**Solution Naïve :**

Le problème qui se pose est de pouvoir transformer un texte donné en une série de 0 et 1 que l’ordinateur peut comprendre. La première solution qui saute aux yeux est d’associer à chaque caractère un code binaire de 8bits (avec 8bits on peut coder 28 = 256 qui est un nombre suffisant pour représenter les caractères spéciaux de plusieurs langues).

Pour que l’ordinateur nous génère une liste des lettres qui existent dans le texte donné en input et les codes binaires correspondants, on va créer un arbre binaire parfaitde profondeur = 8ie un arbre binaire strict (dont tous les nœuds possèdent zéro ou deux fils) dans lequel toutes les feuilles sont à la même profondeur.

**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 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.

Pour cela on va parcourir le texte donné en input pour reconnaitre les différents caractères dont on aura besoin, sans prêter de l'attention à leurs occurrence, car dans le cas de Huffman statique on va se baser sur les fréquences déjà données par le laboratoire CLLE-ERSS (par exemple) et qu’on va stocker dans une liste.

Ensuite on va générer un tableau dont l’indice est le code ASCII du caractère et sa valeur est la fréquence.

On va se servir de ce tableau pour ainsi construire un arbre binaire. Cet arbre est créé de la manière suivante, 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.

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 le décodeur 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.