PR3-S2

Ayoub Bouchama

15 Juin 2023

Utilisation de l'arbre préfixe pour le calcul du nombre d'occurrences d'un mot dans un fichier

Implémentation

Dans mon implémentation, chaque noeud de l'arbre contient :

- Une valeur d'arete. (arete au dessus)
- Une valeur de noeud.
- Une clé.
- Un boolean qui indique si le contient contient uen information.
- Les fils du noeud qui sont des arbres. (stockés dans une file)

J'ai pris la clé comme une file, qui était la meilleure solution pour une telle implémentation car il permet de diviser la clé en de petits bouts de clés. Chaque clé est constitué de la concaténation de la valeur des aretes de chaque noeud qu'on a parcouru à partir de la racine vers le noeud correspondant à la clé.

Les fils ssont stockés dans une file car en premier j'avais prévu de les stocké dans un tableau mais vu que l'insertion n'impose pas necessairement un ordre et ccomme le module des files a été déjà implémenté dans le pr2-s2, je l'ai donc utilisé les fonctions et procedures qu

Utilisation

1. Pour utiliser le programme, il faut d'abord compiler les fichiers 'files.adb' et 'arbre.adb' (ou compiler directement le fichier 'test_arbre.adb').
2. Le fichier 'test_arbre.adb' contient des tests du module Arbre, principalement des tests pour la fonction Ajouter qui insère un nœud dans l'arbre, la fonction Valeur qui retourne la valeur du nœud associée à une clé, et des affichages de l'arbre à chaque étape d'insertion pour observer les différences. Le test contient chaque affichage où l'on affiche les nœuds (arêtes, clés, valeur du nœud) de chaque niveau.

3. Ensuite, il faut compiler les fichiers 'CLI.adb' et 'main.adb' (ou utiliser 'gnatmake main.adb' pour compiler tous les fichiers).

4. Il faut ensuite disposer du fichier dans le même répertoire et lancer la commande :

   ```
   ./main <mot> <fichier>
   ```

   où `<mot>` est le mot à rechercher et `<fichier>` est le nom du fichier de recherche.

5. Si le mot existe, le nombre d'occurrences du mot va s'afficher, sinon le programme va signaler que ce mot n'existe pas.

Réponse à la question 4

Une autre proposition d'application efficace de l'arbre préfixe est la suggestion de mots similaires ou la complétion de mots, telle qu'on les trouve dans les suggestions des claviers numériques des téléphones.

En utilisant un arbre préfixe, il est possible de stocker un large ensemble de mots et de préfixes correspondants. Lorsque l'utilisateur commence à saisir un mot, l'arbre préfixe est consulté pour trouver tous les mots qui ont le même préfixe. Ces mots peuvent être présentés à l'utilisateur comme des suggestions, l'aidant ainsi à compléter le mot ou à choisir une option similaire.

Réponse à la question 5

Je pense que la structure arborescente est très efficace dans les algorithmes de recherche grâce à sa capacité à organiser les données de manière hiérarchique. Elle permet une recherche rapide car elle réduit le nombre de comparaisons nécessaires entre nœuds à chaque étape.

Cependant, je pense que la suppression d'une donnée dans l'arbre peut être complexe. car lorsque nous devons supprimer un nœud, nous devons également supprimer tous ses descendants (ses fils, leurs fils...). Cependant, si nous souhaitons conserver les descendants du nœud que nous supprimons, cela peut poser un problème. Il peut être difficile de gérer cette situation et de maintenir la structure de l'arbre cohérente tout en préservant les fils du nœud supprimé. Cela nécessite une manipulation délicate de l'arbre et peut potentiellement affecter les performances de l'opération de suppression.

J'ai pensé qu'une solution à ce problème serait de simplement désactiver le nœud au lieu de le supprimer, afin de préserver le chemin d'accès vers ses fils.

Une amélioration possible pour réduire le temps de recherche dans l'arbre préfixe est d'indexer les éléments de l'arbre. Plutôt que de parcourir tous les nœuds

pour effectuer une recherche, on peut utiliser des indices associés à chaque nœud pour accéder directement aux fils correspondants. Cela permet d'obtenir rapidement la valeur associée au nœud sans avoir à parcourir l'ensemble de l'arbre. Par exemple, dans notre application de l'arbre préfixe, au lieu d'insérer les fils de chaque nœud de manière aléatoire, on peut utiliser le code de chaque bout de clé (arête) pour accéder directement aux fils appropriés, facilitant ainsi la récupération des valeurs des nœuds.

Bilan

Le projet etait vraiment interessant meme si je n'avais pas compris le concept de la mise en oeuvre de ce type d'arbre en premier temps. J'avais rencontré plusieurs difficultés, je m'etais bloqué plusieurs fois pour des erreurs betes mais j'ai pu débogué mon code à l'aide de l'affichage sur la console pour détecter d'ou vient l'erreur.