# Codes de Huffman

La compression de texte sans perte consiste à coder un texte de sorte que la version codée occupe moins d'octets que le texte d'origine et que le procédé soit réversible, c'est-à-dire que le texte d'origine puisse être recalculé à partir de la version codée. Parmi les algorithmes de compression de texte, on distingue ceux qui nécessitent une analyse statistique préalable de ceux qui peuvent être exécutés à la volée sur n'importe quels textes, comme gzip d'UNIX .

## Les codes de Huffman

D.A. Huffman a mis au point cet algorithme de compression textuelle en 1952. Il est fondé sur une analyse statistique effectuée au préalable sur un texte ou sur sa langue. En effet, pour chaque caractère susceptible d'apparaître dans un texte écrit en français par exemple, il faut connaître a priori sa fréquence d'apparition. Parmi les algorithmes de compression de texte qui utilisent une méthode statistique, le codage de Huffman est optimal

## Préliminaires

1) Visitez le fichier huffman.py dans lequel on va programmer l'intégralité de ce TP.
2) Notez qu'il contient les statistiques (probabilités d'apparition) relatives à un certain jeu de caractères. On limite ainsi notre alphabet aux 26 lettres minuscules allant de 'a' à 'z' et à l'espace ' '. Le dictionnaire des fréquences est prêt à être calculé.
3) Une classe Arbre est fournie dans une version minimale mais suffisante pour construire des arbres de Huffman.

## Exercice 1. Construction de l'arbre de Huffman

Le but de cet exercice est de construire l'arbre de Huffman selon le principe suivant. Au début, il y a autant d'arbres que de caractères. On sélectionne les 2 arbres de moindres probabilités et on les assemble en un seul en mettant à jour les probabilités. On répète ce principe jusqu'à l'obtention d'un arbre unique.
Tout ce travail va être grandement facilité par l'importation du module heapq. Ce module permet d'implémenter la structure de tas binaire.

1) Commencez par lire la documentation P YTHON relative au module heapq. Comprenez notamment heapify(), heappop() et heappush().
2) Initialisez le tas à raison d'un triplet par case. Chaque triplet est de la forme :

(proba,etiquette,arbre)
Pour l'instant, les arbres sont réduits à des feuilles, les probas sont celles d'origne et les étiquettes tiennent en une seule lettre.
3) Utiliser la structure de tas pour extraire les triplets relatifs aux 2 arbres de moindres probabilité une fois que le tas est constitué.
4) Assemblez ces 2 arbres en une seul, mettez à jour le nouveau triplet et ajoutez-le au tas.
5) Quand le tas n'a plus qu'un élément, on peut extraire l'arbre de Huffman.

**Exercice 2. Construction du dictionnaire**
On se propose de construire le dictionnaire qui à chaque caractère associe son mot de code, directement déduit de l'arbre de Huffman. La fonction code-Huffman() déjà écrite suggère que c'est la sous-fonction parcours() qui va faire le travail.
1) Comprenez la fonction code-Huffman().
2) La fonction parcours() fait un parcours infixe classique de l'arbre. Sur une feuille, elle est à même d'associer son code au caractère stocké dans la feuille. Commencez par écrire ce cas d'arrêt.
3) Complétez le reste de la fonction parcours() qui mémorise, selon que l'on descende à gauche ou à droite dans l'arbre, le chemin pris dans la variable préfixe.
4) Testez le tout.

**Exercice 3 : Codage du texte**
A présent que nous disposons de fonctions pour construire et l'arbre de Huffman et le dictionnaire afférent, nous pouvons passer au codage du texte qui est ici une compression.
1) Le texte à encoder a été mis dans la variable texte. Il reste à le lire caractère par caractère et à remplacer chaque caractère connu par son code. On affectera le code de l'espace à tout caractère inconnu pour ne pas compliquer le décodage.
2) Testez votre fonction d'encodage sur le fichier texte fourni.
3) On pourrait se livrer au calcul de la longueur moyenne des mots du code ainsi qu'à celui de l'entropie pour évaluer la qualité de compression. En pratique, on peut aussi se demander : quel est le taux de compression obtenu ?

**Exercice 4 : Pas de codage sans décodage ...**
Comment décompresser un texte compressé par un codage de Huffman donné ?

1) Remplissez la fonction decodage().
2) Décodez ou décompressez le texte précédemment compressé.