



K.R. MANGALAM UNIVERSITY
THE COMPLETE WORLD OF EDUCATION

Course: Data Structure

(Course Code : ENCS205)

UNIT-1: Foundations of Data Structures

School of Engineering & Technology
K.R. Mangalam University, Gurugram (Haryana)

Session 13:

Applications of Arrays and Introduction to Sparse Matrices

Learning Objectives

By the end of this session, will be able to:

- Explain the concept and properties of arrays.
- Identify real-world applications of arrays across various domains.
- Understand how arrays are used to implement other data structures.
- Analyze the advantages and limitations of using arrays.
- Understand what a sparse matrix is and why it is used
- Identify different types of sparse matrices (lower, upper, tri-diagonal)

Introduction to Arrays

An array is a collection of elements of the same data type stored in **contiguous memory locations**.

- **Types of arrays:**
 - **1D Array:** Linear list of elements.
 - **2D Array:** Matrix-like representation.
 - **3D Array:** Data stored in three dimensions.
- **Key Features:**
 - Fixed size.
 - Random access to elements.
 - Homogeneous data storage.
 - Sequential memory allocation.

Core Properties & Benefits

- **Random Access:** Direct retrieval in **$O(1)$** time.
- **Cache Friendliness:** Contiguous memory improves CPU cache performance.
- **Memory Efficiency:** Stores only the required elements without overhead pointers.
- **Compatibility:** Easily handled by most programming languages and hardware.

Arrays as Building Blocks for Other Data Structures

- **Stacks & Queues:** Implemented using arrays for efficiency.
- **Hash Tables:** Arrays store hash buckets.
- **Lists & VLists:** Array-based storage for sequential access.
- **Heaps & Priority Queues:** Represented efficiently in array form.
- **Strings:** Stored as character arrays in many languages.

Mathematical & Algorithmic Applications

- **Matrix Representation:**
 - 2D arrays store matrices for linear algebra.
 - Used in **image processing** for pixel grids.
- **Dynamic Programming:** Arrays store subproblem results for reuse.
- **Lookup Tables:** Precomputed arrays speed up calculations.
- **Control Tables:** Used for controlling program flow efficiently.

System-Level & Specialized Applications

- **CPU Scheduling:**
 - Linux scheduler maintains active and expired queues as arrays indexed by priority.
- **Complete Binary Trees:**
 - Stored efficiently in arrays without pointers.
- **Speech Processing:**
 - Microphone arrays, beamforming, and ASR use array data structures.
- **Cryptography & ML:**
 - Arrays hold data sets, feature vectors, and transformation tables.

Additional Practical Uses

- **Data Buffers:** For file I/O, networking, and streaming.
- **Sorting & Searching:**
 - Sorted arrays allow **$O(\log n)$** binary search.
 - Used in applications needing quick retrieval (floor, ceiling, kth element).
- **Simulation & Modeling:** Arrays model grids, simulations, and game maps.

Limitations of Arrays

- **Fixed Size:** Resizing is costly and requires reallocation.
- **Insertion/Deletion Cost:** Requires shifting elements.
- **Type Restriction:** Homogeneous data only.
- **Memory Wastage:** If array size is larger than required.

Summary Table

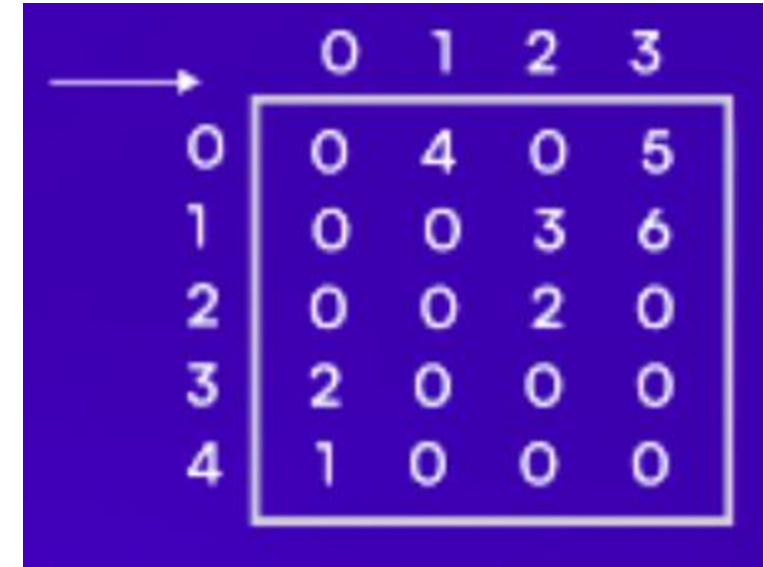
Area	Example Applications
Data Structures	Stacks, Queues, Lists, Heaps, Hash Tables
Algorithmic Uses	Dynamic Programming, Lookup Tables, Control Tables
System Applications	CPU Scheduling, Binary Trees, Speech Processing
Special Domains	Image Processing, Cryptography, Machine Learning
Practical Uses	Data Buffers, Sorting, Searching

Conclusion

- Arrays are **foundational** to computer science and engineering.
- Provide **fast, efficient, and structured** data storage.
- Support **system-level, algorithmic, and application-level** needs.
- Must be chosen wisely depending on flexibility and performance requirements.

What Is a Sparse Matrix?

- A matrix where **most elements are zero**.
- Common in large-scale, real-world datasets—e.g., in machine learning, data science, and graph theory



	0	1	2	3
0	0	4	0	5
1	0	0	3	6
2	0	0	2	0
3	2	0	0	0
4	1	0	0	0

Why Use Special Representations?

- **Problems with Normal Storage:**
 - Wastes memory (stores zeros unnecessarily)
 - More processing time for traversals/operations
- **Advantages of Sparse Storage:**
 - Saves memory
 - Reduces time complexity for operations involving only non-zero elements
- **Example:**
 - A 1000×1000 matrix = **1 million elements**
 - If only 1000 are non-zero, normal storage wastes **99.9% memory**



Thank You



K.R. MANGALAM UNIVERSITY

THE COMPLETE WORLD OF EDUCATION
Recognised under the section 2 (f) of the UGC Act 1956



Empowering the Youth; Empowering the Nation

