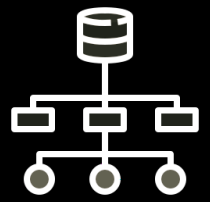
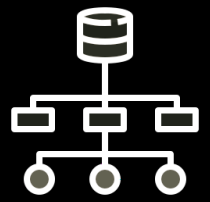
Engineer With SR



Overview

Data Structures

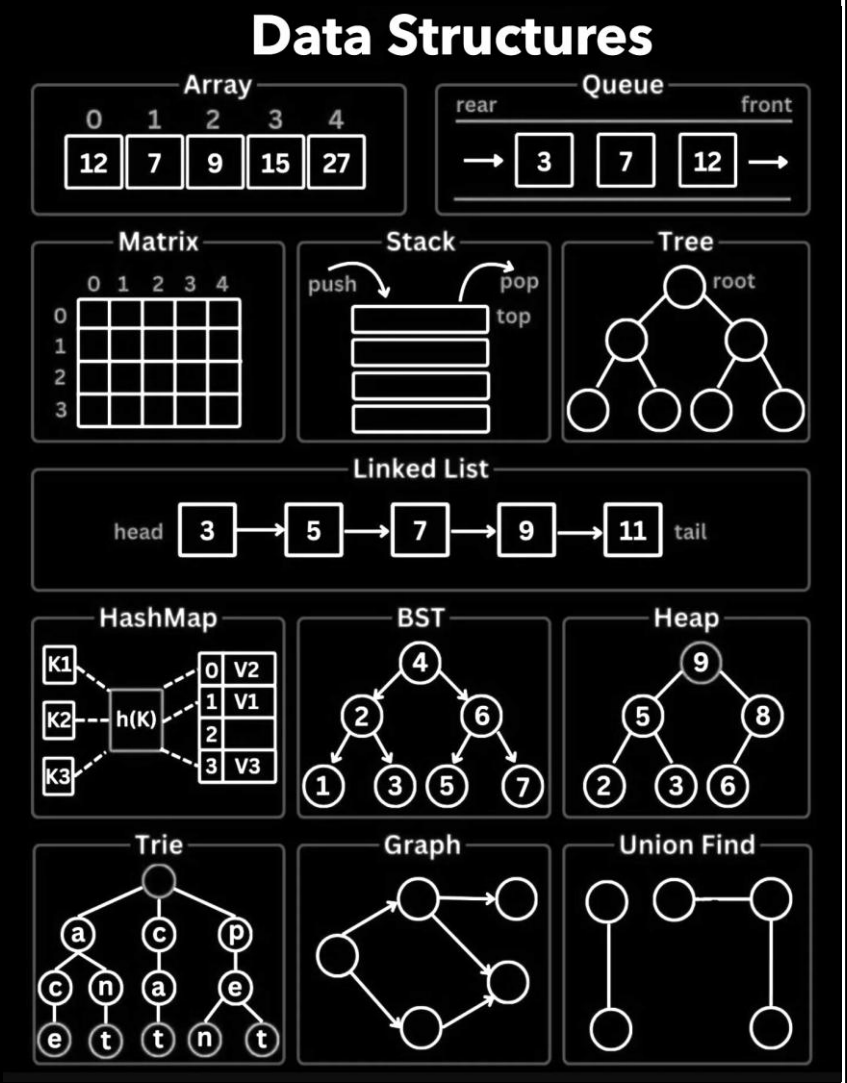


Agenda

- Introduction
- Types
- Characteristics
- Real-World Examples
- Summary

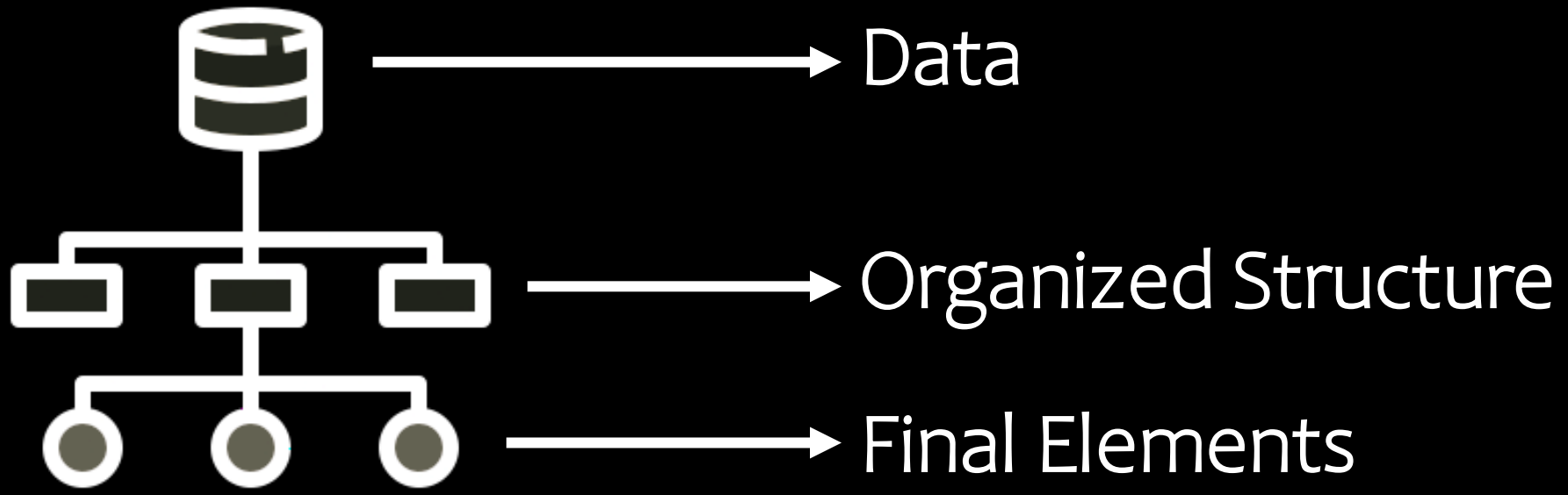
Introduction

- In the modern world, data and its information is an essential part, where data is just collection of facts or set of values that are in particular format and the information is the processed data.
- If the data is not organized effectively, it is very difficult to perform any task on large amount of data. If it is organized effectively then any operation can be performed easily on that data.
- If the data is stored in a well-organized way on storage media and in computer's memory then it can be accessed quickly for processing that further reduces the latency and the user is provided a fast response.
- A data structure is a particular way of organizing a large amount of data more efficiently in a computer so that any operation on that data becomes easy.
- In other words, Data structure is a way of collecting and organizing data in such a way that we can perform operations on these data in an effective way.
- Data structures is about rendering data elements in terms of some relationship, for better performance, organization and storage.

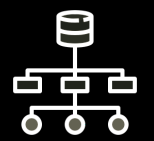


Introduction

- The logical or mathematical model for a particular organization of data is termed as a data structure.
- It represents the knowledge of data to be organized in memory. It should be designed and implemented in such a way that it reduces the complexity and increases the efficiency.
- Data Structure is useful in representing the real-world data and its operations in the computer program.

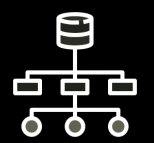


Types



- Based on the organizing method of a data structure, data structures are divided into two types:
  1. Primitive Data Structures (Built-In Data Structures)
  2. Non-primitive Data Structures (User-defined Data Structures)

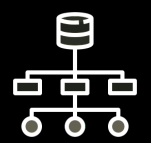
Types



Primitive Data Structures

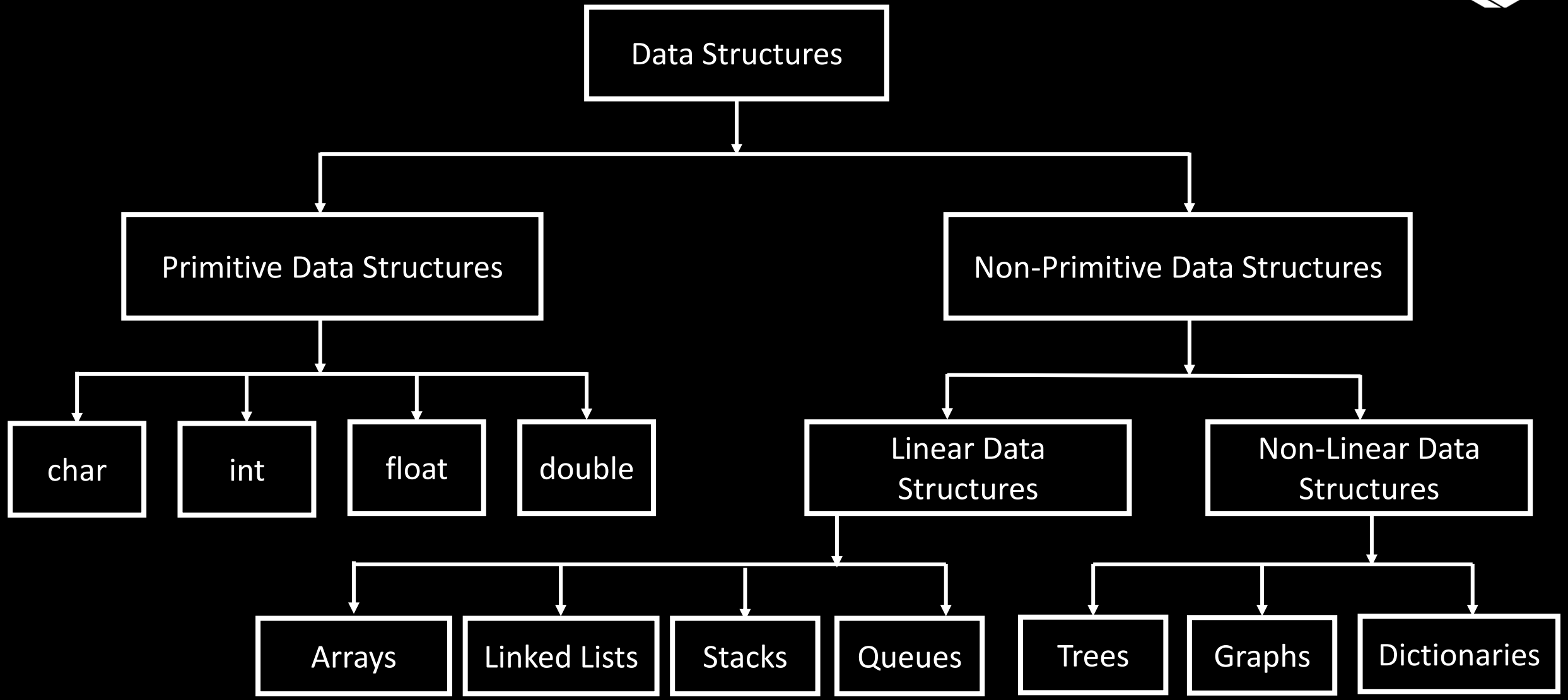
- Primitive data structures are those which are the predefined way of storing data by the system. And the set of operations that can be performed on these data are also predefined.
- Primitive data structures are char, int, double and float.
- Most of the programming languages have built-in support for the primitive data structures.

Types

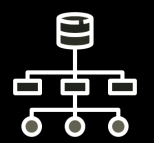


Non-primitive Data Structures

- Non-primitive data structures are more complicated data structures and they are derived from primitive data structures.
- Non-primitive data structures are used to store large and connected data.
- Some example of non-primitive data structures are: Linked List, Tree, Graph, Stack and Queue.


Data Structures
Primitive Data Structures
Non-Primitive Data Structures
char
int
float
double
Linear Data Structures
Non-Linear Data Structures
Arrays
Linked Lists
Stacks
Queues
Trees
Graphs
Dictionaries

Characteristics



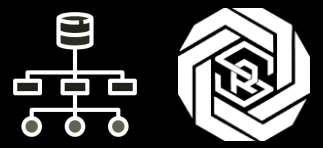
1. Linear Data Structures

- In linear data structures, all the data items are stored in a linear sequence.
- Example : Arrays, Linked Lists, Stacks and Queues.

2. Non-Linear Data Structures

- In Non-Linear data structures, all the data items are stored in random order or hierarchical order.
- The data Items are not stored in a linear sequence.
- Example : Trees, Dictionaries, Graphs and Heaps

Characteristics



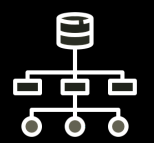
3. Homogeneous Data Structures

- In Homogeneous data structure, all the data items are of same type.
- Example : Arrays

4. Non-Homogeneous Data Structures

- In Non-Homogeneous data structures, all the data items may or may not be of same type.
- Example : Structures

Characteristics



5. Static Data Structures

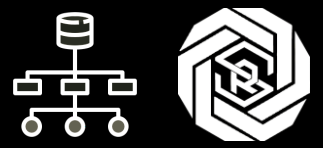
- In Static data structures, the size and structure's associated memory locations are fixed during compile time.
- Example : Arrays

6. Dynamic Data Structures

- Dynamic data structures can expand or shrink depending upon the program need and its execution.
- This expansion and shrinking happens during the program runtime. Also, their associated memory locations can change during program runtime.
- Example : Linked Lists, Stack using Linked Lists, Queues using Linked List, Trees, Heaps, etc.

- All the data structures allow us to perform different operations on data.
- Each data structure has advantages and disadvantages compared to other data structures.
- One should be able to select the best data structures based on which type of operations are required.

Real-World Examples



Examples

- Array: Student marks storage
- Stack: Undo/Redo in text editors
- Queue: Ticket booking system
- Graph: Social networks
- Tree: File system hierarchy

Summary

- Data = Raw facts
- Information = Processed data
- Data Structures = Organized way to store & process data
- Crucial for performance & solving real-world problems

Data Structures

- Primitive
  - int
  - char
  - float, etc.
- Non-Primitive
  - Array, Linked List, Stack, Queue, Graph, etc.

Linear → Sequential (Arrays)

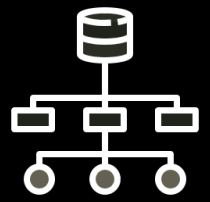
Homogeneous → Same type (Array)

Static → Fixed size (Array)

Non-Linear → Hierarchical (Trees, Graphs)

Non-Homogeneous → Different type (Structures)

Dynamic → Grow/shrink (Linked List, Stack)



End of Notes