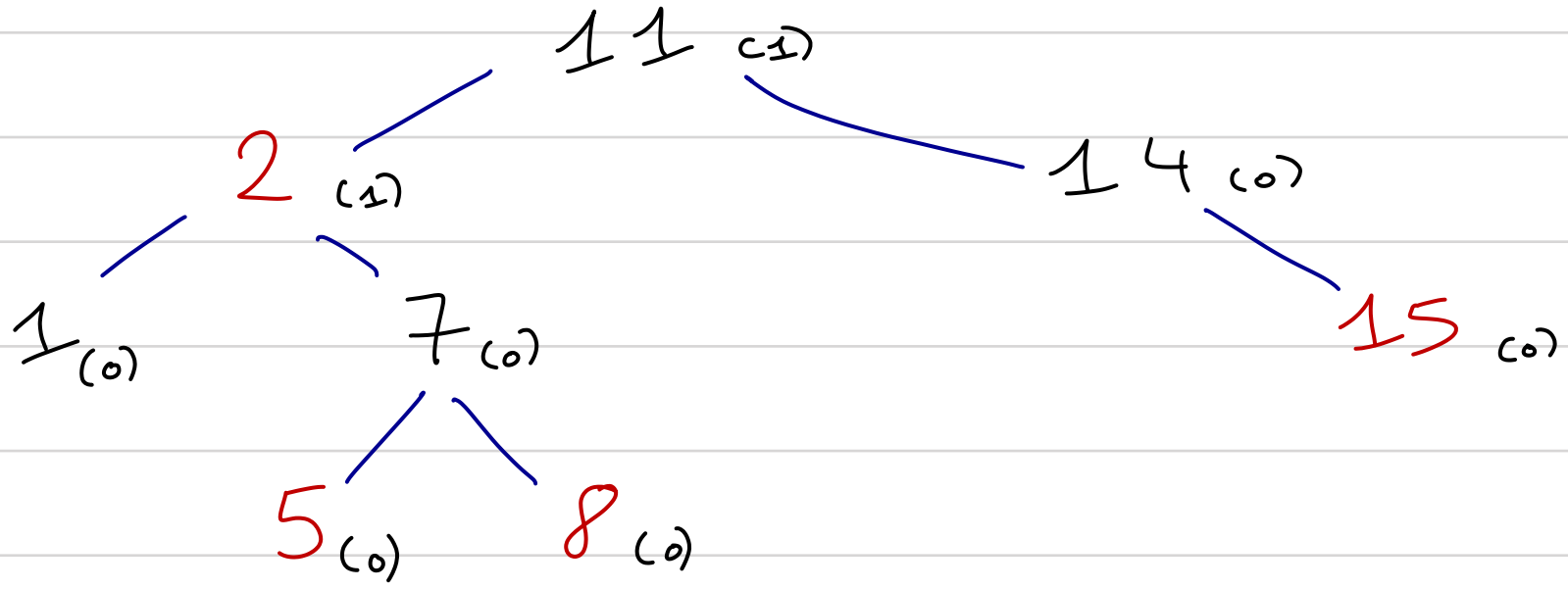
Arbres rouge-noir

ABR
- → Recherche
- → Minimum / Maximum / Suivant
- → Insertion / Suppression

} $O(h)$

Équilibré $h = O(\log n)$



ABR rouge-noir : ABR et chaque noeud se voit associer une couleur rouge ou noire :

→ Si un noeud est rouge, tous ses enfants (s'il en a) sont noirs.

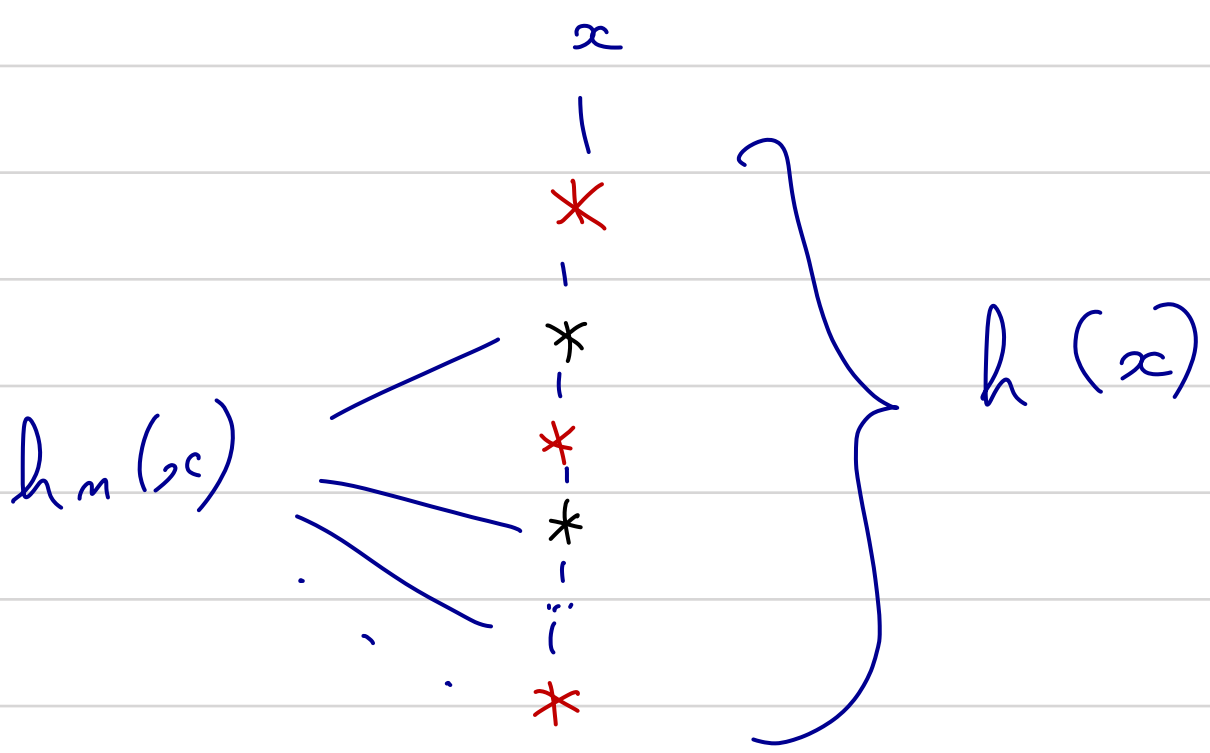
→ $\hat{M}$ nombre de noeuds noirs sur tout chemin d'un noeud x de l'arbre aux descendants (x inclus) ayant 0 ou 1 enfant.

Définition La **hauteur noire** (bh) d'un noeud x (notée $bh(x)$) dans un arbre rouge-noir est le nombre (unique) de noeuds noir sur tout chemin du noeud x à une feuille de l'arbre, le noeud x exclu.

Prop. $bh(x) \geq \dfrac{h(x)-1}{2}$

(**Rem** $bh(x) \leq h(x)$)

Dém.



$2\,bh(x) + 1 \geq h(x)$

$\boxed{bh(x) \geq \dfrac{h(x)-1}{2}}$ □

Prop. : Dans un arbre rouge-noir à n noeuds et de racine r, $n \geq 2^{bh(r)}$.

Dém. Par récurrence, sur la hauteur h de l'arbre rouge-noir.

Cas de base : $h = 0$ $\quad n = 1$ $\quad 1 \geq 2^0$

$bh(r) = 0$

Cas inductif :

On suppose $n \geq 2^{h_n(x)}$ vérifié pour tous les arbres rouge-noir de hauteur $\leq h$. On considère un arbre rouge-noir de hauteur $h+1$.

- Si $h_n(x) = 0$ alors, trivialement,
$$n \geq 2^0 = 1$$
- On suppose $h_n(x) > 0$.

Cela signifie que la racine a exactement deux enfants (sinon, violation de la 2ème propriété des arbres rouge-noir)

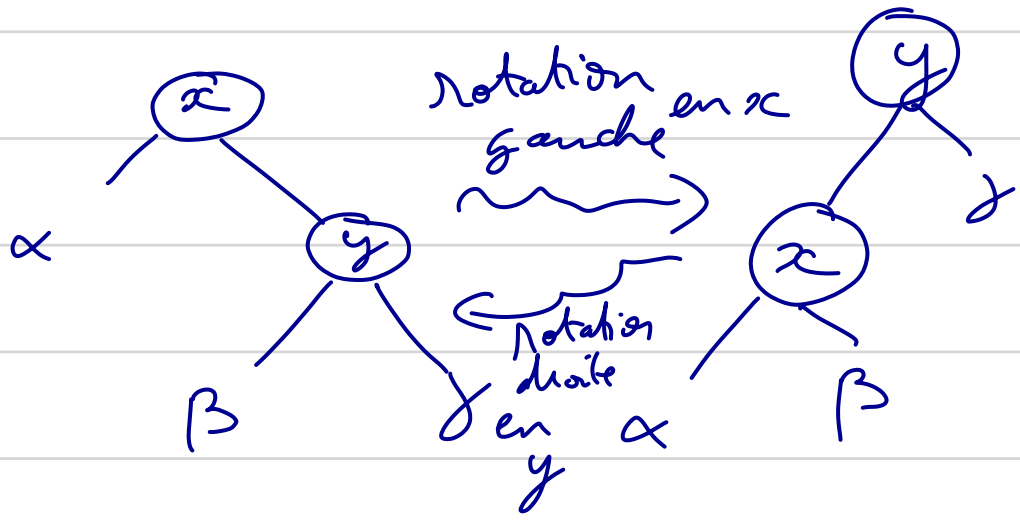
Ces enfants sont racine d'arbre de hauteur h et de hauteur noire $\geq h_n(x) - 1$.

$$n \geq 2 \times 2^{h_n(x)-1} + 1 \geq 2^{h_n(x)} \qquad \square$$

Thm. Les arbres rouge-noir sont équilibrés.

Dém.
$$n \geq 2^{h_n(x)} \geq 2^{\frac{h-1}{2}}$$
$$\iff \log n \geq \frac{h-1}{2} \iff h \leq 2(\log n) + 1$$
Autrement dit $h = O(\log n)$. $\square$

Rotation { gauche / droite



Insertion dans un arbre rouge-noir $O(\log n)$

$O(\log n)$ → Insertion dans l'arbre à son endroit naturel z, coloré en rouge

→ Correction de la violation rouge-noir entre z et son parent p :

$O(1)$ 1/ Si z est enfant de la racine, on colorie la racine en noir.

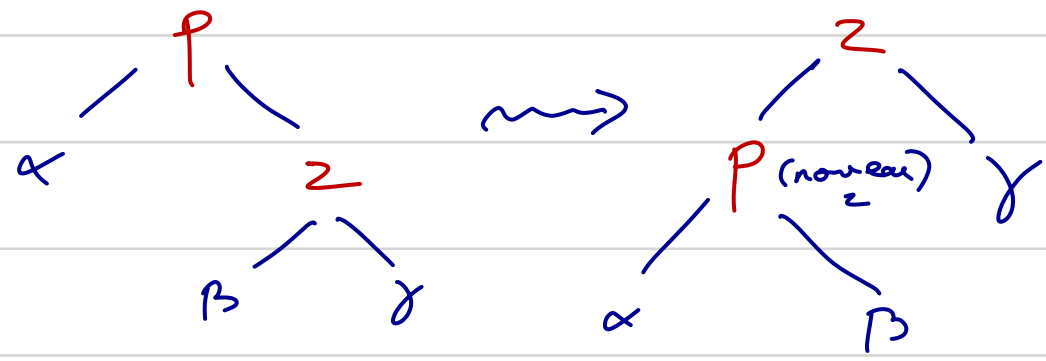
2/ On traite du cas où le parent de z est un enfant gauche (le cas où c'est un enfant droit est symétrique)

$O(1)$ 3/ Si l'oncle w de z est rouge, on colorie w et p en noir, et le grand-parent de z en rouge. On continue avec le grand-parent (nouveau z)

$O(\log n)$

4/ Sinon, si z est un enfant droit, on fait une

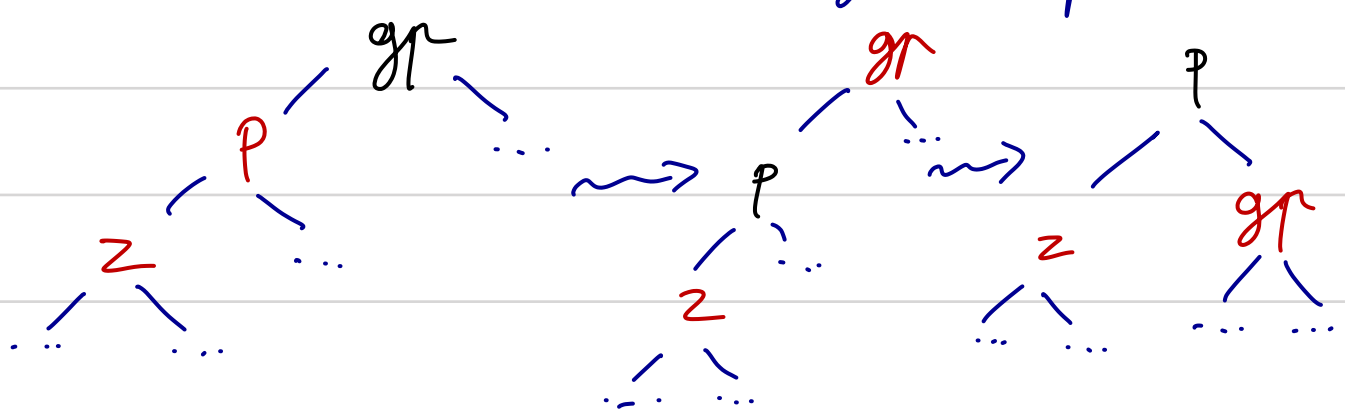
$O(1)$ rotation gauche sur le parent de z, et on considère le nouvel enfant gauche de z pour le cas suivant.



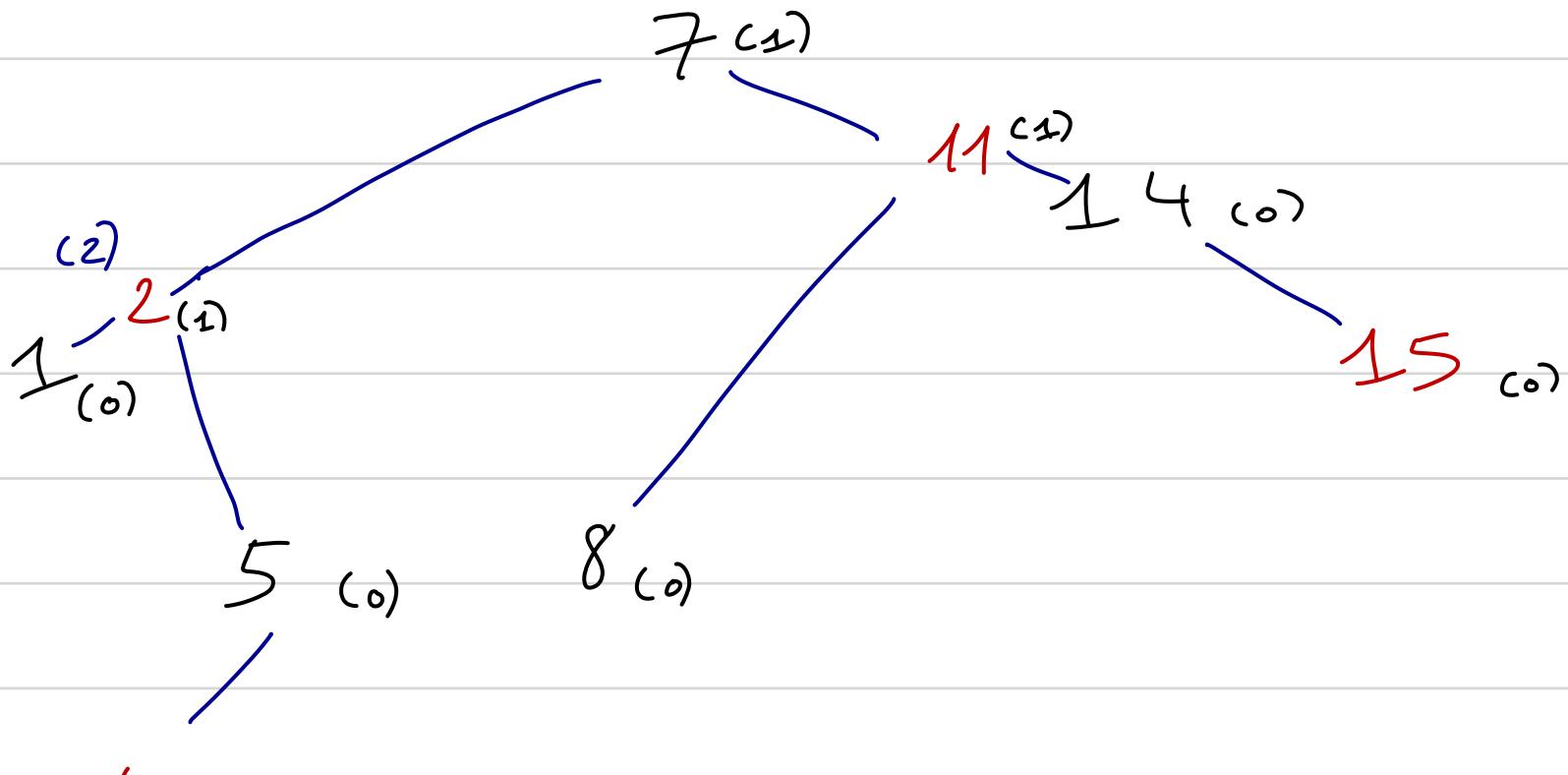
5/ z est un enfant gauche, on colorie son parent en noir, son grand-parent en rouge, puis

$O(1)$ on fait une rotation droite sur le grand-parent.

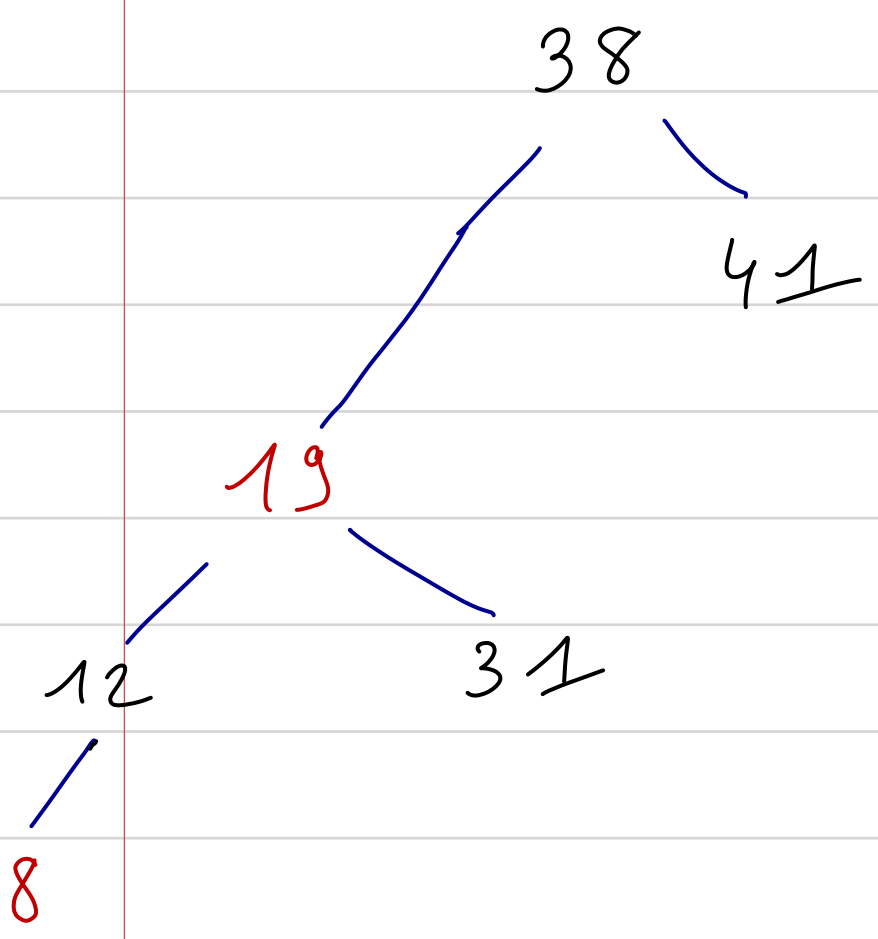
Fini.



Insérer l'élément 4 dans l'arbre exemple :



Insérer à l'arbre vide : 41, 38, 31, 12, 19, 8



Suppression dans un arbre rouge-noir $O(\log n)$

(On recherche dans l'arbre le noeud z)

$O(\log n)$ 1/ On supprime le noeud z :

— Si z est une feuille, on le supprime ;
$O(1)$ si z était noir, on se souvient qu'il faudra corriger les couleurs à l'ancien emplacement de z

— Si z avait un seul enfant x, on remplace
$O(1)$ z par x ; si z était un noeud noir, on se souvient qu'il faudra corriger les couleurs au niveau de x

— Sinon, z a deux enfants.

* On recherche dans le sous-arbre de
$O(\log n)$ l'enfant droit de z l'élément minimal y. Si y est de couleur noire, on se souvient qu'il faudra corriger les couleurs au niveau de l'emplacement actuel de y.

* On remplace y par son enfant droit (s'il
$O(1)$ en a un).

* On remplace z par y, en conservant la
$O(1)$ couleur de y.

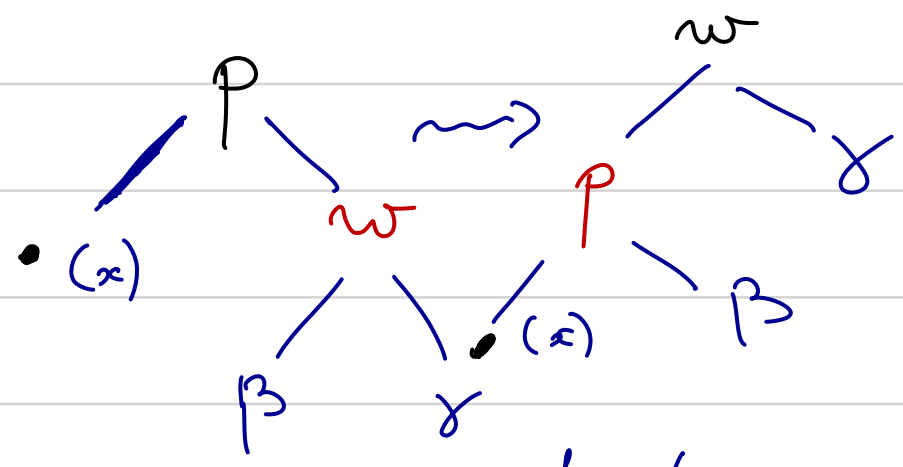
2/ S'il faut corriger les couleurs au niveau d'un emplacement x, ça veut dire qu'il manque une couleur noire au niveau de x.

$O(1)$ — Si x est rouge, on le colorie en noir, et on a fini.

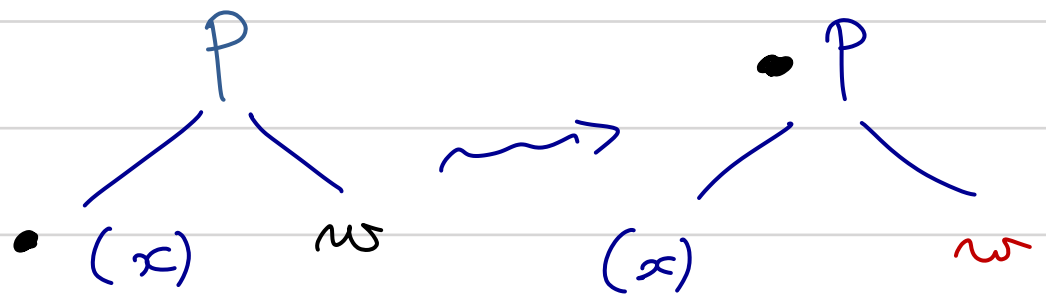
$O(1)$ — Si x est la racine, on a fini.

— Sinon, on suppose que x est un enfant gauche (le cas enfant droit est symétrique).

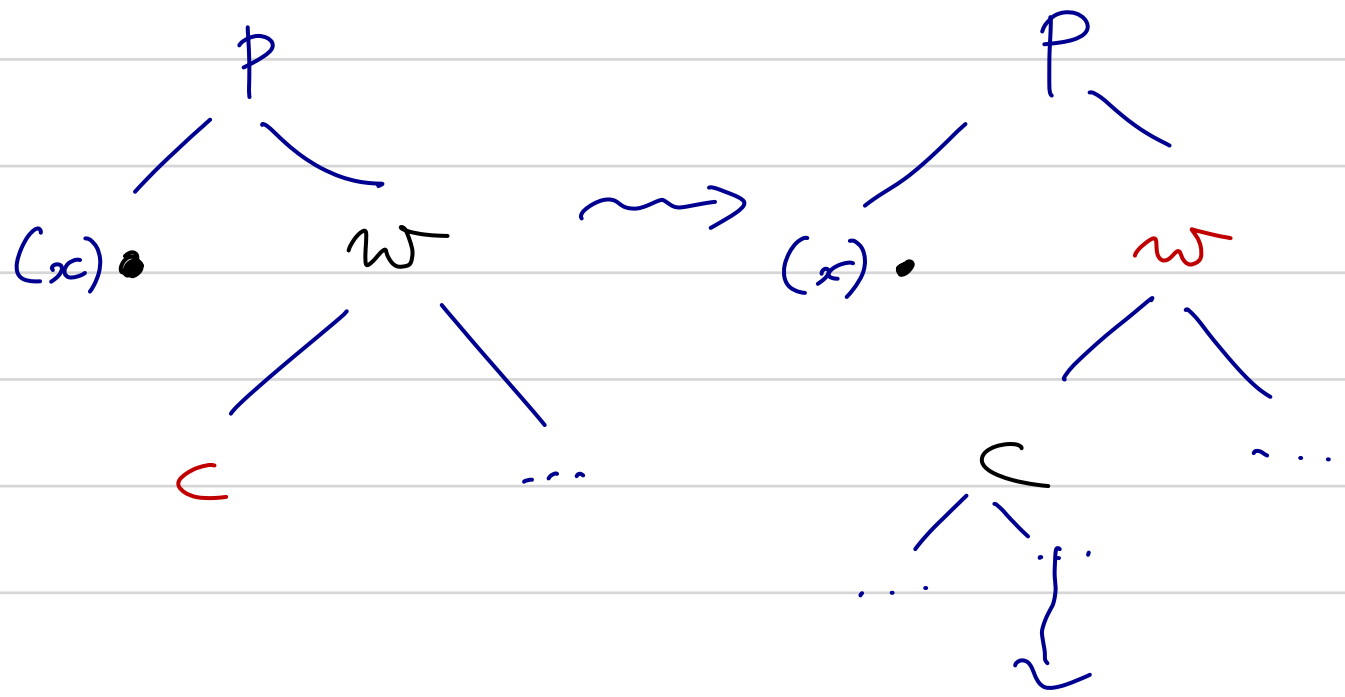
a) Si le frère de x est rouge, on le rend noir, on rend le parent p de w et x rouge, et on fait une rotation gauche $O(1)$ en p. On se retrouve dans le cas d'un frère noir, qu'on traite dans l'un des cas restant.



b) Si w est noir et n'a que des enfants noirs : on rend w rouge, et on répète la procédure au niveau du parent p de w $O(\log n)$ et x, auquel il manque une couleur noire :

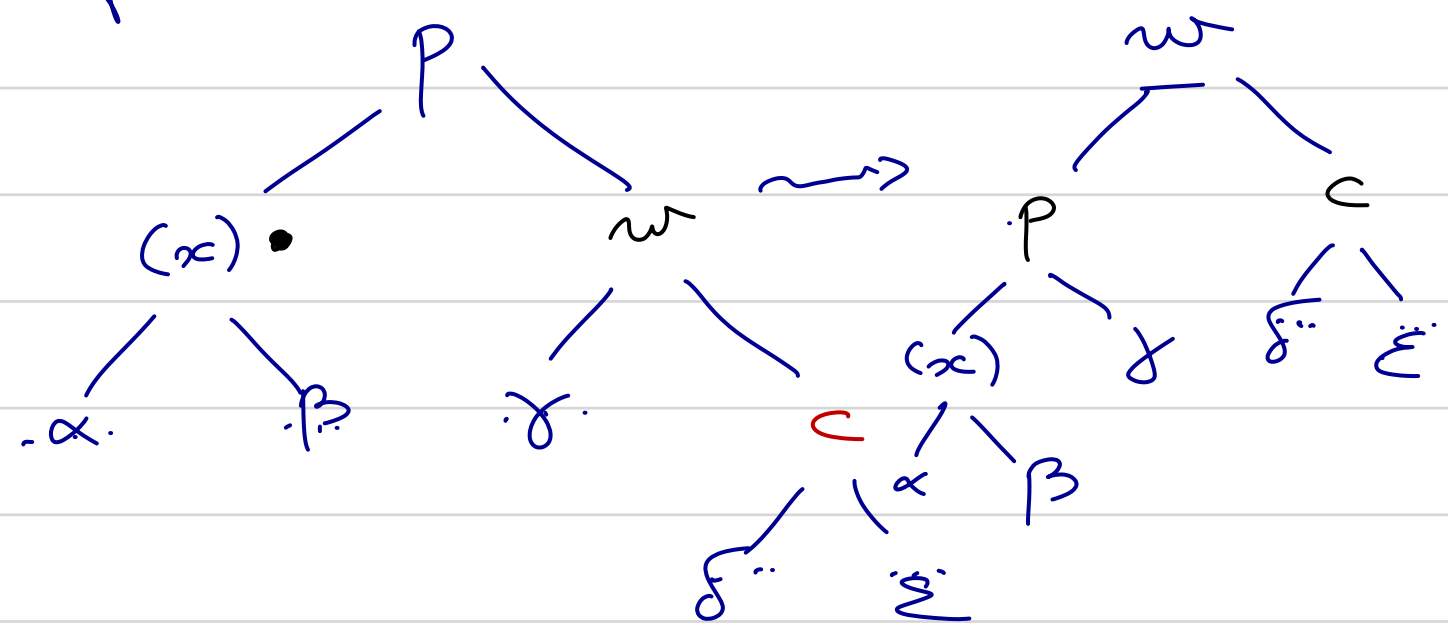


c) Si w est noir et que son enfant gauche est rouge : on rend w rouge et son enfant gauche noir, puis on fait $O(1)$ une rotation droite en w. On passe au cas suivant.



d) Si w est noir et que son enfant droit est rouge, on colorie w dans la couleur de p, on colorie p en noir et l'enfant droit de w en noir et on fait une rotation gauche au niveau de p. On a fini.

$O(1)$



ABRE vs table de hachage

$O(\log n)$ — $O(1)$ (amorti, en moyenne)

y compris pour des opérations exploitant l'ordre des éléments

pas possibilité d'opérations exploitant d'ordre des éléments

Supprimer de l'arbre d'exemple les valeurs

8 12 19 31 38 41

Tri par ABRE $O(n \log n)$

$O(n \times \log n)$ → Insère n éléments dans l'ABRE

$O(n)$ → On parcourt linéairement l'ABRE (de gauche à droite)