

L'objectif de ce TP est d'implémenter un arbre B en mémoire primaire. Nous nous concentrerons sur l'implémentation de deux méthodes principales : la recherche de clé et l'insertion de paire (clé, valeur).

Le fichier `ArbreB.java` contient une classe principale qui représente un arbre B comme vu en cours. Le paramètre `M` représente le nombre de clés qui sont stockées dans les noeuds internes et dans les feuilles. Ce paramètre doit toujours être supérieur ou égal à 2. La classe `Noeud` est utilisée pour représenter à la fois les noeuds internes et les feuilles. Une instance de cette classe est une feuille si et seulement si `estFeuille` a pour valeur `true`.

En fin de fichier, vous trouverez une méthode `main` qui contiendra vos tests. Vous avez déjà à votre disposition deux tests de création d'arbre B de petite et grande taille. Pour exécuter le fichier, il suffit de taper la commande suivante :

```
java ArbreB.java
```

Questions

Méthodes outils

Question 1. Compléter la méthode `positionPour` qui retourne l'indice auquel une clé devrait être insérée dans un noeud interne ou une feuille. On renverra `M` dans le cas, on devrait insérer dans un noeud plein après la dernière clé.

Question 2. Compléter la méthode `decalerDeUn` qui décale dans un noeud non plein les clés d'une case vers la droite pour laisser la position passée en argument vide. On décalera les valeurs dans une feuille de la même manière. Pour les noeuds internes, seules les enfants à droite des clés décalées sont déplacés.

Question 3. Compléter la méthode `insérerA`, qui insère, dans une feuille, une clé et une valeur à une position donnée et dans un noeud interne, insère une clé à une position donnée et un enfant à sa droite.

Ajout simple et recherche

Question 4. Faire une première version de la méthode `ajouter` qui fait appel à une nouvelle méthode auxiliaire récursive `ajouterRec(Noeud n, String cle, String valeur)`. Cette première version doit permettre d'ajouter une association clé, valeur dans un arbre B dont aucune feuille n'est pleine (aucun split ne peut survenir).

Assurez-vous que votre fonction fonctionne correctement à l'aide d'exemples de votre choix construits dans le `main`.

Question 5. Compléter la méthode `recherche` qui retourne la valeur correspondant à une clé dans l'arbre B. Utilisez aussi une fonction auxiliaire récursive.

Ajout et recherche simple

Question 6. En vous inspirant de la méthode `splitInterne`, compléter la fonction `splitFeuille`, qui applique un split sur une feuille pleine au moment de l'ajout d'une clé et de sa valeur. La méthode retourne une paire contenant la clé médiane à faire remonter dans le noeud interne "parent" et la nouvelle feuille issus du split à droite de la valeur médiane. On transformera le noeud sur lequel s'applique le split en le noeud à gauche de la valeur médiane.

Question 7. Proposez une version complète de `ajouter` et `ajouterRec`, qui applique des splits sur les feuilles et les noeuds internes quand cela est nécessaire.

Question 8. Testez votre code avec `testSimple` et `testCommunes`. Ce dernier test charge la liste des communes de France dans un arbre B (voir le fichier `communes.txt`).

Changez de valeur de `M` pour tester votre code.

Bonus : recherche par intervalle

Question 9. Modifiez la classe `Noeud` et le reste de votre code pour pouvoir coder une fonction de recherche par intervalle de clés efficace.

Question 10. Codez une méthode de recherche par préfixe de clé.