - Exemples: compression de la voix et d'images.

Modèle de Markov non linéaire

- Modèle de Markov pour une image binaire: pixels noirs et pixels blancs.
- On définit deux état S_b et S_n
- On définit les probabilité

$$P(S_b): \text{ pixel courant blanc}$$
$$P(S_n): \text{ pixel courant noir}$$

- On définit les probabilité de transition:

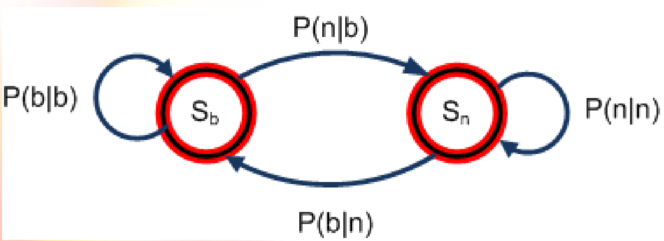


L'entropie du processus:

$$H = P(S_b)H(S_b) + P(S_n)H(S_n)$$
$$H(S_b) = -P(b|b)\log P(b|b) - P(n|b)\log P(n|b)$$
$$H(S_n) = -P(n|n)\log P(n|n) - P(b|n)\log P(b|n)$$
$$p(b|b) = 1 - P(b|n), \quad p(n|n) = 1 - P(n|b)$$

Entropie d'une image binaire
Modèle de probabilité vs Modèle de Markov

- Énoncé: Soit une image binaire tel que: $p \;\; b|b \; = 0.99, \quad p \;\; n|n \; = 0.7$
- On en déduit:



$$p(b|b) = 0.99 \rightarrow p(n|b) = 0.01$$

$$p(n|n) = 0.7 \;\; \rightarrow p(b|n) = 0.3$$

$$P(S_b) = P(S_n)P(b|n) + P(S_b)P(b|b)$$

$$P(S_b) = \;\; 1 - P(S_b) \;\; P(b|n) + P(S_b)P(b|b)$$

$$(1 + P \;\; b|n \;\; - P(b|b))P(S_b) = P(b|n)$$

$$P(S_b) = \frac{P(b|n)}{1 + P(b|n) - P(b|b)} = \frac{0.3}{1 + 0.3 - 0.99} = \frac{30}{31}$$

$$P(S_n) = \frac{1}{31}$$

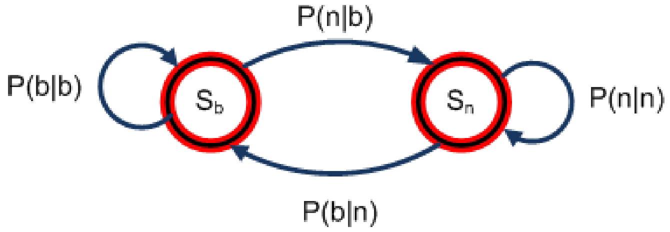
Modèle de probabilité vs Modèle de Markov

- Calcul de l'entropie en utilisant le modèle de probabilité:

$$H_{prob} = -P(S_b)\log_2 P(S_b) - P(S_n)\log_2 P(S_n)$$

$$H_{prob} = -\frac{30}{31}\log_2\frac{30}{31} - \frac{1}{31}\log_2\frac{1}{31} = 0.206 \text{ bits}$$

- Calcul de l'entropie en utilisant le modèle de Markov



$$H(S_b) = -0.01\log_2 0.01 - 0.99\log_2 0.99 = 0.081\text{bits}$$

$$H(S_n) = -0.3\log_2 0.3 - 0.7\log_2 0.7 = 0.881 \text{ bits}$$

$$H_{Markov} = P(S_b)H(S_b) + P(S_n)H(S_n)$$

$$H_{Markov} = \frac{30}{31}0.081 + \frac{1}{30}0.881 = 0.107 \text{ bits}$$

Codage: terminologie

- Codage: assignation de séquences binaires aux éléments de l’alphabet.
- Code: l’ensemble des séquences binaires.
- Taux du code: nombre de bits moyen par symbole.
- Mot-code: un élément du code.
- Alphabet : collection de symboles ou lettres.

- L’alphabet de la langue anglaise contient 26 lettres minuscules, 26 lettres majuscules et une variété de marques de ponctuation.
- Code ASCII: code à longueur fixe. Moins performant qu’un code à longueur variable.

ASCII codes for some of the keys on a keyboard.

SPACE = 00100000

A	=	01000001	a	=	01100001
B	=	01000010	b	=	01100010
C	=	01000011	c	=	01100011
D	=	01000100	d	=	01100100
E	=	01000101	e	=	01100101
F	=	01000110	f	=	01100111
etc ...			etc ...		

RETURN (end of a line) = 00001010

ASCII = American Standard Code for Information Interchange
(American — that's why ‘£’ isn't a standard character on some printers!)

Code uniquement décodable

- Le design d'un code performant doit réduire la longueur moyenne.
- Mais ce n'est pas assez! Il faut garantir que le récepteur peut décoder la séquence binaire.
- Exemple:
  - Soit l'alphabet $=\{a_1, a_2, a_3, a_4\}$
  - $P(a_1)=1/2$, $P(a_2)=1/4$, $P(a_3)=P(a_4)=1/8$,
  - $H=1.75$ bits
  - Longueur moyenne:

$$l = \sum_{i=1}^{4} p(a_i) n(a_i)$$

  - $n(a_i)$: longueur du mot code

Code uniquement décodable

- Considérons les 4 mots code suivants:

Lettres	Probabilité	Code 1	Code 2	Code 3	Code 4
a_1	0.5	0	0	0	0
a_2	0.25	0	1	10	01
a_3	0.125	1	00	110	011
a_4	0.125	10	11	111	0111
	Longueur moyenne	1.125	1.25	1.75	1.875

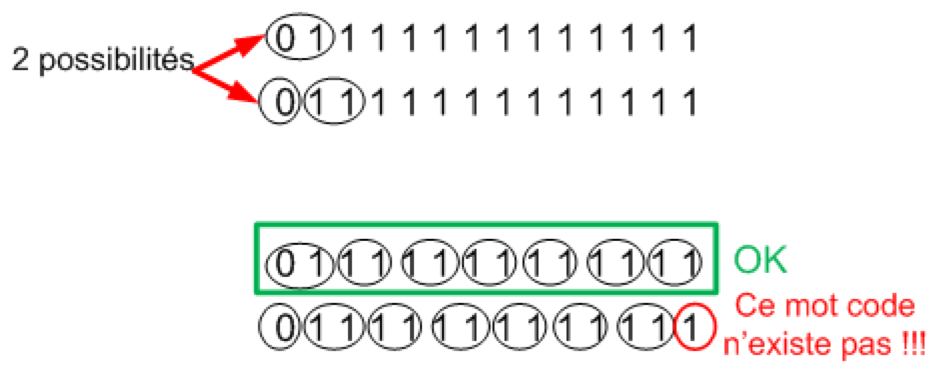
- Code 1: a_1 et a_2 sont 2 mots code identiques => ambigüités quant on reçoit 0.
- Code 2: code unique mais ambigus quant on reçoit 00 => 00= a_3 ou 00= a_1 a_1
- Code 3: uniquement décodable et instantané => la fin du mot code est détecter juste à la réception du dernier bit.
- Code 4: uniquement décodable mais quasi-instantanée => quant on reçoit 011, on ne peut affirmer que c'est a_3 qu'après la réception du prochain bit.

Code instantané vs quasi instantané

- Considérons le code suivant:

Lettre	Mot code
a_1	0
a_2	01
a_3	11

- Décodage:



- Il faut attendre la réception de toute la séquences binaire pour pouvoir décoder.

Test de décodabilité unique

- Quelques définitions: Soit a= 010

  b= 01011

  ==> a est un préfixe de b et le *dangling suffix* est 11

- Algorithme de test de décodabilité unique
  1. Construction d'une liste de tout les mot-codes
  2. Détecter un mot code est un préfixe d'un autre mot code
  3. Rajouter le *dangling suffix* à la liste [à moins qu'il ne soit précédemment ajouté]
  4. Itérer les étapes 2 et 3 en utilisant la nouvelle liste contenant le nouveau *dangling suffix*

  Conditions d'arrêt:
  1. On tombe sur un *dangling suffix* qui est un mot-code. NOK
  2. Il n y a plus de *dangling suffixes*

Application de l'algorithme

- Exemple 1: code=0, 01, 11
  - Étape 1: {0, 01, 11}
  - Étape 2: 0 est un suffixe de 01 et le *dangling suffix* est 1
  - Étape 3: Nouvelle liste= {0, 01, 11, 1}
  - Étape 2: 1 est un suffixe de 11 et le dangling suffix est 1, mais il existe déjà dans la liste.
  - Condition d'arrêt: Il n y a plus de dangling suffixes

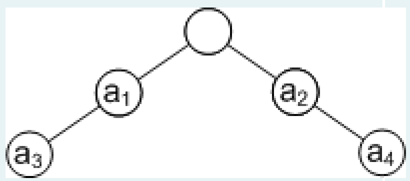
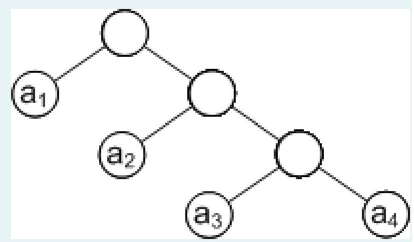
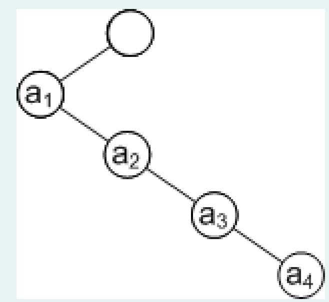
- Exemple 2: code=0, 01, 10,1
- Exemple 3: code=0, 01, 110,111

Code préfixe

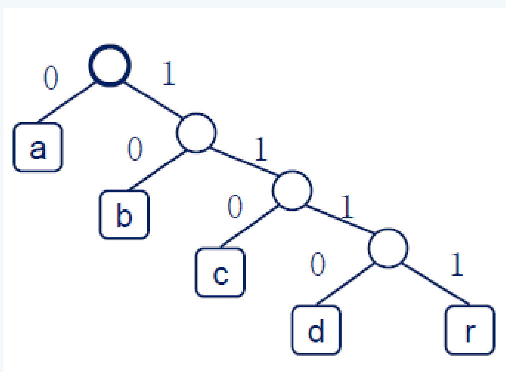
- Pour s'assurer que le test de décodabilité unique soit positif, on peut considérer un code où aucun mot-code n'est préfixe d'un autre. Un tel code est un code préfixe.
- Ainsi on ne peut trouver aucun *dangling suffix* qui est un mot code et l'algorithme du test de décodabilité ne fait aucune itération et se heurte directement à la condition 2éme d'arrêt.

- Pour vérifier si un code est un code préfixe il suffit simplement de tracer l'arbre binaire comme suit:
  - On commence par un nœud racine.
  - Chaque nœud a au maximum 2 branches.
  - Convention: la branche à gauche=0 et la branche droite=1.
  - Code préfixe: tout les mot-codes sont associés à des nœuds externes ou feuilles.

Vérification d'un code préfixe

Lettres	Code 2	Code 3	Code 4
a_1	0	0	0
a_2	1	10	01
a_3	00	110	011
a_4	11	111	0111
Arbre binaire			
Remarque	Code 2 n'est pas un code préfixe car il y a des mot-codes (a_1 et a_2) associés à des nœuds internes	Code 3 est un code préfixe tout les mot-codes sont associés à des nœuds internes	Code 4 n'est pas un code préfixe car il y a des mot-codes (a_1, a_2 et a_3) associés à des nœuds internes

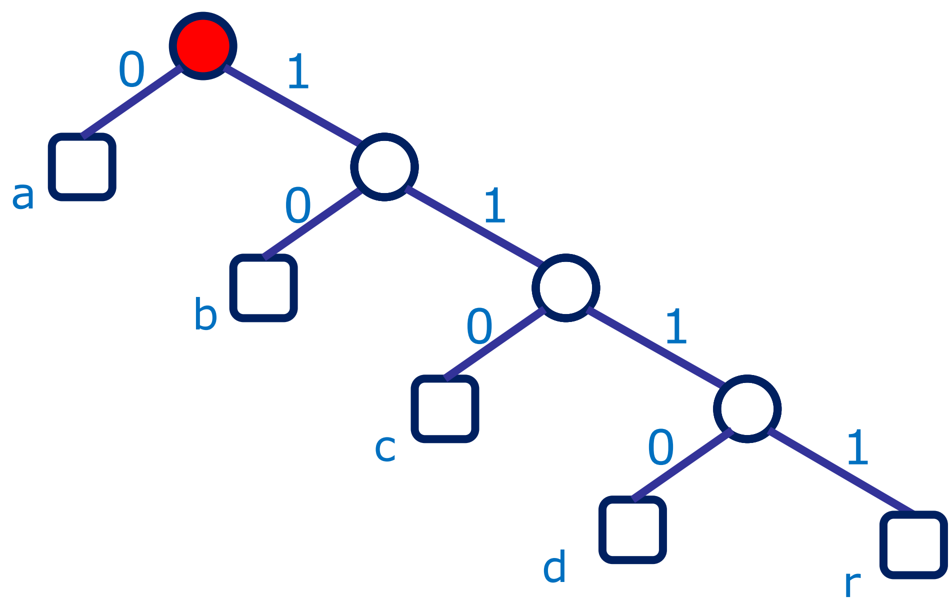
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (1):

Reçu : 01011110110011100101110

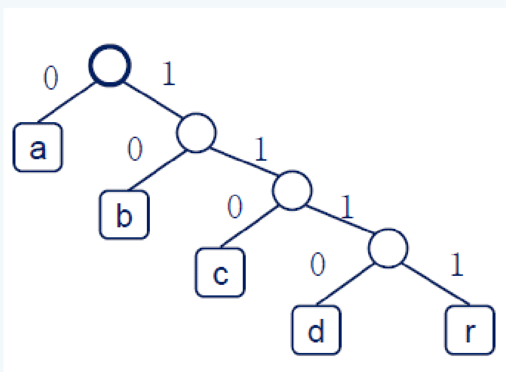
Décodé : -----------



Séquence envoyée:

abracadabra = 01011110110011100101110

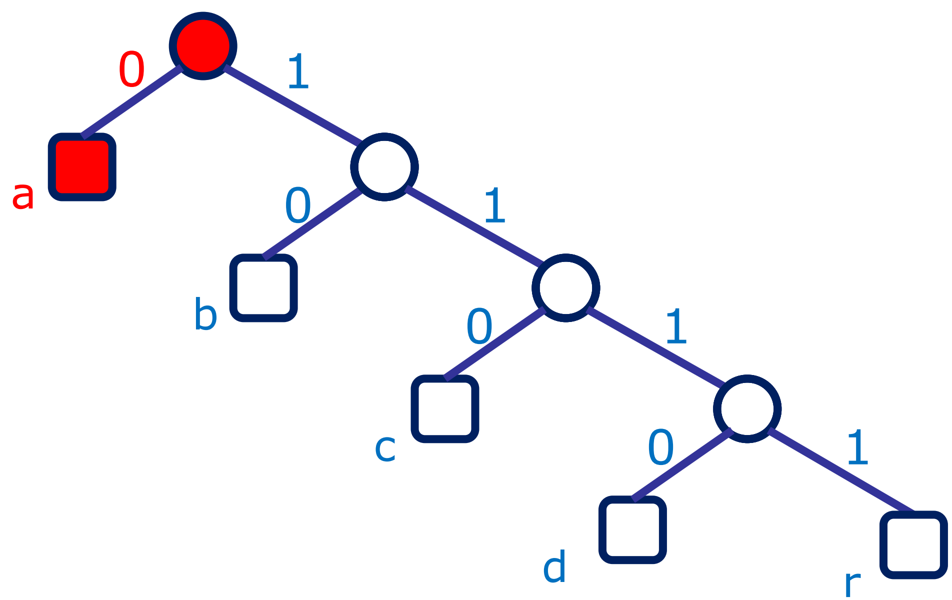
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (2):

Reçu : 010111101100111001011110

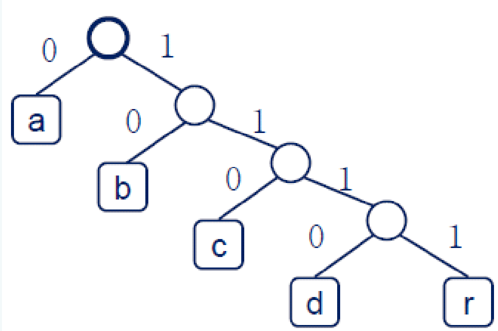
Décodé : a----------



Séquence envoyée:

abracadabra = 010111101100111001011110

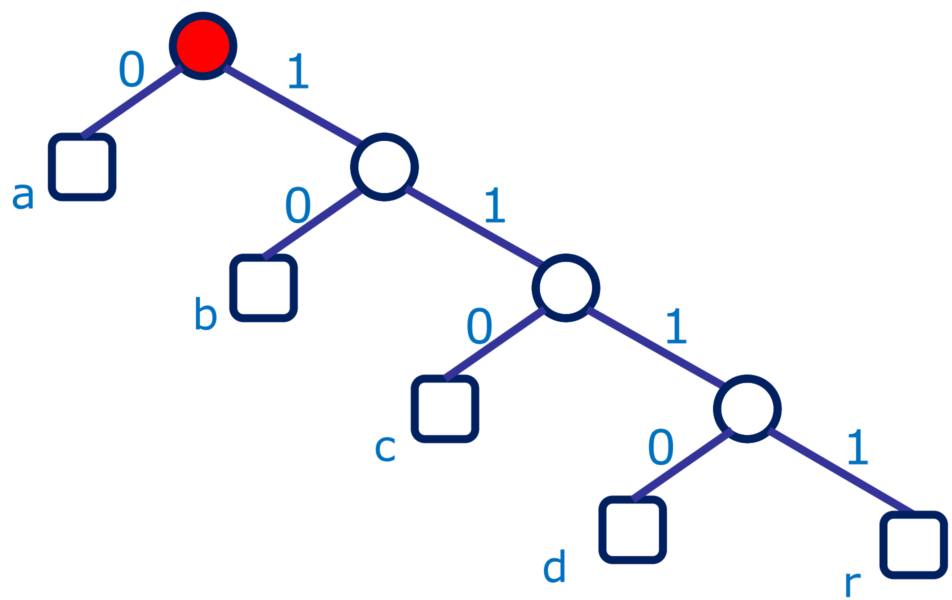
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (3):

Reçu : 010111101100111001011110

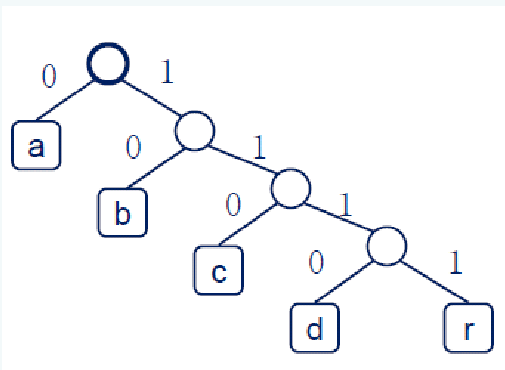
Décodé : a----------



Séquence envoyée:

abracadabra = 010111101100111001011110

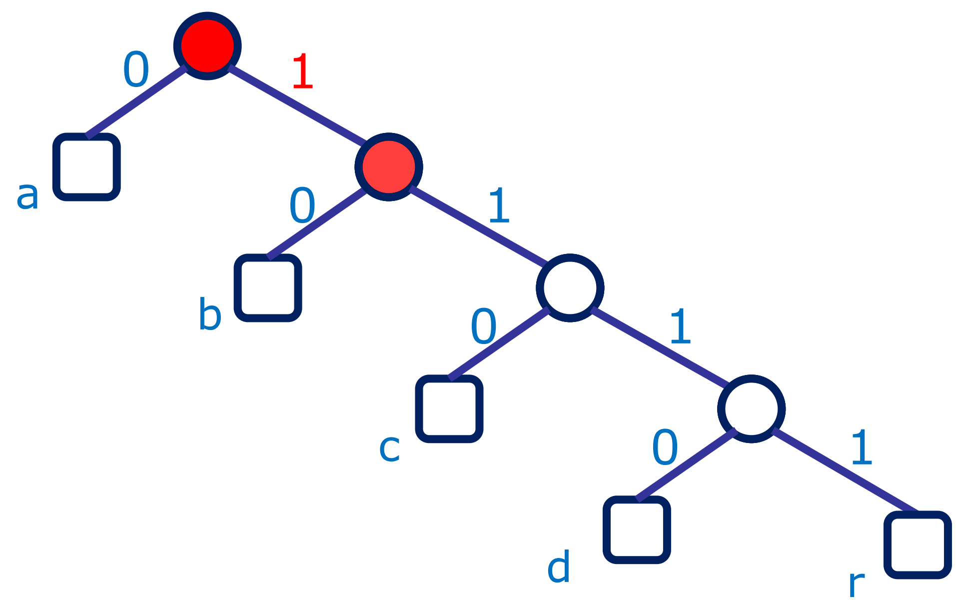
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (4):

Reçu : 010111101100111001011110

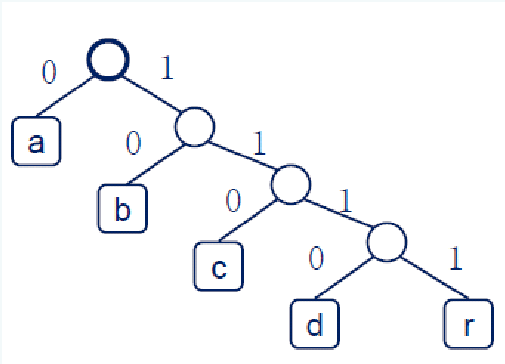
Décodé : a----------



Séquence envoyée:

abracadabra = 010111101100111001011110

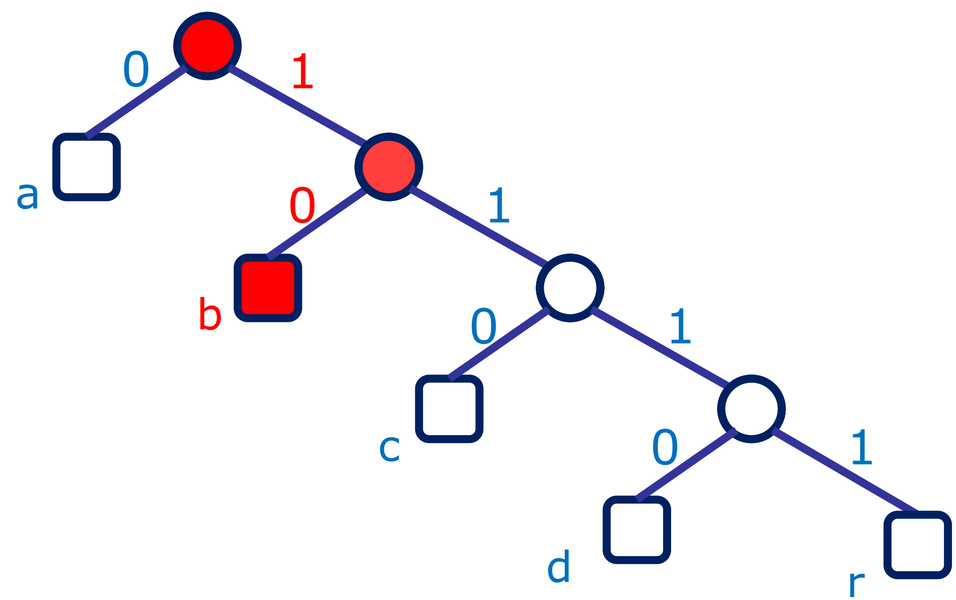
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (5):

Reçu : 01011110110011100101111 0

Décodé : ab----------



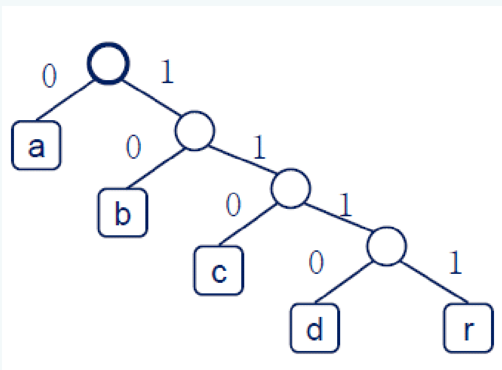
Séquence envoyée:

abracadabra = 01011110110011100101111 0

Mohamed Haj Taieb, Département de génie électrique et de génie informatique

Décodage d'un code préfixe

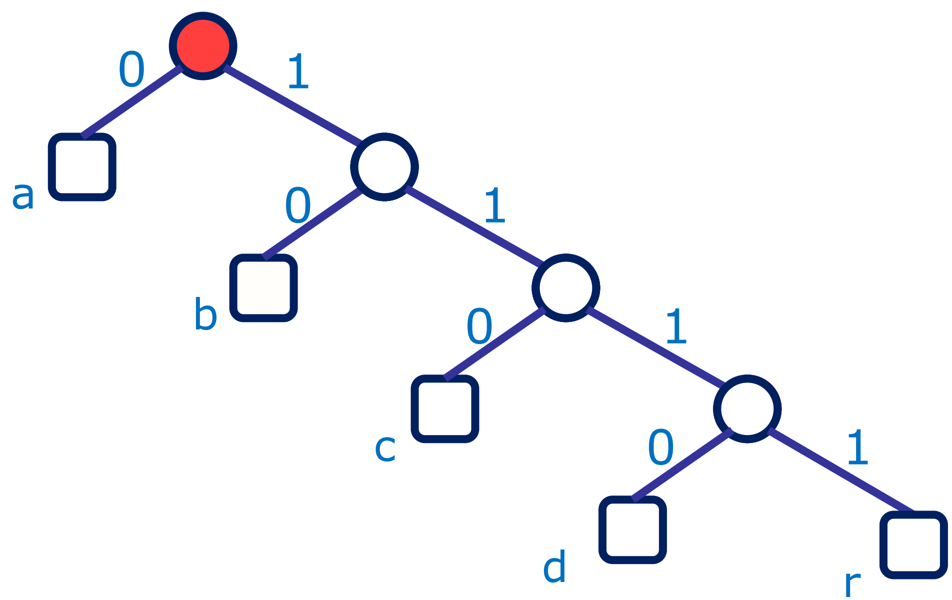
Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	



Processus de décodage (6):

Reçu : 010111101100111001011110

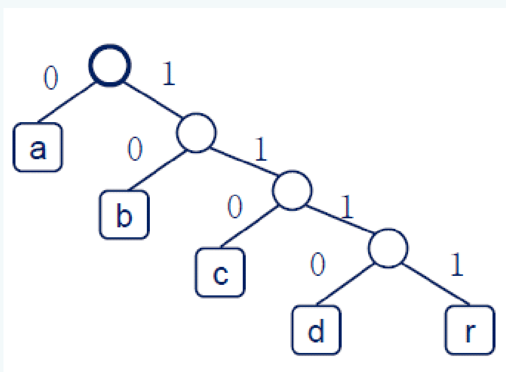
Décodé : ab----------



Séquence envoyée:

abracadabra = 010111101100111001011110

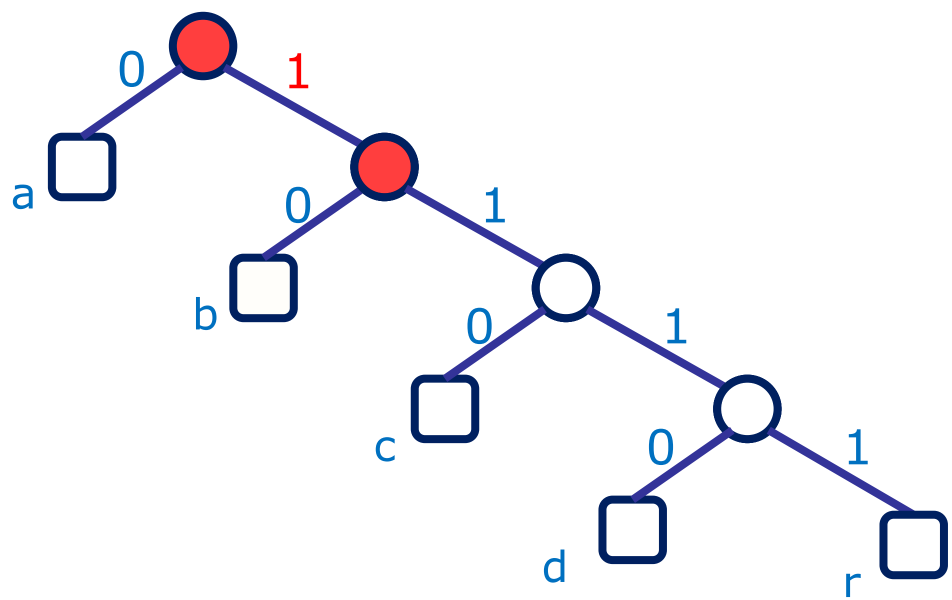
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (7):

Reçu : 010111101100111001011110

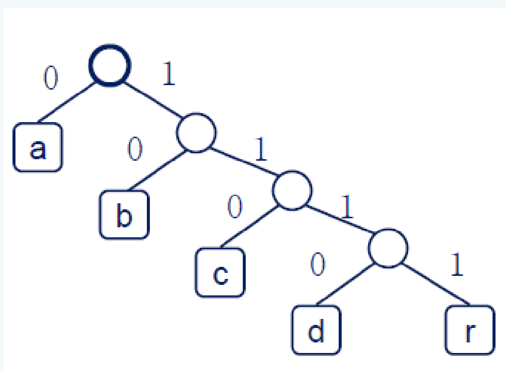
Décodé : ab----------



Séquence envoyée:

abracadabra = 010111101100111001011110

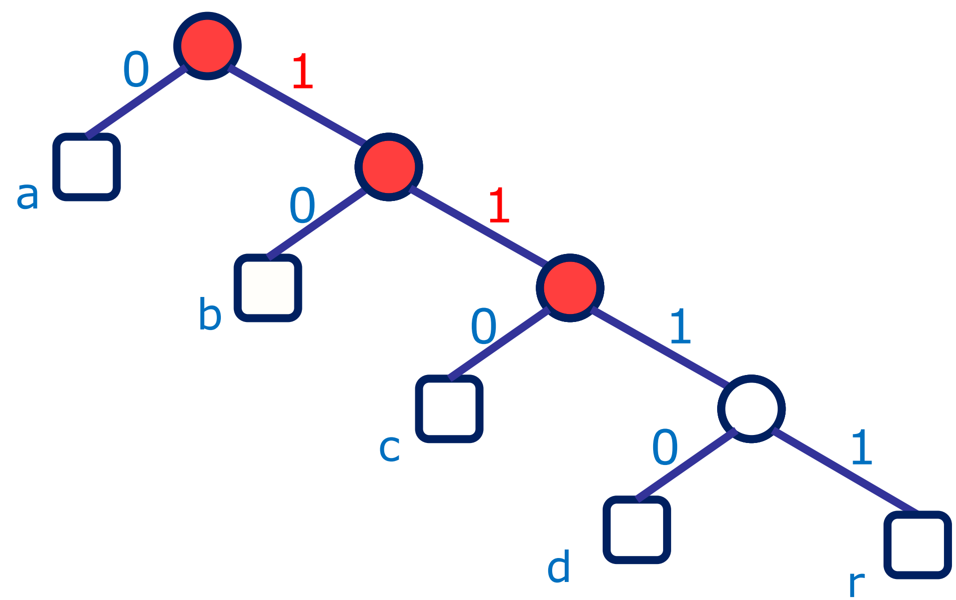
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (8):

Reçu : 010111101100111001011110

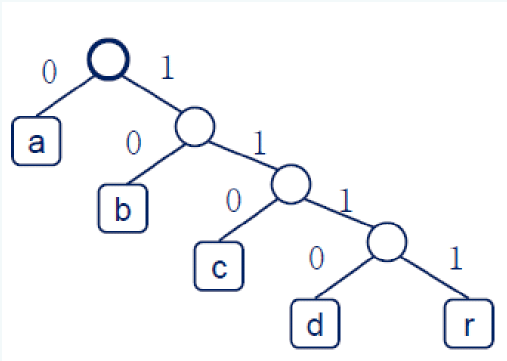
Décodé : ab----------



Séquence envoyée:

abracadabra = 010111101100111001011110

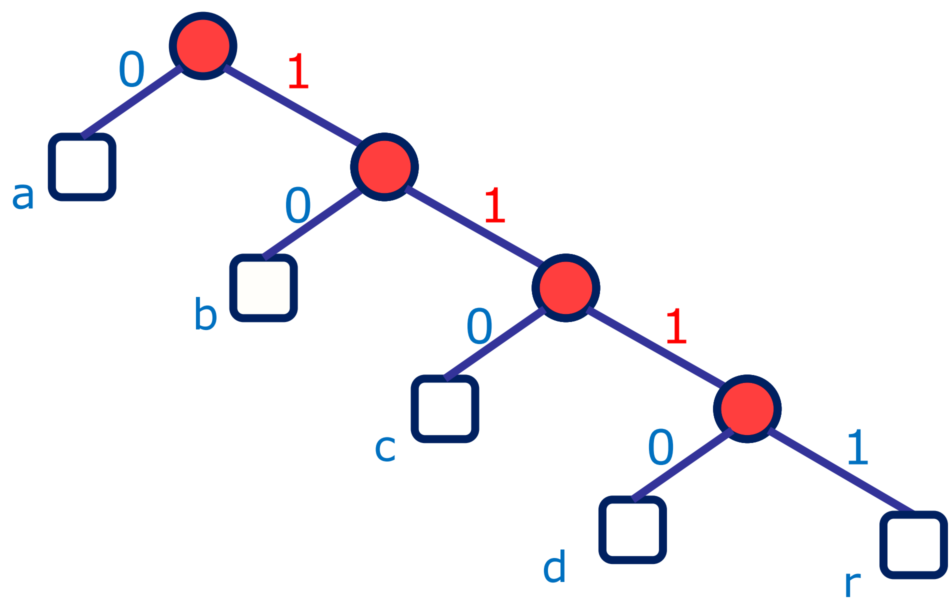
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (9):

Reçu : 01011110110011100101110

Décodé : ab---------

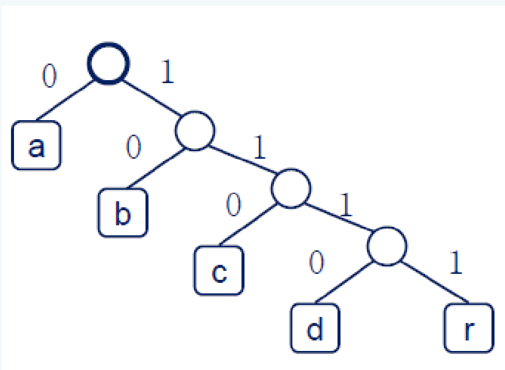


Séquence envoyée:

abracadabra = 01011110110011100101110

Mohamed Haj Taieb, Département de génie électrique et de génie informatique

Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

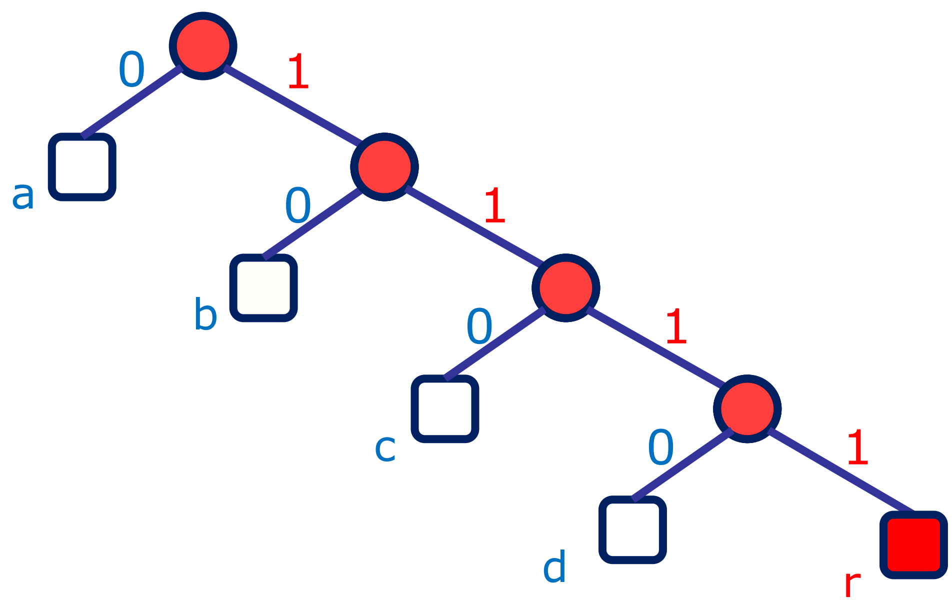
Séquence envoyée:

abracadabra = 01011110110011100101110

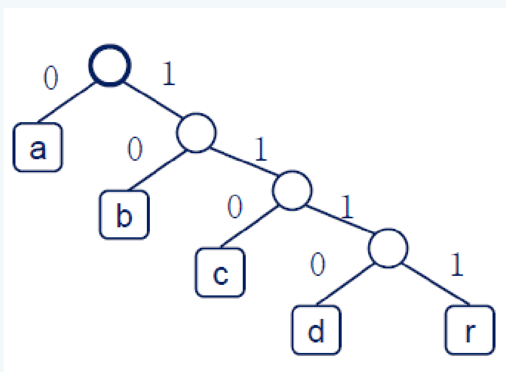
Processus de décodage (10):

Reçu : 01011110110011100101110

Décodé : abr---------



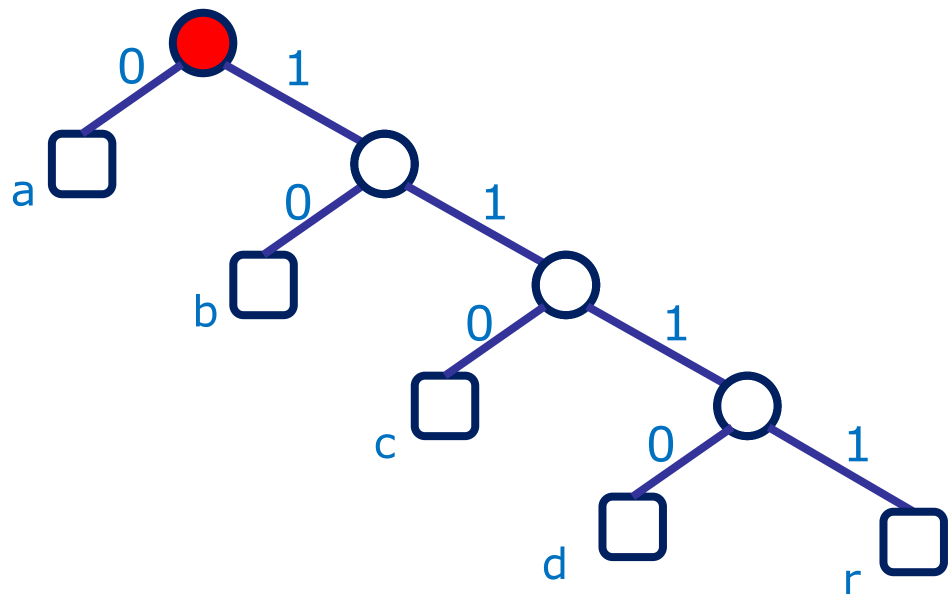
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (11):

Reçu : 0101111011001110010111110

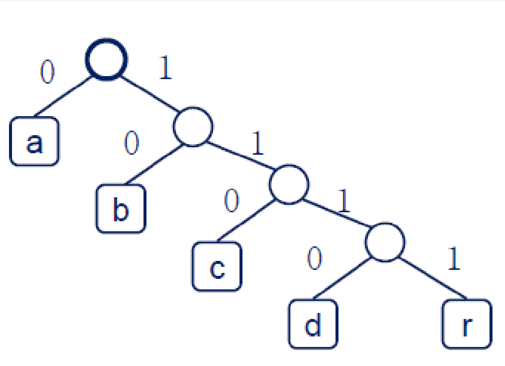
Décodé : abr---------



Séquence envoyée:

abracadabra = 0101111011001110010111110

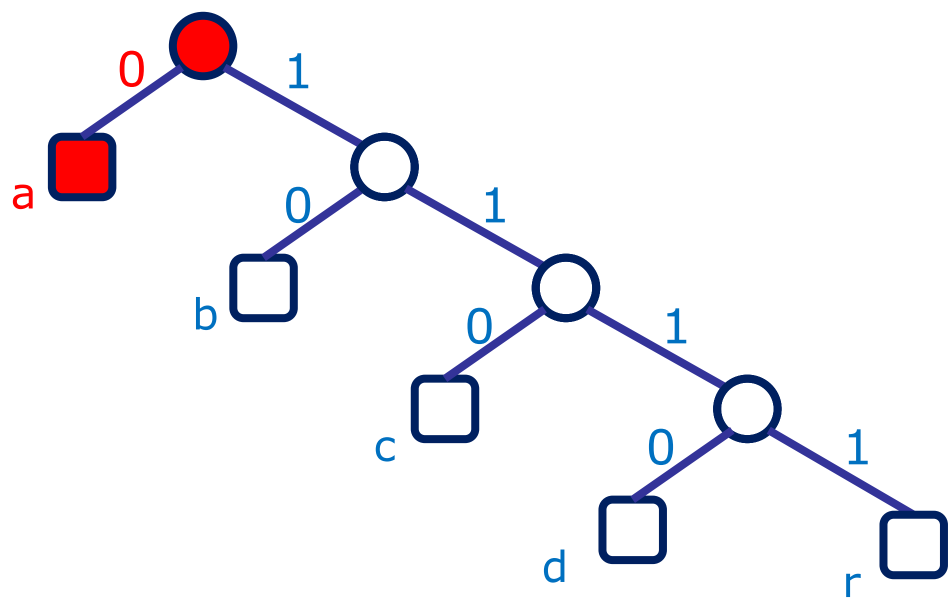
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (12):

Reçu : 010111101100111001011110

Décodé : abra--------

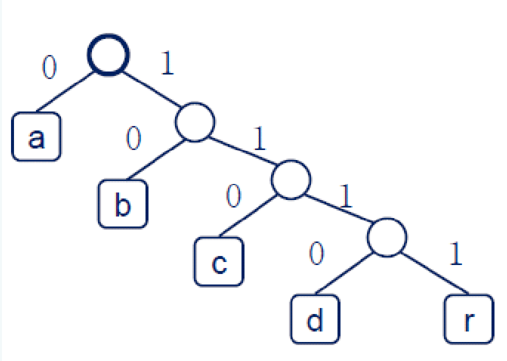


Séquence envoyée:

abracadabra = 010111101100111001011110

Mohamed Haj Taieb, Département de génie électrique et de génie informatique

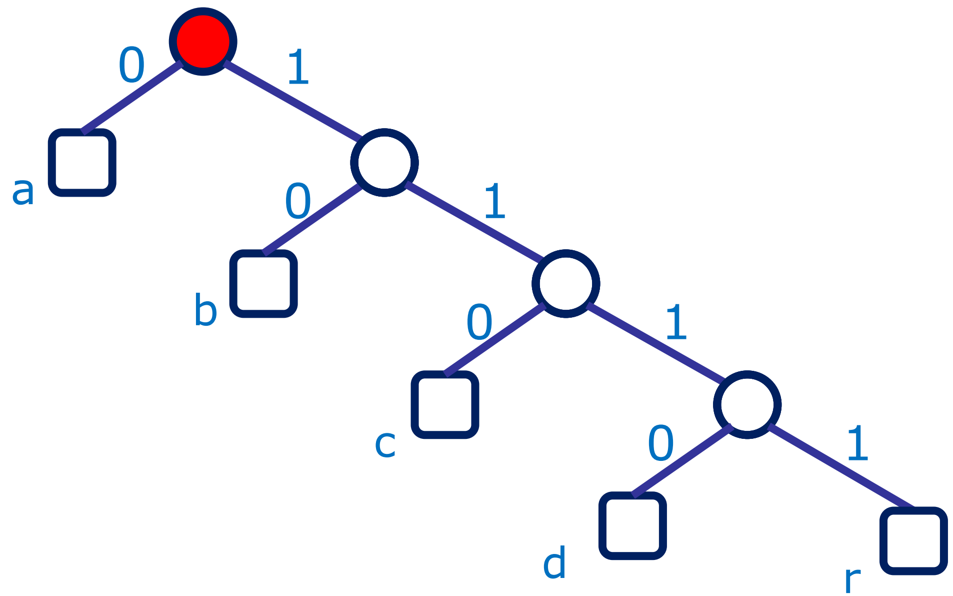
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (13):

Reçu : 0101111011001110010111110

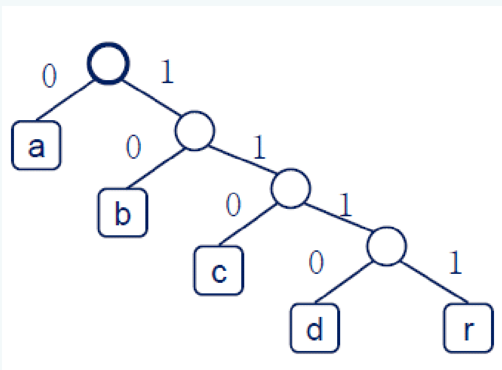
Décodé : abra-------



Séquence envoyée:

abracadabra = 0101111011001110010111110

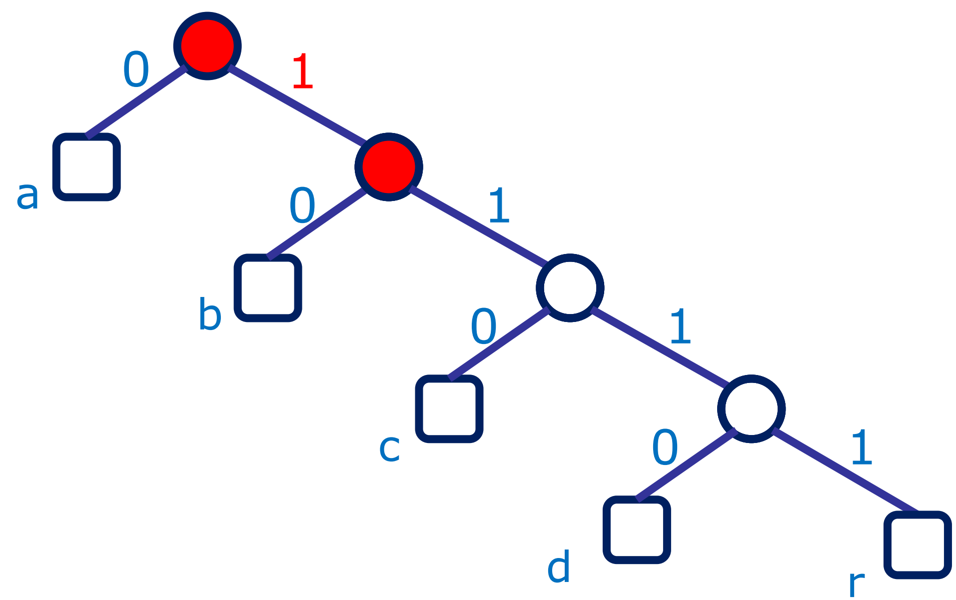
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (14):

Reçu : 01011110110011100101110

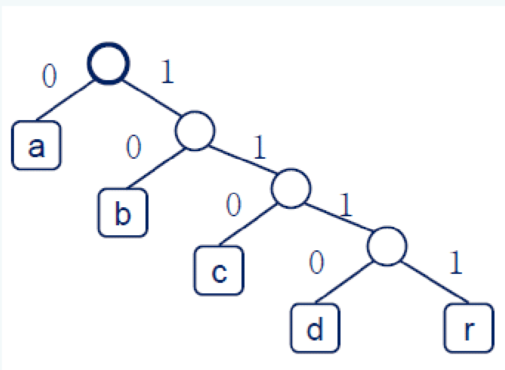
Décodé : abra-------



Séquence envoyée:

abracadabra = 01011110110011100101110

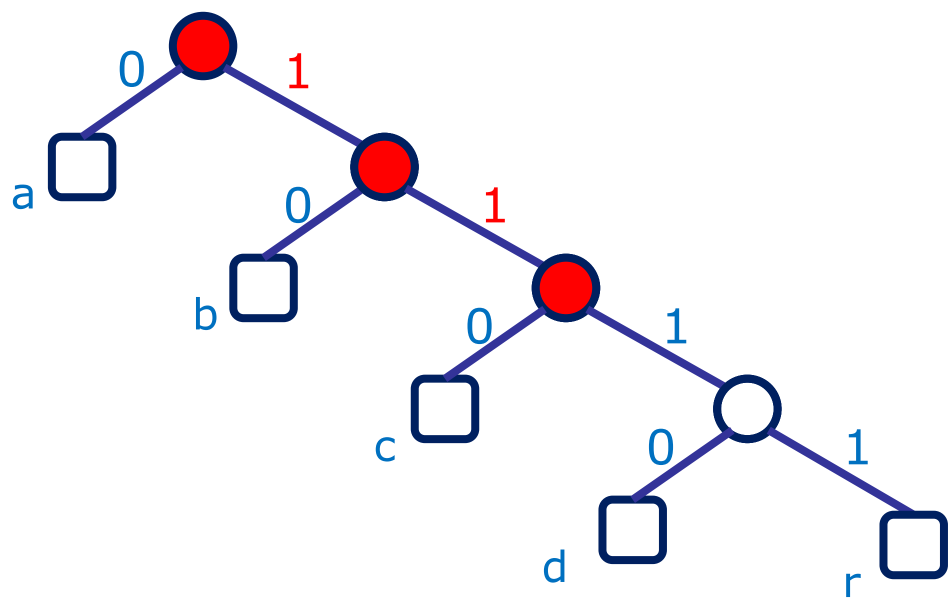
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (15):

Reçu : 01011110110011100101110

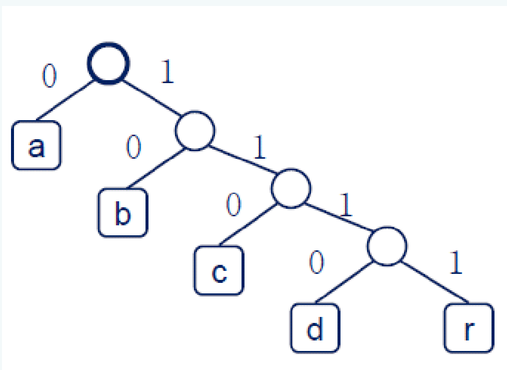
Décodé : abra-------



Séquence envoyée:

abracadabra = 01011110110011100101110

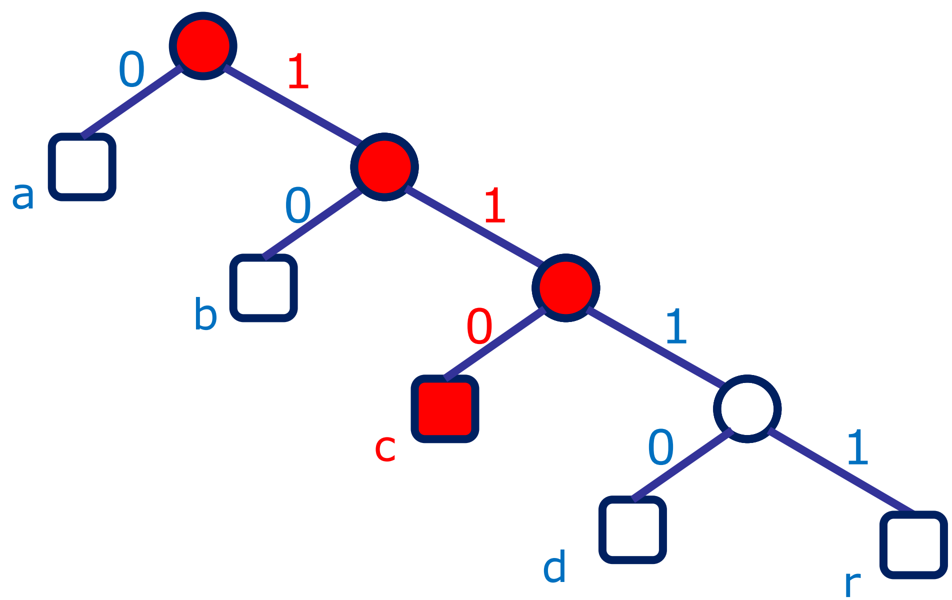
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (16):

Reçu : 010111101100111001011110

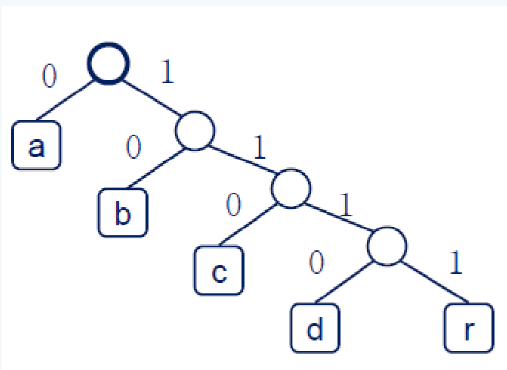
Décodé : abrac------



Séquence envoyée:

abracadabra = 010111101100111001011110

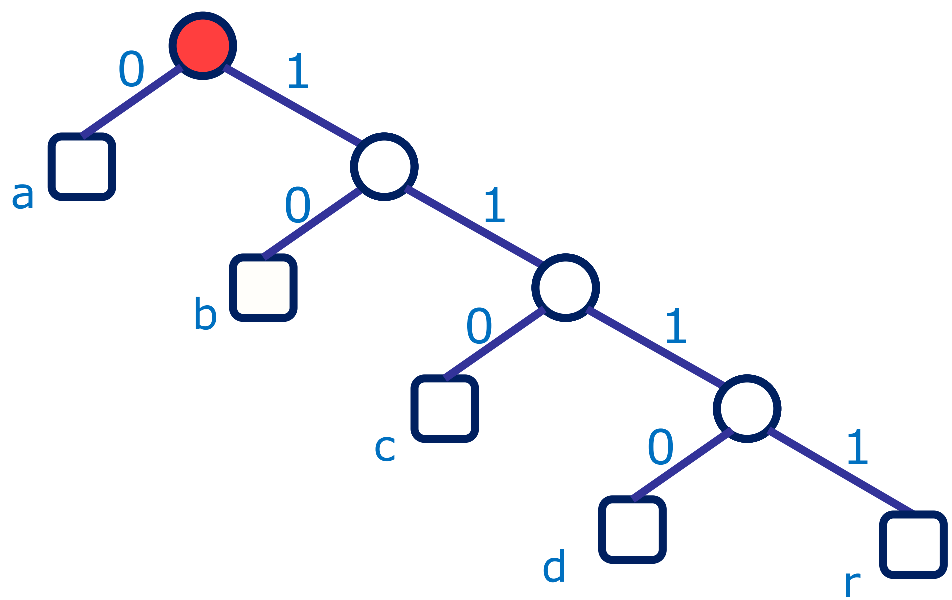
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (17):

Reçu : 010111101100111001011110

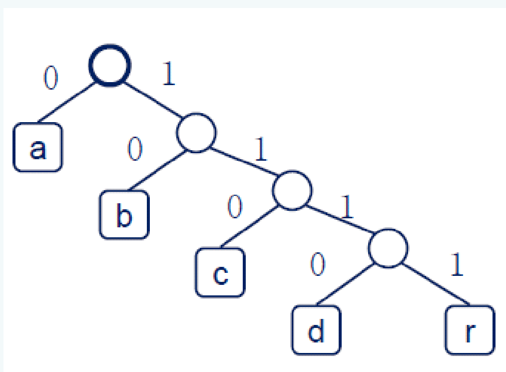
Décodé : abrac------



Séquence envoyée:

abracadabra = 010111101100111001011110

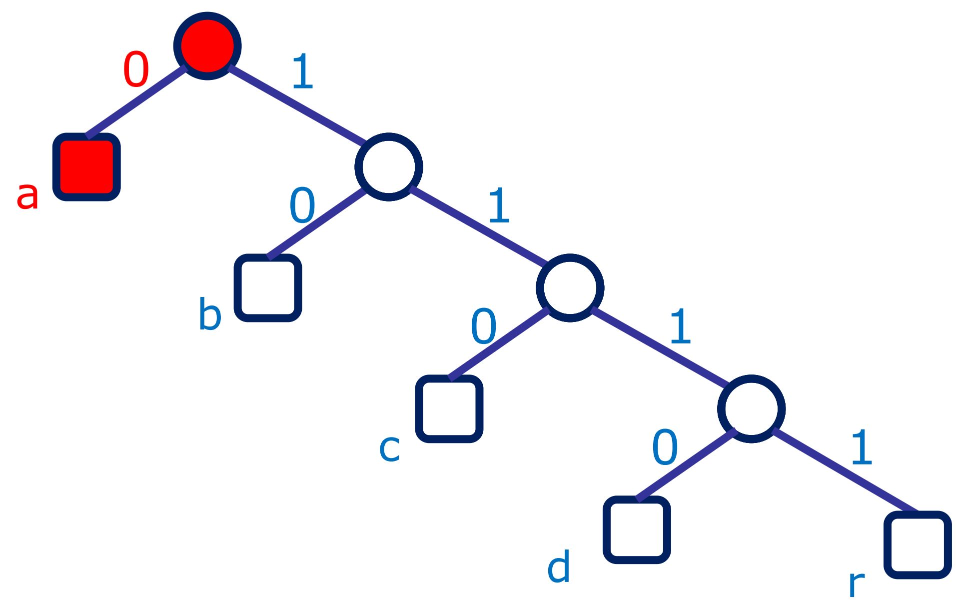
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (18):

Reçu : 010111101100111001011110

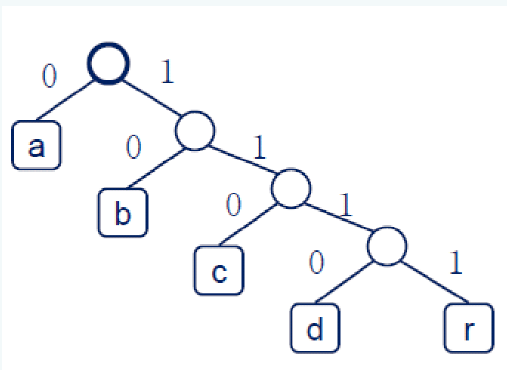
Décodé : abraca-----



Séquence envoyée:

abracadabra = 010111101100111001011110

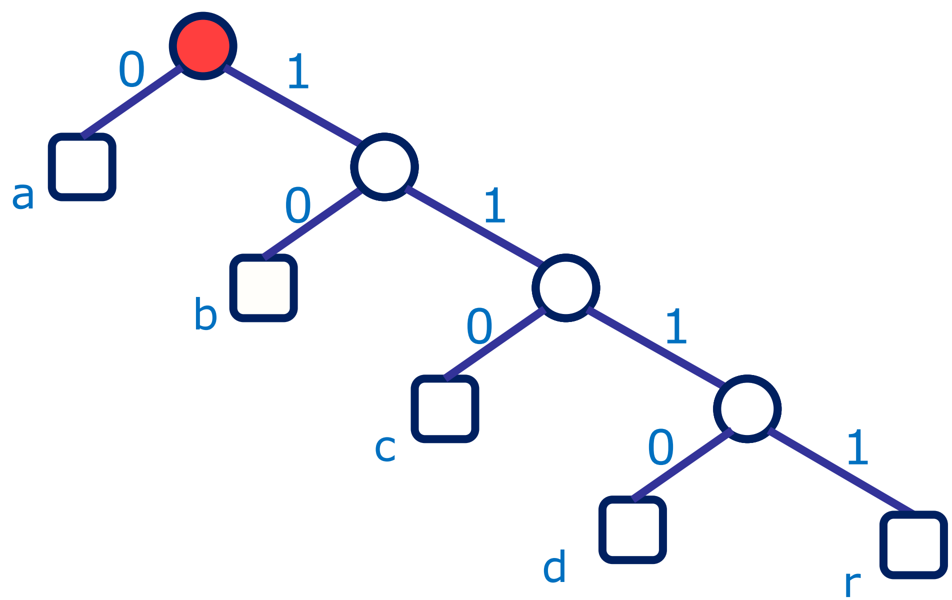
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (19):

Reçu : 010111101100111001011110

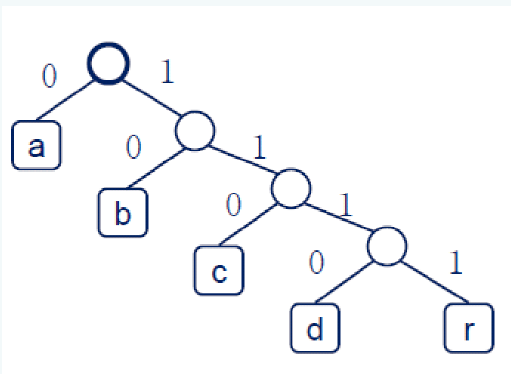
Décodé : abraca-----



Séquence envoyée:

abracadabra = 010111101100111001011110

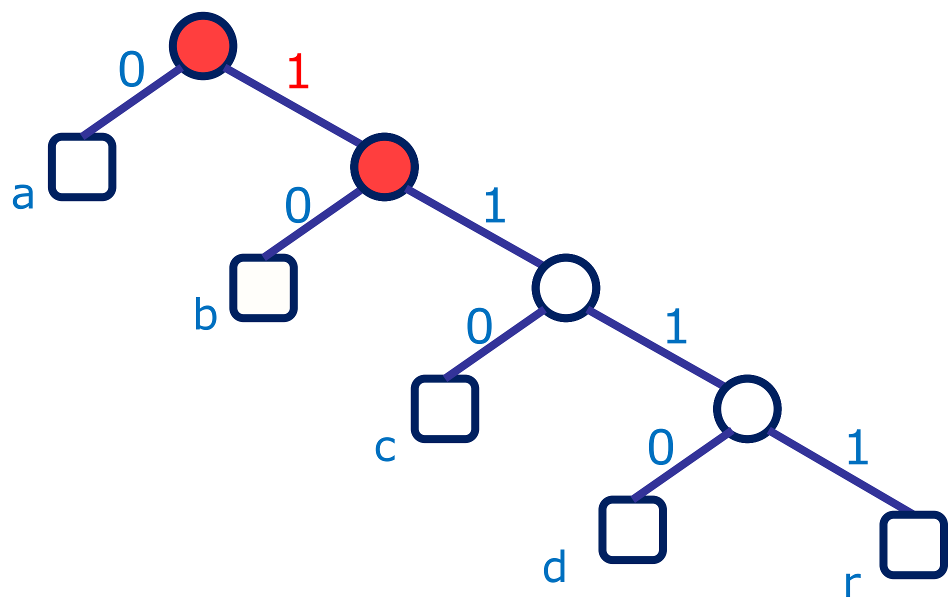
Décodage d’un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (20):

Reçu : 010111101100111001011110

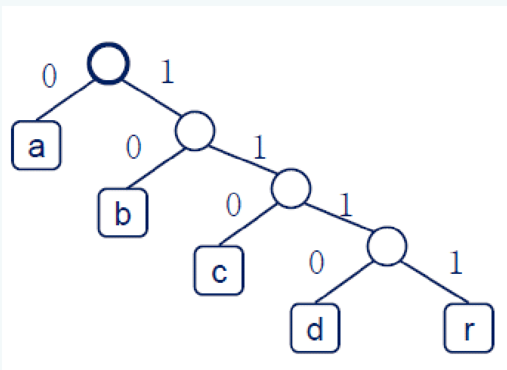
Décodé : abraca-----



Séquence envoyée:

abracadabra = 010111101100111001011110

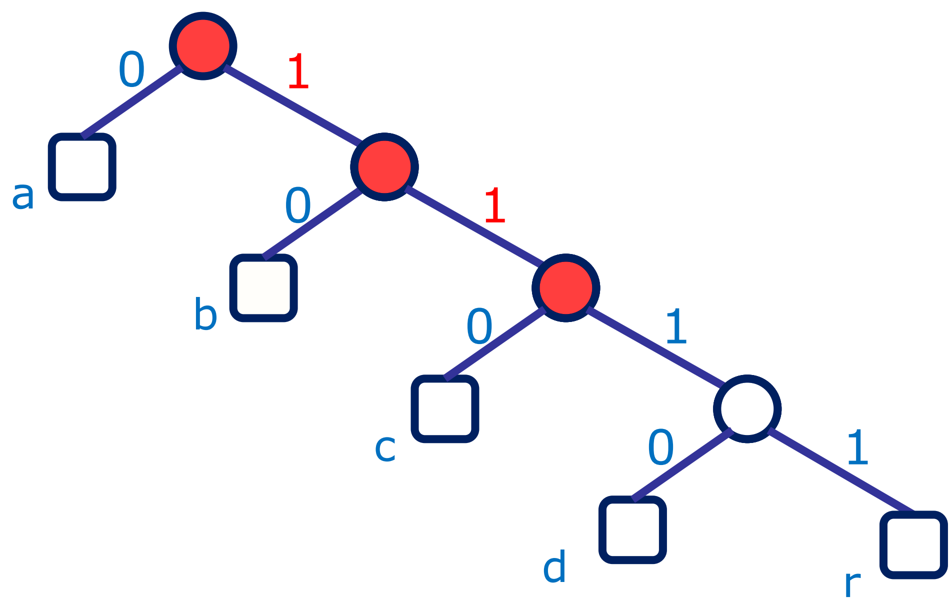
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (21):

Reçu : 0101111011001110010111110

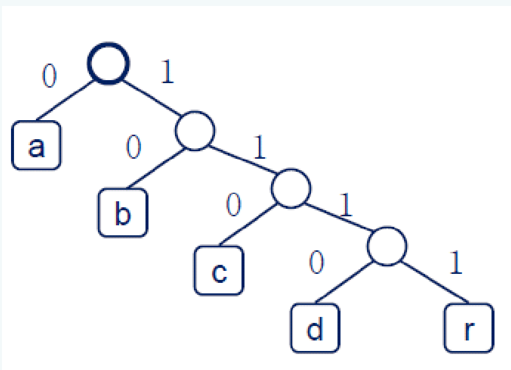
Décodé : abraca-----



Séquence envoyée:

abracadabra = 0101111011001110010111110

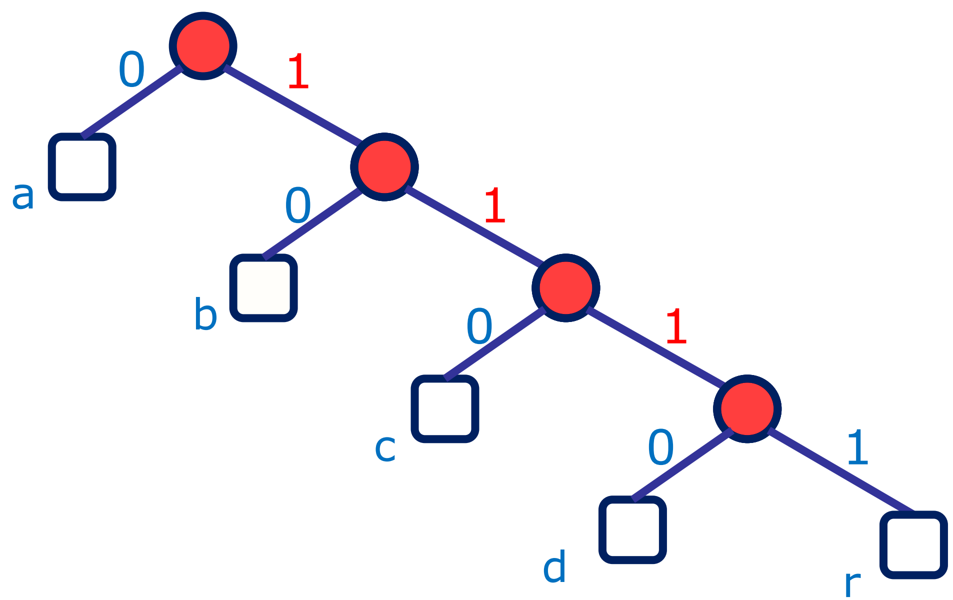
Décodage d'un code préfixe

Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (22):

Reçu : 010111101100111001011110

Décodé : abraca-----

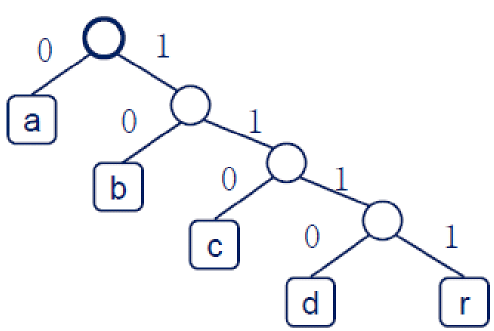


Séquence envoyée:

abracadabra = 010111101100111001011110

Mohamed Haj Taieb, Département de génie électrique et de génie informatique

Décodage d'un code préfixe

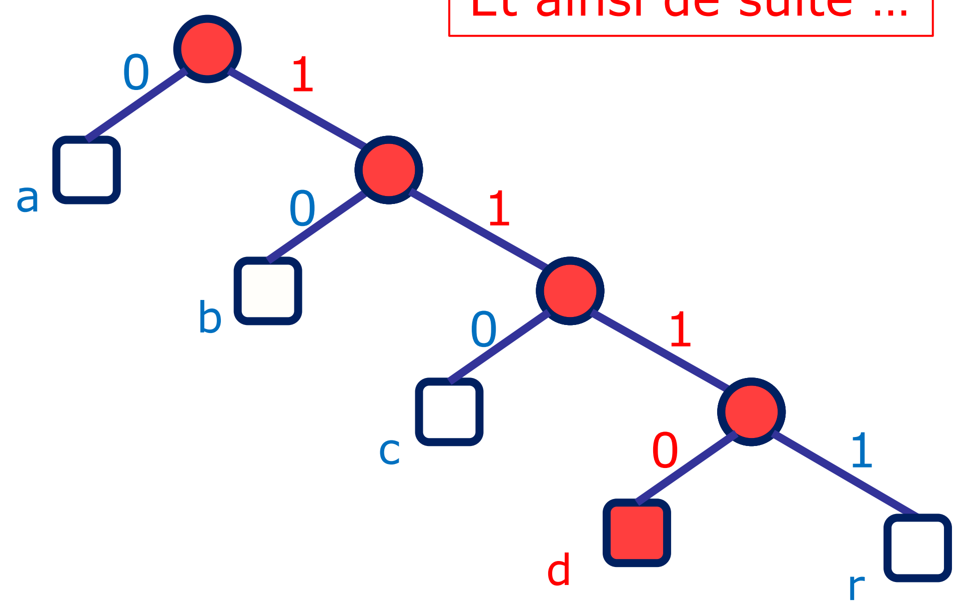
Lettres	Code
a	0
b	10
c	110
d	1110
r	1111
Arbre binaire	

Processus de décodage (23):

Reçu : 01011110110011100101110

Décodé : abracad----

Et ainsi de suite …



Séquence envoyée:

abracadabra = 01011110110011100101110