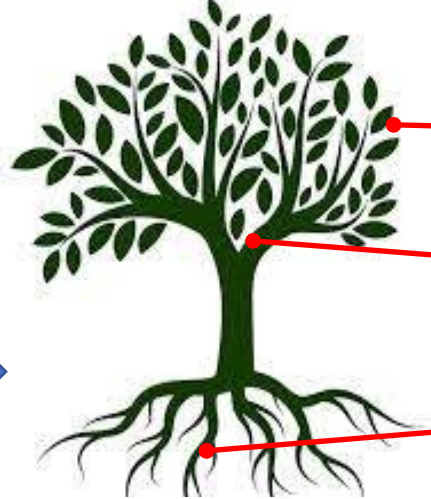
PERTEMUAN 7

Struktur Data & Algoritma

Tree (Pohon)

TREE

REAL WORLD

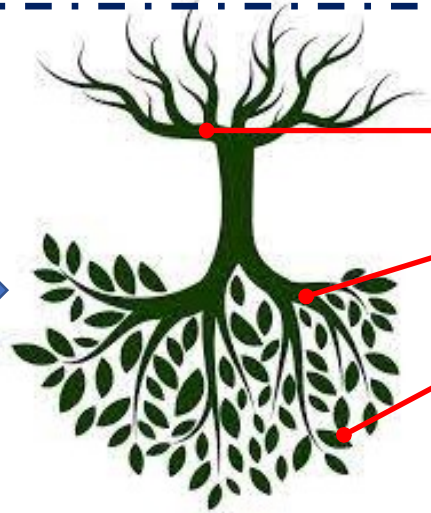


Leaf

Branch

Root

COMPUTER SCIENTIST'S VIEW



Root (akar)

Branch (cabang)

Leaf (daun)

PENGERTIAN TREE

Kumpulan node yang saling terhubung satu sama lain dalam suatu kesatuan yang membentuk layakya struktur sebuah pohon. Struktur pohon adalah suatu cara merepresentasikan suatu struktur hirarki (*one-to-many*) secara grafis yang mirip sebuah pohon, walaupun pohon tersebut hanya tampak sebagai kumpulan node-node dari atas ke bawah.

Suatu struktur data yang tidak linier yang menggambarkan hubungan yang hirarkis (one-to-many) dan tidak linier antara elemen-elemennya.

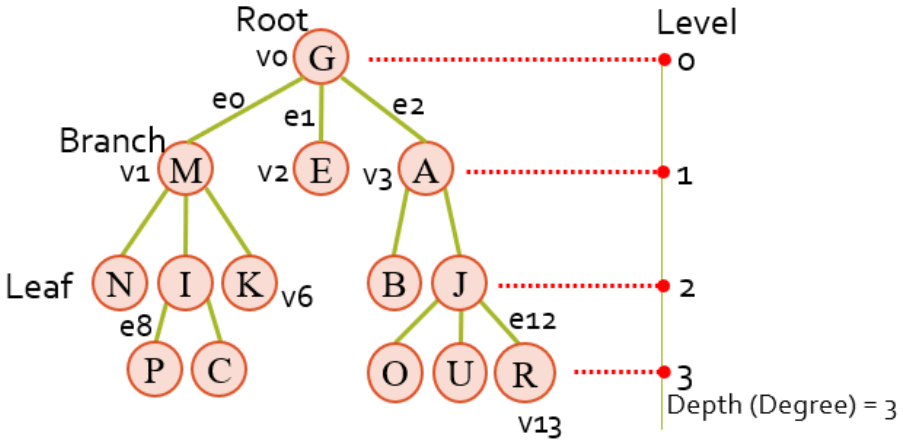
PENGERTIAN TREE (lanj.)

Tree digunakan di banyak bidang informatika termasuk sistem operasi, pemrosessan Bahasa alami, penggalian data web, grafika, sistem database dan jaringan komputer.

Struktur data tree mempuyai suatu *root* (akar), *branches* (cabang) dan *leaves* (daun-daun).

- Pada tree berbentuk list of lists (list di dalam list), kita menyimpan nilai dari node root sebagai elemen pertama dari list.
- Elemen kedua dari list akan merepresentasikan sub-tree kiri.
- Elemen ketiga adalah list lain yang mewakili sub-tree kanan.

TREE (pohon)



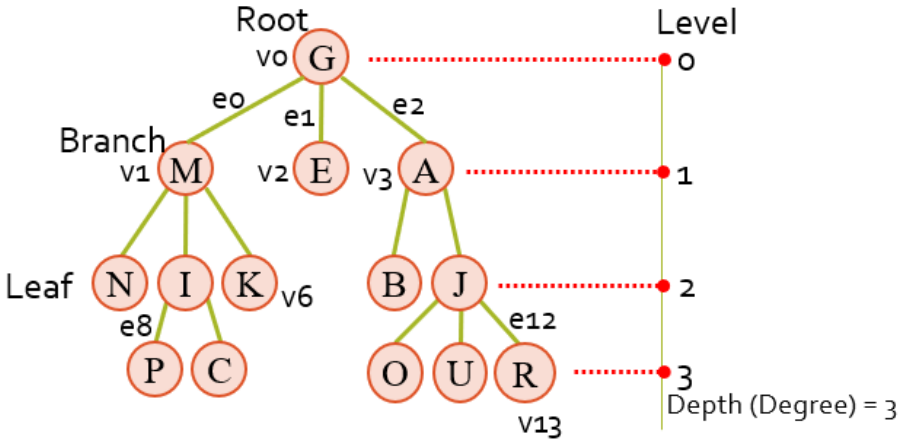
Tree (pohon) merupakan kumpulan dari Simpul (Node/ Vertex) dan Busur (Edge), dimana salah satu Simpul merupakan ***Root*** (akar) dan simpul lainnya membentuk suatu sub Tree.

V = {v0, v1, v2….. v13} dan
E = {e0, e1, e2….. e12}

m = n – 1;
m : jumlah Busur (Edge)
n : jumlah Simpul (Node/ Vertex)

Tentang ilustrasikan struktur data ini, memperlihatkan tree sederhana beserta list yang mewakilinya.

TREE (pohon)



- ✓ Simpul G merupakan Root (akar) yang berada pada Level 0 dan tidak punya Superordinat.
- ✓ Leaf (daun) adalah Simpul yang tidak punya Sub-ordinat (Simpul E, N, K, B, P, C, O, U, R)
- ✓ Simpul M merupakan Superordinat dari Simpul N, I dan K.
- ✓ Simpul M mempunyai Subordinat Simpul N, I dan K.
- ✓ Degree sebuah Tree merupakan jumlah Sub-ordinat terbanyak yg dimiliki suatu Simpul (contoh: Degree = 3)
- ✓ Degree sebuah Simpul merupakan jumlah Sub-ordinat yang dimiliki suatu Simpul (*Degree* Simpul G = 3; Degree Simpul P = 0).

BINARY TREE (pohon)

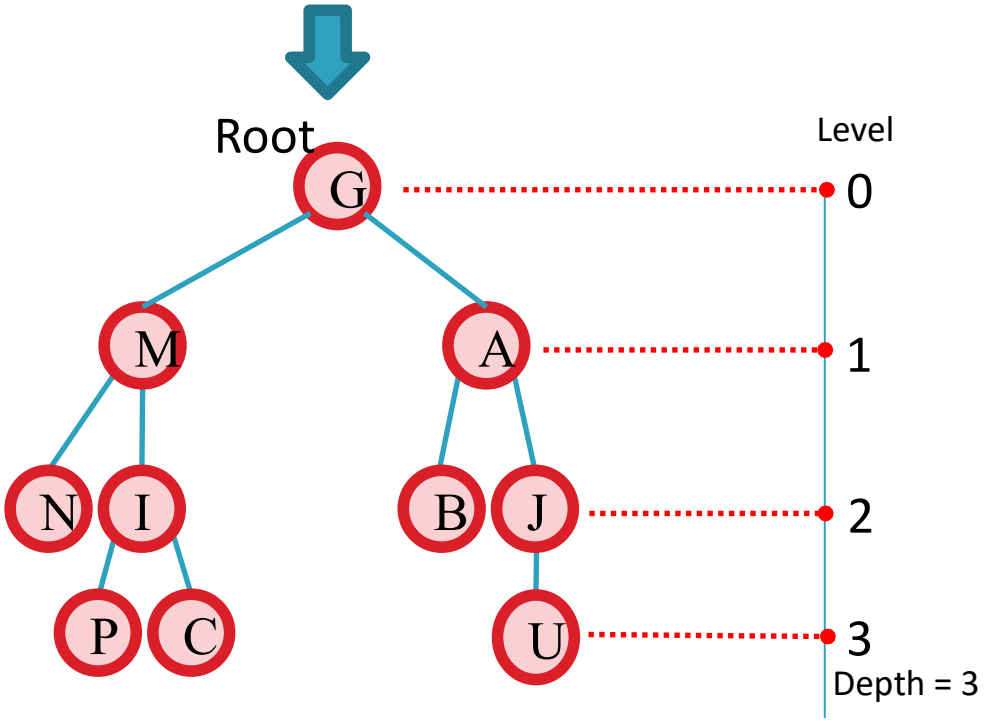
BINARY TREE

Tree dengan syarat bahwa tiap node hanya boleh memiliki maksimal dua sub-pohon dan kedua sub-pohon harus terpisah. Kelebihan struktur Binary Tree :

- Mudah dalam penyusunan algoritma sorting
- *Searching* data relatif cepat
- *Fleksibel* dalam penambahan dan penghapusan data

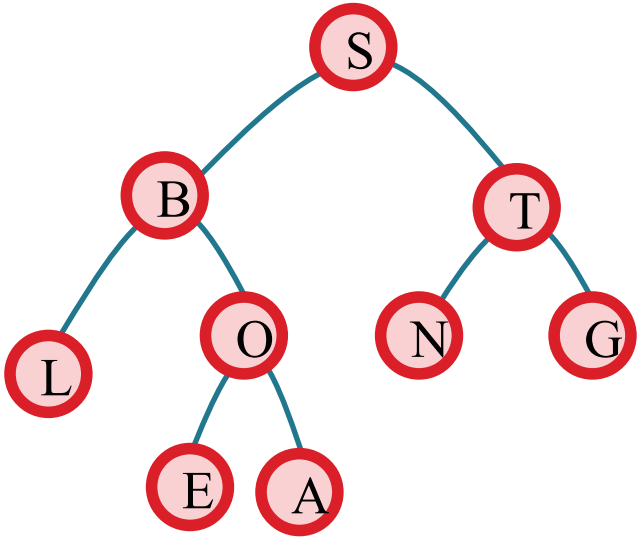
BINARY TREE

Disebut Binary Tree (pohon biner) karena setiap Simpul paling banyak memiliki 2 Simpul Subordinat.



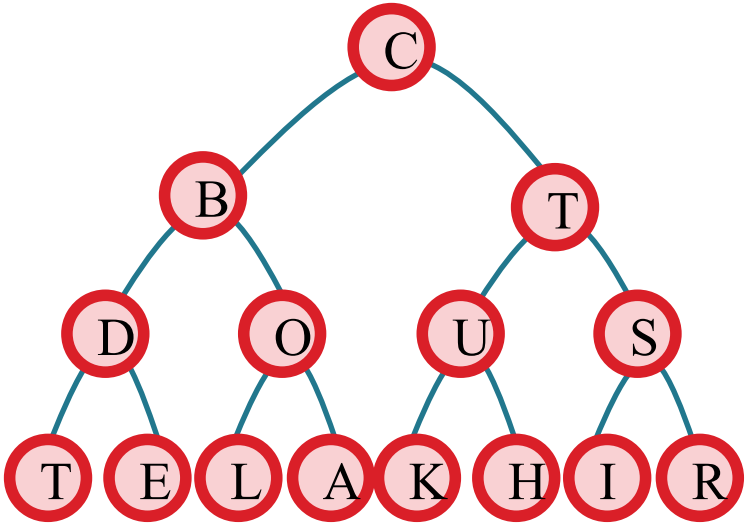
CONTOH BINARY TREE

Strickly Binary Tree: pohon biner yang semua simpulnya mempunyai sub- ordinat lengkap, kecuali simpul Leaf.



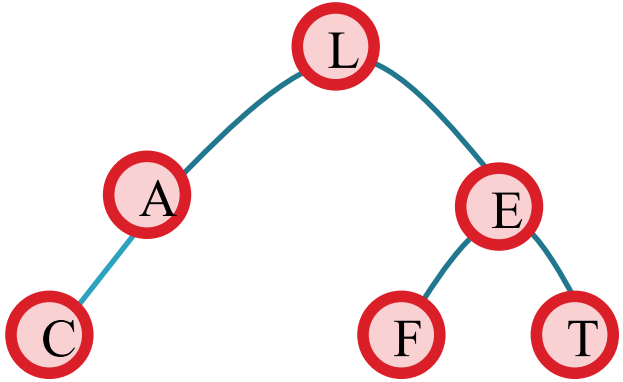
CONTOH BINARY TREE

Complete Binary Tree: pohon biner yang semua Leaf-nya berada pada level terakhir.



CONTOH BINARY TREE

Almost Complete Binary Tree: selain Leaf-nya, terdapat simpul yang memiliki subor-dinat tidak lengkap (***Left & Right***).



CONTOH BINARY TREE

Kita telah melihat dua cara mendapatkan pasangan key-value dalam suatu koleksi. Koleksi ini mengimplementasikan ADT map. Dua implementasi dari ADT map tersebut adalah binary search pada suatu list dan hash tables. ***Binary search*** tree adalah cara lain untuk memetakan dari suatu key ke nilainya.

OPERASI BINARY TREE

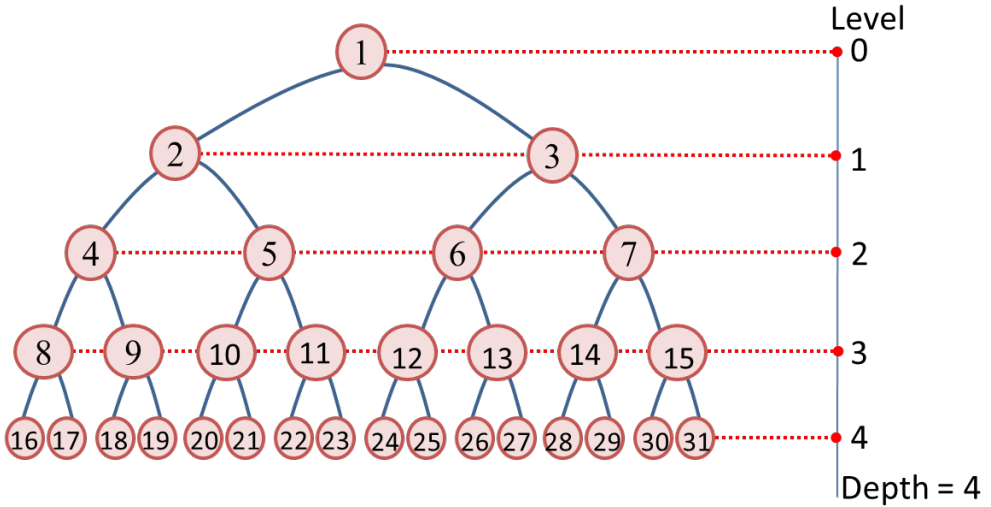
Berikut ini adalah beberapa operasi yang dapat dilakukan terhadap map :

map **()** untuk membuat suatu map baru yang kosong.
Put **(key,val)** menambahkankan suatu pasangan key-value baru ke map. Jika key telah ada dalam map maka ganti nilai lama dengan nilai yang baru.
get (key) diberikan suatu key, kembalikan nilai yang disimpan di dalam map atau *None* jika tidak.
del menghapus pasangan key-value dari map menggunakan pernyataan berbentuk del map[key].
len() mengembalikan jumlah pasangan key-value yang disimpan dalam map.
in mengembalikan True untuk suatu pernyataan dari bentuk key in map, jika key yang diberikan ada dalam map.

BINARY TREE

Menentukan jumlah Simpul:

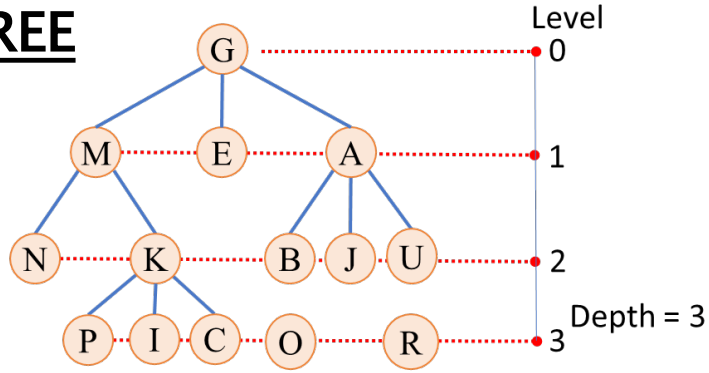
Level	Jumlah Maksimum Simpul (pada level yg sama)	Jumlah Maksimum Seluruh Simpul
0	1	1
1	2	3
2	4	7
3	8	15
4	16	31
5	32	63
6	64	127
n	**2^n**	**{2^(n+1)}-1**



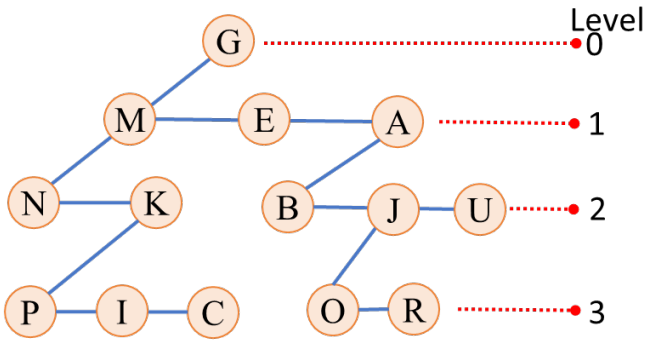
KONVERSI TREE MENJADI BINARY TREE

KONVERSI TREE MENJADI BINARY TREE

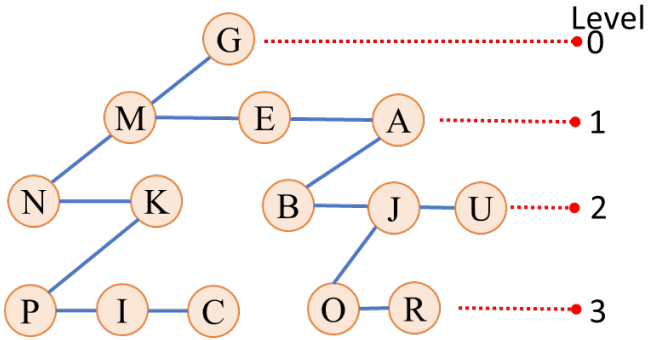
TREE



Langkah 1:



KONVERSI TREE MENJADI BINARY TREE

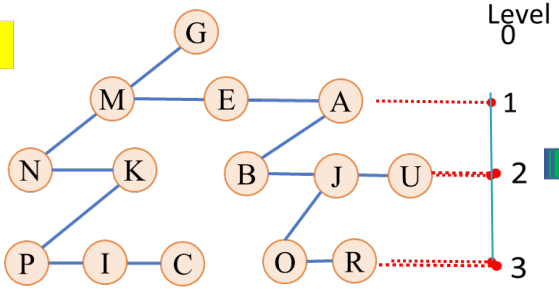


Langkah 1: (proses dimulai dari Simpul atas ke bawah, lalu dari Simpul kiri ke kanan)

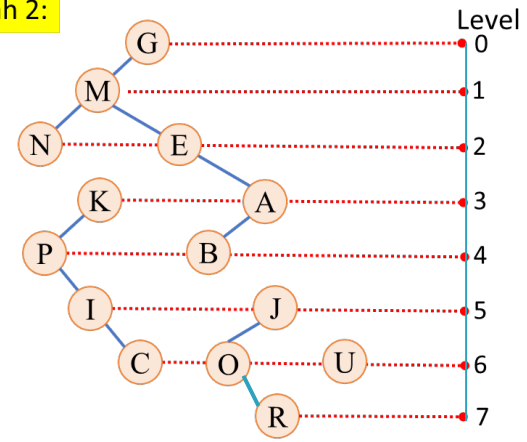
- ✓ Proses konversi dilakukan dengan memandang Simpul G mempunyai 'anak' M, E dan A (karena M berada pada posisi paling kiri, maka M dianggap 'anak tertua' dari G).
- ✓ Lalu ubah struktur Edge dengan menghubungkan G hanya dengan M disebelah bawah bagian kiri G (G sebagai Root tidak punya 'saudara'/ hanya satu Edge terhubung dengan G).
- ✓ E dan A sebagai 'saudara' M, diletakkan disebelah kanan M.
- ✓ M mempunyai anak N dan K, maka N dihubungkan lebih dahulu di posisi bawah kiri dari M.
- ✓ K, sebagai saudara N, dihubungkan sejajar dengan N disebelah kanan, dst……

KONVERSI TREE MENJADI BINARY TREE

Langkah 1:



Langkah 2:



Langkah 2:

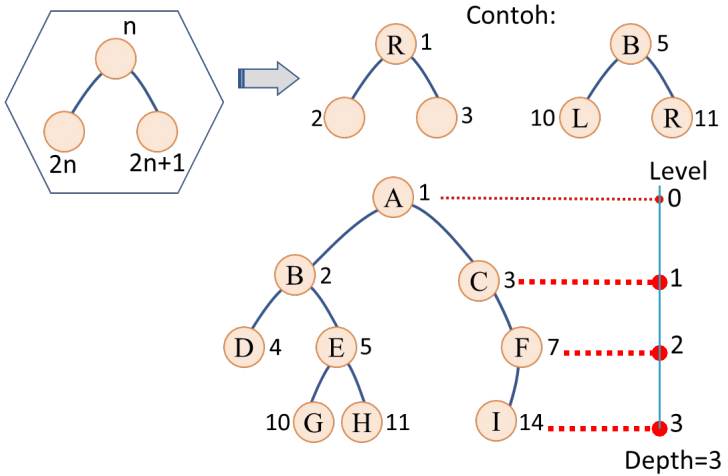
- ✓ Setiap Simpul diposisi kiri yang mempunyai ‘saudara’ (garis sejajar), maka Simpul ‘saudara’ nya dijadikan Subordinat dari Simpul ‘saudara tertuanya’.

PENOMORAN SIMPUL BINARY TREE

PENOMORAN SIMPUL BINARY TREE

- Root diberi nomor 1.
- Bila sebuah Simpul bernomor n, maka Subordinat kiri bernomor 2n dan Subordinat kanan bernomor 2n+1.

Dengan mengetahui nomor setiap Simpul, maka sebuah Binary Tree dapat direpresentasikan menjadi sebuah Array satu dimensi.
Jumlah Elemen = 2^(Depth+1).



D=3,
maka jumlah elemen = 2^(3+1) = 16

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	A	B	C	D	E		F			G	H			I	

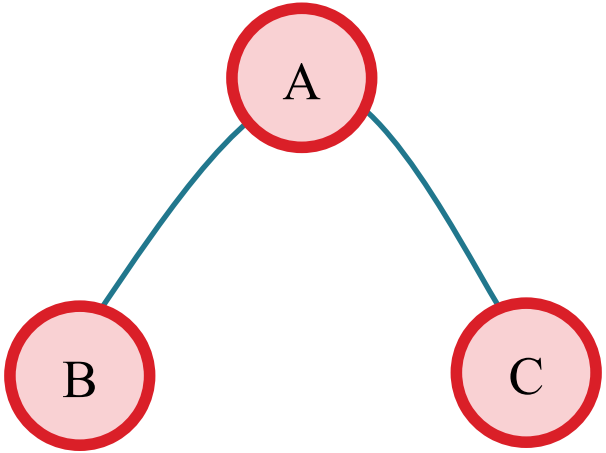
TRAVERSAL BINARY TREE

TRAVERSAL BINARY TREE

Traversal Binary Tree maksudnya adalah membaca Simpul-Simpul dengan urutan/ susunan tertentu. Ada 3 macam Traversal Binary Tree, yaitu:

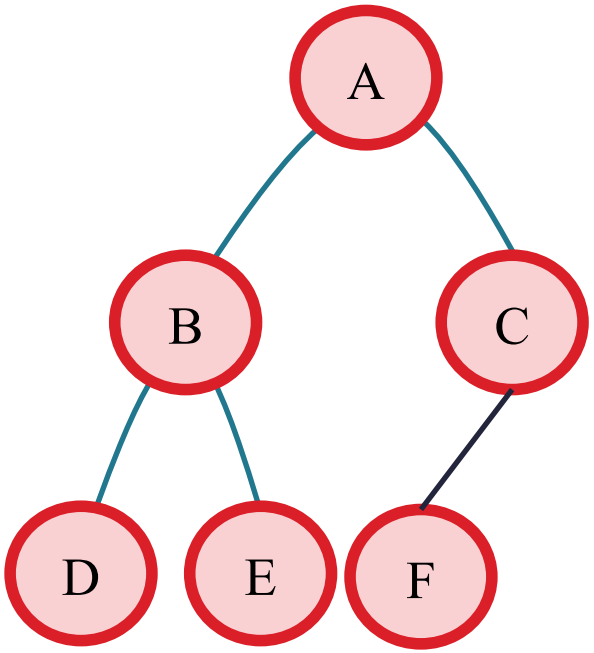
No	Nama Traversal	Proses Traversal
1	Preorder	**Root**, Left, Right
2	Inorder	Left, **Root**, Right
3	Postorder	Left, Right, **Root**

Contoh 1:



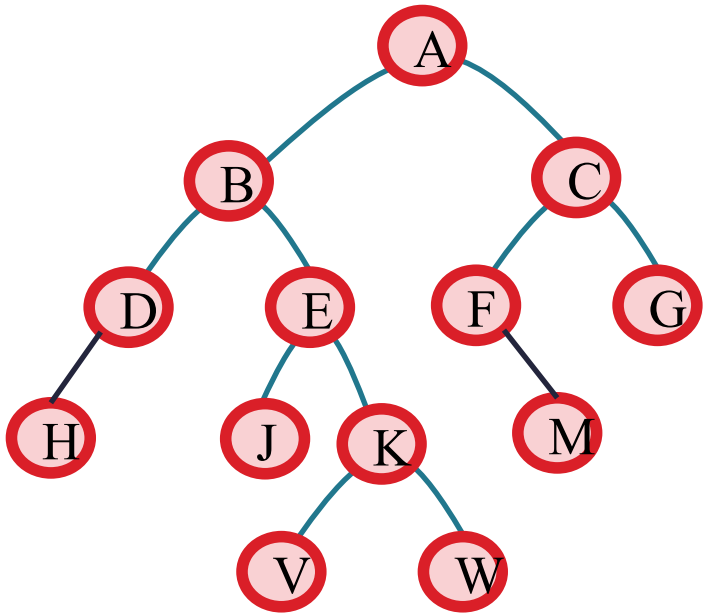
Preorder	ABC
Inorder	BAC
Postorder	BCA

Contoh 2 :



Preorder	ABDECF
Inorder	DBEAFC
Postorder	DEBFCA

Contoh 3 :



Preorder	ABDHEJKVWCFMG
Inorder	HDBJEVKWAFMCG
Postorder	HDJVWKEBMFGCA

ISTILAH DALAM TREE

Predecesor	Node yang berada diatas node tertentu.
Successor	Node yang berada dibawah node tertentu.
Ancestor	Seluruh node yang terletak sebelum node tertentu dan terletak pada jalur yang sama
Descendant	Seluruh node yang terletak setelah node tertentu dan terletak pada jalur yang sama
Parent	Predecessor satu level di atas suatu node.
Child	Successor satu level di bawah suatu node.
Sibling	Node-node yang memiliki parent yang sama
Subtree	Suatu node beserta descendantnya.
Size	Banyaknya node dalam suatu tree
Height	Banyaknya tingkatan dalam suatu tree
Root	Node khusus yang tidak memiliki predecessor.
Leaf	Node-node dalam tree yang tidak memiliki successor.
Degree	Banyaknya child dalam suatu node

SELESAI