MCS-375: Algorithms: Analysis and Design Handout #DS8
San Skulrattanakulchai
Gustavus Adolphus College
Sep 26, 2014

Splay Trees

A splay tree is an efficient form of binary search tree. It achieves $O(\log n)$ time per binary search tree operation in an *amortized* sense instead of the *worst-case* sense. Unlike red-black trees, splay trees keep no balance information. Instead, it uses the following simple heuristic: **each time an item is accessed, move it to the root.** Hence splay trees are *self-adjusting*. However, we have to implement this heuristic correctly in order to achieve the desired efficiency. The correct way to move the accessed item to the root is called *splaying*.

Splaying

Here is the code for a splay operation at node x.

```
procedure SPLAY(x) {
    while x is not the root do
        if x.p is the root then
            rotate x.p so x becomes the root
        else if x and x.p are both left children, or both right children then
            rearrange the zig-zig path to the opposite zig-zig path
        else
            rearrange the zig-zag path to a sibling path
}
```

Splay tree algorithms

SEARCH(`k`) : do a search like in the binary search tree. Let `x` be the last node accessed. Then either `x.key` = `k`, or `k` is not in the tree but `k`'s parent would be `x` if `k` were in the tree. Splay x.

INSERT(`k`) - use the bst algorithm to insert `k`, say in the new node `x`. Splay x.

DELETE(`z`) - execute the bst algorithm. It sets `y` to `z` or its successor, and replaces `y` by child `x`. Splay `y.p`.

Timing
The time for each splay tree operation is proportional to the time for the splay.

Lemma (Splay Tree Lemma). *Starting with any bst of n nodes, any sequence of $m \geq n$ splays takes time $O(m \log n)$.*

Theorem (Splay Tree Theorem). *Starting from an empty tree, any sequence of m search, insert, and delete operations takes total time $O(m \log n)$. Here n is the greatest number of nodes ever in the tree.*

Although some operations can take $\Theta(n)$ time, on the average (over any sequence of operations) each operation takes $O(\log n)$ time.

The proof of the Splay Tree Theorem is beyond the scope of this course. Several conjectures concerning splay tree properties remain open. (See the wiki page for splay tree.) An example is

Conjecture (Dynamic Optimality Conjecture). *Starting with any bst, on any sequence of node accesses, the time for splaying is at most a constant more than any other bst algorithm.*