Illustration du codage de Huffman.

Exemple et illustration tirés de la page Wikipedia (fr).

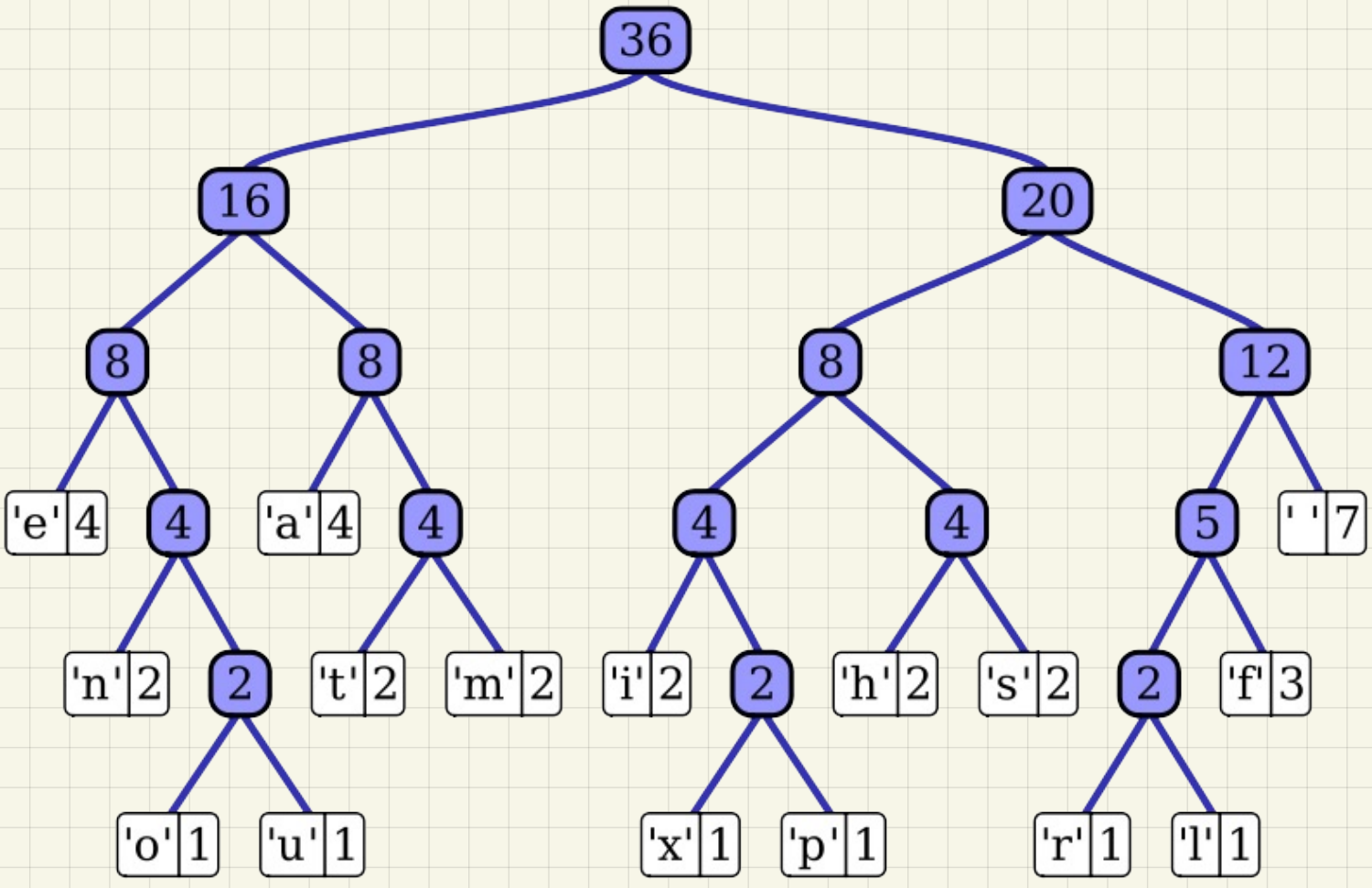
Le message à coder :

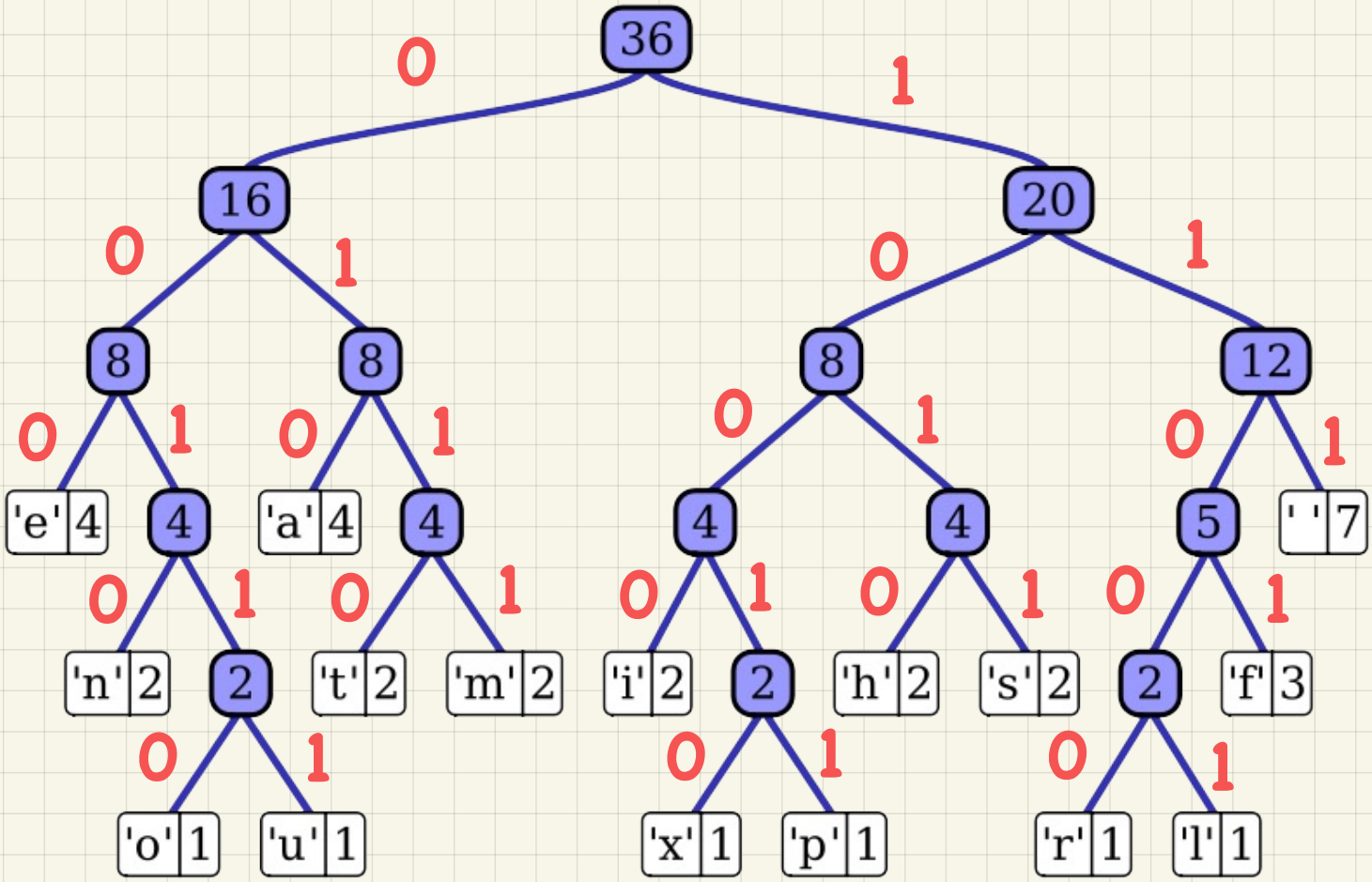
«*this is an example of a huffman tree*»

Ce message comporte (présenté ici par ordre croissant du nombre d'occurrences) :

1 'o', 1 'u', 1 'x', 1 'p', 1 'r', 1 'l', 2 'n', 2 't', 2 'm', 2 'i', 2 'h', 2 's', 3 'f', 4 'e', 4 'a', 7 ' '

Un arbre de Huffman associé est :





Ceci donne la « table de codage » :

‘o’ : 00110
‘u’ : 00111
‘x’ : 10010
‘p’ : 10011
…
‘a’ : 010
‘ ’ : 111

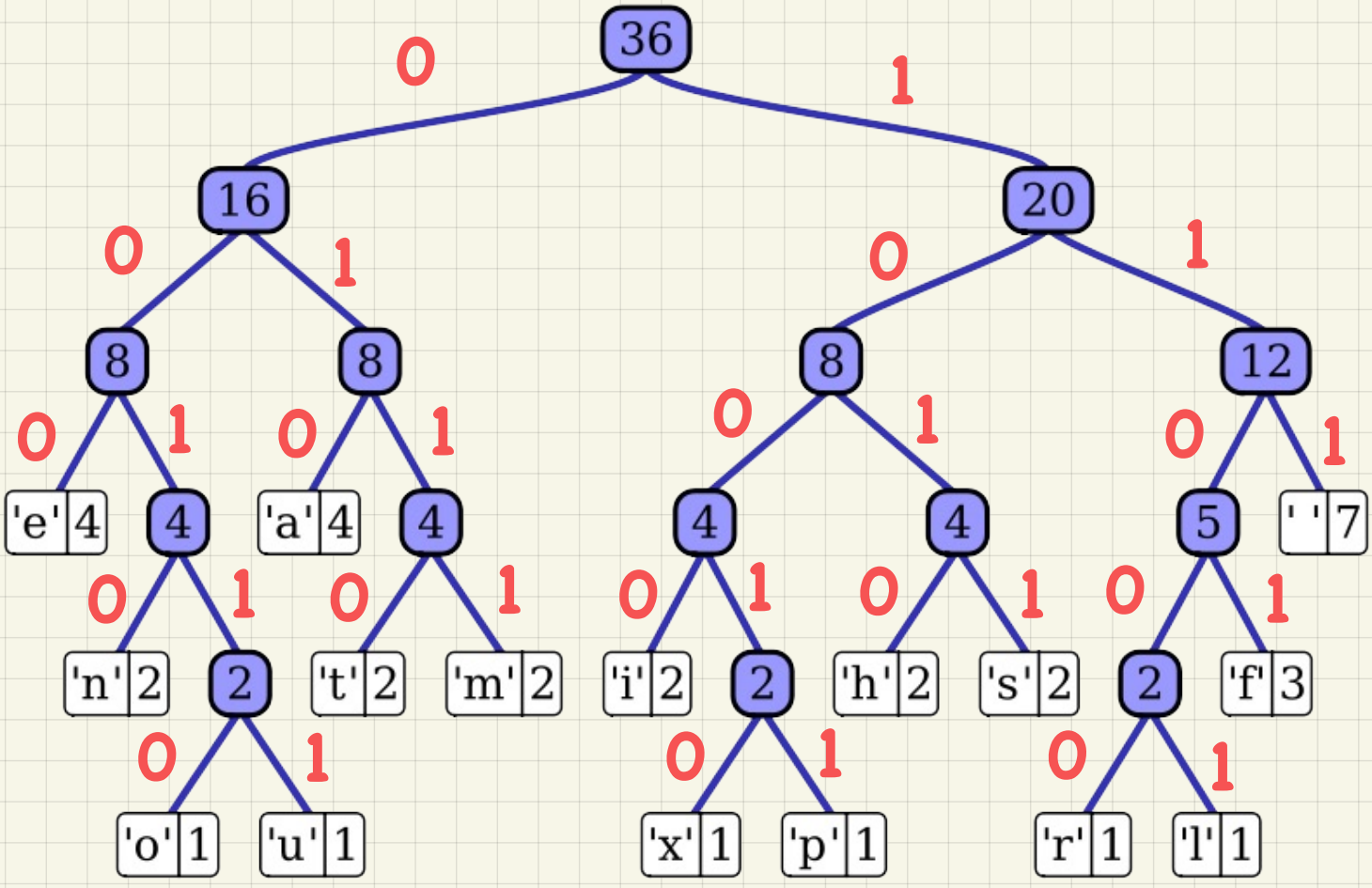
Le message peut alors être codé en binaire, caractère par caractère, en utilisant la table de codage :

01101010100010111111…

Le décodage du message utilise l'arbre de Huffman.

Le principe général :

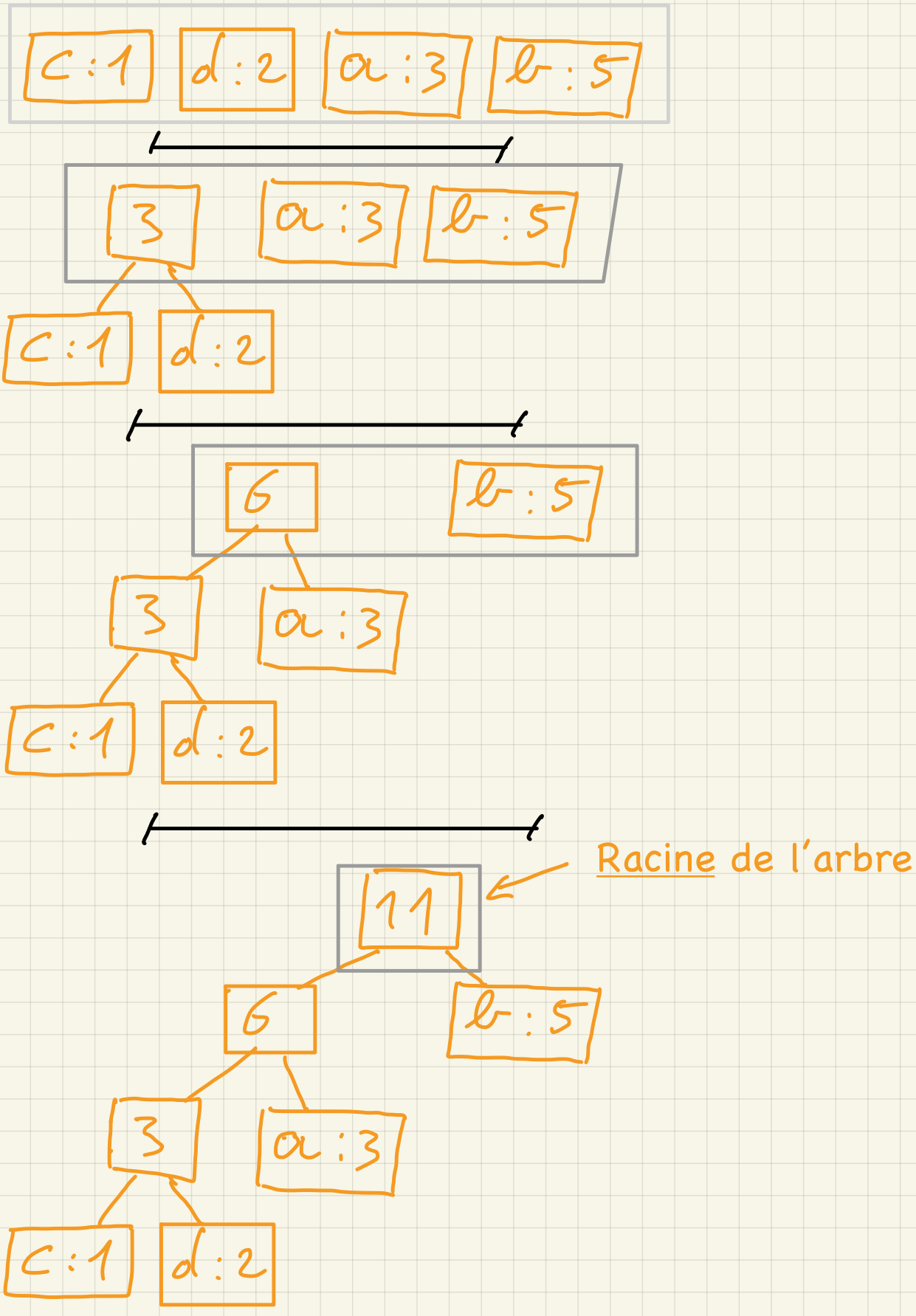
Tant que le message codé n'est pas entièrement décodé faire :
    Partir de la racine de l'arbre de Huffman.
    Tant que l'on n'est pas sur une feuille :
        Lire le prochain bit.
        Si c'est un 0, aller dans le sous-arbre gauche,
        sinon aller dans le sous-arbre droit.
    Retourner le caractère de la feuille atteinte.



01101010100010111111... → this ...

Construction d'un arbre de Huffman.

a a b b a b c b b d d



Voici une description informelle de l'algorithme de création de l'arbre de Huffman à partir du texte (ou fichier) à coder.

+ Partir du texte à coder.
+ Pour chaque caractère c présent dans le texte, compter son nombre d'occurrences. Notons ce nombre n_c (le caractère c est donc présent en n_c exemplaires dans le texte).
+ Pour chaque caractère c présent dans le texte, créer ce que l'on va nommer ici une « cellule » et que l'on va écrire [c : n_c]. Ici n_c sera nommé le « champ numérique de la cellule ».
+ Regrouper toutes ces cellules dans un « ensemble » **S**.

Ceci termine l'initialisation. La construction proprement dite de l'arbre peut débuter.

Tant que S contient au moins deux cellules faire :
  + Notons x et y les deux cellules ayant les plus petits champs numériques (notés n_x et n_y) parmi celles encore présentes dans S.
  + Sortir ces deux cellules x et y de S.
  + Créer une nouvelle cellule NC = [: n_x + n_y] (qui n'a pas de caractère mais qui a un champ numérique dont la valeur est la somme des deux).
  + Ajouter cette nouvelle cellule NC dans S.
  + Les deux cellules x et y deviennent des filles de NC.

Retourner l'arbre dont la racine est l'unique cellule qui reste dans S.

L'arbre de Huffman est utile pour coder et pour décoder

Mais comment le décodeur connait-il l'arbre ?

Il faudrait coder l'arbre à l'intérieur du fichier compressé...

Ainsi, la 1ère étape du décodage reviendrait à décoder et reconstruire l'arbre puis, dans un second temps, de décoder le reste du flux grâce à cet arbre.

Les questions qui se posent sont alors :
+ Comment coder un arbre de Huffman en binaire ? (sa structure et les caractères).
+ Le fichier codé comprendrait ainsi l'arbre et le texte compressé ; Tout cela en binaire. Comment faire la distinction entre les deux dans ce flux binaire ?

Remarquons que dans un arbre de Huffman :
+ Chaque feuille représente un caractère.
+ Un sommet interne ne sert que de « connecteur » (ne représente pas de caractère).
+ Une arête vers un enfant gauche représente 0 et vers un enfant droit représente un 1.
+ Les champs numériques des cellules sont utiles lors de la phase de construction de l'arbre mais ne sont pas du tout utilisés lors du processus de décodage. Ils peuvent donc être ignorés.