Les arbres binaires

- <u>**Définition**</u>

Un arbre binaire de recherche (ABR) est un arbre binaire dans lequel chaque nœud possède une clé d'identification, et tel que chaque nœud du sous-arbre gauche a une clé inférieure à celle du nœud considéré, et que chaque nœud du sous-arbre droit a une clé supérieure à celle-ci.

Les arbres binaires

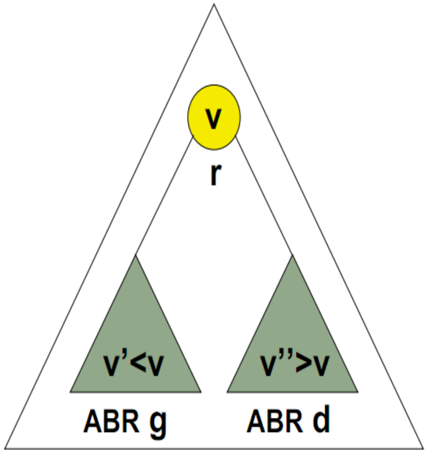
- **Utilité**

- Permettent de rechercher et de trouver un nœud de façon rapide et simple.
- Utiles pour ranger des valeurs ordonnées.
- S’approche de la méthode de recherche dichotomique

Les arbres binaires

- **Terminologies**

Un arbre binaire de recherche (ABR) est un arbre où les nœuds sont organisés en fonction d'un ordre, celui des valeurs contenues dans ses noeuds.



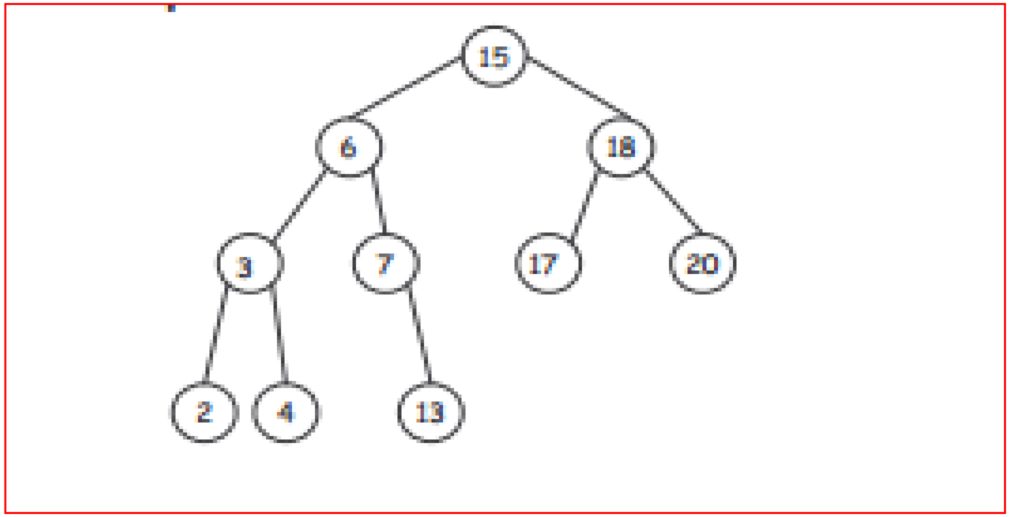
Un ABR a est :

- soit vide : a = ()
- soit non vide : a = (r, g, d)

```
si v est la valeur de la racine r, alors
  si g est non vide, alors ∀ v'∈ g, v'< v
    g est un ABR (sous-arbre gauche)
    si d est non vide, alors ∀ v''∈ d, v''> v
      d est un ABR (sous-arbre droit),
      il n'y a pas de doublon
```

Les arbres binaires

Exemple d’un arbre binaire



Toutes les valeurs des nœuds du sous arbre gauche sont inférieures à la valeur de la racine.

Toutes les valeurs des nœuds du sous arbre droit sont supérieures à la valeur de la racine.

Les arbres binaires

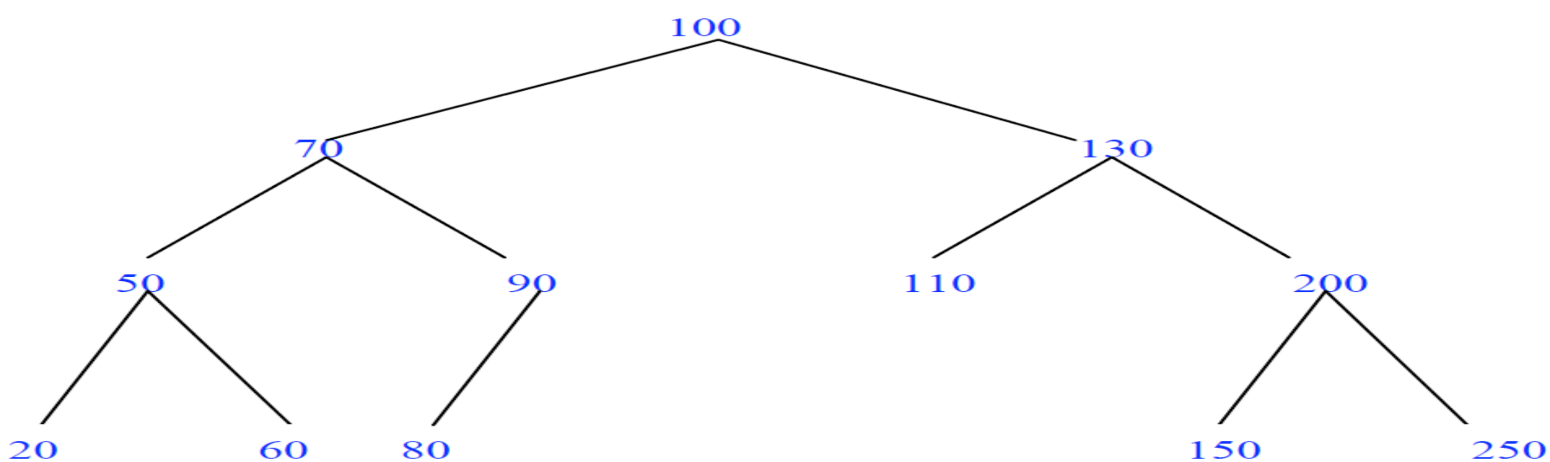
La représentation des arbres binaires

- Il existe plusieurs façons de représenter un arbre:
  - Statique (tableau)
  - Dynamique (allocation dynamique des nœuds, utilisation de pointeurs)
  - Mixte

Les arbres binaires

- **Représentation par un tableau**

Soit l'arbre suivant

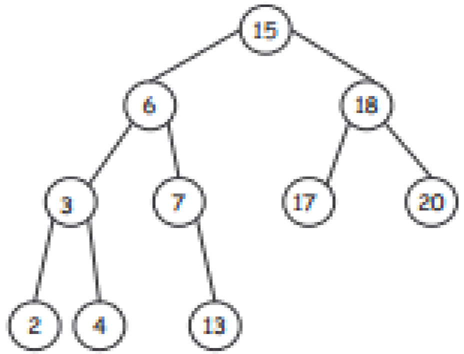


100	70	130	50	90	110	200	20	20	60	80	NUL	NUL
1	2	3	4	5	6	7	8	9	10	11	12	13

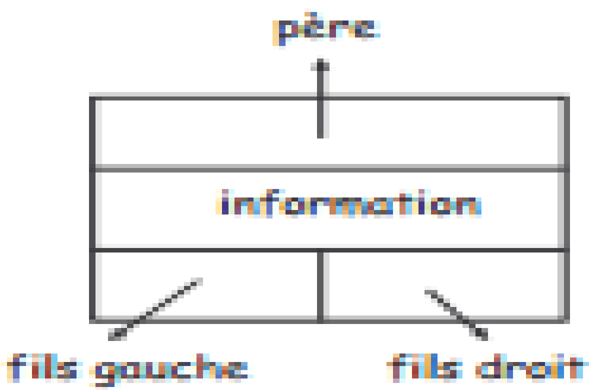
Les arbres binaires

- **Représentation par les pointeurs**

Soit l'arbre suivant

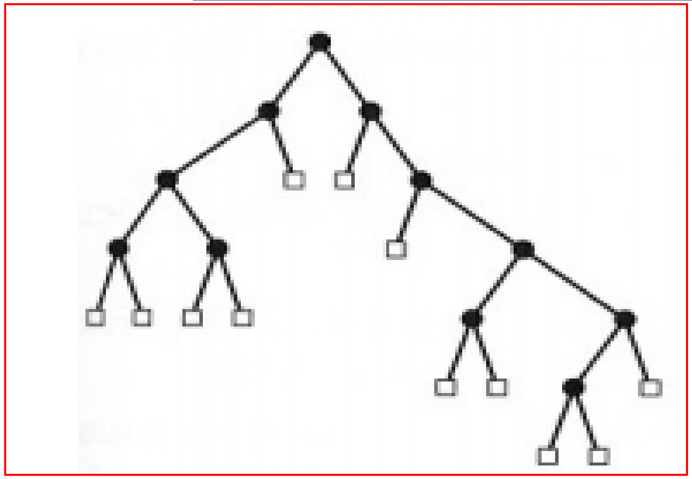


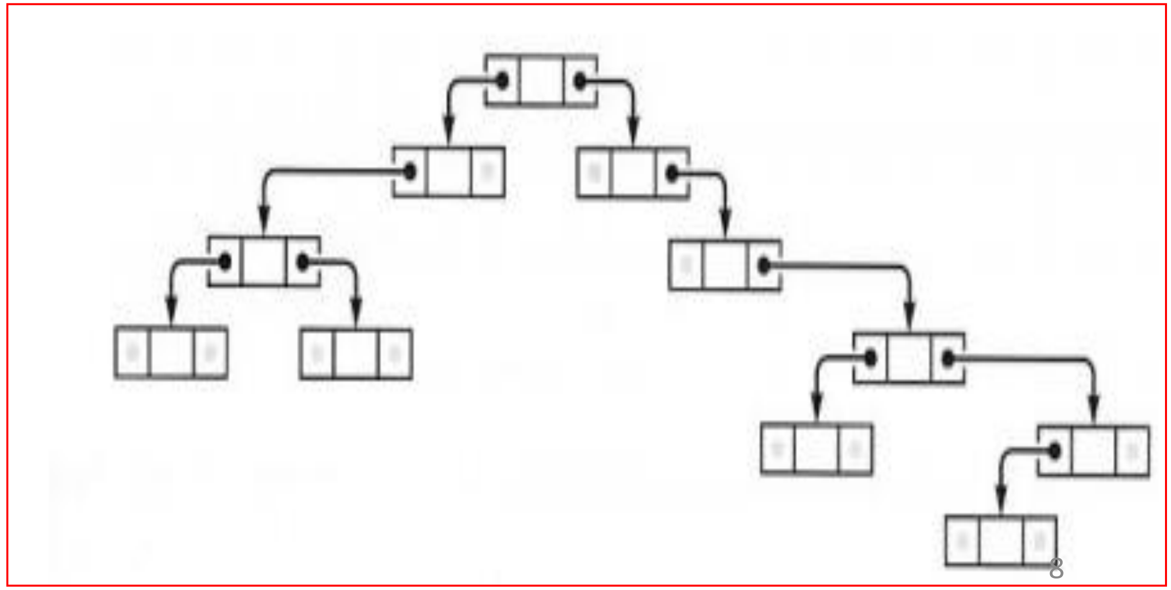
Chaque nœud de l'arbre est représenté comme suit :



Les arbres binaires

Exemple d’une représentation d’un arbre binaire





Les arbres binaires

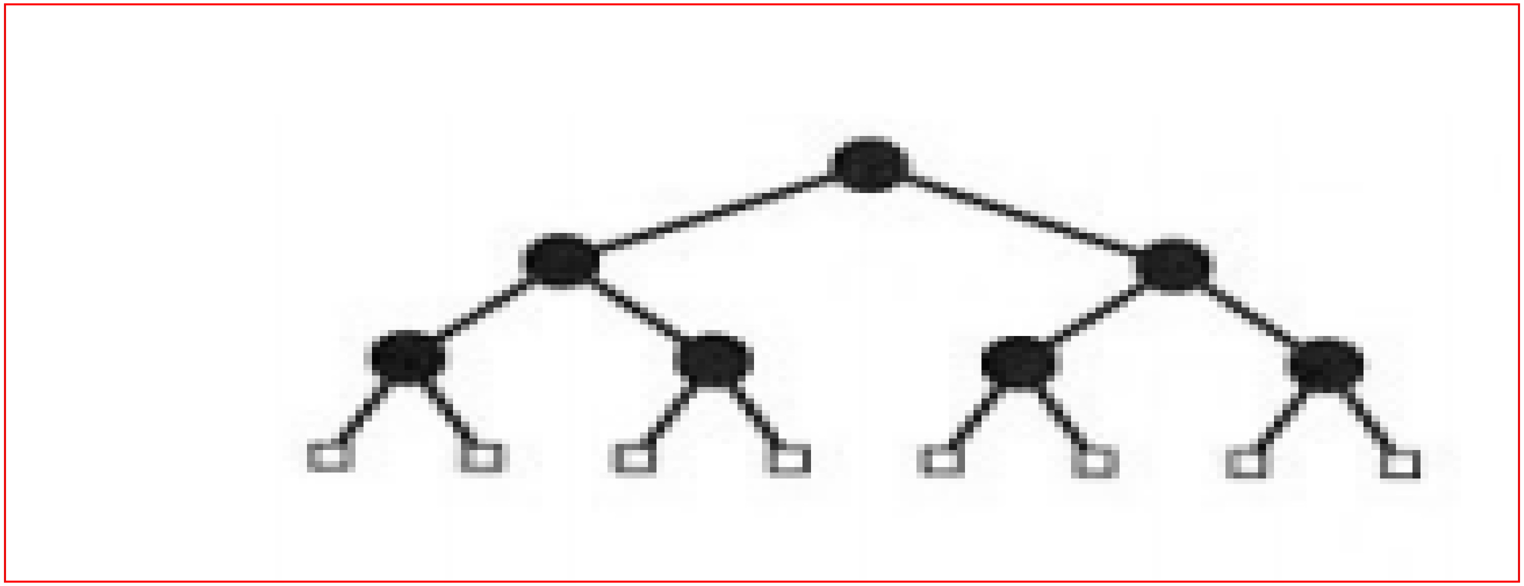
Propriétés des arbres binaires

- Un nœud possède au moins un fils
- Une feuille est un nœud sans fils
- Chaque nœud interne a 2 fils : un fils gauche et un fils droit
- Un arbre binaire avec N nœuds internes :
  - a N+1 nœuds externes
  - a 2N liens en tout : N-1 liens vers des nœuds internes et N+1 liens vers des nœuds externes

Les arbres binaires

- <u>Les arbres binaires complets</u>

Un arbre binaire complet est un arbre où tous les nœuds feuilles apparaissent au dernier niveau.



Les arbres binaires

Arbre binaire ordonné

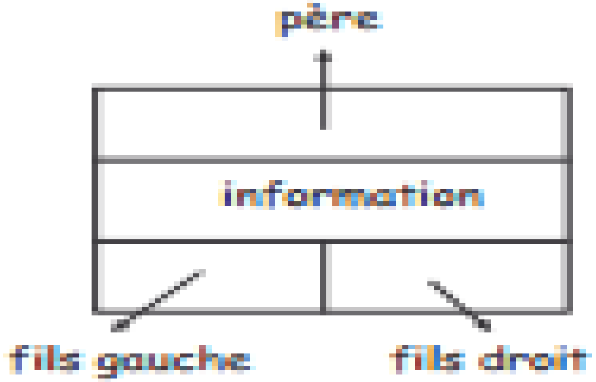
Chaque nœud ne porte que des valeurs plus petites que lui-même dans son sous arbre gauche et que des valeurs plus grandes dans son sous arbre droit.

Cette structure permet de rechercher rapidement des données.

Les arbres binaires

- **Les modèles des arbres binaires**

Dans les opérations formant le modèle des arbres binaires, les paramètres fg et fd sont des pointeurs vers des nœuds alors que v est un type quelconque (l'information ou la valeur du nœud).



Les arbres binaires

- Le nœud est alors formé d'au moins trois champs :

- ➢ v information
- ➢ fg fils gauche, de type pointeur
- ➢ fd fils droit, de type pointeur

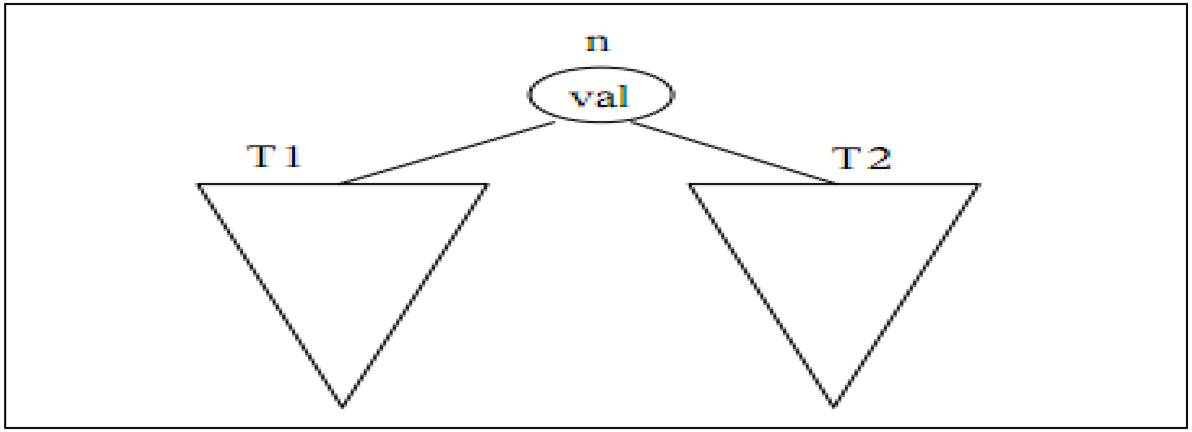
Les arbres binaires

Les principales fonctions sur un arbre :

CreerNoeud(val)	Alloue un nouveau nœud contenant v comme information et NULL comme fils-gauche et fils-droit. La fonction retourne l'adresse du nouveau nœud.
LibererNoeud(a)	Retourne l'information stockée dans le noeud pointé par a
Fg(a)	Retourne le contenu du champs 'fg' du noeud pointé par a
Fd(a)	Retourne le contenu du champs 'fd' du noeud pointé par a
Aff-info(p, val)	Affecte val dans le champs ‘valeur' du nœud pointé par p
Aff-fg(p, q)	Affecte q dans le champs 'fg' du noeud pointé par p
Aff-fd(p, q)	Affecte q dans le champs 'fd' du noeud pointé par p

Les arbres binaires

- **Parcours des arbres**



1- Le parcours préordre

Le parcours *préordre* d'un arbre non vide consiste à visiter le nœud racine, puis parcourir récursivement en *préordre* les sous-arbres T1 puis T2 → [n , T1 , T2]

Les arbres binaires

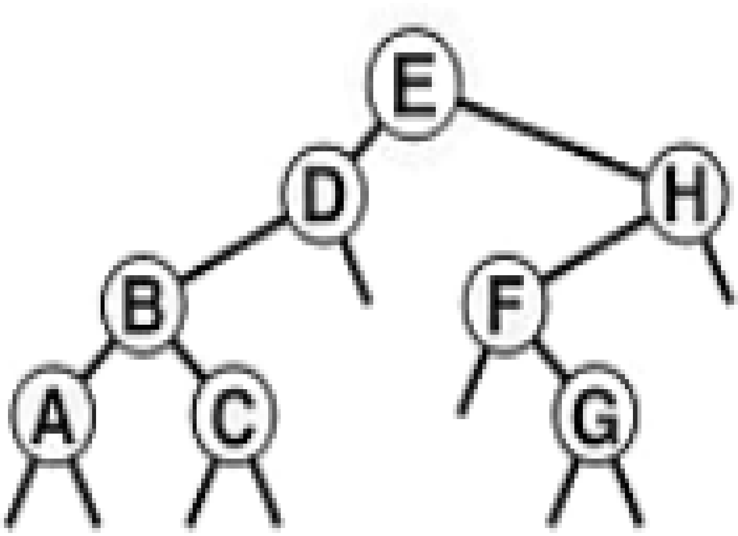
Le parcours préordre

Exemple d’une procédure récursive qui affiche les valeurs (informations) de tous les nœuds d'un arbre de racine R en le parcourant en **préordre** :

```
preordre( R:ptr )
Debut
  SI R <> NULL
    ecrire( info(R) );
    preordre( fg(R) );
    preordre( fd(R) )
  FSI
Fin
```

Les arbres binaires

Exemple d’un parcours pré ordre




```
traverse E
  visit E
  traverse D
    visit D
    traverse B
      visit B
      traverse A
        visit A
        traverse •
        traverse •
      traverse C
        visit C
        traverse •
        traverse •
    traverse •
  traverse H
    visit H
    traverse F
      visit F
      traverse •
      traverse G
        visit G
        traverse •
        traverse •
    traverse •
```

Les arbres binaires

Algorithme Préordre itératif

- L'algorithme itératif nécessite l'utilisation d'une pile P pour sauvegarder tout nœud examiné, afin de pouvoir parcourir son sous-arbre droit après avoir effectué le parcours de son sous-arbre gauche.

  Le parcours du sous-arbre gauche d'un nœud X est réalisé par la procédure parcours fils gauche définie ci-dessous.

Les arbres binaires

Le parcours du sous-arbre gauche d'un nœud X est réalisé par la procédure parcours fils gauche définie ci-dessous :

Procedure Préordre(racine)	Procedure PreordreG(X)

<pre>Debut
  X ← racine ;
  PreordreG(X) ;
  Repeter
    DepilerR(X);
    Si fd(X) ≠ NULL Alors
      X ← fd(X);
      PreordreG(X)
    FSi
   Jusqu'à Vide(P)
Fin</pre>
	<pre>Debut
	  Traiter(X);
	  Empiler(X);
	  TantQue fg(X) Faire
	    X ← Gauche(X);
	    Traiter(X);
	    Empiler(X);
	  Fin-TQ
	Fin</pre>

Les arbres binaires

2- Le parcours inordre

Le parcours *inordre* d'un arbre non vide consiste d'abord à parcourir récursivement en *inordre* le sous-arbre gauche T1, puis visiter le nœud racine (n), ensuite parcourir récursivement en *inordre* le sous-arbre droit T2 → [T1 , n , T2]

Les arbres binaires

Le parcours inordre

Voici une procédure (récursive) qui affiche les valeurs (informations) de tous les nœuds d'un arbre de racine R en le parcourant en *inordre* :

```
Inordre( R:ptr )
Debut
  SI R <> NULL
    inordre( fg(R) );
    ecrire( info(R) );
    inordre( fd(R) )
  FSI
Fin
```

Les arbres binaires

Le parcours inordre itératif

```
Procedure inOrdre(R:noeud)
DEBUT
  SI R est une feuille Alors
    Traiter(R)
  SINON
    DEBUT
      n1 ← fils le plus à gauche de R;
      inOrdre(n1);
      lister n;
      POUR chaque fils de R (sauf n1) de gauche à droite FAIRE
        inOrdre(n:f ils)
      FPOUR
    FSINON
FIN
```

Les arbres binaires

3- Le parcours postordre

Le parcours *postordre* d'un arbre non vide consiste d'abord à parcourir récursivement en *postordre* les sous-arbres T1 puis T2, ensuite visiter le nœud racine (n) → [T1 , T2 , n]

Les arbres binaires

Voici une procédure (récursive) qui affiche les valeurs (informations) de tous les nœuds d'un arbre de racine R en le parcourant en *postordre* :

```
Procedure postordre( R:ptr )
Debut
  SI R <> NIL
    postordre( fg(R) );
    postordre( fd(R) );
    ecrire( info(R) )
  FSI
Fin
```

Les arbres binaires

- **Exemple des différents parcours**

on aura pour le parcours préordre :

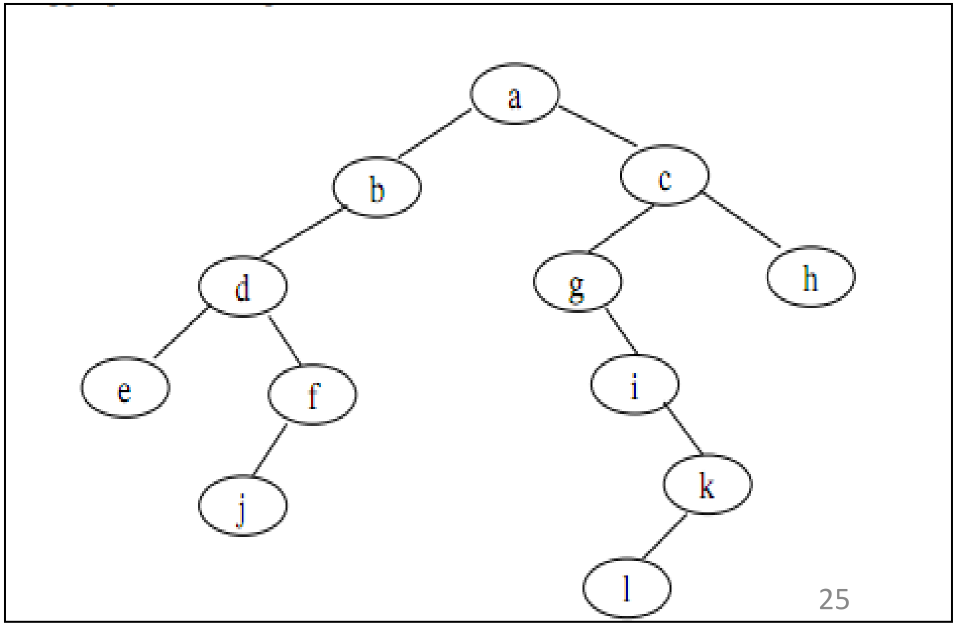
a b d e f j c g i k l h

pour le parcours inordre :

e d j f b a g i l k c h

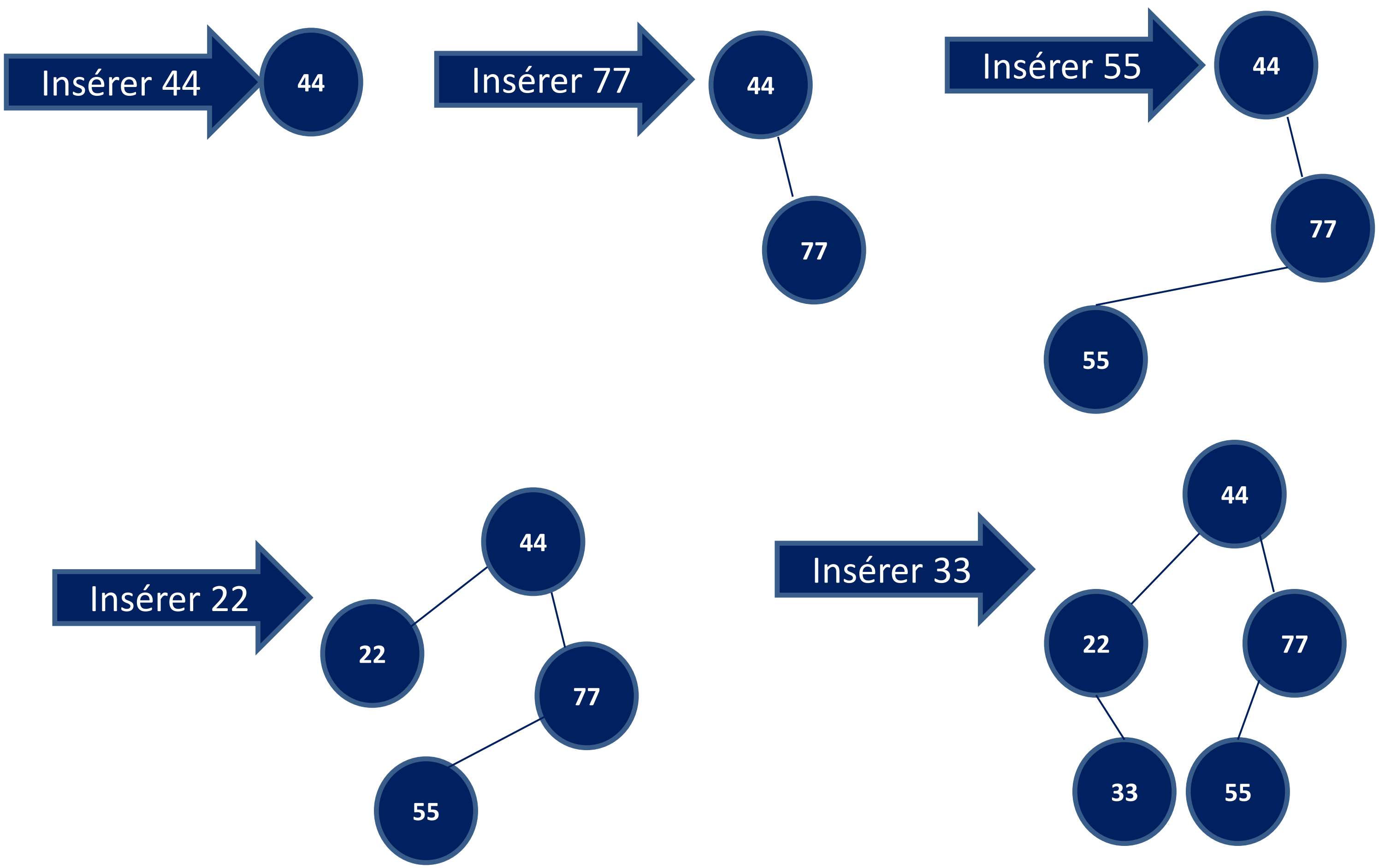
et pour le parcours postordre :

e j f d b l k i g h c a



Les opérations sur les arbres binaires / insertion

Exemple : 44, 77, 55, 22, 33



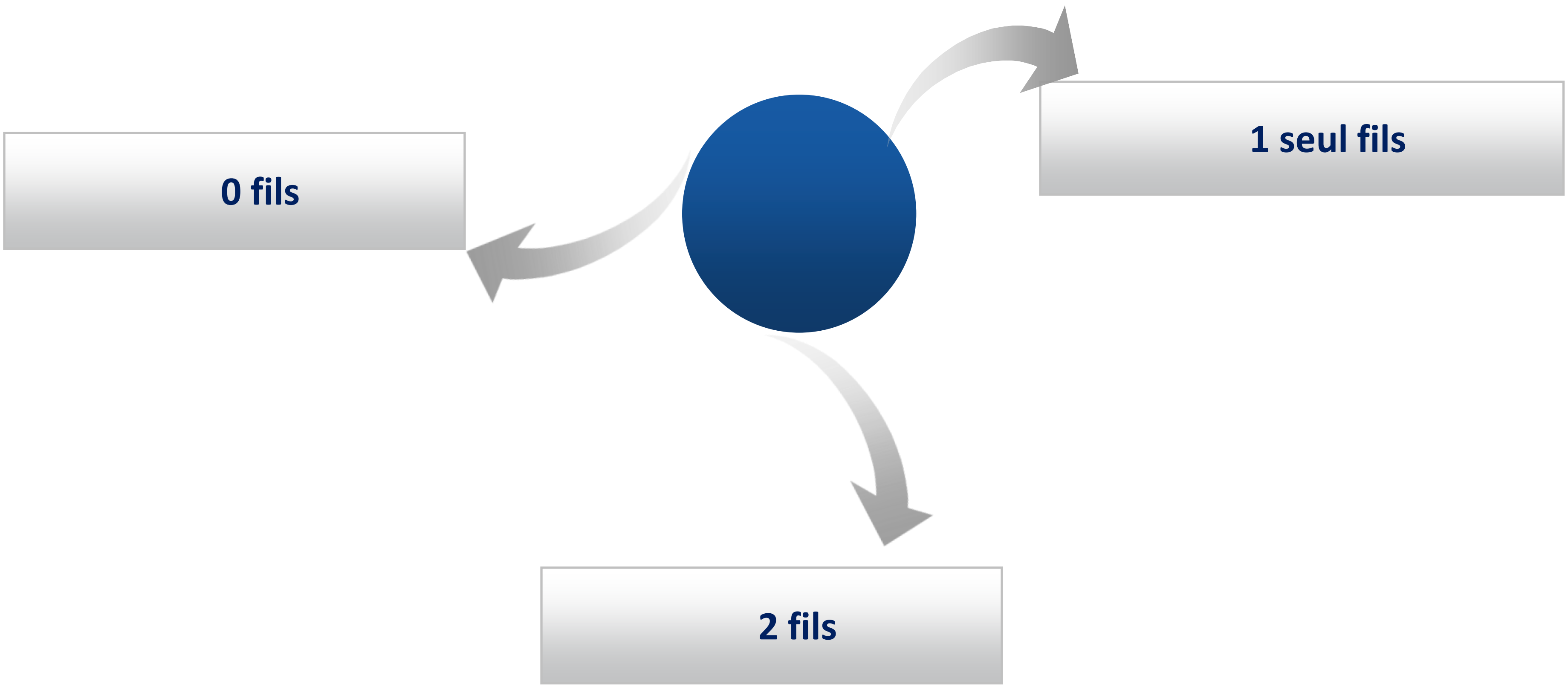
Les arbres binaires

Algorithme d’insertion dans un arbre binaire

```
Arbre NouvelArbre(Element  val , Arbre a, Arbre b)
{
 Arbre c;
 c = (Arbre)malloc (sizeof (struct Noeud ));
 c->contenu = v;
 c->fg = a;
 c->fd = b;
 return c;
}
void AjouterArbre(Element v, Arbre *arbre)
{ Arbre a = *arbre;
  if (a == NULL ) a = NouvelArbre(val, NULL , NULL );
  else
     if (v <= a->element) AjouterArbre(val, &a->fg);
     else AjouterArbre(val, &a->fd);
  *arbre = a;
}
```

Les arbres binaires

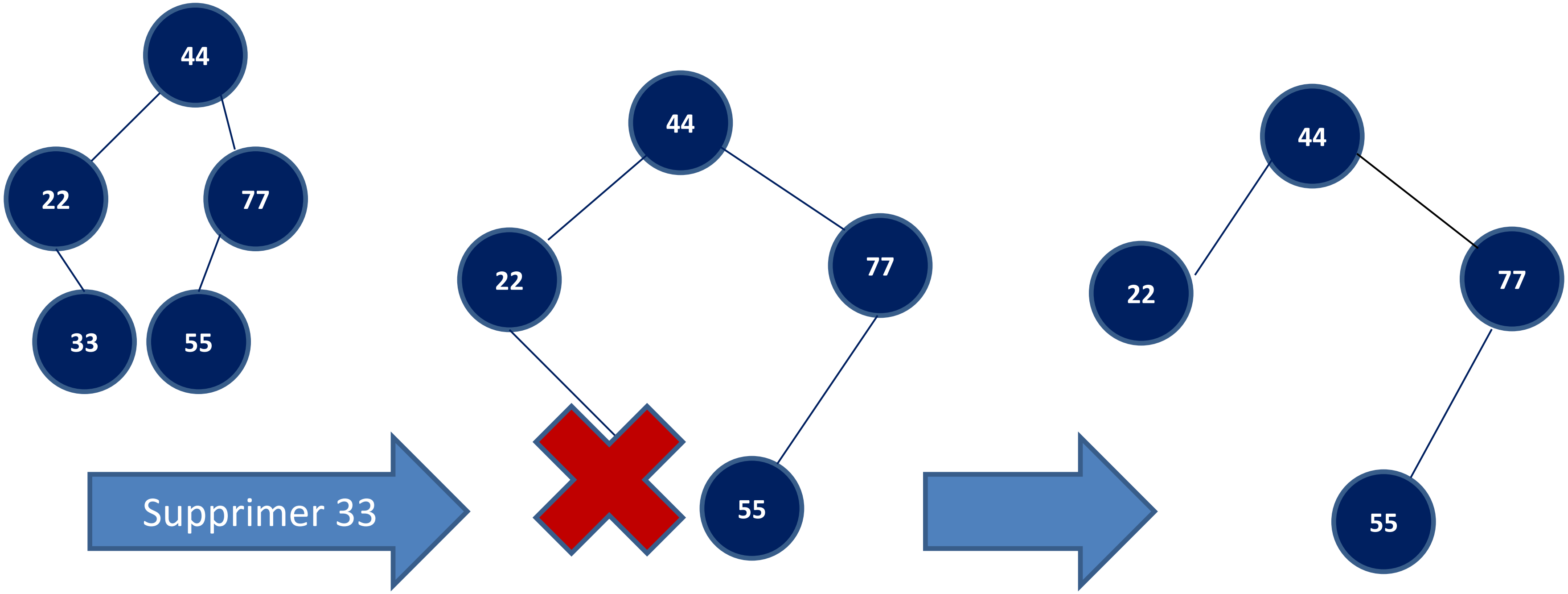
- **Les opérations sur les arbres binaires / Insertion**



Les arbres binaires

- Le nœud à supprimer n'a pas de fils :

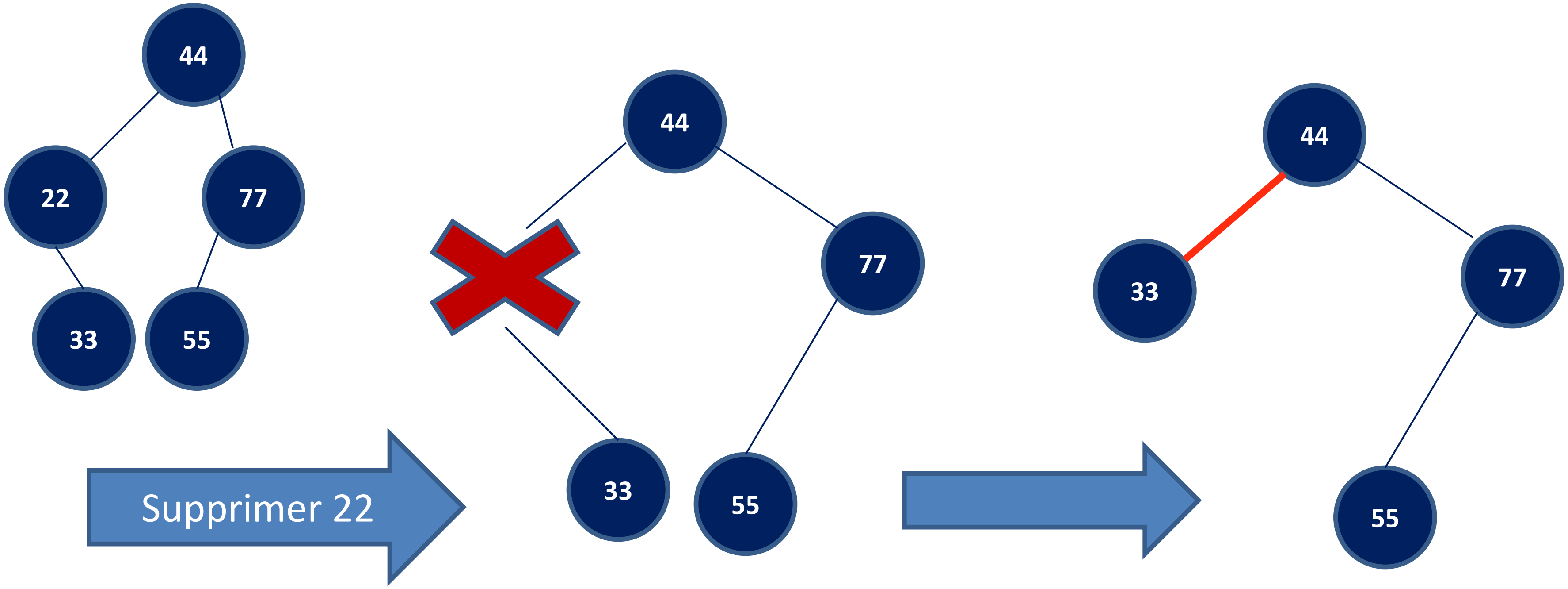
On supprime le nœud et on remplace son adresse par NULL



Les arbres binaires

- 2ème cas: le nœud a un seul fils :

On supprime le nœud et on remplace son adresse dans son père par l'adresse de son fils

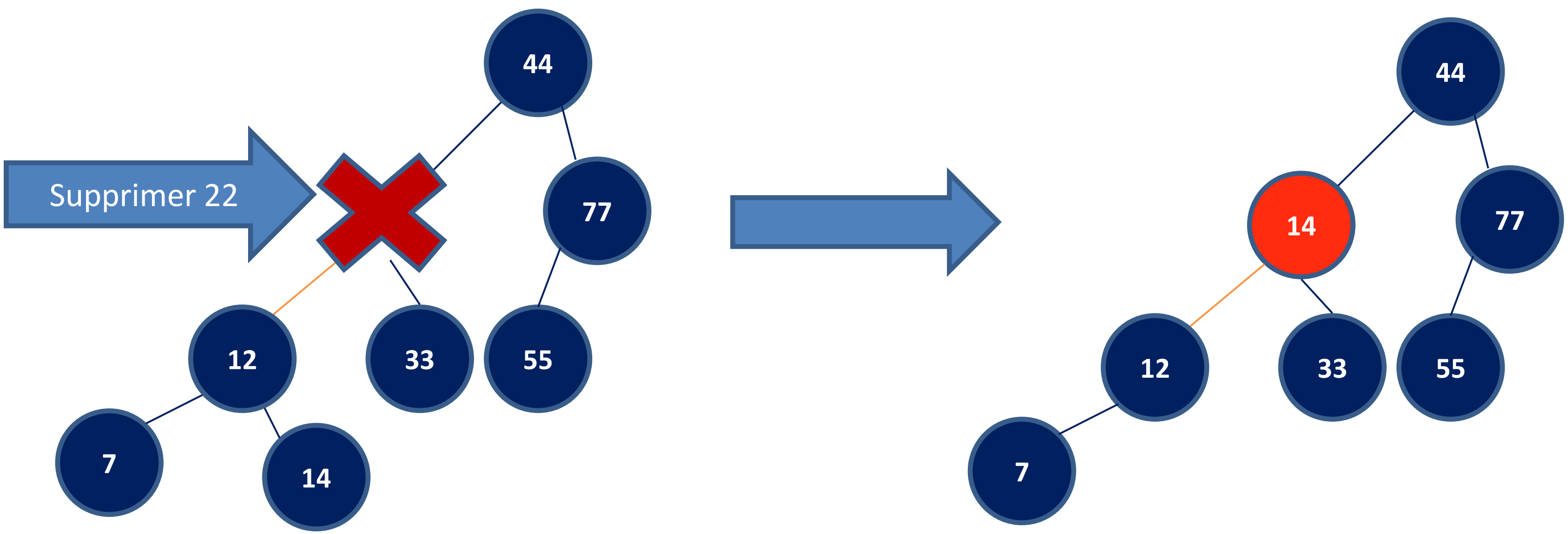


Les arbres binaires

3ème cas: Le nœud a deux fils :

On supprime N et on le remplace par l'élément maximal de son sous arbre gauche puis on le supprime.

On supprime N et on le remplace par l'élément minimal de son sous arbre droit puis on le supprime.



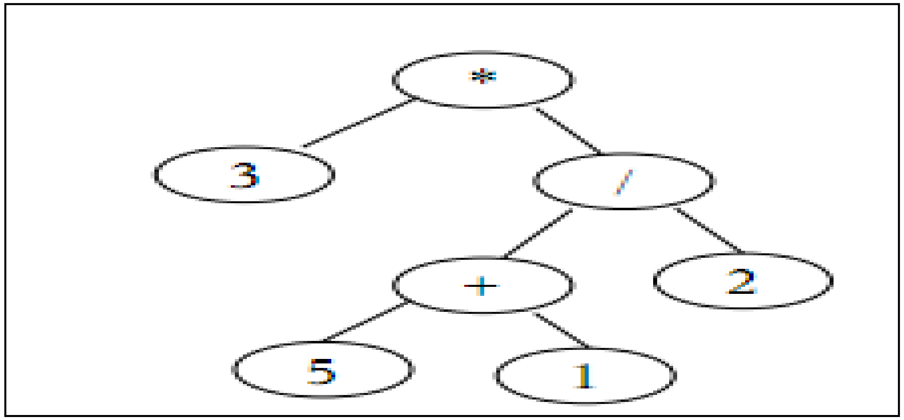
Les arbres binaires

Application des arbres binaires

1)Représentation des expressions arithmétiques

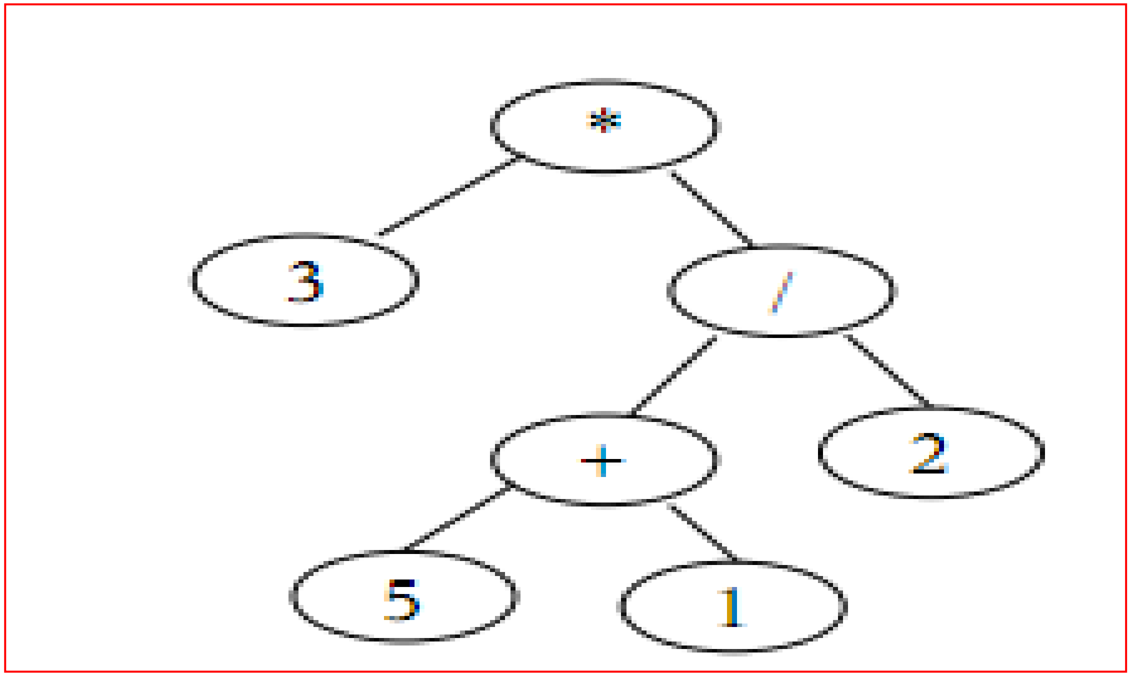
Les expressions arithmétiques peuvent êtres représentées sous forme d'arbre binaire. Les nœuds internes contiennent des opérateurs, alors que les feuilles contiennent des valeurs (opérandes).

Par exemple, l'expression `3*((5+1)/2)` sera représentée par l'arbre suivant :



Les arbres binaires

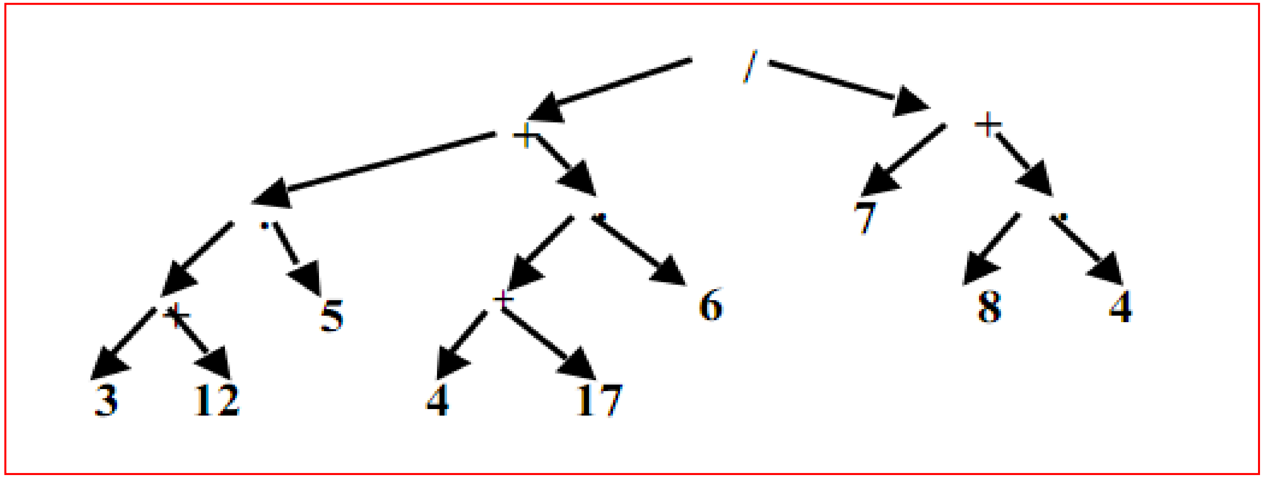
- Par exemple, l'expression `3*((5+1)/2)` sera représentée par l'arbre suivant.



Les arbres binaires

- **Arbres de calcul arithmétique**

Définition : Un arbre de calcul arithmétique est un arbre dont tous les nœuds non feuille ont exactement deux fils, dont les données sur les feuilles sont des nombres et les données sur les nœuds non feuilles sont des signes d'opération.



Les arbres binaires

- **Enumérations**

  On utilise dans ce cas plusieurs énumérations en profondeur

Énumération préfixée : On écrit un nœud avant de visiter ses fils.

Dans l'exemple on obtient :

`/ + . + 3 12 5 . + 4 17 6 + 7 . 8 4`

On peut reconstituer l'arbre à partir de l'écriture

Énumération postfixée : On écrit ou traite un nœud après avoir visité ses fils.

Dans l'exemple 4 on obtient : **`3 12 + 5 . 4 17 + 6 . + 7 8 4 . + /`**

Énumération infixée : On écrit ou traite un nœud entre son 1 er et son 2ème fils. Dans l'exemple 4 on obtient

`3 + 12 . 5 + 4 + 17 . 6 / 7 + 8 . 4`

C'est l'écriture classique, mais sous cette forme elle est ambiguë. On ne peut se passer de parenthèses.

Les arbres binaires

Accélérer l'accès par position dans une liste

On utilise un genre particulier d'arbre de recherche binaire pour rendre l'accès par position beaucoup plus rapide qu'un simple parcours séquentiel.

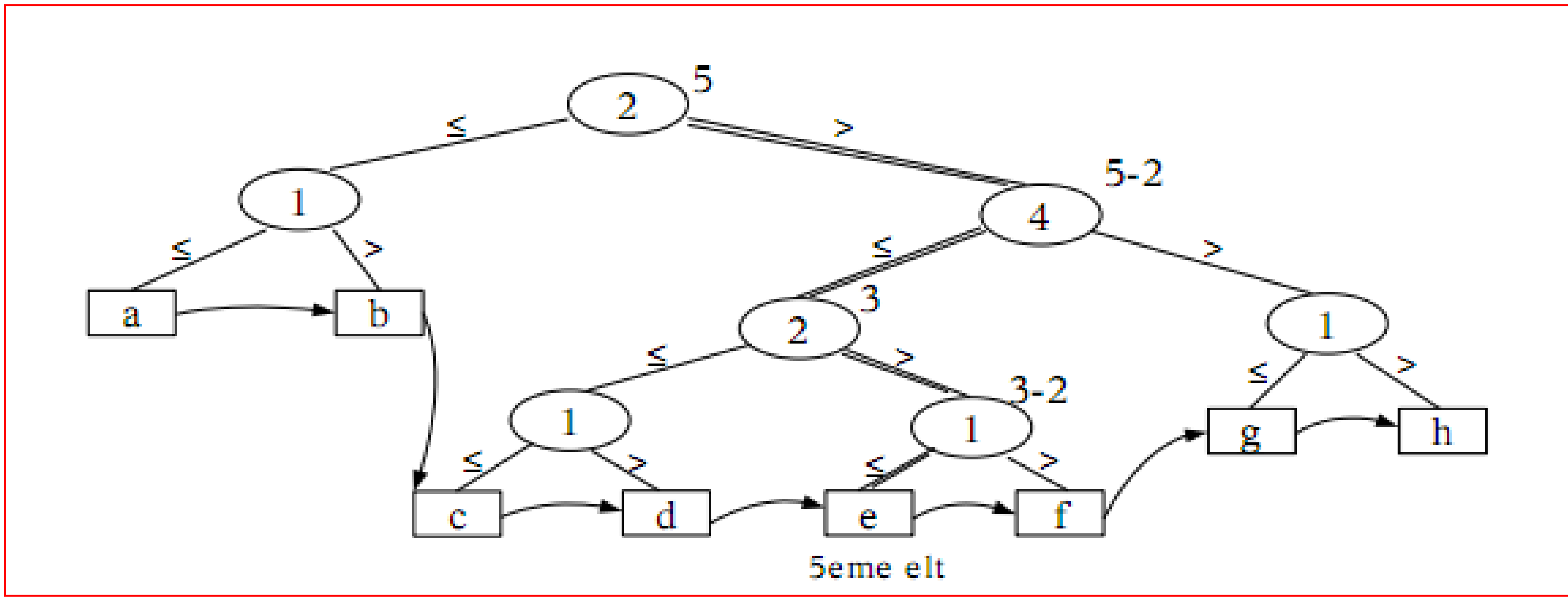
Dans ce type d'arbre, les feuilles contiennent les valeurs d'une liste donnée, alors que les nœuds, internes forment un index de position : chaque nœud interne contient un entier désignant le nombre de feuilles de son sous-arbre gauche.

Les arbres binaires

Accélérer l'accès par position dans une liste

Cette information sera utilisée pour guider la recherche d'une position donnée :

Si la position cherchée est inférieure ou égale à l'information du nœud interne alors on descend à gauche.



Les arbres binaires

Accélérer l'accès par position dans une liste

Le schéma précédent, montre le déroulement de la recherche de la cinquième position :

On compare 5 par rapport à la racine (2), c'est supérieur, donc on descend à droite en retranchant 2 à la position cherchée, car en allant dans le sous arbre droit, on a écarté 2 feuilles (celles du sous arbre gauche).

Maintenant on recherche la position 3, alors que l'information du nœud courant indique qu'il y a 4 feuille à sa gauche, donc on descend à gauche.

On compare 3 avec l'information du nœud qui indique 2 feuilles à sa gauche, donc on descend à droite et on retranche 2.

Les arbres binaires

Accélérer l'accès par position dans une liste

Maintenant on recherche la position 1 et comme c'est inférieur ou égal à l'information du nœud courant (1) on descend à gauche.

Etant arrivé au niveau d'une feuille avec la position cherchée égale à 1, on est donc positionné sur le bon élément.

Les arbres binaires

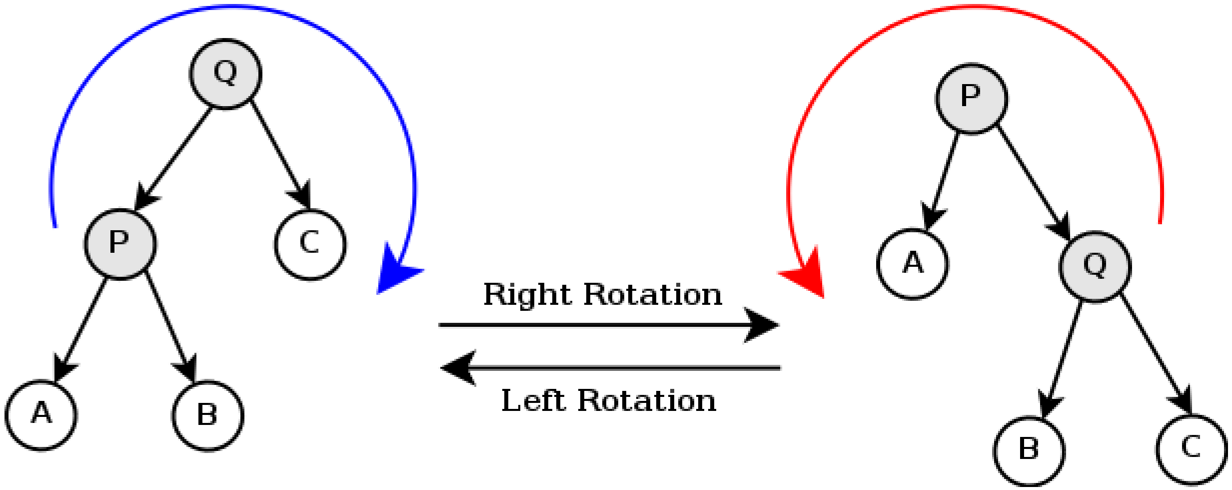
Les rotations sur les arbres binaires

La **rotation d'un arbre binaire de recherche** permet de changer la structure d'un arbre binaire de recherche ou ABR sans invalider l'ordre des éléments.

Une telle rotation consiste en fait à faire remonter un nœud dans l'arbre et à en faire redescendre un autre.

Les arbres binaires

Illustration



Les AVL

- **Introduction**

Si l'insertion ou la suppression provoque un déséquilibre de l'arbre, rétablir l'équilibre.

Le meilleur ABR maintient à tout moment un arbre raisonnablement équilibré.

Toutes les opérations insertion, suppression,… sur un arbre AVL avec N noeuds en O(log N) (en moyenne et dans le pire cas!)

Les AVL

- **Définition**

  Un AVL est un arbre binaire de recherche tel que, pour tout nœud de l'arbre, les hauteurs des sous-arbres gauches et des sous-arbres droits diffèrent **d'au plus** de 1.

❖ Remédier au problème de déséquilibre dans les arbres binaires de recherche.

❖ Ce sont des arbres de recherche équilibrés dont les opérations de recherche, d'insertion et de suppression ne sont pas coûteuses en terme de complexité.

Les AVL

- **Terminologie**

  Un AVL est un ABR tel que la différence des hauteurs du sous-arbre gauche et droit de la racine est d'au plus 1 et les sous-arbres gauche et droit sont des AVL

Principe

Nœud

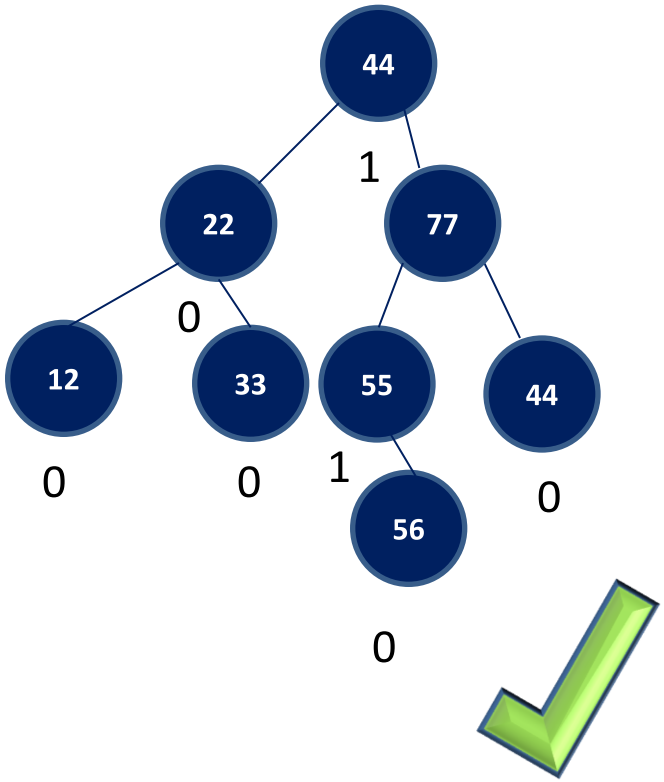
fd	info	BAL	fg

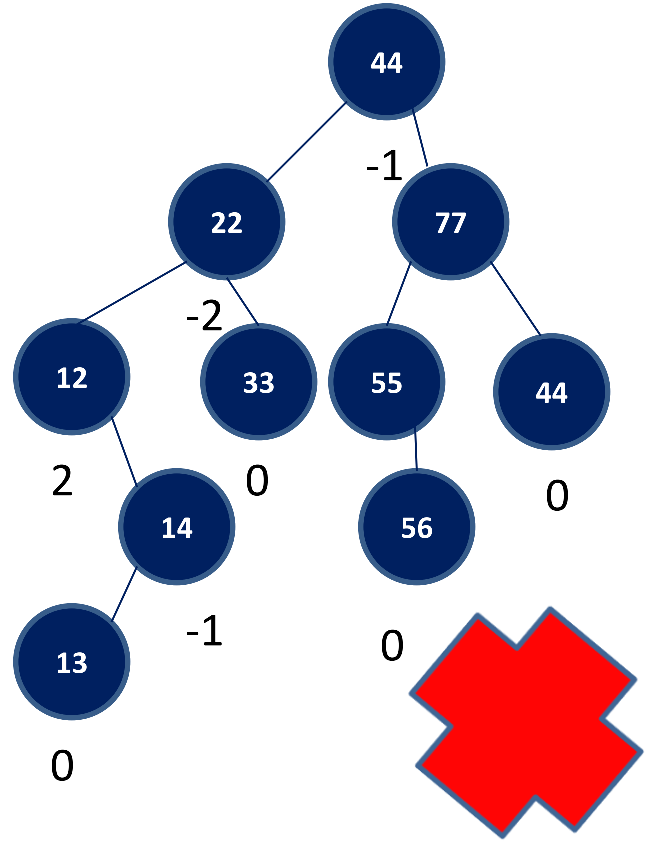
BAL = hauteur(sous arbre droit) – hauteur(sous arbre gauche)

BAL =0
BAL=1
BAL=-1

Les AVL

Exemples





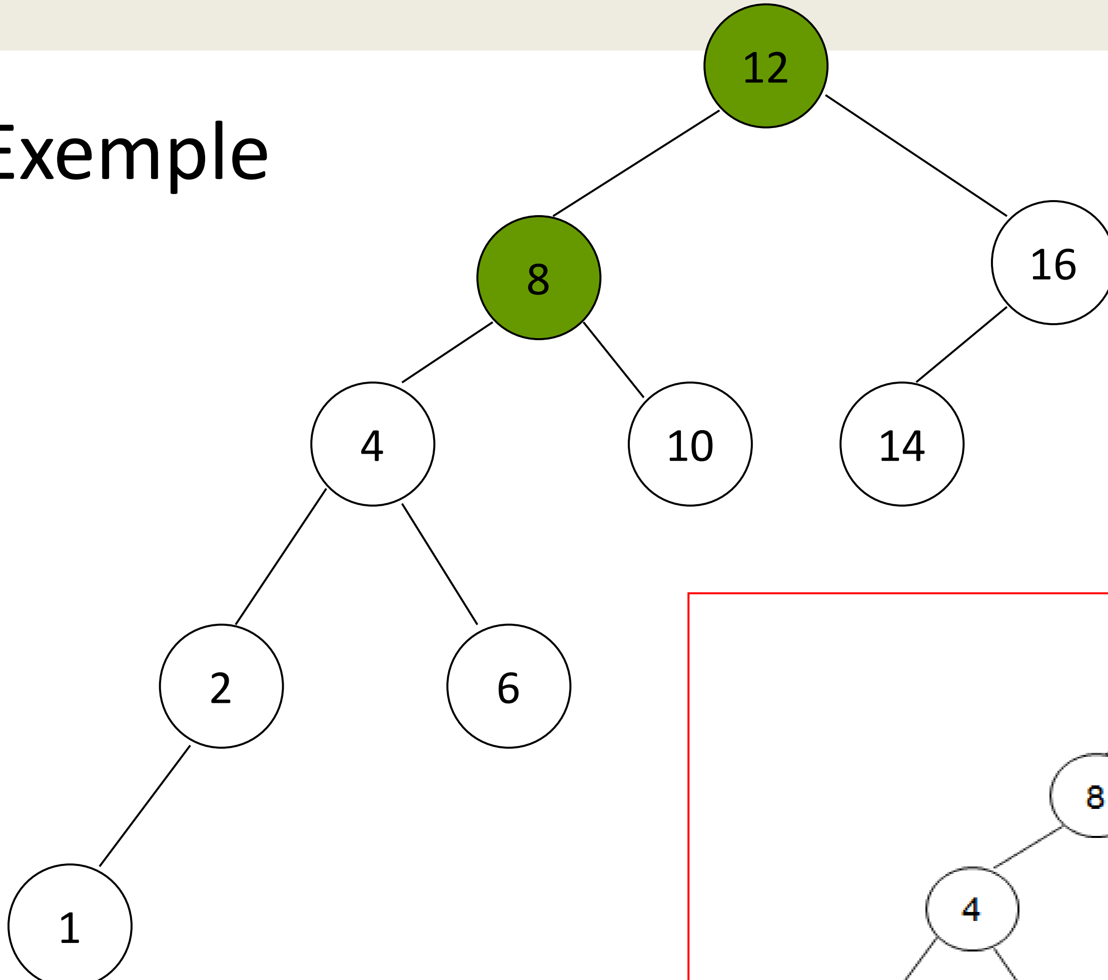
Les AVL

Terminologies

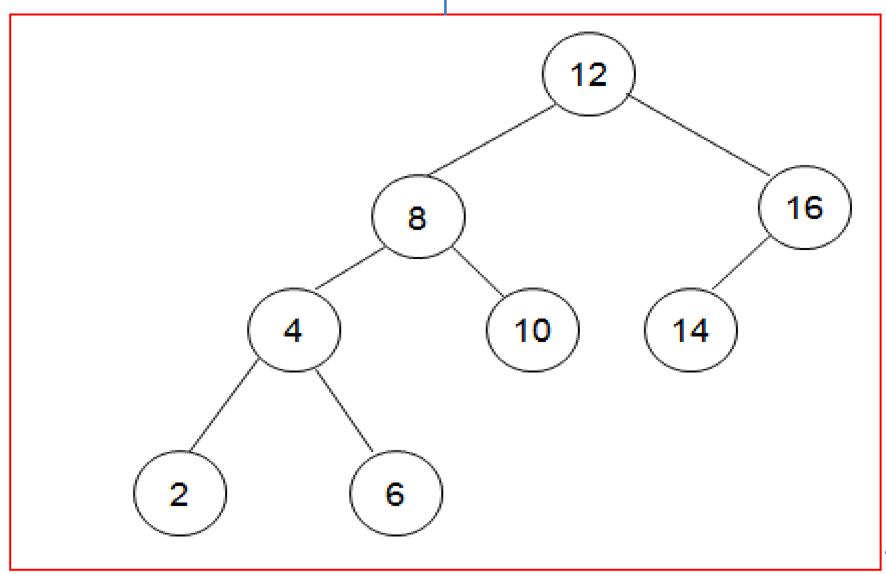
- Un arbre AVL n'est ni un arbre plein ni un arbre niveau-min.
- Insertions et suppressions sont effectuées de la même manière que pour les ABR. Après chaque opération, *on a besoin de vérifier la propriété d'AVL, car l'arbre peut ne plus l'être !*

Les AVL

- Exemple



Ces noeuds violent la condition



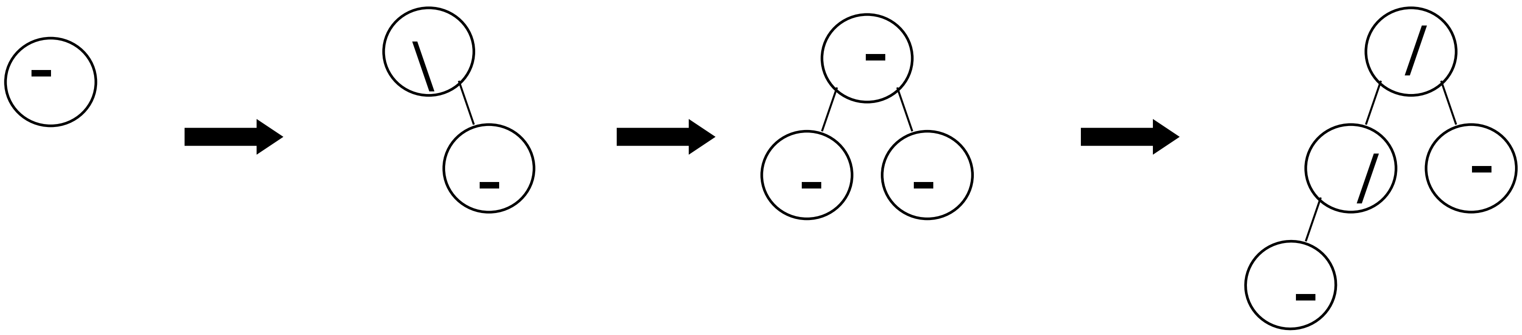
Les AVL

- Il faut, après chaque insertion ou retrait, rétablir l'équilibre s'il a été rompu par l'opération
- <u>Observation importante</u>: après une insertion, seuls les noeuds qui sont sur le chemin du point d'insertion à la racine sont susceptibles d'être déséquilibrés
- Deux cas:
  - insertion dans le sous-arbre de gauche du fils gauche ou dans le sous-arbre de droite du fils droit:
    - $\Rightarrow$ Simple rotation
  - insertion dans le sous-arbre de droite du fils gauche ou dans le sous-arbre de gauche du fils droit:
    - $\Rightarrow$ Double rotation

Les AVL

Les opérations sur les AVL

- Après une insertion, si l'arbre est un AVL alors on ne fait rien.
- Comme sur l'exemple ci-dessous:



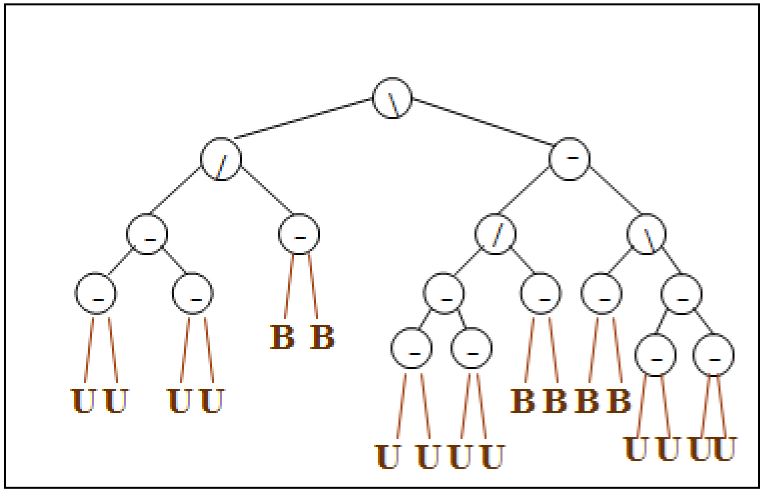
Les AVL

- **Insertion dans un AVL**
- L'arbre devient déséquilibré si l'élément ajouté est le descendant gauche (droit) d'un nœud avec un léger déséquilibre gauche (droit). Alors la hauteur de ce sous-arbre augmente.
- Dans ce shéma on note

  U: nouveaux nœuds

  pouvant déséquilibrer l'arbre

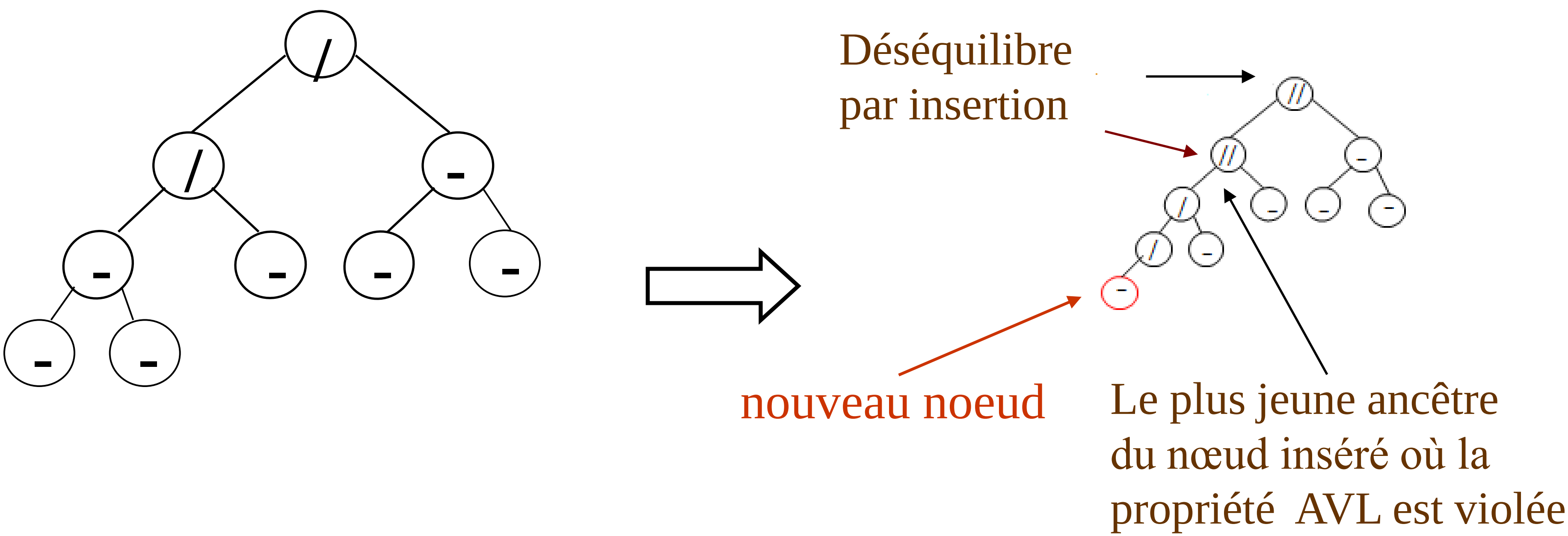
  B: nouveaux laissant équilibré



Les AVL

Insertion dans un AVL

- L'insertion d'un nœud peut provoquer des déséquilibre sur plusieurs nœuds.



Les AVL

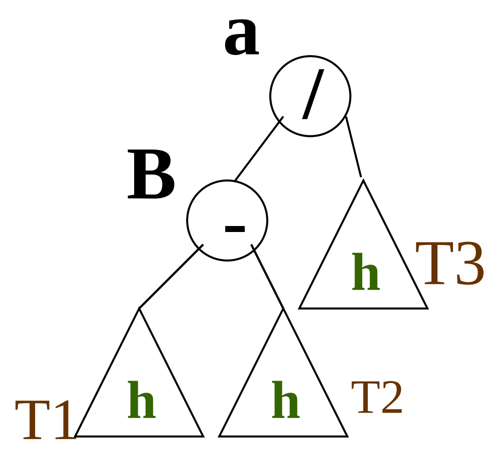
- **Insertion dans un AVL**

Rétablir un arbre AVL déséquilibré après l'insertion en utilisant des **rotations**.

Soit **a** le plus jeune ancêtre où apparaît le déséquilibre

Dans l'arbre AVL, avant l'insertion, T_1, T_2 et T_3 ont une hauteur h

Le même raisonnement peut être utilisé si l'arbre le plus haut est celui de **droite**



Les AVL

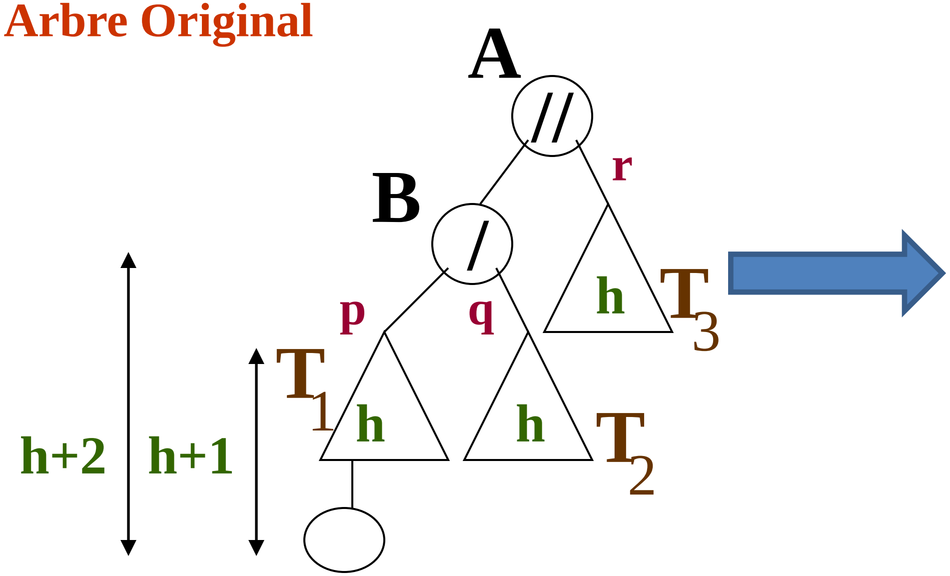
1) Un nouveau nœud est inséré dans T_1

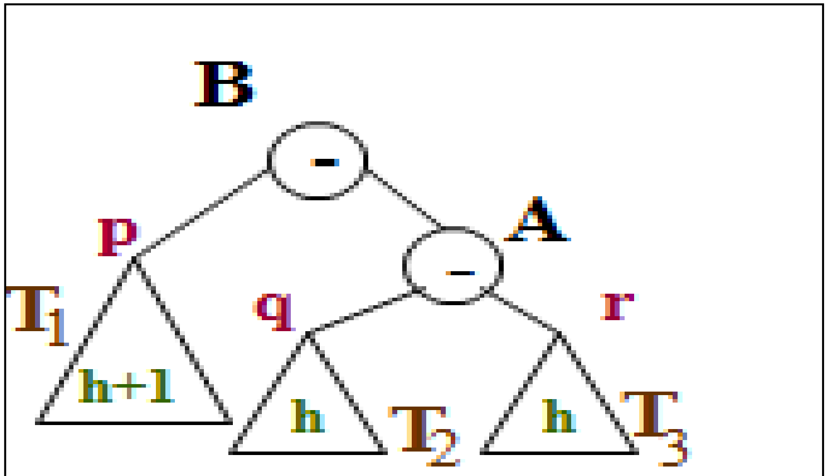
Réequilibrer par **rotation droite:**

P < B < q < A < r

⇨ propriété ABR maintenue !

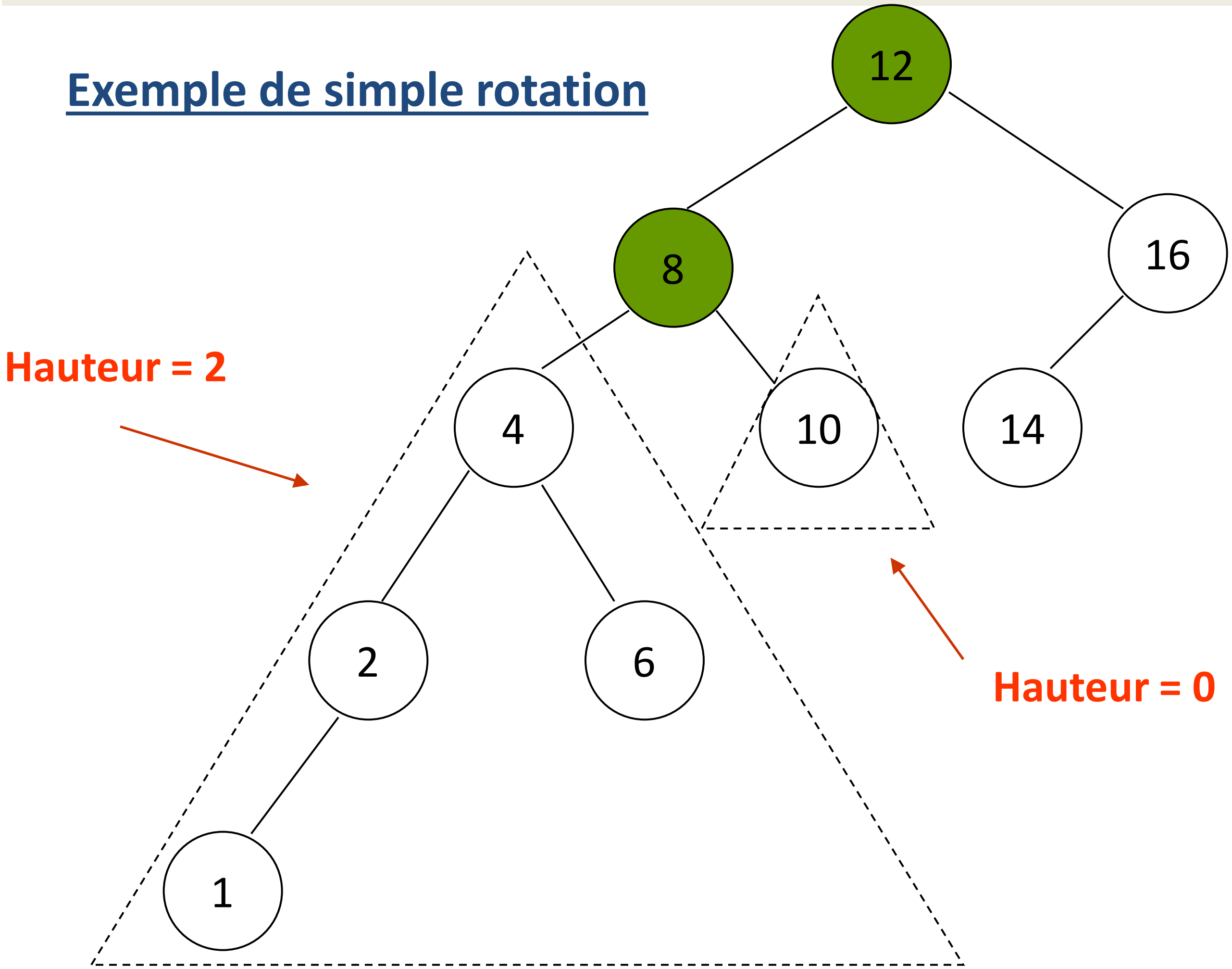
Arbre Original



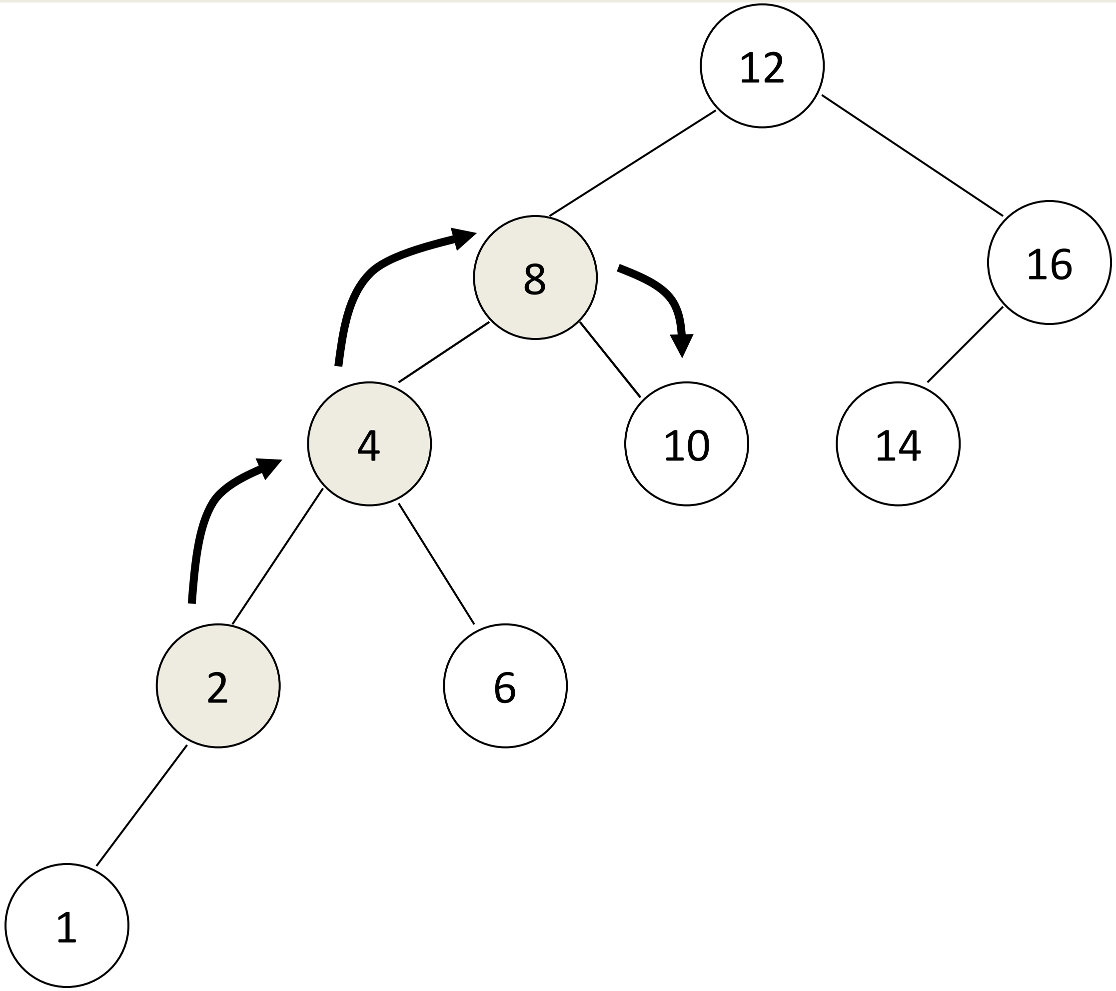


Les AVL

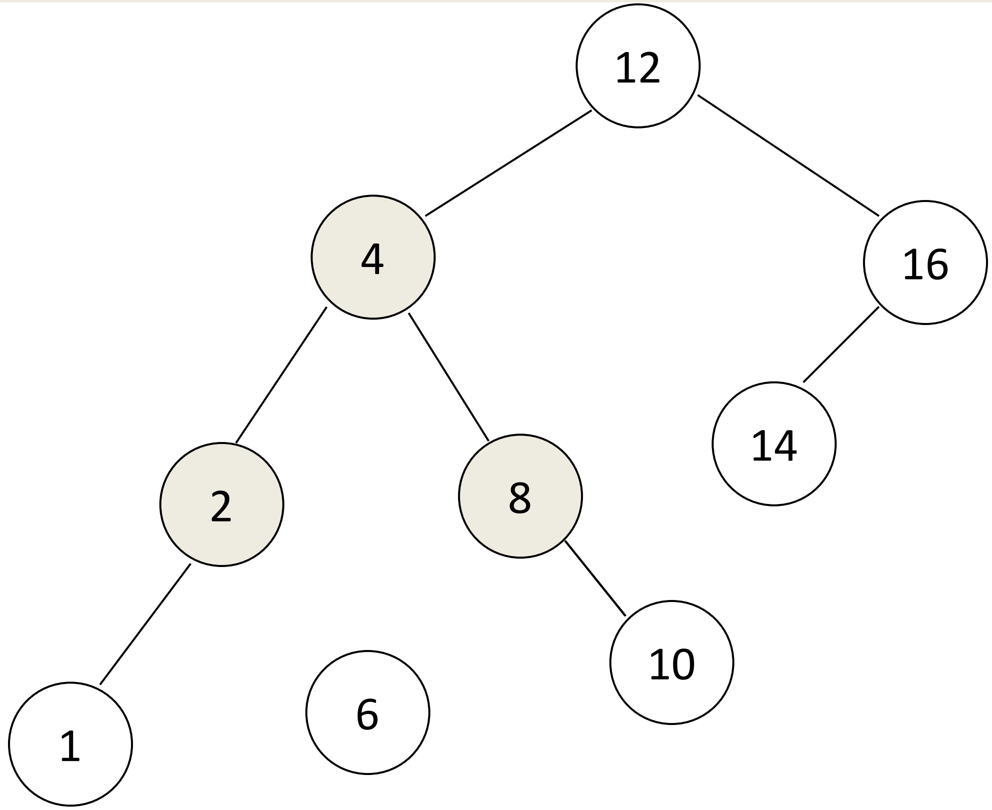
Exemple de simple rotation



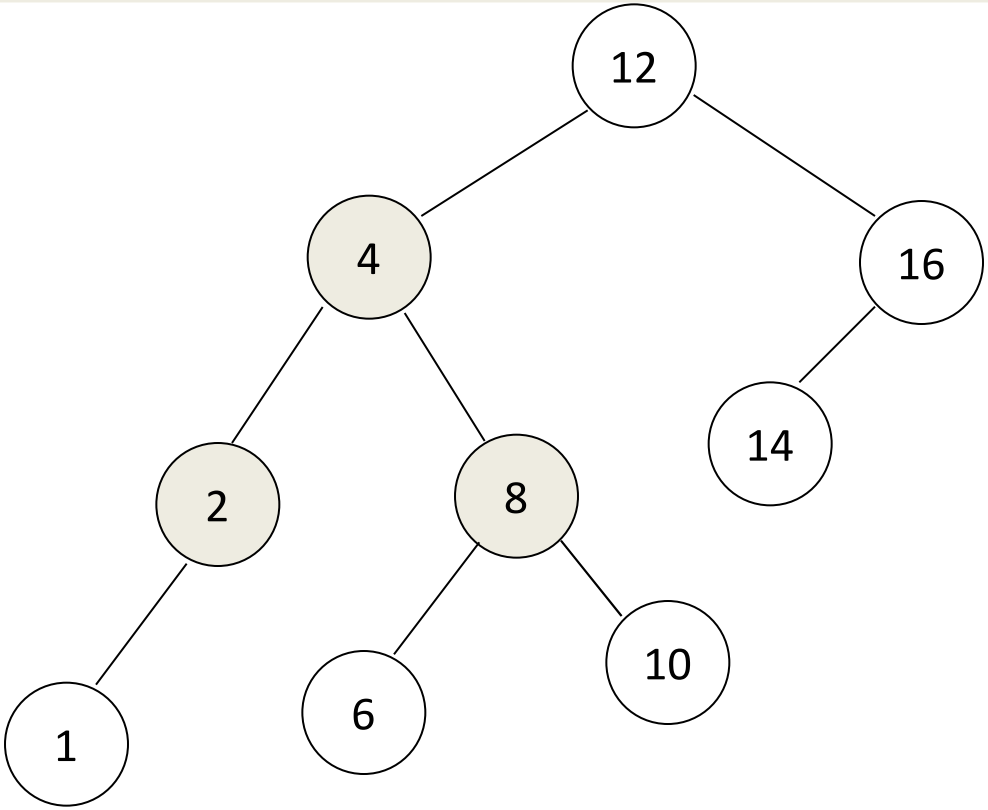
Les AVL



Les AVL



Les AVL

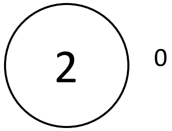


Les AVL

Implementation

- Algorithme récursif.
- Une fois le nœud inséré, en revenant sur notre chemin, il faut vérifier, pour chaque nœud parcouru, les différences de profondeur des sous-arbres gauche et droite.
- La rotation peut être requise à n'importe quel nœud qui se trouve dans le chemin de la racine au point d'insertion.
- Le retrait est passablement plus compliqué que l'insertion (mais demeure **`O(lg N)`**)
- Il y a d'autres types d'arbres équilibrés plus facile à implémenter et plus efficaces.

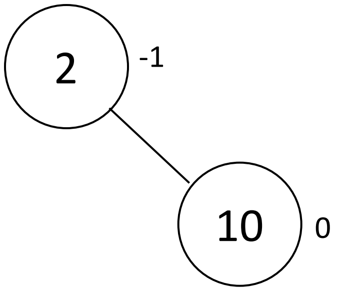
Les AVL



2 10 12 4 16 8 6 14

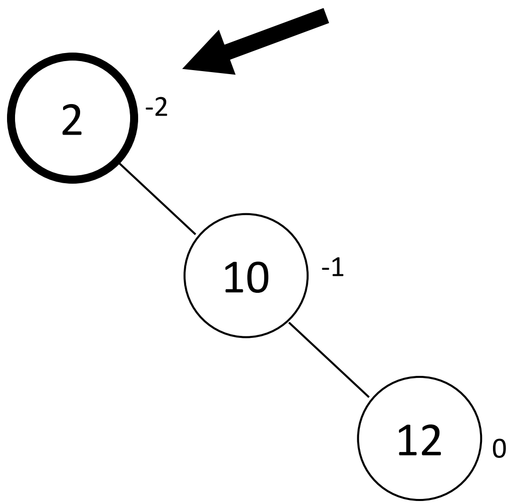
Les AVL

2 10 12 4 16 8 6 14



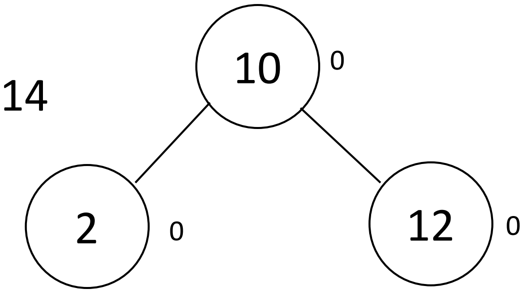
Les AVL

2 10 12 4 16 8 6 14



Les AVL

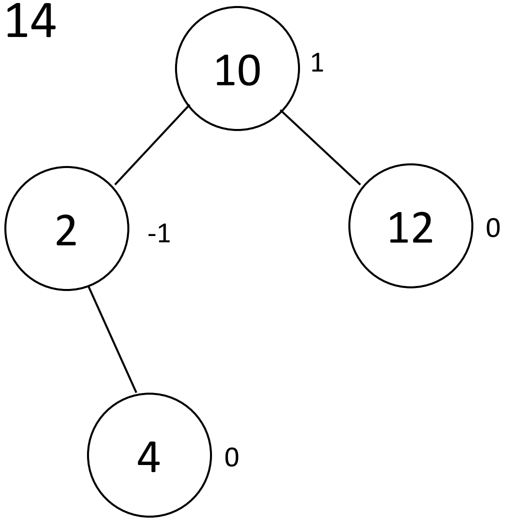
2 10 12 4 16 8 6 14



Rotation simple

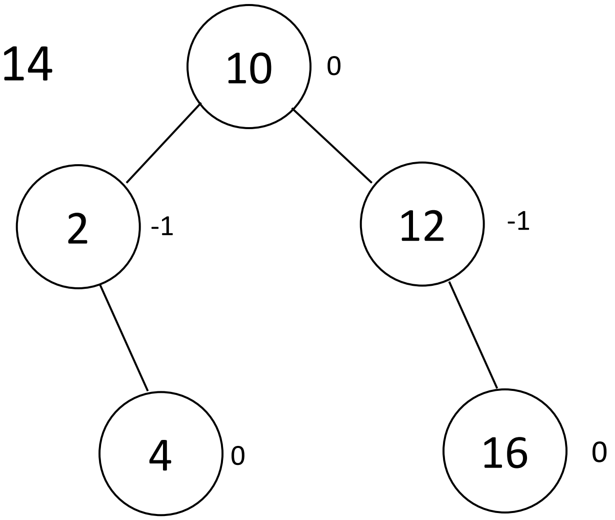
Les AVLs

2 10 12 4 16 8 6 14



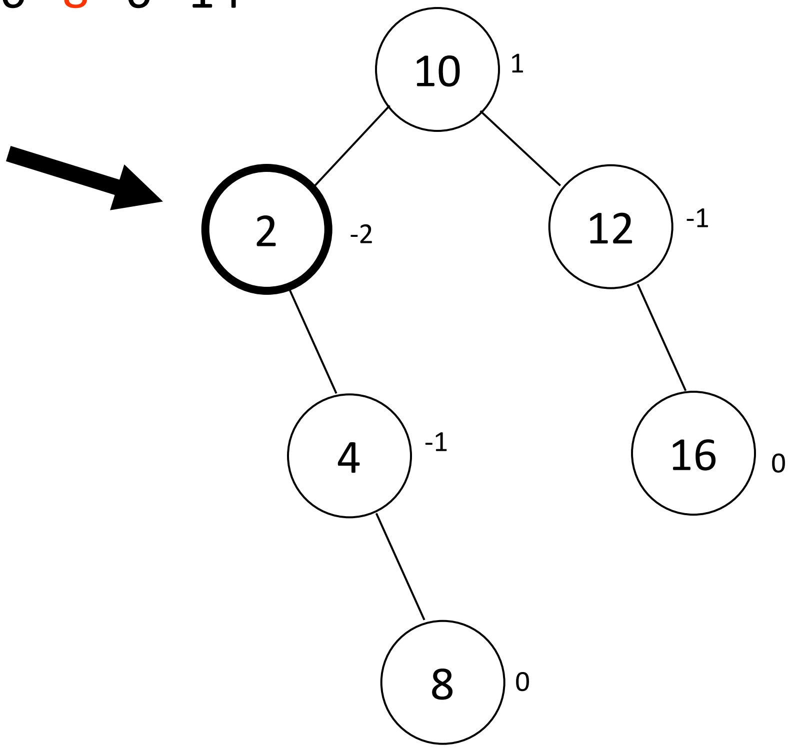
Les AVL

2 10 12 4 16 8 6 14



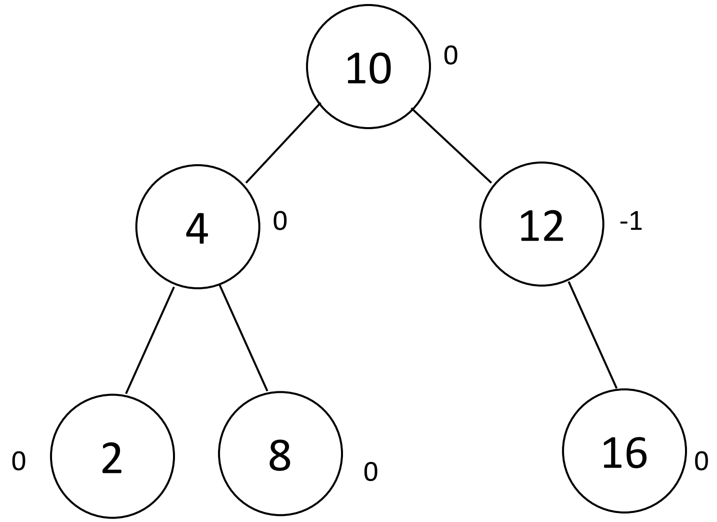
Les AVL

2 10 12 4 16 8 6 14



Les AVL

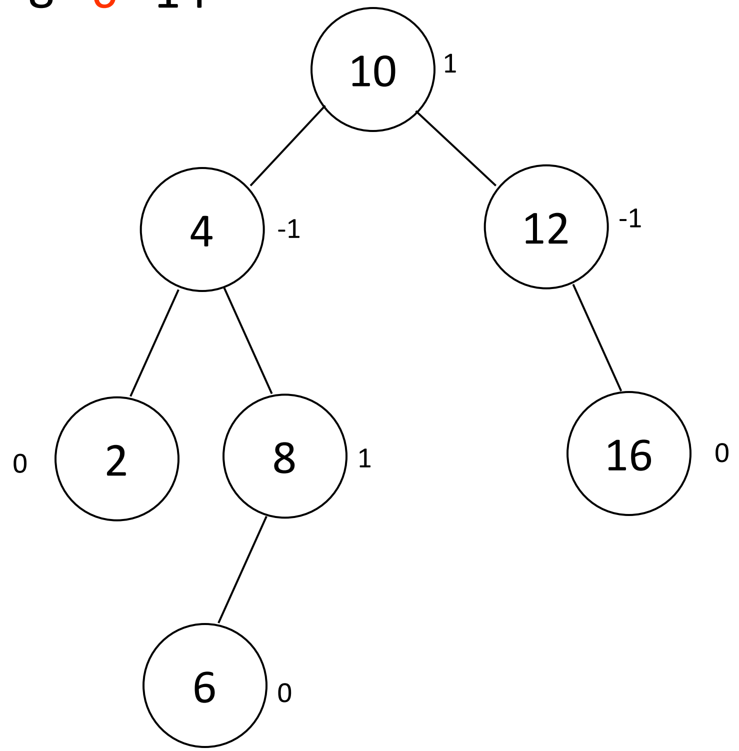
2 10 12 4 16 8 6 14



Rotation simple

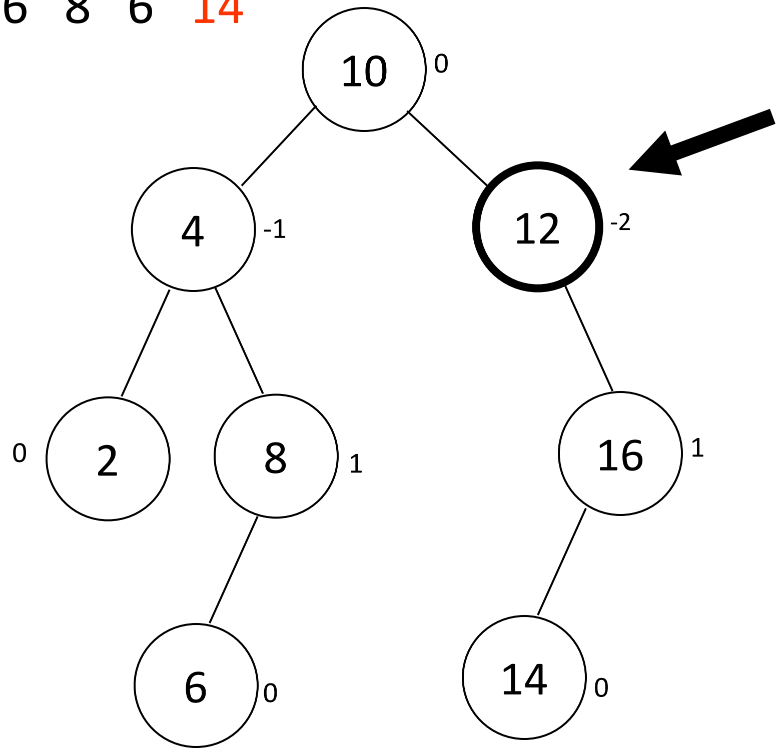
Les AVL

2 10 12 4 16 8 6 14



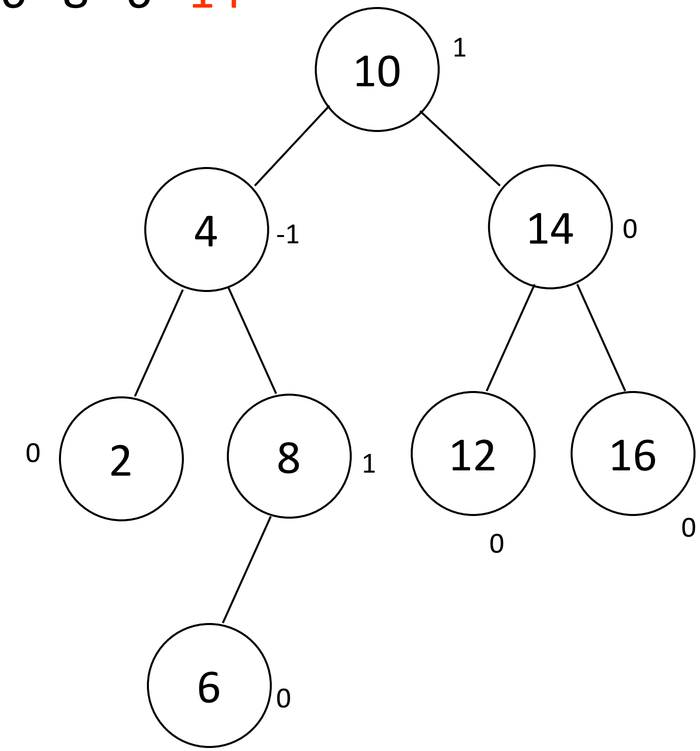
Les AVL

2 10 12 4 16 8 6 14



Les AVL

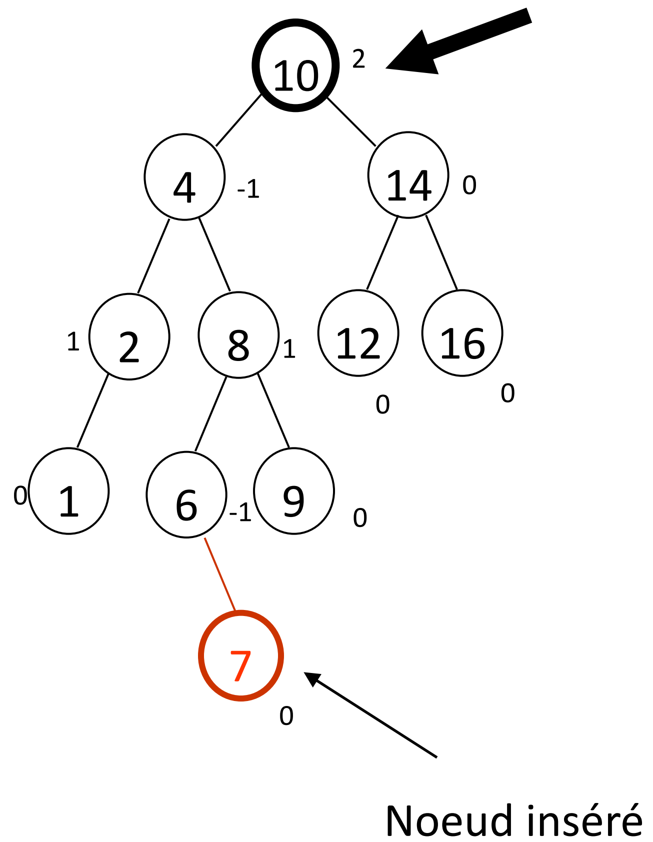
2 10 12 4 16 8 6 14



Rotation double

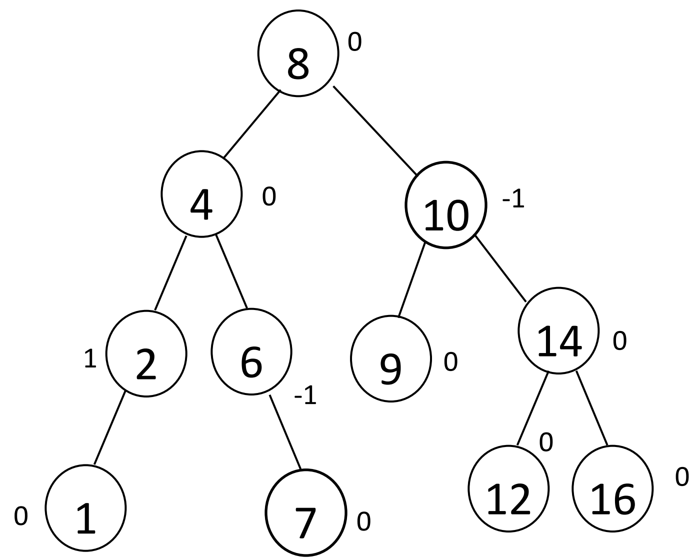
Les AVL

Voici un exemple où la rotation se fait loin du point d'insertion



Les AVL

Voici un exemple où la rotation se fait loin du point d'insertion



Après rotation double

Les arbres rouge & noir

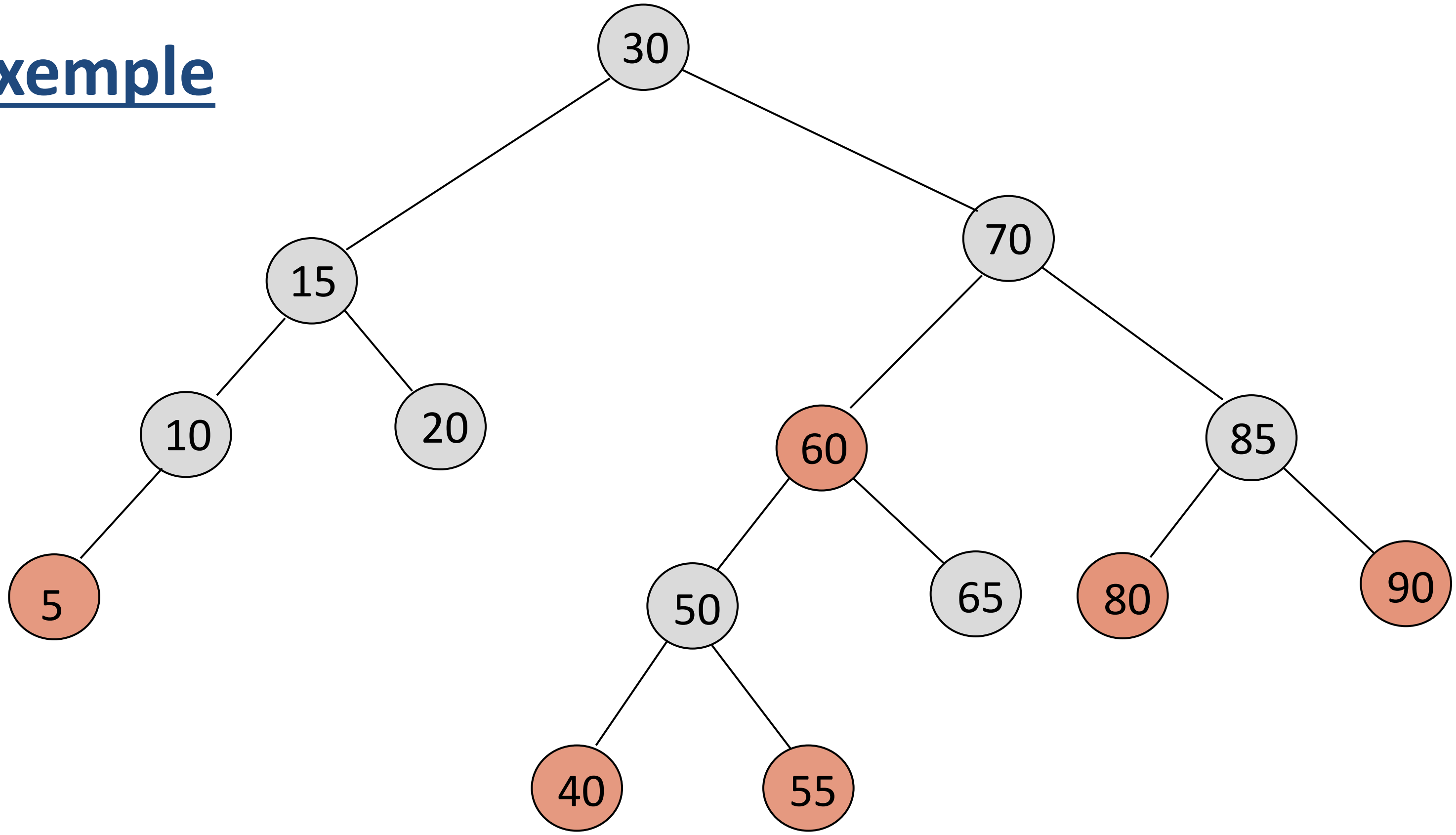
L'arbre a les propriétés suivantes:

1. Chaque nœud est soit rouge soit noir
2. La racine est noire
3. Si un nœud est rouge, tous ses fils doivent être noirs
4. À partir de n'importe quel nœud, tous les chemins de la racine jusqu'à un pointeur NULL doivent avoir le même nombre de noeuds noirs

Comme la racine est noire, et il ne peut y avoir plus de deux noeuds rouges consécutifs, la longueur de tout chemin de la racine à une feuille ne peut être supérieure à 2 fois le nombre de noeuds noirs dans ce chemin.

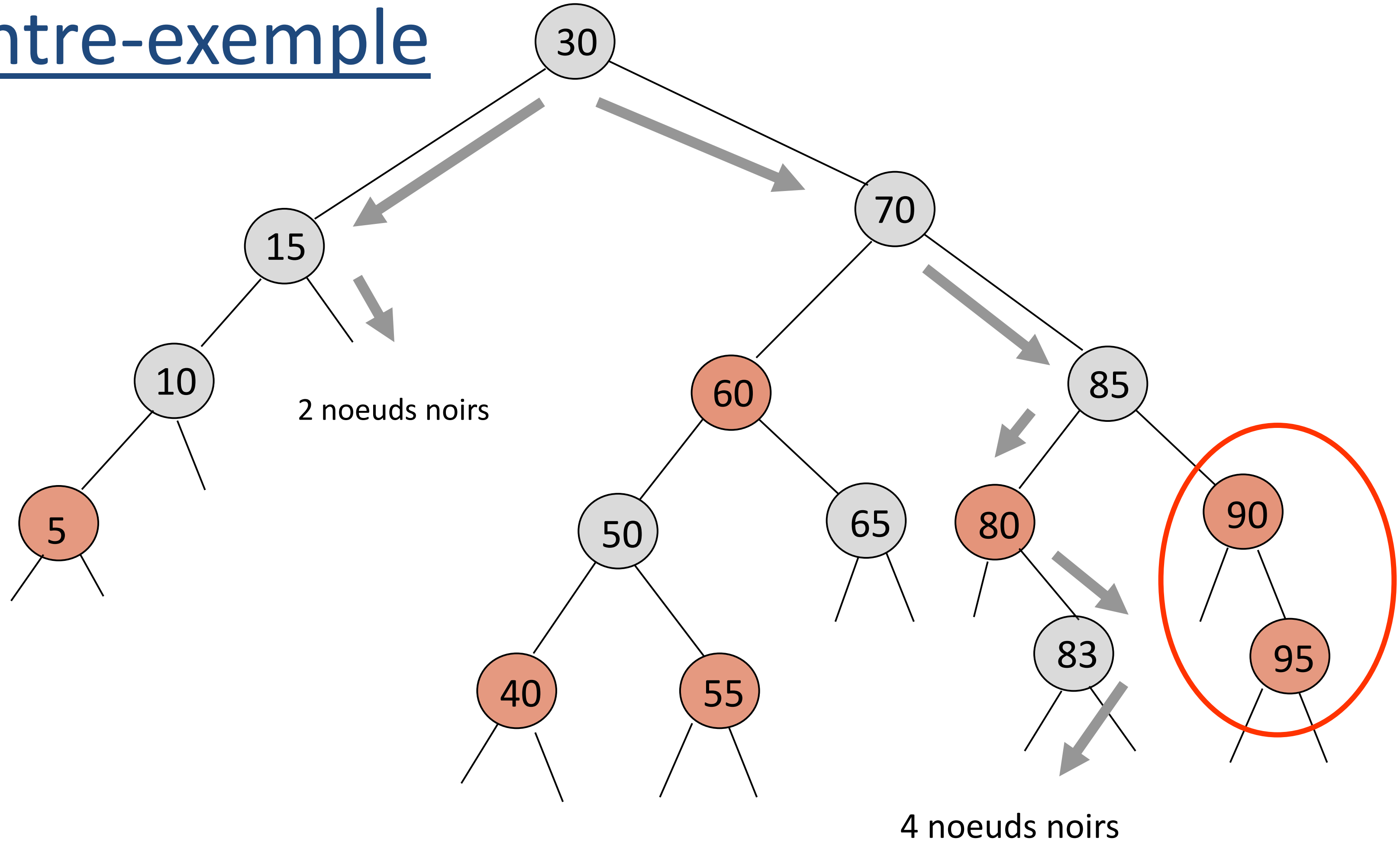
Les arbres rouge & noir

Exemple



Les arbres rouge & noir

Contre-exemple

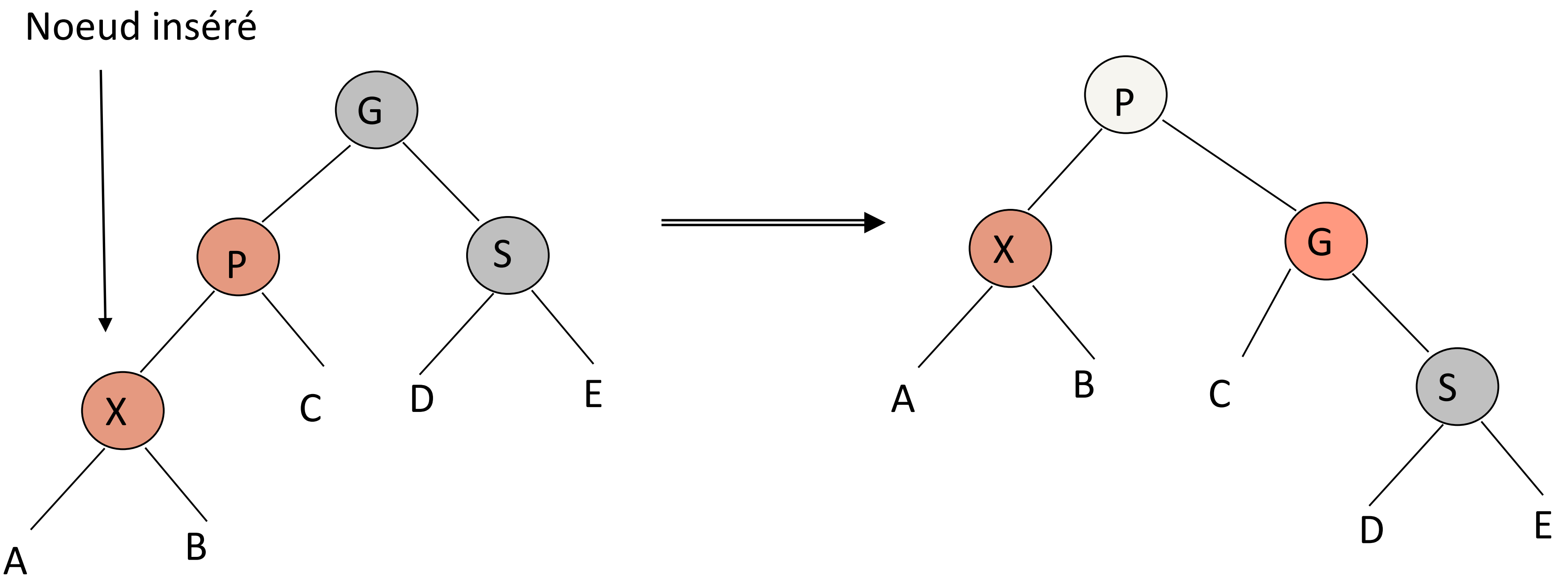


Les arbres rouge & noir

Propriétés

- Un noeud inséré est toujours une feuille.
- On ne peut pas le colorier en noir, puisque cela violerait la condition 4.
- On colore le nœud en rouge.
- Si le père est noir, pas de problème.
- Si le père est rouge, on viole la condition 3. Dans ce cas, on ajuste l'arbre, par le biais de changements de couleurs et de rotations.

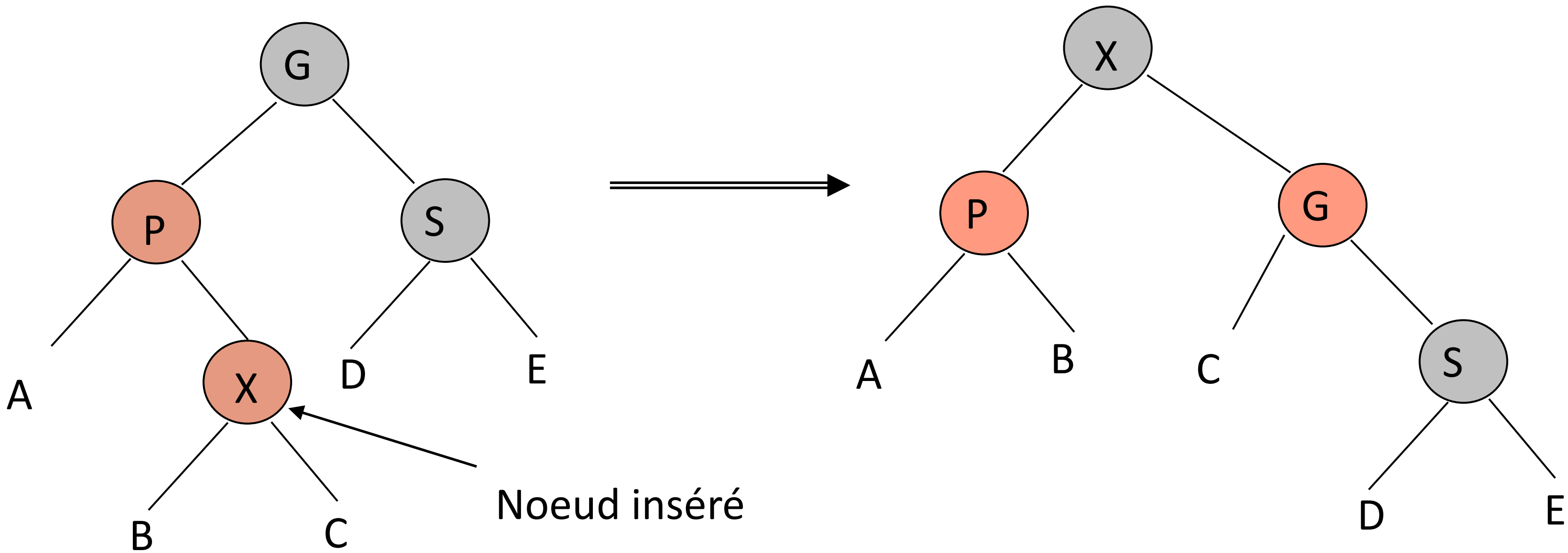
Les arbres rouge & noir



Premier cas: Le frère du nœud parent est noir (on utilise la convention qu'un nœud NULL est noir)

Rotation simple

Les arbres rouge & noir

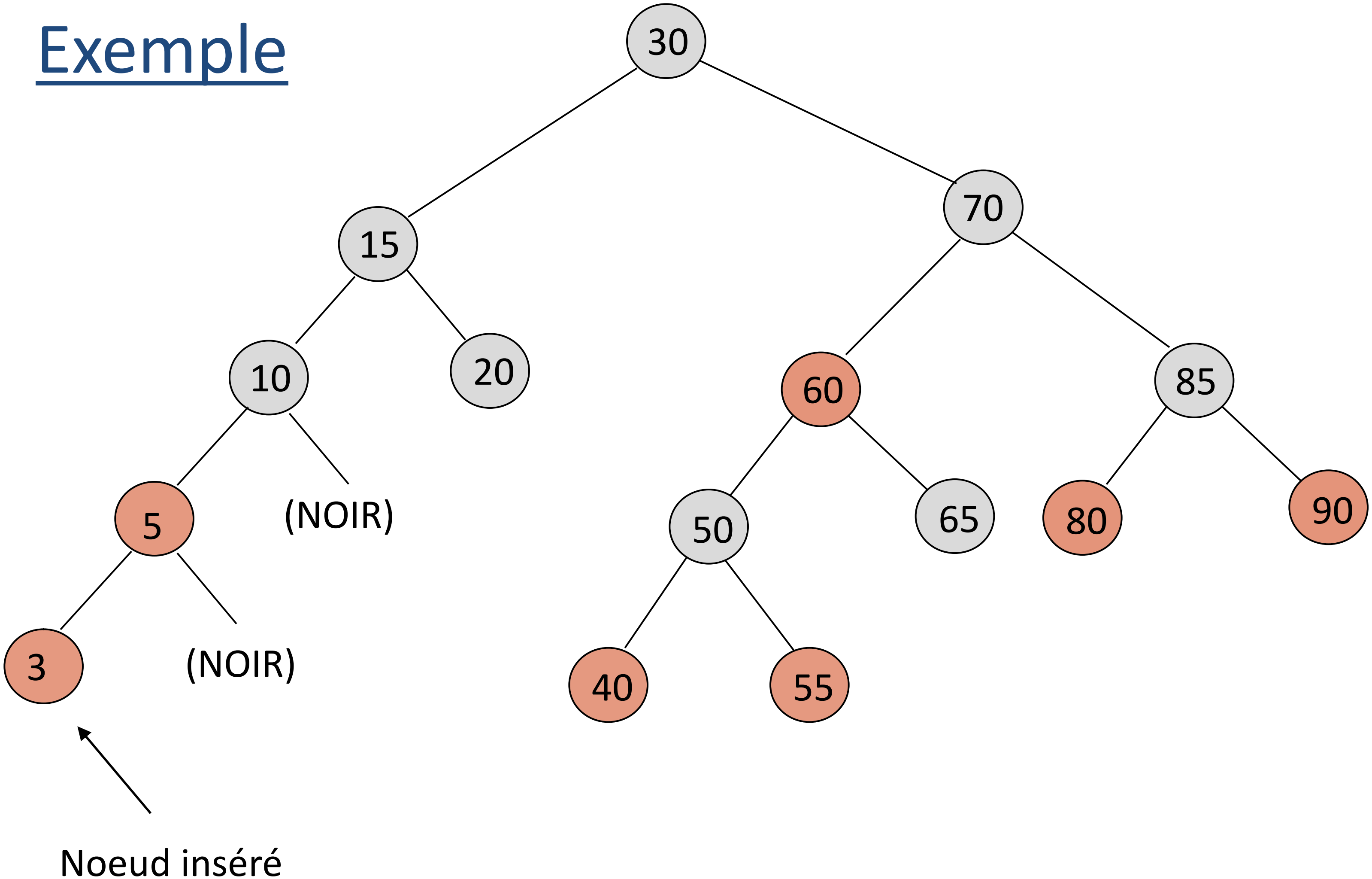


Premier cas: Le frère du nœud parent est noir (on utilise la convention qu'un nœud NULL est noir)

Rotation double

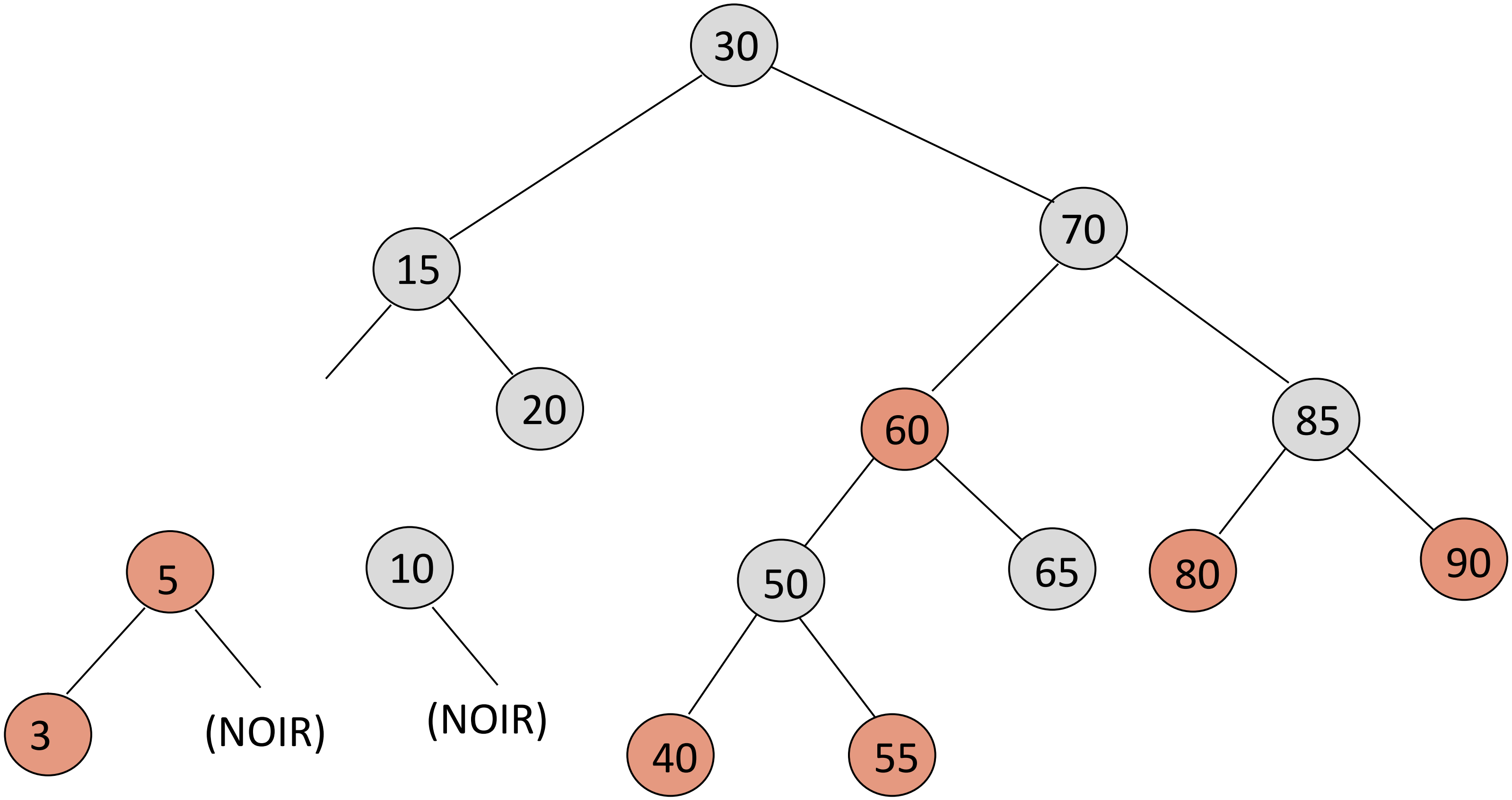
Les arbres rouge & noir

Exemple



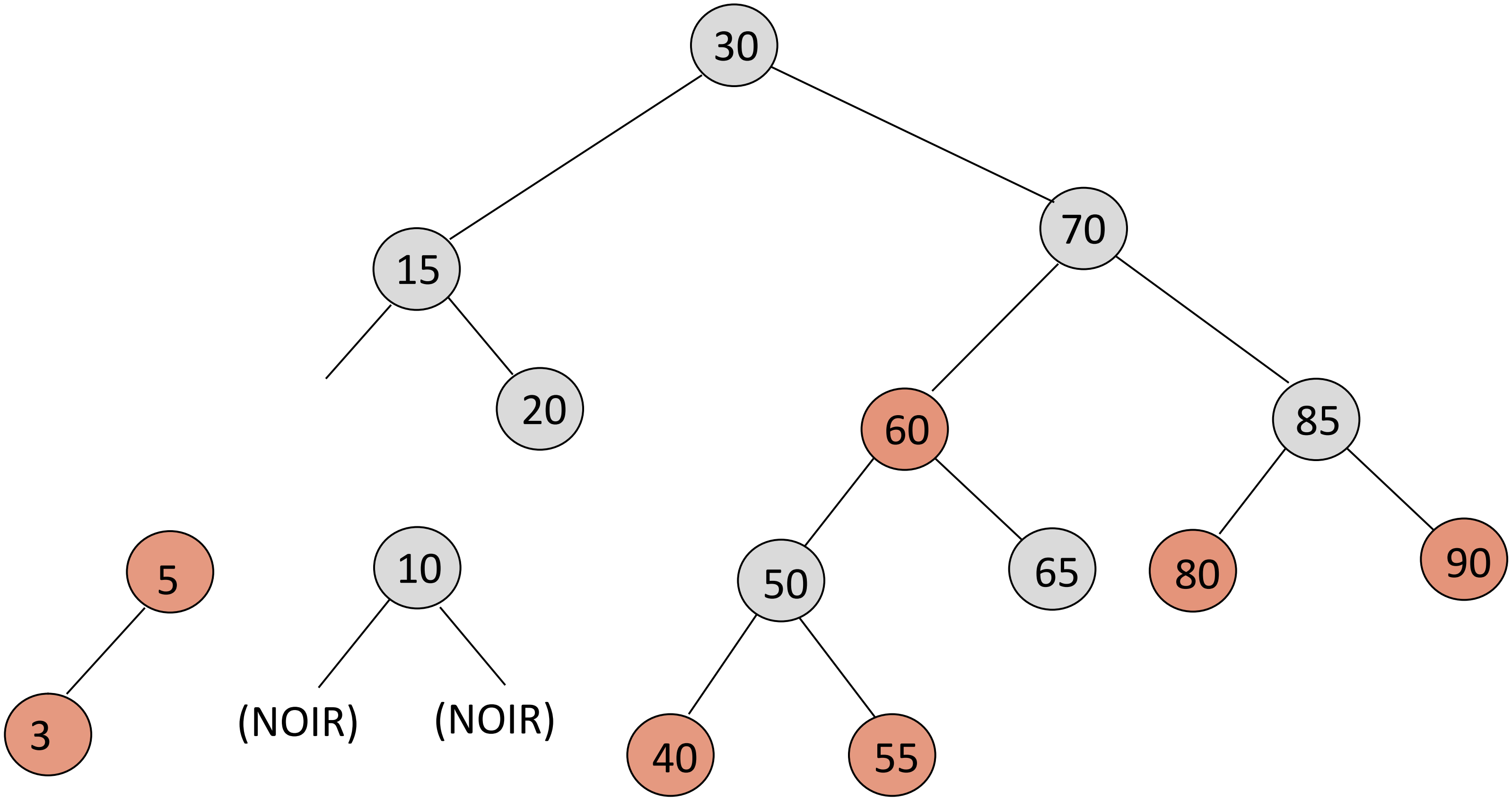
Rotation simple

Les arbres rouge & noir



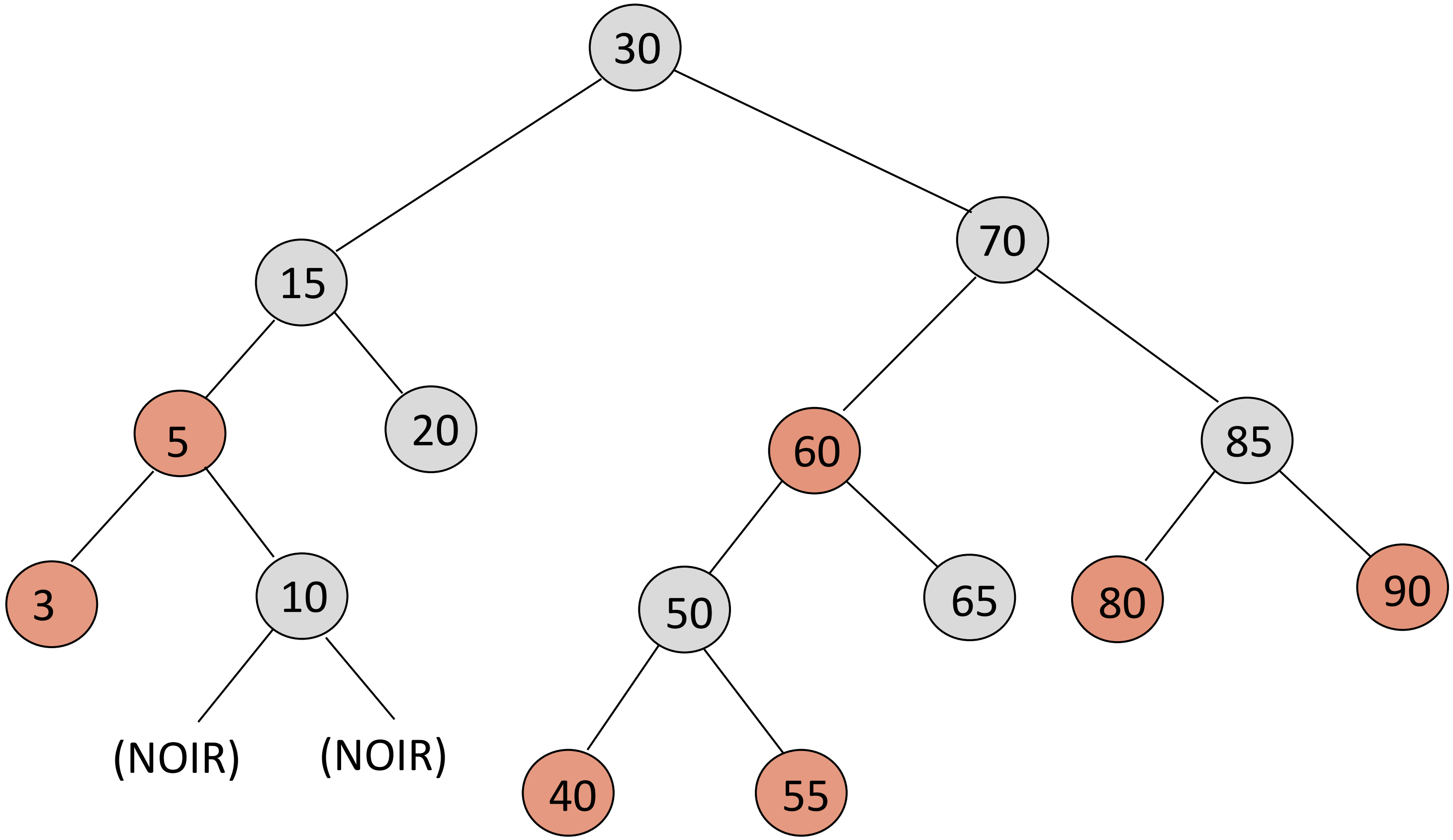
Rotation simple

Les arbres rouge et noir



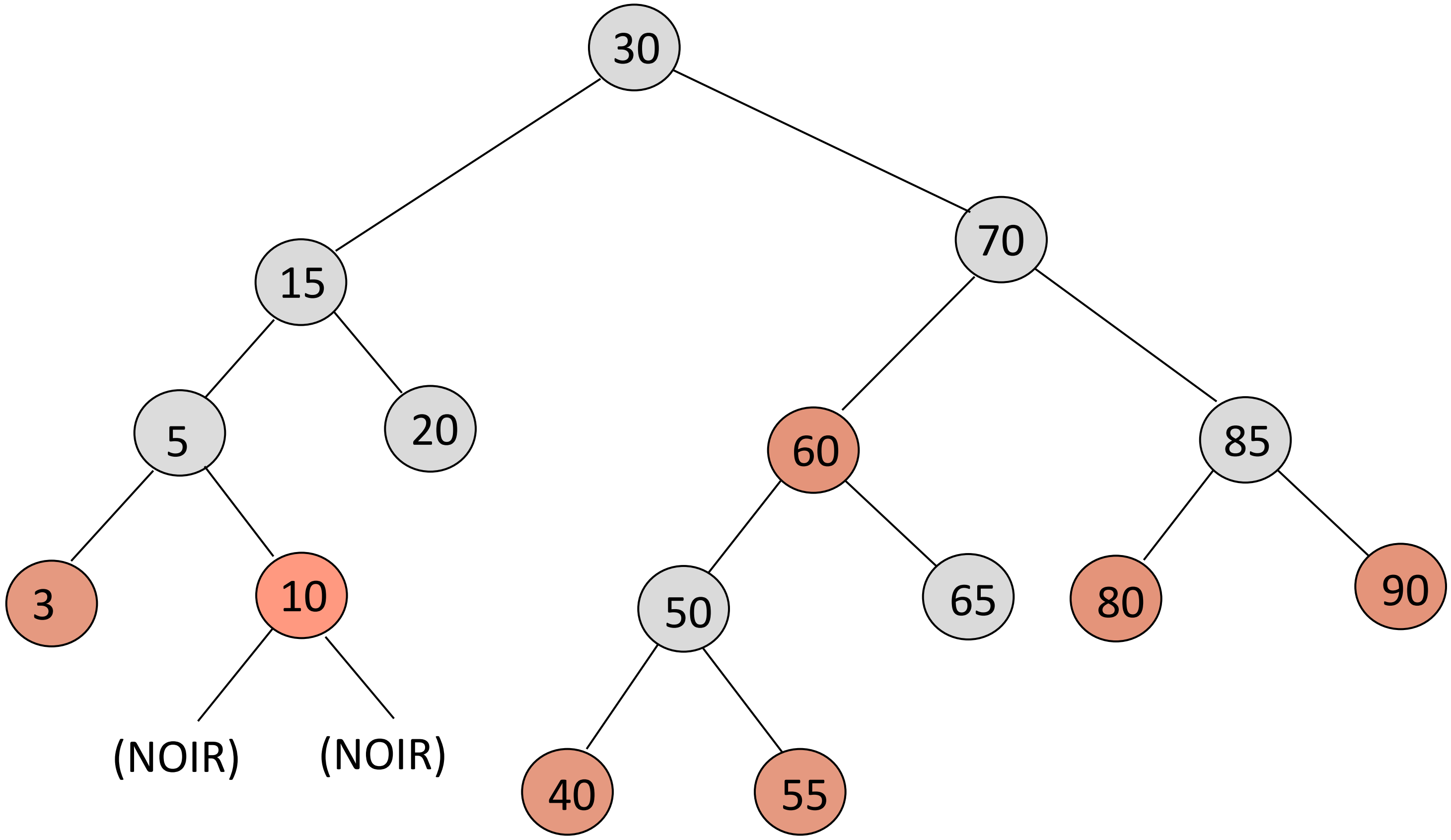
Rotation simple

Les arbres rouge & noir



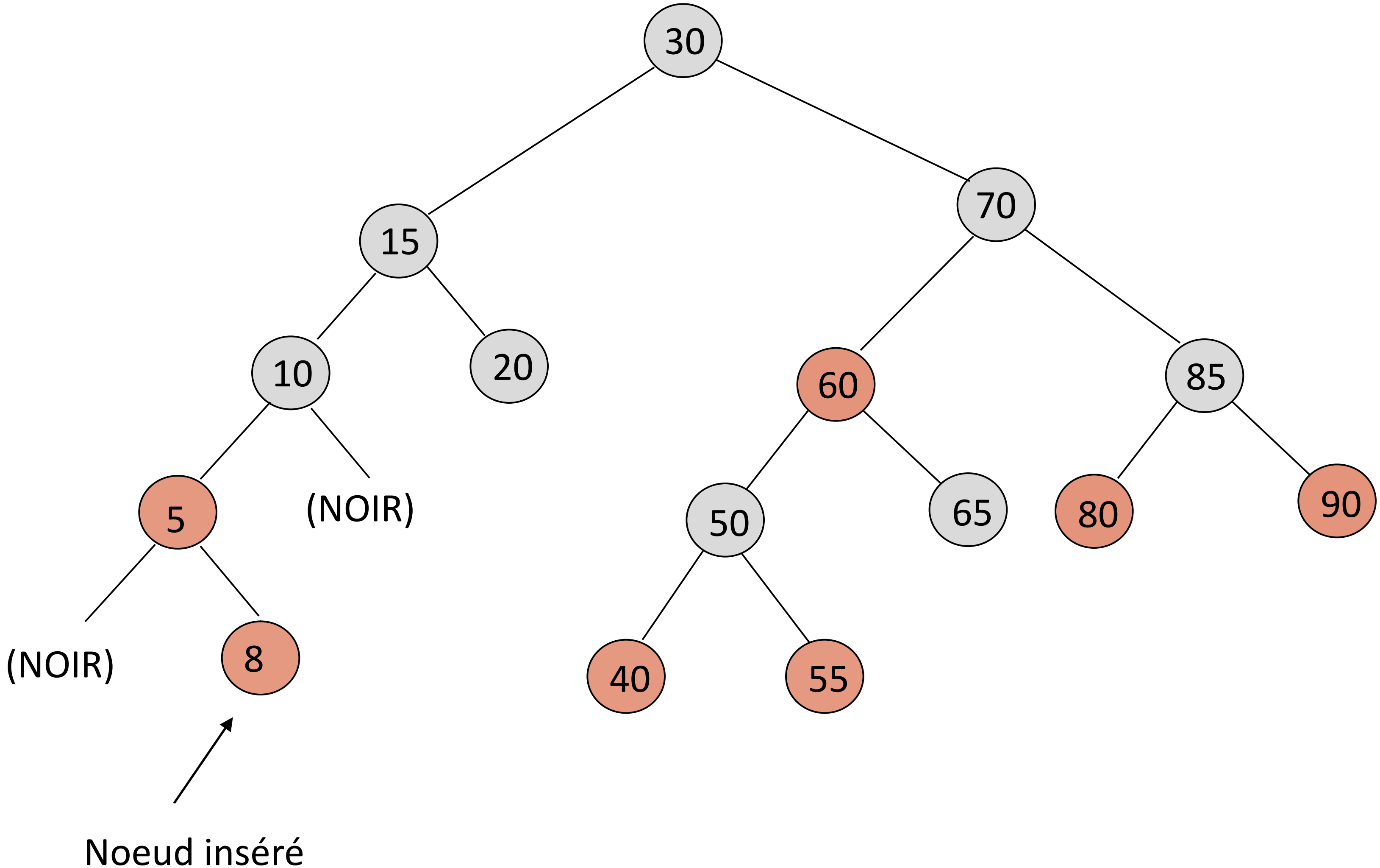
Rotation simple

Les arbres rouge & noir



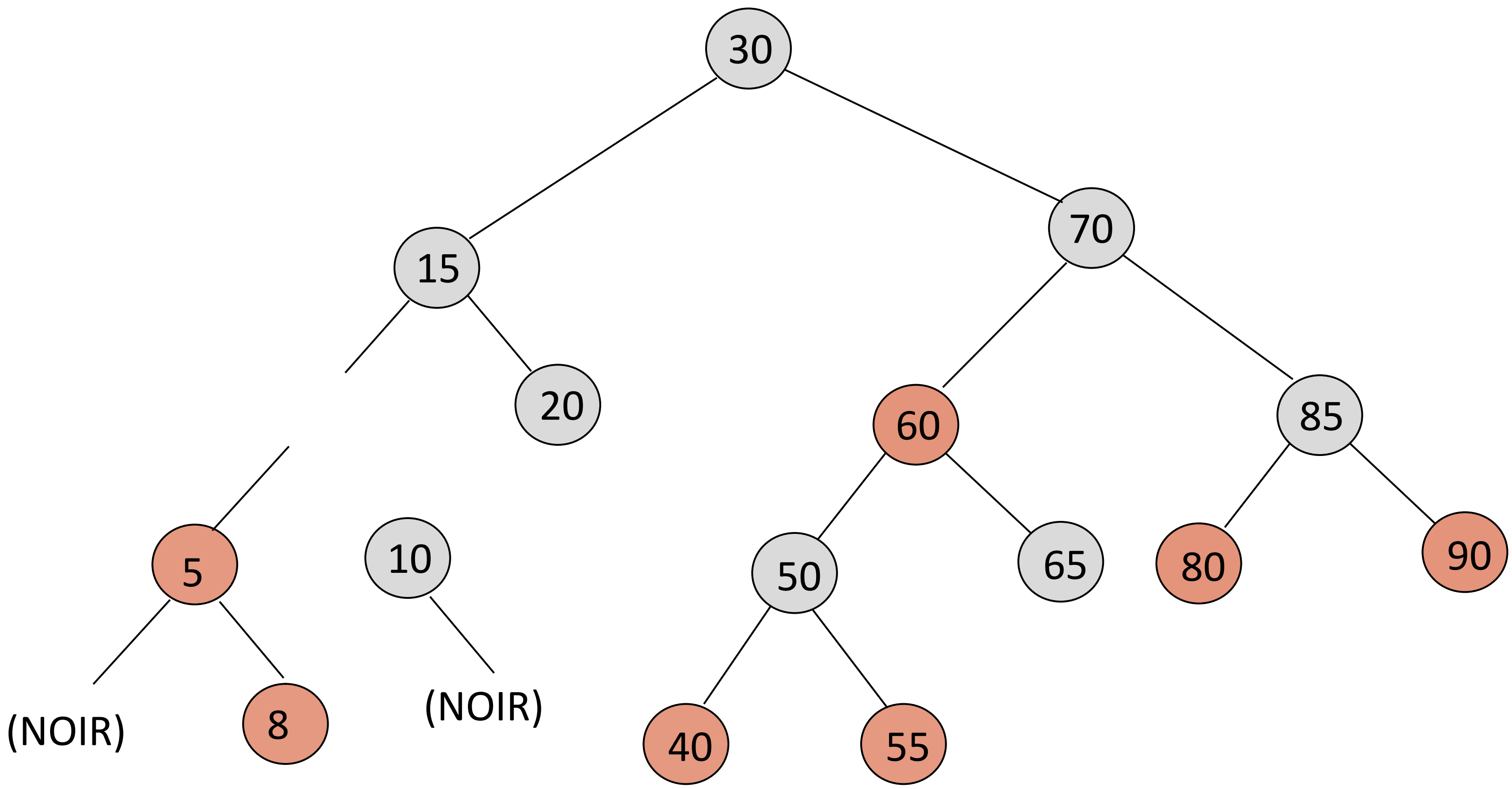
Rotation simple

Les arbres rouge & noir



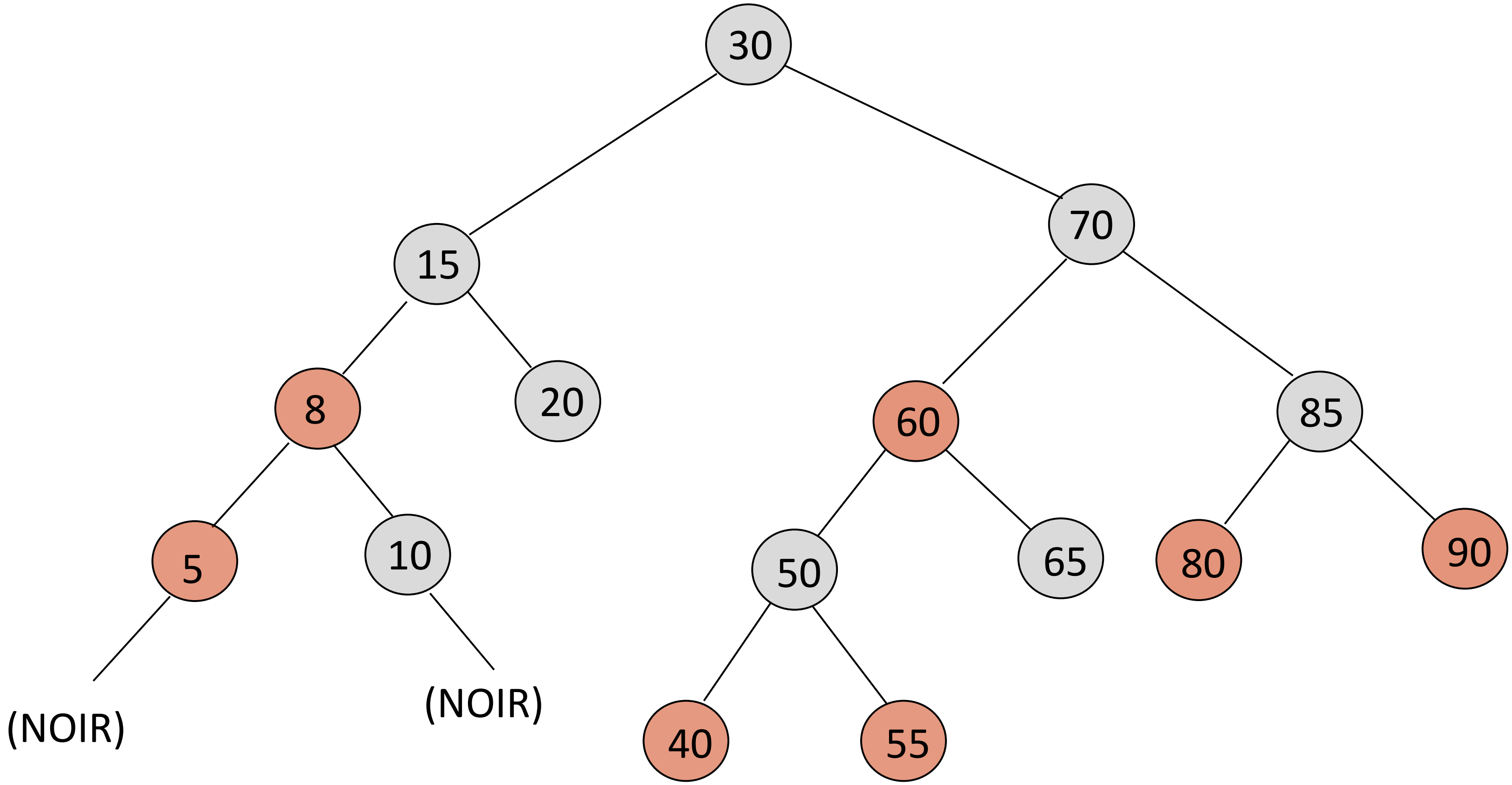
Rotation double

Les arbres rouge & noir



Rotation double

Les arbres rouge & noir



Les arbres rouge & noir