ProjetArbreB

Version numéro 2

Havart Florentin / Groupe 2
Bugnon Florentin / Groupe 7

Lien Git :
https://gitlab-etu.fil.univ-lille1.fr/bugnon/projetarbreb/-/tree/main

Table des matières:

Explication des arbre-B :

Un arbre-B est un arbre (au sens informatique du terme) tel qu'à chaque nœud on associe plusieurs clés prise dans un ensemble donné.

Soient L et U deux entiers naturels non nuls tels que $L \leq U$. En toute généralité, on définit alors un L-U arbre B3 de la manière suivante : chaque nœud, sauf la racine, possède un minimum de $L-1$ clés (appelées aussi éléments), un maximum de $U-1$ clés et au plus U fils. Pour chaque nœud interne — nœud qui n'est pas une feuille —, le nombre de fils est toujours égal au nombre de clés augmenté d'une unité. Si n est le nombre de fils, alors on parle de n-nœud. Un L-U arbre ne contient que des n-nœuds avec $L \leq n \leq U$. Souvent on choisit la configuration $L = t$ et $U = 2 \times t$: t est appelé le degré minimal de l'arbre B.

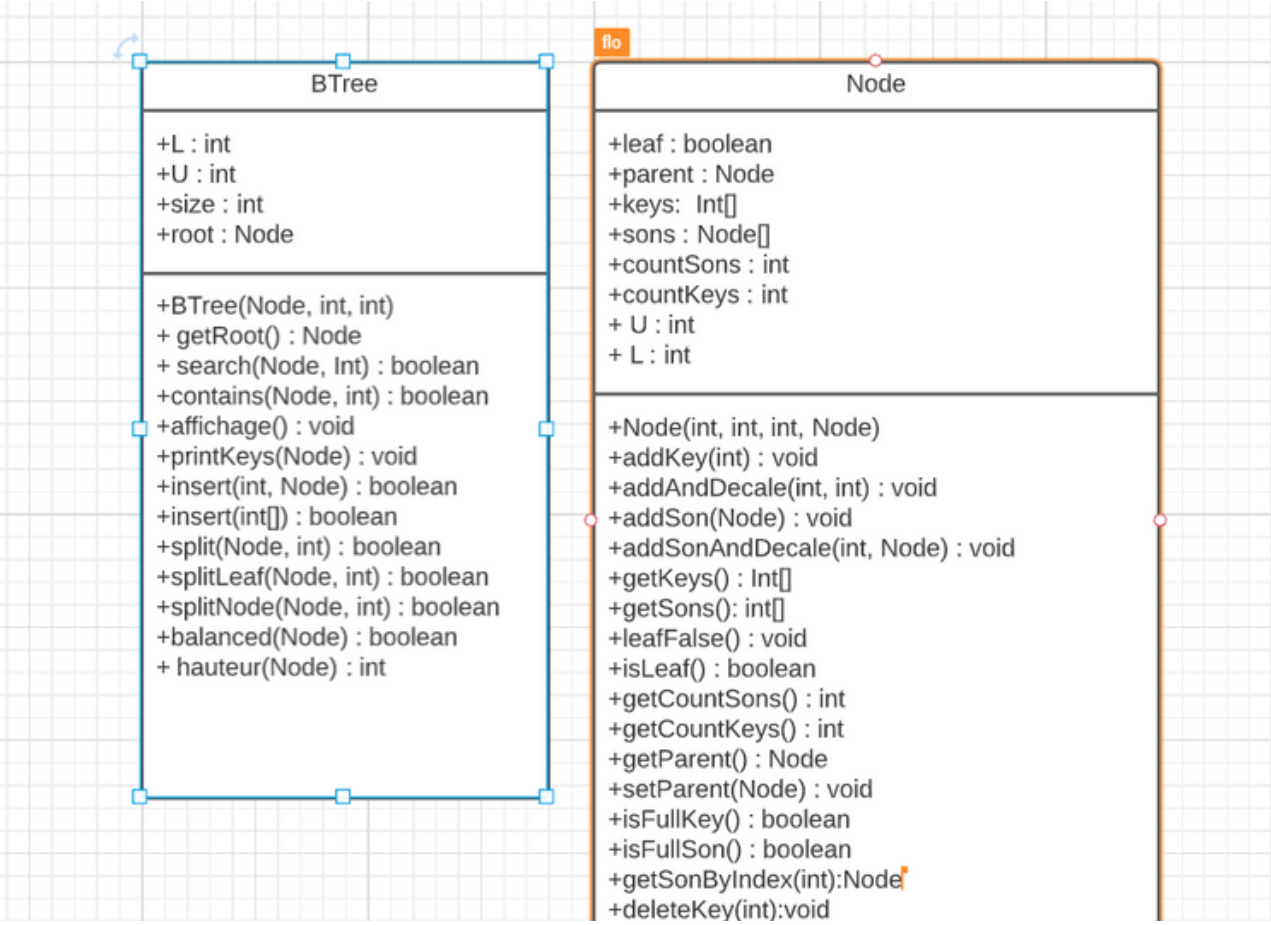
De plus, la construction des arbres B garantit qu'un arbre B est toujours équilibré. Chaque clé d'un nœud interne est en fait une borne qui distingue les sous-arbres de ce nœud. Par exemple, si un nœud a 3 fils — lesquels constituent les racines respectives de trois sous-arbres : sous-arbre gauche, sous-arbre du milieu et sous-arbre droit —, alors il a 2 clés notées c1 et c2 qui délimitent les clés de chaque sous-arbre : les clés du sous-arbre gauche seront inférieures à c1 ; les clés du sous-arbre du milieu seront comprises entre c1 et c2 ; les clés du sous-arbre droit seront supérieures à c2.

Nous avons décidé de coder ce projet en Java.
Jusqu'à aujourd'hui nous avons travaillé sur l'algorithme de recherche et d'insertion.
Concernant la structure de données, nous avons une classe représentant l'arbre B, ainsi qu'une classe représentant un noeud contenu dans l'arbre B.
Nous avons rencontrés un problème lors de la définition de l'ensemble des fils d'un noeud. En premier lieu nous avions choisi une ArrayList qui permettait une allocation dynamique et l'utilisation de toutes les méthodes qui y sont liées, mais cela nous posait des problèmes d'indexation lors du parcours de la liste dans une boucle. Nous avons alors décidé d'utiliser un tableau simple de Java, de taille fixée au début. Ainsi, les cases ne contenant aucune valeur aurait simplement la valeur "null".

Avancement du projet :

12/01/2022	**initial commit**
26/01/2022	**Recherche et structure**
13/01/2022	**Structure de Btree et de Node**
09/02/2022	**Début de l'insertion**
05/03/2022	**Fin de l'insertion**

Le répertoire git contient , un git ignore , un make file (make cls pour compiler , make run pour lancer le projet , make doc pour créer la documentation et make clean pour enlever la documentation et les classes du répertoire) , un dossier image avec l'image de l'uml , le pdf du rapport , un readme , et le dossier ArbreB avec le code du projet.



Contenu de Node.java :

Cette classe représente un noeud d'un arbre-B. On l'a défini avec une valeur, un nombre pour le nombre de valeur minimum, et un nombre pour le nombre de valeur maximum qui sera contenu dans chaque noeud de l'arbre.

On a un tableau de clés et de fils, ainsi que deux attributs représentant le nombre de clés et de fils contenus dans le noeud. On initialise le tableau de clés avec la valeur 0 (à changer)

On initialise le tableau de sons avec des valeurs null.

Il y a aussi un attribut boolean qui est True si le node est une feuille. Il y a un attribut node parent qui représente le parent du noeud. Ce parent est null si le noeud est la racine de l'arbre.

Contenu de Btree.java :

Cette classe représente un arbre N-Aire. On l'a défini avec un noeud racine, un nombre pour le nombre de valeur minimum, et un nombre pour le nombre de valeur maximum qui sera contenu dans chaque noeud de l'arbre.

On a, en plus comme attribut, la taille de l'arbre.

Algorithmes :

Algorithme de recherche :

- On regarde si c'est le noeud est une feuille

  -> Si oui, on regarde la valeur qu'on recherche est dans les clés du noeud

- Sinon

  -> on regarde la valeur qu'on recherche est dans les clés du noeud

  -> Si elle n'y est pas, on cherche le fils qui peut contenir la valeur qu'on cherche, et on applique l'algo de manière récursive sur ce noeud.

on increment une variable de 0 à n tant que:

n est plus petit que le nombre de clé maximum dans un noeud

et que la valeur qu on cherche est plus petite que la valeur à l index i dans le tableau de clé du noeud

et que l'index de i dans le tableau des clés soit différent de 0 (comme on initialise à 0)

  -> on appelle search avec le fils correspond (celui qui corresppond à l index i dans le tableau des fils)

Exemple d'utilisation de la recherche :

exemple de la recherche avec le nombre 6:

le noeud n'est pas une feuille donc on rendre dans le sinon

la valeur est dans le noeud donc on retourne true

exemple de la recherche avec le nombre 10:

le noeud n'est pas une feuille donc on rendre dans le sinon

la valeur n est pas dans le neoud donc on iter sur i

i=0 , i < 3 , 10 > 6 , 6 != 0 -> i=1

i =1 , i<3 , 10 > 0 , 0 n est pas différent de 0 , donc on arrete le tant que

on recommence la recherche sur le noeud 10,14

le noeud n'est pas une feuille donc on rendre dans le sinon

la valeur est dans le noeud donc on retourne true

exemple de la recherche avec le nombre 12:

le noeud n'est pas une feuille donc on rendre dans le sinon

la valeur n est pas dans le neoud donc on iter sur i

i=0 , i < 3 , 12 > 6 , 6 != 0 -> i=1

i =1 , i<3 , 12 > 0 , 0 n est pas différent de 0 , donc on arrete le tant que

on recommence la recherche sur le noeud 10,14

le noeud n'est pas une feuille donc on rendre dans le sinon

la valeur n est pas dans le neoud donc on iter sur i

i=0 , i< 3 , 12 > 10 , 10 != 0 -> i=1

i=1 , i<3 , 12 < 14 -> donc on arrete i=1

on fait la recherche sur le noeud index 0 dans le tableau de fils

le noeud est une feuille , et il contient 12 donc on retourne true

(si la valeur n'est pas dans l'arbre , on retrouvera le même résulat sauf qu'on trouvera pas la valeur dans le noeud et on retournera false

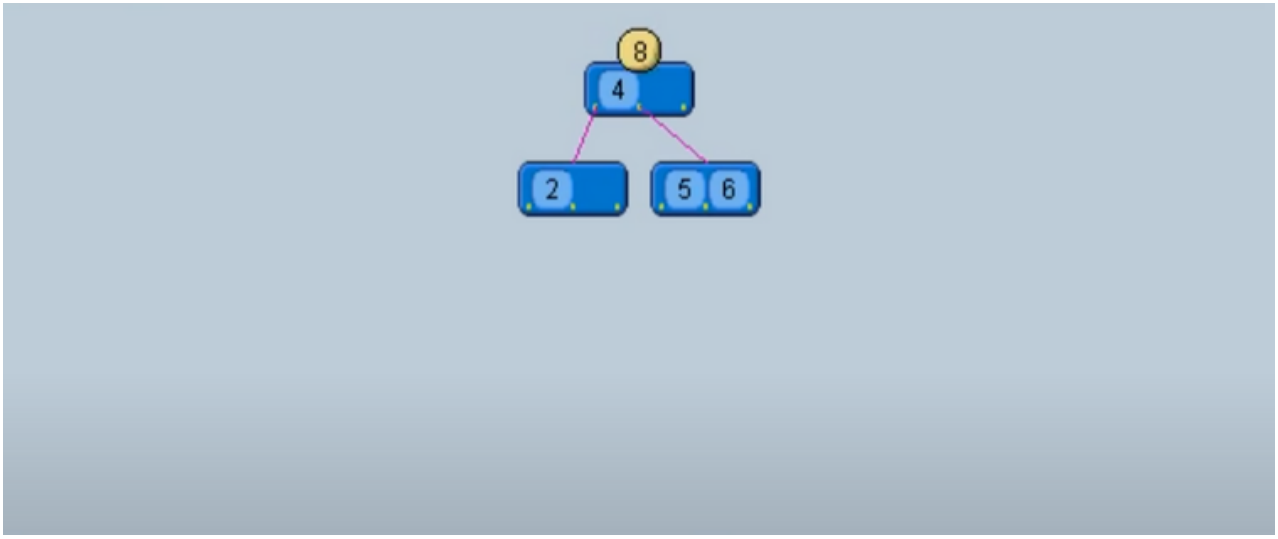
Algorithme d'insertion :

-> Si le noeud est une feuille
-> Si le noeud est plein
on split le noeud en question
-> Sinon on ajoute tout simplement la clé au tableau de clés du noeud
-> Sinon on cherche dans quel enfant il faut continuer l'insertion récursivement

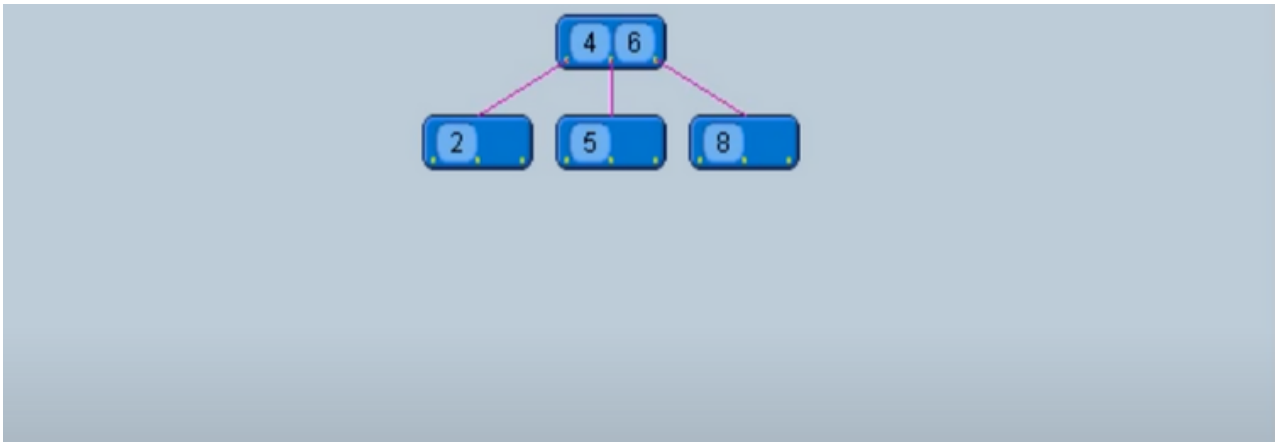
Algorithme Split :

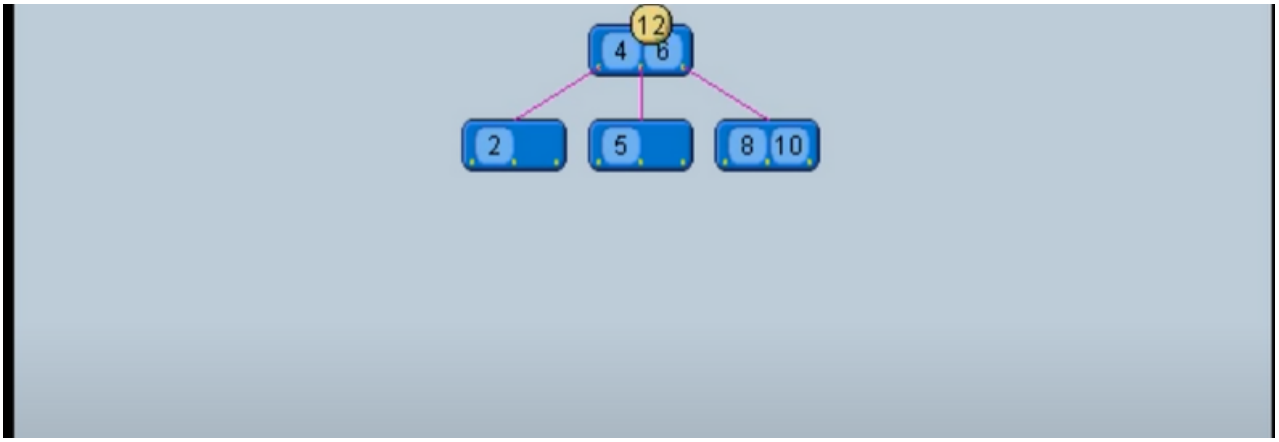
->Si le noeud est une feuille
-> Si le noeud a un parent
-> Si ce parent n'est pas plein (rempli de clés)
On ajoute valeur médiane du noeud au parent
On crée un second noeud, qui aura le même parent, avec les valeurs supérieures à la médiane
-> Sinon
On split le parent
On ajoute valeur médiane du noeud au parent
On crée un second noeud, qui aura le même parent, avec les valeurs supérieures à la médiane
-> Sinon
On crée un nouveau noeud qui sera le parent du noeud actuel
On ajoute valeur médiane du noeud au parent
On crée un second noeud, qui aura le même parent, avec les valeurs supérieures à la médiane
-> Sinon
-> Si le noeud a un parent
-> Si ce parent n'est pas plein (rempli de clés)
On ajoute valeur médiane du noeud au parent
On crée un second noeud, qui aura le même parent, avec les clés et les enfants supérieurs à la médiane
-> Sinon
On split le parent
On ajoute valeur médiane du noeud au parent
On crée un second noeud, qui aura le même parent, avec les clés et les enfants supérieurs à la médiane
-> Sinon
On crée un nouveau noeud qui sera le parent du noeud actuel
On ajoute valeur médiane du noeud au parent
On crée un second noeud, qui aura le même parent, avec les clés et les enfants supérieures à la médiane

Exemples d'insertion :



Insérer 8 :
- Le noeud racine n'est pas une feuille, donc on regarde dans quel enfant on va chercher à insérer 8
- On veut insérer 8 dans le noeud à droite mais il est plein alors on le split
- Cela fait remonter la valeur médiane du noeud au parent (ici c'est 6).
- Voici l'état de l'arbre après insertion de 8





insérer 12 :
-La racine n'est pas une feuille donc on cherche l'enfant dans lequel on va continuer l'insertion
- On veut insérer dans la feuille de droite mais celle-ci est pleine alors on split
-Ce split va entrainer le split du noeud parent
-Voici l'état de l'arbre à la fin de l'insertion