AVL Tree:

* Self-balancing binary search tree
* Height of a nodes child subtrees differ by at most one
* The balance factor of a node N is defined to be the height difference of its two child subtrees
  + BalanceFactor(N) := Height(RightSubtree(N)) – Height(LeftSubtree(N))
* A binary tree is defined to be an AVL tree if the invariant:
  + BalanceFacotr(N) in {-1,0,+1} for all nodes
* A node with balance factor <0 is “left-heavy”, >0 “right-heavy”, =0 “balanced”
* Search:
  + Same as binary search
* Insert:
  + Begin with a BST insert
  + ‘retrace’ the tree checking/ updating balance factors and balancing tree
    - the balance factor after an insertion will be in the range [-2,2]
    - if the balance factor remains in the range [-1,1] stop!
* Deletion:
  + BST deletion with retracing as defined in insert
* Rebalancing:
  + If a nodes balance factor is:
    - Greater than 1:
      * Left Left case
      * Left Right case
    - Less than 1:
      * Right Right case
      * Right Left case

y x

/ \ Zig (Right Rotation) / \

x T3 – - – - – - – - - -> T1 y

/ \ < - - - - - - - - - / \

T1 T2 Zag (Left Rotation) T2 T3