Analysis of splay trees

Effects of splaying mysterious, analysis subtle.

Goal: m operations on an initially empty tree that never has more than n nodes use $O(m \log n)$ time.

Note: some operations may take more than $\log n$ time, but then earlier ones must have taken less.

Strategy: "Store" excess time for later use. Money analogy. With each operation, adds $O(\log n)$ \$ to tree, distributed among nodes. Each rotation (Cases I, II, or III ~~single or double~~) costs \$1. Show that there's always enough money to pay for rotations.

Defn: For any node N, let $w(N)$ be the number of descendents of N, and $r(N) = \lfloor \log_2 w(N) \rfloor$, the rank of N.

Money Invariant: Each node N has $r(N)$ \$ at all times. 10/30/96

~~Cost of Splay Steps~~

Defn: Let P be a node involved in a rotation. $r'(P)$ denotes its rank after the rotation, and $r(P)$ its rank before.

Cost of Splay Steps Lemma: A rotation ~~involving P, P's parent, and possibly P's grandparent~~ to splay P closer to the root can be done with an additional $3(r'(P) - r(P))$ \$, plus \$1 if this was the last rotation in the splay.

Proof: 3 cases, based on type of rotation. We'll only do ~~the simplest~~ the third.

Case 1: P has no grandparent (Fig 7.21). The extra \$1 pays for the rotation. To maintain the Money Invariant, need ~~new~~ \$ as follows:

$$(r'(P) + r'(Q)) - (r(P) + r(Q)) = r'(Q) - r(P) \le r'(P) - r(P).$$

This is only 1/3 of what the lemma allows.

Case 2: Fig 7.22 ~~See book.~~

~~Case 3: Fig 7.23~~

See book.

Case 3: Fig 7.23. Have to reinstate Money Invariant, and pay $1.

$r'(R) \le r(R)$ and $r'(Q) \le r(Q)$.

~~Move~~ Leave R's money ~~to~~ at R, and leave Q's money at Q.

2/6/98
5/3/96

To satisfy at P, ~~move P's money to R, and~~ add additional

$\cancel{r'(R) - r(P) \le}\ r'(P) - r(P)$

to P.

To pay $1:

If $r(P) < r'(P)$, still have additional dollars.

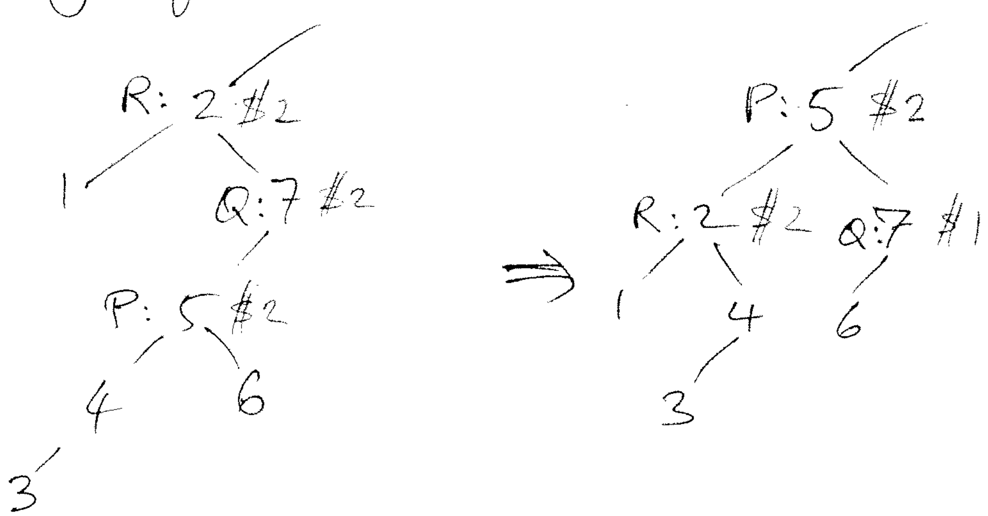
Otherwise, $r'(P) = r(P) = r(R) = r(Q)$.

Either $r'(R) < r'(P)$ or $r'(Q) < r'(P)$, since a node of rank x can't have two children both of rank x.

Thus, either $r'(Q) < r(Q)$ (pay $1 from Q's dollars) or $r'(R) < r(R)$ (~~move all but~~ pay $1 from ~~R to~~ R).

10/26/98

Example of Case 3:



Investment Lemma: Splaying a tree with n ~~at node~~ nodes can be done with an additional $3\lfloor \log_2 n \rfloor + 1$ \$.

Proof: Suppose the splay brings P to the root using k rotations (Case I, II, or III). Let $r^{(i)}(P)$ be the rank of P after i rotations of the splay. By the Cost of Splay Steps Lemma, the number of additional \$ to splay is $3(r^{(1)}(P) - r(P)) + 3(r^{(2)}(P) - r^{(1)}(P)) + \cdots + 3(r^{(k)}(P) - r^{(k-1)}(P)) + 1$

$= 3(r^{(k)}(P) - r(P)) + 1$

$\leq 3\lfloor \log_2 n \rfloor + 1.$

11/1/96

Theorem: Any sequence of m dictionary operations on a ~~self-adjusting~~ splay tree that is initially empty and never has more than n nodes uses $O(m \log n)$ time.

Proof: Show that each operation can be done with an additional $O(\log n)$ \$. (Note that total time of each op is proportional to # of rotations.)

Lookup: Splay costs $O(\log n)$ new \$, by Investment Lemma.

Insert: Cost is splay + investment in new root, which is $\leq \lfloor \log_2 n \rfloor$ \$

Concat: Cost is splay + investment to make T_2 a subtree of root, which is $\leq \lfloor \log_2 n \rfloor$ \$.

Delete: Cost is splay + Concat.

5/6/96
11/6/96
2/9/98
10/28/98

Corollary: