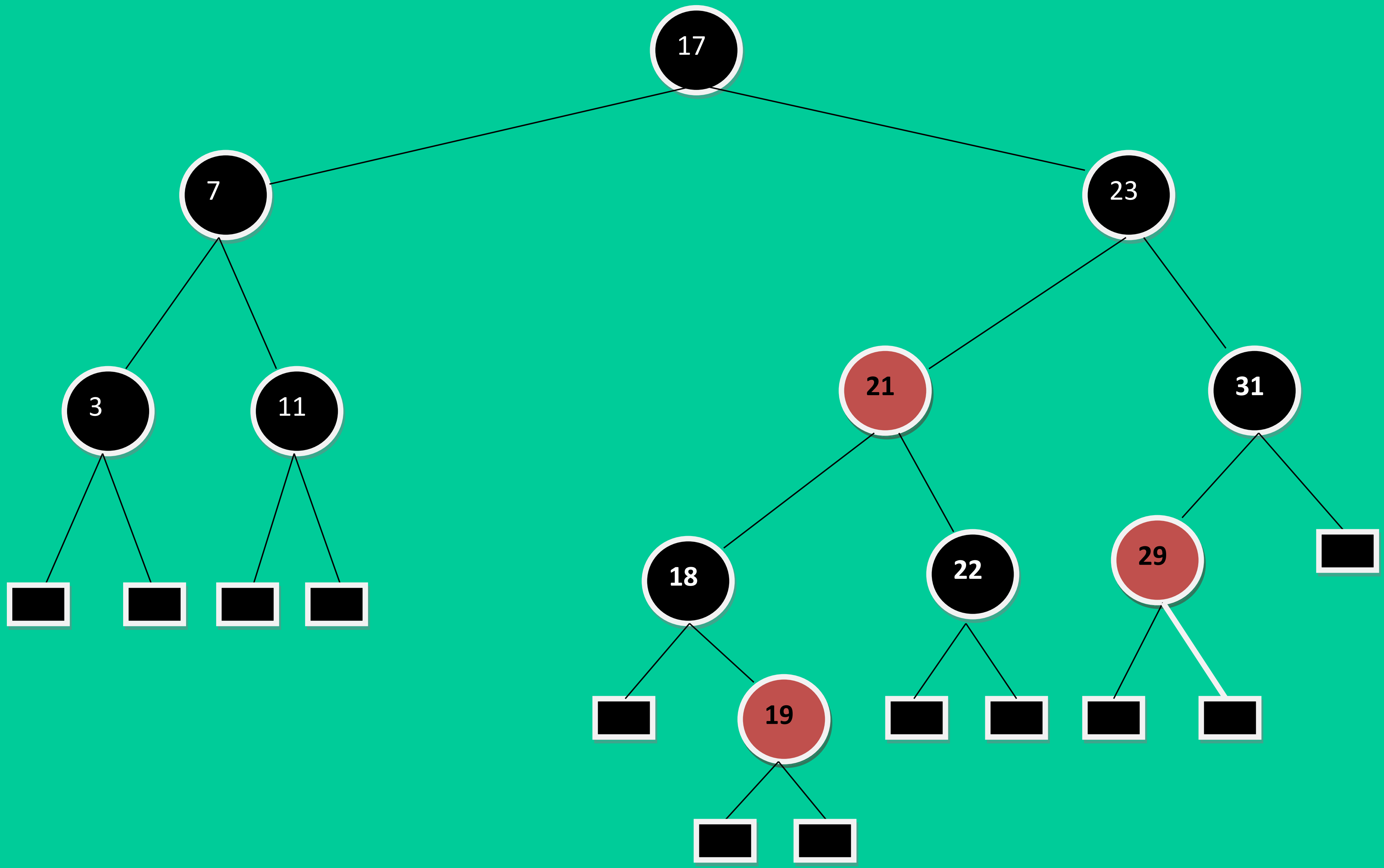
Sur les arbres rouge noir: partie III

Arbres rouge-noir

Inventés par **Bayer** en 1972.

Étudiés en détail par **Guibas** et **Sedgewick** en 1978.

Un ABR est dit rouge-**noir** s'il vérifie les 5 propriétés suivantes illustrées dans l'arbre ci-dessous:



17
7
23
3
11
21
31
18
22
29
19

P1 : chaque nœud est soit **rouge**, soit **noir**;

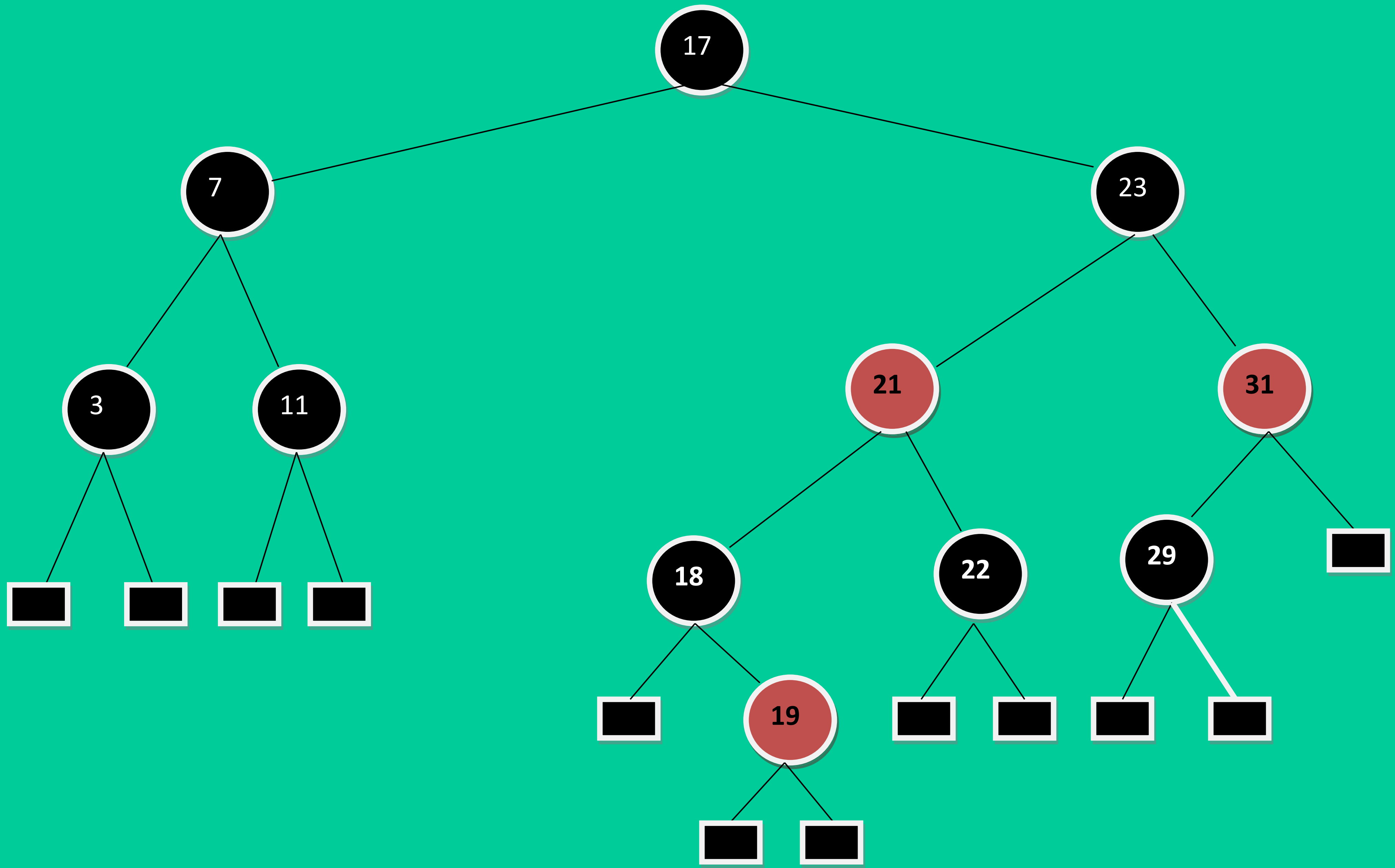
P2 : la racine est **noire** ;

P3 : chaque sentinelle est **noire** ;

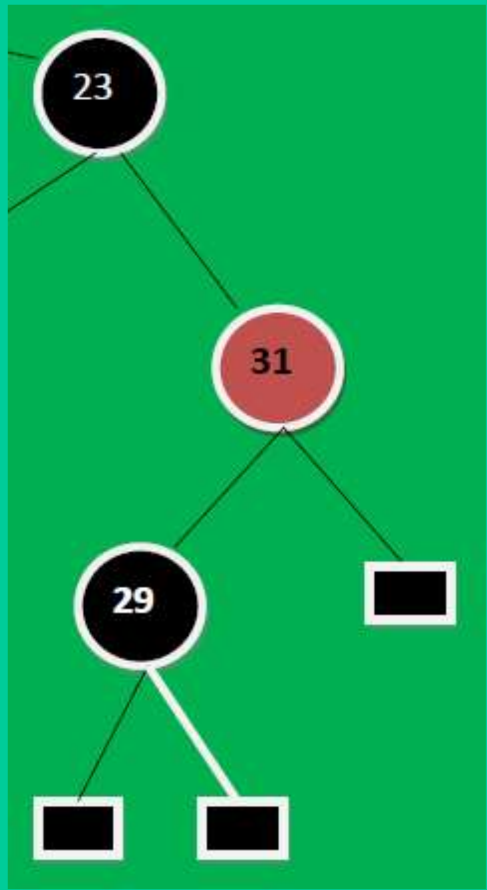
P4 : si un nœud est **rouge**, alors ses deux fils sont **noirs**;

P5 : les chemins reliant un nœud à une **sentinelle** traversent le **même nombre** de nœuds **noirs**.

Cet arbre **n'est pas** rouge noir !

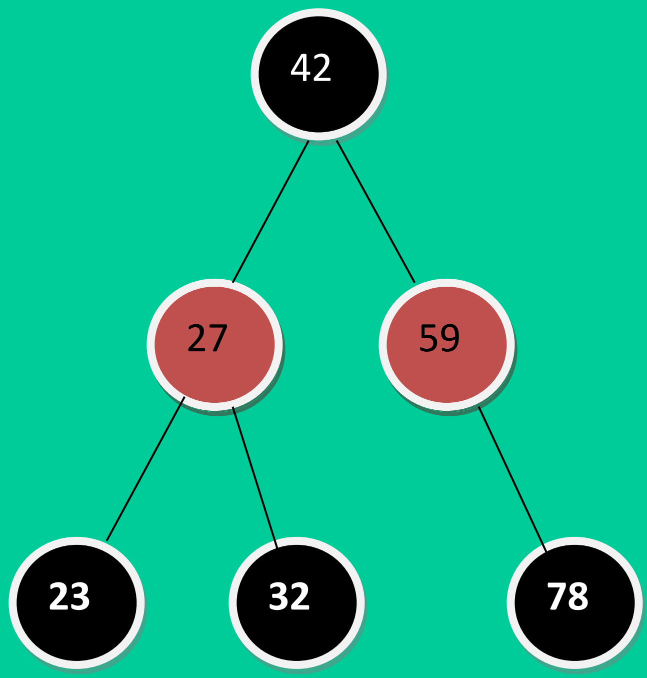


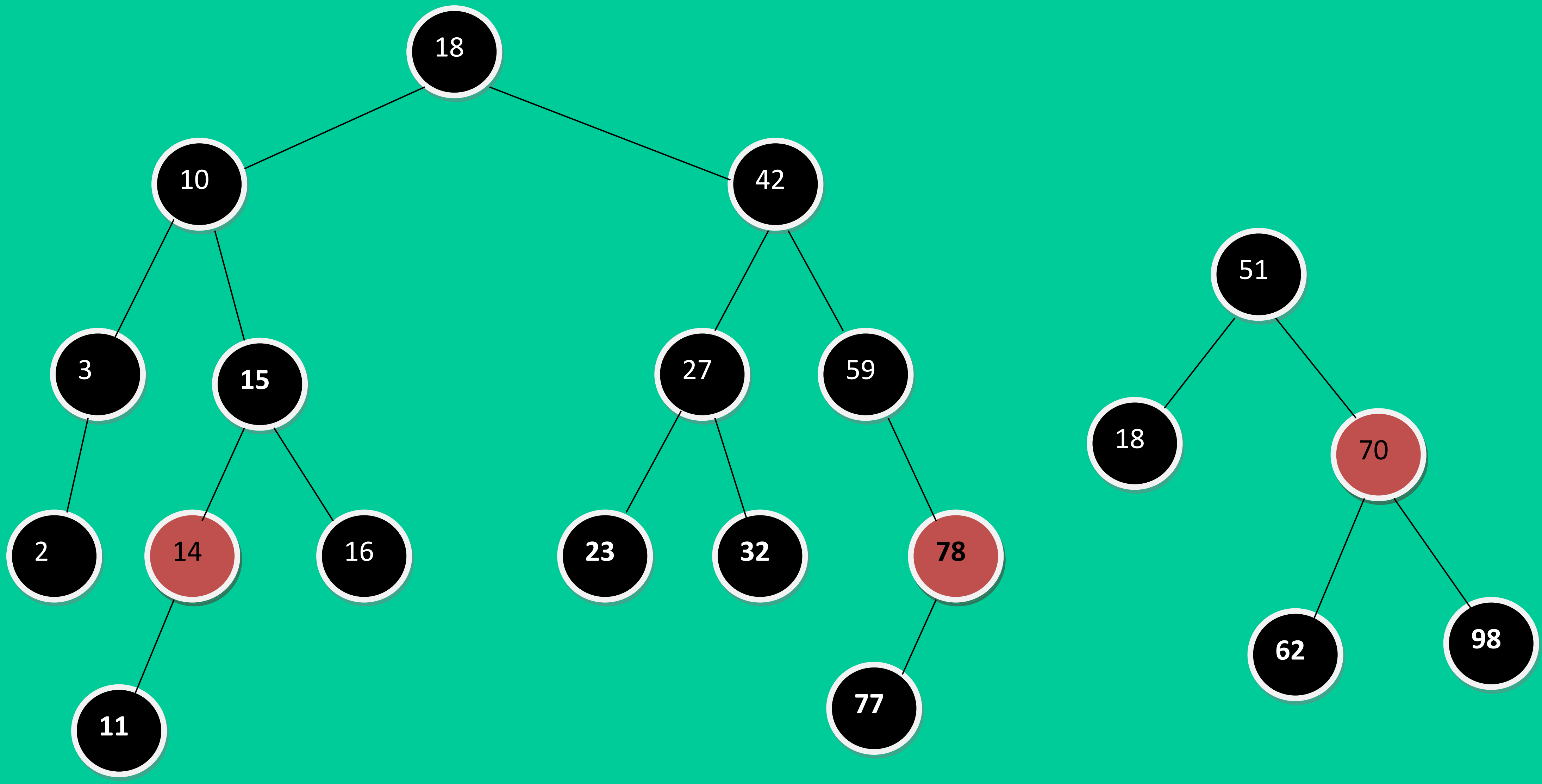
Pourquoi ?



La propriété 5 est violée !

Exemple d’arbres rouge noir




18
10
42
51
3
15
27
59
18
70
2
14
16
23
32
78
62
98
11
77

Analyse des propriétés

La propriété **P1** signifie que chaque nœud a un attribut supplémentaire: sa couleur, qui est soit **rouge** soit **noire**.

Les propriétés **P2** et **P4** stipulent que les nœuds rouges ne doivent pas être trop nombreux.

La propriété **P3** est simplement technique: elle est sans grande importance.

La propriété **P5** est la plus importante: c'est une condition d'équilibre.

Elle signifie que si on fait abstraction les nœuds rouges, on obtient un arbre binaire **parfaitement équilibré**.

Contraintes induites

1- aucun chemin **ne peut avoir deux nœuds rouges consécutifs** (P4).

2-le **plus court chemin** de la racine à une sentinelle n'est formé **que de nœuds noirs** (P5).

3-le **plus long chemin** doit **alterner** entre les nœuds rouges et noirs. (P5).

4-D'après P5, le **plus long chemin** ne peut être **deux fois plus long** que le **plus court**.

Hauteur noire

La **hauteur noire** d'un nœud **x**, notée $H_n(x)$, est le nombre de nœuds **noirs** sur un chemin de **x** à une sentinelle (**x** n'est pas compris).

La **hauteur noire** d'un arbre rouge-noir est la hauteur noire de sa **racine**.

propriété de la hauteur noire

La **hauteur noire** d'un nœud x, $H_n(x)$, est telle que:

$$n \geq 2^{Hn(x)}-1$$

n étant le nombre de nœuds du sous arbre de x .

Démonstration par récurrence

On montre par récurrence sur Hn(x) que :

$$n \geq 2^{hn(x)}-1$$

- Si la hauteur noire de x est 0, alors le chemin contient n=0 ou n= 1 nœud.

$$n \geq \quad 2^{Hn(x)}-1 = 2^{0}-1 = 0$$

- Si la hauteur noire de *x* est >0, alors chacun de ses deux fils x_g et x_d une hauteur noire égale:

  - soit à Hn(x) s'il est rouge,
  - soit à Hn(x)-1 s'il est noir.

Donc, en appliquant l'hypothèse de récurrence aux deux fils x_g et x_d de x, ona :

$$n > 2(2^{Hn(x)-1} - 1) + 2$$

D'où :

$$n > 2^{Hn(x)}$$

Et finalement :

$$n \geq 2^{hn(x)} - 1$$

Hauteur d'un arbre rouge-noir

Soit A un arbre rouge-noir de hauteur H et possédant n nœuds. On a :

$$H \leq 2 \log_2(n+1)$$

Démonstration

Soit H la hauteur d'un arbre rouge-noir, la moitié au moins des nœuds vers une sentinelle doit être noirs.

Donc la hauteur noire d'un arbre rouge-noir est au moins H/2.

$$H_n(\text{racine}) \geq H/2$$

Nous pouvons donc écrire :

$$n \geq 2^{Hn(racine)} - 1$$

$$n+1 \geq 2^{Hn(racine)}$$

ou : $$\log_2(n+1) \geq H_n(racine)$$

or : $$H_n(racine) \geq H/2$$

on déduit finalement: $$H \leq 2\log_2(n+1)$$

Insertion dans un arbre rouge-noir

-L'insertion dans un arbre rouge et noir commence par l'insertion naïve dans un ABR.

-Le **nouveau nœud** sera **rouge**.

Conséquences

-La propriété P5 est préservée.

-Par contre, la propriété P4 n'est plus nécessairement préservée.

-Si le père du nouveau nœud est également rouge, la propriété P4 est violée.

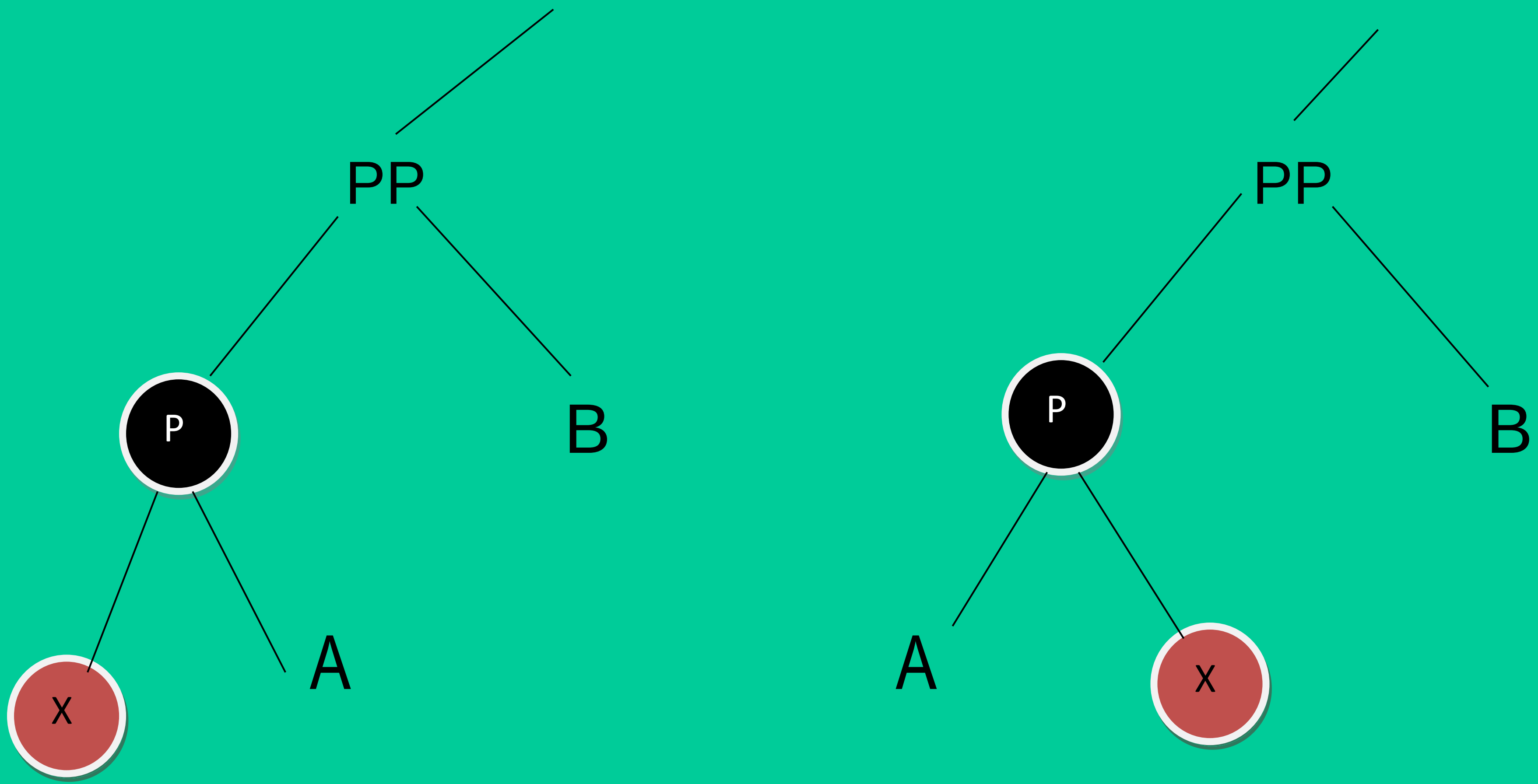
Afin de rétablir P4, l'algorithme modifie l'arbre à l'aide de **rotations**.

Le but de ces modifications est de rééquilibrer l'arbre.

Soit **x** le nœud et **p** son père.

L'algorithme distingue plusieurs cas.

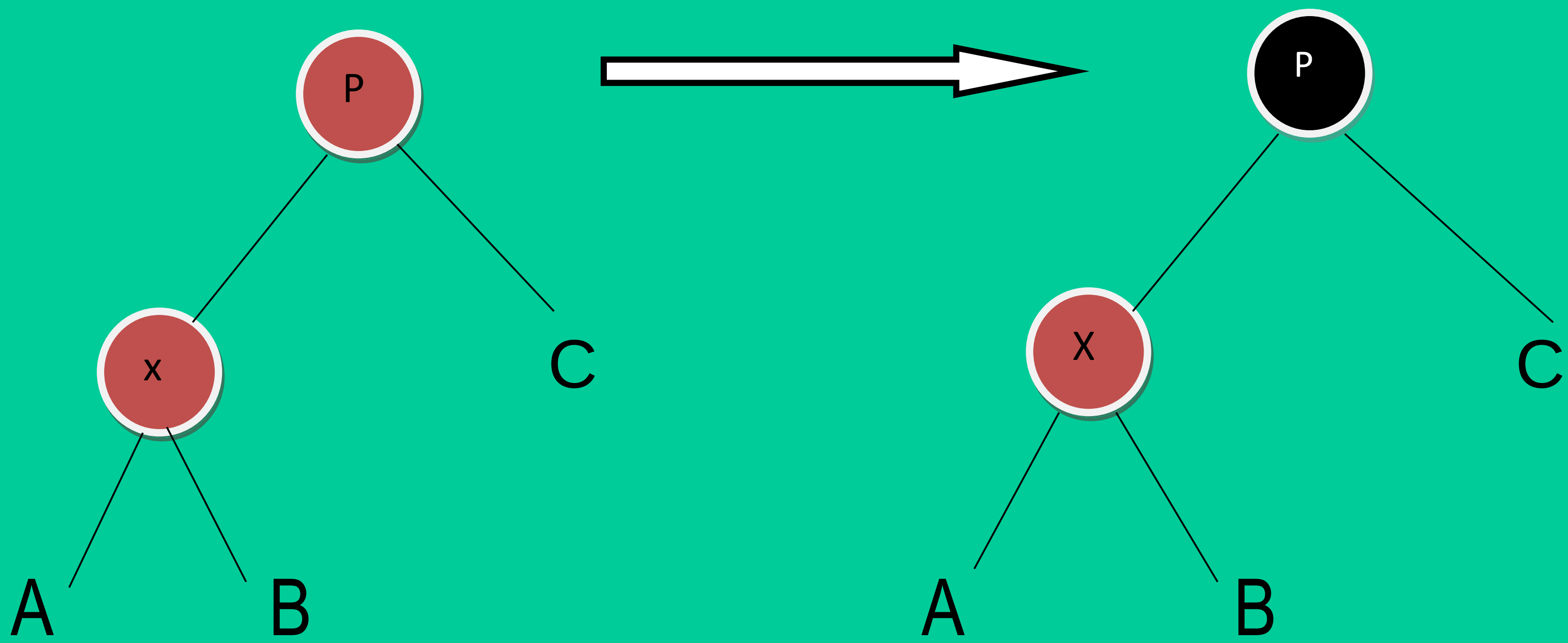
<u>Cas 0</u> :Le père **p** de x est **noir**



L’arbre est **toujours** un arbre rouge-noir.

<u>Cas 1</u>: Le père **p** est rouge

Mais **p** est racine de l'arbre



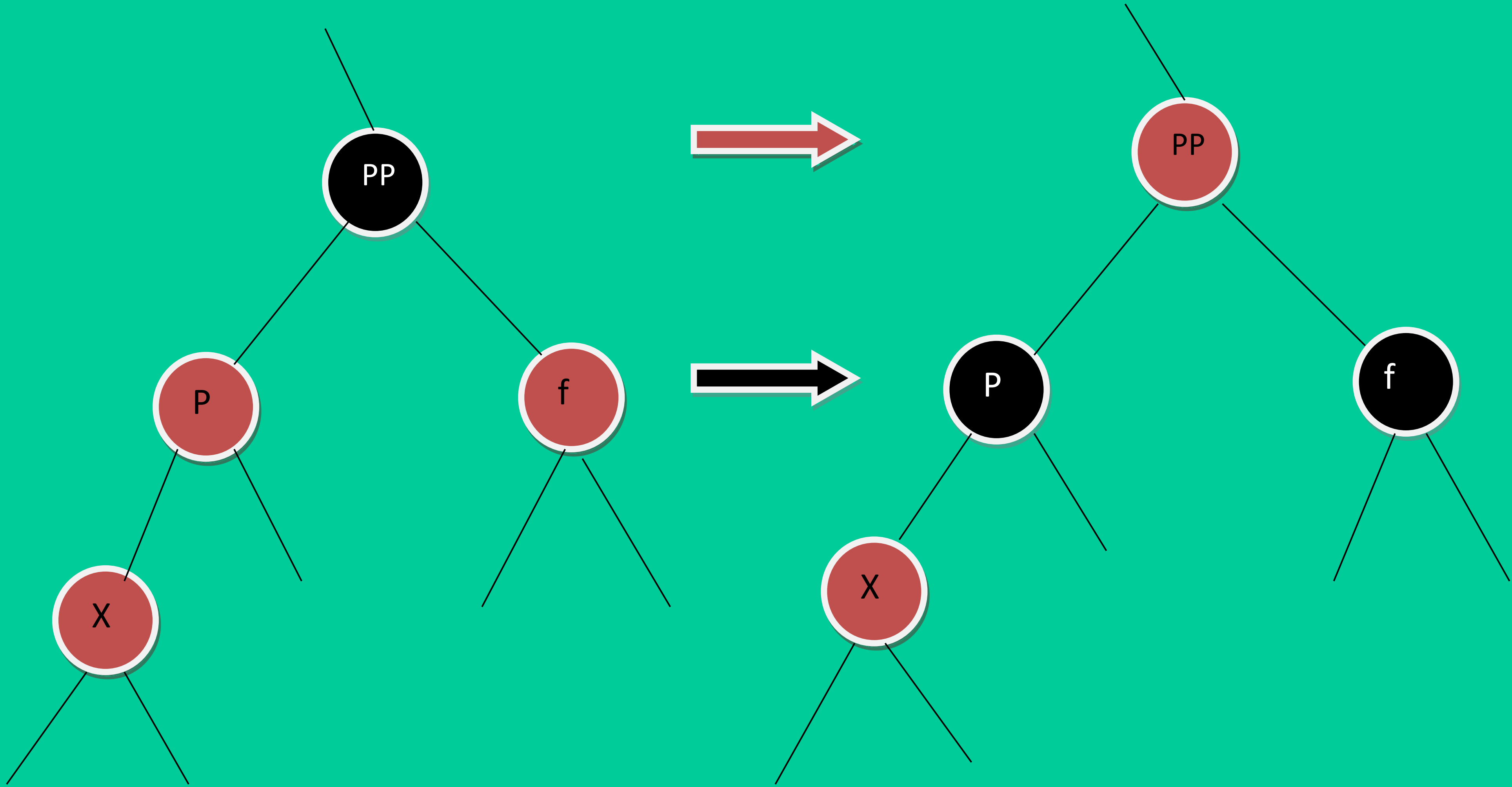
-Le nœud **p** devient alors **noir**.

-Les propriétés P2 et P4 sont maintenant vérifiées

-La propriété P5 le reste.

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

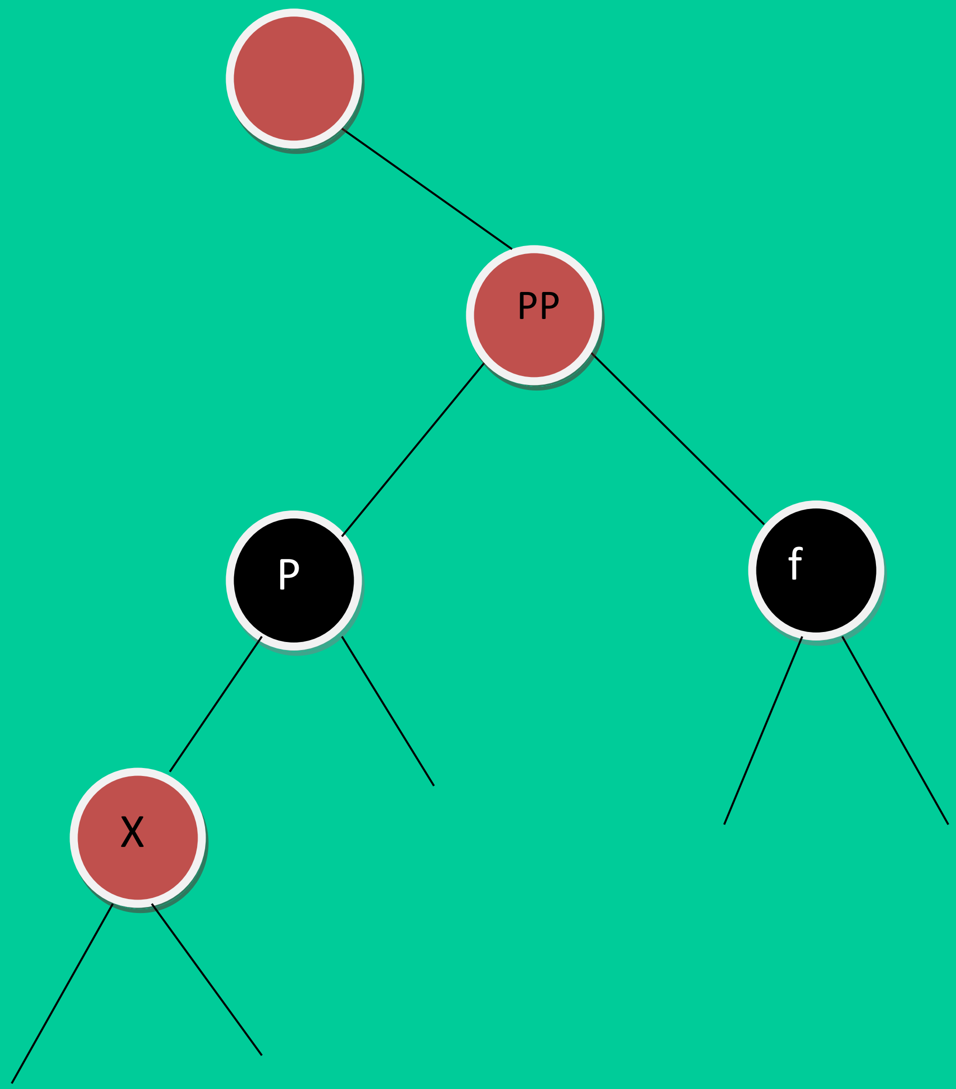
Cas 2 : **p** est rouge et n'est racine de l'arbre
Le frère **f** de **p** est rouge



-Les nœuds **p** et **f** deviennent **noirs** et leur père **pp** devient rouge.

-La propriété P5 reste vérifiée

- Mais la propriété P4 ne l'est pas si le père de **pp** est aussi rouge



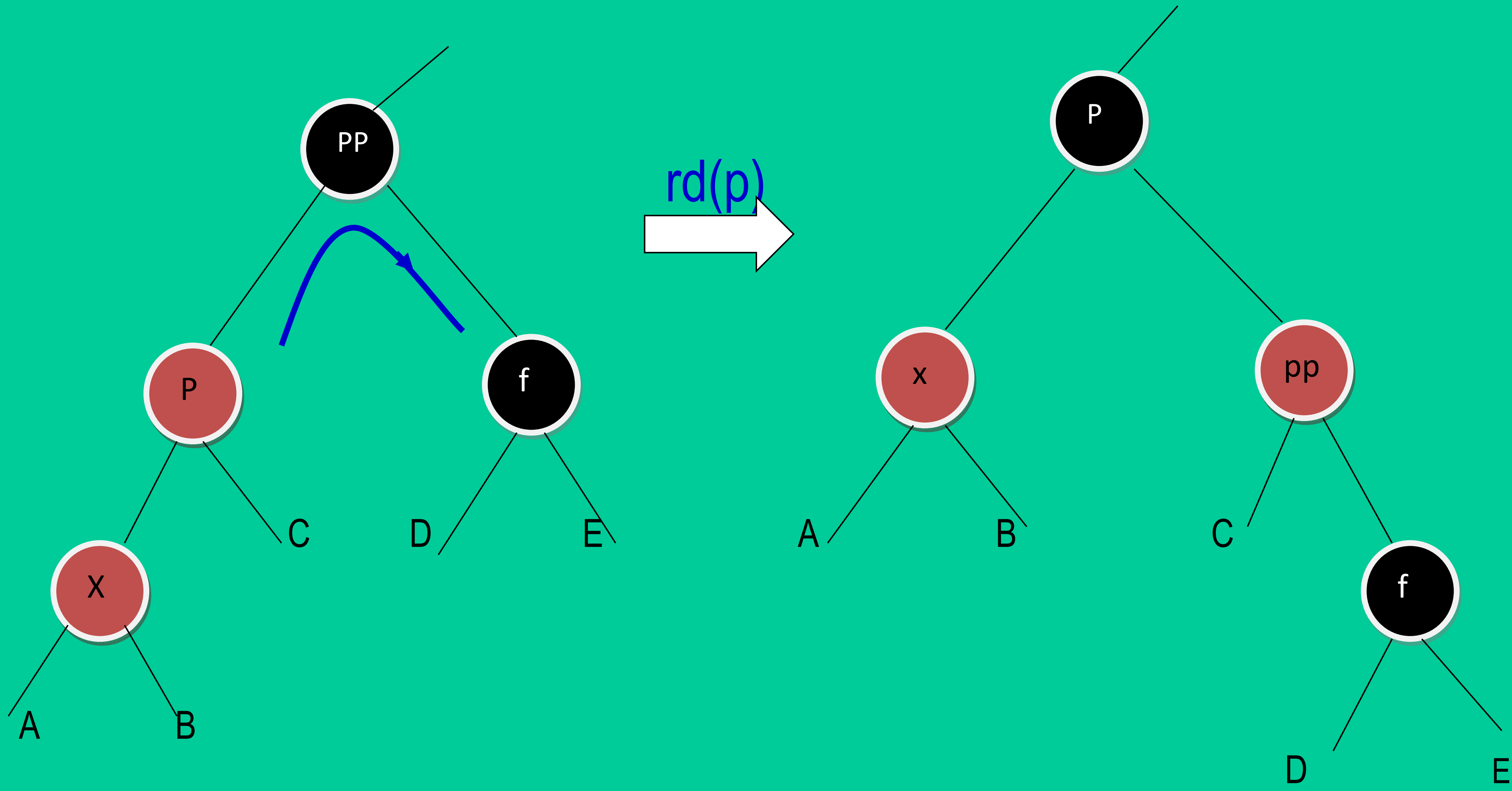
-Par contre, l'emplacement des deux nœuds rouges consécutifs s'est déplacé **vers la racine**. (voir cas 1)

Cas 3 : **p** est rouge et n’est racine de l'arbre
le frère **f** de **p** est **noir**

On suppose que **p** est le fils **gauche** de son père **pp**.
(sinon envisager la symétrie)

L'algorithme distingue deux sous-cas :
- x est le **fils gauche** de p: **cas 3a**
- x est le **fils droit** de p: **cas 3b**.

Cas 3a: x est le fils gauche de p.

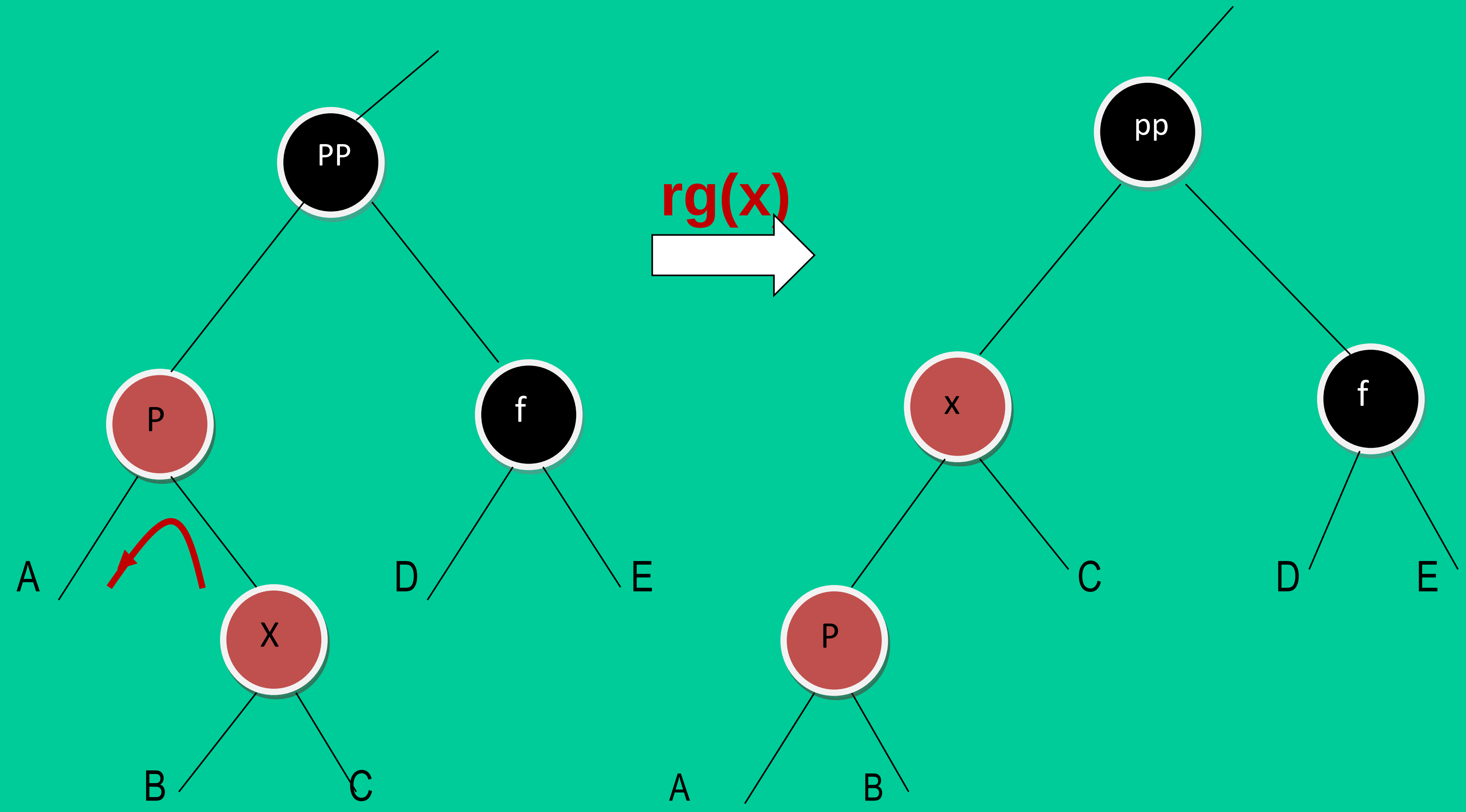


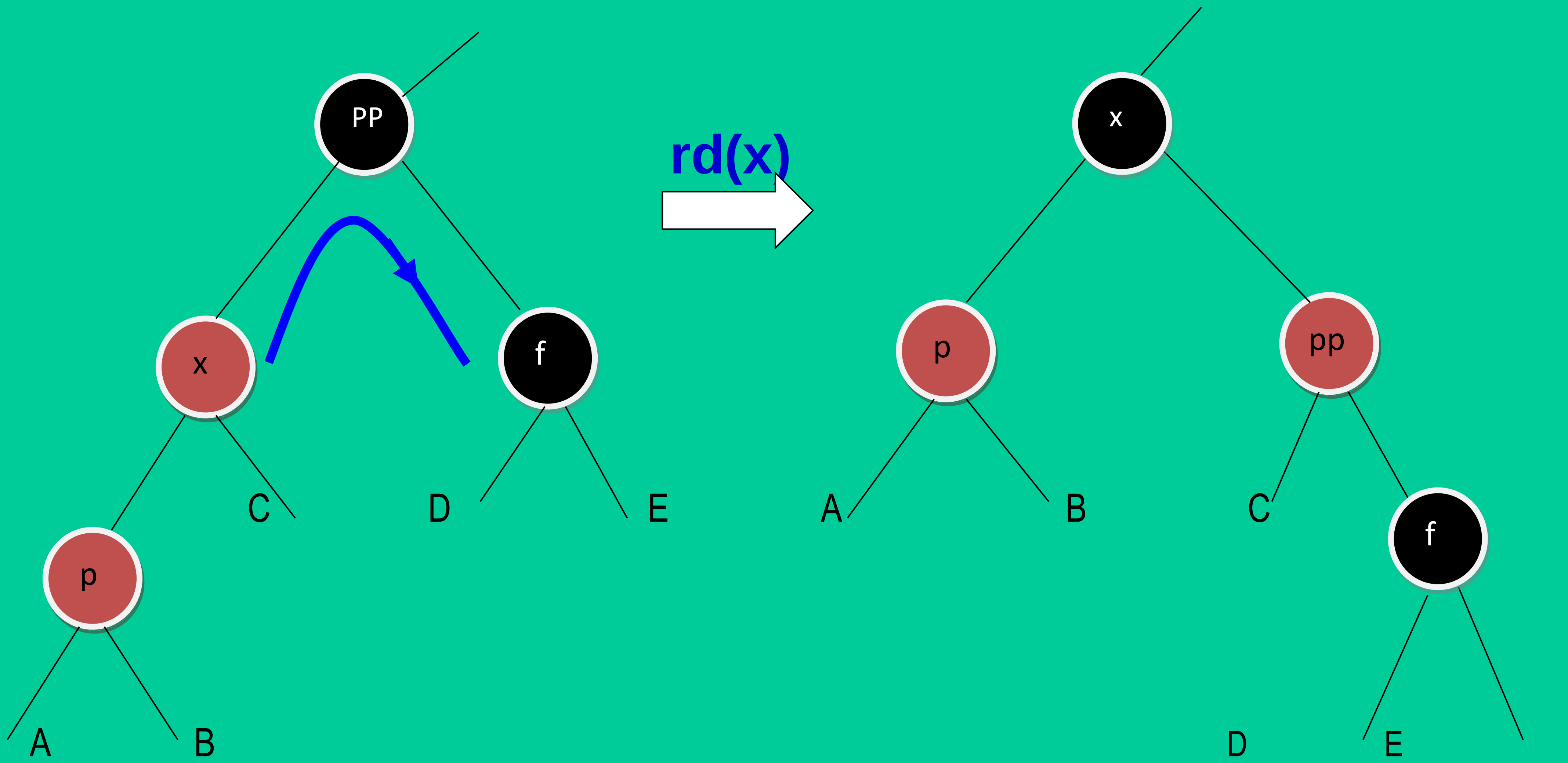
-L'algorithme effectue une rd(p)

-Ensuite, les nœuds **p** et **pp** changent de couleur.

-L'algorithme s'arrête puisque les propriétés P4 et P5 sont vérifiées.

<u>Cas 3b</u>: x est le fils droit de p.




PP
rd(x)
x
f
C
D
E
p
A
B
x
p
pp
A
B
C
f
D
E

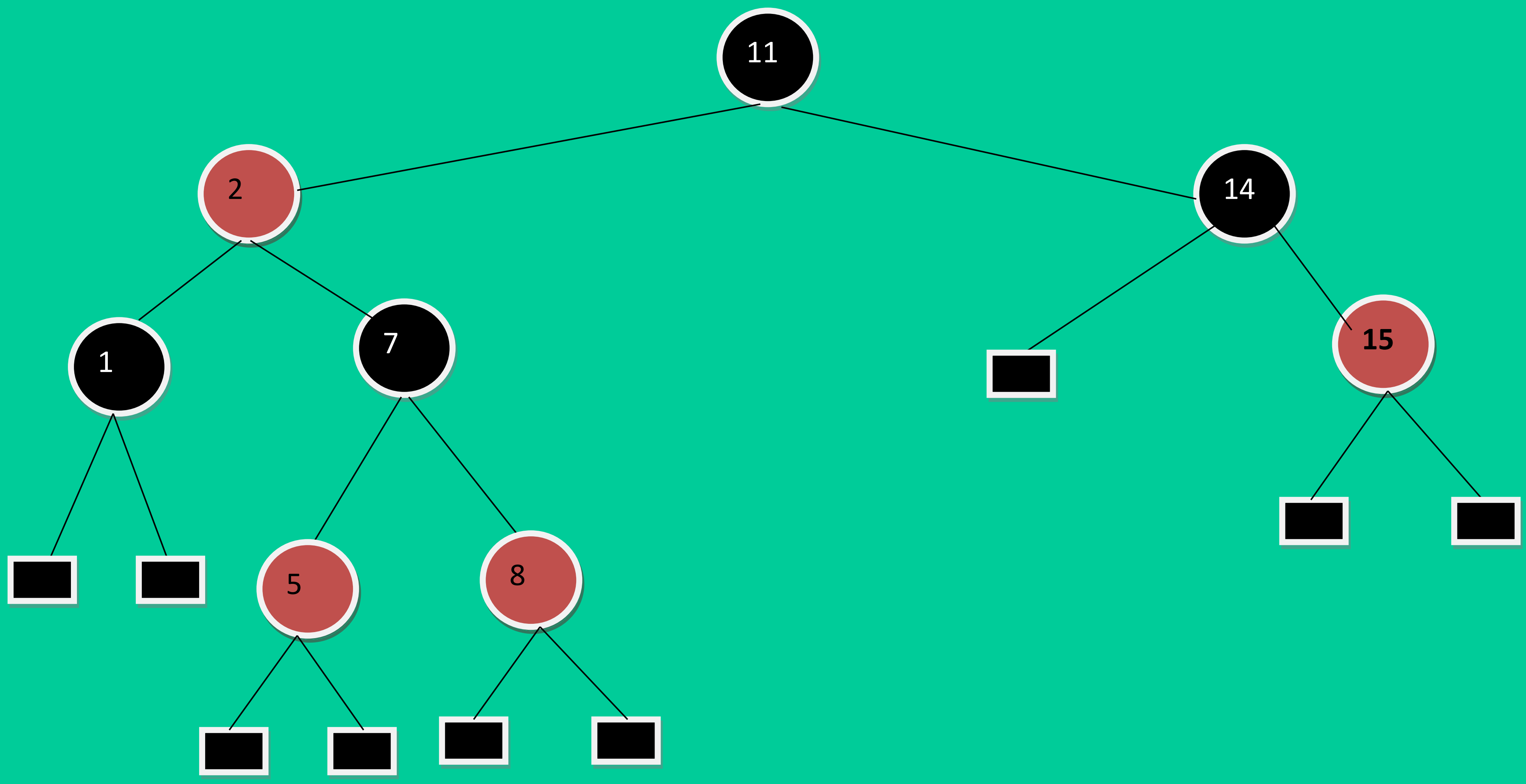
L'algorithme effectue une **rotation rg(x).**

Ensuite une **rotation rd(x).**

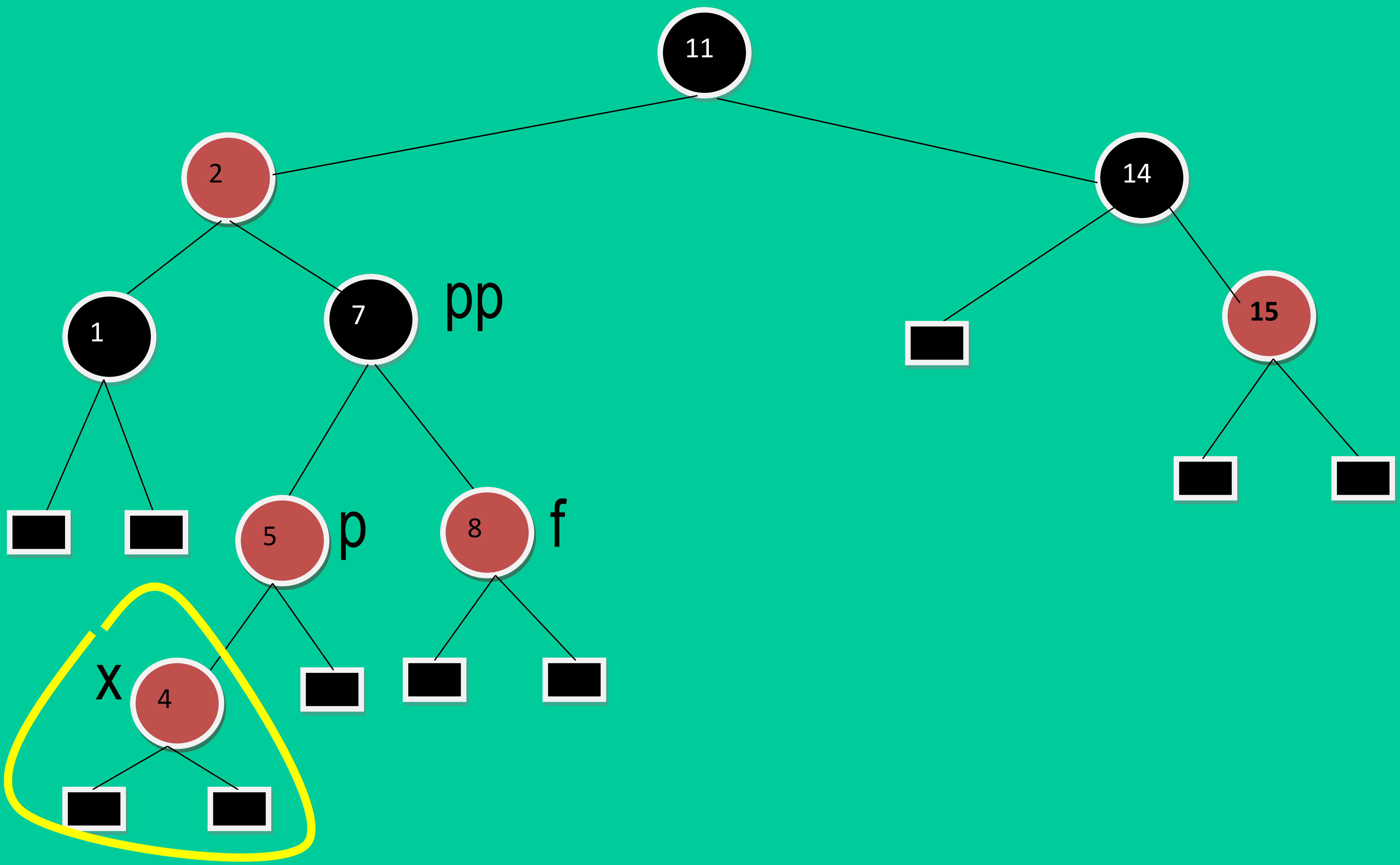
Les nœuds x et pp changent de couleur.

L'algorithme s'arrête: P4 et P5 sont vérifiées.

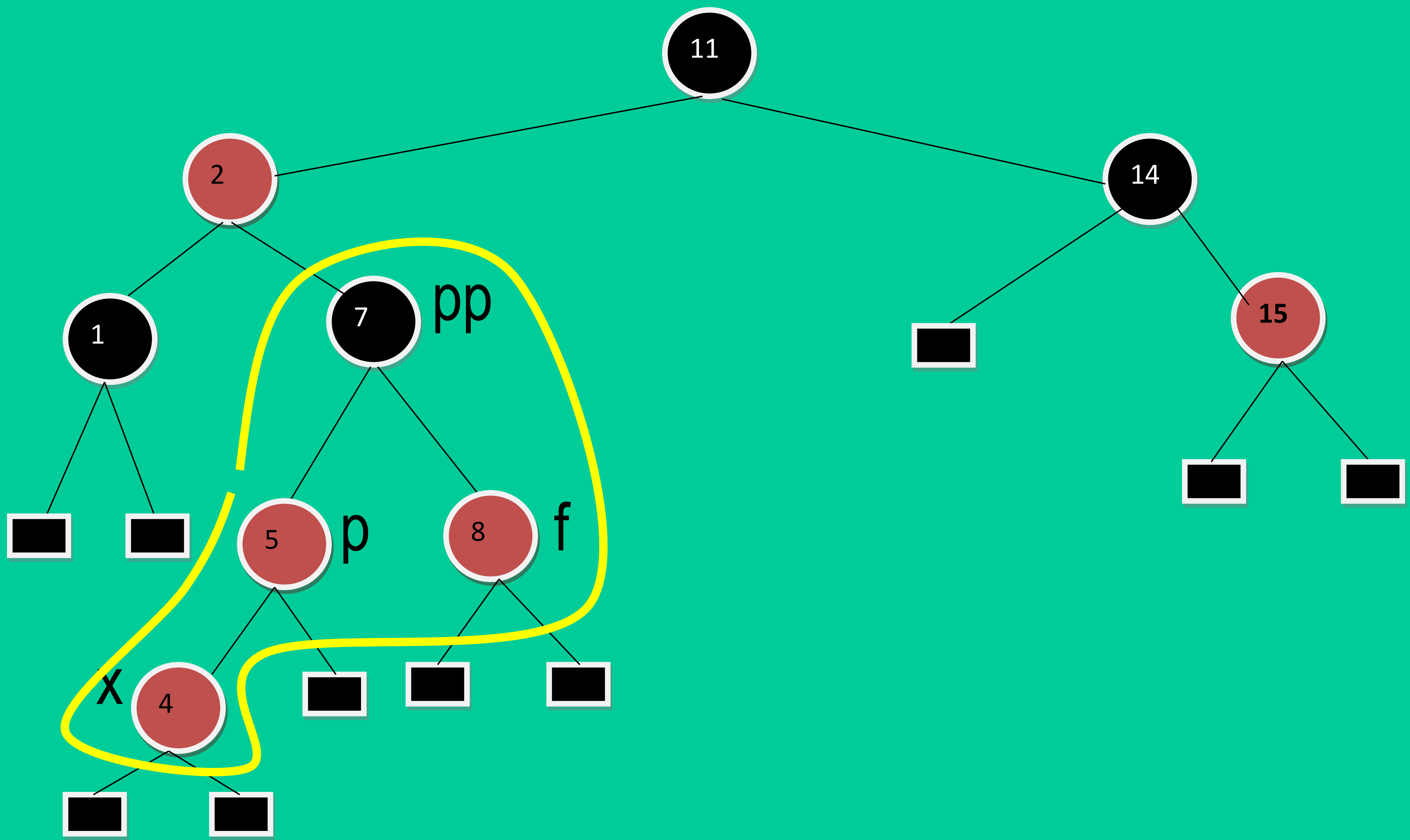
Exemple d'application

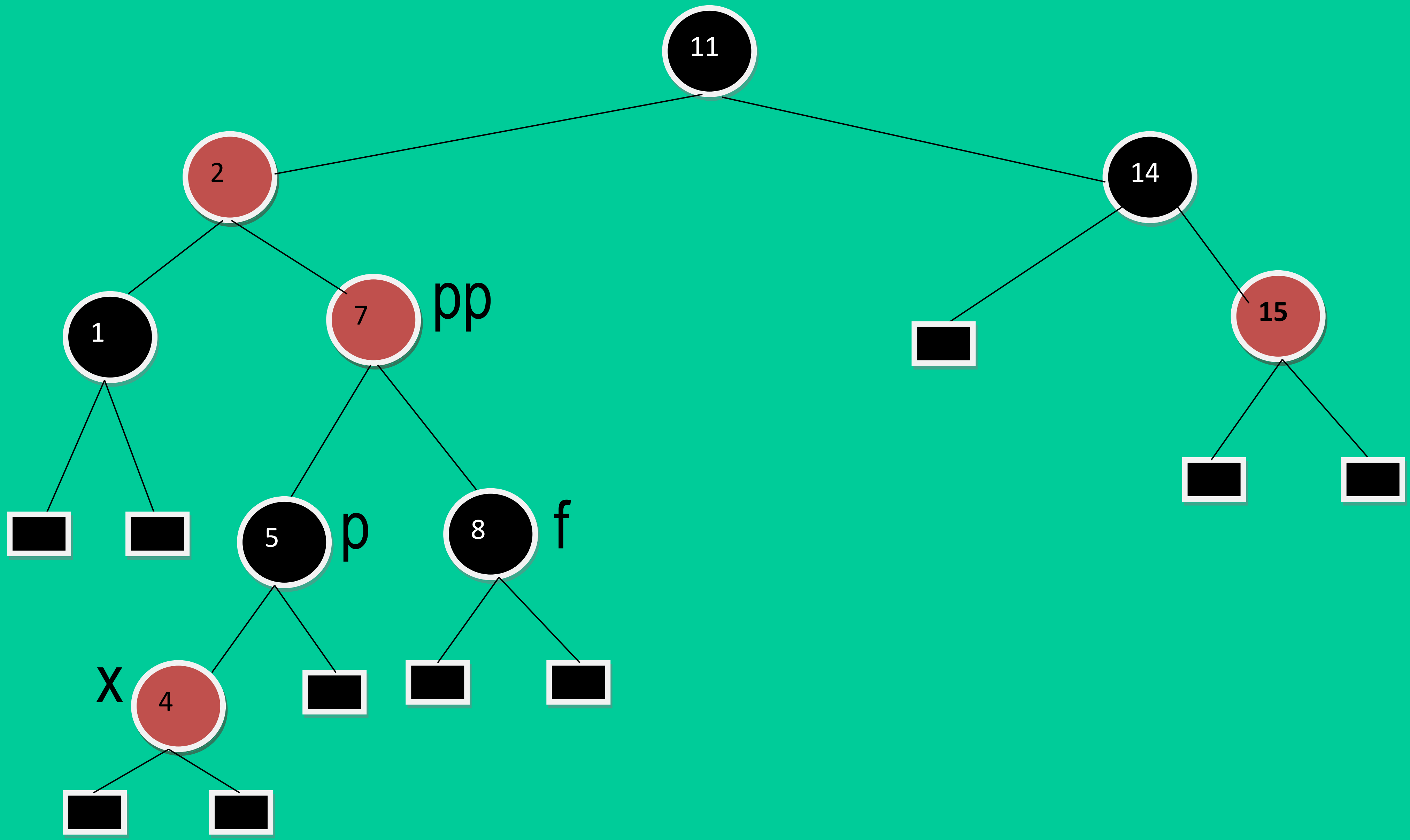


<u>Acte 0</u> : Ajouter 4 en rouge



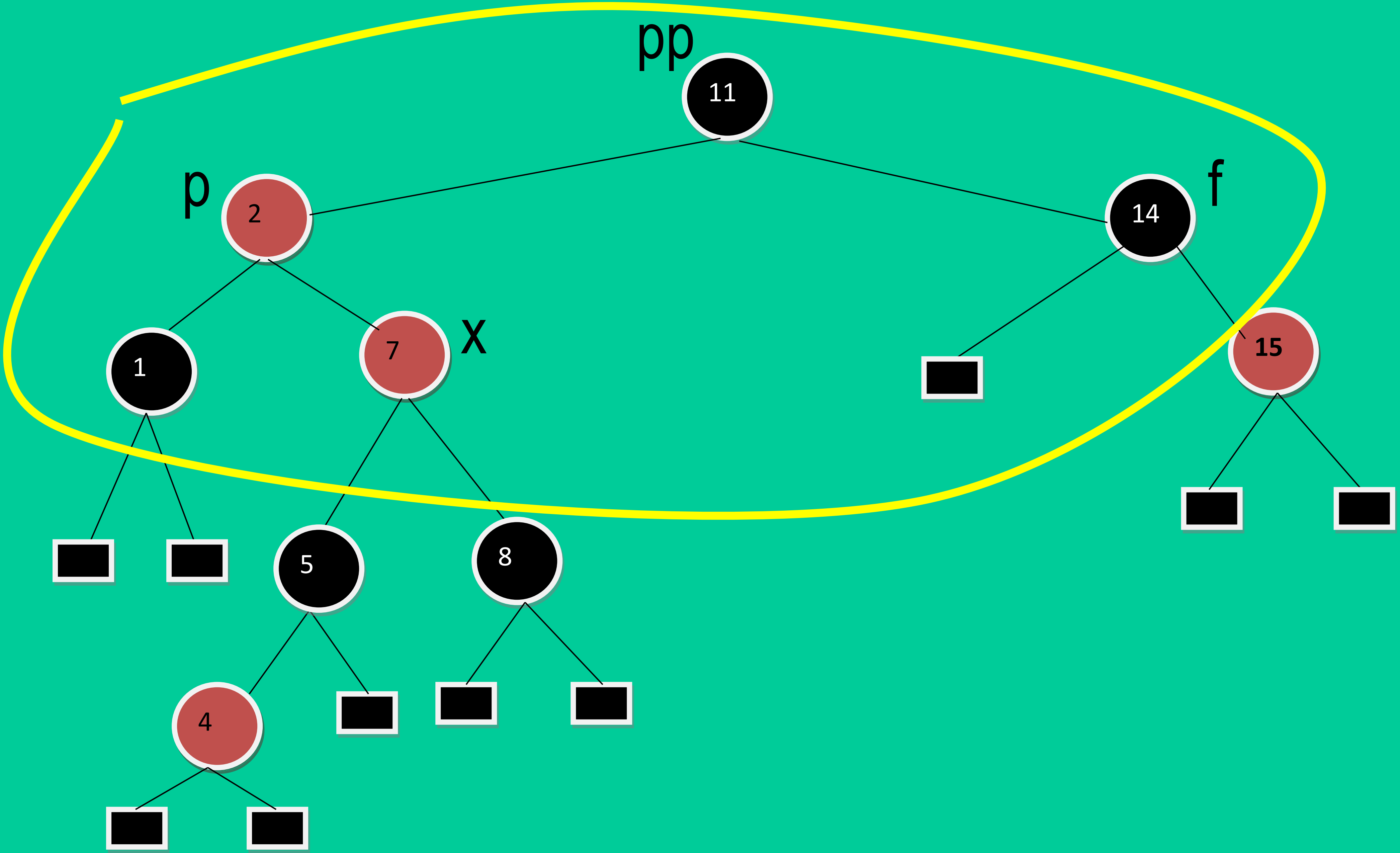
<u>Acte 1</u> : appliquer **Cas 2** : **p**, **pp** et **f** changent de couleur



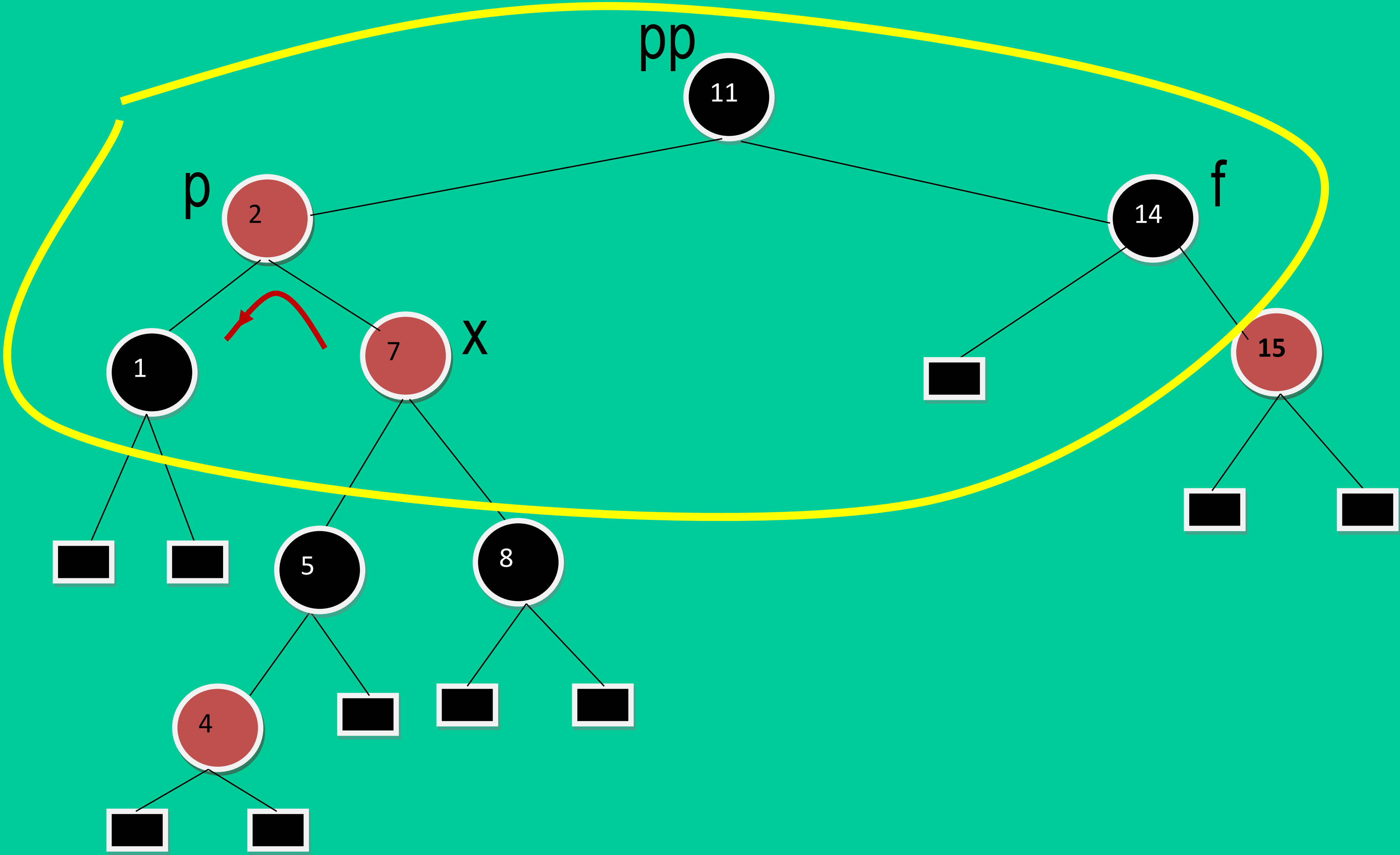


11
2
14
1
7
pp
15
5
p
8
f
x
4

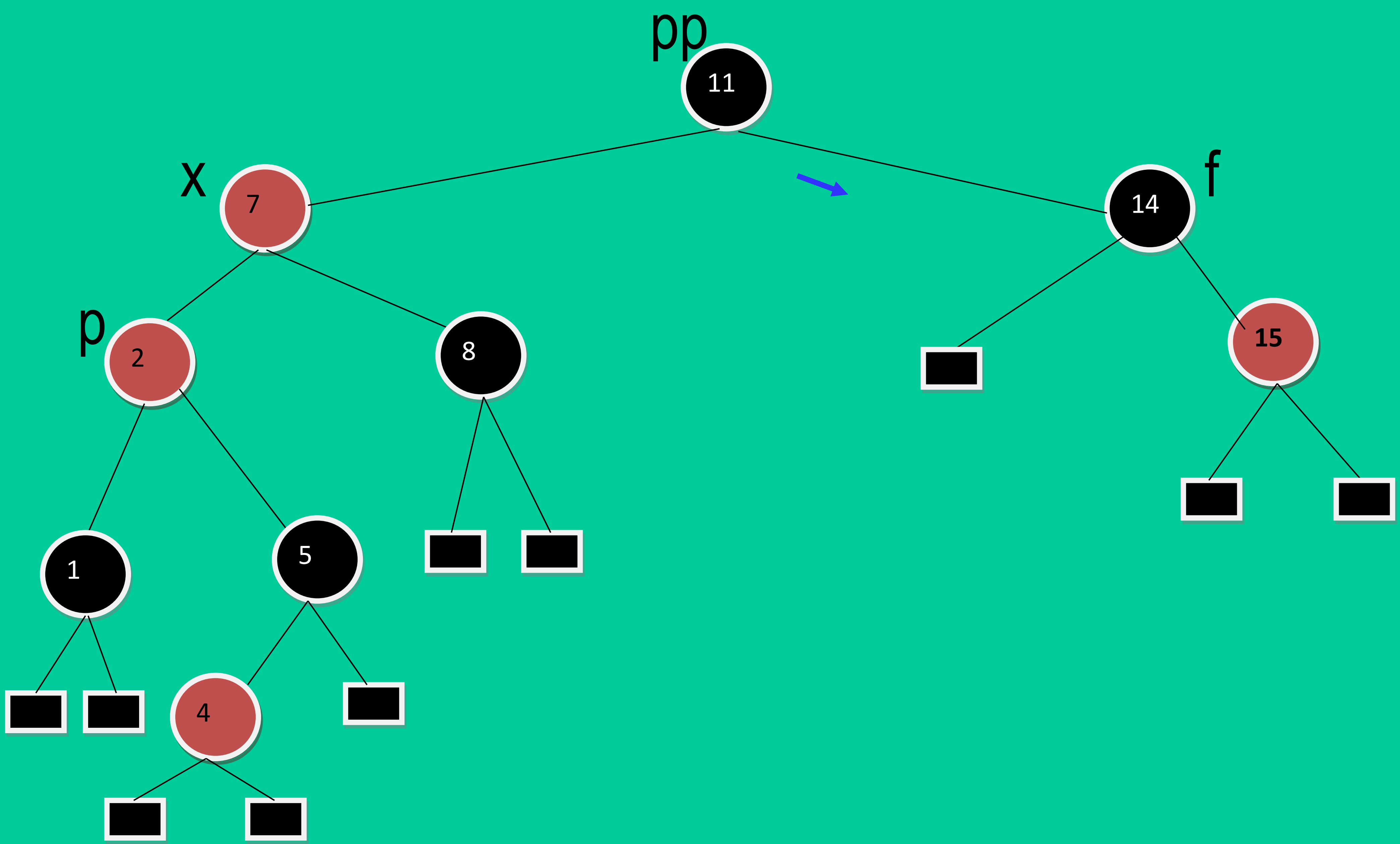
Acte 2 : appliquer Cas 3b



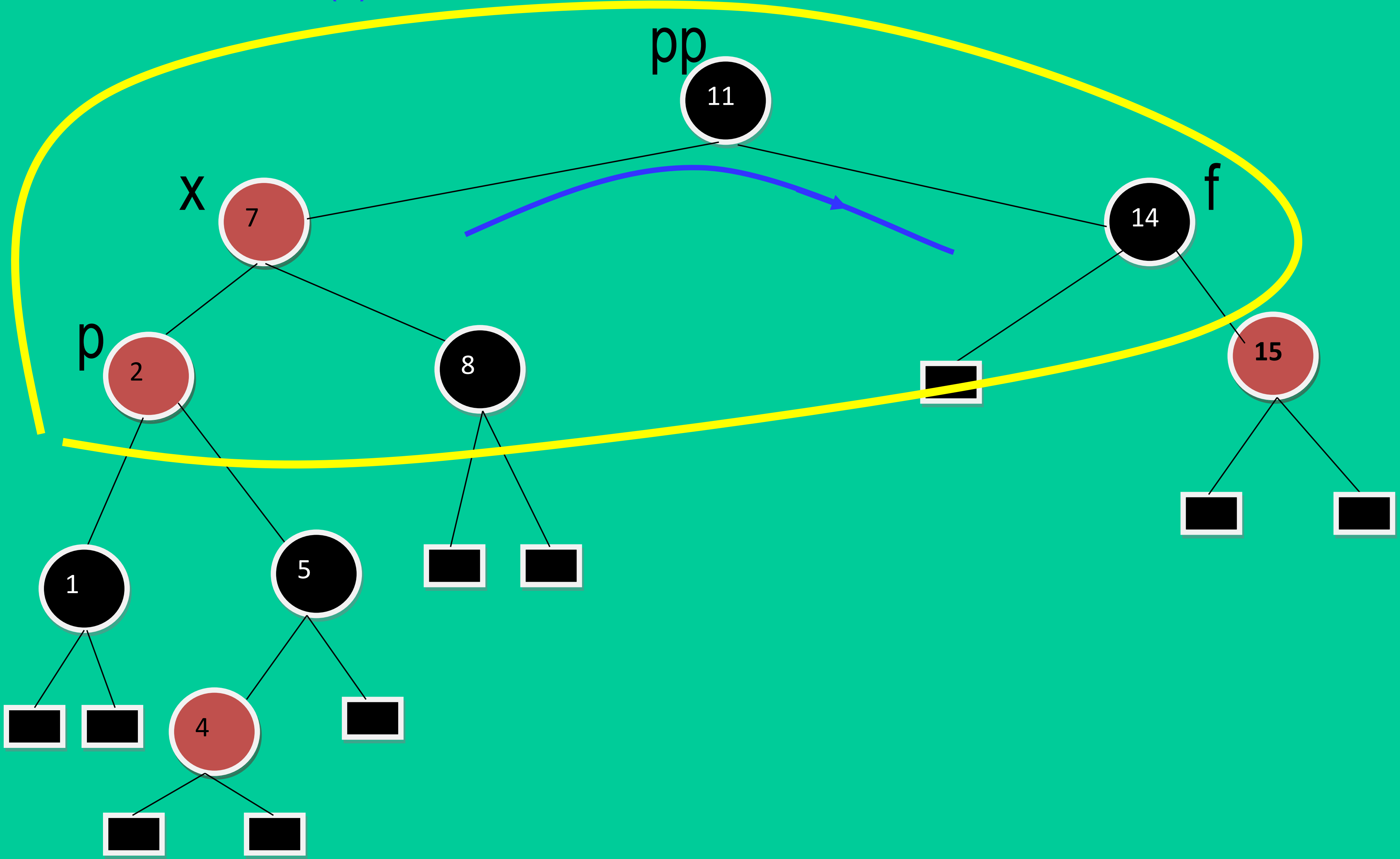
<u>Acte 2</u> : appliquer <u>Cas 3b</u> : rg(x) +…



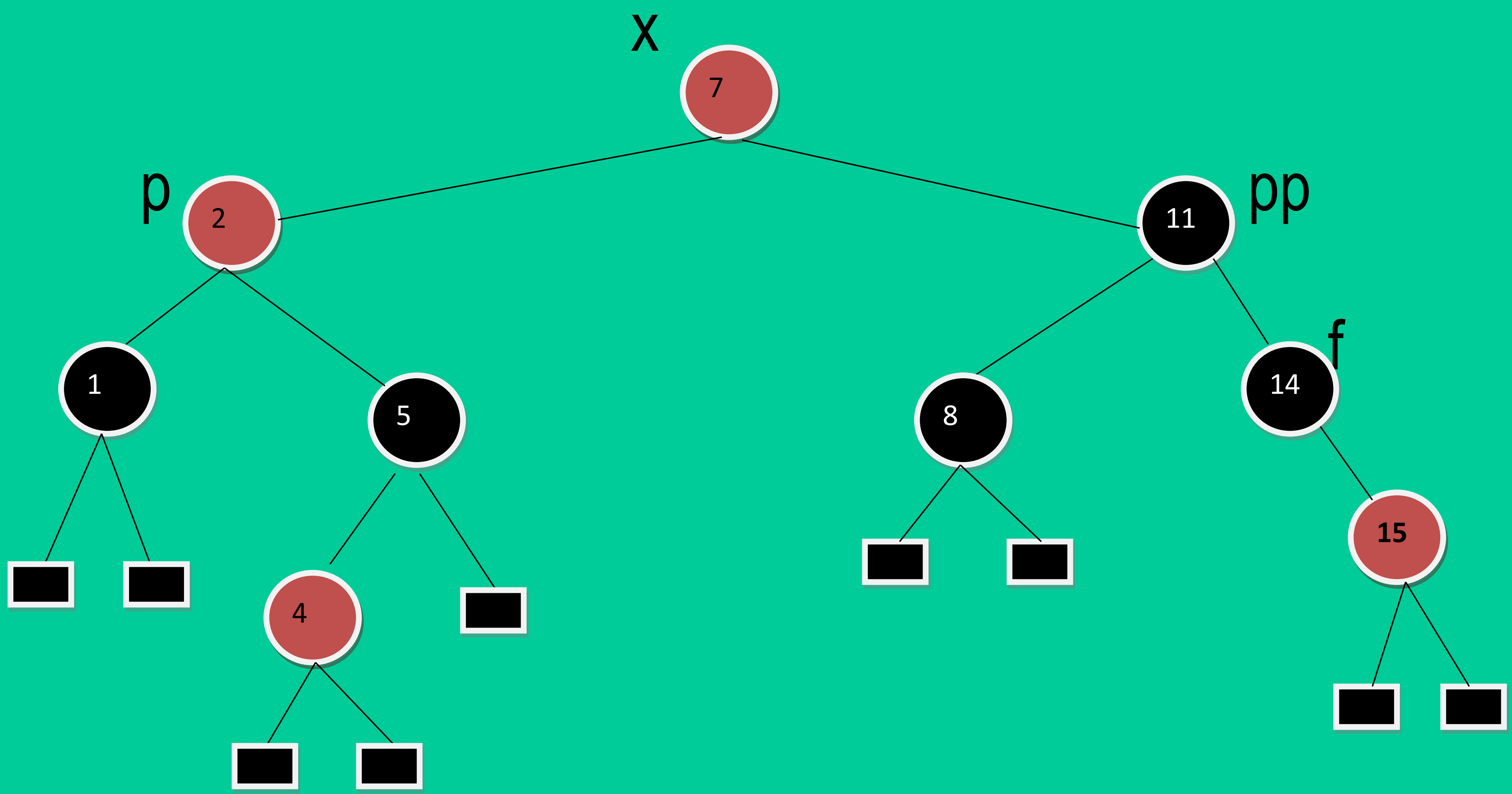
Cas 3b : ...+



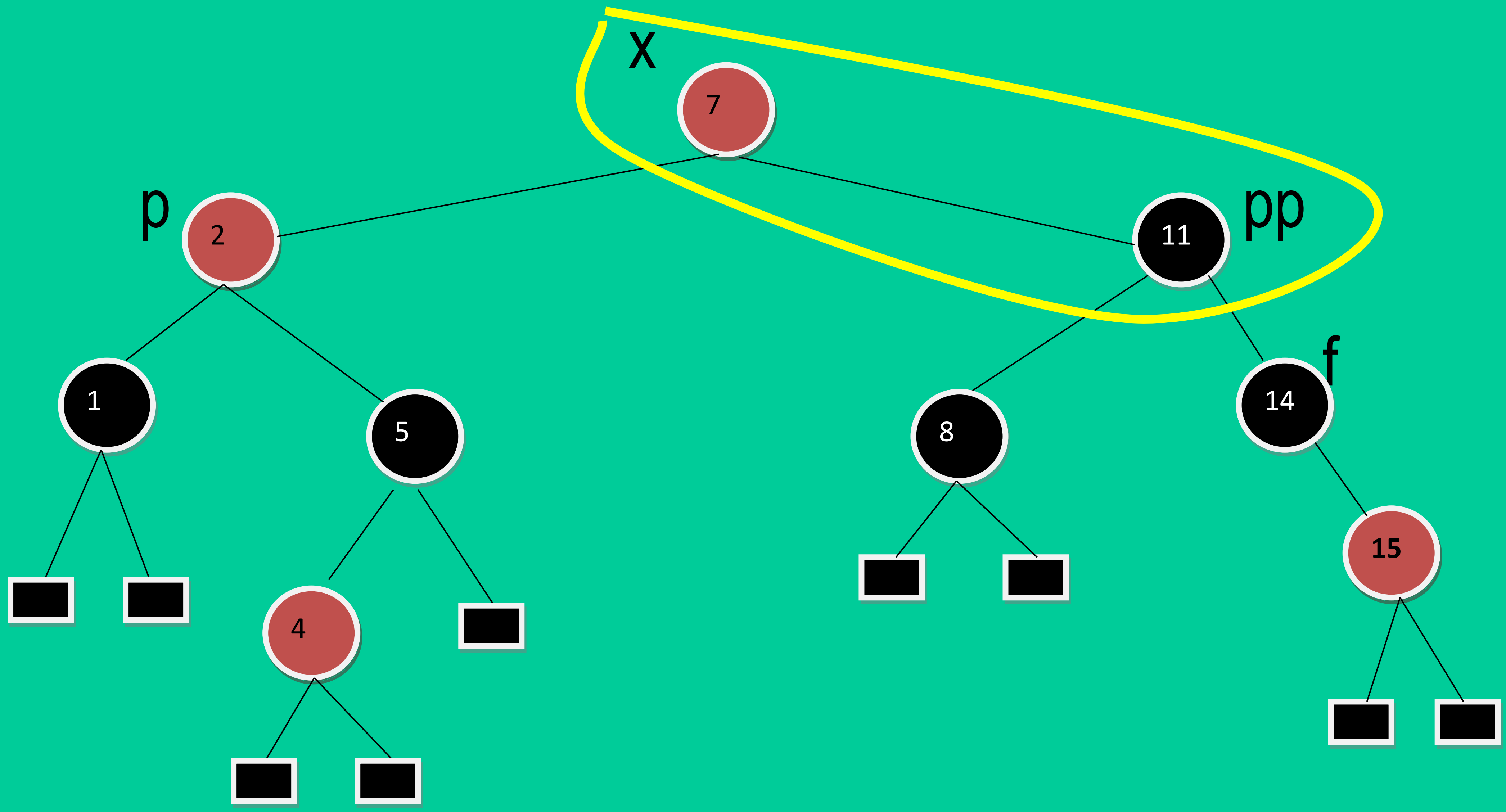
Cas 3b : …+ rd(x) +…



Cas 3b : ... + ...



Cas 3b : **…** + **x** et **pp** changent de couleur



Fin :arbre rouge noir contenant 4