Arrays in DSA: Master the Most Important Data Structure

Subtitle: Learn how arrays work, solve common problems, and build a strong foundation for coding interviews.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Arrays are the most fundamental data structure in DSA. Almost every coding problem builds on array concepts directly or indirectly. If you don't master arrays, you will struggle with advanced topics like trees, graphs, and dynamic programming.

Step 1: What is an Array?

An array is a **collection of elements stored in contiguous memory locations**.

- Elements are accessed using an **index**
- Index starts from **0**

Example:

```
let arr = [10, 20, 30, 40];

console.log(arr[0]); // 10
```

Step 2: Time Complexity of Array Operations

Operation	Time Complexity
Access	**O(1)**
Search	**O(n)**
Insert	**O(n)**
Delete	**O(n)**

Insight:

- Access is fast
- Insert/Delete is slow due to shifting elements

Step 3: Basic Array Operations

1. Traversing an Array

```
for (let i = 0; i < arr.length; i++) {

  console.log(arr[i]);

}
```

2. Insertion

```
arr.push(50); // add at end
```

3. Deletion

```
arr.pop(); // remove last element
```

4. Searching

```
arr.includes(20); // true
```

Step 4: Common Array Patterns (Very Important)

1. Two Pointers Technique

Used for:

- Reversing arrays
- Finding pairs

```
let left = 0;

let right = arr.length - 1;



while (left < right) {

  [arr[left], arr[right]] = [arr[right], arr[left]];

  left++;
```

```
  right--;
}
```

2. Sliding Window

Used for:

- Subarray problems
- Maximum sum problems

```
let sum = 0;
for (let i = 0; i < k; i++) {
  sum += arr[i];
}
```

3. Prefix Sum

Used for fast range calculations

```
let prefix = [];
prefix[0] = arr[0];

for (let i = 1; i < arr.length; i++) {
  prefix[i] = prefix[i - 1] + arr[i];
}
```

Step 5: Important Problems You Must Practice

1. Find Maximum Element

```
let max = arr[0];
for (let num of arr) {
  if (num > max) max = num;
}
```

2. Reverse an Array

Use two pointers (covered above).

3. Remove Duplicates (Sorted Array)

```
let j = 0;


for (let i = 1; i < arr.length; i++) {
  if (arr[i] !== arr[j]) {
    j++;
    arr[j] = arr[i];
  }
}
```

4. Two Sum Problem

```
let map = {};


for (let i = 0; i < arr.length; i++) {
  let target = 10 - arr[i];


  if (map[target] !== undefined) {
    console.log(map[target], i);
  }
  map[arr[i]] = i;
}
```

Step 6: Mistakes Beginners Make

- Ignoring **edge cases** (empty array, single element)
- Writing only brute-force solutions
- Not understanding **time complexity**
- Memorizing solutions instead of patterns

Step 7: Practice Plan for Arrays

Day 1–2:

- Basics + traversal
- 5 easy problems

Day 3–4:

- Two pointers
- 5–7 problems

Day 5–6:

- Sliding window
- 7–10 problems

Day 7:

- Mixed revision
- Solve 10 problems

Mini Exercises

1. Find the **second largest number** in an array
2. Check if an array is **sorted**
3. Move all **zeros to the end**
4. Find the **sum of all elements**

Key Takeaways

- Arrays are the **foundation of DSA**
- Master **patterns (two pointers, sliding window, prefix sum)**
- Focus on **problem-solving, not memorization**
- Practice consistently with increasing difficulty
- Strong array skills make advanced topics easier

Visit haas.dev for more DSA problem sets, detailed guides, and coding practice resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app