Sliding Window Technique — Notes

What is Sliding Window Technique?

Sliding Window is an optimization method used to solve problems involving **contiguous sequences** (subarrays or substrings) in arrays or strings efficiently. It reduces the time complexity by avoiding repeated work on overlapping parts of the data.

Why use Sliding Window?

- Naive solutions often involve nested loops (O(n^2)) checking all subarrays/substrings.
- Sliding Window reduces this to O(n) by maintaining a dynamic window over the data, updating information as the window moves.

How does Sliding Window work?

- Use two pointers or indices (usually called `start` and `end`) to mark the boundaries of the current window.
- The window “slides” through the array/string by moving these pointers.

- Depending on the problem, the window size may be fixed or variable.

- At each step, update the current window's state (like sum, max, frequency, etc.) efficiently without recomputing everything.

Types of Sliding Window:

1. **Fixed-size Sliding Window**

   - The window size remains constant throughout.

   - Common for problems like "maximum/minimum sum of subarrays of size k".

   - Move the window by one element at a time: remove the leftmost element, add the new rightmost element.

2. **Variable-size Sliding Window**

   - The window size changes dynamically depending on a condition.

   - Useful when you want to find the longest or shortest substring/subarray satisfying some property.

   - Adjust the window size by moving `start` and `end` pointers accordingly.

When to use Sliding Window?

- Problems involving contiguous elements in arrays or strings.
- Finding max/min/average/sum of subarrays or substrings.
- Finding longest or shortest substrings/subarrays with certain constraints (e.g., no duplicates, sum ≤ target).
- Optimizing brute-force approaches with nested loops into linear time.

Key Advantages

- Reduces time complexity from $O(n^2)$ to $O(n)$ in many cases.
- Uses constant or linear extra space.
- Efficient for real-time or large data processing.

Common Applications

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

- Smallest subarray with sum ≥ target
- Count of subarrays with certain properties
- Sliding window maximum/minimum
- String pattern matching problems

Important Points to Remember

- Always define what the window represents and what conditions it must satisfy.
- Carefully update the window as you move pointers: add new elements, remove old elements.
- Think about whether your window size is fixed or flexible based on the problem requirements.
- Sliding Window is often combined with data structures like hash maps, sets, or queues for tracking elements inside the window.
-