

## TC Informatique

---

PC N° 7  
20 Décembre 2000

François Sillion

# **Arbres équilibrés** **Tas, *Heapsort***

<http://w3.edu.polytechnique.fr/informatique>

## Parcours d'arbres

---

- Dépend de l'ordre des appels récursifs
- Infixe, préfixe, postfixe
- Exemple de parcours préfixe :

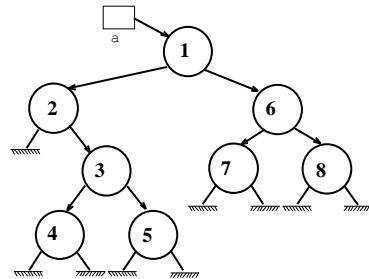
```
static void traverse( ArbreBinaire x )
{
    if ( x == null ) return;
    visite( x );
    traverse( x.gauche );
    traverse( x.droite );
}
```

## Parcours en profondeur d'abord

- équivalent à la version récursive

```
static void traverseProfondeur( ArbreBinaire x )
{
    File f = CreePile();
    Empile( f , x );

    while ( ! PileVide( f ) ) {
        x = Depile(f);
        visite( x );
        if ( x.droite != null )
            Empile( f , x.droite );
        if ( x.gauche != null )
            Empile( f , x.gauche );
    }
}
```



PC 7

François Sillion

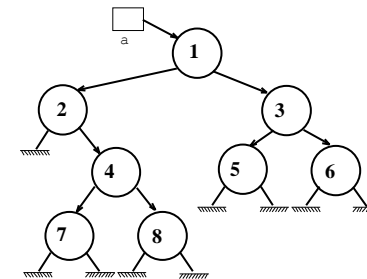
3

## Parcours en largeur d'abord

- Utilisation d'une file au lieu d'une pile

```
static void traverseNiveau( ArbreBinaire x )
{
    File f = CreeFile();
    Ajoute( f , x );

    while ( ! FileVide( f ) ) {
        x = Retire(f);
        visite( x );
        if ( x.gauche != null )
            Ajoute( f , x.gauche );
        if ( x.droite != null )
            Ajoute( f , x.droite );
    }
}
```



PC 7

François Sillion

4

## Arbres de recherche équilibrés

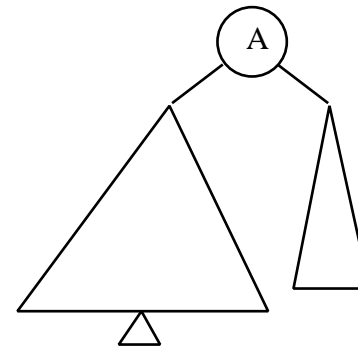
---

- Arbres AVL (existent aussi les arbres 2-3-4, rouge-noir...)
- Force la condition  
| hauteur ( arbre gauche ) - hauteur ( arbre droit ) | ≤ 1
- Modifie la fonction d'insertion
  - Garantit la condition d'équilibre ci-dessus
  - Renvoie la variation de hauteur de l'arbre

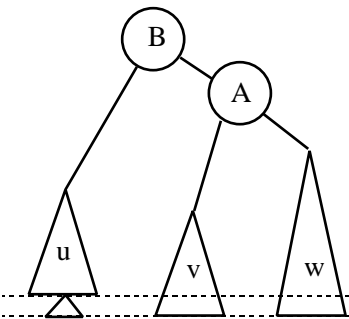
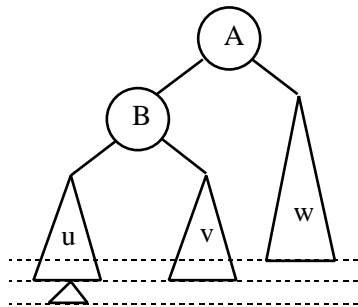
```
class AVLreturnValue {  
    int diffHauteur;  
    Arbre a;  
  
    AVLreturnValue( int h, Arbre b ) {  
        diffHauteur = h;  
        a = b;  
    }  
}  
static AVLreturnValue insere(Arbre t, int x)...
```

## Que faire en cas de déséquilibre ?

---



## Rotation d'un noeud

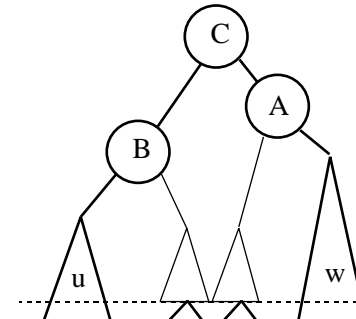
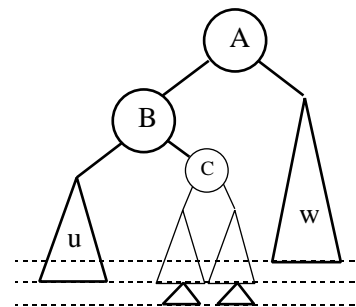


PC 7

François Sillion

7

## Rotation double



PC 7

François Sillion

8

## Insertion AVL

---

```
static AVLReturnValue insere( int v, ArbreBinaire a ){
    AVLReturnValue p;
    int incr, r = 0;
    if ( a == null ) {
        a = new Arbre( v, null, null );
        a.bal = 0;
        r = 1;
    } else {
        if ( v <= a.contenu ) {
            p = insere( v, a.gauche );
            incr = -p.diffHauteur;
            a.gauche = p.a;
        } else {
            p = insere( v, a.droite );
            incr = p.diffHauteur;
            a.droite = p.a;
        }
        // Equilibrage... (voir plus loin)
    }
    return new AVLReturnValue( r , a );
}
```

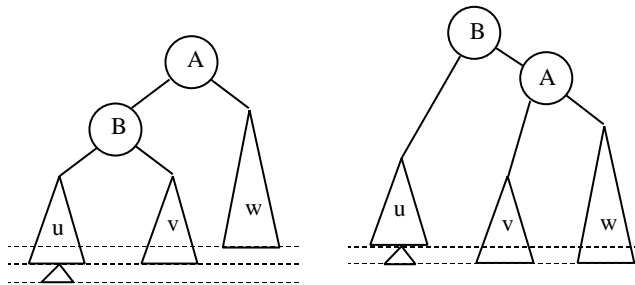
## équilibrage AVL

---

```
...
a.bal = a.bal + incr;
if ( incr != 0 && a.bal != 0 )
    if ( a.bal < -1 )
        // trop de poids à gauche
        if ( a.gauche.bal < 0 )
            a = rotationD( a );
        else {
            a.gauche = RotationG(a.gauche);
            a = rotationD(a);
        }
    else
        if ( a.bal > 1 )
            // trop de poids à droite
            if ( a.droite.bal > 0 )
                a = rotationG( a );
            else {
                a.droite = RotationD( a.droite );
                a = rotationG( a );
            }
        else
            r = 1;
```

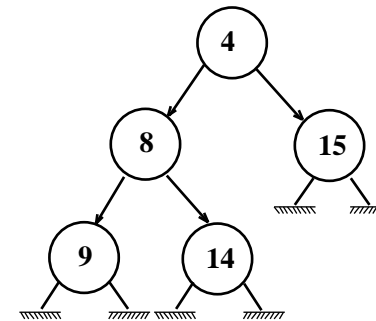
## rotation AVL

```
static Arbre rotationD( Arbre a ) {  
    Arbre b;  
  
    b = a;  
    a = a.gauche;  
    b.gauche = a.droite;  
    a.droite = b;  
  
    // mise à jour du champ bal...  
}
```



## Structure de tas (heap)

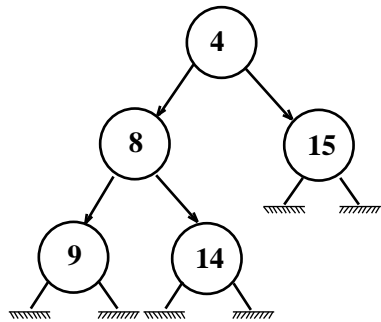
- Propriété de tas: la valeur d'un nœud est inférieure à la valeur de ses fils.
- Par construction, la racine contient la plus petite valeur
- Arbre quasi complet



## Arbre binaire et tableau

---

- Rangement dans le tableau "par couches"
- Les fils de  $t[k]$  sont en  $t[2k]$  et  $t[2k+1]$
- On peut ranger un tas dans un tableau de taille  $N$
- 4    8    15    9    14



## Utilisation du tas

---

- Extraction du minimum: temps constant  $O(1)$
- Suppression du minimum
  - Place le dernier élément du tableau en tête
  - Reconstitue la condition du tas en descendant
- Insertion
  - Place le nouvel élément en queue
  - Reconstitue condition du tas en remontant
- Complexité  $\log N$  dans tous les cas !
- Heapsort
  - Crée un tas
  - Extrait les éléments un par un