

TC Informatique

PC N° 6
14 Décembre 2000

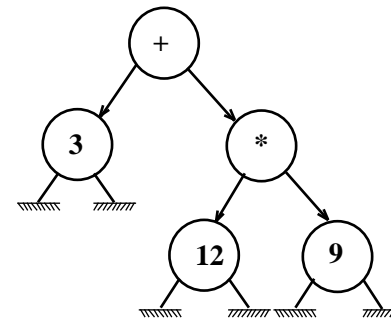
François Sillion

Arbres de recherche Tas

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

Structures arborescentes

- Organisation des données
- Recherche
- Traitement des données dans un ordre choisi (évaluation d'expressions...)



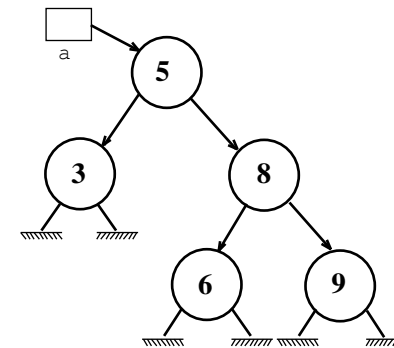
Arbre binaire

```
class Arbre {
    int val;
    Arbre gauche;
    Arbre droite;

    Arbre( int v ) {
        val = v;
        gauche = null;
        droite = null;
    }
    Arbre( int v, Arbre g, Arbre d ) {
        val = v;
        gauche = g;
        droite = d;
    }
    static boolean EstVide( Arbre a ) {
        return a == null;
    }
}
```

Construction de l'arbre

```
Arbre a = new Arbre( 5,
                    new Arbre( 3 ),
                    new Arbre( 8,
                                new Arbre( 6 ),
                                new Arbre( 9 ) ) );
```

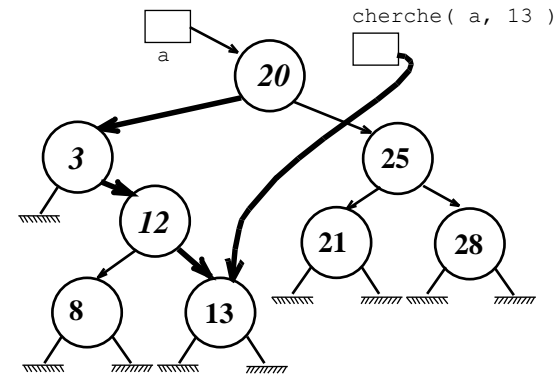


Arbre binaire de recherche

- Propriété fondamentale: toutes les clés inférieures à la clé courante sont dans le sous-arbre gauche
- Recherche dans l'arbre

```
static Arbre cherche( Arbre a, int x ){  
    if ( a == null ) {  
        return null;  
    } else if ( x < a.val ) {  
        return cherche( a.gauche, x );  
    } else if ( x > a.val ) {  
        return cherche( a.droite, x );  
    } else { // égalité x == a.val !  
        return a;  
    }  
}
```

Arbre binaire de recherche

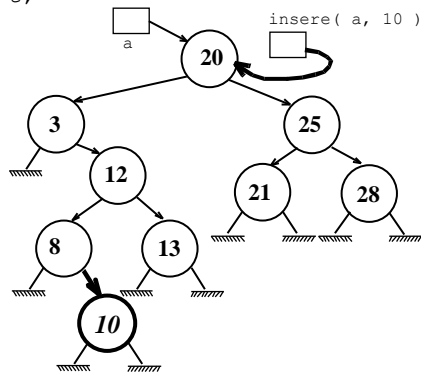


Insertion (avec effet de bord)

```

static Arbre insere(Arbre t, int x) {
    if (EstVide(t)) {
        return new Arbre(x);
    } else if (x < t.value) {
        t.left = insere(t.left, x);
        return t;
    } else {
        t.right = insere(t.right, x);
        return t;
    }
}

```



PC 6

François Sillion

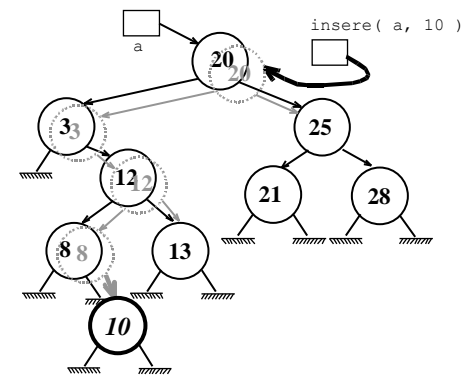
7

Insertion (sans effet de bord)

```

static Arbre insert(Arbre t, int x) {
    if (EstVide(t)) {
        return new Arbre(x);
    } else if (x < t.value) {
        return new Arbre(t.value,
            insere(t.left, x), t.right);
    } else {
        return new Arbre(t.value, t.left,
            insere(t.right, x));
    }
}

```



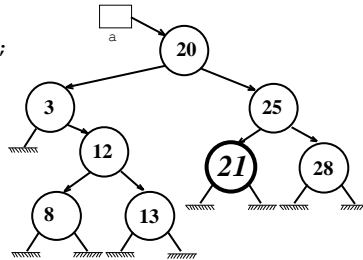
PC 6

François Sillion

8

Suppression de la racine

```
// L'arbre est supposé non-vide
static Arbre deleteRoot(Arbre t)
{
    if (EstVide(t.right) {
        return t.left;
    } else if (Estvide(t.left) {
        return t.right;
    } else if (EstVide(t.right.left) {
        t.right.left = t.left;
        return t.right;
    } else {
        Arbre r = t.right;
        Arbre l = r.left;
        while (!EstVide(l.left)) {
            r = l;
            l = r.left;
        }
        r.left = l.right;
        l.left = t.left;
        l.right = t.right;
        return l;
    }
}
```



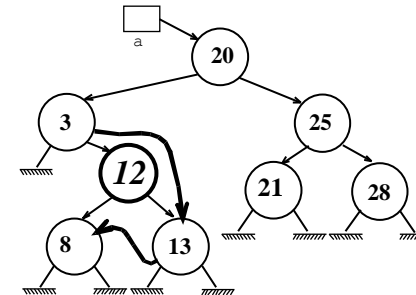
PC 6

François Sillion

9

Suppression d'une valeur

```
static Arbre delete(Arbre t, int x) {
    if (EstVide(t)) {
        return t;
    } else if (x == t.value) {
        return deleteRoot(t);
    } else if (x < t.value) {
        t.left = delete(t.left, x);
        return t;
    } else {
        t.right = delete(t.right, x);
        return t;
    }
}
```



PC 6

François Sillion

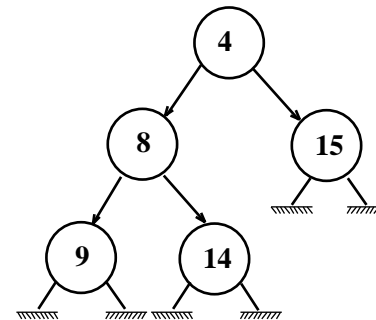
10

Arbres binaires de recherche

- Pas nécessairement équilibrés !
- Dans un arbre équilibré, insertion, recherche, suppression en $\log N$.

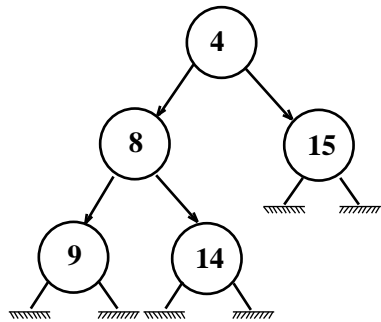
Structure de tas (heap)

- 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

Parcours d'arbres

- Dépend de l'ordre des appels récursifs
- Infixe, préfixe, postfixe
- Parcours en largeur d'abord avec une file