2022-2023, Semestre d'automne
L3, Licence Sciences et Technologies
Université Lyon 1

LIFAPC: Algorithmique, Programmation et Complexité

Chaine Raphaëlle (responsable semestre automne)
E-mail : raphaelle.chaine@liris.cnrs.fr
http://liris.cnrs.fr/membres?idn=rchaine

AVL

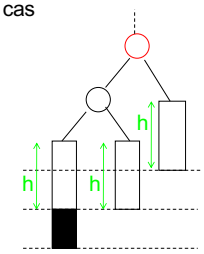
- Du nom des auteurs de la méthode : Adelson-Velskij et Landis
- Garantir après chaque opération que
  - pour chaque nœud, les hauteurs des sous-arbres gauche et droit diffèrent au plus de 1
- Cet équilibre peut-être maintenu à travers l'utilisation judicieuse de 4 opérations :
  - Rotation gauche
  - Rotation droite
  - Rotation droite-gauche (ou rotation double à gauche)
  - Rotation gauche-droite (ou rotation double à droite)

Rééquilibrage après une insertion

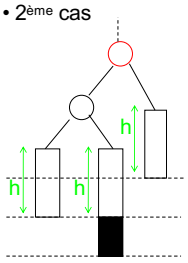
- Si il apparait un nœud ◯ au niveau duquel le déséquilibre devient égal à 2 (mais dont les fils restent équilibrés)

■ Élément inséré

- 1er cas



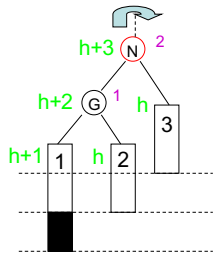
- 2ème cas

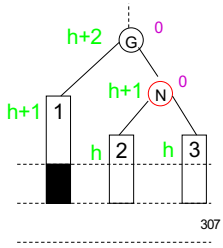


Rotation simple à droite

- A faire dans le 1er cas d'un déséquilibre « à gauche toute »

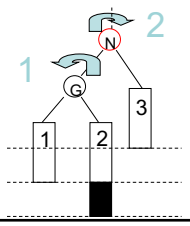
Hauteurs h1=h+1 h2=h3=h





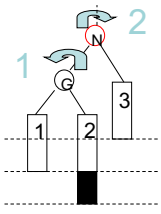
Rotation double à droite

- A faire dans le 2ème cas : déséquilibre « à droite de la gauche»
- La rotation double à droite commence par une rotation gauche sur le sous arbre G **puis** une rotation droite sur N



Rotation double à droite

- Mise à jour des déséquilibres des nœuds : Sous-cas à considérer lorsqu'on fait une rotation double à droite :

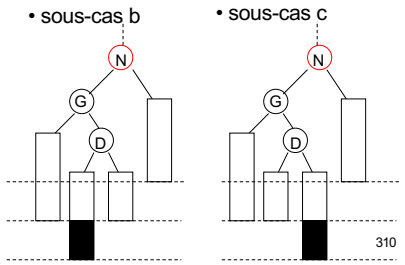


a) Sous-cas où les sous-arbres 1, 2 et 3 sont vides avant l'insertion (h=0)
b) Sous cas où le sous-arbre 2 est déséquilibré sur sa gauche
c) Sous cas où le sous-arbre 2 est déséquilibré sur sa droite

Rotation double à droite

- A faire dans le 2ème cas d'un déséquilibre « à droite de la gauche» de ◯

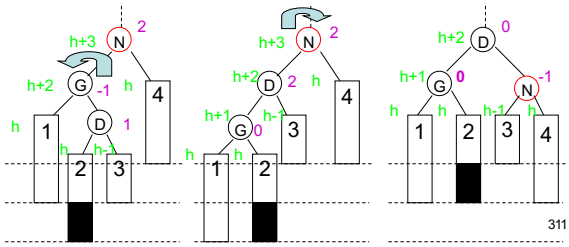
■ Élément inséré



310

Rotation double à droite

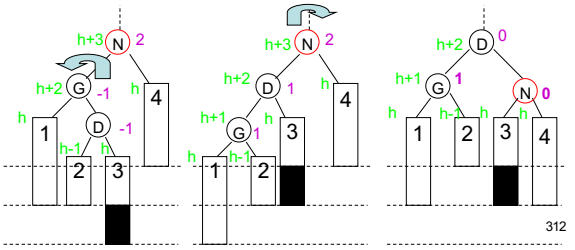
- Cas 2b : déséquilibre à gauche de la droite dans le sous-arbre gauche
  Hauteurs h1=h2=h4=h et h3=h-1
- Combinaison d'une rotation gauche puis d'une rotation droite



311

Rotation double à droite

- Cas 2c : déséquilibre à droite de la droite du sous-arbre gauche
  Hauteurs h1=h3=h4=h h2=h-1
- Combinaison d'une rotation gauche puis d'une rotation droite



312

- Les symétriques de ces opérations existent dans le cas d'un déséquilibre alourdissant un sous-arbre droit après une insertion
  - Rotation gauche
  - Rotation droite-gauche
- Dans les 4 cas (et leurs symétriques) la hauteur du sous-arbre de racine n retrouve sa hauteur d'avant l'insertion de l'élément ■

313

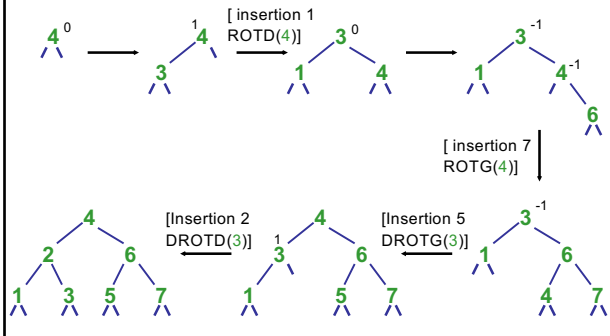
313

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

314

314

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



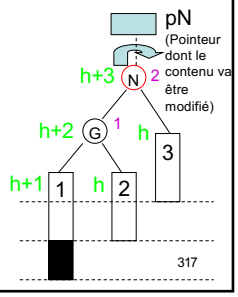
315

- Comment pister / stocker les déséquilibres?
  - 1ère solution : Conserver l'info de hauteur dans chaque nœud
  - 2ème solution : Ne conserver dans chaque nœud que la **différence** de hauteur entre le sous-arbre gauche et le sous-arbre droit

316

Rotation droite

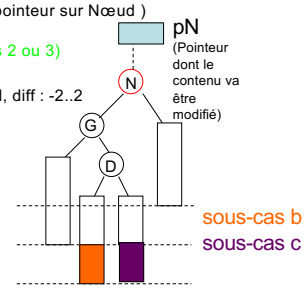
```
procédure AVL_Rotation_Droite
   (donnée-résultat a : ABR, pN : pointeur sur Nœud)
   Précondition : pN->diff=2 et pn->gauche->diff=1 (cas 1)
   variables :
      pG : pointeur sur Nœud
   début
      pG ← pN->gauche
      pN->gauche ← pG->droite
      pG->droite ← pN
      pN ← pG
      pN->diff ← 0; pn->droite->diff ← 0
   fin
```



317

Rotation double à droite

```
procédure AVL_Rotation_Double_Droite
   ( donnée-résultat a : ABR, pN : pointeur sur Nœud )
   Précondition : pN->diff=2 et
         pn->gauche->diff =-1 (cas 2 ou 3)
   variables :
      pG, pGD : pointeur sur Nœud, diff : -2..2
   début
      pG ← pN->gauche
      pGD ← pN->gauche->droite
      pG->droite ← pGD->gauche
      pGD->gauche ← pG
      pN->gauche ← pGD->droite
      pGD->droite ← pN
      diff ← pGD->diff ;
      pN ← pGD
      pN->gauche->diff ← max(0,-diff) ; pN->droite->diff ← min(0,-diff);
      pN->diff ← 0
   fin
```



318

```
procédure équilibrer( donnée-résultat a : ABR, pN : pointeur sur Nœud )
   Précondition : pN->diff=2 ou -2
   début
      si pN->diff=2 alors
         si pN->gauche->diff=1 alors
            AVL_Rotation_Droite(a,pN)
         sinon
            AVL_Rotation_Double_Droite(a,pN)
         finsi
      sinon
         si pN->droite->diff=-1 alors
            AVL_Rotation_Gauche(a,pN)
         sinon
            AVL_Rotation_Double_Gauche(a,pN)
         finsi
      finsi
   fin
```


319

Insertion dans un AVL

- Insertion aux feuilles : parcours d'un chemin menant de la racine au nouvel élément
- Parcours du chemin en sens inverse,
  - pour mettre à jour les différences de hauteur des nœuds ancêtres,
  - et effectuer une opération de rééquilibrage si nécessaire!
- Parcours du chemin en sens inverse :
  - facile si la procédure d'insertion est récursive, (il suffit de mettre les instructions relatives à la remontée après l'appel récursif)
  - la remontée peut s'arrêter à partir du moment où le champ diff d'un nœud n'est plus modifié
  - En itératif, on peut mémoriser l'adresse des Nœuds rencontrés à la descente, par exemple dans une pile … ou bien on peut redescendre une seconde fois!

320

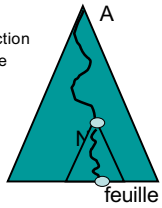
Complexité

- Temps d'une rotation : constant
- Note : Insertion exécute AU PLUS UNE rotation car une rotation sur un nœud rétablit la hauteur initiale (avant insertion) du sous-arbre correspondant
- *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

321

- Version itérative
  - C'est uniquement sur le chemin racine / nouvelle feuille qu'il peut y avoir des rotations
  - On considère sur ce chemin le nœud N le plus bas (le dernier rencontré à la descente) dont le champ *diff != 0*
  - On peut démontrer que si rotation il doit y avoir après l'insertion ca sera sur ce Noeud

Entre A et N l'adjonction n'a pas pu causer de problème.



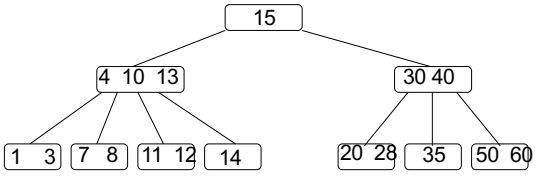
N= nœud le + bas t. que N->diff != 0

322

Arbres 234

- Arbre de recherche dont les nœuds contiennent 1, 2 ou 3 éléments triés et dont toutes les feuilles sont à la même hauteur
- Un nœud contenant $x_1<\ldots<x_{k-1}$ a k sous-arbres tels que :
  - les éléments du 1er sous-arbre sont $\leq x_1$,
  - tous les éléments du $i^{ème}$ sous-arbre sont $> x_{i-1}$ et $\leq x_i$,
  - tous les éléments du $k^{ième}$ sous-arbre sont $> x_{k-1}$

323



10 noeuds, 18 éléments

324

Recherche dans un (sous)arbre-2.3.4

- Même principe que dans un (sous) arbre binaire de recherche
- On compare x avec les éléments contenus dans la racine de A
  - s'il existe j tel que $x=x_j$, alors x est trouvé
  - si $x<x_1$ recherche dans le premier sous-arbre de A
  - si $x_j<x<x_{j+1}$ recherche dans le (j+1) ième sous-arbre de A
  - si $x>\max(x_i)$, recherche dans le dernier sous-arbre

325

Recherche dans un (sous)arbre-2.3.4

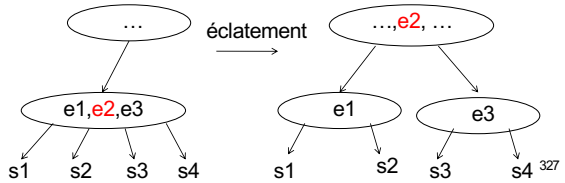
- si la recherche se termine sur une feuille qui ne contient pas x, alors x n'appartient pas à A

- Recherche en O(lg n) puisque l'arbre est équilibré par construction

326

Insertion dans un (sous) arbre 234

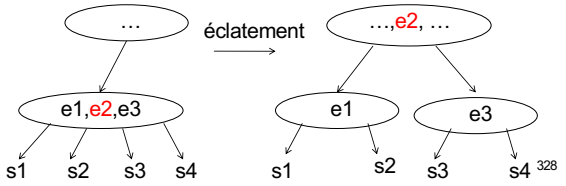
- Pour insérer x
  - On descend **jusqu'à la feuille** où x doit s'insérer (même approche que pour la recherche)
  - Si la feuille est pleine on commence par l'éclater….



327

Insertion dans un (sous) arbre 234

- Lors d'un éclatement, l'élément du milieu remonte dans le père
- Si le père est plein, on l'éclate à son tour…



328

328

Insertion dans un (sous) arbre 234

- 2 stratégies possibles
  - Éclatement en remontée (seulement quand il devient indispensable)
  - Éclatement (préventif) à la descente des nœuds pleins

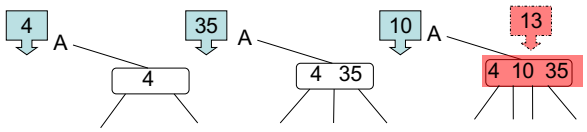
329

329

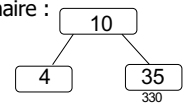
Exemple

Insertion avec éclatement en remontée

Insertion successive de 4, 35, 10, 13, 3, 30,15,12, 7, 40, 20, 11, 6, à partir de l'arbre vide

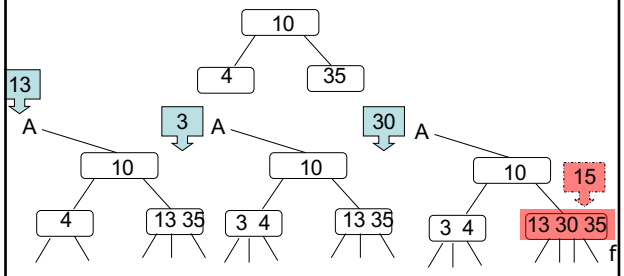


A cette étape, A est équivalent à l'arbre binaire :



330

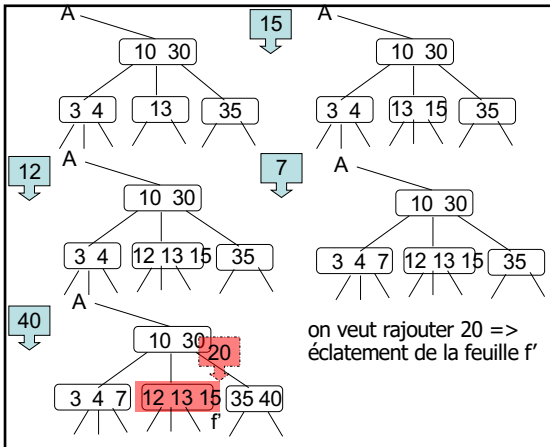
330



Pour ajouter 15, éclatement de la feuille f en 2 sous-nœuds contenant respectivement le plus petit et le plus grand élément de f ; déménagement de l'élément médian dans le père de f

331

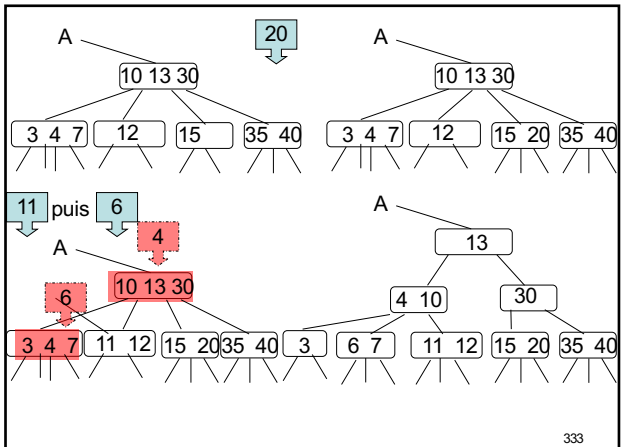
331



on veut rajouter 20 => éclatement de la feuille f'

332

332



333

333

Insertion avec éclatement à la descente

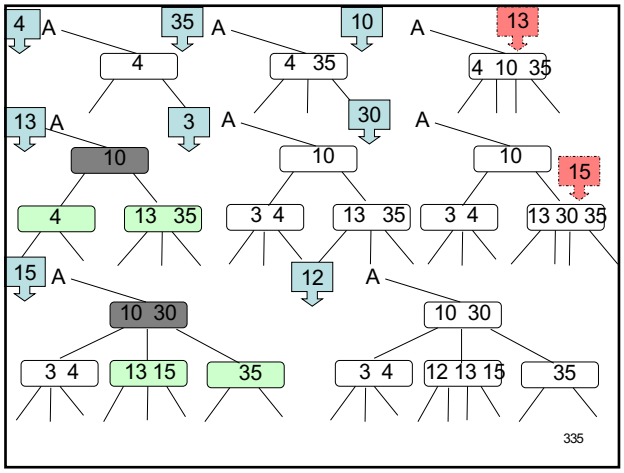
Avec la méthode précédente, éclatements en cascade possibles (éventuellement sur toute la hauteur de l'arbre si le chemin suivi n'est formé que nœuds pleins)

Pour éviter ce phénomène, travail sur des arbres-2.3.4 ne contenant jamais deux nœuds pleins à la suite :
-> pas de cascade d'éclatements lors d'une insertion.

Réalisation de cette condition par éclatement à la descente : pour insérer un élément, on parcourt un chemin dans l'arbre à partir de la racine et on fait éclater les nœuds pleins au fur et à mesure de leur rencontre.

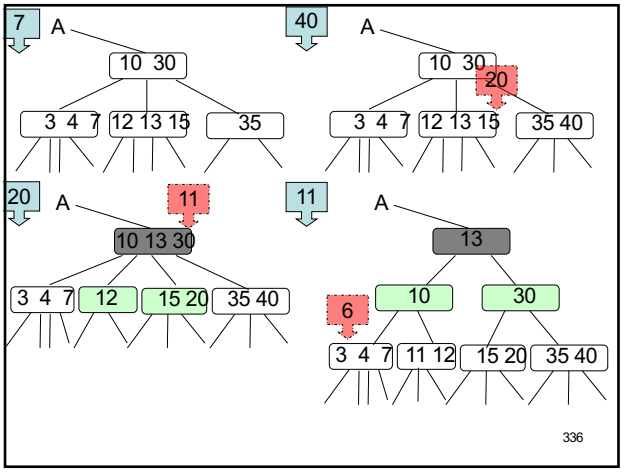
334

334



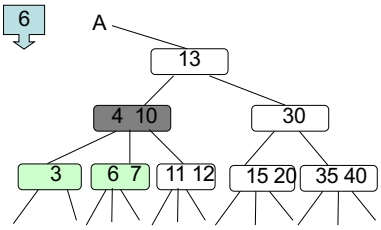
335

335



336

336



Les transformations de rééquilibrage à la descente sont purement locales : lorsqu'un nœud éclate, son père ne peut être un nœud plein car sinon on l'aurait fait éclater avant ; il n'y a donc que 2 cas possibles, suivant que le père contient 1 ou 2 éléments

337

337

Ces transformations n'augmentent pas la hauteur de l'arbre, sauf si on éclate la racine ; dans ce cas la hauteur augmente de 1

Analyse de la complexité

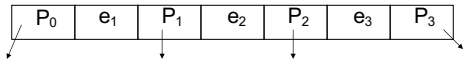
L'insertion est également en $\Theta(\log n)$: dans le cas d'un éclatement à la montée ou à la descente, on opère toujours sur un chemin de la racine à une feuille de l'arbre

338

338

Représentation des arbres-2.3.4

Pour représenter les différents types de nœuds ; on peut choisir une représentation « maximale » :



On peut également opter pour une implantation équivalente : les arbres binaires Rouge / Noir

339

339