When solving **"Array"** related problems on LeetCode or any other programming platform, it's important to keep in mind several key concepts and techniques. Here are some must-know facts and tips for effectively solving array-related problems:

* **1. Array Traversal:**
  + Master the basic skill of traversing an array in both directions (forward and backward).
* **2. Array Manipulation:**
  + Understand and practice basic array manipulation operations, such as swapping elements, reversing an array, and rotating an array.
* **3. Two Pointers Technique:**
  + Learn and apply the two pointers technique for efficiently solving problems requiring traversal or comparison of elements in an array.
* **4. Sliding Window Technique:**
  + Understand and use the sliding window technique for solving problems that involve subarrays or sublists.
* **5. Sorting Arrays:**
  + Familiarize yourself with sorting algorithms (e.g., quicksort, mergesort) as sorting can be useful in solving many array problems.
* **6. Binary Search:**
  + Understand binary search and apply it when solving problems that involve searching in a sorted array.
* **7. Hashing:**
  + Learn about hashing and use it to optimize certain array-related problems, especially those involving frequency counting or checking for duplicates.
* **8. Prefix Sum Technique:**
  + Understand and use the prefix sum technique for solving problems that involve cumulative sums or ranges.
* **9. Arrays and Pointers:**
  + Understand the relationship between arrays and pointers in languages like C/C++. It can be helpful for optimizing certain operations.
* **10. Space Complexity:**
  + Be mindful of space complexity. Some problems can be solved with O(1) space or require extra space in the form of additional data structures.
* **11. Edge Cases:**
  + Consider edge cases when implementing solutions, such as empty arrays, arrays with a single element, or arrays with repeated elements.
* **12. Optimizations:**
  + Look for opportunities to optimize your code. Sometimes, a brute-force approach can be optimized using clever techniques.
* **13. Handling Negative Numbers:**
  + When working with arrays containing negative numbers, consider how negative numbers affect the solution, especially in problems involving sums or differences.
* **14. Time Complexity Analysis:**
  + Analyze the time complexity of your solution. Understanding the time complexity helps in choosing the right approach and optimizing your code.
* **15. Dynamic Programming (if applicable):**
  + For certain array problems, dynamic programming techniques might be applicable. Familiarize yourself with basic dynamic programming concepts.

Remember that practice is key to mastering these concepts. Solve a variety of array-related problems to build your skills and gain confidence in tackling different types of challenges.

Here's a list of some widely used algorithms for solving array-related LeetCode questions:

**Two Pointers:**

Used for problems where you need to maintain two pointers (indices) and move them toward each other or in a specific pattern.

**Sliding Window:**

Useful for solving problems involving subarrays or contiguous elements in an array.

**Kadane's Algorithm:**

Specifically used to find the maximum sum subarray within an array.

**Binary Search:**

Helpful for problems that can be optimized using binary search, especially when the array is sorted or has some order.

**Merge Intervals:**

Used to merge overlapping intervals in an array.

**Prefix Sum:**

Useful for problems that involve cumulative sums or finding subarrays with a specific sum.

**Sorting:**

Sorting the array can help in solving various problems, such as finding pairs with a specific sum or identifying the median.

**Hashing:**

Hash tables can be used to efficiently solve problems related to finding duplicates, frequency counts, or mapping elements.

**Dynamic Programming:**

Often applied to problems with optimal substructure and overlapping subproblems, such as those involving maximum subarrays or longest increasing subsequences.

**Divide and Conquer:**

Useful for solving problems by breaking them down into smaller subproblems, solving each subproblem, and combining the solutions.

**Greedy Algorithms:**

Applicable to problems where making the locally optimal choice at each stage leads to a globally optimal solution.

**Topological Sorting:**

Used for problems involving dependencies, such as course scheduling.

**Dutch National Flag Algorithm:**

Useful for sorting an array of 0s, 1s, and 2s in linear time.

**Max/Min Heap:**

Heaps can be used to efficiently solve problems related to finding the kth largest or smallest element in an array.

**Bit Manipulation:**

Applicable to problems that involve bitwise operations or XOR operations.

These algorithms cover a wide range of array-related problems on platforms like LeetCode. Understanding and practicing these techniques can be beneficial for solving various algorithmic challenges.