Analysis of splay trees

Effects of splaying mystical, analysis subtle.
Goal: most operations on an initially empty tree that never has more than n nodes use $O(\log n)$ 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, add $O(\log n)$ dollars to tree, distributed among nodes. Each rotation (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 descendants of $N$, and $r(N) = \log_2 w(N)$, the rank of $N$.

**Money Invariant:** Each node $N$ has $r(N)$ dollars at all times.

**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 $P$'s parent, and possibly $P$'s grandparent 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:

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

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

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

**Case 2:** Fig 7.22 & see book.

10/30/96
Case 3: Fig. 7.23. Have to restate Money Invariance, and pay #1.

\[ r'(R) \leq r(R) \text{ and } r'(Q) \leq r(Q). \]

Leave R's money at R, and leave G's money at Q.

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

\[ r'(P) = r'(Q) - r'(P). \]

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

Thus, either \( r'(R) < r'(Q) \) (pay #1 from Q's dollars)

or \( r'(R) < r(R) \) (move #1 from P to R).

Example of Case 3:

\[ \begin{array}{c}
R: \ 2 \ #2 \\
\ \ \ \ 1 \\
\ \ \ Q: \ 7 \ #2 \\
\ \ \ \ P: \ 5 \ #2 \\
\ \ \ \ 4 \\
\ \ \ \ \ 6 \\
\ \ \ \ \ 3 \\
\end{array} \]

\[ \Rightarrow \begin{array}{c}
P: \ 5 \ #2 \\
\ \ \ R: \ 2 \ #2 \\
\ \ \ \ Q: \ 7 \ #1 \\
\ \ \ \ 1 \\
\ \ \ \ 4 \\
\ \ \ \ \ 6 \\
\ \ \ \ \ 3 \\
\end{array} \]
Investment Lemma: Splaying a tree with \( n \) nodes can be done with an additional \( 3 \lfloor \log_2 n \rfloor + 1 \) rotations using \( k \)-rotations (Case I, II, or II)

Proof: Suppose the splay brings \( P \) to the root. 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

\[
3 (r(1)(P) - r(P)) + 3 (r(2)(P) - r(1)(P)) + \ldots + 3 (r(k)(P) - r(k-1)(P)) + 1
\]

\[
\leq 3 \lfloor \log_2 n \rfloor + 1.
\]

11/1/96

Theorem: Any sequence of \( m \) dictionary operations on a self-adjusting 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 \( O(\log n) \) \$.

Concat: Cost is splay + investment to make \( T_2 \) a subtree of root, which is \( O(\log n) \) \$.

Delete: Cost is splay + Concat.

Corollary:

5/6/96

11/6/96

2/9/98

10/28/98