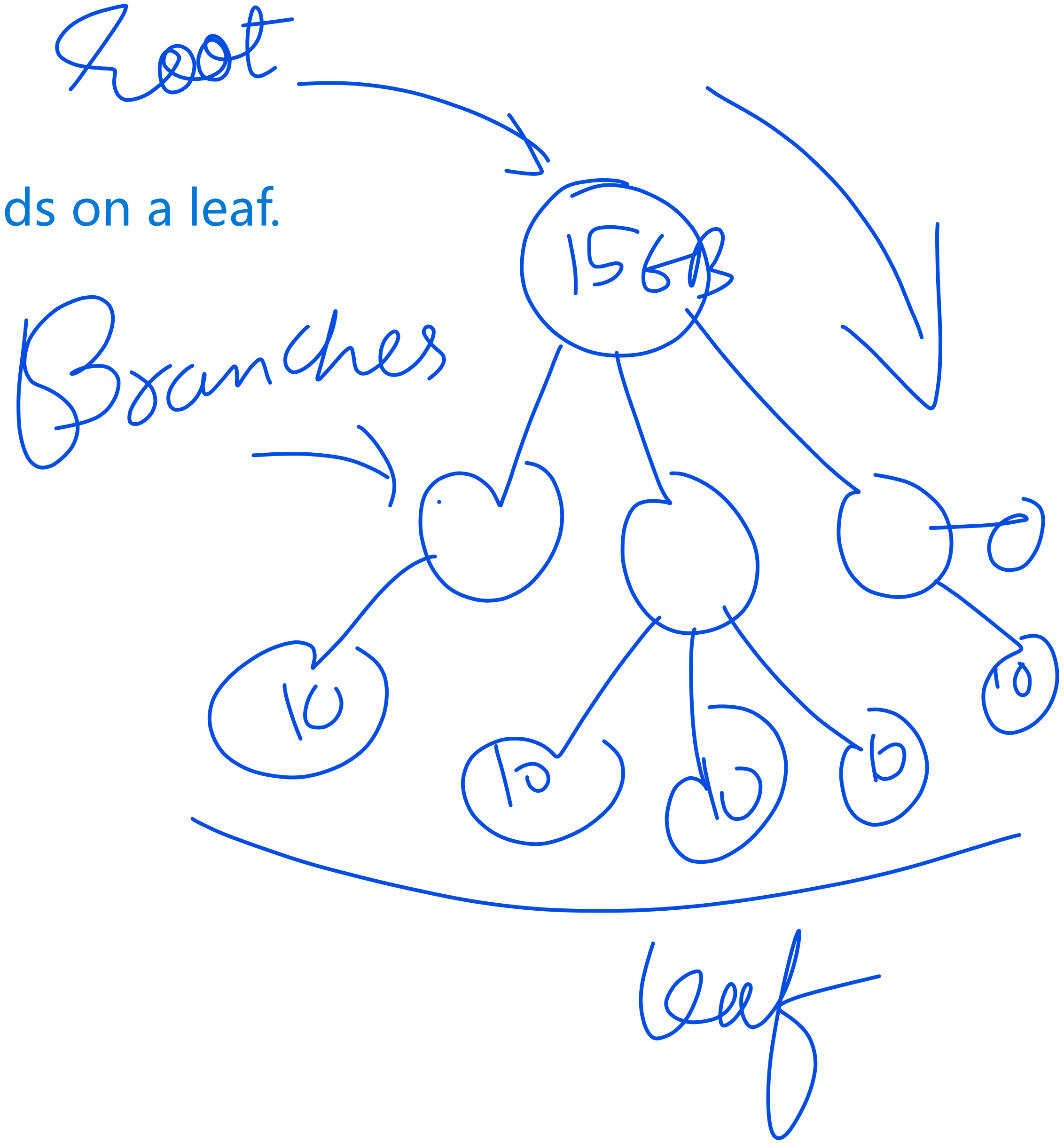
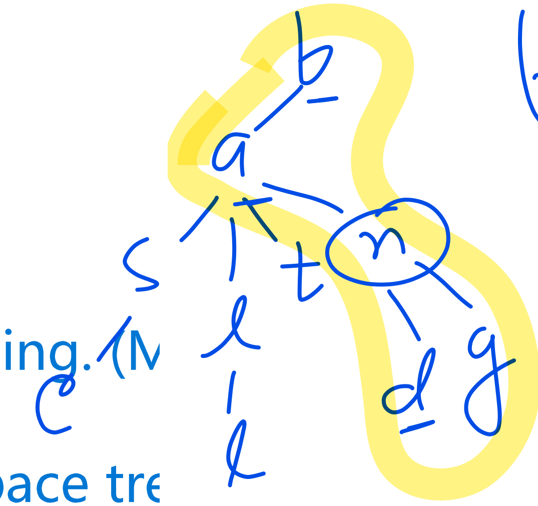
Free is a hierarchical structure.
Requires guided function for providing access.
Starts with the root grows with the branches and ends on a leaf.
tree: most imp root(only stored others referred)

Tree



TREE

Application:
Hierarchical storage.
Large-scale data searching and sorting.
In Geographical Information System for Imaging. (M
Auto-complete, auto-correct
Used by AI in decision-making called state space tre

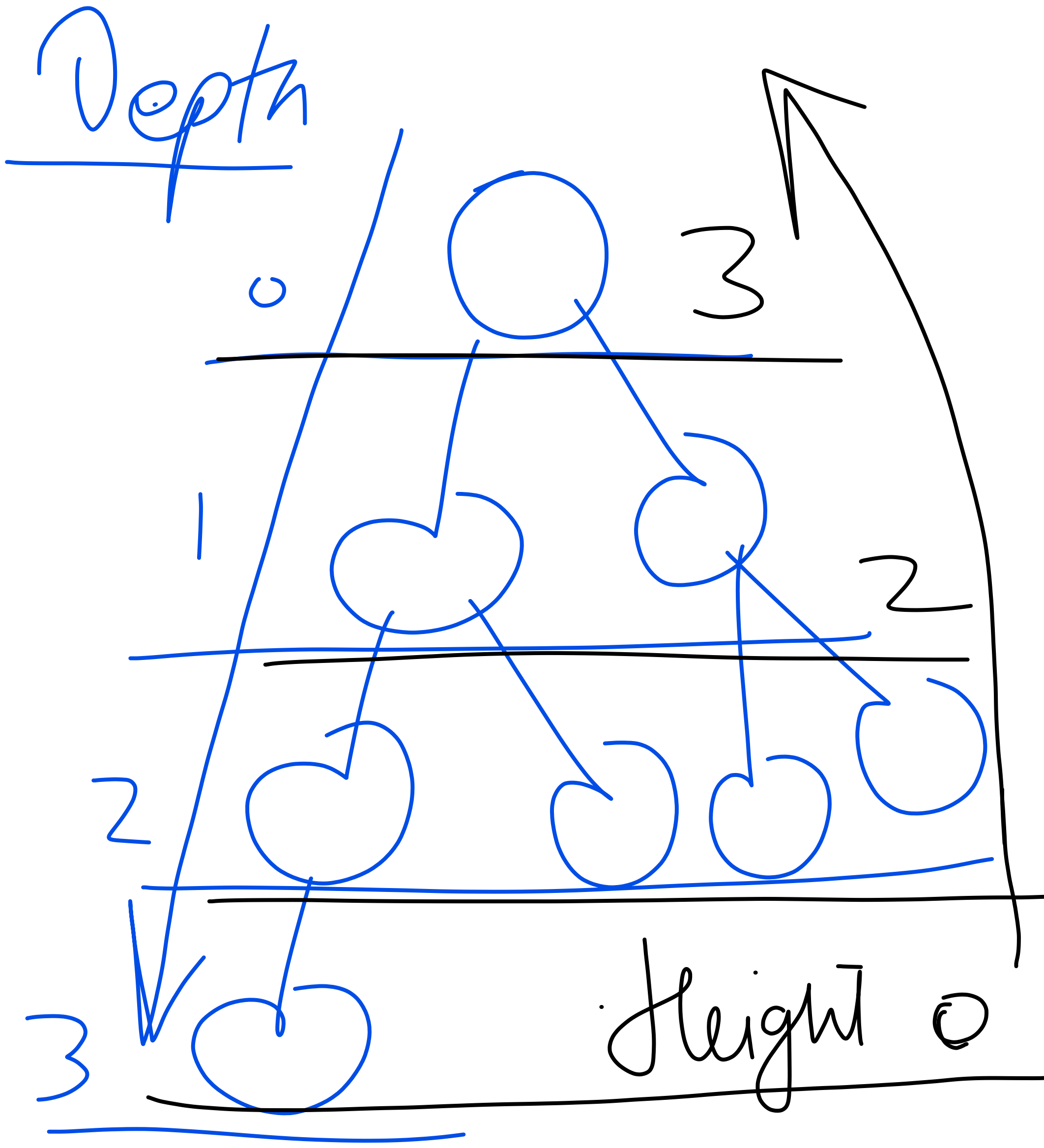


base
bat
ball
band
bang

ban ~~bnd~~ band

Binary Tree At most two children.

- Terms
  - Depth Length of longest running branch. decides speed of access. Lesser The depth, higher the spee
  - Height Higher the height, higher the spee
  - Subtree part of tree
  - Ancestor root of tree
  - Leaf node last node of tree
  - Root node start of tree,at most/at least one
  -



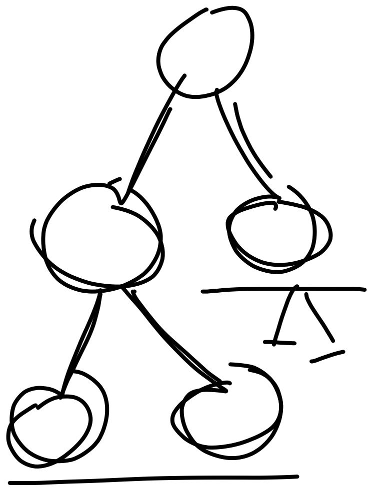
Types of Binary Tree

- Complete All leaf nodes are at exact one depth.
- Nearly complete All leaf nodes except one is at depth +1 or -1
- Strict Each node can either have two children or zero.
- Binary Search Tree

  tree is specially designed for faster searching and sorting.

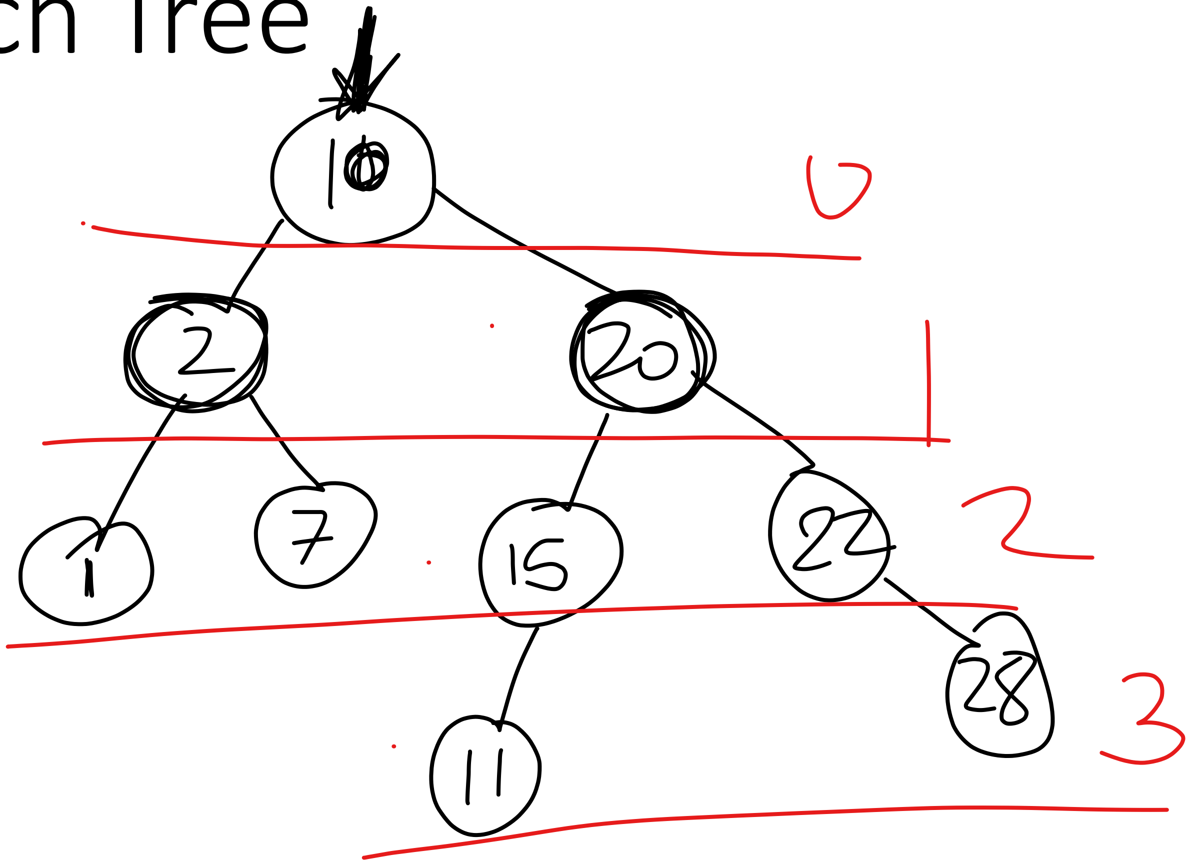
  All elements lesser than the parent are to the left of the parent.

  All elements greater than or equal to the parent are to the right of the parent.



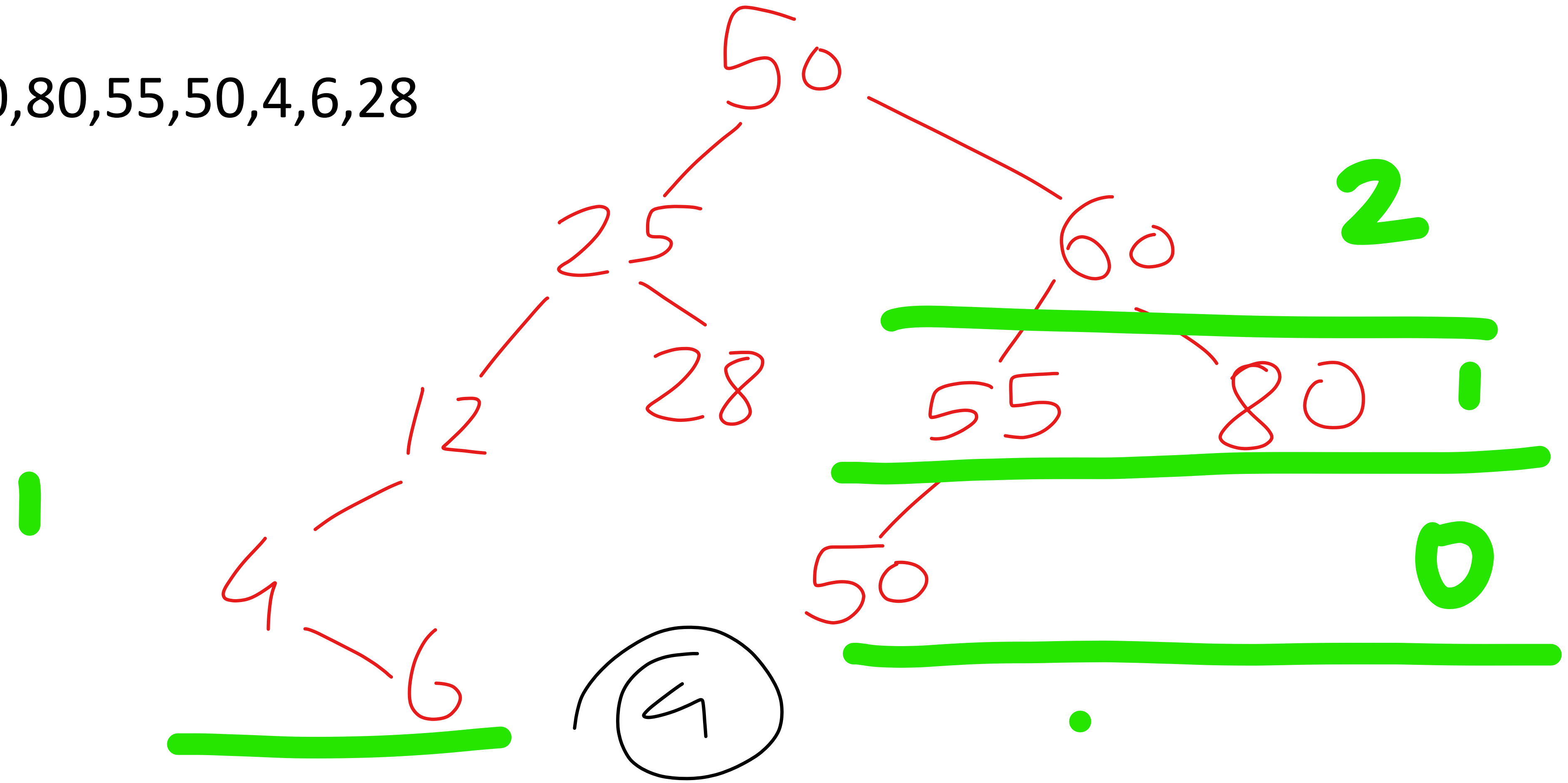
Construct Binary Search Tree

- ~~10,2,7,1,20,15,11,22,28~~



Construct Binary Search Tree

- 50,25,12,60,80,55,50,4,6,28



Tree Traversal

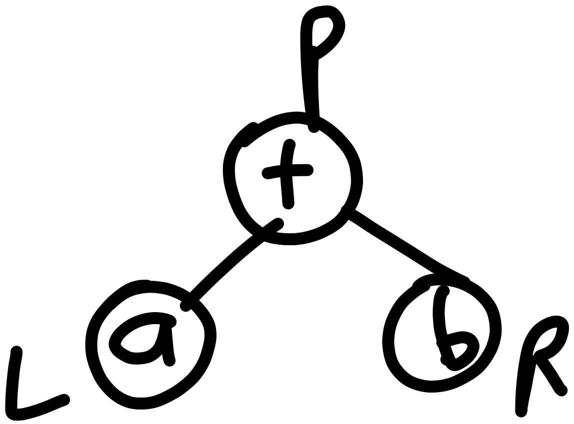
Order of visiting every no
sequence

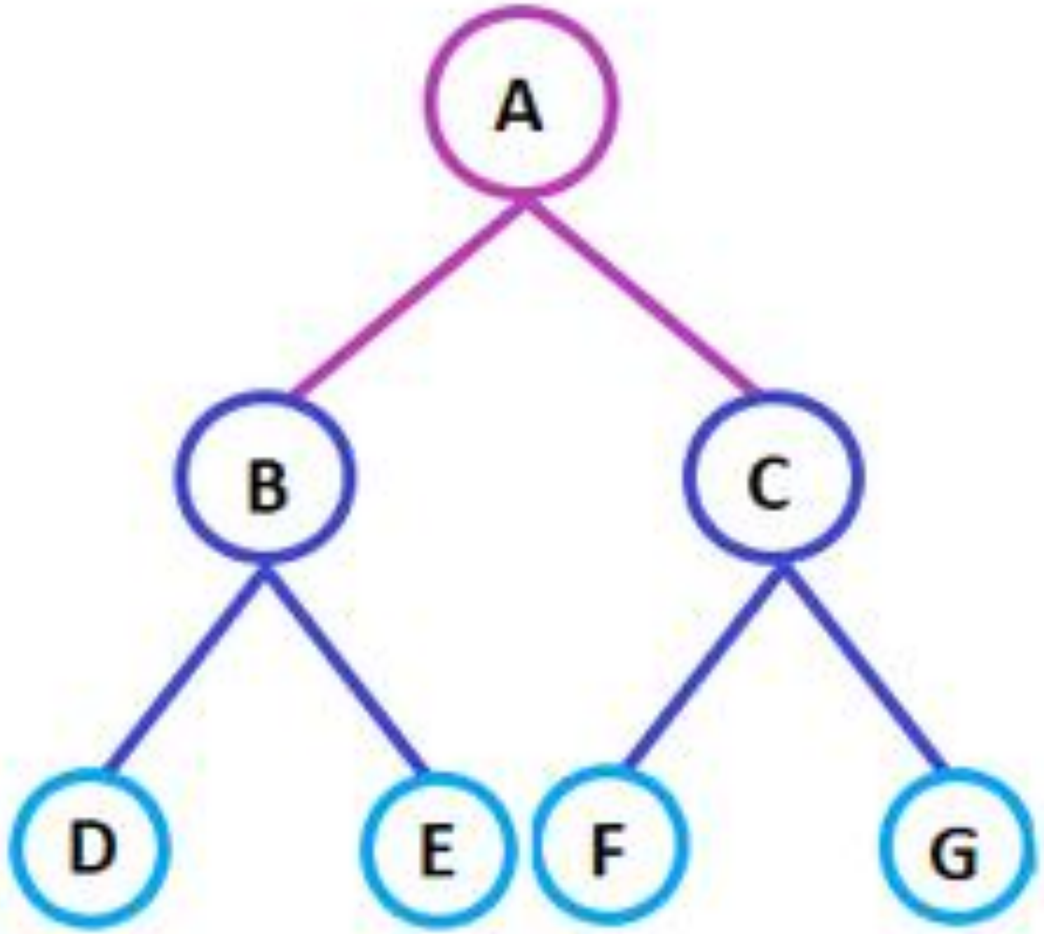
when u cut L:go to left
when u cut R:go to right
when u cut P:print
when all done go to pare

preorder:PLR
+ab

inorder:LPR
a+b

postorder:LRP
ab+



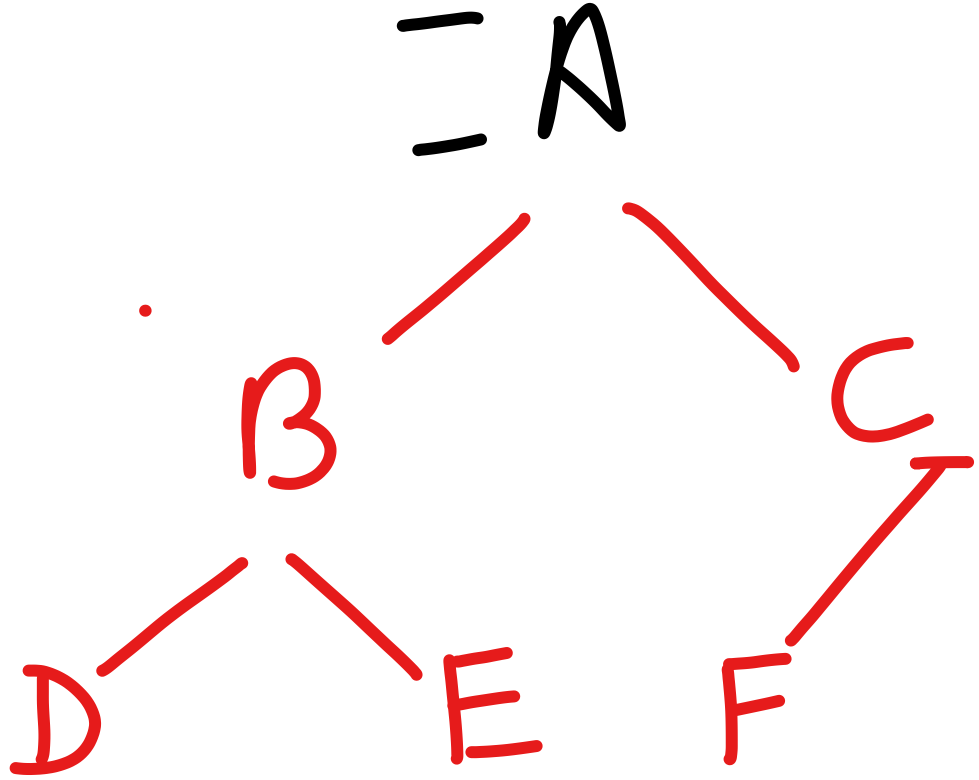


Construct tree

- Inorder sequence: D B E A F C
- Preorder sequence: A B D E C F

We will write in order sequence and us
post-order or pre-order for the
construction of a tree.

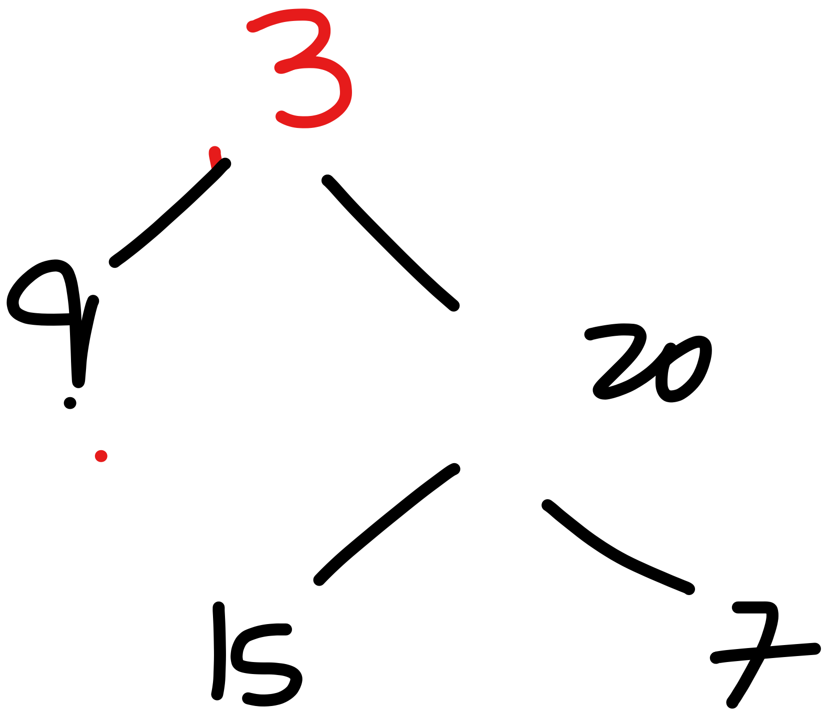
In pre-order go first(root) to last
Cut one node at a time.
Construct left child and right child.
Repeat the process till tree is construct



Construct tree

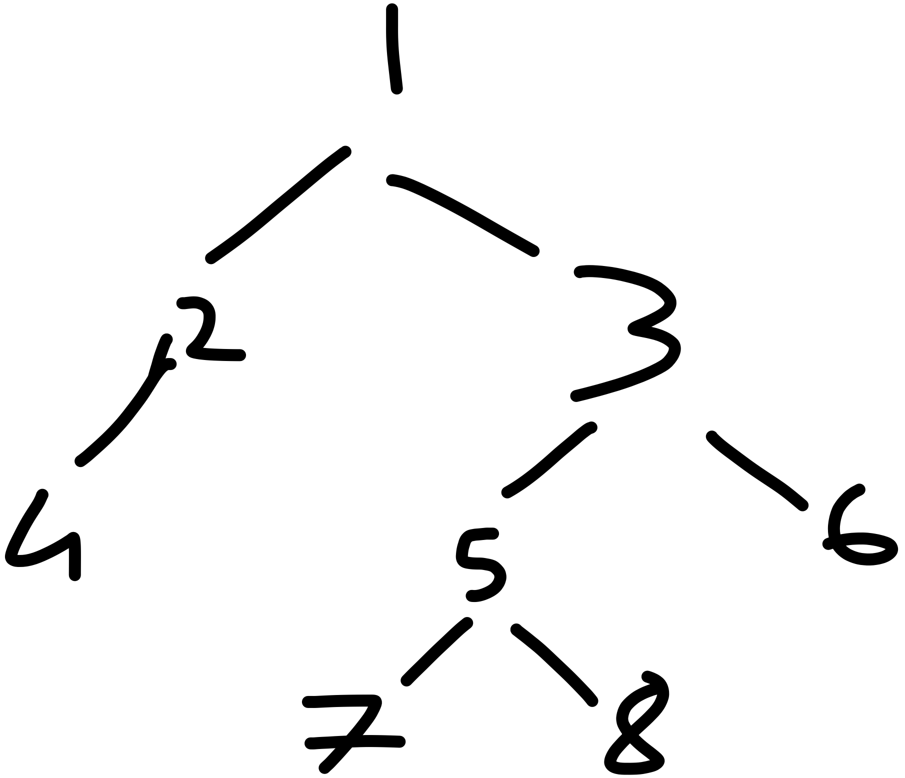
- preorder [~~3~~,~~9~~,~~20~~,15,7]
- inorder = [9,3,15,20,7]

In pre-order go first(root) to last
Cut one node at a time.
Construct left child and right child.
Repeat the process till tree is constructed.



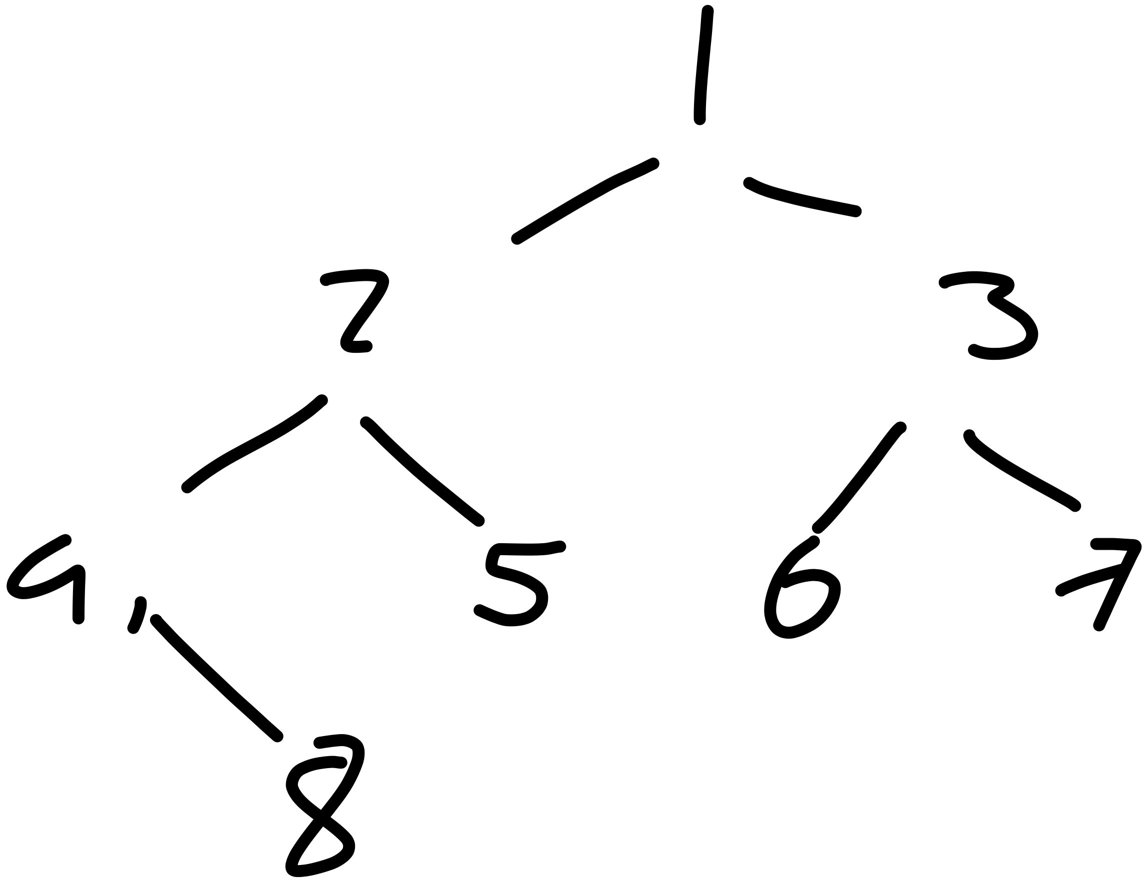
Construct tree

- Inorder Traversal : { 4, 2, 1, 7, 5, 8, 3, 6 }
- Preorder Traversal: { ~~1~~, ~~2~~, ~~4~~, ~~3~~, 5, 7, 8, ~~6~~ }



Construct tree

- in[]   = {4, 8, 2, 5, 1, 6, 3, 7}
- post[] = {8, ~~4~~, ~~5~~, ~~2~~, ~~6~~, ~~7~~, ~~3~~, ~~1~~}



Construct tree

- inorder = [9,3,15,20,7]
- postorder = [9,15,7,20,3]

Construct tree

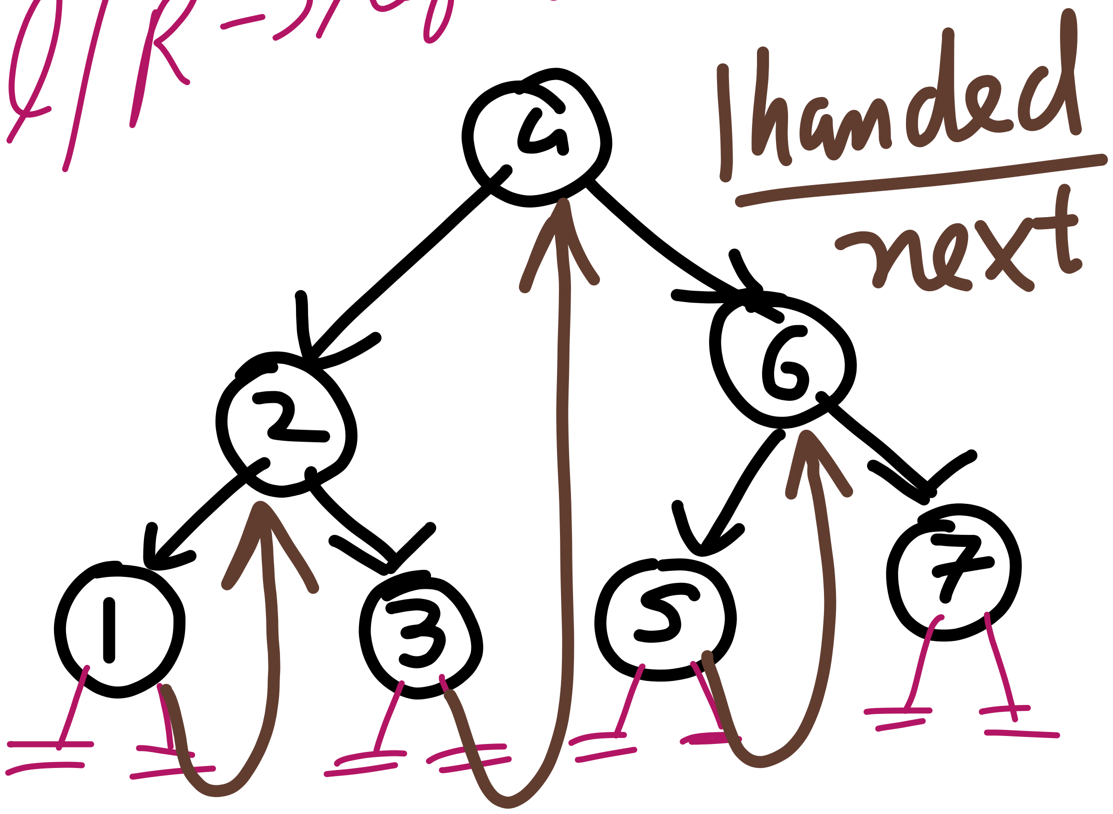
- Inorder Traversal : { 4, 2, 1, 7, 5, 8, 3, 6 }
  Postorder Traversal : { 4, 2, 7, 8, 5, 6, 3, 1 }

Threaded Tree

A special version of a binary tree enhanced for sequential access in-order.
By default, data nature has a sequential nature.

l/R → ref to next

- Inorder traversal of a Binary tree can either be done using recursion or with the use of a auxiliary stack. The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).



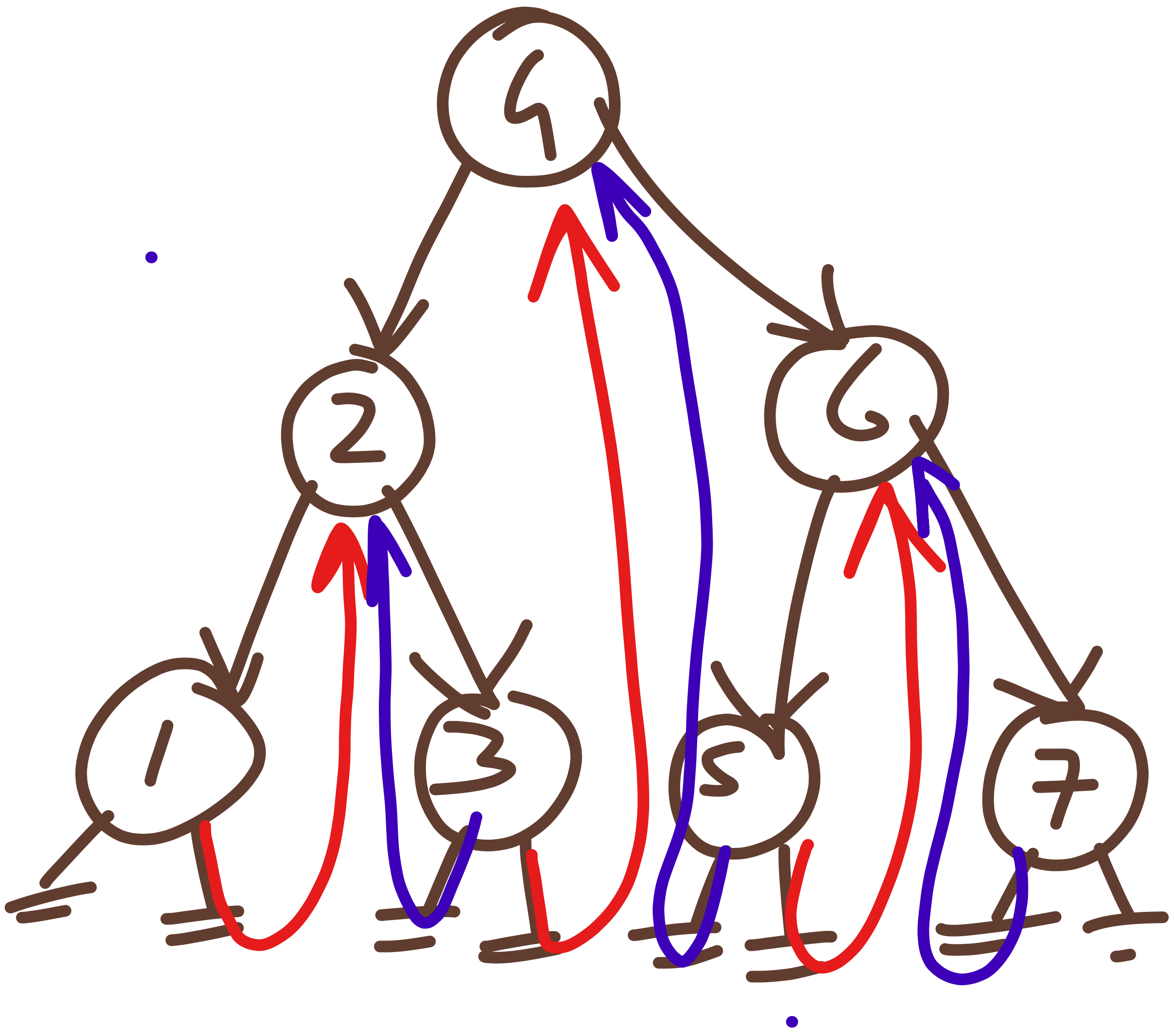
Threaded Tree

- Single Threaded: Where a NULL right pointers is made to point to the inorder successor (if successor exists)

Threaded Tree

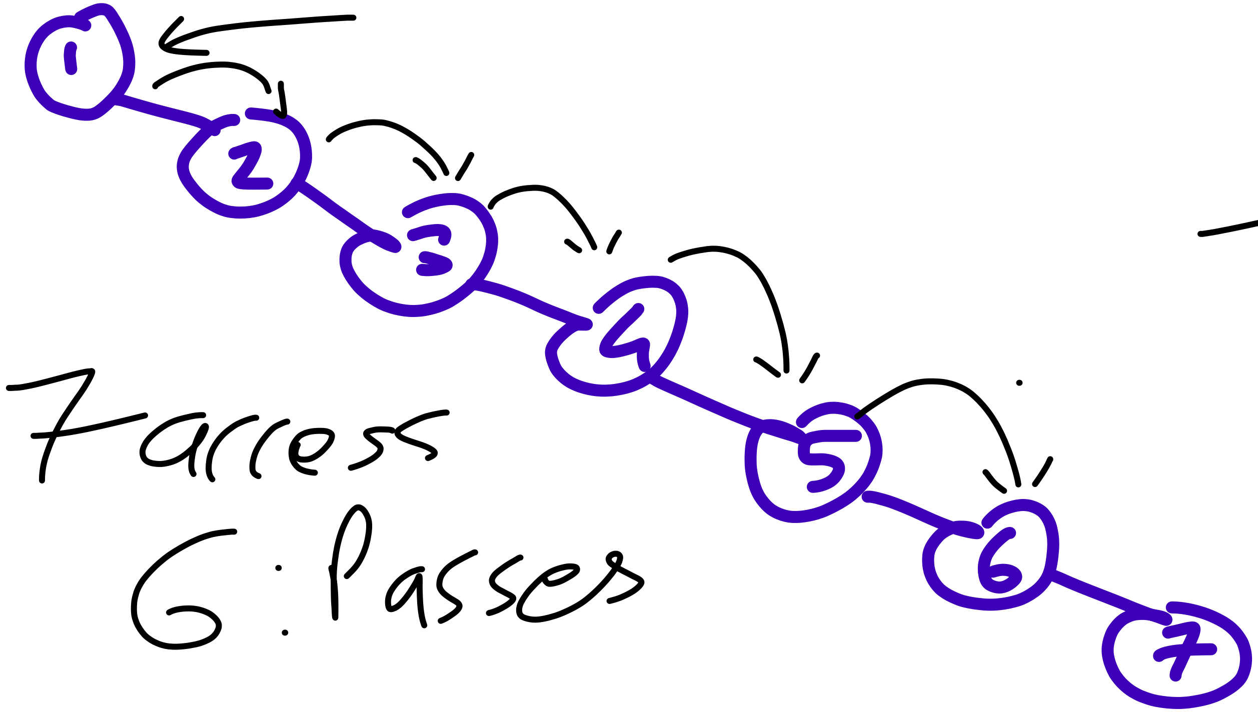
- Double Threaded: Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively. The predecessor threads are useful for reverse inorder traversal and postorder traversal.

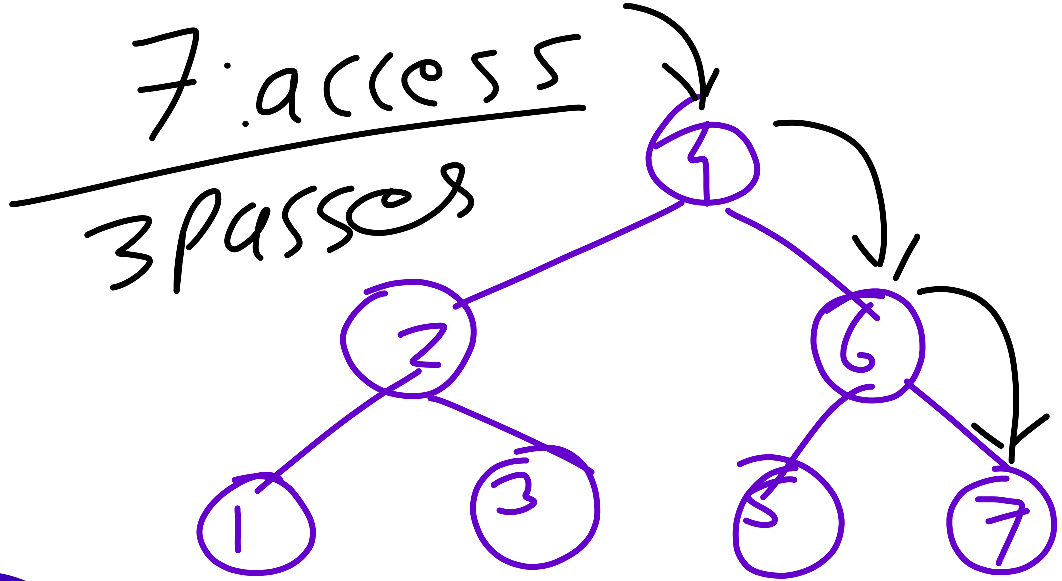
L: Prev
R: next



AVL TREE

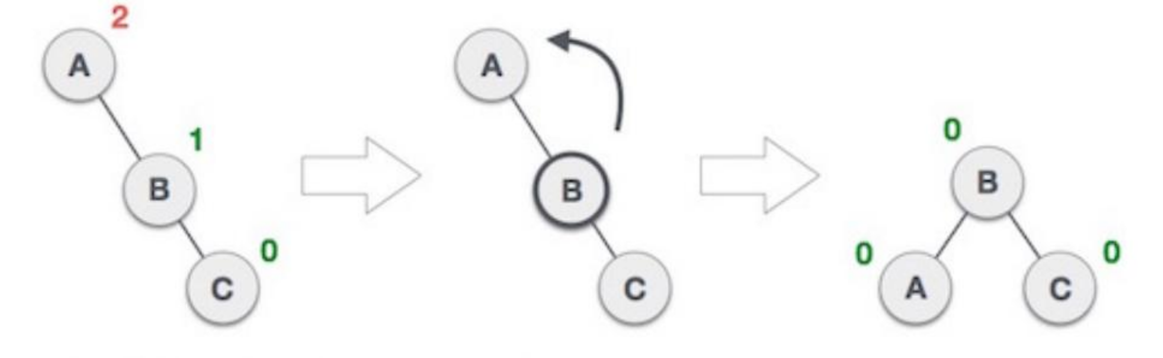
It is self-balancing(if goes out of balance, it will rotate itself to achieve balance)
nearly balanced tree(factor can be between +1 to -1.(+1,0,-1)
balance factor=height of left sub tree-height of right sub tree
If a tree is balanced, it takes lesser time to access the data, increasing the speed of access.





Rotations

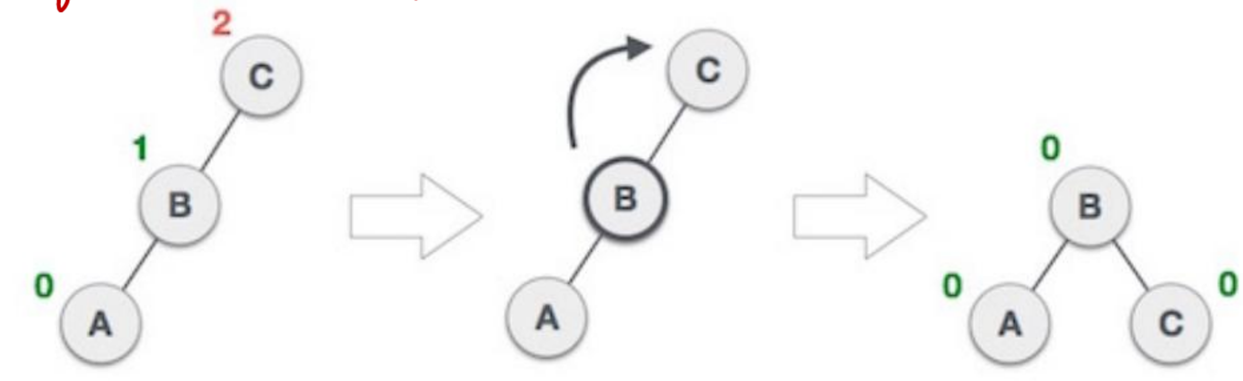
Single



Right unbalanced tree — Left Rotation — Balanced

Right of Right

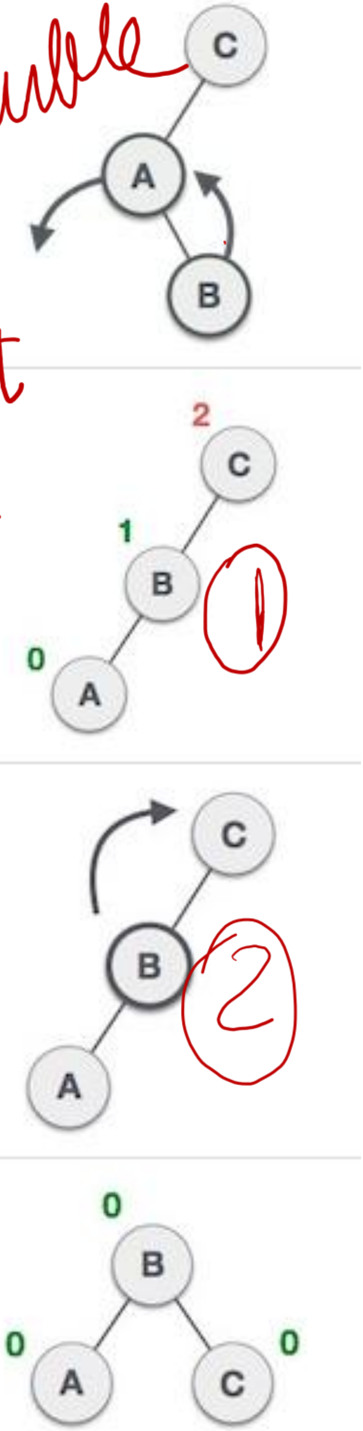
Left of Left



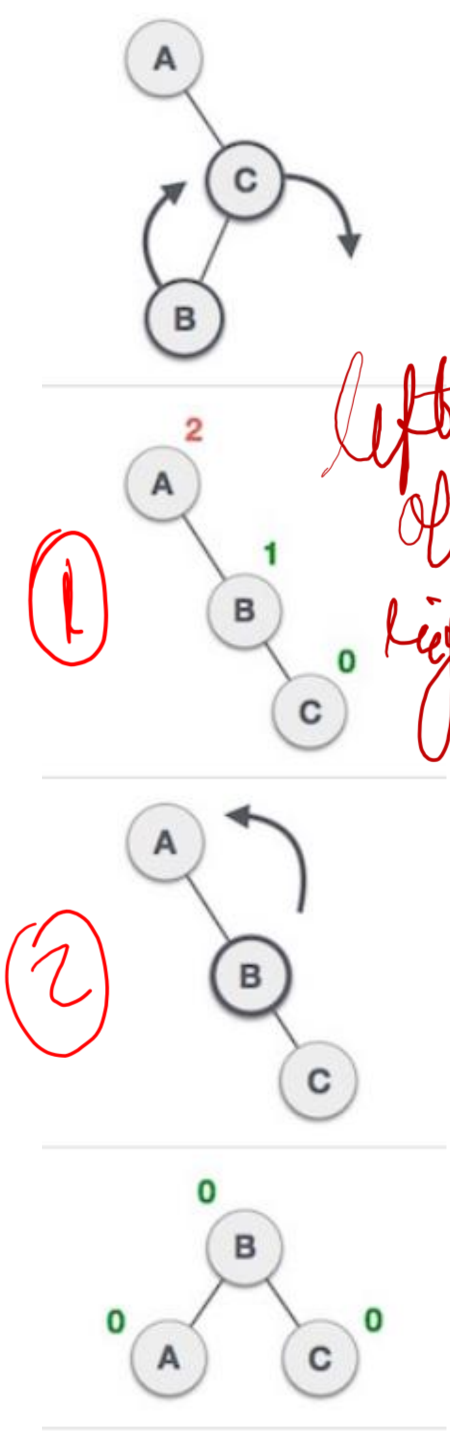
Left unbalanced Tree — Right Rotation — Balanced Tree

Double

Right of Left



Left of Right



B-Tree

It is an example of a multi-way search tree.
In multi-way search tree, a particular node has multiple eleme
decided by its order m. Multi-way search tree allows faster
reading and writing. In a single pass, multiple elements in a no
can be filled.

- B-Tree is a self-balancing search tree.
- Single node consist multiple elements this reduces number of read write and memory swaps.
- the B-Tree node size is kept equal to the disk block size
- All leaves are at the same level.
- B-Tree is defined by the term minimum degree 'M'.
- M is odd.
- On Order M:
  - Maximum M-1 elements at a node
  - Minimum M/2 elements at a node
  - Split when M elements reached
- All keys of a node are sorted in increasing order.
- B-Tree grows and shrinks from the root which is unlike Binary Search Tree.
- Like other balanced Binary Search Trees, the time complexity to search, insert and delete is O(log n).
- Insertion of a Node in B-Tree happens only at Leaf Node in order and BST form.

B+ Tree

BST IMPLEMENTATION

- Static Array
- Dynamic Linked List
- Dynamic Tree node