INF218
Struktur Data & Algoritma

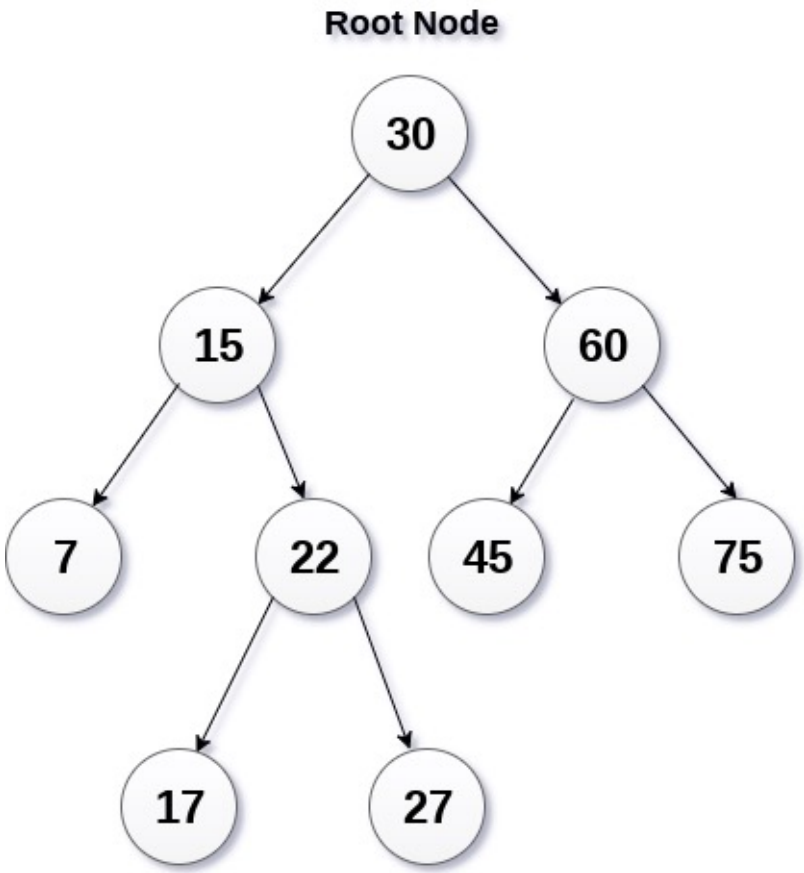
BST dan Tree Traversal

Alim Misbullah, S.Si., M.S.

Binary Search Tree

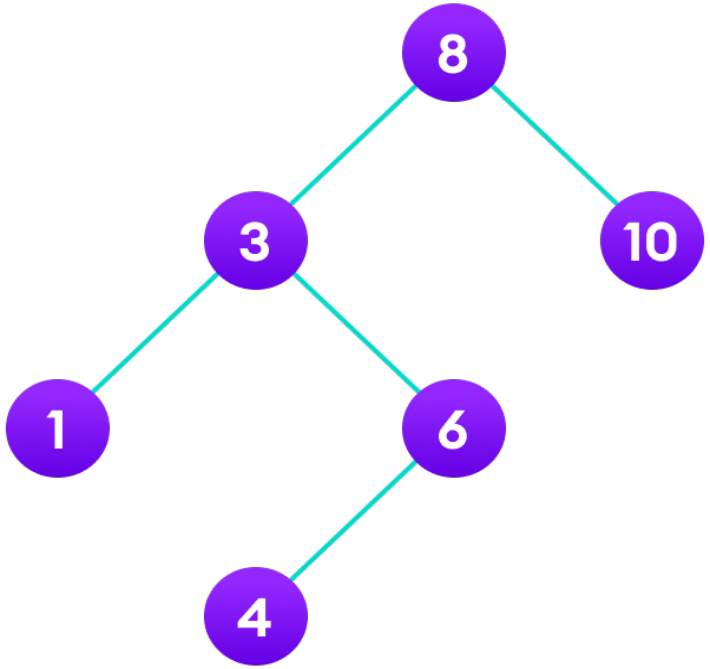
Binary Search Tree

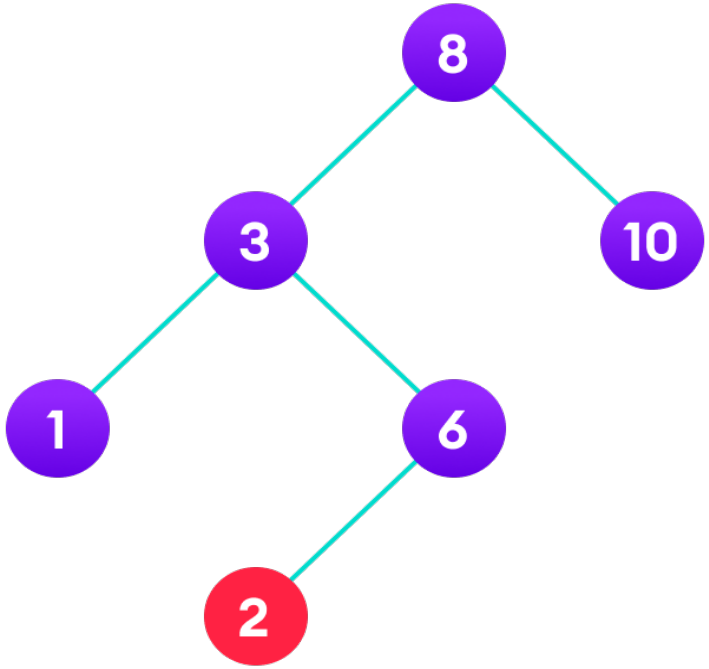
- Binary search tree merupakan sebuah struktur data yang mengizinkan kita secara cepat untuk melakukan pengurutan dari susunan bilangan atau disebut juga dengan **ordered binary tree**
- Pada binary search tree, **nilai dari semua node pada bagian kiri < nilai root**
- Sementara, **nilai dari semua node pada bagian kanan > nilai root**



Binary Search Tree

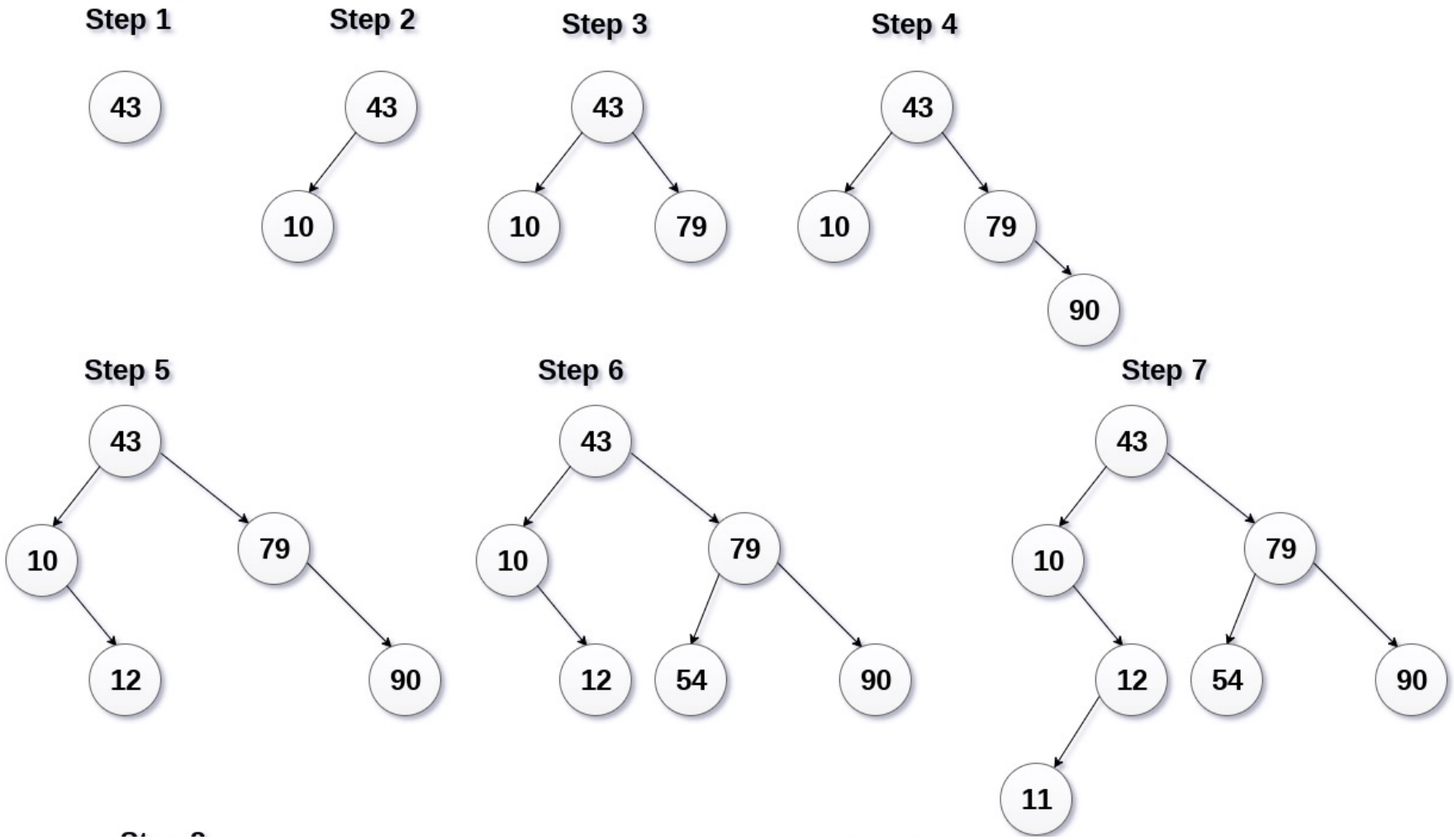
Binary Search Tree



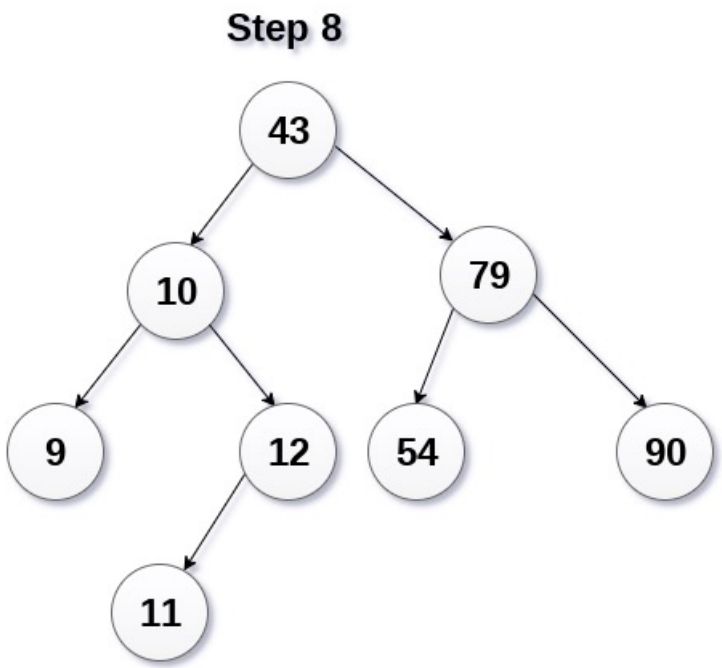


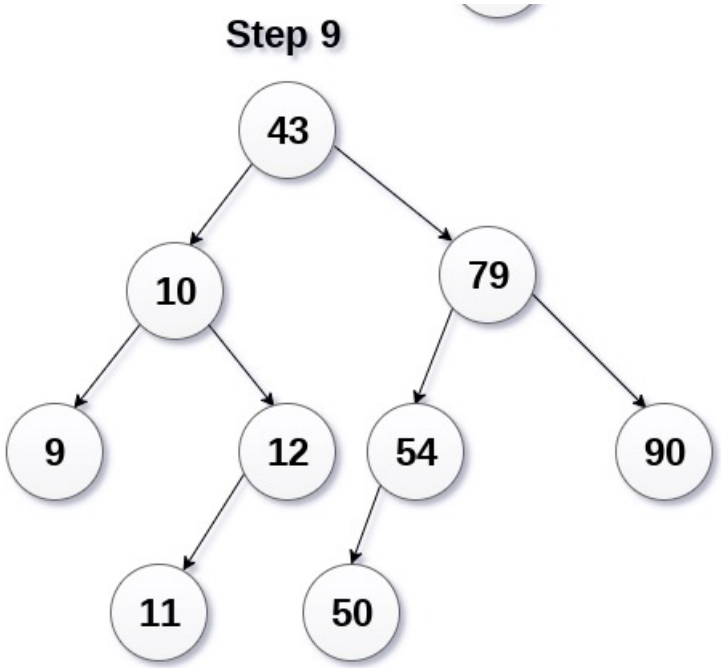
Bagian tree sebelah kanan adalah bukan binary search tree karena terdapat nilai lebih kecil di sebelah kanan “3” yaitu “2”

Proses membuat BST



Proses membuat BST





Operasi pada Binary Search Tree

- Search Operation (Operasi Pencarian)
  - Algoritmanya

```
If root == NULL
    return NULL;
If number == root->data
    return root->data;
If number < root->data
    return search(root->left)
If number > root->data
    return search(root->right)
```

Ilustrasi:
https://www.javatpoint.com/searching-in-binary-search-tree

- Insert Operation (Operasi Penambahan)
  - Algoritmanya

```
If node == NULL
    return createNode(data)
if (data < node->data)
    node->left  = insert(node->left, data);
else if (data > node->data)
    node->right = insert(node->right, data);
return node;
```

Ilustrasi:
https://www.javatpoint.com/insertion-in-binary-search-tree

Operasi pada Binary Search Tree

- Deletion Operation (Operasi Penghapusan)
  - Algoritmanya

Delete (TREE, ITEM)

- **Step 1:** IF TREE = NULL
  Write "item not found in the tree" ELSE IF ITEM < TREE -> DATA
  Delete(TREE->LEFT, ITEM)
  ELSE IF ITEM > TREE -> DATA
  Delete(TREE -> RIGHT, ITEM)
  ELSE IF TREE -> LEFT AND TREE -> RIGHT
  SET TEMP = findLargestNode(TREE -> LEFT)
  SET TREE -> DATA = TEMP -> DATA
  Delete(TREE -> LEFT, TEMP -> DATA)
  ELSE
  SET TEMP = TREE
  IF TREE -> LEFT = NULL AND TREE -> RIGHT = NULL
  SET TREE = NULL
  ELSE IF TREE -> LEFT != NULL
  SET TREE = TREE -> LEFT
  ELSE
  SET TREE = TREE -> RIGHT
  [END OF IF]
  FREE TEMP
  [END OF IF]
- **Step 2:** END

Ilustrasi:
https://www.javatpoint.com/deletion-in-binary-search-tree

Tree Traversal

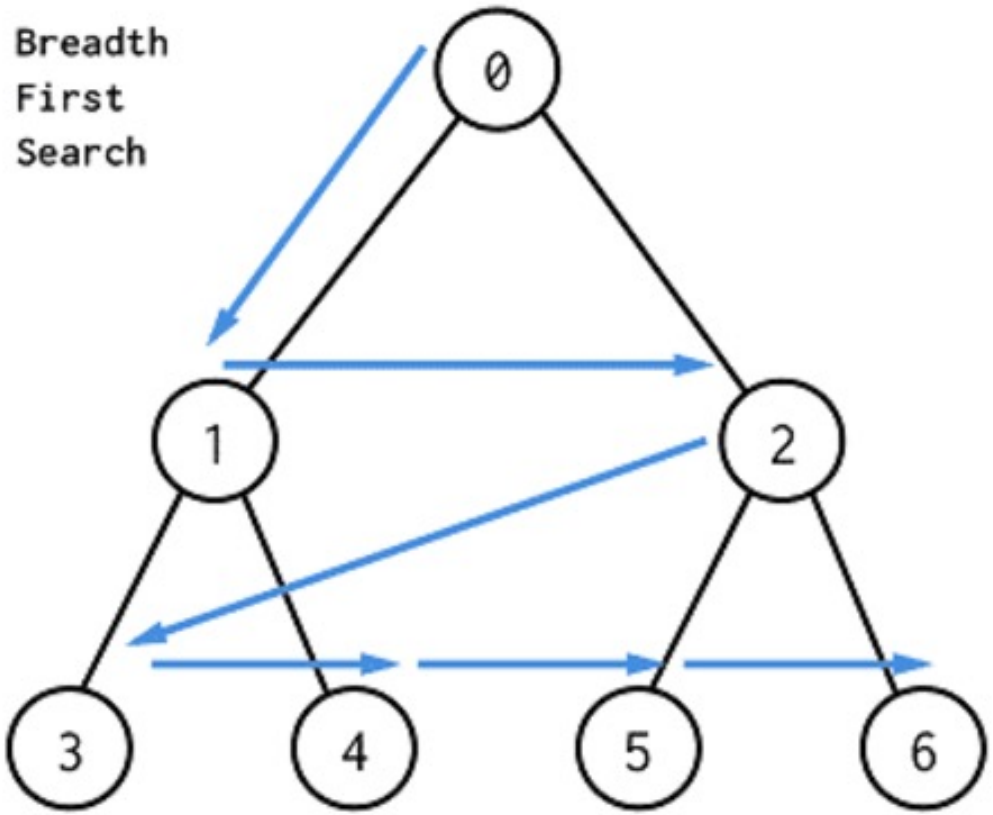
Definisi Tree Traversal

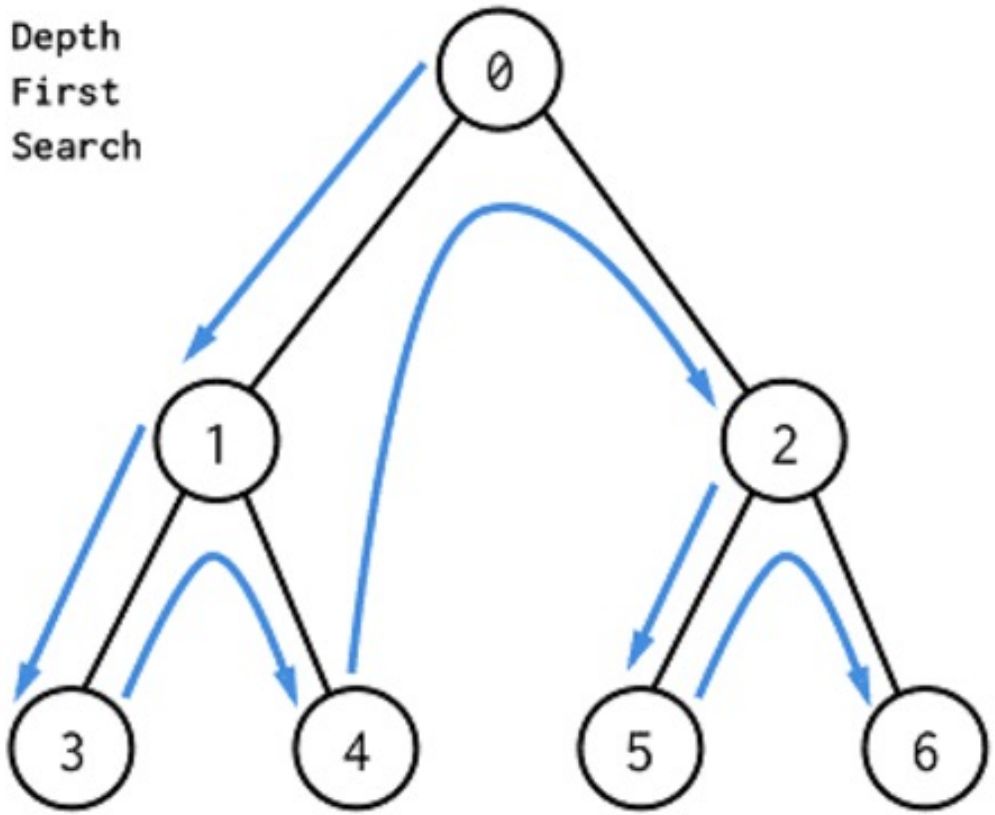
- Traversal maksudnya adalah **penelurusan**
- Tree Traversal adalah menelusuri setiap node yang ada pada sebuah tree
- Misalnya, kita ingin menambahkan data atau mencari data yang ada di dalam tree, maka kita perlu melakukan penelurusan setiap node yang ada di dalam tree tersebut
- Pada linear struktur data seperti array, linkedlist, stack dan queue hanya ada satu cara untuk melakukan penelusuran
- Sementara pada tree, penelusuran dapat dilakukan dengan beberapa cara

Jenis Tree Traversal

- Tree traversal terbagi menjadi 3 jenis menurut order (urutan) penelusuran, yaitu
  - Inorder Traversal
  - Preorder Traversal
  - Postorder Traversal
- Ketiga jenis traversal tersebut digolongkan kedalam teknik **Depth First Traversal**
- Ada teknik lain yang digunakan untuk melakukan penelusuran yaitu **Breadth First Traversal (Level Order Traversal)**. Namun, teknik ini tidak dibahas pada materi struktur data ini

Jenis Tree Traversal





Jenis Tree Traversal

Breadth First Search (BFS)	Depth First Search (DFS)
BFS visit nodes level by level.	DFS visit nodes by depth.
BFS is slower and require more memory.	DFS is faster and require less memory.
It uses queue data structure.	It uses stack data structure.
Application: Find shortest path between 2 node, Find all connected component etc.	Application: Topological sorting, find articulation point, solving puzzle etc.

Depth First Search: Time Complexity- O(|V|+|E|) where V is vertex and E is edge.
Space Complexity- O(|V|)

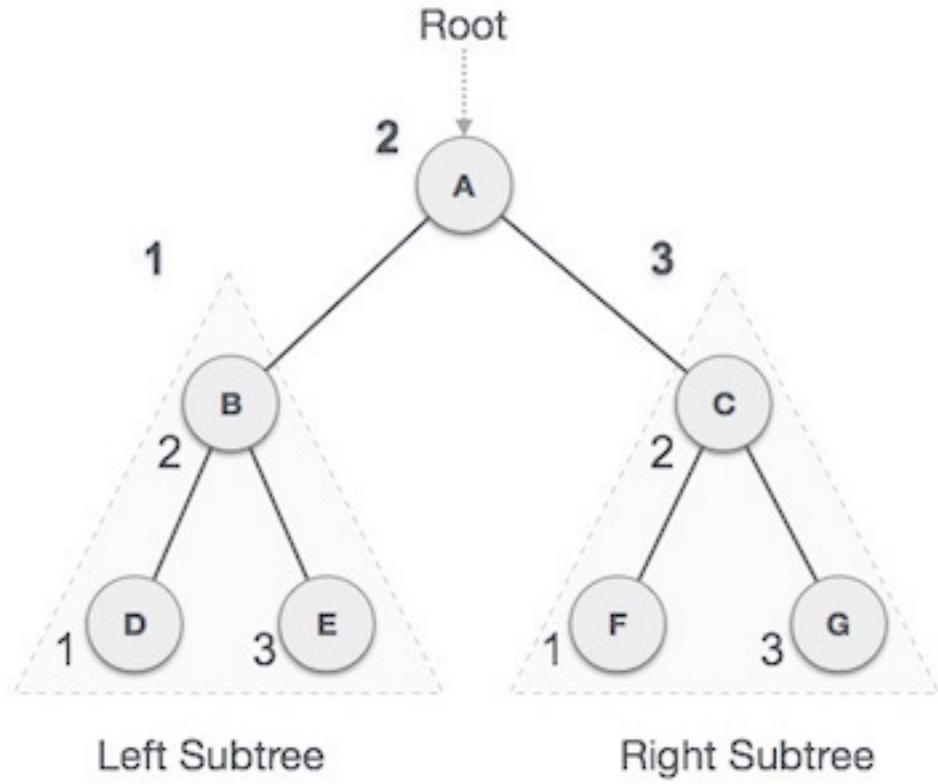
Breadth First Search: Time Complexity - O(|V|+|E|) where V is vertex and E is edge.
Space Complexity- O(|V|)

Inorder Traversal

- Langkah-Langkah pada **inorder traversal** yaitu
  - Pertama, telusuri semua node subtree pada sebelah kiri root
  - Kemudian, penelusuran dilanjutkan ke node root
  - Terakhir, telusuri semua node subtree pada sebelah kanan root

```
inorder(root->left)
display(root->data)
inorder(root->right)
```

- Implementasi
  - https://www.tutorialspoint.com/data_structures_algorithms/tree_traversal_in_c.htm



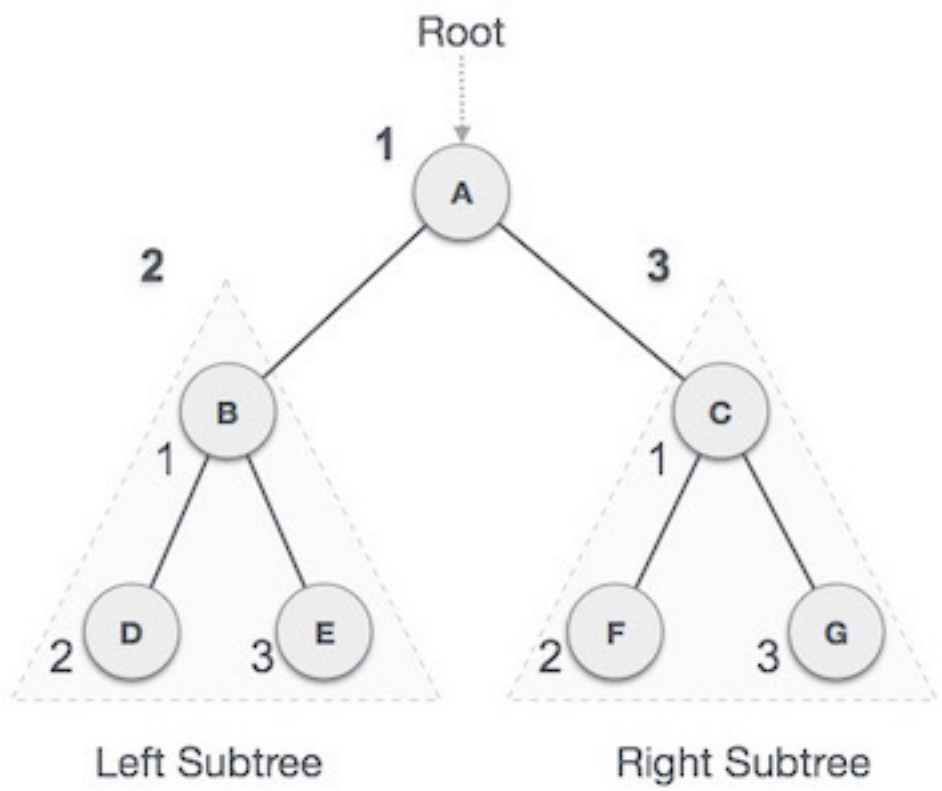
D → B → E → A → F → C → G

Preorder Traversal

- Langkah-Langkah pada **preorder traversal** yaitu
  - Pertama, kunjungi node root
  - Kemudian, telusuri semua node yang ada pada sebelah kiri subtree
  - Terakhir, telusuri semua node yang ada pada sebelah kanan subtree

```
display(root->data)
preorder(root->left)
preorder(root->right)
```

- Implementasi
  - https://www.tutorialspoint.com/data_structures_algorithms/tree_traversal_in_c.htm



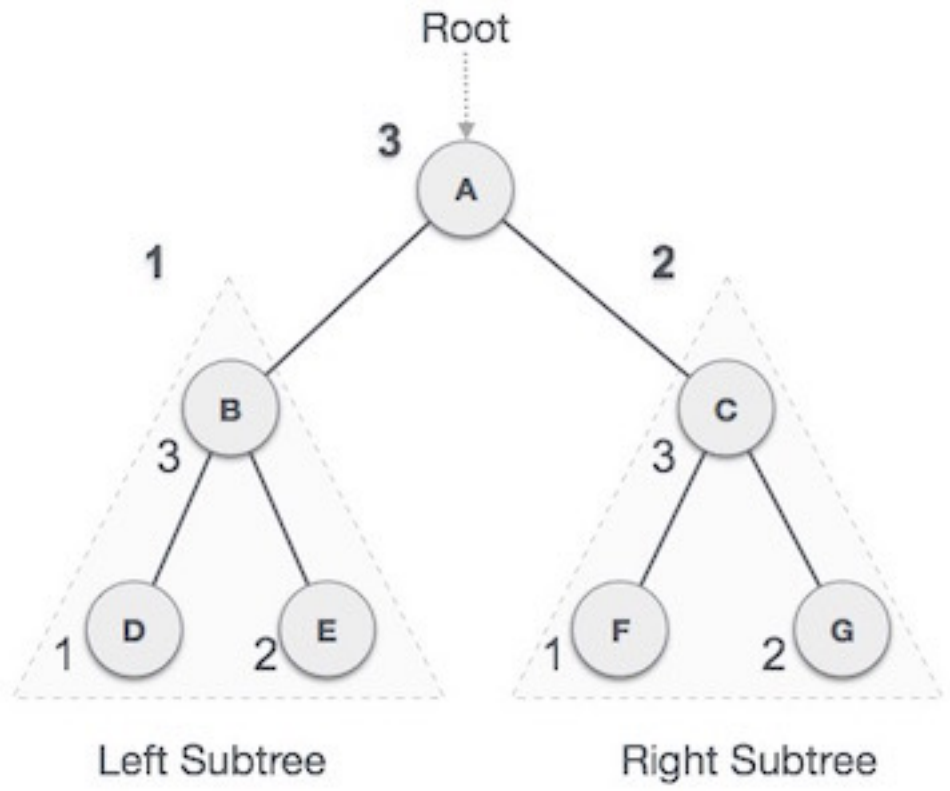
A → B → D → E → C → F → G

Postorder Traversal

- Langkah-Langkah pada **postorder traversal** yaitu
  - Pertama, telusuri semua node yang ada pada sebelah kiri subtree
  - Kemudian, telusuri semua node yang ada pada sebelah kanan subtree
  - Terakhir, telusuri node root

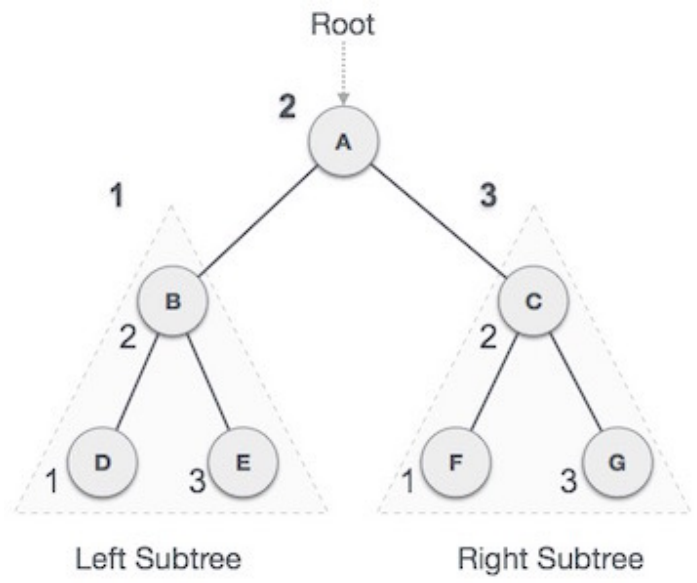
```
postorder(root->left)
postorder(root->right)
display(root->data)
```

- Implementasi
  - https://www.tutorialspoint.com/data_structures_algorithms/tree_traversal_in_c.htm

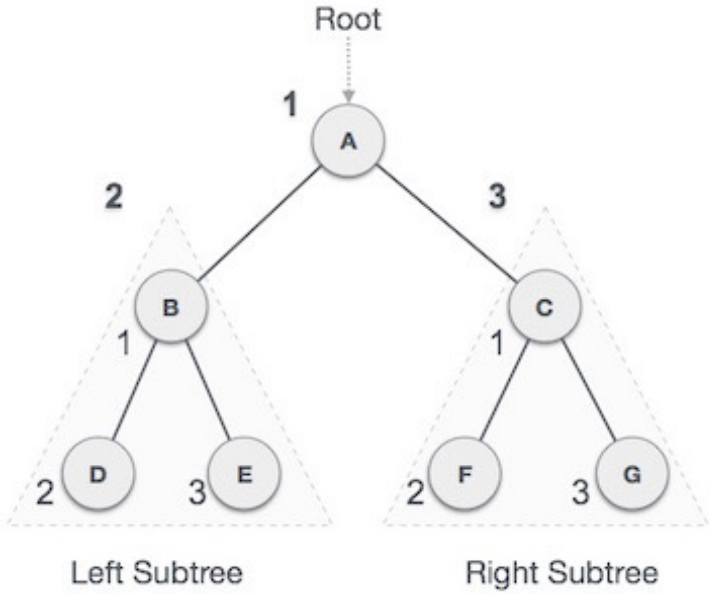


D → E → B → F → G → C → A

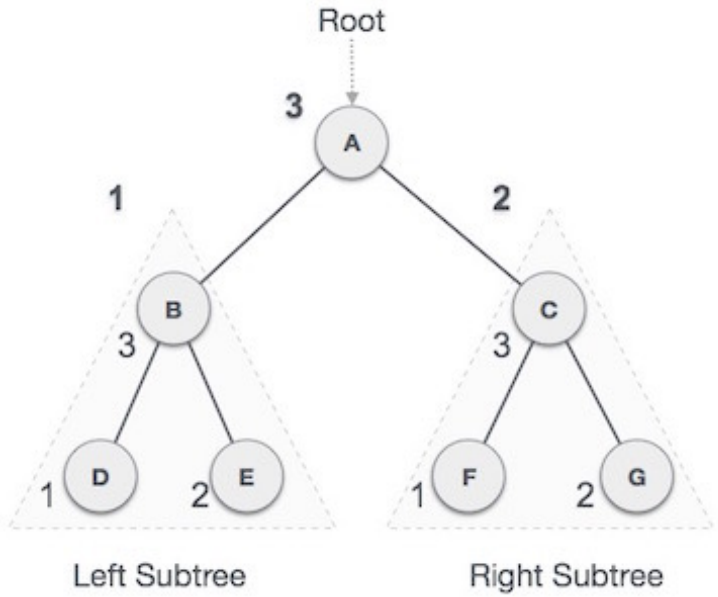
Perbandingan Tree Traversal



In-Order Traversal
D → B → E → A → F → C → G



Pre-Oder Traversal
A → B → D → E → C → F → G



Post-Order Traversal
D → E → B → F → G → C → A

Practice

- In-Order Tree Traversal
  - https://practice.geeksforgeeks.org/problems/inorder-traversal/1
- Pre-Order Tree Traversal
  - https://practice.geeksforgeeks.org/problems/preorder-traversal/1
- Post-Order Tree Traversal
  - https://practice.geeksforgeeks.org/problems/postorder-traversal/1

Q & A

Referensi

- https://www.tutorialspoint.com/data_structures_algorithms/tree_traversal.htm
- https://www.programiz.com/dsa/tree-traversal
- https://www.javatpoint.com/data-structure-tree-travelsal