Problem Solving RoadMap

Block	Category	Subtopics / Concepts	Problem Type / Practice Goals	Trainer Notes & Suggestions
I. Non-DSA Foundations	Conditional Statements	if-else, nested if, switch, ternary	Grade calculator, menu-driven programs	Edge cases (leap year, triangle validity).
	Control Statements	for, while, do-while, break, continue	Sum of digits, factorial, number reversal	Include loop trace dry-runs.
	Pattern Printing	Half, pyramid, diamond, numeric/alphabet	Pattern tracing (nested loop flow)	Ideal for debugging & logic control.
	Arrays – 1D	Traversal, min/max, sum, reverse	Find 2nd largest, rotate array	Reinforce index tracing & memory layout.
	Arrays – 2D	Matrix traversal, diagonal sum, rotation	Spiral print, matrix addition	Emphasize nested loop reasoning.
	String	Reverse, palindrome, substring, comparison	Anagram, word count	Teach both logic and in-built methods.
	Subarray / Subsequence / Subset	Brute → optimized (Kadane, recursion, bitmask)	Max subarray sum, count subsets	Bridge to algorithmic thinking.
	Math / Number Theory	Prime, GCD/LCM, factorial, Armstrong	Numeric patterns, divisibility	Reinforce arithmetic + logic fluency.
II. OOP & Core Programming Concepts	Class & Object	Class syntax, object creation, instance vs static	Create Student/Bank/Employee classes	Introduce basic encapsulation principles.
	Constructors & Overloading	Default, parameterized, copy constructor	Initialize different types of objects	Compare function & constructor overloading.
	Encapsulation & Access Modifiers	private, public, protected	Getter/setter, data hiding	Reinforce concept of controlled access.
	Inheritance	Single, Multilevel, Hierarchical	Reuse of class methods, constructor chaining	Show runtime hierarchy in diagram.
	Polymorphism	Compile-time (overload) & runtime (override)	Area calc, animal sound, shape example	Teach dynamic dispatch mechanism.
	Abstraction	Abstract class, interface, pure virtual fn	Vehicle, payment gateway, student mgmt system	Demonstrate contract-based design.
	Aggregation & Composition	HAS-A relationship	Library, student-course model	Contrast between association & dependency.
	Exception Handling	try-catch-finally, throw, custom exceptions	Handle divide-by-zero, invalid input	Encourage input validation & safe exits.
	File Handling	FileReader/FileWriter, serialization	Read/write structured data to file	Connect to backend data flow mindset.
	Object Array & Collections	Array of objects, ArrayList/Vector basics	Store & process student records	Transition to data structure mindset.
III. Linear DSA	Array (1D, 2D)	Implementation, traversal, insert/delete	Find median, duplicates, prefix sum	Base for sorting/searching.
	Linked List	Single, Double, Circular	Create, insert, delete, reverse	Emphasize pointer and memory visualization.
	Stack	Array/LL, monotonic stack	Push/pop, balanced parentheses	Introduce LIFO memory flow.
	Queue	Linear, Circular, Deque, Priority Queue	Enqueue/dequeue, peek	Show FIFO in memory & application mapping.
	HashMap / HashSet	Key–value, frequency count	Pair sum, prefix sum	Foundation for optimized search/storage.
IV. Non-Linear DSA	Tree	Binary Tree, BST, AVL, Red-Black	Traversals, insertion, height	Recursion and recursion-tree tracing.
	Graph	Directed, Undirected, Weighted	BFS, DFS, Dijkstra	Build intuition for pathfinding & connectivity.
	Heap / Priority Queue	Min/Max heap	Insert, extract, heapify, Kth largest	Array representation + real-world mapping.
	Trie / Segment Tree *(opt)*	Trie, Segment/Fenwick Tree	Prefix search, range queries	Dream track extension.
V. Algorithmic Paradigms	Brute Force	Exhaustive search	Pair/triple sum, permutations	Teach time complexity awareness.
	Recursion	Base + recursive case, call stack	Fibonacci, subset sum	Dry-run recursion tree.
	Backtracking	State-space exploration	N-Queens, Sudoku	Differentiate recursion vs pruning.
	Divide & Conquer	Splitting/merging logic	Merge Sort, Quick Sort, Binary Search	Teach recurrence visualization.
	Dynamic Programming	Memoization & Tabulation	LCS, Knapsack, Grid paths	Build 1D → 2D → optimized pattern thinking.
	Greedy Algorithms	Local optimal → global optimal	Activity selection, coin change	Core interview readiness.
VI. Support / Bridge Topics	Bit Manipulation	AND, OR, XOR, shifts	Count set bits, power of 2	Must-cover for Amazon/Zoho.
	Complexity Analysis	Big-O time/space, nested loops, recurrences	Compare algorithms via runtime	Include short exercises per concept.
	Mixed Problem Practice	Integration across modules	Array+String, Stack+Queue, Tree+Recursion	Perfect for mock readiness.