BAB III TIPE DATA ABSTRAK : TREE

Tujuan Pembelajaran Umum :

Setelah menyelesaikan pembahasan ini, mahasiswa :

1. Dapat memberikan definisi dan menyebutkan istilah-istilah dasar struktur data Tree;
2. Dapat mendefinisikan struktur data non binary tree ataupun binary tree statis maupun dinamis;
3. memahami tahapan representasi data dengan acuan type alokasi memory yang dipilihnya;
4. Dapat menyimpulkan manfaat konversi non binary tree menjadi binary tree khususnya dengan adanya keteraturan proses dan kesederhanaan logika proses.

Tujuan Pembelajaran Khusus :

Setelah menyelesaikan pembahasan ini, mahasiswa :

1. Dapat menyebutkan istilah dasar dari sebuah contoh struktur tree dan dapat menguraikan proses traversalnya;
2. Dapat menjelaskan type data yang didefinisikan untuk merepresentasikan Non Binary Tree dalam sebuah array;
3. Dapat menjelaskan type data yang didefinisikan untuk merepresentasikan Non Binary Tree dalam sebuah record dimana salah satu fieldnya bertype pointer;
4. Dapat menuliskan logika proses dan Tracing proses traversal dari struktur data yang didefinisikannya;
5. Dapat menyusun algoritma iteratif maupun rekursif untuk traversal binary tree;
6. Dapat memahami variasi representasi binary tree baik dengan adanya Treaded Binary Tree maupun Lexicographic binary tree.
7. Memahami spesifikasi ADT Non Binary Tree Statis maupun Dinamis;
8. Dapat menggambarkan proses rotasi atau restrukturisasi binary tree jika diperlukan.

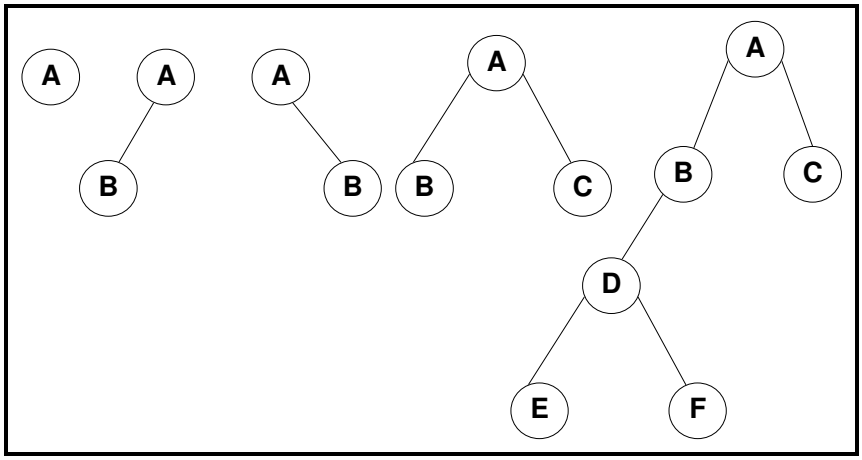
3.1 Istilah Dasar dalam Struktur Data Tree

Definisi Informal :
Merupakan struktur dari sekumpulan elemen dengan salah satu elemennya merupakan akar-nya (*root*), yang berada di puncaknya, dan sisa-nya merupakan bagian-bagian pohon yang terorganisasi dalam suatu susunan hirarki.

Contoh penggunaan : struktur organisasi, hirarki keluarga (silsilah), daftar isi buku, dll.

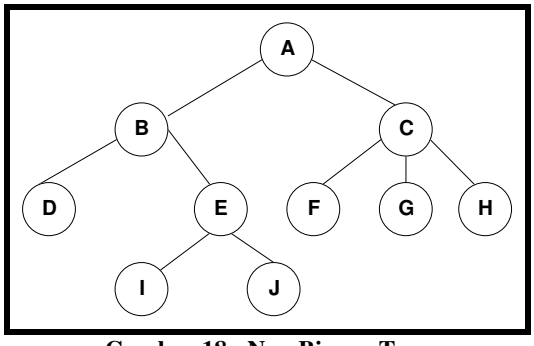
Sifat rekursif *tree* :

- Sebuah simpul tunggal merupakan suatu pohon,



Gambar 17. Representasi Tree Biner

- Bila terdapat simpul n dan beberapa sub pohon T1, T2,...,Tk yang tidak saling berhubungan, yang masing-masing akarnya n1,n2,n3, ..., nk. Dari simpul/sub pohon tersebut dapat dibuat sebuah pohon baru dengan n sebagai root dari simpul/sub pohon n1,n2,...,nk



Gambar 18. Non Binary Tree

Istilah dasar :

1. Simpul/node/vertex adalah elemen-elemen pohon yang mengandung informasi (data) serta menunjuk adanya pencabangan.

2. Akar/Root,

a. Pada contoh di atas, rootnya adalah simpul A

b.Hubungan/relasi antara simpul dianalogikan dengan hubungan keluarga/silsilah

Contoh :

-. A parent (orang tua) dari simpul B dan C

-. Sebaliknya B dan C merupakan sons (anak) dari A.

-. Simpul yang memiliki parent yang sama adalah *siblings* (bersaudara).

Misalnya : B bersaudara dengan C

E bersaudara dengan F, G & H

I bersaudara dengan J

3. Tingkat/level dari simpul.

a. Level dari *root* didefinisikan sebagai tingkat/level 0.

b. Tingkat simpul lainnya didefinisikan sebagai 1+tingkat dari parent simpul tersebut.

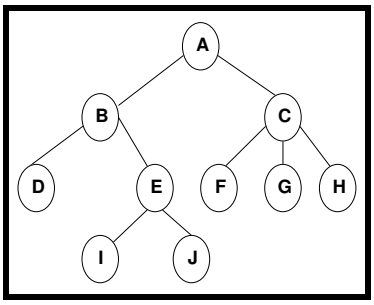
Dari contoh di atas :

Tabel 1. Tingkat dalam Struktur Tree

Tingkat	Simpul-simpul
0	A
1	B,C
2	D,E,F,G,H
3	I,J

4. Depth/Kedalaman dari suatu pohon dinyatakan oleh tingkat yang tertinggi dari simpul yang terdapat pada pohon tersebut. Jadi suatu pohon dikatakan memiliki kedalaman n, bila paling sedikit ada satu simpul yang bertingkat n.

5. Derajat Simpul (*Order*)



Gambar 19. Derajat Simpul

Sebuah pohon dikatakan berorder n, bila memiliki n tahap turunan menuju simpul terendah. Tree di atas memiliki depth = 3(~ n)

Tabel 2. Derajat Non Binary tree

Order	Simpul-simpul
0	I,J,D, F, G
1	E,C
2	B
3	A

Simpul-simpul berderajat/order 0 (nol) adalah simpul yang tidak mempunyai turunan, disebut simpul terminal/ *daun/leaf*.
Simpul yang bukan berderajat 0 disebut simpul non terminal/internal.

3.2 Traversal Non *Binary Tree* :

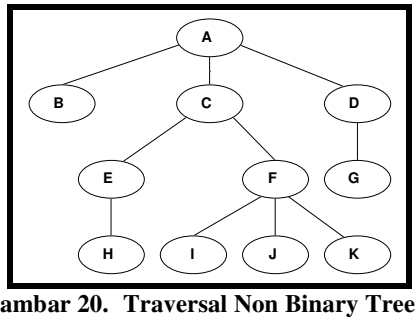
Pengertian : Traversal adalah proses mendatangi setiap simpul dari Non *Binary Tree* secara sistematik masing-masing satu kali.
Traversal ini terbagi menjadi :

1. ***PREORDER,*** yaitu dengan memproses simpul tsb., kemudian proses pada semua anak/cabang yang dimilikinya;

2. ***POSTORDER***, yaitu dengan memproses simpul semua anak/cabang yang dimiliki, kemudian proses simpul tersebut;

3. ***INORDER***, yaitu dengan memproses simpul anak pertama, kemudian proses simpul tersebut dan terakhir proses simpul anak lainnya;

4. ***LEVELORDER***, yaitu dengan memproses simpul ber-dasarkan tingkat dari simpul yang dilalui. Pemrosesan dimulai dari simpul tingkat satu sampai tingkat n (akhir).

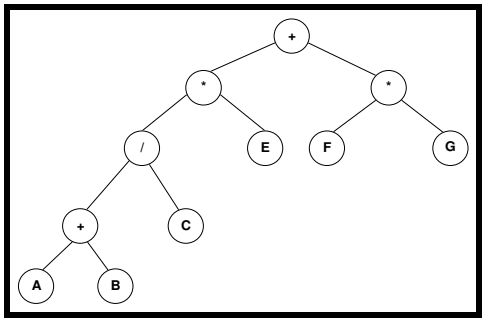


Gambar 20. Traversal Non Binary Tree

Traversal dari *Binary Tree* sering digunakan dalam:

- Suatu ekspresi matematika, pada umumnya suatu notasi matematika ditulis dalam notasi infix, dimana operator seperti +,-,/,* dituliskan di antara dua variabel.
- Pembagian I/O device yang diatur oleh Operating system, biasa menggunakan traversing preorder.

Contoh pemakaian traversal binary tree :
Dari ekspresi matematik berikut : (A+B)/C*E+(F*G) dapat diperoleh binary tree sebagai berikut :



Gambar 21. Pohon Ekspresi

Maka traversal dari binary tree di atas adalah :

- Preorder atau notasi prefix : +*/+ABCE*FG
- Postorder atau notasi postfix : AB+C/E*FG*+
- Inorder atau notasi infix : A+B/C*E+F*G
- Level order : +**/EFG+CAB

Perhatikan bahwa dalam penelusuran ini tidak ditemukan adanya tanda kurung.

3.3 Algoritma Traversal Non Binary Tree

Berikut salah satu alternatif algoritma untuk proses traversal dalam struktur data Non binary tree (alokasi yang digunakan adalah alokasi statis):

1. Traversal PreOrder

```
Procedure PreOrder (X : ArrNB)
Pcur : integer          // Posisi Node Aktif
Resmi : boolean         // Sifat kunjungan resmi
Begin
Pcur ← 1
Write (layer) X[pcur].info
Resmi ← .True.
Repeat
  If X[pcur].fs <> 0 and resmi
  Then
     Pcur ← X[pcur].fs
     Write (layer) X[pcur].info
  Else if X[Pcur].nb <> 0
          Then Pcur ← X[Pcur].nb
             Write (layer) X[pcur].info
             Resmi ← .True.
          Else
             Pcur ← X[pcur].pr
             Resmi ← .false.        //Proses numpang lewat
     Endif
  Endif
Until (X[pcur].pr = 0)
EndProcedure Pre Order
```

2. Algoritma traversal Post Order:

```
Procedure PostOrder (X : ArrNB)
Pcur : integer          // Posisi Node Aktif
Resmi : boolean
Begin
Pcur ← 1
Resmi ← .T.             // Kunjungan resmi
While (pcur <> 0) do
  If X[pcur].fs <> 0 and resmi
  Then
     Pcur ← X[pcur].fs
  Else
     Write (layer) X[pcur].info
     If X[pcur].nb <> 0
        Then
           Pcur ← X[pcur].nb
           Resmi ← .T.
        Else
           Pcur ← X[pcur].pr
           Resmi ← .F.     //Numpang lewat
     Endif
  Endif
EndWhile
EndProcedure Post Order
```

3. Algoritma traversal InOrder:

```
Procedure InOrder (X : ArrNB)
Pcur : integer          // Posisi Node Aktif
Resmi : boolean
```

```
Begin
Pcur ← 1
Resmi ← .T.            // Kunjungan resmi
While (pcur <> 0) do
  If X[pcur].fs <> 0 and resmi
  Then
     Pcur ← X[pcur].fs
  Else
     If (resmi)
     Then Write (layer) X[pcur].info
     Endif

     If (pcur = X[X[pcur].pr].fs)
     Then Write (layer) X[X[pcur].pr].info
     Endif

     If  (X[pcur].nb <> 0)
        Then
           Pcur ← X[pcur].nb
           Resmi ← .T.
        Else
           Pcur ← X[pcur].pr
           Resmi ← .F.     //Numpang lewat
     Endif
  Endif
EndWhile
EndProcedure InOrder
```

4. Algoritma Traversal Level Order:

```
Procedure LevelOrder (X : ArrNB)
Pcur : integer          // Posisi Node Aktif
Q : Queue          // ADT Queue
Begin
Write (layer) X[1].info
pcur ← X[1].fs
Ins_Queue(0)
While (pcur  <> 0) do
Begin
  Write (layer) X[pcur].info
  If X[pcur].fs <> 0
  Then Ins_Queue(X[pcur].fs)
     Pcur ← X[pcur].fs
  End If

  If X[pcur].nb <> 0
  Then
     Pcur ← X[pcur].nb
  Else
     Pcur ← del_queue()
  End If
EndWhile
EndProcedure LevelOrder
```