Arrays – Array Traversal

1. What is Array Traversal?

Array Traversal is the process of **accessing and processing each element of an array one by one**, usually from the **first index to the last index**.

In simple words, traversal means **visiting every element in the array exactly once** to perform some operation on it.

---

2. Why is Array Traversal Important?

Array traversal is one of the **most fundamental operations** in data structures because:

- Almost all array operations depend on traversal
- It is required for searching, updating, and calculating values
- It helps in understanding how arrays are accessed internally

Without traversal:

- We cannot print elements
- We cannot find sum, maximum, or minimum
- We cannot apply conditions to array elements

---

3. How Does Array Traversal Work?

Traversal uses a **loop** that starts from the **first index (0)** and moves step-by-step to the **last index (size – 1)**.

General Logic:

1. Start from index `0`
2. Access the element at the current index
3. Perform the required operation
4. Move to the next index
5. Stop when the last index is reached

---

4. Visual Representation of Traversal

Consider the array:

Index	0	1	2	3	4
Value	5	10	15	20	25

Traversal flow:

```
Start → arr[0] → arr[1] → arr[2] → arr[3] → arr[4] → Stop
```

A pointer moves **sequentially from left to right**, visiting each element once.

---

5. Types of Array Traversal

1️⃣ Forward Traversal

- Starts from index `0`
- Ends at index `n - 1`
- Most commonly used

2️⃣ Reverse Traversal

- Starts from index `n - 1`
- Ends at index `0`
- Used in reversing arrays or backward processing

---

6. Common Operations Using Traversal

Array traversal is used to:

- Print all elements
- Calculate sum of elements
- Find maximum or minimum value
- Count even or odd numbers
- Search for a specific element

Traversal is the **base step** for almost every array algorithm.

---

7. Time Complexity of Array Traversal

- Time Complexity: **O(n)**
- Each element is visited **once**
- Performance depends on array size

Traversal is efficient and unavoidable when working with arrays.

8. Advantages of Traversal

- Simple and easy to implement
- Works for all array sizes
- Forms the base for advanced algorithms
- Helps in understanding memory access

9. Limitations of Traversal

- Sequential access only
- Cannot skip elements randomly
- Slower compared to direct access when processing all elements

10. Real-World Examples of Array Traversal

- Displaying all student marks
- Calculating total sales of a week
- Checking sensor values one by one
- Finding average temperature readings
- Counting votes in an election system

11. Summary

- Array traversal means visiting each element sequentially
- It starts from index `0` and ends at `n - 1`
- Loops are used for traversal
- Traversal has time complexity **O(n)**
- It is the foundation for most array-based operations