Laporan Praktikum Struktur Data

Semester Gasal 2021/2022

NIM	**71200634**
Nama Lengkap	**Egi Granaldi Ginting**
Minggu ke / Materi	**13 / Tree**

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

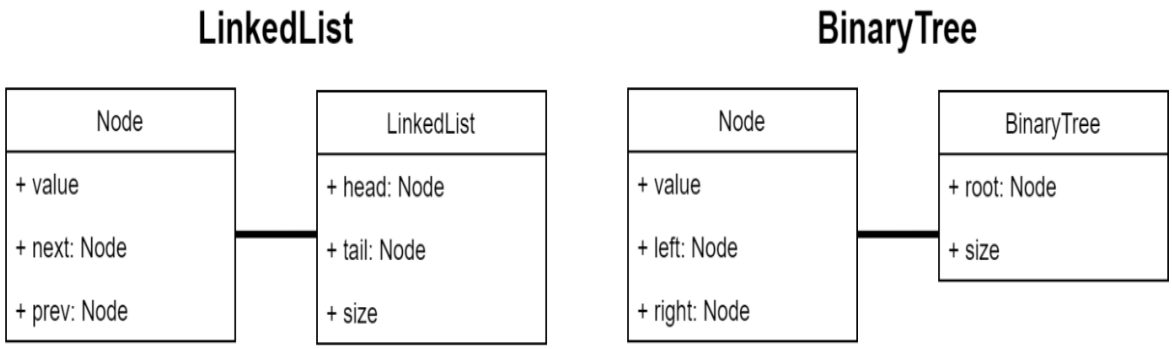
PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2021

BAGIAN 1: Tree

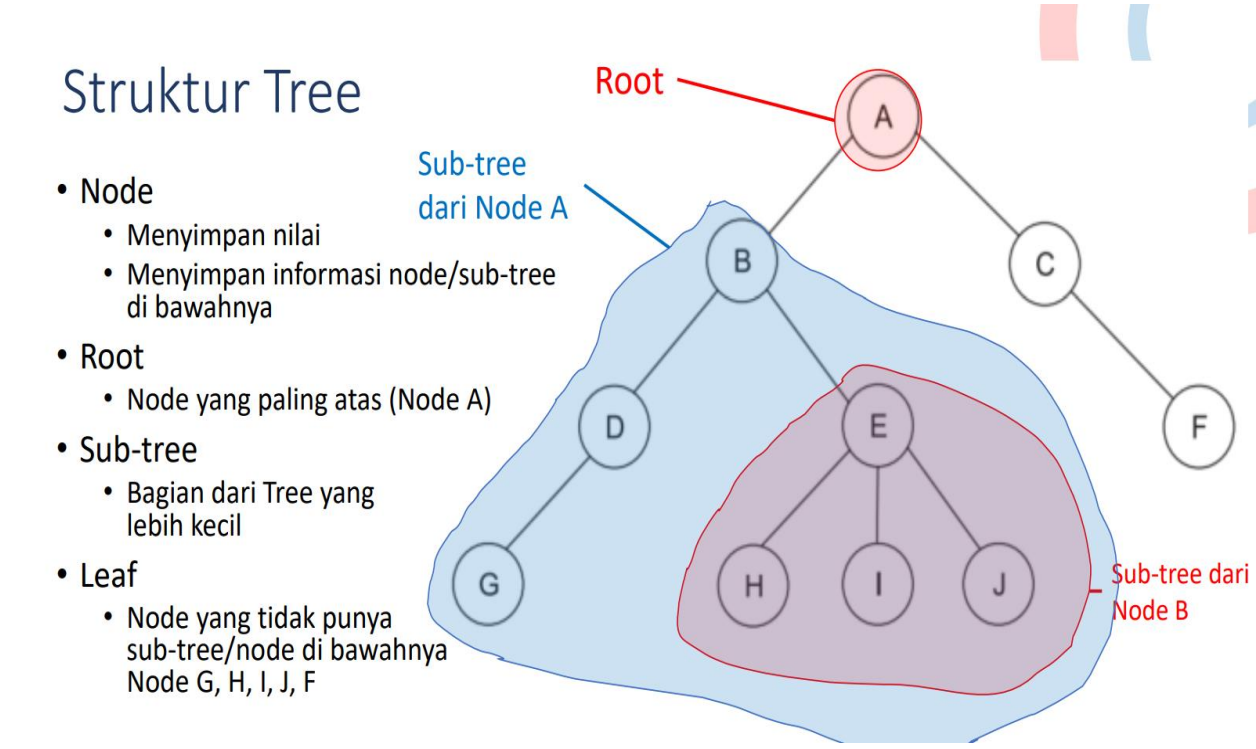
Tree adalah sebuah struktur data yang secara bentuk menyerupai sebuah pohon, yang terdiri dari serangkaian node(simpul) yang saling berhubungan. Node-node tersebut dihubungkan oleh sebuah vector.Setiap node dapat memiliki 0 atau lebih node anak.Sebuah node yang memiliki node anak disebut node induk.Sebuah node anak hanya memiliki satu node induk.Sesuai konvesi ilmu komputer, tree bertumbuh kebawah bukan keatas.

Node yang berada di pangkal tree disebut node root, sedangkan node yang berada paling ujung pada piramida tree disebut node leaf(daun).

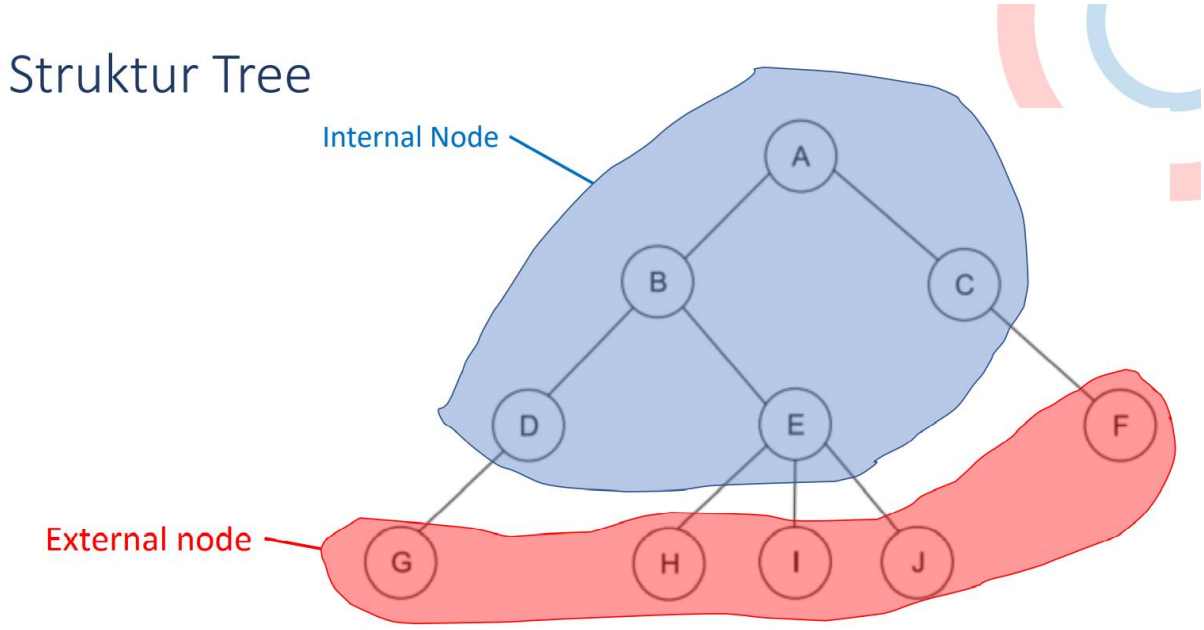
- Struktur data non-linear
  - Membentuk hierarkhi/tingkatan
- Merupakan recursive data struktur
  - Dalam implementasinya, Sebagian / seluruhnya tersusun dari struktur data yang sama dalam bentuk yang lebih kecil.Contohnya Linked List/Tree



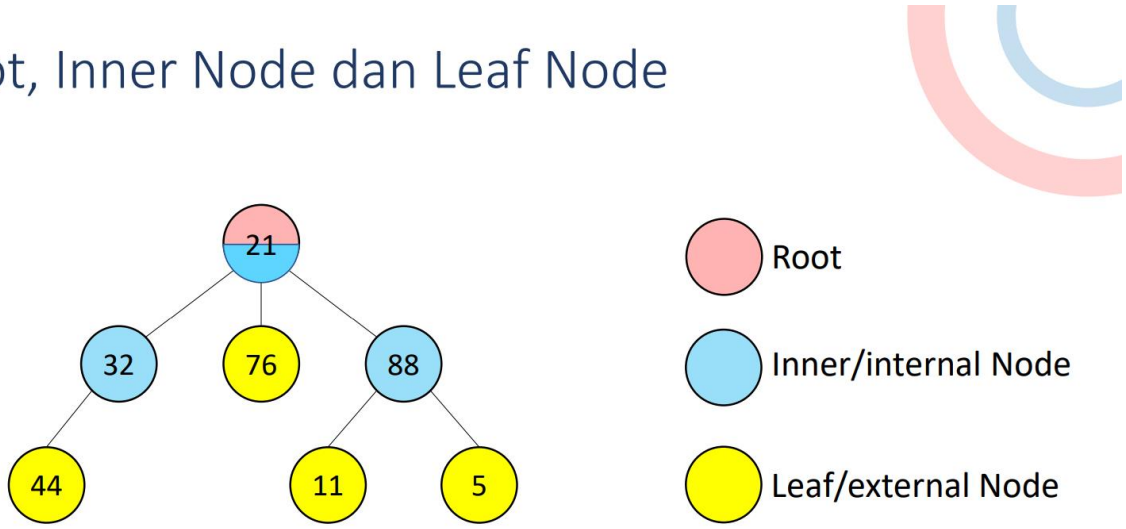
Gambar 1 Linked list & Binary Tree



Gambar 2 Struktur Tree



Gambar 3 Struktur Tree



Gambar 4 Root, Inner Node dan Leaf Node pada struktur tree

Jenis – Jenis Tree

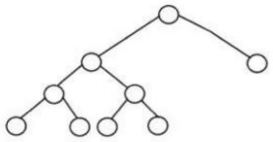
- General Tree
  - Setiap Node-nya bisa memiliki child node dalam jumlah berapapun
  - Dalam implementasinya biasanya child nodes disimpan dalam list
- Binary Tree
  - Setiap Node-nya maksimal dua child nodes
  - Biasanya child nodes disimpan dengan nama left node dan right node
  - Paling banyak dipakai dalam algoritma, memiliki banyak variasi bentuk seperti Binary search tree, AVL Tree, B-Tree, T-Tree, Red Black Tree, dll….

Binary Tree

- Setiap node maksimal memiliki dua child nodes, biasanya dinamakan left node dan right node.
- Binary Tree yang lebih spesifik biasanya memiliki aturan pada peletakan child node, apakah di sebelah kiri atau kanan
- Dari bentuknya, Binary Tree bisa digolongkan menjadi beberapa bentuk berikut ini:
  - Full binary tree, Complete binary tree, Perfect binary tree, Balanced binary tree, Degenerate/skewed binary tree

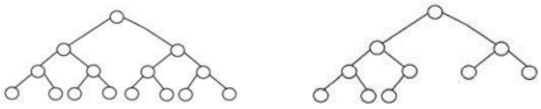
Jenis – Jenis Binary Tree

- Full Binary Tree
  Semua nodes memiliki dua child kecuali external (leaf) node



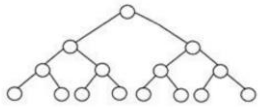
Gambar 5 Full Binary Tree

- Complete Binary Tree
  Seluruh level terisi oleh node, kecuali pada level terendahnya



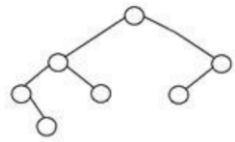
Gambar 6 Complete Binary Tree

- Perfect Binary Tree
  Semua internal nodes memiliki dua child, kemudian seluruh leaf node berada pada level yang sama



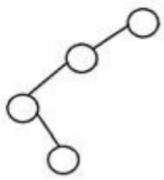
Gambar 7 Perfect Binary Tree

- Balanced Binary Tree
  Height dari tree adalah Log n, dengan n adalah jumlah nodes di dalam tree.Selisih tinggi sebelah kiri dan kanan tidak boleh lebih dari 1



Gambar 8 Balanced Binary Tree

- Degenerate/Skewed Binary Tree
  Seluruh internal node hanya memiliki 1 child saja

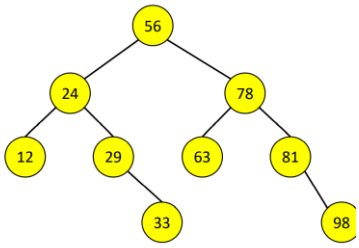


Gambar 9 Degenerate/Skewed Binary Tree

ADT Tree

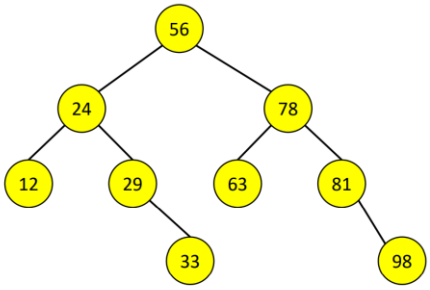
- Operasi – operasi pada Binary Tree
  - Insert
  - Traversing (Find, Print)
  - Delete
- Mode Traversing bisa bermacam-macam
  - Level Order / Breadth First Traversal
  - Depth First Traversal
- Depth First Traversal terdiri dari 3 macam, yaitu Pre-order, In-order dan Post – Order

Traversing Level Order



Hasilnya adalah : 56, 24, 78, 12, 29, 63, 81, 33, 98

Traversing Pre Order



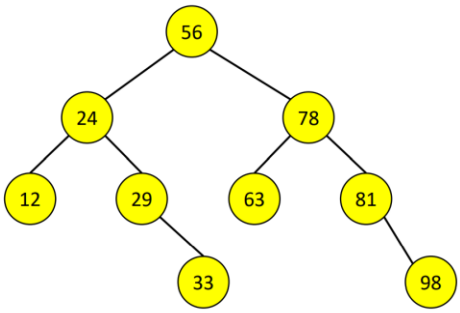
Urutannya adalah :

- Kunjungi root node
- Kunjungi Left node secara rekursif
- Kunjungi right node secara rekursif

Root => Left => Right

Hasilnya adalah : 56, 24, 12, 29, 33, 78, 63, 81, 98

Traversing In-Order



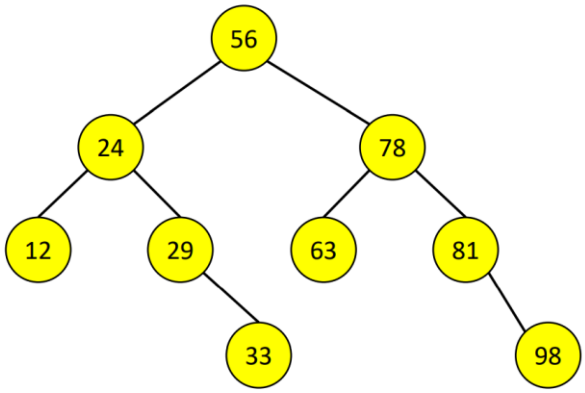
Urutannya adalah :

- Kunjungi left node secara rekursif
- Kunjungi root node
- Kunjungi right node secara rekursif

Left => Root => Right

Hasilnya adalah : 12, 24, 29, 33, 56, 63, 78, 81, 98

Traversing Post – Order



Urutannya adalah :

- Kunjungi left node secara rekursif
- Kunjungi right node secara rekursif
- Kunjungi root node

Left => Right => Root

Hasilnya adalah : 12, 33, 29, 24, 63, 98, 81, 78, 56

Menghapus Node pada Binary Tree

- Binary Tree biasa
  - Jika lead node, hapus langsung
  - Jika bukan leaf node, pilih satu node yang berada di paling kanan dari tree untuk menggantikannya

BAGIAN 2: SOAL UNGUIDED

Unguided 1

```
class Node:
    def __init__ (self, data, parent):
        self._data = data
        self._children = []
        self._parent = parent

    def addChild(self, data):
        self._children.append(data)

    def operator(self):
        return self._data

    def parent(self):
        return self._parent

    def children(self):
        return self._children

    def isRoot(self):
        return self._parent is None

    def isExternal(self):
        return len(self._children) == 0


root =Node(100,None)
a = Node(15,root)
b = Node(23,root)
c = Node(20,a)
d = Node(78,a)
e = Node(24,b)
f = Node(42,c)
g = Node(51,c)
h = Node(76,d)
i = Node(12,d)
j = Node(15,d)
k = Node(8,e)
l = Node(33,e)
a.addChild(c)
a.addChild(d)
b.addChild(e)
c.addChild(f)
```

```
c.addChild(g)
d.addChild(h)
d.addChild(i)
d.addChild(j)
e.addChild(k)
e.addChild(l)
root.addChild(a)
root.addChild(b)


print(root.operator())
print("anak dari node 100 adalah :")
for i in root.children():
    print(i.operator(),end=' ')
print("")
print("anak dari node 15 adalah :")
for i in a.children():
    print(i.operator(),end=' ')
print("")
print("anak dari node 23 adalah :")
for i in b.children():
    print(i.operator(),end=' ')
print(" ")
print("anak dari node 20 adalah :")
for i in c.children():
    print(i.operator(),end=' ')
print("")
print("anak dari node 78 adalah :")
for i in d.children():
    print(i.operator(),end=' ')
print("")
print("anak dari node 24 adalah :")
for i in e.children():
    print(i.operator(),end=' ')
print(" ")
```

```
100
anak dari node 100 adalah :
15 23
anak dari node 15 adalah :
20 78
anak dari node 23 adalah :
24
anak dari node 20 adalah :
42 51
anak dari node 78 adalah :
76 12 15
anak dari node 24 adalah :
8 33
PS C:\Users\M S I>
```

Unguided 2

```python
class Node_Tree:
    def __init__(self, data, parent):
        self._data = data
        self._parent = parent
        self._child = []

    def insert_Child(self, data):
        self._child.append(data)

    def isExternal(self):
        return len(self._child) == 0


    def child(self):
        return self._child

    def get_data(self):
        return self._data

    def preorder(self,node):
        if node is not None:
            print(node.get_data(), end = ' ')
            for data_child in node.child():
                self.preorder(data_child)
```

```
    def sum(self):
        sum_data = 0
        if (self.isExternal() == False):
            for child_data in self._child:
                sum_data += child_data.sum()
        return sum_data + self._data

root = Node_Tree(100, None)
node_a = Node_Tree(15, root)
node_b = Node_Tree(23, root)
node_c = Node_Tree(20, node_a)
node_d = Node_Tree(78, node_a)
node_e = Node_Tree(24, node_b)
node_f = Node_Tree(42, node_c)
node_g = Node_Tree(51, node_c)
node_h = Node_Tree(76, node_d)
node_i = Node_Tree(12, node_d)
node_j = Node_Tree(15, node_d)
node_k = Node_Tree(8, node_e)
node_l = Node_Tree(33, node_e)
root.insert_Child(node_a)
root.insert_Child(node_b)
node_a.insert_Child(node_c)
node_a.insert_Child(node_d)
node_b.insert_Child(node_e)
node_c.insert_Child(node_f)
node_c.insert_Child(node_g)
node_d.insert_Child(node_h)
node_d.insert_Child(node_i)
node_d.insert_Child(node_j)
node_e.insert_Child(node_k)
node_e.insert_Child(node_l)
print('='*8+' Data Tree '+'='*8)
root.preorder(root)
print('\nTotal dari tree:',root.sum())
print('Total dari tree:',node_d.sum())
print('Total dari tree:',node_i.sum())
```

```
PS C:\Users\M S I> & "C:/Users/M S I/AppData/Local/Programs/Python/Python39/python.exe" "d:/Semester 3/Praktikum Struktur Data/laporan 8(2).py"
====== Data Tree ======
100 15 20 42 51 78 76 12 15 23 24 8 33
Total dari tree: 497
Total dari tree: 181
Total dari tree: 12
PS C:\Users\M S I>
```

Penjelasan :

Pada Unguided 1 dan 2 kali ini kita diminta untuk membuat program Tree dengan struktur yang sudah diperintahkan.Setelah itu kita asumsikan data pada tree selalu berupa angka, dan kita membuat function sum yang dapat menghitung keseluruhan node sampai yang paling dalam atau paling bawah.Pertama kita membuat struktur data tree terlebih dahulu Setelah itu kita menambahkan node ke dalam tree. Jika data yang akan kita masukkan lebih besar daripada elemen root, maka akan diletakkan di node sebelah kanan, sebaliknya juga jika data lebih kecil maka akan diletakkan di node sebelah kiri. Untuk data pertama akan menjadi elemen root.Setelah itu function preorder digunakan untuk mencetak node yang dikunjungi yaitu, left dan right.Lalu kita menambahkan sum untuk menjumlahkan keseluruhan node pada data kita tersebut.

Daftar Pustaka :

https://adoc.pub/tree-structure-struktur-pohon.html