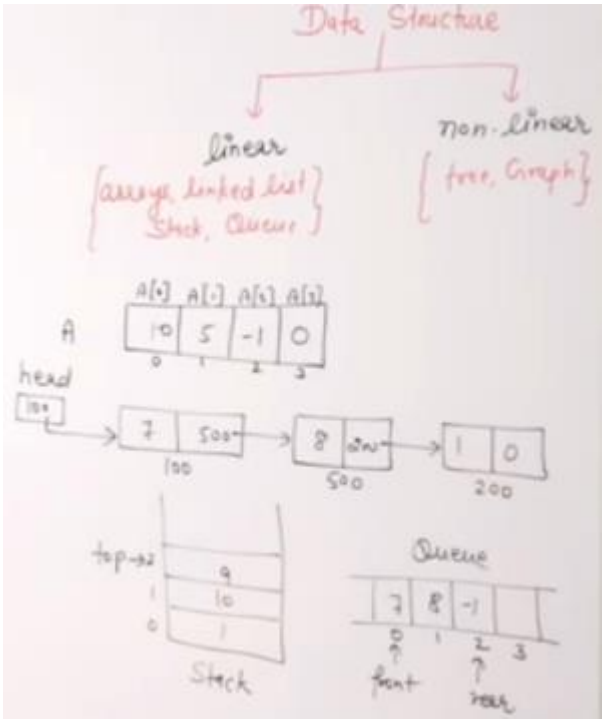
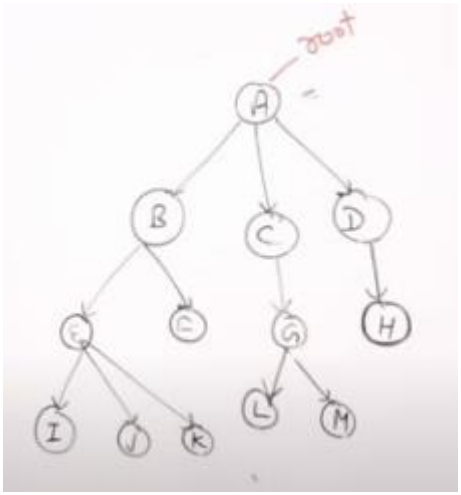
## Introduction to Trees

Stack : LIFO; Queue: FIFO

Struktur Data Linear artinya susunan data secara sekuensial (satu data dengan data lainnya secara terurut/satu per satu), hanya 1 level.




Sedangkan Struktur Data non linear artinya memiliki susunan data secara multiple level/hirarki.




Contoh Tree

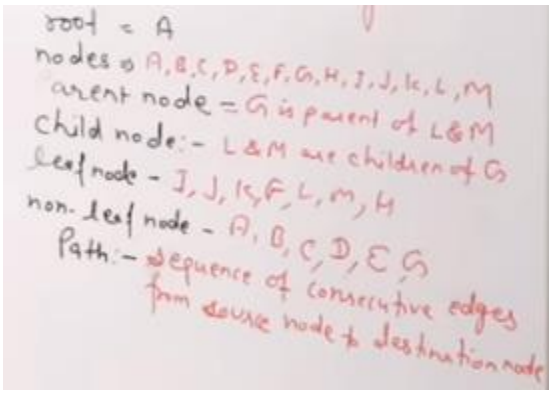
A = root, selain itu sbg child/node.

Misal A sbg direktur memiliki 3 supervisor, kmdn msg2 supervisor memiliki anak buah dst. Shg A sbg level 0; sedangkan B, C, D sbg level 1, kmdn E, F, G, H sbg level 2 dst. Hal tsb yg disebut hirarki.

Tree hanya memiliki 1 arah/ tidak punya arah (krn tdk memiliki child) yaitu arah dari atas ke bawah (level 0 ke level 1, level 1 ke level 2,dst).

Sbgmn tree scr biologi tumbuh dari akar, batang, ranting, daun (tumbuh dari bawah ke atas).

'Tree can be defined as a collection of entities (nodes) linked together to simulate a hierarchy'.



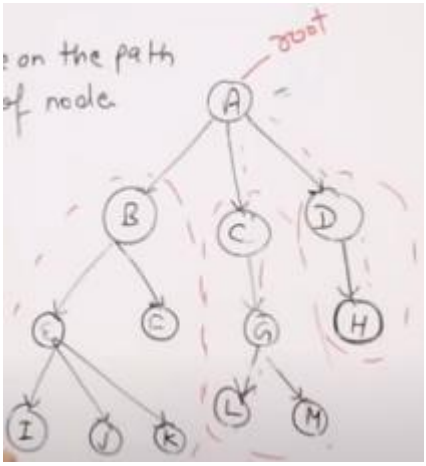

Pada array kita sebut sbg elemen tapi pada tree disebut node. Setiap node memiliki informasi yg mhubkn dg node lainnya. Node A adalah parent dari node B, C, D; sedangkan C mrpkn parent dari G,dst. Jadi parent adalah node dari level sebelumnya (immediate predecessor/previous node). Namun pada Root = tdk memiliki predecessor node/ level 0. Sedangkan child adalah immediate succesors (penerus langsung). Leaf node (eksternal node) = node yg tdk memiliki child. Non leaf node (internal node) = node yg memiliki minimal 1 child.

Path (lintasan)= urutan tepi yang berurutan dari node sumber ke node tujuan. Edge = link antar node. Contoh path = A – C – G – L.

Ancestor = any predecessor nodes on the path from root to that node. Contoh L = A, C, G; H = A, D.

Decendent= any successor node on the path from that node to leaf node.

Contoh: C = G, L, M; B = E, I, J, K, F.




Sub tree dari gambar disamping tdapat 3 sub tree (yg diberi garis putus2 merah).

Sibling = children of same parent, contoh: E & F tapi F & G bukan sibling.

Degree of node = jumlah node (child) yg dipunyai node parent (number of children of that node), contoh: E memiliki 3 degree yaitu I, J, K. Klo degree dari M = 0 degree.

Degree of tree = max degree among all node, dlm cth tree ini = 3.

Depth of node = length of path (jmh edge) from root to that node.

Contoh: Depth of node F = 2; J = 3; D = 1, A = 0, dst.

Height of node = number of edges in the longest path from that node to a leaf.

Contoh: Height of node G = 1; B = 2.

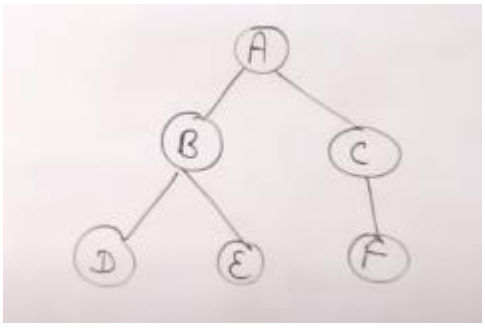
Height of tree = 3.

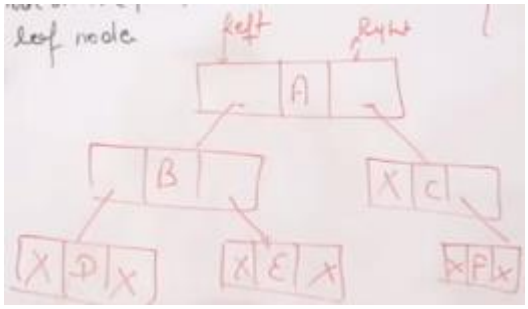
Level of node = jumlah edge dari root ke node tsb. Cth G = 2.

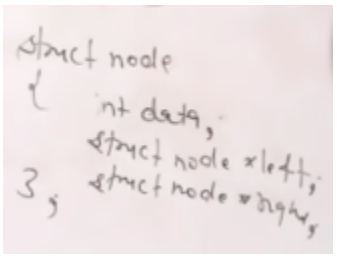
Level = height

Pada tree: n nodes = (n-1) edges

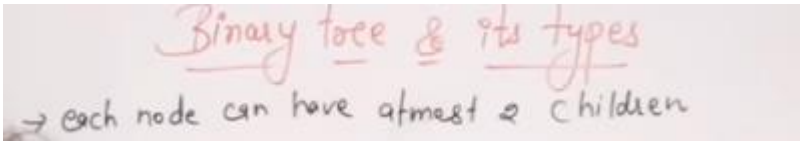
Binary tree: tree yg hy memiliki 2 children.



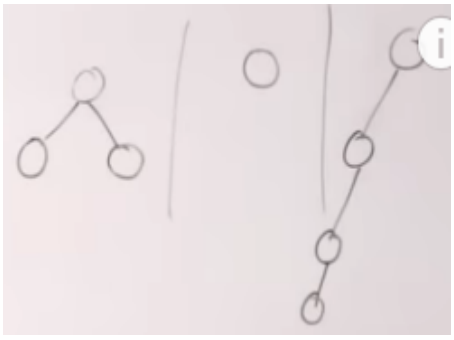
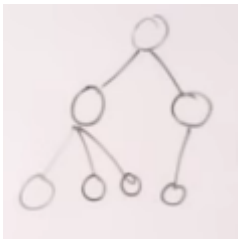
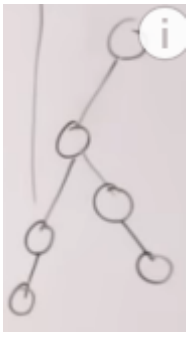
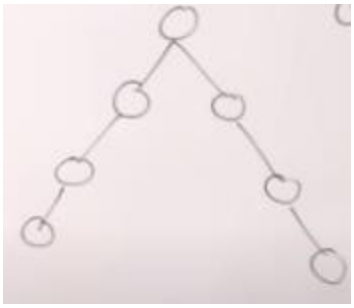



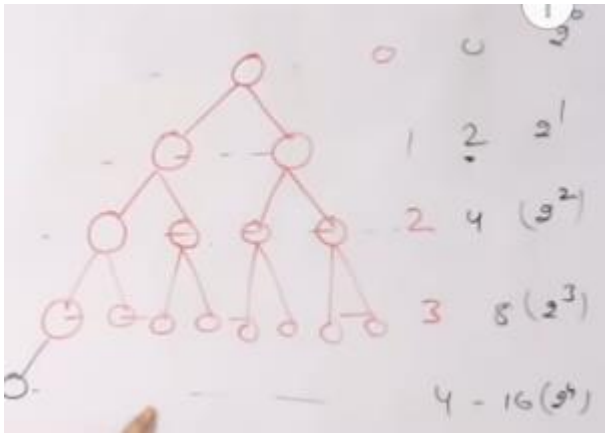


linked list

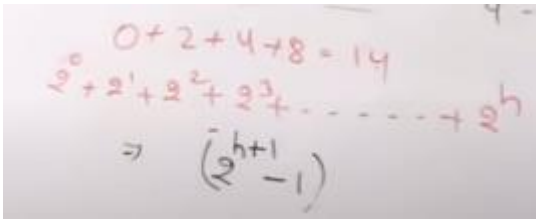


0, 1, 2 (jmh child)

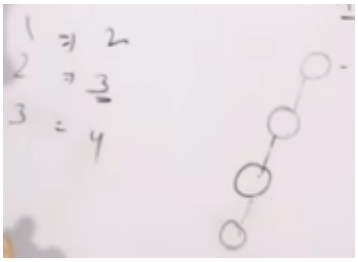




maksimum node tiap level

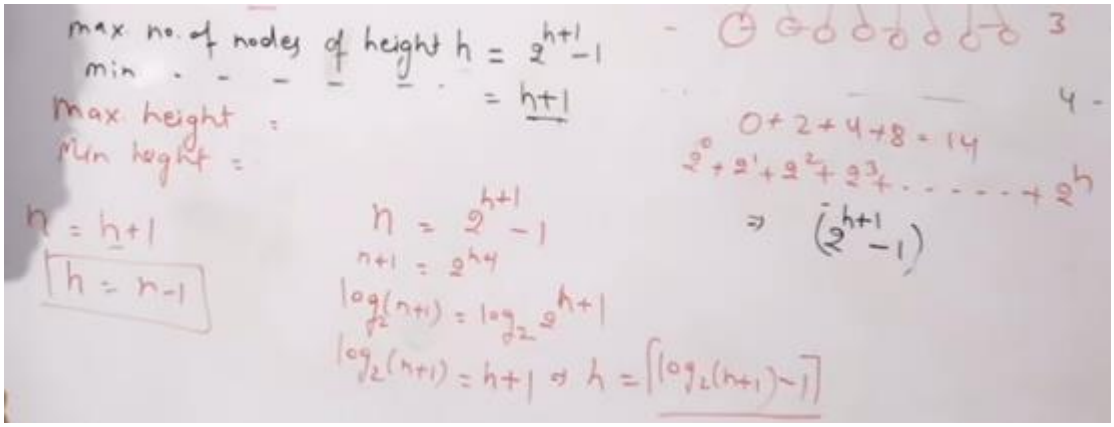


$2^n$, n = level.

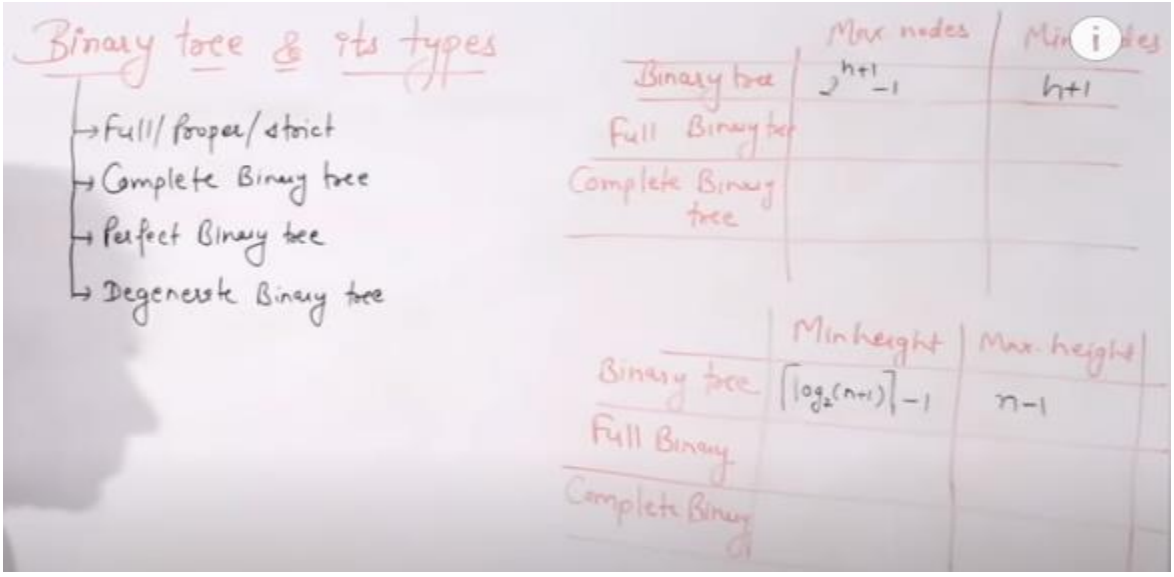


Utk binary tree, maks jmh node dari height = $2^{h+1}$-1, sdgkan min node nya = h+1. Cth jika hy level (height) 0 maka min node = 1 (root).

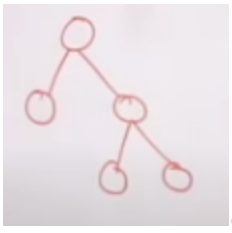
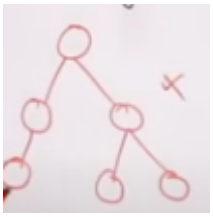
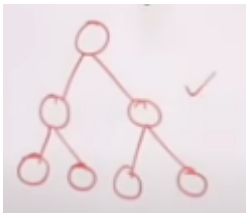
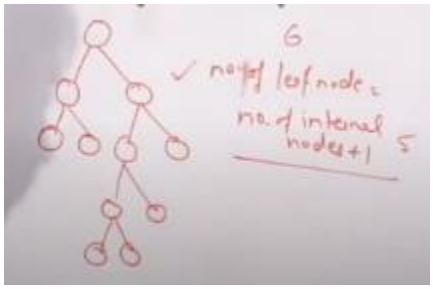



Max height (h) = n-1; min height = h=($\log_2$(n+1)-1).



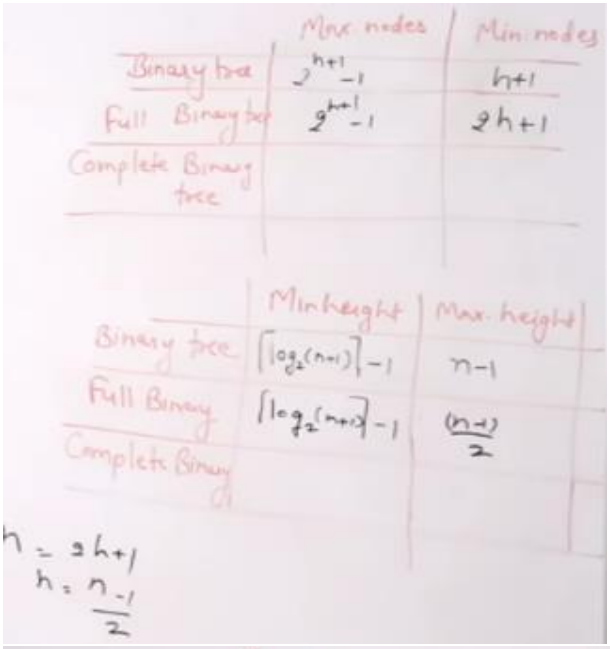

Full binary tree = each node have either 0 or 2 children.

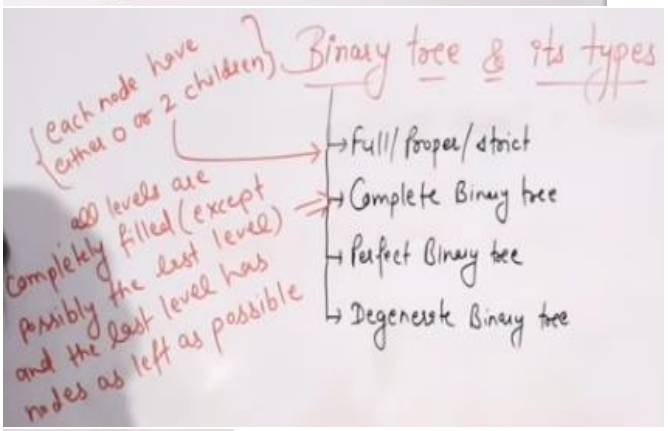

contoh





Number of leaf node = number of internal nodes + 1.

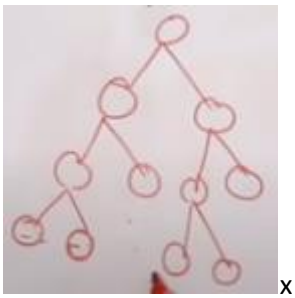
Internal node = node yg memiliki children.

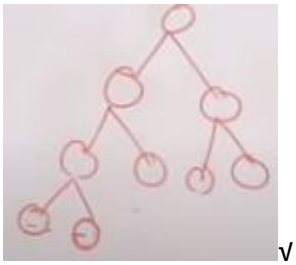
Min node of Height = 2h+1, cth height 3 maka min node = 7



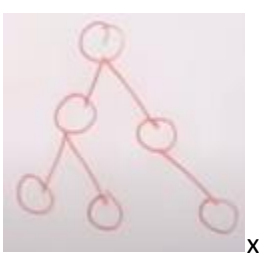
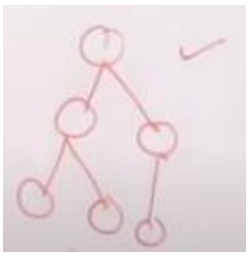




bukan complete binary tree krn level trakhir di bagian kiri kosong (cth gbr pertama).


x

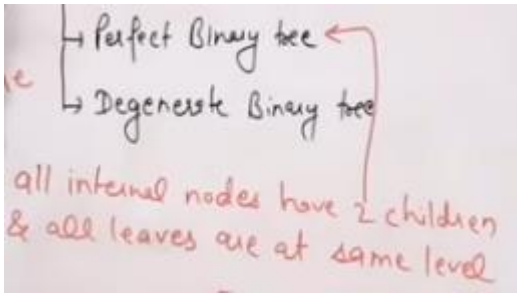

√

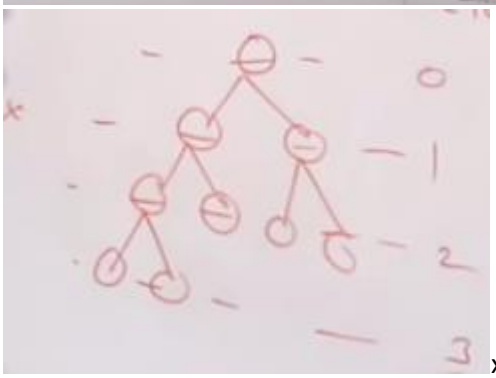

√

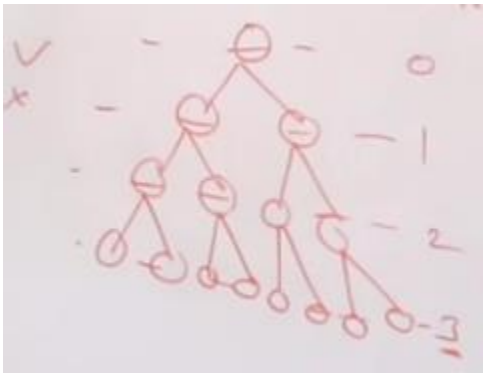

x


| | Max nodes | Min nodes |
|---|---|---|
| Binary tree | $2^{h+1}-1$ | $h+1$ |
| Full Binary tree | $2^{h+1}-1$ | $2h+1$ |
| Complete Binary tree | $2^{h+1}-1$ | $2^h$ |

| | Min height | Max. height |
|---|---|---|
| Binary tree | $\lceil \log_2(n+1) \rceil - 1$ | $n-1$ |
| Full Binary | $\lceil \log_2(n+1) \rceil - 1$ | $\frac{(n-1)}{2}$ |
| Complete Binary | $\lceil \log_2(n+1) \rceil - 1$ | $\log n$ |

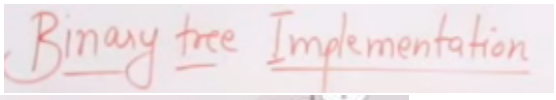


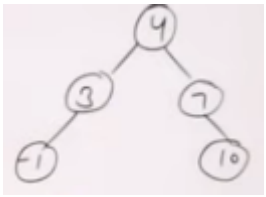


x bukan perfect binary tree

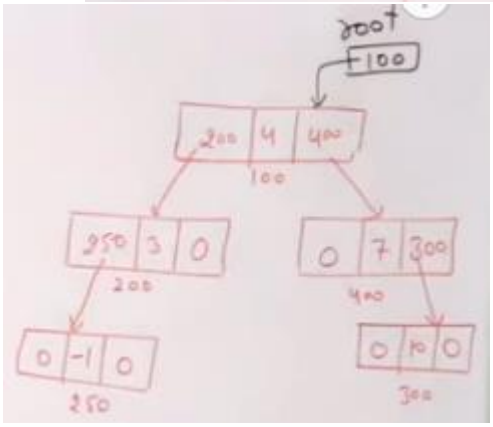

√ perfect binary tree = CBT & FBT

Degenerate binary tree = jika internal node hanya memiliki 1 child

left skew binary tree, right skew binary tree

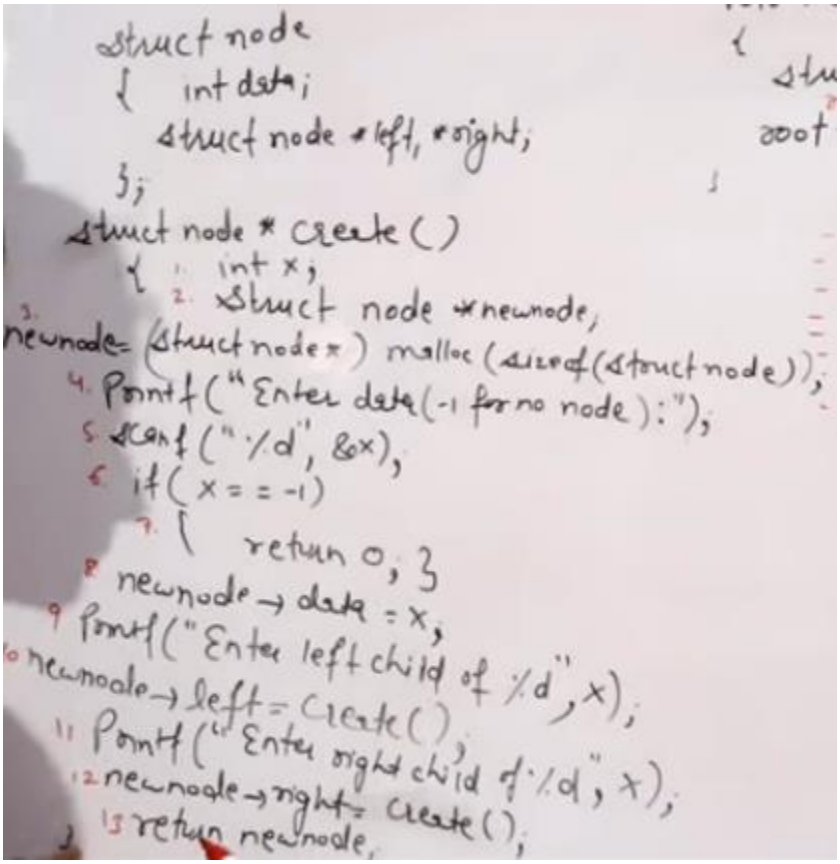



100 = address node of root/ sbg pointer

```
void main()
{
    struct node * root;
    root = 0;
    root = Create();
}
```



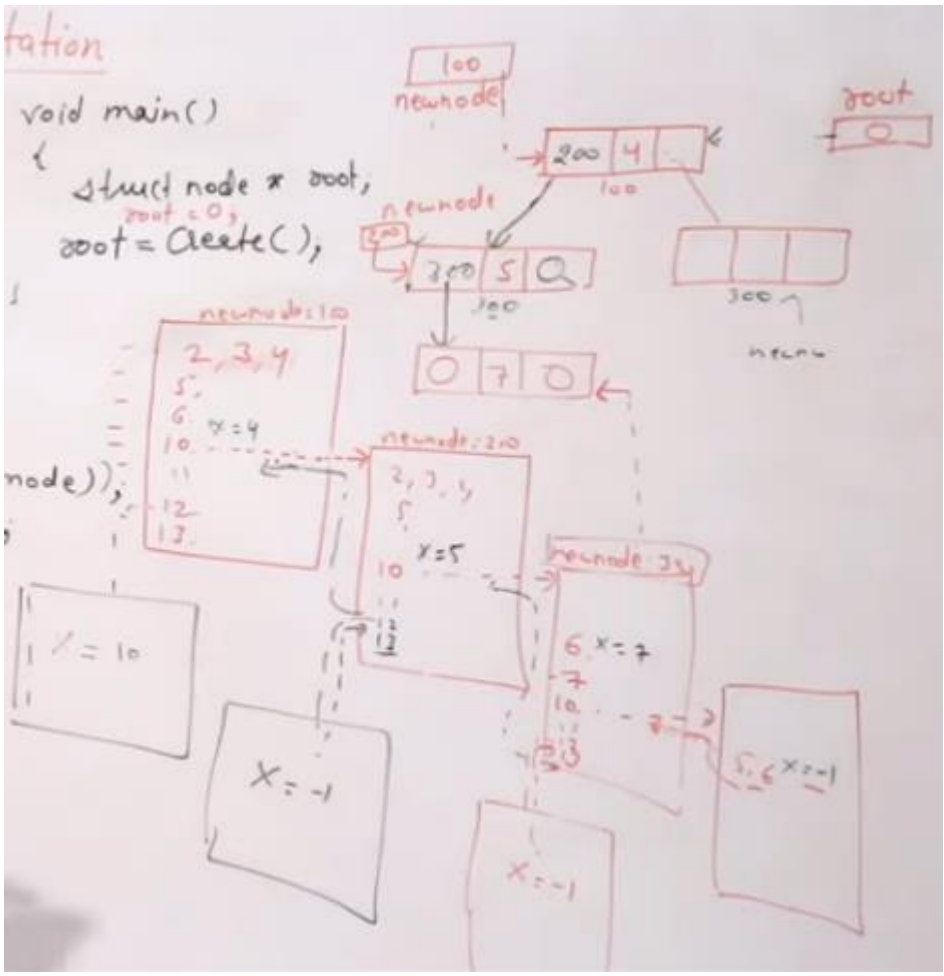

Jika leaf node maka x=-1 artinya return 0 (kembali/back to) dr left ke right node.

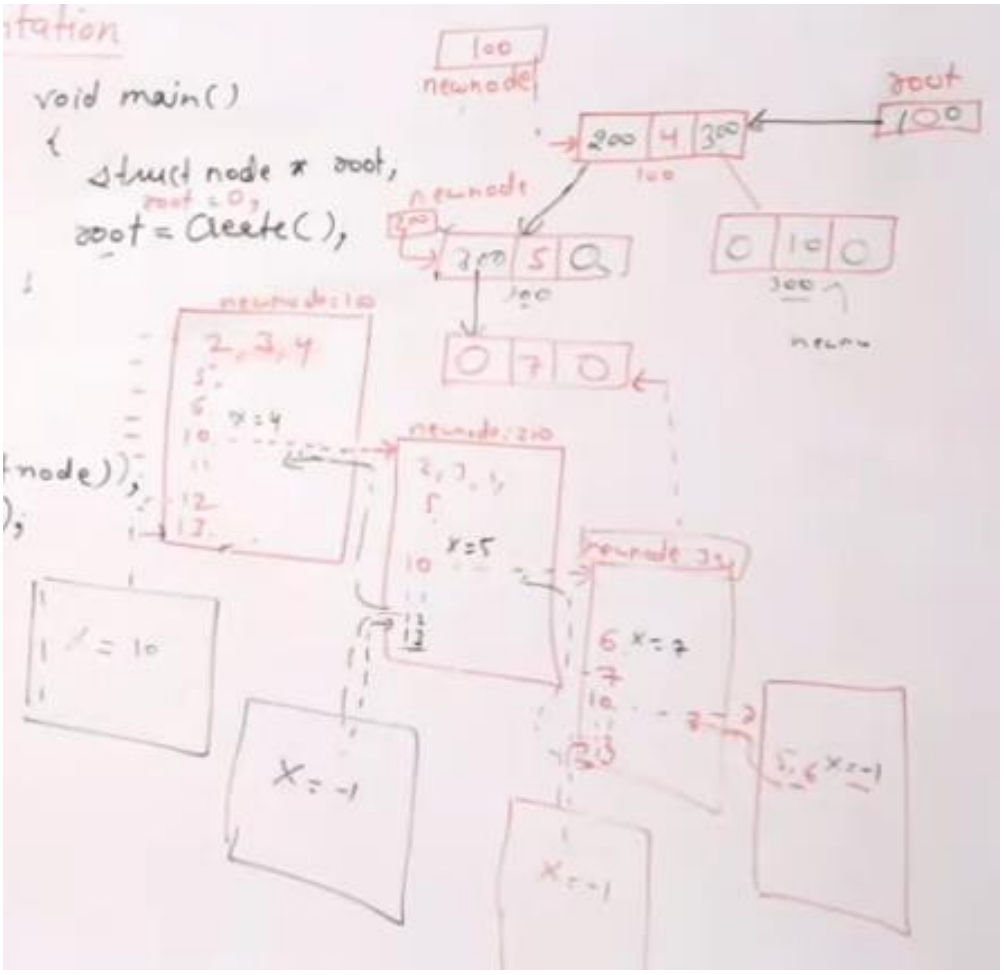
Brktnya create node (left child) dulu secara rekursif (memanggil fungsi dasar/base).

Utk root node, x = 4

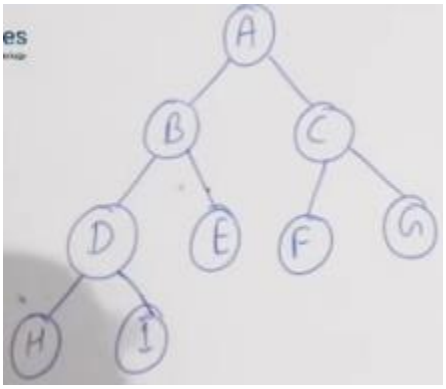
Pada baris 10, ada fungsi create function maka rekursif dijalankn.

Inisialisasi awal root = 0; pointer root = 100



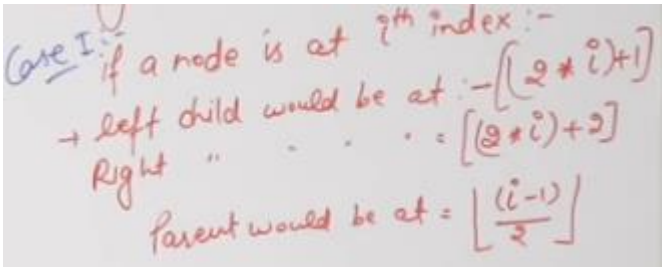


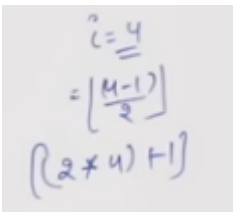

# Array Representation of Binary tree (Sequential Representation)



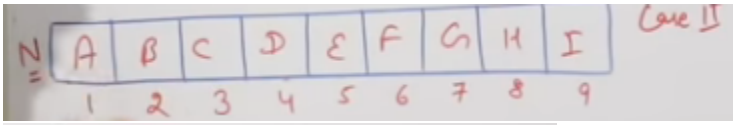

Utk mnulis dlm array, dmulai dari level 0, kmdn dr left node ke right node.

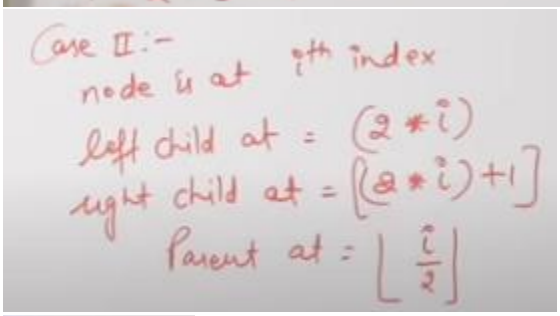

Dlm array tsb kita tdk bs mlihat node parent & child. Maka, utk case 1:

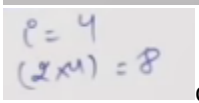


lokasi parent = floor value dr rumus disamping. 

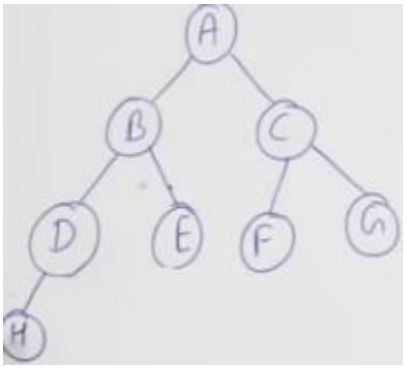
misal E = i = 4 (indeks),kita cek parent&childnya

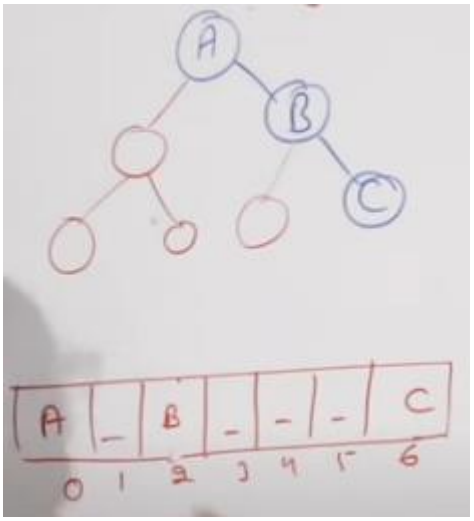


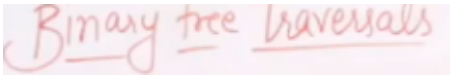



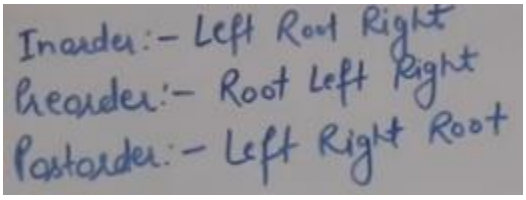
cth D, i = 4 sbg complete binary tree



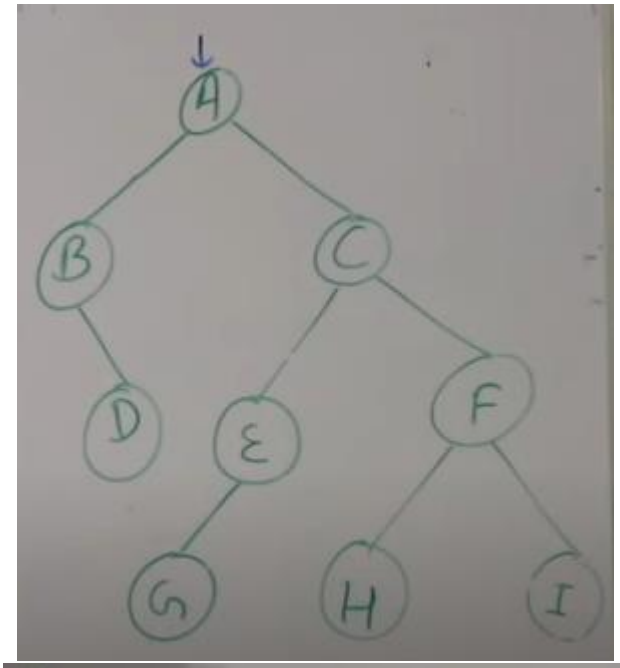
CBT 

Hrs CBT = Complete Binary Tree (ingat definisi CBT)



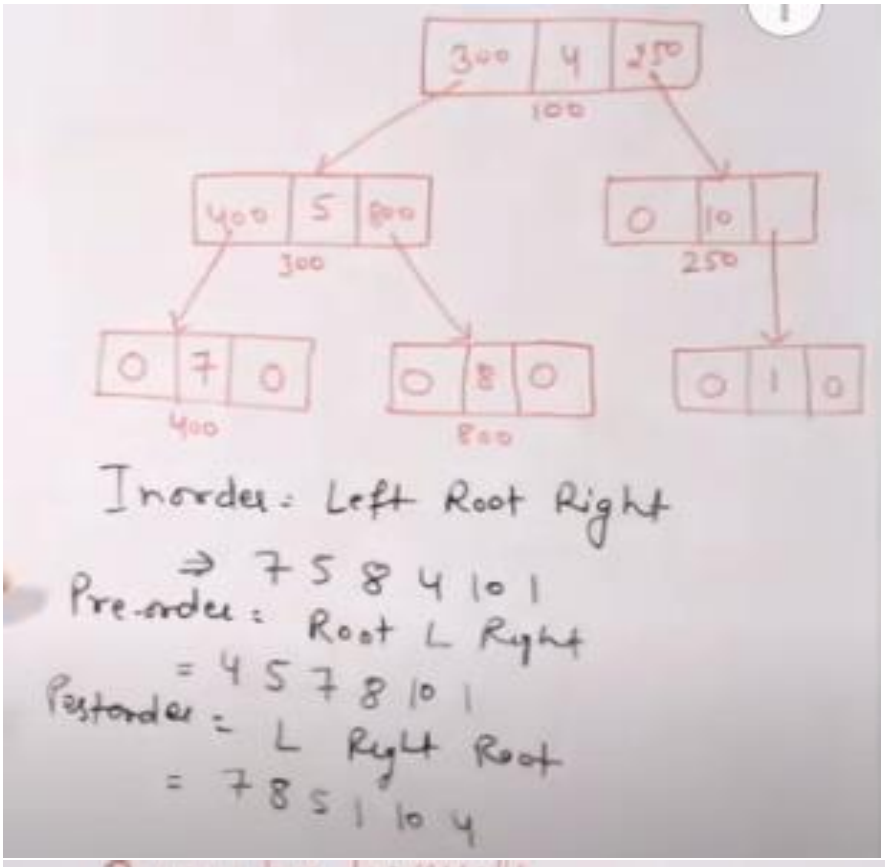



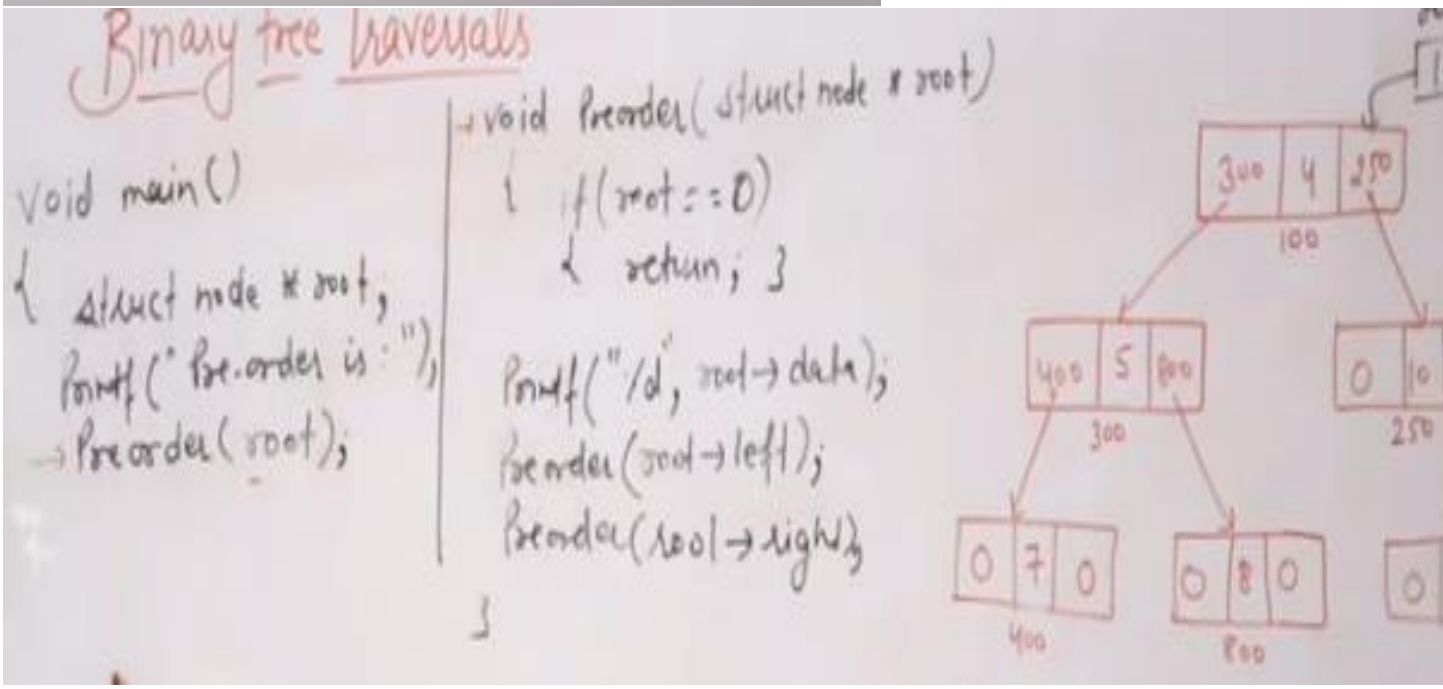
Preorder:- A B D C E G F H I



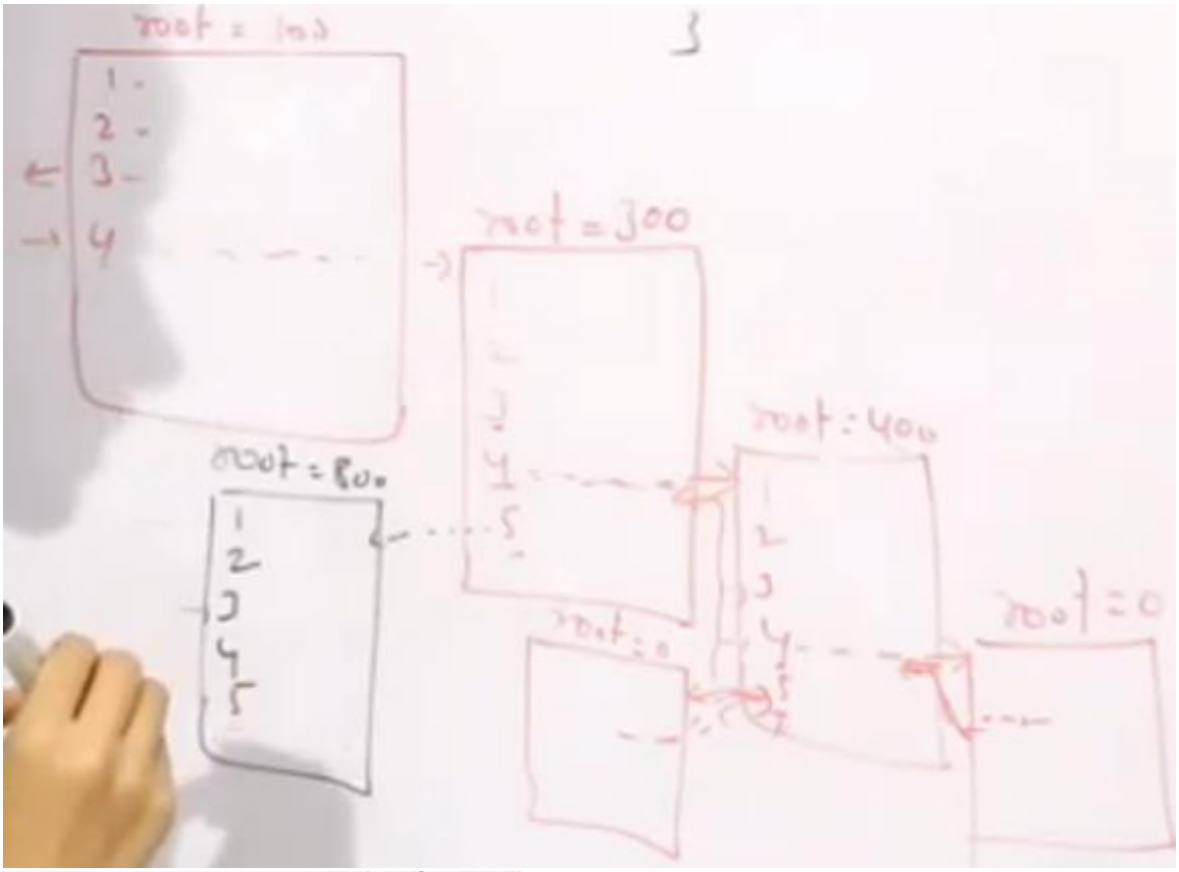

Postorder:- D B G E H I F C A

Inorder:- B D A G E C H F I

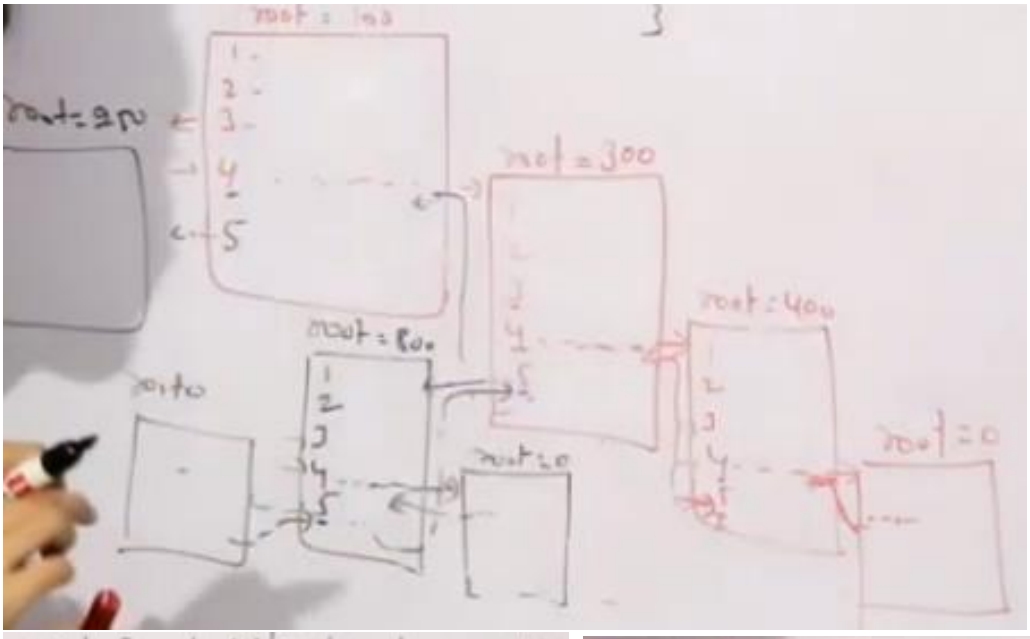



Termination condition= kondisi ketika pada leaf node (bernilai 0) / return.




```
void Preorder(struct node * root)
{
1  if(root==0)
2  { return; }
3  Printf("%d", root->data);
4  Preorder(root->left);
5  Preorder(root->right);
}
```




```
void Inorder(struct node * root)
{  if(root = 0)
      2  .  3
   Inorder(root->left);
   Printf("%d", root->data);
   Inorder(root->right);
}
```

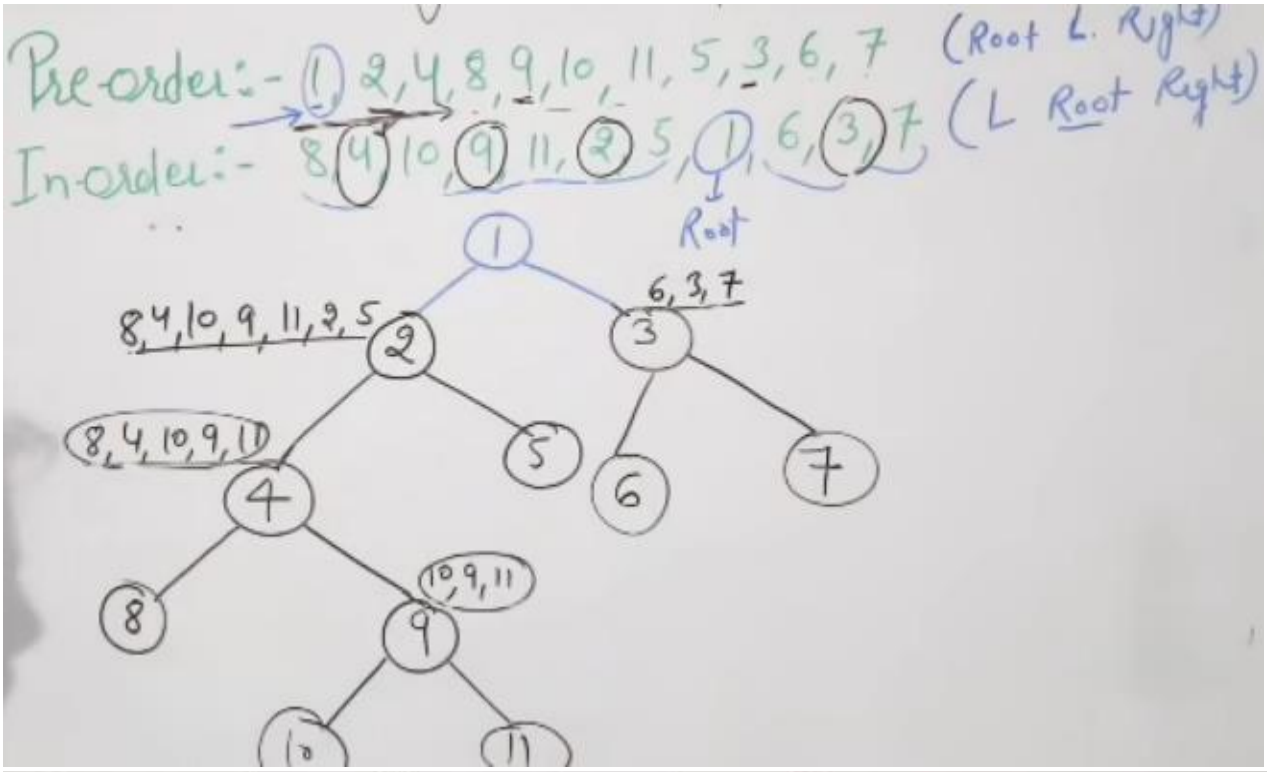
```
void Postorder( -  -  )  {
{  -

   Postorder(root->left)
   Postorder(root->right);  3
   Printf("%d", root->data);
3
```

# Construct a Binary tree from Preorder & Inorder

Preorder:- 1, 2, 4, 8, 9, 10, 11, 5, 3, 6, 7 (Root L. Right)

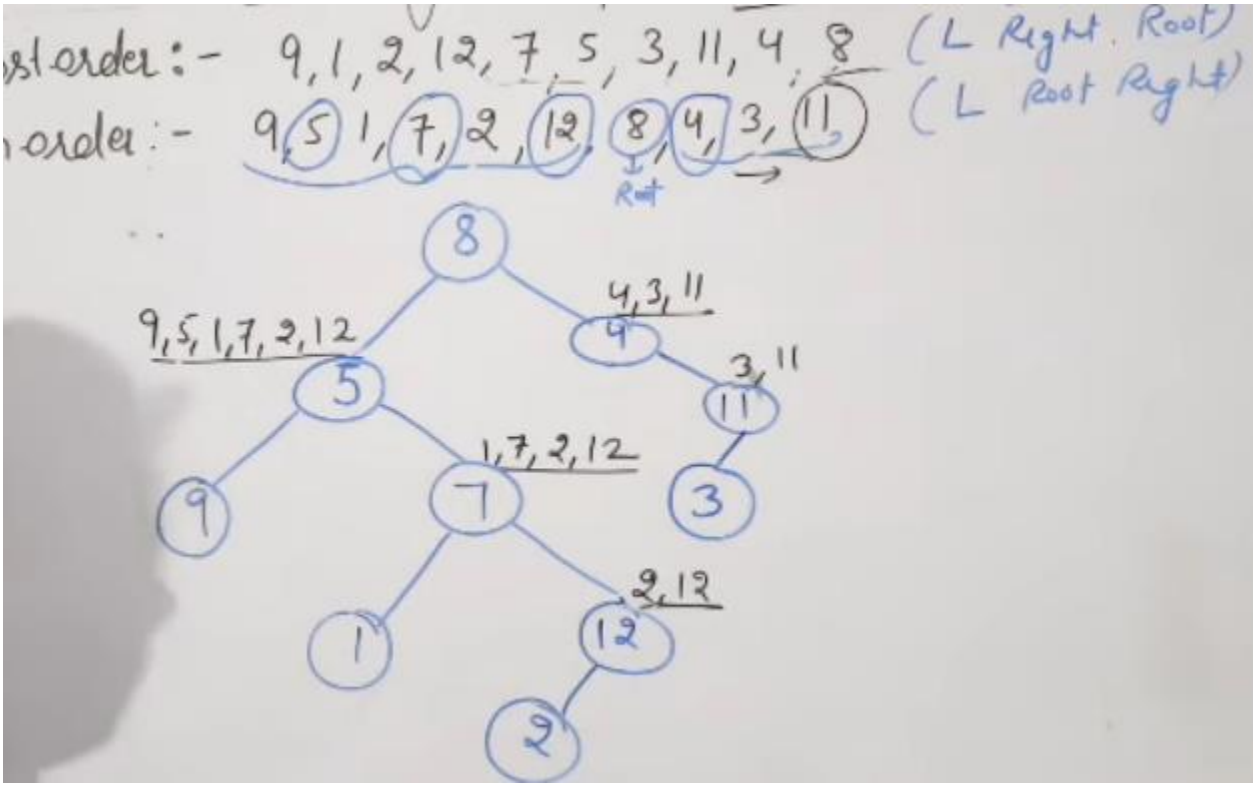
Inorder:- 8, 4, 10, 9, 11, 2, 5, 1, 6, 3, 7 (L Root Right)

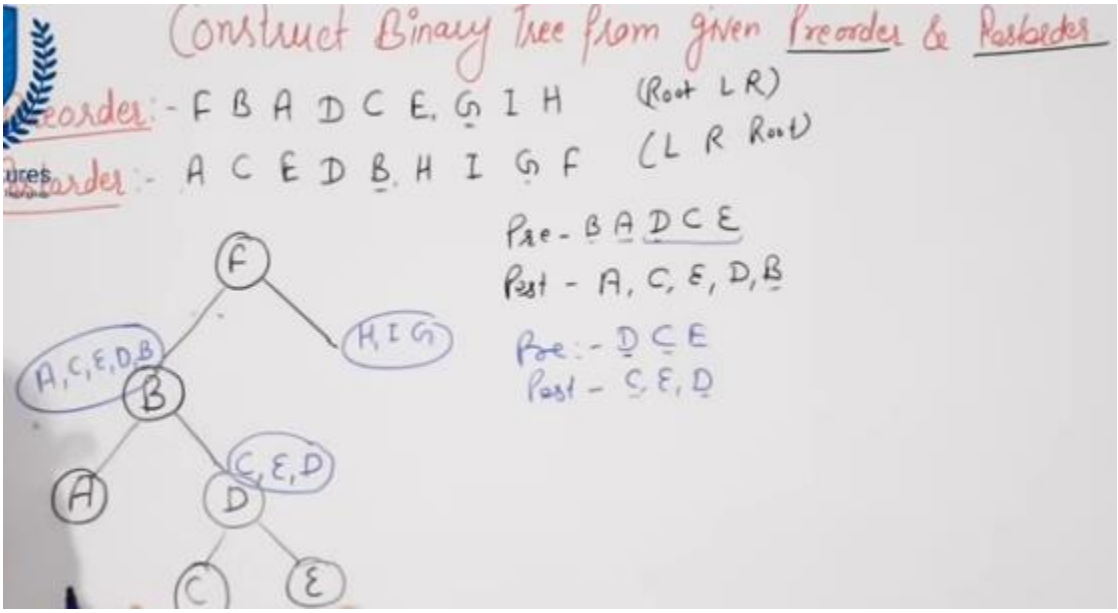


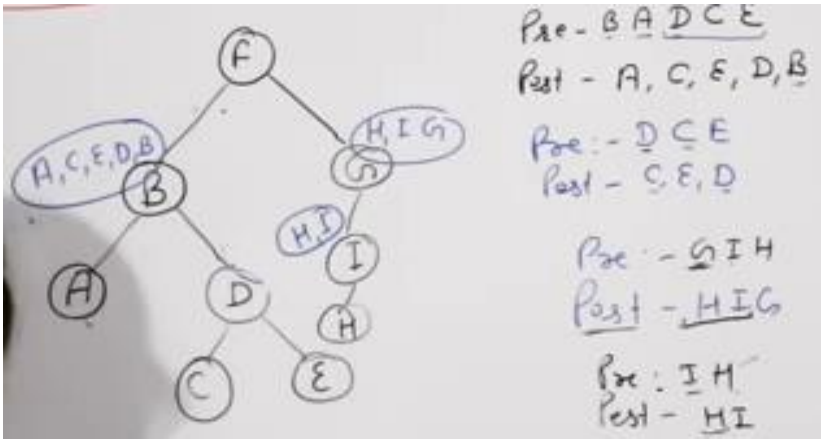

Construct a Binary tree from Post-order & Inorder

Postorder:- 9, 1, 2, 12, 7, 5, 3, 11, 4, 8 (L Right. Root)

Inorder:- 9, 5, 1, 7, 2, 12, 8, 4, 3, 11 (L Root Right)


storder:- 9, 1, 2, 12, 7, 5, 3, 11, 4, 8 (L Right Root)
order:- 9, 5, 1, 7, 2, 12, 8, 4, 3, 11 (L Root Right)
Root
8
9,5,1,7,2,12
4,3,11
5
4
3,11
11
1,7,2,12
9
7
3
2,12
1
12
2


Construct Binary Tree from given Preorder & Postorder
Preorder:- F B A D C E G I H (Root L R)
Postorder:- A C E D B H I G F (L R Root)
Pre - B A D C E
Post - A, C, E, D, B
Pre :- D C E
Post - C, E, D
F
A,C,E,D,B
H,I G
B
A
C,E,D
D
C
E

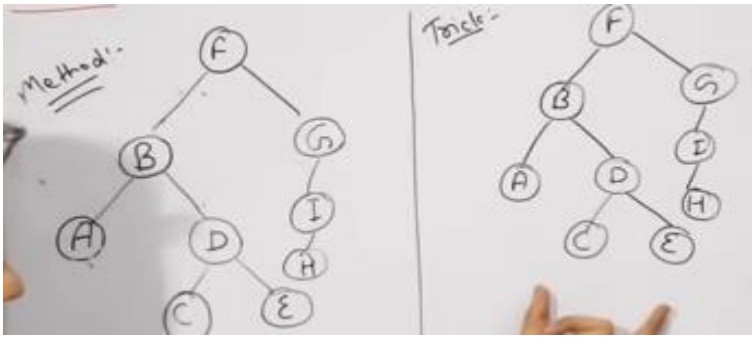



Pre - B A D C E
Post - A, C, E, D, B

Pre : - D C E
Post - C, E, D

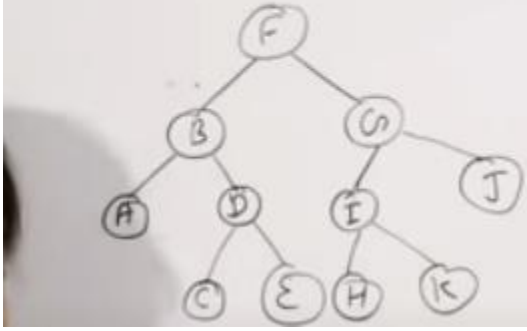
Pre - G I H
Post - H I G

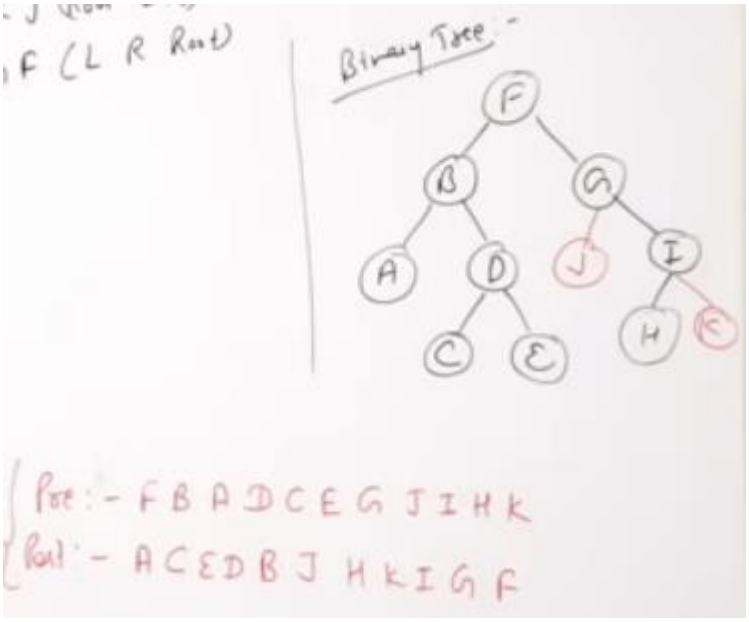
Pre : I H
Post - H I




eorder - F B A D C E G I H K J (Root L R)

arder - A C E D B H K I J G F (L R Root)





Full Binary Tree