Friday, 23 October 2020 6:24 PM

→ Arrays
→ Linked List
→ Stacks
→ Queues
→ Hashmaps

Hierarchical data structure. [tree]

→ Recursive data structures

Binary Tree

Each node of the tree can have atmost 2 child.

class node {
int data;
node left;
node right;
}

→ Inorder
→ Preorder
→ Level

→ Depth of a tree

Q. The max no of nodes at level 'l' of a binary tree.

$l=1 \quad n=1 \quad \rightarrow 2^0$

$l=2 \quad n=2 \quad \rightarrow 2^1$

$l=3 \quad n=4 \quad \rightarrow 2^2$

Q. Maximum no of nodes in a binary tree of height 'h' is $2^{h+1}-1$

$h=0 \quad n=1 \quad 2^1-1=1$

$h=1 \quad n=3 \quad 2^2-1$

$h=2 \quad n=7 \quad 2^3-1$

Full Binary Tree

→ Every node is a LEAF, or children are a siblings.

Complete Binary Tree

→ All levels are completely filled except the last level.

→ Addition of keys is not left as possible

Perfect Binary Tree

→ All internal nodes have two children and all leaves are at same level.

Implementation (Binary Tree)

(1) nodes

(2) Arrays

node → 2

$2^1 = 2 \rightarrow 2^2 (4)$
$2^2 = 4 \rightarrow 2^3 (8)$

$l = h$
$2^{l+1} = 2^{h+1}$
$2^{l+1} \rightarrow 6$

10	20	30	40		50
0	1	2	3	4	5

root node → arr[0]

parent node → i

child node (left) → $2i+1$
child node (right) → $2i+2$

Tree traversals

(1) DFS (Depth first search) → Preorder, Inorder, Postorder

(2) BFS (Breadth first search) → Level order traversal

Preorder N L R
Inorder L N R
Postorder L R N

Preorder.

10 20 40 50 30 60 70

void preorder (root) {
if (root == null)
return;
print (root.data);
preorder (root.left);
preorder (root.right);
}

Inorder (LNR)

40 20 50 10 60 30 70

void inorder (root) {
if (root == null)
return;
inorder (root.left);
print (root.val);
inorder (root.right);
}

Postorder

Breadth first search

(level order traversal)

10 20 30 40 50

60 70 80 90 100 110

void BFS (root) {
q = {}
q.enqueue (root);
while ((!q.isempty()) {
node = q.dequeue();
print (n.val);
if (n.left != null)
q.enqueue (n.left);
if (n.right != null)
q.enqueue (n.right);
}
}

Q. Find a element in BT.

check (60) → true / false

bool check (root, key) {
if (root == null) return false;
if (root.val == key)
return true;
}
else {
bool L = check (root.left);
bool R = check (root.right);
return L || R;
}
}

Q. Find height of tree

$\max(l, r) + 1$

$(h, h) + 1$

height (h)

height (root) {
if (root == null)
return -1;
int h = height (root.left);
int h = height (root.right);
return max(l, r) + 1;
}

Q. Given two traversals of a tree, inorder traversal & preorder traversal construct the tree.

Inorder → [D B E A F C]

preorder → [A B D E F C]