

Notes de cours

Ecole Nationale d'Ingénieurs de Brest

Structures de données

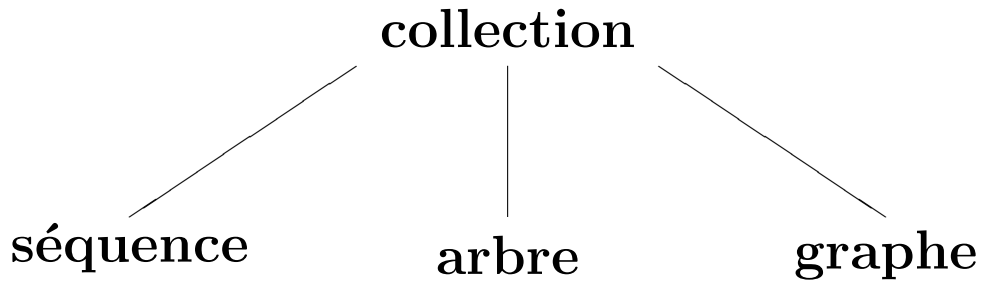
J. Tisseau , S. Morvan

– 1994,1999 –

Collections

- Séquences
- Arbres
- Graphes

Types de collections

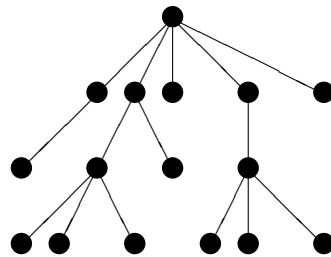
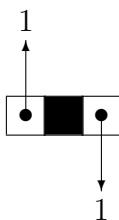


vecteur

chaîne

pile

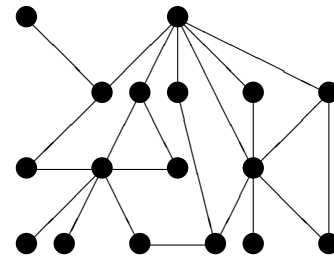
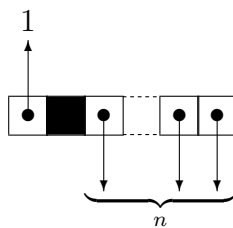
file



arbre de décision

système de fichiers

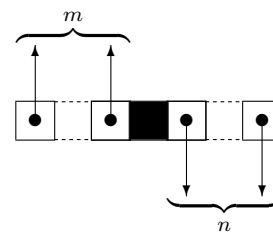
taxinomie



réseau routier

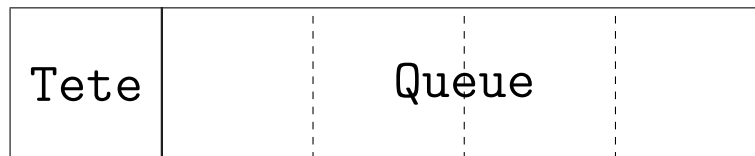
grafcet

réseau sémantique



Séquences

Une séquence est une suite ordonnée (éventuellement vide) d'éléments.



Notations :

1. $[]$: séquence vide
2. \mathcal{S} : ensemble des séquences
3. \mathcal{E} : ensemble des éléments
4. $x \in \mathcal{E}$, $s \in \mathcal{S}$, $x \circ s$: insertion de l'élément x en tête de la séquence s

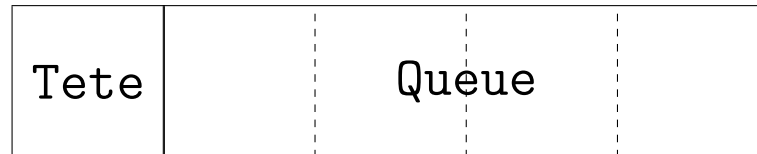
Définition :

1. $[] \in \mathcal{S}$
2. $\forall x \in \mathcal{E}$, $\forall s \in \mathcal{S}$, $x \circ s \in \mathcal{S}$

Axiomes :

1. $\forall x \in \mathcal{E}$, $\forall s \in \mathcal{S}$, $x \circ s \neq []$
2. $\forall x_1, x_2 \in \mathcal{E}$, $\forall s_1, s_2 \in \mathcal{S}$,
 $(x_1 \circ s_1 = x_2 \circ s_2) \Rightarrow (x_1 = x_2) \wedge (s_1 = s_2)$

Tête et queue



Tête :

$$\forall x \in \mathcal{E}, \forall s \in \mathcal{S}, \text{head}(x \circ s) = x$$

Queue :

$$\forall x \in \mathcal{E}, \forall s \in \mathcal{S}, \text{tail}(x \circ s) = s$$

Propriétés :

1. $\forall s \in \mathcal{S}, \text{head}(s) \in \mathcal{E}$ si $s \neq []$
2. $\forall s \in \mathcal{S}, \text{tail}(s) \in \mathcal{S}$ si $s \neq []$
3. $\forall s \in \mathcal{S}, s = \text{head}(s) \circ \text{tail}(s)$ si $s \neq []$

Sous-séquences

Préfixe :

1. $\forall s \in \mathcal{S}, [] = \text{prefixe}(s)$
2. $\forall x \in \mathcal{E}, \forall s, s_1, s_2 \in \mathcal{S},$
 $s_2 = x \circ s_1 = \text{prefixe}(x \circ s)$ ssi
 $s_1 = \text{prefixe}(s)$

Suffixe :

$$\forall s, s_1 \in \mathcal{S},$$
$$s_1 = \text{suffixe}(s) \text{ ssi}$$
$$s_1 = s \vee s_1 = \text{suffixe}(\text{tail}(s))$$

Sous-séquence :

$$\forall s, s_1 \in \mathcal{S},$$
$$s_1 \subset s \text{ ssi}$$
$$s_1 = \text{prefixe}(\text{suffixe}(s))$$

Appartenance et longueur

Appartenance :

1. $\forall x \in \mathcal{E}, \forall s \in \mathcal{S}, x = \text{head}(s) \in s$
2. $\forall x \in \mathcal{E}, \forall s \in \mathcal{S},$
 $x \in s$ ssi $x \in \text{tail}(s)$

Longueur :

1. $\text{length}([\]) = 0$
2. $\forall x \in \mathcal{E}, \forall s \in \mathcal{S},$
 $\text{length}(x \circ s) = \text{length}(s) + 1$

Rang :

1. $\forall x \in \mathcal{E}, \forall s \in \mathcal{S},$
 $(x \circ s)[0] = x$
2. $\forall x \in \mathcal{E}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}$
 $(x \circ s)[n] = s[n - 1]$ ssi $0 < n < \text{length}(x \circ s)$

Concaténation

Concaténation :

1. $\forall s \in \mathcal{S}, \square + s = s$
2. $\forall x \in \mathcal{E}, \forall s, s_1, s_2 \in \mathcal{S},$
 $s_3 = x \circ s_2 = (x \circ s_1) + s$ ssi
 $s_2 = s_1 + s$

Inversion :

1. $reverse(\square) = \square$
2. $\forall x \in \mathcal{E}, \forall s \in \mathcal{S},$
 $reverse(x \circ s) = reverse(s) + (x \circ \square)$

Itérateurs

Application d'une fonction :

foreach

$$\forall x \in \mathcal{E}, \forall s \in \mathcal{S},$$

$$\forall f : \mathcal{E} \rightarrow \mathcal{E},$$

1. $\text{foreach}(f, []) = []$

2. $\text{foreach}(f, x \circ s) = f(x) \circ \text{foreach}(f, s)$

Sélection sous condition :

collect

$$\forall x \in \mathcal{E}, \forall s \in \mathcal{S},$$

$$\forall p : \mathcal{E} \rightarrow \{\text{false}, \text{true}\},$$

1. $\text{collect}(p, []) = []$

2. $\text{collect}(p, x \circ s) = x \circ \text{collect}(p, s)$ si $p(x)$

3. $\text{collect}(p, x \circ s) = \text{collect}(p, s)$ si $\neg p(x)$

Séquences ordonnées

Relation d'ordre total : \leq

L'ensemble $\mathcal{E}' \subset \mathcal{E}$ est totalement ordonné s'il existe une relation d'ordre dans \mathcal{E}' , notée \leq , telle que :

$\forall x, y, z \in \mathcal{E}'$,

1. réflexivité : $x \leq x$
2. antisymétrie : $(x \leq y) \wedge (y \leq x) \Rightarrow x = y$
3. transitivité : $(x \leq y) \wedge (y \leq z) \Rightarrow (x \leq z)$

Séquence ordonnée : *ordered*

$\forall x, y \in \mathcal{E}', \forall s \in \mathcal{S}(\mathcal{E}')$,

1. $\text{ordered}(\square)$
2. $\text{ordered}(x \circ \square)$
3. $\text{ordered}(x \circ (y \circ s))$ ssi
 $(x \leq y) \wedge \text{ordered}(y \circ s)$

Tri d'une séquence

Tri :

sort

$\forall s, s' \in \mathcal{S}(\mathcal{E}'),$
 $s' = \text{sort}(s)$ ssi
 $\text{permutation}(s, s') \wedge \text{ordered}(s')$

Permutation :

permutation

$\forall x \in \mathcal{E}, \forall s, s', s'' \in \mathcal{S}$

1. $\text{permutation}([], [])$
2. $\text{permutation}(s, x \circ s'')$
 $\wedge \text{select}(x, s, s')$
 $\wedge \text{permutation}(s', s'')$

Sélection d'un élément :

select

$\forall x, y \in \mathcal{E}, \forall s, s' \in \mathcal{S}$

1. $\text{select}(x, x \circ s, s)$
2. $\text{select}(x, y \circ s, y \circ s') \wedge \text{select}(x, s, s')$

Tri par sélection

Tri : *selectsort*

$$\forall x, x_{min} \in \mathcal{E}', \forall s, s_{del} \in \mathcal{S}(\mathcal{E}'),$$

1. $\text{selectsort}(\square) = \square$
2. $\text{selectsort}(x \circ s) = x_{min} \circ \text{selectsort}(s_{del})$
 $\wedge x_{min} = \min(x \circ s)$
 $\wedge s_{del} = \text{del}(x_{min}, x \circ s)$

Minimum : *min*

$$\forall x, x_{min} \in \mathcal{E}', \forall s \in \mathcal{S}(\mathcal{E}'),$$

$$x_{min} = \min(s) \text{ ssi } \forall x \in s, x_{min} \leq x$$

Suppression d'un élément : *del*

$$\forall x, y \in \mathcal{E}, \forall s \in \mathcal{S}$$

1. $\text{del}(y, \square) = \square$
2. $\text{del}(y, x \circ s) = s$ si $x = y$
3. $\text{del}(y, x \circ s) = x \circ \text{del}(y, s)$ si $x \neq y$

Remarque : Les différents algorithmes de tri par sélection (tri par sélection/échange, tri bulles, ...) se distinguent par les calculs de $\min(s)$ et $\text{del}(x, s)$.

Tri par insertion

Tri : *insertsort*

$$\forall x \in \mathcal{E}', \forall s \in \mathcal{S}(\mathcal{E}'),$$

1. $\text{insertsort}(\[]) = []$
2. $\text{insertsort}(x \circ s) = \text{insert}(x, \text{insertsort}(s))$

Insertion dans une séquence triée : *insert*

$$\forall x, y \in \mathcal{E}', \forall s \in \mathcal{S}(\mathcal{E}'),$$

1. $\text{insert}(x, []) = x \circ []$
2. $\text{insert}(x, y \circ s) = x \circ (y \circ s)$ si $x \leq y$
3. $\text{insert}(x, y \circ s) = y \circ \text{insert}(x, s)$ si $x \not\leq y$

Remarque : Les différents algorithmes de tri par insertion (tri par insertion, tri par insertion dichotomique, ...) se distinguent par le calcul de $\text{insert}(x, s)$.

Tri rapide

Tri :

quicksort

$\forall x \in \mathcal{E}', \forall s, s_{inf}, s_{sup} \in \mathcal{S}(\mathcal{E}'),$

1. $quicksort([]) = []$

2. $quicksort(x \circ s) =$

$quicksort(s_{inf}) + (x \circ quicksort(s_{sup}))$

$\wedge s_{inf} = collect((x' \leq x \wedge x' \in s), s)$

$\wedge s_{sup} = collect((x' \not\leq x \wedge x' \in s), s)$

Arbres

Un arbre est un ensemble fini (éventuellement vide) de nœuds.

Un nœud est un doublet $\langle x, f \rangle$ composé d'un élément x et d'une forêt f .

Une forêt f est une séquence (éventuellement vide) d'arbres disjoints non vides ($f = [a_1, a_2, \dots, a_n]$).

Notations :

1. \mathcal{A} : ensemble des arbres
2. \mathcal{F} : ensemble des forêts
3. \mathcal{E} : ensemble des éléments
4. $\langle \rangle$: arbre vide
5. $x \in \mathcal{E}, f \in \mathcal{F}, \langle x, f \rangle$: nœud
6. $\langle x, [] \rangle \equiv \langle x \rangle$

Définition :

1. $\langle \rangle \in \mathcal{A}$
2. $\forall x \in \mathcal{E}, \langle x, [] \rangle \in \mathcal{A}$
3. $\forall x \in \mathcal{E}, \forall f \neq [] \in \mathcal{F}, \langle x, f \rangle \in \mathcal{A}$

Largeur et profondeur

Largeur :

width

$$\forall x \in \mathcal{E}, f \in \mathcal{F},$$

1. $\text{width}(\langle \rangle) = 0$

2. $\text{width}(\langle x, f \rangle) = \max(\text{length}(f) \circ \text{foreach}(\text{width}, f))$

Profondeur :

depth

$$\forall x \in \mathcal{E}, \forall a \in \mathcal{A}, \forall f \in \mathcal{F},$$

1. $\text{depth}(\langle \rangle) = 0$

2. $\text{depth}(\langle x, [] \rangle) = 1$

3. $\text{depth}(\langle x, a \circ f \rangle) = \max(\text{foreach}(\text{depth}, a \circ f)) + 1$

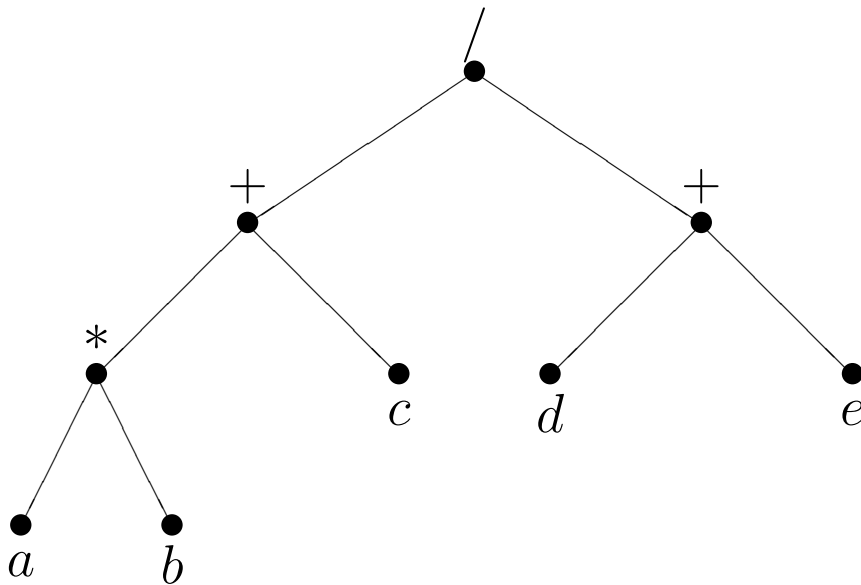
Parcours d'arbres

Parcours :

order

order : $\mathcal{A} \rightarrow \mathcal{S}(\mathcal{E})$

$\langle /, [\langle +, [\langle *, [\langle a \rangle, \langle b \rangle] \rangle, \langle c \rangle] \rangle, \langle +, [\langle d \rangle, \langle e \rangle] \rangle] \rangle$



Parcours : [...?....]

1. Parcours infixé : notation mathématique
2. Parcours préfixé : langage Lisp
3. Parcours postfixé : calculette HP

Parcours infixé

Parcours :

inorder

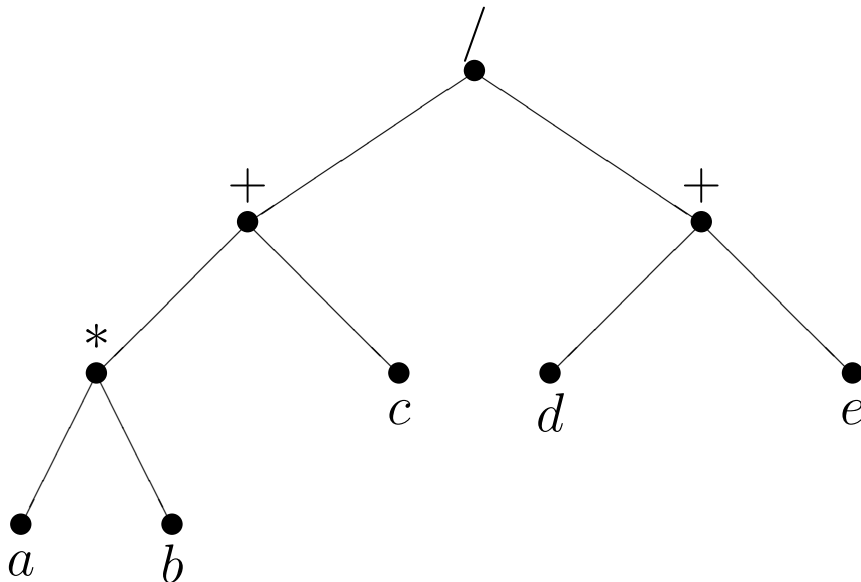
$$\forall x \in \mathcal{E}, \forall a \in \mathcal{A}, \forall f \in \mathcal{F},$$

$$1. \text{inorder}(\langle \rangle) = []$$

$$2. \text{inorder}(\langle x, [] \rangle) = x \circ []$$

$$3. \text{inorder}(\langle x, a \circ f \rangle) = \text{inorder}(a) + (x \circ []) + \text{foreach}(\text{inorder}, f)$$

$\langle /, [\langle +, [\langle *, [\langle a \rangle, \langle b \rangle] \rangle, \langle c \rangle] \rangle, \langle +, [\langle d \rangle, \langle e \rangle] \rangle] \rangle$



Parcours infixé : [a, *, b, +, c, /, d, +, e]

Parcours préfixé

Parcours :

preorder

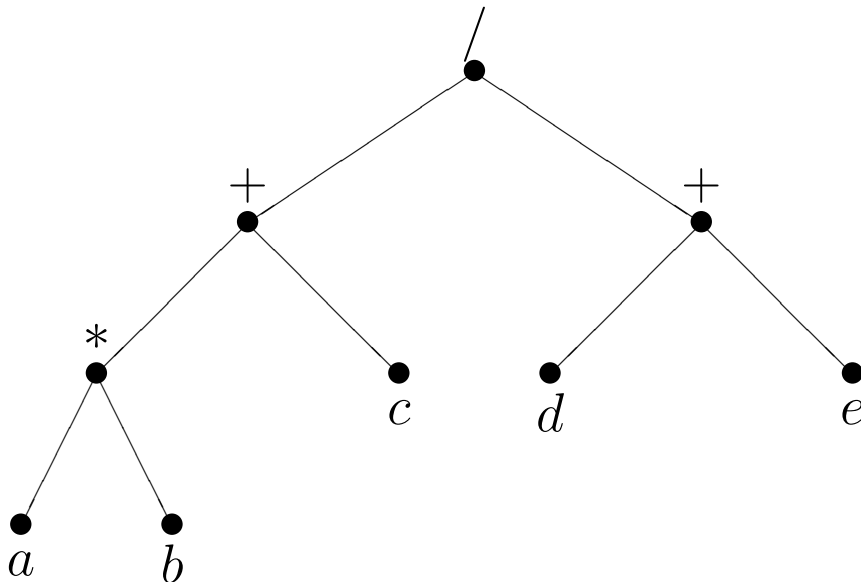
$$\forall x \in \mathcal{E}, \forall a \in \mathcal{A}, \forall f \in \mathcal{F},$$

$$1. \text{preorder}(\langle \rangle) = []$$

$$2. \text{preorder}(\langle x, [] \rangle) = x \circ []$$

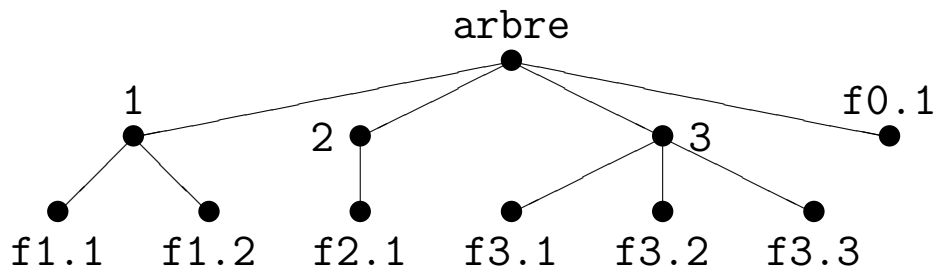
$$3. \text{preorder}(\langle x, a \circ f \rangle) = (x \circ []) + \text{foreach}(\text{preorder}, a \circ f)$$

$\langle /, [\langle +, [\langle *, [\langle a \rangle, \langle b \rangle] \rangle, \langle c \rangle] \rangle, \langle +, [\langle d \rangle, \langle e \rangle] \rangle] \rangle$



Parcours préfixé : $[/, +, *, a, b, c, +, d, e]$

Exemple de parcours préfixé



```
$ ls -R ./arbre
./arbre:
1/ 2/ 3/ f0.1

./arbre/1:
f1.1 f1.2

./arbre/2:
f2.1

./arbre/3:
f3.1 f3.2 f3.3
$
```

Parcours postfixé

Parcours :

postorder

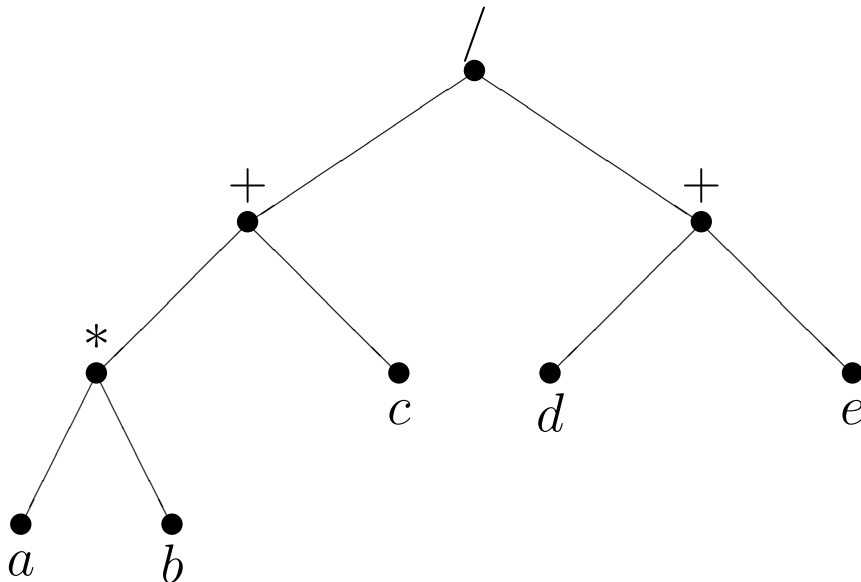
$\forall x \in \mathcal{E}, \forall a \in \mathcal{A}, \forall f \in \mathcal{F},$

1. $\text{postorder}(\langle \rangle) = []$

2. $\text{postorder}(\langle x, [] \rangle) = x \circ []$

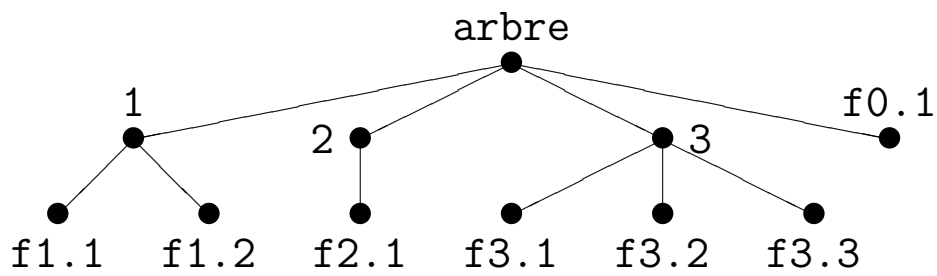
3. $\text{postorder}(\langle x, a \circ f \rangle) =$
 $\text{foreach}(\text{postorder}, a \circ f) + (x \circ [])$

$\langle /, [\langle +, [\langle *, [\langle a \rangle, \langle b \rangle] \rangle, \langle c \rangle] \rangle, \langle +, [\langle d \rangle, \langle e \rangle] \rangle] \rangle$



Parcours postfixé : [a, b, *, c, +, d, e, +, /]

Exemple de parcours postfixé



```
$ du -a arbre
0  arbre/1/f1.1
0  arbre/1/f1.2
8  arbre/1
0  arbre/2/f2.1
8  arbre/2
0  arbre/3/f3.1
0  arbre/3/f3.2
0  arbre/3/f3.3
8  arbre/3
0  arbre/f0.1
32 arbre
$
```

Arbres binaires

Un arbre binaire est un arbre de largeur égale à 2.

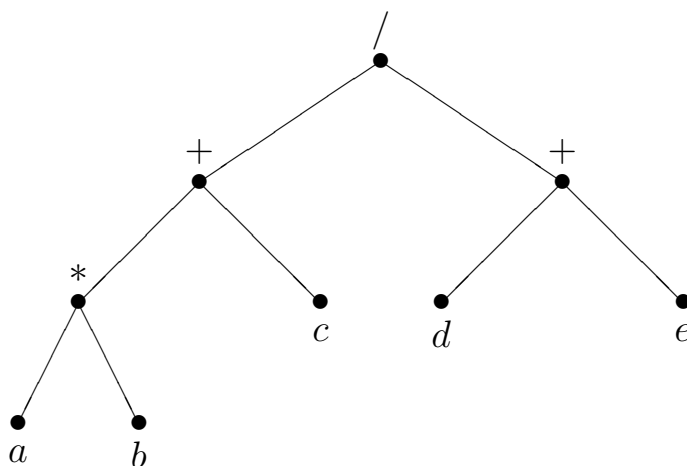
Notations particulières :

1. $\mathcal{A}_2 \subset \mathcal{A}$: ensemble des arbres binaires
2. $x \in \mathcal{E}$, $l, r \in \mathcal{A}_2$, $\langle x, [l, r] \rangle \equiv \langle l, x, r \rangle$
3. $\langle \langle \rangle, x, \langle \rangle \rangle \equiv \langle x \rangle$

Définition :

1. $\langle \rangle \in \mathcal{A}_2$
2. $\forall x \in \mathcal{E}, \forall l, r \in \mathcal{A}_2, \langle l, x, r \rangle \in \mathcal{A}_2$

$\langle \langle \langle a \rangle, *, \langle b \rangle \rangle, +, \langle c \rangle \rangle, /, \langle \langle d \rangle, +, \langle e \rangle \rangle \rangle$



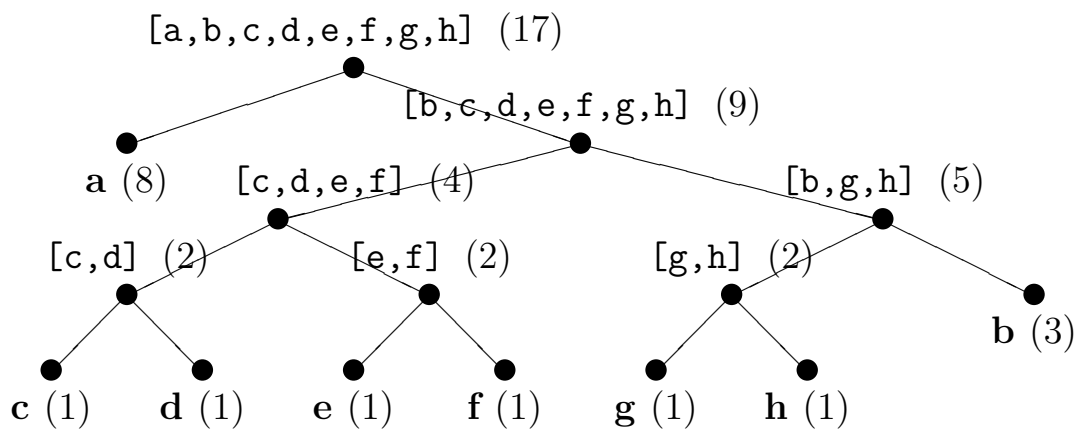
Arbre de Huffman

ASCII : [b, a, c] →

[0,1,1,0,0,0,1,0,0,1,1,0,0,0,0,1,0,1,1,0,0,0,1,1]

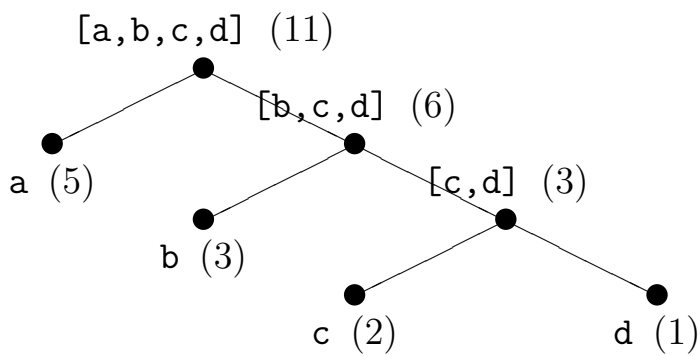
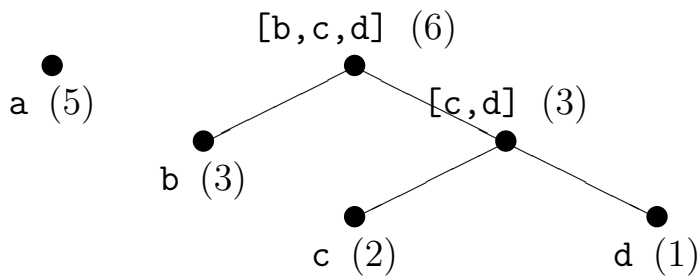
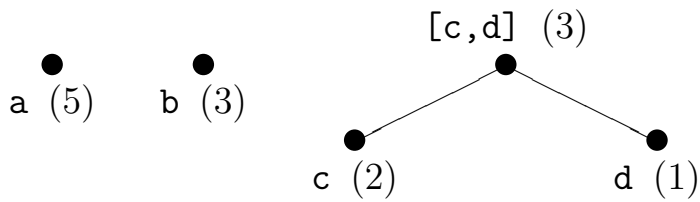
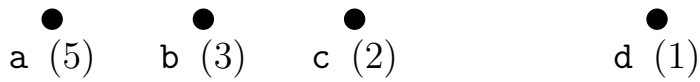
Morse : [b, a, c] → [[0,1,1,1], [1,0], [0,1,0,1]]

Huffman : [b, a, c] → [1,1,1,0,1,0,0,0]

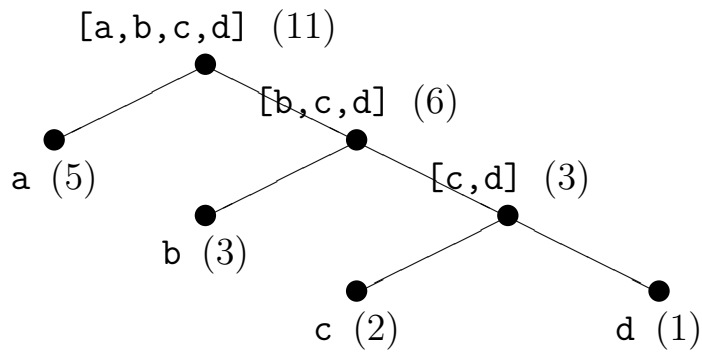


- déplacement vers la gauche : bit à 0
- déplacement vers la droite : bit à 1

Principe de construction



Codage/Décodage



a ↔ [0]

b ↔ [1,0]

c ↔ [1,1,0]

d ↔ [1,1,1]

[b,a] ↔ [1,0,0]

[a,b] ↔ [0,1,0]

[b,a,c] ↔ [1,0,0,1,1,0]

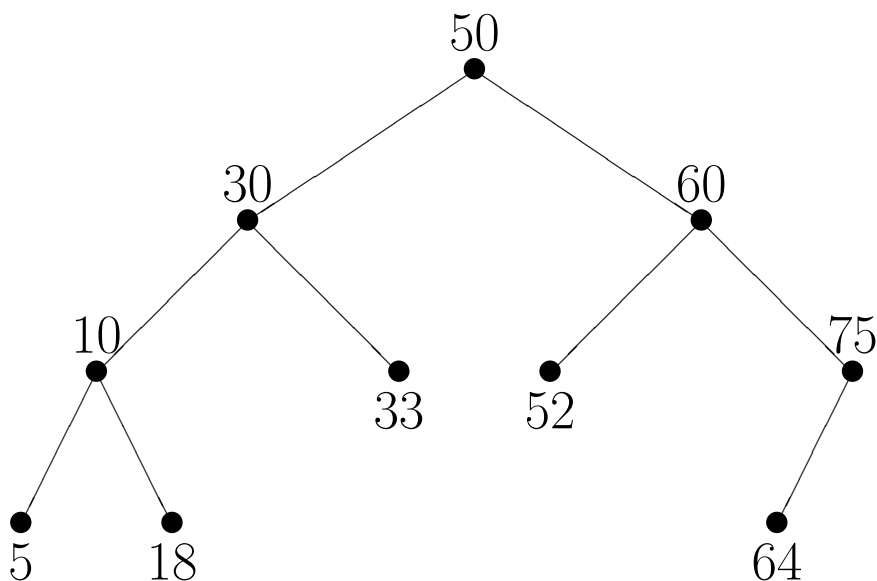
[b,a,b,a] ↔ [1,0,0,1,0,0]

[a,b,c,d] ↔ [0,1,0,1,1,0,1,1,1]

Arbres binaires de recherche

Un arbre binaire de recherche est un arbre binaire tel que pour tout nœud $n = \langle l, x, r \rangle$ de l'arbre

1. les éléments de tous les nœuds du sous-arbre gauche l sont inférieurs à l'élément x contenu dans n ;
2. les éléments de tous les nœuds du sous-arbre droit r sont supérieurs à l'élément x contenu dans n .

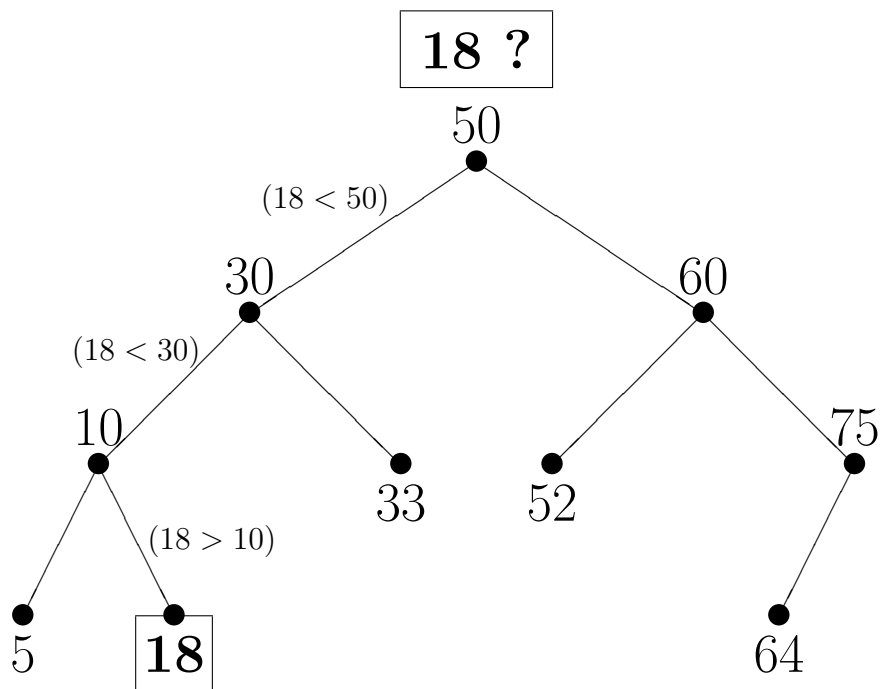


Recherche d'un élément

Recherche $\text{search} : \mathcal{E}' \times \mathcal{A}_2(\mathcal{E}') \rightarrow \text{Booléen}$

$\forall x, y \in \mathcal{E}', \forall l, r \in \mathcal{A}_2(\mathcal{E}')$,

1. $\text{search}(y, \langle \rangle) = \text{false}$
2. $\text{search}(y, \langle l, x, r \rangle) = \text{true}$ si $y = x$
3. $\text{search}(y, \langle l, x, r \rangle) = \text{search}(y, l)$ si $y < x$
4. $\text{search}(y, \langle l, x, r \rangle) = \text{search}(y, r)$ si $y > x$



Insertion d'un élément

Insertion $\text{insert} : \mathcal{E}' \times \mathcal{A}_2(\mathcal{E}') \rightarrow \mathcal{A}_2(\mathcal{E}')$

$\forall x, y \in \mathcal{E}', \forall l, r \in \mathcal{A}_2(\mathcal{E}')$,

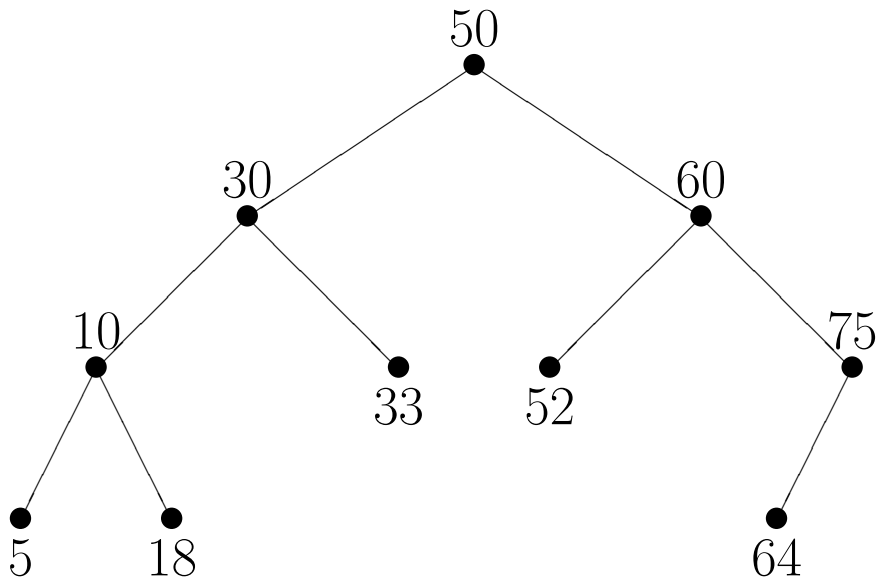
1. $\text{insert}(y, \langle \rangle) = \langle \langle \rangle, y, \langle \rangle \rangle$
2. $\text{insert}(y, \langle l, x, r \rangle) = \langle \text{insert}(y, l), x, r \rangle$
si $y < x$
3. $\text{insert}(y, \langle l, x, r \rangle) = \langle l, x, \text{insert}(y, r) \rangle$
si $y > x$

Le plus grand élément : $\text{max} : \mathcal{A}_2(\mathcal{E}') \rightarrow \mathcal{E}'$

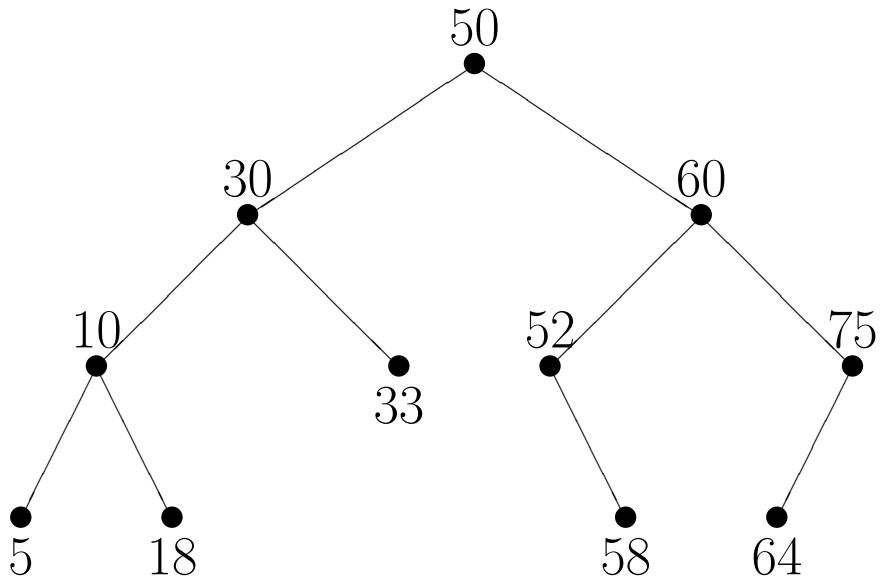
$\forall x \in \mathcal{E}', \forall l, r \in \mathcal{A}_2(\mathcal{E}')$,

1. $\text{max}(\langle l, x, \langle \rangle \rangle) = x$
2. $\text{max}(\langle l, x, r \rangle) = \text{max}(r)$ si $r \neq \langle \rangle$

Exemple d'insertion



`insert(58, <<<<5>, 10, <18>>, 30, <33>>, 50, <<52>, 60, <<64>, 75, <>>>>)`



Suppression d'un élément

Suppression $\text{del} : \mathcal{E}' \times \mathcal{A}_2(\mathcal{E}') \rightarrow \mathcal{A}_2(\mathcal{E}')$

$\forall x, y \in \mathcal{E}', \forall l, r \in \mathcal{A}_2(\mathcal{E}')$,

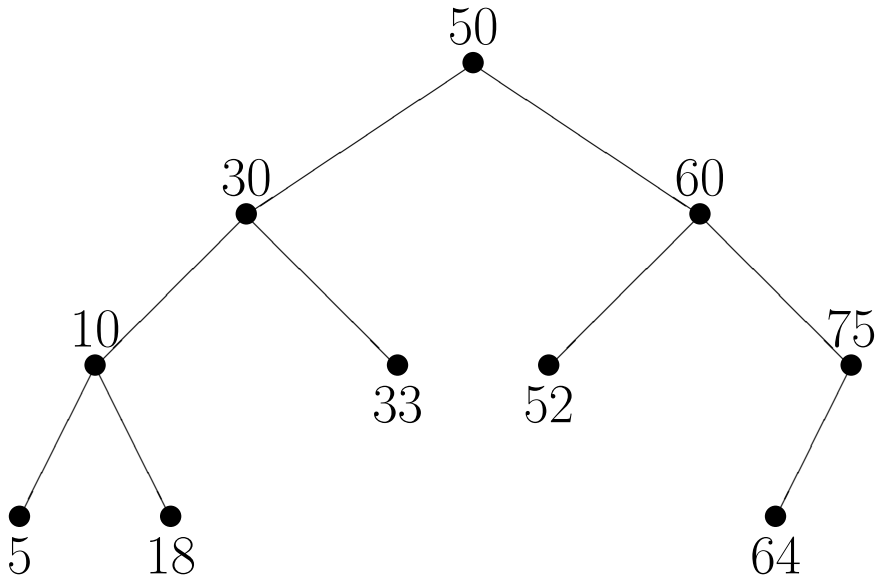
1. $\text{del}(y, \langle l, x, \langle \rangle \rangle) = l$ si $y = x$
2. $\text{del}(y, \langle \langle \rangle, x, r \rangle) = r$ si $y = x \wedge r \neq \langle \rangle$
3. $\text{del}(y, \langle l, x, r \rangle) = \langle \text{delmax}(l), \max(l), r \rangle$
si $y = x \wedge l \neq \langle \rangle \wedge r \neq \langle \rangle$
4. $\text{del}(y, \langle l, x, r \rangle) = \langle \text{del}(y, l), x, r \rangle$
si $y < x$
5. $\text{del}(y, \langle l, x, r \rangle) = \langle l, x, \text{del}(y, r) \rangle$
si $y > x$

$\text{delmax} : \mathcal{A}_2(\mathcal{E}') \rightarrow \mathcal{A}_2(\mathcal{E}')$

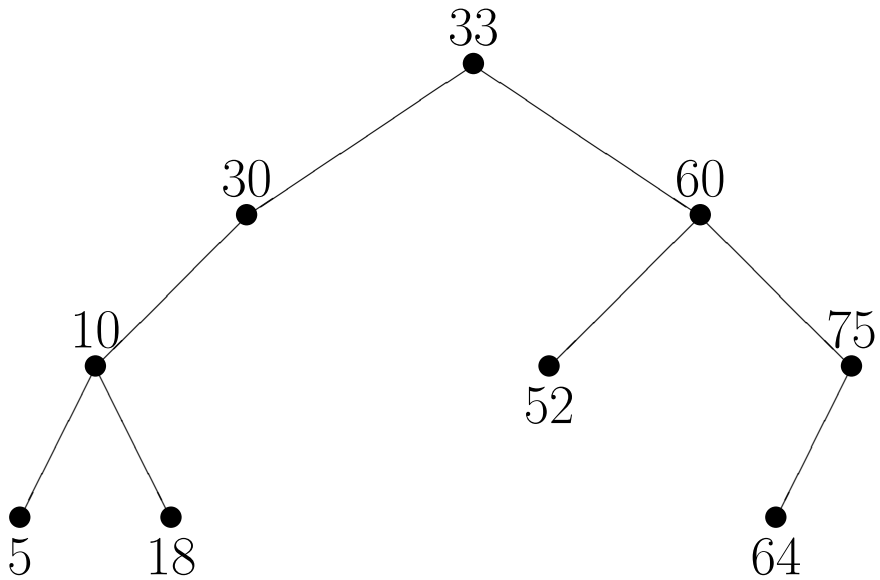
$\forall x \in \mathcal{E}', \forall l, r \in \mathcal{A}_2(\mathcal{E}')$,

1. $\text{delmax}(\langle l, x, \langle \rangle \rangle) = l$
2. $\text{delmax}(\langle l, x, r \rangle) = \langle l, x, \text{delmax}(r) \rangle$ si $r \neq \langle \rangle$

Exemple de suppression



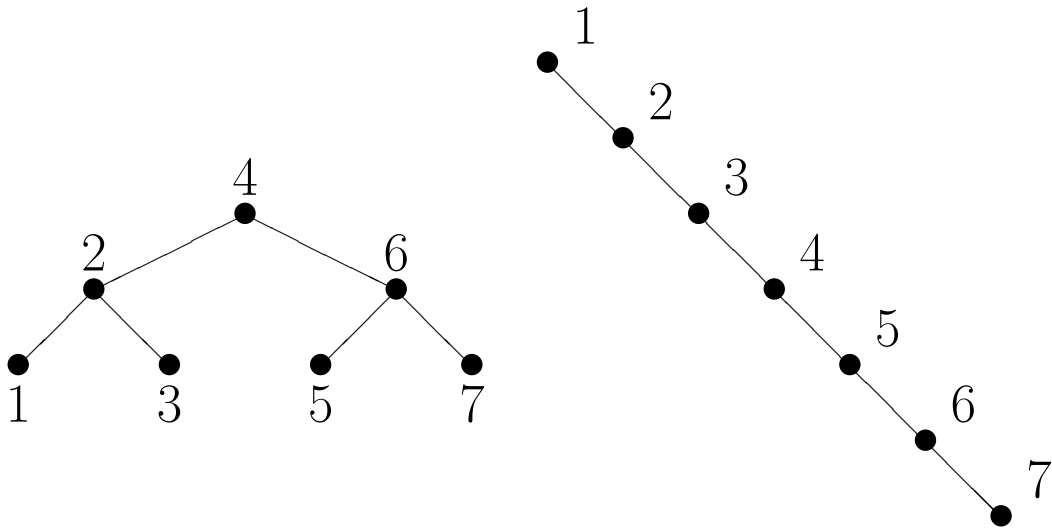
del(50, <<<<5>, 10, <18>>, 30, <33>>, 50, <<52>, 60, <<64>, 75, <>>>>)



Arbres de recherche équilibrés

Problème

1. Complexité en moyenne : $\approx \log(n)$
2. Complexité au pire : $\approx n$

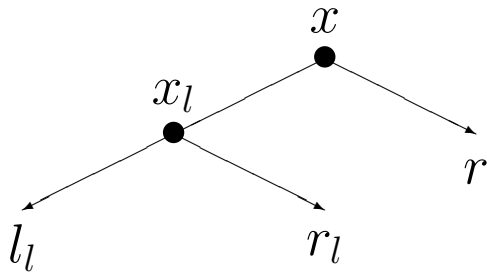


Solutions envisagées

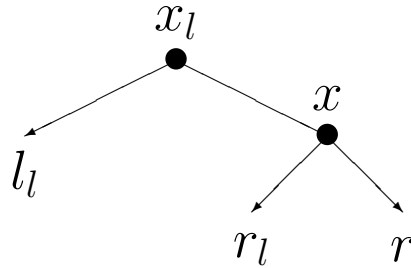
1. Autoriser un léger déséquilibre entre les profondeurs des sous-arbres gauche et droit
→ *arbres AVL*
2. Autoriser les nœuds à avoir un nombre variable de fils
→ *arbres 2-3*

Rééquilibrage à droite

Avant



Après



Rééquilibrage

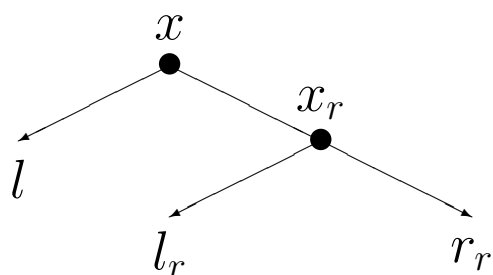
$\text{rreq} : \mathcal{A}(\mathcal{E}) \rightarrow \mathcal{A}(\mathcal{E})$

$\forall x, x_l \in \mathcal{E}, \forall l, r, l_l, r_l \in \mathcal{A}(\mathcal{E}),$

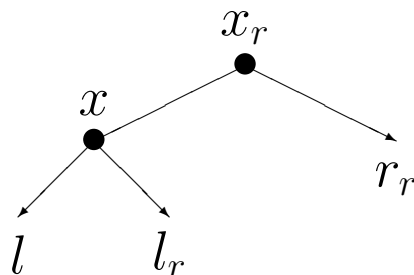
$$1. \text{rreq}(\langle \langle l_l, x_l, r_l \rangle, x, r \rangle) = \langle l_l, x_l, \langle r_l, x, r \rangle \rangle$$

Rééquilibrage à gauche

Avant



Après



Rééquilibrage

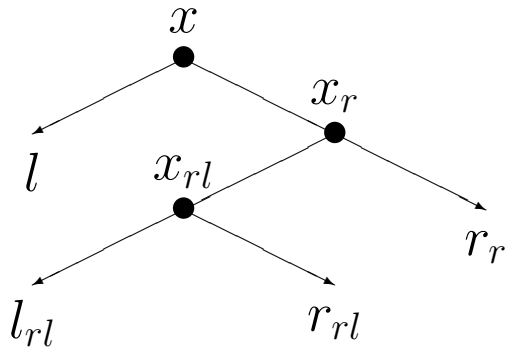
$\text{lreq} : \mathcal{A}(\mathcal{E}) \rightarrow \mathcal{A}(\mathcal{E})$

$\forall x, x_r \in \mathcal{E}, \forall l, r, l_r, r_r \in \mathcal{A}(\mathcal{E}),$

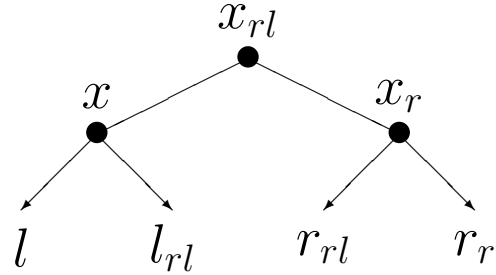
$$1. \text{lreq}(\langle l, x, \langle l_r, x_r, r_r \rangle \rangle) = \langle \langle l, x, l_r \rangle, x_r, r_r \rangle$$

Rééquilibrage droite/gauche

Avant



Après



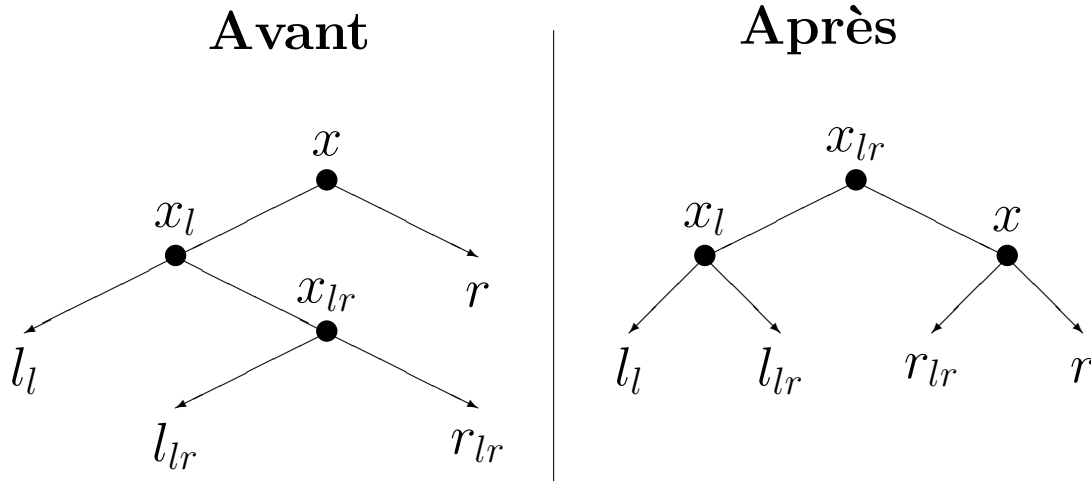
Rééquilibrage

$\text{rlreq} : \mathcal{A}(\mathcal{E}) \rightarrow \mathcal{A}(\mathcal{E})$

$\forall x, x_r, x_{rl}, \forall l, l_{rl}, r_{rl}, r_r \in \mathcal{A}(\mathcal{E}),$

$$1. \text{rlreq}(\langle l, x, \langle \langle l_{rl}, x_{rl}, r_{rl} \rangle, x_r, r_r \rangle \rangle) = \langle \langle l, x, l_{rl} \rangle, x_{rl}, \langle r_{rl}, x_r, r_r \rangle \rangle$$

Rééquilibrage gauche/droite



Rééquilibrage

$\text{lrreq} : \mathcal{A}(\mathcal{E}) \rightarrow \mathcal{A}(\mathcal{E})$

$\forall x, x_l, x_{lr}, \forall r, r_{lr}, l_{lr}, l_l \in \mathcal{A}(\mathcal{E}),$

$$1. \text{lrreq}(\langle \langle l_l, x_l, \langle l_{lr}, x_{lr}, r_{lr} \rangle \rangle, x, r \rangle) = \langle \langle l_l, x_l, l_{lr} \rangle, x_{lr}, \langle r_{lr}, x, r \rangle \rangle$$

Arbres AVL

Un arbre binaire est équilibré si en tout nœud de l'arbre, les profondeurs des sous-arbres gauche et droit diffèrent au plus de 1.

Définition

$avl : \mathcal{A}_2(\mathcal{E}) \rightarrow \text{Booléen}$

$\forall x \in \mathcal{E}, \forall l, r \in \mathcal{A}_2(\mathcal{E}),$

1. $avl(\langle \rangle) = \text{true}$

2. $avl(\langle l, x, r \rangle) =$

$$|\Delta h(\langle l, x, r \rangle)| \leq 1 \wedge avl(l) \wedge avl(r)$$

avec $\Delta h(\langle l, x, r \rangle) = \text{depth}(l) - \text{depth}(r)$

Propriété

Tout arbre AVL à n nœuds a une profondeur h vérifiant

$$\log_2(n + 1) \leq h + 1 < 1.44 \log_2(n + 2)$$

Remarque : Historiquement, ces arbres ont été introduits dans les années 1960 par Adelson-Velskii et Landis (d'où le nom d'AVL).

Insertion dans un AVL

Insertion $\text{insert} : \mathcal{E}' \times \mathcal{A}_2(\mathcal{E}') \rightarrow \mathcal{A}_2(\mathcal{E}')$

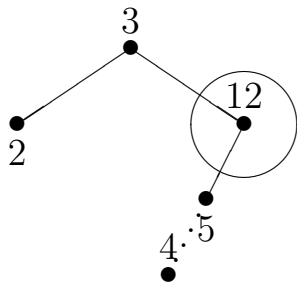
$\forall x, y \in \mathcal{E}' \forall l, r \in \mathcal{A}_2(\mathcal{E}')$,

1. $\text{insert}(y, \langle \rangle) = \langle \langle \rangle, y, \langle \rangle \rangle$
2. $\text{insert}(y, \langle l, x, r \rangle) =$
 $\text{req}(\langle \text{insert}(y, l), x, r \rangle)$ si $y < x$
3. $\text{insert}(y, \langle l, x, r \rangle) =$
 $\text{req}(\langle l, x, \text{insert}(y, r) \rangle)$ si $y > x$

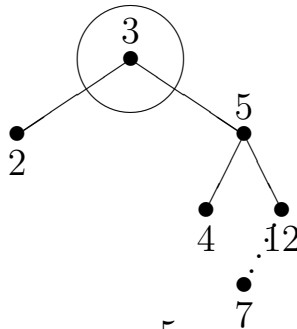
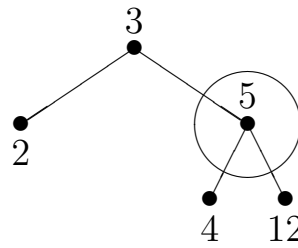
Rééquilibrage $\text{req} : \mathcal{A}_2(\mathcal{E}') \rightarrow \mathcal{A}_2(\mathcal{E}')$

1. $\text{req}(\langle l, x, r \rangle) = \langle l, x, r \rangle$
 si $|\Delta h(\langle l, x, r \rangle)| \leq 1$
2. $\text{req}(\langle l, x, r \rangle) = \text{rreq}(\langle l, x, r \rangle)$
 si $\Delta h(\langle l, x, r \rangle) = 2 \wedge \Delta h(l) = 1$
3. $\text{req}(\langle l, x, r \rangle) = \text{lreq}(\langle l, x, r \rangle)$
 si $\Delta h(\langle l, x, r \rangle) = -2 \wedge \Delta h(r) = -1$
4. $\text{req}(\langle l, x, r \rangle) = \text{lrreq}(\langle l, x, r \rangle)$
 si $\Delta h(\langle l, x, r \rangle) = 2 \wedge \Delta h(l) = -1$
5. $\text{req}(\langle l, x, r \rangle) = \text{rlreq}(\langle l, x, r \rangle)$
 si $\Delta h(\langle l, x, r \rangle) = -2 \wedge \Delta h(r) = 1$

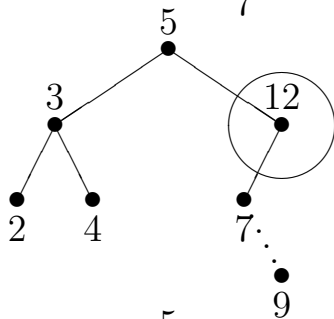
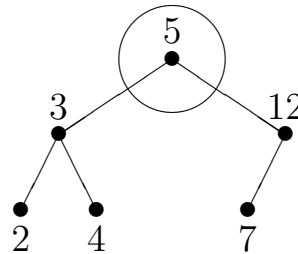
Exemples d'insertions dans un AVL



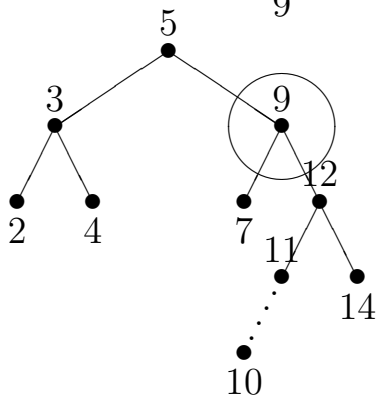
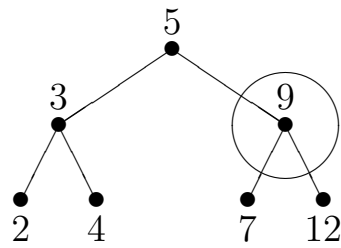
rreq



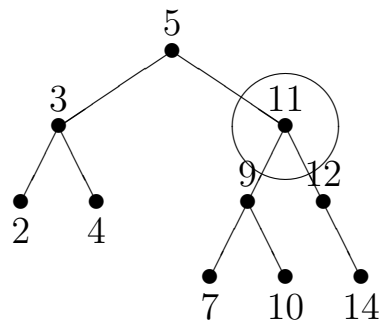
lreq



lrreq



rlreq



Suppression dans un AVL

Suppression $\text{del} : \mathcal{E}' \times \mathcal{A}_2(\mathcal{E}') \rightarrow \mathcal{A}_2(\mathcal{E}')$

$\forall x, y \in \mathcal{E}', \forall l, r \in \mathcal{A}_2(\mathcal{E}')$,

1. $\text{del}(y, \langle l, x, \langle \rangle \rangle) = l$ si $y = x$
2. $\text{del}(y, \langle \langle \rangle, x, r \rangle) = r$
si $y = x \wedge r \neq \langle \rangle$
3. $\text{del}(y, \langle l, x, r \rangle) =$
 $\text{req}(\langle \text{delmax}(l), \text{max}(l), r \rangle)$
si $y = x \wedge r \neq \langle \rangle \wedge l \neq \langle \rangle$
4. $\text{del}(y, \langle l, x, r \rangle) =$
 $\text{req}(\langle \text{del}(y, l), x, r \rangle)$ si $y < x$
5. $\text{del}(y, \langle l, x, r \rangle) =$
 $\text{req}(\langle l, x, \text{del}(y, r) \rangle)$ si $y > x$

$\text{delmax} : \mathcal{A}_2(\mathcal{E}') \rightarrow \mathcal{A}_2(\mathcal{E}')$

$\forall x \in \mathcal{E}', \forall l, r \in \mathcal{A}_2(\mathcal{E}')$,

1. $\text{delmax}(\langle l, x, \langle \rangle \rangle) = l$
2. $\text{delmax}(\langle l, x, r \rangle) =$
 $\text{req}(\langle l, x, \text{delmax}(r) \rangle)$ si $r \neq \langle \rangle$

Exemple de suppression dans un AVL

