Arbres AVL

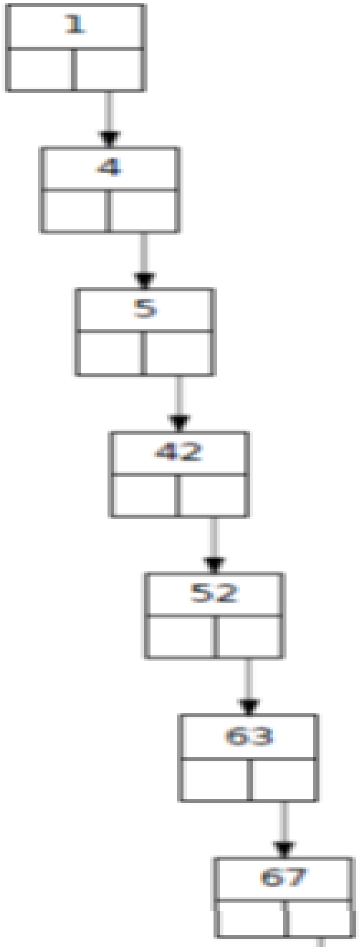
SECTION 4.4 (WEISS)

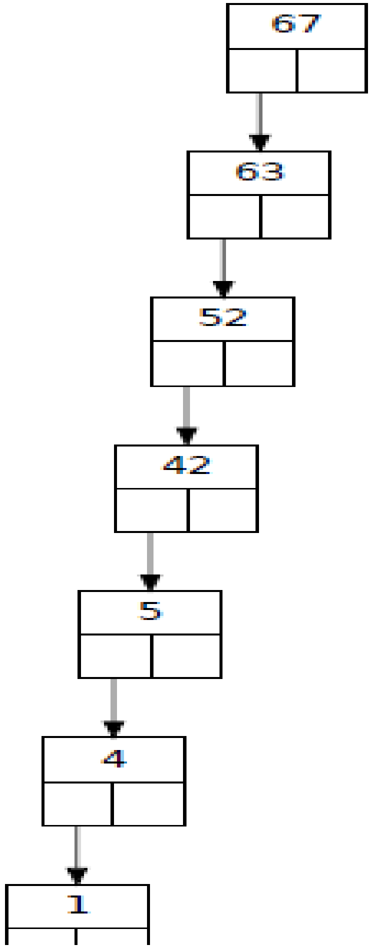
STRUCTURES DE DONNÉES ET ALGORITHMES, NOTES DE COURS (É.BAUDRY)

Arbres AVL

- Les arbres binaires de recherche peuvent s'avérer inefficaces s'ils ne sont pas adéquatement équilibrés
- Le pire cas se produit lorsqu'on insère des éléments totalement ordonnés (croissant ou décroissant)
- Dans le pire cas, la recherche s'effectue en temps O(n) où n est le nombre de nœuds

Arbres binaires dégénérés





Arbres AVL

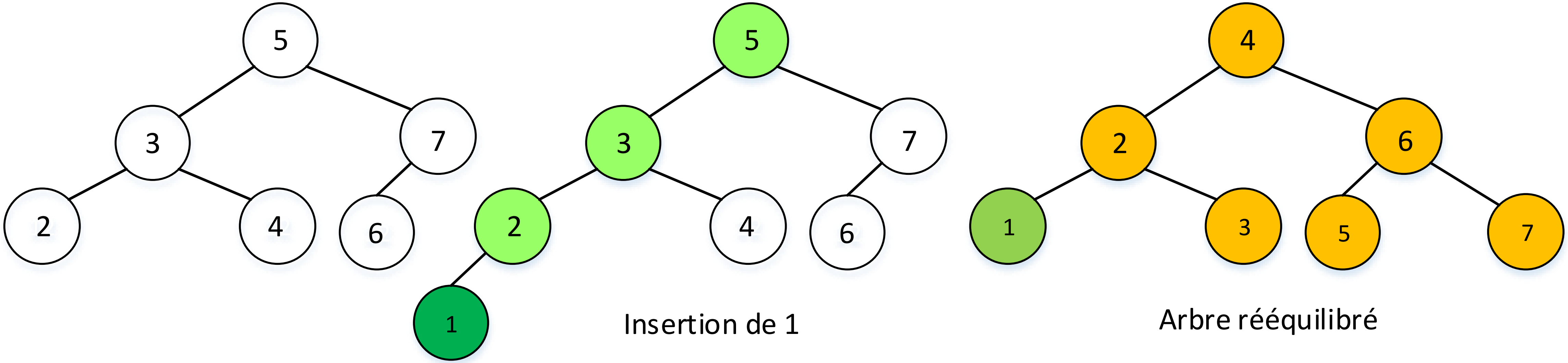
- Pour garantir une recherche plus efficace, s'effectuant en temps O(logn), il faut être en mesure d'éliminer environ la moitié de l'espace de recherche à chaque étape
- Afin de couper par deux l'espace de recherche, les sous arbres de gauche et de droite doivent être équilibrés en terme de nombre de nœuds qu'ils contiennent

Arbres AVL

- Arbre équilibré
- Équilibre parfait
  - sous arbres gauche et droite ont exactement le même nombre de nœuds => la même hauteur
  - difficile à maintenir
- Équilibre quasi parfait
  - tous ses niveaux sont remplis, mais à l’exception du dernier niveau
  - difficile à maintenir

Arbres AVL

- Arbres équilibrés: difficile à maintenir



Insertion de 1

Arbre rééquilibré

Arbres AVL

- Assouplir la définition d'arbre équilibré
- Les arbres AVL [AVL62] portent le nom de leurs inventeurs, les russes Georgii Adelson-Velsky et Evguenii Landis
- Un arbre AVL est un arbre binaire de recherche et équilibré
- **Équilibré**: Pour chacun des nœuds d'un arbre AVL, la différence de hauteur entre ses sous arbre de gauche et sous arbre de droite est d'au plus de un, c.-à-d.

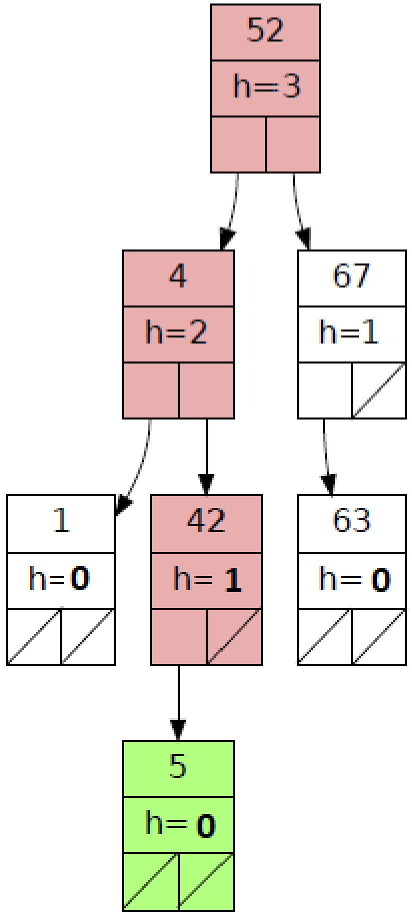
$$\left|h_g - h_d\right| \leq 1$$

Arbres AVL

- Lorsque la contrainte $|h_g - h_d| \leq 1$ est violée, on fait la restructuration d'arbre
  - des rotations sont faites autour des nœuds
- La règle d'équilibre permet de garantir que les opérations de recherche, d'insertion, et d'enlèvement demeurent toujours en temps O(logn)

Arbres AVL

- Arbre équilibré: AVL
- $|h_g - h_d| \leq 1$



Arbres AVL

- Calcul de la hauteur
  - Complexité du calcul de la hauteur d’un sous arbre

$$\Downarrow$$

$$O(n)$$

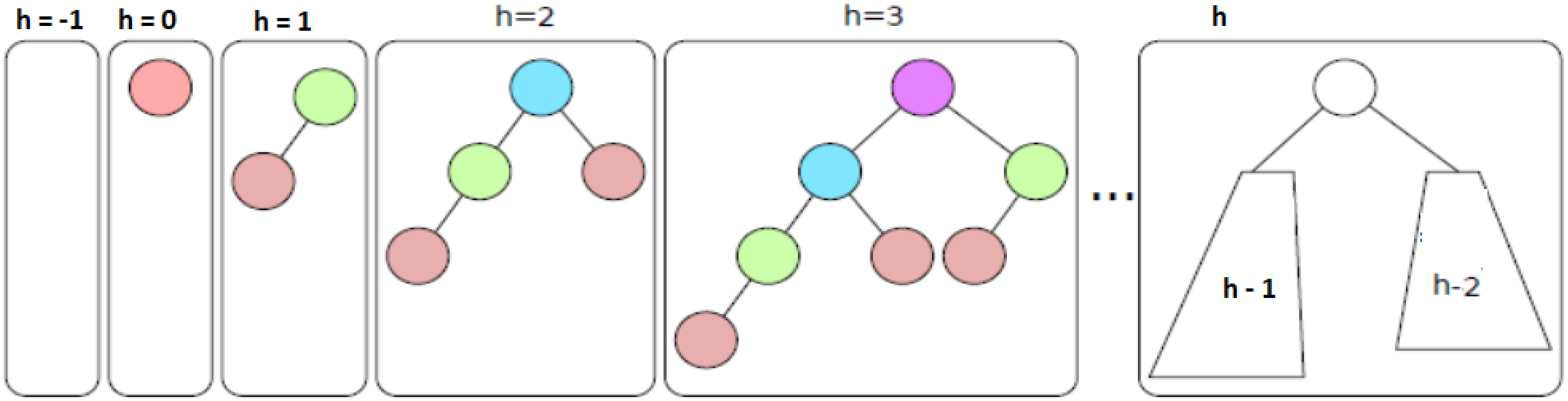
- Au lieu de calculer la hauteur à chaque fois, on peut maintenir à jour un champ hauteur dans chaque nœud

Arbres AVL

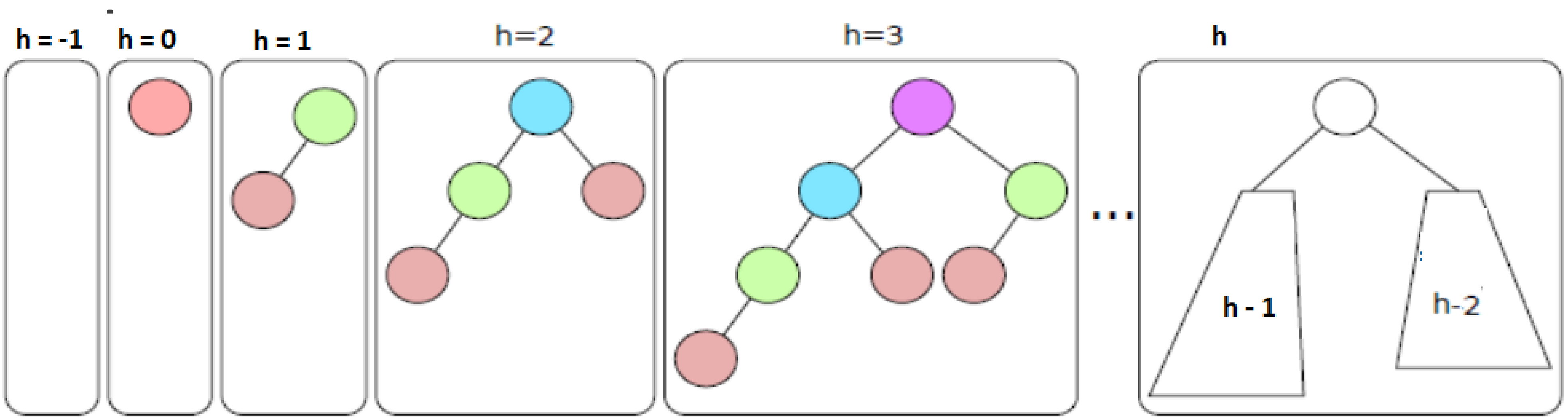
- En réalité, ce n’est pas la hauteur qui nous intéresse !
- Pour chaque nœud, nous voulons connaître la différence de hauteur entre les deux sous arbres
- Au lieu de conserver un champ hauteur, on peut conserver un indice d’équilibre égale à la différence des hauteurs des deux sous arbres

Arbres AVL

- Hauteur
- Les arbres minimaux, pour une hauteur -1, 0, 1, 2, etc.



Arbres AVL

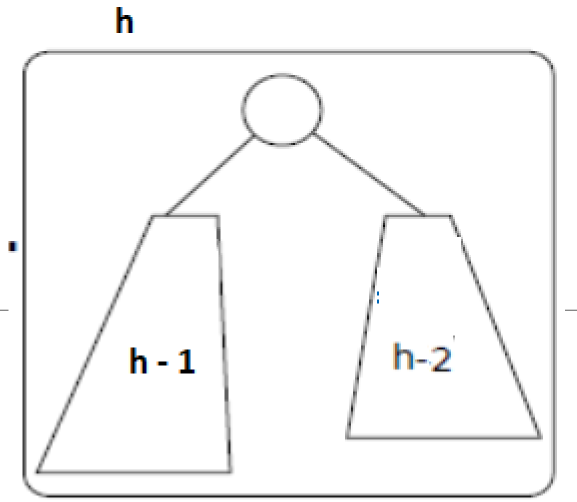


Fonction n(h) - le nombre minimal de nœuds d'un arbre AVL de hauteur h

$$n(h) = \begin{cases} 0, h = -1 \\ 1, h = 0 \\ n(h-1) + n(h-2) + 1, h \geq 1 \end{cases}$$

$$\text{fib}(h) = \begin{cases} 0, h = -1 \\ 1, h = 0 \\ fib(h-1) + fib(h-2), h \geq 1 \end{cases}$$

Arbres AVL



$$n(h) = \begin{cases} 0, h = -1 \\ 1, h = 0 \\ n(h-1) + n(h-2) + 1, h \geq 1 \end{cases}$$

$$\text{fib}(h) = \begin{cases} 0, h = -1 \\ 1, h = 0 \\ fib(h-1) + fib(h-2), h \geq 1 \end{cases}$$

Analyse de Fibonacci: $n(h) > \ \phi^h \ (\phi \approx 1.62)$

Supposons que n- un nombre de nœuds dans un arbre AVL de l'hauteur h:

$$n \ \geq n(h) > \ \phi^h \Rightarrow \ log_\phi n \geq h \Rightarrow h \leq 1.44 log_2 n$$

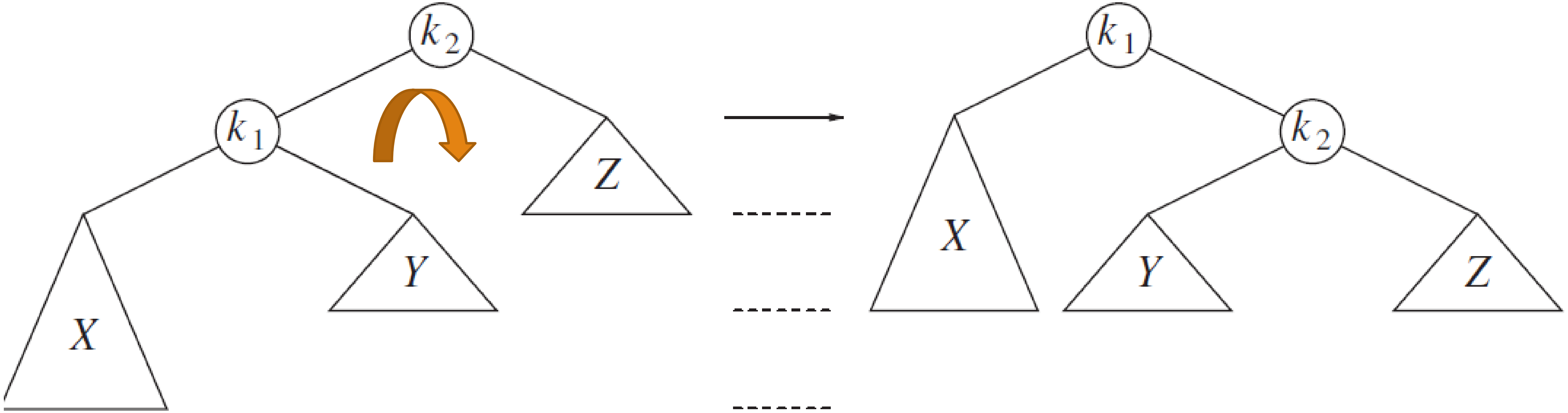
Hauteur maximale d'un arbre AVL de n nœuds : $h < 1.44 log_2(n+2) - 0.328$

Arbres AVL

- Maintenir l’arbre équilibré après chaque modification
- 4 cas d’un déséquilibre: 2+2 (symétriques)
- Cas 1: Une insertion dans le sous arbre de gauche d’un enfant gauche
- Cas 2 (symétrique): Une insertion dans le sous arbre de droite d’un enfant droite
- Cas 3: Une insertion dans le sous arbre de gauche d’un enfant droite
- Cas 4 (symétrique): Une insertion dans le sous arbre de droite d’un enfant gauche

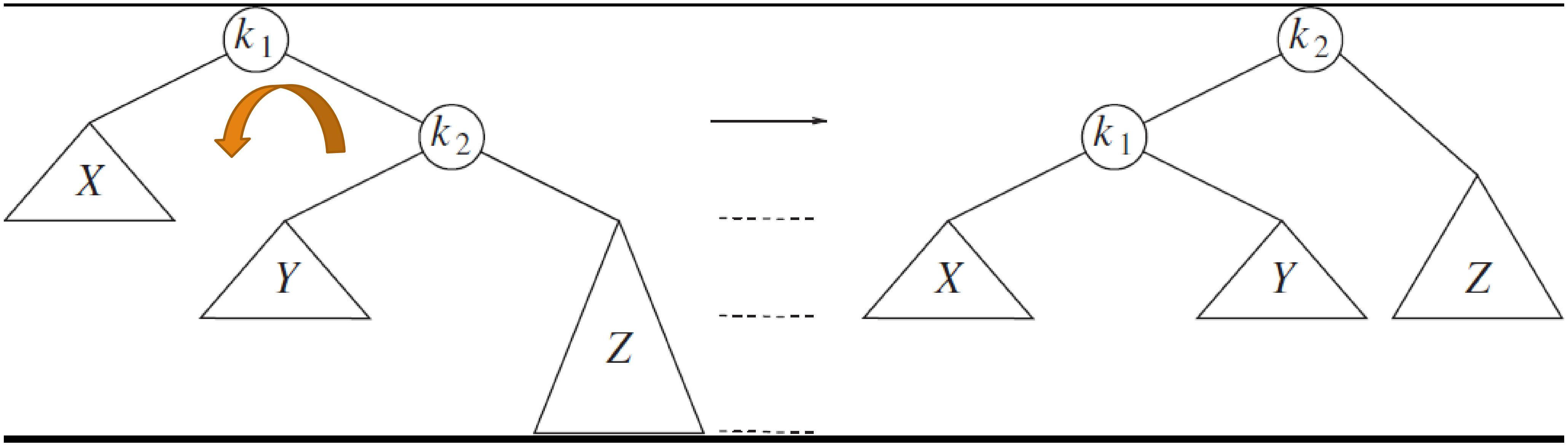
Arbres AVL

- Cas 1: l’arbre de racine k2 penche à gauche et son sous arbre gauche de racine k1 penche à gauche => **Rotation simple à droite**



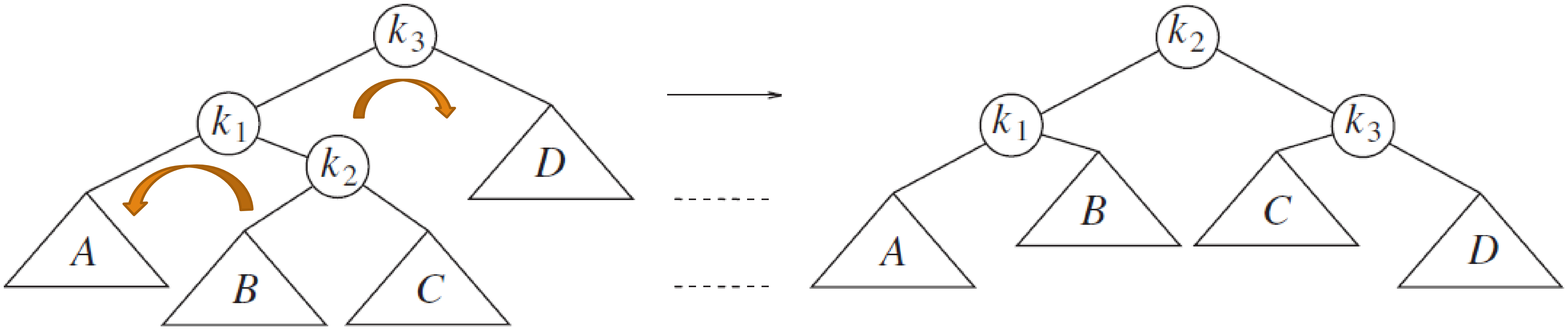
Arbres AVL

- Cas 2: l'arbre de racine k1 penche à droite et son sous arbre droite de racine k2 penche à droite => <u>**Rotation simple à gauche**</u>



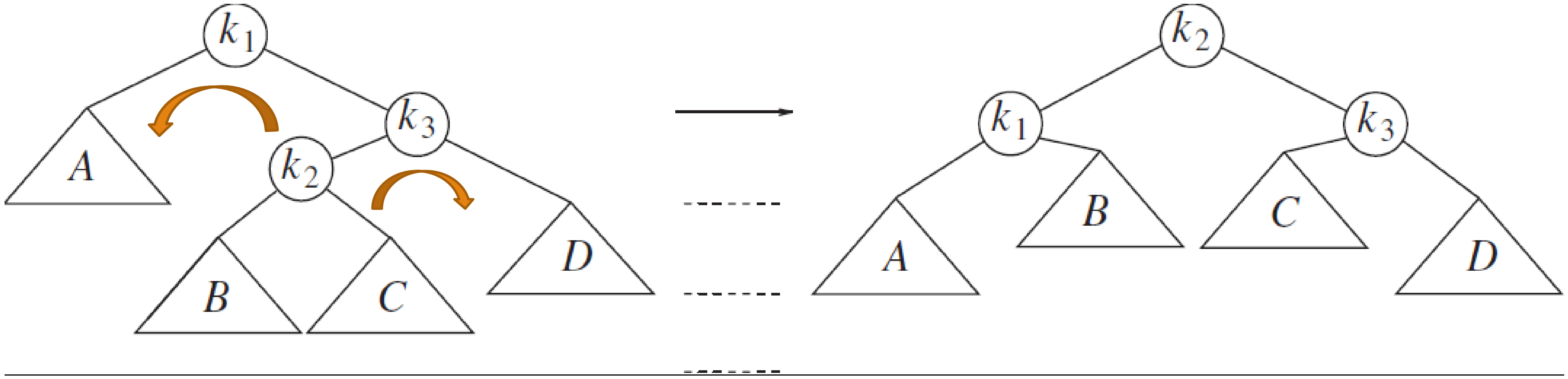
Arbres AVL

- Cas 3: Une insertion dans le sous arbre de gauche d'un enfant droite => <u>**Rotation double gauche - droite**</u>



Arbres AVL

- Cas 4: Une insertion dans le sous arbre de droite d'un enfant gauche => <u>**Rotation double droite - gauche**</u>



Arbres AVL

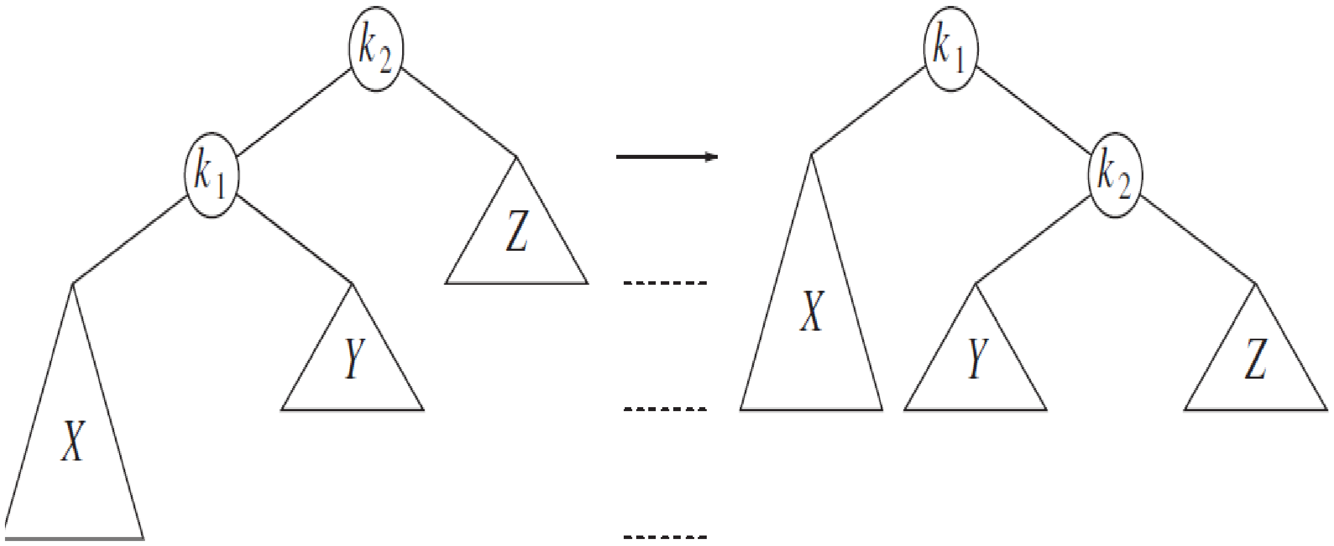
- Représentation, classe AVL Nœud

```
private static class AvlNode<AnyType>{
   AvlNode( AnyType theElement ) { this( theElement, null, null ); }
   AvlNode( AnyType theElement, AvlNode<AnyType> lt,
                                                AvlNode<AnyType> rt )
        { element = theElement; left = lt; right = rt; height = 0; }

  AnyType element; // The data in the node
  AvlNode<AnyType> left; // Left child
  AvlNode<AnyType> right; // Right child
  int balance; // indice d'équilibre
}
```

Arbres AVL

- Rotation simple, classe AVL Nœud

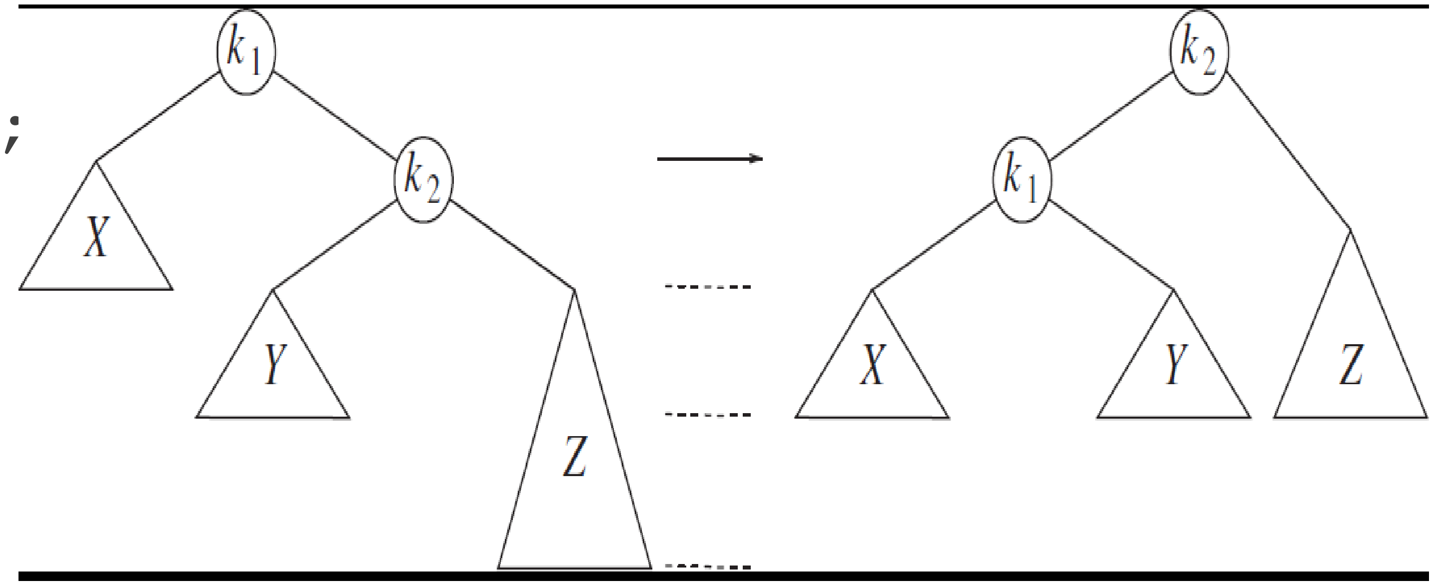
```
 /* Rotate binary tree node with left child. * For AVL
trees, this is a single rotation for case 1.*/
 static AvlNode rotateWithLeftChild( AvlNode k2 ) {
     AvlNode k1 = k2.left;
     k2.left = k1.right;
     k1.right = k2;
     return k1;
  }
```



Arbres AVL

- Rotation simple, classe AVL Nœud

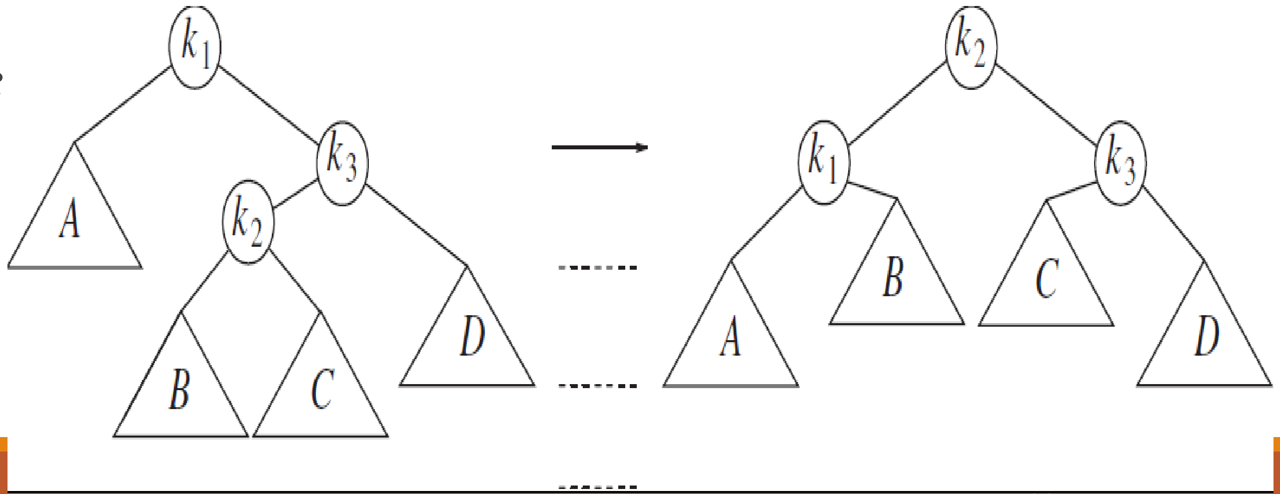
```
/* Rotate binary tree node with right child. * For AVL
trees, this is a single rotation for case 2.*/
static AvlNode rotateWithRightChild( AvlNode k1 )
{
 AvlNode k2 = k1.right;
 k1.right = k2.left;
 k2.left = k1;
 return k2;
}
```



Arbres AVL

Rotation double, classe AVL Nœud

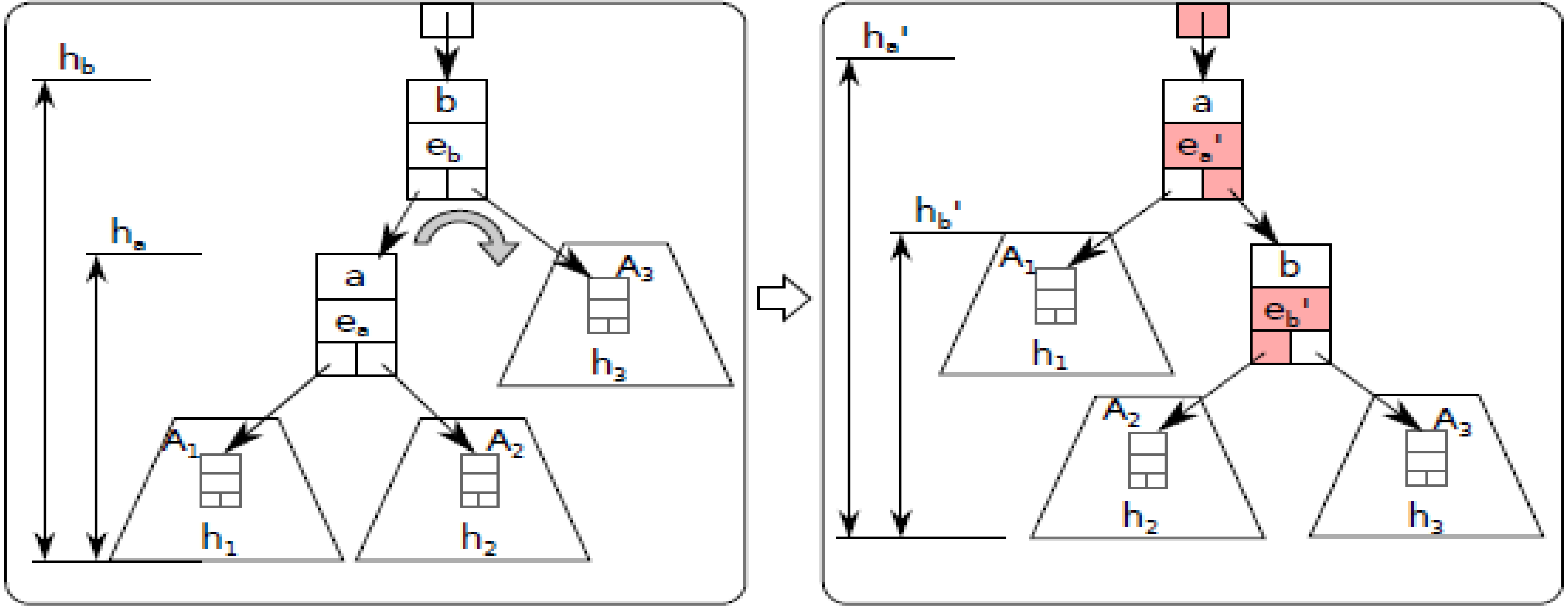
```
/** Double rotate binary tree node: first left child with
its right child; then node k3 with new left child.
For AVL trees, this is a double rotation for case 4.*/

static AvlNode doubleRotateWithLeftChild(AvlNode k3)
{

   k3.left = rotateWithRightChild( k3.left );

   return
     rotateWithLeftChild( k3 );

}
```



Arbres AVL

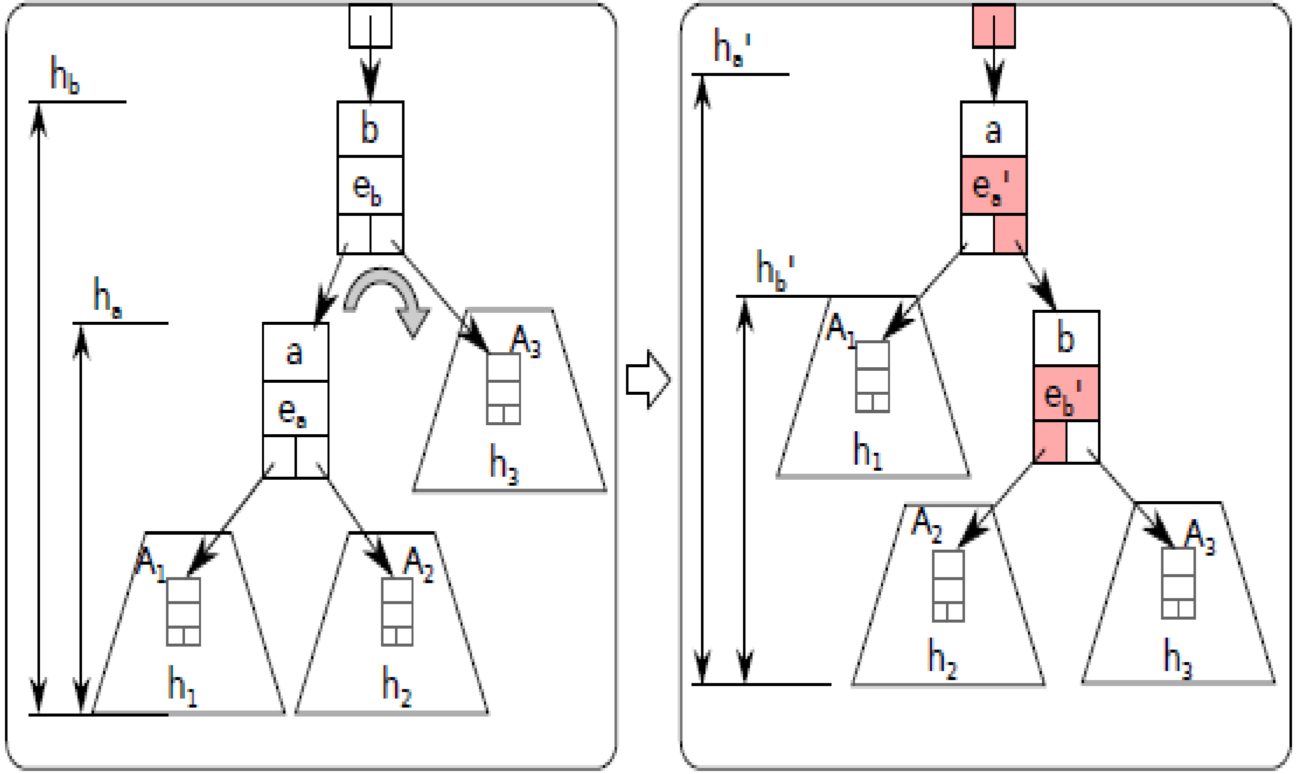
- Calcul d'indice d'équilibre

É. Baudry

- 

Arbres AVL

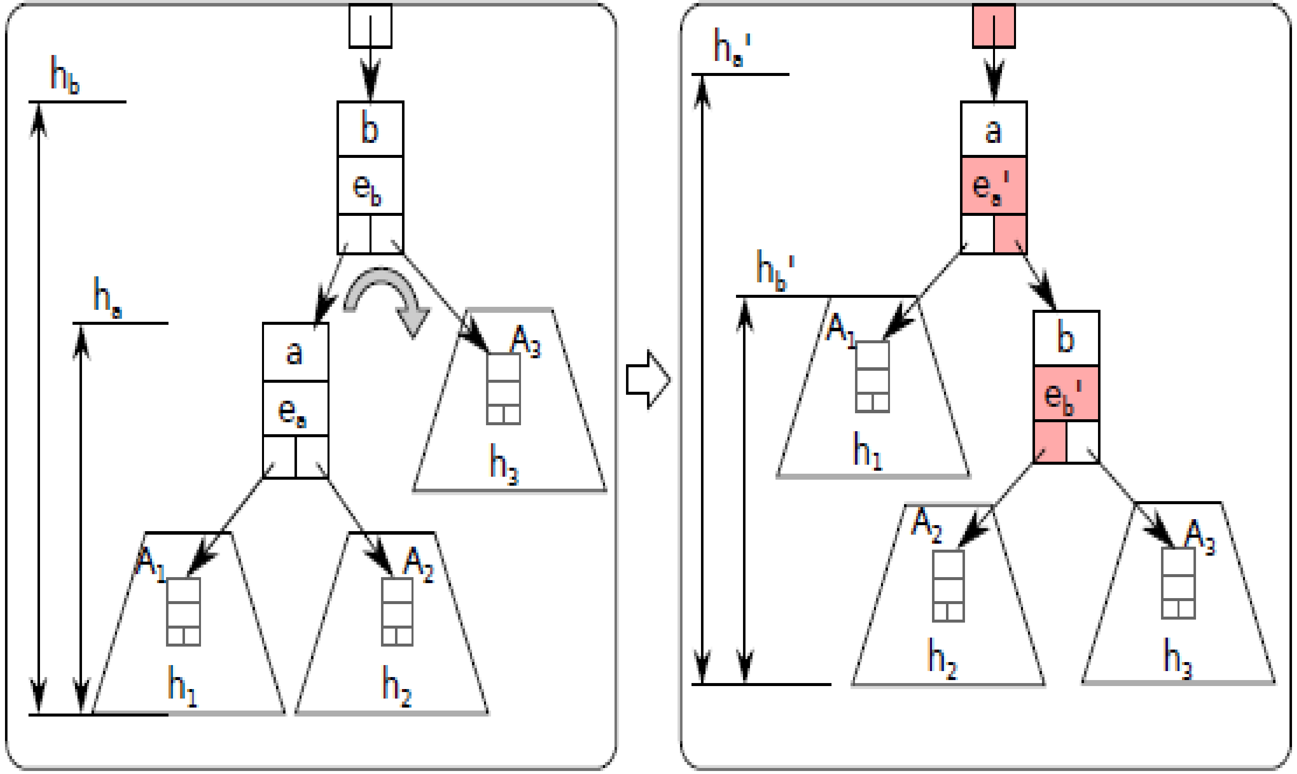
- Calcul d'indice d'équilibre



$$e_b' = h_2 - h_3$$
$$h_3 = h_a - e_b$$
$$h_a = \max(h_1, h_2) + 1$$
$$h_1 = h_2 + e_a$$
$$h_a = \max((h_2 + e_a), h_2) + 1$$
$$= h_2 + \max(e_a, 0) + 1$$
$$e_b' = h_2 - (h_a - e_b)$$
$$= h_2 - ((h_2 + \max(e_a, 0) + 1 - e_b)$$
$$= -\max(e_a, 0) - 1 + e_b$$

Arbres AVL

- Calcul d'indice d'équilibre



$$e'_a = h_1 - h'_b$$
$$h'_b = \max(h_2, h_3) + 1$$
$$h_3 = h_2 - e'_b$$
$$h'_b = \max(h_2, h_2 - e'_b) + 1$$
$$= h_2 + \max(0, -e'_b) + 1$$
$$h_1 = h_2 + e_a$$
$$e'_a = (h_2 + e_a) - (h_2 + \max(0, -e'_b) + 1)$$
$$= e_a - \max(0, -e'_b) - 1$$
$$= e_a + \min(0, e'_b) - 1$$

Arbres AVL

- Exemples au tableau