TP Compression : Codage de Huffman

L'objectif de ce TP est d'une part de mettre en évidence l'intérêt en terme de compression de données du codage de Huffman pour des données réelles et d'autre part de comprendre la méthodologie de développement du codeur. L'outil de développement et d'implémentation est Matlab. L'information que l'on souhaite compresser est le contenu d'un message (une succession de symboles). On considérera que ces symboles sont codés sur 8 bits (valeurs entre 0 et 255). Par exemples, pour un texte, les codes ASCII sont obtenus en convertissant les caractères en nombres entiers (fonction double sous Matlab), pour une image l'information correspond au valeurs de niveaux de gris de chaque pixel.
La méthode de compression / décompression par le codage de Huffman comprend 5 grandes étapes (fonctions Matlab):

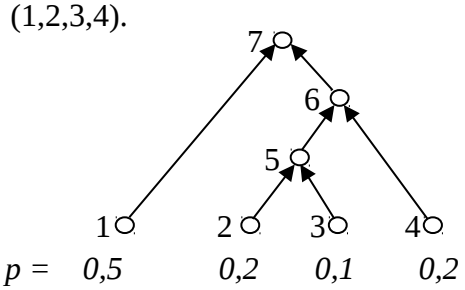
1. calcul de l'entropie : pour un message donné, la fonction renvoie les valeurs de l'histogramme des symboles (le vecteur des probabilités ou des fréquences d'apparition). En plus, pour information, elle retournera la valeur de l'entropie du message et la redondance
2. construction de l'arbre de Huffman en fonction fréquences d'apparition des symboles: la fonction renvoie la matrice d'adjacence du graphe et les noms des nœuds : à chaque niveau, '0' pour le moins fréquent et '1' pour le suivant (ou l'inverse)
3. dictionnaire : la fonction parcourt l'arbre et construit le dictionnaire de symboles en associant les noms des nœuds rencontrés sur chaque chemin
4. codage de l'information entrante et renvoi d'une suite binaire de 0 et 1
5. décodage

Exemple : Soit la suite de caractères 'ABRACARABA'. Les fréquences d'apparition (probabilités) de chaque caractère sont telles que p(A)=0,5 ; p(B)=0,2 ; p(C)=0,1 ; p(R)=0,2

1. entropie : cette fonction doit prendre en variable d'entrée l'information (la succession de symboles) sous forme d'un vecteur d'entiers. La fonction doit renvoyer 4 variables de sortie :
   - un vecteur contenant les codes d'origine des symboles existants (entre 0 et 255)
   - un vecteur des fréquences d'apparition (nb. d'éléments égal au nombre de symboles différents apparaissant dans l'entrée),
   - l'entropie de l'information d'entrée
   - la redondance du codage

   fonctions matlab utilisées :size, hist, sum, log2, find
2. arbre : la fonction prend en entrée le vecteur des probabilités (fréquences d'apparition) des symboles et renvoie la matrice d'adjacence de l'arbre de Huffman (graphe orienté en partant des feuilles vers la racine, donc dans le sens de parcours pour le codage) et un vecteur de noms associés à chaque nœud du graphe..

Sur l'exemple de la figure 1 la matrice d'adjacence associée se lit tel que : chaque sommet possède un seul successeur sauf la racine et chaque sommet a 2 prédécesseurs sauf les feuilles (1,2,3,4).



	1	*2*	*3*	*4*	*5*	*6*	*7*
1	0	0	0	0	0	0	1
2	0	0	0	0	1	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	0	1	0
5	0	0	0	0	0	1	0
6	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0

Figure 1

Proposition d'algorithme :

i. initialisation d'une matrice nulle de taille N x N, avec N le nombre de symboles ; cette matrice code les sommets source du graphe (les feuilles 1 – 4 dans l'exemple précédent)

ii. création d'un nouveau sommet (appelé *k* par exemple) en associant les deux feuilles *i* et *j* de plus faibles probabilités ; ce sommet, de probabilité *p(k) = p(i) +p(j)*, aura 2 prédécesseurs (*i* et *j*), ce qui se traduira par des éléments (*i*, *k*) et (*j*, *k*) égaux à 1 dans le tableau. Dans la structure de l'arbre, intuitivement et par la suite, le sommet *k* deviendra une « feuille » qui remplacera les deux feuilles qui le précèdent (Ex : *i=2, j=3, k=5 et p(5)=p(2)+p(3)*)

iii. attribution de noms : les deux feuilles *i* et *j* seront appelée 0 et 1. En pratique, on créera un vecteur binaire *noms* qui contiendra 0 et 1 dans les éléments *i* et *j*.

iv. les points ii. et iii. seront itérés pour les nouvelles «feuilles» de l'arbre (les feuilles de l'itération précédente, sauf *i* et *j*, qui sont remplacés par *k*) (Ex : *i=4, j=5, k=6* pour l'itération 2, avec *noms*(4)=0 et *noms*(5)=1, *i=1, j=6, k=7* pour l'itération 3, avec *noms*(1)=0 et *noms*(6)=1).

v. l'algorithme s'arrête quand on atteint la racine (de probabilité 1).

Pour l'exemple de la figure 1, la matrice d'adjacence obtenue est celle du tableau, et le vecteur de noms devrait être

	1	*2*	*3*	*4*	*5*	*6*	*7*
noms	1	1	0	0	1	0	

fonctions matlab utilisées :sıze, sort, sparse

3. dictionnaire : la fonction prend en entrée la matrice d'adjacence et le vecteur de noms des sommets et renvoie le dictionnaire (sous forme de « cell array » : un vecteur de vecteurs de longueur différente pour chaque symbole, dont les éléments sont des 1 et 0,).
   Proposition d'algorithme :
   i. parcours de l'arbre en partant de chaque feuille (symbole) et concaténation des noms de chaque sommet pour créer les codes Huffman.

   Pour cet exemple: 'A'= 1, 'B'=011, 'C'=010, 'R'=00 (la solution n'est pas unique)

4. codage: la fonction prend en entrée la matrice de données (image, texte, etc), le dictionnaire et la liste de symboles et renvoie l'information codée (vecteur binaire).
   'ABRACARABA' ➡ 101100101010010111

5. decodage : la fonction prend en entrée la chaîne binaire représentant l'information compressée, le dictionnaire et la liste de symboles et renvoie le vecteur de symboles d'origine.

Les fonctions doivent être réalisées et testées indépendamment. Après validation de chaque étape, le programme complet doit être validé et évalué sur un texte (chaîne de caractères) et/ou une matrice de test (image à télécharger).