### Splay tree operations

- **Find**
  Finding an element in the splay tree follows the same behavior as in a BST. After we find our node, we splay it. *(Note: if the node is not found, we splay the last node reached.)*

- **Find-Min**
  This operation will only go down the left children, until none are left. After we find the min node, we splay it.

- **Find-Max**
  The process for this is the same as for $\text{Find} - \text{Min}$, except we go down the right child.

- **Join**
  Given two trees $T_1$ and $T_2$ with $\text{key}(x) < \text{key}(y)$ $\forall x \in T_1, y \in T_2$, we can join $T_1$ and $T_2$ into one tree with the following steps:
  1. $\text{Find} - \text{Max}(T_1)$. This makes the max element of $T_1$ the new root of $T_1$.
  2. Make $T_2$ the right child of this root.

- **Split**
  Given a tree $T$ and a pivot $i$, the split operation partitions $T$ into two BSTs:
  \[
  T_1: \{x | \text{key}(x) \leq i\} \\
  T_2: \{x | \text{key}(x) > i\}
  \]
  We split the tree $T$ by performing $\text{Find}(i)$. This Find will then splay on a node, call it $x$, which brings it to the root of the tree. We can then cut the tree; everything on the right of $x$ belongs to $T_2$, and everything on the left belongs to $T_1$. Depending on its key, we add $x$ to either $T_1$ or $T_2$. Thus, we either make the right child or the left child of $x$ a new root by simply removing its pointer to its parent.
  Join and Split make insertion and deletion very simple!

- **Insert**
  Let $i$ be the value we want to insert. We can first split the tree around $i$. Then, we let node $i$ be the new root, and make the two subtrees the left and right subtrees of $i$ respectively.

- **Delete**
  To delete a node $i$ from a tree $T$, we first $\text{Find}(i)$ in the tree, which brings node $i$ to the root. We then delete node $i$, and are left with its left and right subtrees. Because everything in the left subtree has key less than everything in the right subtree, we can then join them.