Splay Tree

Splay tree is a self balancing binary tree that performs $splaying$ after certain ADT operations like $insert/find$. A popular choice is to perform $splaying$ after each $find$ operation. The idea is based on the heuristic that an element accessed now will be accessed again in future. The accessed element X is moved to the top of the tree through multiple AVL rotations such that, tree as a whole is more balanced.

There are six rotational operations:

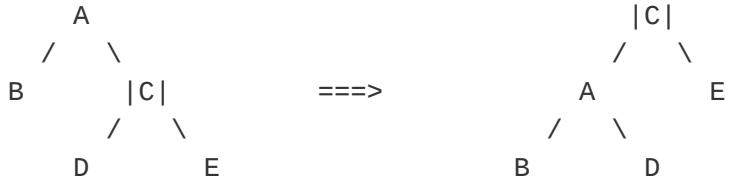
- $Zig\ right$: A single right rotation
- $Zig\ left$: A single left rotation
- $Zig - Zig\ right$: A double right rotation
- $Zig - Zig\ left$: A double left rotation
- $Zig - Zag\ right$: First left and then right rotation
- $Zig - Zag\ left$: First right and then left rotation

The illustration below shows how rotations are chosen. There are two factors in play:

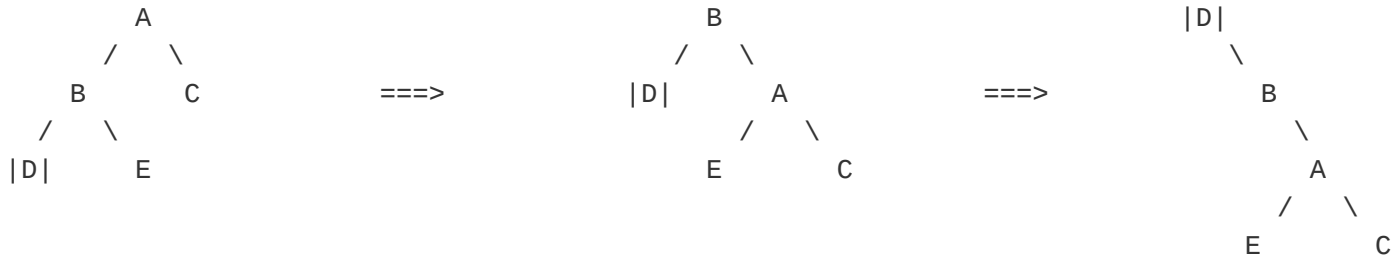
- Whether the accessed node has grandparents
- Whether the accessed node is left child or right child

Access B: $Zig\ right$ rotation

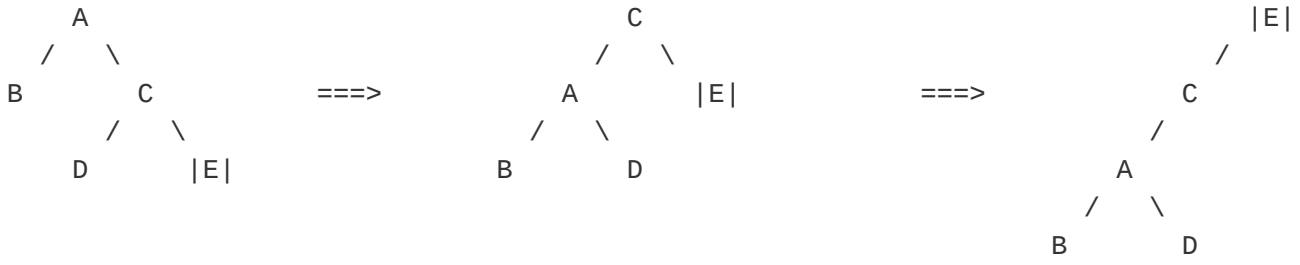
Access C: $Zig\ left$ rotation



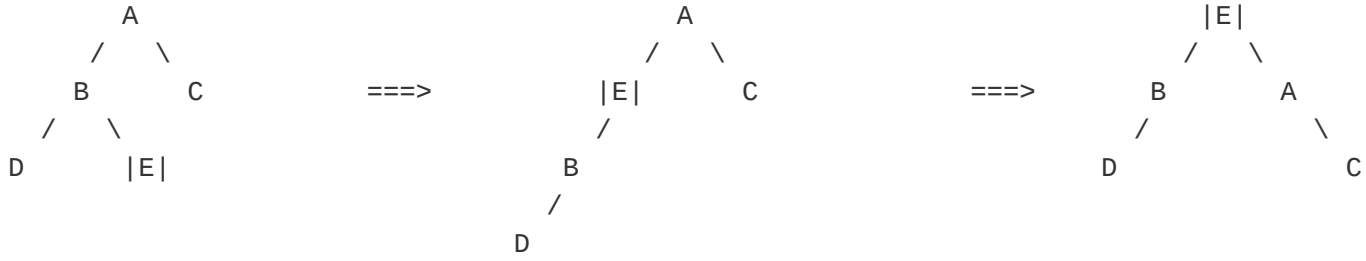
Access D: $Zig - Zig\ right$ rotation



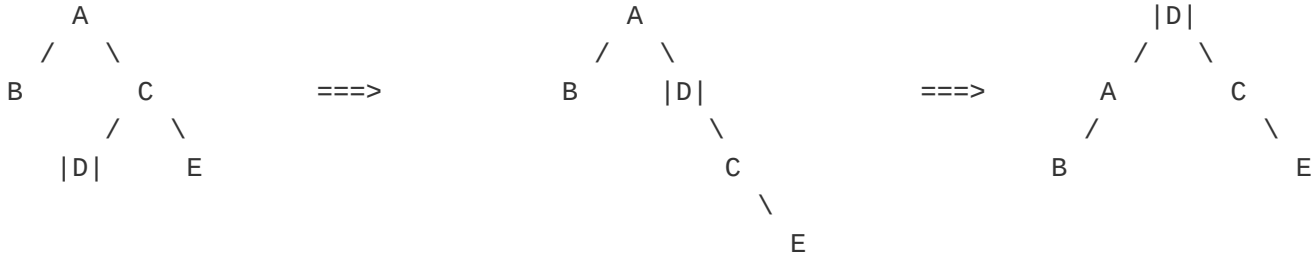
Access E: $Zig - Zig\ left$ **rotation**



Access E: $Zig - Zag\ right$ **rotation**



Access D: $Zig - Zag\ left$ **rotation**



Any sequence of M operations on Splay tree takes $O(MlogN)$ time. So, the amortized running time of one operation is $O(logN)$. Splay tree guarantees that even though an operation can take $O(N)$ time, it is impossible to get long $O(N)$ operation sequences. In absence of splaying, M operations can take $O(MN)$ time.