Splay Trees

Design - Goals: A BST s.t.
1) No stored balance info.
2) The tree is "self-balancing".

Properties of a Splay Tree:
1) Search(x) has side effect of rotating x to root.
   (Splay(x))
2) Subtle rotation rules.
3) Uses amortized analysis
Splay Trees

Two ways to describe them:

1) rotation rule
2) As Treap with rules to change priorities (insertion order)

Example:

```
insertion order

(--- b --- a -----)   (--- ab -----)
move a in front of b.
```
3 Rotation Rules for Splay

Goal Def Splay(x) = rotate x to root.

Def Splay(x,T)

1) Zig = \[ \begin{array}{c}
\text{root} \\
\text{x} \\
\text{root}
\end{array} \] (move x to front)

2) Zig-Zag

a) (move x in front of z)

b) (move y in front of z)

c) (move x" "y) (same as 2)

The interesting case

3) Zig-Zig

\[ \begin{array}{c}
\text{z} \\
\text{x} \\
\text{y} \\
\text{z}
\end{array} \]
Priority version of splay:

Given $x, y, z$ s.t. $\text{Parent}(x) = y$ $\text{Parent}(y) = z$
- move $y$ in front of $z$
- move $x$ "" $y$

Why not simple rule: move $x$ in front of $y$.
- i.e. move $x$ to front!
- no zig-zig rule!

Keys: $0, 1, 2, 3, 4$
Priorities: $(4, 3, 3, 3, 0)$ $(0, 4, 3, 3, 2, 1)$

Tree

0

# rotations = 4
move 1 to front

Priorities: \(1, 0, 4, 3, 2\)

num_rotations = \(3\)

MTF 2

num_rotations = \(2\)
Priorities: \((n, n-1, \ldots, 0)\)

\[
\begin{array}{c}
\text{MTF } 0, \ldots, \text{MTF } n \text{ ops} \\
\text{rotations: } n \uparrow n \uparrow n-1 \uparrow \cdots \uparrow 1 \\
\end{array}
\] \(\mathcal{O}(n^2)\) rotations

Average \# rotations \(\Theta(n)\).
Using all 3 rules

Input: (4, 3, 3, 1, 0)

Splay(0)

Note: Trees may be very unbalanced!
\begin{enumerate}
\item \textbf{Splay Dictionary Ops}
\item \textbf{Insert}(x, T) = 1) Vanilla Insert(x, T) \\
\hspace{1cm} 2) Splay(x, T)
\item \textbf{Delete}(x, T) = 1) Splay(x, T) giving X
\hspace{1cm} 2) If B empty return A.
\hspace{1cm} else Splay(largest(A), A) \\
\hspace{1cm} giving Y return Y \\
\hspace{1cm} C
\hspace{1cm} B
\item \textbf{Lookup}(x, T) =
\hspace{1cm} 1) Search(x, T) \\
\hspace{1cm} 2) If found then return Splay(x, T) \\
\hspace{1cm} else return Splay(parent(x), T) \\
\hspace{1cm} (Why?)
\end{enumerate}
Potential Method $\Phi(T)$

Goal: $\Phi$ large for unbalanced trees

**Definition**

$S(x) = \# \text{nodes in subtree rooted at } x$

$\text{Def } r(x) = \lceil \log_a(S(x)) \rceil$ \quad \text{Rank}

$$\Phi(T) = \sum_{x \in T} r(x)$$

If $T$ is line then $\Phi(T) = \sum_{i=1}^{n} \log i = \Theta(n \log n)$

If $T$ is balanced then $\Phi(T) = \Theta(n)$

Claim $\Phi(T) = \Omega(n) \& \Phi(T) = O(n \log n)$
Amortized Cost = Unit-Cost + \Delta E

Unit-Cost = \# rule applications = \frac{\text{depth}}{2} + \left\lceil \frac{\# \text{rotations}}{2} \right\rceil

Access Lemma

\[ AC(\text{splay}(x)) \leq 3(r(\text{root}) - r(x)) + 1 \]

Cor

\[ AC \leq 3\log n + 1 \]

Simple facts

Rank Rule

\[ \begin{array}{c}
\text{Root Rule} \\
\text{Contrapositive}
\end{array} \]

\[ \begin{array}{c}
s(a), s(b) > 2^r \\
\Rightarrow s(x) > 2^m \\
\Rightarrow r(x) > m
\end{array} \]

\[ \begin{array}{c}
x \Rightarrow r' > r
\end{array} \]

\[ \begin{array}{c}
x \Rightarrow r' < r \quad (\text{or}) \quad \Rightarrow 2r > r' + r''
\end{array} \]
Proof of Access Lemma

Claim: Suffice to prove following two lemmas:

Lema 1 \[ AC(s1g) \leq 3(r(root)-r(child))+1 \]

Lema 2 \[ AC(s1g-s1g) \& AC(s1g-s0g) \leq 3(r(GP)-r(c)) \]

Where \( GP \equiv \text{GrandParent}(c) \)
\( P \equiv \text{Parent}(c) \)
\( C \equiv \text{child} \)
Proof of Claim by example: Consider $Splay(x)$

\[ AC(Splay(x)) = AC(3 \delta_x - 3 \alpha_3) + AC(3 \delta_x - 3 \alpha_3) + AC(3 \delta_x - 3 \alpha_3) + AC(3 \delta_x) \]

\[ \leq 3(\delta_1 - \delta_0) + 3(\delta_2 - \delta_1) + 3(\delta_3 - \delta_2) + 3(\delta_4 - \delta_3) + 1 \]

\[ = 3(\delta_4 - \delta_0) + 1 \]
Proof of zig-lemma

\[ AC(\text{zig}) \leq 3(r(\text{root}) - r(c)) + 1 \quad \text{(to show)} \]

\[ = 3(r-r') + 1 \]

before

\[
\begin{array}{c}
\text{root} \\
\alpha \\
\beta
\end{array}
\]

\[
\begin{array}{c}
r' \\
\gamma
\end{array}
\]

after

\[
\begin{array}{c}
\text{old root} \\
r'' \\
r''
\end{array}
\]

\[
\begin{array}{c}
r'' \\
\gamma
\end{array}
\]

\[
\begin{array}{c}
\gamma
\end{array}
\]

\[
\begin{array}{c}
\beta
\end{array}
\]

\[ P(\text{after}) = r + r'' + \Phi(\alpha) + \Phi(\beta) + \Phi(\gamma) = r + r'' + \Phi(\text{subtrees}) \]

\[ P(\text{before}) = r + r' + \Phi(\text{subtrees}) \]

\[ AC(\text{zig}) = 1 + \Delta P = 1 + r'' - r' \leq 1 + (r - r') \leq 3(r-r') + 1 \]
Table of Potential Changes

<table>
<thead>
<tr>
<th>GP</th>
<th>r</th>
<th>Claim</th>
<th>31g-31g or 31g-3ag</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>b</td>
<td>r &gt; a</td>
<td>ΔF ≤ 2(r - a)</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>r = a</td>
<td>ΔF ≤ -1</td>
</tr>
</tbody>
</table>

Case r(GP) > r(C), r > a

\[ \Xi(\text{before}) = r + b + a + \Xi(\text{subtrees}) \]
\[ \geq r + 2a + \Xi(\text{subtrees}) \]
\[ \Xi(\text{after}) \leq 3r + \Xi(\text{subtrees}) \]
\[ \Delta \Xi \leq 3r - (r + 2a) = 2(r - a) \]
Case \( r = b = a \) \( r(GP) = r(P) = r(C) \)

\( E(\text{before}) = 3r + E(\text{subtrees}) \)

Claim \( E(\text{after}) \leq 3r - 1 + E(\text{subtrees}) \)

Case \( \text{zig-zag} \)

\[ \Rightarrow r(P) + r(GP) \leq r \]

Case \( \text{zig-zig} \)

\( r(GP) < r \)

\( r(GP) \) unchanged

Thus \( \Delta E \leq E(\text{after}) - E(\text{before}) \)

\[ \leq -1 \]
\[ AC \text{ (case } r(C_P) > r(C)\text{)} = 1 + \Delta \delta \leq 1 + 2(r-a) \leq 3(r-a) \]

\[ AC \text{ (case } r(C_P) = r(C)\text{)} = 1 + \Delta \delta \leq 1 - 1 = 0 \leq 3(r-r) \]

We Finished Access Lemma!
Balance Thm: m-splays on an n-node tree
splay does $O(m \log n + n \log n)$ rotation

$\text{AC} \leq 3 \log n + 1 \quad \text{end} \geq 0 \quad \text{begin} \leq O(n \log n)$

$\#\text{rotation} \leq O(m (3 \log n + 1) + \text{begin} - \text{end})$

$\leq O(m \log n + n \log n)$
**Important extensions**

**Static Optimality Thm**

$m$ searches, $g_i = \#$ searches of $K_i$ \( \text{eg } m = \sum g_i \)

Then total cost = \(O(m + \sum_{g_i > 0} g_i \log \frac{m}{g_i})\)

**pf**

Redo access lemma with weighted nodes

Set \( S(x) = \sum_{y \in \text{Subtree}(x)} W(y) \)

Set \( \text{rank}(x) = \log (S(x)) \)

Claim: \( AC(splay(x)) \leq 3(\text{rank(root)} - \text{rank}(x)) + 1 \)

For Opt proof set \( W(x) = \frac{g_i}{m} \)
Dynamic Optimality Conjecture

(Sleator & Tarjan 1985)

Def: Dynamic BST Algorithm:
1) BST
2) Updated with rotations between searches.
3) May look into future requests to determine rotations.

Conjecture: Splay tree do at most a constant factor more work than any dynamic BST.