CORRECTEUR ORTHOGRAPHIQUE

Phase 3

24 AVRIL 2022

SEBBAH VALENTIN RAMOS LUIS

Sommaire

Introduction :

Pour cette 3ème phase nous allons utiliser une nouvelle structure de données qui est l'arbre BK.

Pour chaque mot m mal orthographié d'un texte, nous souhaitons toujours faire des suggestions de correction à l'utilisateur. Cette fois, nous allons utiliser une structure de données adaptée à la recherche approximative de chaines de caractères identiques : un arbre BK, (d'après leurs inventeurs Burkhard and Keller).

Arbres BK :

Cette structure contient plusieurs champs, un champ pour le mot, deux champs pour les fils gauche et le frère droit et finalement un champ qui contient un entier qui représente la distance de Levenshtein entre ce mot et le premier mot de l'arbre.

Insertion :

L'insertion dans un arbre BK se présente de la manière suivante : nous vérifions d'abord que l'arbre n'est pas nul si c'est le cas nous créons la racine et ajoutons le mot. Si le nœud où l'on souhaite ajouter le mot ne possède pas de fils gauche cela signifie que pouvons ajouter un nouveau nœud en tant que fils gauche du nœud actuel. Dans le cas où le nœud où l'on souhaite ajouter le mot possède un fils gauche, nous parcourons un à un ses fils jusqu'à temps que l'on puisse lui trouver une place. La place qui lui est attribuée est définie en fonction de la valeur de la distance Levenshtein entre le mot du nœud et le mot que l'on souhaite ajouter. Si lors du parcours nous retrouvons une valeur Levenshtein similaire avec l'un des fils du nœud, nous appelons récursivement la fonction sur le fils en question.

Libération :

Nous utilisons la même méthode que pour un arbre ATR c'est-à-dire nous parcourons l'arbre en profondeur et nous libérons les éléments, nous appelons cette fonction récursivement sur ses fils gauches et frères droits

Affichage :

Pour afficher l'arbre nous avons décidé d'utiliser les fichiers dot, une fois le correcteur_2 lancé il suffira de taper la commande dot -Tpdf arbre.dot -o arbre.pdf dans le terminal afin de transformer le fichier dot en pdf et pouvoir le visualiser graphiquement

Création :

Pour la création d'un arbre, nous parcourons l'ensemble des mots dans le fichier qui nous sert de dictionnaire, et nous les ajoutons au fur et à mesure à un nouvel arbre BK.

Comparatif temps d'exécution :

```
luis@luis-Swift-SF314-56G:~/Bureau/phase3$ time ./correcteur_2 ressources/a_corriger_1.txt ressources/dico_2.dico
Mot(s) mal orthographié : faix
Proposition(s) : foix
Mot(s) mal orthographié : c
Proposition(s) : et, je, se, il, ma, la, de, du
Mot(s) mal orthographié : faix
Proposition(s) : foix
Mot(s) mal orthographié : vil
Proposition(s) : il
Mot(s) mal orthographié : doigt
Proposition(s) : dit
Mot(s) mal orthographié : foit
Proposition(s) : foix, foie, foi, fois
Mot(s) mal orthographié : etais
Proposition(s) : etait

real    0m0,001s
user    0m0,001s
sys     0m0,000s
luis@luis-Swift-SF314-56G:~/Bureau/phase3$ time ./correcteur_1 ressources/a_corriger_1.txt ressources/dico_2.dico
Mot(s) mal orthographié : faix
Proposition(s) : foix
Mot(s) mal orthographié : c
Proposition(s) : se,ma,la,je,il,et,du,de
Mot(s) mal orthographié : faix
Proposition(s) : foix
Mot(s) mal orthographié : vil
Proposition(s) : il
Mot(s) mal orthographié : doigt
Proposition(s) : dit
Mot(s) mal orthographié : foit
Proposition(s) : foix,fois,foie,foi
Mot(s) mal orthographié : etais
Proposition(s) : etait

real    0m0,001s
user    0m0,001s
sys     0m0,000s
```

Figure 1 : Premier comparatif

```
luis@luis-Swift-SF314-56G:~/Bureau/phase3$ time ./correcteur_1 ressources/a_corriger_1.txt ressources/dico_3.dico
Mot(s) mal orthographié : faix
Proposition(s) : paix,faux,fait,fais,faim
Mot(s) mal orthographié : foie
Proposition(s) : voie,soie,oie,noie,joie,folie,fois,foire,foi
Mot(s) mal orthographié : c
Proposition(s) : y,x,v,s,i,ci,ce,ca,a
Mot(s) mal orthographié : faix
Proposition(s) : paix,faux,fait,fais,faim
Mot(s) mal orthographié : foie
Proposition(s) : voie,soie,oie,noie,joie,folie,fois,foire,foi
Mot(s) mal orthographié : foit
Proposition(s) : voit,toit,soit,fout,fort,font,fois,foi,fit,fait,doit,boit

real    0m0,061s
user    0m0,057s
sys     0m0,004s
luis@luis-Swift-SF314-56G:~/Bureau/phase3$ time ./correcteur_2 ressources/a_corriger_1.txt ressources/dico_3.dico
Mot(s) mal orthographié : faix
Proposition(s) : paix, fait, faim, faux, fais
Mot(s) mal orthographié : foie
Proposition(s) : folie, foire, voie, soie, noie, joie, fois, oie, foi
Mot(s) mal orthographié : c
Proposition(s) : ce, ci, y, x, v, s, i, ca
Mot(s) mal orthographié : faix
Proposition(s) : paix, fait, faim, faux, fais
Mot(s) mal orthographié : foie
Proposition(s) : folie, foire, voie, soie, noie, joie, fois, oie, foi
Mot(s) mal orthographié : foit
Proposition(s) : font, voit, toit, soit, doit, fout, fois, fort, boit, fit, fait, foi

real    0m0,644s
user    0m0,640s
sys     0m0,004s
```

Figure 2 : 2eme Comparatif

Nous remarquons ici que les arbres BK sont plus lents dans notre code, en effet nous nous sommes rendu compte que notre recherche n'avait pas une complexité logarithmique ce qui est le cas de la recherche dans un ATR.

Ceci est explicable du fait qu'on utilise on parcourt en profondeur de l'arbre ce qui ralenti notre programme et augmente le temps de calcul. Les arbres BK étant des arbres n-aires de recherche dans ce cas précis des arbres binaires de recherche leur croissance est d'un facteur exponentiel en nœuds, la parcours en profondeur devant parcourir tous les nœuds devient beaucoup plus lent. Alors que la recherche dans ATR est faite de telle sorte à ce que sa complexité soit de O(hauteur de l'arbre) ce qui est environ logarithmique.

Compilation et Exécution :

Pour compiler le correcteur_2, il suffit de taper make dans le terminal, le makefile créera alors les programmes correcteur_0/1/2

Pour exécuter les correcteurs il faut taper :

./correcteur_nb ressources/texte_a_corriger.txt ressources/dico_nb.dico

Conclusion :

Finalement avec cette partie-là, nous avons appris une nouvelle structure de données les arbres BK qui sont une méthode plus rapide de transformer un fichier texte et l'utiliser un dictionnaire car nous pouvons utiliser le champ valeur pour faire en sorte que la recherche soit en O(h) de plus vu que cet arbre prend des chaines de caractère plutôt que des caractères un à un l'auteur de cet arbre BK est forcément plus petite que la hauteur d'un arbre ternaire de recherche.