

Arbres de recherche AVL

UMLV ©

Implémentation d'ensembles avec opérations :

MIN (A), MAX (A)	} $O(\log(A))$ pire des cas
AJOUTER (x, A)	
ENLEVER (x, A)	
ELEMENT (x, A)	

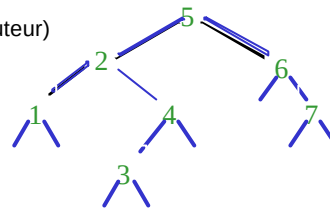
A arbre (binaire de recherche) AVL (équilibré en hauteur)

1/ en tout noeud p de A (si A non vide) :

$$|h(A_d(p)) - h(A_g(p))| \leq 1$$

2/ A vide ou

$A = (r, A_g, A_d)$, A_g, A_d sont des arbres AVL
et $|h(A_d) - h(A_g)| \leq 1$



$$\text{bal}(p) = \text{bal}(A(p)) = h(A_d(p)) - h(A_g(p))$$

1'/ en tout noeud p de A (si A non vide),

$$\text{bal}(p) = -1, 0 \text{ ou } +1$$

277

Implémentation des AVL

UMLV ©

```
struct NOEUD {
    element elt ;
    deuxbits bal ; /* compris entre -1 et +1 */
                  /* -2 et +2 dans la suite */
    struct NOEUD * g, * d ;
} noeud ;
noeud * arbre ;

(arbre,entier) AJOUTER (element x, arbre A) ;
/* rend l'arbre modifié et la différence de hauteur : 0 ou +1*/

(arbre,entier) ENLEVER (element x, arbre A) ;
/* rend l'arbre modifié et la différence de hauteur : -1 ou 0 */

(arbre,entier) OTERMIN (arbre A) ;
/* rend l'arbre modifié et la différence de hauteur : -1 ou 0 */
```

278

Hauteur des arbres AVL

UMLV ©

A arbre AVL à n nœuds

$$\log_2(n+1)-1 \leq h(A) \leq 1,45 \log_2(n+2)$$

ARBRES de FIBONACCI

[nœuds identifiés à leurs étiquettes, 1, 2, ...]

$F_0 = ()$ = arbre vide

$$h(F_0) = -1$$

$F_1 = (1, (), ())$

$$h(F_1) = 0$$

$F_k = (F_{k+1}, F_{k-1}, F_{k-2})$ si $k > 1$

$$h(F_k) = k-1$$

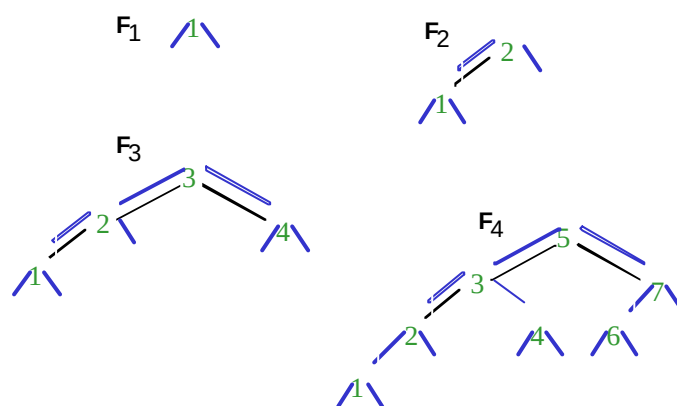
$F'_{k-2} = F_{k-2}$ avec étiquettes augmentées de F_{k+1}

[nombres de Fibonacci : $F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, \dots$]

279

AVL de Fibonacci

UMLV ©



[$F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, F_5 = 5, F_6 = 8, F_7 = 13, \dots$]

280

Propriétés

UMLV ©

L'arbre de Fibonacci F_k

- est un arbre AVL
- est étiqueté par 1, 2, ..., $F_{k+2}-1$
- a pour hauteur $k-1$

Tout arbre AVL de hauteur $k-1$ possède au moins
- $F_{k+2} - 1$ nœuds

Conséquence : pour A AVL avec n nœuds et hauteur $k-1$

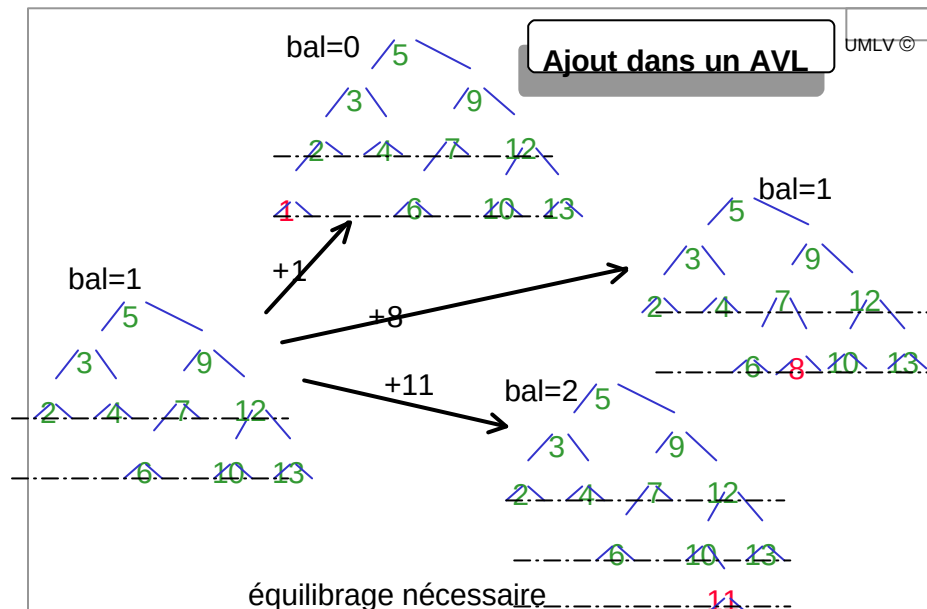
$$n \geq F_{k+2} - 1 > \Phi^{k+2} \sqrt{5} - 2$$

$$h(A) = k - 1 < 1,45 \log_2(n+2) - 0,33$$

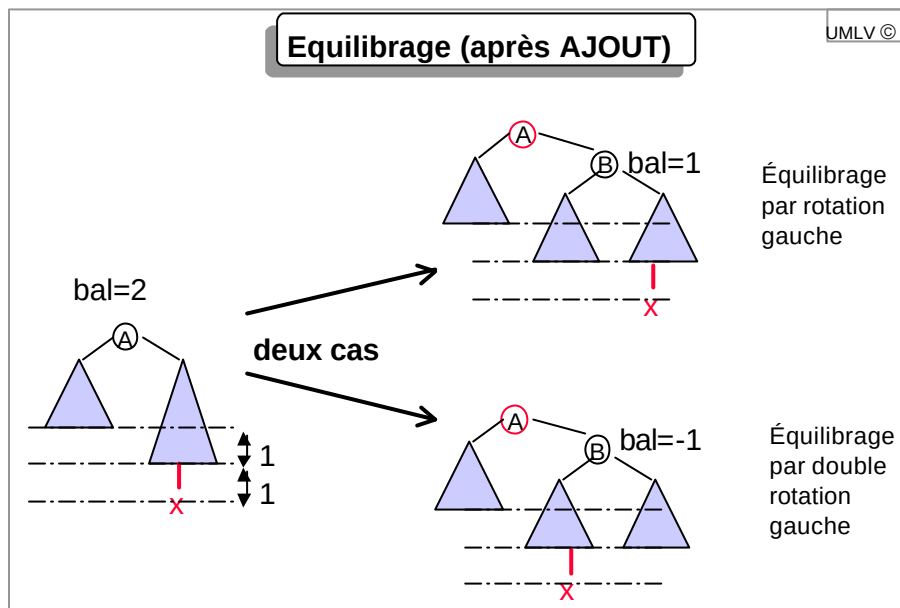
281

Ajout dans un AVL

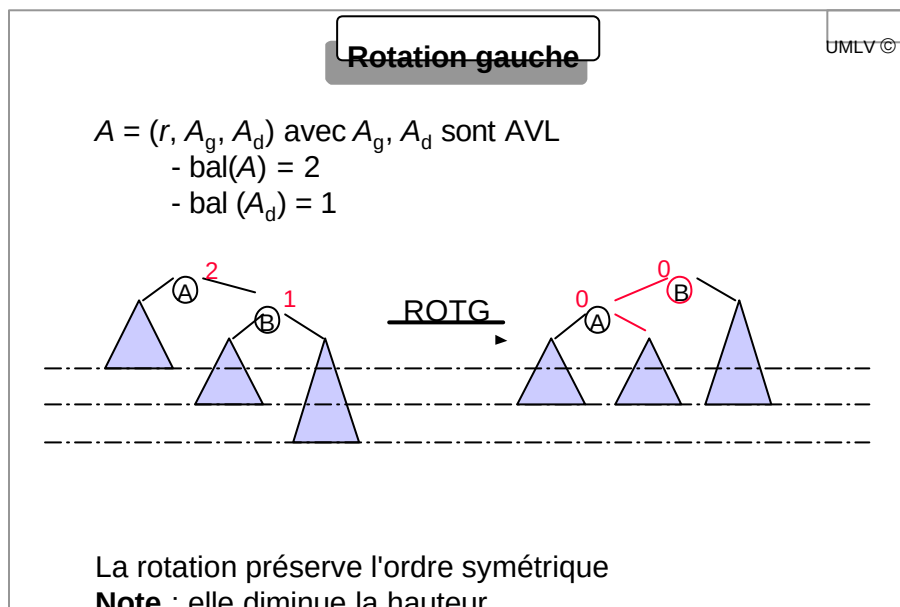
UMLV ©



282



283



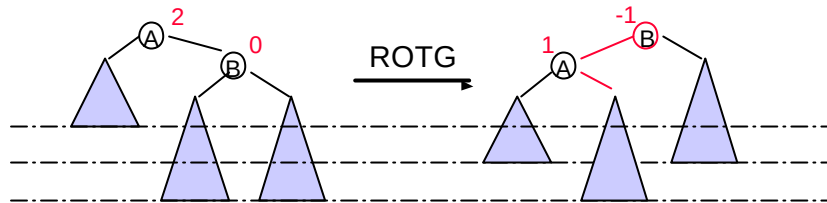
284

Rotation gauche (suite)

UMLV ©

$A = (r, A_g, A_d)$ avec A_g, A_d sont AVL

- $\text{bal}(A) = 2$
- $\text{bal}(A_d) = 0$



La rotation préserve l'ordre symétrique

Note : elle ne diminue pas la hauteur

285

```

arbre ROTG(arbre A) {
    arbre B; int a,b;
    B = A->d;
    a = A->bal;  b = B->bal;
    A->d = B->g; B->g = A;  /* rotation */
    A->bal = a-max(b,0)-1;
    B->bal = min(a-2,a+b-2,b-1);
    return B;
}
    
```

Exercices :

- vérifier les calculs de bal
- écrire la rotation droite ROTD

UMLV ©

286

```

arbre ROTD(arbre A) {
    arbre B; int a,b;
    B = A->g;
    a = A->bal;  b = B->bal;
    A->g = B->d; B->d = A;  /* rotation */
    A->bal = a-min(b,0)+1;
    B->bal = max(a+2,a+b+2,b+1);
    return B;
}

```

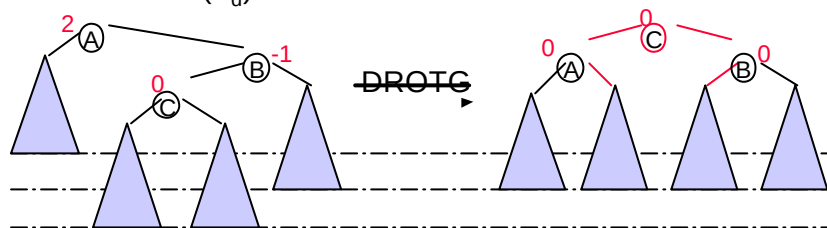
Exercice :

- vérifier les calculs de bal

Double rotation gauche

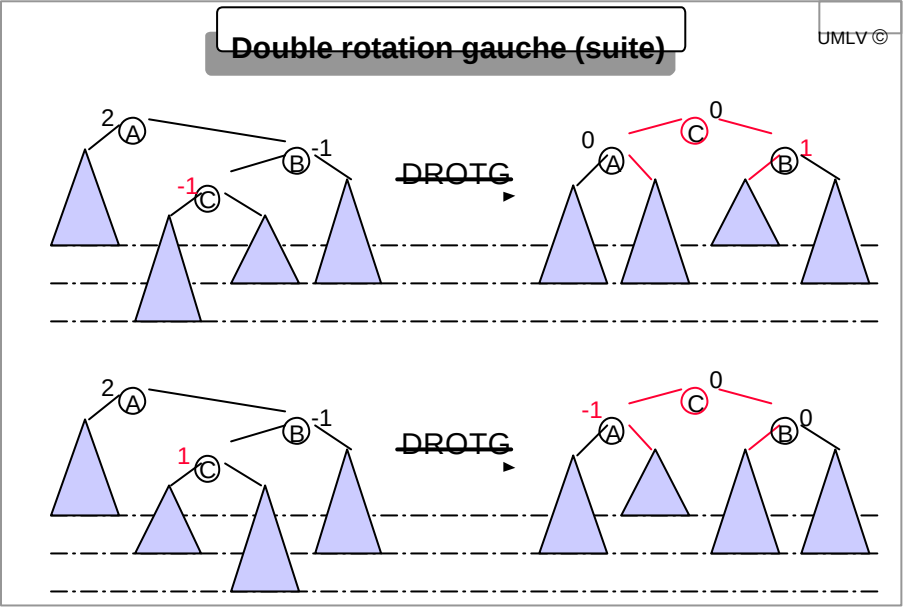
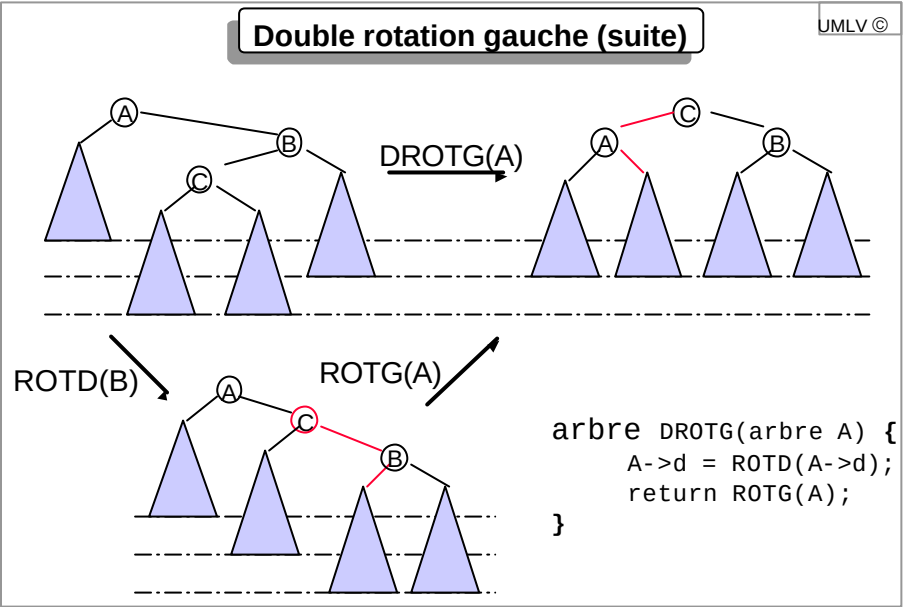
$A = (r, A_g, A_d)$ avec A_g, A_d sont AVL

- $\text{bal}(A) = 2$
- $\text{bal}(A_d) = -1$



La rotation préserve l'ordre symétrique

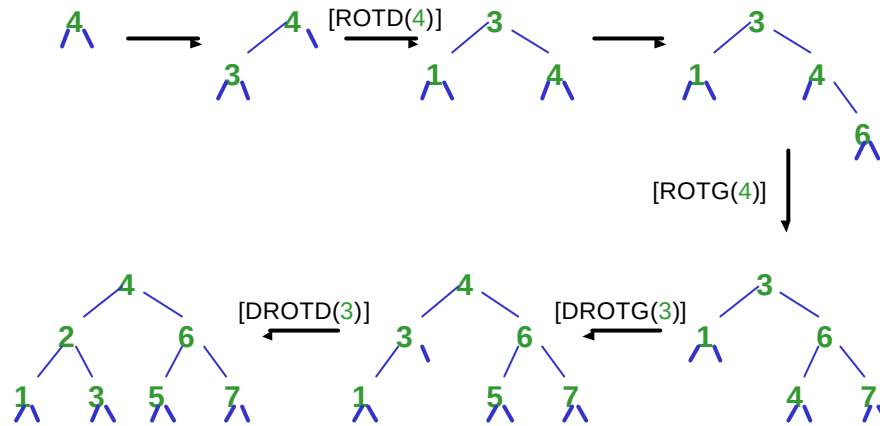
Note : elle diminue la hauteur



Exemples de rotations

UMLV ©

Ajouts successifs de 4, 3, 1, 6, 7, 5, 2 dans l'arbre vide



291

Équilibrage

UMLV ©

Entrée

A arbre tel que
 A_g, A_d sont AVL
 $-2 \leq \text{bal}(A) \leq 2$

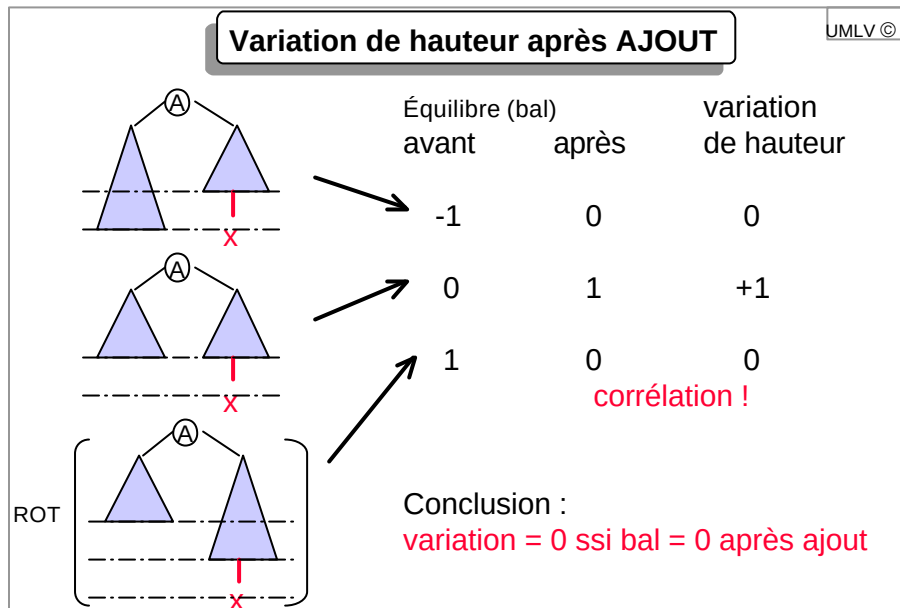
Sortie

A arbre AVL
 $[-1 \leq \text{bal}(A) \leq 1]$

```

arbre EQUILIBRER(arbre A) {
    if (A->bal == 2)
        if (A->d->bal >= 0)
            return ROTG(A);
        else { A->d = ROTD(A->d);
              return ROTG(A);
            }
    else if (A->bal == -2)
        if (A->g->bal <= 0)
            return ROTD(A);
        else { A->g = ROTG(A->g);
              return ROTD(A);
            }
    else return A;
}
    
```

292



293

UMLV ©

```

(arbre,int) AJOUTER(element x, arbre A) {
    if (A == NULL) {
        créer un nœud A;  A->g = NULL;  A->d = NULL;
        A->elt = x;  A->bal = 0;
        return (A,1);
    } else if (x == A->elt)
        return (A,0);
    else if (x > A->elt)
        (A->d,h) = AJOUTER(x,A->d);
    else
        (A->g,h) = AJOUTER(x,A->g); h = -h;
    if (h == 0)
        return (A,0);
    else
        A->bal = A->bal + h;
        A = EQUILIBRER(A);
        if (A->bal == 0) return (A,0); else return (A,1);
}

```

294

Temps d'un ajout

UMLV ©

Temps d'une rotation : constant

Note : **AJOUTER exécute au plus une rotation**
car une rotation dans cette situation rétablit la hauteur initiale de l'arbre auquel elle est appliquée

A arbre AVL à n nœuds

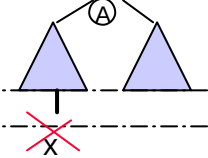
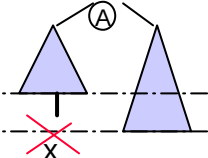
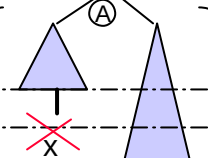
Temps total d'un ajout : $O(h(A)) = O(\log n)$

car une seule branche de l'arbre est examinée

295

Variation de hauteur après RETRAIT

UMLV ©

	Équilibre (bal)		
	avant	après	variation de hauteur
	-1	0	-1
	0	1	0
	1	0 ou -1	-1 0
			corrélation !
ROT			
			Conclusion : variation = -1 ssi bal = 0 après retrait

296

```

(arbre,int) ENLEVER(element x, arbre A) {
    if (A == NULL)
        return (A,0);
    else if (x > A->elt)
        (A->d,h) = ENLEVER(x,A->d);
    else if (x < A->elt)
        (A->g,h) = ENLEVER(x,A->g); h = -h;
    else if (A->g == NULL) {
        /* free */ return (A->d,-1);
    } else if (A->d == NULL) {
        /* free */ return (A->g,-1);
    } else {
        A->elt = min(A->d);
        (A->d,h) = OTERMIN(A->d);
    }
}
. . . .

```

```

. . . .
    if (h == 0)
        return (A,0);
    else {
        A->bal = A->bal + h;
        A = EQUILIBRER(A);
        if (A->bal == 0)
            return (A,-1);
        else
            return (A,0);
    }
}

```

Entrée
A arbre AVL
non vide

```
(arbre, int) OTERMIN(arbre A) {
    if (A->g == NULL) {
        min = A->elt;
        /* free */ return (A->d, -1);
    } else
        (A->g, h) = OTERMIN(A->g); h = -h;
    if (h == 0)
        return (A, 0);
    else {
        A->bal = A->bal + h;
        A = EQUILIBRER(A);
        if (A->bal == 0)
            return (A, -1);
        else
            return (A, 0);
    }
}
```

Temps d'un retrait

Note : ENLEVER et OTERMIN peuvent exécuter une rotation sur chaque ancêtre du nœud supprimé

Exemple : suppression du maximum
dans un arbre de Fibonacci

A arbre AVL à n nœuds

Temps total d'une suppression : $O(h(A)) = O(\log n)$
car ENLEVER et OTERMIN examinent une seule
branche de l'arbre (et temps d'une rotation constant)