Structures de données avancées :

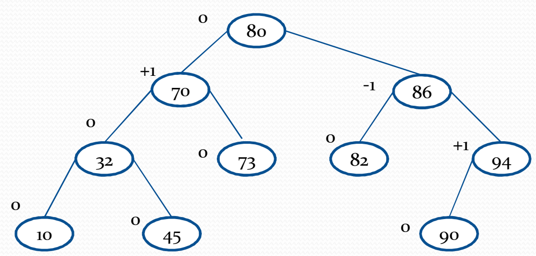
Arbres AVL & Arbres Rouge et Noir

Pr ZEGOUR DJAMEL EDDINE
Ecole Supérieure d'Informatique (ESI)
http://zegour.esi.dz
email: d_zegour@esi.dz

Les arbres AVL

Arbres AVL

Un *arbre AVL* **est un** *arbre de recherche binaire équilibré*

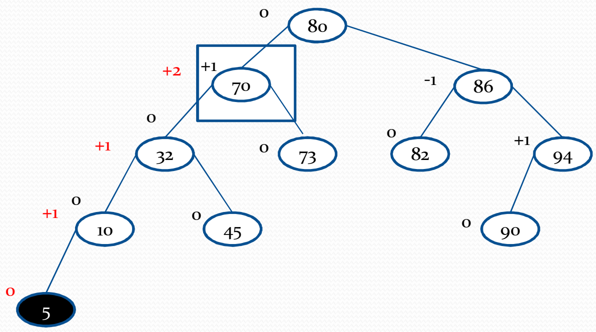


| Profondeur(fg(n))– Profondeur(fd(n)) | <= 1

Ajouter un champ balance (facteur d'équilibrage) au niveau de chaque noeud

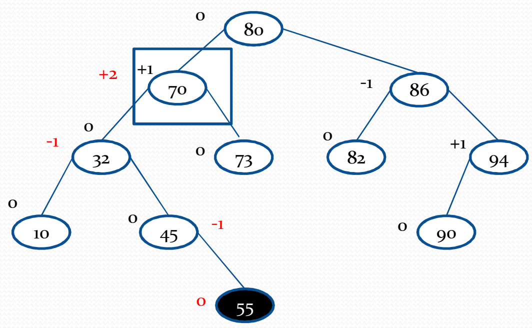
Les arbres AVL

Arbres AVL (*Cas de déséquilibre*)



Les arbres AVL

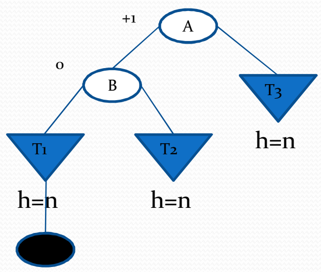
Arbres AVL (*Cas de déséquilibre*)



Les arbres AVL

Arbres AVL (*Techniques d'équilibrage*)

Examinons un sous arbre de racine le plus jeune antécédent qui devient non équilibré suite à une insertion



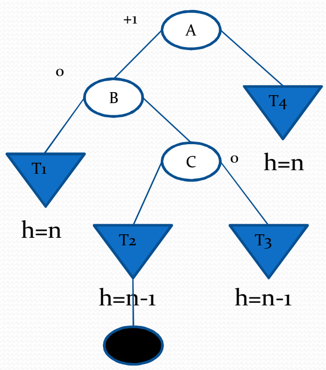
Cas où le facteur d'équilibrage est +1

Le nouveau nœud est inséré dans le sous arbre gauche de B. Donc f(B) devient 1 et f(A) devient 2

Les arbres AVL

Arbres AVL (*Techniques d'équilibrage*)

Examinons un sous arbre de racine le plus jeune antécédent qui devient non équilibré suite à une insertion



Cas où le facteur d'équilibrage est +1

Le nouveau nœud est inséré dans le sous arbre droit de B. f(B) devient -1 et f(A) devient 2.

Les arbres AVL

Arbres AVL (*Techniques d'équilibrage*)

Transformer l'arbre de telle sorte que

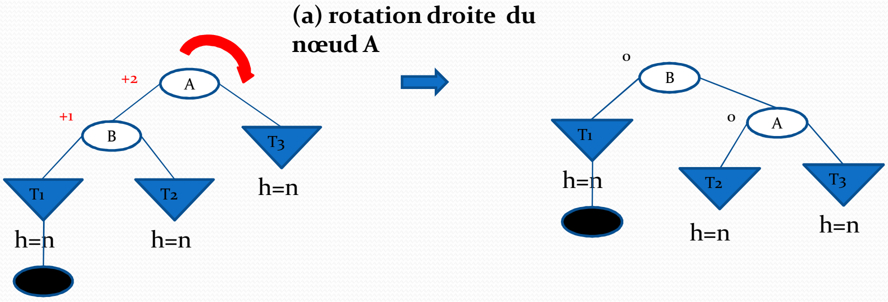
l'inordre soit préservé

l'arbre transformé soit équilibré

Les arbres AVL

Arbres AVL (*Techniques d'équilibrage*)

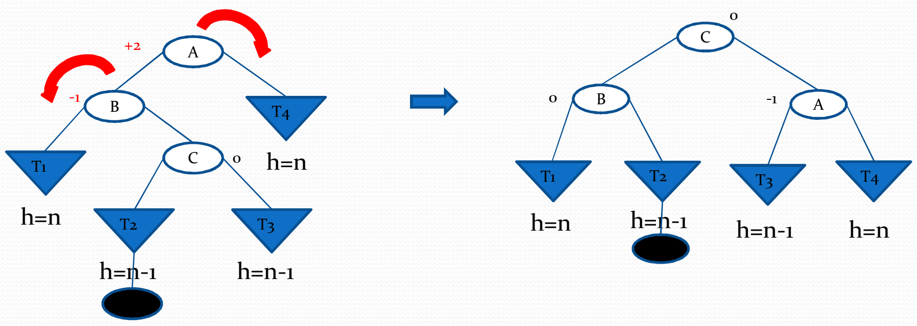
(a) rotation droite du nœud A



Les arbres AVL

Arbres AVL (*Techniques d'équilibrage)*

(b) rotation gauche du nœud B suivie par une rotation droite du nœud A



Les arbres AVL

Arbres AVL (*Algorithme d'insertion*)

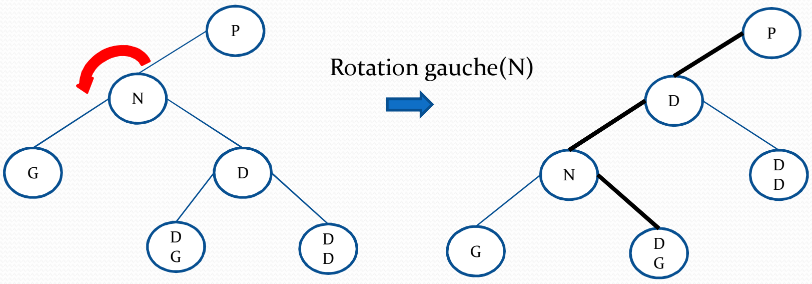
La première partie de l'algorithme consiste à insérer la clé dans l'arbre sans tenir compte du facteur d'équilibrage

Elle garde aussi la trace du plus jeune antécédent, soit Y qui devient non équilibré

La deuxième partie fait la transformation à partir de Y

Les arbres AVL

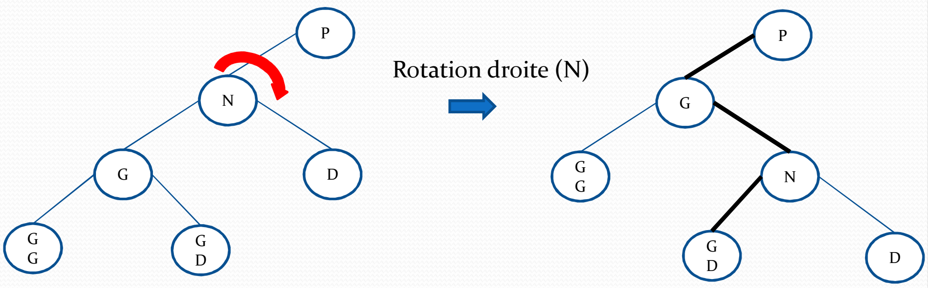
Arbres AVL (*Rotation gauche*)



AFF_FD(N, DG)   AFF_FG(D, N))   AFF_FG(Parent, D)

Les arbres AVL

Arbres AVL (*Rotation droite*)



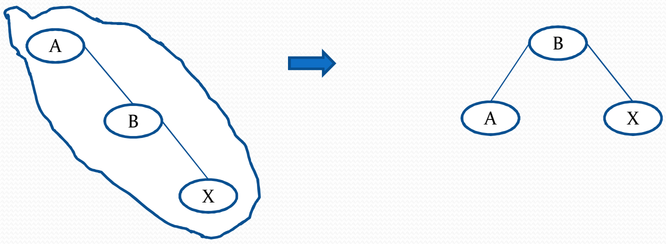
AFF_FG(N, GD)   AFF_FD(G, N))   AFF_FD(Parent, G)

Les arbres AVL

Arbres AVL (*Exemples*)

Insérons la séquence : A, B, X, L, M, C, D, E, H, R, F **dans un arbre AVL.**

Insertion A, B, X

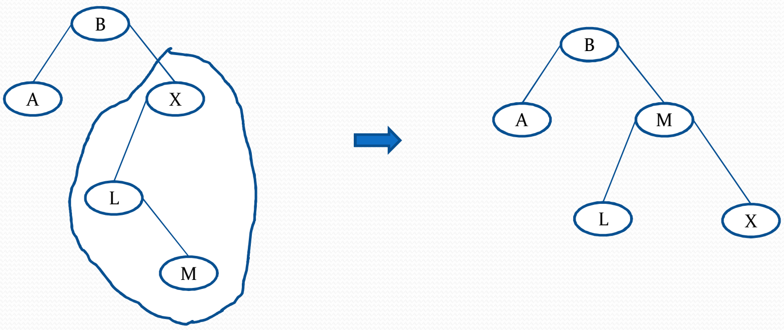


Les arbres AVL

Arbres AVL (*Exemples*)

Insérons la séquence : A, B, X, L, M, C, D, E, H, R, F **dans un arbre AVL.**

Insertion L, M

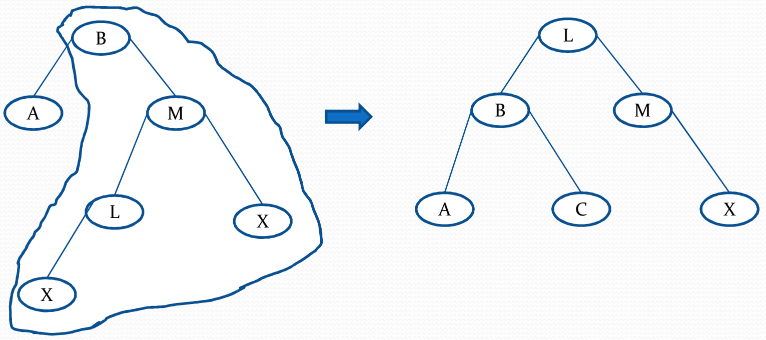


Les arbres AVL

Arbres AVL (*Exemples*)

Insérons la séquence : A, B, X, L, M, C, D, E, H, R, F **dans un arbre AVL.**

Insertion C

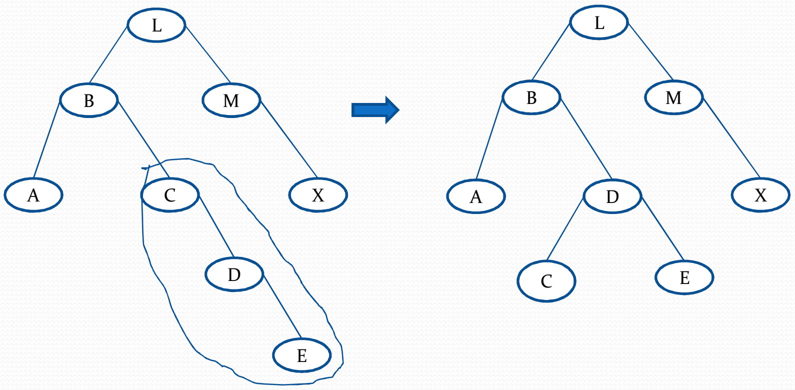


Les arbres AVL

Arbres AVL (*Exemples*)

Insérons la séquence : A, B, X, L, M, C, D, E, H, R, F **dans un arbre AVL.**

Insertion D, E

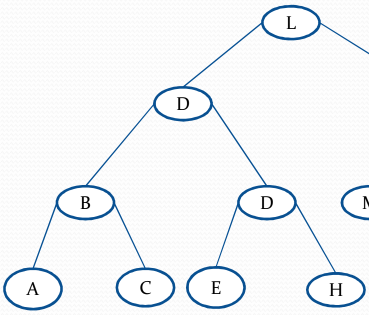


Les arbres AVL

Arbres AVL (*Exemples*)

Insérons la séquence : A, B, X, L, M, C, D, E, H, R, F **dans un arbre AVL.**

Insertion H, R, F



Les arbres AVL

Arbres AVL (*Suppression*)

Étape 1 : comme dans un arbre de recherche binaire ordinaire
Étape 2 : mettre à jour les balances

Cas où la balance d'un nœud A devient +2
→ Le fils gauche B de A doit exister

Les cas suivants peuvent se présenter
a) **B a une balance égale à + 1**
b) **B a une balance égale à – 1**
c) **B a une balance égale à 0**

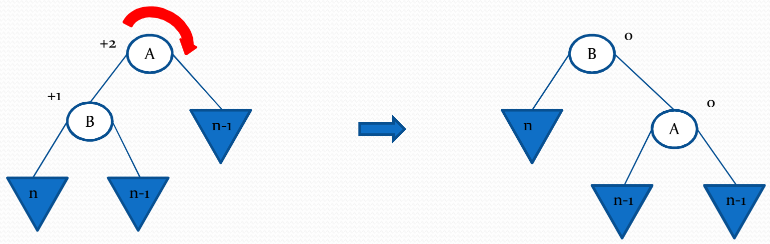
Même traitement symétrique dans le cas où la balance d'un nœud A devient -2

Traitement peut continuer en cascade

Les arbres AVL

Arbres AVL (*Suppression*)

B a une balance égale à + 1

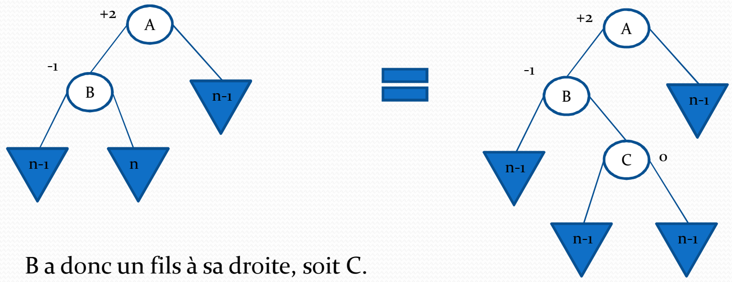


Les arbres AVL

Arbres AVL (*Suppression*)

B a une balance égale à -1

Cas Balance (C)= 0

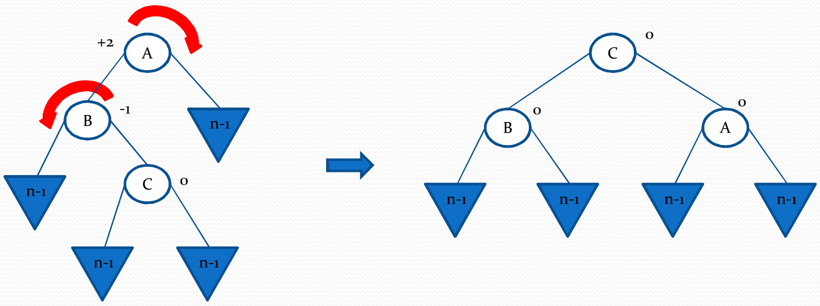


B a donc un fils à sa droite, soit C.

Les arbres AVL

<u>Arbres AVL (*Suppression*)</u>

B a une balance égale à -1 , C son fils droit avec Balance(C)=0

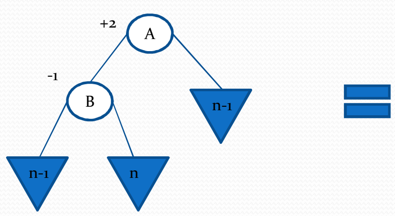


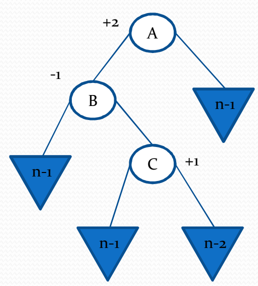
Les arbres AVL

<u>Arbres AVL (*Suppression*)</u>

B a une balance égale à -1

Balance (C)= +1



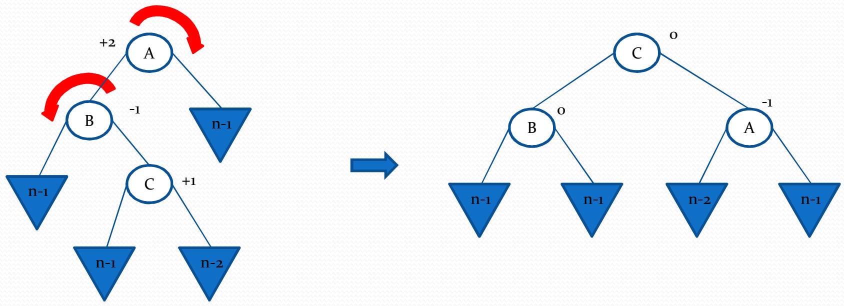


B a donc un fils à sa droite, soit C.

Les arbres AVL

<u>Arbres AVL (*Suppression*)</u>

B a une balance égale à -1 , C son fils droit avec Balance(C)=+1

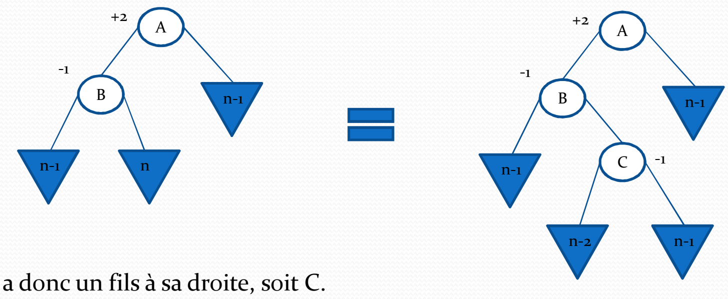


Les arbres AVL

<u>Arbres AVL (*Suppression*)</u>

B a une balance égale à -1

Balance (C)= -1

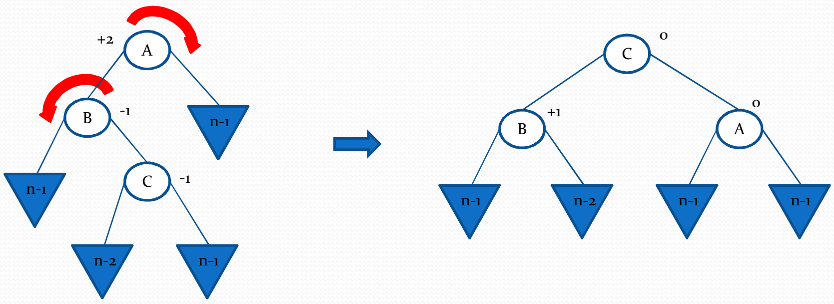


B a donc un fils à sa droite, soit C.

Les arbres AVL

Arbres AVL (*Suppression*)

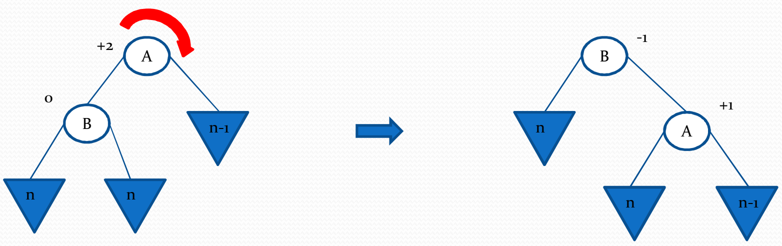
B a une balance égale à -1 , C son fils droit avec Balance(C)=-1



Les arbres AVL

Arbres AVL (*Suppression*)

B a une balance égale à 0



Les arbres AVL

<u>Arbres AVL (*Analyse théorique*)</u>

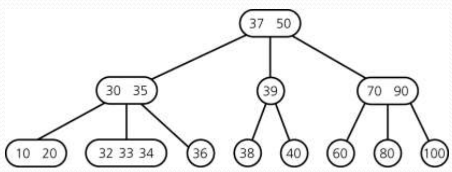
*la profondeur maximale d'un arbre binaire équilibré est 1.44*$Log_2 n$*

La recherche dans un tel arbre *n'exige jamais plus de 44% de plus de comparaisons que pour un arbre binaire complet*

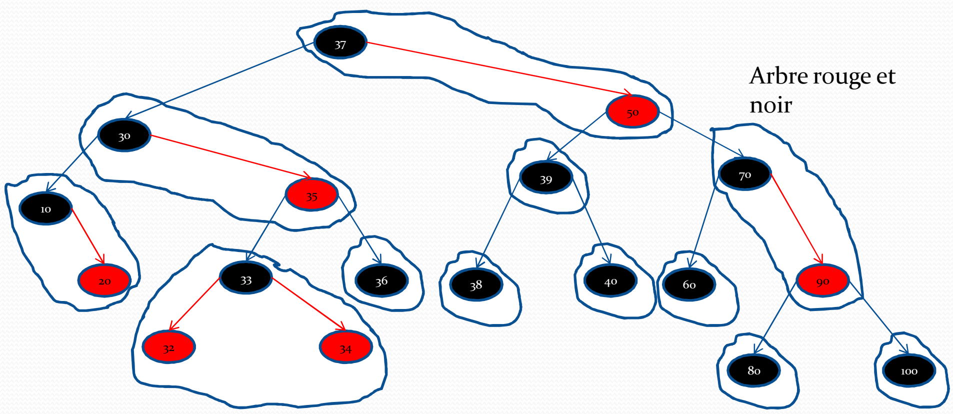
Operations de maintenance :

- **Restructuration** = *1 rotation ou double rotation*
- **Insertion :** *au plus 1 restructuration*
- **suppression :** *au plus Log2 (N) restructurations*

Les arbres Rouge et Noir



Arbre 2-4



Arbre rouge et noir

Les arbres Rouge et Noir

Arbres Red Black

Un arbre rouge et noir (RB-tree) est un arbre binaire de recherche où chaque nœud est de couleur rouge ou noire .

De plus, toutes les branches issues de tout nœud :
1. Ne possèdent pas deux nœuds rouges consécutifs.
2. Possèdent le même nombre de nœuds noirs

La racine est noir

- Nœuds noirs : équilibrage parfait
- Nœuds rouges : tolérer légèrement le déséquilibre

Pire des cas:
Alternance entre les nœuds rouges et noirs.

Les arbres Rouge et Noir

Arbres Red Black (Insertion)

Insertion comme dans un arbre de recherche binaire.

Le nœud inséré est toujours une feuille

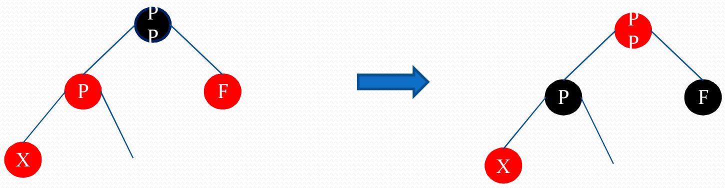
On lui attribue la couleur rouge

Si son père est aussi rouge, un algorithme de maintenance est appliqué

Les arbres Rouge et Noir

Arbres Red Black(Insertion)

CAS 1: le frère F de P est rouge



X : nœud introduit

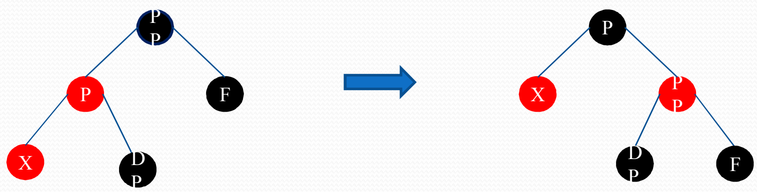
Les nœuds P et F deviennent noirs et leur père PP devient rouge.

Le processus continue en cascade

Les arbres Rouge et Noir

Arbres Red Black(Insertion)

CAS 2: le frère F de P est noir et X est le fils gauche de P.



Rotation droite du nœud PP.

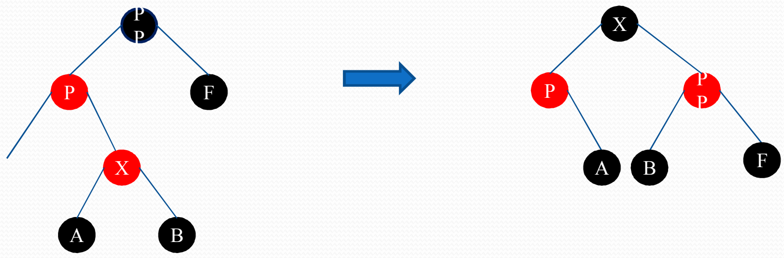
P devient noir et PP rouge.

Le processus se termine

Les arbres Rouge et Noir

Arbres Red Black(Insertion)

CAS 3: le frère F de P est noir et X est le fils droit de P.



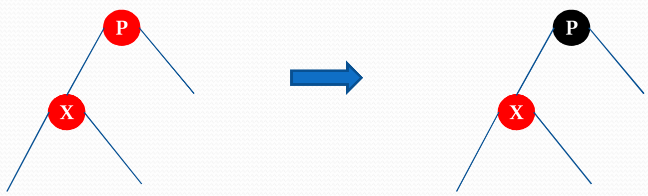
Rotation gauche du nœud P + rotation droite du nœud PP.
X devient noir et PP rouge.

Le processus se termine

Les arbres Rouge et Noir

Arbres Red Black(Insertion)

CAS 4: le nœud père P est la racine de l'arbre



Le nœud père devient noir

C'est le seul cas où la hauteur noire de l'arbre augmente.

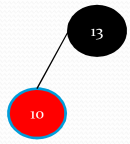
Le processus se termine

Les arbres Rouge et Noir

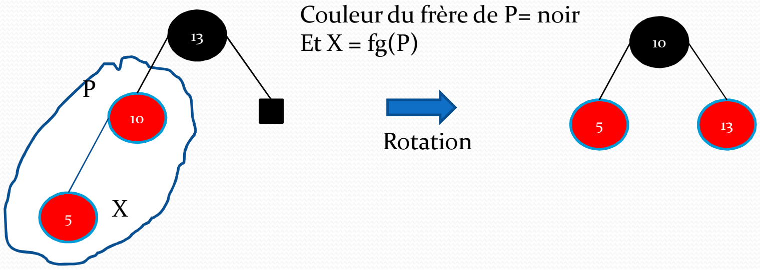
Arbres Red Black (Exemple)

Insérer 13

Insérer 10



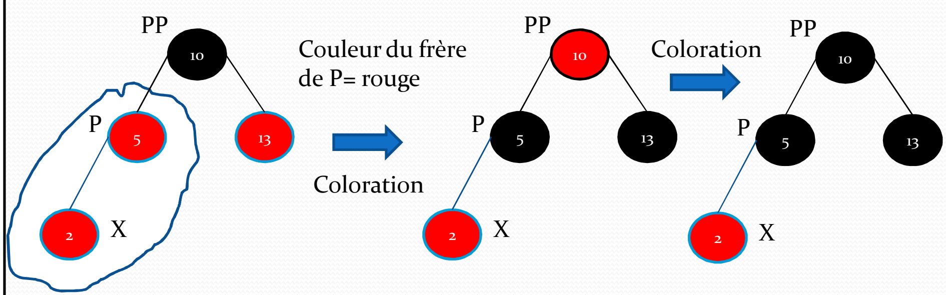
Insérer 5



Les arbres Rouge et Noir

Arbres Red Black(exemple)

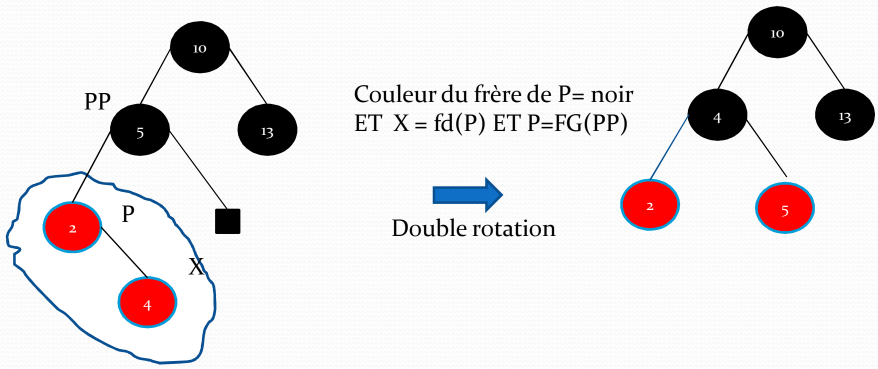
Insérer 2



Les arbres Rouge et Noir

Arbres Red Black (exemple)

Insérer 4



Les arbres Rouge et Noir

Arbres Red Black (Suppression)

Suppression comme dans un arbre de recherche binaire.

Si le nœud physiquement supprimé est noir, un algorithme de maintenance est appliqué.

On considère que le nœud qui remplace le nœud supprimé porte une couleur noire en plus.

Ceci signifie qu'il devient noir s'il est rouge et qu'il devient doublement noir s'il est déjà noir.

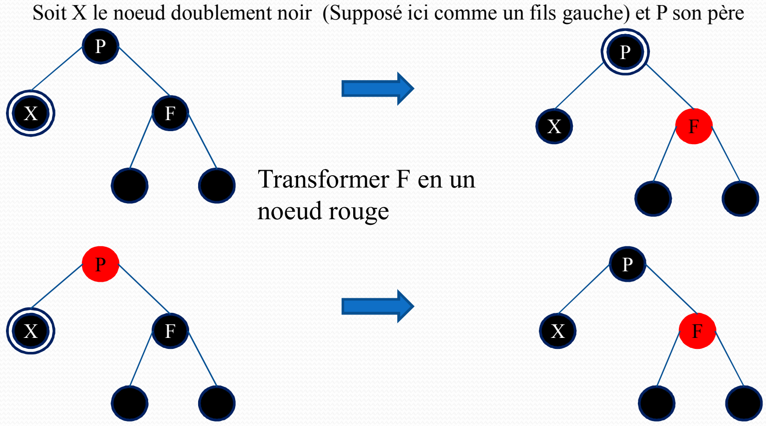
L'algorithme de maintenance a donc pour rôle de supprimer ce nœud doublement noir.

Les arbres Rouge et Noir

Arbres Red Black (Suppression)

CAS 1: Le frere F de X est noir et a deux fils noirs

Soit X le noeud doublement noir (Supposé ici comme un fils gauche) et P son père

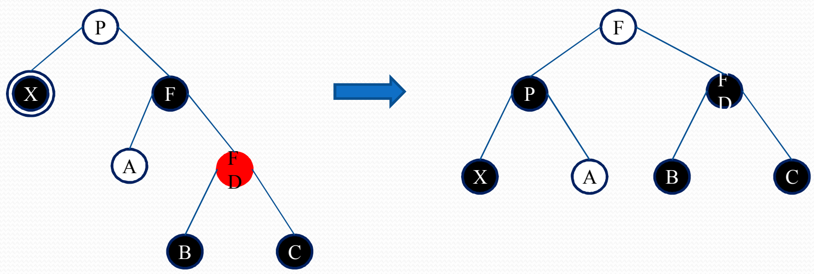


Si le parent P est noir, le processus continue en remontant dans l'arbre. X devient le nouveau noeud doublement noir.

Les arbres Rouge et Noir

Arbres Red Black (Suppression)

CAS 2: Le frère F du noeud X est noir et a un fils droit rouge (FD)



Rotation gauche de P

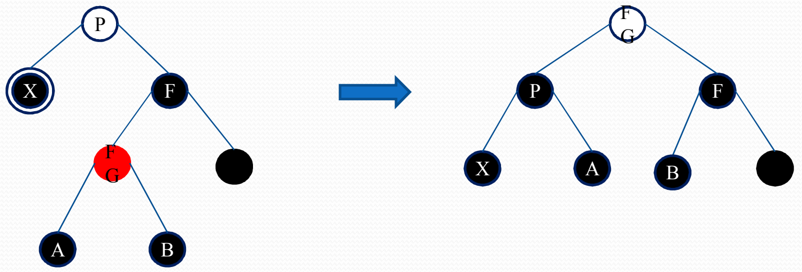
Recolorer : P et FD deviennent noirs et la couleur de F est celle de P avant la transformation.

Le processus se termine

Les arbres Rouge et Noir

Arbres Red Black (Suppression)

CAS 3: si le frere F du noeud X est noir et a un fils gauche rouge(FG)



Rotation droite (F) + rotation gauche (P).

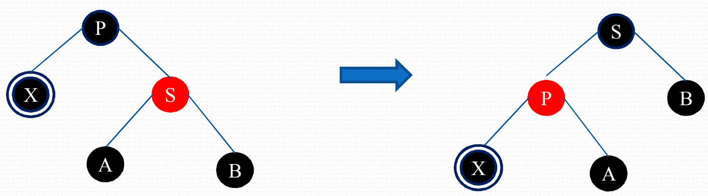
Recolorer : P devient noir et la couleur de FG est celle de P avant la transformation.

Le processus se termine

Les arbres Rouge et Noir

Arbres Red Black (Suppression)

CAS 4: Le frère F de X est rouge



Rotation droite(P)

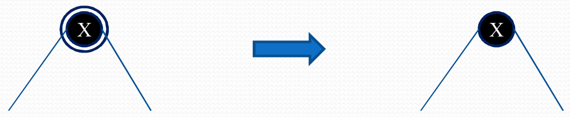
Le processus continue selon le cas 1, 2 ou 3

Les arbres Rouge et Noir

Arbres Red Black (Suppression)

CAS 0: X est la racine de l'arbre

Soit X le noeud doublement noir



X devient simplement un noeud noir.

C'est le seul cas ou la hauteur de l'arbre diminue. Le processus se termine.

Les arbres Rouge et Noir

Arbres Red-Black (*Analyse théorique*)

Operations de maintenance :
- Restructuration et coloration.
- Insertion : au plus 1 restructuration et au plus Log_2 (n) colorations.
- suppression : au plus 2 restructurations et au plus Log_2 (n) colorations.
N : nombre d'élements insérés.

la profondeur maximale d'un arbre binaire équilibré est $2*Log_2(n)$

Recherche, insertion et suppression : $O(Log_2(n))$