**Huffman :**

Le codage de Huffman est un algorithme de compression de données, il utilise un code à longueur variable pour représenter un symbole de la source. Il existe trois variantes de l'algorithme de Huffman (statique, semi-adaptatif, adaptatif), chacune d'elles définissant une méthode pour la création de l'arbre, pour le moment on s’intéresse par la méthode semi-adaptatif.

L’algorithme de Huffman a pour but la transformation d’un message en un mot codé (ou message codé) composé d’un ensemble de codes binaires tel que le mot codé soit composé du moins de bits possibles. Pour ce faire, il convient d’attribuer à chaque caractère distinct du message un code binaire qui le représentera de façon à ce que les caractères les plus fréquents soient remplacés par les codes les plus courts.

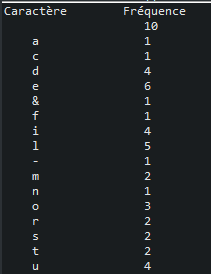
Alors, l’encodage de Huffman semi-adaptatif nécessite une connaissance à priori de la probabilité d’apparition des symboles. Un texte par exemple à coder est tout d’abord lu intégralement de façon à construire l’alphabet et les fréquences d’apparition des éléments de l’alphabet. Le principe repose sur la création d'une structure d'arbre composée de nœuds à partir des poids de chaque alphabet.

Cela doit également se faire en évitant d’introduire des « préfixes » dans nos codes, ce qui signifie qu’il ne doit jamais y avoir d’ambiguïté lors du décodage. Si un ensemble de bits correspond à un code binaire connu, alors aucun autre code ne devrait commencer par ce même ensemble. Par exemple, si « 110 » est présent alors cela rend impossible l’utilisation d’un code binaire qui commencerait par ce préfixe, comme « 1101 », il serait impossible de savoir où s’arrêter lors du décodage si je reçois « 1101101... ».

**Encodage :**

L'algorithme de Huffman génére un code-préfixe à taille variable, à partir de la table des fréquences d'une suite de valeurs, on va construire un programme qui permet de compresser et décompresser un texte par suivre les étapes au-dessous :

**Etap1 : Détermination de l’alphabet et des fréquences.**

Prenons par exemple le message suivant : « sud-cloud & iot est la meilleure filière du monde ». Il est constitué de 50 caractères, nous pouvons le décomposer en un tableau de fréquence par caractère :

Pour garder l’exemple simple, les majuscules et les minuscules sont confondues, les accents ne sont pas comptés pour le moment. Mais dans un cas réel tout va être comptés.

**Etape3 : Créer une feuille pour chaque caractère avec la fréquence associée.**

Un arbre binaire est une structure de données composée de nœuds, permettant de stocker une collection de données de même type. Lesquels sont tous liés à un nœud parent et deux fils au maximum (un fils droit et gauche), à l’exception du nœud racine. Un nœud racine est un nœud qui n’a pas de parent et qui est situé tout en haut de l’arbre. Si un nœud ne possède pas de fils, alors ce nœud s’appelle une feuille.

Dans cette étape on va construire à l’aide du tableau les feuilles de l’arbre contenant les caractères :

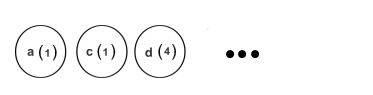


Figure : Initialisation de l’arbre

**Etape4 : Construire l'arbre et retourne la racine.**

Pour chaque itération suivante il faut sélectionner les deux nœuds de plus petites fréquences f1 et f2, pour créer un nouveau nœud de fréquence égal à f1 + f2 et lui affecter les deux nœuds sélectionnés comme fils gauche et droite successivement, jusqu’à obtention de la racine.

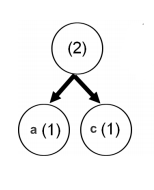
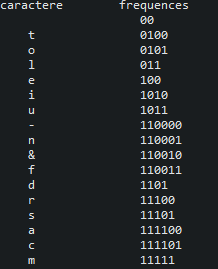


Figure : groupement des nœuds de plus faibles valeurs

**Etape5 : Parcourir l'arbre pour générer le code pour chaque lettre.**

L’arbre est utilisé pour déterminer le code binaire de tous ses nœuds. Pour ce faire, le code est formé en parcourant l’arbre de haut en bas et en ajoutant une valeur binaire à chaque fois que nous passons sur un nœud. Soit la valeur 0 pour les nœuds à gauche et la valeur 1 pour ceux de droite.

Deux facteurs ont une grande influence sur le placement des éléments et donc sur la taille minimale du code.

Le premier est que quand le nombre de lettres différentes augmente, l’arbre s’allonge, déplaçant les lettres les plus fréquentes de plus en plus loin de la racine et créant par la même occasion de plus en plus d’éléments qui ne sont pas des lettres.

Le deuxième facteur est le nombre d’occurrences des lettres les plus fréquentes : si ce nombre est très grand, la lettre aura tendance à rester dans le haut de l’arbre .

**Etape6 : parcourir le texte pour créer l’encodage.**

Le résultat a la particularité de ne pas avoir de séparateur entre les codes de chaque lettre. Une fois l’encodage terminé, le tableau des fréquences est transmis avec le message encodé afin que le receveur puisse procéder au décodage .

**Décodage :**

Le décodage se fait après avoir récupéré le tableau des fréquences qui a été utilisé pour l’encodage. Il suffit ensuite de reconstruire l’arbre et parcourir tout le message encodé et à chaque fois qu’une suite de bits est retrouvée en entier elle est instantanément remplacée par le caractère correspondant.

**Complexité :**

