Binary Trees

Construct Binary Tree

Use stack and pair class and state.

Sum, Size, Max, Height

Simple dfs approach

Pre, in post traversal

Use stack and pair class and state

Levelorder traversal

Use Queue

Find and Node to root path

Using arraylist. Fill in the list if data found.

Print K levels down

High level: Just decrease 'k' in every recursion,

If k == 0 then print data and return.

Print * Nodes k Distance Away

* Get the node to root path
* Print k levels down from every node in list (keep decreasing 'k').
* while going k levels down, take care of the blocker node.

Path to Leaf from root in range

* Use recursion on the way up.

Print Single child nodes

* Just check left and right of node conditions with respect to the parent node.

Remove Leaves in BinaryTree

* with return value
* without return value

Create Left cloned Tree

* Create a new copy, and change the attachments of node

Transform to normal from Left cloned tree

* Use some logic on the left node over each node to transform.

if (node.left != null && node.left.data == node.data)
→ adjust left pointers accordingly.

Tilt of binary Tree

Return sum, calculate tilt

Tilt is defined in two terms

Tilt of node | Tilt of tree

Tilt of node = abs(sum of left sbt - sum of right sbt)

Tilt of tree = Summation of tilt of all nodes

Diameter of binary tree:

Number of edges b/w two farthest nodes of the tree

Tilt of binary tree : Approaches:

1) Using static variable for tilt of tree TC O(n) SC O(n)
2) Using class which returns a pair of (sum and tilt) TC O(n) SC O(n)

Diameter of binary trees | Return height, calculate dia

Distance b/w two farthest nodes interms of edges.

i) lh + rh + 2 (one node left, one node right)

ii) Ldia (both nodes in left)

iii) Rdia (both nodes in right)

Approaches:

* Using Static variable - diameter
* Using ~~static~~ class pair - (diameter & height)

Is Balanced binary tree:

Balanced tree property,

Balancing factor | |lh - rh| ≤ 1

nodeBalanced: (lh - rh) ≤ 1

Tree balanced: when all it's nodes are balanced

Here, you must calculate height interms of node.

Approaches:

* Using static variable
* Using class Pair

Is a BST (Binary Search Tree)

all nodes in leftsubtr < node.data < all nodes in rightsubtr

* You will need to return both min & max of a node from both L.subtr, R.subtr