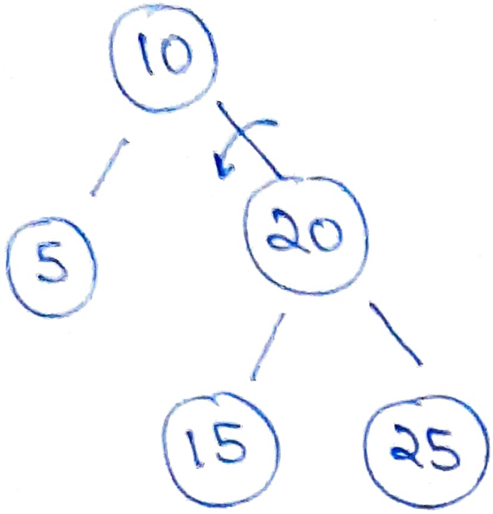
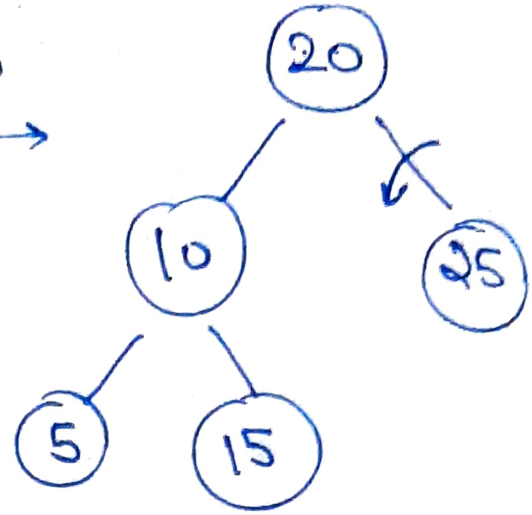
(a.)



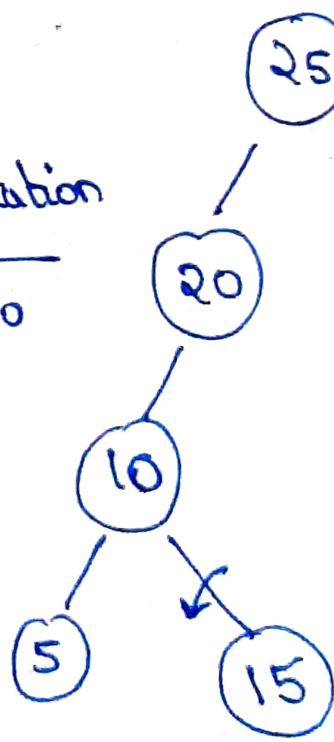
left rotation

about 10



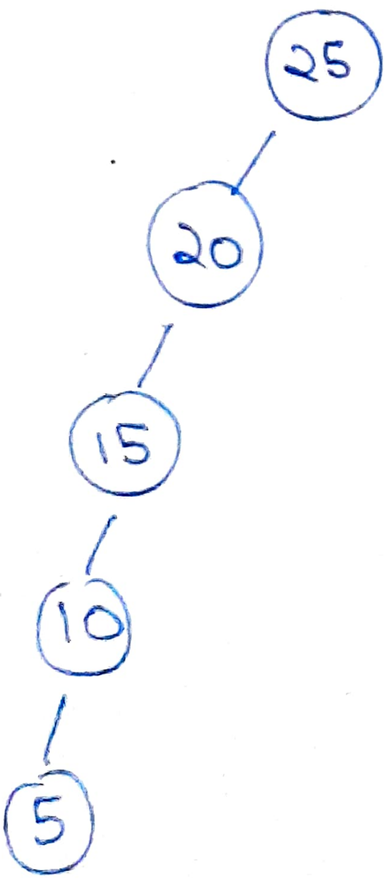
left rotation

about 20



left rotation

about 10



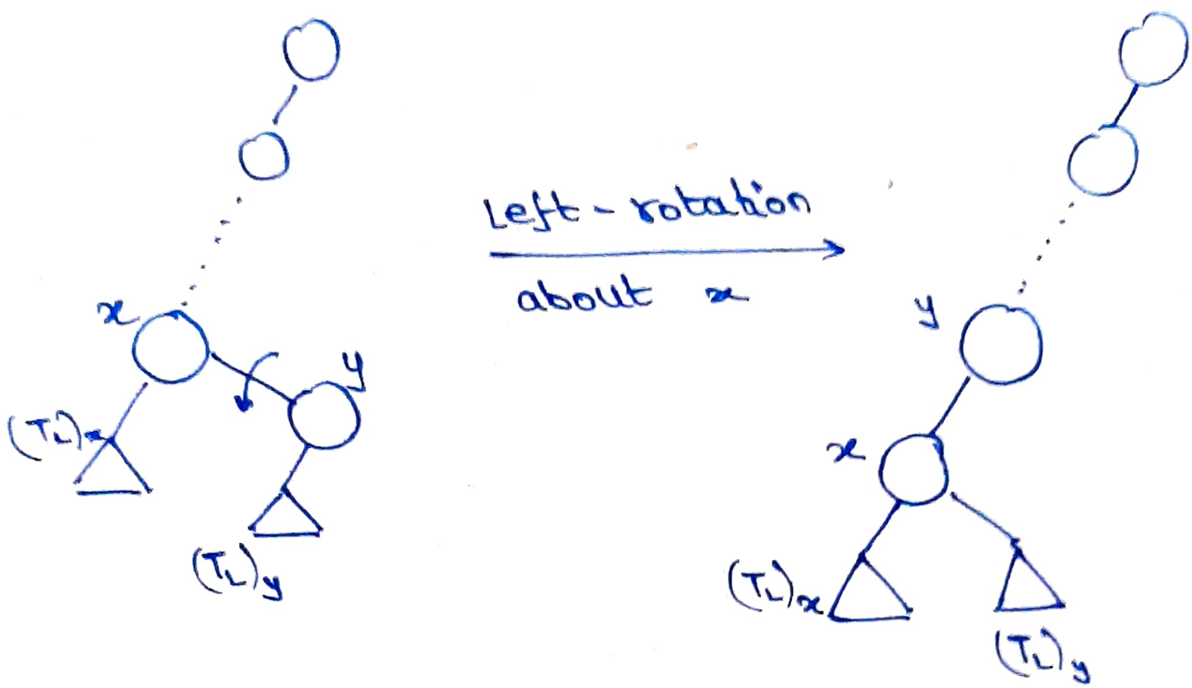
(b) Procedure: First we find the topmost node that has a non-null right child. Then we apply left rotation about it, till there are no such nodes in the entire tree.

- Correctness of the Algorithm:

→ Let 'd' denote the depth of the topmost node with a non-null right child.

→ All the BSTs can be effectively grouped into 2 cases. Let 'x' be the topmost node at any step of algorithm that has a non-null right child 'y'.

Case 1: y has a null right child



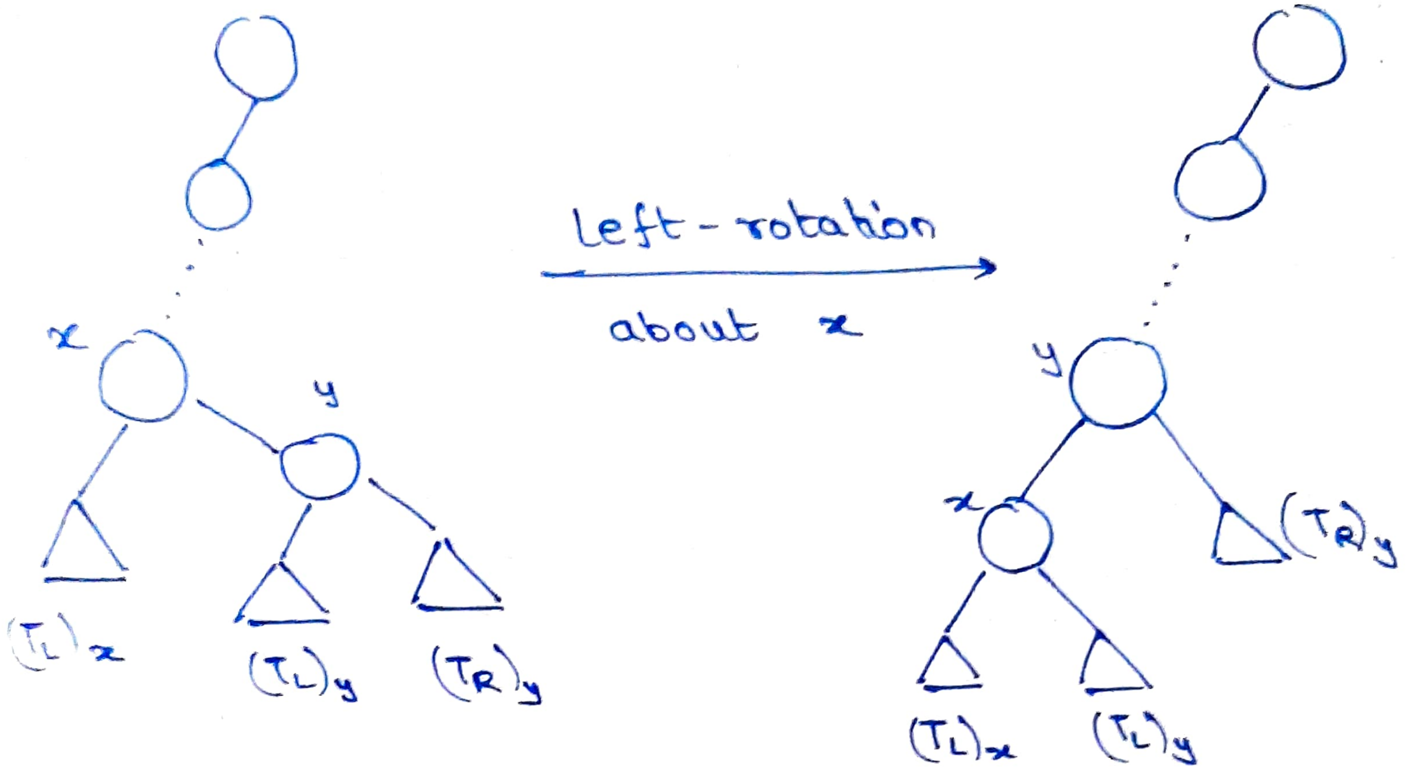
In this case, we perform a left-rotation about x. $(T_L)_x$ = left subtree of x

$(T_L)_y$ = Left subtree of y

Depth of x increases by 1.

Algorithm can be recursively applied again.

Case 2 : Right child of y is non-null.



In this case, we perform left-rotation about x.

Now, the topmost node with a non-null right child is 'y'.

Depth of $(T_R)_y$ from x (the topmost node with non-null right child)

= 2

Depth of $(T_R)_y$ from 'y' after left-rotation is performed $= 1$

Hence, the depth of the right subtree from the topmost nodes with non-null right child decreases.

After almost Height (subtree $((T_R)_y)$) operations, we effectively reduce the case to case 1.

Since the number of nodes are finite the algorithm exhaustively reduces all right subtrees in finite time

→ Algorithm reduces the BST to a left-linear BST in finite number of steps.

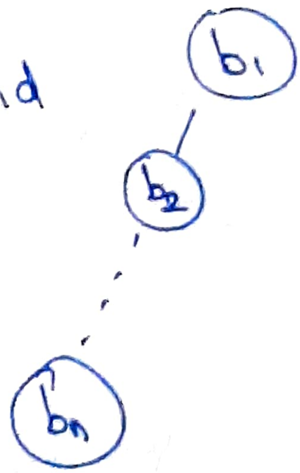
(c) Let S_1 = set of all BSTs

S_2 = set of all BSTs which are left-linear.

(1) Note that for a given set of nodes, the BSTs made from the given set have the same unique left-linear BST.

Proof: Suppose we have 2 BSTs B_1 and B_2 that have the same set of nodes, and have 2 different left-linear BST T_1 and T_2, say

and

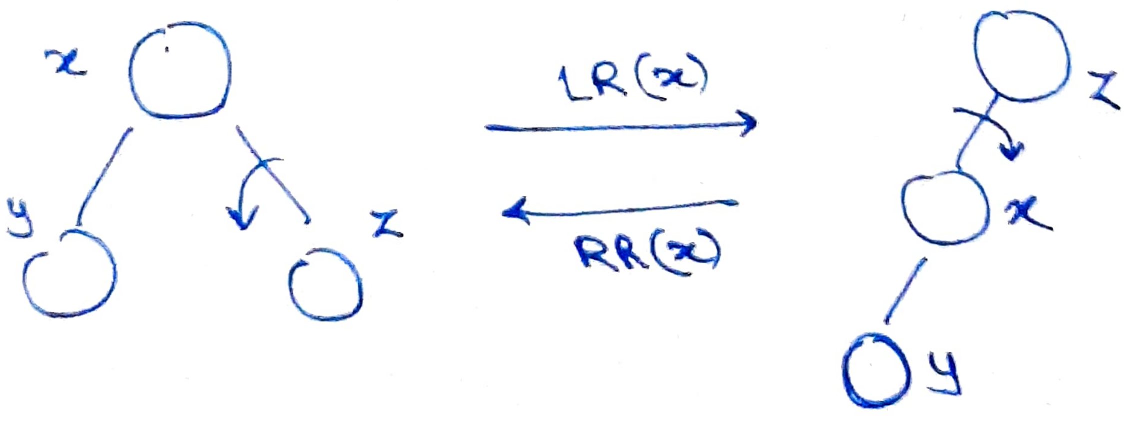


Now, given a set of numbers/nodes, there can be only 1 unique decreasing ~~order~~ (or increasing) order.

Hence, this violates our assumption that T_1 and T_2 are different.

Hence Proved.

(2) Also note that left rotations and right rotations are inverse operations on each other, i.e, when applied on the same edge ~~[illegible]~~, the configuration of the tree remains unchanged.



(Proof not provided as this is given in lecture notes)

(3) Now consider a map from set S_1 to S_2. This is a many to one map [From resut (1)]

This suggets that when we consider the reverse map, a left-linear BST consisting of a certain nodes can be converted to any of its parent BST from which it was made, by apply the reverse operation of right rotations in exactly the reverse order.

Existence of a many to one map is guranteed by the proof in part (b).

So suppose for a given BST B_1, we apply left-rotation operations $[L_1, L_2 \dots L_i]$ on B_1 to convert it to a left-linear BST, then we can apply equal amounts of right-rotations $[R_i, R_{i-1} \dots R_1]$ on the respective edges to get a BST from the left-linear BST.

8.2] (d.) Worst case happens when the BST to be converted to left-linear tree is entirely right-linear tree.

For any right-linear Bstree, with n nodes, maximum (n-1) left-rotations are required.

Hence upper bound on number of left rotations = (n-1)

→ A similar case happens for upper bound on Part (c). For us to convert an entirely left-linear Bstree to right-linear BST, (n-1) right rotations are required.

Hence upper bound on number of right rotations $= (n-1)$