# Notes: Abstract Data Types (ADT) vs Data Structures

**Concept Overview**

* **Abstract Data Type (ADT):**  
  A logical model or blueprint defining **behavior** of data operations without specifying **implementation** details.  
  Comparable to an **interface** or **abstract class** in programming that defines expected behavior but not how it’s achieved.
  + Focus: *What operations are performed?*
  + Ignores: *How they are implemented or in which programming language.*
* **Data Structure:**  
  A **concrete implementation** of an ADT. It represents how data is **stored, organized, and manipulated** in memory.
  + Focus: *How the operations are executed efficiently.*
  + Goal: Achieve the **fastest possible running time**, often **O(1)** (constant time).

**Key Differences**

| **Aspect** | **Abstract Data Type (ADT)** | **Data Structure** |
| --- | --- | --- |
| Definition | Specifies *what* operations will be performed | Specifies *how* operations are implemented |
| Focus | Behavior/model of data | Actual memory representation |
| Implementation | Independent of programming language | Language and memory dependent |
| Example | Stack, Queue, Priority Queue, Associative Array | Array, Linked List, Heap, Hash Table |

**Examples and Relationships**

* **Stack (ADT):**
  + Defines: push() (insert), pop() (remove)
  + Structure: LIFO (Last In, First Out)
  + Can be implemented using:
    - One-dimensional array
    - Linked list
* **Queue (ADT):**
  + Defines: enqueue() (insert), dequeue() (remove)
  + Structure: FIFO (First In, First Out)
  + Can be implemented using:
    - One-dimensional array
    - Linked list
* **Priority Queue (ADT):**
  + Defines priority-based insertion/removal
  + Implemented using:
    - **Heap** (tree-like structure)
* **Associative Array (ADT):**
  + Defines key–value pair associations
  + Implemented using:
    - **One-dimensional array** with **hash functions**

**Efficiency Considerations**

* **Arrays:** Provide linear time search ().
* **Binary Search Trees:** Reduce search to logarithmic time ().
* **Hash Tables (Associative Arrays):** Enable constant time () operations for most cases.
* Objective of data structures: **Minimize running time** for operations like search, insert, and delete.

## Array Operations

