TREE (POHON)

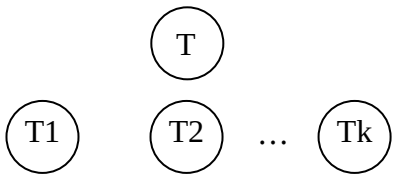
Tree/pohon merupakan struktur data yang tidak linear/non linear yang digunakan terutama untuk merepresentasikan hubungan data yang bersifat hierarkis antara elemen-elemennya.

Definisi tree : “Kumpulan elemen yang salah satu elemennya disebut dengan root (akar) dan sisa elemen yang lain disebut sebagai simpul (node/vertex) yang terpecah menjadi sejumlah himpunan yang tidak saling berhubungan satu sama lain, yang disebut subtree/cabang”.

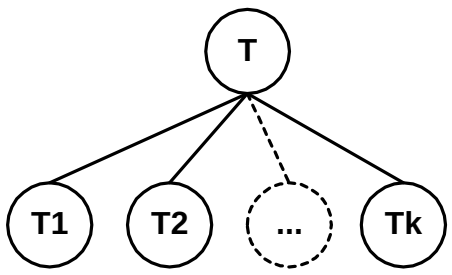
Ilustrasi :

Simpul Tunggal : T

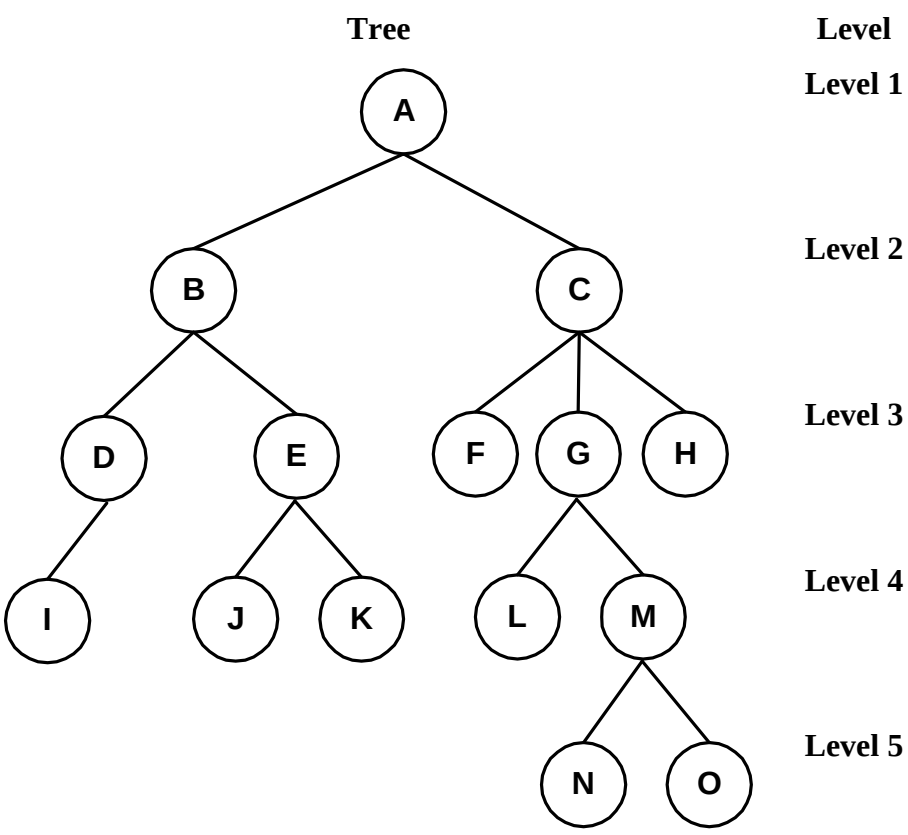
Subtree : T1, T2, ..., Tk



Tree yang terbentuk dari simpul T (root) dan Simpul T1, T2, … Tk adalah :



Perhatikan contoh tree (Gambar 1)



Gambar 1. Contoh Tree

Istilah-istilah objek tree, adalah :

- Simpul adalah elemen tree yang berisi informasi / data dan penunjuk pencabangan.
- Tingkat/level suatu simpul ditentukan dari akar (root), sebagai level 1. Apabila simpul dinyatakan sebagai tingkat N, maka simpul-simpul yang merupakan anaknya berada pada tingkat N+1.
- Derajat/degree menyatakan banyaknya anak/turunan di simpul tersebut.
  Contoh : Simpul A memiliki derajat 2 (B dan C), simpul yang memiliki derajat 0 (nol) disebut leaf (daun) seperti : I, J, K, L, N, dan O.
- Tinggi (height) atau kedalaman (depth) suatu tree adalah tingkat maksimum dari tingkat dalam tree tersebut dikurangi 1.
  Contoh dalam tree di atas, mempunyai depth 4.
- Ancestor suatu simpul adalah semua simpul yang terletak dalam satu jalur dengan simpul tersebut, dari akar sampai simul yang ditinjaunya.
  Contoh Ancestor L adalah A,C dan G.
- Predecessor adalah simpul yang berada di atas simpul yang ditinjau.
  Contoh : Predecessor D adalah B.
- Successor adalah simpul yang berada di bawah simpul yang ditinjau.
  Contoh : Successor D adalah I.
- Descendant adalah seluruh simpul yang terletak sesudah simpul tertentu dan terletak pada jalur yang sama.
  Contoh : Descendant E adalah J dan K.
- Sibling adalah simpul-simpul yang memiliki parent yang sama dengan simpul yang ditinjau.
  Contoh : Sibling J adalah K
- Parent adalah simpul yang berada satu level di atas simpul yang ditinjau.
  Contoh : Parent J adalah E

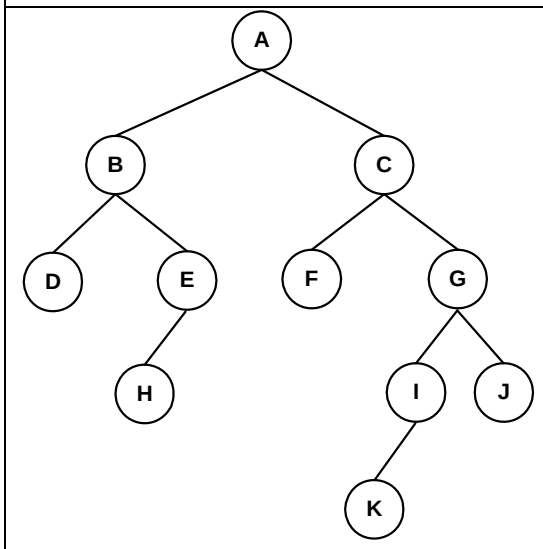
BINARY TREE

Sebuah pohon biner T dapat didefinisikan sebagai sekumpulan terbatas dari elemen-elemen yang disebut node/simpul dimana :

a. T dikatakan kosong (disebut NULL Tree/pohon NULL atau empty tree/pohon kosong)
b. T terdiri dari sebuah node khusus yang dipanggil R, disebut root dari T dan node-node T lainnya membentuk sebuah pasangan terurut dari binary tree T1 dan T2 yang tidak berhubungan yang kemudian dipanggil subtree kiri dan subtree kanan.

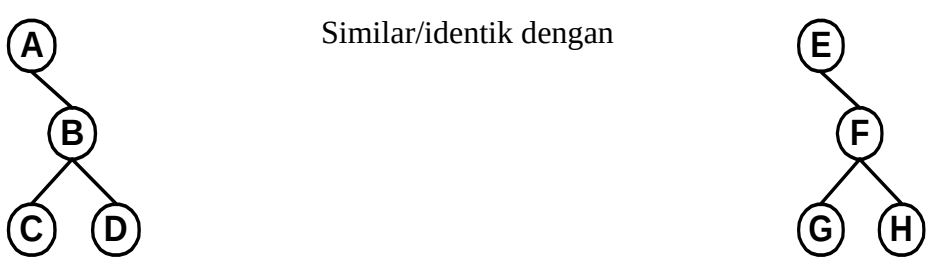
Jika T1 tidak kosong maka rootnya disebut successor kiri dari R dan jika T2 tidak kosong, maka rootnya disebut successor dari R.

Contoh :

Tree (T)	Keterangan
	▪ Root dari T adalah simpul A. ▪ B adalah successor kiri dari simpul A. ▪ C adalah successor kanan dari A ▪ Subtree kiri dari root A adalah simpul B, D, E, dan H ▪ Subtree kanan dari root A adalah simpul C, F, G, I, J dan K

Gambar 2. Contoh Binary Tree

Dua buah binary tree dikatakan similar/identik jika tree tersebut memiliki struktur/bentuk yang sama . Contoh :



Gambar 3. Contoh Binary Tree yang Similar

Terminologi

R adalah sebuah simpul pada T dengan successor kiri S1, dan successor kanan S2, maka R disebut parent dari S1 dan S2. S1 disebut anak kiri (left child) dari R, dan S2 adalah anak kanan (right child) dari R. S1 dan S2 dikatakan sibling (bersaudara).

Derajat tertinggi dari sebuah simpul binary tree adalah 2.

Banyaknya simpul maksimum pada tingkat/level N adalah $2^{(N-1)}$, sehingga maksimum simpul sampai tingkat ke-N adalah

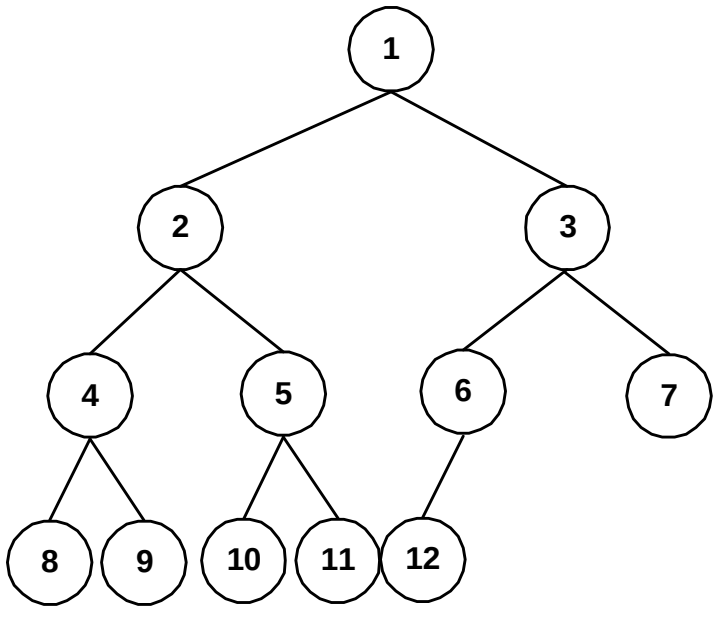
$$MaksSimpul : \sum_{i=1}^{n} 2^{(i-1)} = 2^{N} - 1$$

sehingga jika binary tree lengkap bertingkat 5 maka banyaknya simpul adalah 2^5-1 = 31, terdiri dari 16 leaf (daun) dan banyaknya simpul yang bukan daun termasuk akar adalah 15.

Jenis-Jenis Binary Tree

1. Complete Binary Tree

   Suatu binary tree T akan disebut complete/lengkap jika semua levelnya memiliki child 2 buah kecuali untuk level paling akhir. Tetapi pada akhir level setiap leaf/daun muncul terurut dari sebelah kiri, seperti terlihat pada Gambar 4.



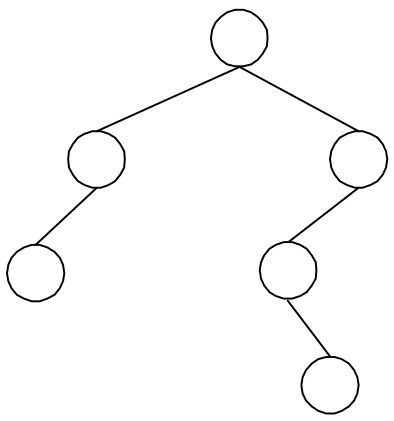
Gambar 4. Contoh Complete Tree T

Sebuah binary tree yang lengkap dapat diberi label dengan sebuah bilangan bulat dari posisi kiri ke kanan. Dengan pemberian label ini, seseorang dapat dengan mudah menentukan child dan parent dari sebuah node/simpul K dalam sebuah complete tree Tn. Khususnya, left child (anak kiri) dari simpul K dapat diketahui dengan rumus 2 * K, dan right child (anak kanan) dari simpul K dapat diketahui dengan rumus 2 * K+1. Perhatikan di gambar bahwa simpul 5 mempunyai anak 10 (dari 2 * 5) dan 11 (dari 2 * 5 + 1). Sedangkan untuk mencari parent, rumusnya adalah K / 2 sehingga ketika simpul 6 yang diperiksa, itu berarti bahwa parent dari simpul 6 adalah 6 / 2 = 3.

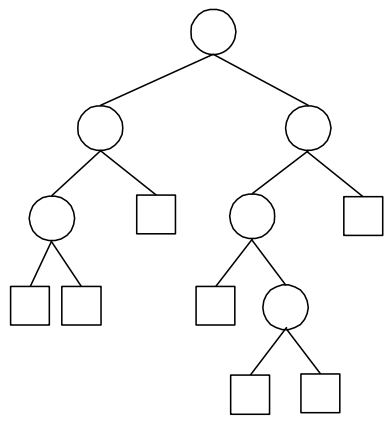
2. Extended Binary Tree : 2-Tree

   Sebuah binary tree dikatakan 2-tree atau extended binary tree jika tiap simpul N memiliki 0 atau 2 anak. Simpul dengan 2 anak disebut dengan simpul internal (internal node), dan simpul dengan 0 anak disebut dengan external node. Kadang-kadang dalam diagram node-node tersebut dibedakan dengan menggunakan tanda lingkaran untuk internal node dan kotak untuk eksternal node.

   Contoh :



Gambar 5. Binary tree T



Gambar 6. Extended 2-tree

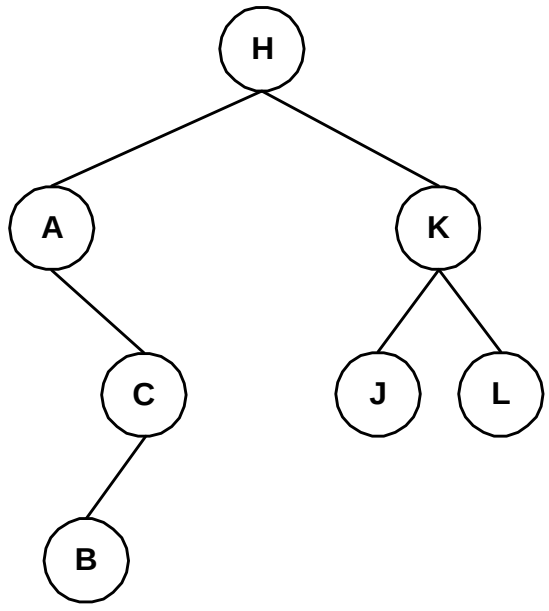
Pembuatan Binary Tree

Pembuatan binary tree lebih mudah menggunakan binary search tree (binary sorted tree) dengan cara : " Jika nilai dari simpul yang akan disisipkan lebih besar dari simpul parent, maka simpul tersebut ditempatkan sebagai subtree kanan. Jika lebih kecil maka simpul baru tersebut disimpan sebagai subtree kiri".

Contoh :

Tree yang akan dibuat adalah : HAKCBLJ

- H dijadikan sebagai root
- A < H : A menjadi subtree kiri H
- K > H : K menjadi subtree kanan H
- C < H → C > A : C menjadi subtree kanan dari A.
- B < H → B > A → B < C : B menjadi subtree kiri dari C.
- L > H → L > K : L menjadi subtree kanan dari K.
- J < H → J < K : J menjadi subtree kiri dari K.

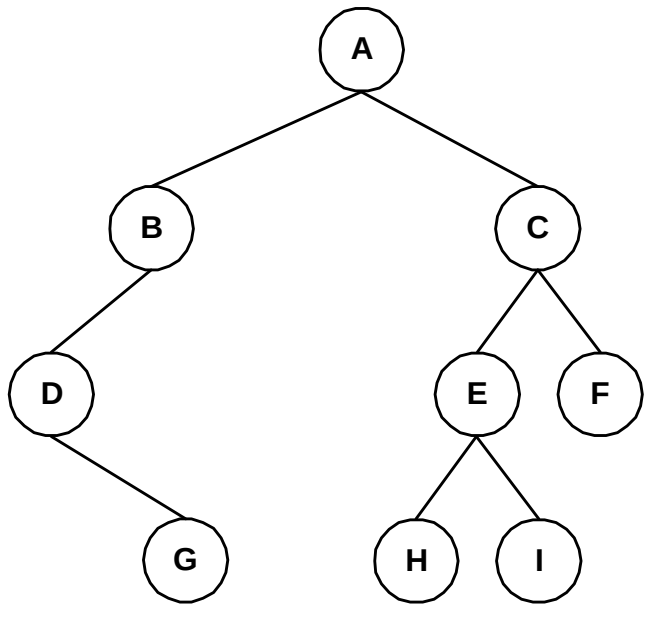


Penelusuran Binary Tree (Traversing Binary Tree)

Ada tiga cara yang standar untuk menelususi sebuah binary tree yaitu :

1. Preorder (Node – Left – Right [NLR])
   - Proses root
   - Telusuri subtree kiri (Left) secara preorder
   - Telusuri subtree kanan (Right) secara preorder
2. Inorder (Left – Node – Right [LNR])
   - Telusuri subtree kiri (Left) secara inorder
   - Proses root
   - Telusuri subtree kanan (Right) secara inorder
3. Postorder (Left – Right – Node [LNR])
   - Telusuri subtree kiri (Left) secara postorder
   - Telusuri subtree kanan (Right) secara postorder
   - Proses root

Contoh :



Secara preorder :

ABDGCEHIF

Secara inorder :

DGBAHEICF

Secara postorder :

GDBHIEFCA

Pembentukan Binary Tree berdasarkan Preorder, Inorder atau Postorder

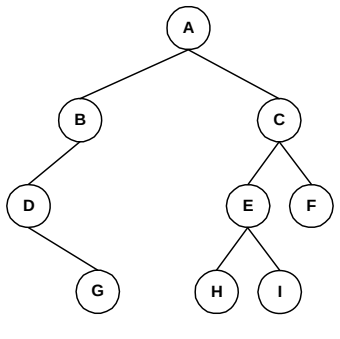
Untuk membentuk suatu binary tree berdasarkan preorder, inorder atau postorder dapat dilakukan dengan syarat menggunakan 2 dari tiga penelusuran tersebut dan salah satunya harus inorder.

Contoh 1: [Menggunakan Preorder dan Inorder]:

Preorder : ABDGCEHIF

Inorder : DGBAHEICF

Caranya adalah :

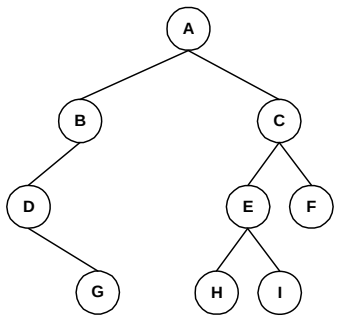


1. Telusuri sepanjang preorder
2. Didapat A, kemudian jadikan sebagai Root
3. Ambil B, lihat di inorder. B berada sebelah kiri dari A, maka B ditulis di kiri dari A.
4. Ambil D, lihat di inorder. D berada di sebelah kiri dari A, tetapi sebelah kiri dari A ada B. Bandingkan posisi D dengan B yang ada di inorder. Ternyata D ada di sebelah kiri dari B sehingga D menjadi subtree B.
5. Ambil G, ikuti cara 4. ternyata G ada disebelah kanan dari D sehingga G menjadi subtree kanan dari G.
6. Ambil C, lihat di inorder, ternyata C ada disebelah kanan dari A, sehingga C menjadi subtree kanan dari A.
7. Ambil E, ternyata sebelah kanan dari A serta sebelah kiri dari C sehingga E menjadi kiri dari C.
8. Ambil H, ternyata ada di sebelah kanan A serta sebelah kanan dari C dan sebelah kiri dari E sehingga H menjadi kiri dari E.
9. Ambil I, ternyata ada di sebelah kanan dari A serta sebelah kiri dari C, dan sebelah kanan E sehingga I menjadi subtree kanan dari E.
10. Ambil F ternyata ada di sebelah kanan C serta sebelah kanan dari C sehingga C menjadi subtree kanan dari C.

Contoh 2 [Menggunakan Postorder dan Inorder]:

Postorder : GDBHIEFCA

Inorder : DGBAHEICF



Cara mirip dengan contoh 1, tetapi penelusuran dilakukan pada postorder secara terbalik (dari paling belakang). Caranya adalah :

1. Ambil dari postorder dapat A, jadikan sebagai root.
2. Ambil C, periksa posisi C di inorder terhadap A, ternyata ada di sebelah kanan. Sehingga C subtree kanan dari A.
3. Ambil F, ternyata ada di sebelah kiri dari A dan sebelah kanan dari C sehingga F menjadi subtree kanan dari C.
4. Ambil E, ternyata ada di sebelah kanan dari A dan sebelah kiri dari C sehingga E menjadi subtree kiri C.
5. Ambil I, ternyata ada di sebelah kanan dari A, sebelah kiri dari C, sebelah kanan dari E sehingga I menjadi subtree kanan dari C.
6. Ambil H, ternyata ada di kanan A, sebelah kiri dari C, sebelah kiri dari E sehingga H menjadi subtree kiri dari E.
7. Ambil B, ternyata B ada disebelah kiri A sehingga B adalah subtree kiri dari A.
8. Ambil D, ternyata ada di kiri A dan dikiri B, sehingga D menjadi subtree kiri dari B.
9. Ambil G, ternyata ada di kiri A, di kiri B dan dikanan dari D sehingga G menjadi subtree kanan dari D.

Latihan-Latihan :

1. Ada sebuah binary tree kosong, kemudian diinsertkan : J R D G T E M H P A F Q
    a. Gambarkan binary tree nya
    b. Tentukan Inorder, Postorder, dan Preordernya
2. Inorder : EACKFHDBG
   Preorder : FAEKCDHGB
    a. Gambarkan binary tree-nya
    b. Tentukan Postordernya