Algorithmique Avancée **Devoir de Programmation : Tries**

Devoir à faire en binôme. Soutenance en séance de TD/TME 10 (entre le 06/12 et 08/12).
Rapport et Code Source à rendre (par mail à vos enseignants) au plus tard le 11/12 à minuit.
Langage de programmation libre.

1 Présentation

Le but du problème consiste à représenter un dictionnaire de mots. Dans cette optique, nous proposons l'implémentation de deux structures de tries concurrentes puis une étude expérimentale permettant de mettre en avant les avantages et inconvénients de chacun des modèles. En plus des implantations des structures de données et des primitives de base, nous envisageons une fonction avancée pour chacun des modèles.

Les deux modèles envisagés sont (1) les Patricia-tries (cf. TD 3-4 exercice 2.3) et (2) les tries hybrides (cf. cours). On rappelle que dans une telle structure, chaque mot encodé n'apparaît qu'une seule fois.

Le dictionnaire que nous considérons est constitué de mots construits sur l'alphabet du code ASCII. Celui-ci est composé de 128 caractères, chacun étant encodé sur 8 bits.

1.1 Structure 1 : Patricia-Tries

Dans l'exercice de TD, l'alphabet utilisé contenait 4 lettres, ainsi qu'un caractère indiquant la fin d'un mot.

Question 1.1 Déterminer judicieusement un caractère parmi les 128 du code ASCII qui sera utilisé pour indiquer la fin d'un mot. Par conséquent, on ne pourra représenter que des mots construits sur les 127 caractères restants.

Question 1.2 Encoder les primitives de base concernant les Patricia-tries.

Question 1.3 Construire par ajouts successifs l'arbre représentant la phrase suivante, appelée par la suite `exemple de base` :
A quel genial professeur de dactylographie sommes nous redevables de la superbe phrase ci dessous, un modele du genre, que toute dactylo connait par coeur puisque elle fait appel a chacune des touches du clavier de la machine a ecrire ?
Quelques erreurs se sont glissées dans la phrase, afin de la rendre compatible avec le code ASCII.

1.2 Structure 2 : Tries Hybrides

Question 1.4 Définir et encoder les primitives de base concernant les tries hybrides.

Question 1.5 Construire par ajouts successifs le trie hybride représentant l'`exemple de base`.

2 Fonctions avancées pour chacune des structures

Question 2.6 Écrire les algorithmes suivants pour chacun des deux modèles d'arbres.

- une fonction de recherche d'un mot dans un dictionnaire : `Recherche`(arbre, mot) → booléen
- une fonction qui compte les mots présents dans le dictionnaire : `ComptageMots`(arbre) → entier
- une fonction qui liste les mots du dictionnaire dans l'ordre alphabétique : `ListeMots`(arbre) → liste[mots]
- une fonction qui compte les pointeurs vers Nil : `ComptageNil`(arbre) → entier
- une fonction qui calcule la hauteur de l'arbre : `Hauteur`(arbre) → entier

— une fonction qui calcule la profondeur moyenne des feuilles de l'arbre :
`ProfondeurMoyenne`(arbre) → entier
— une fonction qui prend un mot A en argument et qui indique de combien de mots du dictionnaire le mot A est préfixe. Ainsi pour l'`exemple de base`, le mot *dactylo* est préfixe de deux mots de l'arbre (dactylographie et dactylo). Noter que le mot A n'est pas forcément un mot de l'arbre :
`Prefixe`(arbre, mot) → entier
— une fonction qui prend un mot en argument et qui le supprime de l'arbre s'il y figure :
`Suppression`(arbre, mot) → arbre

3 Fonctions complexes

Question 3.7 Étant donnés deux Patricia-tries, expliquer les étapes importantes de la fusion de ces deux arbres en un seul Patricia-trie. Puis encoder un algorithme permettant de fusionner deux Patricia-tries en un seul.

Question 3.8 Définir les deux fonctions de conversion permettant de passer d'une structure d'arbre à l'autre.

Question 3.9 Après plusieurs ajouts successifs dans un trie hybride, ce dernier pourrait être plutôt déséquilibré. Après avoir expliqué les idées clé pour rendre un tel arbre mieux équilibré, et donné une idée du seuil permettant de dire si un arbre est déséquilibré ou non, encoder un algorithme d'ajout de mot suivi d'un rééquilibrage si nécessaire.

4 Complexités

Question 4.10 Pour chacune des questions de 2.6. à 3.9, donner, en la justifiant, la complexité de la fonction (après avoir indiqué quelle était la mesure de complexité judicieuse, par exemple, le nombre de comparaisons de caractères, ...)

5 Étude expérimentale

La dernière partie du devoir consiste à comparer expérimentalement les structures.

Question 5.11 Construire le Patricia-trie et le trie hybride encodant l'ensemble des mots de l'œuvre de Shakespeare [1].

Question 5.12 En utilisant les fonctions définies dans les parties précédentes(et éventuellement d'autres), comparer les deux types d'arbres encodant les œuvres de Shakespeare. La comparaison consiste par exemple en :
— temps de construction de la structure complète
— temps d'ajout d'un nouveau mot (éventuellement n'existant pas en anglais) dans chacune des structures
— temps de la suppression d'un ensemble de mots des structures
— profondeur des structures
— ... et toute autre comparaison qui semble pertinente.

Question 5.13 (facultative) Comparer les temps de constructions des arbres soit par ajouts successifs soit par fusions d'arbres encodant chacun une œuvre (cette approche permet des constructions en parallèle).

Question 5.14 (facultative) Afficher un aperçu global et visuel des deux structures afin de pouvoir effectuer certaines comparaisons *de visu*.

1. Une archive de chaque œuvre est disponible : `https://www-master.ufr-info-p6.jussieu.fr:8083/2016/ALGAV/Shakespeare.tar`.