Technical Interview Prep

11-Week Study Syllabus

AJ Consultation Services · ajconsultation.com · (904) 325-6536

This syllabus covers the core data structures, algorithms, and software engineering concepts tested in technical interviews at top companies. Each week builds on the last — complete the problems before moving forward.

Duration: 11 Weeks · **Level:** Beginner → Interview-Ready · **Instructor:** Parth Shah, Senior Software Engineer

Week 1: Arrays & Strings

■ Foundation

Learning Goals

- Understand array indexing, slicing, and mutation
- Work with ASCII values and character manipulation
- Solve problems using iteration and nested loops

Topics Covered

- Array creation, indexing, slicing
- String immutability (Python/Java)
- ASCII / ord() / chr()
- In-place vs. out-of-place operations
- Basic time complexity: O(n), O(n²)

Practice Problems

Problem	Difficulty	Pattern
Reverse a String	Easy	Iteration / two-pointer
Check Palindrome	Easy	String comparison
Find Max/Min in Array	Easy	Single pass

Remove Duplicates	Easy	Set or in-place
Rotate Array by K	Medium	Index math

■ *If you're new to Big-O, spend extra time here. Everything downstream depends on understanding O(n) vs O(n²).*

Week 2: Recursion

■ Core Concept — Required for Trees & Graphs

Learning Goals

- Understand the call stack and base cases
- Convert iterative solutions to recursive ones
- Recognize when recursion is the natural fit

Topics Covered

- Base case + recursive case pattern
- Call stack depth and stack overflow
- Tail recursion awareness
- Memoization intro (top-down)
- Recursion vs. iteration trade-offs

Practice Problems

Problem	Difficulty	Pattern
Factorial	Easy	Classic recursion
Fibonacci (naive → memo)	Easy→Medium	Introduce memoization
Sum of Nested List	Medium	Recursive structure
Power Function (x^n)	Medium	Divide & conquer
Flatten Nested Array	Medium	Recursive traversal

■ *You cannot understand trees or graphs without recursion. Do not skip this week even if it feels basic.*

Week 3: Two Pointers & Sliding Window

■ Pattern Recognition

Learning Goals

- Identify problems suited to two-pointer patterns
- Apply sliding window for substring/subarray problems
- Reduce $O(n^2)$ brute force to $O(n)$

Topics Covered

- Two-pointer: left/right convergence
- Two-pointer: slow/fast (Floyd's cycle)
- Sliding window: fixed vs. variable size
- Window expansion and contraction
- Anagram and substring matching

Practice Problems

Problem	Difficulty	Pattern
Two Sum II (sorted array)	Easy	Left/right pointers
Valid Anagram	Easy	Frequency map / window
Reverse Linked List	Easy	Prev/curr/next pointers
Detect Cycle in Linked List	Medium	Floyd's slow/fast
Longest Substring Without Repeating Chars	Medium	Sliding window
Minimum Window Substring	Hard	Variable window

■ *Sliding window and two-pointer are the same family — one moves both pointers inward, the other expands/contracts a range.*

Week 4: Stacks & Queues

■ Linear Data Structures

Learning Goals

- Implement stacks and queues from scratch
- Recognize LIFO vs. FIFO use cases
- Use stacks for parsing and backtracking

Topics Covered

- Stack: push/pop/peek, LIFO
- Queue: enqueue/dequeue, FIFO
- Deque (double-ended queue)
- Monotonic stack pattern
- BFS preview (queue-driven)

Practice Problems

Problem	Difficulty	Pattern
Valid Parentheses	Easy	Stack matching
Min Stack	Medium	Auxiliary stack
Evaluate Reverse Polish Notation	Medium	Stack ops
Daily Temperatures	Medium	Monotonic stack
Implement Queue Using Stacks	Medium	Design

Week 5: Hash Maps & Sets

■ Lookup Optimization

Learning Goals

- Use hash maps to trade space for time
- Identify O(1) lookup opportunities
- Solve frequency-counting and grouping problems

Topics Covered

- Hash map internals (bucket, collision)
- Dict / HashMap CRUD operations
- Set operations: union, intersection, difference
- Frequency counters
- Grouping by key

Practice Problems

Problem	Difficulty	Pattern
Two Sum	Easy	Hash map complement lookup
Group Anagrams	Medium	Sorted key grouping
Top K Frequent Elements	Medium	Frequency + heap
Longest Consecutive Sequence	Medium	Set membership
LRU Cache	Hard	OrderedDict / doubly linked list

Week 6: Object-Oriented Programming

■ Software Design

Learning Goals

- Write clean, extensible class hierarchies
- Apply the four pillars of OOP
- Design interview-ready data structure classes

Topics Covered

- Encapsulation, Inheritance, Polymorphism, Abstraction
- Interfaces vs. abstract classes
- Composition over inheritance
- SOLID principles (intro)
- __init__, __repr__, __eq__, __hash__

Practice Problems

Problem	Difficulty	Pattern
Design a Stack class	Easy	Encapsulation
Design a Bank Account	Easy	State + methods
Animal hierarchy (Dog, Cat extends Animal)	Medium	Inheritance + polymorphism
Design Parking Lot	Medium	System design + OOP
Design a LRU Cache class	Hard	OOP + DS

■ *OOP questions come up in both coding and system design rounds. Know how to defend your design choices.*

Week 7: Linked Lists & Nodes

■ Pointer Manipulation

Learning Goals

- Build and traverse singly and doubly linked lists
- Manipulate pointers without losing references
- Solve in-place reversal and merge problems

Topics Covered

- Node class: value + next pointer
- Singly vs. doubly linked list
- In-place reversal
- Merging sorted lists
- Finding middle node

Practice Problems

Problem	Difficulty	Pattern
Reverse Linked List	Easy	Iterative + recursive
Merge Two Sorted Lists	Easy	Two-pointer merge
Find Middle of Linked List	Easy	Slow/fast pointer
Remove Nth Node From End	Medium	Two-pass or one-pass
Reorder List	Medium	Find mid + reverse + merge

■ *Draw the pointer diagram on paper first. Every linked list bug is a missed pointer update.*

Week 8: Trees

■ Hierarchical Structures

Learning Goals

- Traverse binary trees in all four orders
- Apply recursion to tree problems naturally
- Recognize BST properties and use them

Topics Covered

- Binary tree node: value, left, right
- Inorder / Preorder / Postorder / Level-order
- BST: insert, search, delete
- Height, depth, diameter
- Lowest Common Ancestor

Practice Problems

Problem	Difficulty	Pattern
Max Depth of Binary Tree	Easy	DFS recursion
Invert Binary Tree	Easy	Swap children recursively
Validate BST	Medium	Range tracking
Lowest Common Ancestor	Medium	Recursive case split
Binary Tree Level Order Traversal	Medium	BFS with queue
Serialize & Deserialize Tree	Hard	Design

Week 9: Graphs

■ Network Traversal

Learning Goals

- Represent graphs as adjacency list and matrix
- Implement DFS and BFS from scratch
- Detect cycles and connected components

Topics Covered

- Directed vs. undirected graphs
- Adjacency list vs. adjacency matrix
- DFS: recursive and iterative
- BFS: level-by-level using queue
- Topological sort (Kahn's algorithm)

Practice Problems

Problem	Difficulty	Pattern
Number of Islands	Medium	DFS flood fill
Clone Graph	Medium	BFS + hash map
Course Schedule (cycle detection)	Medium	Topological sort
Word Ladder	Hard	BFS shortest path
Pacific Atlantic Water Flow	Hard	Multi-source DFS

■ *Graphs trip people up because of representation. Always clarify: directed or undirected? Weighted? Cyclic?*

Week 10: Heaps, DFS & BFS Deep Dive

■ Advanced Patterns

Learning Goals

- Use heaps for efficient min/max retrieval
- Apply DFS for path-finding and backtracking
- Apply BFS for shortest-path and level problems

Topics Covered

- Min-heap / max-heap (heapq in Python)
- K-th largest/smallest pattern
- DFS backtracking (permutations, subsets)
- BFS shortest path in unweighted graph
- Bidirectional BFS (awareness)

Practice Problems

Problem	Difficulty	Pattern
Kth Largest Element	Medium	Min-heap of size K
Find Median from Data Stream	Hard	Two heaps
Subsets / Permutations	Medium	DFS backtracking
Word Search	Medium	DFS + visited matrix
Shortest Path in Binary Matrix	Medium	BFS

Week 11: Greedy, General Techniques & Mock Interviews

■ Final Prep

Learning Goals

- Apply greedy choices where optimal substructure exists
- Know when greedy fails (need DP)
- Complete 3 full mock interviews with debrief

Topics Covered

- Greedy: interval scheduling, activity selection
- Bit manipulation basics (XOR, AND, shifts)
- Sorting tricks: custom comparators
- Dynamic Programming awareness (3 problems only)
- Interview framework: clarify → example → brute → optimize → code → test

Practice Problems

Problem	Difficulty	Pattern
Jump Game	Medium	Greedy reachability
Merge Intervals	Medium	Sort + greedy
Meeting Rooms II	Medium	Min-heap scheduling
Climbing Stairs	Easy	DP intro — Fibonacci pattern
Coin Change	Medium	DP — classic
Longest Increasing Subsequence	Medium	DP — awareness only

■ *DP is included for awareness — most companies do not ask pure DP. Focus on Climbing Stairs and Coin Change to understand the pattern. Do not let DP become a rabbit hole.*