

15 Arbres rouge-et-noir

15.1 Rang et coloriage

W(en)

On associe une valeur entière non-négative à chaque nœud. On va l'appeler le «rang» (avant de trouver un meilleur nom), dénoté par $\text{rang}(x)$. Le rang est croissant vers la racine, comme la hauteur, mais avec un équilibre permissif entre deux sous-arbres frères, parce que parfois le rang du père est le même que celui de l'enfant. Le but est de contrôler le déséquilibre (trop de rangs identiques mènent à une plus grande hauteur possible), mais pas excessivement (il faut ajuster la structure en $O(\log n)$ temps au pire à chaque opération). On va démontrer que les règles à côté accomplissent exactement ce but.

Règles.

1. Pour chaque nœud^a x excepté la racine,

$$\text{rang}(x) \leq \text{rang}(\text{parent}(x)) \leq \text{rang}(x) + 1.$$

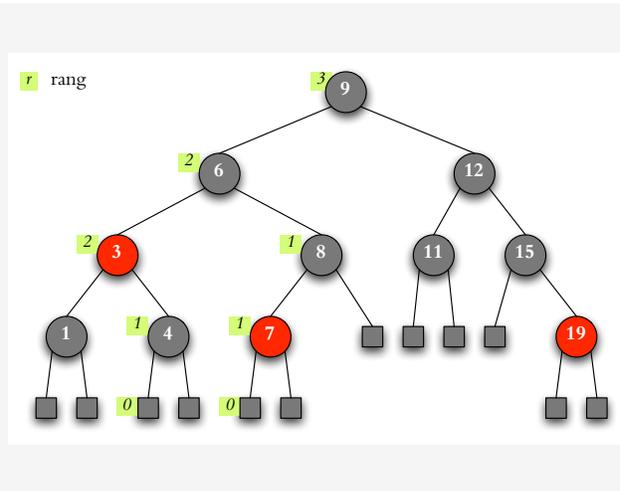
2. Pour chaque nœud x avec un grand-parent

$$\text{rang}(x) < \text{rang}(\text{parent}(\text{parent}(x))).$$

3. Pour chaque nœud externe (null) on a

$$\text{rang}(x) = 0 \quad \text{rang}(\text{parent}(x)) = 1.$$

^aPour les buts de la discussion, on considère les nœuds externes comme ayant une identité distincte, et donc $\text{parent}(x)$ donne le parent du nœud x même si x est externe. Dans le code on représente les nœuds externes par null, comme d'habitude.



Au lieu de stocker les rangs explicitement, il suffit de stocker la différence entre parent et enfant, en coloriant les nœuds par rouge ou noir.

- ★ si $\text{rang}(\text{parent}(x)) = \text{rang}(x)$, alors x est colorié par **rouge**
- ★ si x est la racine ou $\text{rang}(\text{parent}(x)) = \text{rang}(x) + 1$, alors x est colorié par **noir**

Théorème 15.1. Dans un coloriage valide,

- (0) chaque nœud est soit noir soit rouge
- (i) chaque nœud externe (null) est noir
- (ii) le parent d'un nœud rouge est noir
- (iii) tout chemin d'un nœud x à un nœud externe dans son sous-arbre contient le même nombre de nœuds noirs

Démonstration de Théorème 15.1. Propriété (0) \Leftrightarrow Règle 1, (i) \Leftrightarrow Règle 3, (ii) \Leftrightarrow Règle 2. En (iii), le nombre de nœuds noirs égale $\text{rang}(x)$, en justifiant l'appellation **hauteur noire**. ■

Lemme 15.2. Pour chaque nœud x , sa hauteur $h(x) \leq 2 \cdot \text{rang}(x)$.

Démonstration. Sur un chemin jusqu'à un nœud externe, il y a au moins autant de nœuds noirs que des rouges. ■

Lemme 15.3. Le nombre de descendants internes de chaque nœud x est $\geq 2^{\text{rang}(x)} - 1$.

Démonstration. Par induction [dans la hauteur de x]. Le théorème est vrai pour un nœud externe x quand $\text{rang}(x) = 0$. Supposons que le théorème est vrai pour tout x avec une hauteur $h(x) < k$. Considérons un nœud x avec $h(x) = k$ et ses deux enfants u, v avec $h(u), h(v) < k$. Par l'hypothèse d'induction, le nombre des descendants de x est $\geq 1 + (2^{\text{rang}(u)} - 1) + (2^{\text{rang}(v)} - 1) = 2^{\text{rang}(u)} + 2^{\text{rang}(v)} - 1$. Or, $\text{rang}(x) - 1 \leq \text{rang}(u), \text{rang}(v)$, ou $2^{\text{rang}(u)} + 2^{\text{rang}(v)} \geq 2^{\text{rang}(x)}$. Le théorème reste donc vrai pour x avec $h(x) = k$, et, en conséquence pour tout x . ■

Théorème 15.4. La hauteur d'un arbre RN avec n nœuds internes est bornée comme $\lg(n+1) \leq h \leq 2\lg(n+1)$.

Démonstration. La borne inférieure correspond à l'arbre binaire complet. La borne supérieure vient de 15.2 et 15.3 :

$$2^{\text{rang}(\text{racine})} - 1 \leq n \quad \text{par 15.3}$$

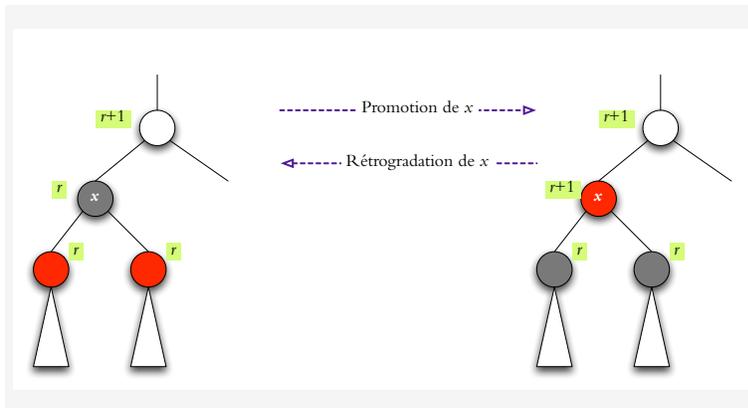
$$\text{rang}(\text{racine}) \leq \lg(n+1)$$

$$h/2 \leq \lg(n+1) \quad \text{par 15.2}$$

■

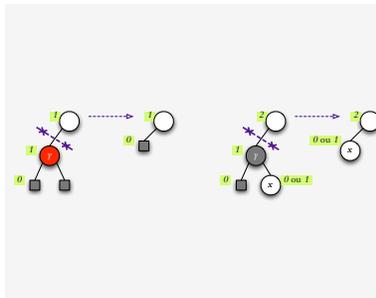
15.2 Arbre RN : temps de calcul des opérations

On ajuste la structure lors d'une insertion ou suppression de nœud en parcourant un chemin vers la racine, en $O(h)$ temps pour un arbre de hauteur h . Par Théorème 15.4, la hauteur d'un arbre RN est toujours $h = \Theta(\log n)$, donc toutes les opérations s'exécutent en $O(\log n)$, même dans le pire cas.



Ajustement de la structure. Pour maintenir l'équilibre, on utilise les **rotations** comme avant + **promotion/rétrogradation** (incrémenter ou décrementer le rang par 1).
 → promotion/rétrogradation change la couleur d'un nœud et ses enfants.
 → on ne peut promouvoir x que s'il est noir avec deux enfants rouges (pour ne pas violer Règle 1 et Propriété (ii) de Théorème 15.1)

Insertion. On insère x avec $\text{rang}(x) = 1 \Rightarrow$ sa couleur est rouge. **Test** : est-ce que le parent de x est rouge ? Si oui, on a un problème ; sinon, rien à faire (cas 0a). Solution : examiner le grand-parent $y = \text{parent}(\text{parent}(x))$: il est forcément noir par Propriété (ii) de Théorème 15.1. Selon le coloriage des enfants de y (le parent et l'oncle de x), on fait une série de promotions (cas 0b : oncle rouge), suivie par une ou deux rotations (cas 1 ou 2 : oncle noir), comme les zig-zag, zag-zig, etc., en déploiement.



Suppression. Pour la suppression, on utilise une technique similaire : procéder comme avec l'arbre binaire de recherche, puis retrogradations en ascendant vers la racine + $O(1)$ rotations (trois au plus) à la fin. L'ajustement de la structure départ par l'examen du nœud y physiquement enlevé (donc, c'est le successeur ou prédécesseur si on a supprimé la clé d'un nœud avec deux enfants non-null). Le nœud y est remplacé par x qui est un des enfants de y . Selon la différence de rangs entre x et y (donc couleur de x), on fait une série de rétrogradations, et quelques rotations.

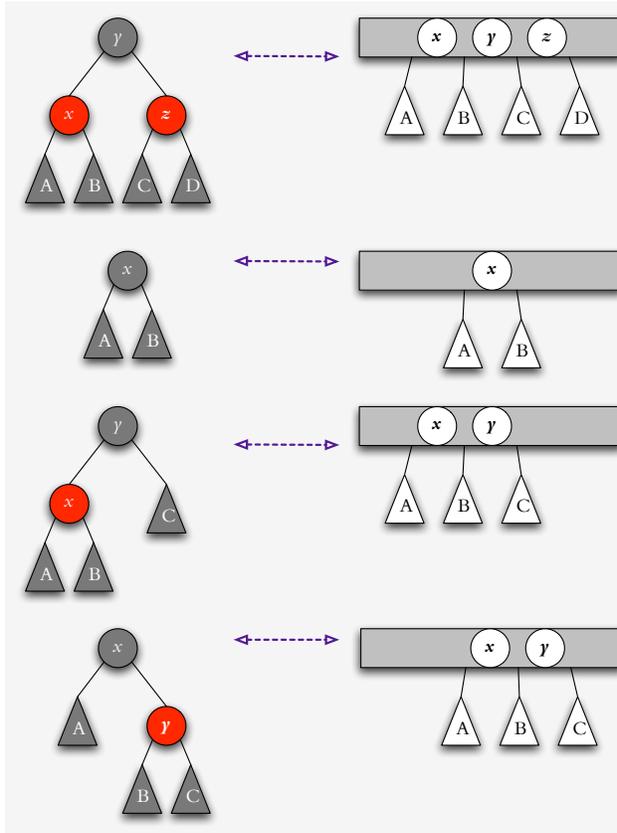
INSERTION

<p>Cas 0a : parent noir</p>	<p>Cas 0b : parent rouge, oncle rouge</p>
<p>Cas 1 : parent rouge, oncle noir, zig-zig</p> <p>rien à faire</p> <p>une rotation simple</p>	<p>Cas 2 : parent rouge, oncle noir, zig-zag</p> <p>promotion du grandparent, continuer avec $x \leftarrow y$</p> <p>une rotation double</p>

SUPPRESSION

<p>Cas 0 : nœud rouge x — il devient noir, et on arrête</p> <p>Cas 1a : sœur noire, neveux noirs</p> <p>retrogradation du parent, continuer avec $x \leftarrow y$ en cas 0, 1, ou 2</p>	<p>Cas 1b : sœur noire, neveu distant rouge</p> <p>une rotation simple</p>
<p>Cas 1c : sœur noire, neveu proche rouge</p> <p>une rotation double</p>	<p>Cas 2 : sœur rouge</p> <p>une rotation simple, continuer avec x en cas 1 (pas de récurion en 1a)</p>

15.3 Arbre 2-3-4



En fusionnant les nœuds rouges avec leurs parents noirs, on obtient un **arbre 2-3-4** : c'est un arbre de recherche *non-binaire* où chaque nœud fusionné comprend

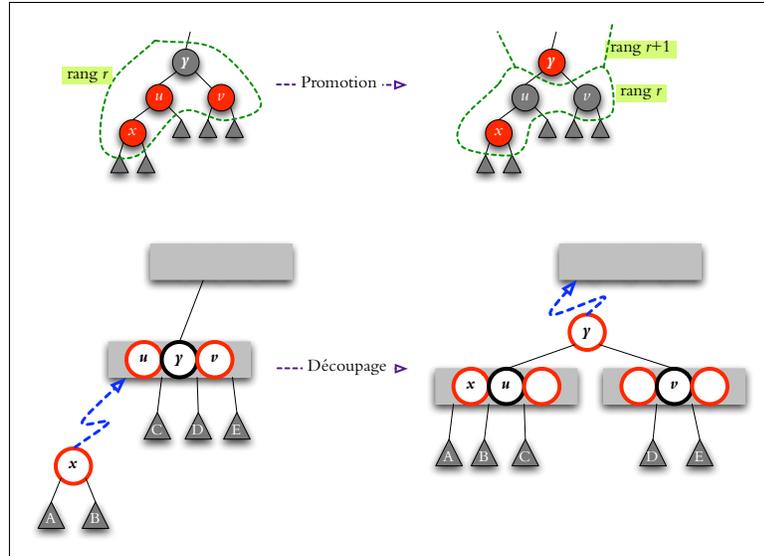
- i. soit 1 clé et 2 enfants (un seul nœud noir),
- ii. soit 2 clés et 3 enfants (nœud noir avec un enfant rouge),
- iii. soit 3 clés et 4 enfants (nœud noir avec deux enfants rouges),

Par la propriété d'équilibre chez les arbres rouge-et-noir, toutes les feuilles de l'arbre 2-3-4 sont au même niveau (une feuille est un nœud interne dont tous les enfants sont externes)

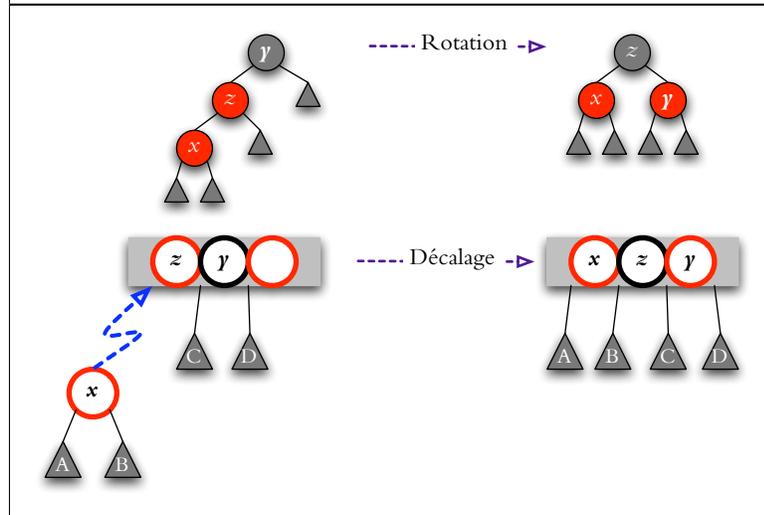
15.4 Insertion dans un arbre 2-3-4

On peut adapter les opérations de l'arbre rouge-et-noir à l'arbre 2-3-4. Par exemple : qu'est-ce qui se passe lors d'une insertion ? On crée un nœud rouge, on performe des promotions en ascendant vers la racine, avec une rotation possible à la fin. Ceci correspond à une séquence de découpages, suivi par un décalage dans le cas de l'arbre 2-3-4.

promotion correspond à **découpage** («éclatement») en ascendant



rotation correspond à **décalage**



promotion de la racine correspond à la création d'une **nouvelle racine**

