Plateforme .NET – ESIEE 2012

Langage C# - Huffman tree

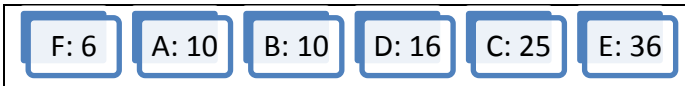
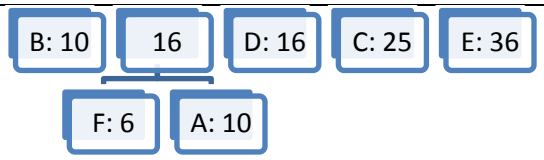
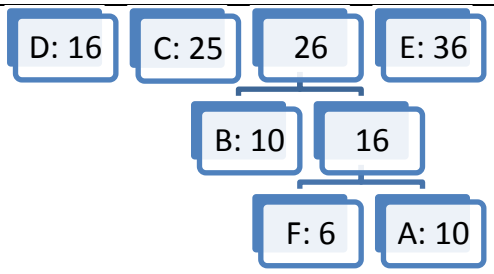
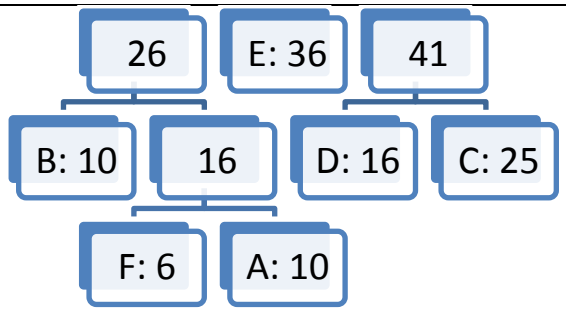
Key Consulting / Olivier Mathieu (olivier.mathieu@keyconsulting.fr)

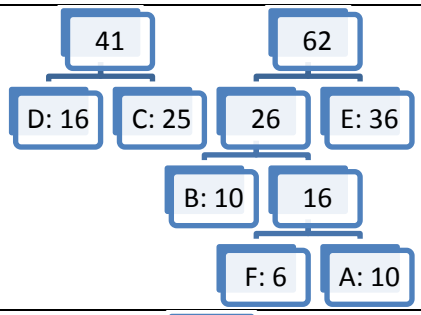
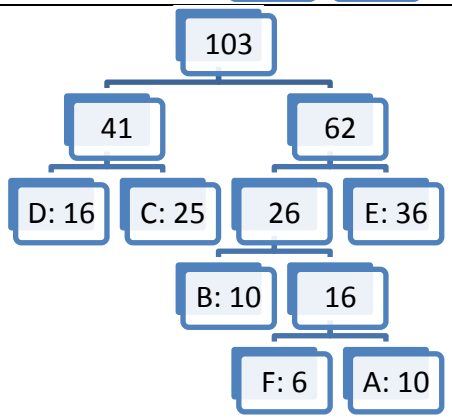
Compression : Construction du dictionnaire (Déterminer le code Huffman)

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 » :
     a. On sélectionne les arbres de poids minimaux (soit n1 et n2), et les remplace par l'arbre binaire pondéré d'enfants n1 et n2.
     b. Tant qu'il y a plusieurs arbres on répète l'opération précédente

	Etape 0
	Etape 1
	Etape 2
	Etape 3

	Etape 4
	Etape 5

3) Construction du code Huffman
    a. Le code d'un symbole est déterminé par le chemin depuis la racine de l'arbre jusqu'à la feuille associée au symbole.
    b. Concaténation d'un « 0 » lorsque le chemin emprunte une branche à gauche
    c. 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

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

Décompression : Remplacer le code Huffman par les symboles

1) Trier le dictionnaire par code Huffman
2) 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.
3) Stocker le résultat dans un tableau d'octets