THIN - THÉORIE DE L'INFORMATION - TP 1 - 2

Ce TP devra être rendu individuellement: une version temporaire à la fin de chacune des séances de TP, puis une version définitive pour vendredi 4 fevrier.
Il est conseillé d'utiliser maple (et dans ce cas, de rendre des fichiers source, et **pas** *des worksheet).*

Liste de quelques fonctions maple qui peuvent être utiles :

- `length` : donne la taille d'une chaîne de caractères.
- `convert`(a,binary) : convertit a en binaire.
- `convert`(a,list) : convertit a (de type `set`, `array`, `string` ou autre) en liste.
- `cat` : permet de concatener plusieurs chaînes de caractères.

En outre, au debut du TP, vous allez télécharger sur ma page web le fichier `alice` (`http://perso.univ-rennes1.fr/anna.morra/temp.html` puis clic droit et télécharger sous...). À l'intérieur de celui-ci sont définies les fonctions `tri` et `insere` et la chaîne de caractères `texte` (qui contient un long texte à comprimer avec les différentes méthodes qu'on verra, après les avoir testées sur des exemples plus petits).

- `tri`(L) : trie une liste de listes en suivant la deuxième composante (et en ordre croissant).
- `insere`(A,L) : insère dans la bonne position (en respectant l'ordre croissant selon la deuxième composante) la liste A dans la liste de listes L.

Remarque : le but est de ne pas utiliser les fonctions avancées pour les chaînes de caractères !!!

Exercice 1 : Codage de Huffman

Étant donnée une chaîne de caractères `texte`, écrire un programme qui

(1) Analyse la source et ses fréquences (i.e. il va construire une liste contenant tous les symboles présents dans `texte`, et une autre liste contenant leur fréquences).
(2) Calcule l'entropie de la source.
(3) Crée le code de Huffman associé à la source.
(4) Calcule la longueur moyenne de ce code et son efficacité (et les compare à celles d'un code à longueur fixe).
(5) Compresse donc, grâce au codage de Huffman ainsi crée, le fichier originaire.
(6) (Facultatif) Construire le code de Huffman pour la source B constituée par les paires de caractères consécutifs et compresser à nouveau le fichier.

En particulier, à chaque étape du programme, il faudra imprimer à l'écran les résultats obtenus.

Exercice 2 : LZ77

- Écrire une fonction qui, étant données deux chaînes de caractères `recherche` et `lecture` de longueurs fixées (respectivement m et n), cherche la plus longue occurrence du début de `lecture` dans `recherche` (si il y en a deux idéntiques on choisit celle plus à gauche) et retourne le triplet
$$(p, l, c)$$
comme vu en cours.
- Écrire un programme qui utilise la fonction ci-dessus pour compresser un texte avec LZ77.
- Tester son programme sur `texte`.
- Écrire une fonction de décompression pour LZ77.

(TOURNEZ SVP)

Date: 19-26 janvier 2011.

Exercice 3 : LZ78

- Écrire un programme qui applique l'algorithme LZ78 pour le compresser une chaîne de caractères (il est conseillé de trier le dictionnaire pour que la recherche soit plus rapide).
- Écrire aussi le programme de decompression.
- Tester son programme sur `texte`.

Exercice 4 (Facultatif)

Modifier les fonctions de compression/décompression des exercices 2 et 3 de manière que les triplets/paires soient codés par des codes binaires à longueur fixe.