Algo

ABR :

Recherche :

```
Fonction  Recherche (A: ABR, x: entier ): ^nœud
Début
    si  (A ≠ nil)  alors
            si  (x= A^.val ) alors  Retourner ( A)
            sinon
              si  ( x < A^.val ) alors
                  Retourner (Recherche (A^.fg, x))
              sinon
                  Retourner (Recherche (A^.fd, x))
              finsi
          finsi
    sinon
            Retourner ( Nil )
    finsi
Fin
```

Insertion :

```
Procédure Insérer ( var A: ABR, x: entier )
Début
    si (A = nil) alors
            Allouer (A)
            A^.val ← x
            A^.fg ← Nil
            A^.fd ← Nil
    sinon
            si ( x ≤ A^.val ) alors
                    Insérer (A^.fg, x)
               sinon
                    Insérer (A^.fd, x)
               finsi
    finsi
Fin
```

Suppression du minimum :

```
1. Fonction Supp_min (var A: ABR): entier
   var     K: ^nœud
           Min: entier
   Début
    si (A^.fg = nil) alors
            Min ← A^.val
            K← A
            A ← A^.fd
            Libérer ( K)
             Retourner (Min)
     sinon
              Retourner (Supp_min (A^.fg))
     finsi
    Fin
```

Suppression :

```
Début
   si (A ≠ nil) alors
         si ( A^.val= x) alors
            k ← A
            si  ( A^.fg= nil) alors
              A ← A^.fd
              Libérér (K)
            sinon
              si  ( A^.fd= nil) alors
                 A ← A^.fg
                 Libérér (K)
             sinon
            A^.val ← Supp-min ( A^.fd) // ou Supp-max(A^.fg)
            finsi
         finsi
```

```
      sinon
        si ( x < A^.val ) alors
                     Supprimer (A^.fg, x)
        sinon
                     Supprimer (A^.fd, x)
        Finsi
      finsi
```

AVLs :

Rotation droite :

Aux = a.fg
A.fg = aux.fd
Aux.fd = a

A = aux

bal(a.fd) = 1 - bal(a)
bal(a) -= 1

Rotation Gauche :

Remplacer fd par fg et vice versa

Bal(a.fg) = -1 - bal (a)
bal (a) += 1

Rotation Gauche Droite :

Rotation gauche sur fils gauche de A
Rotation droite sur A

Mise à jour des balances :
bal(a.fg) = bal (a.fd= = 0
Case bal(a) :
    -1 : bal(a.fg) = 1
    1 : bal(a.fd) = -1
bal(a) = 0

Insertion

Fonction qui retourne un entier
Si A = nill
    Création d'un nouveau noeud
    Allouer(A)....
    Retourner 1

Sinon
    Si x <= A.val
        Si insert-val(A.fg,x) = 0
            Retourner 0
        Sinon
            A.bal ++
            Si a.bal = 2
                Si a.fg.bal = -1
                    RGD(A)
                Sinon
                    RD(A)
        Retourner a.bal
    Sinon
        Meme algorithme précedemnt en remplacant gauche avec droite

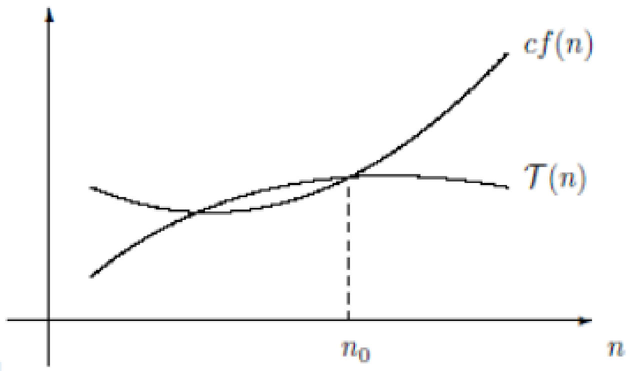
Fin

Est AVL

Fonction Est_AVL (A : AVL) : Booleén
var
Début
Si A = Nil alors Retourner (vrai)
Sinon
Si |A^.bal| > 1 alors Retourner (Faux)
Sinon
Retourner (Est_AVL (A^.fg) et Est_AVL (A^.fd))
fsi
fsi
Fin

‘Complexité :

$\mathcal{O}$ (f(n)) est l’ensemble des fonctions T (n) qui peuvent être bornées supérieurement par **c f(n)** pour **n** suffisamment grand.

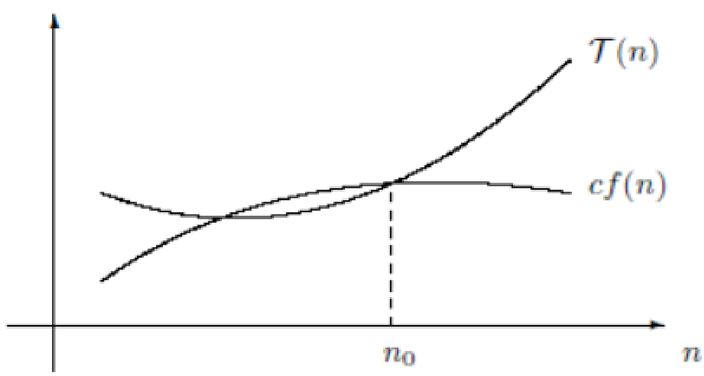


La notation « Omega » permet de trouver un minorant.

On a T (n) $\in \Omega$(f(n)), si et seulement s’il $\exists$ deux constantes n_0 et c > 0 telles que :

$$\forall n \geq n_0, \text{ on a } T(n) \geq cf(n)$$

Ω(f(n)) est l’ensemble de toutes les fonctions T (n) bornées inférieurement par cf(n) pour n suffisamment grand.

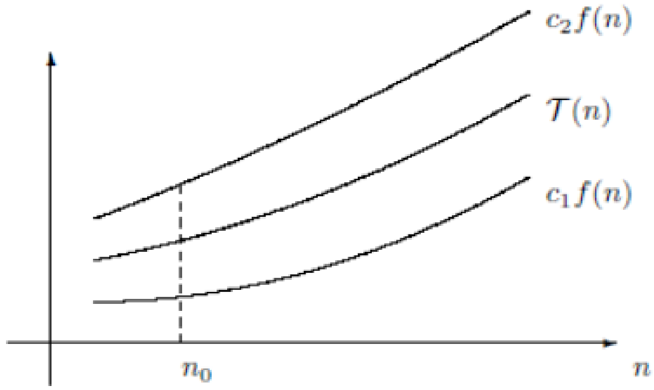


Si une fonction peut être majorée et minorée par une même fonction en notation asymptotique, alors on parle d'ordre exacte (notation « Théta »).

Une fonction T (n) $\in \Theta(f(n))$ si et seulement si elle vérifie à la fois
T (n) $\in \mathcal{O}(f(n))$ et T (n) $\in \Omega(f(n))$.

C'est-à-dire il $\exists$ deux constantes c_1 et c_2 telles que :

$$\forall\ n \geq n_0,\ \text{on a}\ c_1 f(n) \leq T(n) \leq c_2 f(n)$$



$$\lim_{n\to+\infty} \frac{g(n)}{f(n)} = \begin{cases} 0 & \text{alors } g(n) \in \mathcal{O}(f(n)) \\ c > 0 & \text{alors } g(n) \in \Theta(f(n)) \\ +\infty & \text{alors } g(n) \in \Omega(f(n)) \end{cases}$$

Complexité récursive

II. 3 Résolution des récurrences

Théorème:

Soient $a \geq 1$ et $b > 1$ deux constantes.

Soit $f(n)$ et $T(n)$ deux fonctions telles que

$$T(n) = aT(\frac{n}{b}) + f(n)$$

T(n) peut être bornée asymptotiquement comme suit :

— Si $f(n) = \mathcal{O}(n^{(\log_b a)-\epsilon})$ pour une constante $\epsilon > 0$, alors $T(n) = \Theta(n^{\log_b a})$;

— Si $f(n) = \Theta(n^{\log_b a})$, alors $T(n) = \Theta(n^{\log_b a} \log n)$;

— Si $f(n) = \Omega(n^{(\log_b a)+\epsilon})$ pour une constante $\epsilon > 0$, et si $a.f(\frac{n}{b}) \leq c.f(n)$ pour une constante $c < 1$ et n suffisamment grand, alors $T(n) = \Theta(f(n))$;