

# Éléments d'algorithmique : les arbres binaires

## Cours 9


novembre 2023

# Insertion et recherche

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

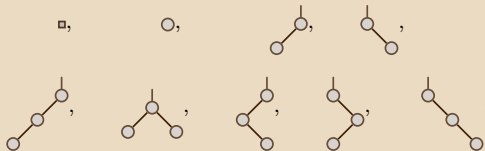
- Pour les tableaux triés, l'insertion est en  $O(n)$  et la recherche d'un élément est en  $O(\log(n))$  où  $n$  est la taille du tableau.
- Pour les listes, l'insertion est en  $O(1)$  et la recherche d'un élément est en  $O(n)$  où  $n$  est la taille de la liste.

# Arbres binaires : définition formelle

Un **arbre binaire**  $t$  de taille  $n \geq 0$  est soit une feuille  $\bigcirc$  soit un nœud  attaché via deux arêtes à deux arbres binaires appelés **sous-arbre gauche** et **sous-arbre droit** de  $t$ , où la **taille** d'un arbre binaire est le nombre de nœuds de  $t$ .

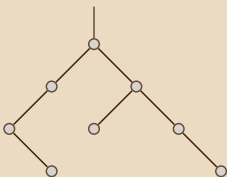
## – Exemples –



Soit  $\square$  l'arbre vide. Les premiers arbres binaires pour  $n \leq 3$  sont




# Arbres binaires : vocabulaire

## – Exemple –



- ▶ la **racine** ;
- ▶ les **nœuds**  ou nœuds internes ;
- ▶ les **feuilles** , ou nœuds externes ;
- ▶ les **arêtes**.

Pour chaque nœud , on a un arbre binaire gauche et un arbre binaire droit appelés respectivement **sous-arbre gauche** et **sous-arbre droit**. Remarquez que ces sous-arbres peuvent être vides et dans ce cas le nœud est une feuille.

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 **fil**s de  $n$  les nœuds reliés directement à  $n$ .

# Type de donnée et prototype (pseudocode)

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

Méthodes pour les arbres binaires :

- ▶ `ABin Vide ()` : renvoie l'arbre vide
- ▶ `ABin Nœud (ABin g, ABin d)` : renvoie un arbre de sous-arbres gauche et droit  $g$  et  $d$ .
- ▶ `Boolean EstVide (ABin t)` : renvoie VRAI si  $t$  est l'arbre vide.
- ▶ `ABin SAG (ABin t)` : renvoie  $g$  si  $t = (g, d)$ , et n'est pas défini sinon.
- ▶ `ABin SAD (ABin t)` : renvoie  $d$  si  $t = (g, d)$ , et n'est pas défini sinon.

# 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**) :

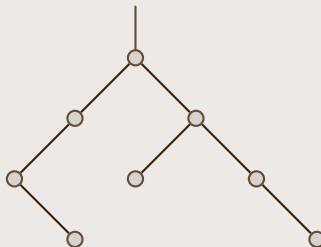
- ▶ application de  $f$  à la racine ;
- ▶ parcours préfixe du sous-arbre gauche ;
- ▶ parcours préfixe du sous-arbre droit.

Parcours **infixe** :

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

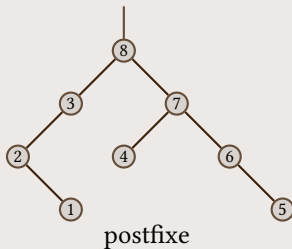
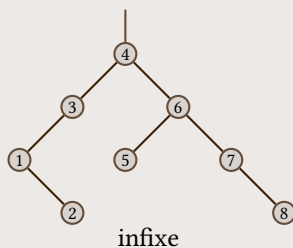
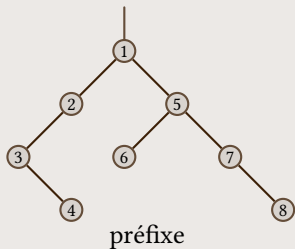
Parcours **postfixe** :

- ▶ 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 pour les valuations.

Méthodes pour les arbres binaires valués :

- ▶  $\text{ABinV}(T) \text{ Vide } ()$  : renvoie l'arbre vide
- ▶  $\text{ABinV}(T) \text{ Nœud } (T \ v, \text{ABinV}(T) \ g, \text{ABinV}(T) \ d)$  : renvoie l'arbre valué  $(v, g, d)$ .
- ▶  $\text{Boolean EstVide } (\text{ABinV}(T) \ t)$  : renvoie VRAI si  $t$  est l'arbre vide.
- ▶  $\text{ABinV}(T) \text{ SAG } (\text{ABinV}(T) \ t)$  : renvoie  $g$  si  $t = (v, g, d)$ , et n'est pas défini sinon.
- ▶  $\text{ABinV}(T) \text{ SAD } (\text{ABinV}(T) \ t)$  : renvoie  $d$  si  $t = (v, g, d)$ , et n'est pas défini sinon.
- ▶  $T \text{ Val } (\text{ABinV}(T) \ t)$  : renvoie  $v$  si  $t = (v, g, d)$ , et n'est pas défini sinon.

# 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

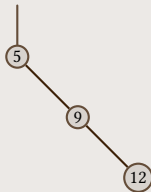
Insertions de 5, 9, 12, 4, 1, 3, 32, 7 dans l'arbre binaire vide :





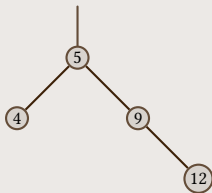
# Exemple d'insertion d'un élément dans un ARB

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



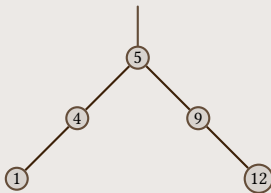
# Exemple d'insertion d'un élément dans un ARB

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



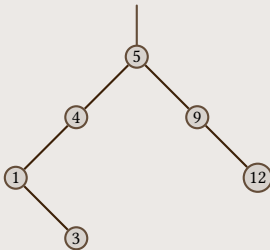
# Exemple d'insertion d'un élément dans un ARB

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



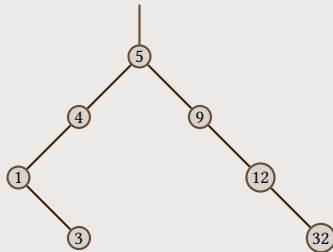
# Exemple d'insertion d'un élément dans un ARB

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 :

