ASD 2

Algorithmique et Structure de Données

Marc Gaëtano

gaetano@polytech.unice.fr

Polytech'Nice-Sophia

Département Sciences Informatiques

930 route des Colles

06903 Sophia Antipolis - France

Les arbres de recherche

- Dictionnaires
- Arbres binaires de recherche
- Arbres AVL

Dictionnaire : objets à clef

Un objet composite dont un des attributs est la clef

- `ObjetAClef(Object info, int clef)` fabrique un objet à clef contenant l'information `info` et de clef `k`
- `Object info()` : retourne l'information contenue dans un objet à clef
- `int clef()` : retourne la clef d'un objet à clef

Deux objets à clef distincts peuvent avoir la même clef

Dictionnaire : définition

Une structure de données contenant des objets à clef qui supporte les opérations

- `void ajouter(ObjetAClef x)` : ajoute l'objet à clef `x` dans le dictionnaire
- `ObjetAClef rechercher(int k)` : retourne l'objet à clef de clef `k` du dictionaire
- `void supprimer(int k)` : supprime l'objet à clef de clef `k` du dictionnaire

Dictionnaire : définition

Cas particuliers

- `void ajouter(ObjetAClef x)` : si un objet de même clef que `x` est déjà présent dans le dictionnaire, ne fait rien
- `ObjetAClef rechercher(int k)` : si aucun objet de clef `k` n'est dans le dictionnaire, retourne `null`
- `void supprimer(int k)` : si aucun objet de clef `k` n'est dans le dictionnaire, ne fait rien

Dictionnaire : implémentation

Plusieurs implémentations possibles

- tableau trié
- arbre binaire de recherche
- arbre équilibré :
  - arbre AVL
  - arbre 2-3-4 (ou arbre rouge et noir)
- table de hachage

Tableau trié : présentation

Caractéristiques

- mise en œuvre très simple mais taille statique
- recherche dichotomique
- idéal si le nombre d'ajouts et de suppressions est faible

Performances

opération	meilleur des cas	pire des cas	cas moyen
`ajouter`	$\Theta(\lg n)$	$\Theta(n)$	$\Theta(n)$
`supprimer`	$\Theta(\lg n)$	$\Theta(n)$	$\Theta(n)$
`rechercher`	$\Theta(1)$	$\Theta(\lg n)$	$\Theta(\lg n)$

ABR : définition

A est un Arbre Binaire de Recherche (ABR)

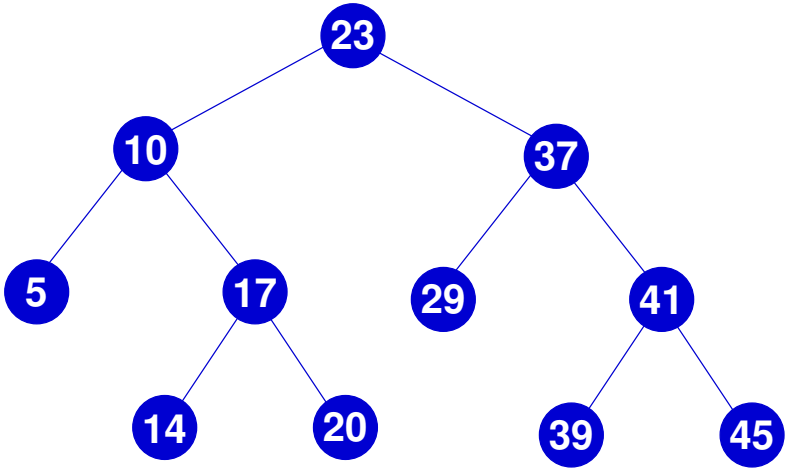
- A contient des objets à clefs
- A vérifie la propriété $\mathcal{P}$:

$$\mathcal{P}(A) \Longleftrightarrow \begin{cases} A = \emptyset, \text{ ou bien} \\ \mathcal{P}(g(A)), \mathcal{P}(d(A)), \text{ et } \forall g \in g(A)), d \in d(A) \\ \text{clef}(g) < \text{clef}(r(A)) < \text{clef}(d) \end{cases}$$

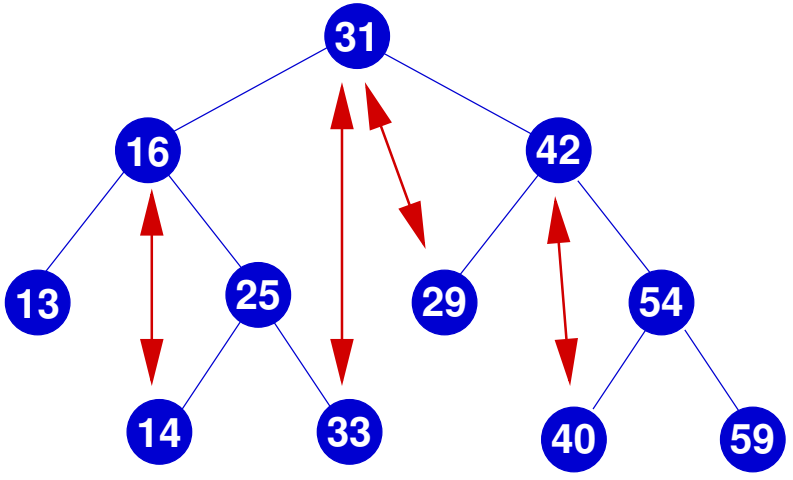
où $g(A)$, $d(A)$ et $r(A)$ sont resp. le sous-arbre gauche, le sous-arbre droit et la racine de l'arbre A

ABR : exemples

Propriété globale et non locale



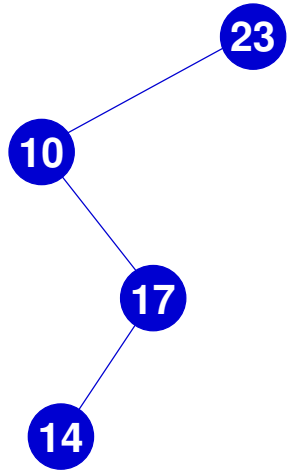
ABR



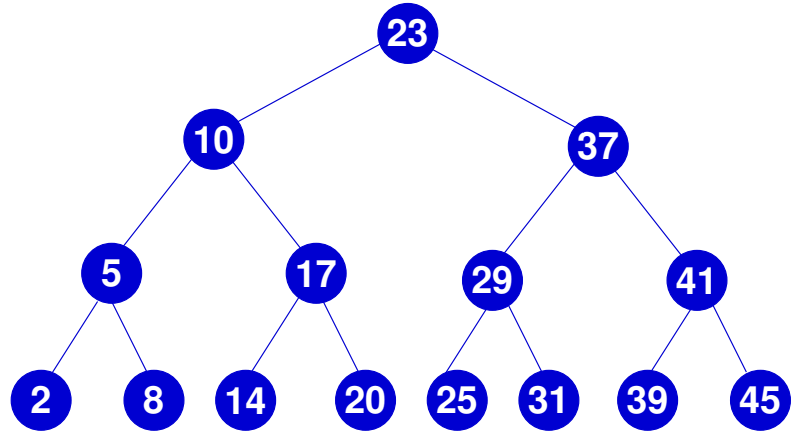
non ABR !

ABR : exemples

Cas particuliers



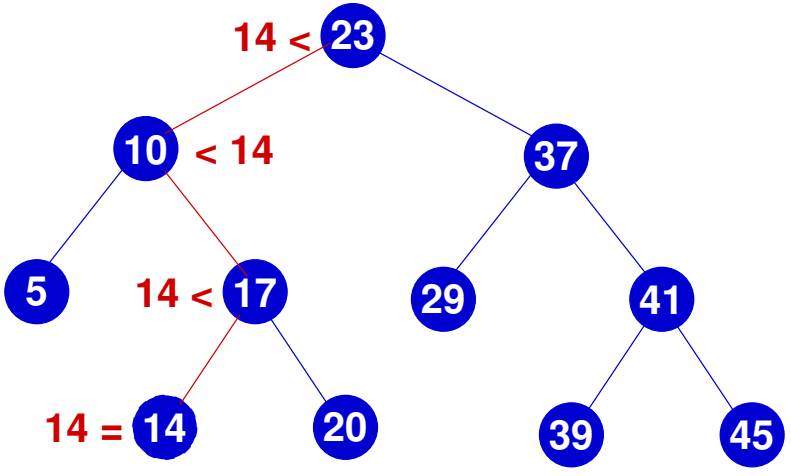
ABR filaire (dégénéré)



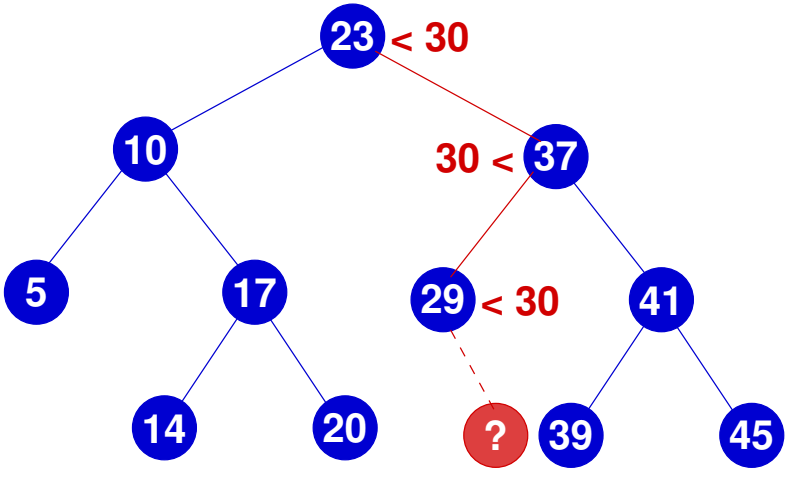
ABR complet

ABR : recherche

Exemple : recherche des objets de clef 14 **et** 30



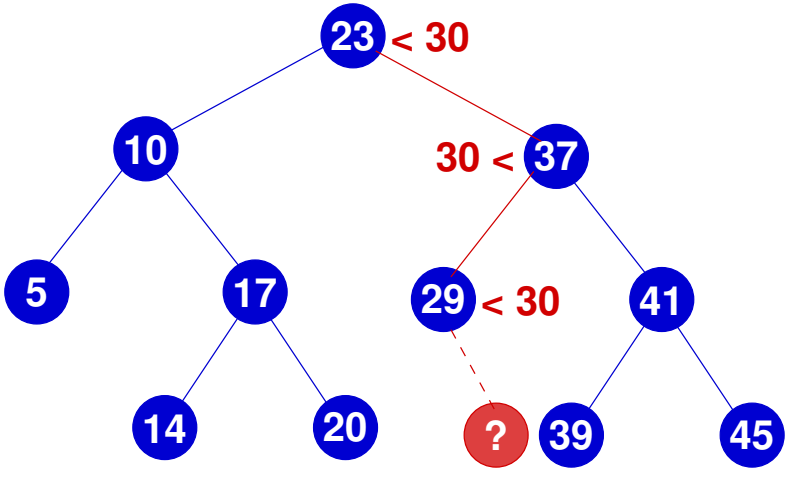
Recherche positive



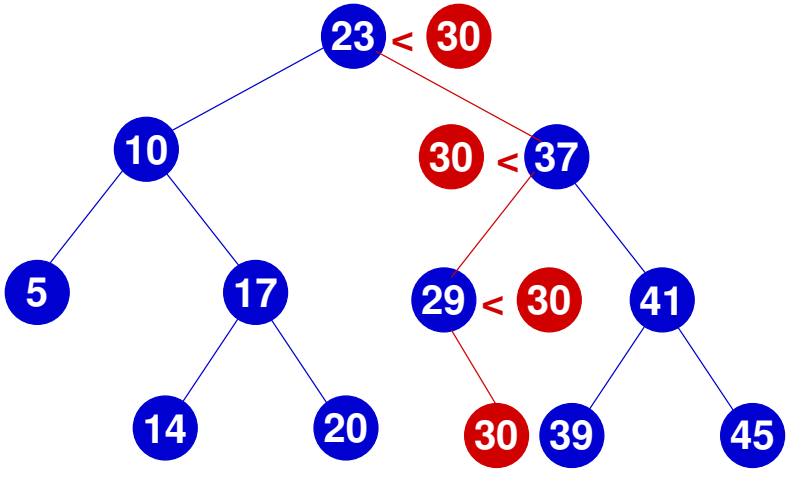
Recherche négative

ABR : ajout

Similaire à une recherche négative



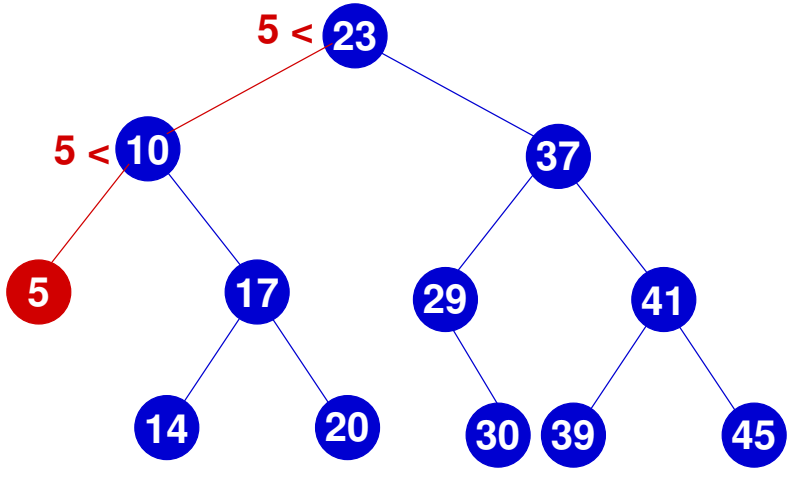
Recherche négative



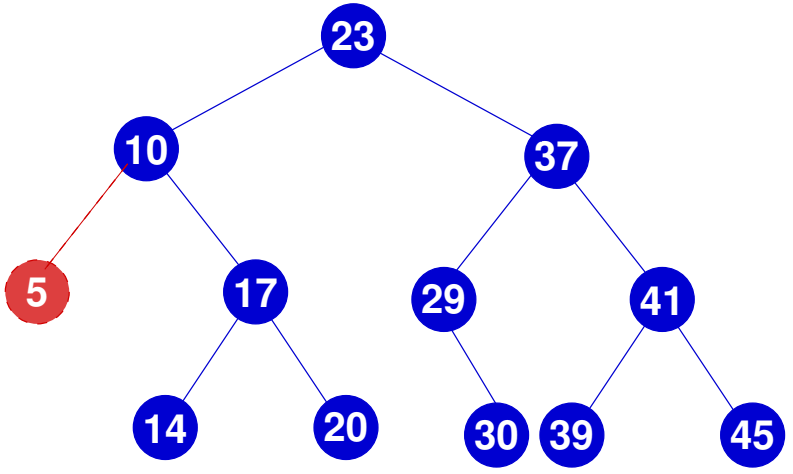
Ajout d'une feuille

ABR : suppression

Suppression d'une feuille (nœud de degré 0)



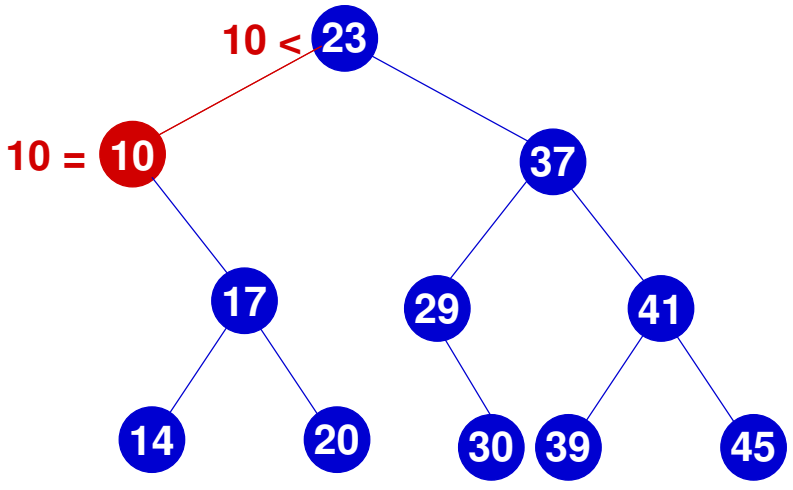
Recherche de la feuille de racine l'objet à supprimer



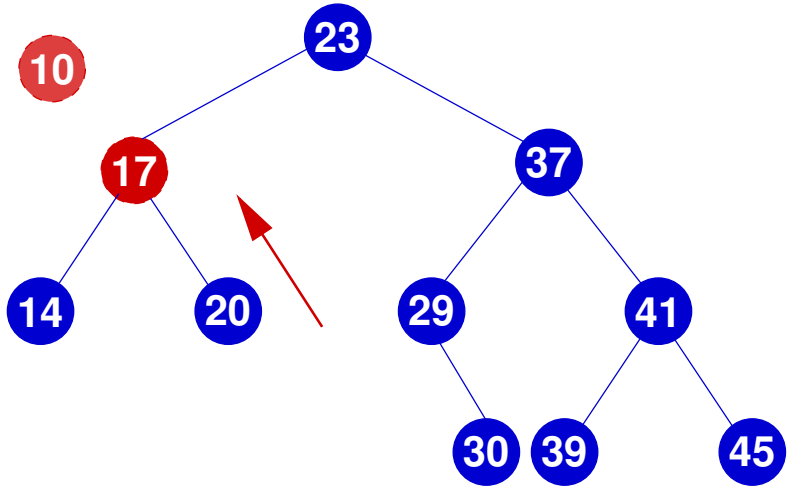
Suppression de la feuille

ABR : suppression

Suppression d'un nœud interne (nœud de degré 1)



Recherche du nœud de racine l'objet à supprimer



Remplacement du nœud par son unique sous-arbre

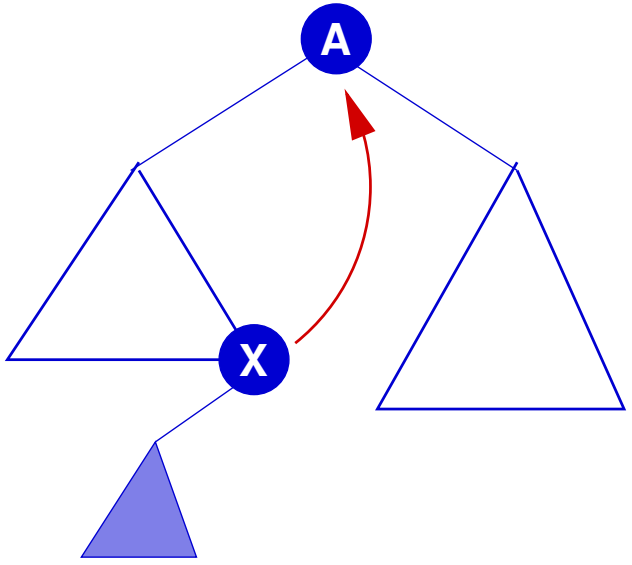
ABR : suppression

Suppression d'un nœud de degré 2

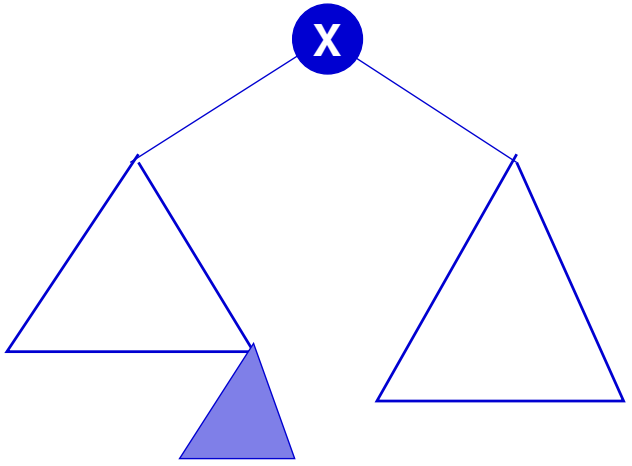
- revient à supprimer la racine d'un sous-arbre A
- on remplace la racine de A par (au choix) :
  - le *maximum* du sous-arbre gauche de A
  - le *minimum* du sous-arbre droit de A
- on supprime ce *maximum* dans le sous-arbre gauche de A ou ce *minimum* dans le sous-arbre droit de A :
  - c'est la suppression d'un nœud de degré *au plus* 1

ABR : suppression

Suppression de la racine d'un arbre de degré 2



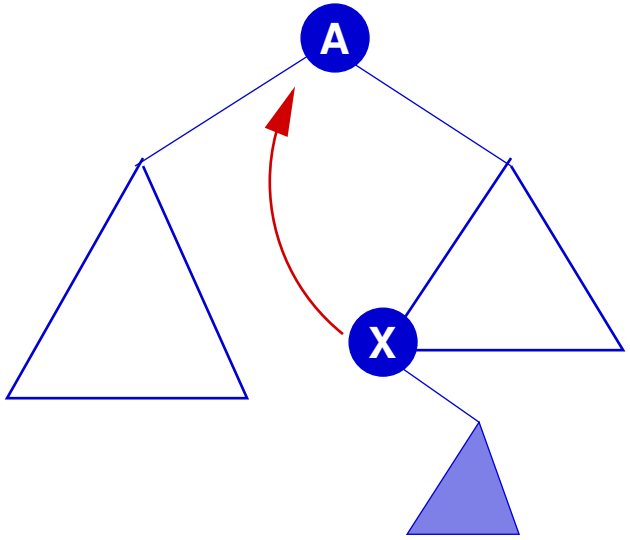
Remplacement de la racine par le *maximum* du sous-arbre gauche



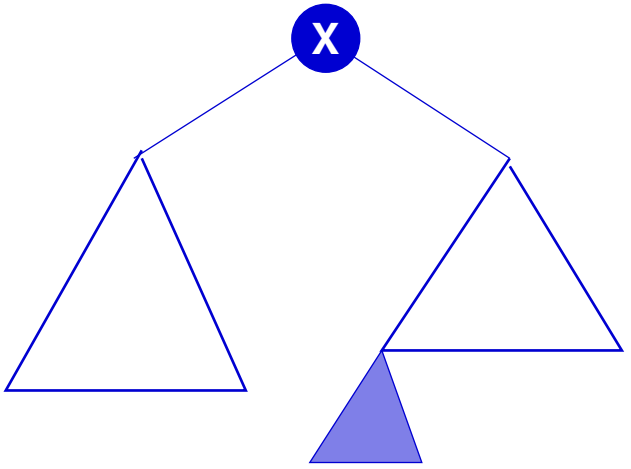
Suppression du *maximum* du sous-arbre gauche

ABR : suppression

Suppression de la racine d'un arbre de degré 2



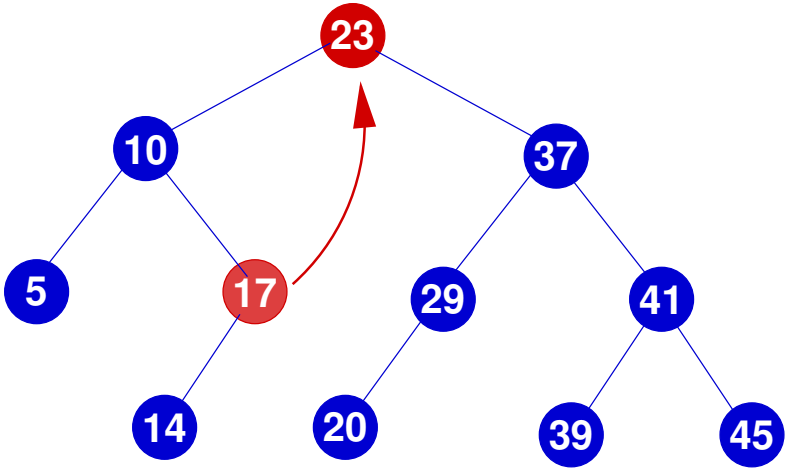
Remplacement de la racine par le *minimum* du sous-arbre droit



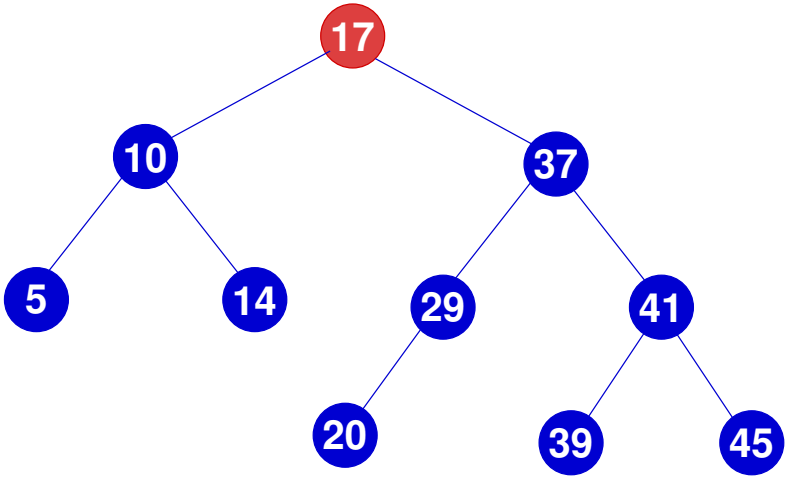
Suppression du *minimum* du sous-arbre droit

ABR : suppression

Exemple



Remplacement de la racine par le *maximum* du sous-arbre gauche

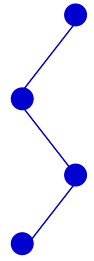


Suppression du *maximum* du sous-arbre gauche

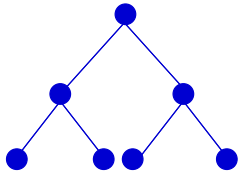
ABR : complexité

Complexité proportionnelle à la hauteur h de l'ABR de taille n

- cas dégénéré : $h = n - 1$



- cas idéal : $h = \lg(n + 1) - 1$



- cas moyen :
  calculer la hauteur moyenne d'un ABR

ABR : complexité

Définitions

$l(A)$, la longueur de cheminement externe d'un arbre binaire A (f désigne une feuille) :

$$l(A) = \sum_{f \in A} p(f)$$

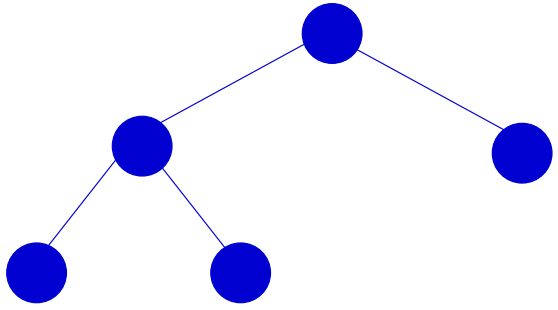
$\overline{h}(A)$, la hauteur moyenne d'un arbre binaire A :

$$\overline{h}(A) = \frac{1}{\phi} l(A)$$

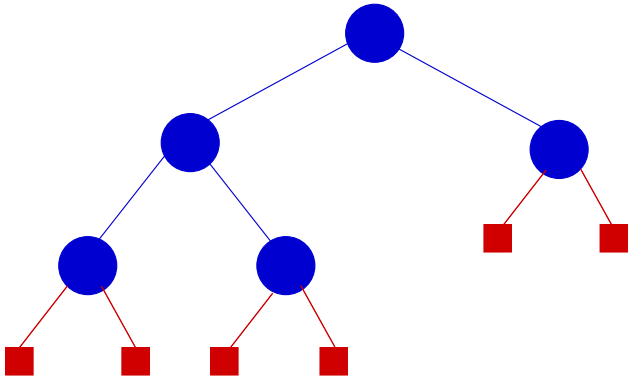
avec p la profondeur et ϕ le nombre de feuilles dans A

ABR : complexité

Arbre binaire complété



A un arbre binaire



A_C son complété

$|A| = n \Longrightarrow A_C$ posséde $n + 1$ nœuds vides

ABR : complexité

Complexité en moyenne

Soit A un ABR et A_C son complété. $\overline{h}(A_C)$, la hauteur moyenne de A_C correspond à la complexité en moyenne :

- d'un ajout dans A
- d'une recherche négative dans A
- d'une suppression d'un élément inexistant dans A

$\Longrightarrow$ c'est une **borne** de la complexité de ces méthodes

ABR : complexité en moyenne

On considère $\mathcal{A}_k$, l'ensemble des ABR obtenus par toutes les séquences d'ajouts successifs de k éléments de clefs distinctes dans un ABR vide. On cherche h_k, la **moyenne** des hauteurs moyennes de A_C pour $A \in \mathcal{A}_k$:

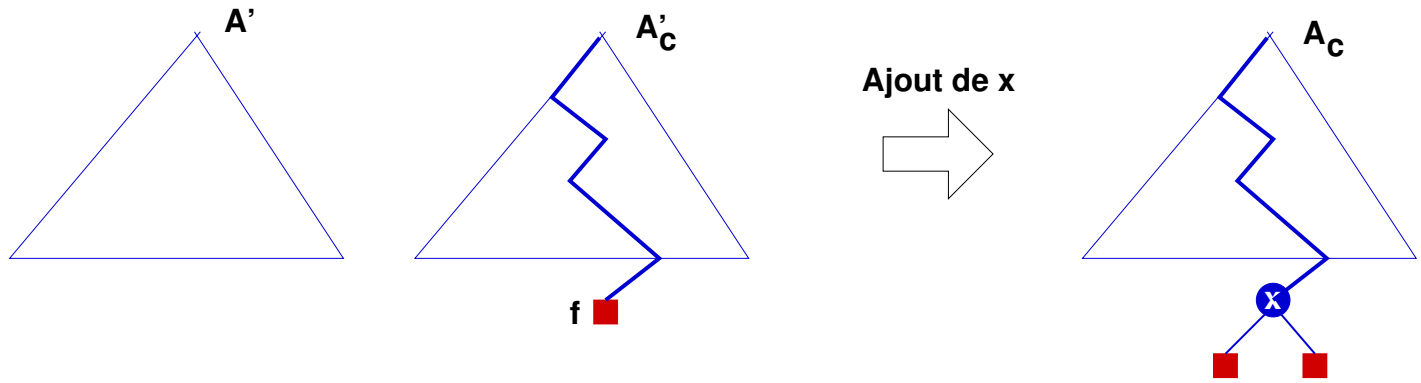
$$\forall A \in \mathcal{A}_k, \overline{h}(A_C) = \frac{1}{k+1} l(A_C)$$

$$h_k = \frac{1}{k!} \sum_{A \in \mathcal{A}_k} \overline{h}(A_C) = \frac{1}{k!} \sum_{A \in \mathcal{A}_k} \frac{l(A_C)}{k+1} = \frac{1}{(k+1)!} \sum_{A \in \mathcal{A}_k} l(A_C)$$

ABR : complexité en moyenne

Relation entre $\mathcal{A}_k$ et $\mathcal{A}_{k-1}$

Tout arbre $A^f \in \mathcal{A}_k$ est obtenu par l'ajout d'un nouvel élément x à un arbre $A' \in \mathcal{A}_{k-1}$. Soit f le nœud vide de A' sur lequel est réalisé l'ajout de x pour obtenir A^f :



$$l(A^f_C) = l(A'_C) - p(f) + 2(p(f)+1) = l(A'_C) + p(f) + 2$$

ABR : complexité en moyenne

Calcul de h_k par récurrence

$$h_k = \frac{1}{(k+1)!} \sum_{A \in \mathcal{A}_k} l(A_C) = \frac{1}{(k+1)!} \sum_{A' \in \mathcal{A}_{k-1}} \Big(\sum_{f \in A'_C} l(A'^f_C) \Big) =$$

$$\frac{1}{(k+1)!} \sum_{A' \in \mathcal{A}_{k-1}} \Big(\sum_{f \in A'_C} [\, l(A'_C) + p(f) + 2 \,] \Big) =$$

$$\frac{1}{(k+1)!} [\, k \sum_{A' \in \mathcal{A}_{k-1}} l(A'_C) + \sum_{A' \in \mathcal{A}_{k-1}} l(A'_C) + 2k(k-1)! \,]$$

ABR : complexité en moyenne

Calcul de h_k par récurrence (suite)

$$h_k = \frac{1}{(k+1)!}[\ 2k(k-1)! + (k+1) \sum_{A' \in \mathcal{A}_{k-1}} l(A'_C)\ ]$$

$$h_k = \frac{2}{k+1} + \frac{1}{k!} \sum_{A' \in \mathcal{A}_{k-1}} l(A'_C)$$

$$h_k = \frac{2}{k+1} + h_{k-1}$$

ABR : complexité en moyenne

Calcul de h_k par récurrence (fin)

$$h_k = \frac{2}{k+1} + h_{k-1} = \frac{2}{k+1} + \frac{2}{k} + h_{k-2} = \frac{2}{k+1} + \frac{2}{k} + ... + \frac{2}{3} + h_1$$

$$h_k = 1 + 2(\frac{1}{3} + \frac{1}{4} + ... + \frac{1}{k+1}) = 2H_{k+1} - 2$$

où $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + ... + \frac{1}{n}$ est le n-ième nombre harmonique ($H_n = \ln(n) + O(1)$)

$\Longrightarrow h_k \approx \lg(k)$ pour k assez grand

ABR : synthèse

Caractéristiques

- mise en œuvre simple et taille dynamique
- la complexité de toutes les opérations est proportionnelle à la hauteur de l'arbre :
  - $\Theta(\lg n)$ si l'arbre est *équilibré*
  - $\Theta(n)$ si l'arbre est *filaire*
- $\Theta(\lg n)$ en *moyenne* si l'arbre est construit par ajouts successifs à partir de l'arbre vide

AVL : arbres H-équilibrés

Définition

$$A \text{ est H-équilibré } \iff |h(g(A')) - h(d(A'))| \leq 1$$

pour tout A' sous-arbre de A, et où $g(X)$, $d(X)$ et $h(X)$ sont resp. le sous-arbre gauche, le sous-arbre droit et la hauteur de l'arbre X

Propriété

Pour tout arbre H-équilibré de taille n et de hauteur h :

$$\lg(n+1) \leq h + 1 < 1,44 \lg(n+2)$$

AVL : arbres H-équilibrés

Preuve

Etant donné un arbre H-équilibré de taille n et de hauteur $h \geq 1$, pour h fixé, n est

- *maximum* : quand l'arbre est *complet*, soit quand $n = 2^{h+1} - 1 \Longrightarrow n + 1 \leq 2^{h+1} \Longrightarrow \lg(n+1) \leq h + 1$
- *minimum* : quand $n = N(h)$ où $N(h)$ est la taille d'un arbre H-équilibré de hauteur h qui a le moins d'éléments.

AVL : arbres H-équilibrés

Calcul de $N(h)$

- $N(h) = 1 + N(h-1) + N(h-2)$ avec $N(0) = 1, N(1) = 2$
- si $F(h) = N(h) + 1$ on a $F(h) = F(h-1) + F(h-2)$ avec $F(0) = 2, F(1) = 3$
- F est une récurrence de Fibonacci qui a pour solution

$$F(h) = \frac{1}{\sqrt{5}}(\phi^{h+3} - \overline{\phi}^{h+3}) \text{ avec } \phi = \frac{1+\sqrt{5}}{2} \text{ et } \phi = \frac{1-\sqrt{5}}{2}$$

AVL : arbres H-équilibrés

Majoration de n

- on a

$$N(h) + 1 = \frac{1}{\sqrt{5}}(\phi^{h+3} - \overline{\phi}^{h+3})$$

- ce qui donne

$$n + 1 \geq \frac{1}{\sqrt{5}}(\phi^{h+3} - \overline{\phi}^{h+3})$$

- qui conduit à

$$h + 1 < 1,44 \lg(n + 2)$$

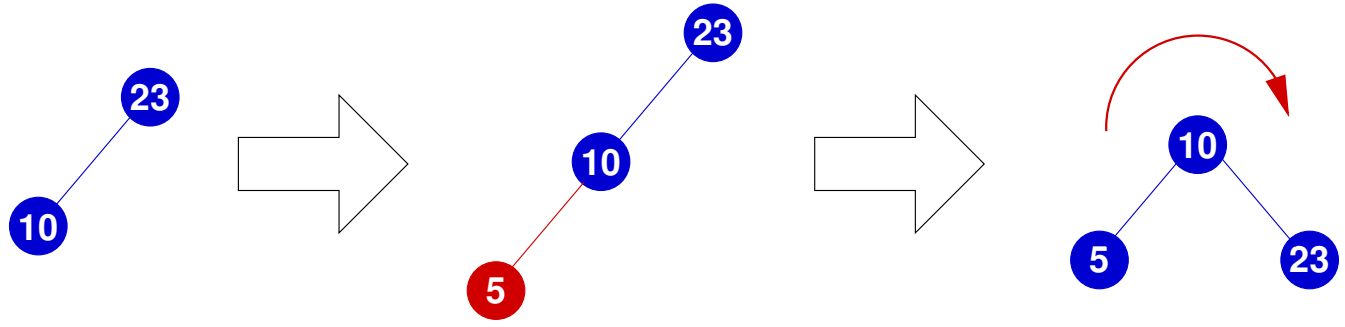
AVL : définition

Inventés par Adelson-Velskii et Landis en 1960

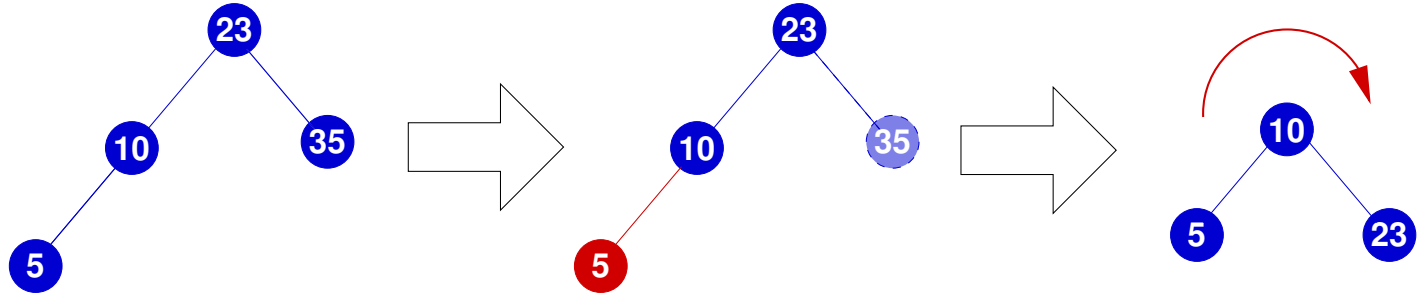
- A est un AVL $\Longleftrightarrow$ A est un ABR H-équilibré
- l'arbre vide est AVL
- si A est AVL, il peut **toujours** le rester après :
  - un ajout
  - une suppression
- éventuellement en modifiant l'ABR obtenu par rotation de certains sous-arbres

AVL : exemples

Ajout

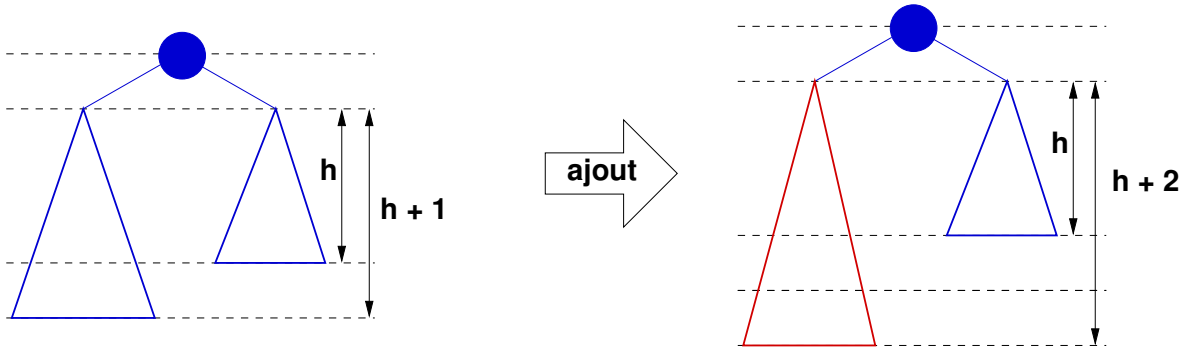


Suppression

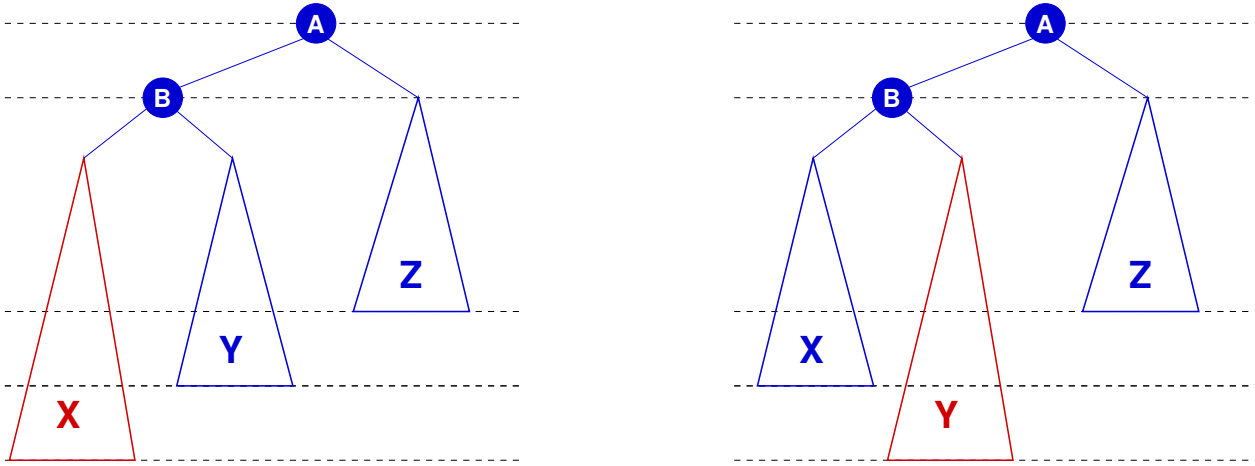


AVL : ajout

Déséquilibre à gauche (et le symétrique à droite)

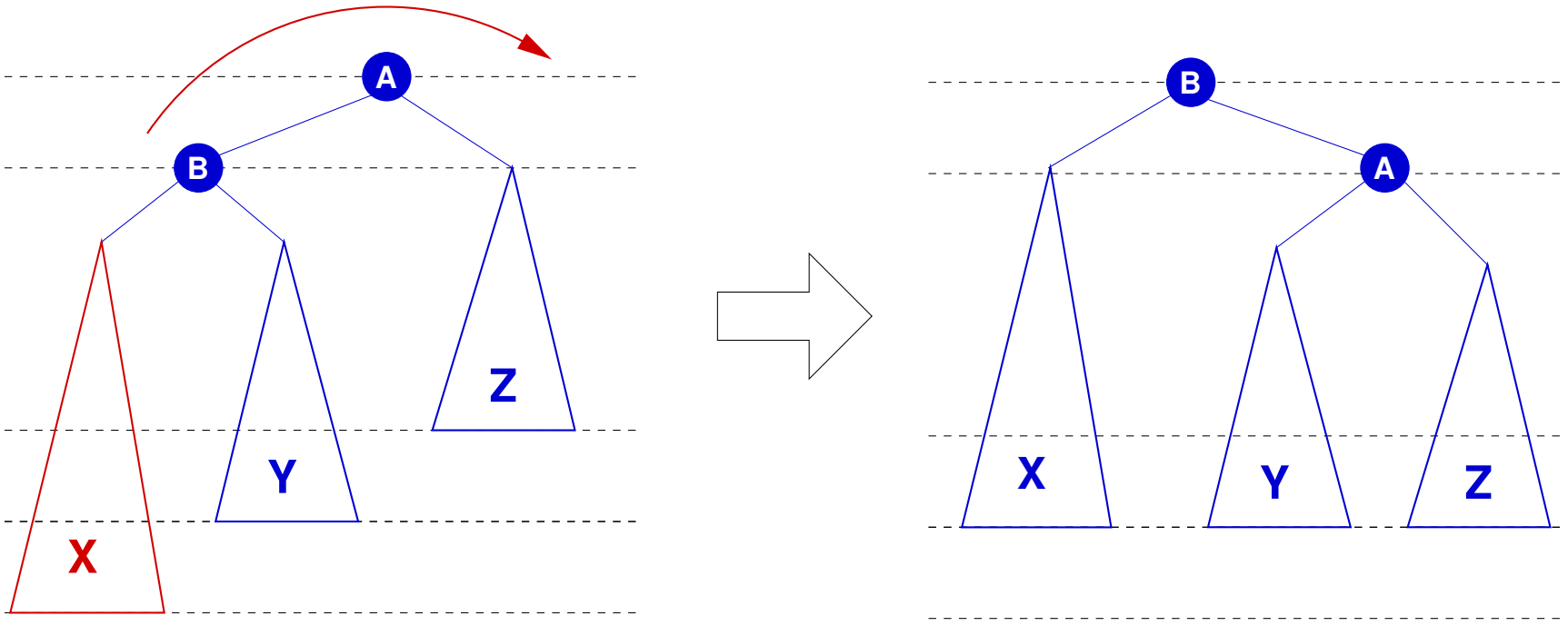


Deux cas (et deux cas symétriques à droite)



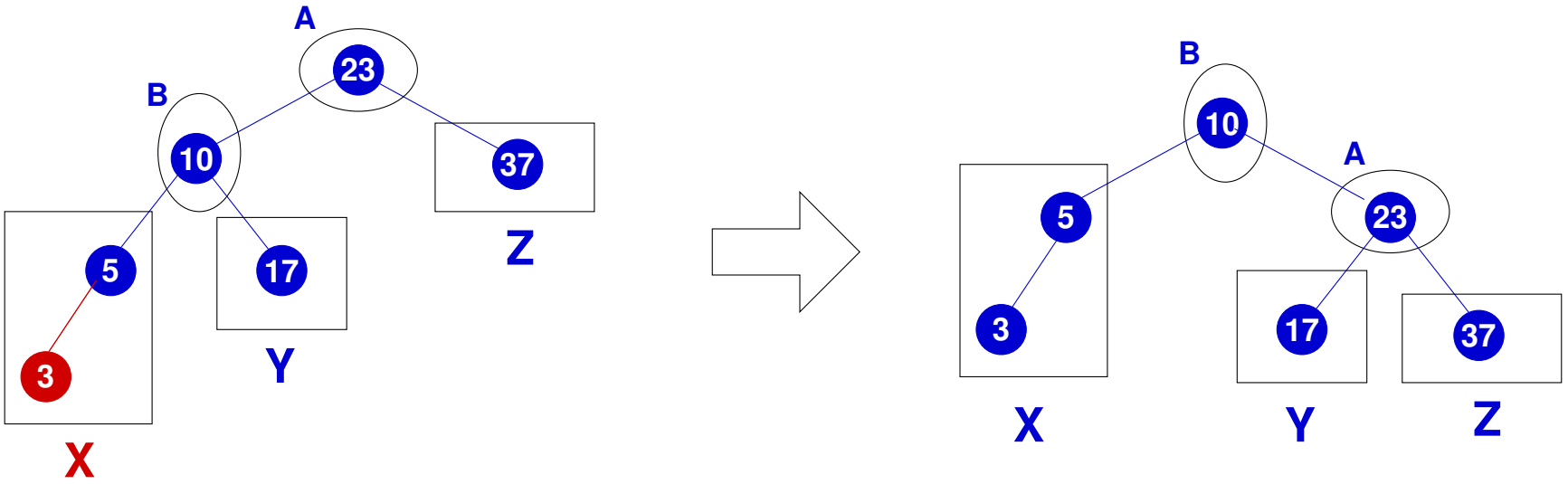
AVL : ajout

Cas 1 : une rotation droite (et symétrique gauche)



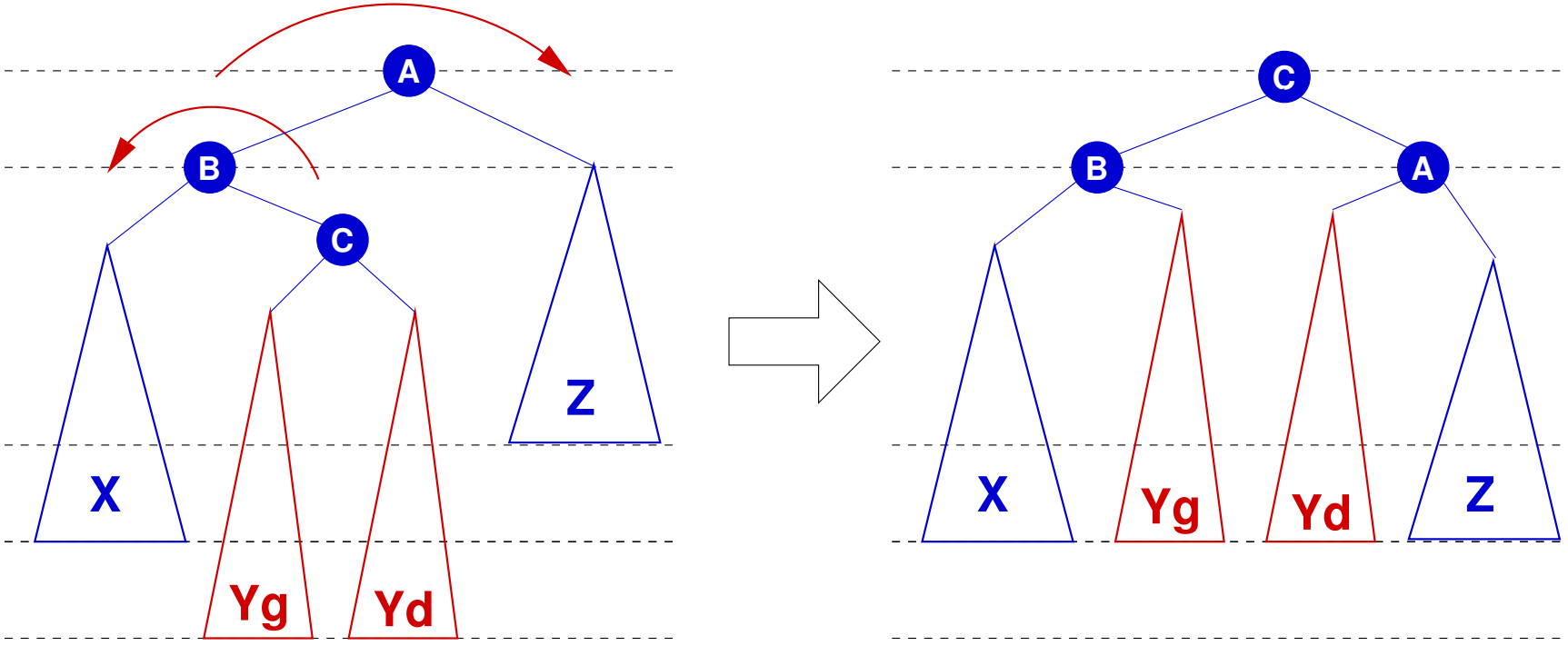
AVL : ajout

Cas 1 (gauche) : exemple



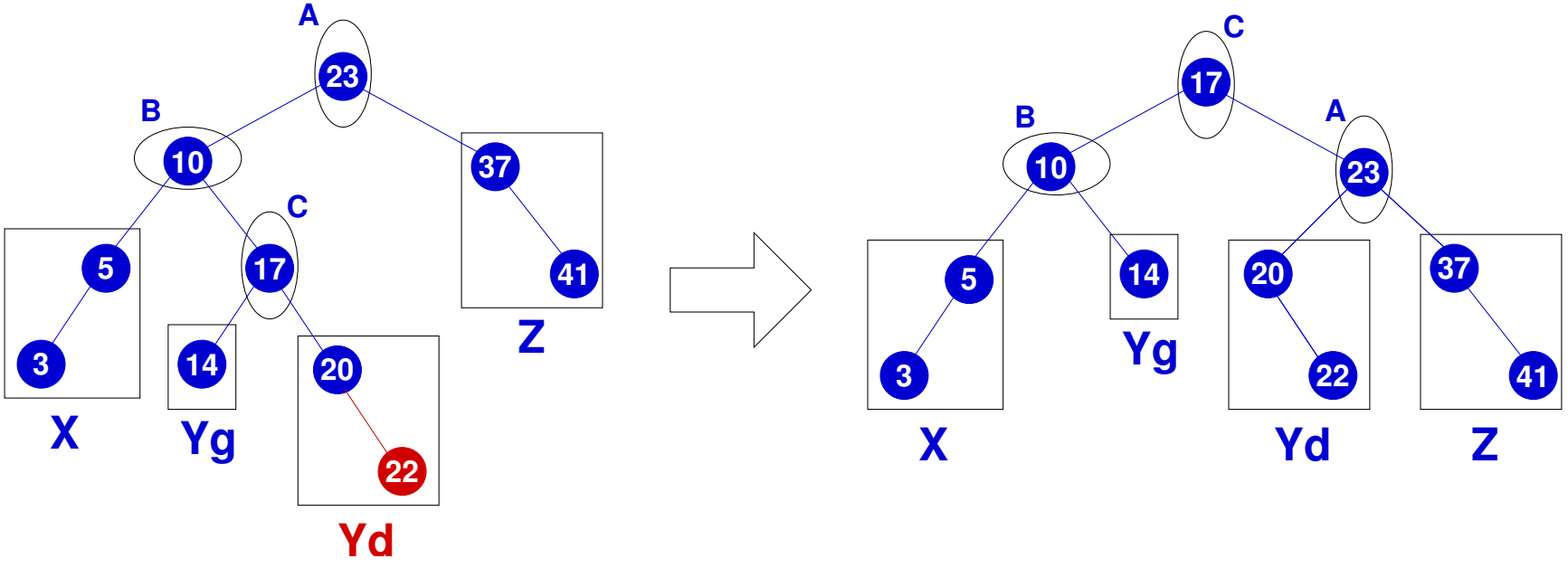
AVL : ajout

Cas 2 : deux rotations gauche-droite (et symétrique droite-gauche)



AVL : ajout

Cas 2 (gauche-droite) : exemple



AVL : synthèse

Caractéristiques

- mise en œuvre simple (ABR + rotations)
- la complexité de toutes les opérations est proportionnelle à la hauteur de l'arbre qui reste en $\Theta(\lg n)$ avec une faible constante
- un ajout entraîne au plus une rotation
- une suppression peut entraîner jusqu'à $1,5 \lg n$ rotations