# 📦 Array Interview Quick-Check Pattern

*Your go-to mental model and techniques for solving any array problem under pressure.*

---

## 1. 🧠 Understand the Type of Array Problem

- ✅ **Search** → Binary Search, Linear Scan
- ✅ **Rearrange** → Move Zeroes, Sort Colors
- ✅ **Count/Frequency** → Duplicates, Majority Element
- ✅ **Subarray** → Kadane's, Sliding Window, Prefix Sum
- ✅ **Transform** → Rotate, Reverse, Merge, Encode
- ✅ **Compare** → Two arrays, Intersection, Union

> 💡 Label it: *"Search? Count? Rearrange? Transform?"* — this triggers the right pattern.

---

## 2. 🔧 Must-Know Techniques

| Technique | Common Use Cases |
|---|---|
| Two Pointers | Remove Duplicates, Reverse, Sorted Pair Search |
| Sliding Window | Longest subarray, min/max sum, character window |
| Hash Map / Set | Frequency, Uniqueness, Duplicates, Intersection |
| Prefix Sum | Subarray range queries, sum difference |
| Sorting | Pair comparison, frequency optimization |
| Stack | Next Greater Element, Monotonic Stack problems |
| Greedy | Maximize/minimize value, merge intervals |
| Kadane's Algorithm | Maximum Subarray Sum (O(n)) |

---

## 3. 🔍 Read the Problem Carefully

- ❓ Do I need a **subarray**, **pair**, **triplet**, or **whole array** result?
- ❓ Is the array **sorted** or can I **sort** it?
- ❓ What is the **return** — index, count, value, or boolean?
- ❓ Are **modifications allowed** in place?

---

## 4. 🧩 Core Templates

---

### ✅ Two Pointers (Sorted Array)

```
function twoSumSorted(arr, target) {
  let left = 0, right = arr.length - 1;
  while (left < right) {
    let sum = arr[left] + arr[right];
    if (sum === target) return [left, right];
    if (sum < target) left++;
    else right--;
  }
  return [];
}
```

---

### ✅ Sliding Window (Max Sum Subarray of Size K)

```
function maxSubArraySum(arr, k) {
  let sum = 0, max = -Infinity;
  for (let i = 0; i < arr.length; i++) {
    sum += arr[i];
    if (i >= k - 1) {
      max = Math.max(max, sum);
```

```
      sum -= arr[i - k + 1];
    }
  }
  return max;
}
```

---

### ✅ Hash Map (Two Sum)

```
function twoSum(nums, target) {
  const map = {};
  for (let i = 0; i < nums.length; i++) {
    const diff = target - nums[i];
    if (map[diff] !== undefined) return [map[diff], i];
    map[nums[i]] = i;
  }
}
```

---

### ✅ Prefix Sum (Subarray Sum Equals K)

```
function subarraySum(nums, k) {
  let map = {0: 1}, sum = 0, count = 0;
  for (let num of nums) {
    sum += num;
    if (map[sum - k]) count += map[sum - k];
    map[sum] = (map[sum] || 0) + 1;
  }
  return count;
}
```

---

### ✅ Kadane’s Algorithm (Maximum Subarray)

```
function maxSubArray(nums) {
  let maxSum = nums[0], currSum = nums[0];
  for (let i = 1; i < nums.length; i++) {
```

```
    currSum = Math.max(nums[i], currSum + nums[i]);
    maxSum = Math.max(maxSum, currSum);
  }
  return maxSum;
}
```

---

## 5. 🧱 Edge Cases You Should Always Think About

- Empty array (`[]`)
- Single element
- Duplicates
- Sorted vs Unsorted
- All negatives (Kadane's edge case)
- Overflow (e.g., large sums)
- In-place modifications required or not
- Return early conditions (early exit on match)

> 🧠 Ask: *"Can my logic break on small, weird, or large input?"*

---

## 6. 🧠 Mental Model to Master Arrays

### 🧩 All array problems fall into these 5 types:

| Category | Trigger Examples | Key Techniques |
|---|---|---|
| **Search** | Find X, index of, Two Sum | Binary Search, Hash Map |
| **Count** | How many subarrays, frequencies | Prefix Sum, Sliding Window |
| **Rearrange** | Move zeroes, reverse, rotate | Two Pointers, Cyclic Replacement |
| **Transform** | Merge sorted, difference array | Sorting, Greedy, Pointers |
| **Max/Min** | Longest, Max Sum, Smallest diff | Kadane's, Sliding Window |

---

## 🔁 Problem Solving Loop

1. ❓ *What is being asked?* (sum, index, count, boolean?)
2. 🏷️ *What's the array type?* (sorted, duplicates, fixed size?)
3. 🧠 *Which pattern matches?* (two pointers, hash map, prefix sum?)
4. 🧪 *What are edge cases?*
5. 📉 *Can I optimize time/space further?*

---

## ✅ Final Checklist Before You Code

- Is the array sorted or do I need to sort it?
- What's the return type: index, value, length, boolean?
- Is the answer in a subarray, prefix, or whole array?
- Can I use hash map, prefix sum, or a window?
- Any obvious edge cases? (length = 0 or 1, negative values, large numbers)