Arbres splay

Benjamin Hellouin

Kozen (algorithms), Chap.12

Un arbre splay est un simple arbre binaire de recherche qui présente la caractéristique d'être *auto-ajustant*, c'est-à-dire qu'il n'est pas nécessairement équilibré mais qu'il se réequilibre de manière transparente quand des opérations sont effectuées. Ceci permet de garantir une complexité amortie de $O(\log(n))$ pour les opérations d'insertion, de suppression, de test d'appartenance, d'union (si $S < S'$) et de séparation (par rapport à un pivot). Le pire cas est cépendant linéaire.

Toutes ces opérations sont effectuables par un nombre constant d'une seule et même opération, le splay : $splay(i, S)$ réorganise l'arbre splay de telle sorte que la racine soit étiquetée par i si $i \in S$, et par un des éléments les plus proches (par excès ou par défaut) sinon. Ainsi,

- le test d'appartenance et la séparation sont évidentes ;
- l'union se fait en appliquant $splay(S, +\infty)$ de telle sorte que l'arbre obtenu n'ait pas de fils droit, puis en plaçant S' à droite ;
- l'insertion de i se fait en appliquant $splay(i, S)$, en élaguant l'arbre à droite ou à gauche selon les cas, en plaçant i en racine de la branche et en joignant les deux arbres ;
- la suppression de i se fait en appliquant $splay(i, S)$ et en unissant les deux fils.

L'opération splay s'implémente à l'aide de l'opération de rotation (dessin).

- On trouve x, ou son plus proche voisin dans l'arbre ;
- Si x est fils droit de fils droit (ou symétriquement), on effectue une rotation sur son père puis sur lui ;
- Si x est fils gauche de fils droit (ou symétriquement), on effectue deux rotations sur lui ;
- Si x a un père et pas de grand-père, on effectue une rotation (cas final).

Analyse. Toutes ces opération prennent un temps $O(d)$, où d est la profondeur du noeud splayé. À une constante près, on suppose que chaque rotation a un coût de 1 et que cela couvre tous les coûts de l'algorithme.

Pour l'analyse, on utilise une méthode de potentiel. On note le *poids d'un sommet* $\mu(x) = \lfloor \log |A(x)| \rfloor$ où $A(x)$ est l'arbre enraciné en x et le *potentiel de l'arbre* est $\nu = \sum_{x \in S} \mu(x)$. Intuitivement, l'arbre sera mieux equilibré si ce potentiel est faible. On prouve que chaque opération splay s'effectue en temps $3(|S| - \mu(x)) + 1 = O(\log n)$ en incluant les coûts d'exécution et la variation du potentiel. Pour chaque rotation, on étudie son coût $C_n = \nu_{n+1} - \nu_n + 1$.

Cas 1. Soit y et z le père et le grand-père de x. Si $\mu_{n+1}(x) \neq \mu_n(x)$:

$$\nu_{n+1} - \nu_n = \mu_{n+1}(x) + \mu_{n+1}(y) + \mu_{n+1}(z) - \mu_n(x) - \mu_n(y) - \mu_n(z)$$

$$= \mu_{n+1}(y) + \mu_{n+1}(z) - \mu_n(x) - \mu_n(y) \quad \text{car } (\mu_{n+1}(x) = \mu_n(z))$$

$\leq 2(\mu_{n+1}(x) - \mu_n(x))$ (chaque terme est plus petit que le terme correspondant)

On obtient une majoration $C_n \leq 3(\mu_{n+1}(x) - \mu_n(x))$. Si au contraire $\mu_{n+1}(x) = \mu_n(x)$, une double majoration montre que le potentiel pour y et z est décroissant. Si on a égalité, écrire $a = |A| + |B| + 1$ et $b = |C| + |D| + 1$ implique que $\lfloor \log a \rfloor = \lfloor \log b \rfloor = \lfloor \log a + b + 1 \rfloor$, ce qui est absurde (si $a \leq b$, $\lfloor \log a + b + 1 \rfloor \geq 1 + \lfloor \log a \rfloor$). Donc le potentiel est strictement décroissant et $C_n \leq 0 \leq 3(\mu_{n+1}(x) - \mu_n(x))$

Cas 2. Ce cas est complètement symétrique au précédent.

Cas final. Soit y le père de x. On a

$$C_n = \mu_{N+1}(x) + \mu_{N+1}(y) - \mu_N(x) - \mu_N(y) + 1 \leq \mu_{N+1}(x) - \mu_N(x)$$

$\leq 3(\mu_{N+1}(x) - \mu_N(x)) + 1$ en tout cas.

Résumé. L'ensemble des opérations a un coût $\sum_{n=0}^{N} 3(\mu_{n+1}(x) - \mu_n(x)) + 1 = 3(|S| - \mu_0(x)) + 1 = O(\log n)$.