Travaux Pratiques d'algorithmique n°2

Cours d'Informatique de Deuxième Année

—DEUG Sciences S4-MI—

Arbres lexicographiques généralisés

L'objet de cette feuille d'exercices est l'implémentation des opérations de base sur une nouvelle structure de données récursive : la structure d'*arbre lexicographique généralisé*. Nous utilisons ici cette structure pour construire un dictionnaire. Nous voulons être capable d'y rechercher un mot de manière efficace.

On considère les déclarations suivantes :

```
struct noeud
{
  char Lettre;                /* donnees liees au noeud            */
  struct noeud * FilsGauche;  /* pointeur sur le sous-arbre gauche */
  struct noeud * FrereDroit;  /* pointeur sur le "frere" droit     */
};

typedef struct noeud Noeud;   /* nouveau type : Noeud              */

typedef Noeud * Arbre;        /* pointeur sur un Noeud             */
```

Un arbre est un arbre lexicographique si les conditions suivantes sont vérifiées :

- à chaque nœud on associe une lettre ;
- chaque chemin allant de la racine à une feuille correspond à un mot ;
- un préfixe commun à plusieurs mots de l'arbre n'apparaît qu'une fois dans l'arbre.

▸ **Exercice 1.** ***(Fonctions de base)*** *Afin de pouvoir manipuler un arbre lexicographique, nous demandons d'écrire les fonctions élémentaires suivantes :*

1. `Noeud * CreerFeuille(char c)` *qui retournera un pointeur sur un nœud contenant le caractère passé en argument ;*
2. `int EstFeuille(Arbre a)` *qui testera si le nœud pointé par* `a` *est une feuille ;*
3. `int EstVide(Arbre a)` *qui testera si l'arbre pointé par* `a` *est vide ;*
4. `int DetruireNoeud(Noeud * n)` *qui libère l'espace mémoire associé au nœud* `n`*. Renvoie* `1` *si la libération s'est effectuée sans problème,* `0` *sinon ;*

▸ **Exercice 2.** ***(Insertion)*** *On désire insérer un mot (et donc tous les nœuds associés) dans un arbre lexicographique.*

1. *Comment peut-on coder la fin d'un mot dans l'arbre ?*

2. *Implémenter la fonction* `InsertionLex` *prenant en paramètre un mot* `mot` *et un arbre lexicographique* `a` *réalisant l'algorithme d'insertion.*

▸ **Exercice 3.** ***(Lecture dans un fichier)*** *Écrire une fonction* `Lecture` *prenant en argument un pointeur sur un fichier* `f` *et un arbre lexicographique* `a`*, et qui insère les mots de* `f` *dans l'arbre* `a`*.*

▸ **Exercice 4.** ***(Affichage)*** *Écrire une fonction* `AffichageMots` *affichant tous les mots contenus dans un arbre lexicographique passé en argument.*

▸ **Exercice 5.** ***(Sauvegarde dans un fichier)*** *Écrire une fonction* `Sauvegarde` *prenant en argument un pointeur sur un fichier* `f` *et un arbre lexicographique* `a`*, et qui écrit tous les mots de l'arbre* `a` *dans le fichier* `f`*.*

▸ **Exercice 6.** ***(Recherche d'un mot)*** *Écrire une fonction* `Appartient` *qui renvoie* 1 *si le mot* `mot` *passé en argument appartient à l'arbre lexicographique* `a`*,* 0 *sinon.*

▸ **Exercice 7.** ***(Recherches sous contrainte)***

1. *Écrire une fonction* `AfficherMotsLongueur` *qui affiche tous les mots de l'arbre* `a` *d'une longueur égale à une longueur donnée.*

2. *Écrire une fonction* `AfficherMotsPrefixe` *qui affiche tous les mots de l'arbre* `a` *qui comportent un préfixe donné.*