# Splay trees

Splay trees are self-adjusting BSTs.

- support ins, del, search in $O(\log n)$ amortized time

Consider perform $m$ accesses in a splay tree.

- Static optimality: suppose every element is accessed at least once, and let $q_x$ be # times $x$ is accessed. Then the cost of performing these $m$ accesses is

$$O\left(m + \underbrace{\sum_x q_x \cdot \log \frac{m}{q_x}}\right)$$

$= m \cdot$ [entropy of a random access]

= cost of searching in the optimal static BST for this sequence (do not change its structure, e.g. by rotation)

- Dynamic finger: cost is also at most

$$O\left(\sum_{i=m} \log(|x_i - x_{i-1}| + 2) + n\right)$$

($x_i$: rank of $i$-th access)

- Working set: cost is also at most

$$O\left(\sum_{i=m} \log(t_i + 2) + n \log n\right)$$

- Dynamic optimality conjecture:

On any access sequence,

cost of splay $\leq O(OPT)$

OPT := least of cost of any BST, possibly tailored for this particular access sequence.

---

Key subroutine: splay(x), which rotates x to the root

ins(x): normal BST insertion, then splay(x)

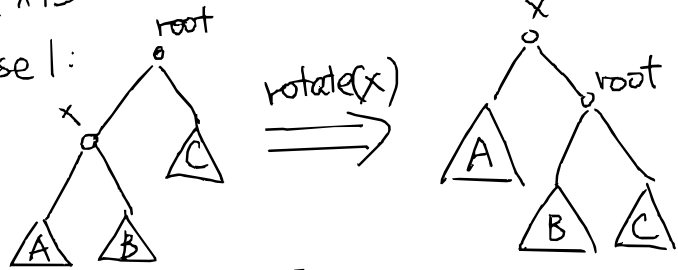
search(x): - - - search, then splay(x)

del(x): - - - - deletion, then splay(parent of deleted node)

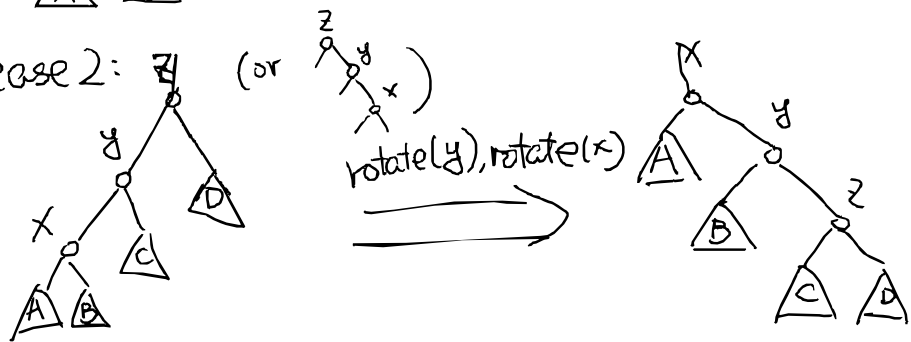
splay(x):

While x is not the root do

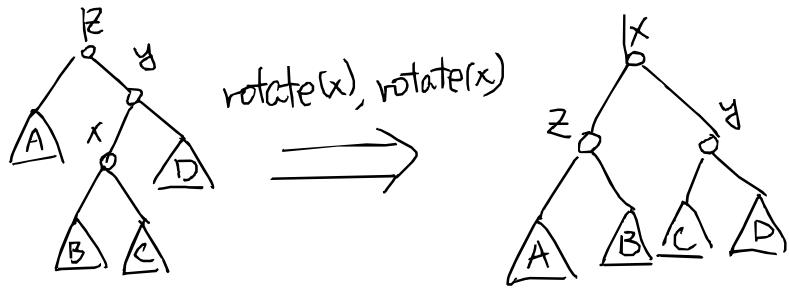
(zig) case 1:




(zig-zig) case 2: (or



(zig-zag) case 3:



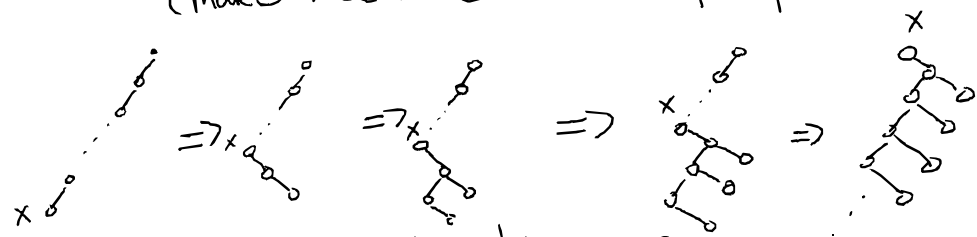

splay(x) takes time O(depth(x)).

The costs of ins, del, search are

O(time of splay(x)).

Analyze amortized cost of splay

Intuition: splay(x) reduces the depth of all nodes from root to x by ≈ a factor of two.
(make tree more balanced if depth(x) is large)




① in case 2&3, both subtrees of x move up by ≥1 level, while x moves up by 2 levels.

② the ancestors of x go under x, then will start moving up with x by ①.

# Potential argument

define a potential function $\Phi$ of the state of the data structure, and bound for each operation

$$(\text{actual time}) + \underbrace{(\Phi_t - \Phi_{t-1})}_{=:\Delta\Phi} \qquad (*)$$

Let $w : \{\text{nodes}\} \to \mathbb{R}^{\geq 0}$ be a weight function.

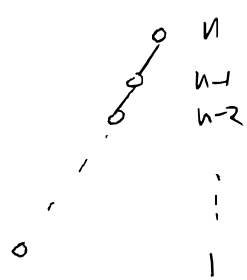
Think of $w(x) = 1$ for all $x$ for now.

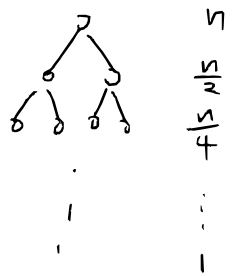
Let $s(x) = \sum_{y \in \text{subtree rooted at } x} w(y)$ (size of the tree)

$r(x) = \log_2 s(x)$. (rank of $x$)

$$\Phi := \sum_x r(x).$$

to get a sense of what $\Phi$ is




$\Phi = \log n + \log n-1 + \cdots + \log 1 = \Theta(n \log n)$

$\Phi = \log n + 2 \cdot \log \frac{n}{2} + 4 \log \frac{n}{4} + \cdots = \Theta(n)$

in general, $\Phi$ small $\Leftrightarrow$ tree is currently balanced.

bounding (actual) + $\Delta\Phi$ is showing

expensive operation must reduce $\Phi$

$\Rightarrow$ balance the tree.

Access lemma: For splay(x),

$$(*) \leq 3\cdot(r(\text{root}) - r(x)) + 1.$$

$$\leq O(\log n) \quad \text{(} \leq \log n \text{)}$$

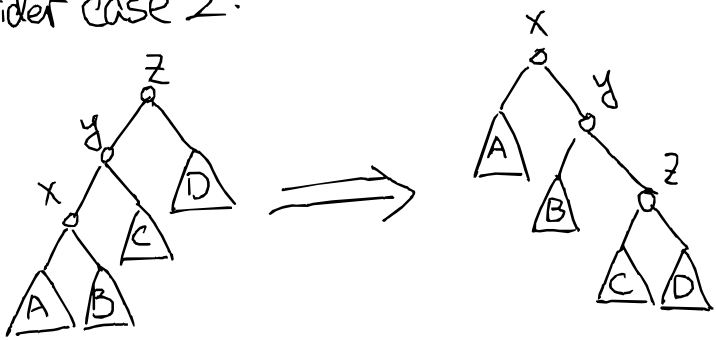
Proof: Prove for each rotation, the amortized cost is

- case 1: $3(r'(x) - r(x)) + 1$
- case 2,3: $3(r'(x) - r(x))$,

where $r'(x)$ is the rank after the rotation.

This implies the lemma by a telescoping sum, and x's final rank is r(root).

Consider case 2:




(actual cost) = 2 rotations

$$\Delta\Phi = \cancel{r'(x)} + r'(y) + r'(z) \\ - r(x) - r(y) - \cancel{r(z)}$$

need: $2 + \Delta\phi \le 3 \cdot (r'(x) - r(x))$.

$$\Delta\Phi \le r'(x) + r'(z) - 2r(x)$$

$$r(x) + r'(z) = \log_2(s(x) \cdot s'(z)) \\ \le \log_2\left(\left(\frac{s(x)+s'(z)}{2}\right)^2\right) \quad \text{AM-GM} \\ \le 2\log_2\left(\frac{s'(x)}{2}\right) \\ = 2r'(x) - 2$$

$$\Delta\Phi \le (r'(x) - 2r(x)) + (2r'(x) - 2 - r(x)) \\ = 3(r'(x) - r(x)) - 2.$$

case 1&3 are similar.

□

Lemma: For ins(x), $\Delta\Phi \le O(r(\text{root})) \le O(\log n)$.

Proof. let $x = y_0, y_1, y_2, \dots, y_k = \text{root}$ be the root-to-x path after insertion, then $\Delta\Phi$

$$= \sum_{i=1}^{k} \log\left(\frac{s(y_i)+1}{s(y_i)}\right) = \log\left(\prod_{i=1}^{k} \frac{s(y_i)+1}{s(y_i)}\right)$$

$$\leq \log\left[\frac{s(y_2)}{s(y_1)} \cdot \frac{s(y_3)}{s(y_2)} \cdots \frac{s(y_k)}{s(y_{k-1})} \cdot \frac{s(y_k)+1}{s(y_k)}\right]$$

$$= \log\left(\frac{s(\text{root})+1}{s(y_1)}\right) \qquad (s(y_i)+1 \leq s(y_{i+1}))$$

$$= O(r(\text{root})). \qquad \square$$

So far only use $w(x)=1$ in the proof.

Setting $w(x) = q_x$ proves static optimality.

Setting $w$ to other values proves different bounds stated earlier.