

# Pengantar Struktur Data Tree (pohon)





# Outline

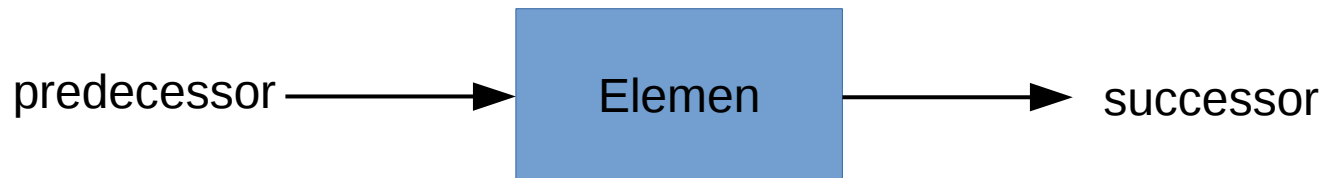
---

- Struktur data linear vs non-linear
- Pengantar graph
- Tree
- Aplikasi tree
- Binary tree
- Implementasi Tree

# Linear vs Non-linear

---

- Data struktur linear : jika elemen-elemen datanya membentuk sequence atau list
- Tiap elemen data pada struktur linear memiliki satu predecessor dan satu successor



- Contoh : Link List dan array
- Data struktur non-linear ? Hubungan antar elemen data bersifat hirarki dan network

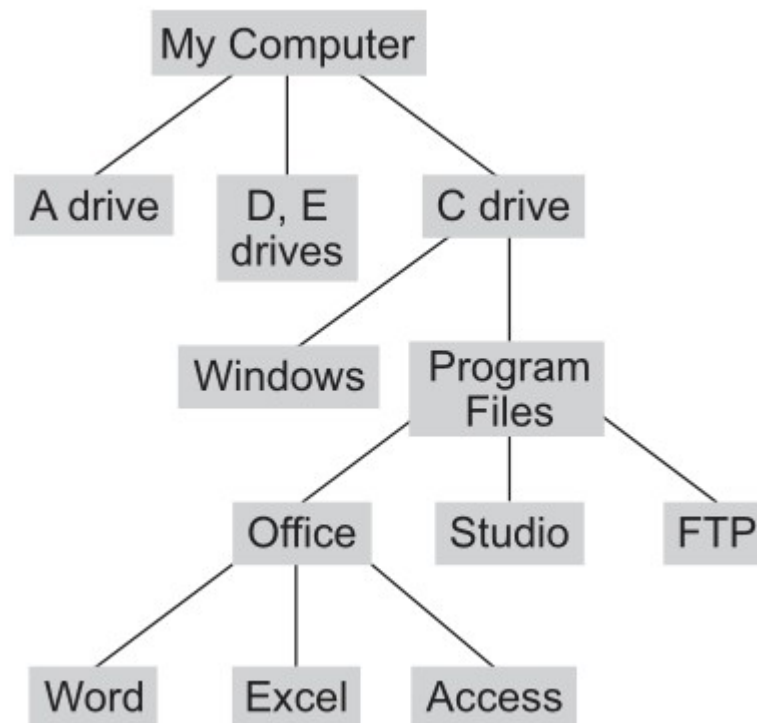
# Non-linear

---

- Tiap elemen data memiliki lebih dari satu predecessor dan successor
- Sehingga kita tidak bisa “menafsirkan” linear sequence ( keterurutan ) seperti pada list
- Struktur data non-linear memiliki kemampuan untuk menggambarkan relasi yang lebih kompleks dari struktur linear
- Contoh struktur non-linear : hirarki keluarga, hirarki organisasi, dll

# Contoh nonlinear

- Organisasi file dan direktori pada komputer

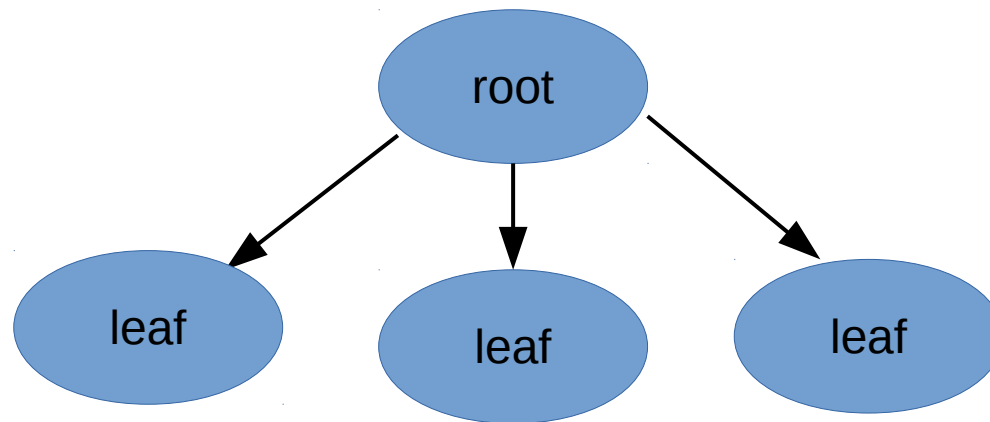


- Didalam direktori terdapat file atau subdirektori

# Nonlinear

---

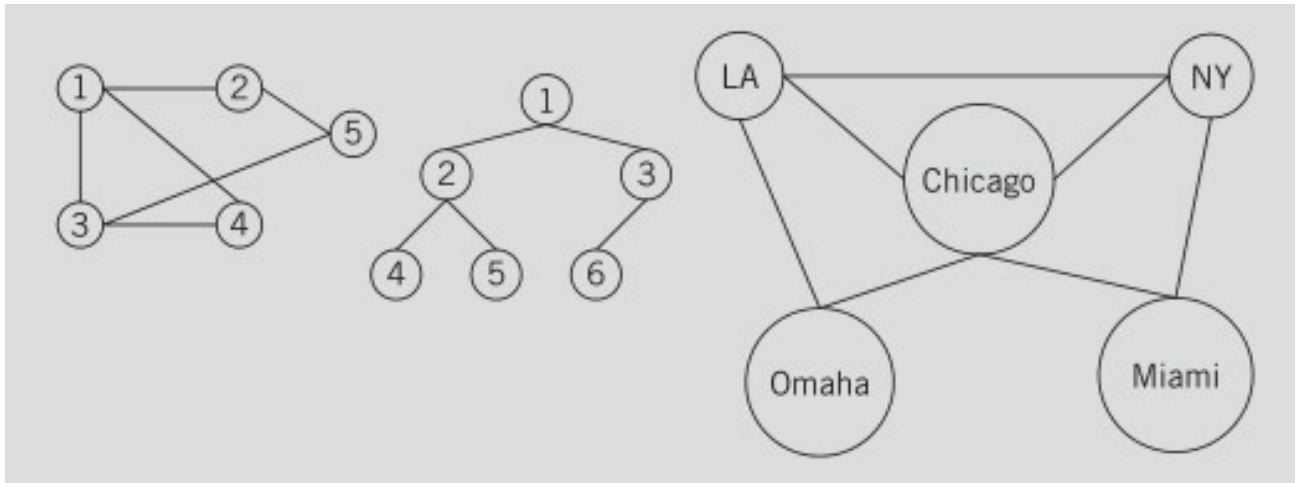
- Bentuk struktur yang menunjukkan hirarki disebut tree (pohon)
- Bagian paling atas disebut root ( akar )



- Turunan (successor) dari root disebut daun (leaf) jika tidak punya turunan dibawahnya

# Nonlinear (3)

- Contoh struktur nonlinear lainnya adalah Graph
- Graph adalah struktur data yang terdiri dari himpunan simpul (node/vertice) dan sisi (edge)



# Apa yang sejauh ini kita ketahui?

---

- Linear : List meliputi array, matriks, linked list, Stack, Queue
- Non-linear : Tree dan Graph
- Dengan struktur data diatas, sebagian besar masalah dalam computer science aka informatika dapat diimplementasikan dalam bentuk program
- Hanya saja, struktur data tersebut perlu di perdetail atau di custom untuk masalah yang spesifik / khusus



# Teori graph

---

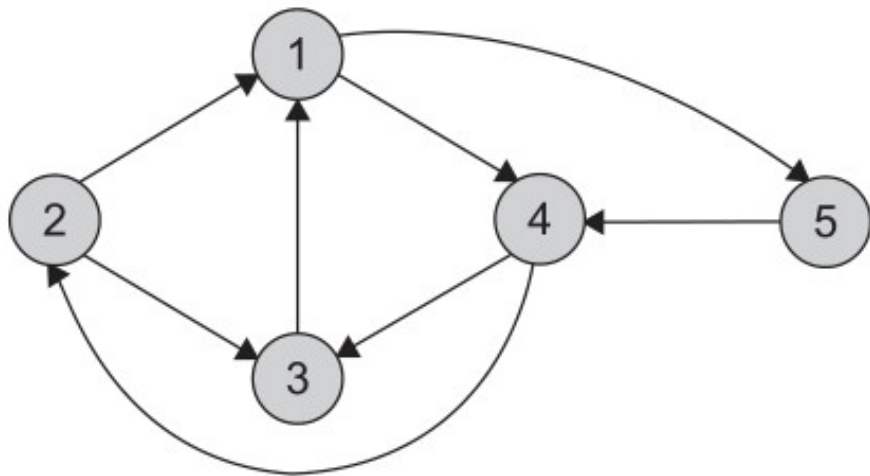
- Dalam literatur, graph sering ditulis dalam notasi matematik:

$G = \{ E, V \}$  yang diartikan graph  $G$  terdiri dari himpunan sisi  $E$  dan himpunan vertice  $V$

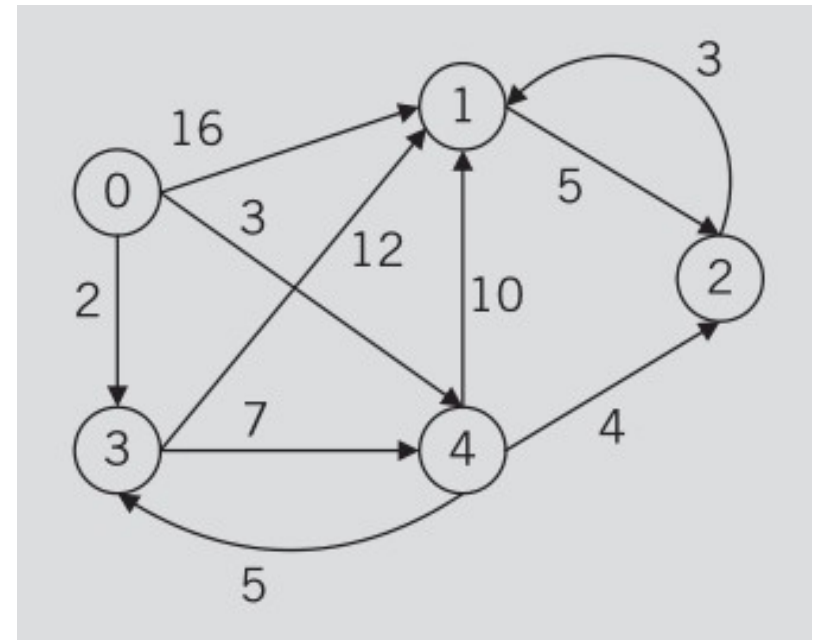
- Dilihat dari sifat sisi (edge)-nya, graph dibagi jadi 2 : directed dan undirected
- Directed graph: sisi memiliki arah, undirected berarti sisi tidak ada informasi arah
- Weighted graph: sisi memiliki bobot / nilai

# Contoh Graph

- Directed Graph

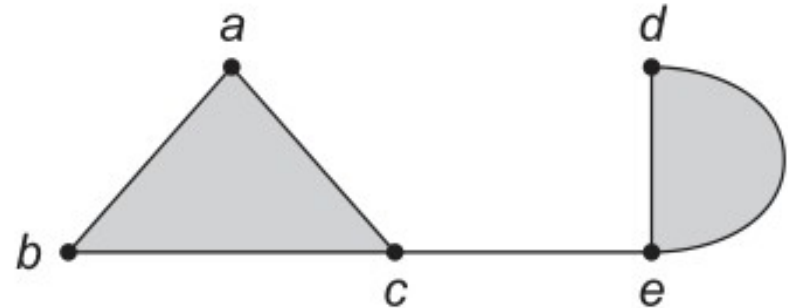
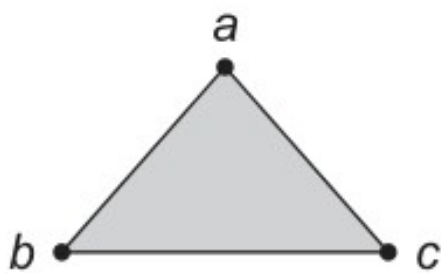


- Weighted graph

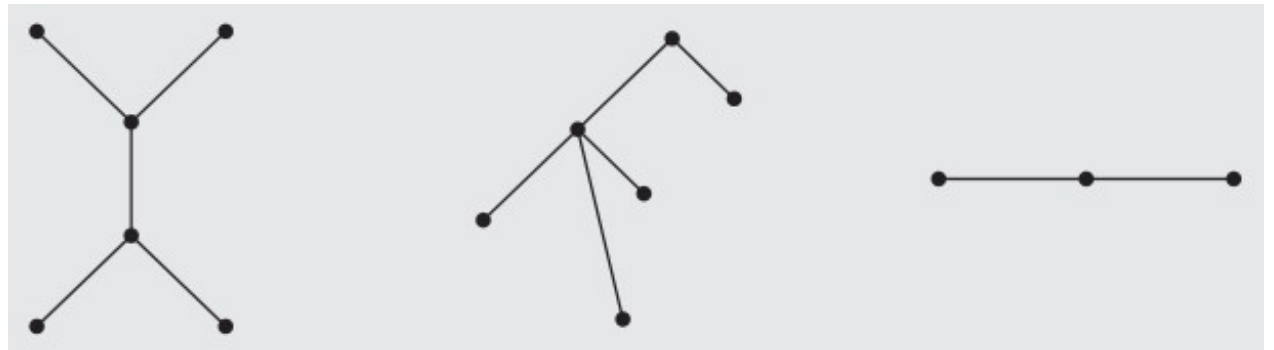


# Tree is a Graph


- Jika kita perhatikan, Tree adalah graph yang **tidak ada** loop ( cycle )
- Contoh loop :

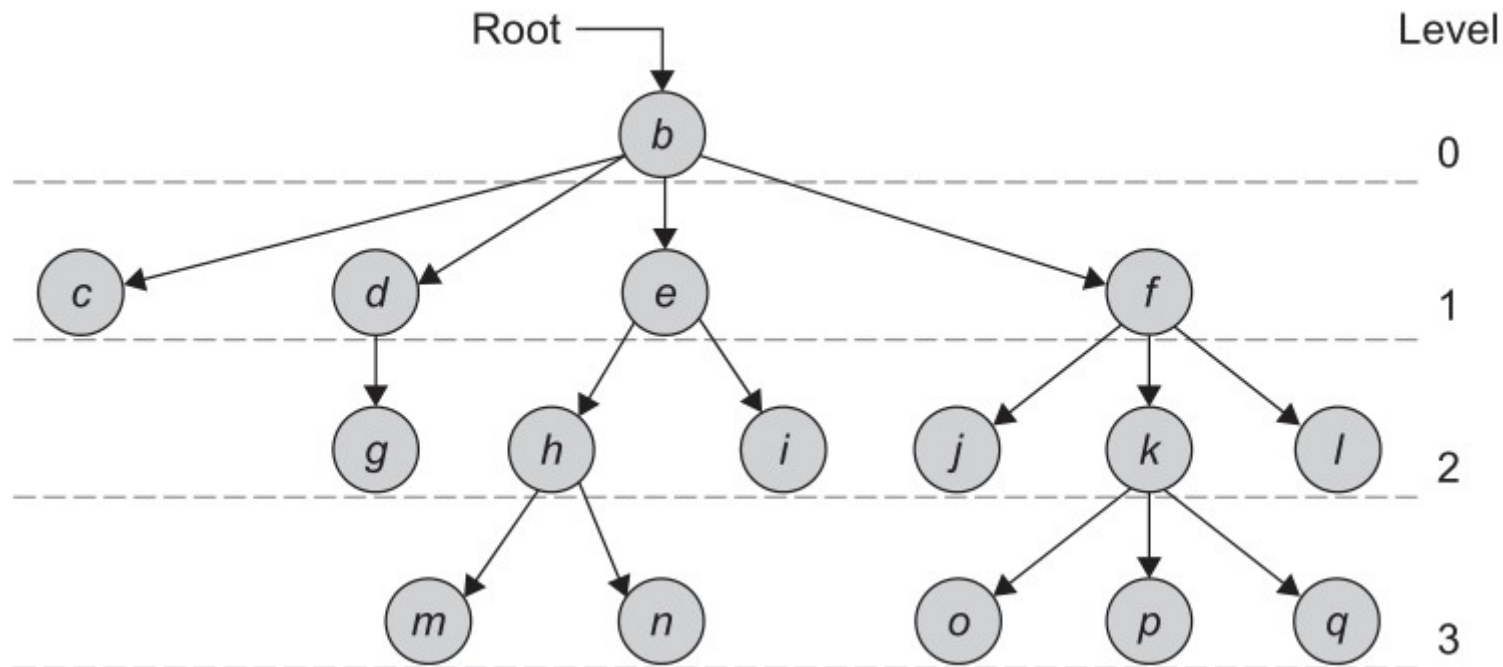


- Contoh Tree :



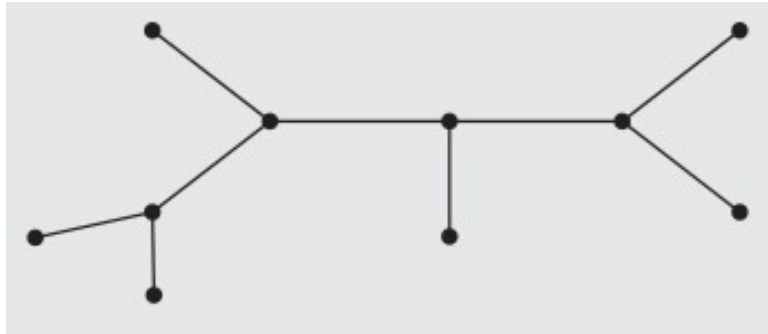
# Macam jenis Tree

- Tree kosong, tidak ada node :)
- Tree dengan satu node yaitu root      Root → 
- Tree dengan root dan beberapa sub-tree

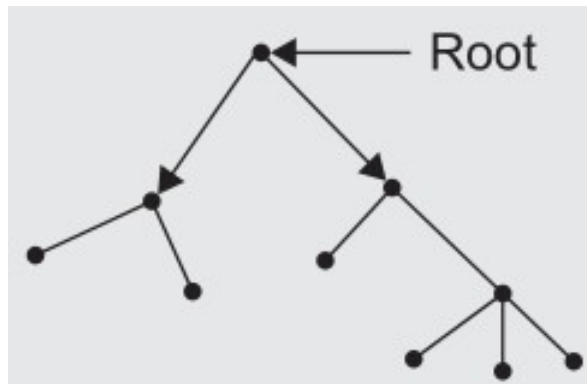


# Macam jenis(2)

- Free tree: connected, acyclic graph



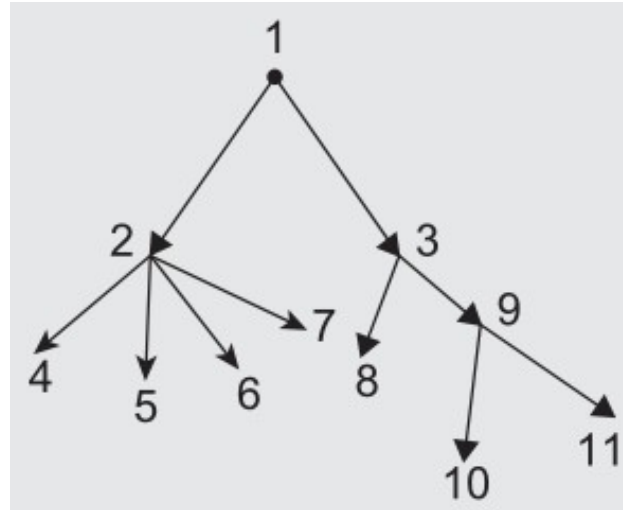
- Rooted tree: directed graph dengan satu node jadi root



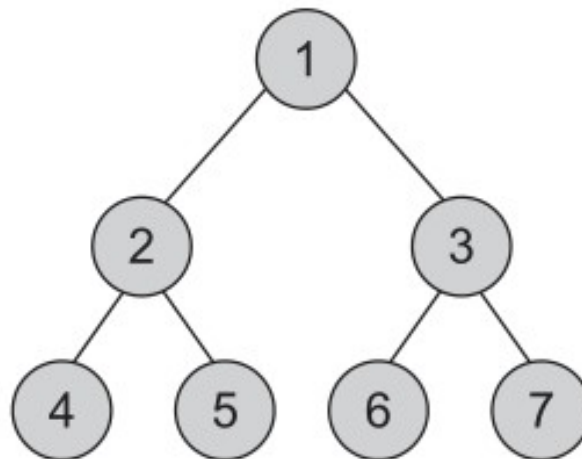


# Jenis tree(3)

- Ordered tree  
tiap node memiliki  
nomor terurut dari root

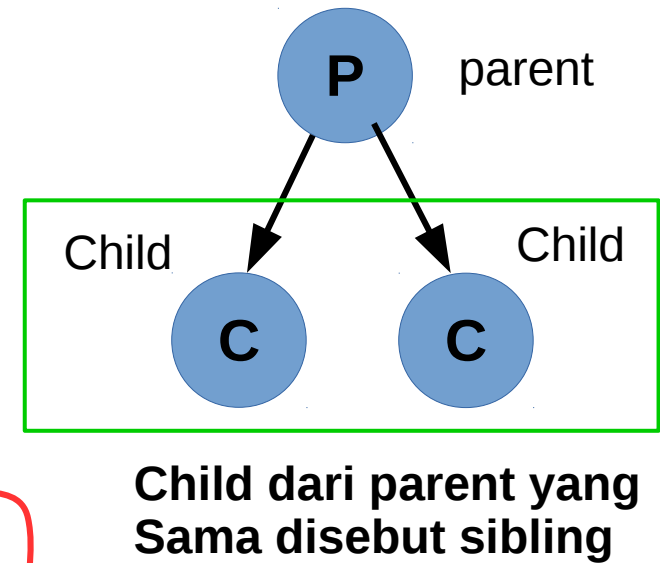
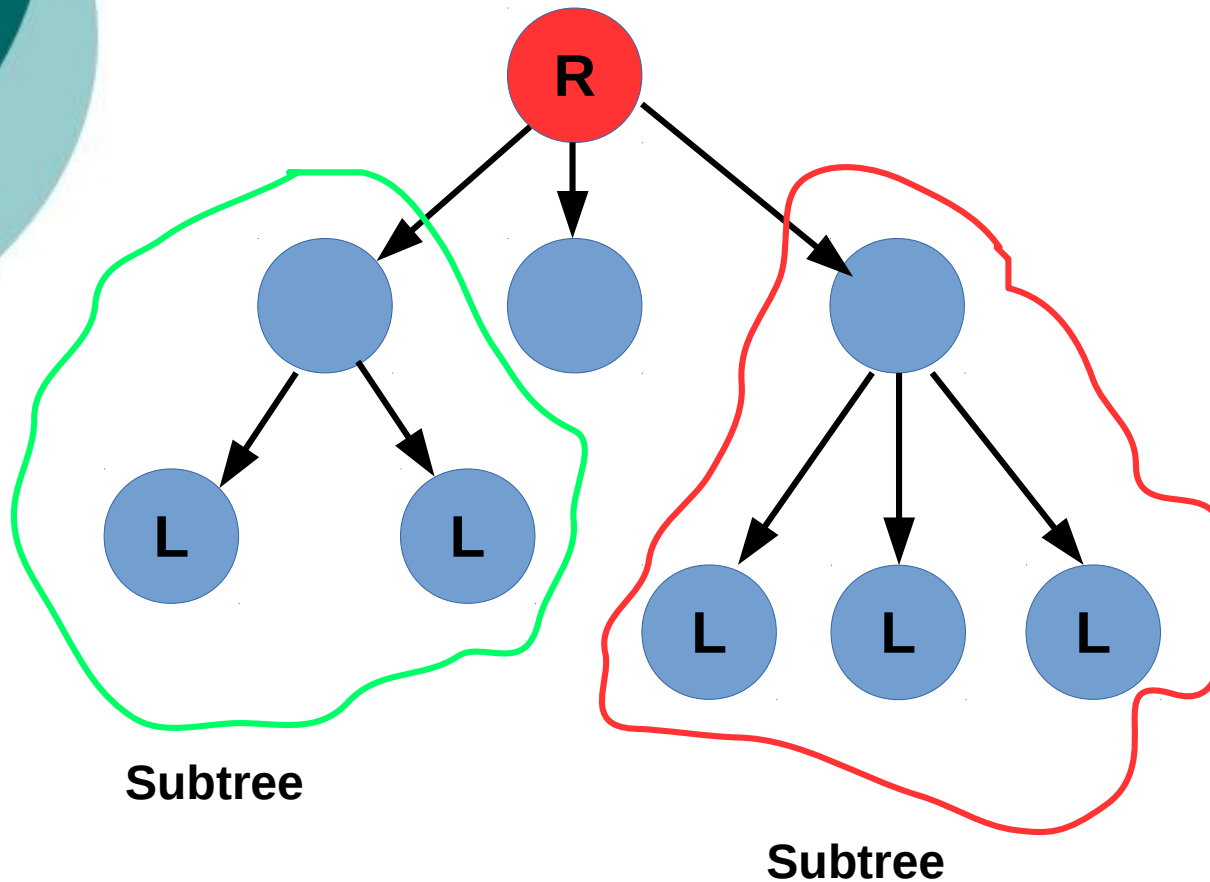


- Binary tree: tree dengan dua subtree



# Anatomi Tree

- R: root, L: leaf



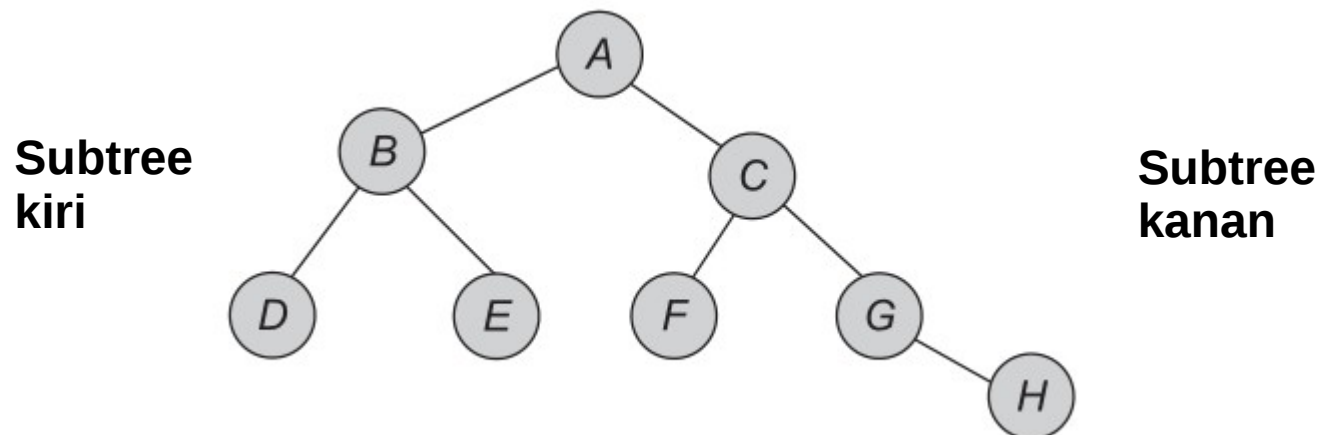
# Aplikasi graph & tree

---

- Dalam ilmu komputer / informatika, graph & tree banyak diaplikasikan seperti contoh berikut:
  - Graph : untuk membuat abstraksi peta. Edge / sisi adalah jalan, simpul/node adalah tempat
  - Graph banyak digunakan dalam Computer Vision
  - Graph juga digunakan untuk membuat model komputer 3D dan animasi ( istilah kerennya Rigging )
  - Tree : banyak digunakan dalam database
  - Tree juga banyak digunakan dalam internal compiler

# Binary Tree

- Jenis Tree yang banyak digunakan untuk keperluan parsing ( mengolah ekspresi matematik )
- Tree dengan satu node root dan dua sub tree



# Definisi binary tree

- Suatu binary tree adalah
  - Tree yang kosong alias tak ada node
  - Atau terdiri dari satu node sebagai root dan dua child ( kanan dan kiri ) yang masing-masing juga merupakan binary tree ( definisi rekursif )



Kotak adalah child berupa tree yang kosong



# Contoh implementasi binary tree

---

```
class TreeNode
{
public:
    char Data;          //memuat info/data karakter
    TreeNode *Lchild;   //pointer ke child kiri
    TreeNode *Rchild;   //pointer ke child kanan
};
```

```
class BinaryTree
{
private:
    TreeNode *Root;
public:
    BinaryTree()        //Konstruktor membuat tree
    {
        Root = NULL;   //kondisi awal: root kosong
    }
    TreeNode *GetNode(); //mendapat pointer ke suatu node
};
```



Contoh implementasi  
Belum lengkap

# Binary tree sebagai ADT

---

- Karena properti binary tree sudah kita bahas didepan, ADT tinggal membuat rincian operasi
  - Membuat binary tree
  - Menyisipkan anak kanan ( subtree kanan )
  - Menyisipkan anak kiri ( subtree kiri )
  - Menghapus anak / node
  - Menelusuri : berpindah dari root ke anak dan seterusnya sampai ke daun
  - dll

