Balanced Binary Search Trees: Splay Tree & AVL Tree

Insert and Delete Complexity Summary
*The Splay tree insert is O(log n) and for delete is amortized O(log n).
*The AVL tree insert is O(log n) and for delete is O(log n).

Splay Tree
The splay tree maybe best implemented when a value could be accessed numerous times. Every time a node is accessed in a splay tree it is moved to the root of the tree. Reducing the time needed to access that node again. This could lead to faster return times but does not guarantee being faster than the AVL tree. The splay tree is also more memory efficient that AVL trees because it does not need to store balance information in the nodes.

Disadvantages:
*The $O(\log\ n)$ time is only amortized; individual operations may take $\Omega(n)$ time and many rotations
*Many rotations

Advantages:
*The $O(\log\ n)$ amortized time for all dictionary operations
*Easy implementation

AVL Tree
The AVL tree guarantees O(log n) for delete and insert when the splay tree maybe a little slower on other operations including delete. It is potentially better to use a splay tree on a smaller data set if the user is using a particular amount of data numerous times. The AVL being a better option if using for a larger data set without normal use of particular values or nodes.

Disadvantages:
*AVL trees are difficult to implement
* Re-balancing expensive operation

Advantages:
*Time complexity
*Search times for keys