Tree

ARNA FARIZA

YULIANA SETIOWATI

Capaian Pembelajaran

Mahasiswa mengerti konsep tree.

Mahasiswa dapat mengimplementasikan tree dalam bahasa pemrograman.

Mahasiswa dapat mengimplementasikan algoritma pembentukan tree.

Mahasiswa dapat mengimplementasikan algoritma penelusuran tree yaitu preorder, inorder dan postorder.

Materi

Tree

Binary Tree

Binary Search Tree

Metode Traversal

- Inorder
- Preorder
- Postorder

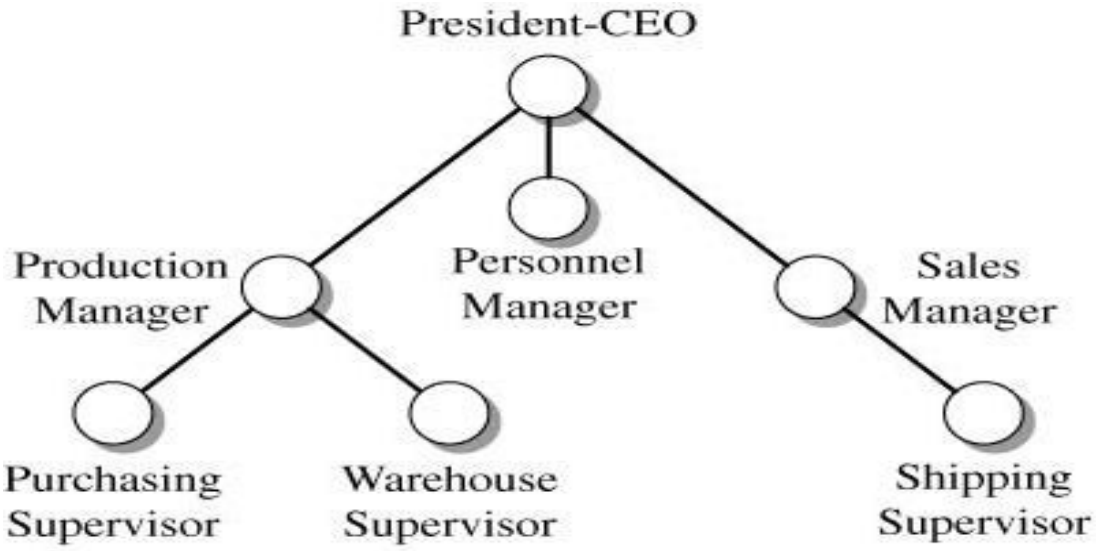
Tree

Struktur pohon (tree) biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada. Contoh penggunaan struktur pohon:

- Silsilah keluarga
- Hasil pertandingan yang berbentuk turnamen
- Struktur organisasi dari sebuah perusahaan

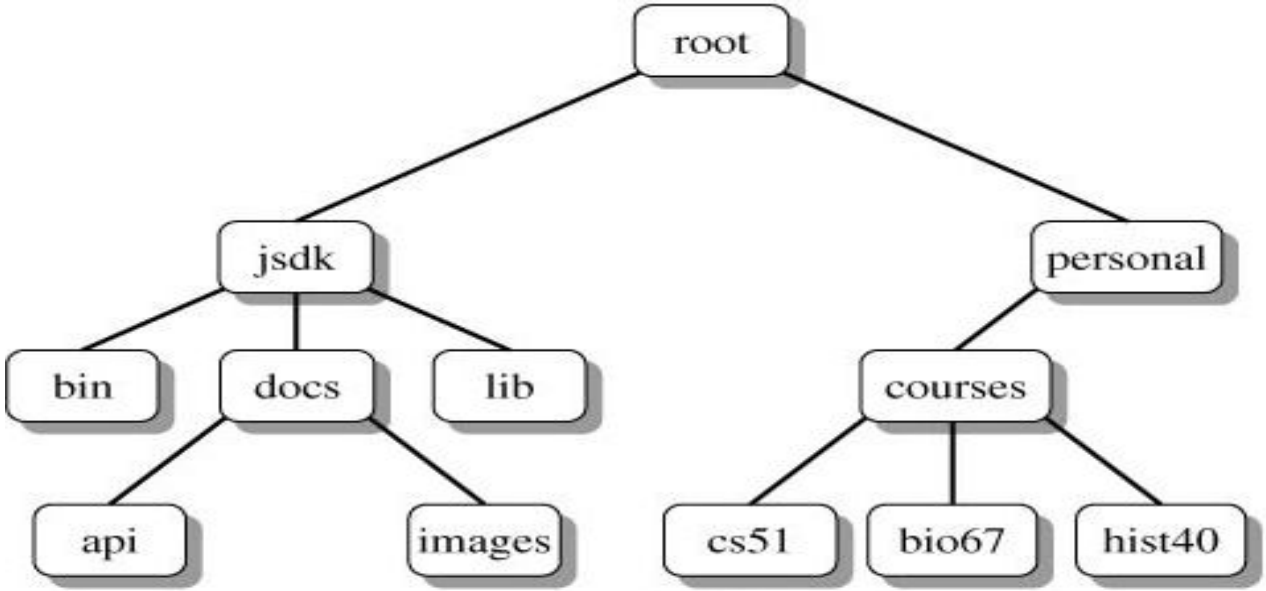
Contoh Tree

- Struktur organisasi dari sebuah perusahaan



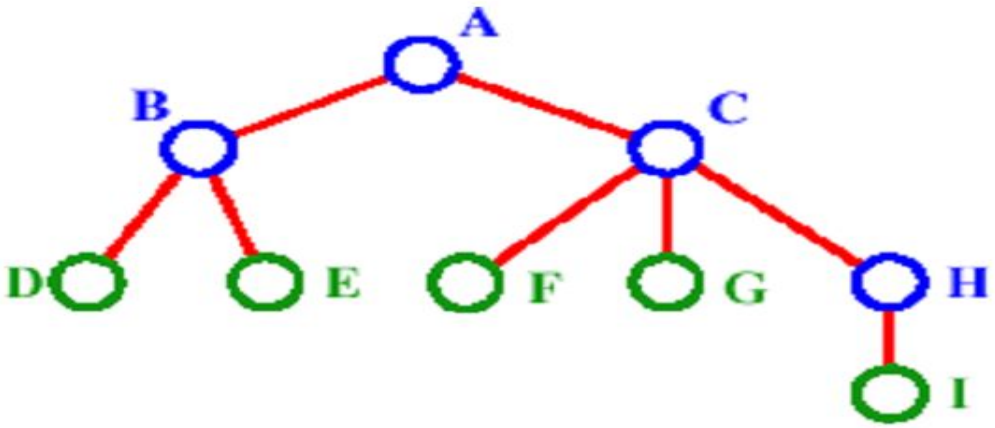
Contoh Tree

Sistem operasi menggunakan tree untuk struktur file



Path name: /jsdk/docs/api

Istilah Umum di Tree



A adalah **root** dari Tree

B adalah **parent** dari D dan E

C adalah **sibling** dari B

D dan E adalah **children**/anak dari B

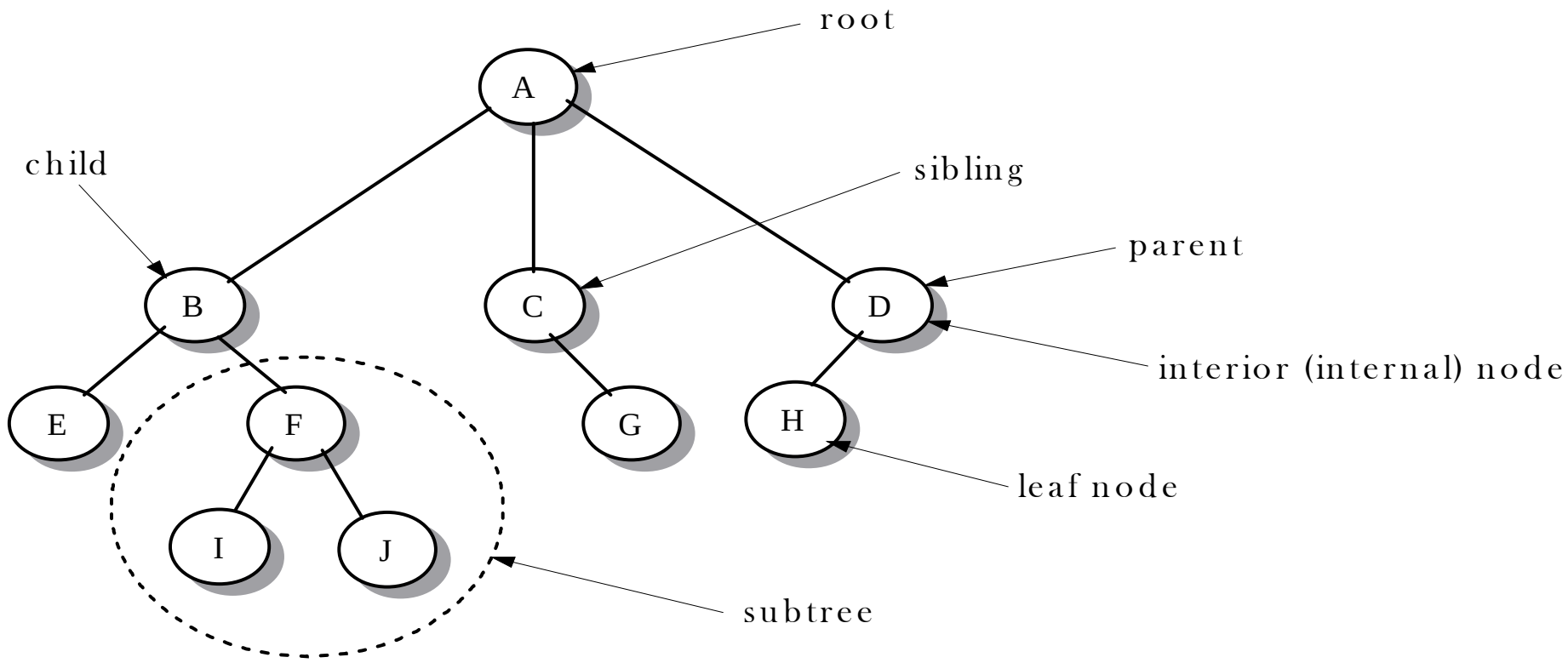
D, E, F, G, I adalah **external nodes** atau **leaf**

A, B, C, H adalah **internal nodes**

Tinggi/**height** dari tree adalah 3

B,D,E adalah **subtree**

Istilah Umum di Tree



Binary Tree

Sebuah pengorganisasian secara hirarki dari beberapa buah simpul, dimana masing-masing simpul tidak mempunyai anak lebih dari 2.

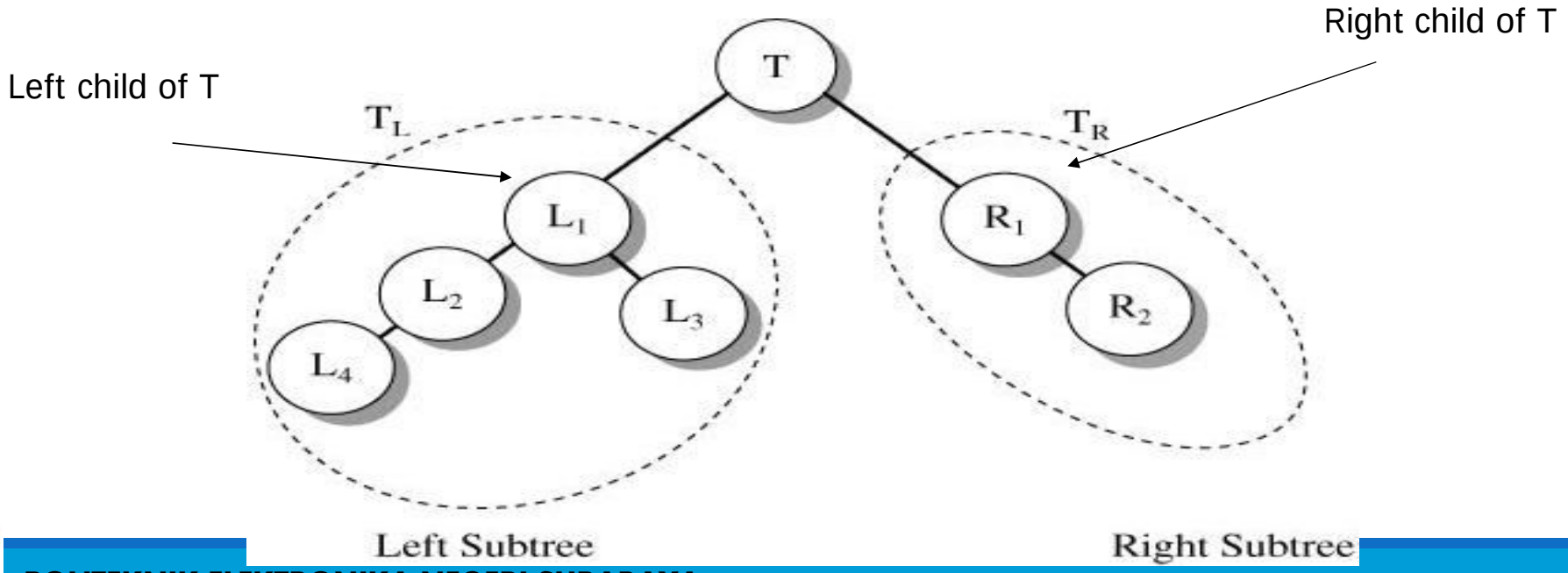
Simpul yang berada di bawah sebuah simpul dinamakan anak dari simpul tersebut.

Simpul yang berada di atas sebuah simpul dinamakan induk dari simpul tersebut.

Binary Tree

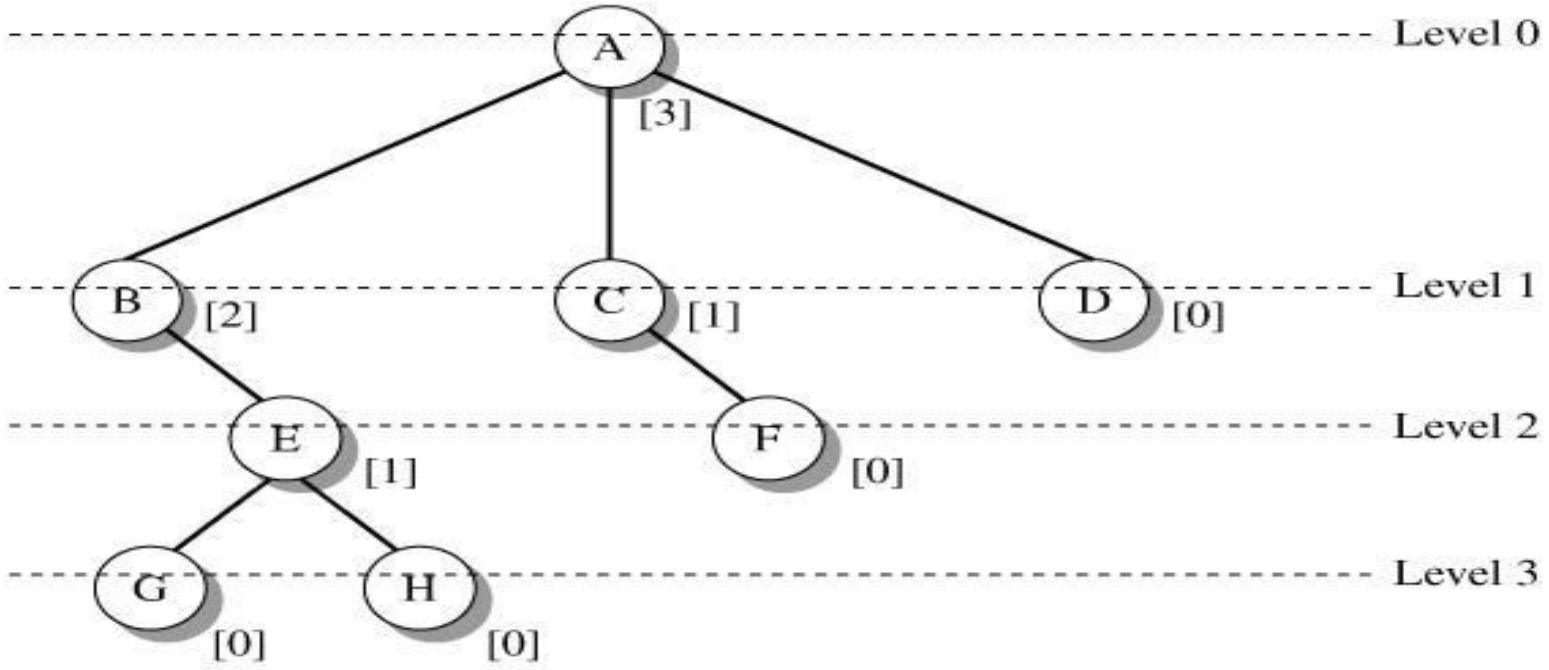
Tiap node pada binary tree adalah **subtree kiri** dan **subtree kanan**.

Setiap **Subtree** adalah juga **tree**.



Level

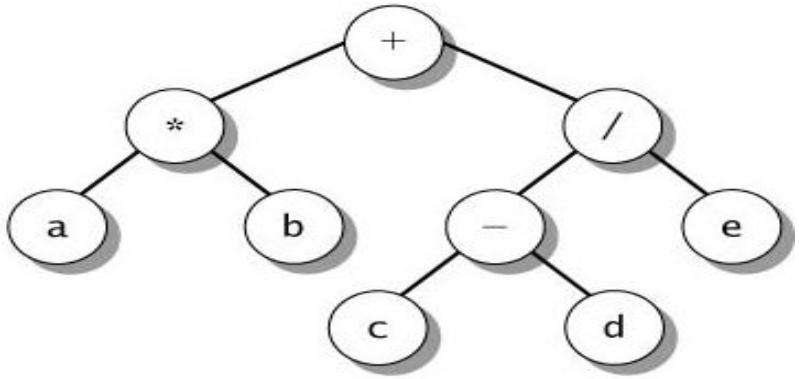
Tree dengan level 3



Tree illustrates the level and height of a node.

Binary Tree

Setiap node dalam Tree mempunyai maksimum dua anak

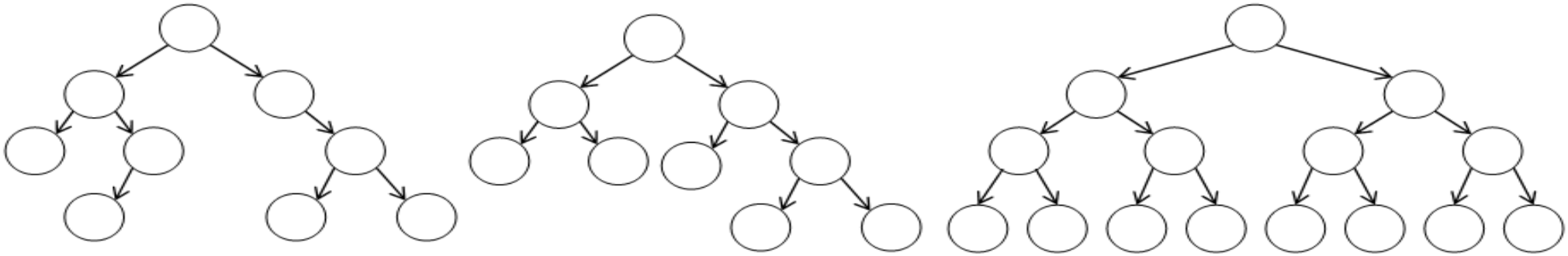


Expression: a* b + (c – d)/e

Binary Tree

Binary tree adalah tree di mana setiap nodes memiliki maksimum 2 anak

Full Binary tree adalah binary tree di mana setiap nodes memiliki anak 0 atau anak 2.



Full Binary Tree

(# external nodes) = (# internal nodes) + 1

(# nodes at level i) $\leq 2^i$

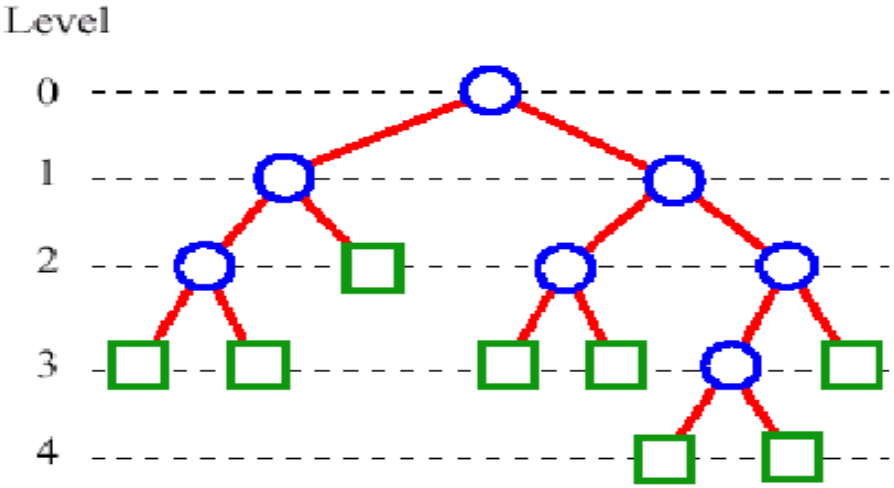
(# external nodes) $\leq 2^{(height)}$

(height) $\geq \log_2$(# external nodes)

(height) $\geq \log_2$(# nodes) - 1

(height) $\leq$ (# internal nodes) = ((# nodes) - 1)/2

Jika tinggi = k, maka #node = 2^{k+1}- 1

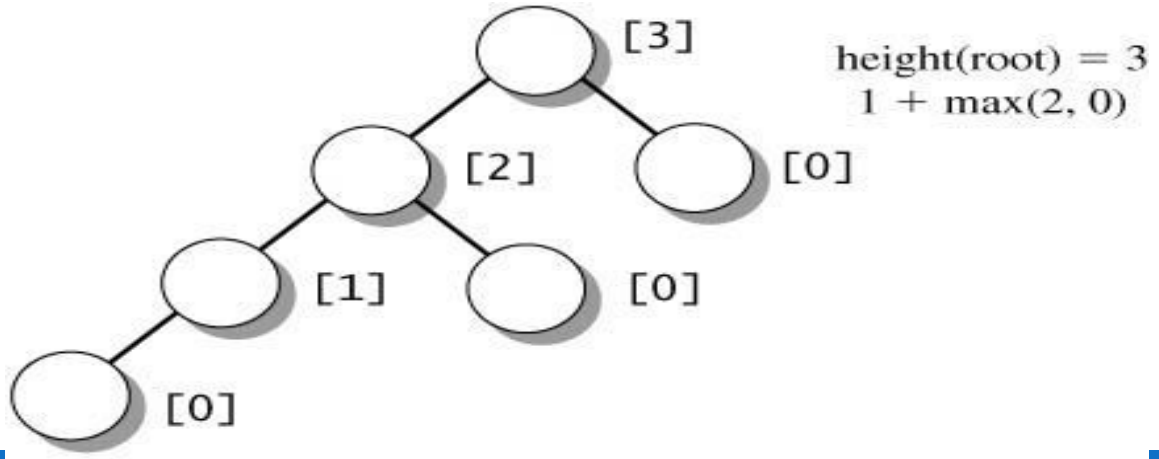


Height dari Binary Tree

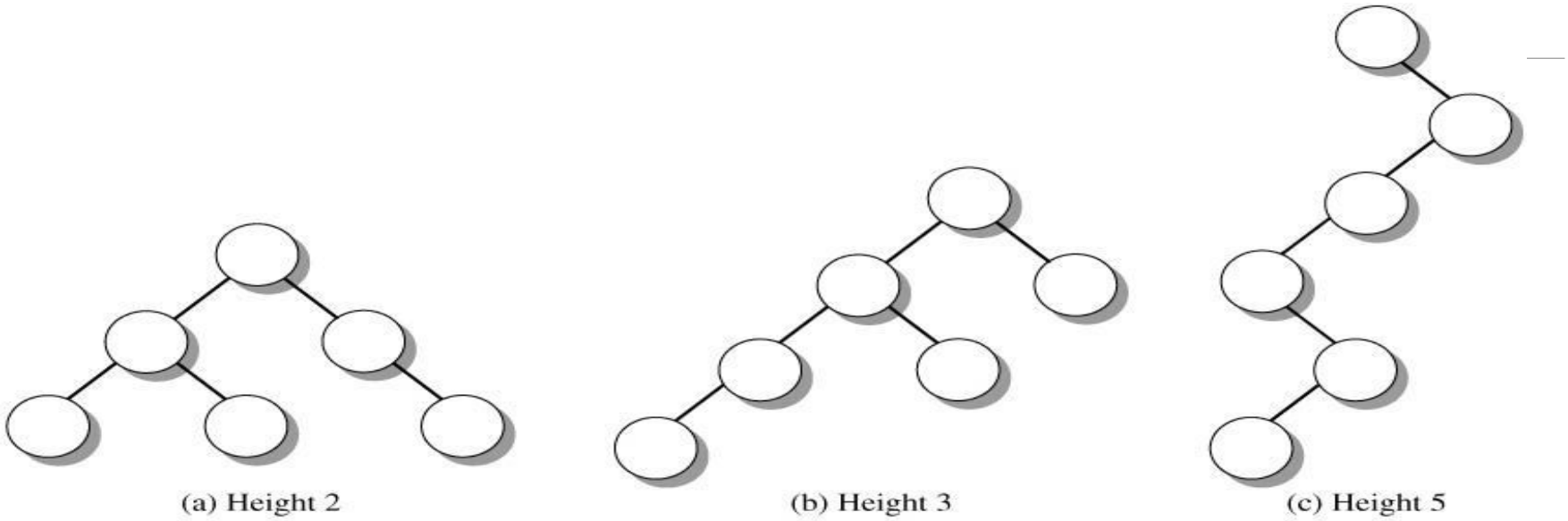
Height/kedalaman tree adalah maksimum level dari tree.

Misal T_N adalah subtree dengan root N dan T_L adalah root subtree kiri dan T_R adalah root dan subtree kanan.

$$height(N) = height(T_N) = \begin{cases} -1 & \text{if } T_N \text{ is empty} \\ 1+\max(\, height(T_L), height(T_R)) & \text{if } T_N \text{ not empty} \end{cases}$$



Height dari Binary Tree



(a) Height 2

(b) Height 3

(c) Height 5

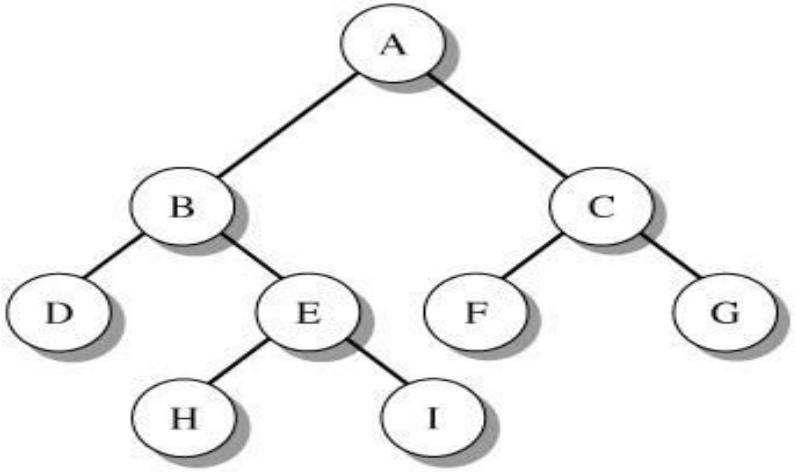
Densitas Binary Tree

Jumlah node di tiap level berada pada range tertentu.

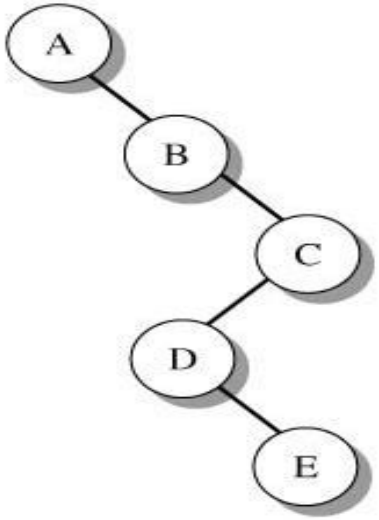
- Level 0, terdapat 1 node yaitu root.
- Level 1, mempunyai 1 atau 2 node
- Level k, jumlah node antara 1 to 2^k

- **Densitas** adalah besar/size tree berdasarkan jumlah node relatif terhadap tinggi/height tree.

Densitas Binary Tree



Tree A
Size 9 Height 3



Tree B
Size 5 Height 4

Binary Search Tree

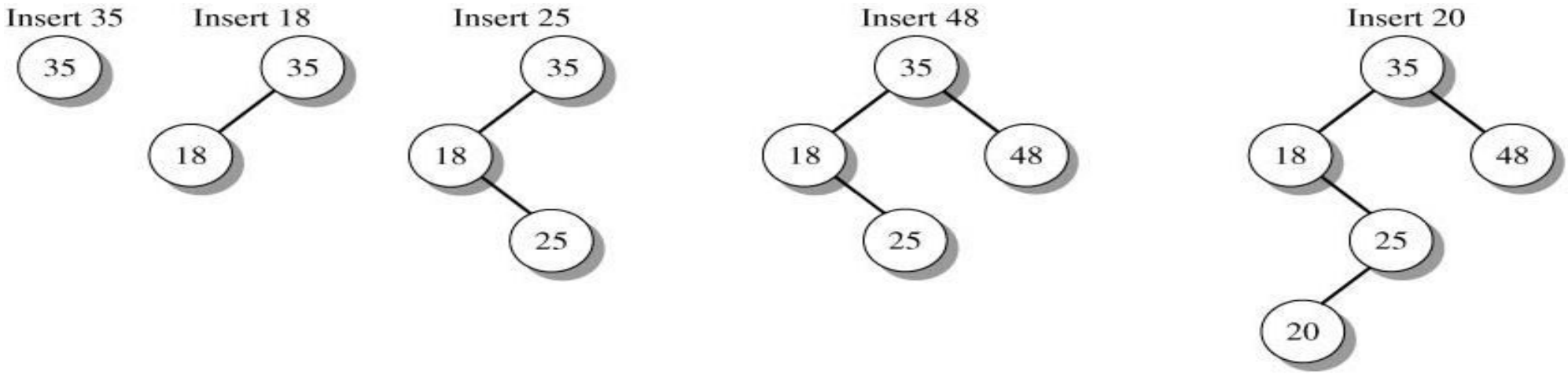
Sebuah node di Binary Search Tree memiliki path yang unik dari root menurut aturan ordering

- Sebuah Node, mempunyai subtree kiri yang memiliki nilai lebih kecil dari node tsb dan subtree kanan memiliki nilai lebih besar dari node tsb.
- Tidak diperbolehkan ada node yang memiliki nilai yang sama.

Membangun Binary Search Tree

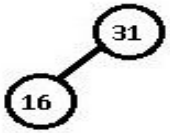
1. Jika value dari node baru sama dengan value dari current node, maka mengembalikan nilai false.
2. Jika value dari node baru kurang dari value dari current node maka :
   1) Jika anak kiri current node tidak null, maka ubah current node ke anak kiri tersebut, lakukan langkah 1.
   2) Jika anak kiri current node adalah null, maka tambahkan node baru tersebut sebagai anak kiri dari current node
3. Jika value dari node baru lebih besar dari value dari current node maka :
   1. Jika anak kanan current node tidak null, maka ubah current node ke anak kanan tersebut, lakukan langkah 1.
   2. Jika anak kanan current node adalah null, maka tambahkan node baru tersebut sebagai anak kanan dari current node

Membangun Binary Search Tree

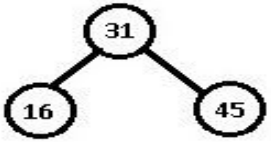


Membangun Binary Search Tree

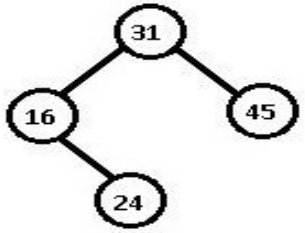
Insert 31



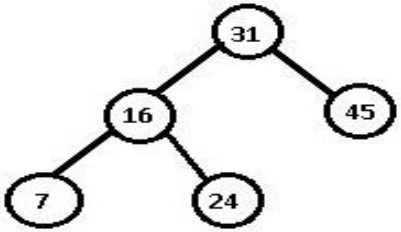
Insert 16



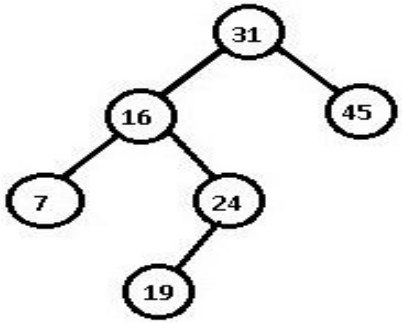
Insert 45



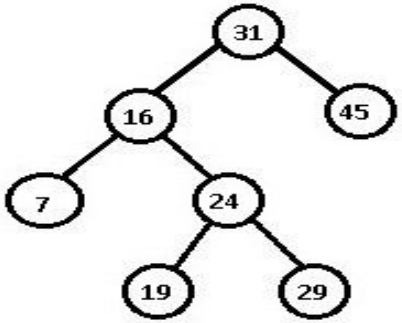
Insert 24



Insert 7

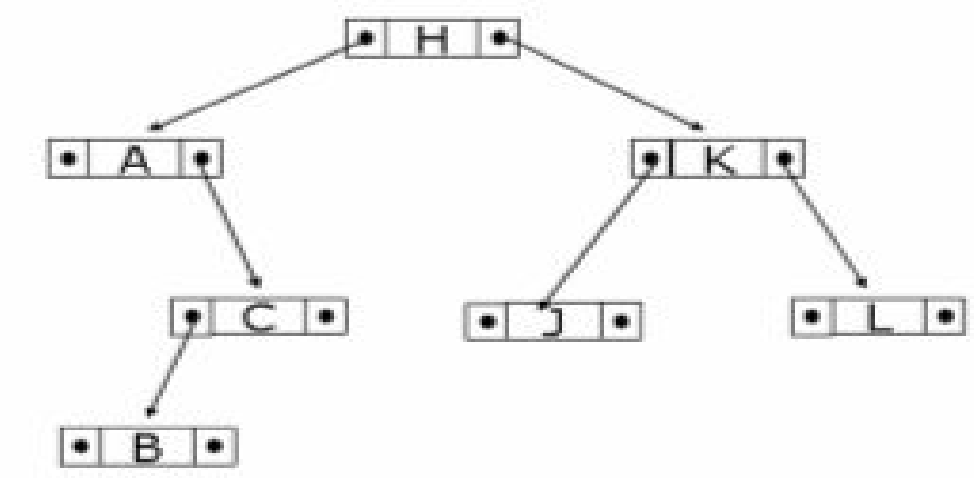


Insert 19



Insert 29

Pembentukan Tree



Metode Traversal

Salah satu operasi yang paling umum dilakukan terhadap sebuah tree adalah kunjungan (traversing)

Sebuah kunjungan berawal dari root, mengunjungi setiap node dalam tree tersebut tepat hanya sekali

- *Mengunjungi artinya memproses data/info yang ada pada node ybs*

Kunjungan bisa dilakukan dengan 3 cara:

1. Preorder
2. Inorder
3. Postorder

Ketiga macam kunjungan tersebut bisa dilakukan secara rekursif dan non rekursif

Preorder

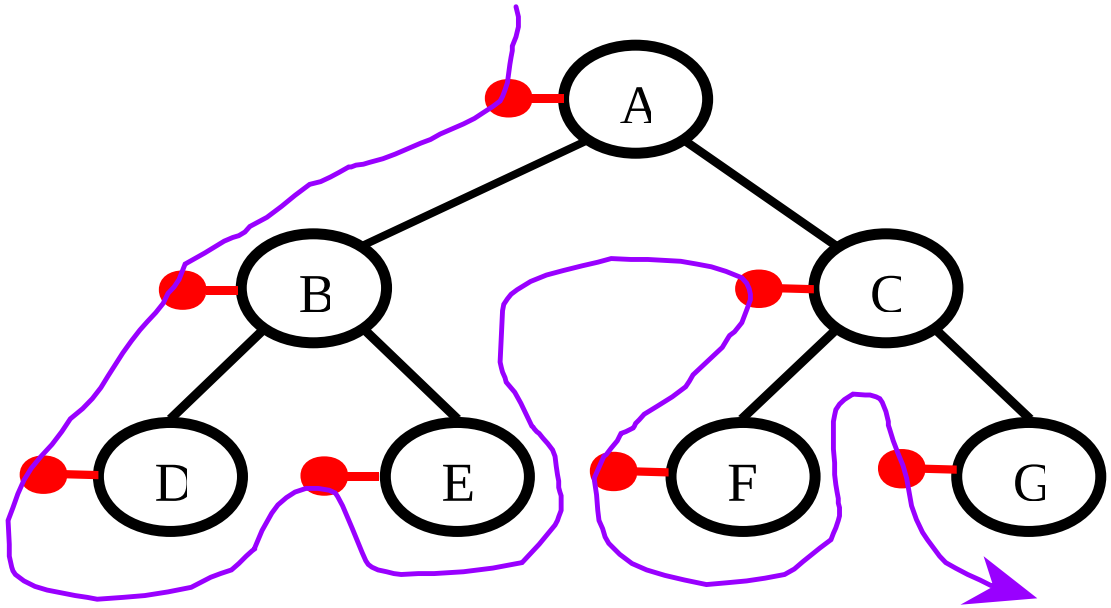
Kunjungan preorder, juga disebut dengan *depth first order*, menggunakan urutan:

- Cetak isi simpul yang dikunjungi
- Kunjungi cabang kiri
- Kunjungi cabang kanan

Preorder

```
void preorder(pohon ph)
{
  if (ph != NULL)
  {
        printf("%c ", ph->info);
        preorder(ph->kiri);
        preorder(ph->kanan);
  }
}
```

Preorder



A B D E C F G

Inorder

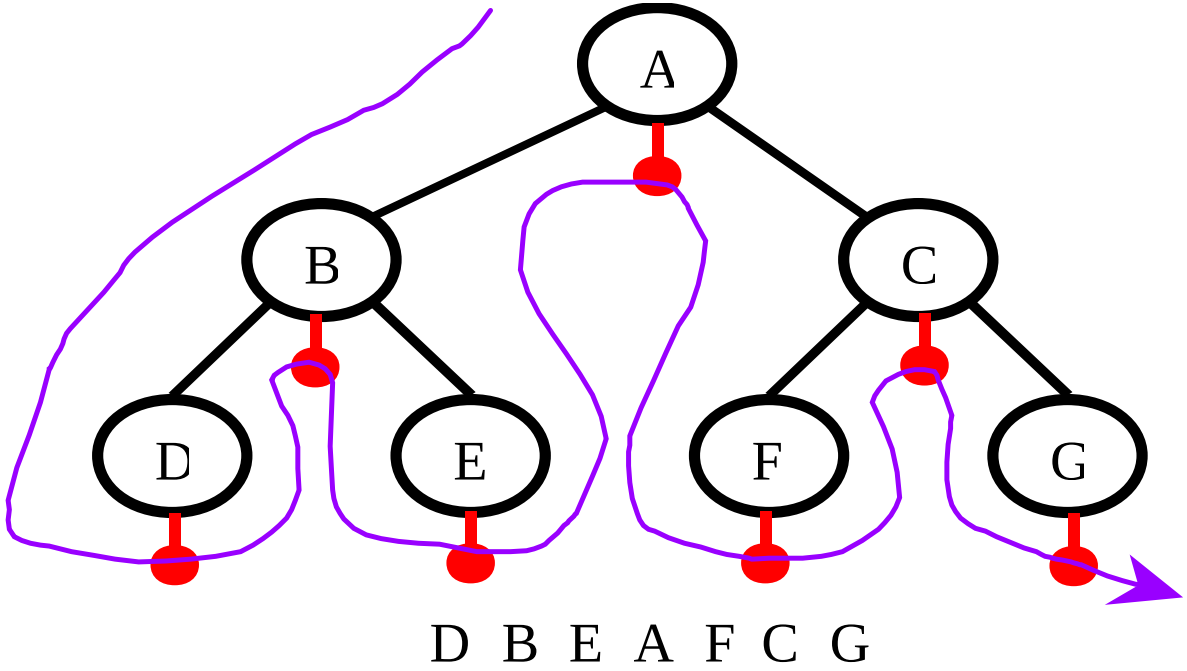
Kunjungan secara inorder, juga sering disebut dengan *symmetric order*, menggunakan urutan:

- Kunjungi cabang kiri
- Cetak isi simpul yang dikunjungi
- Kunjungi cabang kanan

Inorder

```
void inorder(pohon ph)
{
 if (ph != NULL)
 {
        inorder(ph->kiri);
        printf("%c ", ph->info);
        inorder(ph->kanan);
 }
}
```

Inorder



Postorder

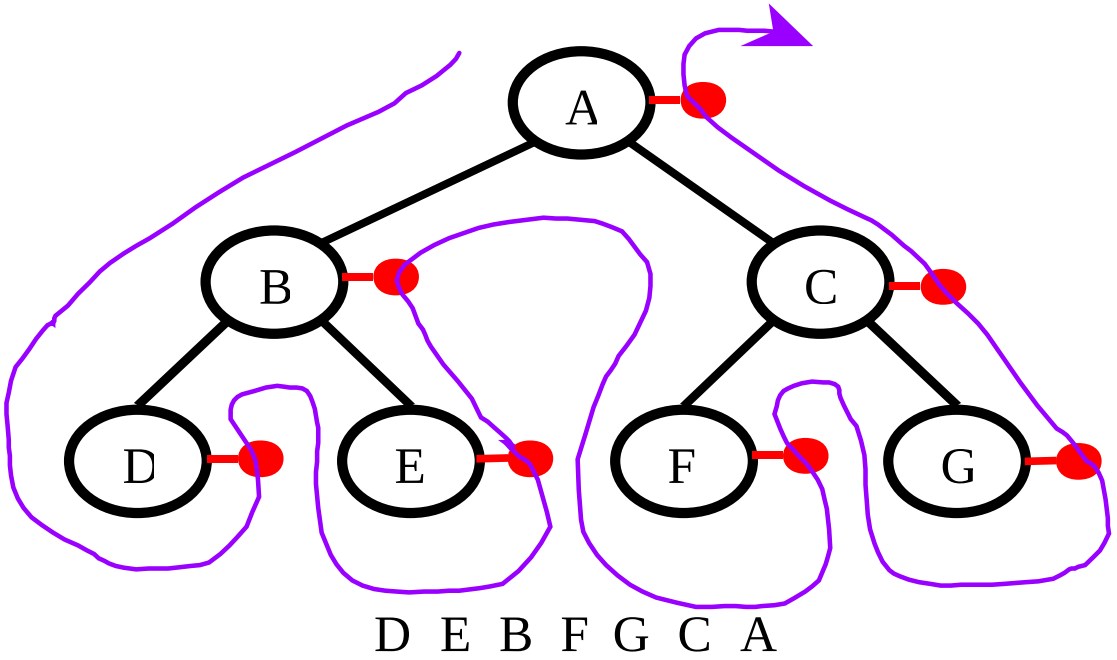
Kunjungan secara postorder menggunakan urutan:

- Kunjungi cabang kiri
- Kunjungi cabang kanan
- Cetak isi simpul yang dikunjungi

Postorder

```
void postorder(pohon ph)
{
 if (ph != NULL)
 {
        postorder(ph->kiri);
        postorder(ph->kanan);
        printf("%c ", ph->info);
 }
}
```

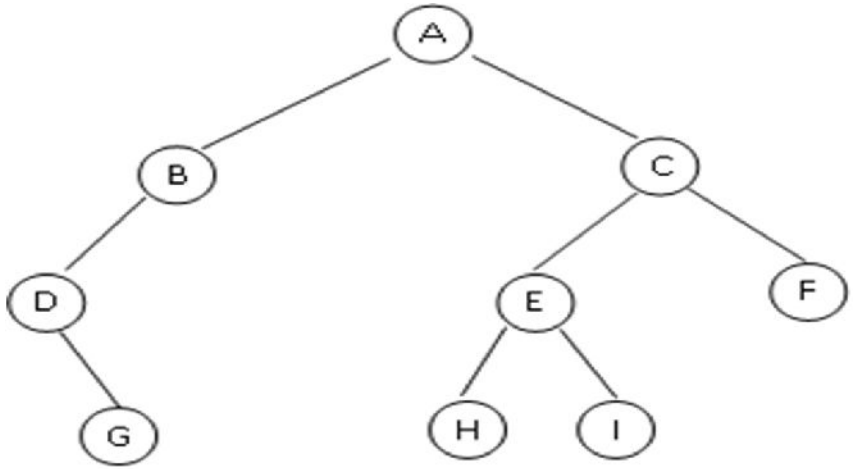
Postorder



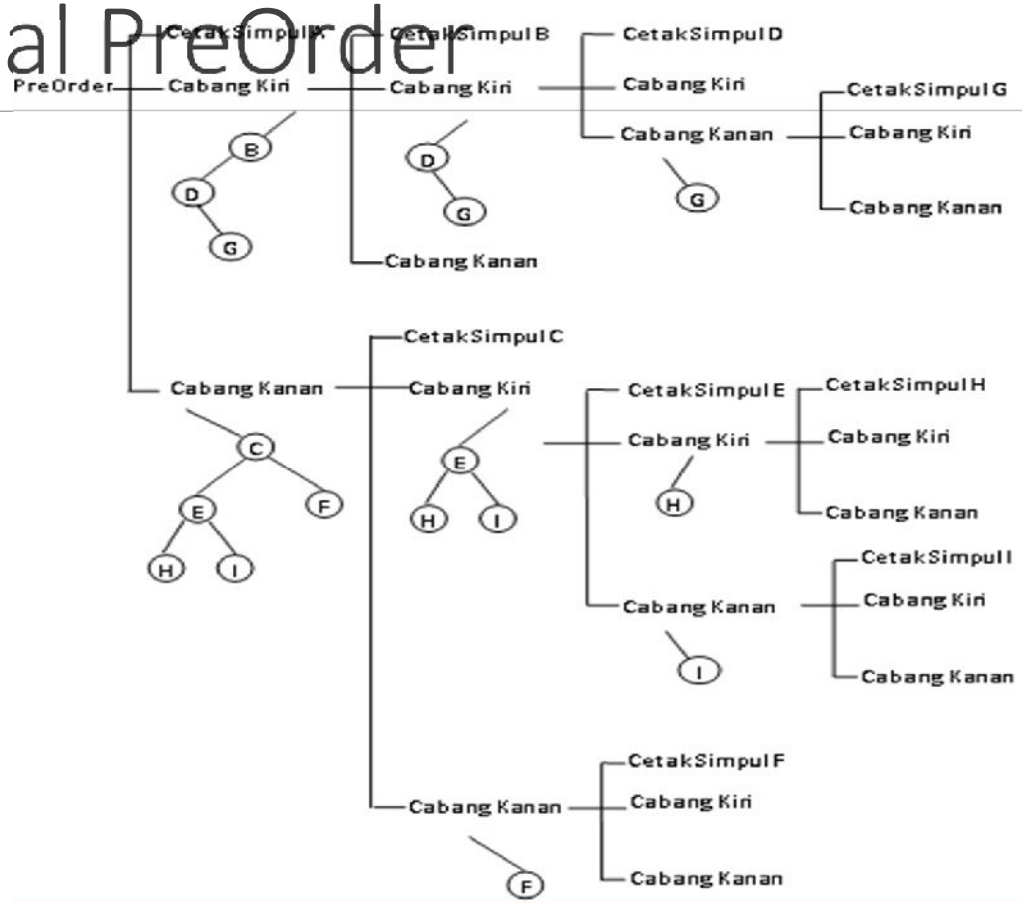
Studi Kasus 1

Terdapat Tree sbb, lakukan traversal tree secara

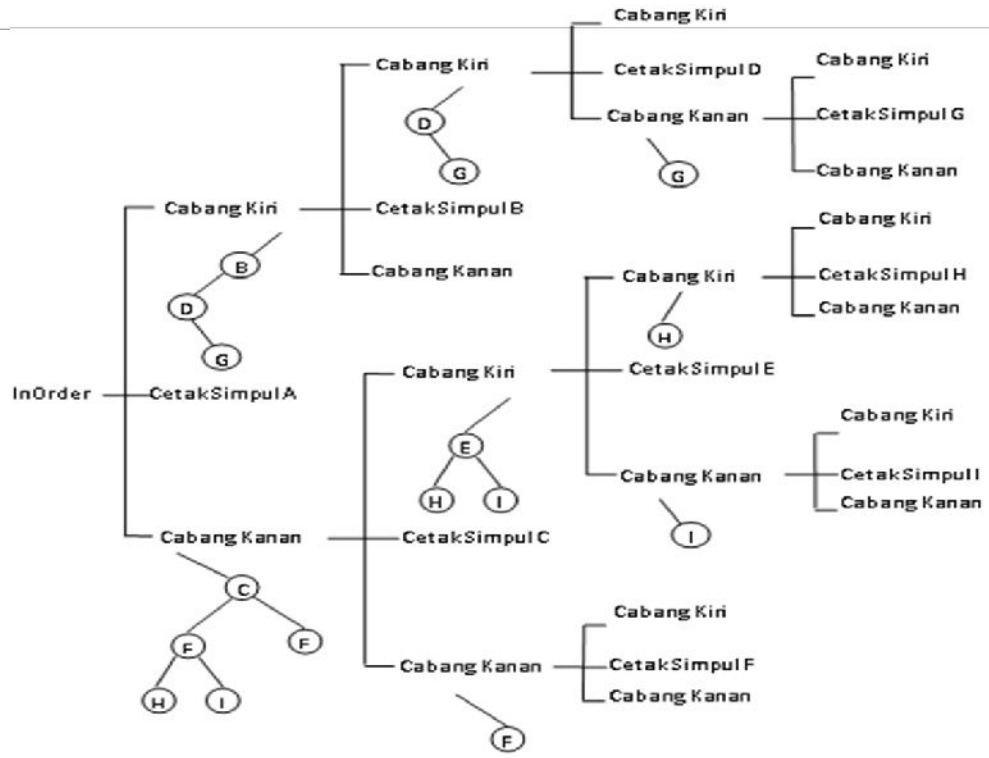
- Inorder
- Preorder
- Postorder



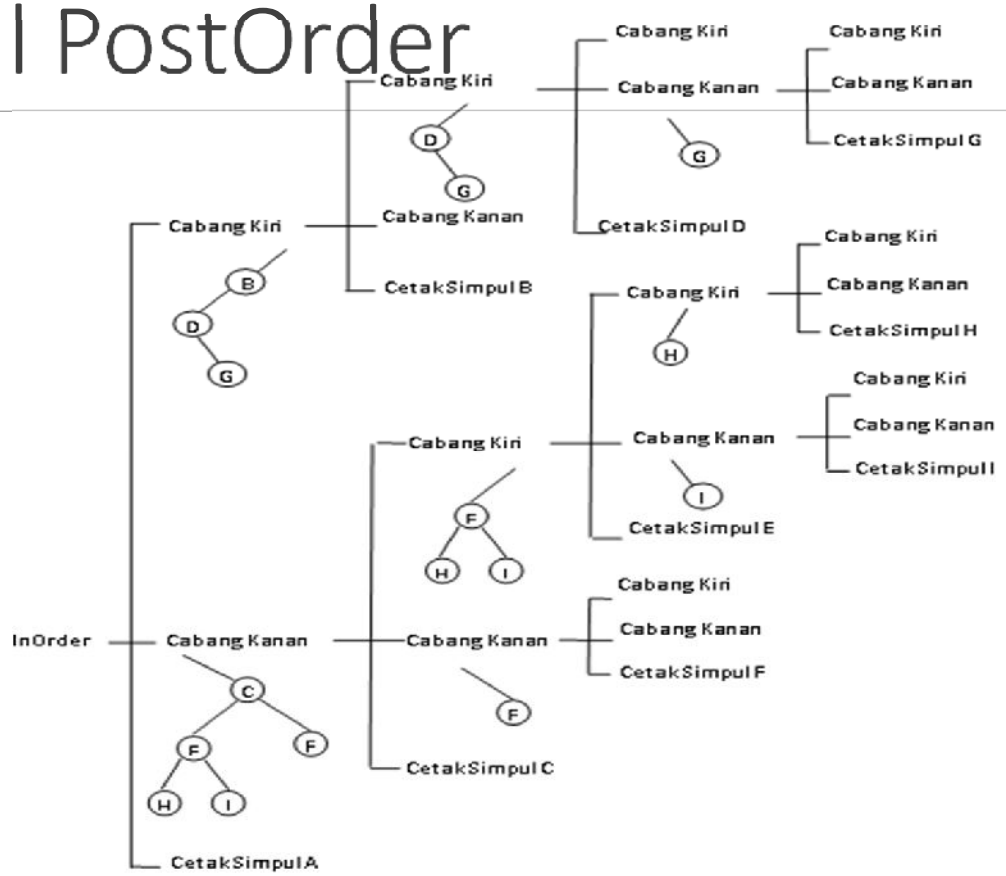
Traversal PreOrder



Traversal InOrder



Traversal PostOrder



Struktur Binary Tree

Masing-masing simpul dalam binary tree terdiri dari tiga bagian yaitu sebuah data dan dua buah pointer yang dinamakan pointer kiri dan kanan.

Deklarasi Tree

```
typedef char TypeInfo;
typedef struct Simpul *Tree;
struct Simpul {
        TypeInfo Info;
        tree Kiri,              /* cabang kiri */
        tree Kanan;             /* cabang kanan */
};
```

Membuat Simpul Baru

```
pohon baru(char hrf)
{
        pohon br;
        br=(pohon)malloc(sizeof(struct tree));
        br->info=hrf;
        br->kiri=NULL;
        br->kanan=NULL;
        return (br);
}
```

Fungsi untuk menyisipkan simpul pada Binary Search Tree yang sudah dibangun

```
void sisip (pohon ph, pohon sp)
{
    pohon P,Q;
    P = ph;
    Q = ph;
     while((sp->info != ph->info)&&(Q!=NULL))
   {
         P = Q;
         if (sp->info < P->info)
              Q = P->kiri;
         Else
                  Q = P->kanan;
     }
     if(sp->info == P->info)
          printf("Sudah ada");
     else
                  if(sp->info < P->info)
                                    P->kiri=sp;
                  Else
                                    P->kanan=sp;
}
```

Kesimpulan

Tree : Struktur pohon (tree) biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada.

Binary Tree : Sebuah pengorganisasian secara hirarki dari beberapa buah simpul, dimana masing-masing simpul tidak mempunyai anak lebih dari 2.

Binary Search Tree

Sebuah node di Binary Search Tree memiliki path yang unik dari root menurut aturan ordering

- Sebuah Node, mempunyai subtree kiri yang memiliki nilai lebih kecil dari node tsb dan subtree kanan memiliki nilai lebih besar dari node tsb.
- Tidak diperbolehkan ada node yang memiliki nilai yang sama.

Kesimpulan

Metode Traversal

- Inorder
- Preorder
- Postorder

Latihan Soal

Buatlah Binary Search Tree, dengan data

7 5 12 3 6 1 4 9

Lakukan Metode Traversal dengan algoritma:

- Inorder
- Preorder
- Postorder