CS2040S Midterm Cheatsheet	CS2040S
Recurrence Solution	Properties > $T(n)=kT(n/k)+O(f(n)) \implies O(f(n)\log\square)$ > $T(n)=T(n/k)+O(n\square) \implies O(n\square)$ > $T(n)=T(n-1)+O(n\square) \implies O(n\square^{+1})$
AVL Tree	Process ::Create AVL Tree > Augment tree (add info) > Define balance condition / invariant > Maintain balance by walking up the tree to reach the lowest unbalanced node > Maximum 2 rotations to fix an insertion ::Augment tree > Store height at every node (v.height = h(v)) > Heights are updated when inserting ::Balance an L-heavy node v > v.L balanced → R-rotate(v) > v.L L-heavy → R-rotate(v) > v.L R-heavy → L-rotate(v.L); R-rotate(v) ::Deletion > Walk up tree and rotate if unbalanced ::Rotation ?> Tree_Rotation Properties ?> AVL_Invariant > Year :- 1962 Adelson-Velsii Landis
Tree Rotation	Process ::Right-rotate node v > Let w = v.L > v.L = w.R > w.R = v Types ?> Double_Rotation
Double Rotation	Process ::Double rotate node v to the right > Let w = v.L > Left-rotate w > Right-rotate v Properties > w.L becomes new root > w.R = w.L.L > v.L = w.L.R
AVL Invariant	Properties > Child height :- |L_Height - R_Height| < 1 > Maximum nodes :- 2^(h+1)-1 > Minimum nodes :- $1+f(n-1)+f(n-2)$
Interval Tree	Definition > Keys are stored as [min, max] > Nodes are comparable by the mins of keys > For each node, store highest max in subtree (h) Usage > Find an interval that covers a certain value Process ::Insert interval node > Find your way to the insertion point and update the h along the way > Balance and update maximum values of rotated nodes ::Search for interval that covers a value > left.h >= key ⟹ go left > left.h < key ⟹ go right Properties > Tree goes right ⟹ no overlapping interval in left subtree > Tree goes left and fails ⟹ there is no point searching in the right side
Orthogonal Range Tree	Definition > Store all points in leaves of tree > Internal nodes should not have the keys > Internal node stores the max of any leaf in its LEFT subtree Usage > Find all values within a range Examples > Find number of points found within box Properties > Invariant :- The search interval for a left-traversal at node v includes the maximum item in the subtree rooted at v > /Search_RTC :- O(k+log□) > /Build_RTC :- O(nlogn) > /Space_Complexity :- O(n) Process ::Build tree > Choose the middle value and make it the root > Recurse with the remaining values until you end up with a tree ::Query(10, 50) > Find /Split_Node where node is between 10 and 50 > Do left traversal, and hug the right > Do right traversal, and hug the left Topics ?> Split_Node
Chaining	Definition > Each bucket contains a /Linked_List of items > So everytime a collision happens, then just attach it in front of the existing linked list Properties > /Insert_RTC :- O(1+h()) > /Search_RTC :- O(n+h()) > /Space_Complexity :- O(buckets+items)