15-451 Assignment 03

Karan Sikka
ksikka@cmu.edu
Collaborated with Dave Cummings and Sandeep Rao.
Recitation: A
September 18, 2014

1: Balanced Splay Trees

The outline of the proof is as follows:

1. We show a lower bound on the decrease in potential when performing a deep splay, $3lg(n)+1$.
2. We show an upper bound on the increase in potential when performing a splay, $3lg(n)+1$.
3. From these facts, we conclude that since we start with a perfectly balanced tree (min potential), a deep splay can only occur at most once after a splay, due to restrictions on the potential function.

(1) Lower bound on decrease in potential during deep splay.

From the access lemma, we know that:

$$\text{a.c. of splay} \leq 3(r(y) - r(x)) + 1$$

Where y is the root node and x is the node being splayed.
We know that $r(y) = lg(n)$ and in the case of a deep splay, $r(x) = 0$ since it's terminal, subtree size is 1, $lg(1) = 0$.

We also know the definition of amortized cost in terms of the potential function. Then in the case of a deep splay,

$$\#\text{ of rots} + \Delta\Phi \leq 3lg(n) + 1$$

In the case of a deep splay, the number of rotations is $d - 1$ since a rotation moves the terminal node 1 level closer to the root.
And a deep splay only occurs when the tree is off-balance which means that $d \geq 6lg(n) + 3$. Combining these facts, we do the following algebra:

$d - 1 + \Delta\Phi \leq 3lg(n) + 1$
$\implies \Delta\Phi \leq 3lg(n) + 1 - (d - 1)$
$\implies \Delta\Phi \leq 3lg(n) + 1 - (6lg(n) + 3 - 1)$
$\implies \Delta\Phi \leq -(3lg(n) - 1)$

(2) Upper bound on increase in potential of a splay.

Combine the access lemma with the defn of amortized cost to establish the following truth:

$$\Delta\Phi \leq 3(r(y) - r(x)) + 1 - \#\text{ of rots}$$

We know that $r(y) = lg(n)$. We know $r(x) \geq 0$ and same with the number of rotations, so we can omit them from our upper bound. Thus, we obtain the following.

$$\Delta\Phi \leq 3lg(n) + 1$$

(3) We start with a perfectly balanced tree, so that the potential function is at its minimum. From there, a deep splay can only occur after a splay, and it can only occur once per splay, otherwise the potential function would go lower than it was when the tree was perfectly balanced.

2: You Be the Adversary

Say there are n elements.

Next we notice the following about the counts of different types of elements:

1. The number of frees at the beginning is n, and the number of frees at the end is 0.
2. The number of tops at the beginning is 0, and the number of tops at the end is 1.
3. The number of bottoms at the beginning is 0, and the number of bottoms at the end is 1.
4. The number of middles at the beginning is 0, and the number of middles at the end is $n - 2$.

The number of tops is not more than 1 because then there is ambiguity as to which is the larger top.
You would have to compare them to determine that and the lower would become a middle. A symmetrical argument holds for the number of bottoms. The number of middles are the remaining elements. No element can be free since that element could be changed to be a Max or Min and it wouldn't be detected by the algorithm.

Define a function Φ to be 3/2 times the number of free elements plus the number of tops, plus the number of bottoms
At the beginning of the algorithm, Φ is $\frac{3}{2}n$ since all elements are free and there are no tops nor bottoms
At the end of the algorithm, Phi is 2 since there is 1 top and 1 bottom.

We observe the following about the life-cycle of the elements:

1. At the start of the algorithm, all n elements are free.
2. A free element compared with any other element will turn into either a top or bottom.
3. A top or bottom compared with any other element will either remain as it was, or turn into a middle.
4. A middle compared with any other element will remain a middle.

Now we compute the change in Φ of each case

1. free =>top or bottom: -3/2 + 1 = -1/2
2. top or bottom =>top or bottom, or middle: 0 or -1
3. middle =>middle: 0.

Notice that two elements may convert on a comparison, so the only way to exceed a change of -1 is by having
two conversions of top or bottom to middles. This could only happen by comparing a top and a bottom, finding
that the bottom is greater than the top, and they both convert to middles.

An adversary can choose an input such that this case does not happen. On some input,
run the algorithm and observe which elements became top or bottom.
Note that an element starts free, then must become only top, or only bottom, and then it may become middle.
Then, add a scalar to all the top elements such that the smallest top element is larger than the largest bottom element.
Now we know that all tops are larger than all bottoms, and the ordering of tops amongst themselves in unchanged.
We can rerun the algorithm with the new input,
and we'll see that all tops before are still tops, and any top compared with any bottom, does not convert both
to middle, because the top will be greater than the bottom.

So we can find an input such that the change in Φ is at most -1, and that means Φ will start at $\frac{3}{2}n$ and end at 2, taking $\frac{3}{2}n - 2$ comparisons to get there.