# 6.854 Advanced Algorithms

Problem Set 2

**John Wang**
Collaborators:

**Problem 1-a:** In class, I stated that single rotations don't work for splay trees. To demonstrate this, consider a degenerate $n$-node linked list shaped binary tree where each node's right child is empty. Suppose the only leaf is splayed to the root by single rotations: show the structure of the tree after this splay. Generalizing, argue that there is a sequence of $n/2$ splays that each take at least $n/2$ work.

**Solution:** If we start with the given degenerate $n$-node tree and splay the bottom-left node (the leaf) $x$, then we will slowly move the node up to the root through a series of "kink" rotations. These kink rotations will move $x$ (which is the global minimum of the tree) and decrease its depth by one each time. Visually, all the nodes above and below $x$ will be left children of their parents, and $x$ will be the only node which is a right child of its parent. The process of moving $x$ up the tree will continue until $x$ becomes the root. At this point, $x$ will only have a right child, which will be the maximum element of the data structure, and everything else will continue down in a series of left children.

Let us call this resulting data structure $T'$. When we splay the new leaf of the tree $x'$ (node which can be found by walking down the left half of the tree), we can see that it will take the same path up to the root through a series of "kink" rotations. Once $x'$ becomes a right child of the root $x$, a rotation will occur which puts $x'$ at the root of the tree, and $x$ as the left child (since $x$ is the absolute minimum and is the only node smaller than $x'$). The right child of $x'$ will be the global maximum. The tree now has depth $n-1$.

Continuing in the same pattern and taking the deepest leaf $x''$ of the tree, we will go through a series of $n-2$ rotations until $x''$ reaches the root. At this point $x'$ will be the left child of $x''$, with $x$ as the left child of $x'$. The maximum node will be the right child of the new root, so the tree will now have depth $n-2$.

It is clear that the deepest leaf $a$ will always be promoted up to the leaf in time equal to the depth of the tree. Once it has reached the root, the previous root $b$ will become its left child, and the maximum node which was previously $b$'s right child, will become $a$'s right child. Therefore, the tree will decrease in depth by 1.

A series of $n/2$ operations that splay the deepest leaf will therefore cost $O(d)$ each, where $d$ ranges from $n, \ldots, n/2$. Therefore, we have shown that there is a sequence of $n/2$ splays that each take at least $n/2$ work. □

**Problem 1-b:** Now from the same starting tree, show the final structure after splaying the leaf with (zig-zig) double rotations. Explain how this splay has made much more progresss than single rotations in improving the tree.

**Solution:** Starting the the only leaf of the tree $x$, we will perform zig-zigs on the way up to the root. For notation, we will say $x$ is the leaf (minimum element), $x'$ is the parent of the leaf (second smallest element), and $x''$ is the parent of $x'$, etc. After the first zig-zig, we see that everything except the last three nodes have been untouched, but $x$ is now the child of $x'''$. Moreover, $x$ has a right child of $x'$ and $x'$ has a right child of $x''$.

Performing another zig-zig operation, the $x'$ to $x''$ chain will become the left subtree to $x'''$. Moreover, $x'''$ will have a right child of $x''''$. □

# 6.854 Advanced Algorithms

Problem Set 2

**John Wang**
Collaborators: