**Description du TP**

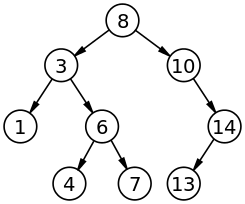
Ce TP a pour objectif de vous faire manipuler différentes structures de type arbre. Il existe bien d’autres types de structures, que vous pouvez approfondir via des recherches sur le sujet. Les énoncés des exercices contiennent un certain nombre d’informations mais elles sont rarement suffisantes pour répondre aux questions. Ils doivent donc essentiellement vous servir de pistes pour vos recherches et votre réflexion.

1. **Les arbres binaires de recherches (ABR)**

Un arbre binaire de recherche est un arbre binaire (chaque nœud a au plus 2 fils) et tel que tout nœud a une valeur supérieure à l’ensemble des valeurs des nœuds de son sous-arbre gauche et inférieure à l’ensemble des nœuds de son sous-arbre droit.

1. INTRODUCTION

- D’après la définition, donner plusieurs exemples d’ABR.



Arbre binaire de recherche

- Voyez-vous d’ores et déjà des applications pratiques à cette structure ?

- Algorithme Min/Max

-

1. INSERER

L'insertion d'une valeur dans un arbre binaire est un processus récursif simple. Il suffit tout simplement d'appliquer la définition d'un arbre binaire de recherche.

- Écrire l’algorithme d’insertion d’un élément dans un arbre binaire de recherche.

static Arbre inserer(int x, Arbre a)

{

if (a == null)

return new Arbre(null, x, null);

if (x < a.contenu)

{

Arbre b = inserer(x, a.filsG);

return new Arbre(b, a.contenu, a.filsD);

}

else if (x > a.contenu)

{

Arbre b = inserer(x, a.filsD);

return new Arbre(a.filsG, a.contenu, b);

}

return a;

}

- Donner la complexité de cet algorithme au pire des cas.

C’est la complexité 0(1)

- Construire l’ABR résultant des insertions successives suivantes :

21, 9, 12, 34, 36, 35, 5, 8, 1, 25

21

9

12

34

36

25

5

8

1

35

1. SUPPRIMER

- Quels sont les différents cas possibles lors de la suppression d’un nœud ?

* Voir réponse question suivante

- Comment procéder dans chacun des cas pour conserver les propriétés des ABR ?

* si le nœud s est une feuille, alors on l’élimine;
* si le nœud s possède un seul fils, on élimine s et on « remonte » ce fils.
* si le nœud s possède deux fils, on cherche le prédécesseur t de s. Celui-ci n’a pas de fils droit. On remplace le contenu de s par le contenu de t, et on élimine t.

- Supprimer l’élément 36 de l’ABR précédent. Donner l’ABR résultant.

static Arbre supprimerRacine(Arbre a)

{

if (a.filsG == null)

return a.filsD;

if (a.filsD == null)

return a.filsG;

Arbre b = a.filsG;

if (b.filsD == null)

{ // cas (i)

a.contenu = b.contenu;

a.filsG = b.filsG;

}

else

{ // cas (ii)

Arbre p = avantDernierDescendant(b);

Arbre f = p.filsD;

a.contenu = f.contenu;

p.filsD = f.filsG;

}

return a;

}

static Arbre avantDernierDescendant(Arbre a)

{

while (a.filsD.filsD != null)

a = a.filsD;

return a;

}

21

9

12

34

35

25

5

8

1

- Supprimer l’élément 9 de l’ABR précédent. Donner l’ABR résultant.

static Arbre supprimerRacine(Arbre a)

{

if (a.filsG == null)

return a.filsD;

if (a.filsD == null)

return a.filsG;

Arbre b = a.filsG;

if (b.filsD == null)

{ // cas (i)

a.contenu = b.contenu;

a.filsG = b.filsG;

}

else

{ // cas (ii)

Arbre p = avantDernierDescendant(b);

Arbre f = p.filsD;

a.contenu = f.contenu;

p.filsD = f.filsG;

}

return a;

}

static Arbre avantDernierDescendant(Arbre a)

{

while (a.filsD.filsD != null)

a = a.filsD;

return a;

}

21

5

12

34

35

25

1

8

1. PARCOURS

Il existe 3 types de parcours d’un ABR : (attention, les définitions ci-dessous sont données de manière récursive, autrement dit en considérant à chaque fois le sous-arbre en cours de traitement).

Parcours Infixe

On parcourt dans l'ordre :

\* le sous-arbre gauche

\* la racine

\* le sous-arbre droit

Parcours Préfixe

\* la racine

\* le sous arbre gauche

\* le sous arbre droit

Parcours Postfixe

\* le sous arbre gauche

\* le sous arbre droit

\* la racine

- Donner le parcours infixe de l’ABR réalisé en b)

static void parcoursInfixe(Arbre a)

{

if (a == null)

return;

parcoursInfixe(a.filsG);

System.out.print(a.contenu + " ");

parcoursInfixe(a.filsD);

}

- Donner le parcours préfixe de ce même ABR

static void parcoursPréfixe(Arbre a)

{

if (a == null)

return;

System.out.print(a.contenu + " ");

parcoursPréfixe(a.filsG);

parcoursPréfixe(a.filsD);

}

- Donner le parcours postfixe de ce même ABR

static void parcoursPostfixe (Arbre a)

{

if (a == null)

return;

parcoursPostfixe(a.filsG);

parcoursPostfixe(a.filsD);

System.out.print(a.contenu + " ");

}

- Que pouvez-vous remarquer sur le résultat du parcours infixe ?

Dans un arbre binaire de recherche, le parcours infixe fournit les contenus des nœuds en ordre croissant.

1. IMPLEMENTATION

- Donner une implémentation possible des ABR à l’aide d’un tableau statique.

- Donner une implémentation possible des ABR à l’aide d’une structure plus évoluée.

1. **Les arbres rouge-noir (ARN)**

Un ARN est un ABR qui satisfait en plus les propriétés suivantes :

1. Chaque nœud est soit rouge, soit noir.
2. Chaque feuille (pointeur NULL) est noire.
3. Si 1 nœud est rouge alors ses 2 fils sont noirs.
4. Chaque chemin simple reliant 1 nœud à 1 feuille descendante contient le même nombre de nœuds noirs.
5. La racine est noire.
   1. INTRODUCTION

- D’après la définition, donner plusieurs exemples d’ARN.

- Voyez-vous d’ores et déjà des applications pratiques à cette structure ?

- Tester sur un ARN l’insertion et la suppression d’une valeur en suivant les algorithmes vus pour les ABR. Que constatez-vous ?

* 1. INSERER

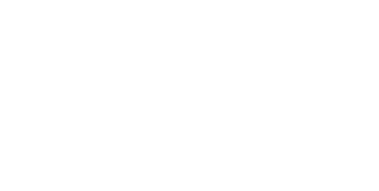
Avant de rentrer dans les détails de l’insertion :

- Un nœud inséré peut-il être colorié en noir ? Pourquoi ?

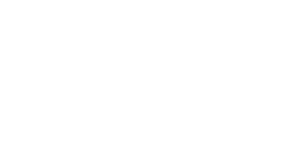
- Dans quel cas l’insertion ne pose-t-elle alors pas de problème ?

Pour les autres cas, il va falloir ajuster l’arbre par le biais de changements de couleurs et de rotations.

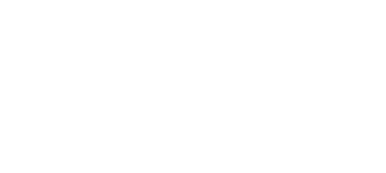
Principe des rotations :



- Sur l’ABR ci-dessous : effectuer la rotation à droite de centre le nœud de valeur 3 puis une rotation à gauche de centre le nœud de valeur 14.



- Sur l’ABR ci-dessous : à l’aide de rotations dont vous préciserez le centre et le sens, amener le nœud de valeur 9 à la racine.



- Les arbres résultants sont-ils toujours des ABR ?

Revenons à l’insertion d’un élément dans un ARN :

Méthode générale : pour réaliser l’insertion dans un arbre rouge noir : on applique l’insertion des arbres binaires de recherche ; puis on colorie en rouge le nouveau nœud ; et enfin on rétablit les propriétés d’arbre rouge noir.

- Que faut-il faire lorsque l’arbre avant insertion ne contenait aucun élément ?

- Que faut-il faire lorsque le parent du nœud que l’on vient d’insérer, N, est noir ?

- On suppose désormais que le parent de N, P est rouge. Quelle est alors la couleur du grand parent G ? N possède-t-il également un oncle O ? Vérifier sur plusieurs exemples.

- Dans le cas où O est noir, montrer que vous pouvez rétablir les propriétés d’arbre rouge noir, par au plus deux rotations et deux recoloriages de nœuds. Montrer sur un exemple.

- Dernier cas : O est rouge. On remarque que P viole la condition 3 et que c’est la seule raison pour laquelle l’arbre n’est pas rouge noir.

Que peut-on faire si G est la racine de l’arbre (indication : utiliser deux recoloriages) ?

Autrement, comment peut-on repousser le problème que pose P, à un problème similaire sur un nœud plus haut dans l’arbre ?

* 1. SUPPRIMER

Même chose que pour l’insertion : on cherche un algorithme pour rétablir les propriétés des ARN après une suppression telle qu’elle a été établie pour les ABR.

- Quels sont les cas faciles ?

- Quels sont les cas difficiles ?

- Donner un algorithme de suppression.

Pour aller plus loin, étudiez de la même manière les structures suivantes :

**Les B-arbres**

**Tas binomial**

**Tas de Fibonacci**

**etc..**