## Martin-Dipp, Neveu Pierre - Rapport B-Tree

Classe Node et NodeComparator

La classe « Node » est une classe interne de la classe BTree. Elle représente un nœud dans une structure de B-Tree. Un nœud, contenant des clés, peut être soit une feuille, contenant un nœud parent, soit un nœud interne avec des références vers ses enfants et un nœud parent, soit une racine, avec des références vers ses enfants.

Nous avons choisi d’utiliser des listes pour représenter les clés et les enfants et pouvoir les trier dans l’ordre croissant.

Afin de créer d’implémenter la notion d’infériorité et de supériorité entre deux noeuds, nous avons choisi de créer une classe « NodeComparator » qui implémente la classe native java “Comparator”. Cette classe est utilisée pour comparer deux nœuds dans le contexte de l'insertion dans la liste des enfants d'un nœud. La comparaison est basée sur les valeurs maximales et minimales des clés des nœuds. Si la valeur maximale du premier nœud n1 est inférieure à la valeur minimale du deuxième n2, alors n1 est inférieur à n2 (supérieur dans le cas contraire . Si les valeurs maximales et minimales sont égales, n1 et n2 sont égaux. Cette comparaison est utilisée pour ordonner les enfants d'un nœud dans l'ordre croissant lors de l'insertion.

Classe Btree - Insertion

L’insertion est une méthode récursive qui utilise la recherche ainsi que la méthode “splitNode”. Elle cherche d’abord le nœud où insérer la clef puis vérifie s' il faut le scinder en deux ou non, sinon elle insère la clé. Si il faut le scinder, la méthode calcule la médiane puis crée un deuxième nœud et procède à la distribution des anciens nœuds enfants et des anciennes clés entre le nœud et le nouveau nœud. Ensuite un test est mis en place pour savoir si le nœud parent est complet et ne peut donc pas accueillir la médiane. Alors la méthode split node est appelée récursivement et ce processus continue jusqu’à qu’il y ait besoin.

La méthode gère le cas particulier où la racine est scindée en deux et créer alors une nouvelle racine en lui assignant la médiane comme clé et en référençant les deux nouveaux noeuds comme les noeuds enfants de la racine

Classe BTree – Suppression

La méthode « delete(int key) » dans la classe BTree est responsable de la suppression d'une clé donnée du B-Tree. Elle utilise une approche récursive en appelant une méthode auxiliaire « delete(Node node, int key) » qui effectue la suppression à partir d'un nœud spécifié. La première fois que cette méthode est appelée, le nœud envoyé est la racine, pour parcourir tout l’arbre.

La méthode commence par vérifier si le nœud actuel est nul, auquel cas elle retourne sans effectuer d'action. Ensuite, elle vérifie si la clé à supprimer n'est pas présente dans le nœud actuel. Si tel est le cas, elle recherche le nœud suivant dans lequel la clé pourrait être présente, et récursivement appelle « delete » sur ce nœud.

Si le nœud suivant devient vide après la suppression, la méthode vérifie s'il n'est pas la racine, puis appelle la méthode « borrowOrMerge » pour emprunter ou fusionner avec des voisins.

Si la clé à supprimer est présente dans le nœud actuel et que le nœud est une feuille, elle supprime simplement la clé. Si le nœud n'est pas une feuille et contient plus d'éléments que nécessaire, elle effectue une opération de déplacement.

Si le nœud n'est pas une feuille et ne contient pas suffisamment d'éléments, la méthode recherche le nœud le plus à droite de l'enfant de gauche et déplace sa clé la plus à droite dans le nœud actuel. Elle effectue ensuite une vérification récursive ascendante pour s'assurer que tous les ancêtres du nœud actuel sont également suffisamment remplis.

emprunter ou fusionner avec les voisins en fonction de la position du nœud par rapport à ses frères.

Les méthodes « borrowFromLeftSibling », « borrowFromRightSibling », « mergeWithLeftSibling », et « mergeWithRightSibling », sont responsables des opérations spécifiques d'emprunt ou de fusion avec les voisins.

La méthode « mergeWithParent » est appelée à chaque fois qu’on fait une fusion (merge). Elle assure, si le nœud auquel on a enlevé la clé est vide, la fusion d'un nœud avec son parent en déplaçant la clé du parent vers le nœud voisin de gauche et en incorporant les clés et les enfants du nœud voisin de droite. Elle gère également les mises à jour nécessaires après la fusion, y compris la récursion vers le parent si celui-ci devient vide.

Le code est structuré de manière à séparer clairement les différentes étapes de suppression

La méthode « borrowOrMerge » est une méthode auxiliaire qui détermine s'il faut et de gestion des cas, rendant le code lisible et compréhensible. Les noms de méthodes et de variables sont descriptifs, ce qui facilite la compréhension du fonctionnement du code. Les commentaires explicatifs sont également présents pour guider le lecteur. En utilisant des méthodes auxiliaires, le code maintient une bonne modularité et permet une compréhension plus claire de chaque étape du processus de suppression.

Check validity method

We implemented a recursive method that goes through the B-Tree node in order to check if they satisfy every condition.