Université de Lille

FACULTÉ DES SCIENCES ET TECHNOLOGIES
Département Informatique

Arbre-B (B-tree)

Projet L3 – S2

Département d'Informatique

02 décembre 2021

Chaabane Djeraba

Table des matières

Introduction

Un arbre B est une structure d'arbre N-aire équilibré. Son histoire est liée à celle des structures de stockage des bases de données. Chaque sommet de l'arbre est composé d'au moins M couples (Clé, Pointeur). La notion de clé est liée à la notion de d'enregistrement ou un uplet dans une table de base de données relationnelle. N est un paramètre statique de l'arbre. Clé est une valeur entière et Pointeur est le lien vers un sommet fils. Les clés sont triées. Ainsi un sommet est composé de : Pointeur_0, (Clé_1, Pointeur_1), (Clé_2, Pointeur_2), … (Clé_K, Pointeur_K). N <= K <= 2N. Ainsi, Pointeur_i, tel que 1 <= i < 2*N, est le lien vers le sommet fils dont les clés sont comprises entre Clé_i et Clé_i+1. Par exemple, Pointeur_1 est le lien vers le sommet fils dont les clés sont comprises entre Clé_1 et Clé_2. Pointeur_0 est le lien vers le sommet fils dont les clés sont inférieures à Clé_1. Pointeur_2N est le lien vers le sommet fils dont les clés sont supérieures à Clé_2N. Trois fonctionnalités de base caractérisent le fonctionnement de l'arbre-B : recherche (clé), insertion (clé) et suppression (clé). De nombreux sites web présentent le fonctionnement de cette structure de données.

Il existe de nombreuses vidéos publiques sur le fonctionnement de l'arbre B. Parmi lesquelles, citons :

- https://www.youtube.com/watch?v=CYKRMz8yzVU
- https://www.canal-u.tv/video/inria/arbre_b.22441

L'arbre-B définit par 2 paramètres : L et U. U est le nombre de nœuds fils max et L le nombre de nœuds fils min. Arbre-B (L, U). n = nombre de clés contenues dans le nœud x. n clefs notées c[1], …, c[n]. Un booléen indiquant si x est une feuille ou non. n+1 fils : f[1], …, f[n+1]. Une feuille ne contient pas de fils. L'arbre-B vérifie les propriété suivantes :

- Toutes les feuilles ont la même profondeur, à savoir la hauteur h de l'arbre
- Si x n'est pas une feuille :
    - pour 2<=i<=n, pour toute clef x du fils[i] : cles[i-1] <= x <=cles[i]
    - Pour toute clef x du fils[1] : x <= cles[1]
    - Pour toute clef x du fils[n+1]: cles[n] <= x
- Si x n'est pas la racine, n est compris entre L-1 et U-1.

Travail à réaliser

Historique de l'arbre-B

Structure d'arbre-B en UML et en Python

Algorithme de recherche et codage en Python

Algorithme d'insertion et codage en Python

Algorithme de suppression et codage en Python

Version arbre-B+

La recherche retourne une valeur booléenne, selon que la clef recherchée est présente ou absente de l'arbre-B. L'insertion insère la clef dans l'arbre. La suppression retire la clef de l'arbre. L'insertion et suppression dans l'arbre-B (L,U) sont valides, si les conditions suivantes sont respectées :

- toutes les feuilles ont la même profondeur, à savoir la hauteur h de l'arbre,
- si x n'est pas une feuille :

- pour 2<=i<=n, pour toute clef x du fils[i] : cles[i-1] <= x <=cles[i]
- pour toute clef x du fils[1] : x <= cles[1]
- pour toute clef x du fils[n+1]: cles[n] <= x

- si x n'est pas la racine, n est comprise entre L-1 et U-1.

Batterie de tests

Lorsque le programme est réalisé, il convient de s'assurer qu'il répond aux spécifications de l'arbre-B, à travers docString de Python.

Test 1

Le noeud contient 2 clés au maximum (U=3) et 1 clé au minimum (L=2).
Insertion dans l'ordre : 2, 4, 5, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 7, 9, 11, 13.
Suppression dans l'ordre : 14, 10, 20, 18, 16, 24, 6
La vidéo ci-dessous visualise l'état de l'arbre à chaque insertions ou suppression.
https://www.youtube.com/watch?v=coRJrcIYbF4
vérification de l'insertion :
arbre1.recherche(42)
False
Arbre1.est_arbre_b()
True
Arbre1.insere(42)
Arbre1.Est_arbre_b()
True
Arbre1.recherche(42)
True

Vérification de la suppression :

Test 2

Suite à cette première batterie de tests, il convient de réaliser d'autres tests.
Le noeud contient 10 clés au maximum (U=11) et 5 clés au minimum (L=6).
Soit la liste des valeurs suivantes : 10, 20, 30, … 5000, 5, 15, 25, 35, ..4995.
Suppression dans l'ordre : 10, 20, 30, … 5000, 5, 15, 25, 35, ..4995

Généralisez les tests en considérant N = (2, 10, 100, 1000, 10000).

Grille d'évaluation

L'évaluation est basée sur une documentation et une démonstration. La démonstration et la documentation sont réalisées et livrées aux cours des deux dernières séances du projet.

La démonstration est réalisée sur le poste de travail, sur une durée de 30 minutes environ, sur les deux batteries de tests présentées. La démonstration vérifie la bonne réalisation des algorithmes.

La documentation est une facette essentielle du projet car elle constitue le référentiel de base concernant le projet après réalisation. La documentation ne doit pas dépasser 10 pages.

La documentation doit décrire :

(1) l'introduction générale : l'historique de l'arbre B. Et quelle est la signification de la lettre « B », l'intérêt des arbres B, la différence entre l'arbre B et l'arbre B+ ?

(2) les algorithmes (codes Python) commentés : l'algorithme de recherche, l'algorithme d'insertion, l'algorithme de suppression. Chaque algorithme doit comporter la documentation (docstring) : entrée, sortie, exemples, estimation de la complexité temporelle des différents algorithmes.

Règles de programmation

Les travail réalisé doit être téléversé sur Gitlab, en au moins 3 versions : fin janvier, fin février et fin mars.

Le projet peut être réalisé par une personne ou un binôme. Exceptionnellement, il est possible pour l'étudiant de choisir un projet différent. Le sujet en question doit être proposé et validé par l'enseignant responsable du groupe et l'enseignant responsable de l'UE, avant le 30 janvier 2022.

Avant de se lancer dans la réalisation et le développement des algorithmes du projet, il est nécessaire de prendre le temps de le comprendre et de le dérouler sur des données tests. Ensuite, organiser développer l'architecture en technologie objet du projet (classes et liens entre les classes).

Annexes

Listes

Le projet nécessite des connaissances sur les listes et arbres de recherche binaire : création de listes d'objets, ajout, suppression, recherche. Version Java : LikedList et ArrayList … Et les opérations get, add et remove … System.nanoTime …

Arbre de recherche binaire et équilibré

Exercice 1 : Structure de données

Expliquez les structures de données suivantes.

```
Class Arbre {
int valeur;
Arbre gauche;
Arbre droit;
……
}
```

Exercice 2 : Création d'un arbre de recherche binaire

Exercice 3 : Parcours dans un arbre binaire

Réalisez les procédures de parcours de l'arbre :

- void ParcoursPrefixe()
- void ParcoursInfixe()
- void ParcoursPostfixe()

Exercice 4 : Mesures

Réalisez les procédures ci-dessous :

- arbresEgaux(Arbre a, Arbre b)
- int hauteur(Arbre a)
- boolean estABR(Arbre a)
- insertion(int value)
- Boolean recherche (int value)