16 Arbres rouge-et-noir

16.1 Rang et coloriage

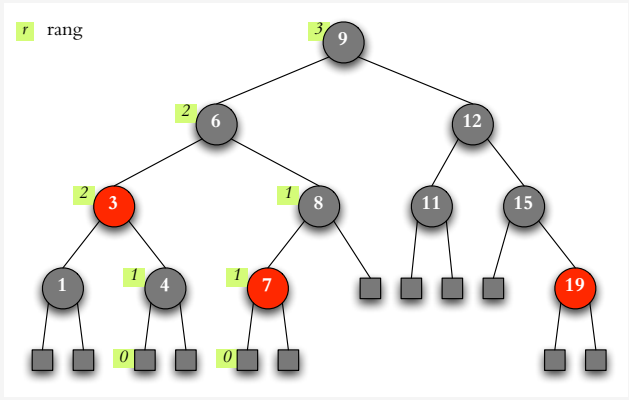
W(en)

On associe une valeur entière non-négative à chaque nœud. On va l'appeler le «rang» (avant de trouver un meilleur nom), dénoté par $\mathsf{rang}(x)$. 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,
$$\mathsf{rang}(x) \le \mathsf{rang}(\mathsf{parent}(x)) \le \mathsf{rang}(x) + 1.$$
2. Pour chaque nœud x avec un grand-parent
$$\mathsf{rang}(x) < \mathsf{rang}\big(\mathsf{parent}(\mathsf{parent}(x))\big).$$
3. Pour chaque nœud externe (null) on a
$$\mathsf{rang}(x) = 0 \qquad \mathsf{rang}(\mathsf{parent}(x)) = 1.$$

[a] Pour les buts de la discussion, on considère les nœuds externes comme ayant une identité distincte, et donc $\mathsf{parent}(x)$ 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 $\mathsf{rang}(\mathsf{parent}(x)) = \mathsf{rang}(x)$, alors x est colorié par **rouge**
- si x est la racine ou $\mathsf{rang}(\mathsf{parent}(x)) = \mathsf{rang}(x)+1$, alors x est colorié par **noir**

Théorème 16.1. *Dans un coloriage valide,*
(0) chaque nœud est soit noir soit rouge
*(i) chaque nœud externe (*null*) est noire*
(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 16.1. Propriété (0) $\Leftrightarrow$ Règle 1, (i) $\Leftrightarrow$ Règle 3, (ii) $\Leftrightarrow$ Règle 2. En (iii), le nombre de nœuds noirs égale $\mathsf{rang}(x)$, en justifiant l'appellation **hauteur noire**. ■

Lemme 16.2. *Pour chaque nœud x, sa hauteur $h(x) \le 2 \cdot \mathsf{rang}(x)$.*

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

Lemme 16.3. *Le nombre de déscendants internes de chaque nœud x est $\ge 2^{\mathsf{rang}(x)} - 1$.*

Démonstration. Par induction [dans la hauteur de x]. Le théorème est vrai pour un nœud externe x quand $\mathsf{rang}(x) = 0$. 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 $\geq 1+(2^{\mathsf{rang}(u)}-1)+(2^{\mathsf{rang}(v)}-1) = 2^{\mathsf{rang}(u)}+2^{\mathsf{rang}(v)}-1$. Or, $\mathsf{rang}(x)-1 \leq \mathsf{rang}(u), \mathsf{rang}(v)$, ou $2^{\mathsf{rang}(u)}+2^{\mathsf{rang}(v)} \geq 2^{\mathsf{rang}(x)}$. Le théorème reste donc vrai pour x avec $h(x) = k$, et, en conséquence pour tout x. ■

Théorème 16.4. *La hauteur d'un arbre RN avec n nœuds internes est bornée comme* $\lg(n+1) \leq h \leq 2\lg(n+1)$.

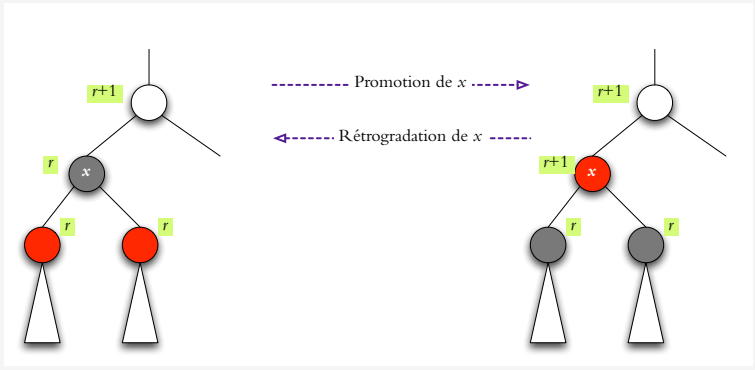
Démonstration. La borne inférieure correspond à l'arbre binaire complet. La borne supérieure vient de 16.2 et 16.3 :

$$\begin{aligned} 2^{\mathsf{rang(racine)}} - 1 &\leq n && \text{par 16.3} \\ \mathsf{rang(racine)} &\leq \lg(n+1) \\ h/2 &\leq \lg(n+1) && \text{par 16.2} \end{aligned}$$

■

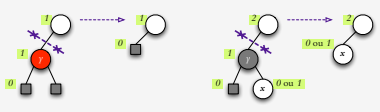
16.2 Arbre RN : temps de calcul des opérations

On ajuste la structure lors d'une insertion ou suppression de noeud en parcourant un chemin vers la racine, en $O(h)$ temps pour un arbre de hauteur h. Par Théorème 16.4, 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 16.1)

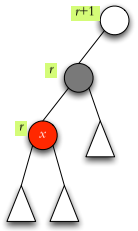
Insertion. On insère x avec $\mathsf{rang}(x) = 1 \Rightarrow$ sa couleur est rouge. **Test** : est-ce que le parent de x est rouge ? Si oui, on a un problème ; sinon, rien à faire (cas 0a). Solution : examiner le grand-parent $y = \mathsf{parent}(\mathsf{parent}(x))$: il est forcément noir par Propriété (ii) de Théorème 16.1. 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), en configurations zig-zag, zig-zig, etc.



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'examination du nœud y physiquement enlevé (donc, c'est le successeur ou prédecesseur 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.

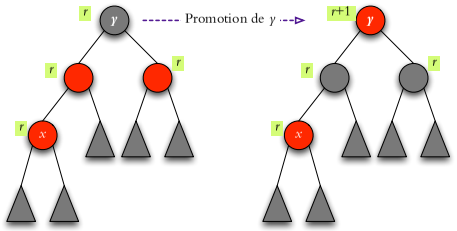
Insertion

Cas 0a : parent noir



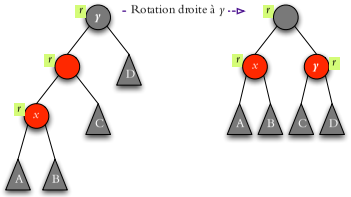
rien à faire

Cas 0b : parent rouge, oncle rouge



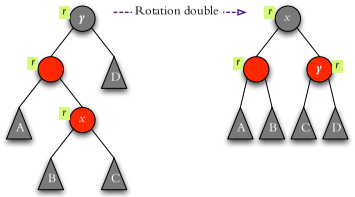
promotion du grandparent, continuer avec $x \leftarrow y$

Cas 1 : parent rouge, oncle noir, zig-zig



une rotation simple

Cas 2 : parent rouge, oncle noir, zig-zag

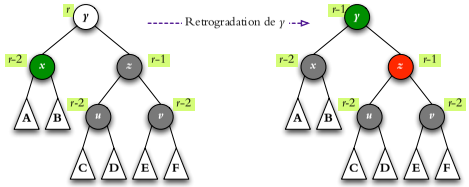


une rotation double

Suppression

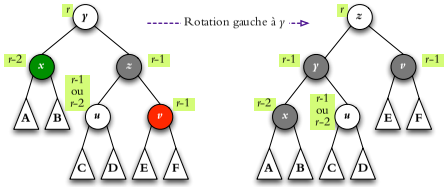
Cas 0 : nœud rouge x — il devient noir, et on arrête

Cas 1a : sœur noire, neveux noirs



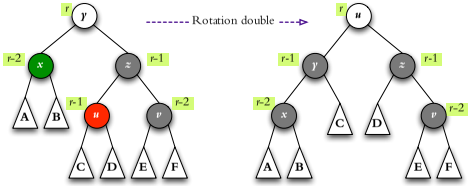
retrogradation du parent, continuer avec $x \leftarrow y$ en cas 0, 1, ou 2

Cas 1b : sœur noire, neveu distant rouge



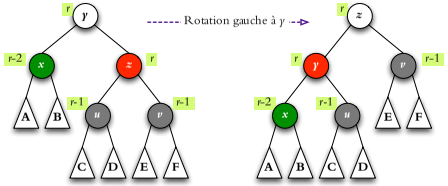
une rotation simple

Cas 1c : sœur noire, neveu proche rouge



une rotation double

Cas 2 : sœur rouge



une rotation simple, continuer avec x en cas 1 (pas de récursion en 1a)