



# STRUKTUR DATA

S1 PENDIDIKAN TEKNOLOGI INFORMASI

UNIVERSITAS NEGERI SURABAYA

PERTEMUAN KE 1

# Biodata

- Nama : Riza Akhsani Setyo Prayoga
- Email : [rizaprayoga@unesa.ac.id](mailto:rizaprayoga@unesa.ac.id)
- WA : 081286153175
- Pendidikan : S1 Sistem Informasi Universitas Brawijaya  
S2 Manajemen Teknologi Informasi

# Rencana Pembelajaran

Pertemuan	Agenda
1	Perkenalan + Tipe Struktur Data
2	Struktur Data Linked List
3	Struktur Data Linked List Lanjutan
4	Queue
5	Stack
6	Tree, Binary Tree, Traversal pada Binary Tree
7	Sertifikasi C++ untuk struktur data
8	Ujian Tengah Semester

# Referensi buku

- Michael T. Goodrich, Roberto Tamassia, David Mount(2011). Data Structures and Algorithm in C++ (2nd ed.).

# Etika perkuliahan

- Mahasiswa/I bisa hadir tepat waktu dengan toleransi keterlambatan 15 menit
- Segala bentuk penugasan bisa dikumpulkan secara tepat waktu. Jika melebihi waktu yang ditentukan maka **tidak dinilai**
- Diperbolehkan menghubungi dosen via wa di jam kerja dan hari kerja ( Senin – Jumat pukul 08.00 – 16.00 WIB)

# Etika perkuliahan





# Pengenalan materi

Pengenalan Struktur Data dan Tipe Data

# Pengantar

- Apa itu struktur data ?



# Pengantar

- Struktur data merupakan salah satu cara **menyimpan maupun mengatur sebuah data** secara rapi dan terstruktur dalam sebuah sistem komputer ataupun database dengan **tujuan memberikan kemudahan dalam mengaksesnya.**

# Tipe Struktur Data

- Array
- Linked List
- Stack
- Queue
- Tree

# Tipe Struktur Data

- Array memberikan kemudahan bagi seseorang yang sedang mencari data acak hanya dengan **menggunakan indeksnya saja**

# Array

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string city[4] = {"Surabaya", "Semarang", "Jakarta", "Bandung"};
    cout << city[2];
    return 0;
}
```

Jakarta

# Array

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string buah[4] = {"Anggur", "Melon", "Jeruk", "Salak"};
    cout << buah[0];
    return 0;
}
```

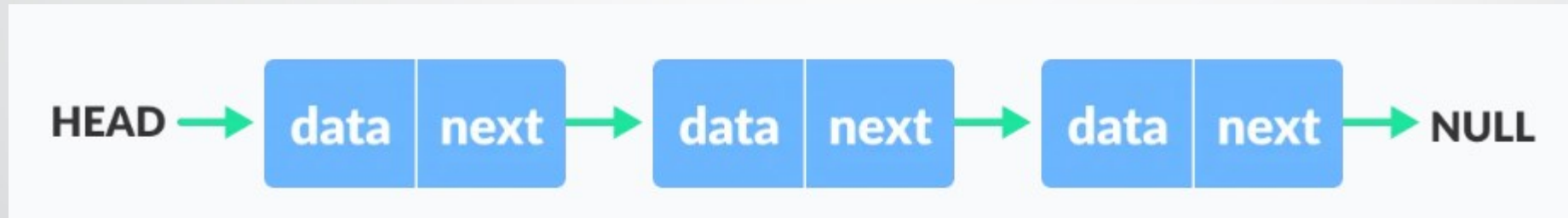
Anggur

# Tipe Struktur Data

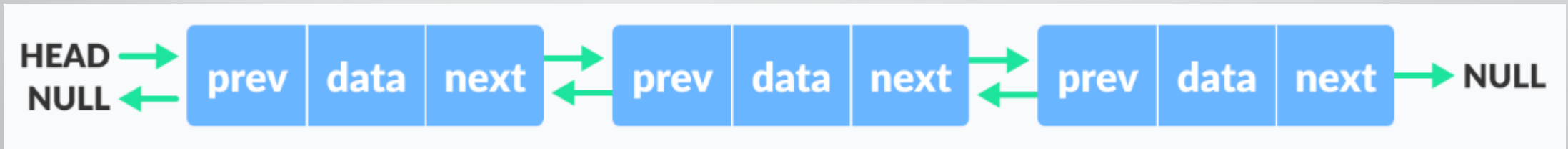
- Linked list merupakan tipe struktur data yang tersusun atas **urutan data yang bersifat liner yang saling terhubung satu sama lain** dan data harus diakses secara manual karena kita tidak dapat mencari suatu data dengan menggunakan sistem acak tersebut.
- Adapun tiga jenis dari sistem linked list antara lain **singly linked list, doubly linked list, dan juga circular linked list.**

# Linked List

## Single Linked List



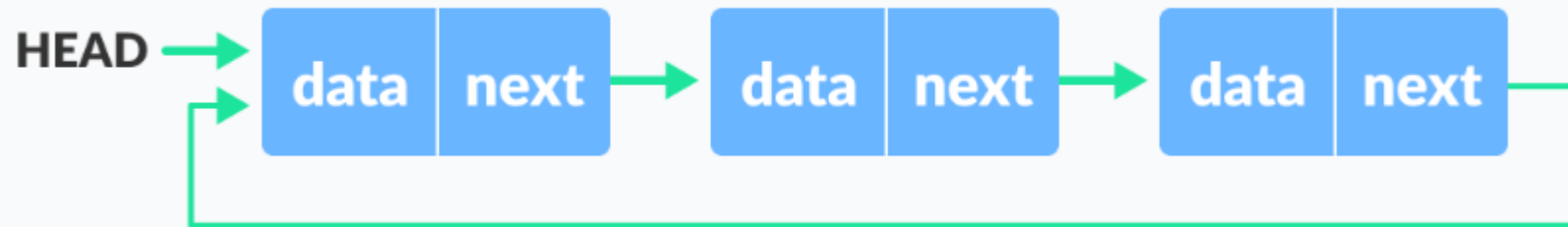
## Double Linked List



<https://dsvisualizer.isatvik.com/linkedlist>

# Linked List

Circular Linked List

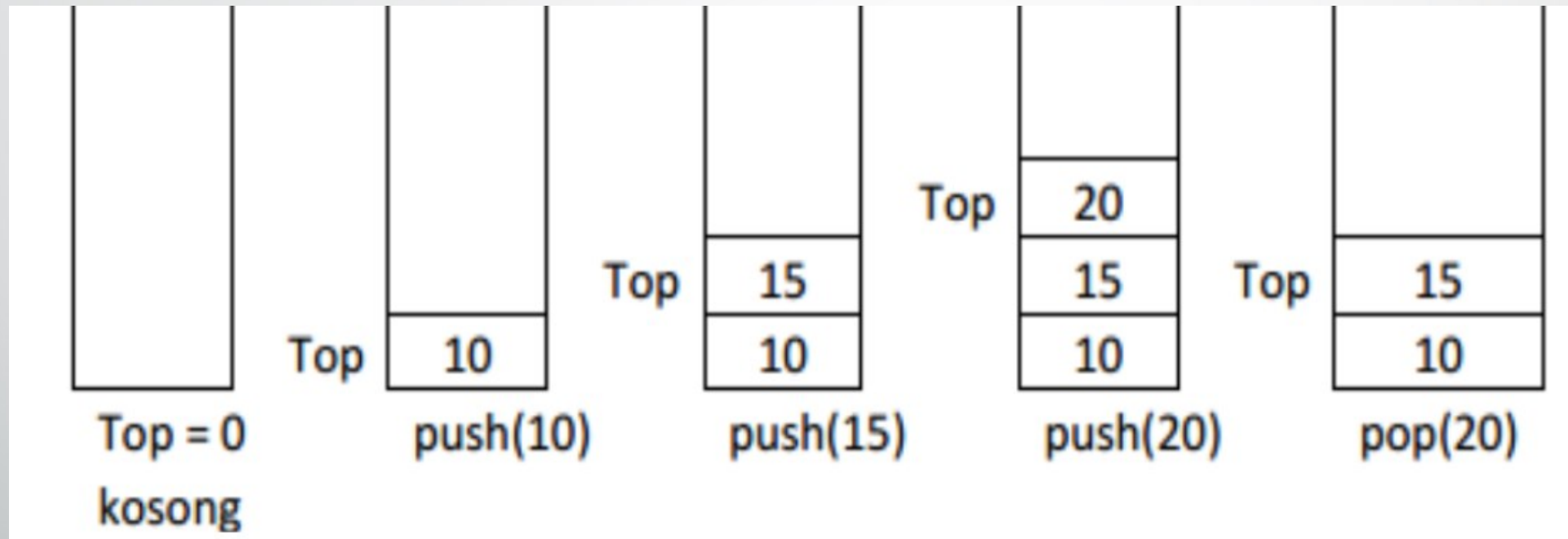




# Tipe Struktur Data

- Stack merupakan tipe struktur data ketiga yang berupa **data linear dengan adanya urutan tertentu**. Urutan yang biasa digunakan adalah **Last In First Out** atau **First In Last Out**, yang mana keduanya memiliki arti yang sama pada intinya data yang masuk terlebih dahulu akan keluar terakhir, begitu sebaliknya.

# Stack



Push untuk menambah data  
Pop untuk menghapus data

<https://dsvisualizer.isatvik.com/stack>


# Stack

```
#include <iostream>
#include <stack>
using namespace std;
int main() {
    // create a stack of strings
    stack<string> makanan;
    // add element to the Stack
    makanan.push("Nasi Goreng");
    makanan.push("Gulai Kambing");
    makanan.push("Sate Ayam");
    makanan.push("Pecel");
    // print top element
    cout << makanan.top();
    return 0;
}
```



Pecel

# Stack



Initial Stack: Sate Ayam, Gulai Kambing, Nasi Goreng,  
Final Stack: Gulai Kambing, Nasi Goreng,

```
#include <iostream>
#include <stack>
using namespace std;
void display_stack(stack<string> st);
int main() {
    stack<string> makanan;
    makanan.push("Nasi Goreng");
    makanan.push("Gulai Kambing");
    makanan.push("Sate Ayam");
    cout << "Initial Stack: ";
    display_stack(makanan);
    makanan.pop();
    cout << "Final Stack: ";
    display_stack(makanan);
    return 0;
}
void display_stack(stack<string> st) {
    while(!st.empty()) {
        cout << st.top() << ", ";
        st.pop();
    }
}
```

# Tipe Struktur Data

- Queue merupakan tipe data linear dengan adanya urutan tertentu yakni data yang pertama kali masuk akan menjadi data yang pertama kali diambil atau istilahnya **First In First Out**.


# Queue



<https://dsvisualizer.isatvik.com/queue>

# Queue

Initial Queue: Es Teh, Es Degan, Es Doger,  
Final Queue: Es Degan, Es Doger,



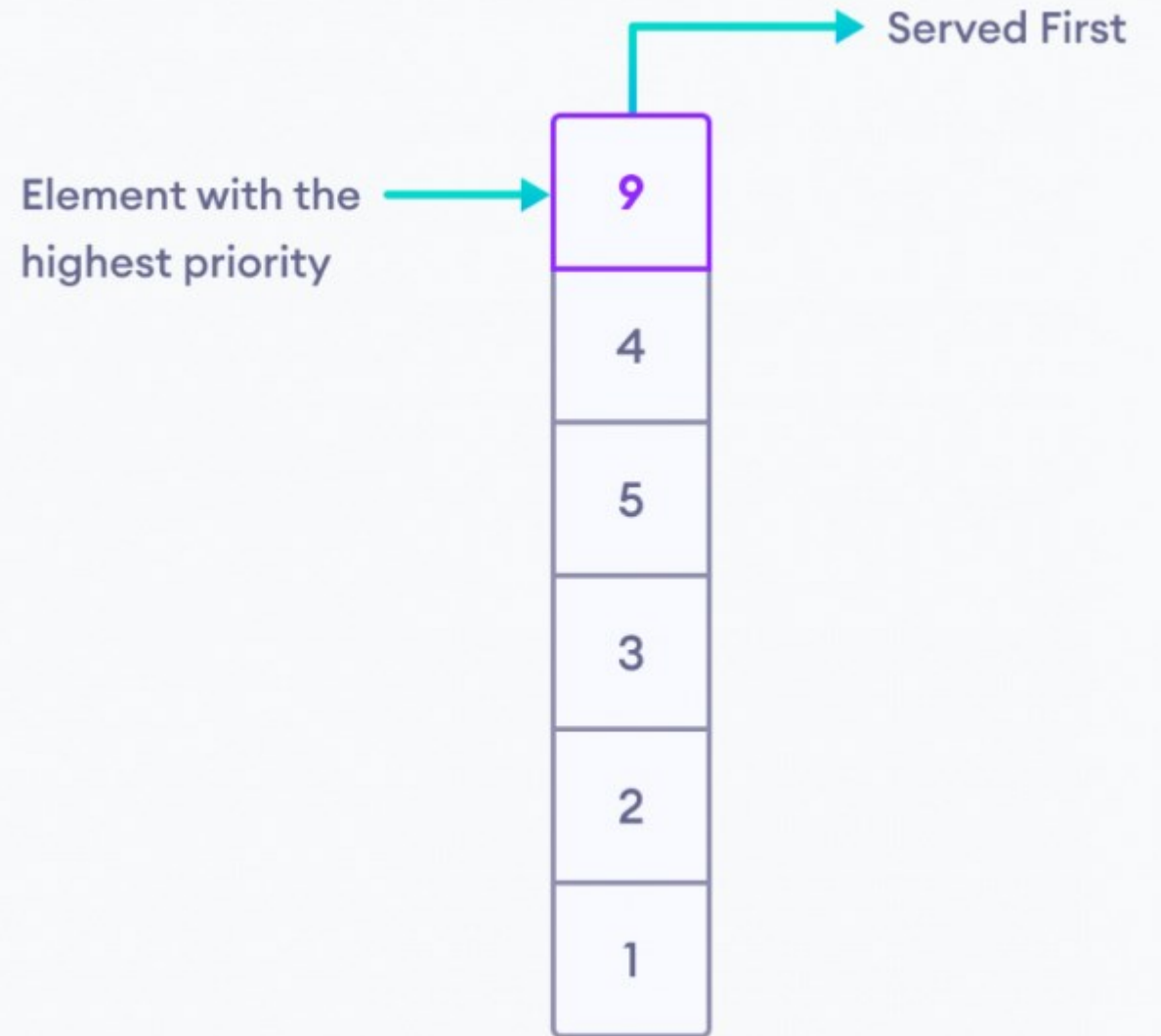
```
#include <iostream>
#include <queue>
using namespace std;
void display_queue(queue<string> q);
int main() {
    queue<string> minuman;
    minuman.push("Es Teh");
    minuman.push("Es Degan");
    minuman.push("Es Doger");
    cout << "Initial Queue: ";
    display_queue(minuman);
    minuman.pop();
    cout << "Final Queue: ";
    display_queue(minuman);
    return 0;
}
void display_queue(queue<string> q) {
    while(!q.empty()) {
        cout << q.front() << ", ";
        q.pop();
    }
}
```

# Tipe Struktur Data

- Priority Queue merupakan tipe queue khusus dimana **setiap elemen dikaitkan dengan nilai prioritas** serta elemen disajikan dengan prioritasnya




# Priority Queue



<https://dsvisualizer.isatvik.com/priorityqueue>

# Priority Queue

Priority Queue: 100000, 10000, 100,



```
#include<iostream>
#include <queue>
using namespace std;

int main() {

    priority_queue<int> nomor;

    nomor.push(100);
    nomor.push(100000);
    nomor.push(10000);

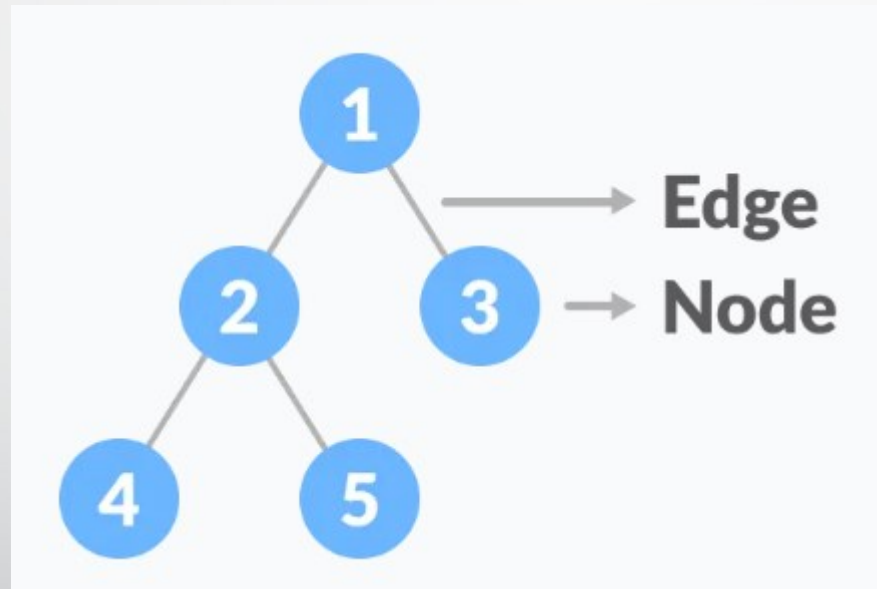
    cout << "Priority Queue: ";

    while(!nomor.empty()) {
        cout << nomor.top() << ", ";
        nomor.pop();
    }
```

# Tipe Struktur Data

- Tree merupakan tipe struktur data yang berbentuk menyerupai pohon. Tipe ini sangat efisien terutama **dalam melakukan penyimpanan data secara hierarkis** karena dapat tersusun pada beberapa level.

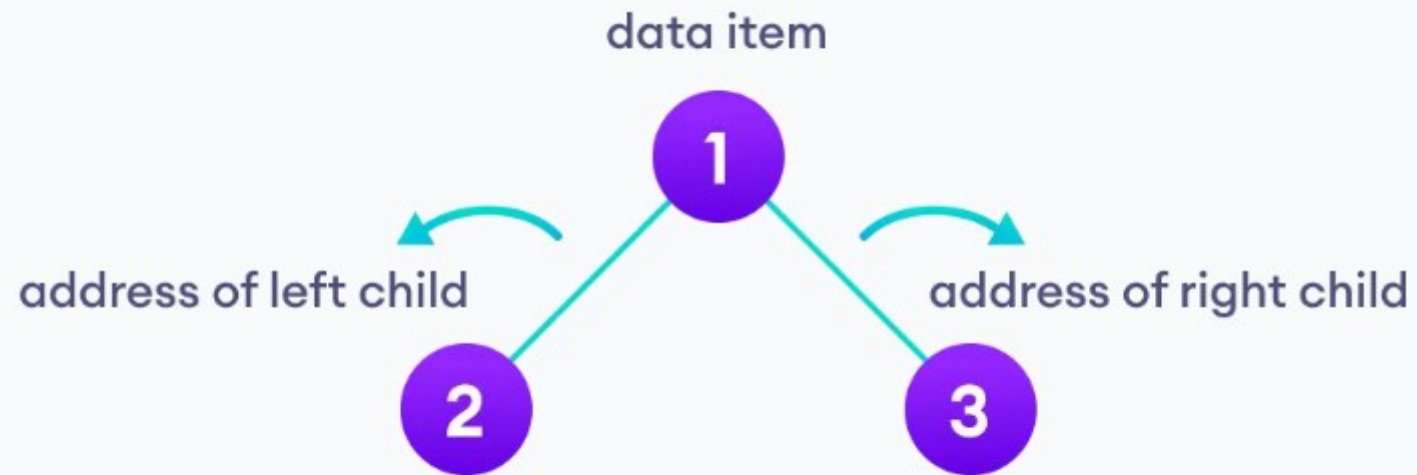
# Tree



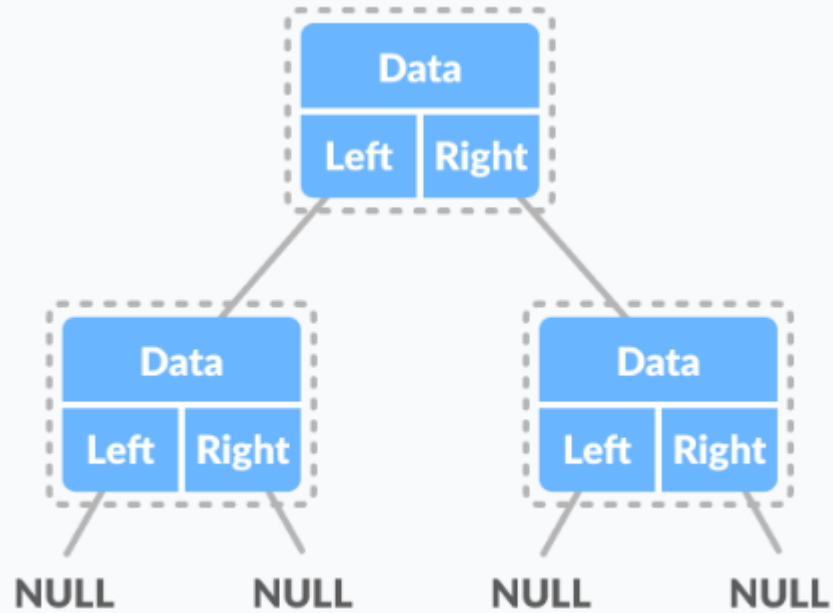
# Binary Tree

- Binary Tree adalah struktur data yang dipresentasikan dalam **bentuk pohon** dimana **setiap induk dapat dua anak**

# Binary Tree



# Binary Tree



# Tree Traversals

- Tree Traversals : Proses dalam penggunaan **model pohon yang mengevaluasi node pohon secara sistematis**. Terdapat 3 jenis model
- Pre Order Traversal : Mengunjungi root, lalu ke kiri subtree dan kanan subtree
- In Order Traversal : Mengunjungi kiri subtree, lalu ke root dan kanan subtree
- Post Order Traversal : Mengunjungi kiri subtree, lalu ke kanan subtree dan root.




# Tree Traversals

- Ilustrasi bisa dicek di <https://dsvisualizer.isatvik.com/treetraversals>



Terima kasih



# Post Tes Time Quizizz

<https://quizizz.com/join>