Pertemuan 11

Binary Tree

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- mendemonstrasikan operasi pada Binary Tree.
- menerapkan Binary Tree pada program aplikasi komputer.

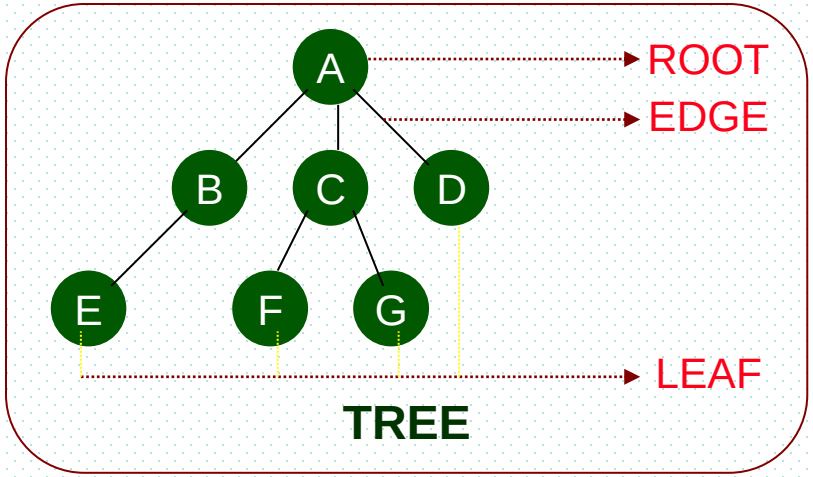
Outline Materi

- Karakteristik
- Terminologi
- Jenis
- Operasi
- Representasi
- Implementasi
- Expression Tree

Karakteristik

- Merupakan tipe Struktur Data dg hubungan One-to-Many / Nested / Hirarki.
- Terdapat 1 node yang unique, yang tidak memiliki predecessor. Node ini disebut ROOT.
- Terdapat satu atau beberapa node yang tidak mempunyai successor. Node-node ini disebut LEAF.
- Setiap node, kecuali ROOT, pasti memiliki 1 predecessor yang unique.
- Setiap node, kecuali LEAF, pasti memiliki 1 atau lebih successor.

Contoh :



Terminologi

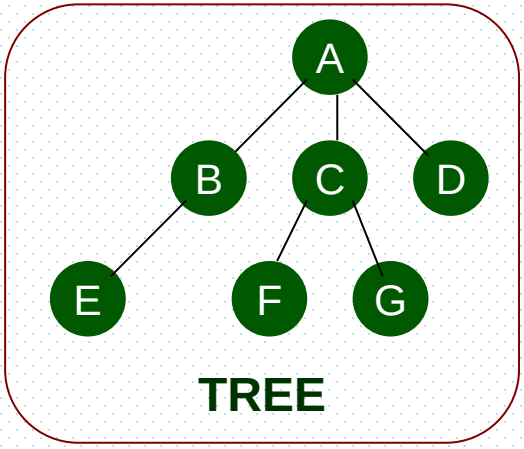
HUBUNGAN PARENT-CHILD

- PARENT adalah predecessor langsung dari suatu node.
- CHILD adalah successor langsung dari suatu node.
- Node-node yang memiliki PARENT yang sama disebut SIBLING.

HUBUNGAN ANCESTOR-DESCENDANT

- ANCESTOR adalah semua node yang berada di atas / sebelum node tertentu, yang terdapat pada path yang sama.
- DESCENDANT adalah semua node yang berada di bawah / setelah node tertentu.

Contoh :



PARENT dari C adalah A

CHILD dari A adalah B, C, D

SIBLING dari F adalah G

ANCESTOR dari F adalah A, C

DESCENDANT dari C adalah F, G

Terminologi (2)

REKURSIVE pd TREE

Tree memiliki sifat yang dinamakan Rekursive,yang dapat didefinisikan sbb:

- Untuk sebuah node tertentu, node tersebut beserta semua descendantnya adalah **SUBTREE** dari parentnya.
- **SUBTREE** tersebut memiliki semua karakteristik dari suatu **TREE.**

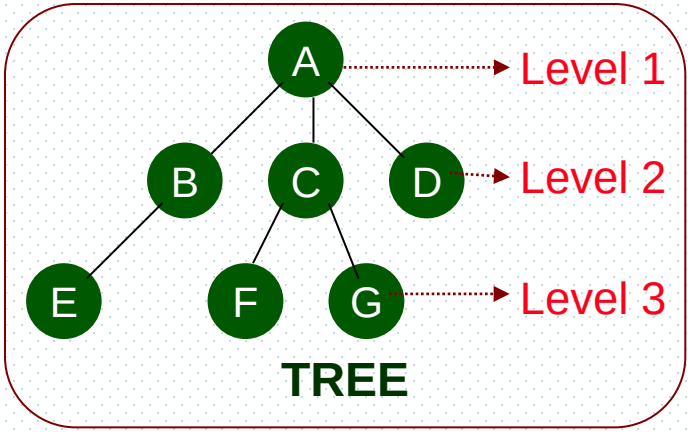
TREE PATH-LENGTH

- Utk SIMPLE TREE, path length adalah sama dengan JUMLAH NODE dikurangi 1.
- Secara umum, path length diambil dari path length dari subtree yang terpanjang.

TREE HEIGHT

- Level : Setiap node pasti akan berada pd tingkat/level tertentu.
- TREE Height : Level tertinggi dari suatu TREE.

Contoh :



PATH-LENGTH = 3 - 1= 2

HEIGHT = 3

Jenis

Complete Binary Tree (HEAP)

Bila semua node, kecuali LEAF memiliki 0 atau 2 children. Subtree dalam Heap dapat mempunyai path length yang berbeda.

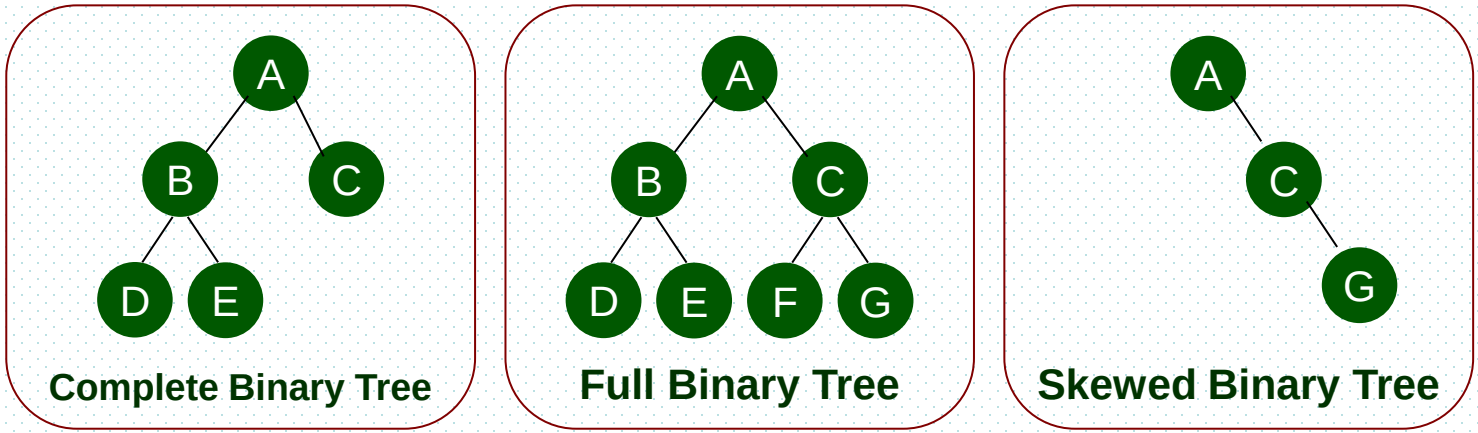
Full Binary Tree

Bila semua node kecuali LEAF memiliki 2 chldren dan semua subtree harus memiliki path length yang sama.

Skewed Binary Tree (Binary Tree Miring)

Bila semua node, kecuali LEAF memiliki hanya 1 child.

Contoh :



Operasi

Traverse (TypeOrd Ord)

- Pre : Tree is not empty.
- Post: Each node in the tree has been proceeded exactly once. The order in which nodes are proceeded depends on the value of Ord.

Insert (TypeElm E, TypePos Pos, Boolean *Fail)

- Pre : Either Pos=root and tree is empty or Pos<>root and tree is not empty.
- Post: E may have been added to the tree, depending on the value of Pos. If E was added, the node that contains E is the current node.

Deletesub ()

- Pre : there is not an empty tree.
- Post: The subtree of T-pre whoser root is c-pre has been removed from the tree. The root node is the current node.

Update (TypeElm E)

- Pre : The tree is not empty.
- Post: The current node’s element has the value of E.

Retrieve (TypeElm *E)

- Pre : The tree is not empty.
- Post: E has value of the standard element in the current node.

Operasi (2)

Characteristics (Status S)

- Post: S contains a size, height, and average length of a path from the root to a leaf node.

Find (TypePos Pos, Boolean *Fail)

- Pre: The tree is not empty.
- Post: The current node is determined by the value of Pos.

Empty ()

- Post: If T-pre is empty then empty is true else empty is false.

Create ()

- Pre: None.
- Post: An empty BT exists.

Clear ()

- Post: The BT is empty.

Traversal

Adalah operasi penelusuran terhadap node-node di dalam binary tree.
Pola-pola traversal :

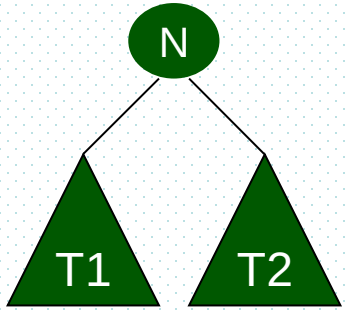
1. PreOrder
2. InOder
3. PostOrder

PreOrder :
- Kunjungi Root (N)
- Kunjungi Left SubTree secara PreOrder (T1)
- Kunjungi Right SubTree secara PreOrder (T2)

InOrder :
- Kunjungi Left SubTree secara InOrder (T1)
- Kunjungi Root (N)
- Kunjungi Right SubTree secara InOrder (T2)

PostOrder :
- Kunjungi Left SubTree secara PostOrder (T1)
- Kunjungi Right SubTree secara PostOrder (T2)
- Kunjungi Root (N)

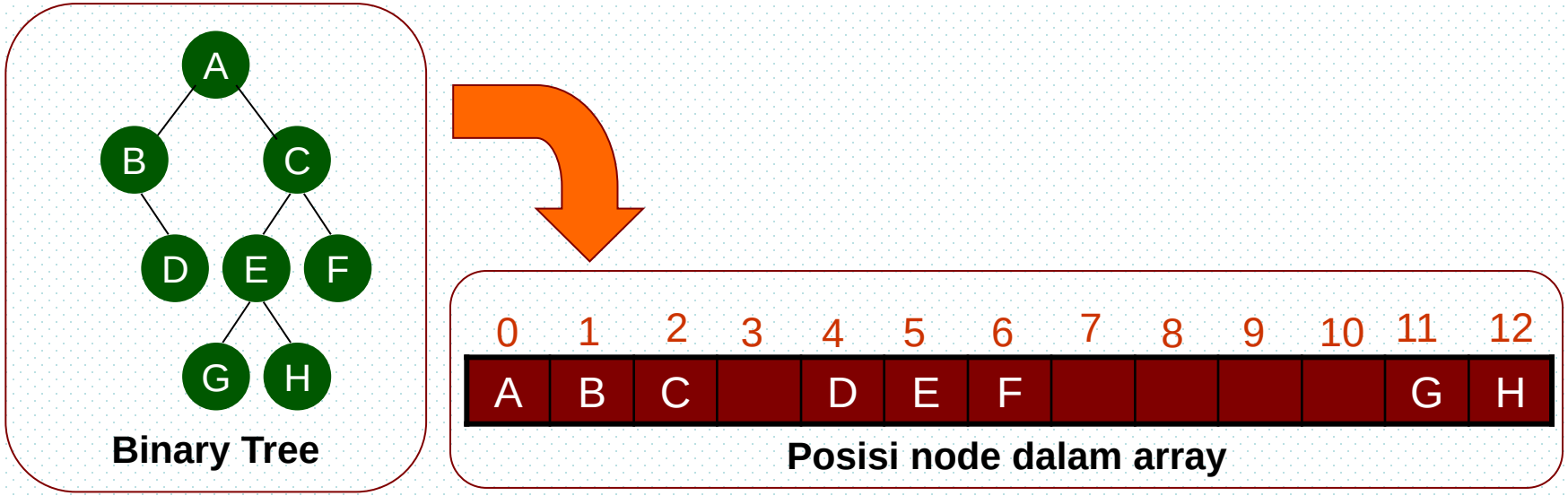
Algoritma Traversal pada Binary Tree



Representasi dg Array

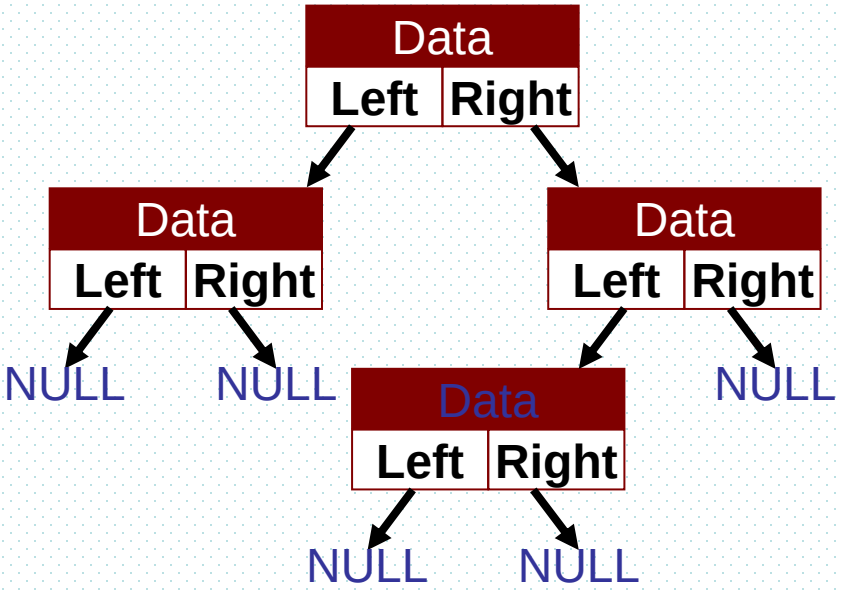
- Indeks pd array menyatakan nomor node
- Indeks 0 adalah Root node
- Indeks Left Child adalah 2p + 1, dimana p adalah indeks parent
- Indeks Right Child adalah 2p + 2
- Indeks Parent adalah (p-1)/2

Contoh :



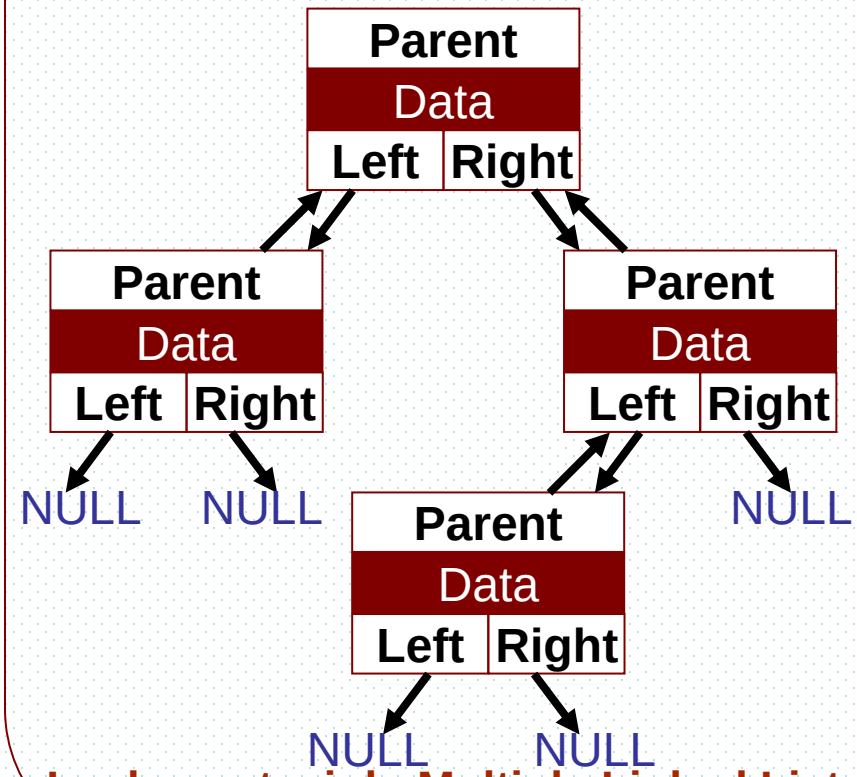
Representasi dg Linked List

```
struct node {
        Elemen_Type Data;
        struct node *Left;
        struct node *Right; }
```



Implementasi dg Double Linked List

```
struct node {
        Elemen_Type Data;
        struct node *Left;
        struct node *Right;
        struct node *Parent;}
```



Implementasi dg Multiple Linked List

Implementasi Traversal

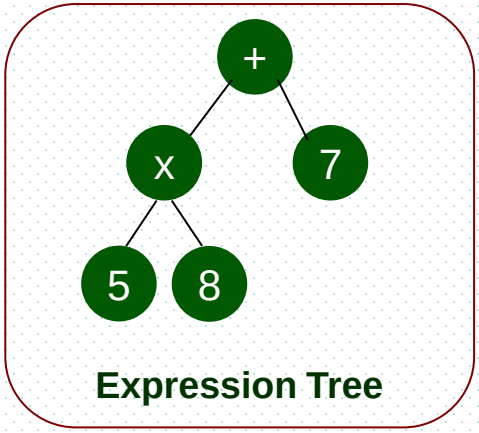
```
PreOrder (ROOT n)
{       if (n != NULL){
                printf(n->info);
                PreOrder(n->Left);
                PreOrder(n->Right);
        }       }
InOrder (ROOT n)
{       if (n != NULL){
                InOrder(n->Left);
                printf(n->info);
                InOrder(n->Right);
        }       }
PostOrder (ROOT n)
{       if (n != NULL){
                PostOrder(n->Left);
                PostOrder(n->Right);
                printf(n->info);
        }       }
```

Implementasi Traversal dg bahasa C

Expression Tree

- Binary Tree yang dapat menampung ekspresi aritmatika, dimana setiap node akan berisi operand dan operator.
- Hasil traversal akan membentuk pola notasi Prefix, Infix, Suffix/Postfix
  - Traversal InOrder akan menghasilkan notasi Infix
  - Traversal PreOrder akan menghasilkan notasi Prefix
  - Traversal PostOrder akan menghasilkan notasi Postfix

Contoh :



Expression Tree

Traversal	Notasi
InOrder	5 x 8 + 7
PreOrder	+ x 5 8 7
PostOrder	5 8 x 7 +

Selesai