

Christiane Frougny
cf@ai.univ-paris8.fr

Licence d'Informatique – L3
2009 – 2010

Algorithmique Avancée
Algorithmes de recherche

Projet **Arbres ternaires de recherche**

Les arbres ternaires de recherche (ATR) sont des structures de données permettant de représenter efficacement des ensembles de mots. Par exemple, ils sont utilisés pour représenter des dictionnaires d'anglais dans certains logiciels commerciaux. Ces structures sont généralement plus rapides que le hachage, permettent des modes de recherche avancée, et sont bien adaptées au classement.

Structure

Un nœud d'un ATR a trois fils, le gauche FG, le droit FD, et celui du milieu FM. L'étiquette du nœud est un caractère. Tout nœud d'un ATR est représenté par la structure suivante en C:

```
typedef struct tnode *Tptr;
typedef struct tnode {
    char etiq;
    Tptr FG, FM, FD;
} Tnode;
```

Recherche

On suppose ici que les chaînes de caractères se terminent par le caractère `'\0'`, comme en C. Ce caractère `'\0'` est plus petit que tous les autres caractères. Dans ce qui suit, `premier(s)` denote le premier caractère de `s`, et `suite(s)` le mot privé de son premier caractère.

Pour rechercher un mot `s` dans l'arbre, on regarde le premier caractère de `s`. S'il est plus petit que la racine, on recherche `s` à gauche, s'il est plus grand, on recherche `s` à droite, et s'il est égal, on descend dans le fils du milieu, et on y recherche la suite du mot `s`.

```

fonction Recherche(p,s)
debut
  si p = NULL alors Retourner(0);
  si premier(s) < p->etiq alors Retourner(Recherche(p->FG,s));
  si premier(s) > p->etiq alors Retourner(Recherche(p->FD,s));
  si premier(s) = '\0' alors Retourner(1)
    sinon Retourner(Recherche(p->FM,suite(s)))
fin

```

Insertion

Pour insérer un nouveau mot s , on compare son premier caractère à la racine, et on prend la bonne branche.

```

fonction Insertion(p,s)
debut
  si p = NULL alors
    { p := (Tpr) malloc(sizeof(Tnode));
      p->etiq := premier(s);
      p->FG := NULL;
      p->FM := NULL;
      p->FD := NULL;}
  si premier(s) < p->etiq alors p->FG := Insertion(p->FG,s);
  si premier(s) > p->etiq alors p->FD := Insertion(p->FD,s);
  si premier(s) = p->etiq alors
    {si premier(s) != '\0' alors p->FM := Insertion(p->FM,suite(s))}
fin

```

Parcours

On peut parcourir l'arbre de façon à obtenir les mots dans l'ordre lexicographique.

```

fonction Traverse(p,w,i)
debut
  si p = NULL alors Retourner();
  Traverse(p->FG,w,i);
  w[i] := p -> etiq;
  si p->etiq != '\0' alors Traverse(p->FM,w,i+1)
  sinon Afficher(w);
  Traverse(p->FD,w,i);
fin

```

L'appel se fait avec la fonction suivante.

```

fonction Parcours(p)
debut
  Traverse(p,word,0)
fin

```

Création de l'arbre

Les n mots sont stockés dans un fichier sur le disque. Chaque mot utilise une ligne. Pour obtenir un ATR qui soit aussi équilibré que possible, le programme commence par lire le fichier et ranger les mots dans l'ordre croissant dans un tableau A . Pour construire l'arbre on commence par insérer l'élément du milieu, puis récursivement les éléments plus petits et les éléments plus grands.

```
fonction Insertion_tous(A,n)
debut
  si n < 1 alors Retourner();
  m := n div 2;
  racine := Insertion(racine,A[m]);
  Insertion_tous(A,m);
  Insertion_tous(A+m+1, n-m-1)
fin
```

L'appel se fait avec la fonction suivante.

```
fonction Creation(A,n)
debut
  racine := NULL;
  Insertion_tous(A,n)
fin
```

Recherche de type "Mots-croisés"

On recherche un mot avec des jokers, un joker étant noté ".". Par exemple, si dans le dictionnaire figurent les mots "bas" et "tas", la recherche de ".a." donne "bas" et "tas".

La structure de la fonction qui effectue cette recherche est semblable à la fonction `Traverse`.

```
fonction Rech_mc(p,s,w,i)
debut
  si p = NULL alors Retourner();
  si premier(s) = "." ou premier(s) < p->etiq alors Rech_mc(p->FG,s,w,i);
  w[i] := p -> etiq;
  si premier(s) = "." ou premier(s) = p->etiq alors
    si p->etiq != '\0' et premier(s) != '\0' alors Rech_mc(p->FM,suite(s),w,i+1);
  si premier(s) = '\0' et p->etiq = '\0' alors Afficher(w);
  si premier(s) = "." ou premier(s) > p->etiq alors Rech_mc(p->FD,s,w,i)
fin
```

Projet

1. Utiliser la structure d'ATR pour implanter un dictionnaire français avec les opérations qui ont été décrites.
2. En option écrire une fonction qui supprime un mot du dictionnaire.
3. Constituer un lexique français - anglais et l'utiliser.

Rapport de projet

Remettre un rapport écrit contenant :

- une exposition du projet, des structures de données et des algorithmes utilisés
- un listing commenté du programme
- un choix d'exécutions variées.

Date de remise du rapport de projet : le 26 janvier 2010