TP N°2: OPÉRATIONS SUR LES ARBRES ROUGES ET NOIRS

PRÉPARER PAR

IDRISS ABDELMAGHNI KEBLADJ.
BOUCHACHI MEHDI MOHAMED.

RESPONSABLE DU TP:

MME S.AROUSSI

Visual graph tree

ENVIRONNEMENT **UTILISÉ**

👉 **Processeur (CPU)**	**I7 5eme Gen**	*Assure la rapidité et la stabilité lors du développement et de la compilation.*
👉 **Mémoire vive (RAM)**	**16 GB DDR4**	*Permet d'exécuter plusieurs outils et serveurs sans ralentissement.*
👉 **système d'exploitation**	Windows 10	*Fournit un environnement fiable et compatible pour le développement web.*
👉 **Langage de programmation**	JavaScript	*Utilisé pour créer des interfaces interactives et dynamiques.*
👉 **Environnement**	Visual Studio Code	*Outil principal d'édition et de gestion du code source.*
👉 **Bibliothèques**	**React**	*Ensemble de modules facilitant la création d'applications réactives.*
👉 **Other tools**	**React icons/ React hot toast/ Styled-components/ Assistant Utilisée, ChatGPT 20%**	

INTRODUCTION

👉 **Les arbres rouges et noirs** sont **un type d'arbres binaires de recherche** dits **équilibrés.**

👉 Ce module présente **leurs définitions**, puis décrit les opérations de **rotation, insertion et suppression, la recherche** étant identique à celle **des arbres binaires de recherche.**

DÉFINITIONS, REPRÉSENTATION

👉 Un **arbre binaire de recherche** est **un arbre rouge et noir** s'il satisfait **les propriétés suivantes:**

1. Chaque nœud est soit rouge, soit noir.
2. Chaque feuille (Nil) est noire.
3. Si un nœud est rouge, alors ses deux fils sont noirs.
4. Tous les chemins descendants reliant un nœud donné à une feuille (dans le sous-arbre dont il est la racine) contiennent le même nombre de nœuds noirs.

DÉFINITIONS, REPRÉSENTATION

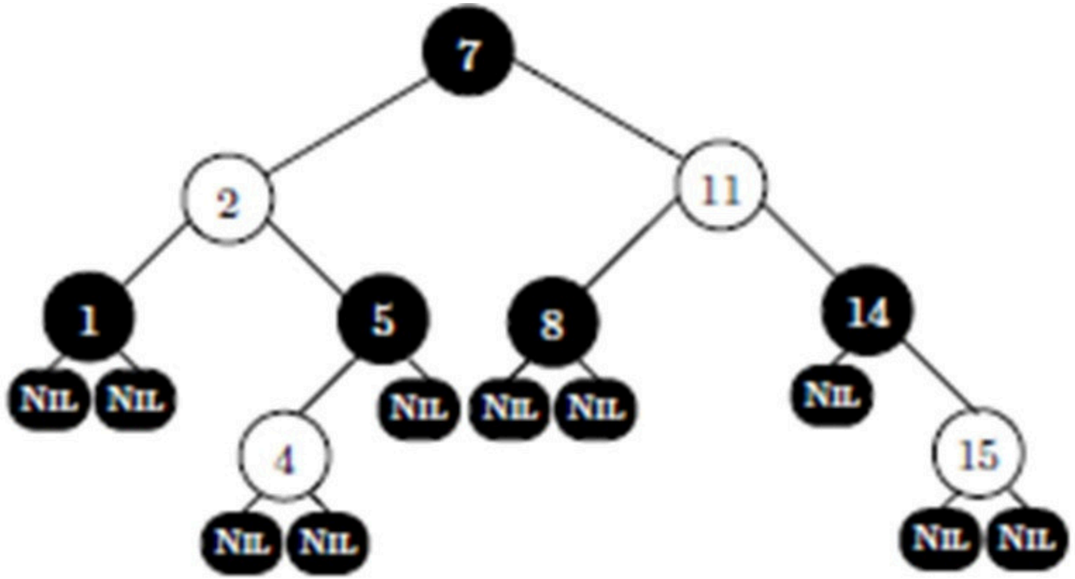
👉 Hauteur noire

- On appelle **hauteur noire** d'un **nœud x**, le **nombre de nœuds noirs** sur un **chemin descendant** reliant x à une **feuille.**

💡 Example:

👉 Lafigure présente un **exemple d'arbre rouge et noir.**

👉 **NOTE: les nœuds noirs** sont à **fond noir** et **les nœuds rouges** à **fond blanc.**



DÉFINITIONS, REPRÉSENTATION

👉 Fonctionnalités

En plus de celles de **l’arbre binaire de recherche:**

1. Méthodes liées à la couleur
2. Méthodes du grand-père

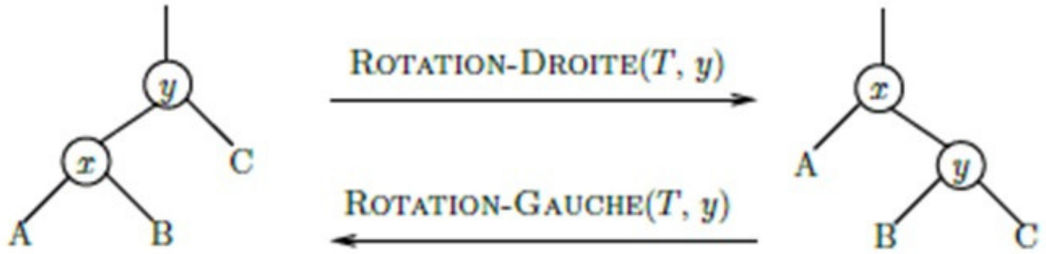
👉 Représentation

Un **nœud** est représenté par : **étiquette**, **fils gauche**, **fils droit**, **père**, **couleur**.

ROTATIONS

👉 Les rotations

La figure présente les transformations d'arbres binaires appelées rotations, utilisées pour rééquilibrer la structure de l'arbre.

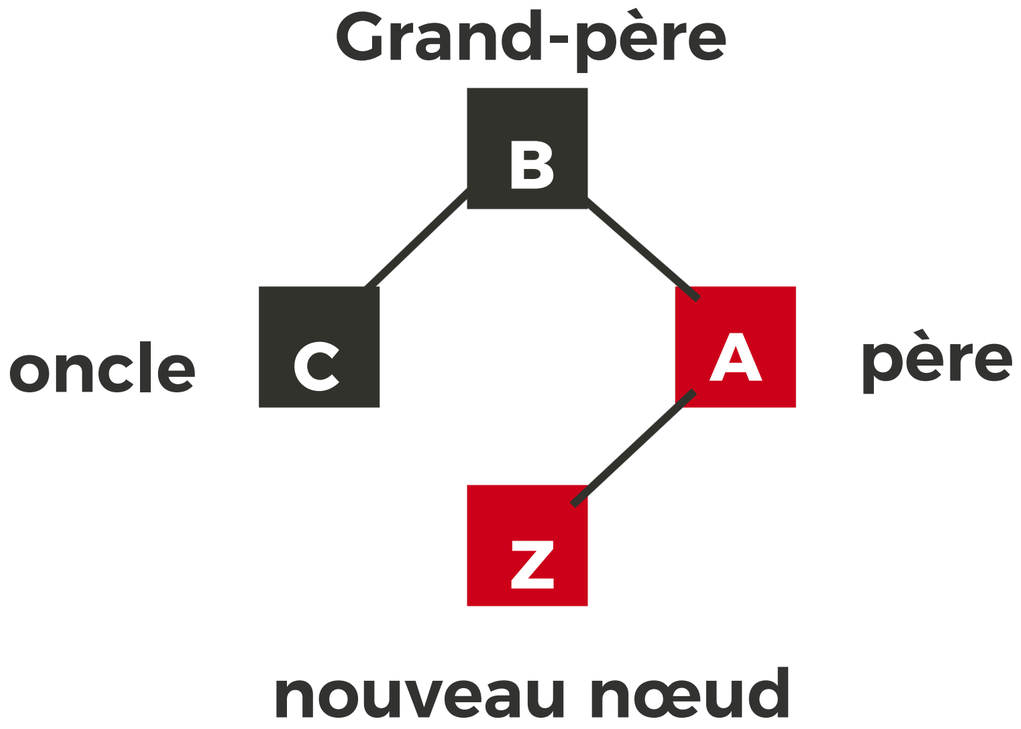


❗❗Attention

Les rotations conservent la propriété des arbres binaires de recherche, mais pas nécessairement celle des arbres rouges et noirs.

INSERTION D'UN ÉLÉMENT

👉 Z Relations



INSERTION D'UN ÉLÉMENT

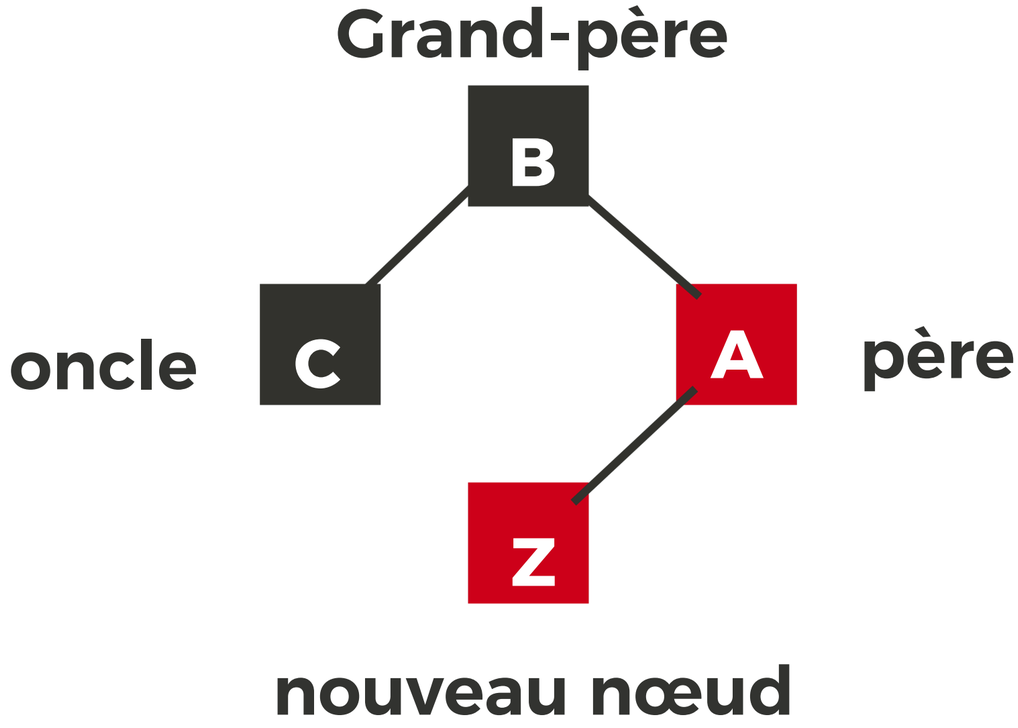
👉 Après l’insertion du nœud (dont la couleur est rouge), il existe plusieurs scénarios possibles :

1□ Z= la racine

2□ Z.oncle = rouge

3□ Z.oncle = noir (triangle)

4□ Z.oncle = noir (ligne)



INSERTION D'UN ÉLÉMENT

👉 **Cas 01:** Z = la racine

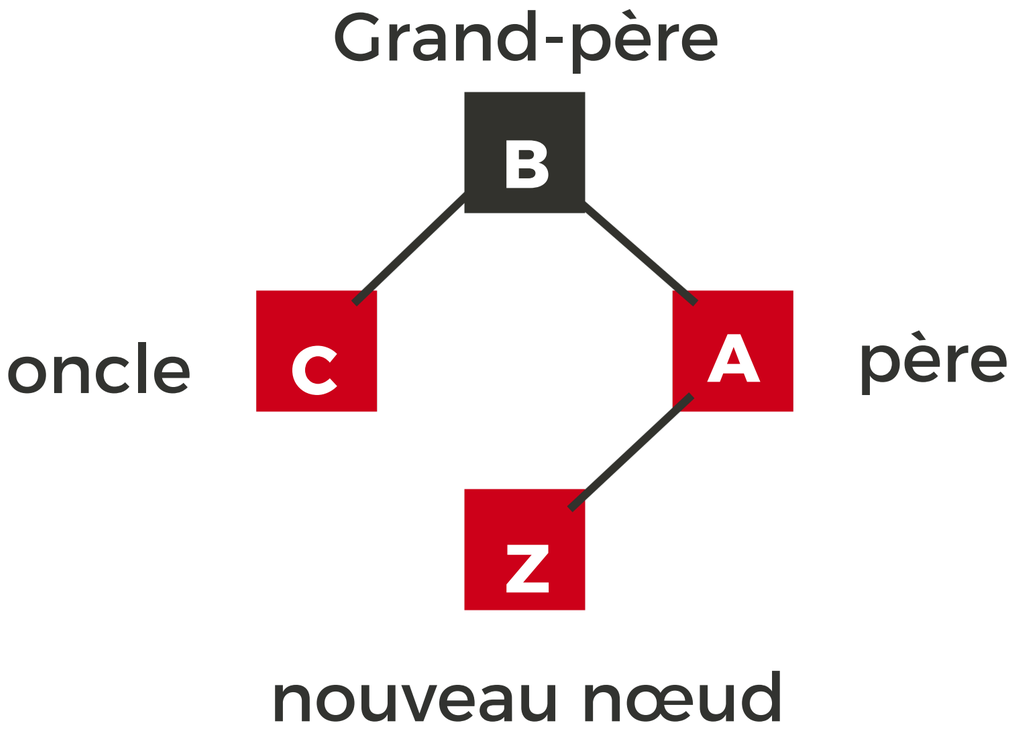
nouveau nœud

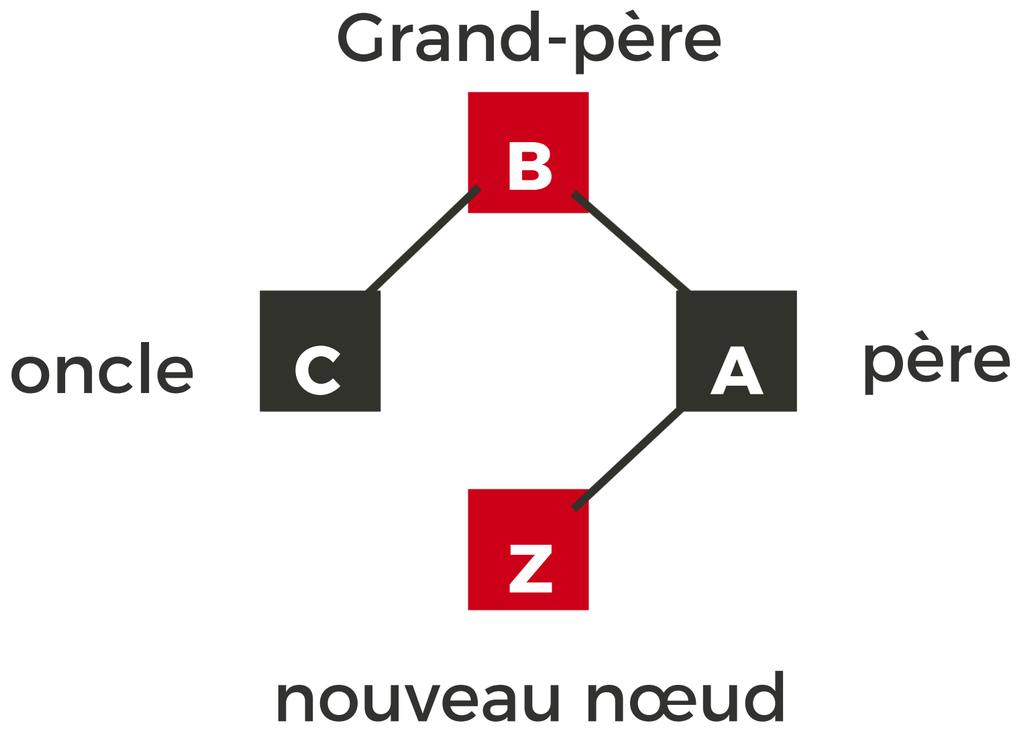
nouveau nœud

💡 Solution: **recolorer** le nouveau nœud en noir.

INSERTION D'UN ÉLÉMENT

👉 **Cas 02:** Z.oncle = rouge



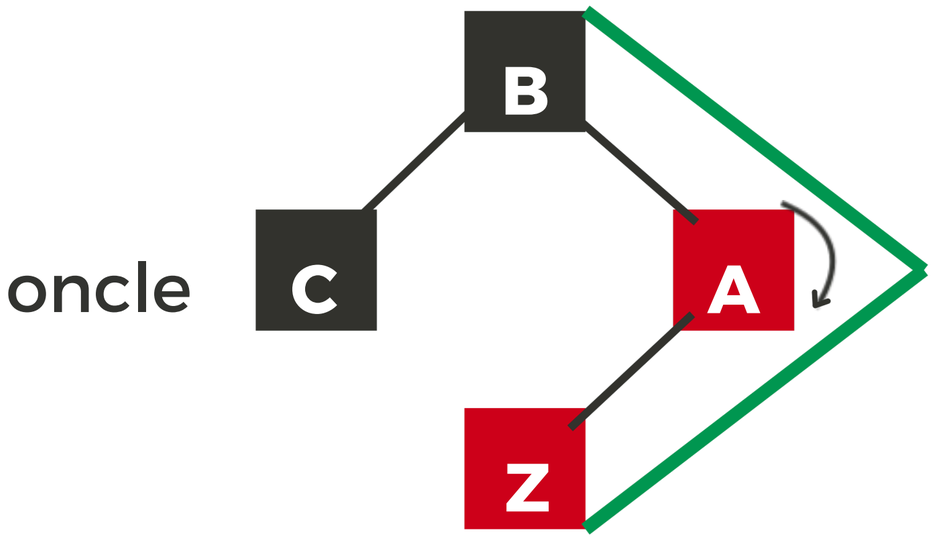


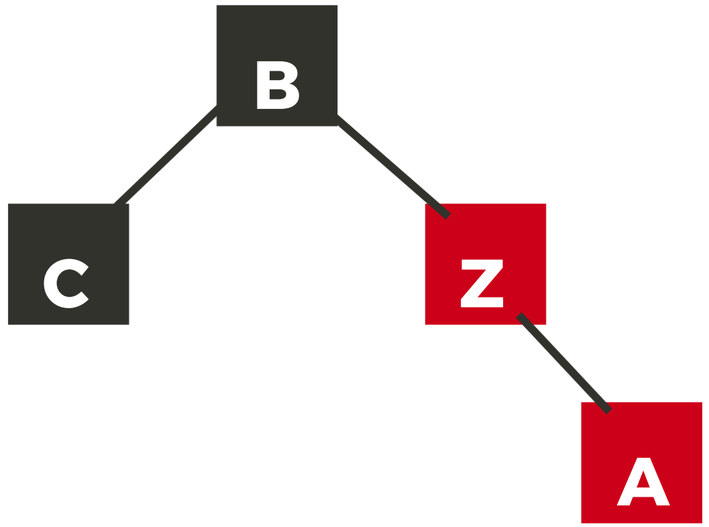
💡 Solution: dans ce cas, on **recolore le père, le grand-père et l'oncle** de z.

😀 **Remarque:** vous pouvez remarquer que la racine est rouge, mais gardez à l'esprit qu'il s'agit uniquement **d'un sous-arbre.**

INSERTION D'UN ÉLÉMENT

👉 **Cas 03:** Z.oncle = noir (traingle)

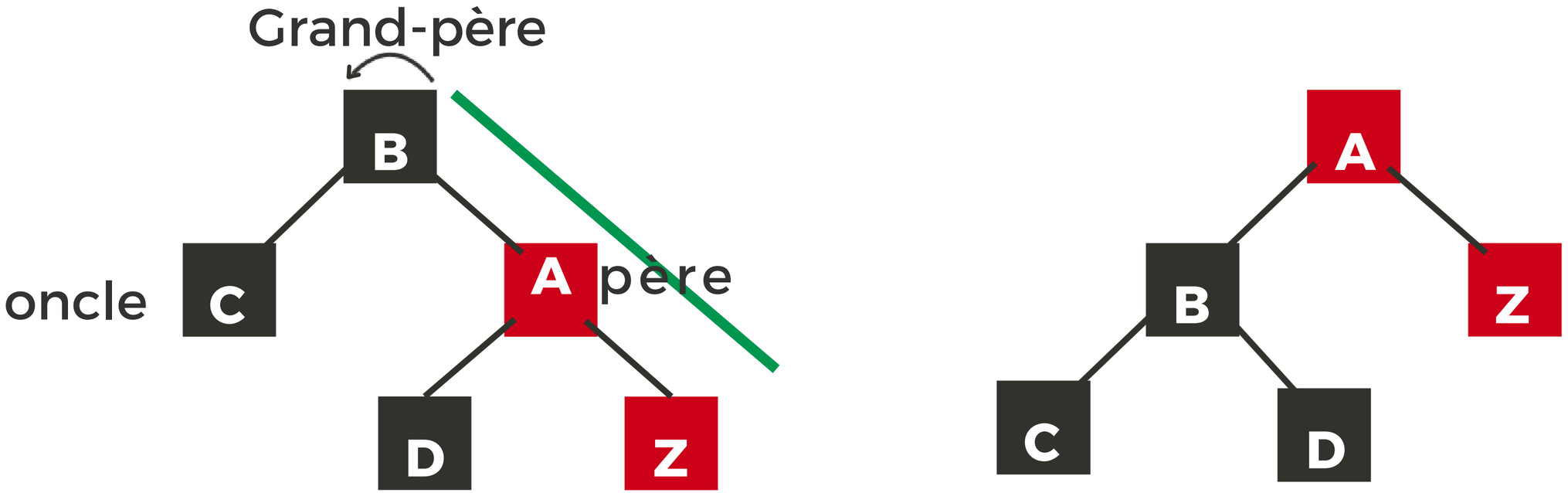




💡 Solution: lorsque l'oncle de z est noir (cas "triangle"), on effectue une rotation sur le père de z dans la direction opposée à celle de z.

INSERTION D'UN ÉLÉMENT

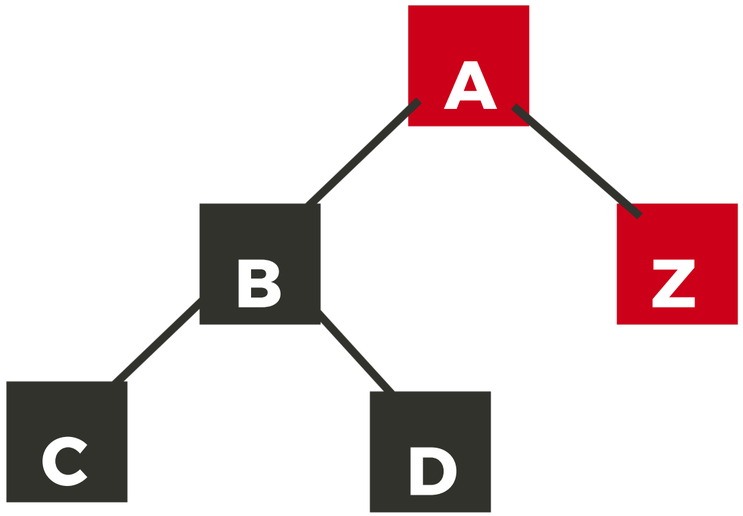
👉 **Cas 04:** Z.oncle = noir (ligne)

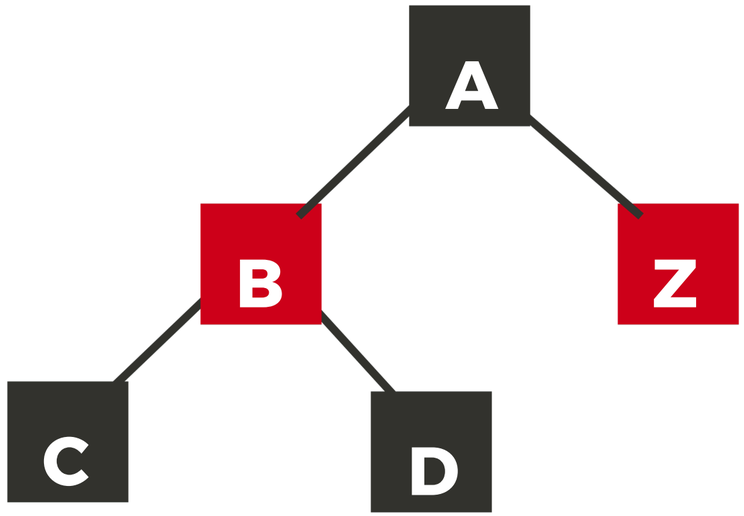


💡 Solution: on effectue une rotation sur le grand-père de z, puis on échange les couleurs du père et du grand-père.

INSERTION D'UN ÉLÉMENT

👉 **Cas 04:** Z.oncle = noir (ligne)





💡 Solution: on effectue une rotation sur le grand-père de z, puis on échange les couleurs du père et du grand-père.

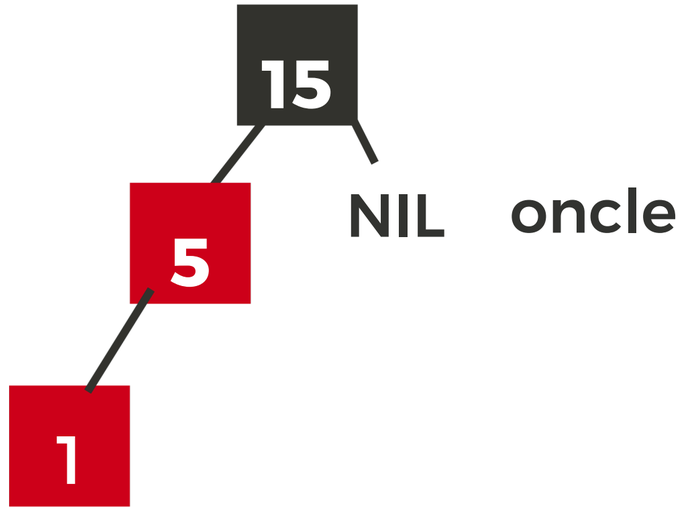
INSERTION D'UN ÉLÉMENT

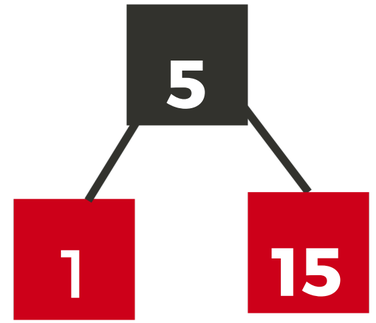
👉 **Example**

Insert 15

Cas 01: Z = la racine
(on échange la couleur)

Insert 5 et 1

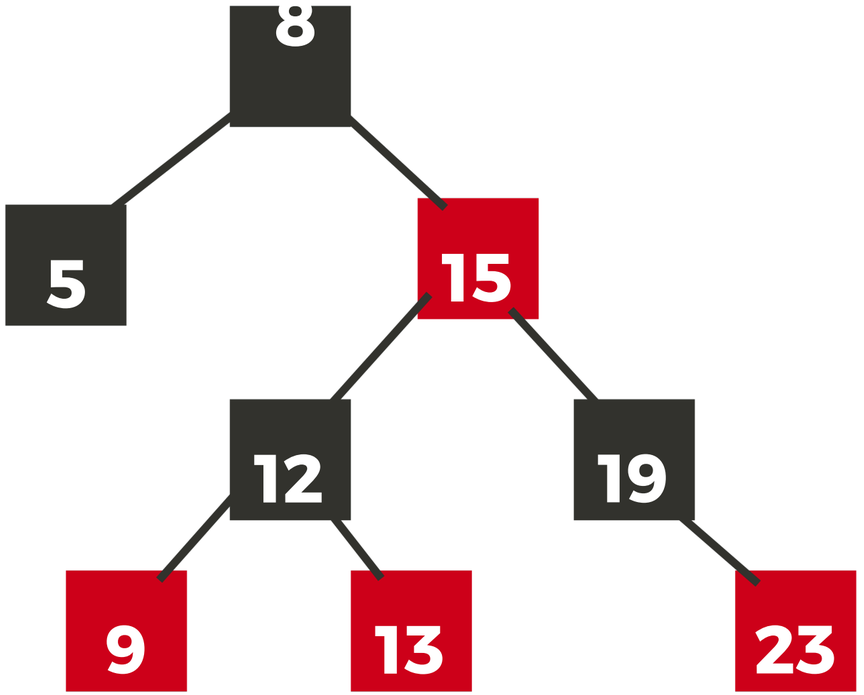




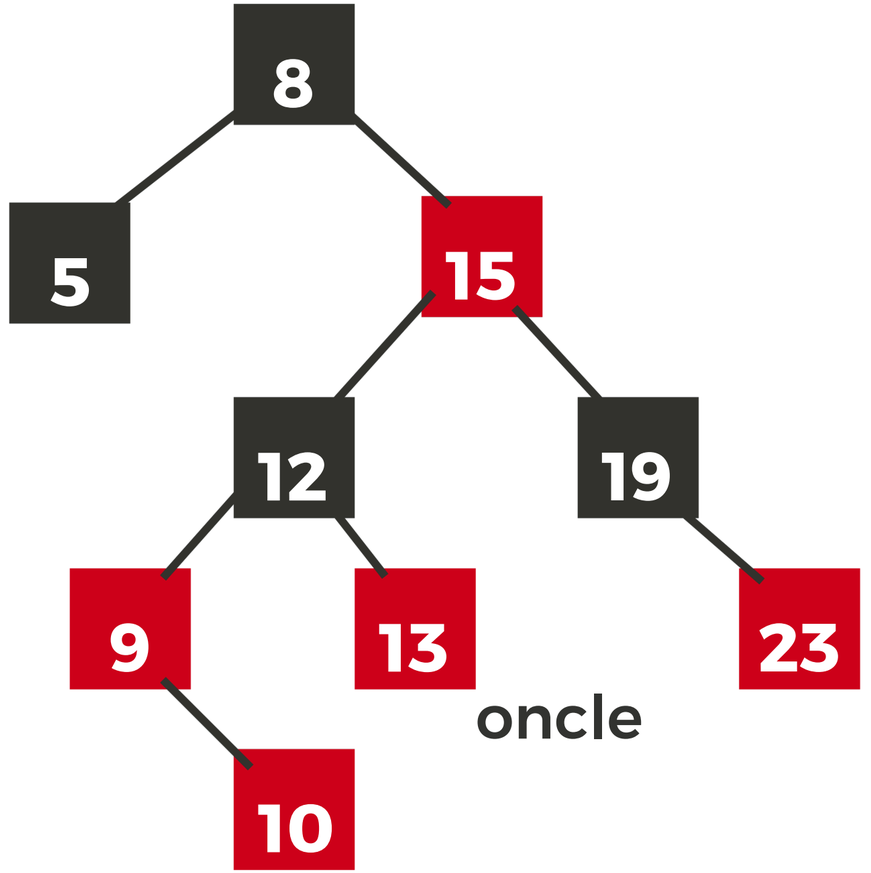
Cas 04: Z.oncle = noir (ligne)

INSERTION D'UN ÉLÉMENT

👉 **Example**



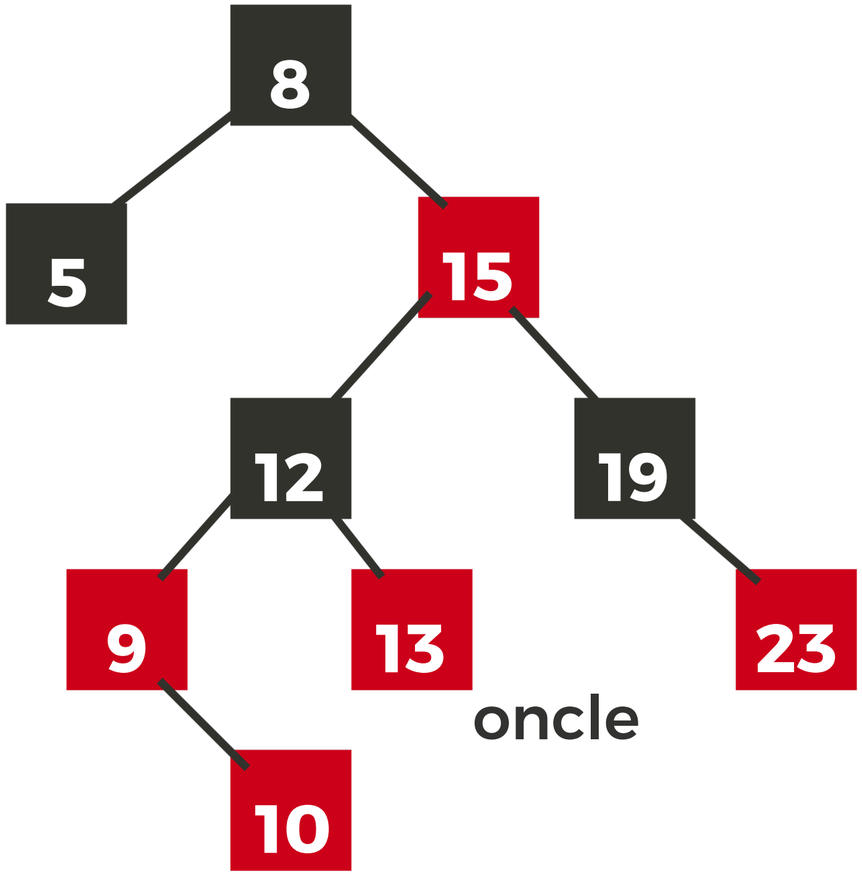
Insert 10



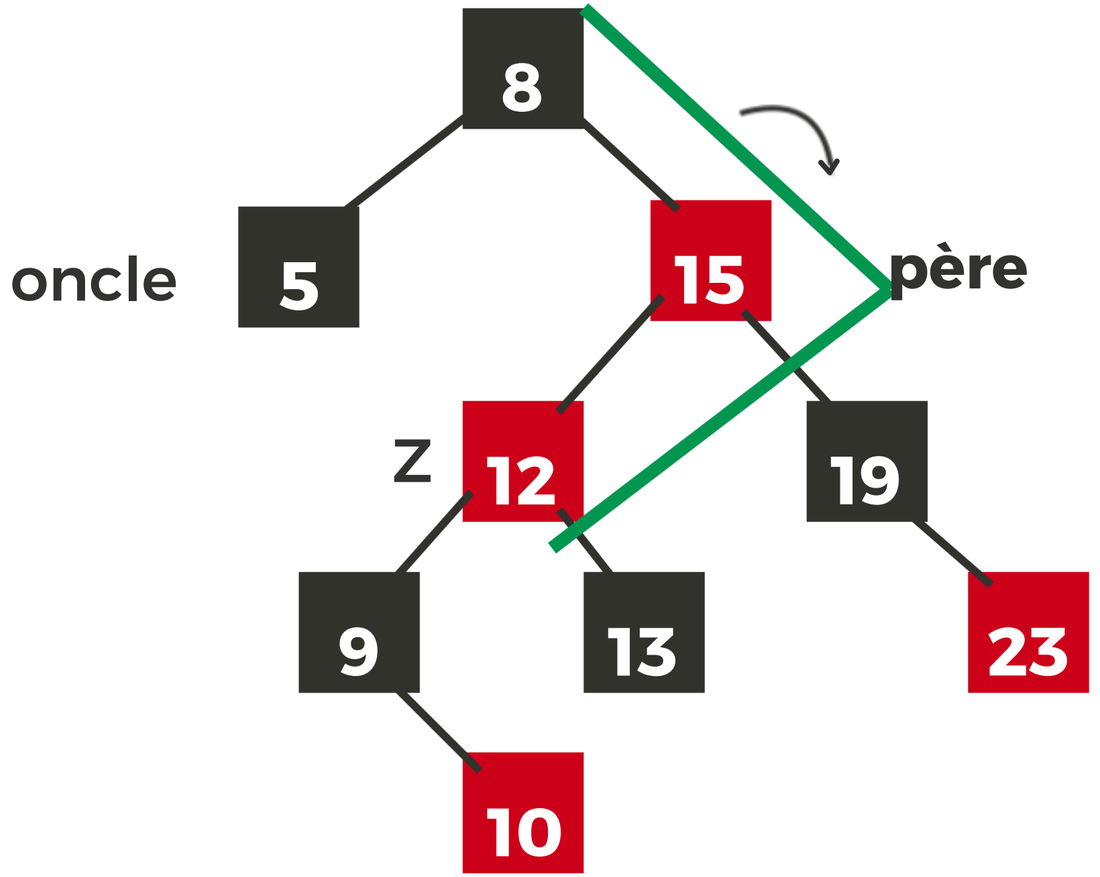
Cas 02: Z.oncle = rouge

INSERTION D'UN ÉLÉMENT

👉 **Example** Insert 10



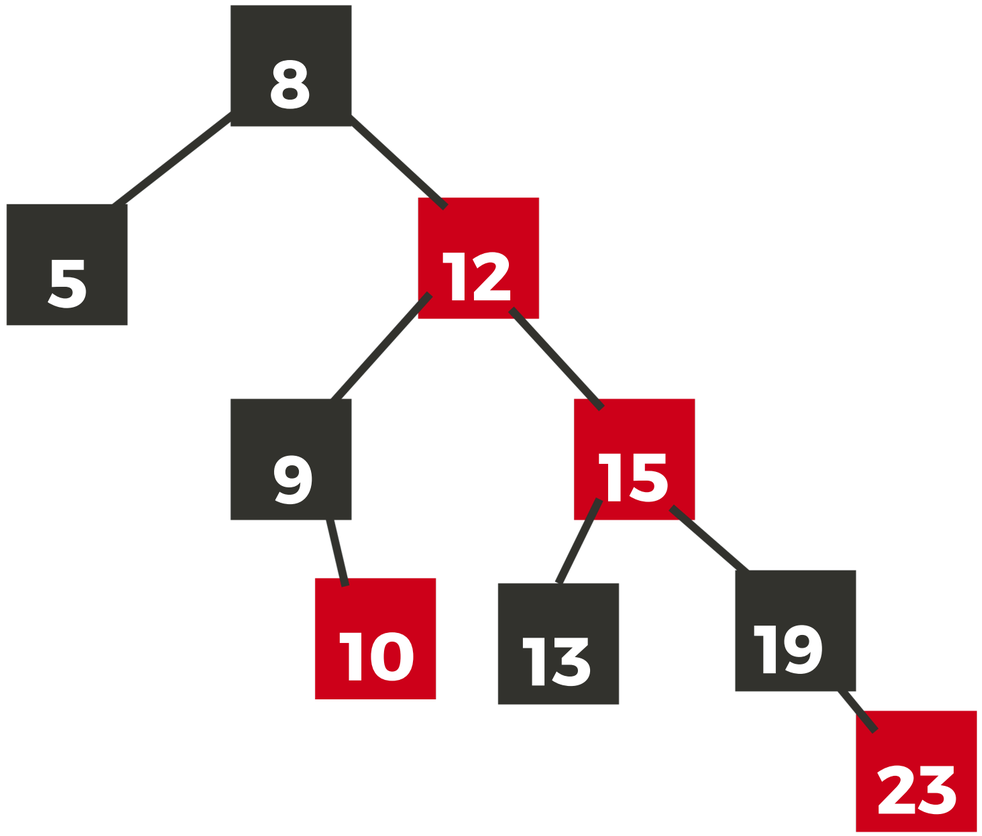
Recolorer le père, l'oncle et le grand-père de z.



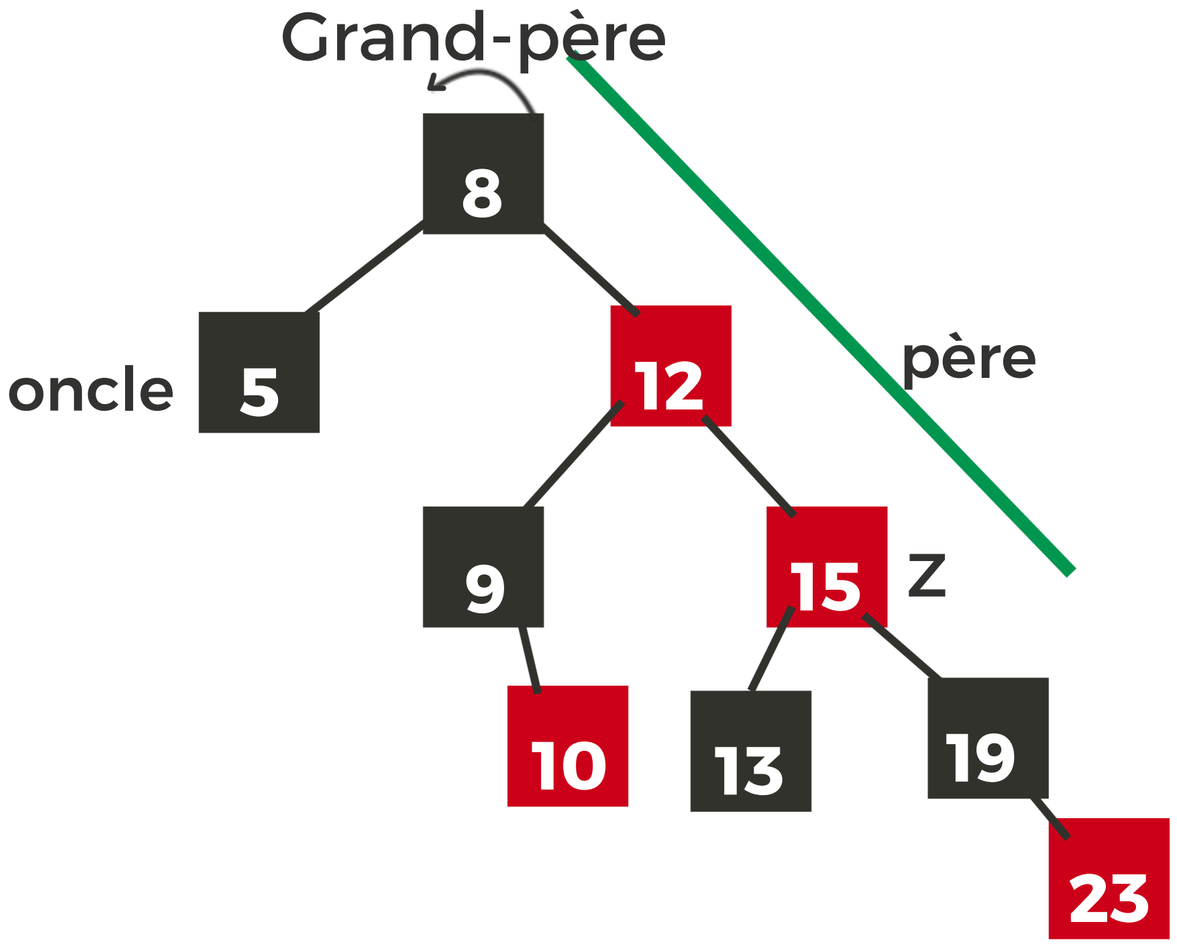
Cas 3: z.oncle = noir (triangle)

INSERTION D'UN ÉLÉMENT

👉 **Example** Insert 10



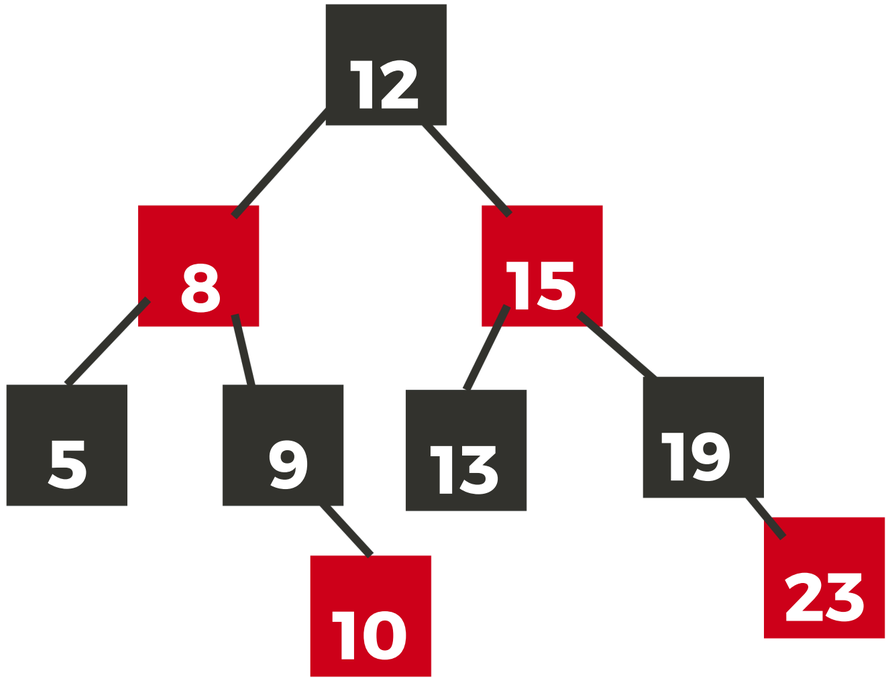
rotation



Cas 4: z.oncle = noir (ligne)

INSERTION D'UN ÉLÉMENT

👉 **Example** Insert 10



PRINCIPE DE LA SUPPRESSION

On applique d’abord l’algorithme de suppression d’un arbre binaire de recherche.
Si le nœud supprimé est rouge, aucune propriété de l’arbre rouge-noir n’est violée.
En revanche, si le nœud supprimé est noir, la propriété 4 (même nombre de nœuds noirs sur tous les chemins) peut être rompue.

Pour corriger cela, on introduit temporairement un nœud doublement noir (appelé « double noir ») sur le chemin affecté. Ce nœud servira à rétablir la balance par recolorations et rotations successives.

PRINCIPE DE LA SUPPRESSION

Lorsqu’un nœud noir est supprimé, le déséquilibre provoque un double noir.

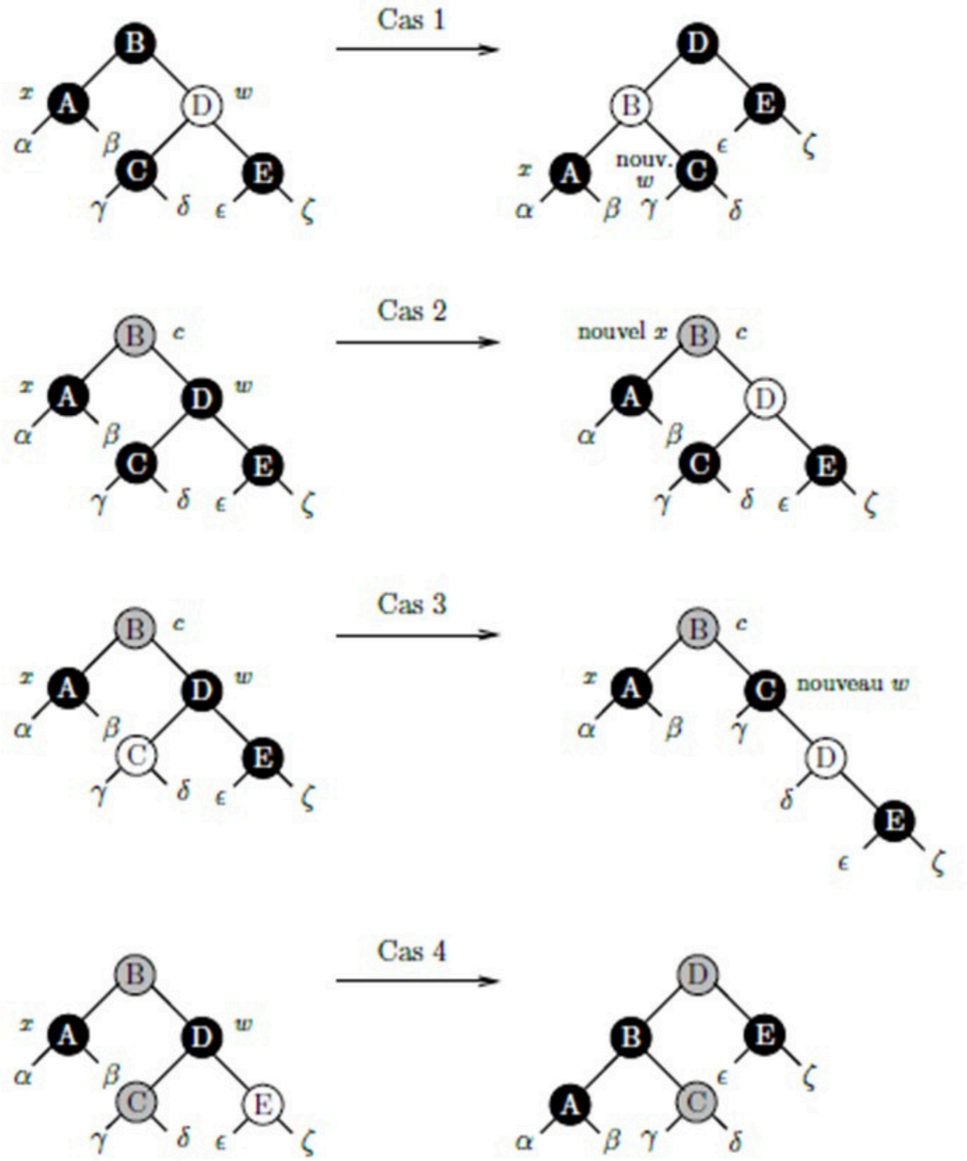
Cas 1 : Le frère du nœud double noir est rouge.
Solution: Rotation autour du père + inversion des couleurs du père et du frère.

Cas 2 : Le frère est noir et ses deux fils sont noirs.
Solution: Le frère devient rouge, et le double noir remonte au père.

Cas 3 : Le frère est noir, son fils proche est rouge, et son fils éloigné est noir.
Solution: Rotation autour du frère pour transformer le triangle en ligne (prépare Cas 4).

Cas 4 : Le frère est noir, son fils éloigné est rouge.
Solution: Rotation autour du père + recoloration du père et du frère.

PRINCIPE DE LA SUPPRESSION



Merci Pour Votre Attention