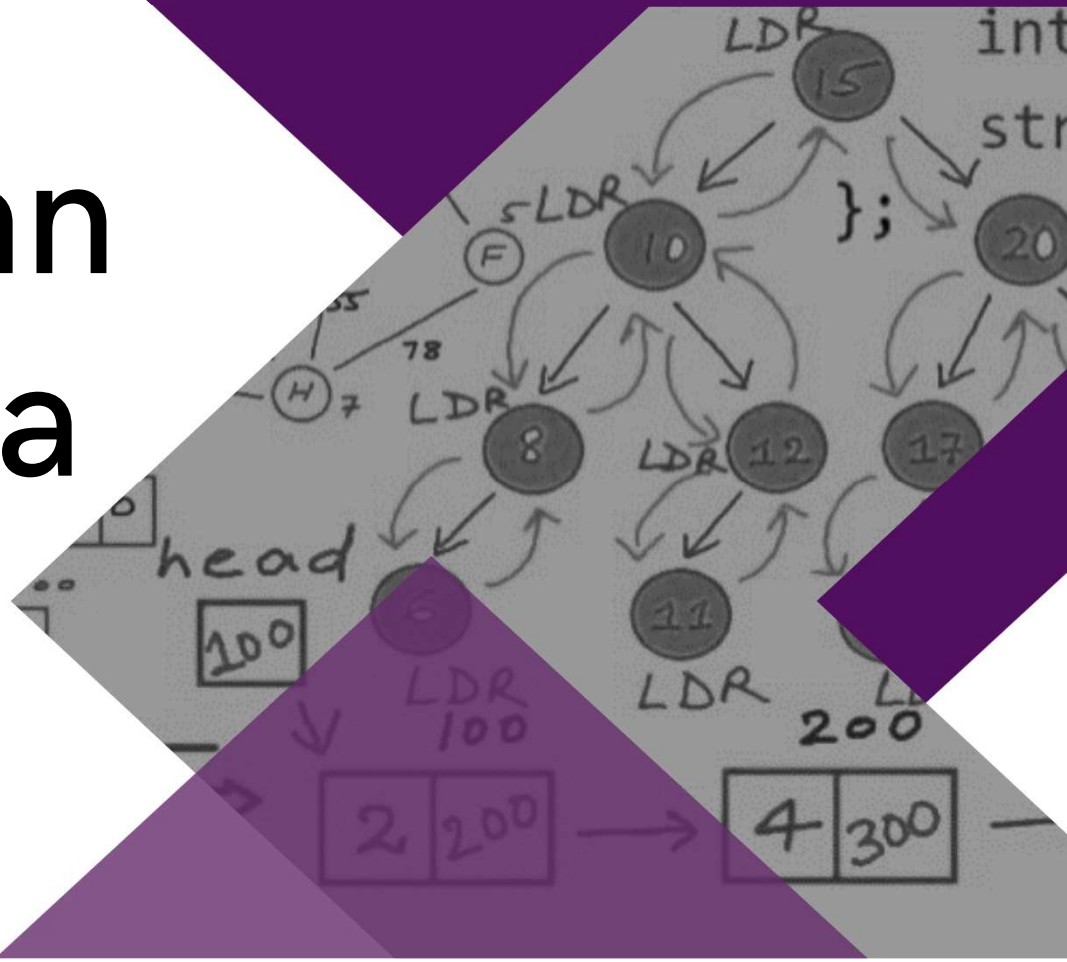
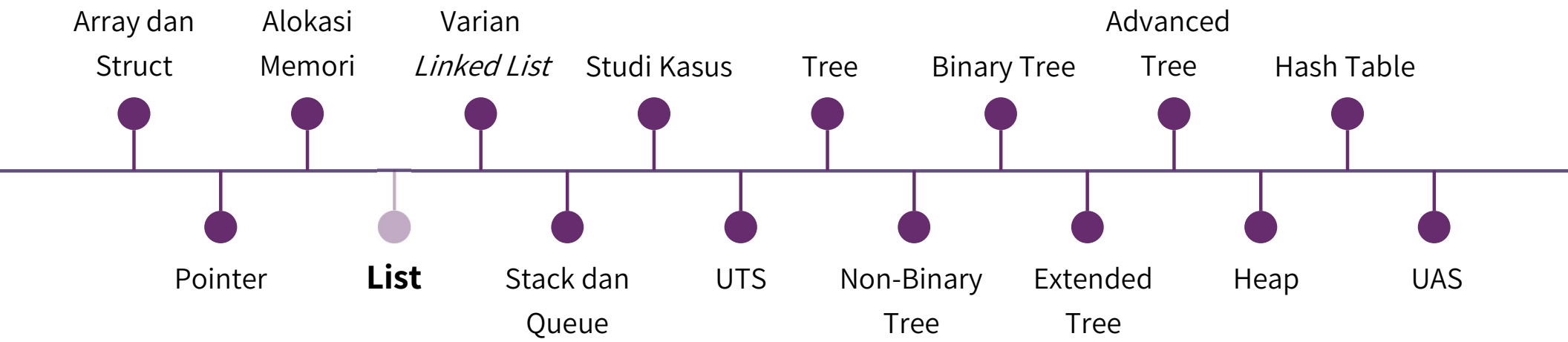
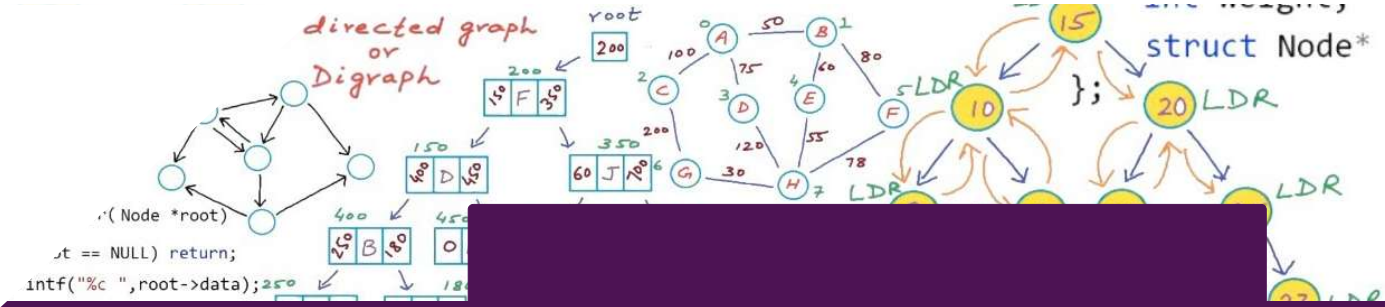
Algoritma dan Struktur Data



Pekan 4

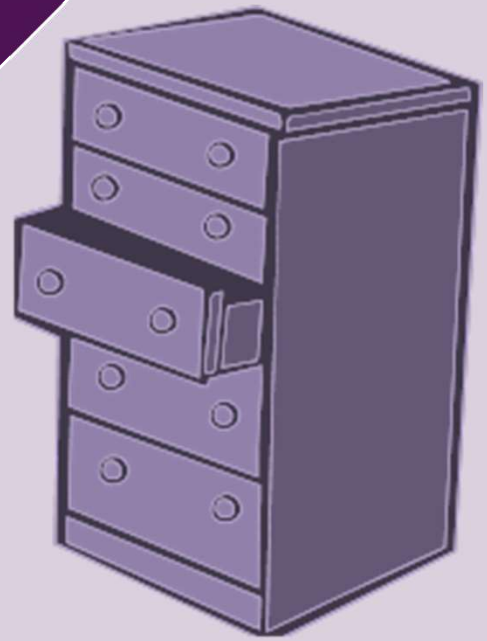

Array dan Struct
Alokasi Memori
Varian *Linked List*
Studi Kasus
Tree
Binary Tree
Advanced Tree
Hash Table
Pointer
List
Stack dan Queue
UTS
Non-Binary Tree
Extended Tree
Heap
UAS

Tujuan



1	Mahasiswa mengenal dan memahami fungsi *abstract data type* (ADT) list
2	Mahasiswa memahami operasi dasar yang dapat dilakukan ADT-list
3	Mahasiswa mampu mengimplementasikan ADT-list
4	Mahasiswa memahami kompleksitas waktu dari operasi-operasi list.

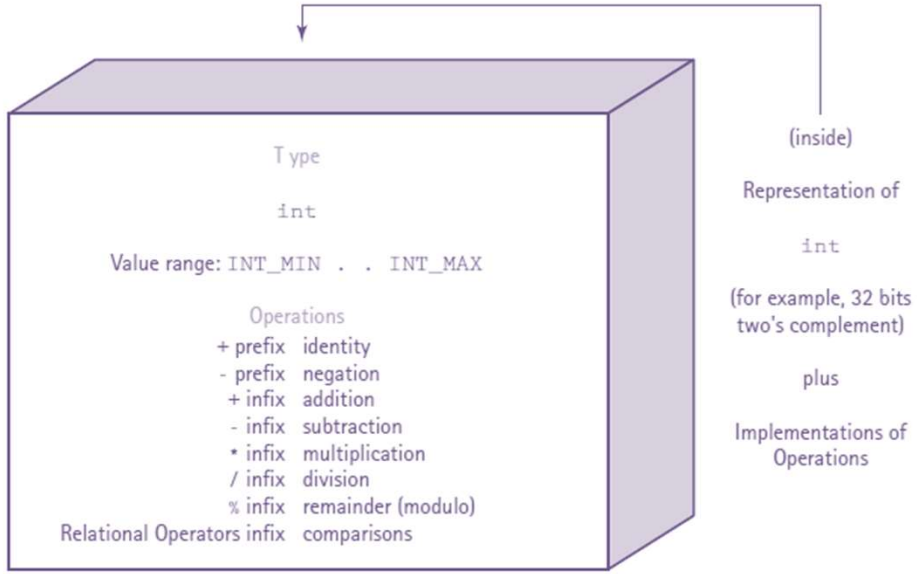
Abstract Data Type



Data Abstraction & Encapsulation

Binary 10011001

Decimal:	153	–25	–102	–103	99
Representation:	Unsigned	Sign and magnitude	One's complement	Two's complement	Binary-coded decimal



Abstract Data Type

Abstract Data Type

A data type whose properties (domain and operations) are specified independently of any particular implementation.

Data Structure

A collection of data elements whose organization is characterized by accessing operations that are used to store and retrieve the individual data element; the implementation of the composite data members in an abstract data type

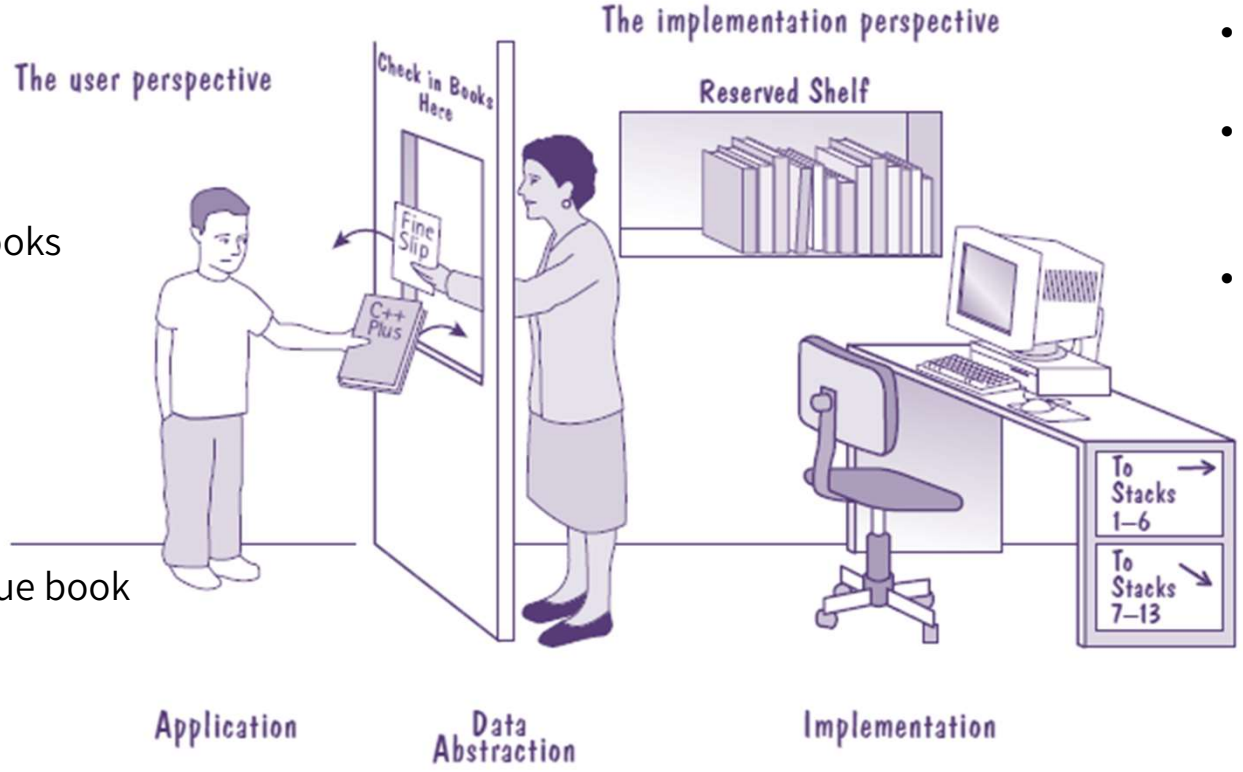
ADT from three perspective

Application (or user) level	A way of modelling real-life data in a specific context; also called the problem domain.
Logical (or abstract) level	An abstract view of the data values (the domain) and the set of operations to manipulate them.
Implementation level	A specific representation of the structure to hold the data items, and the coding of the operations in a programming language (if the operations are not already provided by the language)

ADT from three perspective

Library = collection of books

Operations:
- Check out a book
- Check in a book
- Reserve a book that is currently checked out
- Pay a fine for an overdue book
- Pay for a lost book



- How are the books catalogues?
- How are they organized on the shelf?
- How does the library process a book when it is checked in?

ADT - List

List

A homogenous collection of elements, with a linear relationship between the elements.

Linear relationship (at logical level) each element except the first has a unique predesessor, and each element except the last has a unique successor.

List

Unsorted	A list in which data items are placed in no particular order; the only relationships between data elements are the list predecessor and successor relationship
Sorted	A list that is sorted by the value in the key; a semantic relationship exists among the key of the items in the list.

Key: a member of a record (struct) whose value is used to determine the logical and/or physical order of the items in a list

Key Example

A list of students on the honor roll can be sorted :

- Alphabetically by name → Key : Name
- Numerically by student ID → Key : ID Number

Operasi Dasar

Operasi Dasar

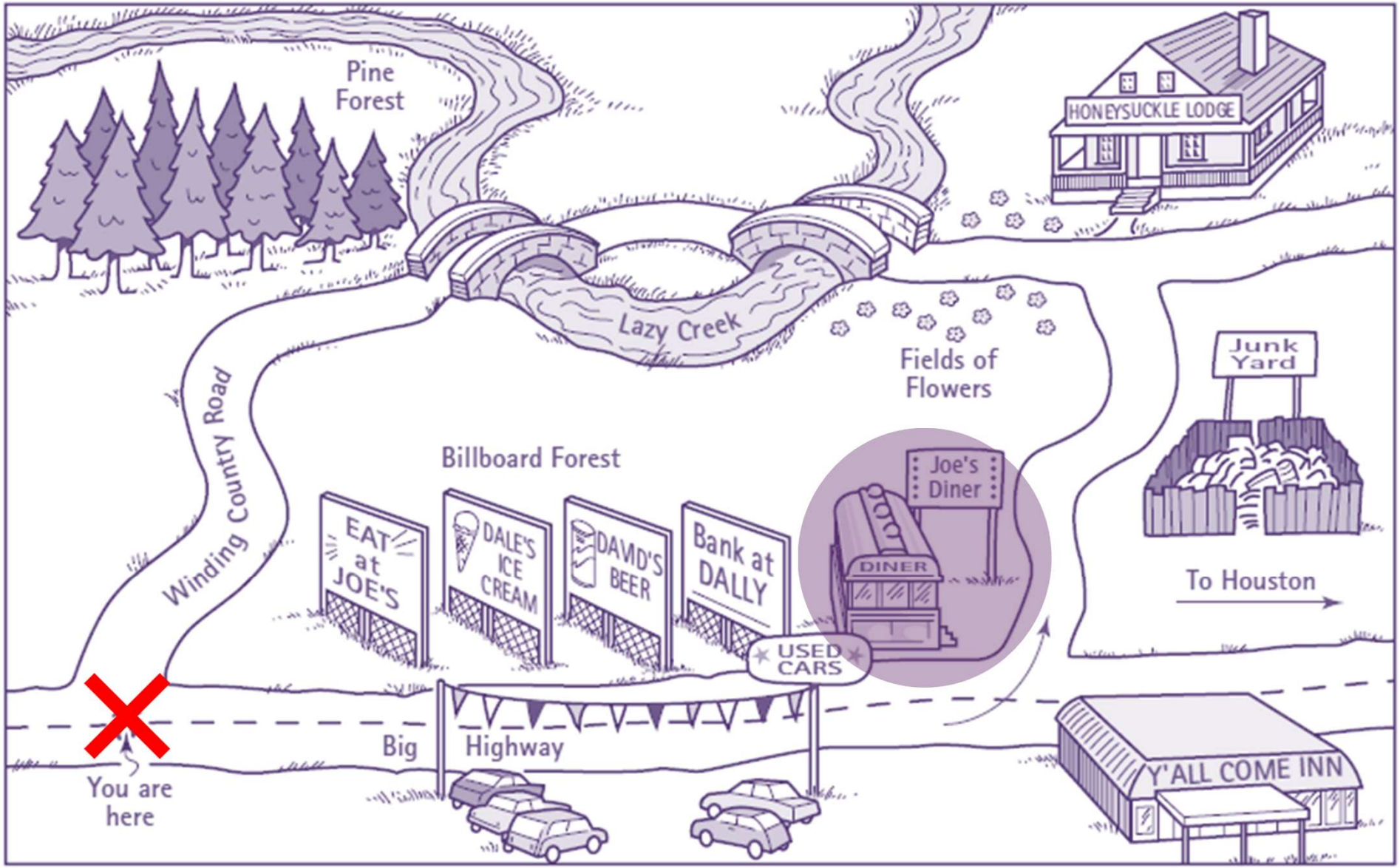
- `InsertItem(item)`
- `DeleteItem(item)`
- `RetrieveItem(item)`
- `MakeEmpty`
- `IsFull`
- `Lengths`

Implementasi

Implementasi

- Array
- Array Dinamis
- Linked-list

Kompleksitas


Pine Forest
HONEYSUCKLE LODGE
Lazy Creek
Fields of Flowers
Junk Yard
Winding Country Road
Billboard Forest
Joe's Diner
DINER
EAT at JOE'S
DALE'S ICE CREAM
DAVID'S BEER
Bank at DALLY
USED CARS
To Houston
You are here
Big Highway
Y'ALL COME INN

Big-O Notation

(Order of magnitude)

A notation that expresses computing time (complexity) as the term in a function that increases most rapidly relative to the size of a problem.

$$f(N) = N^4 + 100N^2 + 10N + 50$$

N = size of the problem

Big-O Notation

(Order of magnitude)

Most of problem in this course involve data structure and abstract data structure (list, stack, queue, tree). Each structure composed of elements.

We develop algorithms to add an element to the structure and to modify or delete an element from the structure.

We can describe the work done by these operations in terms of N, where N is the number of elements in the structure.

Big-O Example

(Order of magnitude)

```
Open the file
while more elements in list do
    Write the next element
```

(N * time_to_write_one_element) +
time_to_open_the_file

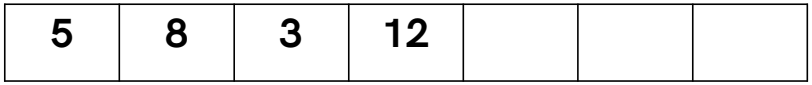
O(N)

Common order of magnitude

$O(1)$	Bounded time
$O(\log_2 N)$	Logarithmic time
$O(N)$	Linear time
$O(N \log_2 N)$	$N \log_2 N$ time
$O(N^2)$	Quadratic time
$O(N^3)$	Cubic time
$O(2^N)$	Exponential time

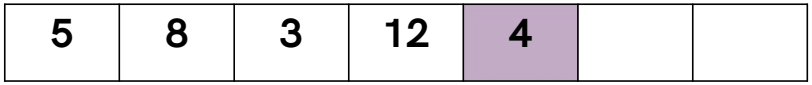
Kompleksitas Implementasi - Array

	Add	Remove
Beginning		
End		
Middle		



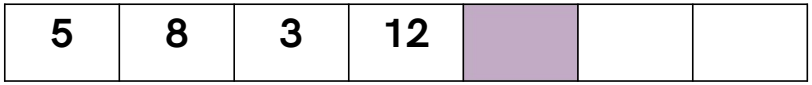
Kompleksitas Implementasi - Array

	Add	Remove
Beginning		
End	O(1)	
Middle		



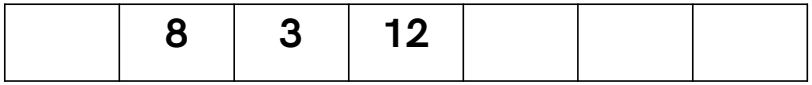
Kompleksitas Implementasi - Array

	Add	Remove
Beginning		
End	O(1)	O(1)
Middle		



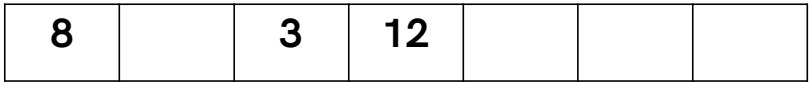
Kompleksitas Implementasi - Array

	Add	Remove
Beginning		O(n)
End	O(1)	O(1)
Middle		



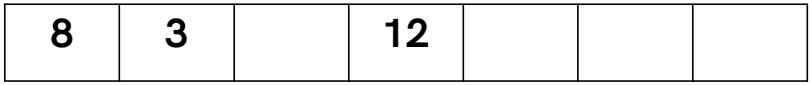
Kompleksitas Implementasi - Array

	Add	Remove
Beginning		O(n)
End	O(1)	O(1)
Middle		



Kompleksitas Implementasi - Array

	Add	Remove
Beginning		O(n)
End	O(1)	O(1)
Middle		

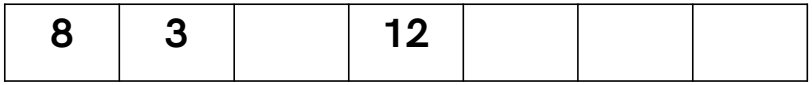


Kompleksitas Implementasi - Array

	Add	Remove
Beginning		O(n)
End	O(1)	O(1)
Middle		

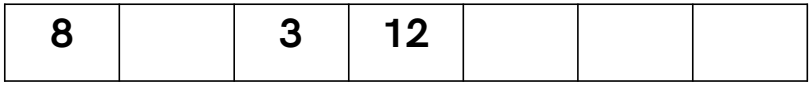
Kompleksitas Implementasi - Array

	Add	Remove
Beginning	O(n)	O(n)
End	O(1)	O(1)
Middle		



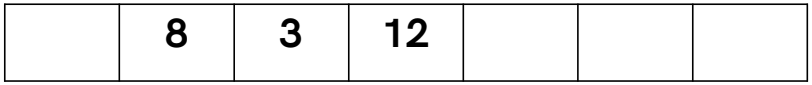
Kompleksitas Implementasi - Array

	Add	Remove
Beginning	O(n)	O(n)
End	O(1)	O(1)
Middle		



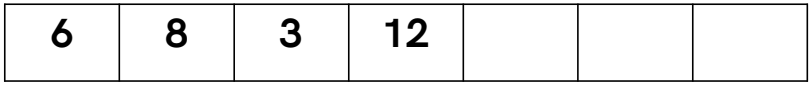
Kompleksitas Implementasi - Array

	Add	Remove
Beginning	O(n)	O(n)
End	O(1)	O(1)
Middle		



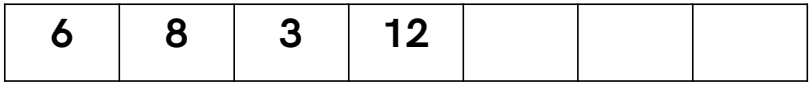
Kompleksitas Implementasi - Array

	Add	Remove
Beginning	O(n)	O(n)
End	O(1)	O(1)
Middle		



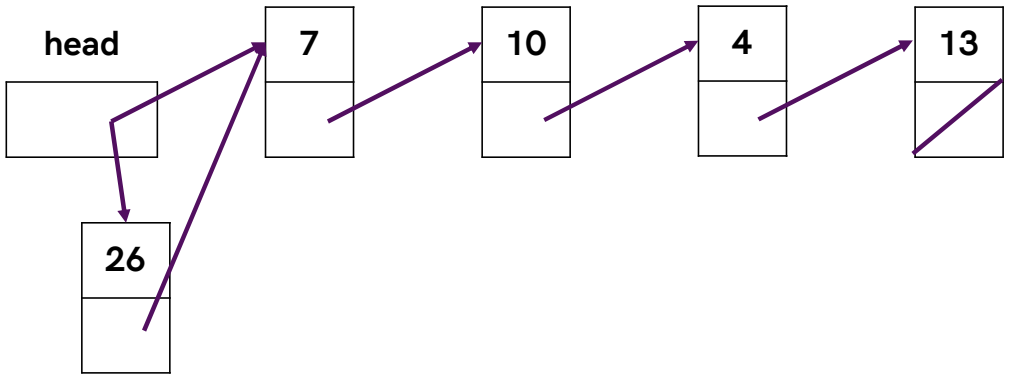
Kompleksitas Implementasi - Array

	Add	Remove
Beginning	O(n)	O(n)
End	O(1)	O(1)
Middle	O(n)	O(n)



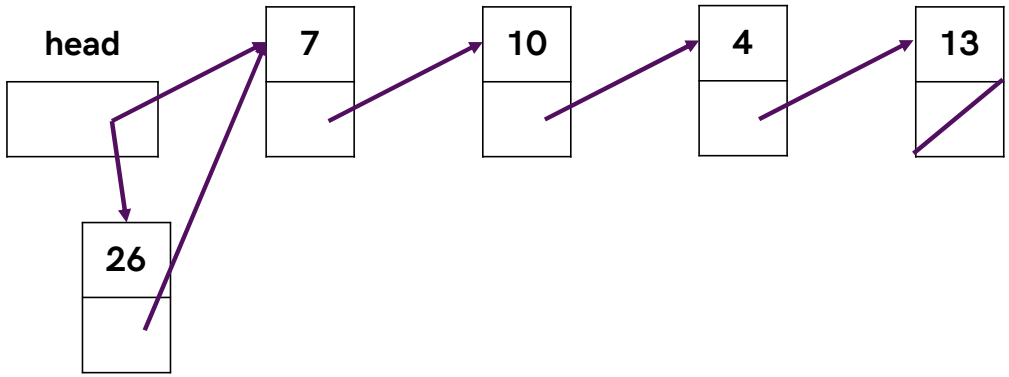
Kompleksitas Implementasi – Linked List

PushFront O(1)



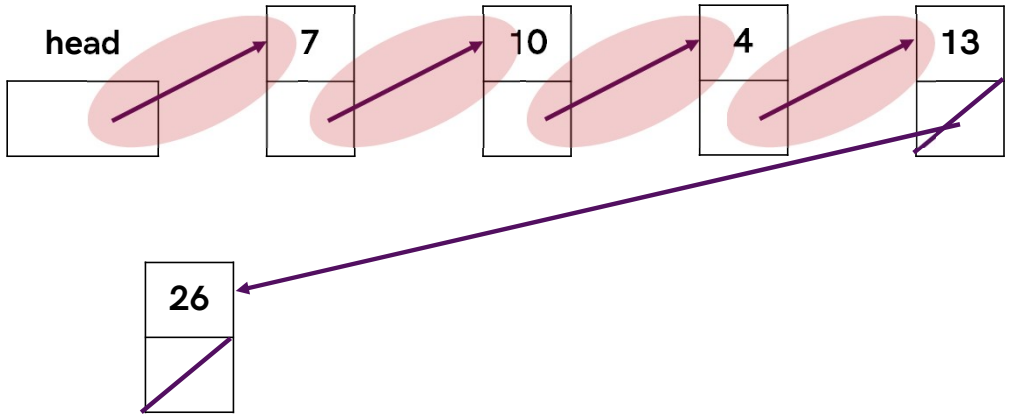
Kompleksitas Implementasi – Linked List

PopFront **O(1)**



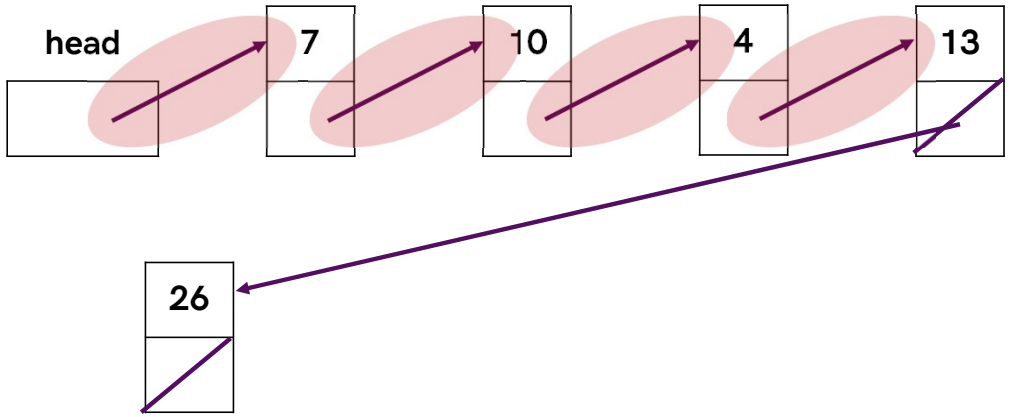
Kompleksitas Implementasi – Linked List

PushBack (No Tail) O(n)



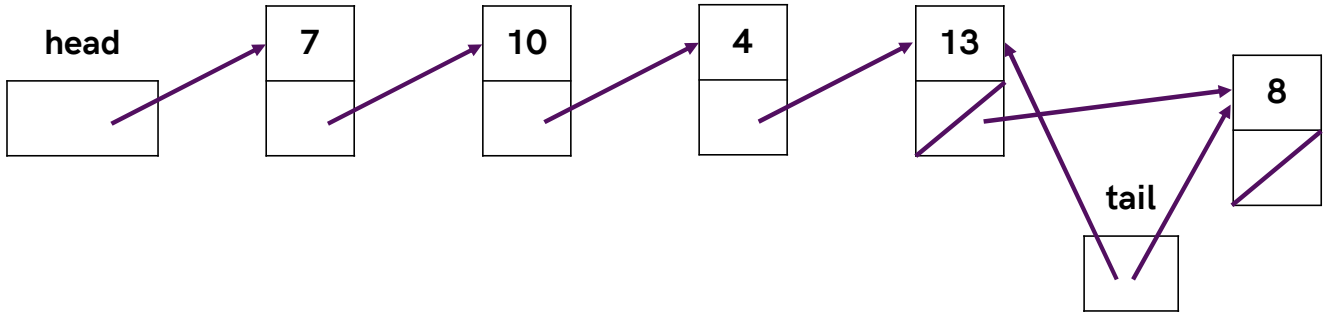
Kompleksitas Implementasi – Linked List

PopBack (No Tail) **O(n)**



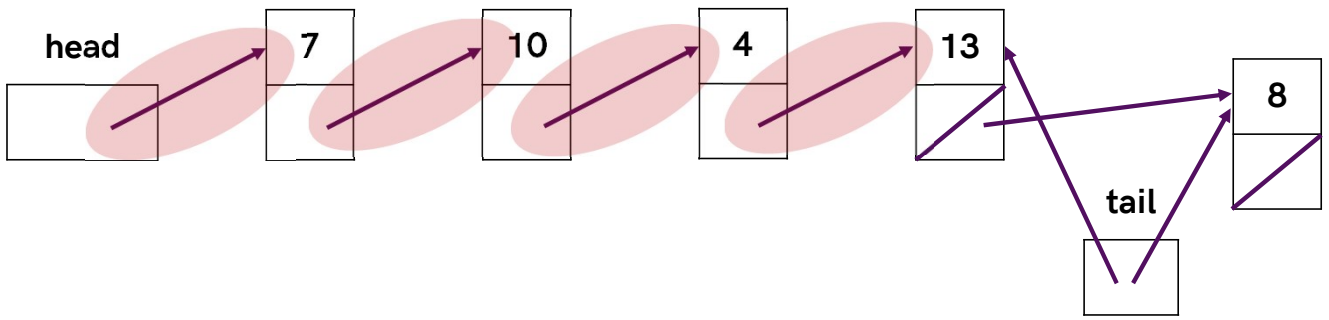
Kompleksitas Implementasi – Linked List

PushBack (With Tail) **O(1)**



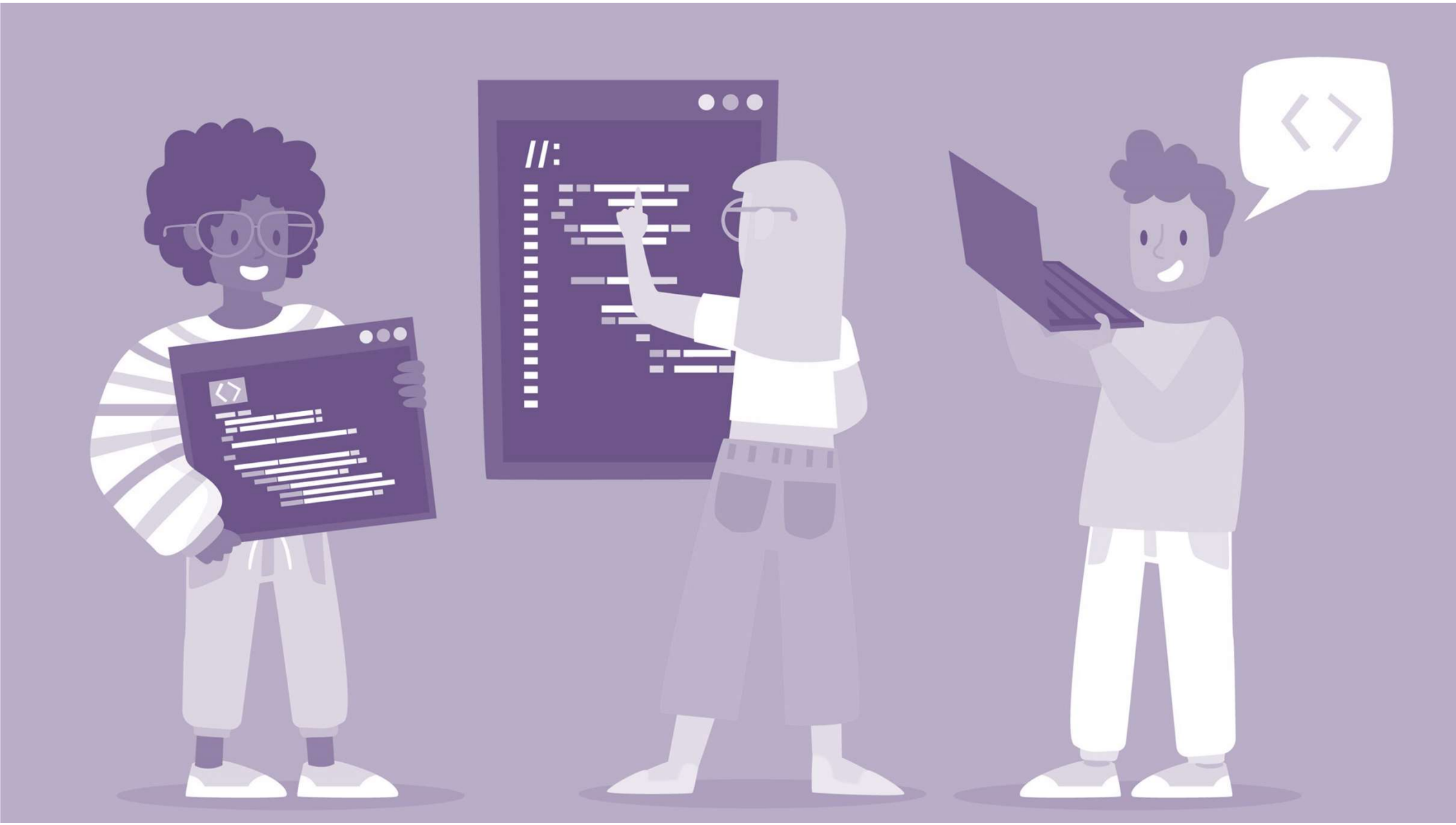
Kompleksitas Implementasi – Linked List

PopBack (With Tail) **O(n)**



Kompleksitas Implementasi (Array)

Operation	Unsorted List	Sorted List
`MakeEmpty`	O(1)	O(1)
`LengthIs`	O(1)	O(1)
`IsFull`	O(1)	O(1)
`ResetList`	O(1)	O(1)
`GetNextItem`	O(1)	O(1)
`RetrieveItem`	O(N)	Linear search: O(N) Binary search: O($\log_2 N$)
`InsertItem`		
Find	O(1)	O(N)
Put	O(1)	O(N)
Combined	O(1)	O(N)
`DeleteItem`		
Find	O(N)	O(N)
Remove	O(1)	O(N)
Combined	O(N)	O(N)



Tugas 2

Pemilik kedai kopi menyimpan data stok bubuk kopi yang dimiliki. Data yang ingin disimpan adalah nama kopi, merek kopi, rating, harga beli per gram, apakah kopi produk lokal atau import, tanggal pembelian, dan berat kopi (dalam gram). Untuk membuat satu gelas kopi dibutuhkan kira-kira 10 gram bubuk kopi. Setiap penjualan kopi akan mengurangi berat kopi pada stok dan setiap pembelian akan menambah berat kopi pada stok. Setiap akhir bulan, pemilik kopi ingin mendapatkan laporan stok bubuk kopi yang diurutkan berdasarkan tanggal pembelian dan berat kopi.

1. Buatlah program untuk menyimpan data stok bubuk kopi menggunakan struktur data berupa linked list!
2. Buatlah program untuk mencatat pembelian kopi dan penjualan kopi sesuai dengan merek kopi yang dibeli/dijual (gunakan struktur data linked list untuk menyimpan data penjualan)!
3. Buatlah program untuk mencari data stok bubuk kopi yang paling banyak!

Terapkan *modular programming* dalam mengimplementasikan program
Tiap mahasiswa bebas mendefinisikan data elemen apa yang akan disimpan untuk menyimpan data penjualan