Data Structures: Organizing Information

Data structures are the foundation for efficient software. They organize, manage, and store data effectively. This enhances performance, scalability, and maintainability.

Understanding Data and Types

Raw Data

Data is raw, unorganized facts needing processing. It is the basic building block.

Processed data in a meaningful context becomes information.

Data Types

Integers, floats, characters, and booleans classify data. These types define storage and operations.

Classifying Data Structures

Linear

Elements are arranged in a sequence, like arrays.

Non-Linear

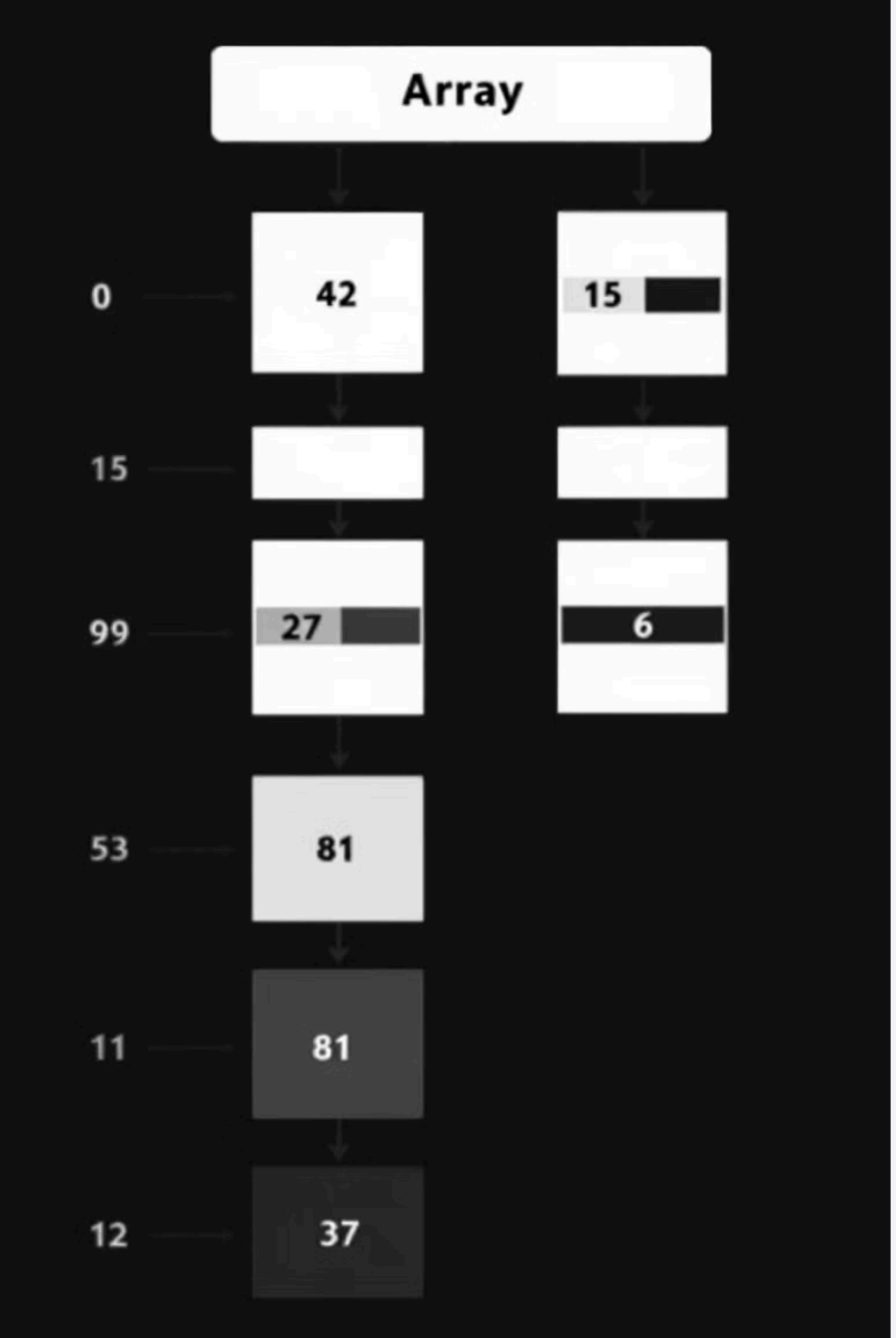
Elements are not sequentially arranged, such as trees.

Homogenous

Elements are of the same data type, like arrays.

Non-Homogenous

Elements may or may not be of the same type.



Arrays: Basic Collections

1

Definition

Arrays are collections of same-type elements.

2

Key Features

Fixed size with direct access via index.

3

Operations

Accessing, inserting, and deleting elements.

Linked Lists: Dynamic Data

Definition

Sequence of nodes containing data.

Dynamic

Efficient insertion and deletion.

Types

Singly, doubly, circular lists exist.

Linked lists offer flexibility in data management.

Stacks and Queues: LIFO & FIFO

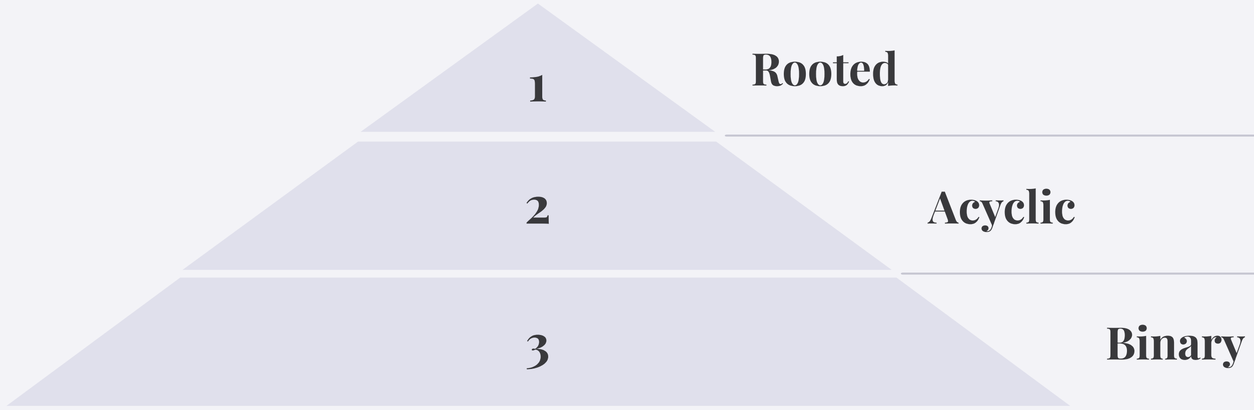
Stacks (LIFO)

- Last-In-First-Out
- Push (add), Pop (remove)
- Function call stack

Queues (FIFO)

- First-In-First-Out
- Enqueue (add), Dequeue (remove)
- Print queue is an example

Trees: Hierarchical Structures



Trees represent hierarchical data. File systems use tree structures.

Graphs: Representing Relationships

Nodes

Vertices in the graph.

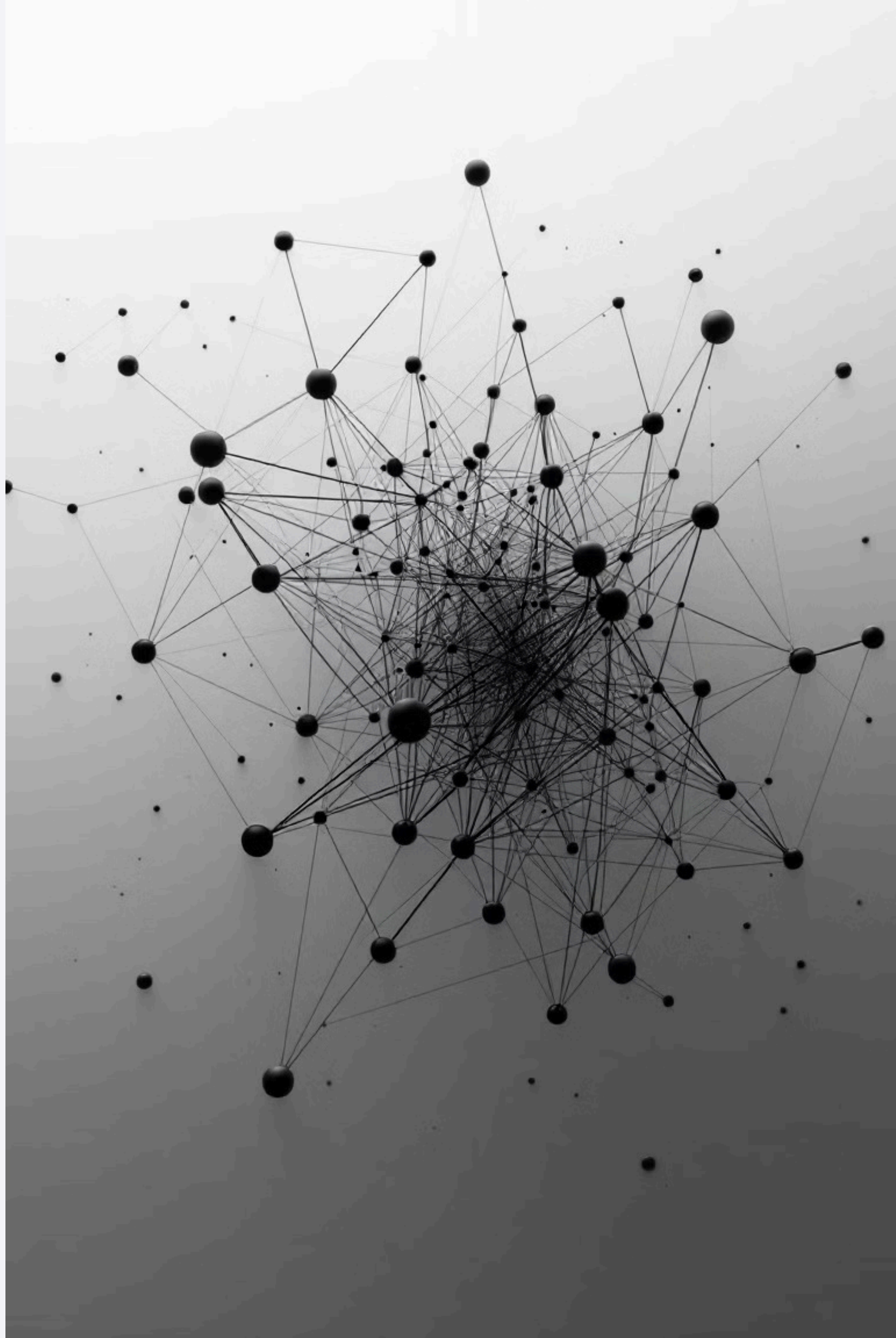
Edges

Connect the nodes.

Types

Directed, undirected.

Graphs model relationships between entities. Social networks are graph examples.



Data Structures vs. ADTs

1

Data Structure

Concrete implementation.

2

ADT

Conceptual model.

3

Relationship

Structures implement ADTs.

Lists are ADTs, implemented with arrays or linked lists.

Key Takeaways

Fundamentals

Structures are fundamental building blocks.

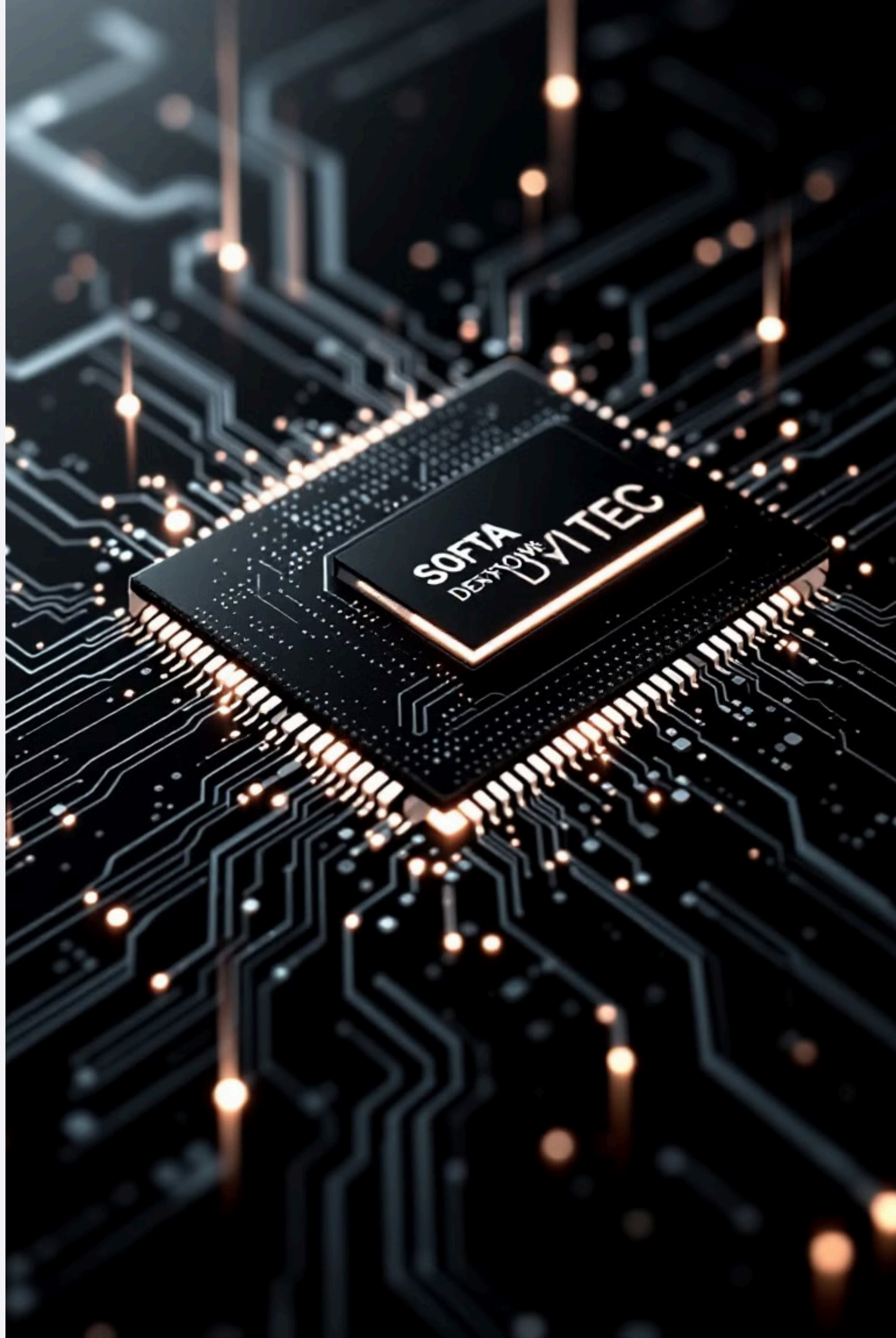
Choosing

Selection depends on requirements.

Understanding

ADTs aid in designing reusable code.

Explore complexity analysis for evaluating data structure performance.



Data Structures: Linear vs. Nonlinear

Explore the world of data structures. Understand how they organize data for efficient access. Discover the differences between linear and nonlinear structures. Learn how the right structure impacts program performance.

What are Data Structures?

Organization

Data structures organize and store data.

Key Operations

They support insertion, deletion, searching and sorting.

Efficiency

Data structures impact algorithm and program performance.

Choosing the right data structure is like selecting the right tool. This enables efficient data management and boosts overall performance.

Linear Data Structures

Arrays

Arrays use contiguous memory. They offer fast access via index. They're great for storing student IDs.

Linked Lists

Linked Lists use dynamic memory. They are flexible in size, and implement playlists.

Stacks

Stacks follow LIFO. They are used for undo/redo functionality.

Queues

Queues follow FIFO. They are used for print job management.

Linear data structures arrange data elements sequentially. Accessing elements follows a linear order, making them single-level structures.

Nonlinear Data Structures

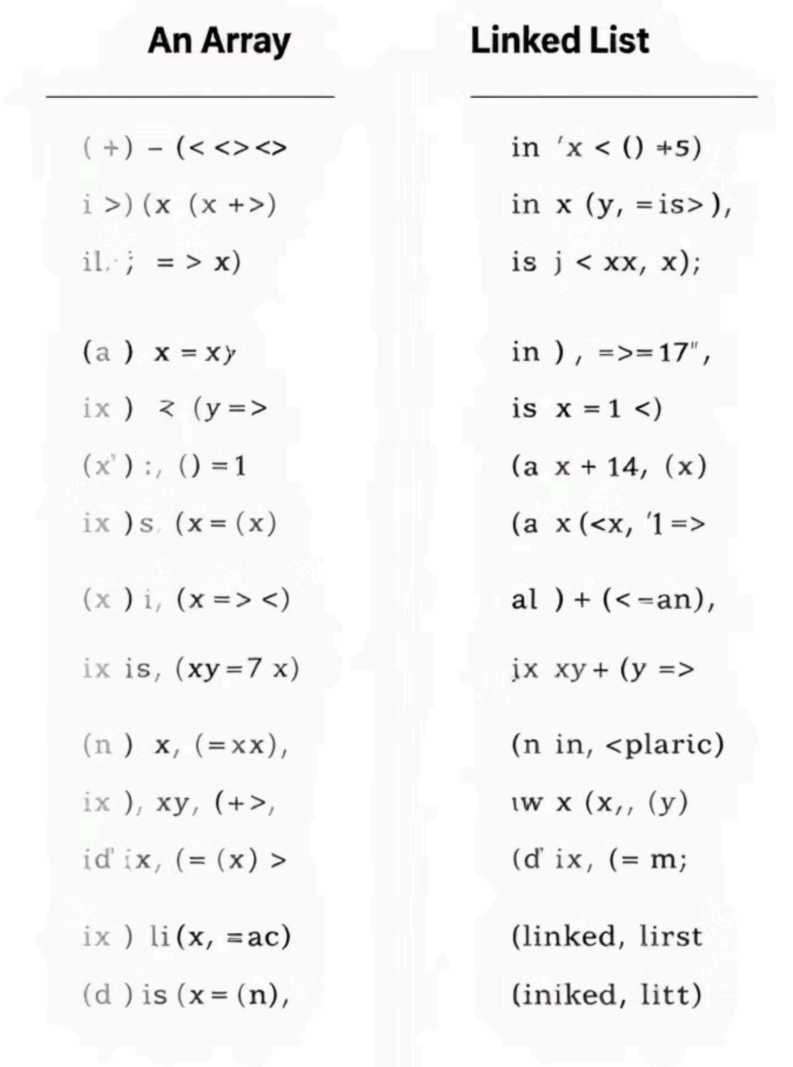
Trees

Trees have a hierarchical structure. They're used in file systems.

Graphs

Graphs are networks of nodes. They represent relationships in social networks.

Nonlinear structures have multiple connections. They feature hierarchical or network-like arrangements. These structures offer multiple levels of data organization.



An Array	Linked List
(+) – (< <><>	in ′x < () +5)
i >) (x (x +>)	in x (y, =is>),
il.·; = > x)	is j < xx, x);
(a) x = xy	in) , =>=17″,
ix) ≷ (y =>	is x = 1 <)
(x′) :, () = 1	(a x + 14, (x)
ix) s, (x = (x)	(a x (<x, ′1 =>
(x) i, (x => <)	al) + (< =an),
ix is, (xy =7 x)	jx xy + (y =>
(n) x, (=xx),	(n in, <plaric)
ix), xy, (+>,	ιw x (x,, (y)
id′ ix, (= (x) >	(d′ ix, (= m;
ix) li (x, =ac)	(linked, lirst
(d) is (x = (n),	(iniked, litt)

Arrays vs. Linked Lists

1

Arrays: Fast Access

Arrays offer fast element access using an index.

2

Arrays: Fixed Size

Arrays require a fixed size, complicating insertions.

3

Linked Lists: Dynamic Size

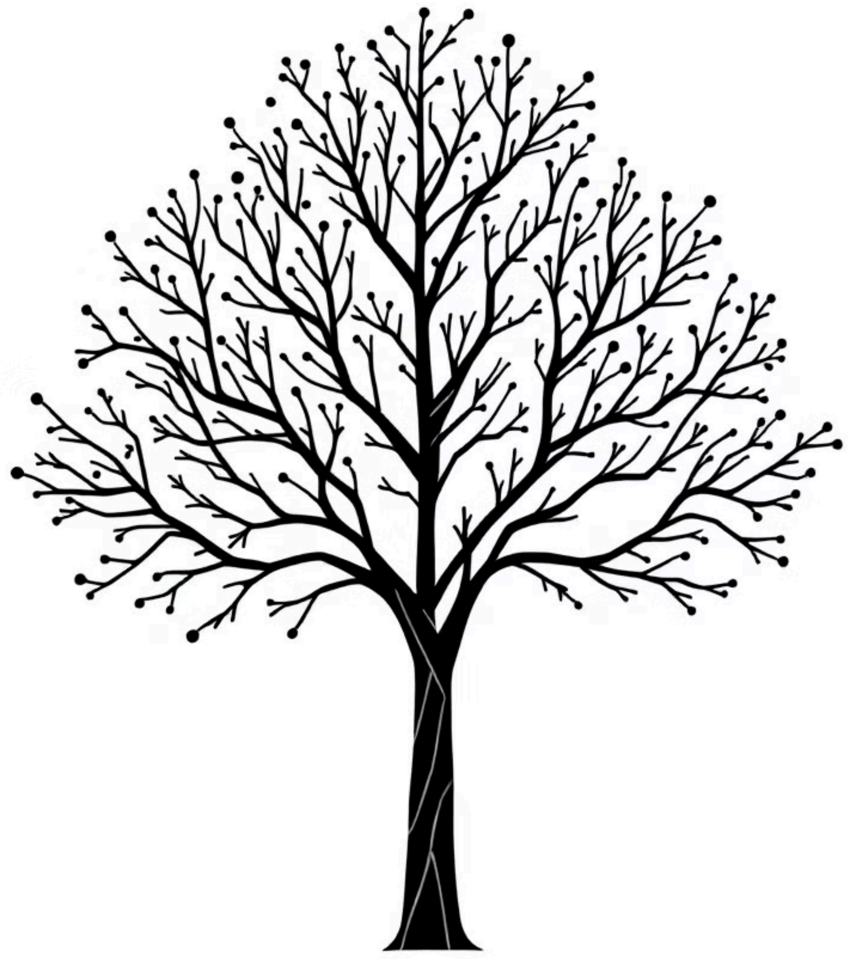
Linked Lists allow efficient insertions and deletions.

4

Linked Lists: Slower Access

Linked Lists have slower element access due to pointers.

Arrays are best for static datasets. Linked Lists excel with dynamic data.



Trees vs. Graphs

1

Trees: Hierarchical

Trees provide efficient searching. They are limited to hierarchical data.

2

Trees: Unbalanced

Trees can become unbalanced affecting performance.

3

Graphs: Flexible

Graphs offer flexible relationship modeling.

4

Graphs: Complex

Graph algorithms can be computationally expensive.

Use trees for hierarchical data. Use graphs for interconnected networks.

Real-World Applications

Playlists

Music playlists use linked lists for flexible song management.

Browser History

Browser history uses stacks to track visited pages.

Task Scheduling

Task scheduling uses queues to manage print jobs.

Social Networks

Facebook employs graphs to map user connections.

Data structures are all around us. Operating systems use trees to organize files efficiently.

Conclusion

Recap

Linear structures arrange data sequentially. Nonlinear structures have complex relationships.

Importance

Choosing the right structure is key for specific problems.

Exploration

Explore advanced structures and algorithm design.