1. Array Basics

- **Definition:** Contiguous memory collection of elements of the same type.
- **Indexing:** 0-based in most languages.
- **Time Complexity (1D array):**

Operation	Complexity
Access	O(1)
Search (linear)	O(n)
Search (binary)	O(log n)
Insertion at end	O(1) amortized (dynamic array)
Insertion middle	O(n)
Deletion	O(n)

2. Types of Arrays

Type	Description	Examples / Notes
1D Array	Single list	[1,2,3,4]
2D Array (Matrix)	Grid (row × column)	arr[i][j]

Multi-D Array	3D or higher	arr[layer][row][col]
Dynamic Array	Resizable (vector, ArrayList)	Automatic resizing
Circular Array	Wrap-around indexing	Queue simulation
Sparse Array	Mostly zeros	Optimize memory
Sorted Array	Elements sorted	Binary search possible

3. Key Concepts

- **Prefix Sum:** Fast subarray sum queries
  `prefix[i] = arr[0] + arr[1] + ... + arr[i]`
- **Two Pointer Technique:** For pairs/triplets in sorted arrays
- **Sliding Window:** Max/min subarray sums, fixed or variable length
- **Frequency Counting / HashMap:** Count duplicates, pairs with sum k
- **In-place modification:** Rotate, reverse, move zeros
- **Kadane's Algorithm:** Max subarray sum in O(n)
- **Binary Search:** For sorted arrays / rotated arrays

4. Common Patterns & Problems

Pattern	Use Case	Example Problems
Two Pointer	Pairs, triplets, sum problems	Two Sum, Triplets, Container With Most Water

Sliding Window	Subarray sum/max/min	Max sum subarray of size k, Longest substring without repeating
Prefix Sum	Range sum queries	Subarray sum = k, 2D prefix sum
Hash / Frequency Count	Count duplicates or pairs	Count pairs with sum k, Find duplicates
In-place modification	Optimize space	Reverse array, Rotate array, Move zeros
Kadane's	Max subarray sum	Maximum subarray, Max product subarray
Binary Search	Sorted / rotated arrays	Search, Find floor/ceiling, Rotated array search

5. 1D Array Problems

Easy:

- Reverse array
- Min/Max element
- Sum of array
- Linear search
- Remove duplicates (sorted)

Medium:

- Two Sum / Three Sum
- Move zeros to end
- Rotate array
- Kadane's maximum sum
- Subarray with given sum

Hard:

- Trapping rainwater
- Maximum product subarray
- Merge intervals
- Sliding window maximum
- Subarray problems with constraints

6. 2D Array / Matrix Problems

- Row / Column sum
- Spiral traversal
- Rotate matrix 90°
- Search in sorted 2D matrix
- Submatrix sum / 2D prefix sum
- Unique paths / DP on grid

7. Problem Solving Workflow (Interview Ready)

1. **Understand problem** → Input, output, constraints
2. **Check examples manually** → edge cases
3. **Identify pattern** → Two-pointer, sliding window, prefix sum
4. **Brute force** → $O(n^2)$ for understanding
5. **Optimize step by step** → $O(n)$ or $O(\log n)$
6. **Write clean code** → Comment logic
7. **Test edge cases** → empty, single element, duplicates, negative numbers
8. **Analyze complexity** → Time & Space

8. Tips for Mastery

- **Visualize array transformations** on paper
- **Memorize patterns** rather than individual problems
- **Practice problems daily** on LeetCode / HackerRank / Codeforces
- **Use Markdown notes** to track formulas, patterns, and edge cases
- **Mix 1D, 2D, and dynamic arrays** for full coverage

ARRAY PATTERN MAP

1. Two Pointer Pattern

When to use:

- Array is sorted (or can sort)
- Find pairs, triplets, subarrays
- Opposite ends traversal

Key Idea:

- Maintain two pointers (left and right) moving toward each other
- Compare sum or condition

Example Problems:

- Two Sum (sorted array)
- Triplets with sum = 0
- Container With Most Water
- Remove duplicates in-place

2. Sliding Window Pattern

When to use:

- Subarray of fixed/variable size
- Need max, min, or sum of subarray
- Longest/shortest subarray meeting a condition

Key Idea:

- Maintain a window (start, end)
- Slide window forward while updating result
- Avoid recomputing sum every time

Example Problems:

- Maximum sum subarray of size k
- Longest substring without repeating characters
- Minimum window substring

3. Prefix Sum / Cumulative Sum

When to use:

- Range sum queries
- Subarray sum problems

Key Idea:

- Precompute prefix[i] = sum(arr[0..i])
- Subarray sum arr[i..j] = prefix[j] - prefix[i-1]

Example Problems:

- Subarray sum equals k
- Range sum query (1D / 2D)
- Maximum average subarray

4. Hashing / Frequency Count

When to use:

- Count occurrences of numbers or characters
- Find duplicates or pairs efficiently

Key Idea:

- Use hashmap or array to track frequency
- Avoid nested loops

Example Problems:

- Count pairs with sum = k
- Find duplicates in array
- Longest consecutive sequence

5. In-place Modification / Rearrangement

When to use:

- Optimize space O(1)
- Rearrange elements according to condition

Key Idea:

- Two pointers or swapping technique
- Avoid extra array

Example Problems:

- Move zeros to end
- Reverse array / rotate array
- Partitioning problems (Dutch National Flag)

6. Kadane's Algorithm / Max Subarray

When to use:

- Find maximum sum or product subarray

Key Idea:

- Keep running maximum
- Update global max whenever local sum improves

Example Problems:

- Maximum subarray sum

  Maximum product subarray

7. Binary Search on Array

When to use:

- Sorted arrays / rotated arrays
- Find element, floor, ceiling, peak, or pivot

Key Idea:

- Divide and conquer (left/right halves)
- Check mid against target or condition

Example Problems:

- Search in rotated sorted array
- First/last occurrence
- Find peak element

8. 2D Array / Matrix Patterns

Pattern	Use Case	Examples
Row/Column Traversal	Summing / searching	Row sum, Column sum
Spiral Traversal	Print elements in spiral order	Spiral print of matrix
Rotate / Transpose	Rotate 90° or flip	Rotate matrix in-place
Search in Sorted 2D	Binary search + row/col property	Search matrix

9. Bonus Tips

- **Edge cases**: empty array, single element, all duplicates, negatives
- **Start with brute force → optimize using patterns**
- **Combine patterns**: sliding window + hashmap, two-pointer + prefix sum