INF2010

TP3-Partie 2

Analyse des différents arbres binaires de recherches

Partie 1 : analyse de l’arbre Splay

Analyse de l’arbre Splay

L'implémentation de l'arbre Splay en Java repose sur deux classes principales : SplayNode et SplayTree. La classe SplayNode définit la structure de chaque nœud de l'arbre Splay. Chaque nœud contient des références vers ses enfants de gauche et de droite, ainsi que vers son parent. Cette structure de nœud comprend également une variable "element" destinée à stocker la valeur associée au nœud. La classe SplayTree encapsule la structure de l'arbre Splay et détient une référence vers la racine de l'arbre, ainsi qu'un compteur du nombre de nœuds présents dans l'arbre. De plus, la classe SplayTree contient un ensemble de 18 méthodes regroupées en différentes catégories en fonction de leur fonctionnalité.

Les deux classes de l’implémentation de l’arbre Splay

1. Classe SplayNode:
   1. Définition de la structure de chaque nœud de l'arbre Splay.
   2. Références vers les enfants gauche et droit, ainsi que vers le parent.
   3. Variable "element" pour stocker la valeur associée au nœud.
2. Classe SplayTree:
   1. Référence vers la racine de l'arbre.
   2. Compteur du nombre de nœuds dans l'arbre.

Les 18 méthodes de l’implémentation de l’arbre Splay

1. Méthodes de Base:
   1. Constructeur *SplayTree()*: Initialisation de l'objet SplayTree.
   2. Méthode *isEmpty()*: Vérification de si l'arbre est vide.
   3. Méthode *clear()*: Réinitialisation de l'arbre.
   4. Méthode *countNodes()*: Récupération du nombre de nœuds dans l'arbre.
2. Méthode d'Insertion:
   1. Méthode *insert(int ele)*: Ajout de nouveaux éléments à l'arbre Splay.
3. Méthodes d'Élimination:
   1. Méthode *remove(int ele):* Suppression d'un élément.
   2. Méthode *remove(SplayNode node*): Suppression d'un nœud spécifique.
4. Méthodes de Rotations:
   1. Méthode *makeLeftChildParent(SplayNode c, SplayNode p):* Rotation pour placer un nœud à gauche en tant que parent.
   2. Méthode *makeRightChildParent(SplayNode c, SplayNode p):* Rotation pour placer un nœud à droite en tant que parent.
5. Méthodes de Splaying:
   1. Méthode *splay(SplayNode x):* Effectue les rotations nécessaires pour placer le dernier nœud accédé à la racine.
6. Méthodes de Recherche:
   1. Méthode *search(int val):* Détermine la présence d'un élément dans l'arbre et effectue un splay du nœud accédé pour le placer à la racine.
7. Méthodes de Parcours de l'Arbre:
   1. Méthodes pour réaliser un parcours en en-ordre (*inorder()*, *inorder(SplayNode r)*), pre-ordre (*preorder()*, *preorder(SplayNode r)*) et post-ordre (*postorder()*, *postorder(SplayNode r)*) de l’arbre Splay.

Comparaison des découvertes

L'implémentation des deux classes qui constituent l'arbre Splay, ainsi que l'arbre binaire de recherche, semble se conformer aux principes et concepts présentés à la page 137 de la troisième édition du manuel "Data Structure and Analysis in Java" rédigé par Mark Allen Weiss. Il s'agit d'un arbre binaire de recherche effectuant des rotations consécutives pour maintenir son équilibre et pour amener les derniers éléments accédés à la racine.

Bien que les arbres AVL et les arbres Splay effectuent tous deux des rééquilibrages par rotation après une opération d'insertion ou de suppression, l'arbre Splay se distingue par une caractéristique unique. À chaque accès à un élément, il procède à une série de rotations pour déplacer cet élément vers la racine. Cette caractéristique est représentée par la méthode "Splay" dans l'implémentation. Les arbres BST, AVL et Splay partagent tous la caractéristique commune d'être des arbres de recherche binaire. Par conséquent, ils suivent la règle fondamentale selon laquelle les éléments du sous-arbre gauche sont plus petits que le parent, tandis que les éléments du sous-arbre droit sont plus grands que le parent.

Les arbres AVL et les arbres Splay implémentés partagent l'objectif commun de maintenir un équilibre pour éviter le pire cas où l'arbre se transformerait en une liste chaînée de complexité O(n). Cependant, l'implémentation de l'arbre Splay tire parti de la notion qu'un élément récemment accédé a de fortes chances d'être réaccédé. Par conséquent, il est avantageux de rapprocher cet élément de la racine de l'arbre, ce qui justifie l'utilisation des opérations de splaying. Ainsi, l’arbre Splay n’empêche pas totalement la possibilité du pire cas O(n), mais diminue les probabilités de son arrivé et on améliore l’efficacité réel d’accès aux éléments de l’arbre

Partie 2 : Performance des arbres binaires BST, AVL et Splay

Nous avons effectué des tests de performance temporelle sur les arbres binaires de recherche (BST), les arbres AVL et les arbres Splay en ce qui concerne les opérations d'insertion, de suppression et de recherche d'un élément. Voici les résultats obtenus :

* Temps de retrait (cas normal) :
  + Arbre BST : 2400 ns
  + Arbre AVL : 3900 ns
  + Arbre Splay : 3100 ns
* Temps de retrait (pire cas) :
  + Arbre BST : 4600 ns
  + Arbre AVL : 5800 ns
  + Arbre Splay : 5900 ns
* Temps d'insertion (cas normal) :
  + Arbre BST : 4600 ns
  + Arbre AVL : 8900 ns
  + Arbre Splay : 4400 ns
* Temps d'insertion (pire cas) :
  + Arbre BST : 6400 ns
  + Arbre AVL : 30000 ns
  + Arbre Splay : 11500 ns
* Temps de recherche (pire cas) :
  + Arbre BST : 6100 ns
  + Arbre AVL : 2400 ns
  + Arbre Splay : 4000 ns
* Temps de recherche (cas normal) :
  + Arbre BST : 1600 ns
  + Arbre AVL : 1100 ns
  + Arbre Splay : 2000 ns

Il est important de noter que les pires temps pour les arbres AVL et Splay se produisent lorsqu'ils doivent effectuer un rééquilibrage important. Dans le cas de l'arbre Splay, les pires cas se produisent lors d'accès consécutifs à des données très éloignées les unes des autres, ce qui confirme la théorie. Cependant, la complexité de l'arbre AVL reste en O(logn) même dans les pires cas. Les pires cas des arbres BST et Splay sont en O(n), cependant, en situation réelle, la probabilité d'accéder à des éléments très éloignés les uns des autres est relativement faible, ce qui diminue la probabilité d'occurrence d'un pire cas pour l'arbre Splay.