STRUKTUR DATA DAN ALGORITMA

Pembelajaran Digital Kolaboratif Tahun 2025

UNIVERSITAS AN NUUR
UNIVERSITAS INDONESIA TIMUR

2025

Struktur Data & Algoritma

(Project Base Learning)

Oleh:

Andri Triyono

Gafur

Editor :

Eko Supriyadi

2025

UNIVERSITAS AN NUUR

Kata Pengantar

Puji syukur kehadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga Buku Praktikum Struktur Data & Algoritma untuk program studi Strata 1 (S1) Ilmu Komputer ini dapat diselesaikan dengan baik. Buku ini merupakan wujud komitmen kami dalam menyediakan sumber belajar yang relevan dan aplikatif di lingkungan Universitas An Nuur Purwodadi.

Mata kuliah Struktur Data & Algoritma adalah fondasi krusial bagi setiap mahasiswa Ilmu Komputer. Buku ini dirancang khusus dengan pendekatan Project Based Learning (PBL), yang bertujuan untuk menjembatani kesenjangan antara teori yang dipelajari di kelas dengan implementasi nyata di dunia industri.

Keunggulan utama buku ini terletak pada:

1. Fokus Implementasi: Materi disajikan melalui bahasa pemrograman Python dengan penekanan pada kode contoh (*Guided Practice*) dan latihan mandiri (*Unguided Practice*).
2. Relevansi Kontekstual: Setiap pertemuan dibuka dengan *Driving Question* yang mengaitkan konsep Struktur Data (seperti *Stack*, *Queue*, *BST*, dan *Graph*) dengan aplikasi teknologi berskala besar, mulai dari sistem *Undo/Redo* hingga optimasi kecepatan *e-commerce* dan sistem navigasi.
3. Asesmen Berbasis Proyek: Mahasiswa didorong untuk merancang *Mini Project* di akhir setiap unit pembelajaran, culminating pada Proyek Akhir yang menguji kemampuan integrasi konsep.

Kami menyadari bahwa Buku ini tidak luput dari kekurangan. Oleh karena itu, kritik dan saran yang membangun dari berbagai pihak, khususnya pengguna Buku, akan kami terima dengan tangan terbuka demi penyempurnaan di masa mendatang.

Harapan kami, Buku ini tidak hanya menjadi bahan ajar, tetapi juga menjadi referensi praktis yang mampu meningkatkan kompetensi mahasiswa dalam menganalisis dan merancang solusi perangkat lunak yang efisien dan optimal.

Purwodadi, 04 Agustus 2025

Daftar Isi

Buku Pertemuan 1

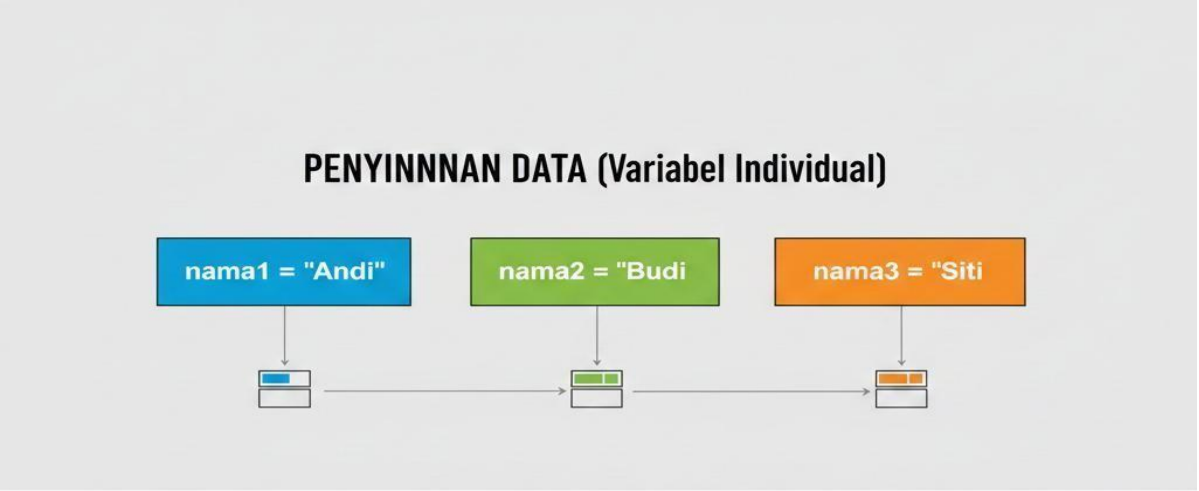
Driving Question:

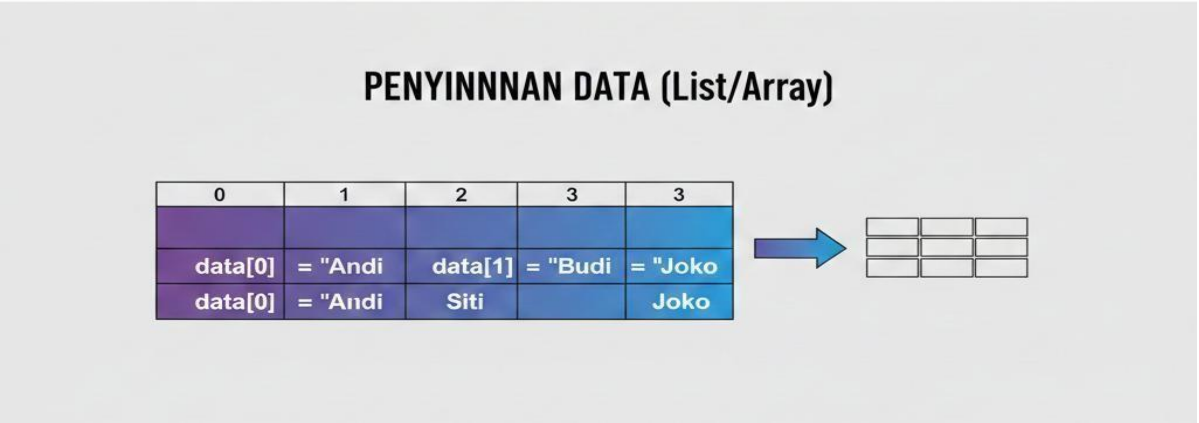
"Mengapa aplikasi seperti Tokopedia, Gojek, atau Instagram membutuhkan struktur data yang efisien untuk bisa digunakan jutaan orang sekaligus?"

Tujuan Pembelajaran:

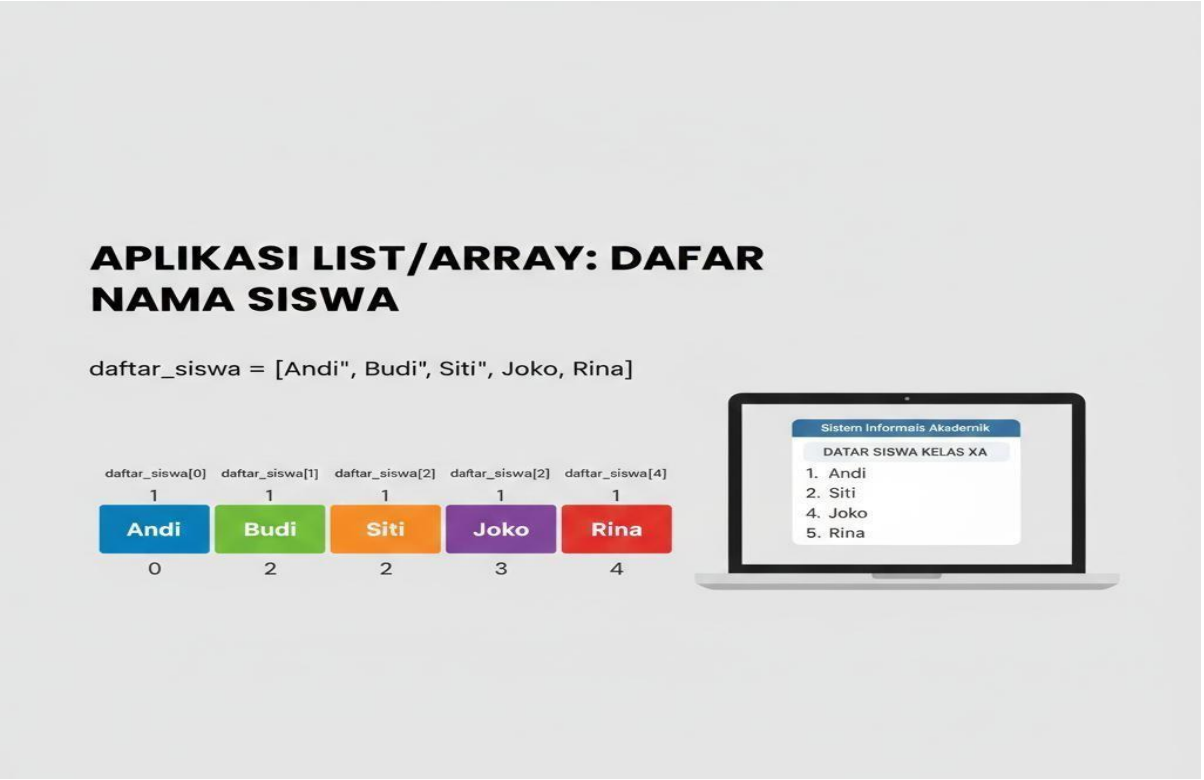
Mahasiswa dapat:

- Menjelaskan struktur data dasar (list, array, dsb)
- Mengimplementasikan array/list di Python
- Membandingkan efisiensi penyimpanan data dengan dan tanpa struktur data(Volk & Turner, 2019)





Gambar 1.1. List dan Array(1)



Gambar 1.2. List dan Array

A. Guided Practice (Panduan + Kode Contoh)

Apa itu Struktur Data?

Struktur data adalah cara menyimpan dan mengatur data agar bisa digunakan secara efisien. Dalam Python, struktur data dasar yang umum digunakan adalah list.

Contoh Program 1: Tanpa Struktur Data

```
a = 80
b = 90
c = 70

print("Nilai Mahasiswa:")
print(a)
print(b)
print(c)
```

Contoh Program 2: Menggunakan List

```
nilai = [80, 90, 70]

print("Nilai Mahasiswa:")
for i in range(len(nilai)):
    print(f"Mahasiswa ke-{i+1}: {nilai[i]}")
```

Visualisasi Sederhana (Opsional)

```
import matplotlib.pyplot as plt

nilai = [80, 90, 70]
nama = ["Andi", "Budi", "Citra"]

plt.bar(nama, nilai)
plt.title("Nilai Mahasiswa")
plt.ylabel("Nilai")
plt.show()
```

B. Unguided Practice (Latihan Mandiri di Kelas)

Mahasiswa mengerjakan soal tanpa melihat pembahasan, bisa dikerjakan langsung di Google Colab.

.**Soal 1:** Apa perbedaan antara variabel biasa dengan list dalam hal penyimpanan data?

.**Soal 2:** Buat program yang menyimpan 5 nama mahasiswa dalam list, lalu cetak satu per satu.

.**Soal 3:** Modifikasi program nilai tadi, dan tampilkan nilai rata-rata dari mahasiswa.

C. Tugas Mandiri (Dikerjakan di Rumah)

Tugas: Daftar Mahasiswa

Buatlah program Python untuk:

- Meminta input dari pengguna: Nama, NIM, dan Nilai dari 3 mahasiswa
- Simpan data tersebut dalam 3 list terpisah
- Tampilkan output seperti berikut:

Mahasiswa 1: NIM = 2104111001, Nama = Andi, Nilai = 85
Mahasiswa 2: NIM = 2104111002, Nama = Budi, Nilai = 78

Bonus: Tambahkan logika: jika nilai >= 75, tampilkan keterangan "Lulus".

Refleksi:

- Saya memahami bahwa struktur data bukan sekadar teori.
- Saya bisa mengaitkan struktur data dengan aplikasi nyata.

Buku Pertemuan 2

Mata Kuliah: Struktur Data & Algoritma
Pertemuan ke: 2
Topik: Array, Pointer, dan Memory Layout
Bahasa: Python
Kode CPMK: CPMK072-1
Taksonomi Bloom: Understanding

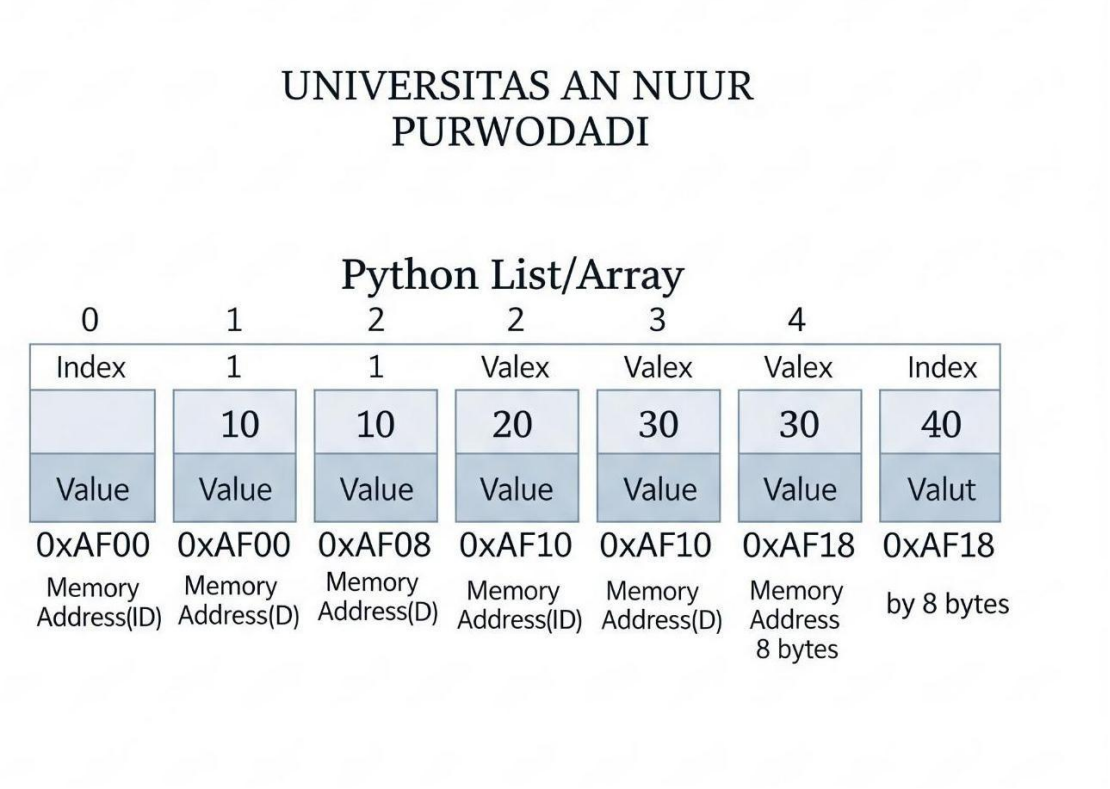
Driving Question:

"Bagaimana menyusun memori untuk data sensor suhu di sebuah smart home secara efisien dan cepat?"

Tujuan Pembelajaran:

Mahasiswa dapat:

- Menjelaskan bagaimana data disimpan dalam array (list)
- Memahami konsep dasar pointer dan referensi dalam Python
- Memvisualisasikan layout memori sederhana(Zhao et al., 2025)



Gambar 2. Memory Layout

A. Guided Practice (Panduan + Contoh Kode)

1. Array dalam Python = List

```
angka = [10, 20, 30, 40]

for i in range(len(angka)):
    print(f"Index ke-{i} berisi: {angka[i]}")
```

Catatan: List di Python bersifat dinamis (panjangnya bisa berubah), namun dalam pembelajaran struktur data, kita bisa memperlakukannya seperti array tetap untuk latihan awal.

2. Visualisasi Memory Layout

```
angka = [10, 20, 30, 40]

for i in range(len(angka)):
    print(f"Index {i}, Value: {angka[i]}, Memory Address: {id(angka[i])}")
```

Penjelasan: id() di Python bisa dianggap sebagai alamat memori. Ini tidak persis seperti pointer C/C++, tapi cukup memberi intuisi.

3. Simulasi Pointer di Python

```
a = 100
b = a  # b mengacu ke nilai yang sama

print(f"Nilai a: {a}, id(a): {id(a)}")
print(f"Nilai b: {b}, id(b): {id(b)}")
```

Ketika kita ubah a atau b, kita bisa lihat apakah memori berubah (mutable vs immutable types).

Bonus Visualisasi (Memory Layout)

```
import matplotlib.pyplot as plt

angka = [10, 20, 30, 40]
alamat = [id(x) for x in angka]

plt.figure(figsize=(10, 2))
plt.bar(range(len(angka)), alamat, tick_label=angka)
plt.title("Visualisasi Alamat Memori (id) Tiap Nilai dalam Array")
plt.ylabel("Alamat Memory (id)")
plt.xlabel("Nilai")
plt.show()
```

B. Unguided Practice (Latihan Mandiri)

Dikerjakan oleh mahasiswa di kelas, tanpa melihat solusi.

Soal 1
Buat list dengan isi [5, 10, 15, 20, 25]. Cetak nilai dan id() untuk masing-masing elemen.

Soal 2
Buat dua variabel x dan y, lalu tunjukkan bahwa x dan y mengarah ke objek yang sama atau berbeda (gunakan id()).

Soal 3
Buat fungsi tambah_satu(arr) yang menerima list, menambah 1 ke tiap elemen, dan print id() setiap elemen sebelum dan sesudah diubah.

C. Tugas Mandiri (Dikerjakan di Rumah)

Tugas: Simulasi Penyimpanan Nilai Mahasiswa

1. Buat list nilai = [75, 80, 85, 90, 70]
2. Cetak:
    - Nilai tiap elemen
    - Indeks
    - Alamat memori (gunakan id())
3. Tambahkan fitur:
    - Cari nilai rata-rata
    - Cari nilai maksimum dan minimum
    - Cari indeks dari nilai tertinggi
4. **(Bonus)** Visualisasikan posisi elemen dan alamat memori menggunakan matplotlib

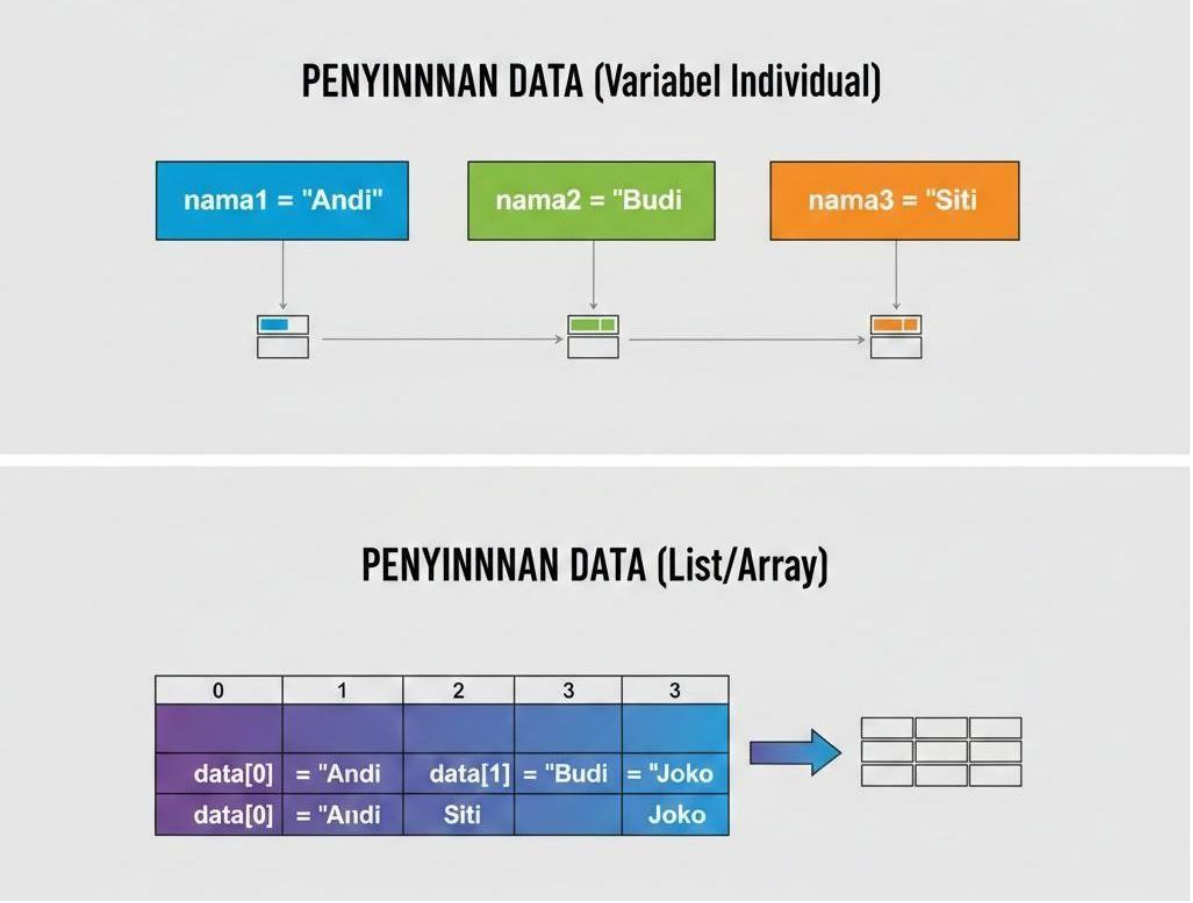
Buku Pertemuan 3

Mata Kuliah: Struktur Data & Algoritma
Pertemuan ke: 3
Topik: Stack & Operasi Dasar (Push, Pop, Peek)
Bahasa: Python
Kode CPMK: CPMK072-3
Taksonomi Bloom: Applying

Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa mampu:

- Menjelaskan konsep stack dan prinsip **LIFO (Last In First Out)**
- Mengimplementasikan operasi dasar stack: push, pop, dan peek
- Menggunakan stack untuk memecahkan masalah sederhana(Gołka et al., 2025)



Gambar 3. Stack

A. Guided Practice (Panduan + Contoh Kode)

1. Apa itu Stack?

Stack adalah struktur data linear yang mengikuti prinsip **LIFO**, yaitu data terakhir yang dimasukkan akan menjadi data pertama yang keluar.

2. Implementasi Stack Sederhana dengan List

python
CopyEdit

```
# Stack menggunakan list
stack = []

# Push
stack.append("A")
stack.append("B")
stack.append("C")

# Pop
print(stack.pop())  # Output: C
print(stack.pop())  # Output: B

# Peek (lihat elemen paling atas)
print(stack[-1])    # Output: A

# Stack saat ini
print(stack)        # Output: ['A']
```

Catatan:

- append() = push
- pop() = hapus dan kembalikan elemen terakhir
- stack[-1] = peek (melihat atas stack tanpa menghapus)

3. Visualisasi Sederhana

python
CopyEdit

```
def tampilkan_stack(s):
    print("Isi stack (atas → bawah):")
    for elemen in reversed(s):
        print("|", elemen, "|")
    print("-----")

stack = []
stack.append("Login")
stack.append("Buka Profil")
```

```
stack.append("Edit Nama")
tampilkan_stack(stack)
stack.pop()  # Undo: Edit Nama
tampilkan_stack(stack)
```

B. Unguided Practice (Latihan Mandiri)

Mahasiswa mengerjakan di kelas secara mandiri.

Soal 1

Buat stack kosong. Tambahkan item "X", "Y", "Z", lalu:

- Cetak isi stack
- Lakukan 1 kali pop()
- Cetak isi stack lagi

Soal 2

Buat program yang menanyakan 5 input nama mahasiswa, lalu simpan dalam stack, lalu tampilkan dari atas ke bawah.

Soal 3

Buat fungsi is_empty(stack) untuk mengecek apakah stack kosong.

C. Tugas Mandiri (Dikerjakan di Rumah)

Tugas: Simulasi Tombol Undo

1. Buat stack kosong
2. Simulasikan 5 aksi pengguna:
    - "Tulis Judul"
    - "Tulis Isi"
    - "Bold Teks"
    - "Tambahkan Gambar"
    - "Preview"
3. Tampilkan isi stack.
4. Lakukan pop() dua kali untuk simulasi tombol **Undo**.
5. Tampilkan kembali isi stack.

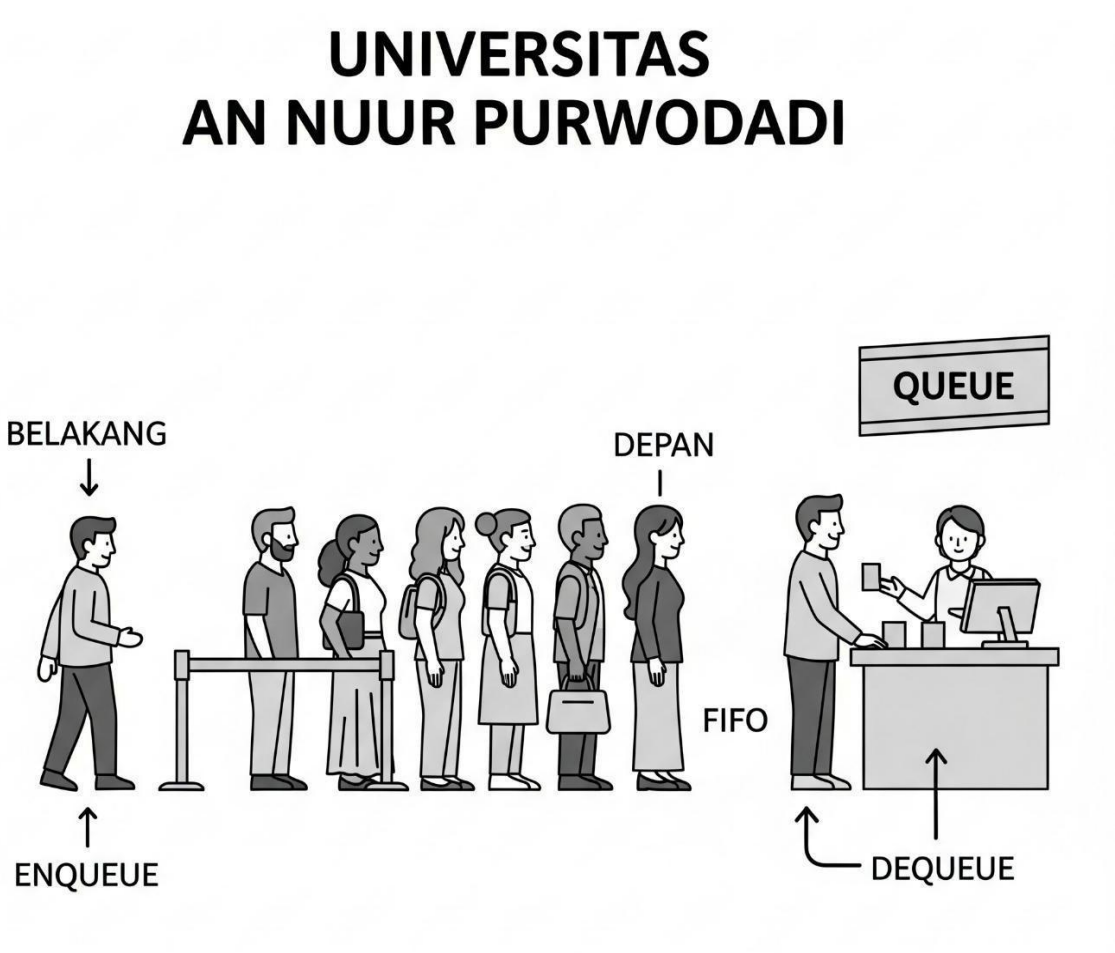
Buku Pertemuan 4

Topik: Queue FIFO & Implementasi

CPMK: CPMK072-3
Bloom: Applying

Tujuan Pembelajaran

- Mahasiswa memahami konsep queue (antrian)
- Mahasiswa mengimplementasikan operasi enqueue, dequeue
- Mahasiswa dapat mensimulasikan antrian sederhana dalam program(Pan et al., 2024)



Gambar 4. Queue FIFO

A. Guided Practice

python
CopyEdit

```
from collections import deque

# Membuat queue
queue = deque()

# Enqueue
queue.append("Andi")
queue.append("Budi")
queue.append("Citra")

# Dequeue
print(queue.popleft())  # Output: Andi

# Tampilkan isi queue
print(queue)  # Output: deque(['Budi', 'Citra'])
```

Visualisasi Sederhana:

python
CopyEdit

```
def tampilkan_queue(q):
    print("Queue sekarang (depan → belakang):", list(q))

tampilkan_queue(queue)
```

B. Unguided Practice

1. Buat queue kosong, tambahkan 3 nama, lalu dequeue 1x dan tampilkan.
2. Buat fungsi is_empty(queue) dan peek(queue)
3. Simulasikan antrian mahasiswa masuk ke ruang dosen

c. Tugas Mandiri

Simulasikan antrian printer:

- Tambahkan 5 job (cetak tugas, laporan, CV, dll)
- Tampilkan isi queue
- Dequeue 2 kali (print selesai)
- Tampilkan sisa job

Refleksi

- Saya paham FIFO
- Saya bisa menggunakan deque
- Saya bisa membuat simulasi antrian

Buku Pertemuan 5

Topik: Singly Linked List
CPMK: CPMK072-1
Bloom: Understanding

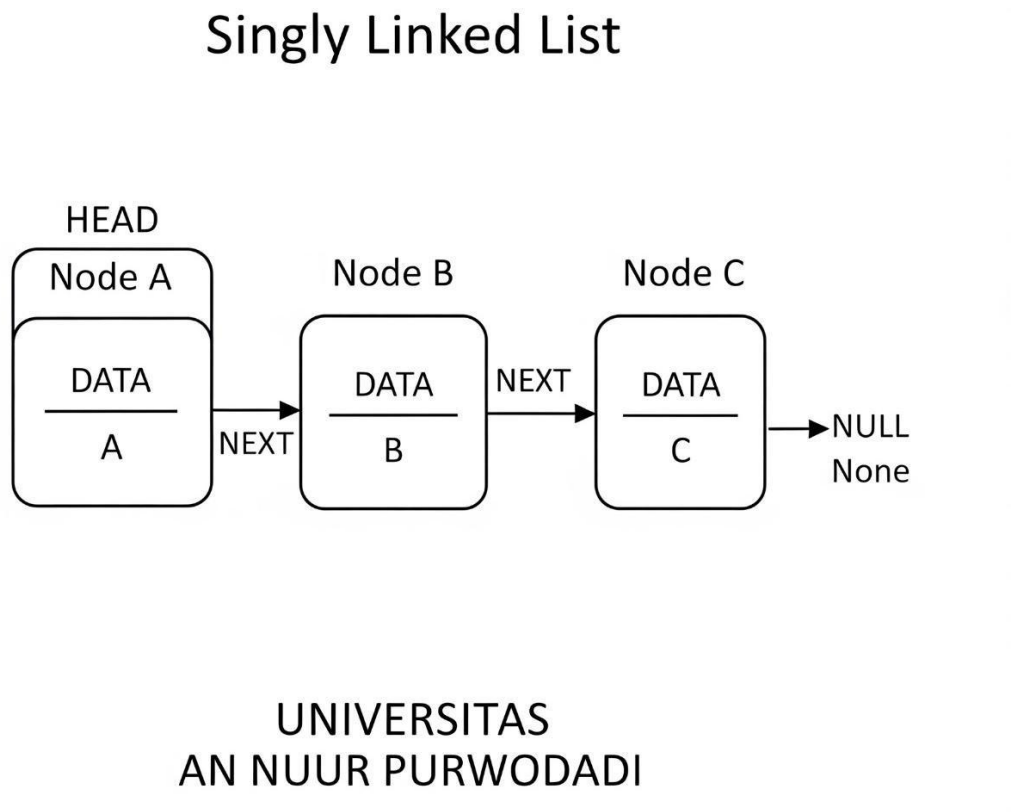
Driving Question:

Bagaimana struktur *Linked List* dapat membantu dalam membuat sistem daftar tunggu rumah sakit atau antrean dinamis pada layanan pelanggan digital?

Tujuan Pembelajaran

Mahasiswa dapat:

- Memahami struktur dan konsep Node
- Membuat *linked list* manual di Python
- Mengenali perbedaan antara list biasa dan *linked list*
- Menghubungkan *linked list* ke skenario dunia nyata(Platz et al., 2020)



Gambar 5. Linked List

A. Guided Practice (Panduan + Contoh Kode)

Definisi Node dan Linked List

```
python
CopyEdit
class Node:
    def _init_(self, data):
        self.data = data
        self.next = None

# Buat node manual
a = Node("A")
b = Node("B")
c = Node("C")

# Hubungkan node
a.next = b
b.next = c

# Traversal dari head
head = a
while head:
    print(head.data)
    head = head.next
```

Catatan: Di dunia nyata, *linked list* digunakan dalam pengelolaan memori dinamis dan sistem antrean yang berubah-ubah ukurannya.

B. Unguided Practice (Latihan Mandiri)

1. Buat 3 node mahasiswa (Nama), sambungkan dengan .next
2. Cetak isi *linked list* mulai dari head
3. Modifikasi isi data node ke-2 menjadi nama lain dan tampilkan ulang seluruh isi list

C. Tugas Mandiri (Dikerjakan di Rumah)

Proyek Mini: Sistem Daftar Tunggu Mahasiswa

Simulasikan sistem antrean pengajuan bimbingan skripsi:

- Buat struktur Node yang menyimpan:
    - nama_mahasiswa
    - nim
    - keperluan (misalnya: "bimbingan", "izin seminar", "revisi")
- Buat minimal 5 node dan sambungkan
- Buat fungsi tampilkan_antrian(head) untuk mencetak isi list:

yaml
CopyEdit
1. NIM: 2104111001, Nama: Andi, Keperluan: Bimbingan
2. NIM: 2104111002, Nama: Budi, Keperluan: Revisi

- **Bonus:** Buat fungsi hapus_antrian_pertama() untuk menghapus node terdepan (seolah antrean dipanggil)

Refleksi

- Saya paham konsep Node dan penghubung next
- Saya tahu kapan *linked list* lebih berguna daripada list biasa
- Saya bisa membuat sistem antrean sederhana berbasis *linked list*

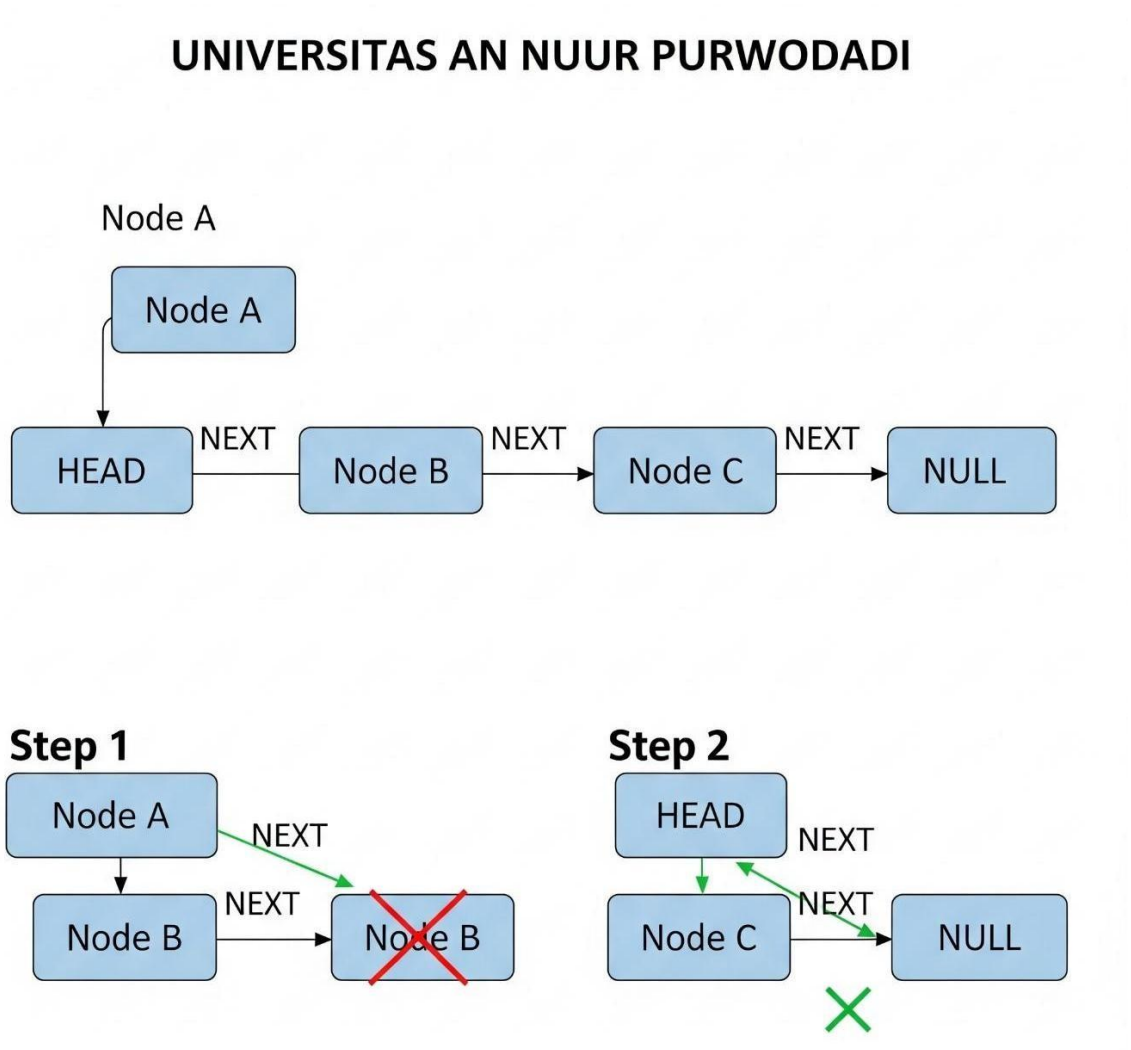
Buku Pertemuan 6

Topik: Operasi Insert/Delete pada Linked List

CPMK: CPMK072-3
Bloom: Applying

Tujuan Pembelajaran

- Mahasiswa dapat menambahkan dan menghapus node
- Mahasiswa memahami traversal, insert di tengah, dan delete node(Wang et al., 2025)



Gambar 6. Operasi linked list

A. Guided Practice

python
CopyEdit
# Sisip di awal
new_node = Node("Awal")
new_node.next = a # a adalah node pertama sebelumnya
head = new_node

# Hapus node kedua
head.next = head.next.next

B. Unguided Practice

1. Buat linked list 3 elemen
2. Tambahkan node di awal
3. Hapus node tengah

C. Tugas Mandiri

Buat fungsi:

- insert_at_end(head, data)
- delete_by_value(head, value)

Uji dengan membuat linked list mahasiswa dan lakukan operasi insert/delete.

Refleksi

- Saya bisa sisip node
- Saya bisa hapus node tertentu
- Saya bisa traversal sebelum/selesai operasi

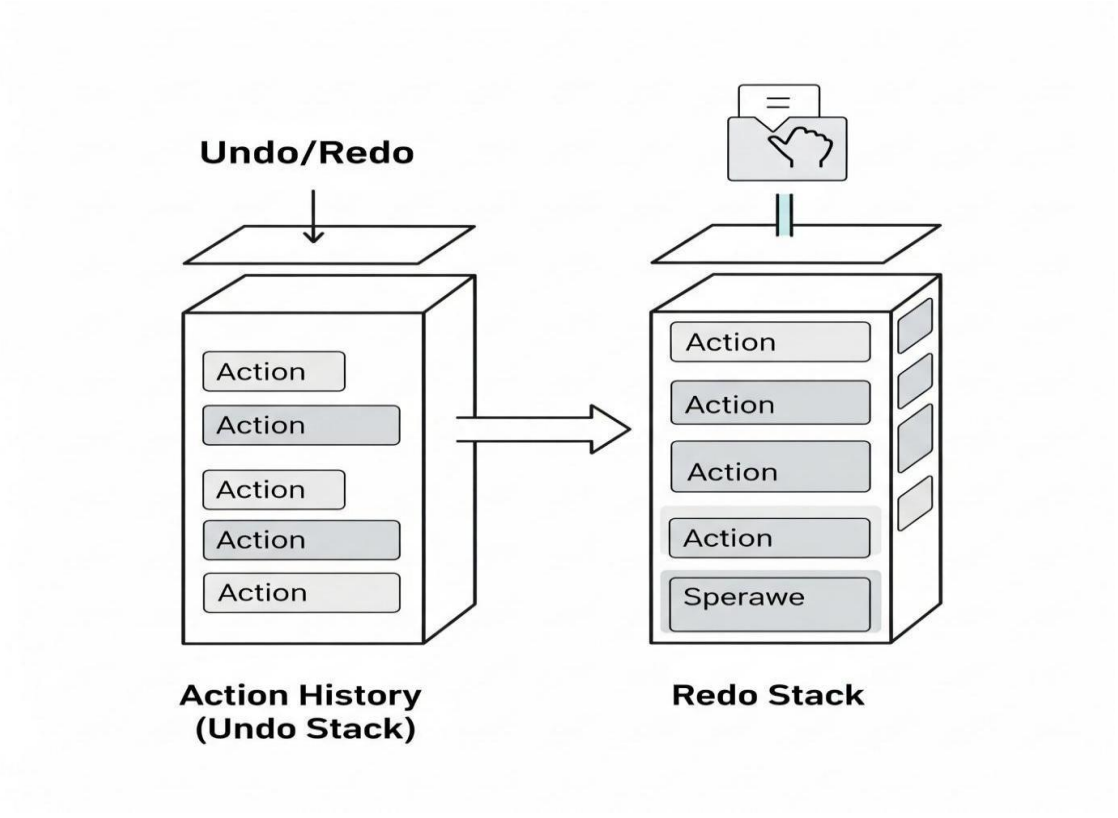
Buku Pertemuan 7

Topik: Stack/Queue Real Use Case (Undo, Antrian Nyata)

CPMK: CPMK072-2
Bloom: Analyzing

Tujuan Pembelajaran

- Mahasiswa mampu menganalisis penggunaan stack dan queue dalam dunia nyata
- Mahasiswa mampu menyimulasikan kasus seperti **Undo/Redo**, **Antrian Dokter**, **Customer Service**(Cacciari, 2005)



Gambar 7. Stack & queue

A. Guided Practice

Simulasi Undo/Redo (Stack):

python
CopyEdit

```
undo_stack = []
redo_stack = []

# Aksi
undo_stack.append("Tulis Judul")
undo_stack.append("Bold")

# Undo
redo_stack.append(undo_stack.pop())  # Undo "Bold"

# Redo
undo_stack.append(redo_stack.pop())  # Redo "Bold"
```

Simulasi Antrian Loket (Queue):

python
CopyEdit

```
antrian = deque(["Ali", "Budi", "Citra"])
print("Melayani:", antrian.popleft())
```

B. Unguided Practice

1. Simulasikan Undo 3 aksi lalu Redo 1 aksi
2. Simulasikan antrian customer service
3. Tampilkan sisa stack/queue

C. Tugas Mandiri

Proyek Mini: Simulasi Skenario Dunia Nyata
Buat aplikasi simulasi berbasis teks untuk:

- Aplikasi Teks dengan Undo/Redo
- Antrian Pasien Klinik

Berikan menu interaktif (1. Tambah, 2. Undo, 3. Redo, dll)

Buku Pertemuan 8

Topik: Big-O & Space-Time Trade-off
CPMK: CPMK072-4
Bloom: Analyzing

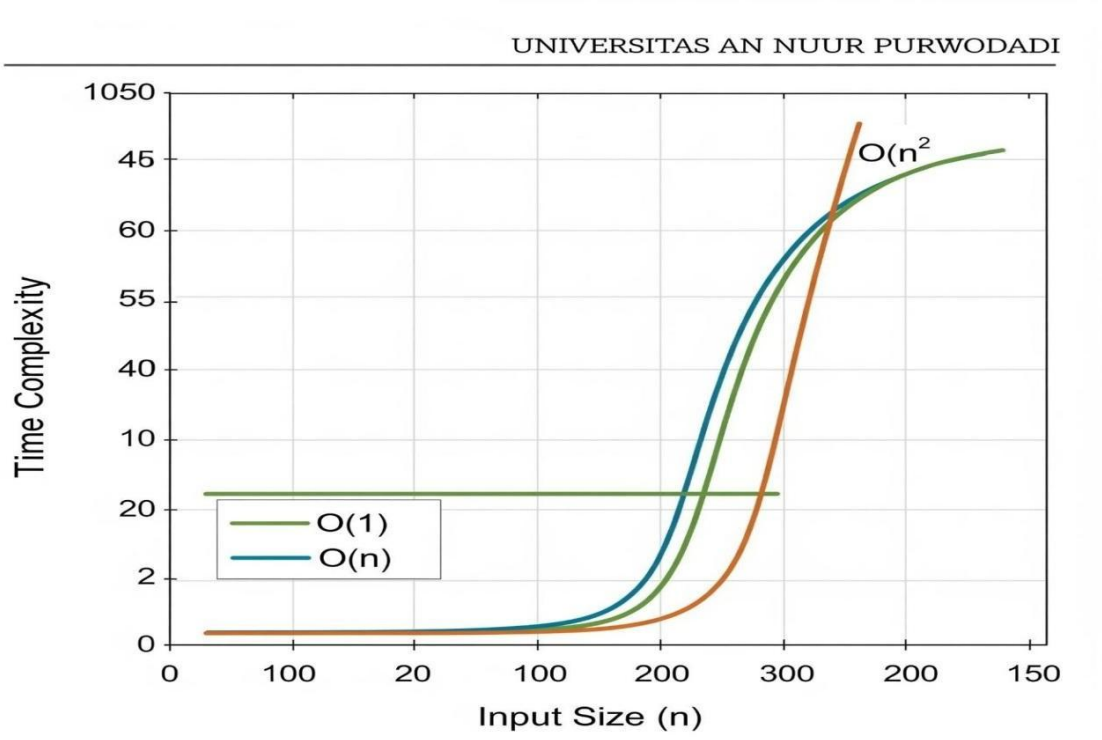
Driving Question:

"Bagaimana kita bisa mengoptimalkan algoritma agar aplikasi seperti YouTube atau e-commerce tetap cepat meskipun jutaan data diproses tiap detik?"

Tujuan Pembelajaran

Mahasiswa dapat:

- Memahami notasi Big-O dan konsep efisiensi algoritma
- Menganalisis efisiensi waktu dan ruang dari beberapa algoritma sederhana
- Mengevaluasi kompleksitas dari berbagai pendekatan kode Python(Li, 2025)



Gambar 8. Big O

A. Guided Practice

1. Konsep Big-O

- O(1) – waktu konstan
- O(n) – waktu linier

- O(n^2) – kuadratik

2. Contoh Implementasi

```
# O(1)
def akses_pertama(data):
    return data[0]
# O(n)
def cetak_semua(data):
    for d in data:
        print(d)

# O(n^2)
def pasangan(data):
    for i in data:
        for j in data:
            print(i, j)
```

3. Visualisasi Waktu Eksekusi

```
import time
import matplotlib.pyplot as plt

n_list = [10, 100, 1000, 2000]
waktu = []

for n in n_list:
    data = list(range(n))
    start = time.time()
    for i in data:
        for j in data:
            _ = i + j
    end = time.time()
    waktu.append(end - start)

plt.plot(n_list, waktu)
plt.title("Waktu vs Ukuran Input (O(n^2))")
plt.xlabel("n")
plt.ylabel("Detik")
plt.show()
```

B. Unguided Practice

1. Bandingkan dua fungsi:
    - Fungsi yang mencetak 1 elemen (O(1))
    - Fungsi yang mencetak seluruh elemen (O(n))
2. Hitung estimasi waktu eksekusi untuk O(n), O(n^2) menggunakan time.time()
3. Buat fungsi yang menampilkan elemen ke-i dari list, dan diskusikan kompleksitas waktunya

C. Tugas Mandiri

Proyek Mini: Simulasi Algoritma Pencarian Mahasiswa
Simulasikan efisiensi beberapa strategi pencarian:

1. Buat dataset mahasiswa dengan format:

mahasiswa = ["Andi", "Budi", "Citra", ..., "Zaki"] # isi 1000 data

2. Buat dan uji 3 fungsi:

- cari_nama_O1() → akses langsung index tertentu
- cari_nama_On() → iterasi dari awal sampai ketemu
- cari_nama_On2() → bandingkan semua pasangan nama (tidak efisien)

3. Uji dengan time.time() dan tampilkan perbandingan waktu secara visual dengan matplotlib

+**Bonus:** Simulasikan hasil dalam bentuk tabel atau grafik di Google Colab

Refleksi

- Saya memahami Big-O dan penerapannya
- Saya bisa membedakan O(1), O(n), O(n^2) melalui eksperimen
- Saya sadar pentingnya efisiensi dalam sistem real-time dan skala besar

Buku Pertemuan 9

Topik: Tree, Binary Search Tree dasar
CPMK: CPMK072-1
Bloom: Understanding

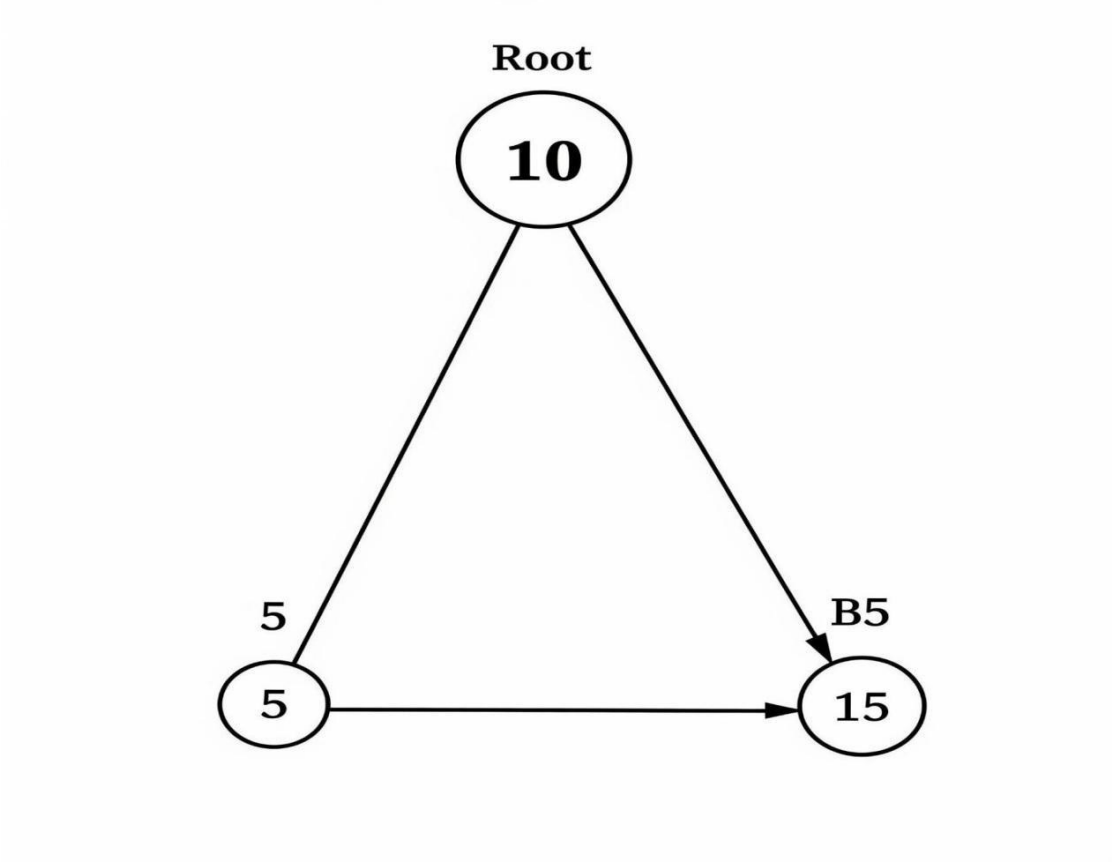
Driving Question:

"Bagaimana cara merepresentasikan struktur organisasi atau sistem direktori file secara hierarkis dalam program?"

Tujuan Pembelajaran

Mahasiswa dapat:

- Memahami konsep struktur tree
- Mengimplementasikan tree sederhana dan BST
- Menghubungkan struktur tree dengan data real-life(Brodal, 2025)



Gambar 9. BST

A. Guided Practice

Struktur Tree Dasar

```
class Node:
    def _init_(self, data):
        self.data = data
        self.left = None
        self.right = None

root = Node(10)
root.left = Node(5)
root.right = Node(15)
```

Traversal Sederhana (In-Order)

```
def inorder(node):
    if node:
        inorder(node.left)
        print(node.data)
        inorder(node.right)

inorder(root)
```

B. Unguided Practice

1. Buat node dengan data [20, 10, 30, 5, 15, 25, 35] dalam struktur BST
2. Buat fungsi inorder, preorder, dan postorder
3. Cetak hasil traversal dan visualisasikan hasil secara manual dalam bentuk skema pohon

C. Tugas Mandiri

Proyek Mini: Simulasi Folder

Simulasikan sistem folder seperti:

```
Root
├── Documents
│   └── Tugas
└── Pictures
```

- Implementasi tree sederhana dengan node sebagai folder
- Buat fungsi cetak struktur folder secara rekursif

Refleksi

- Saya memahami konsep tree & BST
- Saya bisa menghubungkan struktur pohon dengan sistem organisasi data

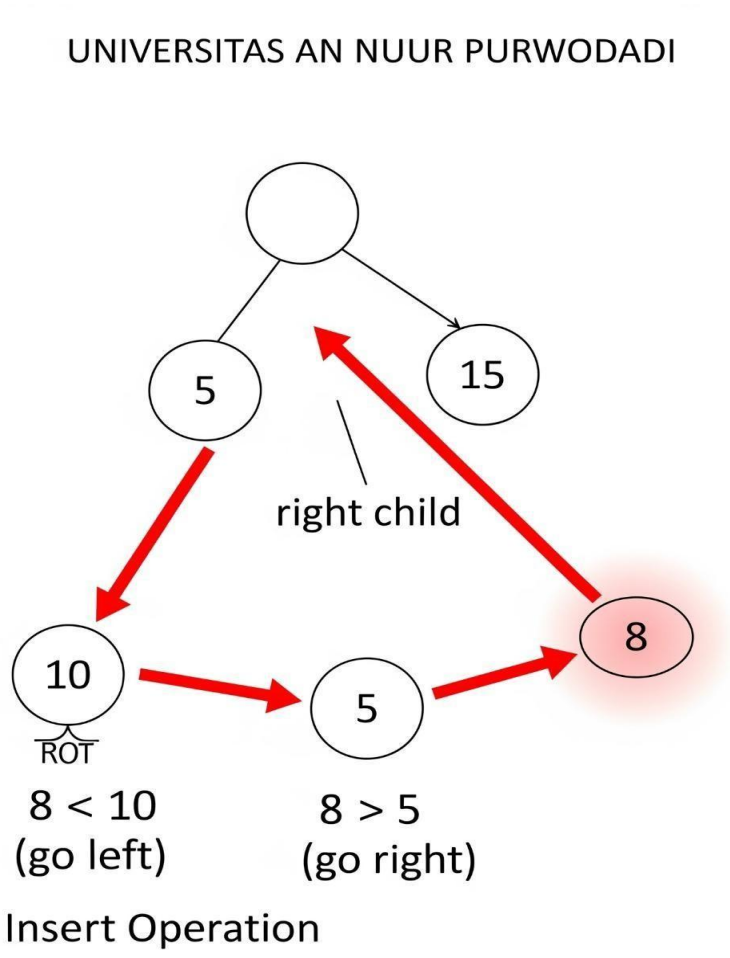
Buku Pertemuan 10

Topik: BST Insert/Delete/Search

CPMK: CPMK072-3
Bloom: Applying

Tujuan Pembelajaran

- Mahasiswa dapat melakukan operasi dasar pada BST: tambah, cari, dan hapus
- Mahasiswa dapat menguji properti BST setelah operasi(Kupriyashin & Borzunov, 2016)



Gambar 10. Operasi BST

A. Guided Practice

1. Insert dan Search

python
CopyEdit

```
class BST:
    def _init_(self):
        self.root = None

    def insert(self, data):
        def _insert(node, data):
            if node is None:
                return Node(data)
            elif data < node.data:
                node.kiri = _insert(node.kiri, data)
            else:
                node.kanan = _insert(node.kanan, data)
            return node

        if self.root is None:
            self.root = Node(data)
        else:
            _insert(self.root, data)

    def search(self, data):
        def _search(node, data):
            if not node:
                return False
            if data == node.data:
                return True
            elif data < node.data:
                return _search(node.kiri, data)
            else:
                return _search(node.kanan, data)

        return _search(self.root, data)
```

B. Unguided Practice

1. Tambahkan fungsi min() dan max() untuk BST
2. Buat menu interaktif:
    - Tambah node
    - Cari node
    - Tampilkan inorder

C. Tugas Mandiri

Implementasikan fungsi delete() untuk BST. Uji dengan skenario:

- Hapus node tanpa anak
- Hapus node dengan satu anak
- Hapus node dengan dua anak

Cetak inorder tree sebelum dan sesudah penghapusan.

Refleksi

- Saya bisa menambah node ke BST
- Saya bisa cari node dengan search
- Saya bisa hapus node sesuai kondisi

Buku Pertemuan 11

Topik: Hashing dan Penanganan Tabrakan
CPMK: CPMK102-1
Bloom: Understanding

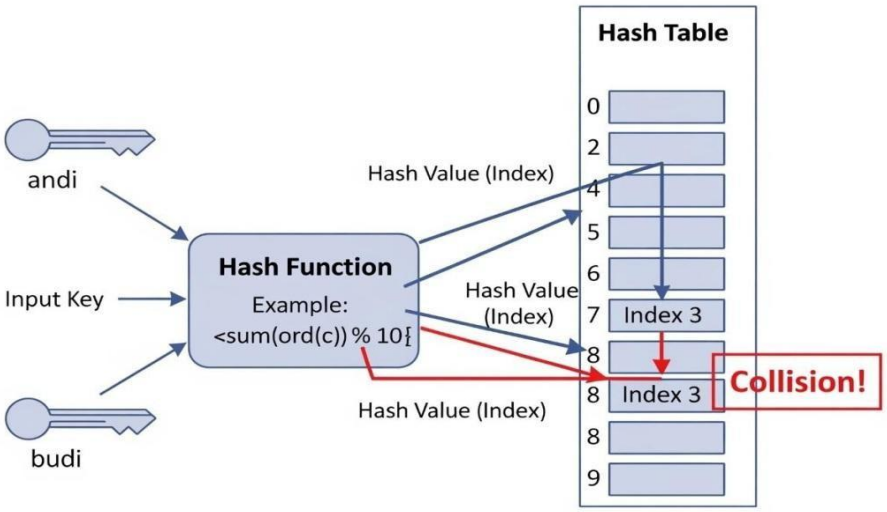
Driving Question:

"Bagaimana sebuah situs bisa menyimpan jutaan akun pengguna dan tetap cepat saat login?"

Tujuan Pembelajaran

Mahasiswa dapat:

- Memahami cara kerja hashing
- Mengidentifikasi potensi collision
- Menerapkan metode penanganan collision(O'Brien et al., 2024)



Gambar 11. Hashing

A. Guided Practice

a. Hash Function Sederhana

```
def hash_function(key):
    return sum(ord(c) for c in key) % 10

print(hash_function("andi"))
print(hash_function("budi"))
```

b. Penanganan Collision (Chaining)

```
hash_table = [[] for _ in range(10)]
def insert(key):
    idx = hash_function(key)
    hash_table[idx].append(key)

insert("andi")
insert("budi")
insert("citra")
print(hash_table)
```

B. Unguided Practice

1. Ubah fungsi hash agar case-sensitive
2. Tambahkan fungsi search(key) untuk mencari apakah key sudah ada
3. Tambahkan fungsi delete(key

C. Tugas Mandiri

Proyek Mini: Sistem Login Email

- Buat simulasi penyimpanan akun email berbasis hashing
- Input: nama, email, password → hashing email sebagai key
- Tampilkan hasil hash table setelah semua data dimasukkan
- Tambahkan fitur pencarian akun

Refleksi

- Saya bisa membuat hash function sederhana
- Saya tahu cara menangani collision
- Saya paham kenapa hashing sangat penting untuk efisiensi

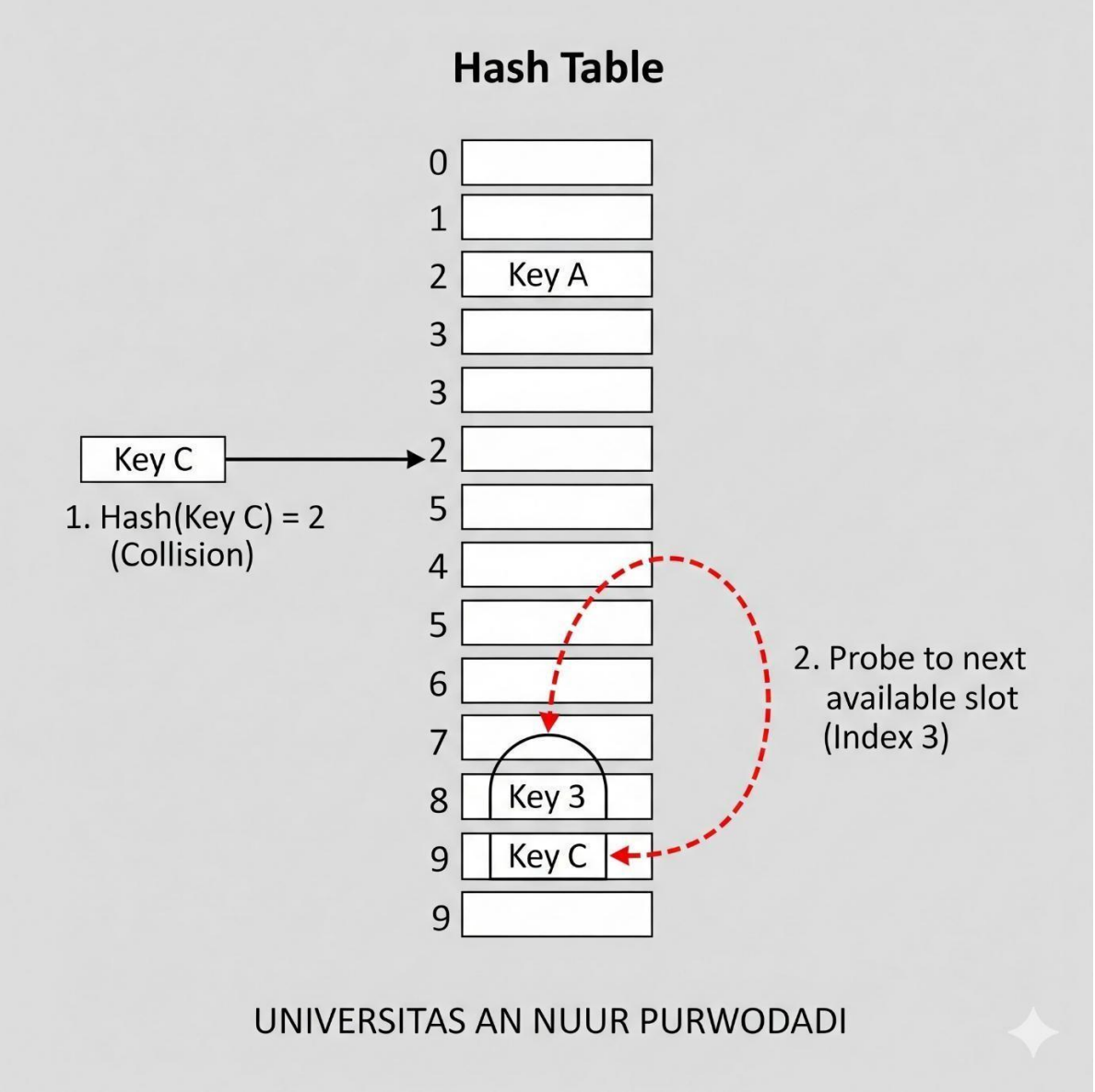
Buku Pertemuan 12

Topik: Array-Based Hashing

CPMK: CPMK102-3
Bloom: Applying

Tujuan Pembelajaran

- Mahasiswa dapat mengimplementasikan hash table dengan array
- Mahasiswa dapat memahami konsep probing dan efisiensinya(Brun et al., 2012)



Gambar 12. Array based hashing

A. Guided Practice: Linear Probing

```
python
CopyEdit
table = [None] * 10

def  linear_probe_insert(key,  value):
   idx = simple_hash(key, len(table))
   while table[idx] is not None:
      idx = (idx + 1) % len(table)
   table[idx] = (key, value)
```

B. Unguided Practice

1. Ubah insert menjadi quadratic probing
2. Tambahkan fungsi search(key) untuk hash table tersebut
3. Uji dengan data mahasiswa

C. Tugas Mandiri

Buat program:

- Array size: 7
- Insert data: ("A", 90), ("B", 80), ..., ("G", 85)
- Tangani collision pakai linear probing
- Tambahkan fitur pencarian dan hapus data

Refleksi

- Saya bisa menggunakan probing
- Saya paham bedanya chaining vs probing
- Saya bisa membandingkan efisiensi keduanya

Pertemuan 13

Topik: Graph dan Representasi
CPMK: CPMK102-1
Bloom: Understanding

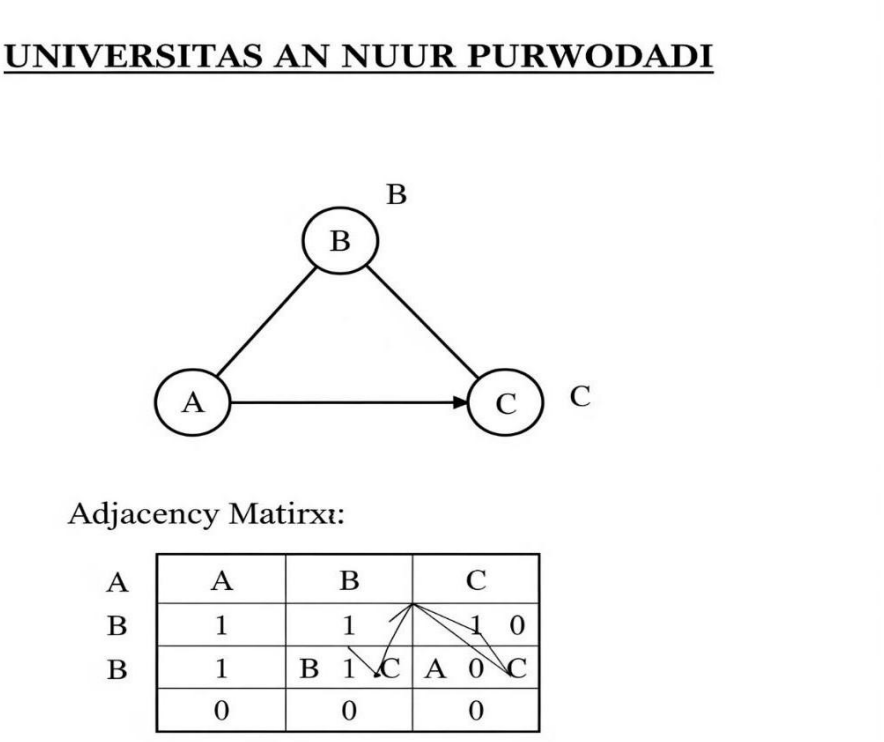
Driving Question:

"Bagaimana Google Maps mengetahui jalur tercepat dari satu titik ke titik lain?"

Tujuan Pembelajaran

Mahasiswa dapat:

- Memahami konsep dasar graph
- Mewakili graph dengan adjacency matrix dan list
- Menyusun representasi data koneksi antar objek(Kostyukova & Tchemisova, 2025)



Gambar 13. Graph

A. Guided Practice

Representasi dengan Matrix

```
# Graph tidak berarah
graph = [
    [0, 1, 1],
    [1, 0, 0],
    [1, 0, 0]
]
```

Representasi dengan Adjacency List

```
graph = {
    'A': ['B', 'C'],
    'B': ['A'],
    'C': ['A']
}
```

B. Unguided Practice

1. Representasikan koneksi antar 5 kota dalam adjacency matrix dan list
2. Tambahkan fungsi tampilkan_koneksi(kota)
3. Buat grafik manual koneksi antar kota

C. Tugas Mandiri

Proyek Mini: Social Graph

- Simulasikan koneksi pertemanan seperti media sosial
- Input: daftar user dan relasi pertemanan
- Buat visualisasi sederhana (pakai dict atau matplotlib networkx jika mau)
- Fitur tambahan: cari teman dari teman (2nd-degree connection)

Refleksi

- Saya bisa membedakan adjacency list dan matrix
- Saya tahu struktur graph digunakan untuk menyelesaikan masalah nyata

Buku Pertemuan 14

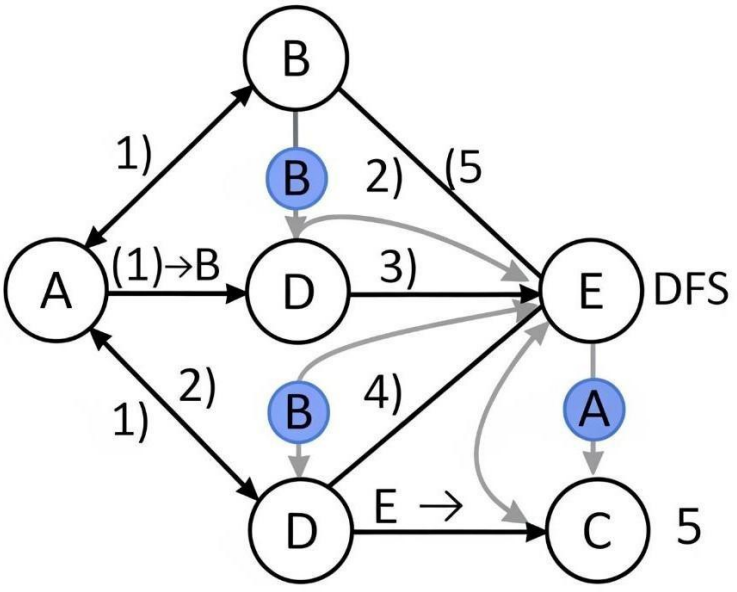
Topik: Graph Traversal (DFS & BFS)

CPMK: CPMK102-3
Bloom: Applying

Tujuan Pembelajaran

- Mahasiswa memahami cara menelusuri graf
- Mahasiswa dapat mengimplementasikan **DFS** dan **BFS**(Scheffler, 2022)

UNIVERSITAS AN NUUR PURWODADI



Gambar 14. DFS & BFS

A. Guided Practice

```
python
CopyEdit
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    print(start)
    for tetangga in graph[start]:
        if tetangga not in visited:
            dfs(graph, tetangga, visited)

# BFS
from collections import deque
def bfs(graph, start):
    visited = set()
    queue = deque([start])
    while queue:
        simpul = queue.popleft()
        if simpul not in visited:
            print(simpul)
            visited.add(simpul)
            queue.extend(graph[simpul])
```

B. Unguided Practice

1. Buat graf dengan 6 simpul
2. Implementasi DFS dan BFS
3. Cetak urutan kunjungan simpul

C. Tugas Mandiri

- Buat program menu:
    - Tambah simpul
    - Tambah edge
    - DFS dari simpul tertentu
    - BFS dari simpul tertentu

Refleksi

- Saya bisa jelaskan cara kerja DFS/BFS
- Saya bisa traversal graf secara manual
- Saya tahu perbedaan depth-first vs breadth-first

Buku Pertemuan 15

Topik: Proyek Akhir

CPMK: CPMK102-2
Bloom: Creating

Tujuan Pembelajaran

- Mahasiswa mampu membangun mini-project dari topik yang telah dipelajari
- Mahasiswa mampu menyajikan hasil program dalam bentuk presentasi

A. Kegiatan

- Mahasiswa dibagi kelompok atau individu
- Tema bebas: simulasi stack, queue, linked list, BST, hash table, atau graf

B. Contoh Proyek:

- **Simulasi Antrian RS**
- **Aplikasi Undo/Redo**
- **Sistem Organisasi Mahasiswa (graf)**
- **BST untuk pencarian buku**

C. Penilaian:

- Logika & efisiensi kode
- Struktur data digunakan tepat
- Antarmuka / CLI interaktif
- Presentasi & dokumentasi

Refleksi

- Saya bisa menggabungkan konsep yang dipelajari
- Saya bisa membuat proyek nyata
- Saya bisa mempresentasikan solusi dengan struktur data

Buku Pertemuan 16

Topik: Refleksi Akhir & UAS

CPMK: CPMK102-4
Bloom: Evaluating

A. Kegiatan

- Review hasil proyek dan refleksi
- Diskusi pengalaman belajar
- Penilaian UAS berbasis studi kasus dan/atau coding

B. UAS (Contoh Format):

- Studi kasus (misalnya data mahasiswa)
- Buat struktur data yang sesuai
- Terapkan operasi utama: tambah, cari, hapus, tampilkan

Refleksi Akhir

Kompetensi	Nilai Diri
Saya memahami dan bisa menerapkan Stack	Ya / Tidak
Saya memahami dan bisa menerapkan Queue	Ya / Tidak
Saya bisa buat BST, Hash, dan Graf	Ya / Tidak
Saya percaya diri menghadapi soal berbasis struktur data	Ya / Tidak

Daftar Pustaka

Brodal, G. S. (2025). Bottom-up rebalancing binary search trees by flipping a coin. *Theoretical Computer Science*, *1055*. https://doi.org/10.1016/j.tcs.2025.115543

Brun, E., Guittet, A., & Gibou, F. (2012). A local level-set method using a hash table data structure. *Journal of Computational Physics*, *231*(6), 2528–2536. https://doi.org/10.1016/j.jcp.2011.12.001

Cacciari, M. (2005). NOMES DE LUGAR: CONFIM. *Source: Revista de Letras*, *45*(1), 13–22. https://doi.org/10.2307/26459823

Gołka, Ł., Suszyński, R., & Poczekajło, P. (2025). Forth Language Extensions for DSP: Configurable FIFO and LIFO Structures. *Procedia Computer Science*, *270*, 2542–2551. https://doi.org/10.1016/j.procs.2025.09.376

Kostyukova, O. I., & Tchemisova, T. V. (2025). Representation of zeros of a copositive matrix via maximal cliques of a graph. *Linear Algebra and Its Applications*, *717*, 40–55. https://doi.org/10.1016/j.laa.2025.03.020

Kupriyashin, M. A., & Borzunov, G. I. (2016). Computational Load Balancing Algorithm for Parallel Knapsack Packing Tree Traversal. *Procedia Computer Science*, *88*, 330–335. https://doi.org/10.1016/j.procs.2016.07.444

Li, Y. (2025). Performance Analysis of Efficient Sorting Algorithms in Big Data Processing. *Procedia Computer Science*, *262*, 44–50. https://doi.org/10.1016/j.procs.2025.05.026

O'Brien, J. G. K., Conway, L. P., Ramaraj, P. K., Jadhav, A. M., Jin, J., Dutra, J. K., Evers, P., Masoud, S. S., Schupp, M., Saridakis, I., Chen, Y., Maulide, N., Pezacki, J. P., am Ende, C. W., Parker, C. G., & Fox, J. M. (2024). Mechanistic differences between linear vs. spirocyclic dialkyldiazirine probes for photoaffinity labeling. *Chemical Science*, *15*(37), 15463–15473. https://doi.org/10.1039/d4sc04238g

Pan, X., Wang, Y., Lu, Y., & Sun, N. (2024). Improved artificial bee colony algorithm based on two-dimensional queue structure for complex optimization problems. *Alexandria Engineering Journal*, *86*, 669–679. https://doi.org/10.1016/j.aej.2023.12.011

Platz, K., Mittal, N., & Venkatesan, S. (2020). Practical concurrent unrolled linked lists using lazy synchronization. *Journal of Parallel and Distributed Computing*, *139*, 110–134. https://doi.org/10.1016/j.jpdc.2019.11.005

Scheffler, R. (2022). On the recognition of search trees generated by BFS and DFS. *Theoretical Computer Science*, *936*, 116–128. https://doi.org/10.1016/j.tcs.2022.09.018

Volk, J. M., & Turner, M. A. (2019). PRMS-Python: A Python framework for programmatic PRMS modeling and access to its data structures. *Environmental Modelling and Software*, *114*, 152–165. https://doi.org/10.1016/j.envsoft.2019.01.006

Wang, Z., Timlin, D., Thapa, R., Fleisher, D., Beegum, S., Han, E., Schomberg, H., Mirsky, S., Sun, W., Reddy, V., Horton, R., & Tully, K. (2025). Process-based vegetative growth model for cereal rye winter cover crop using object-oriented programming and linked-list data structure. *Computers and Electronics in Agriculture*, *231*. https://doi.org/10.1016/j.compag.2025.109964

Zhao, L., Liu, X., & Jin, J. (2025). A novel adaptive parameter zeroing neural network for the synchronization of complex chaotic systems and its field programmable gate array implementation. *Measurement: Journal of the International Measurement Confederation, 242*. https://doi.org/10.1016/j.measurement.2024.115989

Glosarium

Istilah	Keterangan	Buku Terkait
Adjacency List	Cara merepresentasikan Graph di mana setiap simpul (vertex) memiliki daftar simpul-simpul lain yang terhubung langsung dengannya[1].	Pertemuan 13
Adjacency Matrix	Cara merepresentasikan Graph menggunakan matriks dua dimensi, di mana nilai 1 (atau berat edge) menunjukkan adanya koneksi antar simpul[2222].	Pertemuan 13
Algoritma	Serangkaian langkah-langkah yang terdefinisi dengan baik untuk memecahkan suatu masalah atau mencapai suatu tujuan.	Umum
Array (List)	Struktur data dasar di Python yang digunakan untuk menyimpan koleksi item. Dalam konteks Buku, sering disamakan dengan list[3333].	Pertemuan 1, 2
Big-O Notasi	Notasi matematika yang menggambarkan batas atas (worst-case) dari kompleksitas waktu atau ruang suatu algoritma saat ukuran input (n) bertambah[4444].	Pertemuan 8
Binary Search Tree (BST)	Struktur data Tree di mana nilai dari setiap *node* di sub-pohon kiri selalu lebih kecil, dan nilai di sub-pohon kanan selalu lebih besar daripada *node* itu sendiri[5].	Pertemuan 9
Breadth-First Search (BFS)	Algoritma penelusuran Graph yang menjelajahi simpul-simpul per level, bergerak melebar sebelum turun ke level berikutnya (menggunakan Queue)[6666].	Pertemuan 14
Chaining	Metode penanganan *Collision* (tabrakan) dalam Hashing, di mana semua elemen yang memiliki nilai hash yang sama disimpan dalam sebuah *linked list* pada indeks *hash table* tersebut[7].	Pertemuan 11
Collision	Kejadian di mana fungsi Hash menghasilkan indeks yang sama untuk dua kunci (key) yang berbeda[8].	Pertemuan 11
Dequeue	Operasi untuk menghapus dan mengembalikan elemen pertama (terdepan) dari sebuah **Queue** (antrian)[9999].	Pertemuan 4
Depth-First Search (DFS)	Algoritma penelusuran Graph yang menjelajahi sejauh mungkin di sepanjang cabang sebelum kembali (menggunakan Stack atau Rekursi)[10101010].	Pertemuan 14
Enqueue	Operasi untuk menambahkan elemen baru ke bagian belakang (tail) dari sebuah **Queue** (antrian)[11111111].	Pertemuan 4

Graph	Struktur data non-linear yang terdiri dari sekumpulan simpul (Vertex) yang dihubungkan oleh sisi (Edge)[12].	Pertemuan 13
Hashing	Proses mengubah data input berukuran acak (kunci/key) menjadi nilai berukuran tetap (nilai hash/indeks)[13].	Pertemuan 11
LIFO (Last In First Out)	Prinsip yang digunakan oleh Stack, di mana data yang terakhir masuk adalah data yang pertama kali keluar14141414.	Pertemuan 3
Linked List	Struktur data linear di mana elemen data tidak disimpan di lokasi memori yang berdekatan. Setiap elemen (*Node*) menunjuk ke elemen berikutnya melalui *pointer* (.next)[1515].	Pertemuan 5
Node	Unit dasar pembangun dalam struktur Linked List dan Tree. Biasanya terdiri dari data dan penunjuk (pointer) ke node berikutnya16161616.	Pertemuan 5
O(1)	Kompleksitas waktu konstan. Waktu eksekusi tidak bergantung pada ukuran input data (n)17171717.	Pertemuan 8
O(n)	Kompleksitas waktu linier. Waktu eksekusi berbanding lurus dengan ukuran input data (n)18181818.	Pertemuan 8
O(n^2)	Kompleksitas waktu kuadratik. Waktu eksekusi berbanding lurus dengan kuadrat ukuran input data (n)19191919.	Pertemuan 8
Peek	Operasi untuk melihat atau mengembalikan elemen teratas dari Stack atau elemen terdepan dari Queue tanpa menghapusnya20202020.	Pertemuan 3
Pointer (Referensi)	Konsep yang mengacu pada alamat memori suatu objek. Dalam Python, dapat dianalogikan dengan fungsi id()21212121.	Pertemuan 2
Pop	Operasi untuk menghapus dan mengembalikan elemen teratas dari sebuah **Stack**22222222.	Pertemuan 3
Probing	Metode penanganan *Collision* dalam Hashing yang mencari slot kosong di Hash Table dengan memodifikasi indeks secara sistematis (misalnya Linear Probing)23232323.	Pertemuan 12
Push	Operasi untuk menambahkan elemen baru ke bagian atas (top) dari sebuah **Stack**24242424.	Pertemuan 3
Queue	Struktur data linear yang mengikuti prinsip FIFO, berfungsi seperti antrian[25].	Pertemuan 4
Stack	Struktur data linear yang mengikuti prinsip LIFO, berfungsi seperti tumpukan26262626.	Pertemuan 3

Struktur Data	Cara menyimpan dan mengatur data agar dapat digunakan secara efisien[27].	Pertemuan 1
Traversal	Proses mengunjungi atau memproses setiap elemen (node) dalam suatu struktur data (seperti Linked List, Tree, atau Graph) tepat satu kali[28282828].	Pertemuan 5, 9, 14

Table: Glosarium