CODAGES ENTROPIQUES

INTRODUCTION

Codage entropique : appelé également codage statistique à longueur variable, fait partie des méthodes de codage de source sans pertes.

Son objectif principal, comme toutes les techniques de compression avec/sans pertes, est de modifier les données issues d'une source (capteur) pour réduire leurs taille en vue d'une transmission sur un canal de communication ou tout simplement un stockage sur une unité de mémoire (disque dur, DVD, mémoires EEPROM …etc).

Les principaux types de codage entropique sont :

- le codage de Huffman
- le codage arithmétique.
- le codage Lempel-Ziv

DEFINITION

Le codage entropique est basé essentiellement sur des statistiques liées aux données issues de la source. L'objectif est d'arriver à construire et remplacer ces données sources, par un code de longueur plus petite mais généralement variable.

Pourquoi des codes de tailles variables ?

L'idée est simple : le symboles (ou données) de source n'ont pas la même importance (ne contiennent pas la même quantité d'information).

En effet, le codage entropique tente généralement à attribuer les mots de codes les plus courts aux symboles de source les plus fréquents (contiennent moins d'informations et bien sûr les codes les plus longs aux symboles source les plus rares (contiennent plus d'information). Ainsi, nous aurons comme principe:

- ❑ attribuer des mots de code courts pour les symboles probables
- ❑ attribuer des mots de code longs pour les symboles peu probables

DEFINITION

Le codage entropique est basé essentiellement sur les concepts de la théorie de l'information, où l'information à coder (donnée source) est considérée comme une variable aléatoire discrète (appartenant à un ensemble fini).

Le théorème du codage de source, établit l'entropie comme la limite de la possibilité de compression.

DEFINITION

une source est représentée habituellement par :

1. l'ensemble des données ou de de symboles qu'elle peut fournir, appelé souvent un alphabet. On peut les noter par exemple $\{m_1, m_2, ..., m_k\}$

2. La loi de probabilité qui contrôle les caractéristiques d'émission de ces données que l'on notera $\{P(m_1), P(m_2), ..., P(m_k)\}$

Si les symboles émis par cette source sont indépendants et régis par la même loi de probabilité alors la source est dite simple ou sans mémoire

DEFINITION

Exemple 1: Soit une source qui va émettre le mot suivant:

Alphabet Source = le mot ‘’annaba”={a, n, n, a, b, a}

Les symboles générés par la source sont formés par 3 lettres de l’alphabet français à savoir a, n et b, mais pas avec la même loi de probabilité, si on se restreint à cet exemple basic.

La loi de probabilité est donc (pour ce cas de figure) :

P(‘’a”)=1/2, P(‘’n”)=1/3 et P(‘’b”)=1/6

Commet allons nous coder chacun de ces symboles?

- Est-ce qu’il est préférable d’utiliser un code de taille fixe ? Comme le code ASCII
- Ou un code de taille variable ?

Pour répondre à ces questions, nous devons d’abord rappeler les fondements de base de la théorie de l’information.

DEFINITION

Exemple 2:

Soit un bloc d'images en niveaux de gris de taille 8 × 8 pixels

10	**10**	**10**	**15**	**200**	**200**	**200**	**200**
15	5	255	255	200	200	200	200
10	15	10	5	200	200	200	200
10	10	10	15	200	200	200	200
15	5	255	255	100	200	100	200
10	15	10	5	100	200	100	200
20	30	30	30	100	200	100	200
20	150	20	20	100	200	100	200

- Ces niveaux de gris sont normalement compris entre 0 et 255, ce qui justifie le choix habituel d'un codage de taille fixe de 8 bits par pixel.
- Mais dans ce cas de figure est il judicieux de garder ce type de codage à taille fixe de 8 bits ou bien doit on le changer ?
- Surtout, que les niveaux de pixels de ce bloc n'ont pas la même probabilité d'apparition. Pour vérifier calculez la loi de probabilité de ces niveaux de gris.

ENTROPIE D'UNE SOURCE

Comme nous venons de le voir, chaque symbole de l'alphabet M={m_1, m_2,....., m_K}, de longueur l_k fixe ou variable, d'une source est régit par une probabilité d'apparition {$P(m_1)$, $P(m_2)$,....., $P(m_k)$}. Ceci, nous permet donc de définir les concepts de base de la théorie d'information à savoir :

- L'information associée à chaque symbole de la source est :

$$I[m_k] = -\log_2 P(m_k)$$

- L'entropie, qui représente la quantité d'information moyenne, d'une source M est donnée par:

$$H(M) = E[I(m_k)] = -\sum_k P(m_k) \times \log_2\big(P(m_k)\big)$$

Comme nous pouvons le remarquer, cette quantité moyenne d'une source, qui est l'entropie, dépend essentiellement de la loi de probabilité P.

ENTROPIE D'UNE SOURCE

- L'entropie, qui représentée la quantité d'information moyenne, d'une source M est donnée par:

$$H(M) = -\sum_k P(m_k) \times \log_2(P(m_k))$$

L'entropie d'une source représente donc la quantité d'information moyenne par symbole de l'alphabet délivrée par cette source. Elle représente aussi une incertitude moyenne par symbole. Elle peut exprimée en bit par symbole, c'est le cas de l'expression ci-dessus, si nous travaillons en base 2 ($\log_2$).

Plus l'entropie de la source est grande, plus il y'a de l'information délivrée par la source et bien entendu l'incertitude est plus élevée.

On montre que pour une loi de probabilité équiprobable ($\forall m$, $P(m_k)=p$, et $\sum p=1$) nous aurons une entropie maximale égale à :

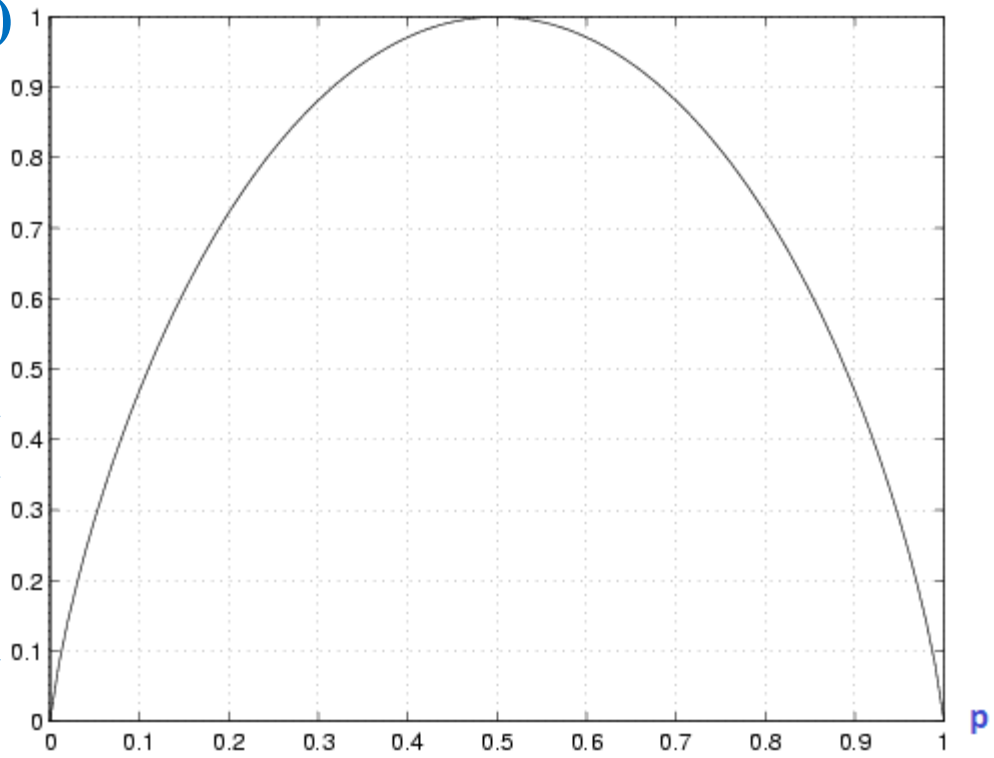
$$H(M) = -\log_2(P(m_k)) = -\log_2(p)$$

ENTROPIE D'UNE SOURCE

Exemple 1 : Cas d'une source binaire (alphabet = 0 ou 1) avec : (P(0)=p et P(1)=1-p), alors cette entropie sera:

$$H(M) = -p\log_2(p) - (1-p)\log_2(1-p)$$

Si maintenant, nous allons représenter graphiquement cette entropie en fonction de p, nous aurons:



Comme nous l'avons énoncé plus haut, cette entropie est maximale quand la loi de probabilité est équiprobable, ici p=0.5 (P(0)=P(1)=0.5.

On remarque aussi que cette entropie est nulle pour p=0 et pour p=1 représentant respectivement une certitude que la source n'émet que des 0 ou que des 1 (pas d'incertitude donc pas d'information et l'entropie est donc nulle)

ENTROPIE D'UNE SOURCE

Exemple 2 : Cas d'une source de k symboles $\{m_1, m_2, ..., m_K\}$ équiprobables $\{P(m_1) = P(m_2) = ,....., = P(m_K) = 1/K\}$, alors cette entropie sera:

$$H(M) = -\sum_{k=1}^{K} \frac{1}{K} \log_2\left(\frac{1}{K}\right) =$$

$$= -\frac{1}{K} \log_2\left(\frac{1}{K}\right) =$$

$$= \log_2 K$$

Evidemment, cette entropie est maximale pour cette source compte tenu que la loi de probabilité qui la régie est équiprobable.

Dans le cas d'une loi quelconque régissant une source de K éléments nous avons:

$$H(M) \leq \log_2 K < H(M) + 1$$

REDONDANCE D'UNE SOURCE

Comme nous venons de le voir pour une source M de longueur K, l'entropie où quantité d'information moyenne délivrée par cette source est notée H(M).

Elle est maximale pour une loi de probabilité équiprobable :

$$H_{\max}(M) = \log_2 K$$

On définit la redondance de la source comme étant l'écart ou la différence entre la valeur maximale possible (lorsque tous les symboles sont équiprobables) de son entropie et son entropie réelle.

$$\text{Re}\,dondance = H_{\max}(M) - H(M) = \log_2 K - H(M)$$

Une source dont l'entropie est faible est plus redondante. Autrement dit, il y'a moins d'incertitude et donc moins d'information délivrée par la source et par conséquence trop de redondance.

DEBIT D'UNE SOURCE

Le débit d'information d'une source X dépend bien sur de la quantité d'information délivrée par cette source H(X) (l'entropie) et la durée moyenne d'un symbole de cette source que l'on notera τ.

En effet, si un symbole dure en moyenne τ secondes alors le nombre moyen N_{moyen} de symboles délivré par cette source en une seconde (une sorte de fréquence) est:

$$N_{moyen} = \frac{1}{\tau}$$

Maintenant, comme chaque symbole contient en moyenne un information représentée par l'entropie de la source H(X) alors, on déduit le débit d'information comme étant:

$$D_{\inf} = H(X) \times N_{moyen} = \frac{H(X)}{\tau}$$

THEOREME DE SHANNON

Comme nous l'avons déjà énoncé plus haut chaque symbole de l'alphabet délivré par une source peut être codé en vue de le transmettre ou de le stocker.

❑ Ce codage peut être de longueur fixe, où chaque symbole se voit attribuer un mot de code de même taille, comme par exemple le codage ASCII pour les caractère alphanumérique utilisé entre autres avec les imprimantes et les transmissions séries asynchrones ou encore le codage des niveaux de gris des pixels d'une image numérique.

❑ Mais on peut aussi opter pour un codage de longueur variable où chaque symbole prendra un code dont la longueur va dépendre de la quantité d'information qu'il contient (un symbole moins probable, contient plus d'information donc il sera codé avec une longueur plus grande…etc).

THEOREME DE SHANNON

Le théorème fondamental de Shannon dit que quel que soit le code utilisé pour représenter les symboles d'une source M la longueur moyenne N_{moyen} des symboles de cette source est toujours supérieure ou égale à l'entropie de la source.

$$N_{moyen} \geq H(M)$$

L'entropie de la source représente donc une limite inférieure du nombre de bits nécessaires pour coder les symboles de cette source.

Dans les codages entropiques, l'idée est de réaliser un code dont la longueur moyenne est la proche possible de $H(M)$. L'exemple typique d'un tel codeur est celui de Huffman.

PROPRIETES DES CODES SOURCES

Code régulier

Un code est dit régulier ou non-singulier si tous les mots de code sont distincts.

Code déchiffrable

Un code régulier est déchiffrable ou à décodage unique (ou tout simplement décodable) si pour deux données différentes nous aurons deux codes différents. Autrement dit, toute séquence codée est décodable par une unique séquence de symbole de source.

Pour les codes de longueur variable, cette propriété est satisfaite pour les codes dits sans préfixes, obtenus en évitant qu'un mot du code ne soit identique au début d'un autre.

Code prefixe

Un code est dit préfixe (ou sans préfixe), il s'agit surtout de code à longueur variable, si aucun mot de code n'est le début d'un autre mot de code.
Un code de Prefixe est un code déchiffrable (ou décodable),

Par exemple, l'ensemble {0, 100, 101, 111, 1100, 1101} est un code préfixe.

PROPRIETES DES CODES SOURCES

Code instantané

On dit d'un code qu'il est à décodage instantané s'il est possible de décoder les mots de code dès lors que tous les symboles qui en font partie ont été reçus. Un code Instantané est un code sans préfixe (ou code Préfixe).

Code avec Séparateur

On consacre un symbole de l'alphabet de destination comme séparateur de mot de code.

PROPRIETES DES CODES SOURCES

Exemples

On code les trois caractères a, b et c respectivement par 0, 01 et 10. S'agit il d'un code déchiffrable?

On remarque bien que si nous aurons la séquence de code suivante : 010, nous aurons une ambiguïté pour son décodage, En effet, il peut s'agir de ac comme il peut s'agir de ba,

Par contre si ces trois lettres sont codées respectivement par 0, 10 et 11 nous aurons un code Préfixe qui par définition il est déchiffrable,

Alors que si ces trois lettres sont codées respectivement par 0, 01 et 11 le code est déchiffrable mais il n'est pas Préfixe,

- **Un code Préfixe est une condition suffisante mais pas nécessaire pour que ce code soit déchiffrable.**

- **Tout code déchiffrable de longueur fixe est Préfixe.**

PROPRIETES DES CODES SOURCES

Exemples

Que peut on dire des codes suivantes?

- C_1={0; 11; 101}
- C_2={00; 01; 001}
- C_3={0; 01; 10}
- C_4={000; 001; 01; 1}
- C_5={000100; 100101; 010101; 111000}
- C_6={0; 01; 11}.
- C_7={0; 10; 110; 1110}.
- C_8={0; 10; 110; 1111}.
- C_9={0; 01; 011; 0111}.
- C_{10}={0,10,101,0101}.
- C_{11}={0,10,110,111}.

PROPRIETES DES CODES SOURCES

Condition nécessaire et suffisante de déchiffrabilité

L'extension d'ordre n d'un code est le code formé par les séquences de n mots du code initial.

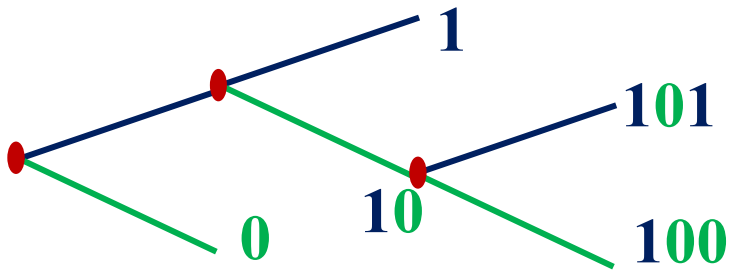
Extension d'un code

Pour qu'un code soit déchiffrable il faut et il suffit que toutes ses extensions soient régulières.

Codes et arbres

À un code (binaire) Préfixe nous pouvons toujours associer un arbre binaire, où tous les mots de code se situent sur les feuilles de l'arbre.

Exemple le code suivant: C1={0, 11, 101, 100}: C'est un code binaire Préfixe déchiffrable. Dès lors nous pouvons lui associer un arbre binaire,



INEGALITE DE KRAFT

L'inégalité de Kraft est la plus importante propriété des codes déchiffrables et plus précisément les codes Préfixes :

Théorème : Pour qu'un code Préfixe (code instantané ou encore code déchiffrable), correspondant à la source d'alphabets $M=\{m_1, m_2, ,\ldots, m_K\}$ existe il faut et il suffit que ses longueurs de mots, notées l_k où $k=\{1, 2,\ldots, K\}$, vérifient l'inéquation suivante:

$$\sum_{k=1}^{K} 2^{-lk} \leq 1$$

Cette même condition a été aussi établie par M_cMillan pour les codes uniquement déchiffrables, A cette effet, on l'appelle souvent l'inégalité de Kraft-M_cMillan.

Code complet

Un code est dit complet s'il vérifie l'égalité :

$$\sum_{k=1}^{K} 2^{-lk} = 1$$

INEGALITE DE KRAFT

Exemples:

A partir de l'inégalité de Kraft, que pouvez vous dire à propos des codes suivants:

- C1 = {00, 01, 10, 11}.
- C2 = {0, 100, 110, 111}.
- C3 = {0, 10, 110, 11}.

CODAGE DE HUFFMAN

Cet algorithme, proposé par David Huffman en 1952, est un codeur entropique de type statistique.

C'est un code optimal: Un code optimal est un code préfixe de longueur moyenne minimale.

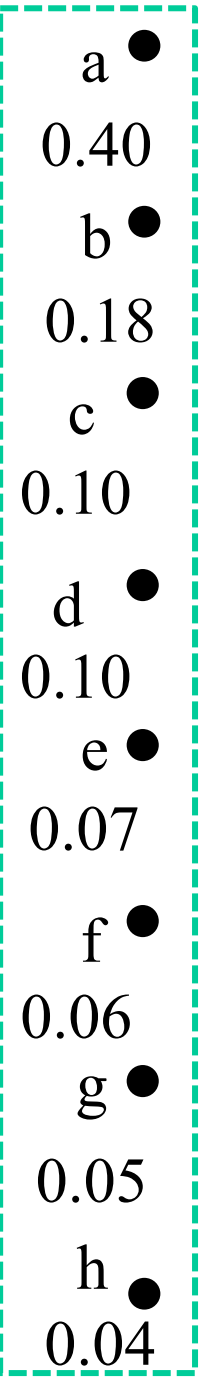
Dans ce cas la compression est d'autant plus importante que la longueur moyenne des mots de code est faible.

Le principe ressemble au code morse.

En effet, l'idée adoptée dans ce codeur, comme pour tous les codeurs statistiques, est d'affecter :

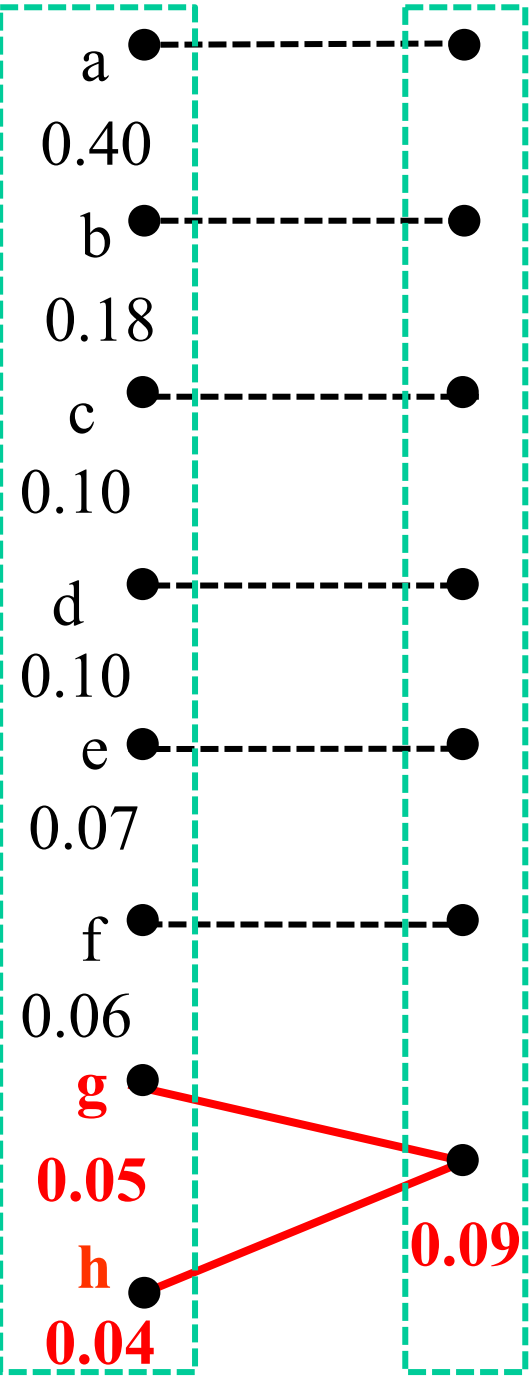
- une longueur faible pour les symboles fréquents (probables),
- un code plus long pour les symboles improbables ou rares.

EXEMPLE D'UN CODAGE DE HUFFMAN



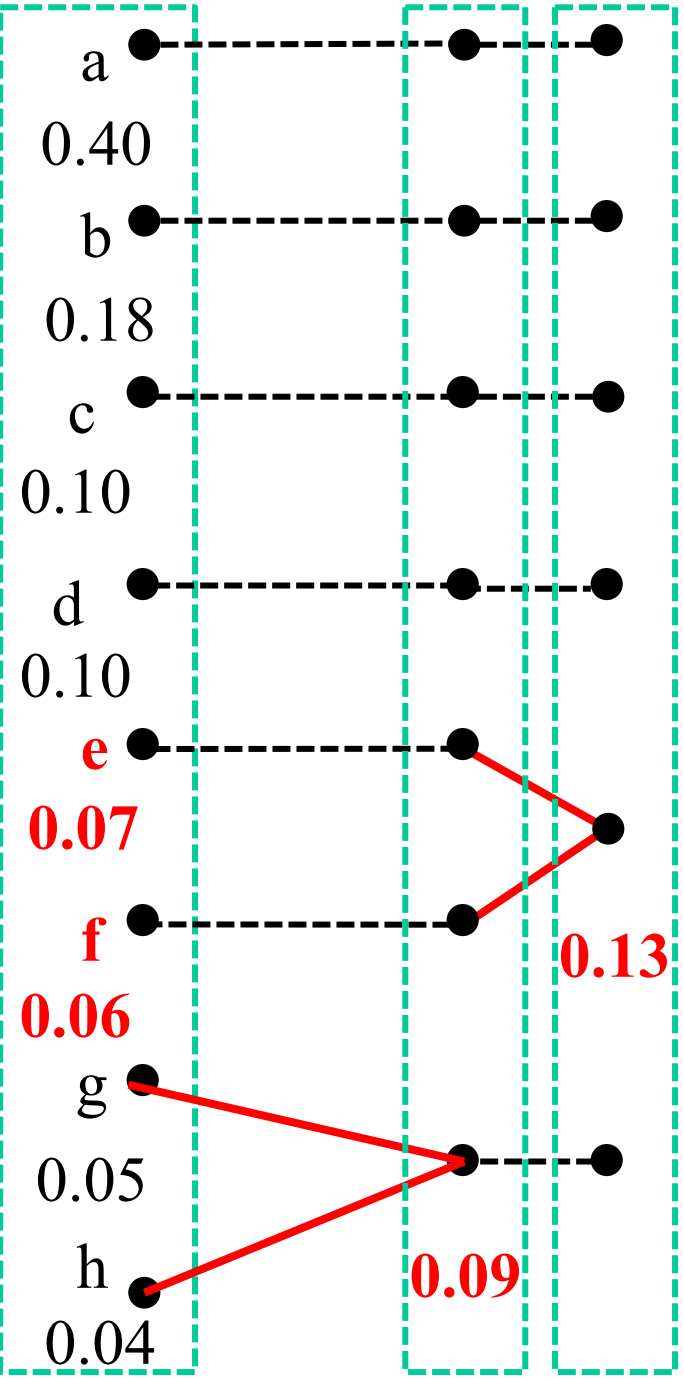
CLASSEMENT DES PROBABILITES D 'APPARITION PAR ORDRE DECROISSANT

Première étape



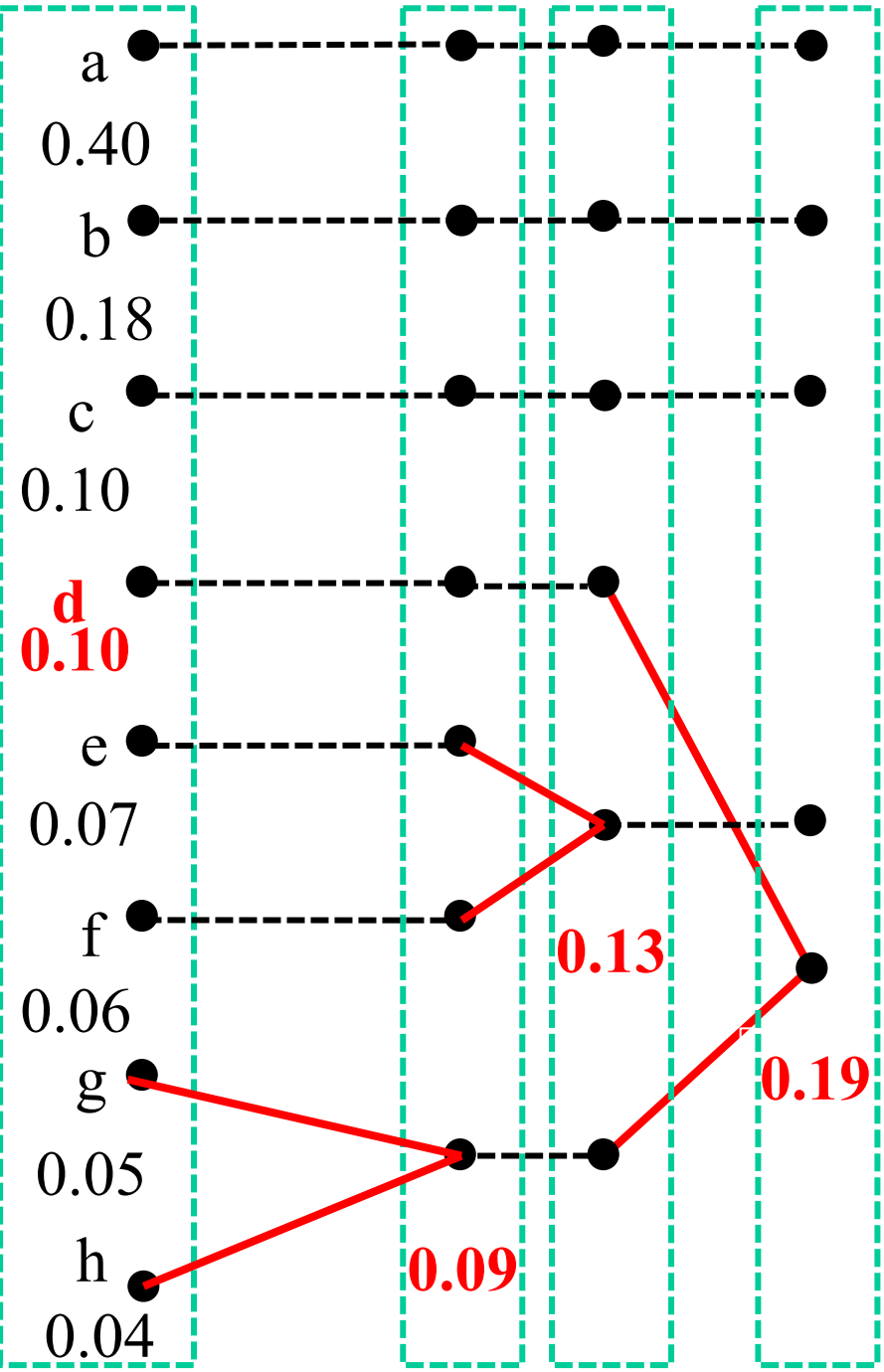
Regroupement des deux lettres
de plus faible probabilité
la probabilité du regroupement
est la somme des deux probabilités

Deuxième étape

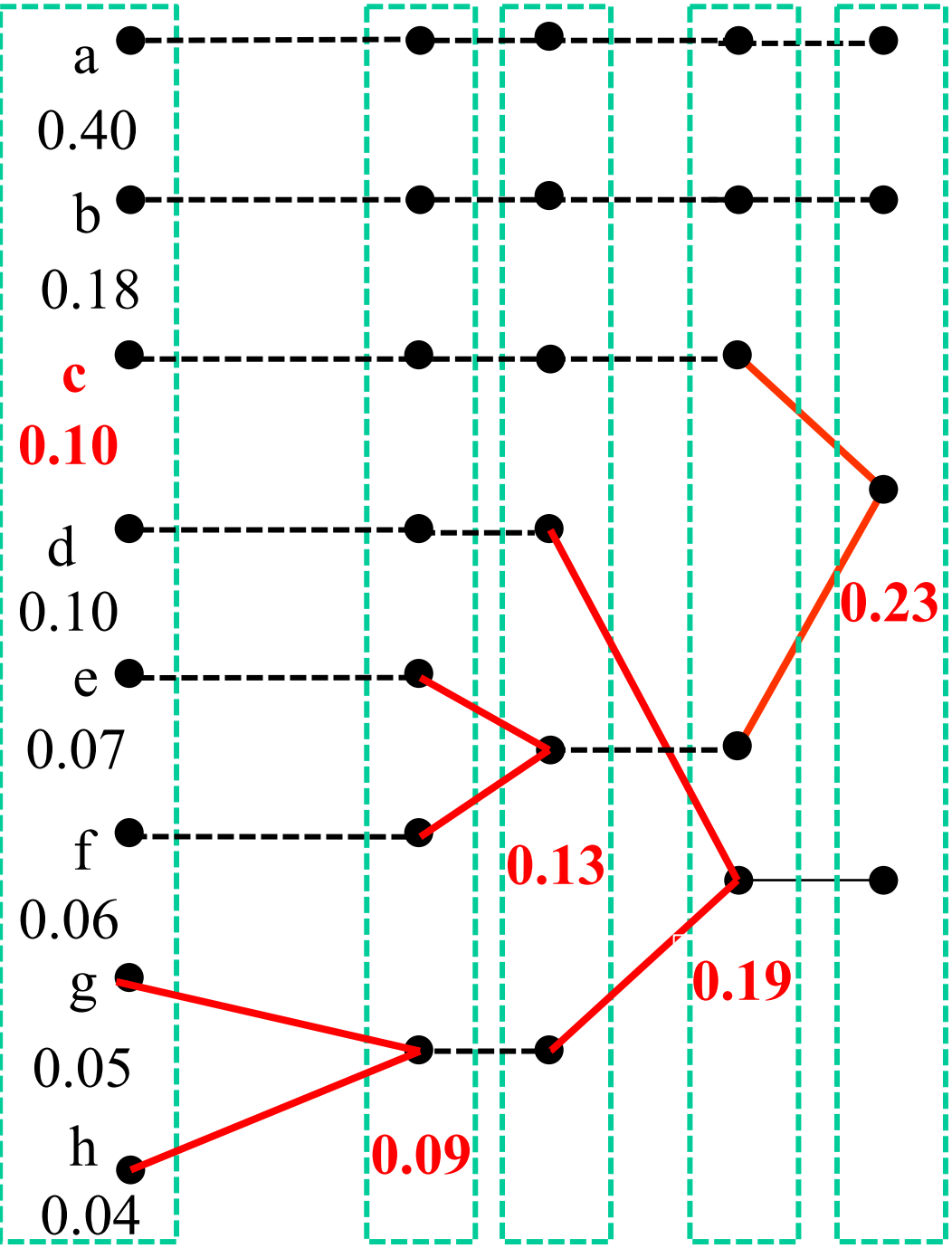


Regroupement des deux lettres
ou groupements de lettres
de plus faible probabilité

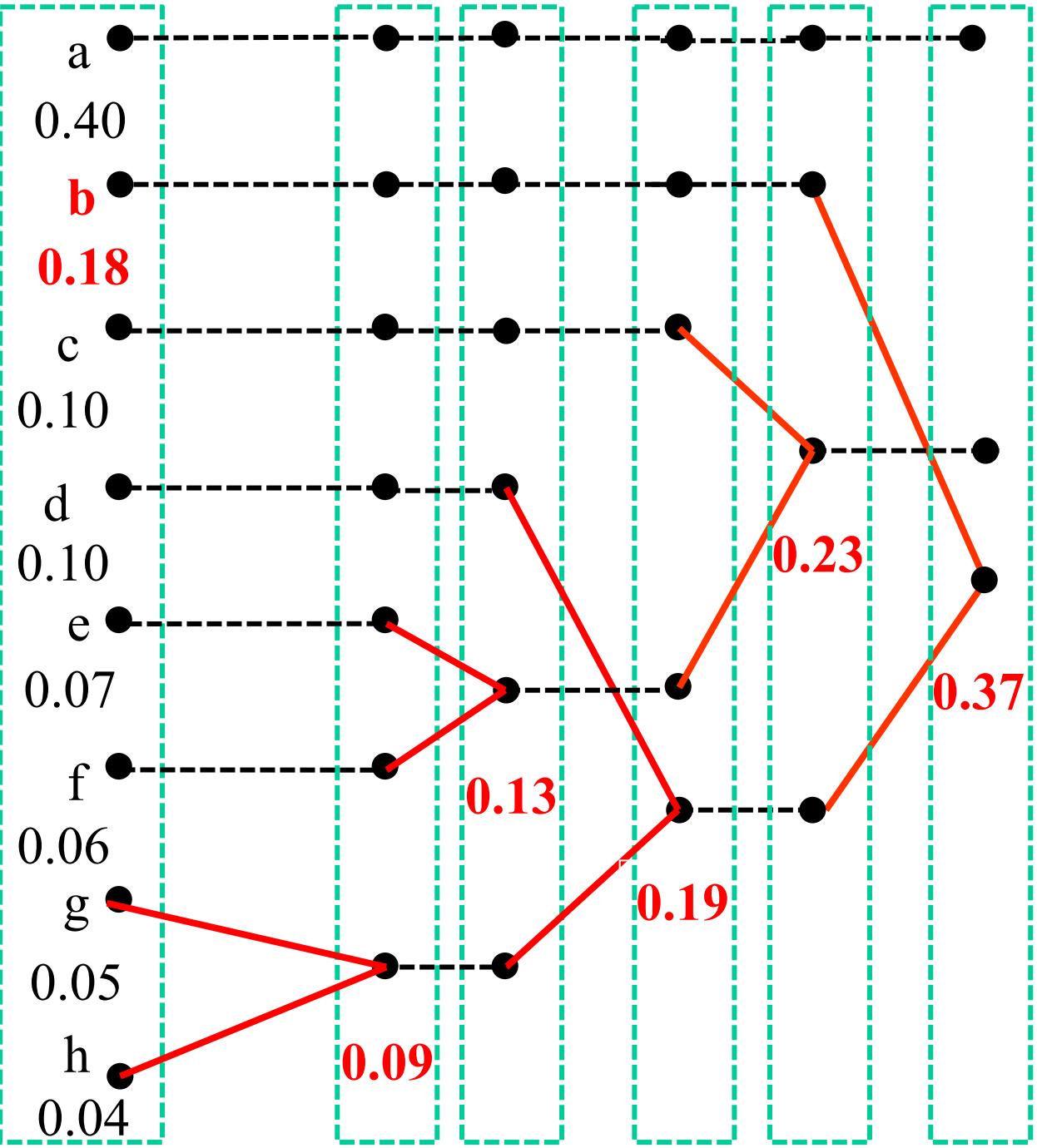
Troisième étape



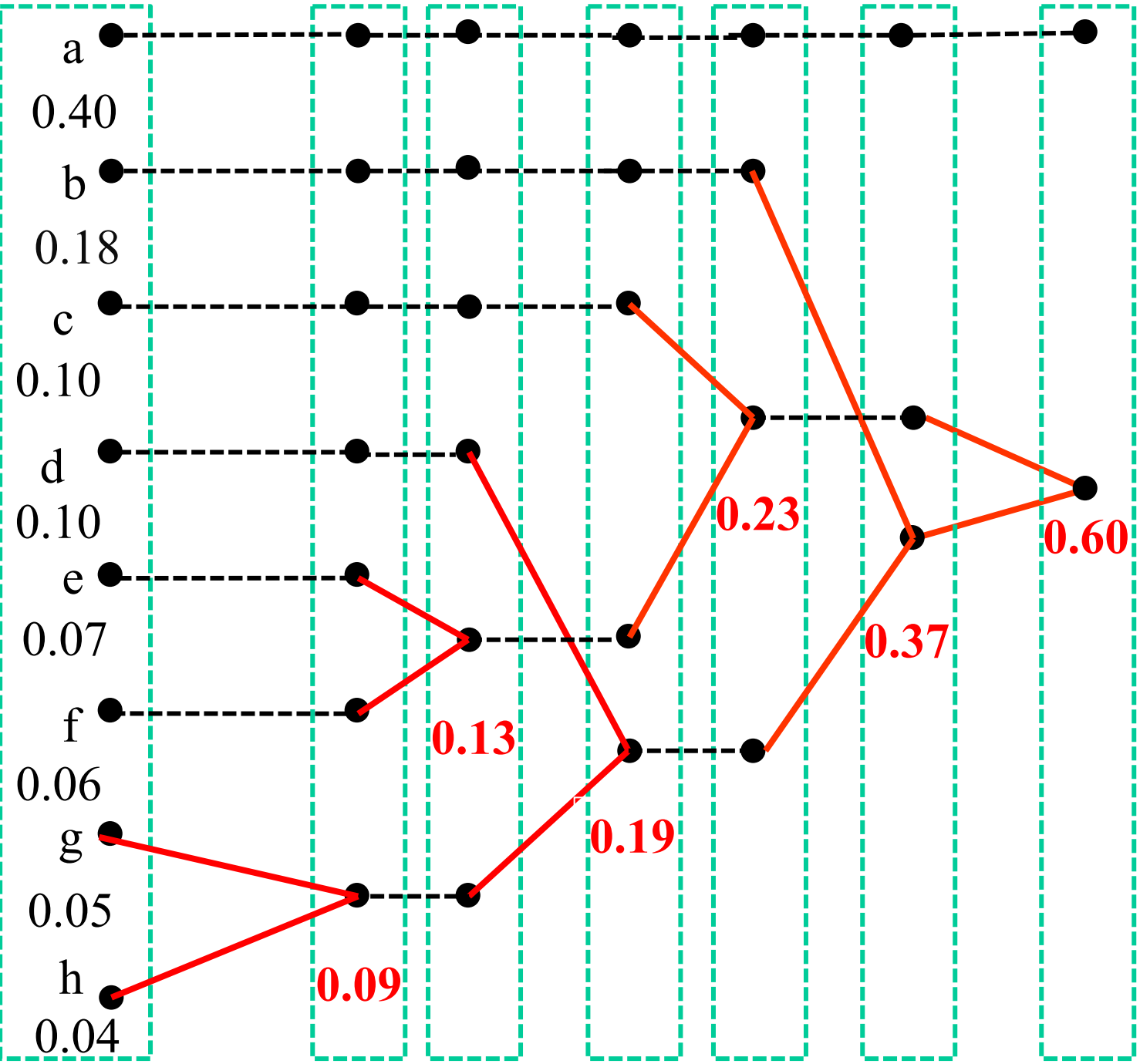
Quatrième étape

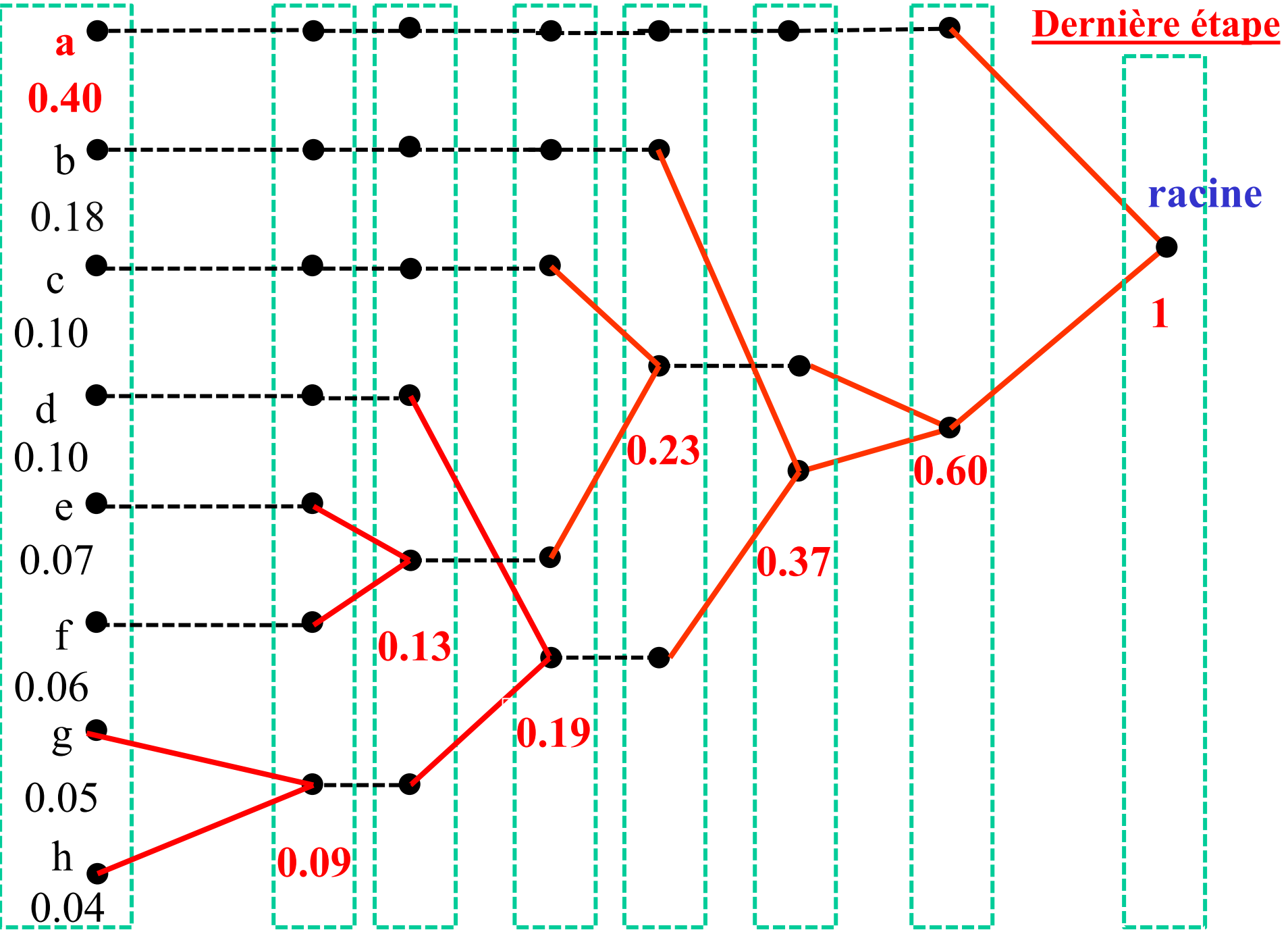


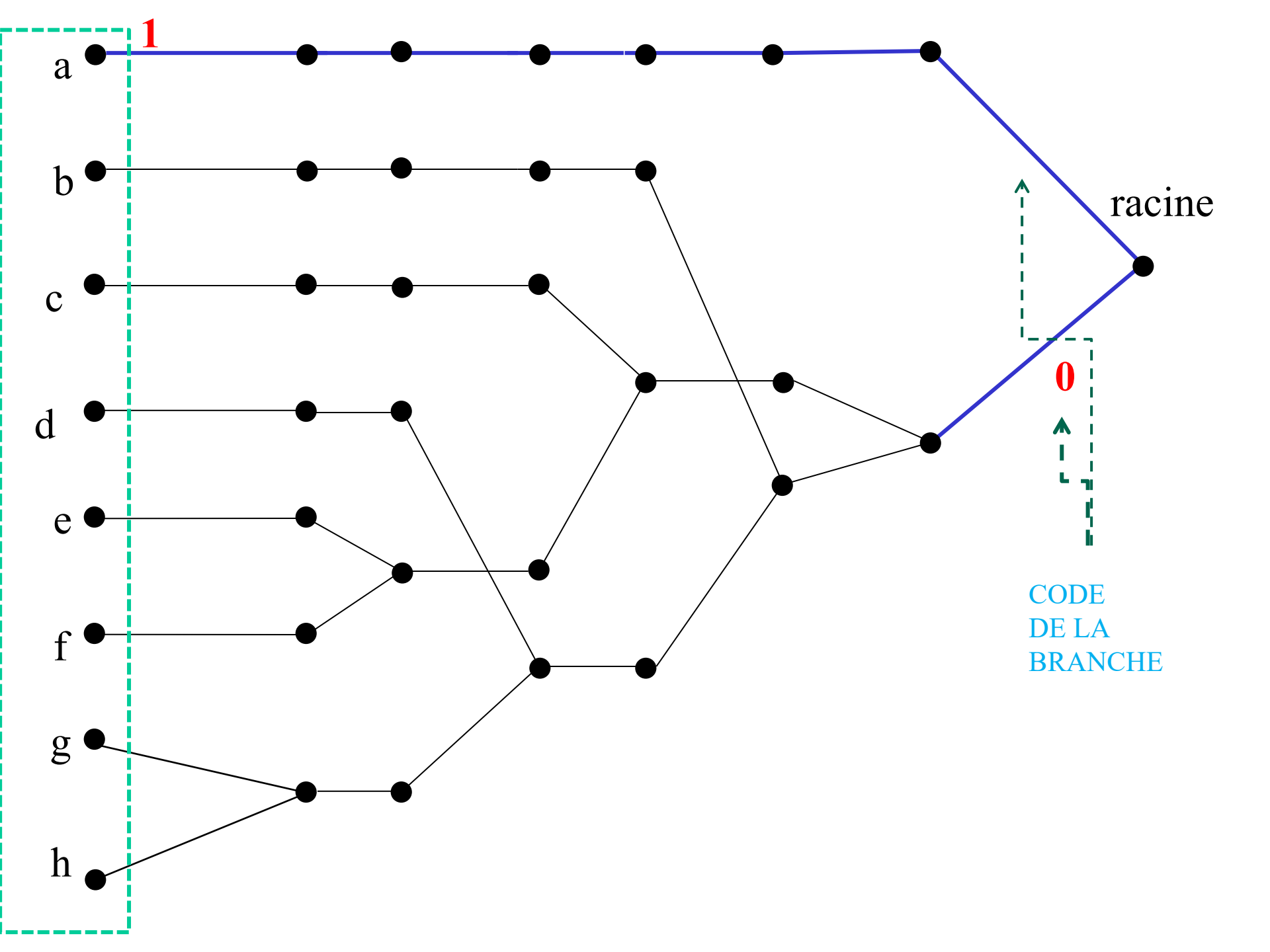
Cinquième étape

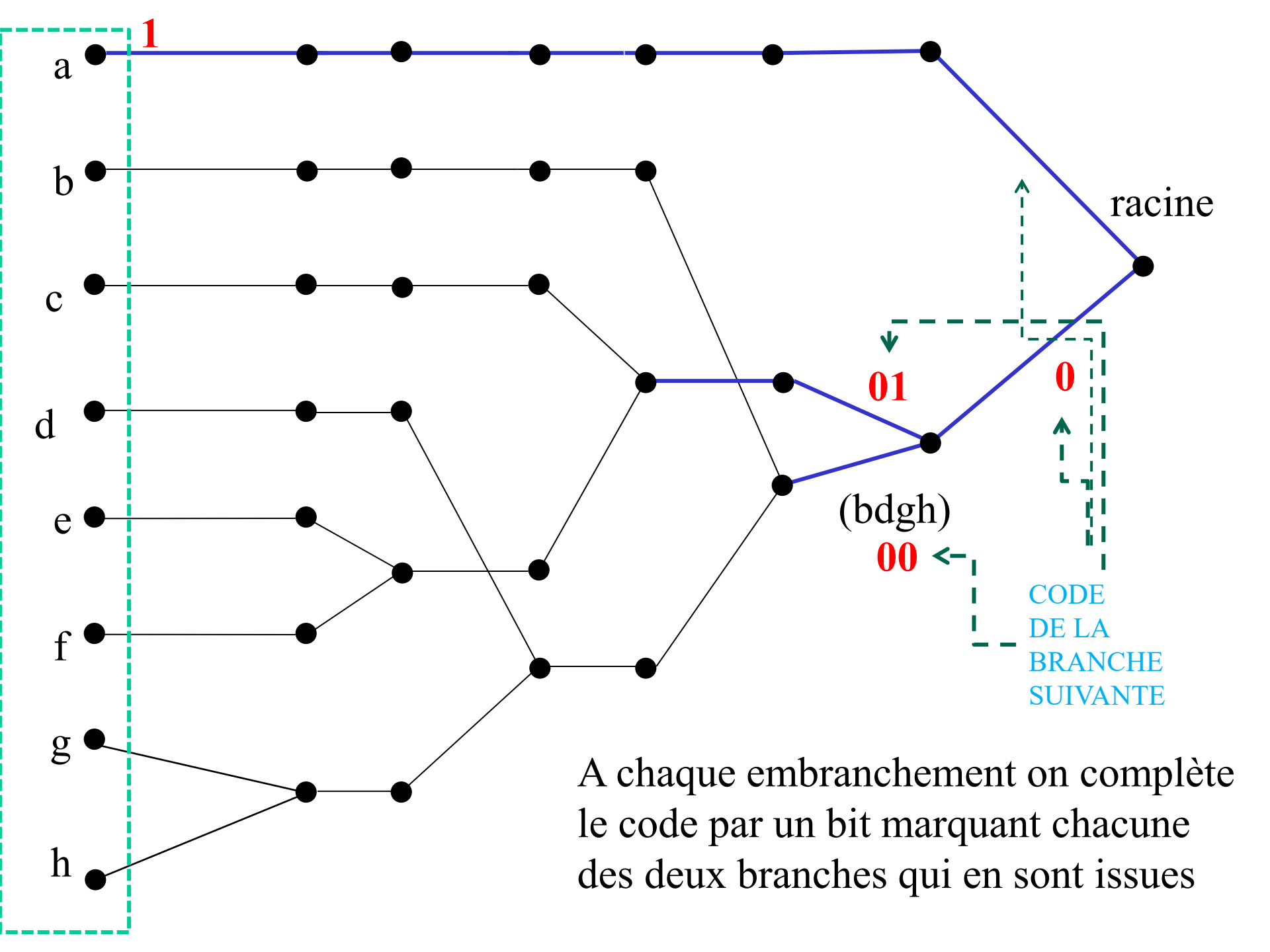


Sixième étape

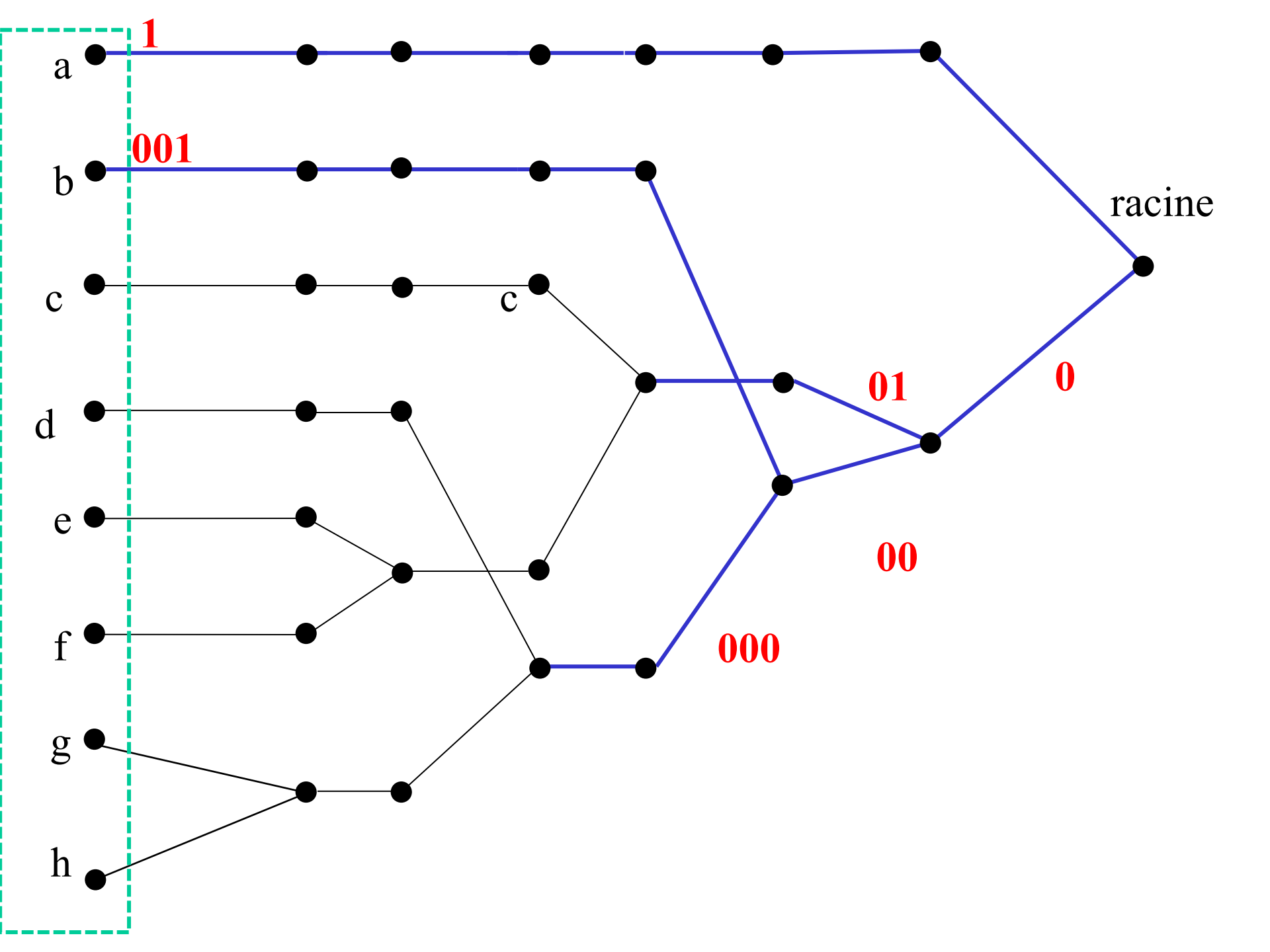


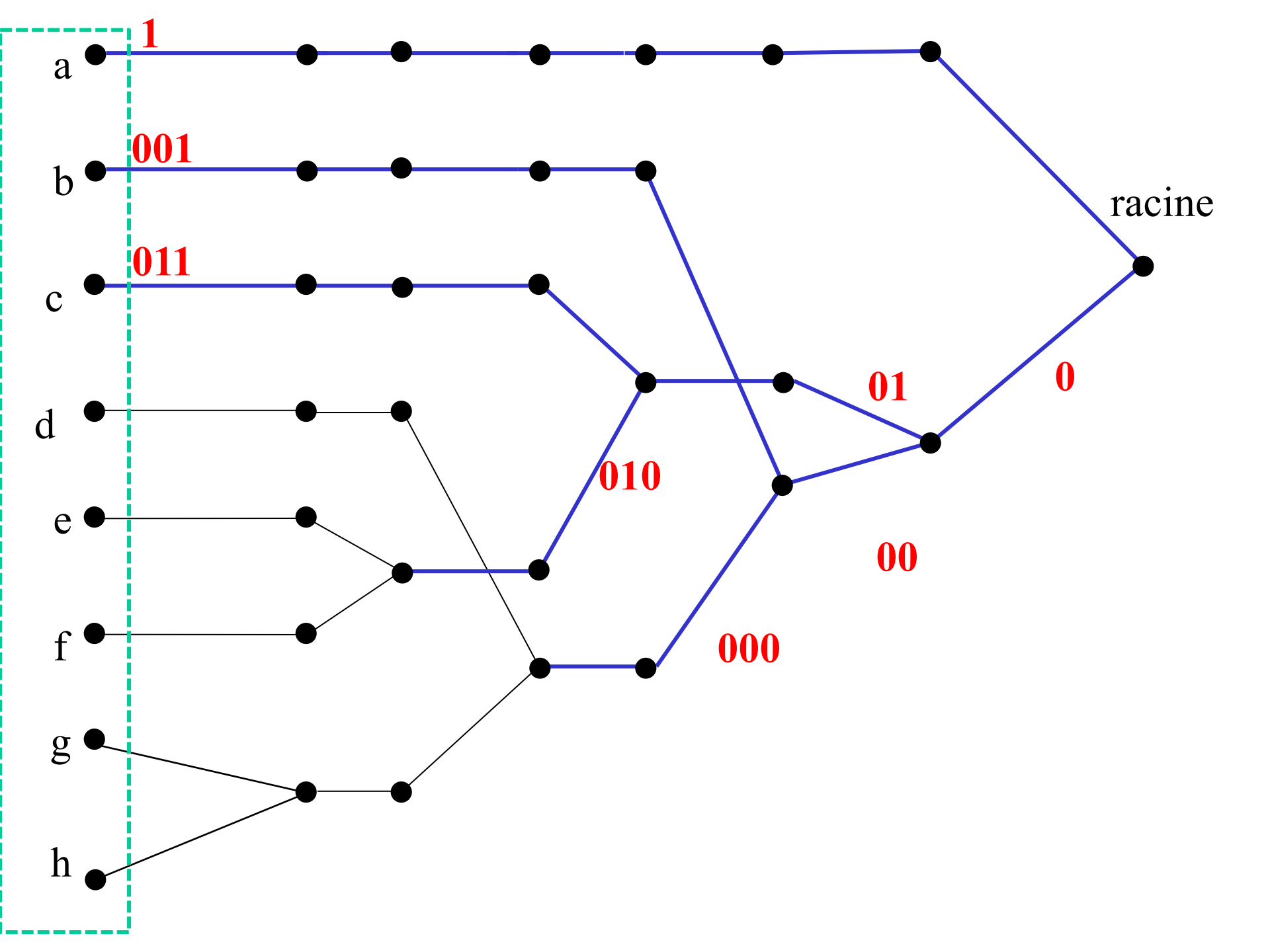

Dernière étape
racine
1
a
0.40
b
0.18
c
0.10
d
0.10
e
0.07
f
0.06
g
0.05
h
0.04
0.09
0.13
0.19
0.23
0.37
0.60

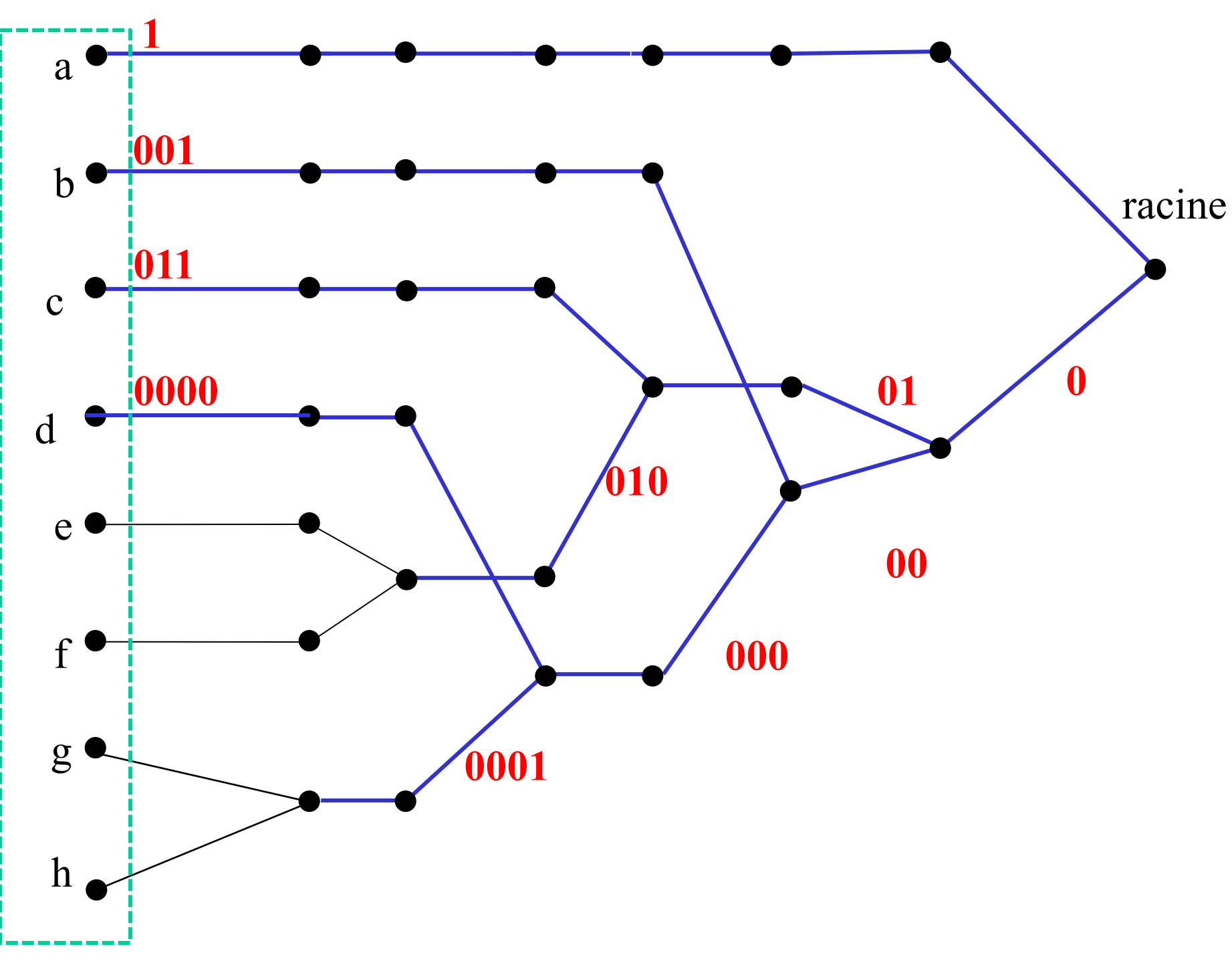

1
a
b
c
d
e
f
g
h
racine
0
CODE
DE LA
BRANCHE

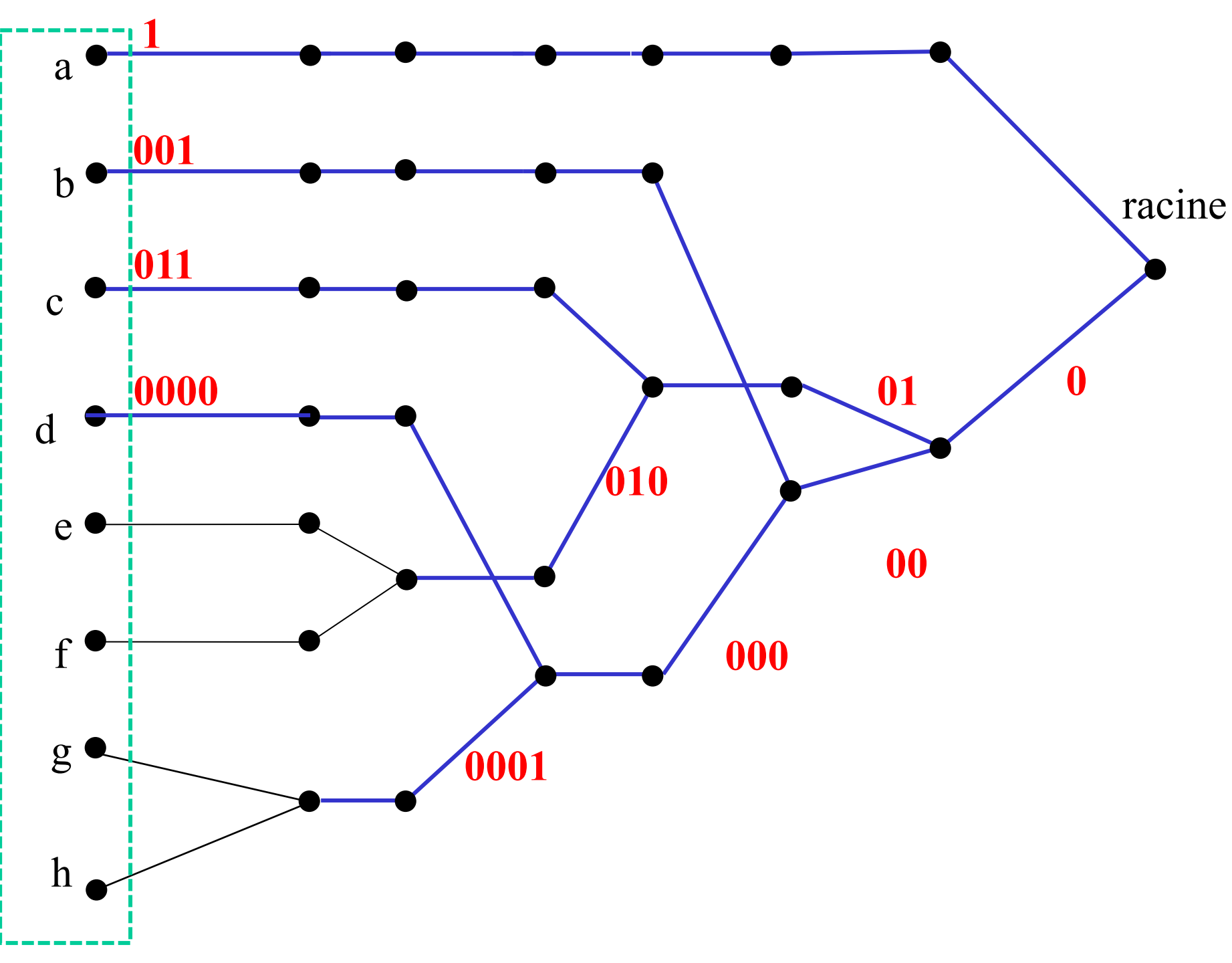


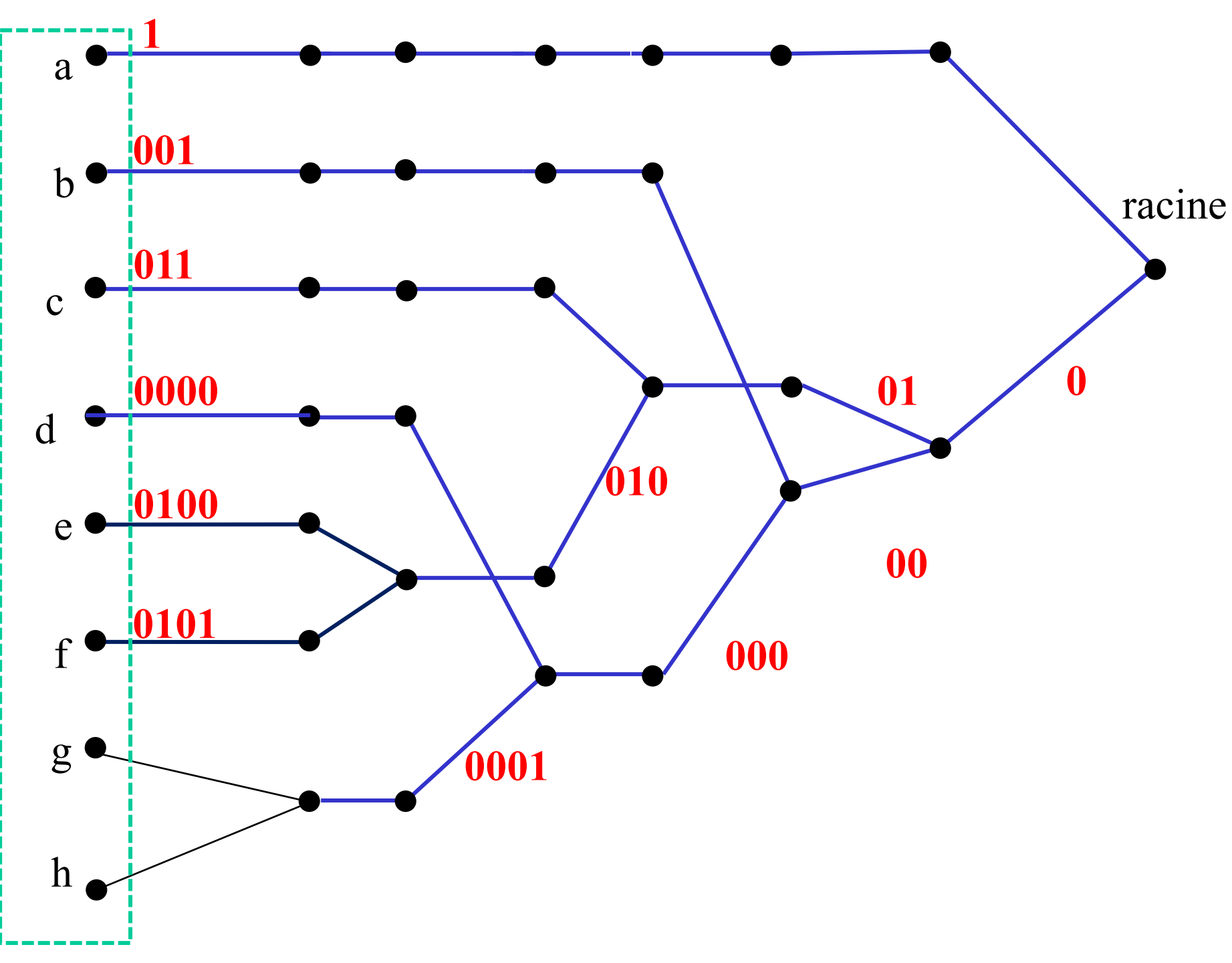
A chaque embranchement on complète le code par un bit marquant chacune des deux branches qui en sont issues

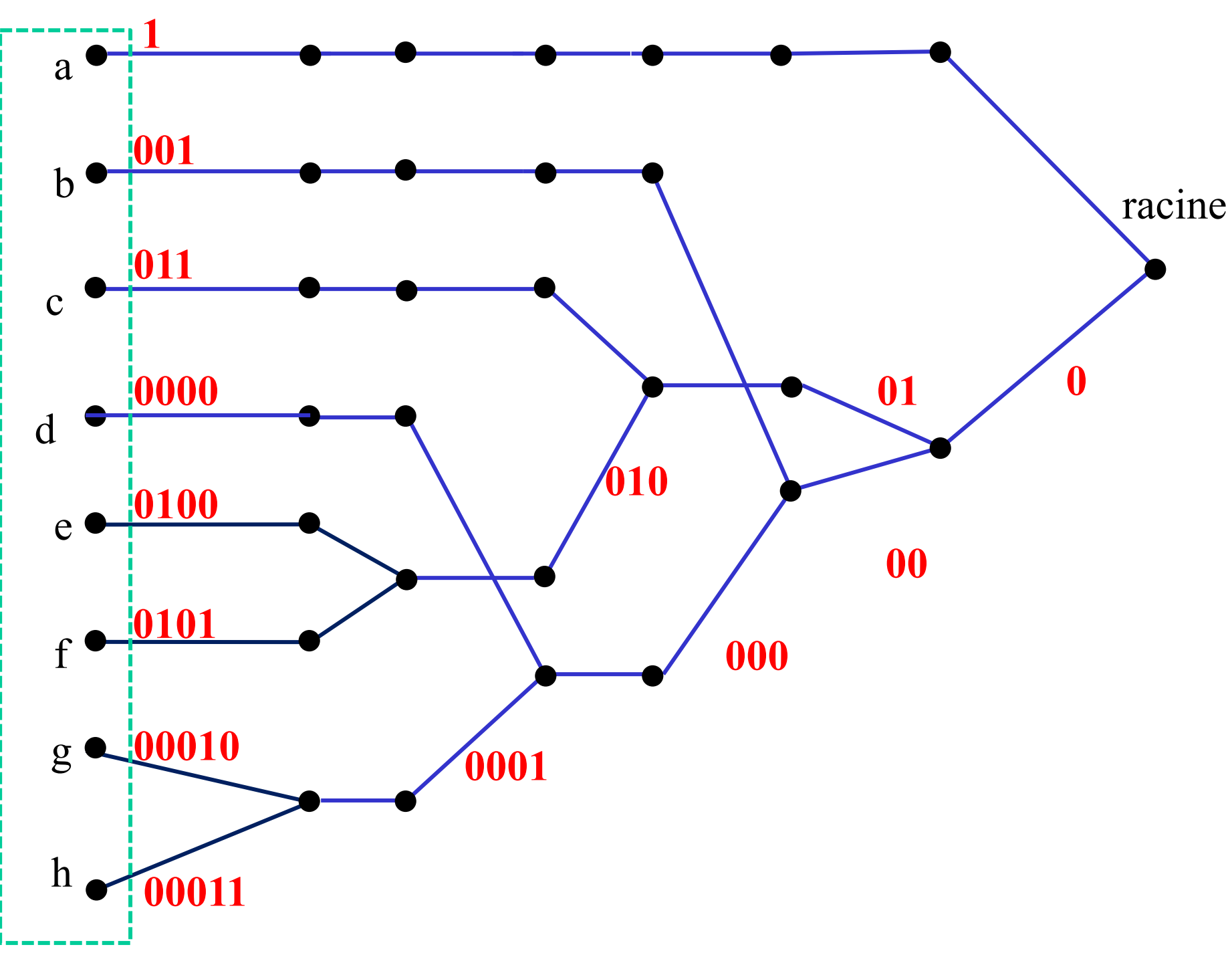

1
a
001
b
racine
c
c
01
0
d
e
00
f
000
g
h


a
b
c
d
e
f
g
h
1
001
011
010
000
00
01
0
racine


a
b
c
d
e
f
g
h
1
001
011
0000
010
0001
000
00
01
0
racine


a
b
c
d
e
f
g
h
1
001
011
0000
010
0001
000
00
01
0
racine


a
b
c
d
e
f
g
h
1
001
011
0000
0100
0101
0001
010
000
00
01
0
racine


a
1
b
001
c
011
d
0000
e
0100
f
0101
g
00010
h
00011
0001
000
010
00
01
0
racine

CODAGE DE HUFFMAN

Le résultats du codage de Huffman pour l'exemple précédent est donné sur le tableau suivant

SYMBOLE	CODAGE HUFFMAN
a	1
b	001
c	011
d	0000
e	0100
f	0101
g	00010
h	00011

Pour cette exemple, la longueur moyenne du code obtenu peut être calculé ainsi :

$$N_{moyen,Huff} = \sum_{k=1}^{7} P(S_k) \times Code_{Huff}(S_k)$$

A lors que son entropie est donnée par:

$$H(X) = -\sum_{k=1}^{7} P(S_k) \times \log_2 P(S_k)$$

$$N_{moy,Huff} = 0.4 \times 1 + 0.18 \times 3 + 0.10 \times 3 + 0.10 \times 4 + 0.07 \times 4 + 0.06 \times 4 + 0.05 \times 5 + 0.04 \times 5 = 2,33$$

$$H(X) = 0.4 \log_2 0.4 + 0.18 \log_2 0.18 + 0.10 \log_2 0.10 + 0.10 \log_2 0.10 + 0.07 \log_2 0.07 +$$
$$+ 0.06 \log_2 0.07 + 0.05 \log_2 0.05 + 0.04 \log_2 0.04 =$$

CODAGE DE SHANNON-FANO

Il s'agit d'un codage entropique pour la compression sans pertes de données, élaboré par Robert Fano à partir d'une idée de Claude Shannon, produisant un code très proche de celui de Huffman de type statistique à longueur variable (VLC=Variable Length Coding), bien qu'il n'est pas toujours optimal, comme c'est le cas pour le code de Huffman.

CODAGE DE SHANNON-FANO

Principe de l'algorithme Shannon-Fano:

- Classement dans un tableau, des probabilités d'apparition ou occurrences des symboles de la séquence par ordre décroissants,

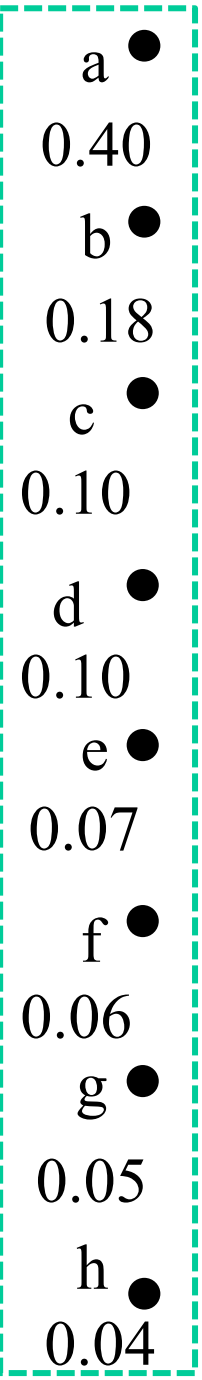
- Séparer les symboles en deux sous-groupes de sorte que le total des nombres d'occurences soient sensiblement égaux dans les deux sous-groupes

- Le tableau est coupé en deux groupes de symboles S0 et S1 dont la somme des probabilités de chaque groupe avoisine 0.5

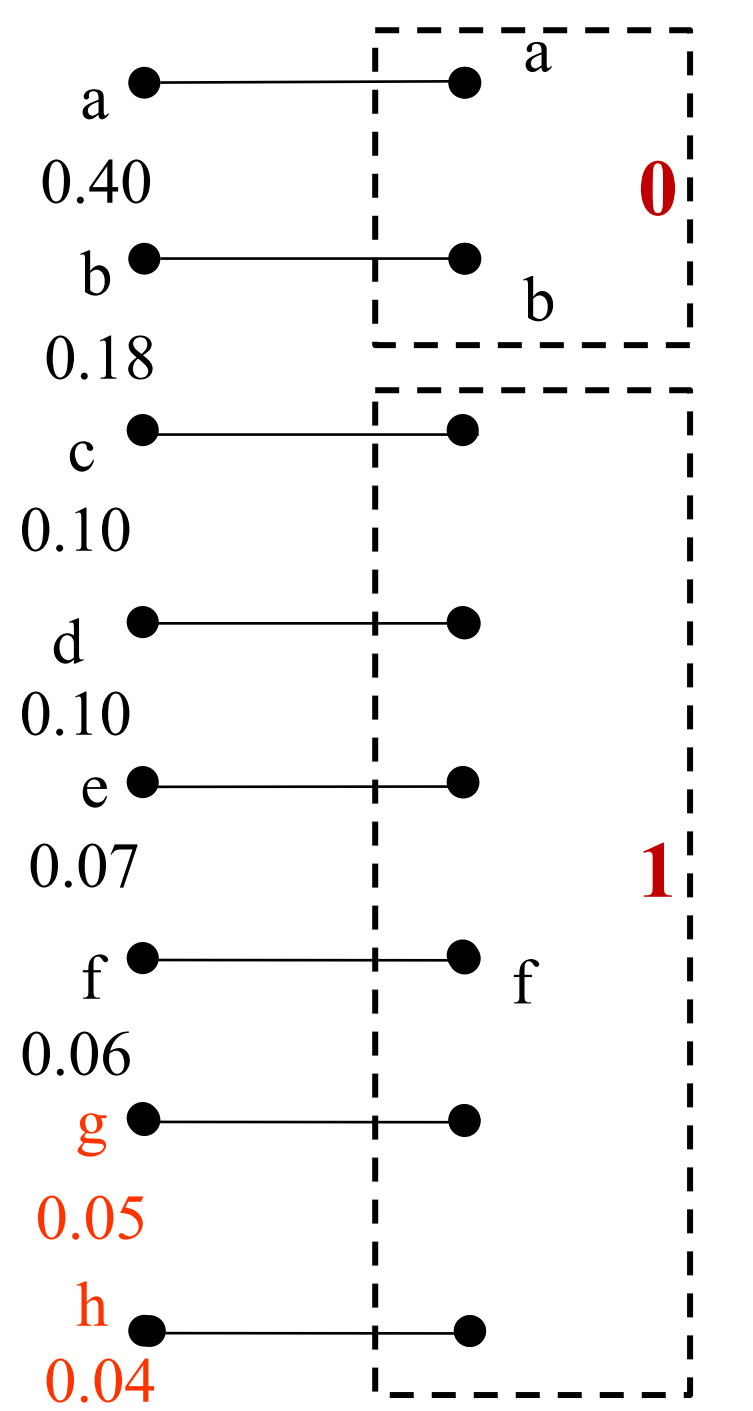
- Le groupe S0 est codé par un "0" et S1 par un "1 »,

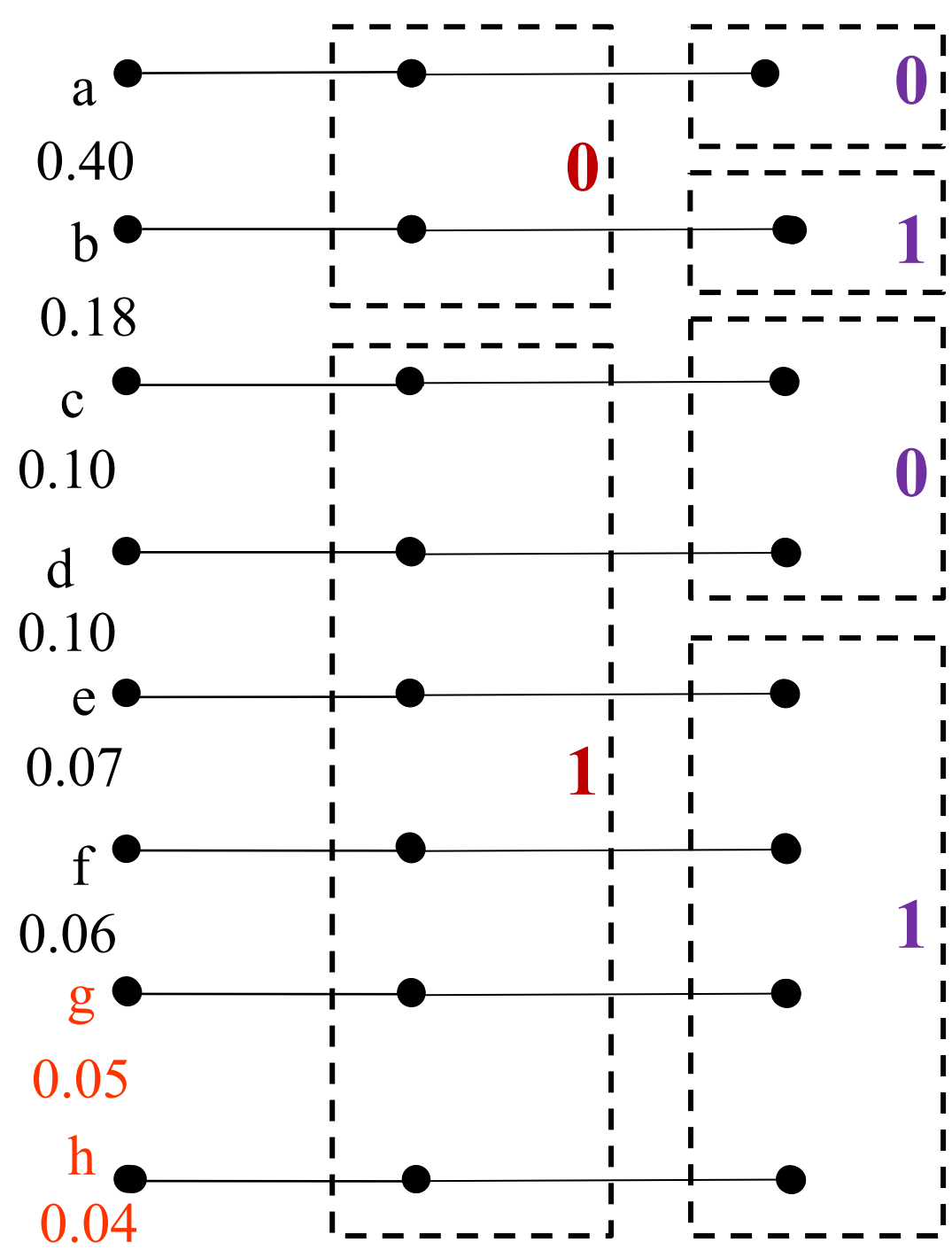
- Recommencer pour chacun des sous-groupes, jusqu'à ce qu'ils n'aient plus qu'un seul élément.

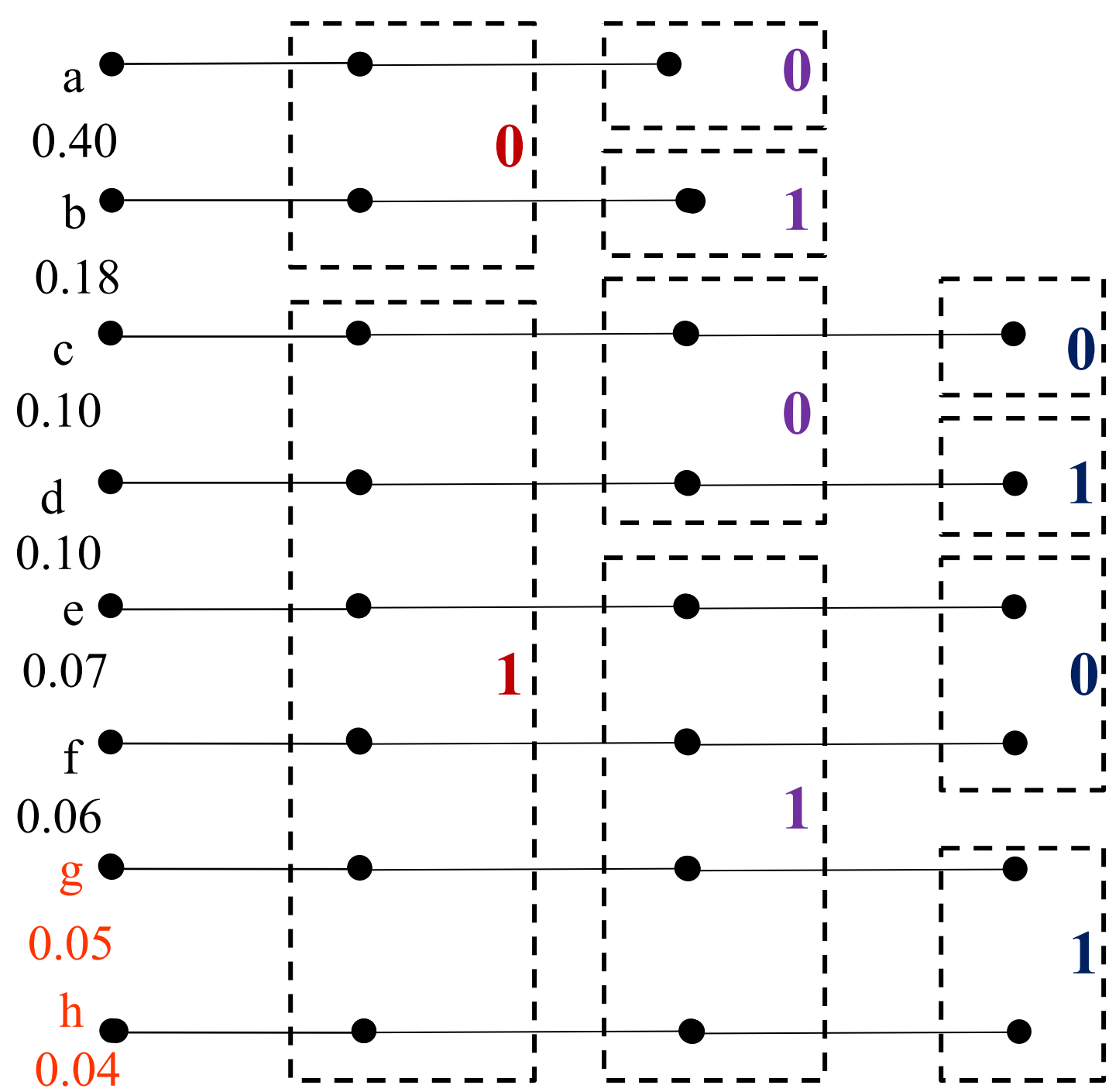
EXEMPLE D'UN CODAGE SHANNON-FANO

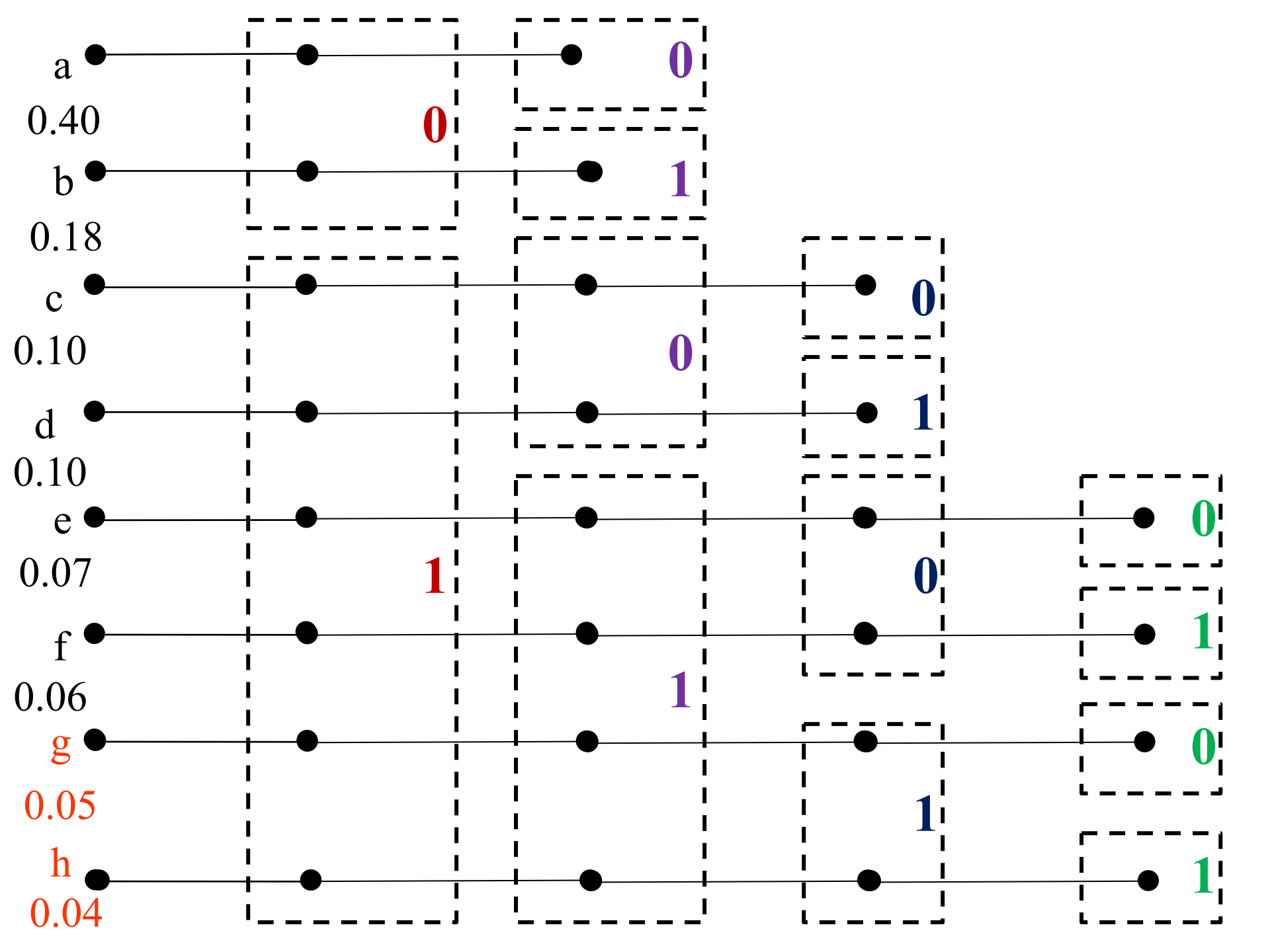


CLASSEMENT DES PROBABILITES D 'APPARITION PAR ORDRE DECROISSANT


a
a
0.40
0
b
b
0.18
c
0.10
d
0.10
e
0.07
1
f
f
0.06
g
0.05
h
0.04


a
0.40
b
0.18
c
0.10
d
0.10
e
0.07
f
0.06
g
0.05
h
0.04
0
1
0
1
0
1


a
0.40
b
0.18
c
0.10
d
0.10
e
0.07
f
0.06
g
0.05
h
0.04
0
0
1
0
0
1
1
0
1
1


a
0.40
b
0.18
c
0.10
d
0.10
e
0.07
f
0.06
g
0.05
h
0.04
0
0
1
0
1
0
1
0
1
0
1
0
1
0
1

CODAGE DE SHANNON-FANO

Le résultats du codage de Shannon-Fano pour l'exemple précédent est donné sur le tableau suivant

SYMBOLE	CODAGE SHANNON-FANO
a	00
b	01
c	100
d	101
e	1100
f	1101
g	1110
h	1111

Pour cette exemple, la longueur moyenne du code obtenu peut être calculé ainsi :

$$N_{moyen,Fano} = \sum_{k=1}^{7} P(S_k) \times Code_{Fano}(S_k)$$

A lors que son entropie est donnée par:

$$H(X) = -\sum_{k=1}^{7} P(S_k) \times \log_2 P(S_k)$$

$$N_{moy,Fano} = 0.4 \times 2 + 0.18 \times 2 + 0.10 \times 3 + 0.10 \times 3 + 0.07 \times 4 + 0.06 \times 4 + 0.05 \times 4 + 0.04 \times 4 = 2{,}64$$

$$H(X) = 0.4 \log_2 0.4 + 0.18 \log_2 0.18 + 0.10 \log_2 0.10 + 0.10 \log_2 0.10 + 0.07 \log_2 0.07 +$$
$$+ 0.06 \log_2 0.07 + 0.05 \log_2 0.05 + 0.04 \log_2 0.04 =$$

HUFFMAN vs SHANNON-FANO

Nous allons prendre en considération l'exemple précédent, pour comparer les résultats des deux codeurs entropiques.

SYMBOLE	CODAGE SHANNON-FANO	CODAGE HUFFMAN
a	00	1
b	01	001
c	100	011
d	101	0000
e	1100	0100
f	1101	0101
g	1110	00010
h	1111	00011

En ce qui concerne la longueur moyenne de chacun des deux codes obtenus pour le même exemple : $N_{moy,Huff} = 2{,}33$ $N_{moy,Fano} = 2{,}64$

Que nous pouvons les comparer à l'entropie de cet alphabet $H(X) = ?$

CODAGE ARITHMETIQUE

Il s'agit aussi d'un un codage statistique, entropique, à longueur variable qui est utilisé dans la compression sans perte, comme les deux précédents codages.

Il est plus efficace que le codage de Huffman (sauf dans le cas particulier où tous les poids des feuilles/nœuds/racines de l'arbre de Huffman sont des puissances de 2).

Malgré un meilleure efficacité, le codage arithmétique, il a été très peu utilisé dans la pratique à cause peut être de sa relative complexité.

Cette meilleure efficacité du codage arithmétique par rapport au codage de Huffamn est due essentiellement qu'il peut coder un caractère **sur une fraction de bit alors** que Huffman code toujours sur un nombre entier de bits.

Prenons l'exemple d'un symbole qui se répète 9 fois sur 10, Huffman va le coder vraisemblablement sur 1 bit alors que le codage arithmétique le codera 0.15 bit

CODAGE ARITHMETIQUE

Pour présenter la compression, nous allons utiliser un exemple et nous décrirons chaque étape de compression. Codons le mot "ESIPE" à l'aide du codage arithmétique.

La première étape consiste à décompter chaque lettre du mot. Nous avons donc 2 'E', 1 'S', 1 'I' et 1 'P'. Nous en générons alors une probabilité de présence dans le mot soit 40% de chance de trouver un E et 20% de chance pour les autres lettres. Dernière actions à effectuer pour cette première partie, nous affectons à chaque lettre un intervalle entre 0 et 1 de la manière suivante :

La lettre 'E' à une probabilité de 40% (soit 0.4). Son intervalle est donc [0,0.4[
La lettre 'P' a une probabilité de 20% (soit 0.2). Son intervalle est donc [0.4,0.6[
Etc...

On obtient dès lors le tableau suivant :

CODAGE ARITHMETIQUE

Lettre	Probabilité	Intervalle
E	4/10	[0,0.4[
S	2/10	[0.4,0.6[
I	2/10	[0.6,0.8[
P	2/10	[0.8,1.0[

CODAGE ARITHMETIQUE

Le codage va maintenant consister à remplacer le mot ESIPE par un nombre flottant lui correspondant. Pour cela, le mot va se voir affecter un intervalle compris entre 0 et 1 où chaque nombre compris entre les deux intervalles permettra de retrouver le mot ESIPE.

L'algorithme appliqué est le suivant : le mot commence avec un intervalle de [0,1[. Puis pour chaque lettre croisée, nous appliquons la formule suivante :

La borne inférieure (BI) du mot est modifiée avec le résultat du calcul "***BI** + **(BS – BI) * Borne_Inférieure_Lettre***"

La borne supérieure (BS) du mot est modifiée avec le résultat du calcul "***BI** + **(BS – BI) * Borne_Supérieure_Lettre***"

Le tableau suivant montre les étapes du calcul:

CODAGE ARITHMETIQUE

Lettre	Borne Inférieure	Borne Supérieure
	0.0	1.0
E	0.0	0.4
S	0.16	0.24
I	0.208	0.224
P	0.2208	0.224
E	0.2208	0.22208

Dès lors, tous nombre flottant entre 0.2208 et 0.22208 est le format compressé du mot "ESIPE"

CODAGE ARITHMETIQUE

Décompression

De la même manière, nous allons présenter la décompression par l'exemple en décompressant notre format compressé.

Prenons le nombre 0.2208 qui code le mot "ESIPE". Le principe de la décompression est très simple. Celle-ci suit les deux étapes suivantes qui se répète jusqu'à l'obtention du mot :

•La prochaine lettre du mot est celle dont l'intervalle contient le nombre du mot actuel (Ex : 0.2208 est dans l'intervalle de E donc la première lettre est E).

•On modifie le nombre représentant le mot à l'aide du calcul ***« (nombre du mot – borne inférieure de la lettre) / probabilité de la lettre*** (Ex : nombre du mot = (0.2208 – 0.0) / 0.4 = 0.552)

Le tableau suivant montre les différentes étapes de la décompression :

CODAGE ARITHMETIQUE

Mot	Lettre	Nouveau code
	E	0.552
E	S	0,76
ES	I	0,8
ESI	P	0,0
ESIP	E	

Ainsi, nous avons récupéré notre premier mot: ESIPE