Tree

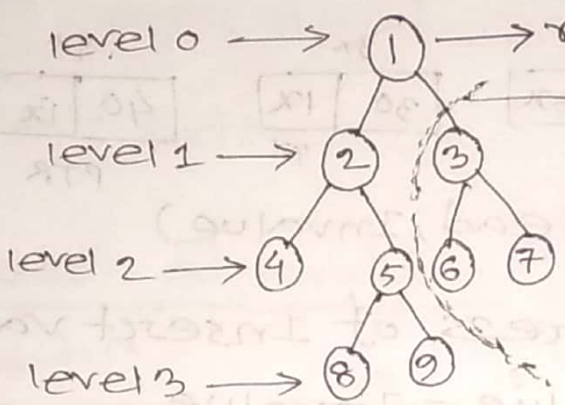


fig: Tree

- what is tree?

Ans: Tree is a type of data structure in which each element is attached to one or more elements directly beneath it.

- Branching Factor: In tree type data structure, the branching Factor is the number of children at each node, the out degree.

- Root: The topmost node in a tree is called root. [① or 1 is root]

- **Internal node:** An internal node is any node of a tree that has child & parent node.
  [2,3,5 is internal node]

- **Leaf:** Leaf is any node that does not have child nodes. [4,8,9,6,7 is called leaf]

- **siblings:** siblings is those node which has the same parent. [889 is called siblings]

- **Subtree:** Any part of a tree is called subtree.

- **Ancestors:** The ancestors of a vertex other than the root are the vertices in the path from the root to this vertex.

- **strictly binary tree or 2-tree or extended binary tree:** A strictly binary tree that is every node can have either no children or two children.
  if n leaf then contains $(2n-1)$ nodes.

Four types of tree:

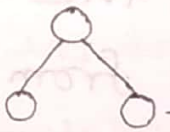
1. Binary search tree.
2. Heap tree:
   - i) Max Heap tree.
   - ii) Min Heap tree.
3. Complete Binary tree.
4. Huffman tree.

Binary search tree:

Condition:

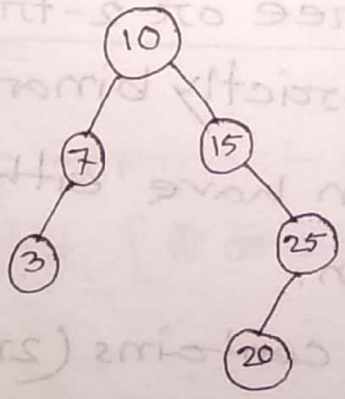
i) Right child will be high than root.

ii) Left child will be low than root.



→ Right বড় আসবে

(10), (15), (25), (7), (3), (20)



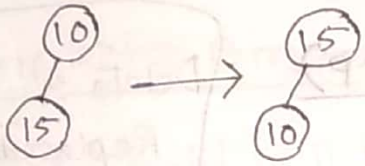
Heap tree (Max Heap)

condition:

i) maximum value will be root than left and Right child.

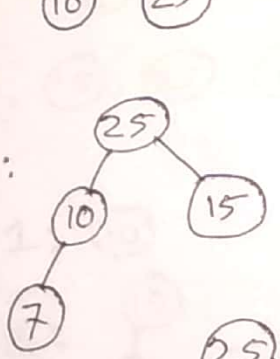
10, 15, 25, 7, 3, 20, 30

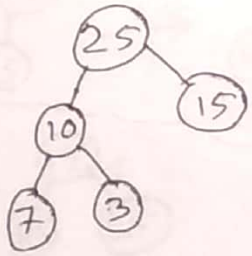
step1: 

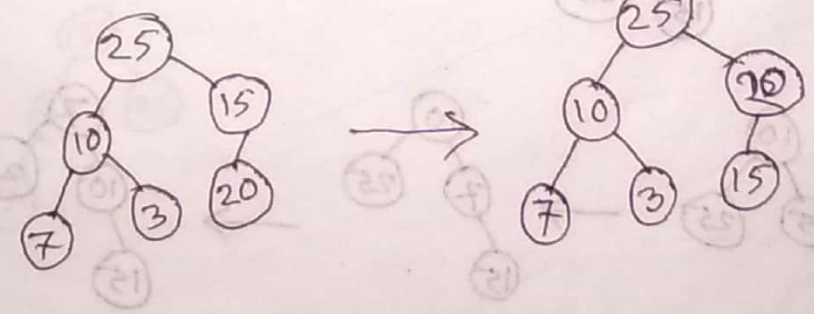
step2: 

step3: 

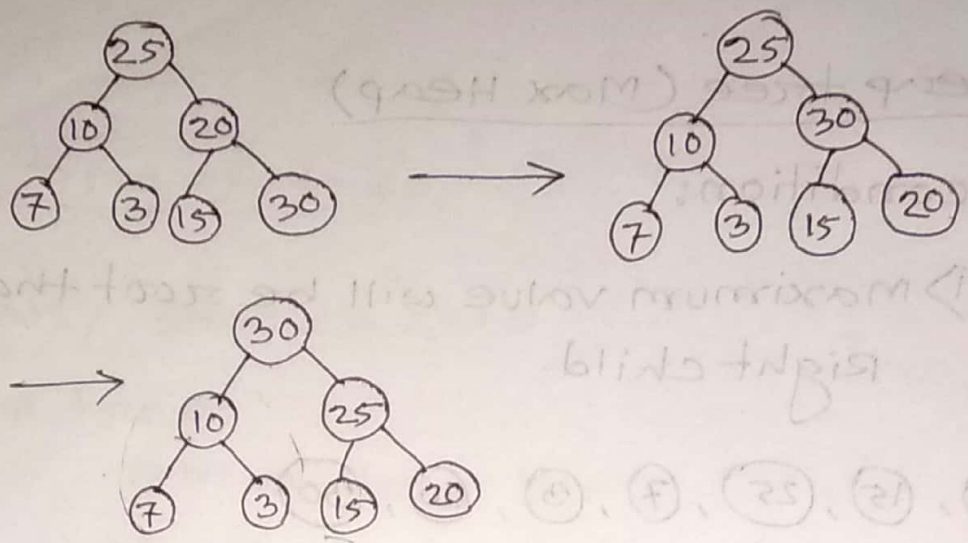
Descendant: A node reachable from parent to child.

step4: 

step5: 

step6: 

step7:



Heap tree (Min Heap)

Delete :- Root Node Replaced by Last Node Then Hepify

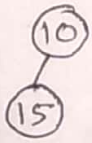
Condition:

1) Minimum value will be root than left and Right child.

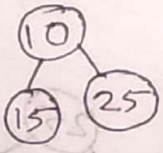
(10), (15), (25), (7), (3), (20)

step1: (10)

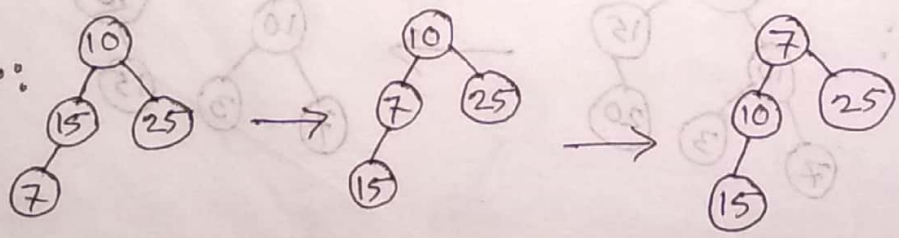
step2:



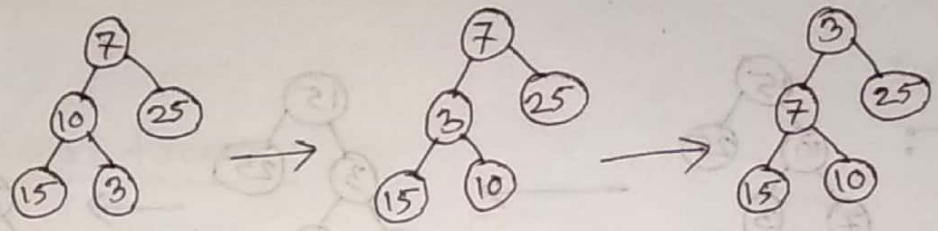
step3:



step4:



step5:



step6:

complete Binary tree:

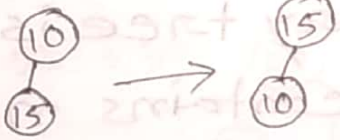
i) left child will be low than root.

ii) Right " " " high " " .

(10), (15), (25), (7), (3), (20)

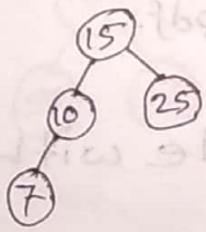
step1: (10)

step2:

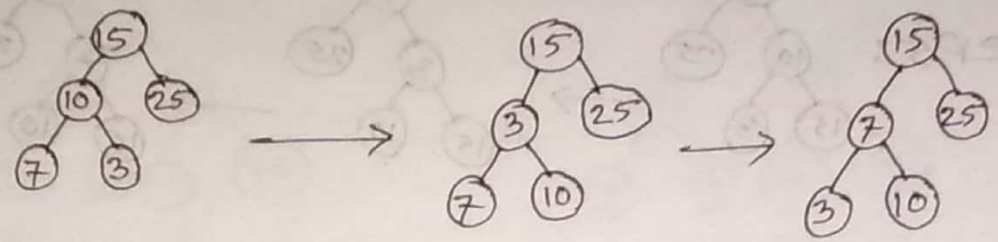


step3:

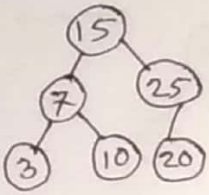
step4:



Steps:



step6:



- complete (full) binary tree: A complete (full) binary tree is the strictly binary tree, where all the leaves are at same level.

- Binary tree: A binary tree is a tree in which each node contains no more than two children.

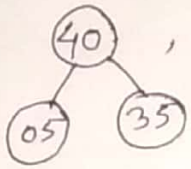
- Fibonocci tree: 275 Page.pdf.

- Terminal node: The node with no successo are called terminal nodes.

Huffman tree:

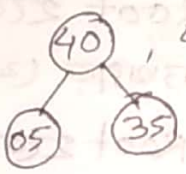
I=50, T=35, D=55, C=05, M=45

step1: 05, 35, 45, 50, 55 [ascending ভাবে সাজাতে হবে]

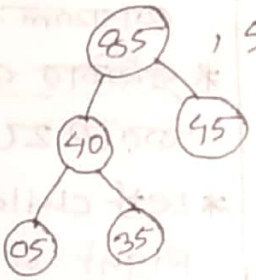


, 45, 50, 55

step2:

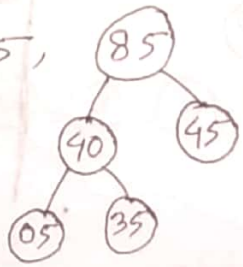


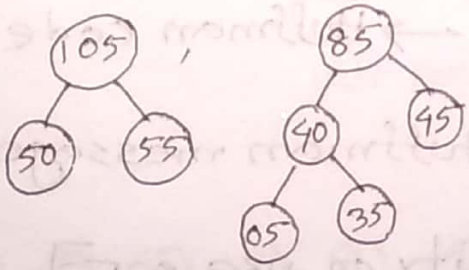
, 45, 50, 55



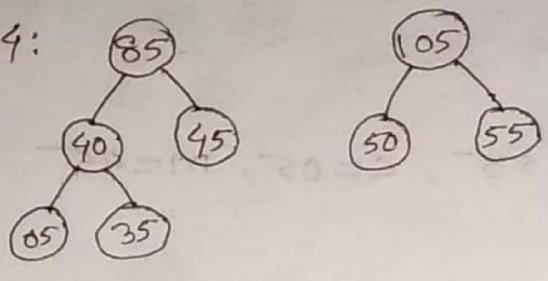
, 50, 55

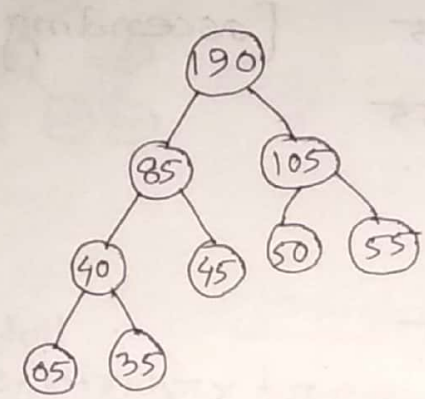
step3: 50, 55,



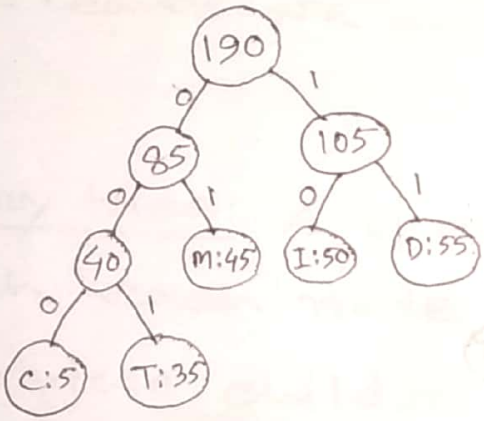


Step 4:





∴ Huffman tree:



* Root হতে সবগুলো সংখ্যার যোগফল
* Root হতে সমস্তটির যোগফল.
* প্রতিবার ascending করতে হবে।
* Left child কে 0 দ্বারা Right কে 1 দ্বারা
* Leaf হতে element গুলো.

T = XX, XX = Last two digits of your sid.

ICT = $\frac{10}{I}\ \frac{000}{C}\ \frac{001}{T}$ → Huffman code

ID = $\frac{10}{I}\ \frac{11}{D}$ → Huffman message

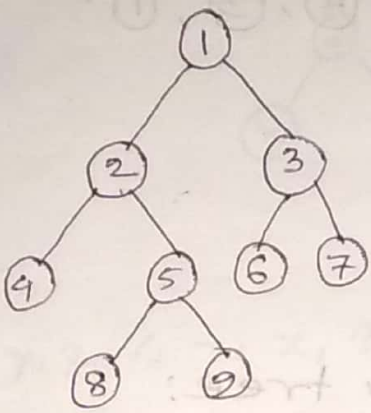
[এই Problem security তে use হয়]

Tree traversing:-

(I) Pre-order: Root, left, ~~child~~ Right

(II) In-order: left, Root, Right.

(III) Post-order: left, Right, Root.



(I) Pre-order:

Root left Right

Root left Right → Root left Right

Root left Right → Root left Right

1, 2, 4, 5, 8, 9, 3, 6, 7.

(II) In-order:

left Root Right

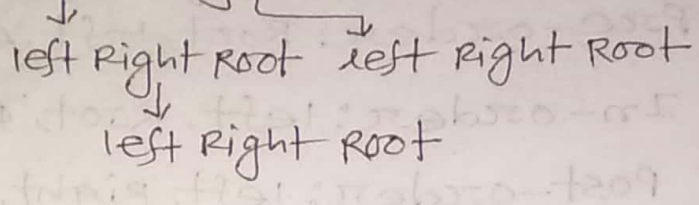
left Root Right → left Root Right

left Root Right → left Root Right

4, 2, 8, 5, 9, 1, 6, 3, 7

(iii) Post order:

left Right Root



(4), (8), (9), (5), (2), (6), (7), (3), (1).

Traversing binary tree:

Preorder: (i) Process the Root R.

(ii) Traverse the left subtree of R in Preorder.

(iii) Traverse the right subtree of R in preorder.

$x=6$
$y=1$

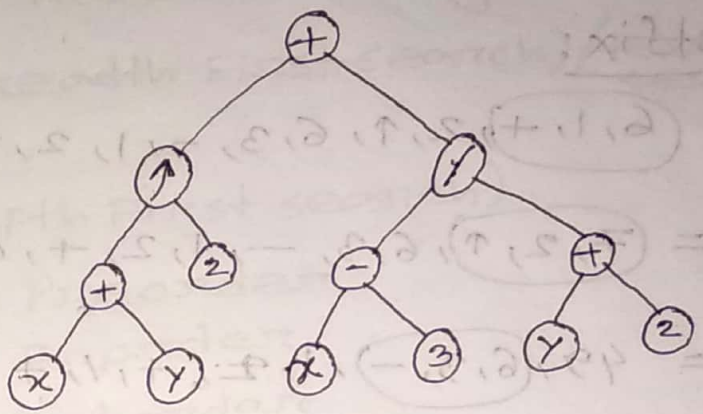
$(x+y)^{\wedge} + (x-3)/(y+2)$

$=(6+1)^{\wedge} + (6-3)/(1+2)$

$= 7^{\wedge} + 3/3$

$= 49+1$

$= 50$



Preorder:

+, ↑, +, x, y, 2, /, −, x, 3, +, y, 2

Prefix notation/polish notation:

+, ↑, +, 6, 1, 2, /, −, 6, 3, +, 1, 2

= +, ↑, +, 6, 1, 2, /, −, 6, 3, 3

= +, ↑, +, 6, 1, 2, /, 3, 3

= +, ↑, +, 6, 1, 2, 1

= +, ↑, 7, 2, 1

= +, 49, 1

= 50.

Preorder-এ সাজানোর পর যখন মান বসিয়ে Evaluate করানো যায় তখন তাকে Prefix বলা। Prefix এ শেষের দিক থেকে কাজ হয়। operator যখন পাওয়া যাবে তখন সেখান থেকে সে দুটি সংখ্যা নিয়ে কাজ করতে হয়।

Postorder:

x, y, +, 2, ↑, x, 3, −, y, 2, +, /, +

Postfix:

(6, 1, +), 2, ↑, 6, 3, −, 1, 2, +, /, +

= (7, 2, ↑), 6, 3, −, 1, 2, +, /, +

= 49, (6, 3, −), 1, 2, +, /, +

= 49, 3, (1, 2, +), /, +

= 49, (3, 3, /), +

= (49, 1, +)

= 50.

Assignment:

1. Prefix ⟶ Postfix
   ⟶ infix

2. postfix ⟶ Prefix
   ⟶ infix

3. infix ⟶ prefix
   ⟶ Postfix

} আমরা যেভাবে infix থেকে Prefix & Postfix.

Tree searching:

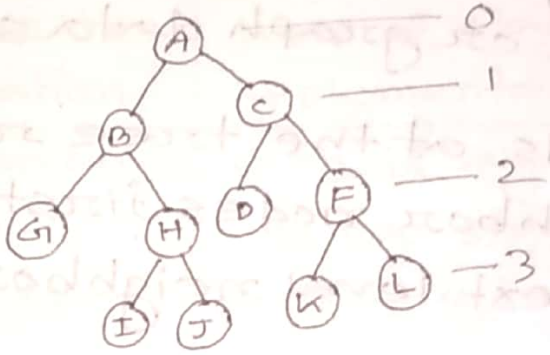
Two types of tree searching.

(i) BFS (Breadth First search)/level search.

(ii) DFS (Depth First search).

- Preorder
- Inorder
- Postorder

[DFS search-এ কোন order উল্লেখ না থাকলে এটা must preorder হবে।



search: (H)

1. BFS search/level search:

(A), (B), (C), (G), (H), (D), (F), (I), (J), (K), (L)

(H) → 5th position

2. DFS search:

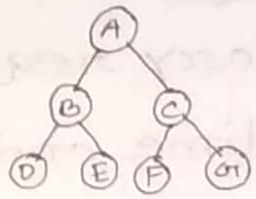
(A), (B), (G), (H), (I), (J), (C), (D), (F), (K), (L)

(H) → 4th position.

(H) → এর জন্য DFS search better.

Pre order: A, B, D, E, C, F, G

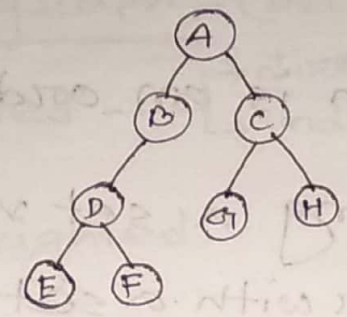
In-order: D, B, E, A, F, C, G.



Postorder: Pre-order and inorder দেওয়া থাকলে post order বের করা যাবে।
যেহেতু Pre order-এ first-এ Root থাকে সেহেতু এখানে A হচ্ছে Root।
∴ In order-এ A হচ্ছে Root এবং A থেকে বামের গুলো left and ডানেরগুলো Right।
Post order-এ সর্বশেষে A কারণ যেহেতু Root সেহেতু বসে। A এর ডানগুলো (F, C, G) দেখতে হবে Pre-order-এ কে আগে আছে। যে আগে থাকবে সে Root। যে A এর আগে আছে পর্যায়ক্রমে একইভাবে হতে থাকবে।

~~Pre~~ Post order:

D, E, B, F, G, C, A.



Post order: E, F, D, B, G, H, C, A.

In-order: E, D, F, B, A, G, C, H.

Pre order: Pre-order বের করতে হলে আগের মত ডান হতে but উল্টো। যেমন, Post order-র Root শেষে থাকে। তাই Preorder এ এ এটা আসে। তারপর In-order-এ যেখানে A আছে তার আশপাশ left এর এবং right. left এর element গুলো post order-এ কে শেষে আছে সে Root অর্থাৎ সে আগে আসে। এভাবে শেষপর্যন্ত যেতে হবে।

Pre-order: A, B, D, E, F, C, G, H.