Université Pierre et Marie Curie **Maitrise d'Informatique**
Juin 2007 – Session 2 **Compression, codage, cryptographie**

Examen-Compression
Durée : 1h30. Tous documents autorisés

1 Interclassement de listes tries (6 points)

On considère p listes **triées** $L_1, \ldots, L_p$ de tailles $n_1, \ldots, n_p$. Le but de cet exercice est d'utiliser des arbres de Huffman pour interclasser ces p listes en une seule liste triée L. Vous expliciterez :

1. les structures de données utilisées
2. l'algorithme d'interclassement
3. la complexité de l'algorithme d'interclassement, en place et en nombre de comparaisons, dans le cas le pire. (On rappelle que l'interclassement de deux listes triées L_1 et L_2 de tailles n_1 et n_2 nécessite $O(n_1 + n_2)$ comparaisons dans le cas le pire.)

2 Compression avec dictionnaire (14 points)

Étant donné un texte T, on le compresse de la manière suivante : on le lit de gauche à droite et, à chaque étape, on cherche le plus long motif qui a déjà été rencontré. On remplace alors ce motif par deux nombres : la position de sa première occurrence suivie de sa longueur, à condition que le motif soit suffisamment long pour qu'il y ait compression. Dans le texte compressé, il faut bien sûr indiquer ce qui correspond à un couple position-longueur et ce qui correspond à un caractère non compressé : un couple position-longueur sera précédé du bit "0" et chaque caractère non compressé sera précédé du bit "1".

Dans la suite de l'exercice, on suppose que le texte initial ne comporte pas plus de 2^{12} caractères et que l'on code un caractère isolé sur 8 bits, une position sur 12 bits et une longueur sur 3 bits. Il y aura donc compression à partir d'une séquence de 2 caractères.

Par exemple, le texte
"AIDE-TOI-LE-CIEL-T-AIDERA"
se compresse en
"1A1I1D1E1-1T1O1I1-1L0(3,2)1C1I1E1L0(4,2)1-0(0,4)1R1A"
(le premier caractère du texte est en position 0).

1. Décompresser le texte compressé :
   “1R1O1U1D0(1,3)0(1,2)1-1E1T1-1S1C0(1,2)1B1I0(3,3)0(17,5)”.

2. Le texte initial est représenté par un tableau T de caractères, indexé de 0 à $n-1$. Le texte compressé est représenté par un tableau C de bits.

   (a) On suppose que l'on dispose des procédures suivantes :

   `EcrirePositionLongueur(C,i,pos,lg)` qui, étant donnés un tableau C de bits, un indice i, une position *pos* et une longueur lg remplit $C[i, i+11]$ avec les 12 bits qui codent *pos* et $C[i+12, i+14]$ avec les 3 bits qui codent lg.

   `CalculerPositionLongueur(T,i,pos,lg)` qui, étant donnés un tableau T de caractères et un indice i calcule la longueur lg (inférieure ou égale à 2^3) du plus long motif $T[i, i+lg-1]$ qui apparaît dans $T[0, i-1]$ et la position *pos* de la première occurrence de $T[i, i+lg-1]$.

   Écrire un algorithme de compression.

   (b) On suppose que l'on dispose des procédures suivantes :

   `LireCaractere(C,i,car)` qui, étant donnés un tableau C de bits et un indice i, calcule le caractère *car* codé par les 8 bits de $C[i, i+7]$.

   `LirePositionLongueur(C,i,pos,lg)` qui, étant donnés un tableau C de bits et un indice i, calcule le nombre *pos* codé par les 12 bits de $C[i, i+11]$ et le nombre lg codé par les 3 bits de $C[i+12, i+14]$.

   Écrire un algorithme de décompression.

3. Exposer une ou plusieurs méthodes pour implanter la procédure `CalculerPositionLongueur`.

4. Comme la position est codée sur 12 bits, il a fallu limiter la taille du texte initial à 2^{12} caractères. Comment faire pour traiter des textes plus grands ?