# Sliding Window – Core Patterns Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Pattern | Core Idea | Mental Idea | Typical Scenarios |
| 1 | Fixed-Length Window | The window size is constant; move it one step at a time, maintaining sums, counts, or averages. | I know my window size; just slide and update incrementally. | Maximum Sum Subarray of Size K, Average of Subarray of Size K |
| 2 | Variable-Length Window (Expand–Shrink) | Expand the window until invalid, then shrink it until valid again. | I’ll grow the window while it’s valid; when it breaks, shrink from the left. | Longest Substring Without Repeating Characters, Minimum Window Substring |
| 3 | Dynamic Condition Window | Window validity depends on a running condition (e.g., count ≤ k, sum ≤ target). | Maintain a running condition — adjust boundaries when the condition fails. | Longest Subarray with Sum ≤ K, Fruits Into Baskets |
| 4 | Counting Window (Frequency Map) | Maintain counts of elements/characters in the window to compare against a target count. | I’ll match my window’s frequency map with the target’s map. | Permutation in String, Minimum Window Substring |
| 5 | Distinct / K-Type Window | Maintain a window with at most or exactly K distinct elements. | Use a map to track frequency and shrink when distinct count > K. | Longest Substring with K Distinct Characters, Fruits Into Baskets |
| 6 | Sum / Product Constraint Window | Use prefix sum or product to maintain numeric conditions on the window. | Keep the running sum/product and shrink until it fits the limit. | Subarray Product Less Than K, Longest Subarray Sum ≤ Target |
| 7 | Monotonic Queue (Optimized Sliding Window) | Maintain a deque that tracks max/min elements for every sliding window efficiently. | I’ll use a queue to always know the max/min within the window. | Sliding Window Maximum, Minimum Window Subarray |
| 8 | Two-Array Sliding Window / Alignment Window | Use two arrays/strings, sliding one over the other to measure overlap or matching. | I’ll slide one sequence over another and measure match quality. | Find All Anagrams in a String, Longest Common Substring (optimized) |
| 9 | Prefix-Sum-Aided Window (for variable target) | Use prefix sums or hashing to handle negative numbers or non-monotonic conditions. | Classic window fails with negatives — prefix sums can bridge the gap. | Subarray Sum Equals K (hash map + window hybrid) |