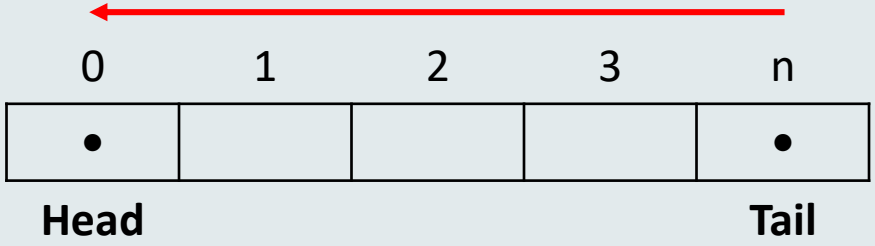
Struktur Data
Tree

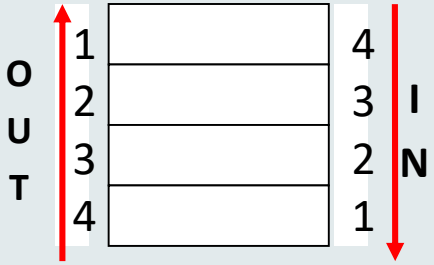
Nisa'ul Hafidhoh, MT

Struktur Data Linier

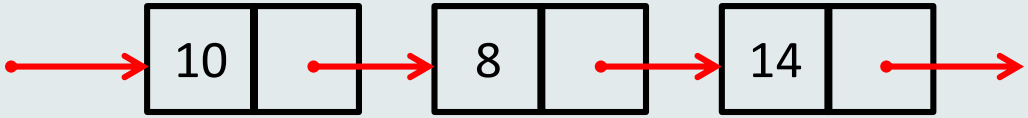
ARRAY



QUEUE



STACK



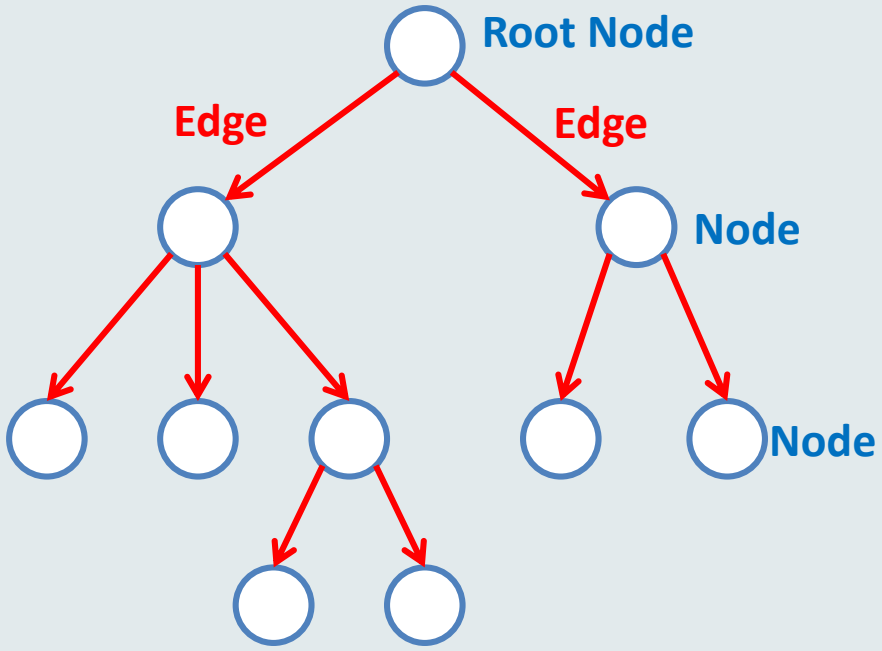
LINKED LIST

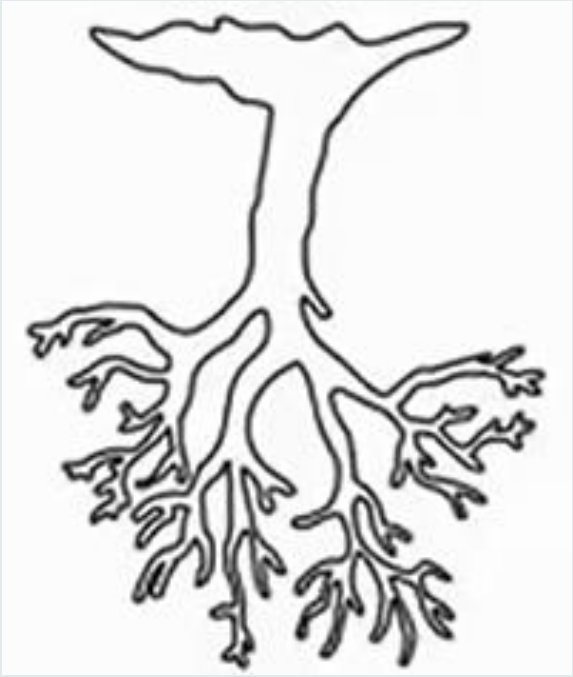
Struktur Tree

- Struktur Tree adalah kumpulan node yang saling terhubung dengan struktur data hirarki (one-to-many)
- Struktur pohon adalah struktur yang penting dalam informatika, yang memungkinkan untuk :
  - mengorganisasi informasi berdasarkan struktur logik
  - memungkinkan cara akses yang khusus terhadap suatu elemen
- Contoh persoalan yang tepat untuk direpresentasi sebagai pohon:
  - pohon keputusan
  - pohon keluarga dan klasifikasi dalam botani
  - pohon sintaks dan pohon ekspresi aritmatika

Tree

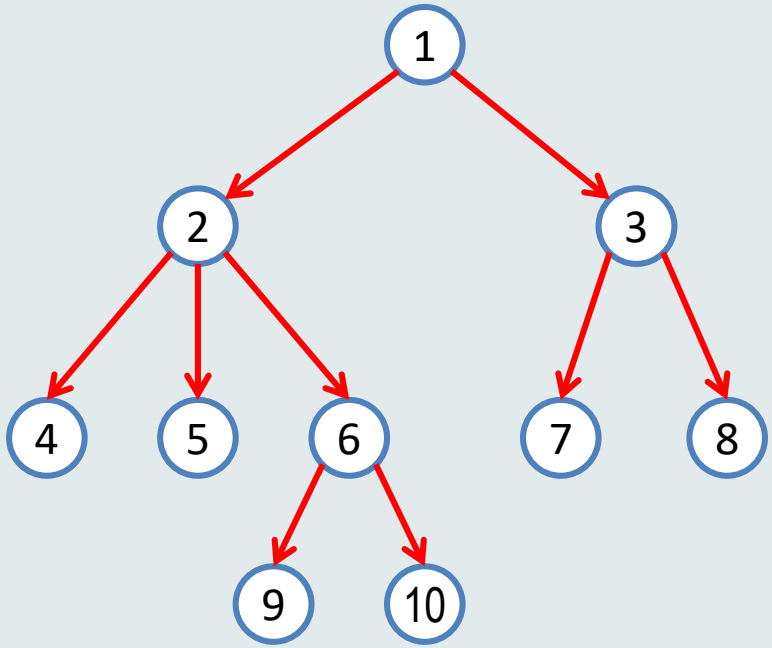
- Tree adalah struktur data yang terdiri dari entitas yang disebut node yang terkait melaui sebuah edge
- Node paling atas disebut dengan root





Tree

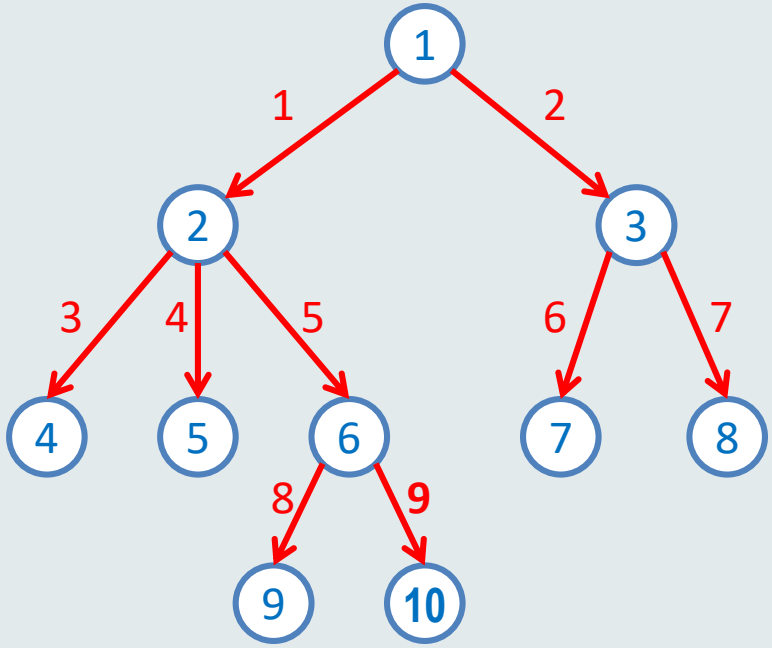
- Node pd posisi yg lebih tinggi disebut parent dan yang lebih rendah disebut children
- Node dengan posisi yang yang sama disebut sibling
- Node dengan posisi paling rendah disebut leaf



- 1 adalah root
- 1 adalah parent dari 2 dan 3
- 2 dan 3 adalah children dari 1
- 2 adalah parent dari 4,5, dan 6
- 4, 5, dan 6 adalah sibling
- 7 dan 8 adalah children dari 3
- 7 dan 8 adalah sibling
- 9 dan 10 adalah leaf

Tree

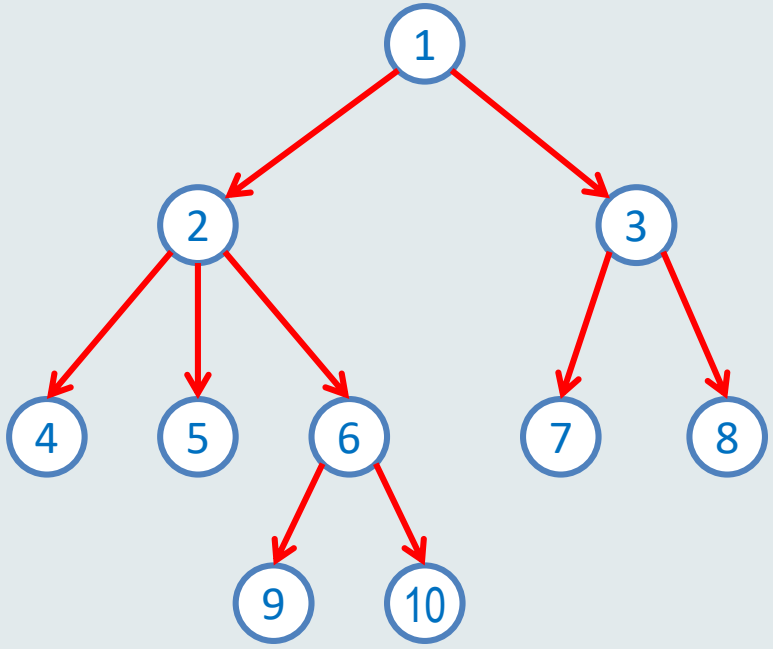
- Tree mempunyai :
  - n node
  - n-1 edge



- Jumlah node adalah 10
- Jumlah edge adalah 9

Tree

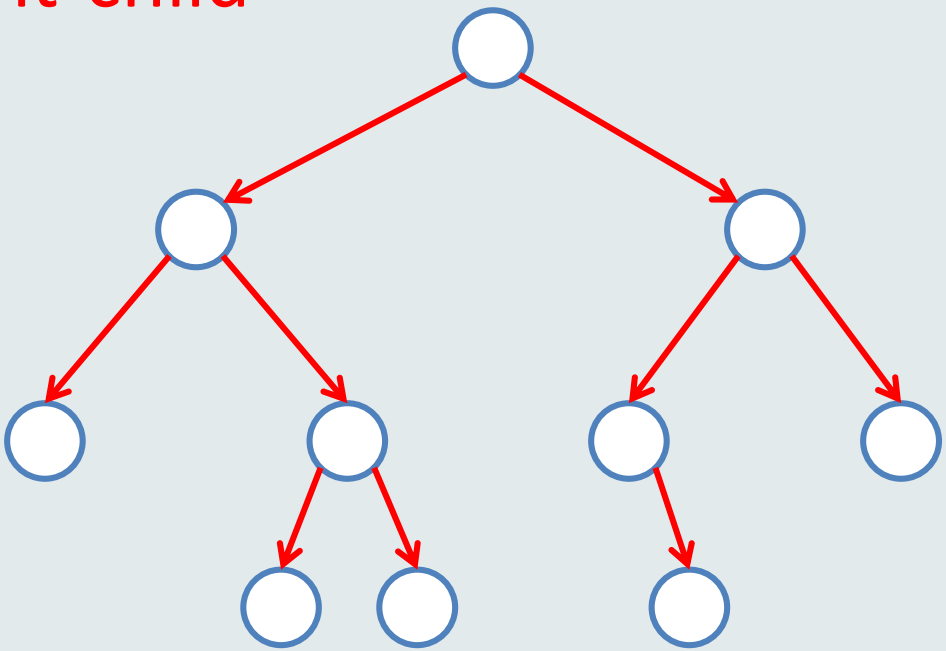
- Depth of Node : jumlah edge dari root ke node
- Height of Node: jumlah edge terpanjang dari node ke leaf
- Height of Tree : height of root node



- Depth of node 1 adalah 0
- Height of node 1 adalah 3
- Depth of node 6 adalah 2
- Height of node 6 adalah 1
- Depth of node 9 adalah 3
- Height of node 9 adalah 0
- Height of tree adalah 3

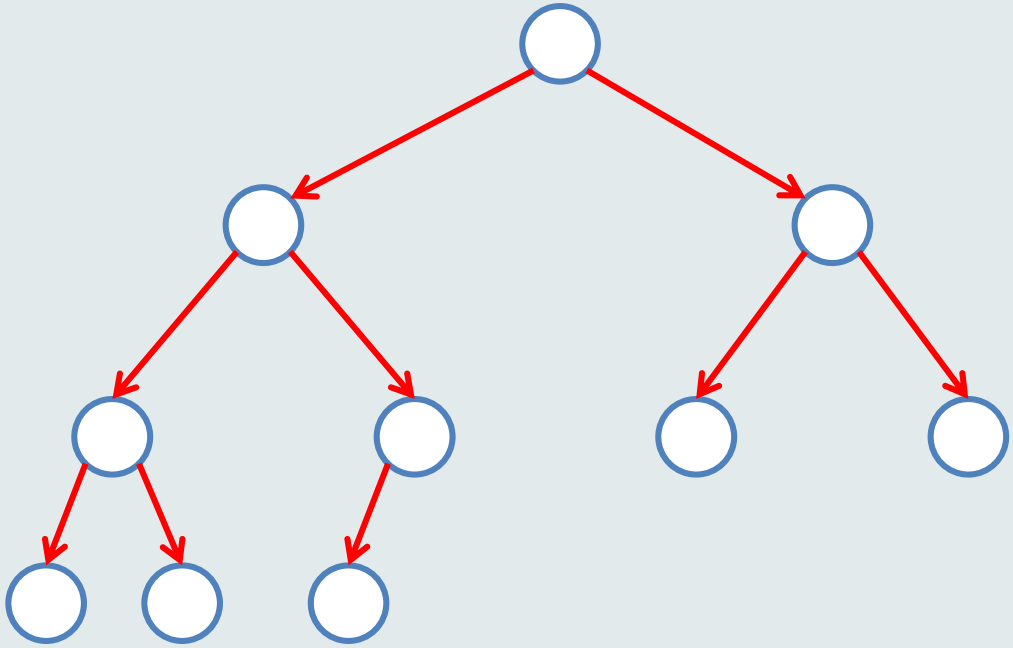
Binary Tree

- Binary Tree adalah tree dimana setiap node mempunyai paling banyak 2 children
- Children dari setiap node disebut left-child dan right-child



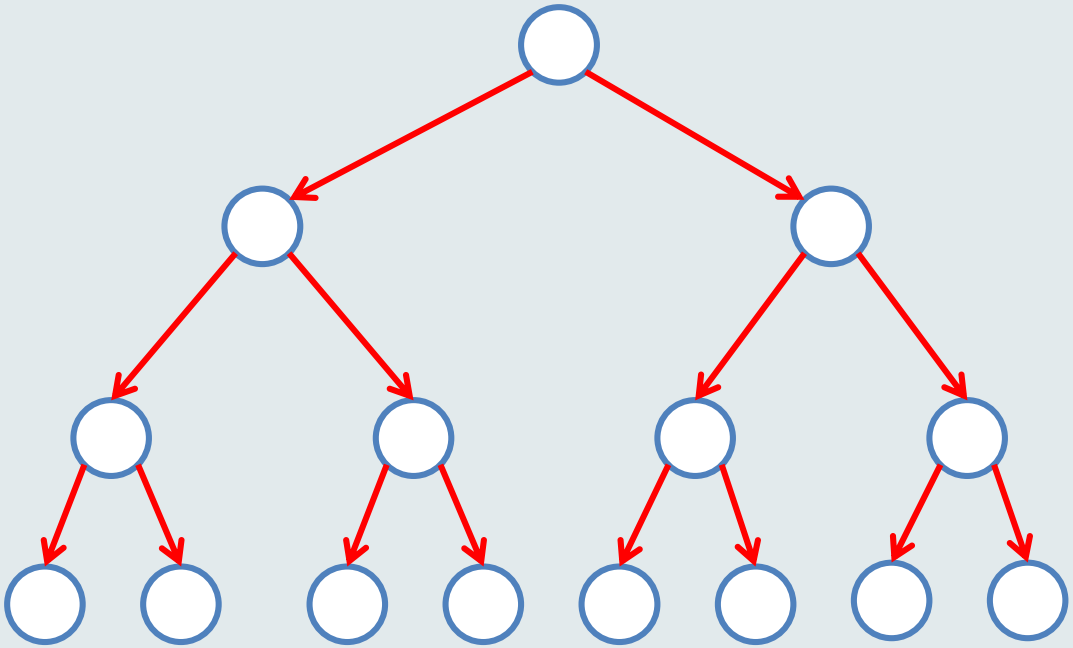
Binary Tree

- Complete Binary Tree
  semua level selain level terakhir pada tree terisi lengkap dan semua node kiri terisi lebih dahulu



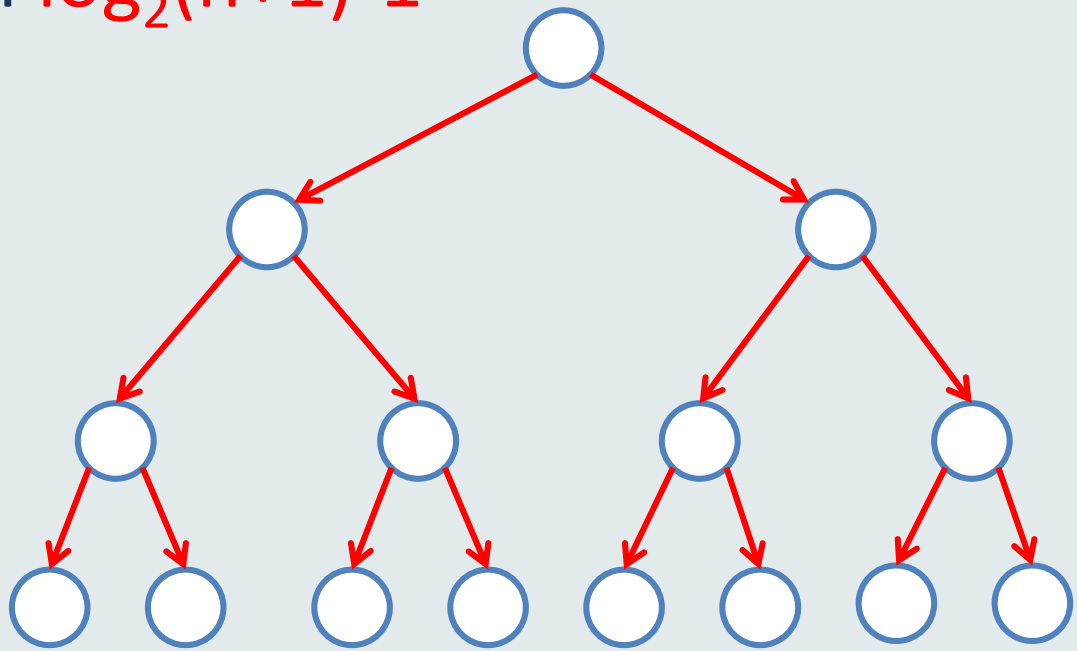
Binary Tree

- Perfect Binary Tree
  semua level pada tree terisi lengkap



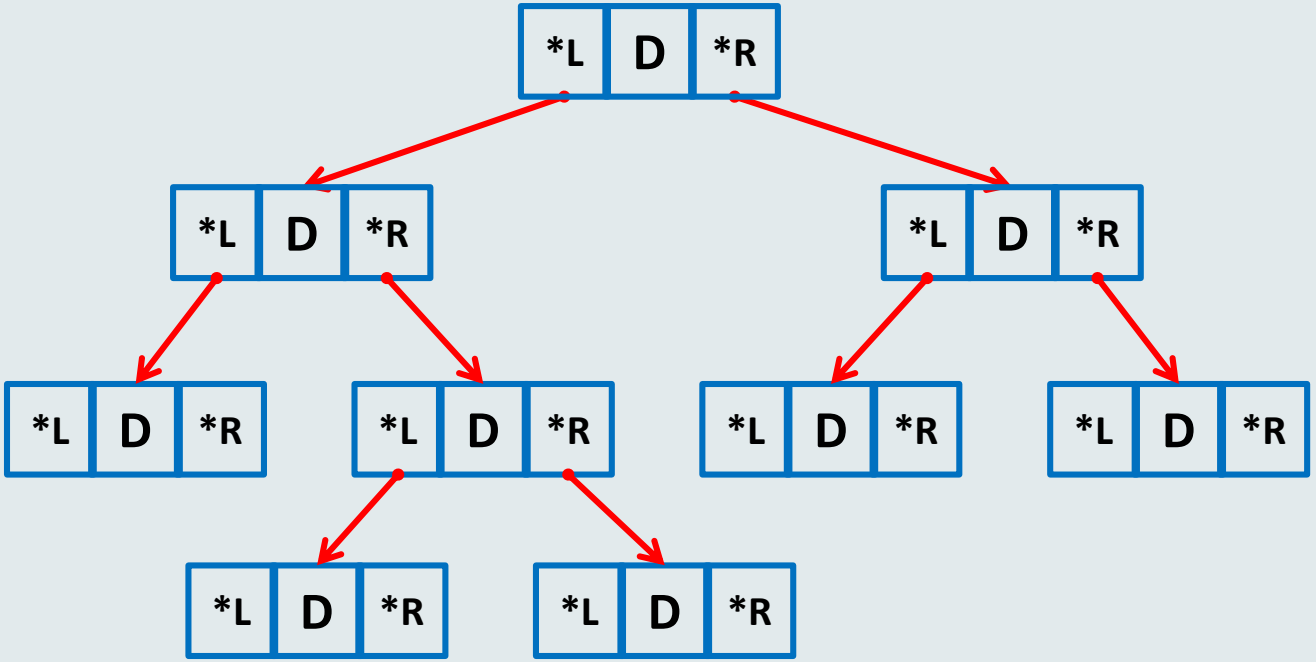
Binary Tree

- Jumlah node maksimal pada perfect binary tree dengan height n adalah $2^{n+1}-1$
- Height dari perfect binary tree dengan n node adalah $\log_2(n+1)-1$



Pembentukan Binary Tree

- Binary tree dibentuk dengan node yang mempunyai **D**ata dan dua buah pointer/link (***L**eft dan ***R**ight)



Struktur Dasar Binary Tree

Struktur Data Tree mirip dengan Double Linked List, hanya beda penamaan.

- **DATA** : berisi informasi setiap 1 buah elemen tree, bisa berupa bilangan (int), string, maupun address.
- **LEFT** : merupakan bagian elemen yang bertipe dasar "Address", yaitu berisi alamat untuk menunjuk tree bagian kiri.
- **RIGHT** : merupakan bagian elemen yang bertipe dasar "Address", yaitu berisi alamat untuk menunjuk tree bagian kanan.

Deklarasi Tree

```
typedef char typeInfo;
struct node {
   typeInfo data;
   node *left;       /* cabang kiri */
   node *right;      /* cabang kanan */
};
```

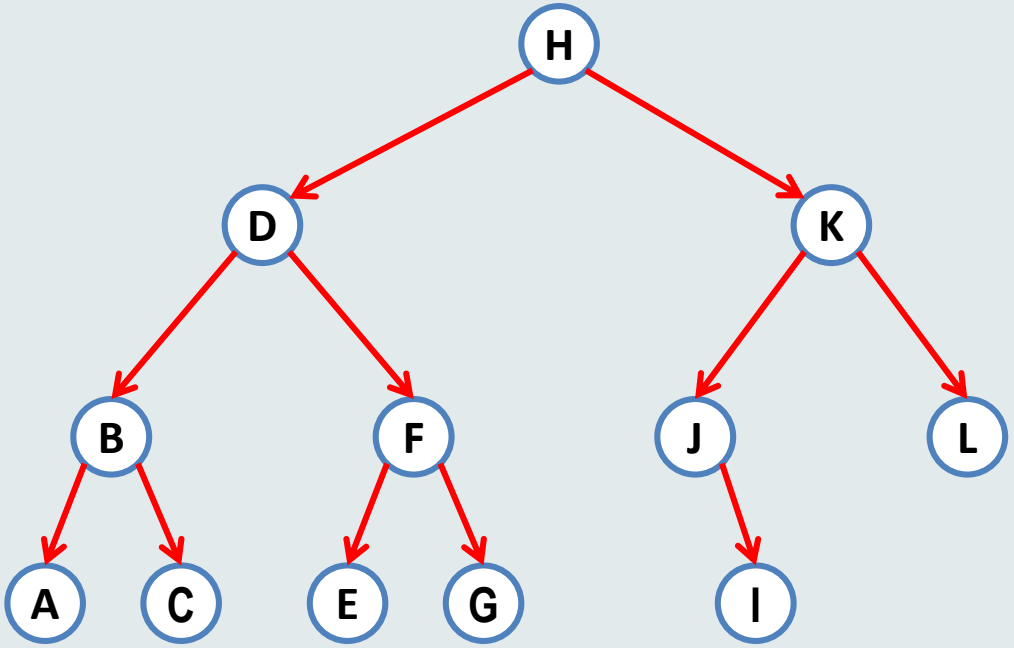
Pembentukan Tree

Langkah-langkah Pembentukan Binary Tree

1. Siapkan node baru
   - alokasikan memory-nya
   - masukkan info-nya
   - set pointer kiri & kanan = NULL
2. Sisipkan pada posisi yang tepat
   - **penelusuran** → utk menentukan posisi yang tepat; info yang nilainya lebih besar dari parent akan ditelusuri di sebelah kanan, yang lebih kecil dari parent akan ditelusuri di sebelah kiri
   - **penempatan** → info yang nilainya lebih dari parent akan ditempatkan di sebelah kanan, yang lebih kecil di sebelah kiri

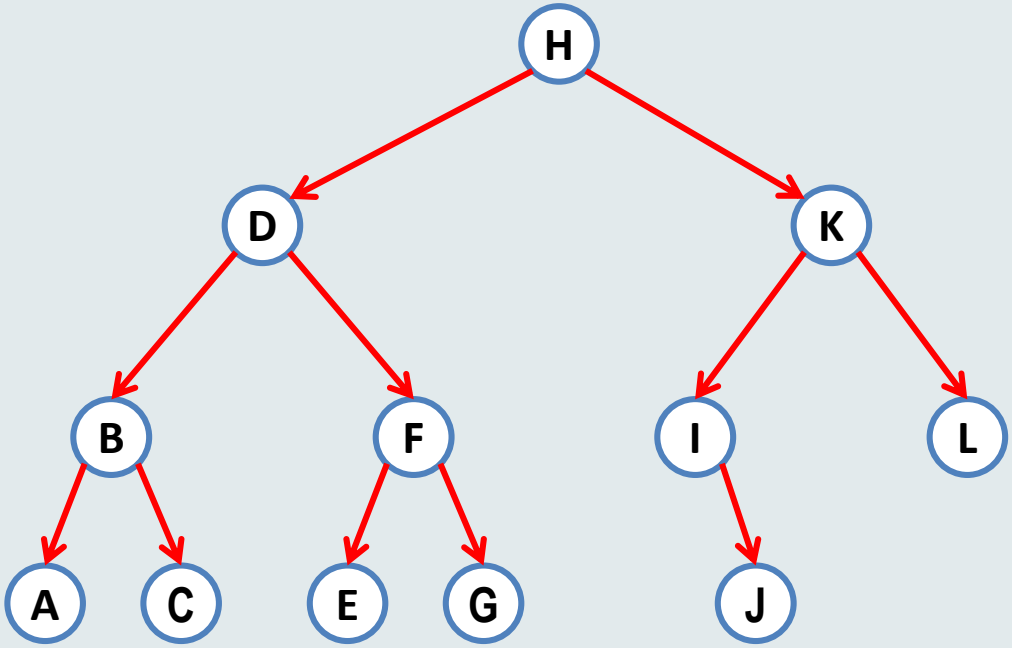
Binary Tree Traversal

- Binary Tree Traversal
  - Breadth First : Level order
  - Depth First :
    - Preorder
    - Inorder
    - Postorder



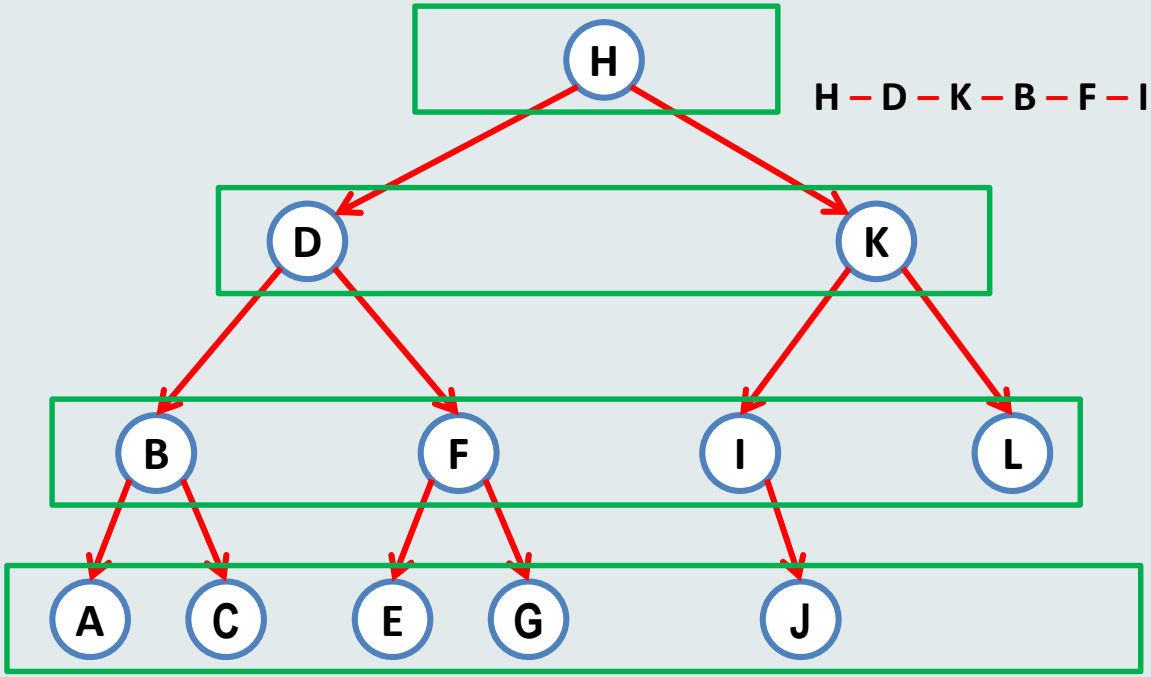
Binary Tree Traversal

– Level Order Traversal
  mengunjungi setiap node dari level teratas kemudian bergerak ke node sebelah kiri kemudian node sebelah kanan pada level dibawahnya.



Binary Tree Traversal

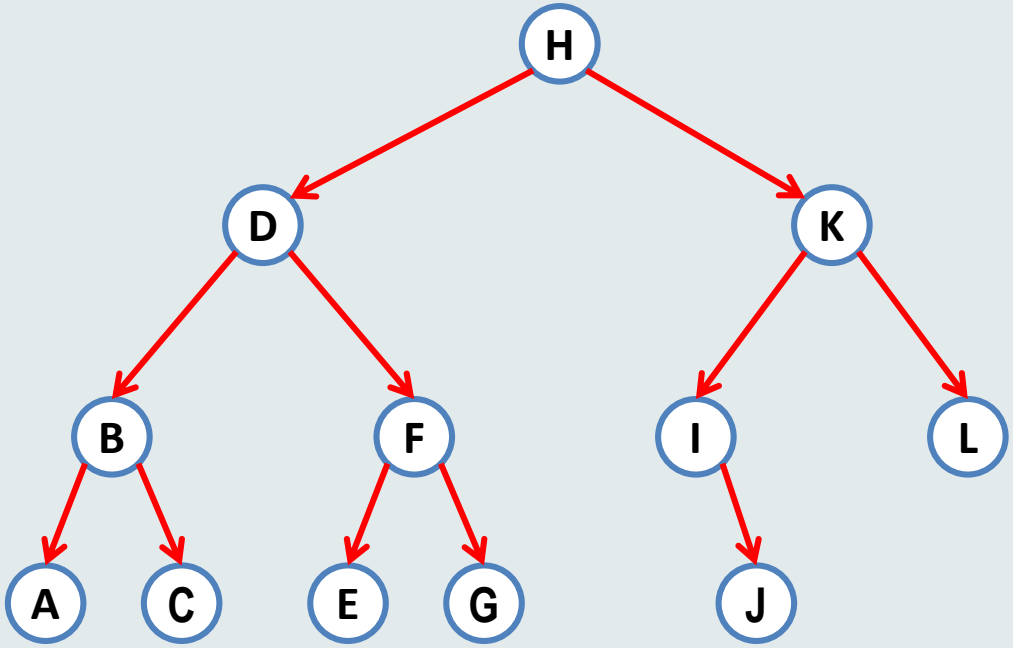
– Level Order Traversal
  mengunjungi setiap node dari level teratas kemudian bergerak ke node sebelah kiri kemudian node sebelah kanan pada level dibawahnya.



H – D – K – B – F – I – L – A – C – E – G – J

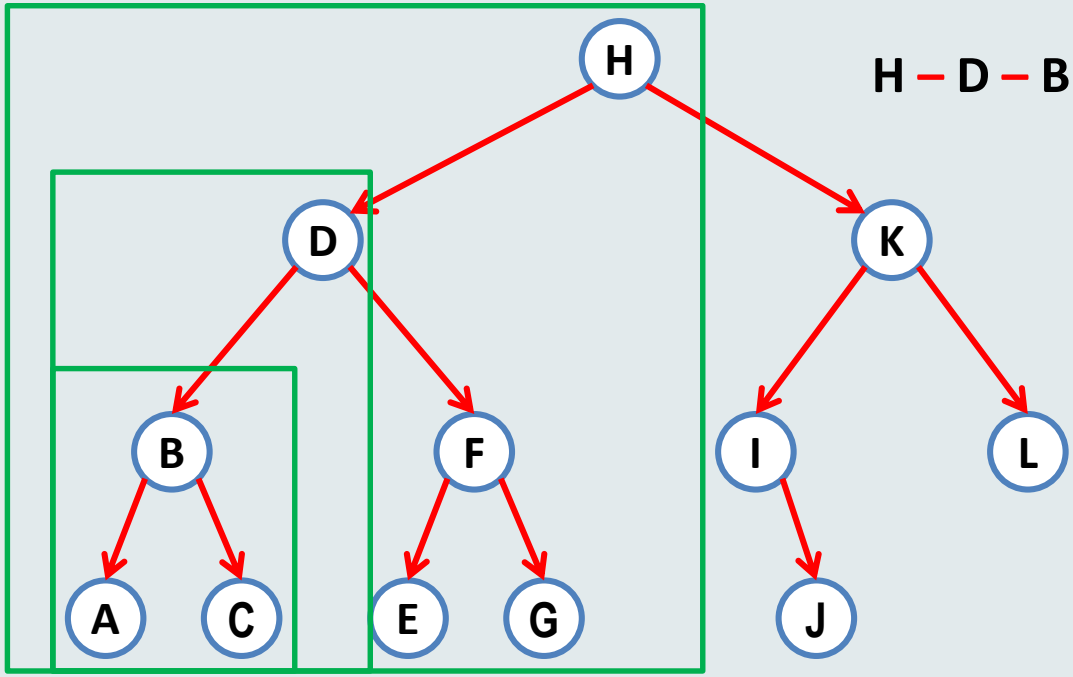
Binary Tree Traversal

– Preorder traversal
  mengunjungi node terbawah hingga mencapai setiap children node dengan urutan:
  Parent → Left children → Right children



Binary Tree Traversal

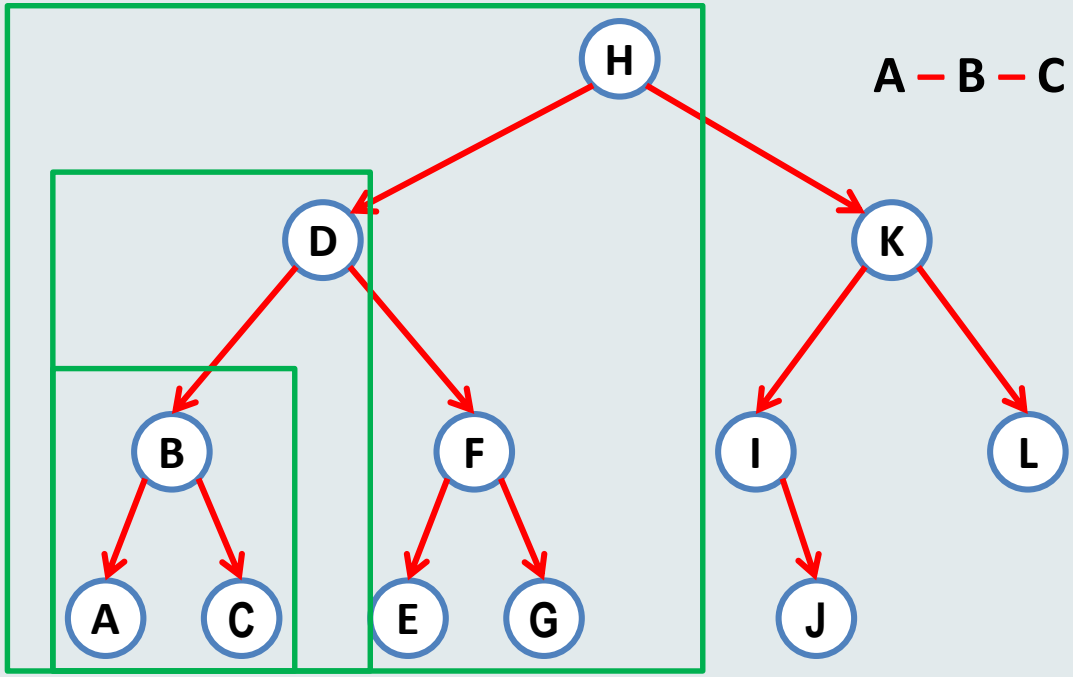
– Preorder traversal
  mengunjungi node terbawah hingga mencapai setiap children node dengan urutan:
  Parent → Left children → Right children



H – D – B – A – C – F – E – G – K – I – J – L

Binary Tree Traversal

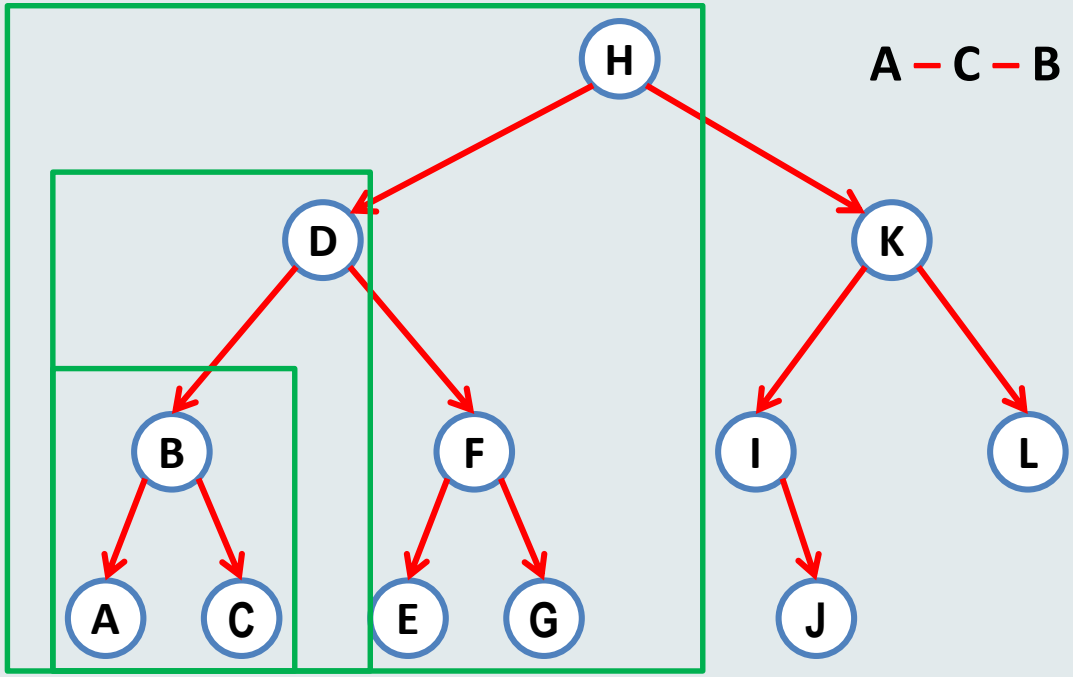
– Inorder traversal
  mengunjungi node terbawah hingga mencapai setiap children node dengan urutan:
  Left children → Parent → Right children



A – B – C – D – E – F – G – H – I – J – K – L

Binary Tree Traversal

– Postorder traversal
  mengunjungi node terbawah hingga mencapai setiap children node dengan urutan:
  Left children → Right children → Parent



A – C – B – E – G – F – D – J – I – L – K – H

Terimakasih