Arbre Binaire de Recherche

Yann Strozecki
yann.strozecki@uvsq.fr

Décembre 2016

Organisation

- Rendez-moi les DM !
- C'est le dernier CM.
- Il reste encore deux TD. Le dernier TD sera le mardi 3 janvier 9h45-13h pour le gr2 en salle 22 et 13h30-16h45 pour le gr1.
- Les étudiants qui ont été absent au contrôle continu doivent venir me voir à la fin du cours où m'envoyer un mail rapidement pour organiser un rattrapage.

La structure de donnée dictionnaire

C'est une structure de donnée qui stocke des éléments de type quelconque identifiés par une clé unique entière et qui permet trois opérations :

- recherche d'un élément
- insertion d'un élément
- suppression d'un élément

Elle permet de représenter des ensembles ou des fonctions (associative array). Même utilisation que les tables de hachage.

La structure de donnée dictionnaire

C'est une structure de donnée qui stocke des éléments de type quelconque identifiés par une clé unique entière et qui permet trois opérations :

- recherche d'un élément
- insertion d'un élément
- suppression d'un élément

Elle permet de représenter des ensembles ou des fonctions (associative array). Même utilisation que les tables de hachage.

Arbre binaire de Recherche

Un arbre binaire de recherche vérifie les conditions suivantes :

- la clé d'un noeud est supérieure aux clés dans le sous-arbre gauche
- la clé d'un noeud est inférieure aux clés dans le sous-arbre droit
- les sous-arbres droit et gauche sont des arbres binaires de recherche

On notera n la taille de l'arbre et h sa hauteur.

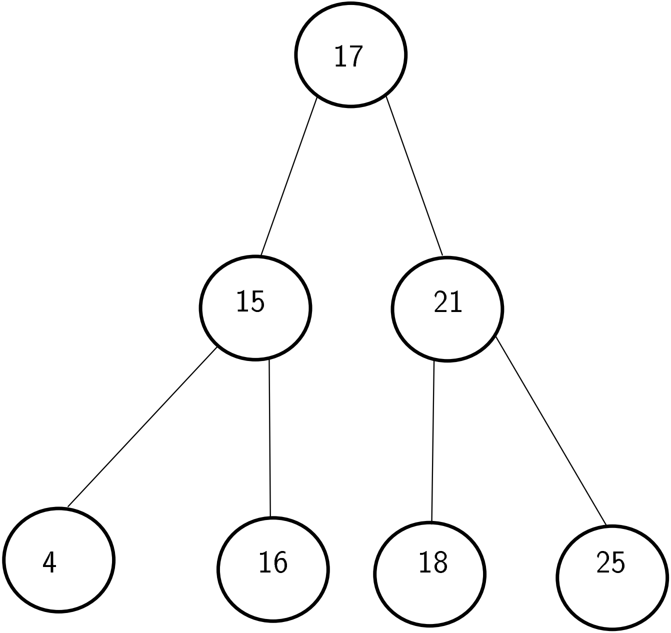
Arbre binaire de Recherche

Un arbre binaire de recherche vérifie les conditions suivantes :

- la clé d'un noeud est supérieure aux clés dans le sous-arbre gauche
- la clé d'un noeud est inférieure aux clés dans le sous-arbre droit
- les sous-arbres droit et gauche sont des arbres binaires de recherche

On notera n la taille de l'arbre et h sa hauteur.

Un exemple



Recherche

Algorithme 1 Recherche d'un élément

```
Recherche(r : Nœud, clé : Entier)
▷ Entrée : r (la racine d'un arbre)
▷ Sortie : le noeud qui contient la clé
 Début
      si (r.clé == clé)
           retourner r
      si (r.clé < clé)
           retourner Recherche(r.filsg,clé)
      sinon
           retourner Recherche(r.filsd,clé)

 Fin
```

Complexité en $O(h)$.

Recherche

Algorithme 2 Recherche d'un élément

```
Recherche(r : Nœud, clé : Entier)
▷ Entrée : r (la racine d'un arbre)
▷ Sortie : le noeud qui contient la clé
 Début
     si (r.clé == clé)
          retourner r
     si (r.clé < clé)
          retourner Recherche(r.filsg,clé)
     sinon
          retourner Recherche(r.filsd,clé)

 Fin
```

Complexité en $O(h)$.

Insertion

Algorithme 3 Insertion d'un élément

Insertion(r : Nœud, clé : Entier)
▷ ***Entrée : r (la racine d'un arbre)***
▷ ***Sortie : la racine de l'arbre***

```
Début
    si (r == NIL)
        r = CréerNœud(clé)
    si (r.clé < clé)
        r.filsg = Insertion(r.filsg,clé)
    si (r.clé > clé)
        r.filsd = Insertion(r.filsd,clé)
    retourner r

Fin
```

La complexité de cet algorithme est en $O(h)$.

Insertion

Algorithme 4 Insertion d'un élément

```
Insertion(r : Nœud, clé : Entier)
▷ Entrée : r (la racine d'un arbre)
▷ Sortie : la racine de l'arbre
 Début
     si (r == NIL)
         r = CréerNœud(clé)
     si (r.clé < clé)
         r.filsg = Insertion(r.filsg,clé)
     si (r.clé > clé)
         r.filsd = Insertion(r.filsd,clé)
     retourner r

 Fin
```

La complexité de cet algorithme est en $O(h)$.

Suppression

Il y a trois cas distincts :

- Le noeud à supprimer est une feuille
- Le noeud à supprimer a un seul descendant
- Le noeud à supprimer a deux feuilles

Dans le dernier cas il faut le remplacer par un autre noeud. Lequel ?

Suppression

Il y a trois cas distincts :

- Le noeud à supprimer est une feuille
- Le noeud à supprimer a un seul descendant
- Le noeud à supprimer a deux feuilles

Dans le dernier cas il faut le remplacer par un autre noeud. Lequel ?

Deux possibilités ...

Suppression

Il y a trois cas distincts :

- Le noeud à supprimer est une feuille
- Le noeud à supprimer a un seul descendant
- Le noeud à supprimer a deux feuilles

Dans le dernier cas il faut le remplacer par un autre noeud. Lequel ?

Deux possibilités ...

Suppression(II)

On supprime le sommet v qui a deux enfants :

1. Trouver le sommet w de valeur plus petite que celle de v et la plus grande possible.
2. Supprimer w, facile car il a au plus un enfant.
3. Remplacer v par w.

On peut vérifier que cette opération se fait en temps $O(h)$ et préserve la propriété d'ABR.

Suppression(II)

On supprime le sommet v qui a deux enfants :

1. Trouver le sommet w de valeur plus petite que celle de v et la plus grande possible.
2. Supprimer w, facile car il a au plus un enfant.
3. Remplacer v par w.

On peut vérifier que cette opération se fait en temps $O(h)$ et préserve la propriété d'ABR.

On peut aussi faire l'énumération des éléments dans l'ordre en $O(n)$ (tableau).

Suppression(II)

On supprime le sommet v qui a deux enfants :

1. Trouver le sommet w de valeur plus petite que celle de v et la plus grande possible.
2. Supprimer w, facile car il a au plus un enfant.
3. Remplacer v par w.

On peut vérifier que cette opération se fait en temps $O(h)$ et préserve la propriété d'ABR.

On peut aussi faire l'énumération des éléments dans l'ordre en $O(n)$ (tableau).

Comparaison des implémentations d'un dictionnaire

Structure	Insertion	Suppression	Recherche
Tableau	$O(1)$	$O(n)$	$O(n)$
Tableau trié	$O(n)$	$O(n)$	$O(\log(n))$
Liste	$O(1)$	$O(n)$	$O(n)$
ABR	$O(h)$	$O(h)$	$O(h)$

Mais que vaut h en général ?

Comparaison des implémentations d'un dictionnaire

Structure	Insertion	Suppression	Recherche
Tableau	$O(1)$	$O(n)$	$O(n)$
Tableau trié	$O(n)$	$O(n)$	$O(\log(n))$
Liste	$O(1)$	$O(n)$	$O(n)$
ABR	$O(h)$	$O(h)$	$O(h)$

Mais que vaut h en général ?

Équilibre

Problème de complexité des opérations de base.

Quand l'arbre est déséquilibré, on peut avoir h de l'ordre de n (arbre filiforme).

Plusieurs solutions :

- randomisation des clés, bonne hauteur moyenne
- arbres rouges noirs
- les arbres binaires équilibrés ou AVL

Équilibre

Problème de complexité des opérations de base.

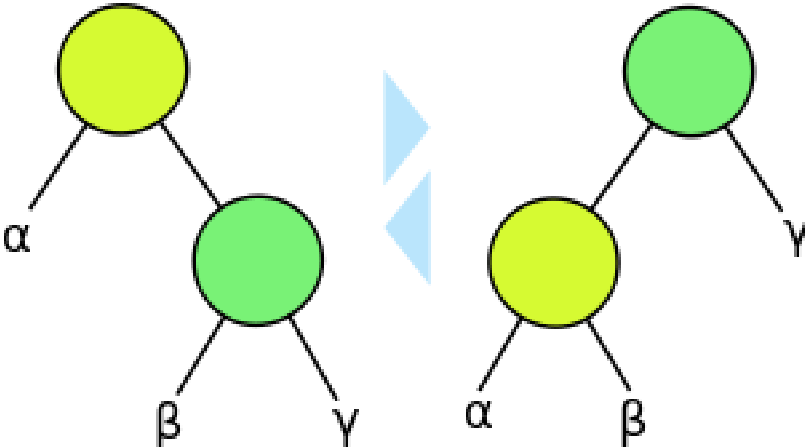
Quand l'arbre est déséquilibré, on peut avoir h de l'ordre de n (arbre filiforme).

Plusieurs solutions :

- randomisation des clés, bonne hauteur moyenne
- arbres rouges noirs
- les arbres binaires équilibrés ou AVL

Comment garder son équilibre

Opération de rotation.
Pour équilibrer un arbre on va appliquer des rotations.



AVL

Dans chaque noeud de l'arbre on stocke le facteur d'équilibre : hauteur du sous arbre gauche moins hauteur du sous-arbre droit.
Dans un AVL on veut que le facteur d'équilibre soit compris dans $\{-1, 0, 1\}$.

Lemma

L'opération de rotation préserve les propriétés d'un ABR.

Lemma

L'opération de rotation quand γ est plus profond que β et α améliore l'équilibrage de l'arbre.

AVL

Dans chaque noeud de l'arbre on stocke le facteur d'équilibre : hauteur du sous arbre gauche moins hauteur du sous-arbre droit.
Dans un AVL on veut que le facteur d'équilibre soit compris dans $\{-1, 0, 1\}$.

Lemma

L'opération de rotation préserve les propriétés d'un ABR.

Lemma

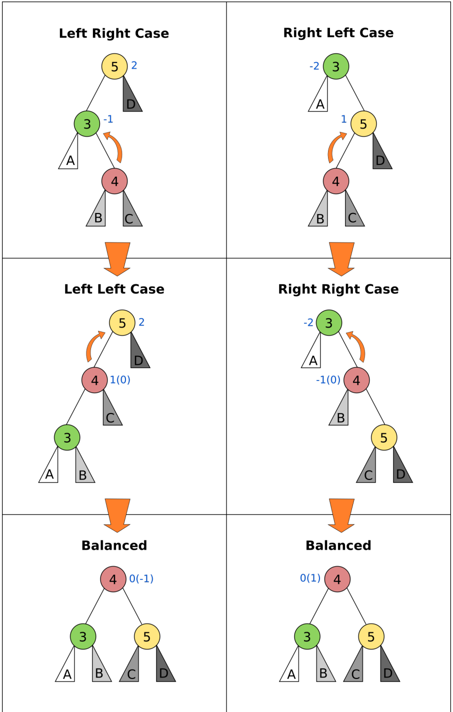
L'opération de rotation quand γ est plus profond que β et α améliore l'équilibrage de l'arbre.

Insertion

On insère le noeud comme dans un ABR.

Dans un deuxième temps on corrige les déséquilibres introduits par l'insertion. Il y a quatre cas décrits dans le slide suivant.

Un exemple au tableau d'insertion.
Complexité en fonction de la hauteur ?


Left Right Case
5
2
D
3
-1
A
4
B
C
Right Left Case
-2
3
A
1
5
D
4
B
C
Left Left Case
5
2
D
4
1(0)
C
3
A
B
Right Right Case
-2
3
A
-1(0)
4
B
5
C
D
Balanced
4
0(-1)
3
5
A
B
C
D
Balanced
0(1)
4
3
5
A
B
C
D

Suppression

On supprime normalement un noeud et on remonte jusqu'à la racine en rééquilibrant.

Intérêt des arbres AVL. Les complexités des fonctions d'insertion, suppression et recherche sont en $O(h)$ et on a en plus :

Théorème

Dans un AVL de hauteur h et avec n sommets on a

$$h < 1,5 \log(n)$$

Suppression

On supprime normalement un noeud et on remonte jusqu'à la racine en rééquilibrant.

Intérêt des arbres AVL. Les complexités des fonctions d'insertion, suppression et recherche sont en $O(h)$ et on a en plus :

Théorème

Dans un AVL de hauteur h et avec n sommets on a

$$h < 1,5\log(n)$$