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.

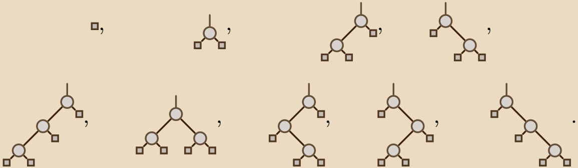
- ⋆ Pour les tableaux triés, l'insertion est en $O(n)$ et la recherche d'un élément est en $O(\log(n))$ où n est la taille du tableau.
- ⋆ Pour les listes, l'insertion est en $O(1)$ et la recherche d'un élément est en $O(n)$ où n est la taille de la liste.

Arbres binaires : définition formelle

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

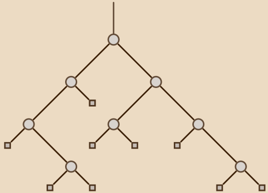
– Exemples –

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.

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.

Type de donnée abstrait

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

Opérations :

- ⋆ Vide : $\{\} \to$ ABin
- ⋆ Nœud : ABin $\times$ ABin $\to$ ABin
- ⋆ EstVide : ABin $\to$ Booleen
- ⋆ SAG, SAD : ABin $\to$ ABin

Préconditions :

- ⋆ SAD$(\mathfrak{t})$, SAG$(\mathfrak{t})$ sont définis si et seulement si non EstVide$(\mathfrak{t})$

Axiomes :

- ⋆ EstVide(Vide()) = VRAI
- ⋆ EstVide(Nœud($\mathfrak{g}$, $\mathfrak{d}$)) = FAUX
- ⋆ SAG(Nœud($\mathfrak{g}$, $\mathfrak{d}$)) = $\mathfrak{g}$
- ⋆ SAD(Nœud($\mathfrak{g}$, $\mathfrak{d}$)) = $\mathfrak{d}$
- ⋆ Nœud(SAG($\mathfrak{t}$), SAD($\mathfrak{t}$)) = $\mathfrak{t}$ si non EstVide($\mathfrak{t}$)

Taille et hauteur

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

Le nombre de nœuds, appelé Taille($\mathfrak{t}$) :

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

Le nombre de nœuds d'un plus long chemin entre la racine et une feuille, appelé Hauteur($\mathfrak{t}$) :

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

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) :

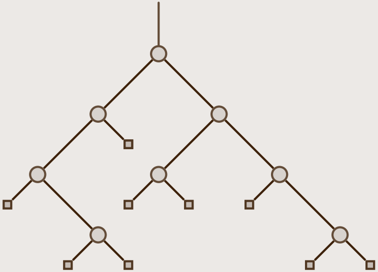
- ⋆ application de f à la racine ;
- ⋆ parcours préfixe du sous-arbre gauche ;
- ⋆ parcours préfixe du sous-arbre droit.

Parcours infixe :

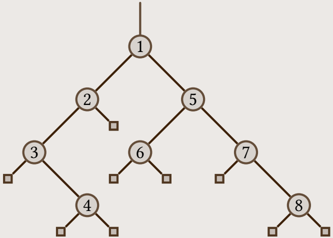
- ⋆ parcours infixe du sous-arbre gauche ;
- ⋆ application de f à la racine ;
- ⋆ parcours infixe du sous-arbre droit.

Parcours postfixe :

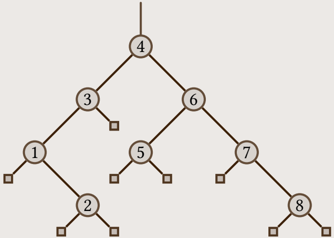
- ⋆ parcours postfixe du sous-arbre gauche ;
- ⋆ parcours postfixe du sous-arbre droit ;
- ⋆ application de f à la racine.



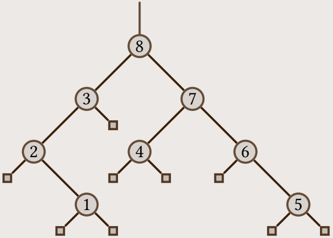
Parcours préfixe, infixe et postfixe : exemples



préfixe



infixe



postfixe

Arbres binaires valués

Soit T un type.

Opérations :

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

Préconditions :

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

Axiomes :

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

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 = Val(t) alors
       renvoyer VRAI
sinon si e <= Val(t) alors
       renvoyer EstDansABR(SAG(t))
sinon
       renvoyer EstDansABR(SAD(t))
```

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

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

Algorithme d'insertion d'un élément dans un ABR

Algorithme InsertABR

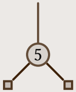
- Entrée : un ABR t et un élément e.
- Sortie : un ABR s obtenu en remplaçant une feuille de t par un nœud contenant e.

```
si EstVide(t) alors
    renvoyer Nœud(e, Vide(), Vide())
sinon si e <= Val(t) alors
    renvoyer Nœud(Val(t), InsertABR(SAG(t), e), SAD(t))
sinon
    renvoyer Nœud(Val(t), SAG(t), InsertABR(SAD(t), e))
```

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

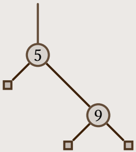
Exemple d'insertion d'un élément dans un ARB

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



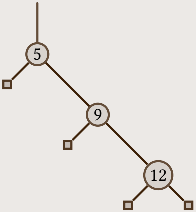
Exemple d'insertion d'un élément dans un ARB

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



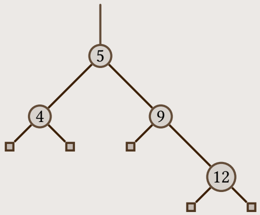
Exemple d'insertion d'un élément dans un ARB

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



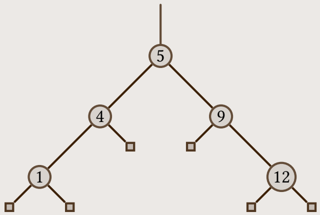
Exemple d'insertion d'un élément dans un ARB

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



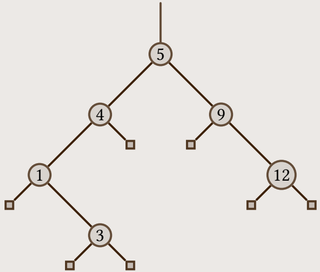
Exemple d'insertion d'un élément dans un ARB

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



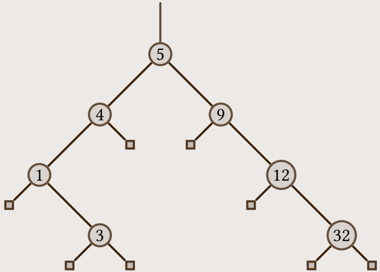
Exemple d'insertion d'un élément dans un ARB

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 :