Chapitre 2 : Le codage du son

1. La numérisation du son

➔ Voir le fonctionnement du C.A.N.

On peut jouer sur la **période d'échantillonnage** ou la **fréquence d'échantillonnage**.

On peut également jouer sur le **nombre de bits N** du C.A.N ou r**ésolution** en modifiant le **pas de conversion** ou **quantification**.

On montre ainsi que le pas de conversion est donné par :

$$p = \frac{\text{plage de conversion}}{2^N}$$

➔ Voir la conversion d'un nombre décimal en nombre binaire ; l'algorithme consiste à faire des divisions euclidiennes par 2 aussi longtemps que le quotient demeure strictement positif.

Le théorème d'échantillonnage, dit aussi théorème de Shannon, énonce que l'échantillonnage d'un signal exige un nombre d'échantillons par unité de temps supérieur au double de l'écart entre les fréquences minimale et maximale qu'il contient.

Les fréquences audibles étant comprises entre 20 Hz et 20 kHz, on attend une fréquence d'échantillonnage appropriée voisine de 40 kHz.

2. La compression d'un fichier audio

➔ Le MP3 a incontestablement rendu la musique plus portable et « partagée ». Les fichiers au format .mp3 ne prennent pas autant de mémoire que ceux d'un CD pour une qualité audio qui s'en approche : 700 Mo d'un CD audio peuvent être réduits en .mp3 à 63 Mo, ce qui correspond à un taux de compression τ défini par :

$$\tau = \frac{\text{taille du fichier après compression}}{\text{taille du fichier avant compression}} = \frac{63}{700} = 0.09 = 9\%$$

Mais comment réduit-on ainsi la taille du fichier ?

La compression MP3 s'effectue « avec perte d'informations », ce qui signifie qu'à partir d'un fichier compressé, il n'est plus possible de retrouver le fichier d'origine. Des informations sonores sont simplement enlevées et perdues à jamais. L'algorithme mathématique qui convertit les fichiers .wav (format du CD) en .mp3 repose sur certaines méthodes qui prennent en compte des principes de l'audition humaine appelés « psychoacoustiques ».

• L'oreille humaine (ou plutôt le couple oreille-cerveau auditif) entend certains sons mieux que d'autres de sorte que toutes les fréquences supérieures à 15,5 kHz sont définitivement supprimées des fichiers .mp3 et celles comprises entre 1 et 4 kHz, auxquelles l'oreille humaine est le plus sensible, sont légèrement accentuées.

• L'oreille humaine sait mieux de quelle direction provient un son aigu (comme celui d'une sirène d'ambulance), qu'un son grave. Donc, pour tirer parti de ce fait et économiser de la mémoire, l'algorithme MP3 réduit les informations stéréo des sons de basse fréquence en informations mono.

• L'oreille humaine a du mal à entendre un son dit « masqué », c'est-à-dire couvert par des sons plus intenses. L'algorithme MP3 réduit les informations correspondantes.

• L’oreille humaine peut ne pas percevoir des sons séparés d’un court laps de temps et les confond alors en un seul. Le MP3 supprime donc les informations relatives à un son « voisin » d’un autre à tel point que les sons de réverbération ou de cymbale seront reproduits différemment pour un MP3 et pour un CD.

Nous pouvons ainsi réaliser une compression de données numériques par suppression psychoacoustique ici et là sans dégrader beaucoup la fidélité.

➔ Le codage de Huffman est une technique de compression sans perte qui utilise un arbre binaire pour représenter les données de manière efficace. Voici une explication détaillée de son fonctionnement :

Étapes du Codage de Huffman

1. **Calcul des Fréquences des Symboles** :
    - Comptez la fréquence d'apparition de chaque symbole dans le texte à compresser.
2. **Construction de l'Arbre de Huffman** :
    - Créez un nœud feuille pour chaque symbole avec sa fréquence associée.
    - Ajoutez tous les nœuds dans une file de priorité (ou un tas), ordonnée par fréquence (les nœuds avec les fréquences les plus basses sont en tête).
    - Répétez les étapes suivantes jusqu'à ce qu'il ne reste plus qu'un seul nœud dans la file :
        - Retirez les deux nœuds avec les plus basses fréquences de la file.
        - Créez un nouveau nœud interne avec ces deux nœuds comme enfants et une fréquence égale à la somme de leurs fréquences.
        - Ajoutez ce nouveau nœud dans la file.
    - Le dernier nœud restant est la racine de l'arbre de Huffman.
3. **Génération des Codes de Huffman** :
    - Parcourez l'arbre de Huffman à partir de la racine pour assigner des codes binaires à chaque symbole.
    - À chaque nœud interne, assignez "0" à la branche gauche et "1" à la branche droite (ou vice versa).
    - Les codes binaires pour chaque symbole sont obtenus en suivant les branches de la racine aux feuilles.
4. **Encodage des Données** :
    - Remplacez chaque symbole du texte par son code binaire correspondant pour obtenir la séquence compressée.

Exemple Illustratif

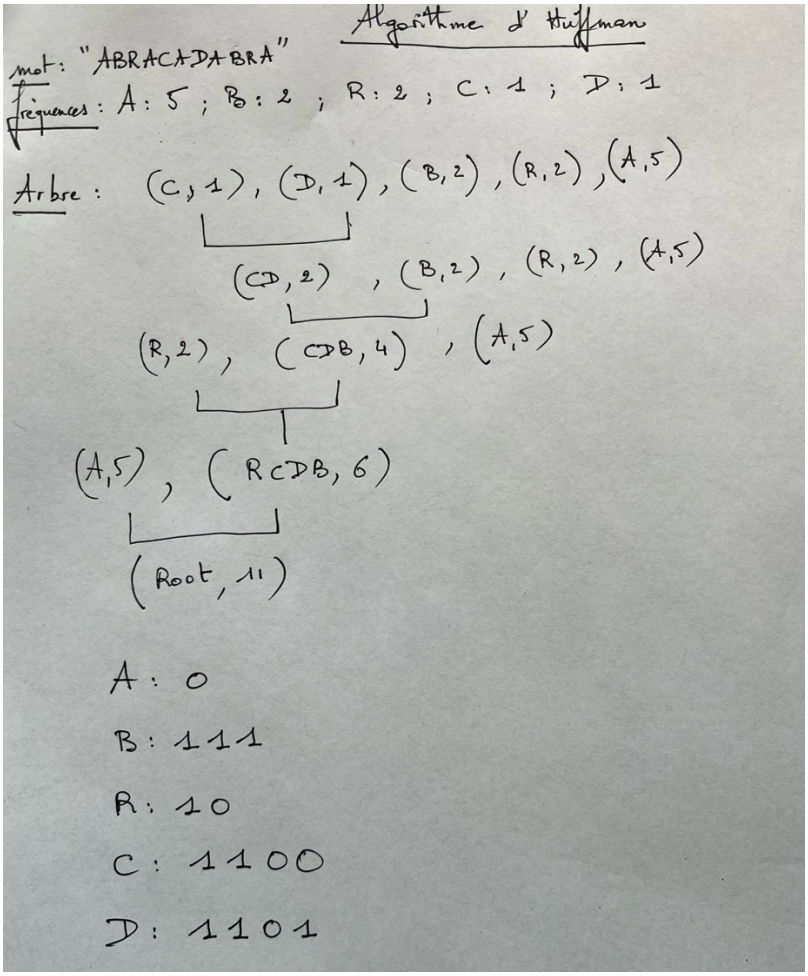
Supposons que nous voulons compresser la chaîne "ABRACADABRA".

1. **Calcul des Fréquences** :
    - A: 5, B: 2, R: 2, C: 1, D: 1
2. **Construction de l'Arbre** :
    - Créez les nœuds pour chaque symbole :
        - A: 5, B: 2, R: 2, C: 1, D: 1
    - Insérez-les dans une file de priorité :
        - [(C, 1), (D, 1), (B, 2), (R, 2), (A, 5)]
    - Combinez les deux nœuds avec les fréquences les plus basses :
        - Combinez (C, 1) et (D, 1) en un nœud interne (CD, 2)
        - [(CD, 2), (B, 2), (R, 2), (A, 5)]
    - Répétez ce processus :
        - Combinez (CD, 2) et (B, 2) en un nœud (CDB, 4)
        - [(R, 2), (CDB, 4), (A, 5)]
        - Combinez (R, 2) et (CDB, 4) en un nœud (RCDB, 6)
        - [(A, 5), (RCDB, 6)]
        - Combinez (A, 5) et (RCDB, 6) en un nœud (Root, 11)

3. **Génération des Codes** :
    - Parcourez l'arbre de la racine aux feuilles pour générer les codes :
        - A: 0
        - R: 10
        - C: 1100
        - D: 1101
        - B: 111
4. **Encodage des Données** :
    - Remplacez chaque symbole par son code :
        - "ABRACADABRA" devient "01111010110011000111110010"

Conclusion

Le codage de Huffman est efficace car il utilise des codes plus courts pour les symboles les plus fréquents et des codes plus longs pour les symboles moins fréquents, ce qui minimise la longueur totale du texte encodé. Il est largement utilisé dans diverses applications de compression de données, y compris les formats de fichiers tels que JPEG (image) et MP3.



Exemple de Décodage Pas à Pas

Partons du texte encodé `01111001100011010011100` et décomposons-le bit par bit.

- **0** : Va à l'enfant gauche → Trouve `A` → Texte décodé : `A`
- **111** : Droite, Droite, Droite → Trouve `B` → Texte décodé : `AB`
- **10** : Droite, Gauche → Trouve `R` → Texte décodé : `ABR`
- **0** : Gauche → Trouve `A` → Texte décodé : `ABRA`
- **1100** : Droite, Droite, Gauche, Gauche → Trouve `C` → Texte décodé : `ABRAC`

- **0** : Gauche → Trouve `A` → Texte décodé : `ABRACA`
- **1101** : Droite, Droite, Gauche, Droite → Trouve `D` → Texte décodé : `ABRACAD`
- **0** : Gauche → Trouve `A` → Texte décodé : `ABRACADA`
- **111** : Droite, Droite, Droite → Trouve `B` → Texte décodé : `ABRACADAB`
- **10** : Droite, Gauche → Trouve `R` → Texte décodé : `ABRACADABR`
- **0** : Gauche → Trouve `A` → Texte décodé : `ABRACADABRA`