

Éléments d'algorithmique : les arbres binaires

Cours 9

novembre 2020

Insertion et recherche

Objectif : obtenir une structure de donnée où l'insertion et la recherche sont efficaces.

- Pour les tableaux triés, l'insertion est en $O(n)$ et la recherche d'un élément est en $O(\log(n))$ où n est la taille du tableau.
- Pour les listes, l'insertion est en $O(1)$ et la recherche d'un élément est en $O(n)$ où n est la taille de la liste.

Arbres binaires : définition formelle

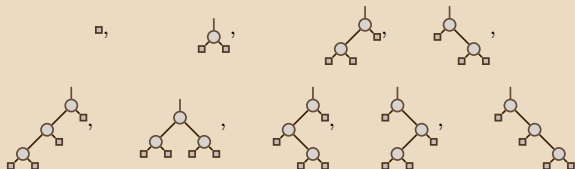
Un **arbre binaire** est une structure de données définie récursivement sur un ensemble fini de nœuds, qui

- ▶ est soit vide, noté \square ,
- ▶ soit contient un nœud racine \bigcirc attaché via deux arêtes à deux arbres binaires appelés **sous-arbre gauche** et **sous-arbre droit** de t .

La **taille** n d'un arbre binaire est le nombre de nœuds de t .

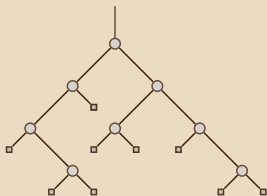
– Exemples –


Les premiers arbres binaires pour $n \leq 3$ sont



Arbres binaires : vocabulaire

– Example –



- ▶ la **racine**;
- ▶ les **nœuds** ;
- ▶ les **arêtes**.

Pour chaque nœud \circ , on a un arbre binaire gauche et un arbre binaire droit appelés respectivement **sous-arbre gauche** et **sous-arbre droit**.

Soit n un nœud d'un arbre binaire, on appelle **descendants** de n tous nœuds appartenant au sous-arbre gauche et au sous-arbre droit de n . On appelle **fils** de n les nœuds reliés directement à n .

Une **feuille** est un nœud de l'arbre sans enfant non-vide. Formellement, les descendants d'une feuille sont des arbres vides.

Type de donnée abstrait

Soit t un arbre binaire, g son sous-arbre gauche et d son sous-arbre droit.

Opérations :

- ▶ $Vide : \{\} \rightarrow ABin$
- ▶ $Nœud : ABin \times ABin \rightarrow ABin$
- ▶ $EstVide : ABin \rightarrow Booleen$
- ▶ $SAG, SAD : ABin \rightarrow ABin$

Préconditions :

- ▶ $SAD(t), SAG(t)$ sont définis si et seulement si non $EstVide(t)$

Axiomes :

- ▶ $EstVide(Vide()) = VRAI$
- ▶ $EstVide(Nœud(g, d)) = FAUX$
- ▶ $SAG(Nœud(g, d)) = g$
- ▶ $SAD(Nœud(g, d)) = d$
- ▶ $Nœud(SAG(t), SAD(t)) = t$ si non $EstVide(t)$

Taille et hauteur

Soit t un arbre binaire, g son sous-arbre gauche et d son sous-arbre droit. On définit deux fonctions sur les arbres binaires.

Le nombre de nœuds, appelé **Taille**(t) :

- ▶ $\text{Taille}(\text{Vide}) = 0$
- ▶ $\text{Taille}(\text{Nœud}(g, d)) = 1 + \text{Taille}(g) + \text{Taille}(d)$

Le nombre de nœuds d'un plus long chemin entre la racine et une feuille, appelé **Hauteur**(t) :

- ▶ $\text{Hauteur}(\text{Vide}) = 0$
- ▶ $\text{Hauteur}(\text{Nœud}(g, d)) = 1 + \max \{ \text{Hauteur}(g), \text{Hauteur}(d) \}$

Parcours dans les arbres binaires

Un **parcours** est un algorithme qui appelle une fonction f sur tous les nœuds ou sous-arbres d'un arbre.

L'**ordre** sur les nœuds dans lequel la fonction est appelée doit être spécifié.

Exemples :

- ▶ si f est une fonction d'affichage, ceci va afficher le contenu des nœuds de l'arbre selon l'ordre spécifié ;
- ▶ si f est une fonction de somme, ceci va calculer la somme des contenus des nœuds de l'arbre.

Parcours préfixe, infixe et postfixe

Parcours **préfixe** (ou **en profondeur**) :

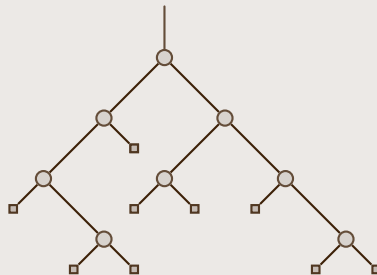
- ▶ application de f à la racine ;
- ▶ parcours préfixe du sous-arbre gauche ;
- ▶ parcours préfixe du sous-arbre droit.

Parcours **infixe** :

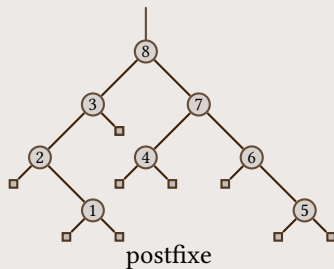
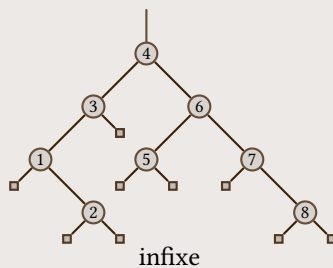
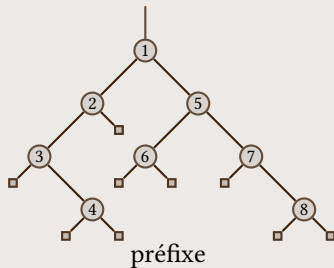
- ▶ parcours infixe du sous-arbre gauche ;
- ▶ application de f à la racine ;
- ▶ parcours infixe du sous-arbre droit.

Parcours **postfixe** :

- ▶ parcours postfixe du sous-arbre gauche ;
- ▶ parcours postfixe du sous-arbre droit ;
- ▶ application de f à la racine.



Parcours préfixe, infixe et postfixe : exemples



Arbres binaires valués

Soit T un type.

Opérations :

- ▶ $\text{Vide} : \{\} \rightarrow \text{ABinV}(T)$
- ▶ $\text{Nœud} :$
 $T \times \text{ABinV}(T) \times \text{ABinV}(T) \rightarrow \text{ABinV}(T)$
- ▶ $\text{EstVide} : \text{ABinV}(T) \rightarrow \text{Booleen}$
- ▶ $\text{SAG}, \text{SAD} : \text{ABinV}(T) \rightarrow \text{ABinV}(T)$
- ▶ $\text{Val} : \text{ABinV}(T) \rightarrow T$

Préconditions :

- ▶ $\text{SAD}(t), \text{SAG}(t), \text{Val}(t)$ sont définis si et seulement si non $\text{EstVide}(t)$

Axiomes :

- ▶ $\text{EstVide}(\text{Vide}()) = \text{VRAI}$
- ▶ $\text{EstVide}(\text{Nœud}(v, g, d)) = \text{FAUX}$
- ▶ $\text{SAG}(\text{Nœud}(v, g, d)) = g$
- ▶ $\text{SAD}(\text{Nœud}(v, g, d)) = d$
- ▶ $\text{Val}(\text{Nœud}(v, g, d)) = v$
- ▶ $\text{Nœud}(\text{Val}(t), \text{SAG}(t), \text{SAD}(t)) = t$ si non $\text{EstVide}(t)$

Arbres binaires de recherche

Un **arbre binaire de recherche** (ABR) est un arbre binaire valué par un type dont les données sont totalement comparables qui, s'il n'est pas vide, est tel que

- ▶ ses sous-arbres gauche et droit sont des ABR;
- ▶ les valeurs des nœuds du sous-arbre gauche sont inférieures à la valeur de la racine de l'arbre;
- ▶ les valeurs des nœuds du sous-arbre droit sont strictement supérieures à la valeur de la racine de l'arbre.

Opérations sur les arbres binaires de recherche

Comme pour les tableaux et les listes, plusieurs opérations sont possibles dans les arbres binaires de recherche :

- ⇒ recherche d'un élément ;
- ⇒ insertion d'un élément ;
- ⇒ suppression d'un élément.

Algorithme de recherche d'un élément dans un ABR

Algorithme EstDansABR

- ▶ Entrée : un ABR t et un élément e .
- ▶ Sortie : VRAI si e apparaît dans t , FAUX sinon.

```
si EstVide( $t$ ) alors
    renvoyer FAUX
sinon si  $e = \text{Val}(t)$  alors
    renvoyer VRAI
sinon si  $e \leq \text{Val}(t)$  alors
    renvoyer EstDansABR(SAG( $t$ ))
sinon
    renvoyer EstDansABR(SAD( $t$ ))
```

⇒ Complexité : $O(\text{Hauteur}(t))$.

Note : en pratique, $\text{Hauteur}(t) \leq \text{Taille}(t)$.

Algorithme d'insertion d'un élément dans un ABR

Algorithme InsertABR

- ▶ Entrée : un ABR t et un élément e .
- ▶ Sortie : un ABR s obtenu en remplaçant une feuille de t par un nœud contenant e .

```
si EstVide( $t$ ) alors
    renvoyer Nœud( $e$ , Vide(), Vide())
sinon si  $e \leq \text{Val}(t)$  alors
    renvoyer Nœud( $\text{Val}(t)$ , InsertABR(SAG( $t$ ),  $e$ ), SAD( $t$ ))
sinon
    renvoyer Nœud( $\text{Val}(t)$ , SAG( $t$ ), InsertABR(SAD( $t$ ),  $e$ ))
```

⇒ Complexité : $O(\text{Hauteur}(t))$.

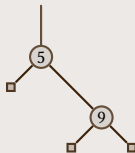
Exemple d'insertion d'un élément dans un ARB

Insertions de 5, 9, 12, 4, 1, 3, 32, 7 dans l'arbre binaire vide :



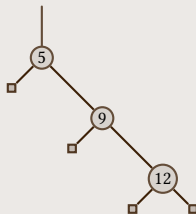
Exemple d'insertion d'un élément dans un ARB

Insertions de 5, 9, 12, 4, 1, 3, 32, 7 dans l'arbre binaire vide :



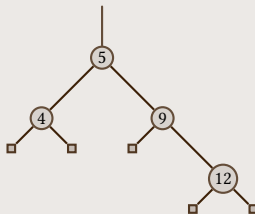
Exemple d'insertion d'un élément dans un ARB

Insertions de 5, 9, 12, 4, 1, 3, 32, 7 dans l'arbre binaire vide :



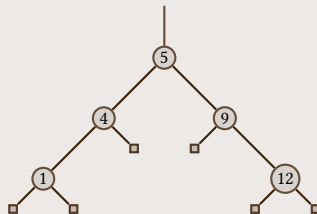
Exemple d'insertion d'un élément dans un ARB

Insertions de 5, 9, 12, 4, 1, 3, 32, 7 dans l'arbre binaire vide :



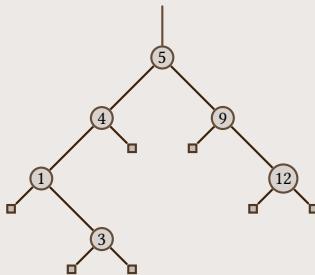
Exemple d'insertion d'un élément dans un ARB

Insertions de 5, 9, 12, 4, 1, 3, 32, 7 dans l'arbre binaire vide :



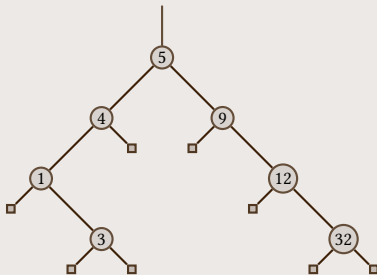
Exemple d'insertion d'un élément dans un ARB

Insertions de 5, 9, 12, 4, 1, 3, 32, 7 dans l'arbre binaire vide :



Exemple d'insertion d'un élément dans un ARB

Insertions de 5, 9, 12, 4, 1, 3, 32, 7 dans l'arbre binaire vide :



Exemple d'insertion d'un élément dans un ARB

Insertions de 5, 9, 12, 4, 1, 3, 32, 7 dans l'arbre binaire vide :

