INF218
Struktur Data & Algoritma

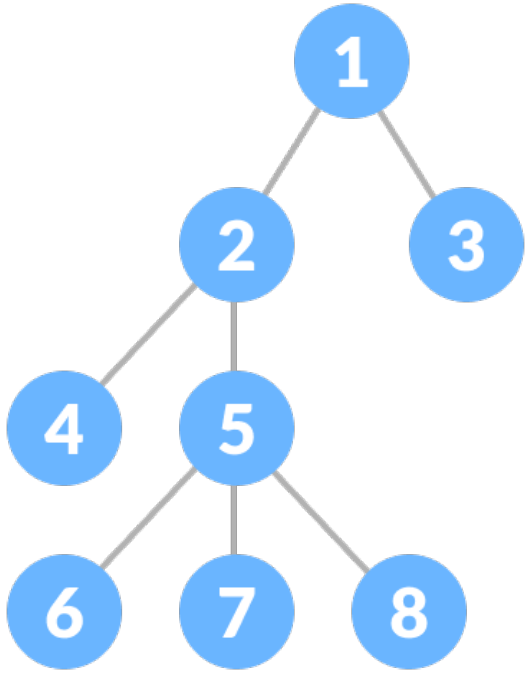
Struktur Data Tree

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

Definisi Tree

Definisi Tree

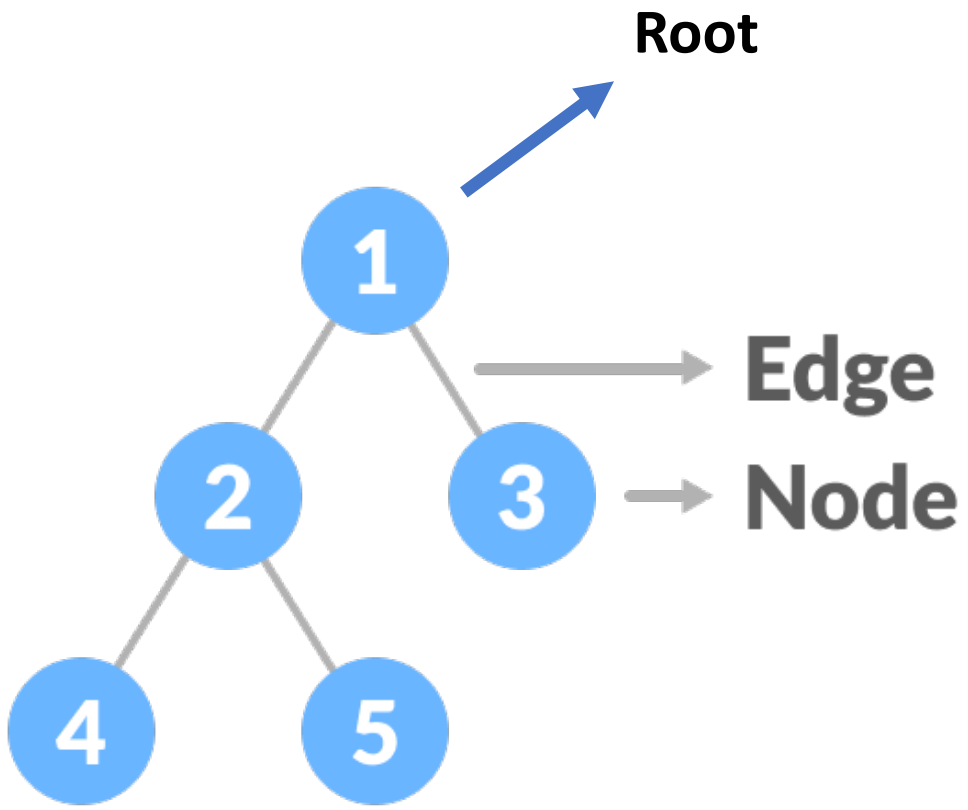
- **Tree** adalah non-linear hirarki struktur data yang terdiri dari **node** yang saling terhubung melalui **edge**
- Mengapa perlu tree?
  - Struktur data yang lain seperti array, linkedlist, stack dan queue merupakan linear struktur data yang datanya saling terhubung secara berurutan
  - Pada linear struktur data, penambahan data yang banyak akan membutuhkan waktu yang banyak pula sehingga tidak bisa digunakan pada komputasi saat ini
  - Tree menjadi solusi karena memungkinkan untuk mengakses data secara cepat dan mudah karena tree merupakan non-linear struktur data



Tree

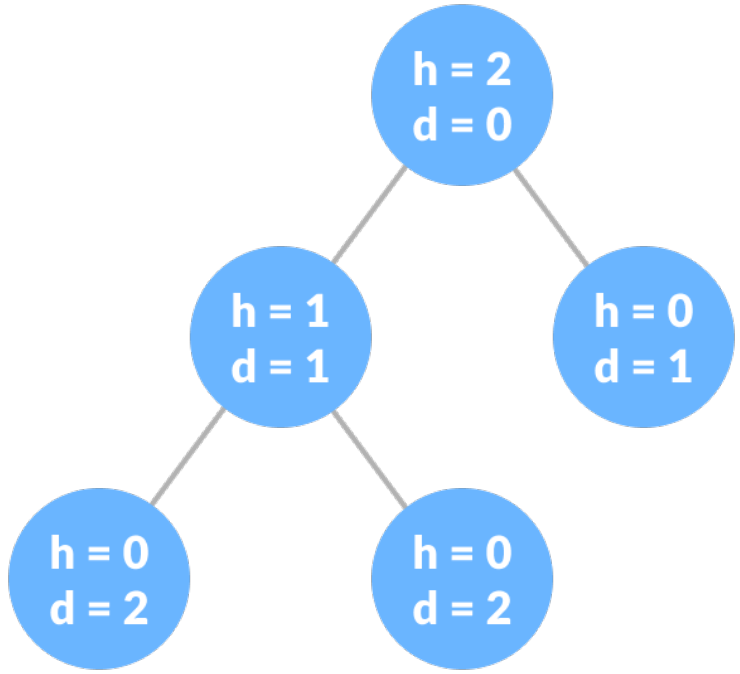
Terminologi Tree

- **Nodes**
  - Sebuah entitas yang mengandung key or value dan pointer ke nodes dibawahnya (child nodes)
  - Node yang terakhir dari setiap path disebut **leaf nodes or external nodes** yang tidak memiliki hubungan (link)/pointer ke child node
  - Node yang memiliki setidaknya satu child nodes disebut **internal node**
- **Edge**
  - Link/penghubung antara dua node
- **Root**
  - Node yang teratas dari sebuah tree



Terminologi Tree

- **Height of a node**
  - Jumlah edges dari sebuah node ke node terakhir di bawahnya (leaf) atau path terjauh dari node ke leaf
- **Depth of a node**
  - Jumlah edges dari root ke node tersebut
- **Height of a tree**
  - Tinggi dari sebuah root node atau kedalaman dari sebuah node
- **Degree of a node**
  - Jumlah total percabangan dari sebuah node



Height and depth of each node in a tree

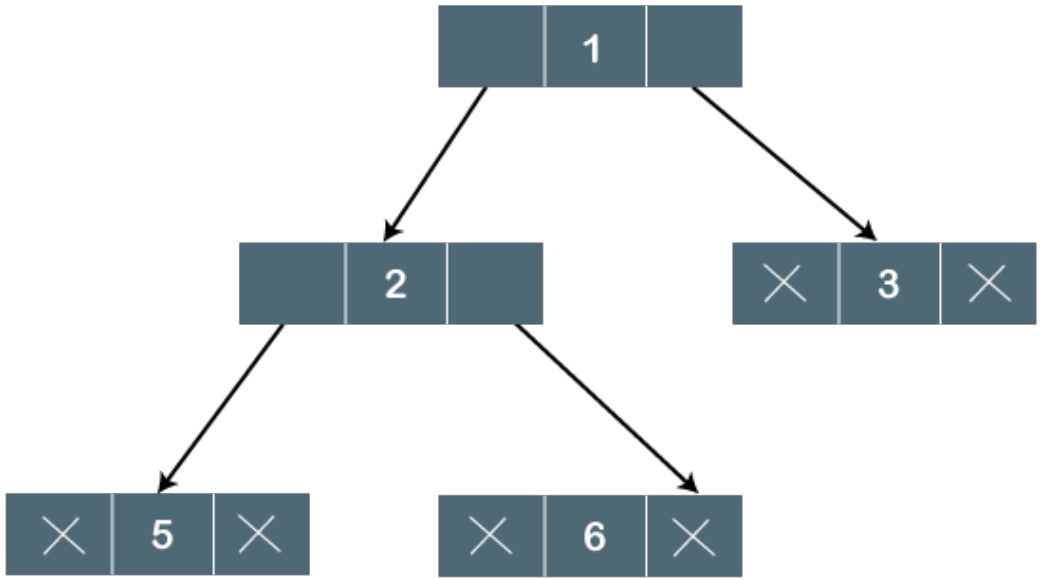
Jenis-jenis Tree

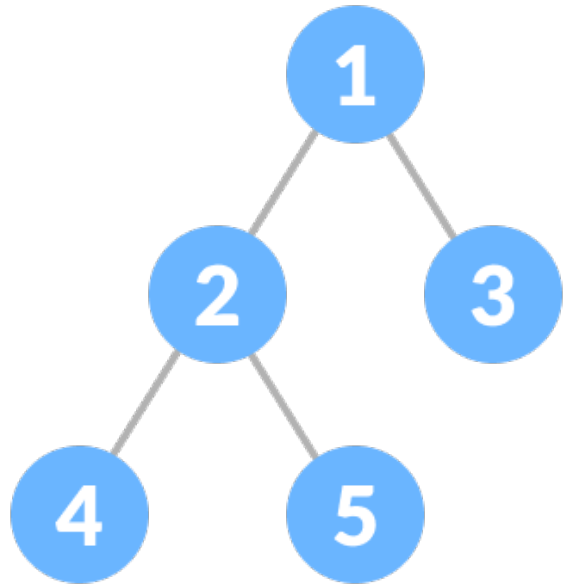
- **Binary Tree**
  - Sebuah struktur data tree yang setiap parent node dapat mempunyai paling banyak 2 children nodes
- **Binary Search Tree**
  - Sebuah struktur data tree yang memiliki aturan seperti binary tree, namun BST disebut sebagai ordered binary tree sehingga sangat efisien untuk proses pencarian
- **AVL Tree**
  - Sebuah self-balancing binary search tree yang mana setiap node akan menjaga sebuah balance factor diantara nilai -1, 0, dan 1
- **B-Tree**
  - Sebuah self-balancing BST yang mana setiap node dapat mengandung lebih dari 1 key dan lebih dari 2 children nodes. B-Tree ada bentuk umum dari BST

Binary Tree

Binary Tree

- Sebuah tree yang setiap parent node memiliki paling banyak 2 (dua) child node
- Dinamai BT karena setiap node dapat memiliki 0, 1 atau 2 children

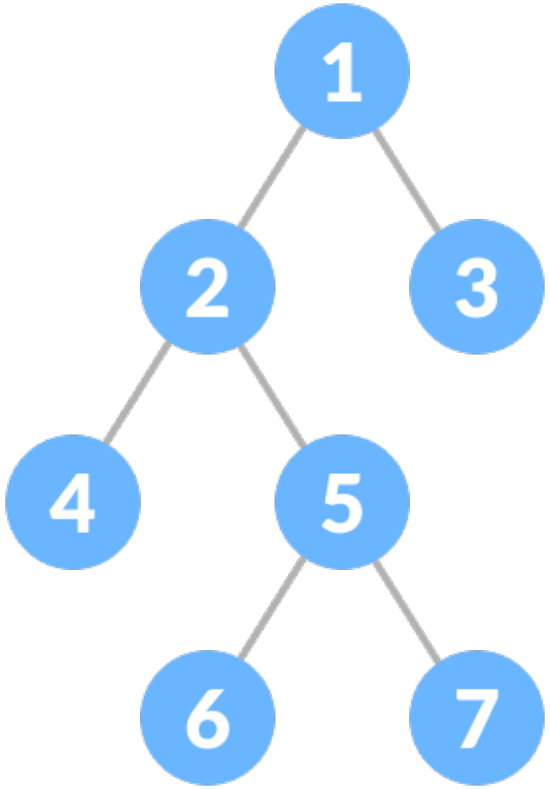




Binary Tree

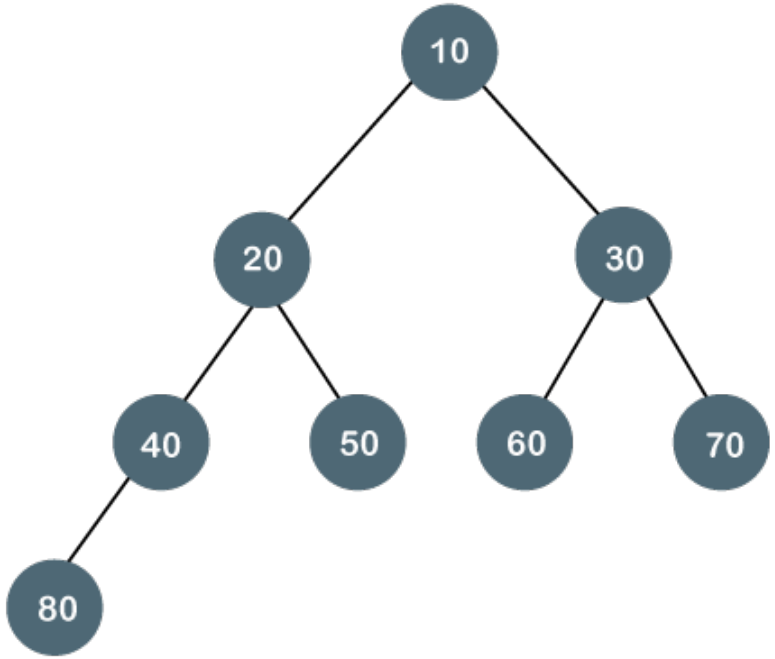
Jenis Binary Tree

- Full Binary Tree
  - Jenis binary tree yang setiap parent/internal node mempunyai 2 (dua) atau tidak memiliki children sama sekali
  - Tree ini juga disebut dengan **proper binary tree**
- Theorema Full Binary Tree
  - https://www.programiz.com/dsa/full-binary-tree
- Implementation in C
  - https://www.programiz.com/dsa/full-binary-tree#c-code



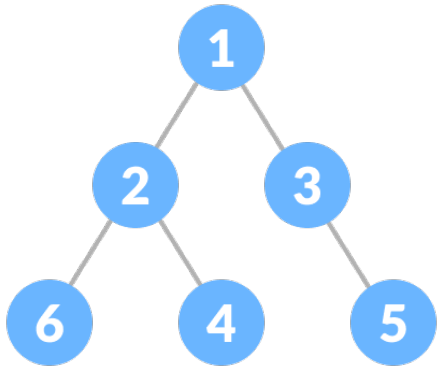
Jenis Binary Tree

- Complete Binary Tree
  - Jenis binary tree yang semua node harus terisi di setiap level, terkecuali level terakhir
  - Pada level terakhir, semua nodes harus sebisa mungkin bersandar ke kiri
  - Pada complete binary tree, node harus ditambahkan mulai dari kiri
- Implementation
  - https://www.programiz.com/dsa/complete-binary-tree#c-code

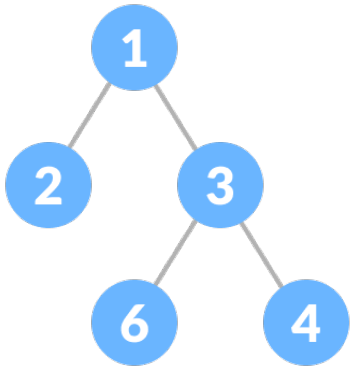


Jenis Binary Tree

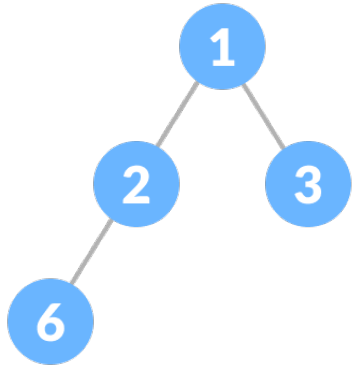
- Perbandingan Complete Binary Tree dan Full Binary Tree



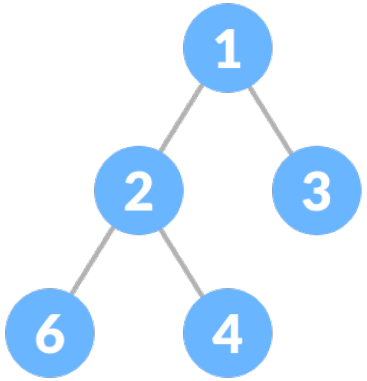
✗ Full Binary Tree
✗ Complete Binary Tree



✓ Full Binary Tree
✗ Complete Binary Tree



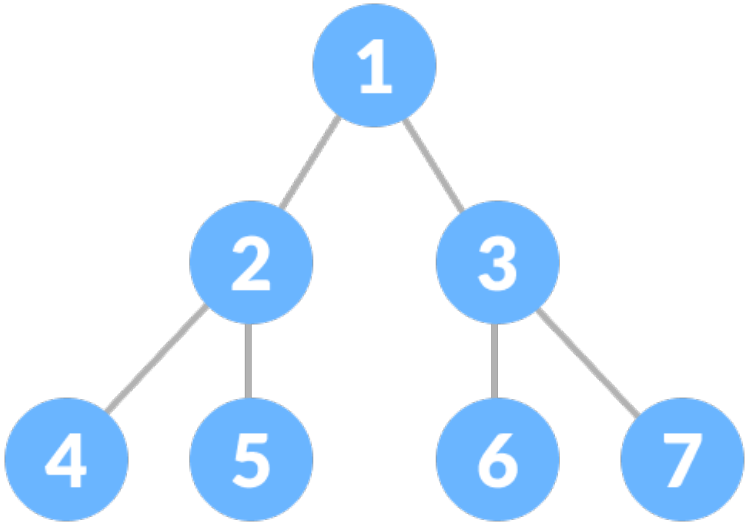
✗ Full Binary Tree
✓ Complete Binary Tree

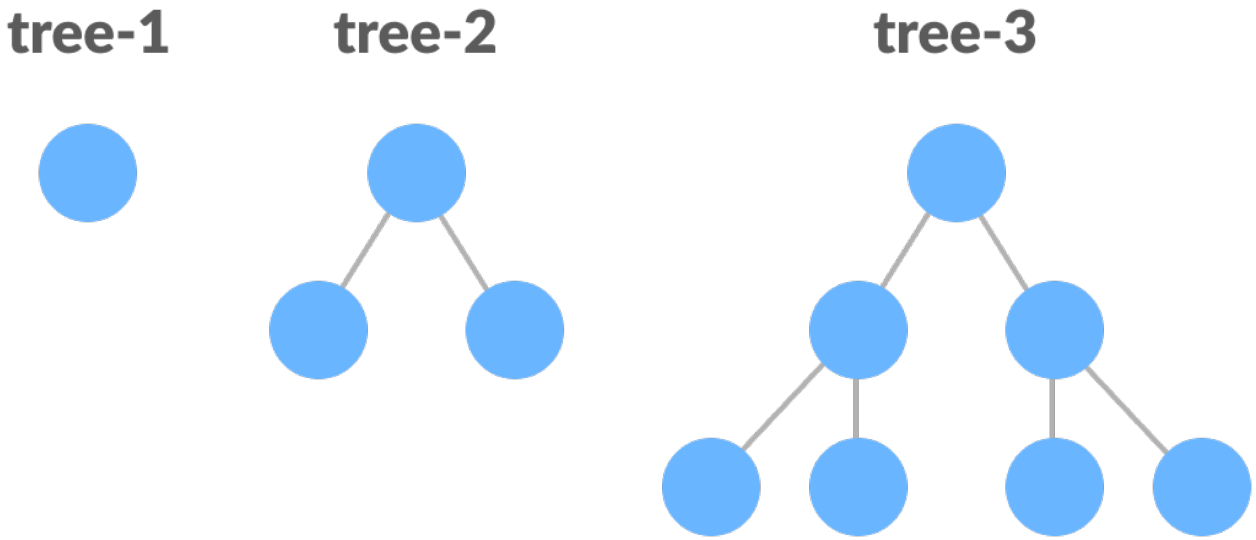


✓ Full Binary Tree
✓ Complete Binary Tree

Jenis Binary Tree

- Perfect Binary Tree
  - Jenis binary tree yang setiap internal node harus mempunyai 2 (dua) children dan leaf node pada level yang sama
- Jika tree tidak memiliki children, maka disebut perfect binary tree dengan h = 0

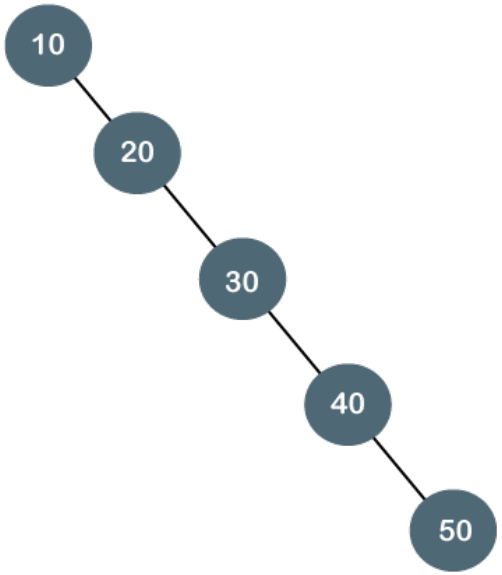




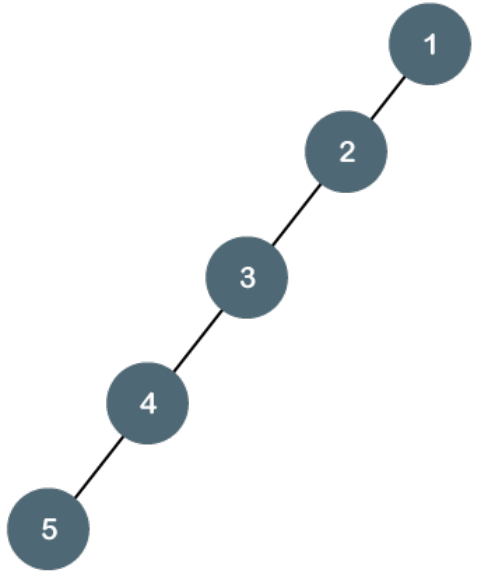
Implementation:
https://www.programiz.com/dsa/perfect-binary-tree#c-code

Jenis Binary Tree

- Degenerate/Pathological Binary Tree
  - Jenis binary tree yang hanya memiliki satu children baik pada sebelah kiri atau kanan



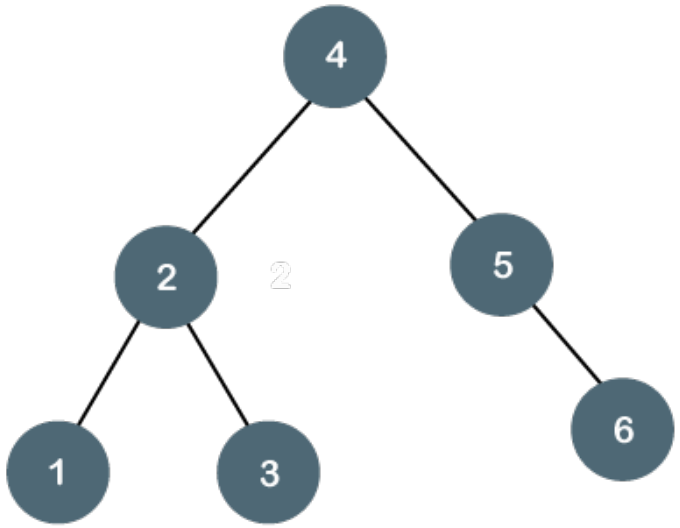
Right-skewed or right child only



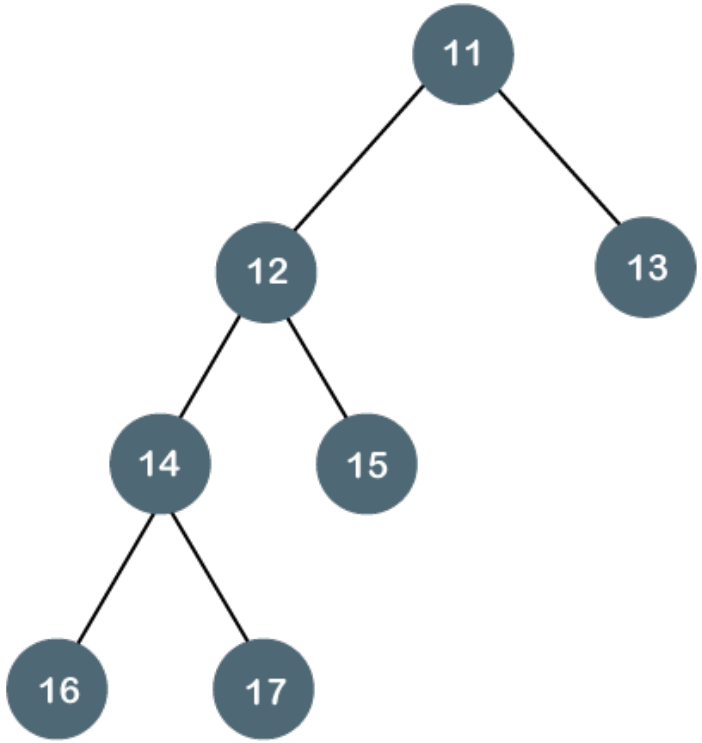
Left-skewed or left child only

Jenis Binary Tree

- Balance Binary Tree
  - Jenis binary tree yang height bagian kiri dan kanan dari tree berbeda maksimum 1



Balance Binary Tree
H(L) – H(R) = 0

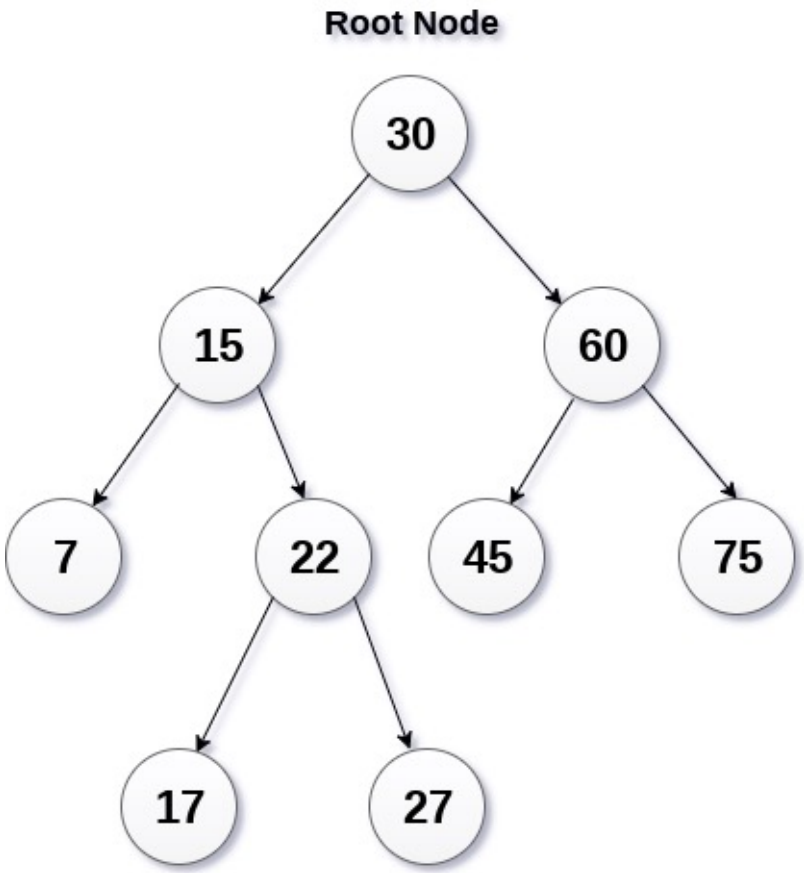


Bukan Balance Binary Tree
H(L) – H(R) = 2

Binary Search Tree

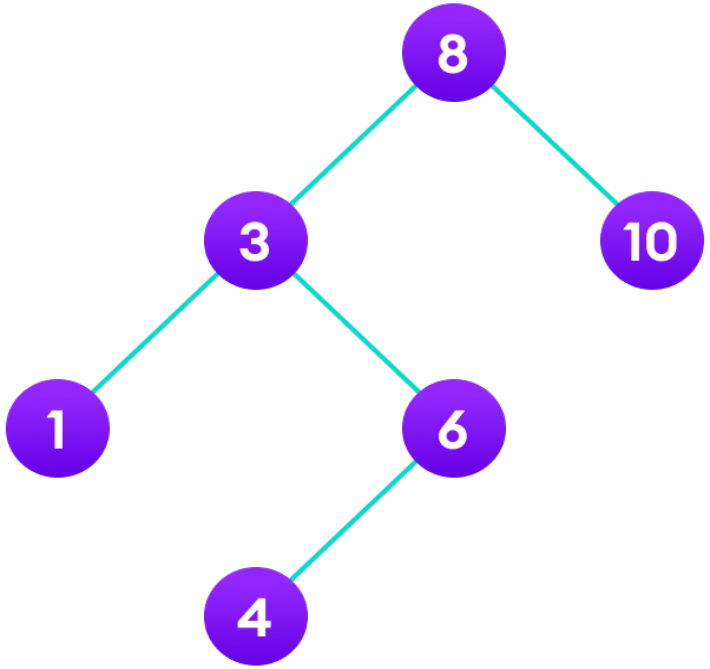
Binary Search Tree

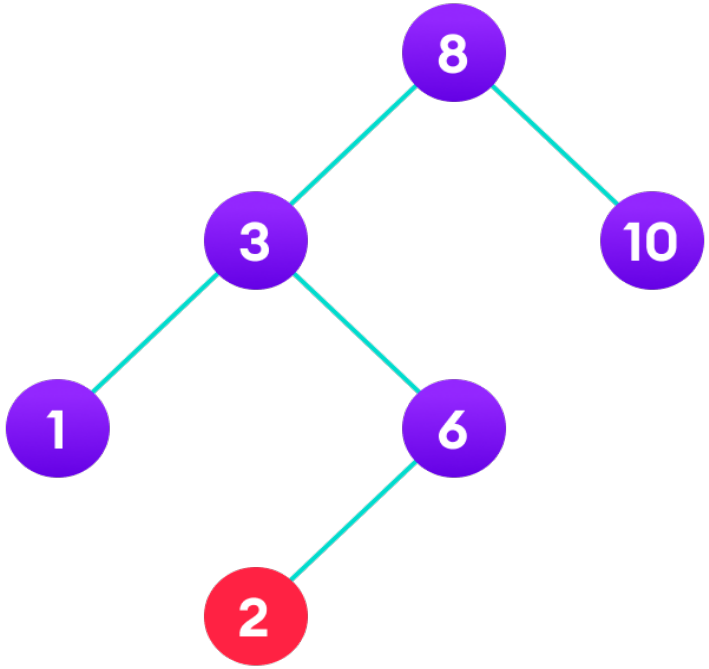
- Binary search tree merupakan sebuah struktur data yang mengizinkan kita secara cepat untuk melakukan pengurutan dari susunan bilangan atau disebut juga dengan **ordered binary tree**
- Pada binary search tree, **nilai dari semua node pada bagian kiri < nilai root**
- Sementara, **nilai dari semua node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



Binary Search Tree

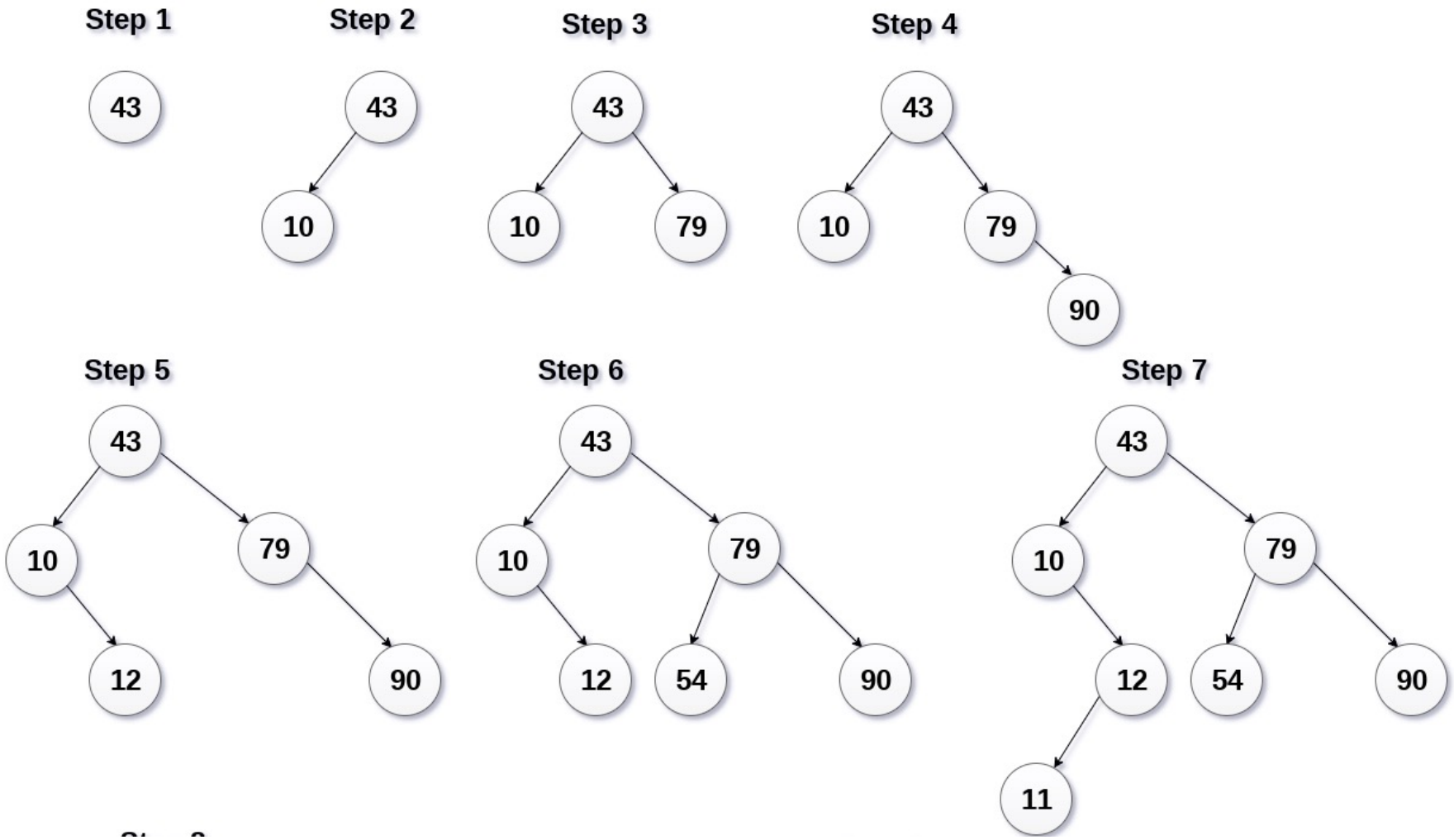
Binary Search Tree



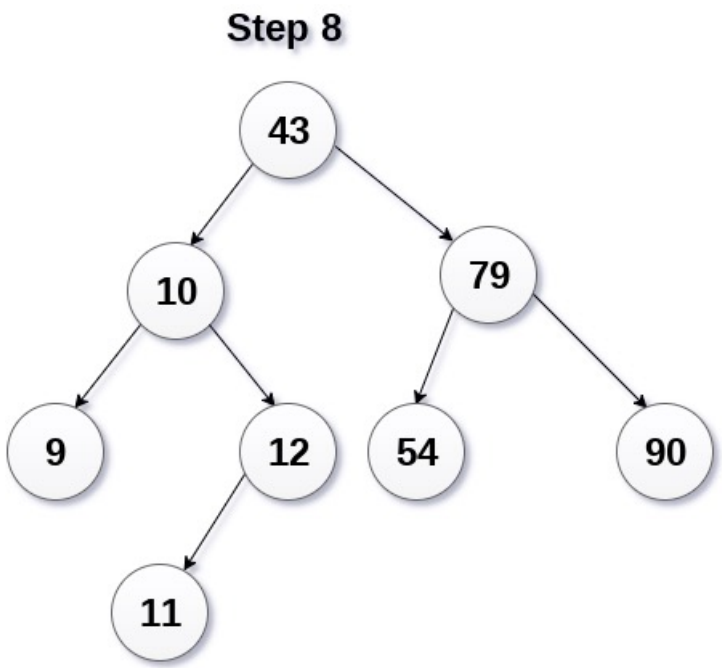


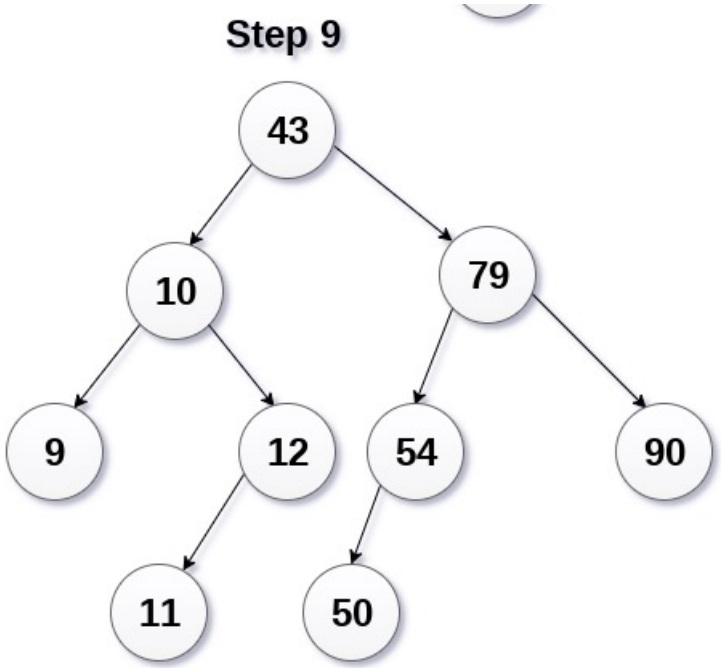
Bagian tree sebelah kanan adalah bukan binary search tree karena terdapat nilai lebih kecil di sebelah kanan “3” yaitu “2”

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

Q & A

Referensi

- https://www.programiz.com/dsa/trees
- https://www.programiz.com/dsa/binary-search-tree
- https://www.javatpoint.com/binary-search-tree