Algorithme de Huffman

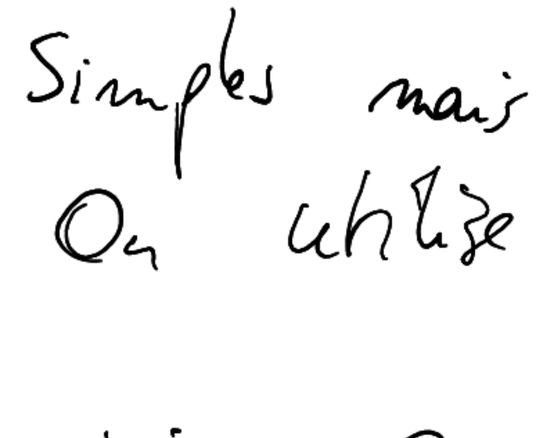
Motivation : Présenter le codage de Huffman et son optimalité pour une sous-catégorie des encodages.
Il est notamment utilisé pour la compression avec des taux élevés sur des données réelles.

On raisonne ici sur un encodage binaire des caractères : chaque caractère du texte initial est représenté par une chaîne binaire unique, appelée code.

Codes de longueur fixe
↳ Simples mais très inefficaces
↳ On utilise des codes de longueur variable.

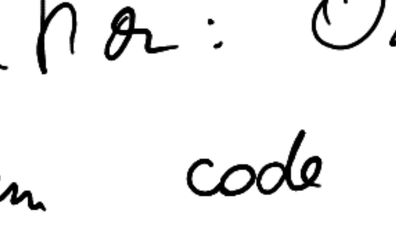
Définition : On dit qu'un codage est préfixe si aucun code n'est préfixe d'un autre code.

Exemples :



est un code non préfixe de longueur variable.

$a \mapsto 0$
$b \mapsto 00$
$c \mapsto 01$
$d \mapsto 1$



est un code préfixe.

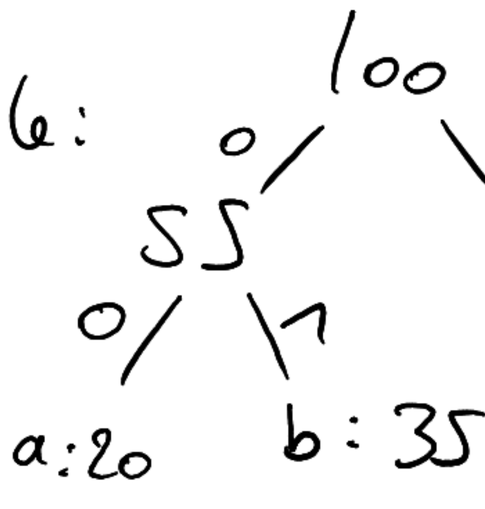
$a \mapsto 00$
$b \mapsto 01$
$c \mapsto 1$

Remarque : Dans l'arbre correspondant à un codage préfixe, tous les caractères sont des feuilles.

Définition : On appelle coût d'un code le nombre de bits nécessaires pour encoder un fichier. A partir de l'arbre T correspondant à C, on a :

$$B(T) = \sum_{a \in \Sigma} f_\Sigma(a) \times d_T(a)$$ où $f_\Sigma(a)$ est le nb d'occurrences de a dans le texte, et $d_T(a)$ la profondeur de a dans T.

Exemple :



$$B(T) = 20 \times 2 + 35 \times 2 + 45 \times 1$$
$$(= 40 + 70 + 45)$$
$$= 155$$

Problème :

Entrée : Σ un alphabet et f_Σ une fonction d'occurrences

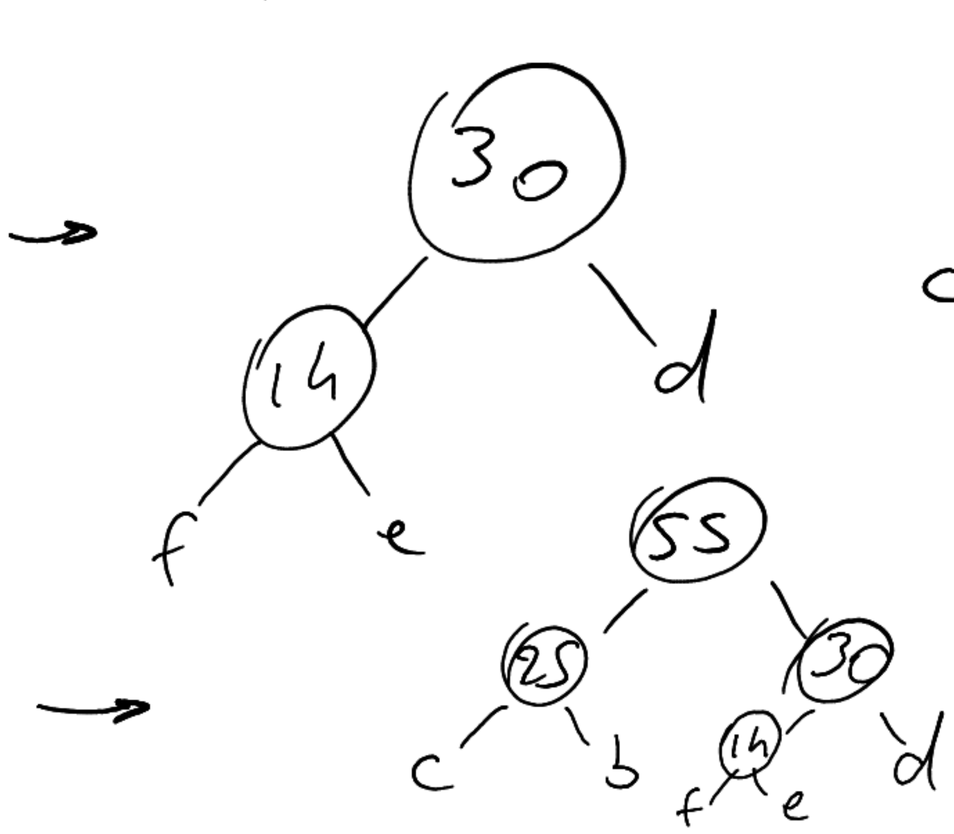
Sortie : C code de coût optimal

```
Fonction huffman(C) : // C ensemble (caractère, nb d'occurrences)
    n = |C|
    Q = creer_tas_min(C)
    Pour i de 1 à n-1
        z = creer_noeud()
        x = extraire_min(Q)
        y = extraire_min(Q)
        z.gauche = x
        z.droit = y
        z.freq = x.freq + y.freq
        inserer(Q, z)
    Retourner extraire_min(Q)   // racine de l'arbre
```

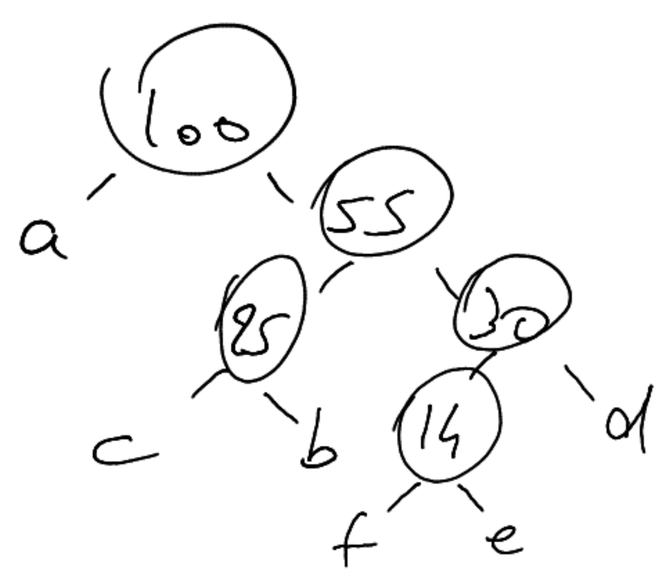
Exemple : [f:5] [e:9] [c:12] [b:13] [d:16] [a:45]

c:12 b:13 d:16 a:45

d:16 a:45



a:45



Terminaison : OK

Complexité : $O(n \log(n))$ avec $n = |\Sigma|$

$O(n \log(\log(n)))$ avec une structure adaptée (arbres de Van Emde Boas)

Correction :

Lemme : Si x et y sont deux caractères de C ayant les fréquences les plus basses, il existe un codage préfixe optimal pour C dans lequel les codes de x et y ont la même longueur et ne diffèrent que par le dernier bit.

Lemme : Avec $\Sigma' = (\Sigma \cup \{z\}) \setminus \{x, y\}$ et $f_{\Sigma'}(z) = f_\Sigma(x) + f_\Sigma(y)$, si T' est un arbre représentant un codage préfixe optimal de C' sur Σ', alors le codage préfixe C de Σ obtenu en remplaçant la feuille z par $\nearrow_x \ _y$ est optimal.

Théorème : L'algorithme de Huffman produit un code préfixe optimal.