## Arrays

### Common Patterns and Techniques

* **Two-Pointer Technique**: Useful for problems involving pairs (e.g., finding two numbers that sum to a target).
* **Sliding Window**: Used for problems that involve contiguous subarrays (e.g., finding the maximum sum of a subarray of size k).
* **Binary Search**: A method for finding an element in a sorted array efficiently.

### Sorting Algorithms

* **Bubble Sort, Selection Sort, Insertion Sort**: Basic sorting algorithms with O(n²) complexity.
* **Merge Sort, Quick Sort**: More efficient sorting algorithms with O(n log n) complexity.
* **Understanding the trade-offs**: Space complexity, stability, and performance of different algorithms.

### Searching Algorithms

* **Linear Search**: Iteratively checking each element in the array. O(n) time complexity.
* **Binary Search**: A more efficient search method for sorted arrays. O(log n) time complexity.

### Common Interview Problems

* **Finding Duplicates**: Check for duplicates in an array.
* **Subarrays and Subsequences**: Finding the maximum sum of a subarray, longest increasing subsequence.
* **Merging and Sorting**: Merging two sorted arrays, sorting an array.
* **Rotation**: Rotate an array to the left or right.

### Space Complexity

* **Understanding how space is allocated**: Fixed size vs. dynamic size and implications on performance.
* **In-place algorithms**: Techniques to modify arrays without using extra space.

### Edge Cases

* **Empty Arrays**: Handling scenarios where the input may be an empty array.
* **Single Element Arrays**: Understanding behavior with one-element arrays.
* **Large Arrays**: Consideration of performance and memory usage with large datasets.

### Practical Applications

* **Data Storage**: Arrays are often used to store lists of data.
* **Matrices**: 2D arrays for mathematical computations.
* **Buffers**: Arrays are used in implementing various buffers in applications.

## Hashes