

Compression : Construction du dictionnaire (Déterminer le code Huffman)


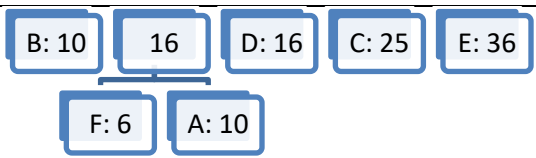
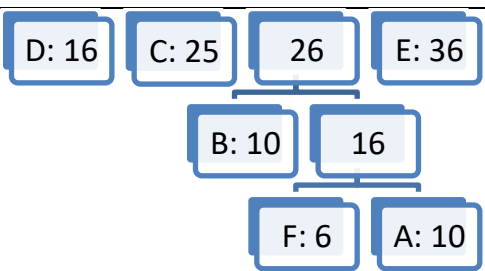
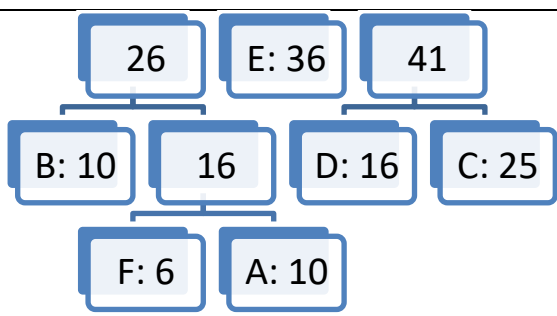
1) Recherche du nombre d'occurrences de chaque symbole (caractère – ici 1 octet)

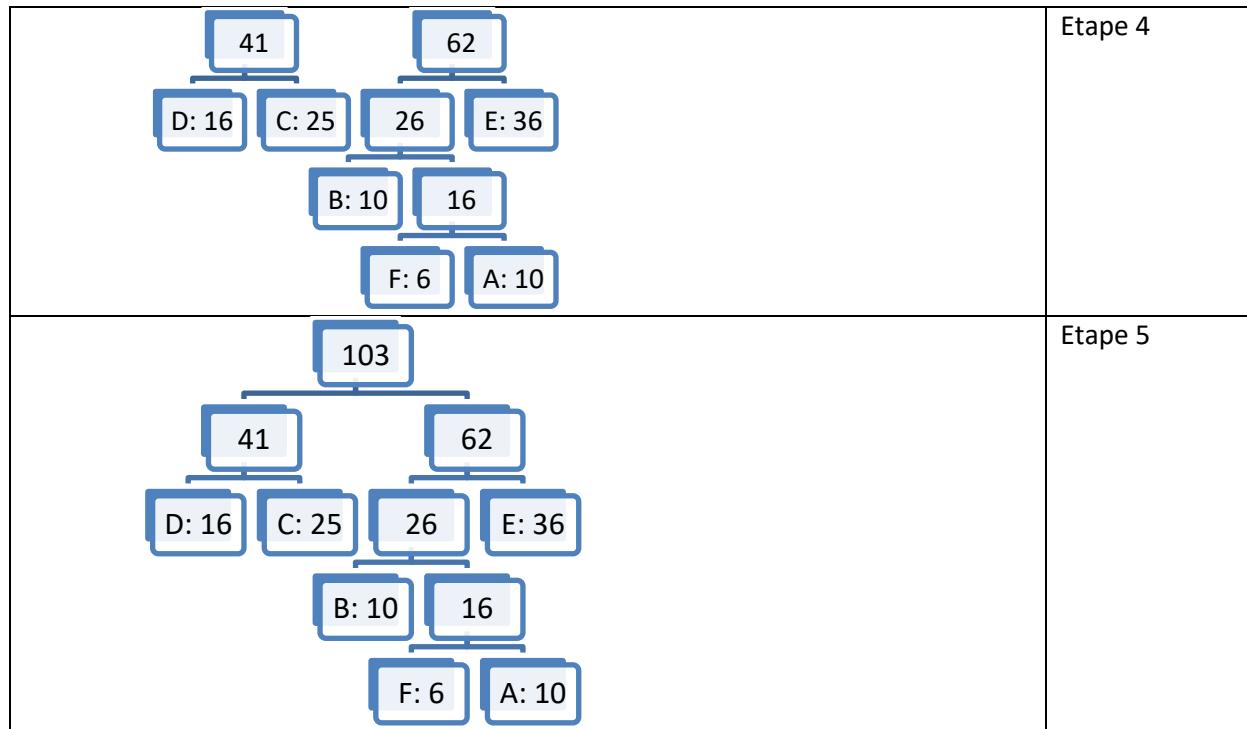
⇒ Création d'une table de fréquences :

A : 10	B : 10	C : 25	D : 16	E : 36	F : 6
--------	--------	--------	--------	--------	-------

2) Création d'un arbre binaire :

- ⇒ Chaque élément de la table de fréquence est une feuille terminale de l'arbre
- ⇒ Chaque nœud de l'arbre est la somme des fréquences des feuilles ou des nœuds fils
- ⇒ On commence avec une forêt, la forêt finale est composée d'un unique arbre.
- ⇒ Algorithme « greedy » :
 - On sélectionne les arbres de poids minimaux (soit n1 et n2), et les remplace par l'arbre binaire pondéré d'enfants n1 et n2.
 - Tant qu'il y a plusieurs arbres on répète l'opération précédente

	Etape 0
	Etape 1
	Etape 2
	Etape 3



3) Construction du code Huffman

- Le code d'un symbole est déterminé par le chemin depuis la racine de l'arbre jusqu'à la feuille associée au symbole.
- Concaténation d'un « 0 » lorsque le chemin emprunte une branche à gauche
- Concaténation d'un « 1 » lorsque le chemin emprunte une branche à droite

A : 1011	B : 100	C : 01	D : 00	E : 11	F : 1010
----------	---------	--------	--------	--------	----------

Compression : Remplacer les symboles par leur code Huffman

- Trouver une structure de donnée efficace pour manipuler des bits
- Le résultat doit être stocké dans un tableau d'octets (byte[])

Décompression : Remplacer le code Huffman par les symboles

- Trier le dictionnaire par code Huffman
- La lecture des bits des données compressées se fait par bit. Dès que l'on rencontre un code Huffman connu, celui-ci est remplacé par son symbole.
- Stocker le résultat dans un tableau d'octets