Travaux Pratiques n°4

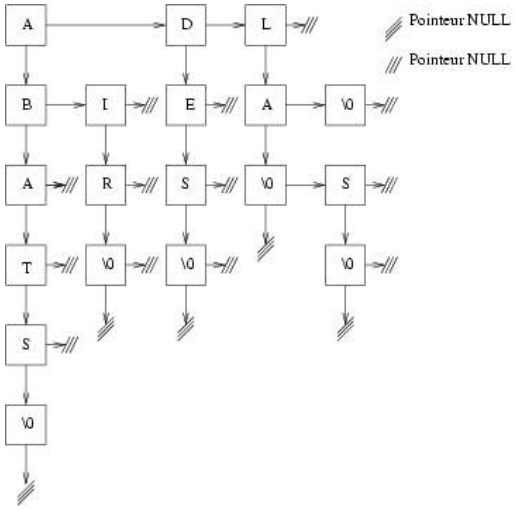
Cours d'Informatique de Deuxième Année

—Licence MI L2.2—

Arbre lexicographique

L'objet de cette feuille d'exercices est l'implantation 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 la déclaration suivante de la structure classique d'un arbre lexicographique :




```
struct noeud
{
    char Lettre;
    struct noeud * FilsGauche;
    struct noeud * FrereDroit;
};

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 est associée une lettre,
– chaque chemin de la racine vers une feuille correspond à un mot,
– un préfixe commun à plusieurs mots de l'arbre n'apparaît qu'une fois dans l'arbre (factorisation des chemins préfixes).

▸ **Exercice 1.** ***Fonctions de base***

Afin de pouvoir manipuler un arbre lexicographique, nous demandons d'écrire les fonctions élémentaires suivantes :

– `Noeud * CreerFeuille(char c)` *qui retournera un pointeur sur un nœud contenant le caractère passé en argument,*
– `int EstFeuille (Arbre a)` *qui testera si le nœud pointé par* `a` *est une feuille,*
– `int EstVide (Arbre a)` *qui testera si l'arbre pointé par* `a` *est vide,*
– `int DetruireArbre (Arbre a)` *qui libère par récurrence l'arbre lexicographique* `a`*.*

▸ **Exercice 2.** ***Insertion***

– *On désire insérer un mot et donc tous les nœuds associés dans un arbre lexicographique. Comment peut-on coder la fin d'un mot dans l'arbre sans modifier la structure proposée ? En supposant que l'alphabet des mots soit l'ensemble des caractères ASCII, comment modifier la structure pour signaler les nœuds terminaux ?*
– *Implanter la fonction* InsertionLex *prenant en paramètre un mot et un arbre lexicographique et réalisant l'opération d'insertion. Quelle est la complexité temporelle de cette fonction ? En déduire la complexité globale de construction de l'arbre lexicographique. Proposer (sans implanter) des variantes pour la représentation de nœuds frères afin d'améliorer la complexité.*

▸ **Exercice 3.** ***Lecture dans un fichier***

Ecrire 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.*
Comment pourrait-on utiliser la structure d'arbre lexicographique afin de réaliser un tri des mots ? Modifier la fonction InsertionLex *afin de pouvoir réaliser le tri (si ce n'est pas déjà fait). Cette modification permet-elle d'améliorer la complexité temporelle de la construction de l'arbre ?*

▸ **Exercice 5.** ***Sauvegarde dans un fichier***

Ecrire 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, dans l'ordre lexicographique.*

▸ **Exercice 6.** ***Recherche d'un mot***

Écrire une fonction Appartient *qui renvoie* 1 *si le mot passé en argument appartient à l'arbre lexicographique* `a`, 0 *sinon.*

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

- *Ecrire une fonction* AfficherMotsLongueur *qui affiche tous les mots de l'arbre* `a` *de longueur donnée* `k`*. Quelle est la complexité temporelle de cette fonction ?*
- *Écrire une fonction* AfficherMotsPrefixe *qui affiche tous les mots de l'arbre* `a` *de préfixe commun* `p`*.*