

PRACTICUM REPORT
ALGORITHM AND DATA STRUCTURES
MODUL 10 : BINARY TREE



Disusun Oleh :
ONIC AGUSTINO
L200234275
X

INFORMATICS ENGINEERING
FACULTY OF COMMUNICATION AND INFORMATICS
UNIVERSITAS MUHAMMADIYAH SURAKARTA
YEARS 2024/2025

1.11 Questions

1. Create function `ukuranPohon(akar)` which will get the size of a binary tree.

- **Program Code**

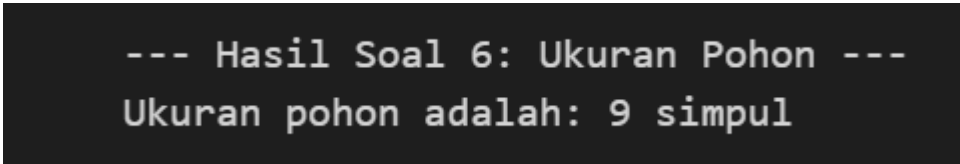
```
class _SimpulPohonBiner(object):
    def __init__(self, data):
        self.data = data # Menyimpan data pada simpul
        self.kiri = None # Link ke anak kiri, awalnya kosong
        self.kanan = None # Link ke anak kanan, awalnya kosong

# --- Fungsi untuk Soal Nomor 6 ---
def ukuranPohon(akar):
    """Menghitung dan mengembalikan jumlah total simpul (ukuran) dari pohon."""
    if akar is None:
        return 0
    else:
        # Ukuran adalah 1 (simpul saat ini) + ukuran sub-pohon kiri + ukuran
        sub-pohon kanan
        return 1 + ukuranPohon(akar.kiri) + ukuranPohon(akar.kanan)
```

Code 1.1 The code of `ukuranPohon(akar)`

The class `_SimpulPohonBiner` represents a node in a binary tree, storing some data along with links to its left and right children. The function `ukuranPohon` calculates the total number of nodes in the tree by recursively summing 1 (for the current node) with the size of the left and right subtrees.

- **Practicum Result Screenshot**



```
--- Hasil Soal 6: Ukuran Pohon ---
Ukuran pohon adalah: 9 simpul
```

Gambar 1.2 The output `ukuranPohon(akar)`

2. Create function `tinggiPohon(akar)` which will get the height of a binary tree.

- **Program Code**

```
class _SimpulPohonBiner(object):
    def __init__(self, data):
        self.data = data # Menyimpan data pada simpul
        self.kiri = None # Link ke anak kiri, awalnya kosong
        self.kanan = None # Link ke anak kanan, awalnya kosong

# --- Fungsi untuk Soal Nomor 7 ---
```

```
def tinggiPohon(akar):
    """Menghitung dan mengembalikan tinggi (jumlah level) dari pohon."""
    if akar is None:
        return 0
    else:
        # Tinggi adalah 1 (level saat ini) + tinggi maksimum dari sub-pohon kiri
        # atau kanan
        tinggi_kiri = tinggiPohon(akar.kiri)
        tinggi_kanan = tinggiPohon(akar.kanan)
        return 1 + max(tinggi_kiri, tinggi_kanan)
```

Code 2.2 The code of tinggiPohon(akar)

The class _SimpulPohonBiner defines a binary tree node that holds data and has pointers to left and right child nodes. The function tinggiPohon calculates the height of a binary tree by recursively finding the maximum height between the left and right subtrees and adding 1 for the current level.

- Practicum Result Screenshot

```
--- Hasil Soal 7: Tinggi Pohon ---
Tinggi pohon adalah: 4 level
```

Gambar 2.2 The output tinggiPohon(akar)

3. Create a function that prints data for each node as well as the level at which the node is located. Please choose whether to use preorder

traversal, inorder traversal, or postorder traversal. An example of a piece of the result is as below (if you use preorder traversal).

```
>>> cetakDataDanLevel(A)
```

Ambarawa, level 0

Bantul, level 1

Denpasar, level 2

Enrekang, level 2

Halmahera Timur, level 3

Cimahi, level 1

- Program Code

```
class _SimpulPohonBiner(object):
    def __init__(self, data):
        self.data = data # Menyimpan data pada simpul
        self.kiri = None # Link ke anak kiri, awalnya kosong
        self.kanan = None # Link ke anak kanan, awalnya kosong

# --- Fungsi untuk Soal Nomor 8 ---
def cetakDataDanLevel(akar):
    """Fungsi utama untuk mencetak data dan level setiap simpul."""
    # Memanggil fungsi pembantu rekursif dengan level awal 0
    _cetakDataDanLevelBantu(akar, 0)

def _cetakDataDanLevelBantu(subpohon, level):
    """Fungsi pembantu yang melakukan penelusuran preorder secara rekursif."""
    if subpohon is not None:
        # 1. Kunjungi simpul (cetak data dan levelnya)
        print(subpohon.data, ", level", level)
        # 2. Telusuri sub-pohon kiri di level berikutnya
        _cetakDataDanLevelBantu(subpohon.kiri, level + 1)
        # 3. Telusuri sub-pohon kanan di level berikutnya
        _cetakDataDanLevelBantu(subpohon.kanan, level + 1)

if __name__ == "__main__":

    # --- Membuat Pohon Contoh ---
    # Struktur pohon ini dibuat sama persis dengan contoh pada halaman 121 di
    # buku.
    print("Membangun pohon dari contoh buku...")
    A = _SimpulPohonBiner('Ambarawa')
    B = _SimpulPohonBiner('Bantul')
    C = _SimpulPohonBiner('Cimahi')
    D = _SimpulPohonBiner('Denpasar')
    E = _SimpulPohonBiner('Enrekang')
    F = _SimpulPohonBiner('Flores')
    G = _SimpulPohonBiner('Garut')
    H = _SimpulPohonBiner('Halmahera Timur')
    I = _SimpulPohonBiner('Indramayu')

    # Menghubungkan simpul-simpul sesuai struktur pohon di buku
    A.kiri = B; A.kanan = C
    B.kiri = D; B.kanan = E
    C.kiri = F; C.kanan = G
    E.kiri = H
    G.kanan = I
    print("Pohon berhasil dibuat.\n")
```

```

# --- Memanggil Fungsi dan Menampilkan Hasil ---

# Memanggil fungsi ukuranPohon
ukuran = ukuranPohon(A)
print("--- Hasil Soal 6: Ukuran Pohon ---")
print("Ukuran pohon adalah:", ukuran, "simpul\n")

# Memanggil fungsi tinggiPohon
tinggi = tinggiPohon(A)
print("--- Hasil Soal 7: Tinggi Pohon ---")
print("Tinggi pohon adalah:", tinggi, "level\n")

# Memanggil fungsi cetakDataDanLevel
print("--- Hasil Soal 8: Cetak Data dan Level (Preorder) ---")
cetakDataDanLevel(A)

```

Code 3.1 The code of cetakdanLevel

This code defines a binary tree node and uses preorder traversal to print each node's data alongside its depth level in the tree. The helper function `_cetakDataDanLevelBantu` visits the current node first, then recursively traverses the left and right subtrees, increasing the level count as it goes deeper.

By running this script, a specific tree structure is built using Indonesian city names, and then three results are printed: the number of nodes (`ukuranPohon`), the height of the tree (`tinggiPohon`), and the data of each node with its corresponding level (`cetakDataDanLevel`). Want me to draw a visual of the tree structure for you?

- **Practicum Result Screenshot**

```

--- Hasil Soal 8: Cetak Data dan Level (Preorder) ---
Ambarawa , level 0
Bantul , level 1
Denpasar , level 2
Enrekang , level 2
Halmahera Timur , level 3
Cimahi , level 1
Flores , level 2
Garut , level 2
Indramayu , level 3

```

Gambar 3.2 The output cetakDataDanLevel

Binary Tree

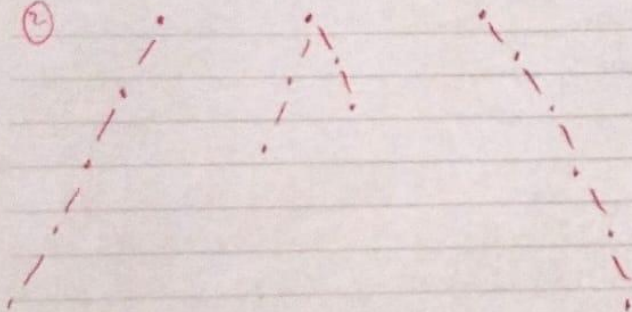
① a) Maximum Level

- $n = 10$, max: 10 level
- $n = 35$, max: 35 level
- $n = 76$, max: 76 level
- $n = 345$, max: 345 level

b) Minimum Level

- $n = 10$, $\lceil \log_2 10 \rceil + 1 = 3.32 + 1 = 4$ level
- $n = 35$, $\lceil \log_2 35 \rceil + 1 = 5.13 + 1 = 6$ level
- $n = 76$, $\lceil \log_2 76 \rceil + 1 = 6.25 + 1 = 7$ level
- $n = 345$, $\lceil \log_2 345 \rceil + 1 = 8.43 + 1 = 9$ level

②



- ③
- $h = 3$, $2^3 - 1 = 8 - 1 = 7$ Simple
 - $h = 4$, $2^4 - 1 = 16 - 1 = 15$ Simple
 - $h = 5$, $2^5 - 1 = 32 - 1 = 31$ Simple
 - $h = 6$, $2^6 - 1 = 64 - 1 = 63$ Simple

④ Tree (a):

- (a) Properties = Complete
- (b) Size = 9 nodes
- (c) Height = 4 levels
- (d) Width = 4

Tree (b):

- (a) Properties = Full, Perfect, and Complete
- (b) Size = 15 nodes
- (c) Height = 4 levels
- (d) Width = 8

Tree (c):

- (a) Properties : Not full, not Perfect, not complete
- (b) Size : 8 nodes
- (c) Height : 3 levels
- (d) Width : 1

Tree (d):

- (a) Properties : Not full, not perfect, not complete
- (b) Size : 6 nodes
- (c) Height : 3 levels
- (d) Width : 2

Tree (e):

- (a) Properties : Full
- (b) Size : 9 nodes
- (c) Height : 3 levels
- (d) Width : 3

- 5- (a) i. Preorder : 60, 10, 17, 52, 39, 21, 91, 70, 9, 03, 23, 89
 ii. Inorder : 17, 10, 21, 39, 91, 52, 60, 9, 70, 23, 03, 89
 iii. Postorder : 17, 21, 91, 39, 52, 10, 9, 23, 89, 03, 70, 60

(b) Leaf : 17, 21, 91, 9, 23, 89

(c) Internal : 60, 10, 52, 39, 70, 03

(d) Level 1 : 17, 9

(e) i. 03 = 19, 70, 9, 83

ii. 39 = ~~19, 70, 52, 03~~ 19, 70, 39

iii. 9 = 19, 2, 23, 9

iv. 9 = 19, 70, 52, 03, 9

(f) i. = 83, 19,

iii. = 19, 70

iv. = 39, 60, 23

(g) i, 20 = 1

ii, 41 = 0

iii, 60 = 0

iv, 19 = 0