01158 - ALGORITMA DAN STRUKTUR DATA 2

TREE

06

OLEH : ANDRI HERYANDI, M.T.

MATERI

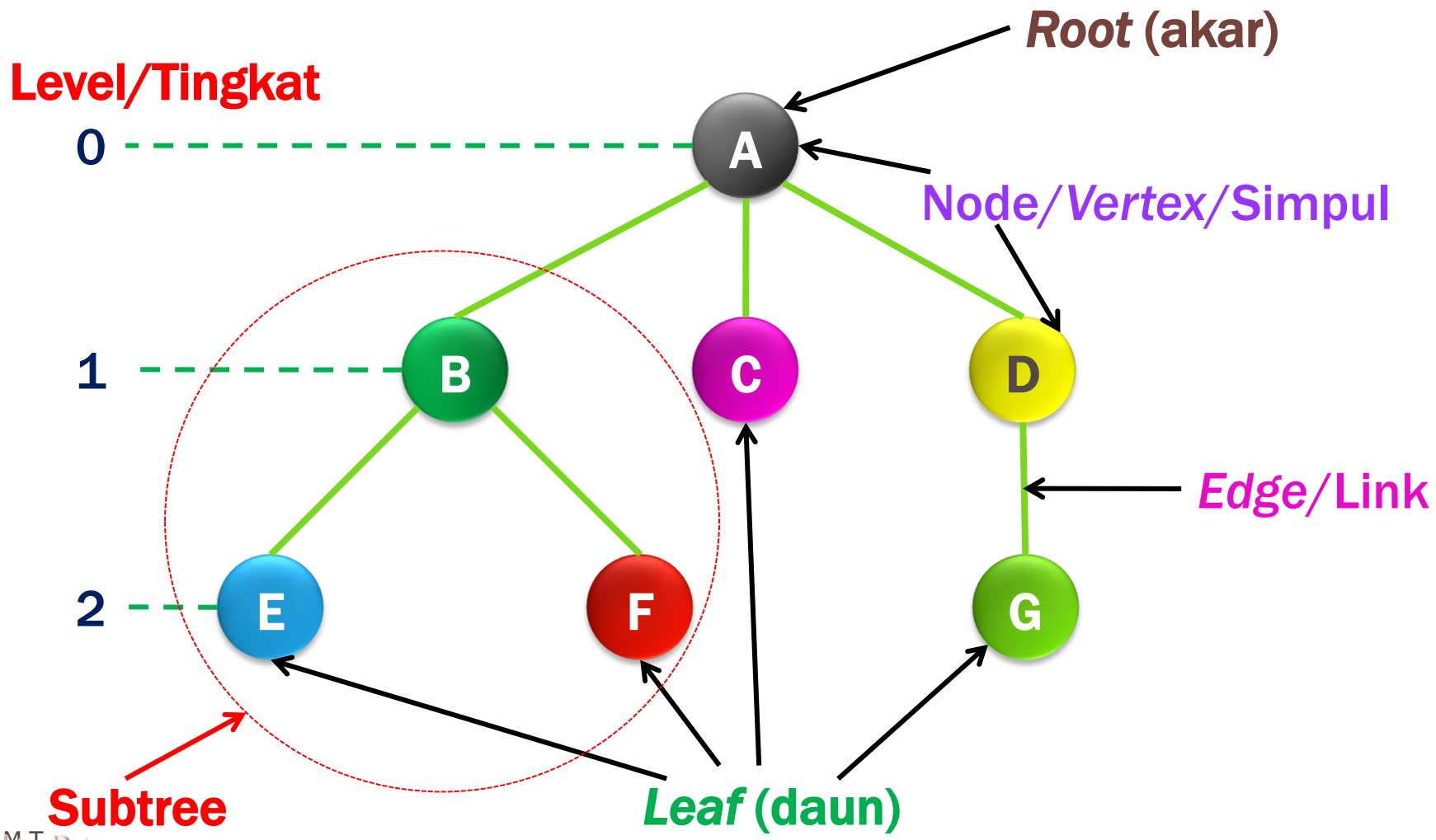
01158 - ALGORITMA DAN STRUKTUR DATA 2

- Definisi Tree
- Jenis-Jenis Tree
  - General Tree
  - Binary Tree
    - Binary Search Tree
    - Heap

DEFINISI TREE

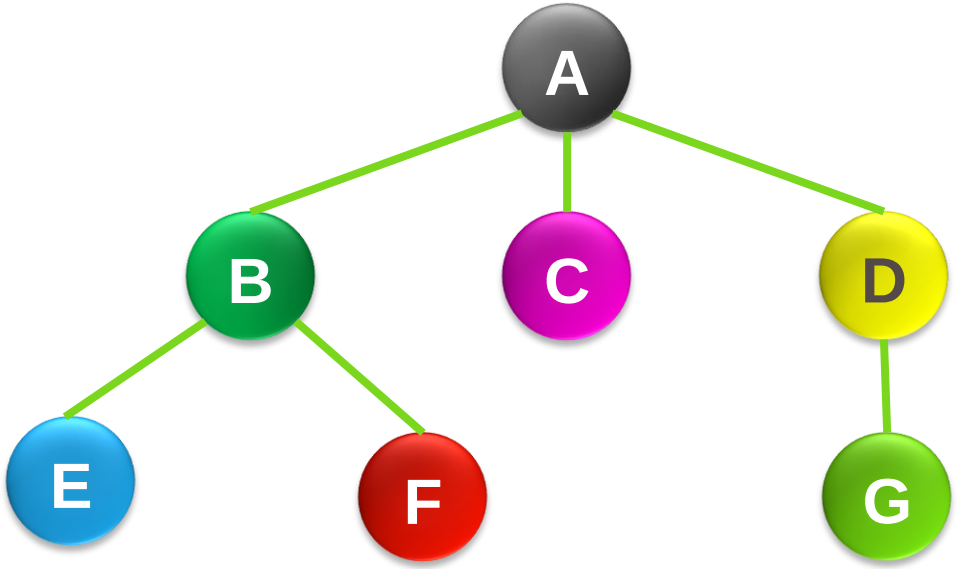
- Tree (pohon) adalah struktur data non linier
- Tree digunakan untuk untuk menyajikan data yang berisi hubungan hirarki antar elemennya.
- Contoh penggunaan tree
  - Silsilah Keluarga
  - Struktur Organisasi
  - Turnamen
  - Daftar Isi
  - Susunan Folder/File dalam media penyimpanan
  - Dokumen html
  - Kompresi Data
  - Pohon Keputusan

STRUKTUR TREE



ISTILAH DALAM TREE

- Parent(X) : Node yang berada satu level di atas node X.
- Child(X) : Node-node yang berada satu level di bawah node X.
- Sibling(X) : Node-node yang mempunyai parent yang sama dengan node X.
- Degree(X) : Banyak child yang dimiliki oleh node X.
- Height(X) : Banyaknya edge dari suatu node ke node paling bawah dari turunannya.
- Path(X,Y) : Jalur dari node X ke node Y.



Parent(E)	: B	*Child*(A)	: B, C, D
Sibling(C)	: B, D	*Degree*(A)	: 3
Height(A)	: 2	*Path*(A,G)	: A – D – G

JENIS-JENIS TREE

- Ada banyak jenis tree.
- Setiap tree mempunyak kegunaan masing-masing.
- Ada tree yang susunannya bebas, tetapi ada juga yang susunannya berdasarkan aturan tertentu.
- Ada tree yang tidak mempunyai maksimum degree yang sama, tapi ada juga yang mewajibkan maksimum degree yang sama.
- Berikut adalah beberapa jenis tree yang banyak digunakan :
  - General Tree
  - Binary Tree
    - Binary Seacrh Tree (BST)
    - Heap
  - Dan lain-lain.

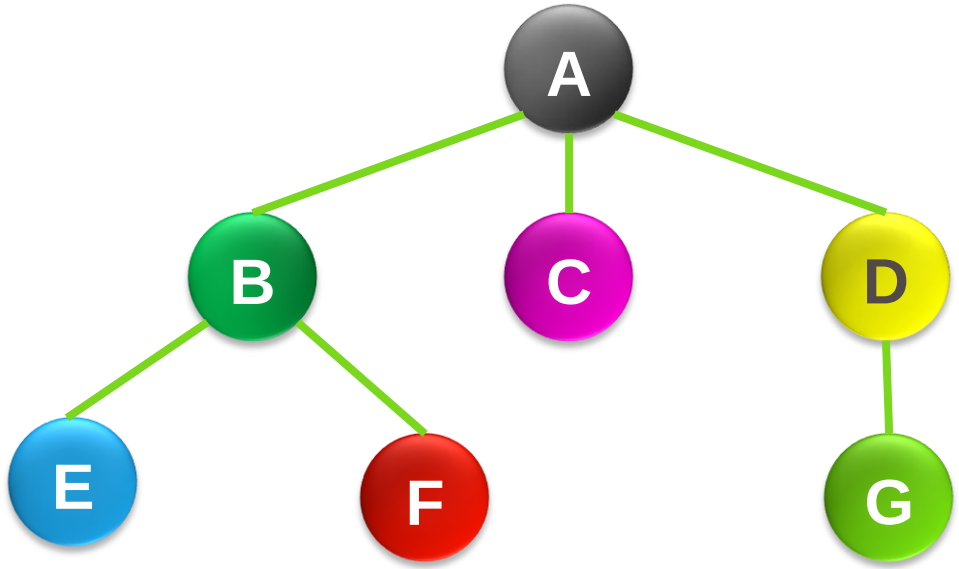
01158 - ALGORITMA DAN STRUKTUR DATA 2

GENERAL TREE

OLEH : ANDRI HERYANDI, M.T.

GENERAL TREE

- General tree adalah tree yang umum. Tree ini tidak mempunyai batasan, baik dari maksimum degree atau pun levelnya.
- Setiap node bisa memiliki 0 sampai dengan N node anak.



Contoh General Tree

STRUKTUR DATA GENERAL TREE

- Untuk membuat struktur data general tree, maka terlebih dahulu anda harus memahami sifat dari general tree, yaitu :
  - Tidak ada aturan dalam penempatan nodenya,
  - Setiap node bisa memiliki 0 atau banyak node anak (tidak dibatasi)
- Berdasarkan karakteristik di atas, mungkin saja anda beranggapan bahwa sebuah node akan memiliki sebuah linked list yang berisi node-node anak dan juga sebuah node yang menunjuk ke node berikutnya (node sibling). Kenapa pakai linked list? Ini karena banyaknya node anak dari sebuah node tidak dibatasi.
- Struktur untuk tree bisa disusun seperti pada gambar di bawah ini

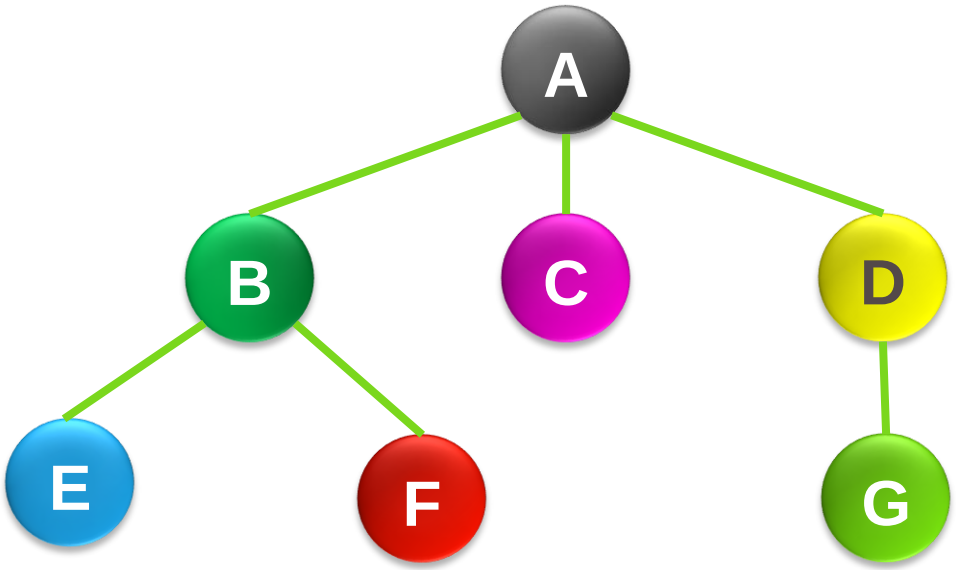
STRUKTUR DATA GENERAL TREE

- Pendeklarasian struktur data untuk general tree (Bahasa Python).

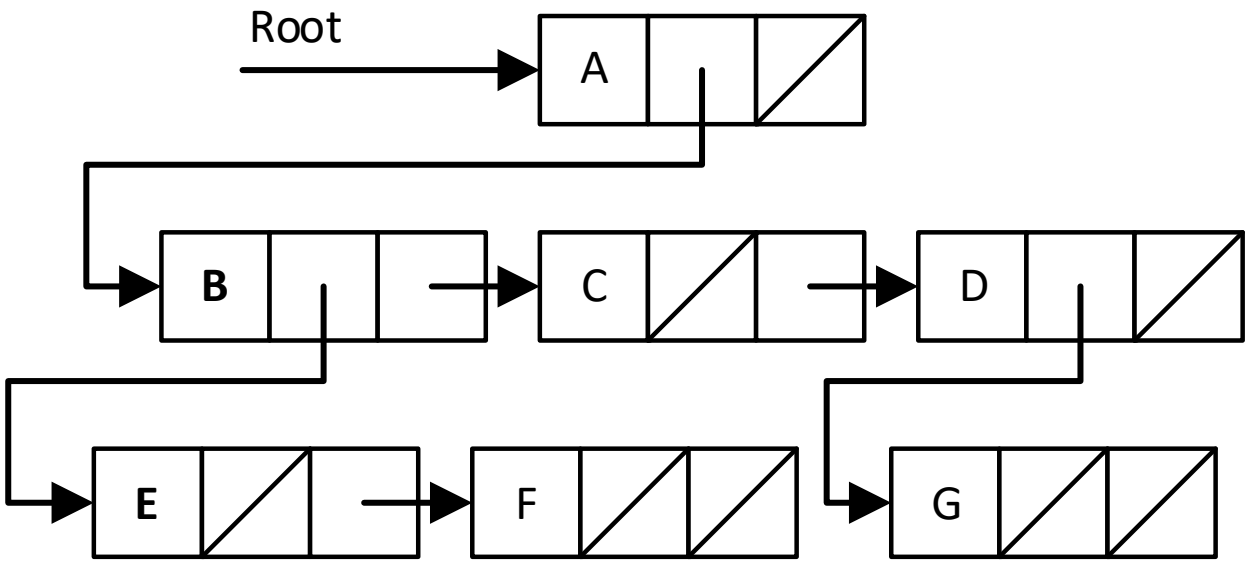
```
class Node:
    def __init__(self, data):
        self.data = data
        self.firstChild = None
        self.nextSibling = None
```

STRUKTUR DATA GENERAL TREE

Contoh General Tree dan implementasi struktur data di dalam memori



General Tree



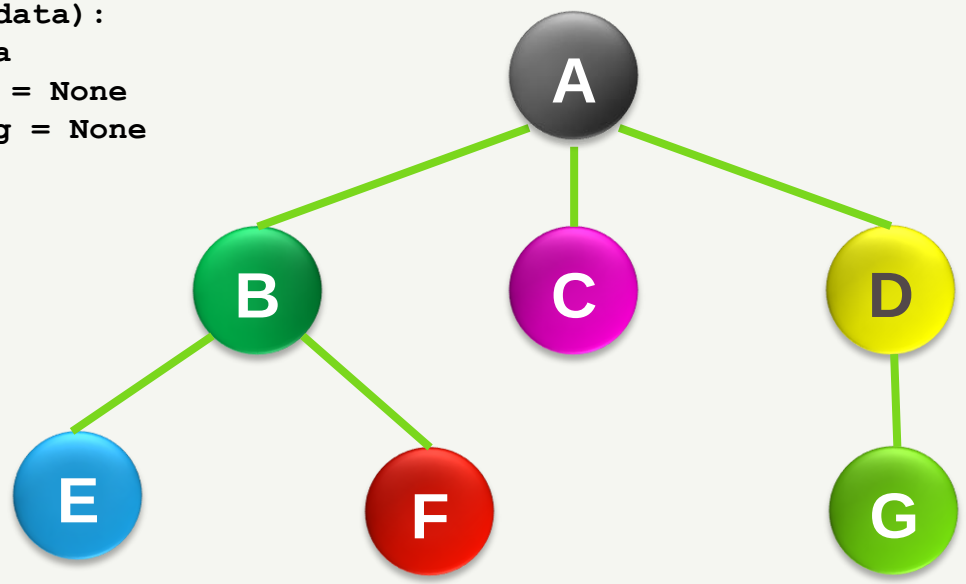
General Tree Dalam Memori

STRUKTUR DATA GENERAL TREE

- Contoh :

```
class Node:
    def __init__(self, data):
        self.data = data
        self.firstChild = None
        self.nextSibling = None

A = Node("A")
B = Node("B")
C = Node("C")
D = Node("D")
E = Node("E")
F = Node("F")
G = Node("G")
A.firstChild=B
B.nextSibling=C
C.nextSibling=D
B.firstChild=E
E.nextSibling=F
D.firstChild=G
print("Root : ", A.data)
print("Root -> First Child  : ", A.firstChild.data)
print("Root -> First Child -> Next Sibling : ", A.firstChild.nextSibling.data)
print("Root -> First Child -> First Child : ", A.firstChild.firstChild.data)
```



Hasil Run :

```
Root :  A
Root -> First Child  :  B
Root -> First Child -> Next Sibling :  C
Root -> First Child -> First Child :  E
```

STRUKTUR DATA GENERAL TREE

- Jika anda perhatikan dalam struktur general tree, di dalam sebuah node terdiri dari 3 bagian yaitu :
  - Data yang dimilikinya (info).
  - 2 buah pointer yang menunjuk ke node anak pertama dan pointer yang menunjuk ke node sibling berikutnya.
- Dapat dilihat bahwa setiap node memiliki 2 node pointer. Sifat hal ini juga yang dimiliki oleh tree yang bernama Binary Tree (dimana memiliki 2 pointer, yaitu pointer ke anak kiri dan pointer ke anak kanan).
- Dikarenakan hal tersebutlah, biasanya general tree juga sering diperlakukan sebagai binary tree. Sehingga sebuah general tree dapat diubah menjadi sebuah binary tree.

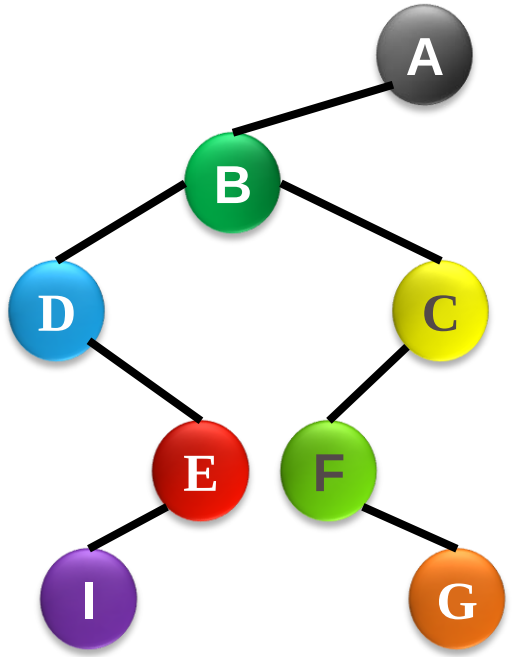
01158 - ALGORITMA DAN STRUKTUR DATA 2

BINARY TREE

OLEH : ANDRI HERYANDI, M.T.

BINARY TREE

- Binary Tree adalah tree yang hanya memiliki maksimal 2 node anak.
- 2 Node anak tersebut biasanya disebut sebagai node anak Kiri (Left) dan node anak Kanan (Right).
- Dalam binary tree, urutan node tidak harus berdasarkan aturan tertentu.



STRUKTUR DATA BINARY TREE

- Karakteristik dari binary tree adalah
  - Setiap node bisa memiliki maksimum 2 node anak.
  - Node anak pertama sering disebut sebagai node Left
  - Node anak kedua sering disebut sebagai node Right.
  - Sebuah node dianggap tidak memiliki anak jika node Left bernilai NIL dan node Right juga bernilai NIL.
- Struktur Data untuk binary tree adalah sebagai berikut :

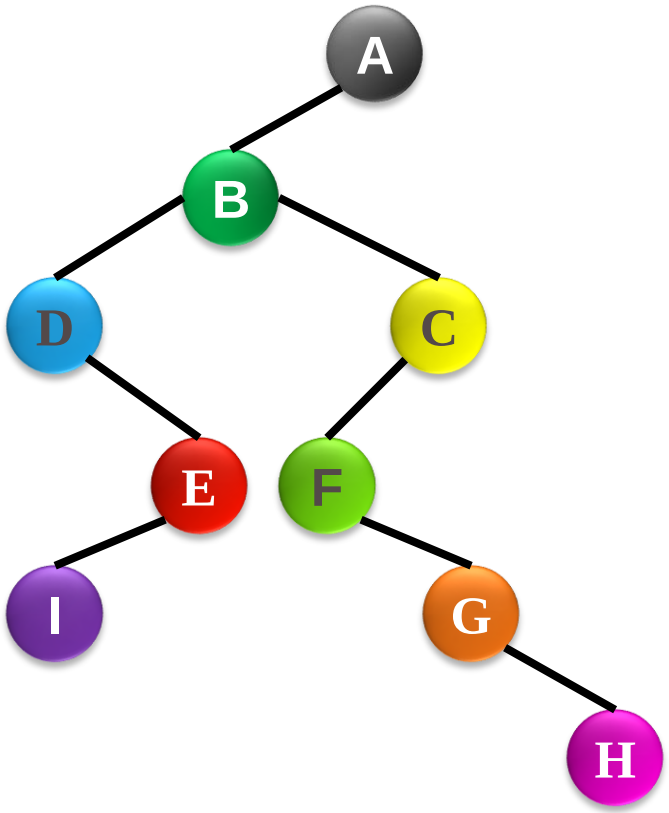
STRUKTUR DATA BINARY TREE

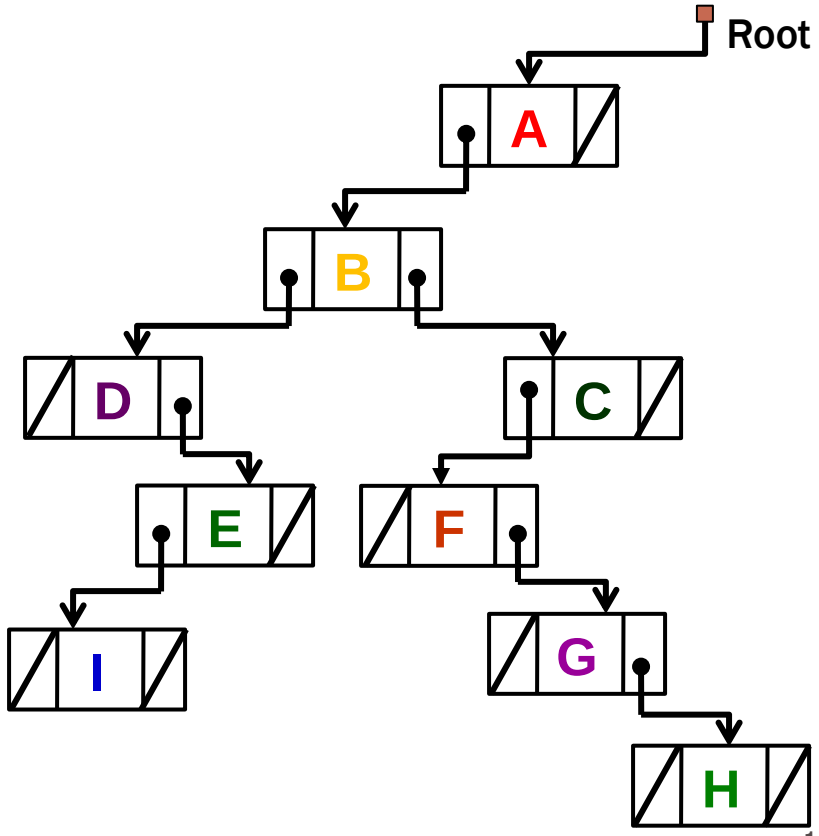
- Pendeklarasian struktur data untuk binary tree (Bahasa Python).

```
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None
```

STRUKTUR DATA BINARY TREE

- Perbandingan Binary Tree dan Implementasi Binary Tree di Memori





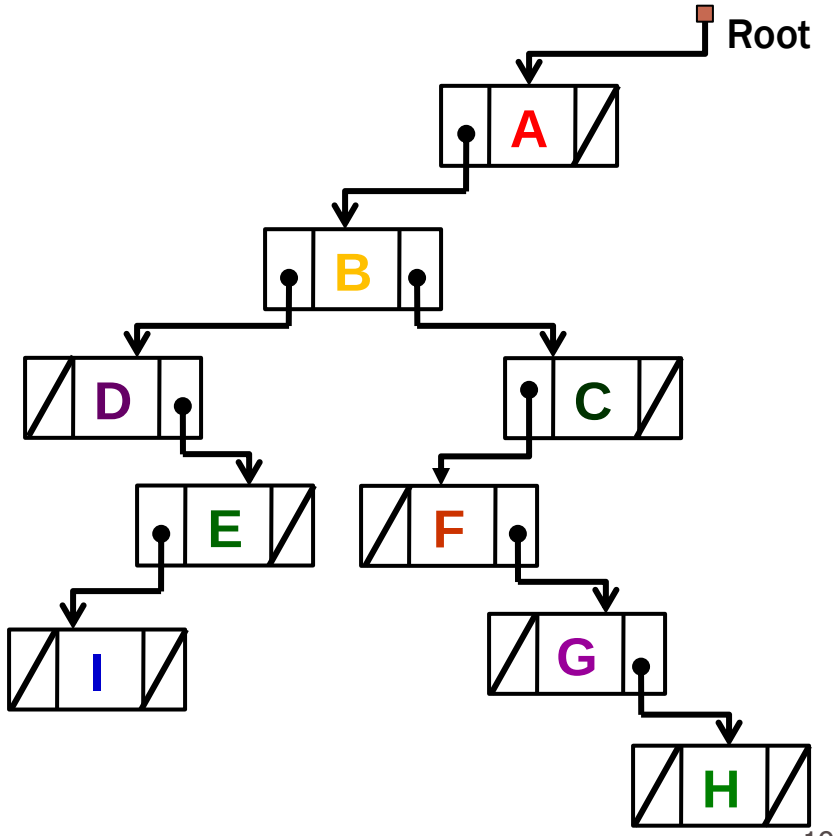
STRUKTUR DATA BINARY TREE

- Contoh :

```
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

A = Node("A")
B = Node("B")
C = Node("C")
D = Node("D")
E = Node("E")
F = Node("F")
G = Node("G")
H = Node("H")
I = Node("I")

A.left = B
B.left = D
B.right= C
C.left = F
D.right= E
E.left = I
F.right= G
G.right= H
```



OPERASI PADA BINARY TREE

- Operasi-operasi yang dapat dilakukan dalam binary tree adalah :
  - Penelusuran Tree
  - Insert Node Baru
  - Penghapusan Node
  - Pembentukan Tree

PENELUSURAN PADA BINARY TREE

- Penelusuran tree dilakukan untuk mengunjungi setiap node yang ada dalam tree.
- Ada 3 cara penelurusan binary tree yaitu :
  - Secara Preorder, yaitu proses menelusuri binary tree dengan cara memproses informasi pada node tersebut, dilanjutkan dengan memproses node anak kiri (Left) dan dilanjutkan dengan memproses node anak kanan (Right). Penelusuran ini disebut juga sebagai penelusuran NLR (Node – Left – Right), disesuaikan dengan cara urutan penelusurannya.
  - Secara Inorder, yaitu proses menelusuri binary tree dengan cara memproses informasi pada node anak kiri (Left) terlebih dahulu, dilanjutkan dengan memproses informasi pada node tersebut dan dilanjutkan dengan memproses node anak kanan (Right). Penelusuran ini disebut juga dengan penelusuran secara LNR (Left – Node – Right).
  - Secara Postorder, yaitu proses menelusuri binary tree dengan cara memproses informasi pada node anak kiri (Left) terlebih dahulu, kemudian diikuti dengan memproses pada node anak kanan (Right) dan diakhiri dengan memproses informasi pada node tersebut. Penelusuran ini disebut juga dengan penelusuran secara LRN (Left – Right – Node).

ALGORITMA PENELUSURAN PADA BINARY TREE

- Algoritma Penelusuran Binary Tree secara PreOrder untuk tree dimulai dari node N adalah sebagai berikut :
  1. Proses informasi yang ada di node N. Misalnya menampilkan ke layar yang ada di field info pada node N.
  2. Jika anak kiri dari node N tidak NIL, maka lakukan PreOrder pada node anak kiri dari N. Hal ini dilakukan secara rekursif.
  3. Jika anak kanan dari node N tidak NIL, maka lakukan PreOrder pada node anak kanan dari N. Ini juga dilakukan secara rekursif.
- Implementasi penesuruan binary tree secara PreOrder di Bahasa Python.

```
def preorder(N):
    print(N.data, end=" ")
    if N.left is not None:
        preorder(N.left)
    if N.right is not None:
        preorder(N.right)
```

ALGORITMA PENELUSURAN PADA BINARY TREE

- Algoritma Penelusuran Binary Tree secara InOrder untuk tree dimulai dari node N adalah sebagai berikut :
  1. Jika anak kiri dari node N tidak NIL, maka lakukan InOrder pada node anak kiri dari N. Hal ini dilakukan secara rekursif.
  2. Proses informasi yang ada di node N. Misalnya menampilkan ke layar yang ada di field info pada node N.
  3. Jika anak kanan dari node N tidak NIL, maka lakukan InOrder pada node anak kanan dari N. Ini juga dilakukan secara rekursif.
- Implementasi penesuruan binary tree secara InOrder di Bahasa Python.

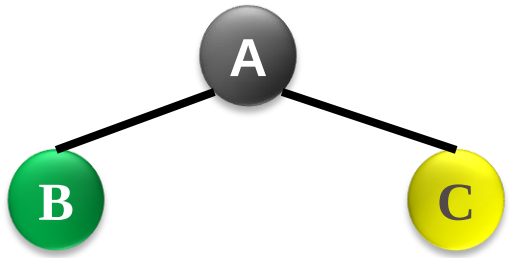
```
def inorder(N):
    if N.left is not None:
        inorder(N.left)
    print(N.data, end=" ")
    if N.right is not None:
        inorder(N.right)
```

ALGORITMA PENELUSURAN PADA BINARY TREE

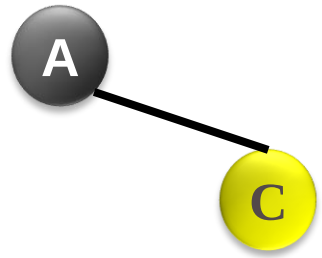
- Algoritma Penelusuran Binary Tree secara PostOrder untuk tree dimulai dari node N adalah sebagai berikut :
  1. Jika anak kiri dari node N tidak NIL, maka lakukan PostOrder pada node anak kiri dari N. Hal ini dilakukan secara rekursif.
  2. Jika anak kanan dari node N tidak NIL, maka lakukan PostOrder pada node anak kanan dari N. Ini juga dilakukan secara rekursif.
  3. Proses informasi yang ada di node N. Misalnya menampilkan ke layar yang ada di field info pada node N.
- Implementasi penesuruan binary tree secara PostOrder di Bahasa Python.

```
def postorder(N):
    if N.left is not None:
        postorder(N.left)
    if N.right is not None:
        postorder(N.right)
    print(N.data, end=" ")
```

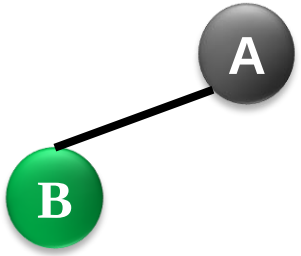
PENELUSURAN PADA BINARY TREE



PreOrder (NLR)	InOrder (LNR)	PostOrder (LRN)
A B C	B A C	B C A

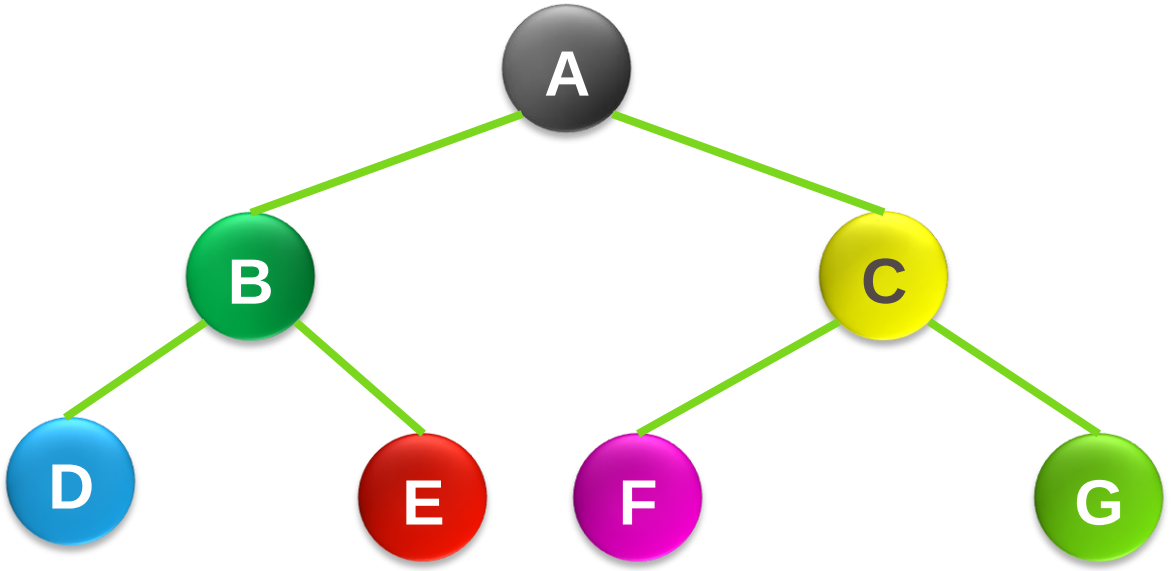


PreOrder (NLR)	InOrder (LNR)	PostOrder (LRN)
A C	A C	C A



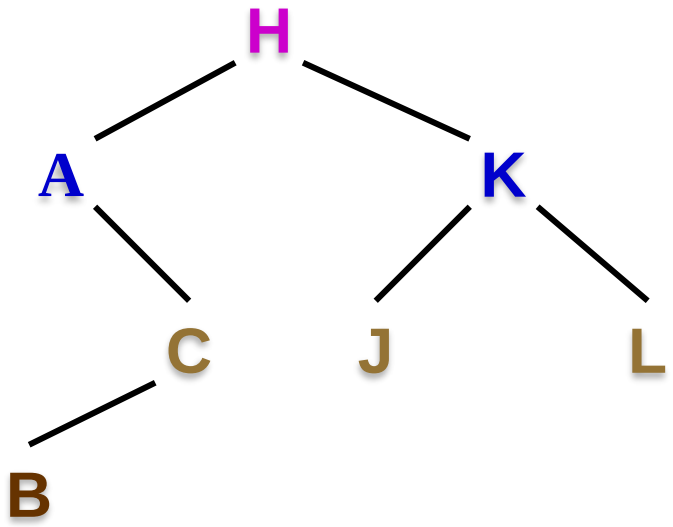
PreOrder (NLR)	InOrder (LNR)	PostOrder (LRN)
A B	B A	B A

PENELUSURAN PADA BINARY TREE



PreOrder (NLR)	InOrder (LNR)	PostOrder (LRN)
A B D E C F G N L R N L R N L R	D B E A F C G L N R L N R L N R	D E B F G C A L R N L R N L R N

PENELUSURAN PADA BINARY TREE

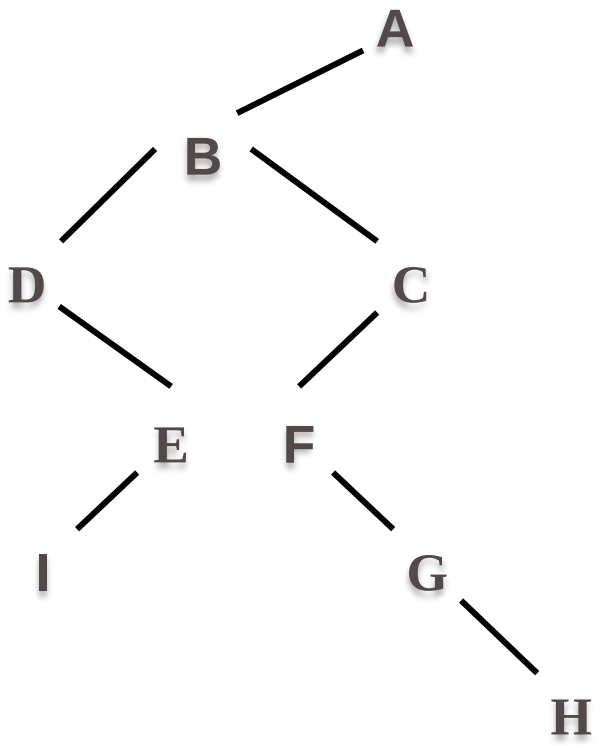


Preorder (NLR) : H A C B K J L

Inorder (LNR) : A B C H J K L

Postorder (LRN) : B C A J L K H

PENELUSURAN PADA BINARY TREE



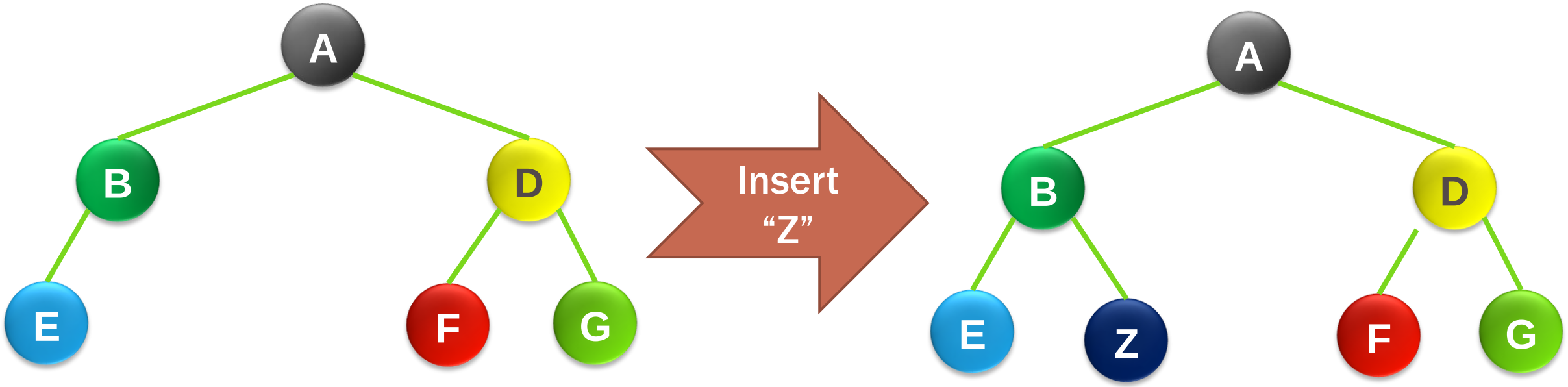
Preorder (NLR) : A B D E I C F G H
Inorder (LNR) : D I E B F G H C A
Postorder (LRN) : I E D H G F C B A

INSERT NODE BARU PADA BINARY TREE

- Operasi insert node baru dalam binary tree dilakukan untuk menyisipkan node baru agar menjadi bagian dari tree.
- Langkah-langkah melakukan penambahan/insert node baru ke sebuah binary tree adalah sebagai berikut :
  1. Cari node yang masih belum memiliki 2 node anak. Periksa dari node di level 1 ke level-level berikutnya.
  2. Jika telah ditemukan node yang belum memiliki 2 anak (misalnya belum punya anak sama sekali, atau punya anak kiri saja, atau punya anak kanan saja), maka sambungkan node baru tersebut sebagai node anak dari node tersebut. Prioritaskan node baru pada node sebelah kiri (jika node belum memiliki anak kiri).

INSERT NODE BARU PADA BINARY TREE

- Ilustrasi insert node baru dalam binary tree.

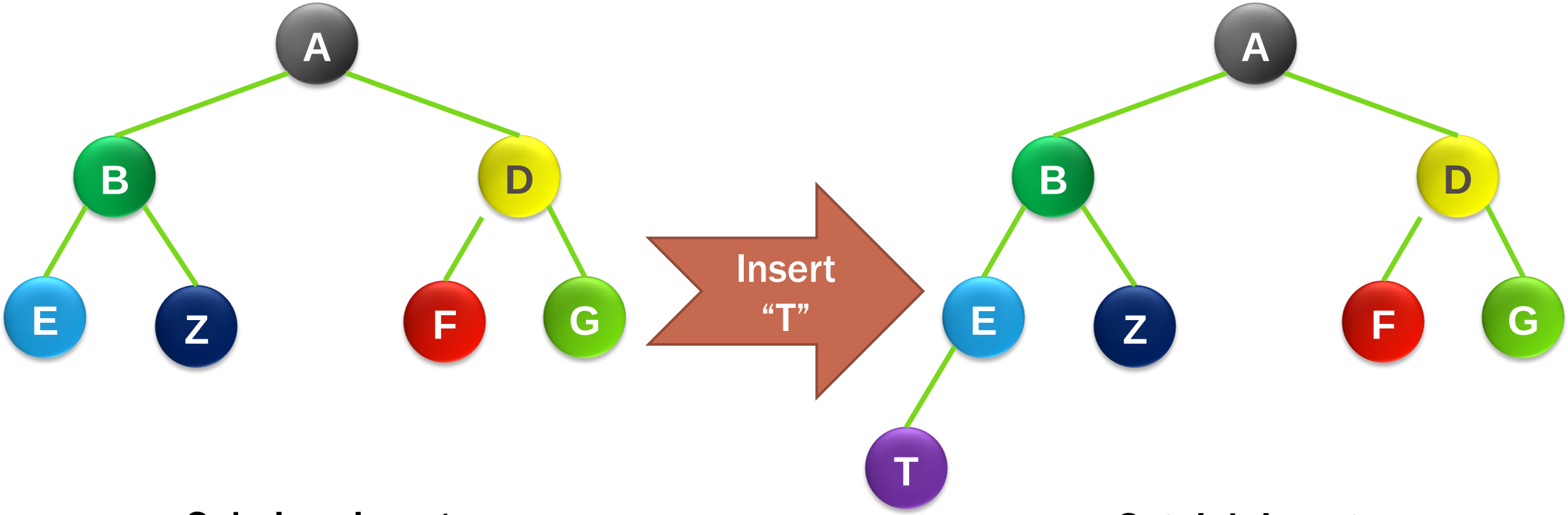


Sebelum Insert

Setelah Insert

INSERT NODE BARU

- Ilustrasi insert node baru dalam binary tree.



Sebelum Insert

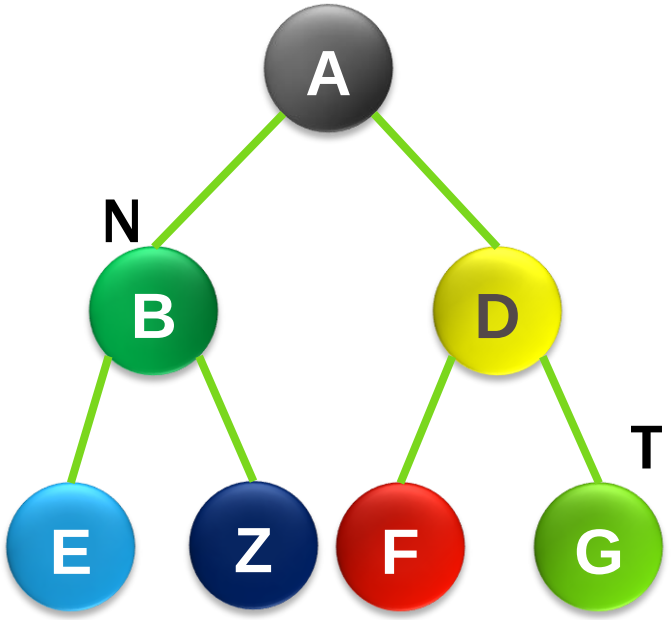
Setelah Insert

PENGHAPUSAN NODE DALAM BINARY TREE

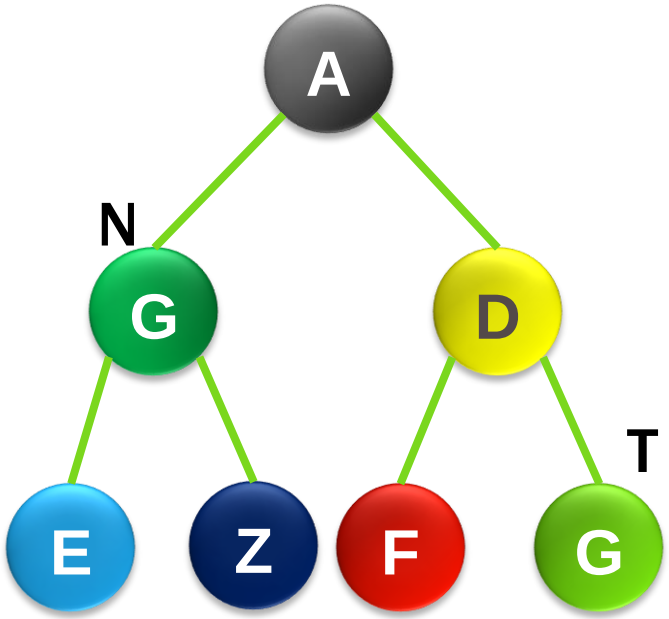
- Penghapusan sebuah node dalam binary tree.
- Langkah-langkah melakukan penghapusan node pada sebuah binary tree adalah sebagai berikut :
  1. Cari node yang akan dihapus (sebut sebagai N).
  2. Cari node yang berada pada posisi paling kanan di level paling bawah (sebut sebagai node T).
  3. Isikan informasi dari node T ke node N. (isi node T diisikan ke node N sebagai node pengganti yang akan dihapus).
  4. Hapus node T.

PENGHAPUSAN NODE DALAM BINARY TREE

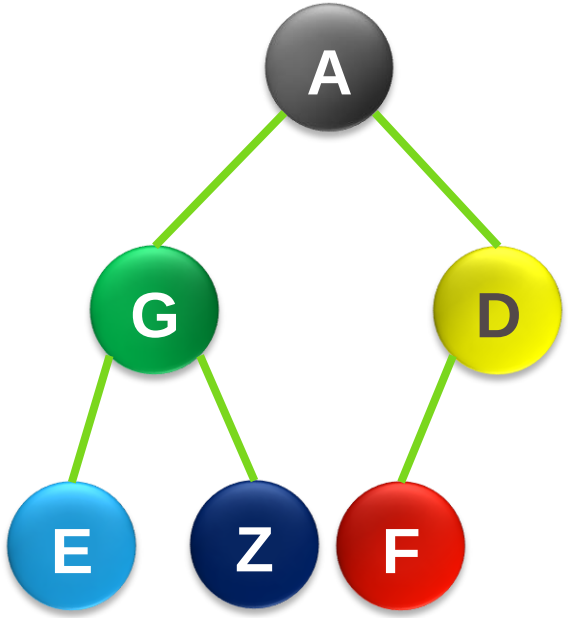
- Penghapusan sebuah node dalam binary tree. Contoh : Hapus node “B”



Cari node yang akan dihapus (N) dan node paling akhir (T)



Isikan info node T ke node N



Hapus node terakhir (level paling bawah, paling kanan)

PEMBENTUKAN BINARY TREE

- Pembentukan binary tree dilakukan ketika kita sudah memiliki sekumpulan data yang akan dijadikan binary tree.
- Ada 3 cara untuk membentuk binary tree, yaitu :
  - Pembentukan binary tree dari data masukan.
  - Pembentukan binary tree dari general tree.
  - Pembentukan binary tree dari hasil penelusuran binary tree.

PEMBENTUKAN BINARY TREE DARI DATA MASUKAN

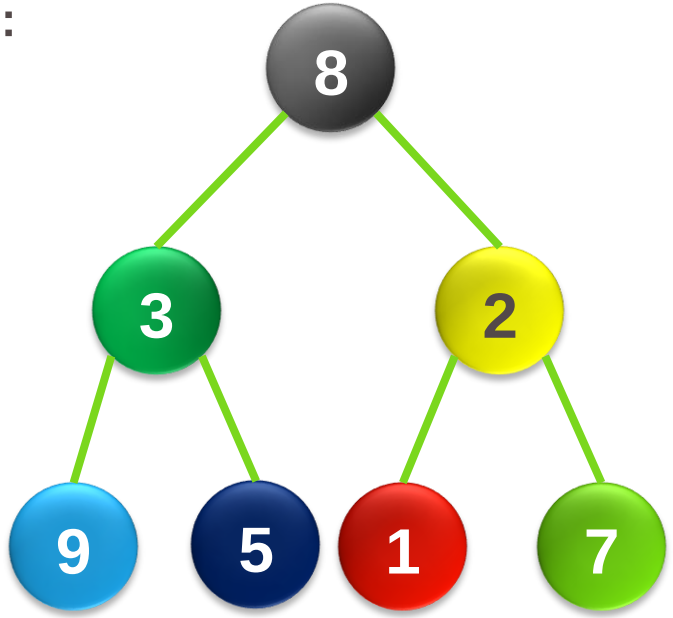
- Pembentukan binary tree dari masukan data berarti proses pembentukan data berdasarkan sekumpulan data yang sudah tersedia sebelumnya.
- Langkahnya adalah dengan melakukan proses insert node baru (lihat operasi insert node baru) satu per satu ke dalam tree.

PEMBENTUKAN BINARY TREE DARI DATA MASUKAN

Ilustrasi pembentukan binary tree dari data masukan.

Contoh data masukan :

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

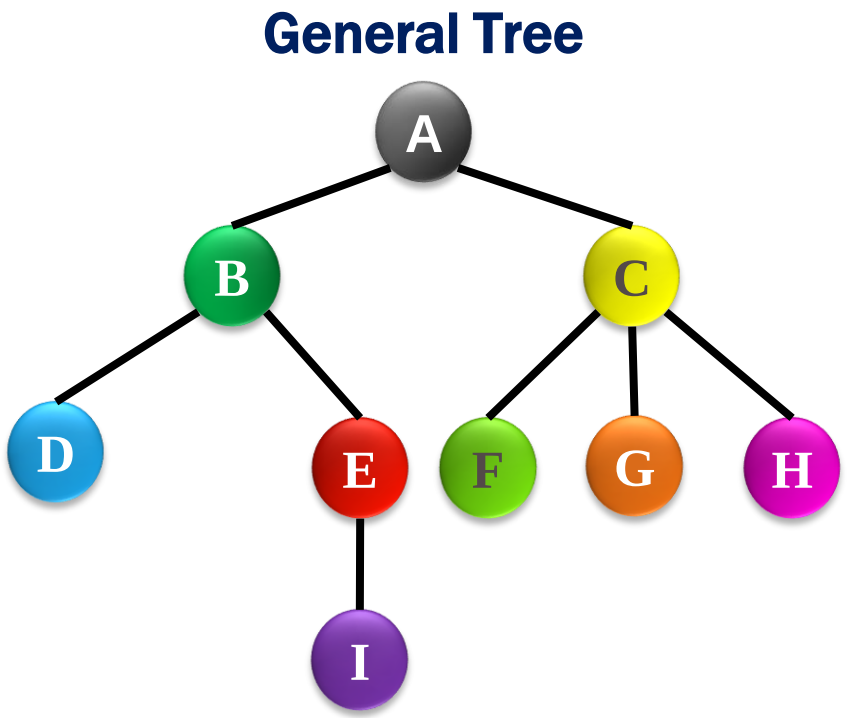


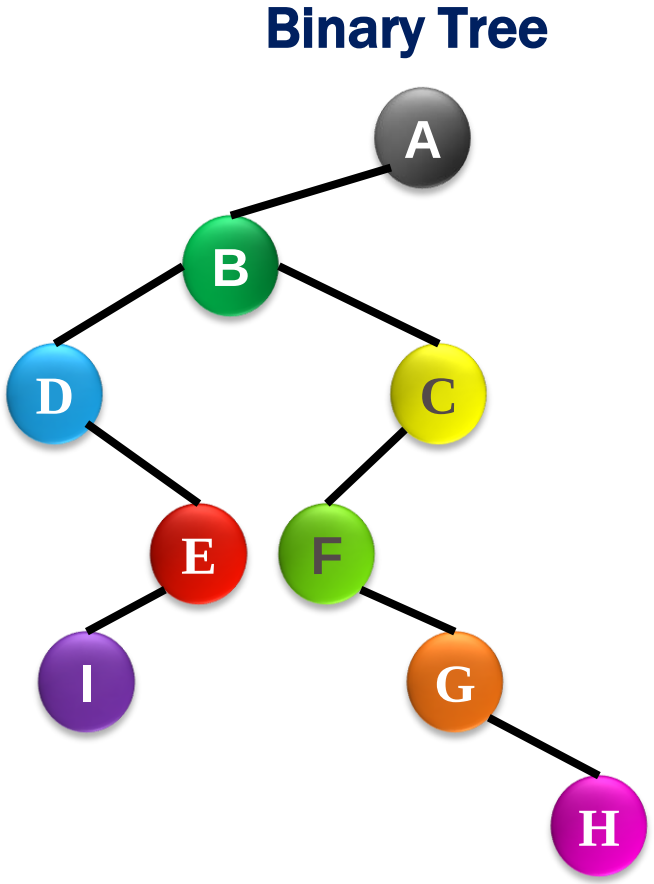
PEMBENTUKAN BINARY TREE DARI GENERAL TREE

- Sebuah general tree dapat dijadikan binary tree.
- Langkah untuk mengkonversinya adalah dengan ketentuan sebagai berikut :
  a. Anak pertama di *general tree* menjadi anak kiri di *binary tree*
  b. Saudaranya (***next sibling***) di *general tree* menjadi anak kanan di *binary tree*

PEMBENTUKAN BINARY TREE DARI GENERAL TREE

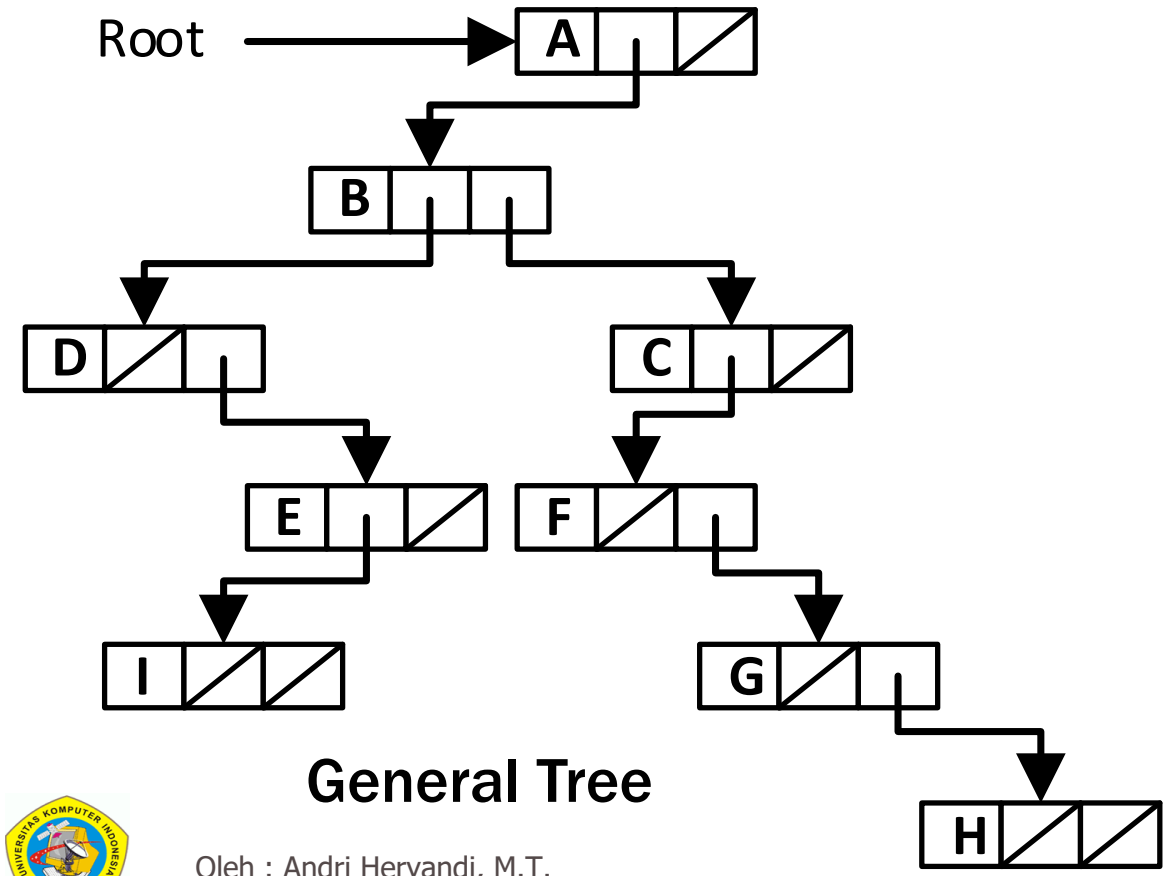
Contoh konversi general tree menjadi binary tree.



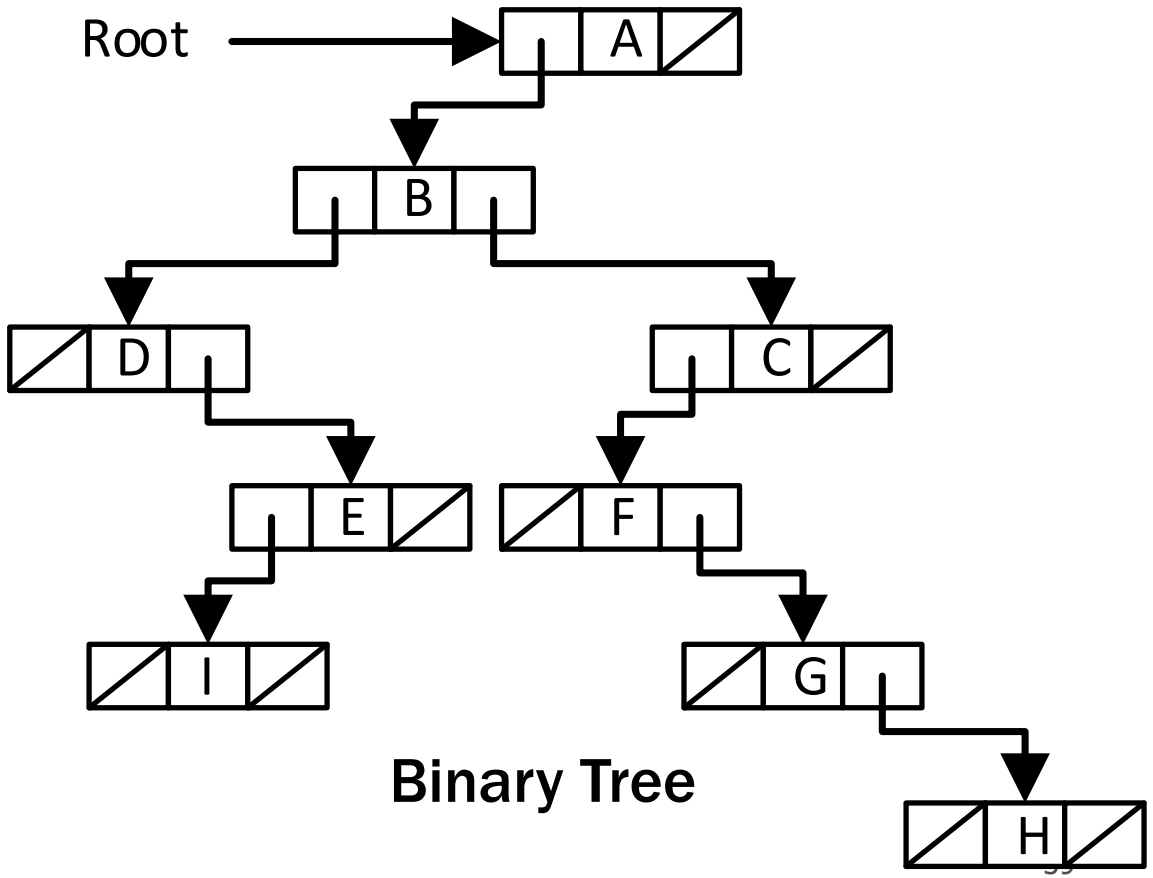


PEMBENTUKAN BINARY TREE DARI GENERAL TREE

Contoh konversi general tree menjadi binary tree di memori.



General Tree



Binary Tree

PEMBENTUKAN BINARY TREE DARI HASIL PENELUSURAN

- Pembentukan binary tree dari hasil penelusuran tree dimungkinkan dengan syarat melibatkan 2 hasil penelusuran yang salah satunya adalah penelusuran InOrder.
- Hasil penelusuran PreOrder dan PostOrder digunakan untuk urutan eksekusi pembentukan treenya, sedangkan hasil penelusuran InOrder digunakan untuk mengetahui lokasi suatu node berdasarkan node sebelumnya.

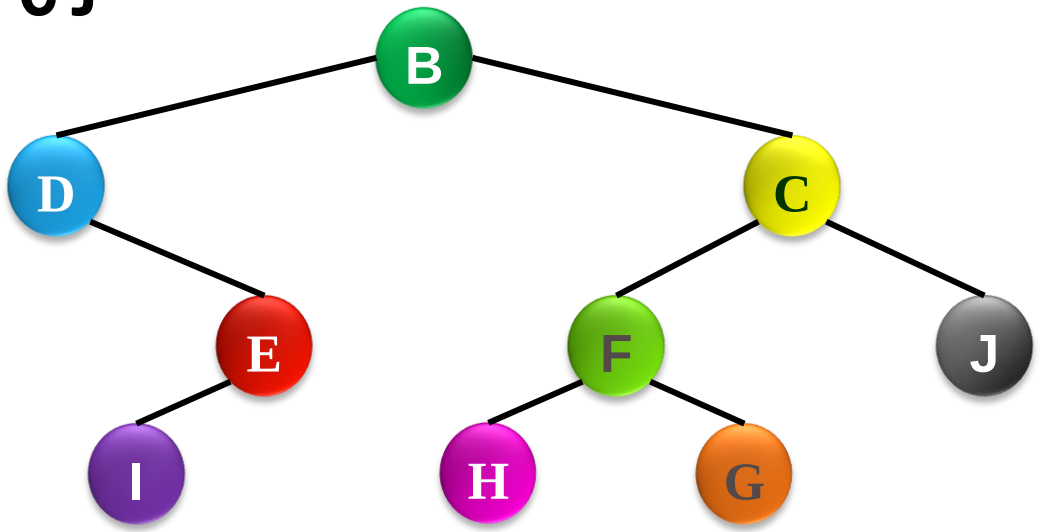
PEMBENTUKAN BINARY TREE DARI HASIL PENELUSURAN

- Contoh pembentukan binary tree dari hasil penelusuran InOrder dan PreOrder

Diketahui dua hasil penelusuran sbb:

Preorder : **B D E I C F H G J**

Inorder : **D I E B H F G C J**



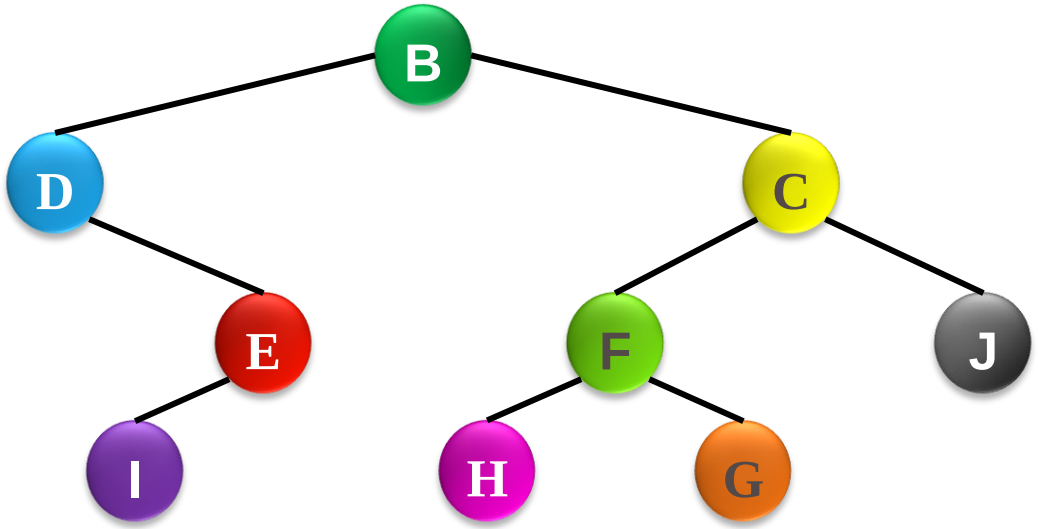
PEMBENTUKAN BINARY TREE DARI HASIL PENELUSURAN

- Contoh pembentukan binary tree dari hasil penelusuran InOrder dan PostOrder

Diketahui dua hasil penelusuran sbb:

InOrder : **D I E B H F G C J**

PostOrder : **I E D H G F J C B**



FORUM DISKUSI

LMS UNIKOM
https://lms.unikom.ac.id

Group Whatsapp Perkuliahan

Youtube Comments
https://unikom.id/YTCStrukturData