AlgoZenith Cohort Plan

Comprehensive roadmap for the shared Google Drive curriculum.

Generated on December 10, 2025.

Purpose: clarify what should happen in each folder/week, define deliverables, and make expectations transparent.

Program guardrails

Weekly cadence: 5 instructional days + 1 contest or retrospective day; Sunday reserved for optional rest or peer debugging.

Success metrics: consistent sub-20 minute solves on medium problems, written retros for every contest, and shrinking backlog of unsolved tasks.

Module	Weeks	Theme	Primary outcomes
1	1-4	Foundations and math	Confident complexity analysis, STL mastery, math refresh
2	5-8	Recursion to DP	Search discipline, DP states, optimization catalog
3	9-12	Data structures & graphs	Segment trees, graph algorithms, string mastery
4	13-16	Expert patterns	Advanced DP, graph decomposition, probabilistic methods

Operating rhythm

Every week follows the same loop: plan (set goals, define states), learn (live session + notes), implement (code labs), compete (contest), review (retrospective, backlog trim).

Artifacts to capture: mentor-ready notes, reusable templates, metrics dashboard, and backlog labels.

Module 1 - Week 1: Foundations and Tooling

Focus: Reset baseline math and ensure everyone can reason about constraints before diving into pattern-specific training.

Priority topics: Time complexity drills, C++ STL and templates, Foundational mathematics

Daily cadence:

- Day 1: Kickoff, metric review, and writing baseline solutions for two prior AZ problems.
- Day 2: Complexity case studies, derive O bound for nested loops, create cheat sheet of growth rates.
- Day 3: STL containers (vector, deque, priority_queue) with implementation walkthroughs.
- Day 4: STL algorithms and iterators lab, refactor two baseline problems using library calls.
- Day 5: Number theory warmups (gcd, mod arithmetic) and discrete math recap.
- Day 6: Mixed practice set and retrospective; Sunday reserved for rest or optional peer review.

Deliverables:

- Complexity reference one-pager shared in Drive.
- Two refactored STL-based solutions reviewed with a peer.
- Diagnostic quiz covering logarithms and modular reasoning.

Checkpoint: Confirm everyone can estimate limits for N up to 1e6 and choose the correct STL primitive without hesitation.

Module 1 - Week 2: Prefixes, Bits, Sorting, Divide and Conquer

Focus: Layer reusable array tricks while sharpening debugging discipline.

Priority topics: Prefix and difference arrays, Bit manipulation, Sorting strategies, Divide and conquer

Daily cadence:

- Day 1: Implement prefix sum template plus sliding difference arrays for range update problems.
- Day 2: Bit masking patterns (lowbit, subset iteration) followed by targeted Codeforces drills.
- Day 3: Sorting + custom comparators, stability considerations, and debugging log template.
- Day 4: Divide and conquer walkthrough (merge sort tree intro, closest pair demo).
- Day 5: Integrated lab solving two problems using combo of prefix + bitset.
- Day 6: Mock contest featuring four tasks from the week; post-contest root cause writeups.

Deliverables:

- Reusable prefix sum snippet pushed to personal template repo.
- Bit manipulation flashcards and error checklist.
- Contest writeup with key bugs documented.

Checkpoint: Average solve time for easy prefix/bit tasks under 12 minutes; zero WA from mis-ordered comparators.

Module 1 - Week 3: Binary Search and Two Pointers

Focus: Automate thinking in monotonic search spaces and subarray scanning.

Priority topics: Binary search on answer, Ternary search where applicable, Two-pointer sweeps, Invariants and proofs

Daily cadence:

- Day 1: Classify binary search problem types; derive monotonic predicate for three historical tasks.
- Day 2: Implement lower_bound / upper_bound wrappers and handle precision-driven ternary search.
- Day 3: Two-pointer playbook (window expansion, shrinking) with dry runs on paper.
- Day 4: Code lab for streaming window problems and maximum subarray variants.
- Day 5: Debug clinic focusing on off-by-one and overflow traps.
- Day 6: Mini-contest plus solution deep dive; compile list of reusable predicates.

Deliverables:

- Binary search decision tree added to notion space.
- Two-pointer template with guardrails for indexes and invariants.
- Contest postmortem referencing at least two lessons learned.

Checkpoint: Team can articulate feasibility proof for each binary search attempt before coding.

Module 1 - Week 4: Combinatorics, Number Theory, Misc Techniques

Focus: Round out core math tools and solidify by targeted practice problems.

Priority topics: Combinatorial mathematics, Number theoretic math, Inclusion-exclusion and constructive tricks, Practice problem marathon

Daily cadence:

- Day 1: Build factorial, nCr mod prime utilities, and pre-computation strategy.
- Day 2: Number theory drills (CRT basics, fast exponentiation) with immediate coding exercises.
- Day 3: Misc techniques (meet constraints, constructive proofs) and whiteboard practice.
- Day 4: Practice problem rotation with pair-programming review.
- Day 5: Timed worksheet mixing combinatorics with prefix/bit insights.
- Day 6: Peer teaching session where each member explains one tricky concept.

Deliverables:

- Math helper module merged into template repository.
- One recorded lightning talk per student explaining chosen combinatorics trick.
- Practice log capturing accuracy and time per task.

Checkpoint: Confidently solve nCr constraints up to 1e6 and explain CRT usage scenarios.

Module 2 - Week 5: Brute Force, Recursion, Meet-in-the-Middle

Focus: Develop disciplined search patterns and pruning heuristics before scaling to DP.

Priority topics: Brute force baselines, Recursion hygiene, Meet-in-the-middle, Sweep line intuition

Daily cadence:

- Day 1: Start every new topic with brute force baseline and identify pruning levers.
- Day 2: Deep recursion session (call stack visualisation, memoization when needed).
- Day 3: Meet-in-the-middle subset sum lab plus complexity comparison to brute force.
- Day 4: Sweep line intro with events sorting, apply to geometry-lite tasks.
- Day 5: Mixed assignment requiring at least two strategies for same problem.
- Day 6: Reflection and backlog triage.

Deliverables:

- Checklist for turning brute force into optimized version.
- Document describing meet-in-the-middle template with sample states.
- Post-lab summary of sweep line pitfalls.

Checkpoint: Everyone can explain when MITM beats straight recursion and estimate memory footprint.

Module 2 - Week 6: Greedy and DP Launch + AZ Contest 7 & 8

Focus: Bridge from greedy proofs to first wave of DP states while maintaining contest cadence.

Priority topics: Greedy exchange arguments, Classic DP1 (1D, 2D), Contest readiness

Daily cadence:

- Day 1: Catalog greedy patterns (interval scheduling, Huffman) and prove correctness.
- Day 2: Fail-fast session identifying counterexamples for weak greedy hypotheses.
- Day 3: DP basics (state definition, transitions) using coin change and knapsack.
- Day 4: DP debugging lab with memory optimization exercises.
- Day 5: Contest warmup problems plus strategy talk.
- Day 6: Run AZ Contest 7 & 8 back-to-back, capture metrics and review.

Deliverables:

- Greedy proof rubric with sample counterexample template.
- DP state catalog page shared with cohort.
- Contest analytics sheet (score, penalty, mistakes).

Checkpoint: DP transitions written before code in 100 percent of attempts; greedy proofs include loop invariants.

Module 2 - Week 7: LR DP States and Mixed Practice

Focus: Scale DP thinking to prefix/suffix combinations and cross-train with mixed sets.

Priority topics: Left-right DP, Path reconstruction, Stress testing

Daily cadence:

- Day 1: Introduce LR DP pattern (prefix best + suffix best) with examples.
- Day 2: Implement reconstruction and trace outputs for debugging.
- Day 3: Mixed DP practice (profiles, transitions) with pair reviews.
- Day 4: Stress-test harness writing and random test generation.
- Day 5: Case study of DP pitfalls plus targeted remedial tasks.
- Day 6: Unrated mini contest emphasising LR states.

Deliverables:

- Template for LR DP plus explanation of prefix/suffix arrays.
- Stress-test script committed to repo.
- Mini contest scoreboard snapshot with insights.

Checkpoint: Average time to implement LR DP under 25 minutes including reconstruction.

Module 2 - Week 8: DP Optimizations

Focus: Adopt advanced optimizations so DP stays feasible for large constraints.

Priority topics: Divide and conquer DP, Convex hull trick, Bitset speedups, Memory compression

Daily cadence:

- Day 1: Overview of optimization catalogue and selection rubric.
- Day 2: Divide and conquer DP derivation and implementation workshop.
- Day 3: Convex hull trick plus Li Chao tree hands-on.
- Day 4: Bitset and SOS DP practise; evaluate memory/time tradeoffs.
- Day 5: Optimization showdown where pairs refactor naive DP to pass tighter limits.
- Day 6: Capstone review: share best patterns and plan revision week.

Deliverables:

- Decision matrix for when to apply each optimization.
- Two refactored problems demonstrating runtime drop.
- Checklist for analyzing constraints before writing DP.

Checkpoint: Students justify optimization choice with numeric estimates before coding.

Module 3 - Week 9: Segment Trees and AZ202 Contest 11 & 12

Focus: Master range query data structures and apply them under contest pressure.

Priority topics: Segment tree basics, Lazy propagation, Iterative segment trees, Contest review

Daily cadence:

- Day 1: Build base segment tree and visualize recursion tree.
- Day 2: Lazy propagation deep dive with update/query walkthroughs.
- Day 3: Iterative segment tree and Fenwick comparisons.
- Day 4: Live session replication focusing on tricky bugs.
- Day 5: Practice problem rotation on sums, minimums, k-th queries.
- Day 6: Run AZ202 Contest 11 & 12; hold round-table analysis.

Deliverables:

- Segment tree template with debug logging toggles.
- Lazy propagation example documented with diagrams.
- Contest debrief noting data structure choices.

Checkpoint: No TLEs from tree implementations; ability to extend tree to custom structs within one session.

Module 3 - Week 10: Graph Algorithms and Practice

Focus: Advance through traversal, shortest paths, and MST patterns while keeping practice volume high.

Priority topics: Graph traversals, Shortest path algorithms, Union Find & MST, Practice problems

Daily cadence:

- Day 1: DFS/BFS refresh plus connected components labs.
- Day 2: Dijkstra, 0-1 BFS, and Bellman-Ford comparisons with cost modeling.
- Day 3: Union-Find, MST (Kruskal, Prim) and DSU optimizations.
- Day 4: Problem set focusing on multi-source shortest paths and DSU on tree.
- Day 5: Dedicated practice chamber curated from AZ folder 4.
- Day 6: AZ Contest 13 & 14 simulation, followed by VOD review.

Deliverables:

- Graph algorithm crib sheet with when-to-use guidance.
- DSU implementation with rollback option for future topics.
- Contest review form capturing improvement targets.

Checkpoint: Able to derive complexity for each path algorithm and justify DSU usage, including edge cases like disconnected graphs.

Module 3 - Week 11: String Algorithms, Trie DS, Contests 15 & 16

Focus: Expand to linear string processing and data structures supporting them.

Priority topics: Prefix-function/KMP, Z-function, Trie and bitwise trie, Suffix structures overview

Daily cadence:

- Day 1: Build prefix-function and Z-function from scratch with pen-and-paper traces.
- Day 2: KMP applications plus pattern constraints labs.
- Day 3: Trie DS, bitwise tries for XOR queries, and memory tradeoff discussion.
- Day 4: Optional suffix array/automaton overview for stretch goals.
- Day 5: Practice problem block mixing string DP and tries.
- Day 6: Run AZ Contest 15 & 16 and annotate tricky string tasks.

Deliverables:

- String algorithm summary page with failure function diagrams.
- Trie implementation supporting queries for XOR and lexicographic order.
- Contest insight doc emphasising modeling.

Checkpoint: Implement prefix-function perfectly on first attempt; memory usage for tries fits limits.

Module 3 - Week 12: Digit and Tree DP + Contests 17 & 18

Focus: Blend combinatorics with structural DP to tackle high-difficulty tasks.

Priority topics: Digit DP templates, Tree DP for rerooting, Advanced contest practice

Daily cadence:

- Day 1: Digit DP framework (position, tight, sum states) with workbook problems.
- Day 2: Implement multi-constraint digit DP and memoization optimizations.
- Day 3: Tree DP introduction with rerooting technique.
- Day 4: Apply tree DP to paths, independent sets, and rerooting sums.
- Day 5: Focused drilling plus code reviews for clarity.
- Day 6: Run AZ Contest 17 & 18; emphasise state planning before coding.

Deliverables:

- Digit DP skeleton with annotation for each state variable.
- Rerooting explanation documenting transitions.
- Contest summary listing error categories.

Checkpoint: State planning time under 10 minutes; transitions validated with manual sample before runtime tests.

Module 4 - Week 13: Advanced Tree DP, Bitmask DP, Revision

Focus: Polish DP depth then dedicate time to systematic revision.

Priority topics: Heavy tree DP, Bitmask DP for subsets, DP revision week

Daily cadence:

- Day 1: More tree DP patterns (centroid, DP on virtual tree).
- Day 2: Bitmask DP (TSP style, subset transitions) plus complexity analysis.
- Day 3: Combine tree and bitmask thinking for hybrid problems.
- Day 4: Dedicated revision block revisiting weakest topics per student dashboard.
- Day 5: Peer instruction sessions to explain revised topics.
- Day 6: Cumulative DP checkpoint quiz and reflection.

Deliverables:

- Bitmask DP template with subset iteration patterns.
- Personal revision doc listing reopened topics and action plans.
- Quiz scorecard with remediation steps.

Checkpoint: Confidence rating for each DP category documented and tracked.

Module 4 - Week 14: Integration and Mock Interviews (Buffer)

Focus: Use the observed gap between Week 13 and 15 folders as a structured buffer for integration, interviews, and backlog cleanup.

Priority topics: Backlog problems, Mock interviews, Systems thinking

Daily cadence:

- Day 1: Audit open tasks from earlier modules; tag each with revisit reason.
- Day 2: Run one 60-minute mock interview per student focused on DP/graphs.
- Day 3: Implement review-driven fixes and refactors discovered in mocks.
- Day 4: Optional stretch goal problems from external judges.
- Day 5: Collaborative debugging dojo tackling hardest unsolved question.
- Day 6: Publish progress journal; plan adjustments for remaining weeks.

Deliverables:

- Mock interview feedback forms stored in shared drive.
- List of reopened problems with remediation strategy.
- Progress journal noting qualitative and quantitative gains.

Checkpoint: Backlog under control (<5 open problems) and interview feedback shows improvement on communication rubric.

Module 4 - Week 15: Bridges, SCC, DFS Tree, Binary Lifting

Focus: Tackle advanced graph decompositions and prepare for tree powering techniques.

Priority topics: Bridge and articulation detection, Strongly connected components, DFS tree applications, Binary lifting

Daily cadence:

- Day 1: Derive Tarjan style bridge/cutpoint algorithm with dry run diagrams.
- Day 2: Strongly connected components via Kosaraju and Tarjan; analyze complexity.
- Day 3: DFS tree usage (bridge tree collapse, SCC DAG) applied to sample tasks.
- Day 4: Binary lifting for LCA plus path queries; extend to k-th ancestor problems.
- Day 5: Practice set mixing bridges + LCA to cement interplay.
- Day 6: Review circle to synthesize takeaways and ensure implementations are contest ready.

Deliverables:

- Graph decomposition notebook showing step-by-step snapshots.
- Binary lifting utilities (build + query) integrated into template repo.
- One annotated solution combining SCC and DP.

Checkpoint: Bridge/SCC solutions accepted on first submission and binary lifting ready for reuse.

Module 4 - Week 16: Probability, Matrix Exponentiation, Game Theory, Contests 21 & 22

Focus: Finish with math-heavy topics and final contest push.

Priority topics: Probability and expectation, Matrix exponentiation, Game theory, AZ Contest 21 & 22

Daily cadence:

- Day 1: Probability refresh, linearity of expectation, variance basics with coding tasks.
- Day 2: Matrix exponentiation for linear recurrences, implement fast power template.
- Day 3: Impartial game theory (Grundy numbers) and sample problems.
- Day 4: Integrative lab combining matrix methods with DP.
- Day 5: Pre-contest systems check and targeted drilling.
- Day 6: Run AZ Contest 21 & 22; final retrospective and celebration.

Deliverables:

- Probability cheat sheet with sample derivations.
- Matrix exponentiation module tested against Fibonacci and DP transitions.
- Final contest retrospective capturing long-term growth metrics.

Checkpoint: Students clearly explain modeling steps for probability and complete contests without conceptual blockers.

Final recommendations

Document unresolved questions every Friday, schedule extra help early, and keep contest writeups in the shared folder so progress is easy to audit.

Use the buffer (Week 14) to clear blockers, rehearse interviews, and revisit analytics before the final wave of advanced topics.