Topic 14: Splay Trees

[Kozen, 12]

CPS 230, Fall 2001

1. Splay trees

- We have previously discussed binary search trees and how they can be kept balanced ($O(\log n)$ height) during insert and delete operations (red-black trees).
  - Rebalancing rather complicated
  - Extra space used for the color of each node
- We also discussed skip lists which are a lot simpler than red-black trees
  - Only guarantee $O(\log n)$ expected performance
  - No extra information is used for rebalance information though
- Splay trees are search trees that “magically” balance themselves (no rebalance information is stored) and have amortized $O(\log n)$ performance.
- Recall the basic properties of search trees:
  - Binary tree with elements stored in nodes
  - If node $v$ holds element (key value) $e$ then
    * all elements in left subtree are $< e$
    * all elements in left subtree are $> e$
- Splay tree:
  - Normal (possibly unbalanced) search tree $T$
  - All operations implemented using one basic operation, SPLAY:

  SPLAY($x; T$) searches for $x$ in the tree $T$ and reorganizes the tree so that the new root is either $x$ (if $x$ is in $T$) or else the minimum element $> x$ or the maximum element $< x$ (if $x$ is not in $T$).
- **SEARCH(x, T):** SPLAY(x, T) and then inspect the root.
- **INSERT(x, T):** SPLAY(x, T) and create a new root with x.

```
  T
  ↓
splay(x,T)

  r
  ↓
T1  T2

  x
  ↓
T1  T2
```

- **DELETE(x, T):**
  * SPLAY(x, T) and remove the root, thus yielding two subtrees T1 and T2.
  * SPLAY(x, T1).
  * Make T2 right son of new root of T1 after the SPLAY.

```
  T
  ↓
splay(x,T)

  x
  ↓
T1  T2
```

```
  r
  ↓
splay(x,T1)

  T1'
  ↓
T2

  r
  ↓
T1'  T2
```

⇒ All operations perform O(1) SPLAYs and use O(1) extra time.
⇒ If SPLAY runs in O(log n) amortized time, then so do all operations.

- **Implementation of SPLAY:**

  - Search for x like in normal search tree
  - Repeatedly rotate x up until it becomes the root.

We distinguish between three cases:

1. **x is child of root (no grandparent):** Do **rotate(x)**

```
        y
            ↘
              x
          T1  T2
          ↓  ↓
        T3  T1
```

2. **x has parent y and grandparent z and x and y are both left children or both right**
children: Do rotate(y) followed by rotate(x)

e.g.

3. x has parent y and grandparent z and one of x and y is a left child and the other is a right child: Do rotate(x) followed by rotate(x)

e.g.

- Note:
  - A SPLAY can take O(n) worst-case time (very unbalanced tree)
  - But splay trees somehow seem to stay nicely balanced \(\Rightarrow O(\log n)\) amortized SPLAY.

Examples: SPLAY(1, T)

SPLAY(5, T)
2 Analysis of Splay Trees

- We will use the *accounting method* to show that all operations take $O(\log n)$ amortized time.
  - We will imagine that each node in the tree has credits on it.
  - We will use some credits to pay for (part of) rotations during a SPLAY.
  - We will see that in addition to the SPLAY cost, each INSERT and DELETE requires placing $O(\log n)$ new credits on the new root node.

- We will ignore cost of searching for $x$, since the rotations cost at least as much as the search. (That is, if we can bound amortized rotation cost, we can also bound search cost.)

- Let $T(x)$ be the subtree of $T$ that is rooted at $x$. We will maintain the *credit invariant* that the number of credits on each node is

$$\mu(x) = \lfloor \log |T(x)| \rfloor.$$  

Equivalently, we could use the potential function

$$\Phi(T) = \sum_x \mu(x).$$

- We will prove the following lemma:

  \[ \text{At most } 3(\mu(T) - \mu(x) + O(1)) \text{ new credits are needed to perform the SPLAY}(x,T) \text{ operation and maintain the credit invariant.} \]

- This lemma implies that a SPLAY operation uses at most $3\lfloor \log n \rfloor + O(1) = O(\log n)$ new credits (amortized time).

- In addition, each INSERT or a DELETE requires us to place onto the new root at most $O(\log n)$ new credits, so the total number of new credits placed counting those done by the SPLAY is $O(\log n)$, thus giving the $O(\log n)$ amortized time bound.

- Proof of lemma:
  - Let $\mu$ and $\mu'$ denote the value of $\mu$ before and after a rotate operation.
  - During a SPLAY operation we perform some number (say, $k$) of case 2 and 3 operations and possibly one case 1 operation.
  - We will show that the actual cost of an operation is as follows:
    * Case 1: $3(\mu'(x) - \mu(x)) + O(1)$
    * Case 2: $3(\mu'(x) - \mu(x))$
    * Case 3: $3(\mu'(x) - \mu(x))$
  
  \[ \Rightarrow \text{When we sum the actual costs over all } \leq k + 1 \text{ operations in the SPLAY, we get} \]

$$3(\mu(T) - \mu(x)) + O(1),$$ (1)

where $\mu(x)$ is the number of credits on $x$ before the SPLAY.
  - Note that it is important that the additive $O(1)$ term appears only in case 1. If the $O(1)$ additive term also appeared in cases 2 and 3, we would get an additive $O(k)$ term in (1), so this approach wouldn’t work.
• Case 1:
  - We have $\mu'(x) = \mu(y)$, $\mu'(y) \leq \mu'(x)$, and all other $\mu$’s are unchanged.
  - To maintain invariant, we use the following number of credits:
    \[
    \mu'(x) + \mu'(y) - \mu(x) - \mu(y) = \mu'(y) - \mu(x) \\
    \leq \mu'(x) - \mu(x) \\
    \leq 3(\mu'(x) - \mu(x))
    \]
  - To do the actual rotation, we use $O(1)$ credits.

• Case 2:
  - We have $\mu'(x) = \mu(z)$, $\mu'(y) \leq \mu'(x)$, $\mu'(z) \leq \mu'(x)$, $\mu(y) \geq \mu(x)$, and all other $\mu$’s are unchanged.
  - To maintain the invariant, we use the following number of credits:
    \[
    \mu'(x) + \mu'(y) + \mu'(z) - \mu(x) - \mu(y) - \mu(z) = \mu'(y) + \mu'(z) - \mu(x) - \mu(y) \\
    = (\mu'(y) - \mu(x)) + (\mu'(z) - \mu(y)) \\
    \leq (\mu'(x) - \mu(x)) + (\mu'(x) - \mu(x)) \\
    = 2(\mu'(x) - \mu(x))
    \]
  - This means that we can use the remaining $\mu'(x) - \mu(x)$ credits to pay for rotation, unless $\mu'(x) = \mu(x)$ (which can happen because of the floor function, since $\mu(x) = \lfloor \log |T(x)| \rfloor$).
  - We will show by contradiction that if $\mu'(x) = \mu(x)$ then $\mu'(x) + \mu'(y) + \mu'(z) < \mu(x) + \mu(y) + \mu(z)$, which means that the operation actually releases credits, which we can use for the rotation:
    * Assume that $\mu'(x) = \mu(x)$. To set up the proof by contradiction, let’s assume that $\mu'(x) + \mu'(y) + \mu'(z) \geq \mu(x) + \mu(y) + \mu(z)$
    * We have $\mu(x) = \mu'(x) = \mu(x)$ and $\mu(x) \leq \mu(y) \leq \mu(z)$
    \[
    \implies \mu(z) = \mu(x) = \mu(y) \\
    \implies \mu'(x) + \mu'(y) + \mu'(z) \geq \mu(x) + \mu(y) + \mu(z) \\
    = 3\mu(x) \\
    = 3\mu'(x)
    \]
    \[
    \implies \mu'(y) + \mu'(z) \geq 2\mu'(x).
    \]
    * Since $\mu'(y) \leq \mu'(x)$ and $\mu'(z) \leq \mu'(x)$, we get $\mu'(x) = \mu'(y) = \mu'(z)$.
    * Therefore, we have
    \[
    \mu(x) = \mu(y) = \mu(z) = \mu'(x) = \mu'(y) = \mu'(z)
    \]
    \[
    \text{(2)}
    \]
  * We will now show that (2) cannot possibly be true (which will complete the proof by contradiction):
  Let $a$ be $|T(x)|$ before the rotations (i.e., $a = |T1| + |T2| + 1$).
  Let $b$ be $|T(z)|$ after rotations (i.e., $b = |T3| + |T4| + 1$).
  Since $\mu(x) = \mu'(z) = \mu'(x)$, we have $\lfloor \log a \rfloor = \lfloor \log b \rfloor = \lfloor \log (a + b + 1) \rfloor$, but then we have the following contradiction:
    - if $a \leq b$, then $\lfloor \log (a + b + 1) \rfloor \geq \lfloor \log 2a \rfloor = 1 + \lfloor \log a \rfloor > \lfloor \log a \rfloor$
    - if $a > b$, then $\lfloor \log (a + b + 1) \rfloor \geq \lfloor \log 2b \rfloor = 1 + \lfloor \log b \rfloor > \lfloor \log b \rfloor$

• Case 3:
  - Can be proved analogously to case 2.