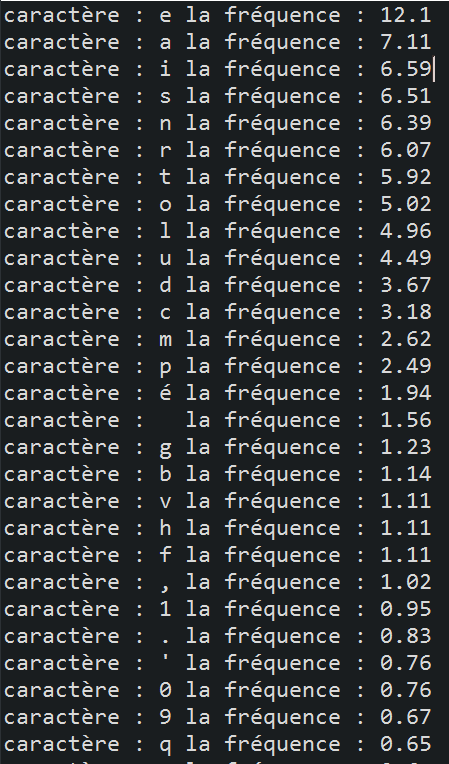
**Huffman statique :**

Dans le codage de Huffman statique, chaque octet a un code prédéfini par des statistiques faites préalablement sur des textes qui ont un nombre de caractères énormes. Le problème de ce type de codage est qu'il ne peut s'effectuer que sur un seul type de fichier (ex. : un texte en français, où les fréquences d'apparition du 'e' sont généralement énormes ; celui-ci aura donc un code très court).

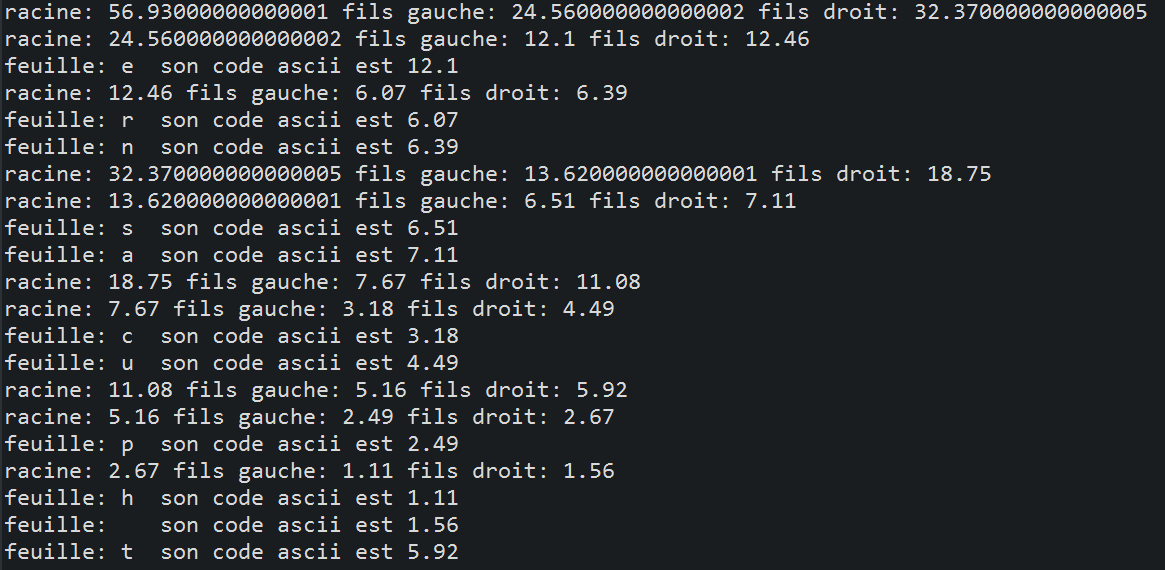
Le corpus de « Wikipédia en français », en 2008, a été segmenté en mots par le laboratoire CLLE-ERSS qui a ensuite recensé les occurrences de ces derniers (à peu près 900 000 000 caractère), permettant ainsi le calcul de la fréquence des caractères dont on va se servir dans notre code.

**Initialement**, on va créer un tableau qui contient 84 caractères et leurs fréquences respectives selon le labo CLLE-ERSS. Voici les premiers caractères qui apparaissent dans le tableau.

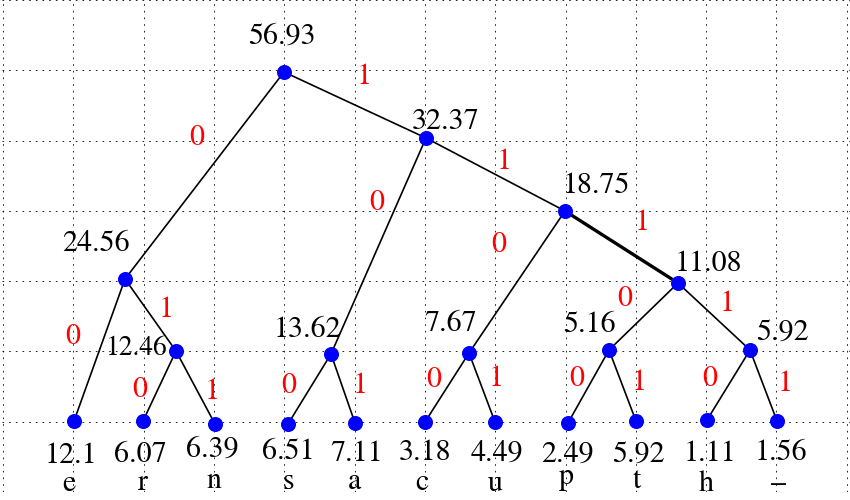
**Ensuite**, on va parcourir le texte donné en input pour reconnaitre les différents caractères dont on aura besoin, sans prêter de l'attention à leur occurrence, car dans le cas de Huffman statique on va se baser sur les fréquences déjà stockées dans le tableau précèdent.

On va générer un tableau dont l’indice est le code ASCII du caractère et sa valeur est la fréquence. Ce tableau nous sera ainsi utile pour construire un arbre binaire. Cet arbre est créé de la manière suivante, premièrement on initialise l’arbre par des feuilles dont la valeur est le caractère et le poids est sa fréquence. Ensuite, on associe chaque fois les deux nœuds de plus faibles poids, pour donner un nouveau nœud dont le poids équivaut à la somme des poids de ses fils. On réitère ce processus jusqu'à n'en avoir plus qu'un seul nœud : la racine. On associe ensuite par exemple le code 0 à chaque embranchement partant vers la gauche et le code 1 vers la droite.

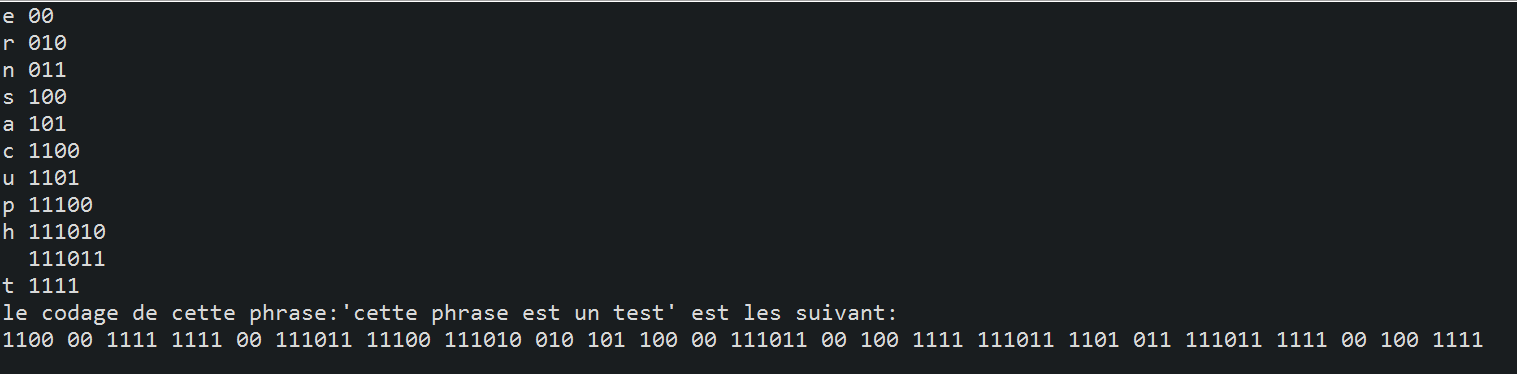
On prend par exemple la phrase « cette phrase est un test ». Son arbre se présentera comme suit :



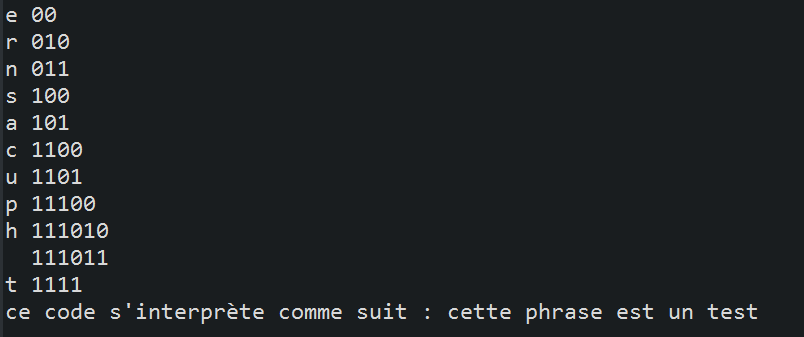
Maintenant, partant de la racine, on associe à chaque fils gauche un « 0 » et à chaque fils droit un « 1 ». Pour mieux comprendre, voici une représentation de l’arbre que j’ai réalisé à l’aide d’un outil appelé géophar :



Pour obtenir le [code binaire](https://fr.wikipedia.org/wiki/Syst%C3%A8me_binaire) de chaque caractère, on remonte l'arbre à partir de la racine jusqu'aux feuilles en rajoutant à chaque fois au code un 0 ou un 1 selon la branche suivie.



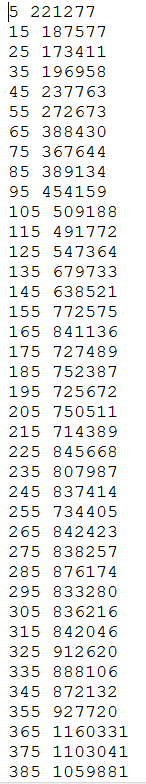
Quant au décodage il faut que la personne qui décode ait la même liste des fréquences et procède au même algorithme pour construire l’arbre. Ensuite il peut générer un tableau qui contient les caractères et leurs codes. C’est ainsi qu’il peut déchiffrer le code reçu.



En ce qui concerne la complexité, on a créé une fonction qui prend en input la longueur du mot souhaité et génère ensuite un mot qui se compose de caractères aléatoires et donne ensuite son code binaire et le temps d’exécution du code. On répète cette opération 50 fois pour la longueur du mot pour ainsi calculer la moyenne des temps d’exécution. Dans un tableau qui doit contenir la longueur du mot et son temps d’exécution, on stocke cette moyenne calculée. Ensuite on augmente le nombre de caractères et on répète 50fois encore, et on procède ainsi 100fois.

Le tableau qu’on obtient à la fin va être utile pour tracer la courbe de complexité.

Extrait du tableau :



La courbe de complexité :