Page No.
Date

College Name :

DELHI GLOBAL INSTITUTE OF TECHNOLOGY

Name : BAZGHA RAZI

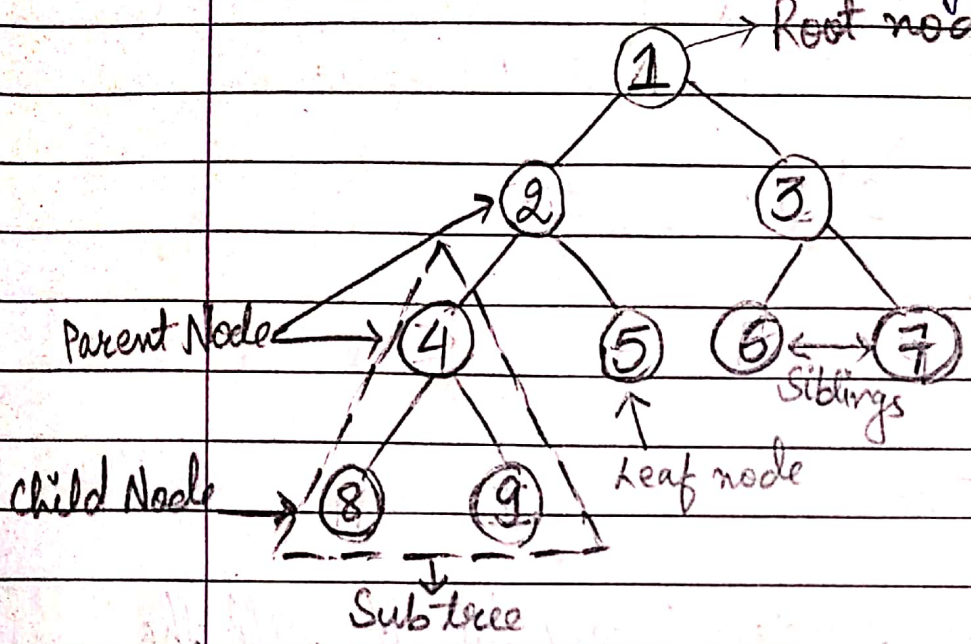
Course Code : PCC-CSE-203G

Subject : Data Structure and Algorithm

Session : 2019 - 2023

Ans 1(c) Binary Tree

A tree whose elements have atmost two children is called a binary tree.



A node in binary tree has 0, 1 or 2 successors.

A parent can have atmost Two children i.e., left child, right child.

The data of left child is less than the parents data/ root node's data.

The data of right child is greater than the parent's data / root node's data.

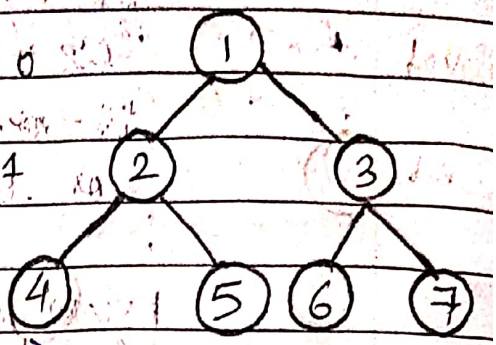
Nodes having the same parent are called siblings.

Page No.
Date

Properties of binary tree

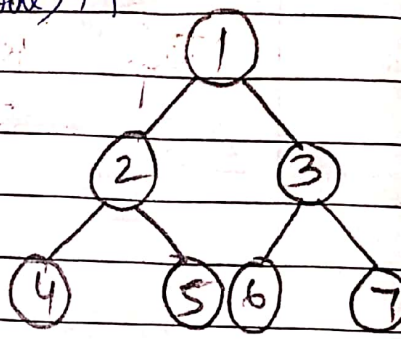
(i) No. of internal nodes = (leaf node - 1)

In this figure;
Leaf node = 4
Therefore,
No. of internal nodes = (leaf node - 1)
= 3

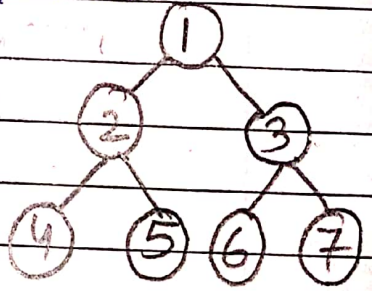


(ii) Total number of nodes = (2 * internal node) + 1

In this figure;
Internal node = 3
Therefore,
Total no. of nodes = (2*3) + 1
= 7



(iii) Max. number of nodes = $(2^{H+1} - 1)$ at level L
where; H is the height of the tree
So, height of the tree = 2
⇒ Maximum no. of nodes = $(2^{H+1} - 1)$
$= 2^{2+1} - 1$
$= 7$



(iv) Minimum no. of nodes = (H+1)
⇒ Minimum no. of nodes = 3

(v) Maximum number of nodes at any level 'L' = 2^L
⇒ If level, L = 3
⇒ Max. no. of nodes = $2^3 = 8$

Page No.
Date

Ans 1 d) Threaded Binary Tree

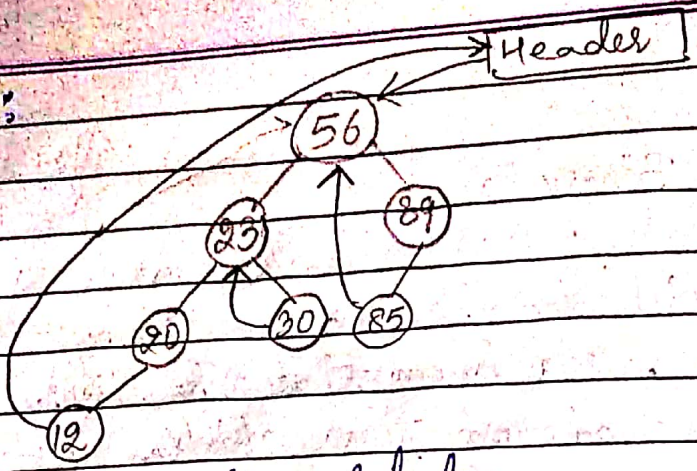
Here we will see the threaded binary tree data structure. We know that the binary tree nodes may have atmost two children. But if they have only one children, the link part in the linked list representation remains null. Using threaded binary tree representation, we can reuse that empty links by making some threads.

If one node has some vacant left or right child area, that will be used as thread. There are two types of threaded binary tree. The single threaded tree or full threaded binary tree. In single threaded mode, there are another two variations. Left threaded and right threaded.

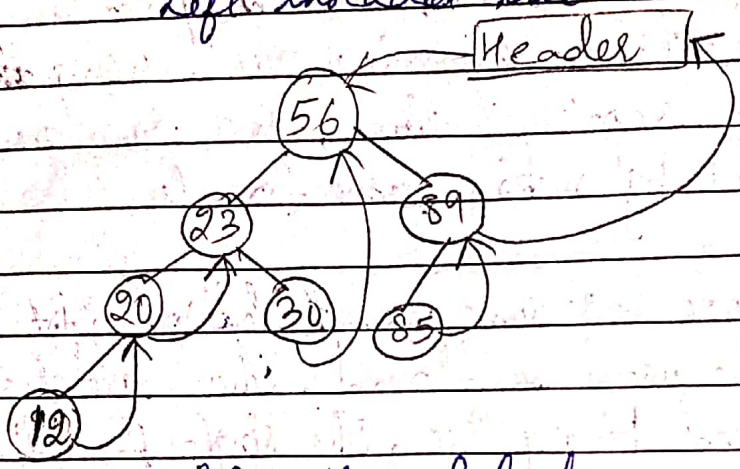
In the left threaded mode if some node has no left child, then the left pointer will point to its inorder predecessor, similarily in the right threaded mode if some node has no right child, then the right pointer will point to its inorder successor in both cases, if no successor or predecessor is present, then it will point to header node.

For fully threaded binary tree, each node has five fields. Three fields like normal binary tree node, another two fields to store Boolean value to denote whether link of that slide is actual link or thread.

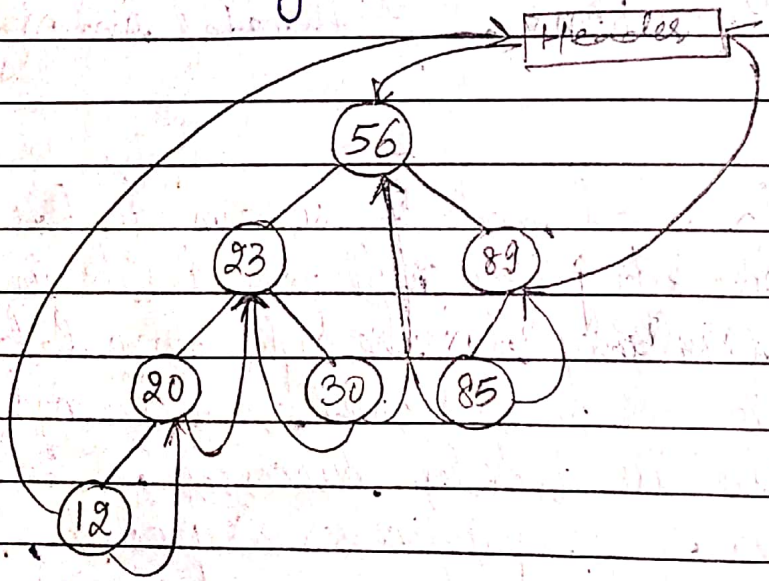
Example:



Left threaded tree



Right threaded tree

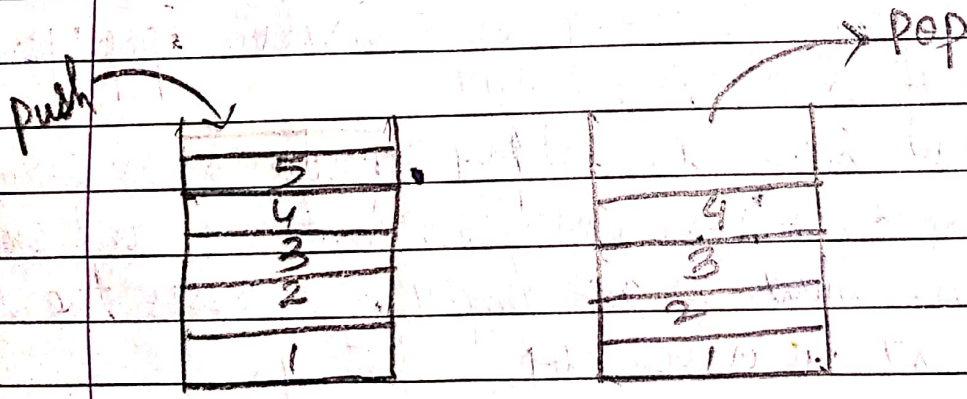


Fully threaded tree

Page No.
Date

Ans 2(a) Stack: It is a linear data structure which follows a particular order in which the operations are performed:
The order may be LIFO (Last In First Out) or FIFO (First In First Out)

Mainly the following basic operations are performed in the stack:
Push: Adds an element / item in the stack. If the stack is full then it is said to be an overflow condition.
Pop: Removes an item / element from the stack. The items are popped in reversal order in which they are pushed. If the stack is empty then the condition is underflow

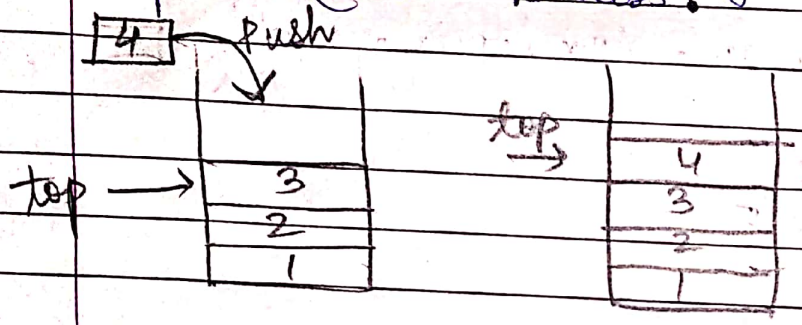


Stack: Last In First Out

A stack can be implemented by means of Array, structure, pointer and linked list.
Stack can either be a fixed size one or it may have a sense of dynamic resizing

Now, following steps involved in push operation:

Step 1: Check if the stack is full.
Step 2: If the stack is full, produces an error or overflow condition and exit.
Step 3: If the stack is not full, increments top to point next empty space.
Step 4: Adds data element to the stack location, where top is pointing.
Step 5: Returns success.



Applications of stack

i) Expression handling: The stack can be used to convert some of infix to postfix or postfix to infix expression. As well as infix to prefix & prefix to infix expression.

ii) Backtracking Procedure: It is one of the algorithm designing technique.

iii) Stack is also used during the function call and return process.

To use a stack efficiently, we need to check the status of stack aswell. For this, following functionality is added to stacks:

peek() – get the top data element of the stack, without removing it.

isEmpty() – Check if stack is empty.

isFull() – Check if stack is full.

Every time we maintain a pointer to the last Pushed data on the stack. As this pointer always represents the top of stack. Top pointer provides top value of the stack without removing it.

~~Now, Below are the follow steps involving~~
Now, here are the following steps involving for pop operation:

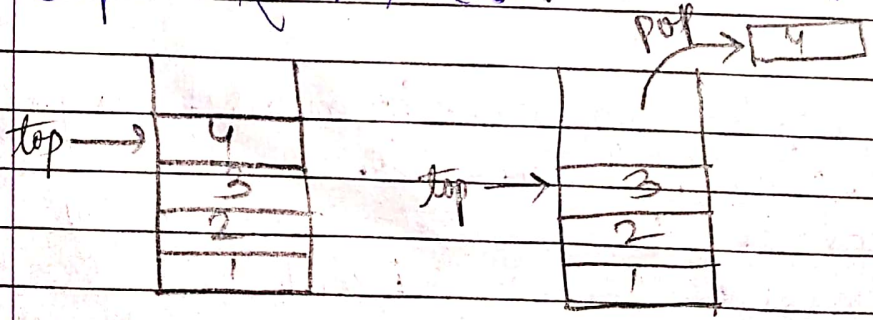
Step 1: Check if stack is empty

Step 2: If the stack is empty, produces an error ~~and exit~~ or underflow condition and exit.

Step 3: If the stack is not empty, accesses the data element at which top is pointing.

Step 4: Decreases the value of top by 1.

Step 5: Return success.



Page No.
Date

Ans 3a) AVL trees are balanced trees whose worst case time complexity is equivalent to binary search.

AVL tree was invented by Adelson Velsky and Landis in 1962. So tree was named in honour of its inventors.

AVL tree mainly depends on two factor:

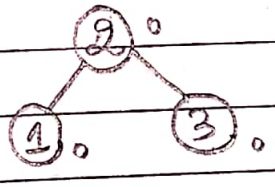
i) It should be a binary tree
ii) Its balanced factor should be 0, 1, -1

Balanced factor mean,

Balanced factor = $H_L - H_R$

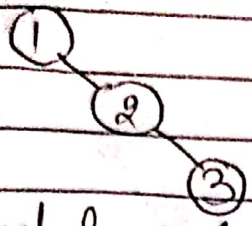
where: H_L → height of left subtree
H_R → height of right subtree

Example:



Balanced

In this tree node 3 and 1 has zero balanced factor because they are leaf node and balance factor of 2 is also zero because the height of the left subtree is also 1 and height of right subtree is also one so difference is zero. Hence, this tree is balance or we can say it is an AVL tree.

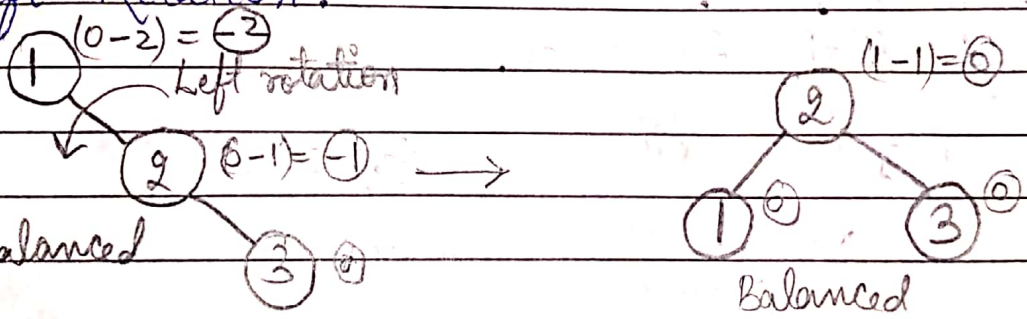


Unbalanced

In this figure the balanced factor of 3 is zero as 3 is a leaf node. Balanced factor of 2 is -1 because the difference of height of left tree and height of right tree is -1.
But the balanced factor of 1 is -2 because the difference of height of left tree and height of right tree is -2.
Hence this tree is unbalanced / Unbalanced.

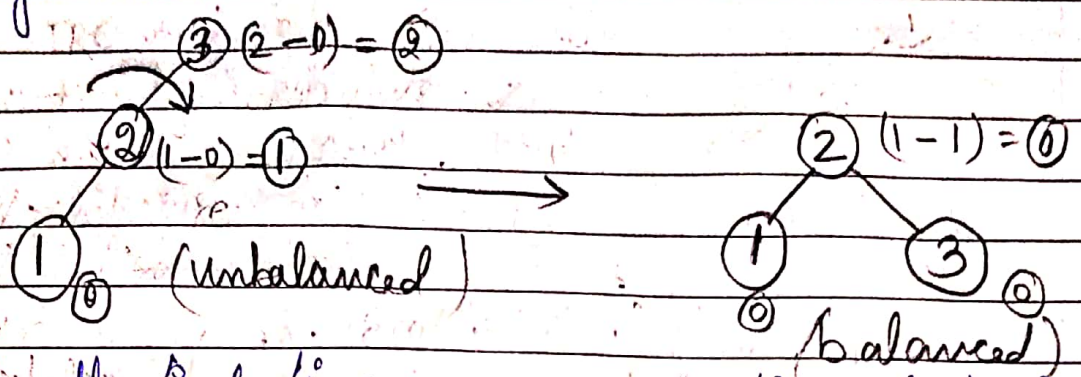
To make these type of unbalanced tree as balanced tree then the following rotation is done:

i) Left Rotation.



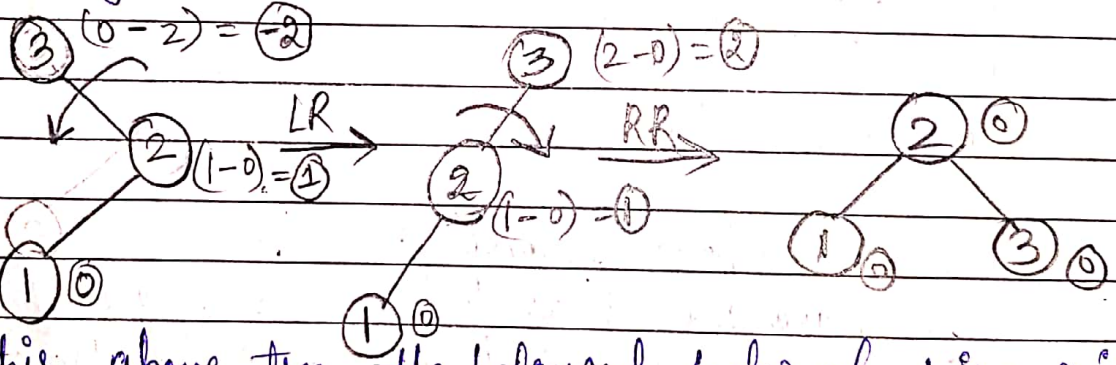
In the above example:
the balanced factor of 3 is 0, 2 is -1 and 1 is -2
So it is unbalanced because of the balanced factor of 1 is -2. Hence, we have to do a rotation i.e., left rotation to balanced this tree.
After left rotation the balanced factor of 1 is zero and 2 is zero and 3 is also zero hence the tree is balanced.

ii) Right Rotation



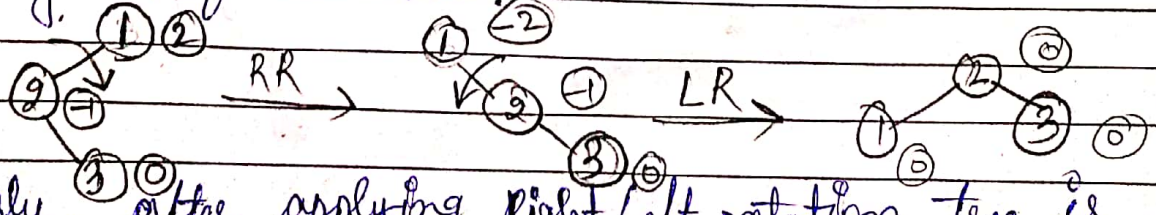
In the first figure, we can see that the balanced factor of 1 is 0, 2 is 1 and 3 is 2. So, balanced factor of 3 is 2 therefore it is unbalanced. So we have to do a right rotation so that tree becomes balanced.
Hence after applying right rotation, the balanced factor of 1 is 0, 2 is 0, 3 is 0 hence it is balanced tree.

iii) Left-Right Rotation.



In this above tree the balanced factor of 1 is 0, 2 is 1 and 3 is -2. Then we have to do a left rotation first after left rotation the tree is still unbalanced so we do a right rotation. Now, after left-right rotation tree is balanced.

iv) Right-Left Rotation.



Similarly after applying Right Left rotation tree is balanced.