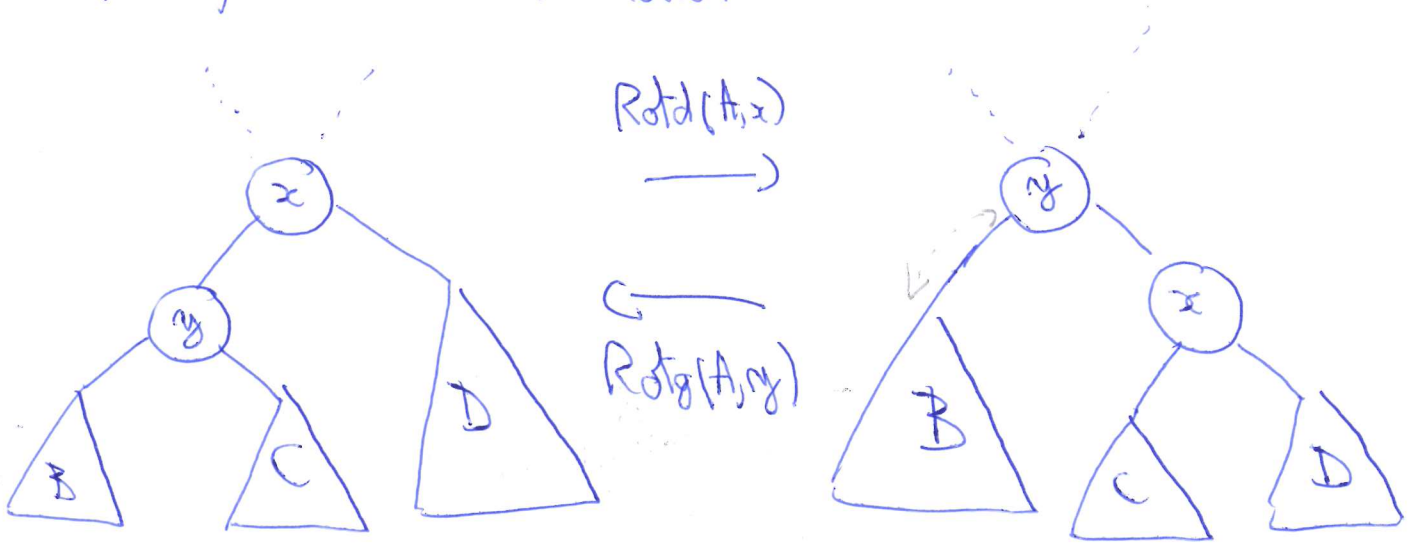
NF16 - ABR, Rotations - Exercice 1 - 1

208 A527



Rotd(A,x) définie si
- A non vide
- SAG(A,x) non vide

Rotg(A,y) définie si
- A non vide
- SAD(A,y) non vide

A) - $B = SAG(A,y)$ donc $de[z] < de[y]\ \forall z \in B$ $y \hookleftarrow B$

- $y \in SAG(A,x)$ donc $de[y] < de[x]$ $y \hookleftarrow x$

- ~~$C \in SAG(A,x)$~~

- $C = SAD(A,y)$ donc $de[z] > de[y]\ \forall z \in C$ $y \hookleftarrow C$

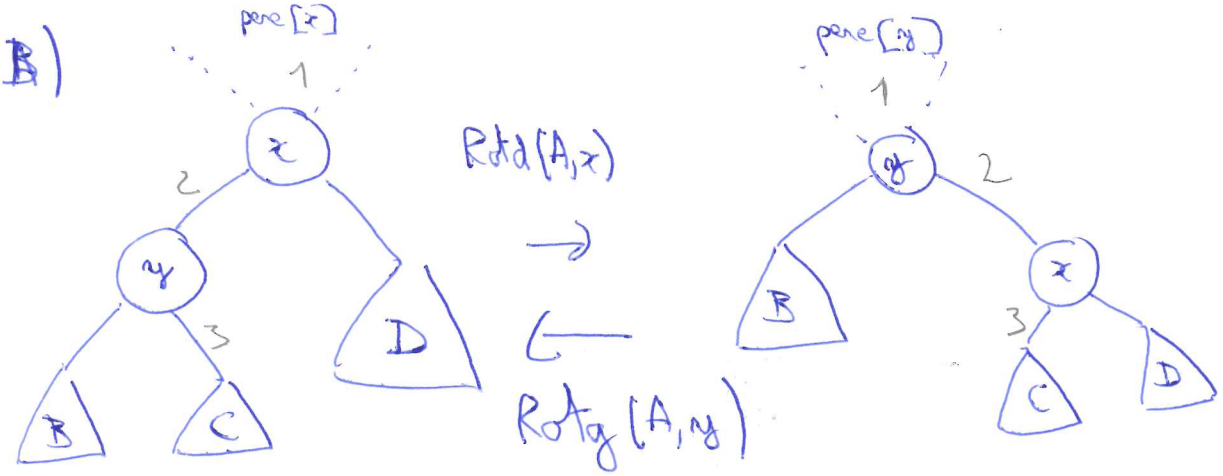
- $C \in SAG(A,x)$ donc $de[z] > de[x]\ \forall z \in C$ $x \hookleftarrow C$

- $D \in SAD(A,x)$ et $y \in SAG(A,x)$ donc $de[y] < de[x] < de[z]$ $\forall z \in D$. $y \hookleftarrow D$

- $D = SAD(A,x)$ donc $de[x] < de[z]\ \forall z \in D$ $x \hookleftarrow D$

NF16 - ABR, Rotations - Exercice 1-2

B)



Algorithme Rotd(A,x)

```
Si gauche[x] ≠ NULL
    y := gauche[x]
    c := droit[y]
    père[y] := père[x]                    // Relie y à l'ancien père de x
    Si père[x] ≠ NULL                                                    1
        Si gauche[père[x]] = x
            gauche[père[x]] := y
        Sinon
            droit[père[x]] := y
    droite[y] := x                        // Relie y à x                2
    père[x] := y
    gauche[x] := c                        // Fils gauche de x devient
    Si c ≠ NULL                                                          3
        père[c] := x
```

Algorithme Rotg(A,y)

```
Si droit[y] ≠ NULL
    x := droit[y]
    c := gauche[x]
    père[x] := père[y]                                                   1
    Si père[y] ≠ NULL
        Si gauche[père[y]] = y
            gauche[père[y]] := x
        Sinon
            droit[père[y]] := x
    gauche[x] := y                                                       2
    père[y] := x
    droit[y] := c                                                        3
    Si c ≠ NULL
        père[c] := y
```

NF16 - ABR, Rotations - Exercice 1 - 3

c)

$h(B) = -1$ si l'arbre B est vide

$h = 1 + \max(1 + h(B), 1 + h(C), h(D))$

$h' = 1 + \max(h(B), 1 + h(C), 1 + h(D))$

Montrons que $\Delta = -1 \Leftrightarrow h(B) > \max(h(D), h(C))$

Si $h(B) > \max(h(D), h(C))$:

$h = 1 + (h(B) + 1)$
$h' = 1 + h(B)$
$\Big\} \Delta = -1$ OK

Si $h(D) > \max(h(B), h(C))$:

$h = 1 + h(D)$
$h' = 1 + (1 + h(D))$
$\Big\} \Delta = 1$ OK

Si $h(B) \leq \max(h(D), h(C))$ et $h(D) \leq \max(h(B), h(C))$:

$h = 2 + \max(h(B), h(C))$
$h' = 2 + \max(h(C), h(D))$
$\Big\} \Delta = \max(h(C), h(D)) - \max(h(B), h(C))$

* Si $h(C) \geq h(D)$: $\Delta = h(C) - \max(h(B), h(C))$

Or $h(B) \leq \max(h(D), h(C))$ donc $h(B) \leq h(C)$

$\Delta = h(C) - h(C) = 0$

* Si $h(C) \geq h(B)$ $\Delta = \max(h(C), h(D)) - h(C)$

or $h(D) \leq \max(h(B), h(C))$ donc $h(D) \leq h(C)$ $\Delta = h(C) - h(C) = 0$

Si $h(C) < h(D)$ et $h(C) < h(B)$

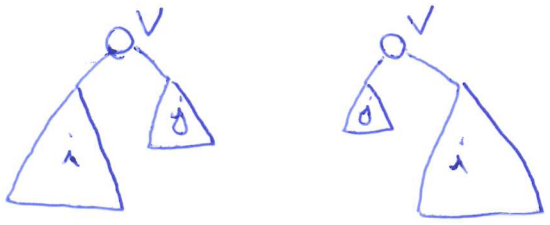
$\Delta = h(D) - h(B)$

$h(B) \leq \max(h(D), h(C))$ donc $h(B) \leq h(D)$
$h(D) \leq \max(h(B), h(C))$ donc $h(D) \leq h(B)$
$\Big\} h(B) = h(D)$ et $\Delta = 0$

NF16 - ABR, Rotations - Exercice 1 - 4

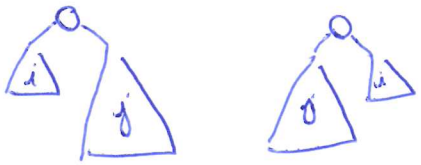
D) On note i le sous-arbre modifié et j le sous-arbre non modifié

Si $h(i) > h(j)$:



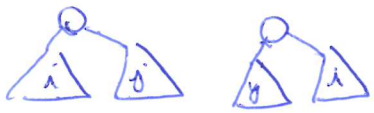
Si $h(i)$ ↗ 1, $h(v)$ ↗ 1
Si $h(i)$ ↘ 1, $h(v)$ ↘ 1

Si $h(i) < h(j)$



Si $h(i)$ ↗ 1, $h(v)$ inchangé
Si $h(i)$ ↘ 1, $h(v)$ inchangé

Si $h(i) = h(j)$



Si $h(i)$ ↗ 1, $h(v)$ ↗ 1
Si $h(i)$ ↘ 1, $h(v)$ inchangé

En notant Δ la variation : $\begin{cases} h'(v) = h(v) + \Delta \text{ si } h(i) > h(j) \text{ ou } \{h(i) = h(j) \text{ et } \Delta = 1 \\ h'(v) = h(v) \text{ sinon} \end{cases}$

E) D'après C) : La variation de hauteur du sous-arbre modifié est 0, -1 ou 1.

D'après D) : On sait ~~dans ce~~ calculer la hauteur du père en fonction de la hauteur de i et de j.

VF16 - ABR, Rotations - Exercice 1 - 5

Algorithme RotD (A, x) (Suite)

```
hB := h[x]
Si gauche[y] ≠ NULL
| hB := h[gauche[y]]
Sinon
| hB := -1
Si C ≠ NULL
| hC := h[C]
Sinon
| hC := -1
Si droite[x] ≠ NULL
| hD := h[droite[x]]
Sinon
| hD := -1
```

Init : Recupère les hauteurs de B, C et D

```
h[x] := 1 + max(hC, hD)
h[y] := 1 + max(h[x], hB)
```

Maj h(x) et h(y)

```
Δ := h[y] - h        } Calcul Δ
i := y
p := pere[y]
Tantque (Δ ≠ 0 et p ≠ NULL)
    Si gauche[p] = i
    | j := droite[p]
    Sinon
    | j := gauche[p]
    Si j ≠ NULL
    | hj := h[j]
    Sinon
    | hj := -1
    Si (h[i] > hj ou (h[i] = hj et Δ = 1))
    |   h[p] := h[p] + Δ
    |   i := p
    |   p := pere[p]
    Sinon
    | Δ := ∅
```

Correction du chemin entre y et la racine de l'arbre

Recherche hauteur du fils non modifié

$O(h)$

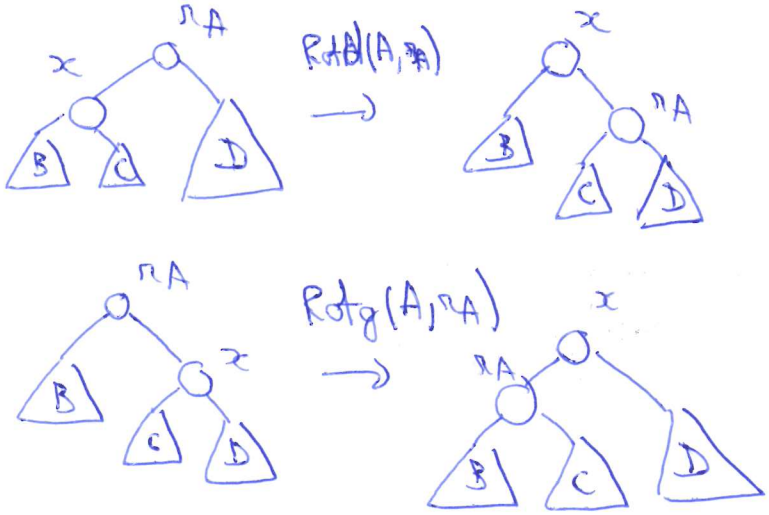
Formule question D

Maj Δ du père

Condition d'arrêt

NF16-ABR, Rotations - Exercice 2 - 1

A)



B)

```
enracine (A:ABR; e:entier)
    Si (de [rA] = e)
        retourner rA
    Si (de [rA] > e)
        enracine (gauche[rA], e)   // Remonte e jusqu'au fils gauche de rA
        retourne rotd (A, rA)      // Rotd pour enracine le fils gauche de rA ↪ e
    Si (de [rA] < e)
        enracine (droit[rA], e)    // Remonte e jusqu'au fils droit de rA
        retourne rotg (A, rA)      // Rotg pour enracine le fils droit de rA ↪ e
```

VF16 - ABR, Rotations - Exercice 2-2

Montrons que enraciner(A,e) termine en au plus k appels récursifs et renvoie un ABR de racine e pour tout arbre A de hauteur k contenant e.

Initialisation : Pour un arbre A de hauteur 0 contenant e enracinerr(A,e) termine directement sans modifier l'arbre qui contient e à sa racine.

Hérédité : Soit A un arbre de hauteur $k+1$ contenant e.

Si $\text{de}[r_A] = e$: L'algorithme se termine et est valide

Si $\text{de}[r_A] < e$: L'arbre A contient e donc $\underset{\text{SAD}}{SAG}(A, r_A)$ contient e. $SAG(A, r_A)$ est un arbre de hauteur k contenant e. Par hypothèse de récurrence, l'appel à enracina$(\underset{\text{droite}}{\text{gauche}}[r_A], e]$ se termine en k appels au plus et renvoie un ABR de racine e.

↪ $k+1$ appels au plus à enracinerr sont realisées

↪ $\underset{\text{droite}}{\text{gauche}}[r_A]$ a pour de e.

La rotation $\underset{\text{droite}}{\underset{\text{gauche}}{}}$ est en $O(1)$ et permet d'enracina $\underset{\text{droite}}{\text{gauche}}[r_A]$, c'est à dire e.

Conclusion : L'algorithme se termine et renvoie un ABR de racine e → validité

k appels à enracinerr au plus sont realisés pour un arbre de hauteur k.

Les autres opérations (dont la rotation) sont en $O(1)$

→ Complexité $O(h)$