7-Day Revision Plan with Curated Reading Material

All problems are handpicked from the comprehensive lists you provided.

Day	Topic	Theory & Patterns (Read These First)	Problem 1: The Foundation	Problem 2: The Classic	Problem 3: The Variation	Problem 4: The Hard One
1	**Hashing & Prefix Sums**	1. **Hashing:** GeeksforGeeks Tutorial (A great overview of the concept). 2. **Prefix Sums:** CP-Algorithms Article (The competitive programmer's choice for this topic).	#1 Two Sum	#560 Subarray Sum Equals K	#128 Longest Consecutive Sequence	-
2	**Two Pointers**	1. **Two Pointer Technique:** GeeksforGeeks Article (Covers all the basic variations you need to know).	#167 Two Sum II	#11 Container With Most Water	#15 3Sum	#42 Trapping Rain Water
3	**Sliding Window**	1. **Sliding Window:** GeeksforGeeks Article (Provides a solid template for both fixed and variable-size windows).	#3 Longest Substring Without Repeating Chars	#209 Minimum Size Subarray Sum	#567 Permutation in String	#76 Minimum Window Substring
4	**Substrings & Palindromes**	1. **Longest Palindromic Substring:** LeetCode Solution Article (Read the "Approach 4: Expand Around Center" explanation).	#242 Valid Anagram	#5 Longest Palindromic Substring	#647 Palindromic Substrings	-
5	**Subsets & Backtracking**	1. **Backtracking:** GeeksforGeeks Introduction (Focus on the state-space tree concept).	#78 Subsets	#90 Subsets II	#494 Target Sum	-
6	**Subsequences (DP)**	1. **LIS:** CP-Algorithms Article (Explains both the $O(N^2)$ and $O(N \log N)$ solutions). 2. **LCS:** GeeksforGeeks Article (A clear, standard DP explanation).	#392 Is Subsequence	#300 Longest Increasing Subsequence	#1143 Longest Common Subsequence	-
7	**Advanced DP on Strings**	1. **Edit Distance:** GeeksforGeeks Article (The classic tutorial for this must-know DP problem).	#516 Longest Palindromic Subsequence	#72 Edit Distance	#115 Distinct Subsequences	-

Arrays & String Topics

1. **Basic Array Operations**
   – Remove Duplicates, Merge, Watermelon, Majority Element

2. **Prefix Sums & Range Queries**
   – Range Sum, Subarray Sum K, Pivot Index, Chef/Subarray

3. **Array Rotation & Cyclic Operations**
   – Rotate Array, Rotate Function, Target Practice, First Missing Positive

4. **Frequency Counting & Hashing**
   – Two Sum, Disappeared Numbers, Triple, Longest Consecutive Sequence

5. **Matrix Operations**
   – Rotate Image, Spiral Matrix, String Solitaire, Set Matrix Zeroes

6. **Two Pointers (Basic → Advanced → String Variants)**
   – Two Sum II, Valid/Reverse/Vowels Palindrome, Move Zeroes
   – 3-Sum, 4-Sum & Variants
   – Dutch Flag, Container With Most Water, Trapping Rain Water
   – Nth Node From End, Interval Intersections, Sorted Squares
   – Subsequence checks, Longest Word delete, Valid Palindrome II

7. **Sliding Window (Fixed-Size → Variable → Advanced)**
   – Max Avg Subarray, Max Vowels, Find Anagrams
   – Longest Substring w/o Repeats, Min-Size Subarray, K-Distinct
   – Min-Window Substring, Permutation in String, Repeating Char Replacement
   – Sliding Window Maximum/Median, Subarrays w/ K Different

8. **Hybrid Window + Two-Pointer Patterns**
   – Fruit Into Baskets, Min Swaps to Group 1s, Beat The Odds

9. **String Matching & Transformation**
   – `strStr()`, Repeated Substring, Rotate String, Anagram Checks
   – Palindromic Substrings, Partitioning, Longest Palindromic Substring
   – Group Anagrams, Min Steps to Anagram, Valid Anagram

10. **Subsets & Bitmask-DP**
    – Subsets I/II, Target Sum, Subset-Sum Variants

11. **Subsequences & Sequence-DP**
    – Is Subsequence, LCS, LIS, Number of LIS
    – Distinct Subsequences I/II, Palindromic Subsequence, Edit-Distance–style

12. **Advanced Substring/DP Integration**
    – Edit Distance, Interleaving, Shortest Common Supersequence
    – Longest Valid Parentheses, String Compression, Valid Palindrome III
    – Longest Duplicate Substring, Delete Operation Two Strings

13. **Mixed-Pattern & Contest-Style Combinations**
    – Problem sets that blend multiple above patterns under contest time pressure

14. **Final Mixed Challenge**
    – Hard mash-ups (e.g., Job Scheduling, Super Egg Drop, CodeChef Mixes)

Pattern Recognition

1. Two-Pointers

When to spot it:

- You need to find pairs (or triples) in an **array/string** where you can move two indices from ends or a left/right boundary.
- **Array is sorted** (or you can sort it without breaking index requirements).
- Typical tasks:
    - Sum to a target (two-sum II, 3-sum, closest sum)
    - Container/trapping water (max area, trapping rain water)
    - Merging intervals or checking palindromes on strings

Rule of Thumb:

If you see "find two indices i < j such that …" **and** either the input is sorted or can be sorted, reach for two-pointers.

---

2. Sliding Window

When to spot it:

- You're looking for a **contiguous subarray or substring** that optimizes/minimizes something (sum, length, count).
- The problem asks for "longest," "smallest," or "exact" window satisfying a condition.
- You can maintain counts or sums incrementally by expanding/contracting a window.

Rule of Thumb:

If you see "subarray" or "substring" **and** a question about sum/count/unique-characters over it ("at most k," "exactly k," "max/min size"), use sliding window.

---

3. Subsets / Backtracking

When to spot it:

- You need to **generate** all combinations/subsets/permutations.
- The output asks for "all possible …" or "count of subsets," or you need to test each subset for some property.
- Constraints are small enough (n ≤ 15–20).

Rule of Thumb:

If the prompt says "find all subsets/combinations" or "choose any k out of n," lean on backtracking/bitmask DP.

---

4. Substrings vs Subsequences

Pattern	Definition	When to use
Substring	Contiguous sequence of characters	Sliding window, KMP, two-pointer on strings
Subsequence	Not necessarily contiguous (order only)	Greedy scan (isSubsequence), LCS, DP on sequences

Rule of Thumb:

"Substring" → contiguous → sliding window or KMP.
"Subsequence" → skip allowed → greedy two-pointer scan or DP (LCS-style).

---

5. Dynamic Programming (DP)

When to spot it:

- Problem asks for **optimal value** (max/min count, way counts) under overlapping subproblems.
- You see phrases like "maximum sum," "longest," "count ways," "edit distance," or constraints up to n ≈ 10^3–10^5 but with DP-friendly structure.
- State can be defined by indices (i, j), masks, or lengths.

Rule of Thumb:

If you can define dp[i], dp[i][j], or dp[mask] with smaller subproblems, go DP.

---

6. Hashing & Frequency Counting

When to spot it:

- You need to detect duplicates, anagrams, or count occurrences.
- Queries about “most frequent,” “pairs with equal x,” or “exists two with same difference.”

Rule of Thumb:

If the core operation is “counting” rather than “ordering,” a hash map (or frequency array) is your friend.

---

7. Graph / Tree Patterns

When to spot it:

- Input describes nodes/edges or a binary tree string.
- You’re asked about connectivity, shortest paths, spanning trees, or tree traversals.

Rule of Thumb:

Node-edge input → BFS/DFS. Weighted paths → Dijkstra/Bellman-Ford. Tree → recursion/tree DP.

---

8. Greedy Algorithms

When to spot it:

- You need a locally optimal choice leading to global optimum (“minimum number of intervals,” “maximize number of events attended”).
- Sort + scan pattern.

Rule of Thumb:

If a sorted order and a “take-it-or-leave-it” choice at each step works, try greedy.

🧰 Quick Pattern-Recognition Checklist

1. **Pairs in sorted data?** → Two Pointers
2. **Contiguous subarray/substring optimize?** → Sliding Window
3. **Generate all combinations?** → Backtracking / Subsets
4. **Skip allowed?** → Subsequences / LCS DP
5. **Optimal value with overlapping subproblems?** → DP
6. **Count or frequency?** → Hashing
7. **Nodes & edges?** → Graph/Tree Algo
8. **Sort + simple selection?** → Greedy

Use this checklist when you read the problem statement: highlight keywords (“subarray,” “k distinct,” “choose,” “max/min,” “path,” “interval”), match them to the rules above, and pick your pattern. With practice, pattern identification becomes almost instinctive. Good luck!