Rapport
Projet ALGAV Trie
Implantation du Trie Hybride et du Patricia Trie

Amel Arkoub 3301571
Ling-Chun SO 3414546

22 décembre 2017

Table des matières

Chapitre 0

Introduction

Présentation

Nous souhaitons représenter un dictionnaire de mots, c'est-à-dire implanter une structure de données éfficace stockant des mots. Pour cela, nous allons nous servir de la structure de Trie. Dans cette optique, nous proposons l'implantation de deux structures de tries concurrentes que sont le Trie Hybride et le Patricia Trie. De cette implantation, une étude expérimentale sera analysées afin de mettre en exergue les avantages et inconvénients de chacune de ces structures. Le langage choisit pour l'implantation de ces deux structures est JAVA.

Trie

Un Trie est une représentation arborescente d'un ensemble de clés en évitant de répéter les préfixes communs.

Trie Hybride

Un Trie Hybride est un arbre ternaire dont chaque noeud contient un caractère et une valeur (non vide lorsque le noeud représente une clé). Chaque noeud a 3 pointeurs : un lien Inf (resp. Eq, res. Sup) vers le sous arbre dont le premier caractère est inférieur (resp. égal, resp. supérieur) à son caractère. Il permet en outre de réduire le nombre de pointeurs vide par rapport à un R-Trie.

Patricia Trie

Un Patricia Trie est un arbre dont le but est de réduire la taille des R-Trie tout en conservant une recherche efficace. Pour ce faire, plutôt que chaque noeud interne permette de distinguer une lettre, il permet de distinguer la plus longue sous-chaîne de lettres communes à plusieurs mots.

Chapitre 1

Trie Hybride

1.1 Implantation

1.1.1 Structure

Dans notre implantation JAVA, un Trie Hybride est un objet qui contient 5 éléments :

— un char “letter”, qui correspond à la lettre stockée.
— une valeur “value”, qui correspond à la représentation de fin de mot (-1 le noeud n’est pas la fin d’un mot, sinon la valeur correspond à l’ordre d’ajout croissant).
— un pointeur vers un Trie Hybride “fc”, ce Trie Hybride correpond au fils central, c’est-à-dire on parcours ce Trie Hybride si le caractère recherché au Trie Hybride courant est correct.
— un pointeur vers un Trie Hybride “fg”, ce Trie Hybride correpond au fils gauche, c’est-à-dire on parcours ce Trie Hybride si le caractère recherché au Trie Hybride courant est plus petit dans l’ordre alphabétique.
— un pointeur vers un Trie Hybride “fd”, ce Trie Hybride correpond au fils droit, c’est-à-dire on parcours ce Trie Hybride si le caractère recherché au Trie Hybride courant est plus grand dans l’ordre alphabétique.

Voici une représentation ci-dessous du Trie Hybride résultant des l’ajouts successifs des mots :

— don
— le
— a
— dort

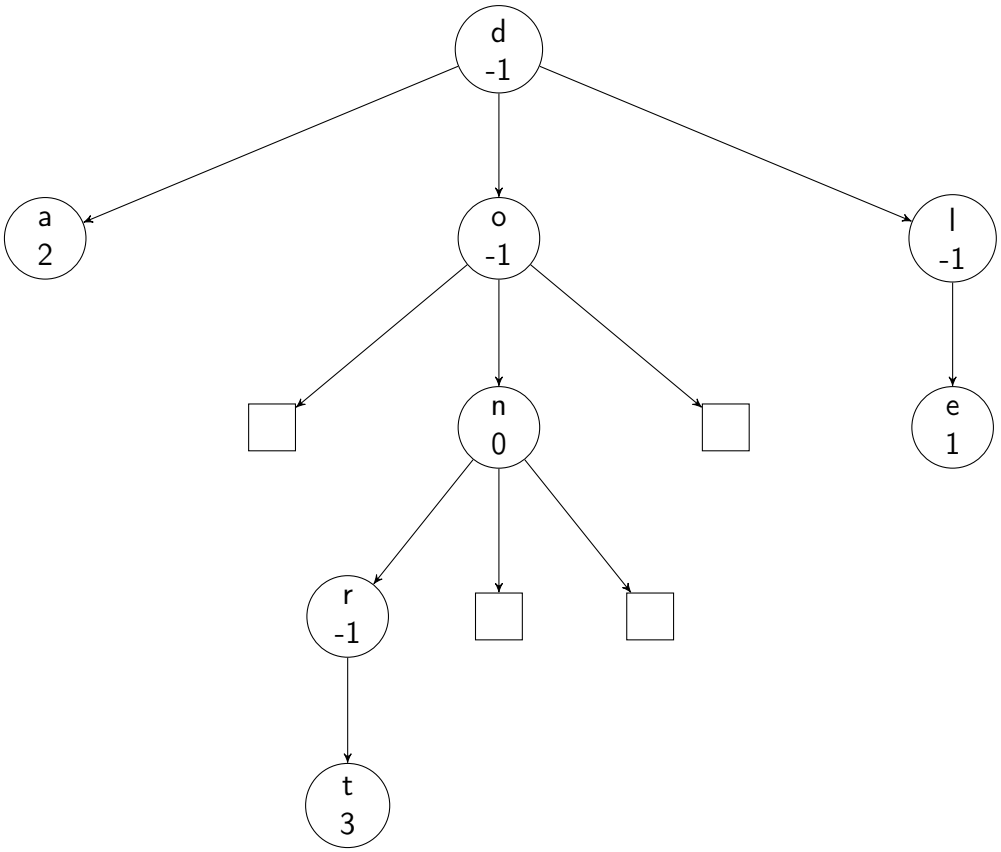

FIGURE 1.1 – Représentation d'un Trie Hybride

1.2 Description des algorithmes

1.2.1 Algorithmes triviaux

La plupart des algorithmes de l'implantation du Trie Hybride sont relativements triviaux, notamment les fonctions :

- Recherche(arbre, mot) → booléen
- ComptageMots(arbre) → entier
- ListeMots(arbre) → liste[mots]
- ComptageNil(arbre) → entier
- Hauteur(arbre) → entier
- ProfondeurMoyenne(arbre) → entier
- Prefixe(arbre) → entier

Elles correpondent pour la plupart d'un simple parcours de la structure du Trie Hybride, dont le traitement dépend de la fonction.

1.2.2 AjoutMot

La fonction ajoutMot permet d'ajouter un mot dans le Trie Hybride et donc aussi la sa construction dont la signature est : ajoutMot(mot, arbre) → void Les étapes sont donc :

- le cas d'arrêt lorsque le mot à été ajouté.
- si on est dans la racine qui est encore vide alors on ajoute le mot.
- si le mot est de longueur un, alors on verifie si le Trie Hybride courant correspond à la bonne lettre sinon on fait des appels récursifs sur le fils gauche ou droit.

— dans le cas général, on compare le caractère du mot et le caractère du Trie Hybride courant, si les lettres concordent alors on appel récursivement sur le fils central, s'ils sont différents on fait un appel récursif sur le fils gauche ou droit suivant l'ordre alphabétique des caractères.

1.2.3 Suppression

La fonction ajoutMot permet la suppression d'un mot dans le Trie Hybride et maintient le Trie Hybride cohérent en supprimant les noeuds dont il n'existe pas de mot, la signature est : suppression(arbre, mot) → void Les étapes sont donc :

— si le Trie Hybride courant est un pointeur null alors on a un cas d'arrêt
— si le mot est réduit à une lettre alors on vérifie que le caractère du mot et du Trie Hybride courant, s'ils ne sont pas égaux alors on fait un appel récursif sur le fils gauche ou droit suivant la comparaison de l'ordre alphabétique des lettres
— dans le cas général, on compare le caractère du mot et le caractère du Trie Hybride courant, si les lettres concordent alors on appel récursivement sur le fils central, s'ils sont différents on fait un appel récursif sur le fils gauche ou droit suivant l'ordre alphabétique des caractères.
— on calcul le nombre de mots issus de chaque fils, s'il existe un fils dont le nombre de mots est égal à 0, alors on détruit le fils en question.

1.3 Complexité

1.3.1 Recherche

Pour la recherche d'un mot de L caractères et un Trie Hybride de taille n, en prenant la comparaison de caractère comme mesure, on obtient une complexité dans le cas général :

— en Θ(L), si $L < n$;
— en Θ(n) sinon.

Sinon on a une complexité en Θ(n) dans le pire cas.

1.3.2 Comptage Mots

Le comptage de mots revient à faire un parcours de l'arbre en entier. Soit un arbre de taille n, en prenant la comparaison de la valeur du noeud comme mesure, on a donc une complexité Θ(n).

1.3.3 Liste Mots

La récupération des mots dans un Trie Hybride correspond aussi à un parcours de l'arbre en entier en gardant le *préfixe* en argument dans les appels de fonctions. Soit un arbre de taille n, en prenant la comparaison de la valeur du noeud comme mesure, on a donc une complexité en Θ(n).

1.3.4 Comptage Nil

Le comptage de pointeur vers null correspond à un parcours de tout les noeuds pour compter le nombre de fils null. Soit un arbre de taille n, en prenant la comparaison de la valeur du noeud comme mesure, on a une complexité en Θ(n) puisque pour n noeuds, on a un nombre de comparaison $\leq 3n$.

1.3.5 Hauteur

La détermination de la hauteur du Trie Hybride est un parcours complet de l'arbre, en prenant l'existence de fils comme mesure , on obtient une complexité en Θ(n).

1.3.6 Profondeur Moyenne

La profondeur moyenne d'un Trie Hybride est un parcours jusqu'aux feuilles dont on ajoute la profondeur dans une liste et on effectue une division. En prenant la comparaison d'existence de fils comme mesure, on obtient une complexité de Θ(n).

1.3.7 Préfixe

La recherche du préfixe peut dans le pire des cas en Θ(n) puisque le pire des cas est atteint lorsque le Trie Hybride revient à être une "liste chaînée" et donc à hauteur h=n.

1.3.8 Suppression

La suppression d'un mot est une recherche dans le Trie Hybride, ce qui correspond à une complexité en Θ(n) comparaison dans le pire cas, cependant il y a aussi 3 appels à la fonction comptageMots de complexité Θ(n). On a donc une complexité en $\mathcal{O}(n^2)$.

Chapitre 2

Patricia Trie

2.1 Implantation

2.2 Complexité

2.2.1 Recherche

2.2.2 Comptage Mots

2.2.3 Liste Mots

2.2.4 Comptage Nil

2.2.5 Hauteur

2.2.6 Profondeur Moyenne

2.2.7 Préfixe

2.2.8 Suppression

Chapitre 3

Fonctions complexes

3.1 Implantation

3.2 Complexité

3.2.1 Fusion de Patricia Trie

3.2.2 Conversion de Patricia Trie en Trie Hybride

3.2.3 Conversion de Trie Hybride en Patricia Trie

3.2.4 Rééquilibrage de Trie Hybride

Chapitre 4

Etude expérimentale