EISTI. **Année scolaire 2009-2010**

DÉPARTEMENT " INFORMATIQUE "

THÉORIE DE L'INFORMATION. PÔLE "INFORMATIQUE THÉORIQUE"

Livrable N^{o}1. **Instructions.**

I. Langage de programmation

Le langage de programmation recommandé pour ce livrable est **Scilab**. Néanmoins, il est autorisé de programmer en Java. Attention ! Le projet nécessite une certaine maîtrise de Java. Il n'est pas conseillé d'utiliser ce langage si vous ne l'avez pas pratiqué suffisamment !

Dans les deux cas vous devez livrer un code qui s'exécute sans erreur (après compilation dans le cas de Java). Tout code doit être dûment commenté. Le barème de notation prévoit 2 points pour les commentaires et 1 point pour l'organisation générale de votre projet informatique.

II. Aide fournie

Pour faciliter la prise en main du projet et en particulier, la manipulation d'arbres binaires, un ensembles de fonctions (Scilab et Java) vous est fourni. Vous pouvez les utiliser dans votre projet. Dans le cas de Java, il s'agit d'un ensemble de classes couvrant les besoins du projet dans lesquelles certaines méthodes sont à compléter.

Dans les deux cas, les fonctions fournies permettent de définir, construire et manipuler les arbres binaires et des ensembles d'arbres, appelés forêts. Chaque arbre est défini par quatre éléments :

1. Poids de la racine : nombre réel
2. Chaîne de caractères associée à la racine
3. Sous-arbre gauche
4. sous-arbre droit

Scilab Dans le cas particulier d'arbres qui n'ont qu'une racine (des feuilles) les deux derniers paramètres sont des constantes.

Les fonctions fournies utilisent les structures de scilab , tlist. Vous pouvez trouver tous les renseignements nécessaires sur ces structures dans l'aide de scilab. Cependant, aucune connaissance de tlist n'est nécessaire pour utiliser les fonctions fournies. Vous pouvez les manipuler comme une mini librairie qui réalise en scilab les types abstraits "Arbre Binaire" et "Forêt Ordonnée" avec un minimum de fonctions.

Java Les classes à compléter sont Source.java, Code.java, Texte.java.

III. Travail à faire

---

1. Ecrire une fonction qui construit le code de Huffman.

   Scilab Elle doit prendre en entrée deux tableaux :

   (a) tableau de n caractères, l'alphabet ;

   (b) tableau de n réels, la distribution de probabilités associée

   La fonction doit renvoyer la table de code, associant à chaque caractère une séquence de bits sous forme de chaîne de caractères.

   Java Cette fonction doit être une méthode de classe Source. La table de caractères et la distribution de probabilités sont alors les attributs de cette classe et n'ont donc pas besoin d'être passés en paramètres explicites de la méthode.

   Pour réaliser cette fonction, inspirez vous de l'algorithme de fusion d'arbres vu en TD3 de "théorie de l'information".

2. Ecrire une fonction qui calcule l'entropie d'une source à partir de sa distribution de probabilités (membre de la classe Source en Java)
3. Ecrire une fonction qui calcule la longueur moyenne des mots de code à partir de la table de code (membre de la classe Code en java)
4. Ecrire une fonction qui encode un message composé de caractères de l'alphabet donné en utilisant une table de code (membre de la classe Code en java).
5. Ecrire une fonction qui permet d'extraire l'alphabet et la table de probabilités à partir d'un texte passé en paramètre comme chaîne de caractères (constructeur de la classe Texte en Java).
6. Ecrire une fonction qui lit le contenu d'un fichier texte dont le nom est passé en paramètre et qui construit les données d'une source d'information associée : la table de probabilités et l'alphabet.

   Scilab Il suffit d'utiliser la commande `s=mgetl('fichier.txt');` pour récupérer en un seul coup le contenu du fichier dans une cha)ne de caractères.

   Java L'algorithme de lecture "caractère par caractère" dans un fichier est fourni (dans la classe Source).

7. A partir des fonctions réalisées composer un ensemble d'instructions (le main() en Java) permettant de lire le contenu texte d'un fichier, construire le code de Huffman associé et produire la chaîne de bits issue d'encodage du texte avec le code obtenu. Calculer l'entropie du texte ainsi que la longueur moyenne des mots de code obtenu. Calculer le taux de compression.

IV. Données

Pour tester vos programmes utilisez d'abord un petit alphabet (comme celui étudié en TD) pour pouvoir vérifier manuellement la justesse de vos résultats. Vous pourrez ensuite appliquer votre programme sur un vrai texte. **POUR LIMITER LA TAILLE D'ALPHABET, UTILISER UNIQUEMENT LES CARACTÈRES MAJUSCULES, SANS ACCENTS.**

Un ensemble de fichiers avec de petits extraits de texte sont mis à votre disposition sur AREL. Vous pouvez également en utiliser un de votre choix (300-500 caractères environ).

V. A livrer

1. Code source commenté
2. Fichier texte utilisé pour la compression
3. Fichier texte contenant la chaîne binaire obtenue par codage de Huffman
4. un rapport décrivant le codage de Huffman, résultats théoriques concernant les taux de compression, description des expériences réalisées avec votre programme : entropie du texte, longueur moyenne de mots du code, taux de compression maximal possible (théorique), taux obtenu. Voir AREL pour le descriptif de rapport.