Travaux Dirigés d'informatique n°7

Algorithmique

Code de Huffman.

Exercice 1 Code.
Lesquels de ces ensembles forment un code.

1. $\{100, 0100, 010\}$.
2. $\{00, 01, 10, 11\}$.
3. $\{000, 001, 10, 1100, 1101, 111\}$.
4. $\{00, 11, 001, 010, 100\}$.
5. $\{00, 01, 10, 11\} \cup \{w\}$ *pour tout mot binaire* w.

Exercice 2 Codage.

1. *Trouver un code de Huffman pour les textes suivants, vous donnerez l'arbre binaire et le tableau construit pour obtenir ce code:*
   - *la belle et la bete*
   - *tagada tsoin tsoin*
2. *expliquer comment compresser les messages précédents.*

Exercice 3 Décodage.

1. *Expliquer comment décompresser le message suivant:*

   111001['e']0001['a']1['d']01['g']1['n']001['o']1['y']1['t']0001['i']1['p']1[' ']001['c']1['m']01['r']1['s']
   1100001100100010100111101111011101011000101110110011111000110110000011 1
   0101110100111111110100001010001010100100110011100110110010111001001110

2. *quelle est la hauteur minimale et la hauteur maximale d'un arbre de Huffman?*

Exercice 4 Occurence.
Ecrire une fonction `int Compter(char nom[], int occ[])` *qui initialise le tableau* `occ`*, de taille* 256*, en remplissant chaque case par le nombre d'occurences de la lettre, contenue dans le fichier* `nom`*, dont le code ASCII est l'indice de la case. Pour représenter les lettres, on utilisera le type* `unsigned char`*. La fonction renverra le nombre de caractères distincts contenu dans le fichier ou* `-1` *en cas d'erreur.*

Nous allons représenter l'arbre sous la forme d'un tableau de

```
typedef struct
{
    char lettre;
    int occurence;
    int gauche, droite;
} NoeudHuffman;
```

L'entier `gauche` (resp. `droite`) représente l'indice du fils gauche (resp. droit).

Exercice 5 Arbre de Huffman.
Ecrire une fonction `int CreerArbre(NoeudHuffman arbre[], int occ[])` *qui crée l'arbre de Huffman à partir du poids des lettres contenus dans le tableau* `occ` *et renvoie le nombre de nœuds de l'arbre ou* `-1` *en cas d'erreur.*

```
typedef struct
{
    char code[256];
    int nombrebit;
} Codage;
```

Exercice 6 Code d'un caractère.
Ecrire une fonction `void CreerCode(Codage C[], NoeudHuffman arbre[], int nbfeuilles)` *qui, pour chaque lettre* `x` *présente dans l'arbre, initialise la case* `C[x]` *avec le code de Huffman de la lettre* `x`*.*

Exercice 7 Compression.
Ecrire une fonction `int Compression(char FicBrut[], char FicCom[])` *qui compresse le texte contenu dans le fichier* `FicBrut` *et sauvegarde le code dans le fichier* `FicCom`*.*