NATIONAL SCHOOL OF SCIENCES

Roadmap to the International Olympiad in Informatics

A Comprehensive Syllabus and Preparation Guide for NEB Grade 11–12 Students

---

PREPARED BY

Divya Darsheel Sharma

September 2026

Document Approval & Preface

This comprehensive roadmap has been meticulously prepared by Divya Darsheel Sharma and formally approved by the Department of Computer Science of the National School of Sciences. It is designed to serve as the definitive guide for students under the National Examination Board (NEB) curriculum in Nepal who aspire to transition from standard school computer science to the rigorous demands of the Nepal Olympiad in Informatics (NOI) and ultimately the International Olympiad in Informatics (IOI).

> *Approved by: The Department of Computer Science, National School of Sciences. Prepared by: Divya Darsheel Sharma. Date: September 2026.*

Informatics is the biggest outlier among the academic Olympiads because the curriculum is not standard school Computer Science. It is competitive programming: algorithms, data structures, complexity analysis, implementation, and mathematical problem solving. In Nepal, the NOI pipeline progresses through District Qualifiers, Provincial Qualifiers, Pre-Finals, and Finals. The international target is the IOI, whose official syllabus explicitly defines prerequisite knowledge and deliberately excludes many advanced textbook topics to focus on algorithmic thinking and efficient implementation.

Executive Summary

This document provides a structured, rigorous, and exhaustive syllabus breakdown for NEB Grade 11–12 students. The curriculum is divided into progressive levels, from Level 0 (Programming Foundations) to Level 15 (National/International Specialization), followed by a strategic breakdown of the District, Province, National, and International competition tiers. A central tenet of this roadmap is that being good at general software engineering does not automatically translate to competitive programming strength. The IOI cares about a narrow, deep combination of algorithmic thinking, data structures, computational modeling, and efficient C++ implementation.

Level 0 – Programming Foundations

Before algorithms, students must become fluent enough in a programming language that coding itself stops being the bottleneck. The primary recommendation for competitive programming is C++, as it is the language most commonly used by IOI competitors and provides the necessary control over memory and execution speed.

Core C++ Programming (Priority: Maximum)

- Variables, integer/floating-point types, characters, booleans, and strings.
- Operators, expressions, and standard input/output.
- Control flow: if/else, switch, for loops, and while loops.
- Functions, parameters, return values, and scope.
- Arrays, vectors, basic references, basic pointers, and recursion.

What to Avoid Early On

Do not over-invest in GUI programming, web development, OOP-heavy application design, frameworks, databases, or software architecture. These are useful for general software engineering but are entirely outside the focus of the IOI syllabus.

Level 1 - Problem Solving & Complexity

This is the point where competitive programming begins. Students must learn to move through the cycle: Understand, Plan, Implement, Test, Optimize.

Algorithmic Thinking

- Breaking a problem into subproblems and identifying constraints.
- Finding patterns and constructing a brute-force solution.
- Looking for bottlenecks, improving them, and testing edge cases.

Time & Space Complexity (Priority: Maximum)

This is non-negotiable. Master O(1), O(log n), O(n), O(n log n), O(n^2), O(n^3), and exponential complexity. Understand time vs. memory trade-offs, input constraints, and upper bounds. The golden rule is to ask: Given the constraints, what complexity is even possible? That question alone separates beginners from competitors.

Level 2 - Basic Data Structures

Arrays, Vectors & Strings

- Traversal, searching, updating, sorting, reversal, and rotation.
- Prefix sums, frequency arrays, histograms, difference arrays, and multidimensional arrays.
- String traversal, comparison, basic substring handling, and frequency counting.

Important correction: Do NOT jump into advanced string algorithms early. The current IOI syllabus explicitly excludes KMP, Rabin-Karp, suffix arrays, and Aho-Corasick.

Stacks, Queues & Heaps

- Stack, Queue, Deque, and monotonic-stack ideas.
- Priority Queue / Binary Heap: Min/max heap, push, pop, top, and heap sort concepts.
- Disjoint Set Union (DSU): Union-Find, path compression, union by rank, and offline connectivity.

Levels 3, 4 & 5 – Searching, Recursion & Greedy

Searching & Sorting

- Linear search, binary search, lower/upper bound, and binary search on the answer.
- Quicksort concept, merge sort, heap sort, and comparator-based sorting.

Recursion, Brute Force & Backtracking

- Recursive functions, base cases, recursion trees, and divide-and-conquer.
- Enumerating possibilities, generating subsets/permutations, and constraint pruning.
- Branch-and-bound and recursive search.

Greedy Algorithms (Priority: Maximum)

Learn the greedy choice, exchange arguments, sorting-based greedy, interval scheduling, and activity selection. The most important skill is to prove why the greedy choice is safe. Olympiad programming requires the proof idea, not just the intuition that it seems optimal.

Level 6 - Prefix & Range Techniques

These techniques are simple but ridiculously useful for optimizing brute-force approaches.

- Prefix Sums: 1D/2D prefix sums, range queries, and prefix frequencies.
- Difference Arrays: Range updates, reconstruction, and difference techniques.
- Two Pointers / Sliding Window: Scans, window maintenance, monotonic properties, and frequency constraints.
- Coordinate Compression: Ranking values and mapping arbitrary coordinates to compact indices.

Level 7 – Graph Theory

Graph theory is one of the central pillars of the IOI. Students must master representation (adjacency matrix/list, edge list) and traversal (DFS, BFS, multi-source, grid).

Trees, DAGs & Shortest Paths (Priority: Maximum)

- Trees: Rooted trees, subtrees, depth, height, diameter, binary trees, and traversals.
- DAGs: Directed acyclic graphs, topological sorting, and DAG dynamic programming.
- Shortest Paths: BFS for unweighted, Dijkstra, Bellman-Ford, and Floyd-Warshall.

MST, Connectivity & Flow

- Minimum Spanning Trees: Kruskal, Prim, and DSU connections.
- Connectivity: Connected components, bridges, articulation points, and strongly connected components.
- Matching & Flow: Bipartite matching, augmenting paths, maximum flow, and minimum cut (National/International tier).

Level 8 - Dynamic Programming

Dynamic Programming is arguably the single most important advanced algorithmic topic. Master the five questions: What is the state? What does it mean? What are the transitions? What is the base case? What is the answer?

DP Variants

- Basic DP: 1D, 2D, sequence, grid, counting, and optimization.
- Knapsack Family: 0/1 knapsack, unbounded knapsack, subset-sum, and partition.
- String / Sequence DP: Longest common subsequence, edit distance, LIS, and interval DP.
- Tree DP: DP on rooted trees, subtree states, independent sets, and rerooting concepts.
- Bitmask DP: Subset states, small-n exponential DP, and traveling-salesperson modeling.

DP Optimization (National/International)

Divide-and-conquer DP, convex-hull trick, and Knuth optimization. The IOI emphasizes understanding why the optimization applies rather than mechanically applying it.

Level 9 - Advanced Data Structures

- Fenwick Tree: Point update, prefix query, and range-related transformations.
- Segment Tree: Range query, point/range update, lazy propagation, and min/max/sum/GCD applications.
- Balanced Search Trees: Ordered sets/maps conceptually and augmented balanced trees.
- Lowest Common Ancestor (LCA): Binary lifting and tree ancestor relationships.
- Tree Decomposition: Heavy-light decomposition and centroid decomposition.
- Persistent Data Structures: Persistent segment trees, versioned structures, and historical queries.

Levels 10, 11 & 12 - Math, Geometry & Strings

Number Theory for Informatics

- Integer Algorithms: GCD, Euclidean algorithm, LCM, fast exponentiation, prime testing, and Sieve of Eratosthenes.
- Modular Arithmetic: Modulo, addition/multiplication, exponentiation, and basic combinatorics.
- Note: The IOI syllabus specifically marks modular division/inverse elements as excluded prerequisite knowledge.

Computational Geometry

- Basics: Points, lines, segments, vectors, Euclidean distance, and polygon area.
- Predicates: Collinearity, cross/dot product, orientation, and line intersection.
- Advanced: Convex hull (monotonic chain) and sweep line algorithms.
- Note: The IOI prefers formulations that avoid numerical precision issues, favoring exact integer arithmetic.

String Algorithms

Prioritize basic string processing, character counting, prefix/suffix reasoning, naive search, and string DP. Do not spend months on KMP, suffix arrays, or suffix automata, as these are explicitly excluded from the IOI syllabus.

Levels 13, 14 & 15 – Techniques, Correctness & Specialization

Problem-Solving Techniques

- Invariants: Values that never change, parity, and conservation properties.
- Modeling: Turning a messy statement into an array, graph, tree, DP, or flow network.
- Offline vs. Online Processing: Sorting queries and deferred processing.
- State Compression: Bitmasks and compact state representations.

Correctness & Debugging

Master preconditions, postconditions, loop invariants, and proof by cases. Develop a systematic debugging process: find the smallest counterexample, inspect the assumption, isolate the bug, fix, and retest. Watch for off-by-one errors, overflow, wrong initialization, and infinite loops.

National/International Specialization

Combine techniques: Graph + DP, Tree + Data Structure, Greedy + Proof, Binary Search + Feasibility Test, DP + Optimization, Geometry + Sweep Line, DSU + Offline Queries, and Flow + Graph Modeling.

The Four Olympiad Levels

District Level - Can you code?

Focus on C++ basics, I/O, conditions, loops, arrays, and strings. Master basic implementation, simple simulation, brute force, sorting, searching, arithmetic, and prefix sums. Goal: Become fast and reliable at easy programming problems.

Province Level - Can you recognize standard algorithms?

Add Stacks, Queues, DSU, Two Pointers, Binary Search on Answer, DFS/BFS, Connected Components, Basic Shortest Path, and Basic DP. Start asking: What is the intended complexity? rather than Can I brute force this?

National / Pre-Finals - Can you invent and combine algorithms?

Focus on Advanced DP, Graph Algorithms (MST, SCC, Bridges), Fenwick/Segment Trees, LCA, Tree Decomposition, Greedy Proofs, Invariants, and Modeling. Goal: Solve difficult problems under contest pressure.

International (IOI) - Can you solve unfamiliar problems efficiently?

Combine advanced algorithms, data structures, mathematical reasoning, and implementation. Master DP optimization, rerooting, heavy-light decomposition, persistent structures, max flow, and amortized analysis. The IOI features six tasks over two days with substantial partial scoring.

Final Priority Map

S-Tier - Master First (The Foundation)

- Programming: C++, Arrays, Strings, Functions, Recursion.
- Algorithms: Complexity, Sorting, Binary Search, Greedy, DFS, BFS, Dynamic Programming.
- Data Structures: Arrays, Stack, Queue, Priority Queue, DSU.
- Graphs: Trees, Shortest Paths, MST, Connectivity.

A-Tier - Very Important

- Fenwick/Segment Trees, LCA, Binary Lifting, SCC, Bridges, DAGs, Tree/Bitmask DP.
- Coordinate Compression, Two Pointers, Offline Algorithms, Max Flow, Bipartite Matching, Computational Geometry.

B-Tier - Advanced National/IOI

- Heavy-Light/Centroid Decomposition, Persistent Data Structures, Advanced DP Optimization, Amortized Analysis.

Exclusions - Do Not Waste Early Preparation On

The official IOI syllabus explicitly excludes or deprioritizes: Calculus, Linear Algebra, FFT, Complex Numbers, Cryptography, Machine Learning, KMP, Aho-Corasick, Suffix Arrays/Trees, and General Graph Matching. These are bad priorities for someone targeting the IOI.

The Clean Master Checklist

Domain	Core Topics
Programming	C++, variables, I/O, loops, functions, arrays, strings, recursion
Complexity	Big O, time/space complexity, constraints analysis, invariants
Basic DS	Arrays, strings, stack, queue, deque, priority queue, DSU
Basic Algos	Brute force, sorting, binary search, prefix sums, two pointers, greedy
Graphs	DFS/BFS, trees, DAGs, shortest paths, MST, SCC, bridges, matching, flow
Dynamic Prog.	1D/2D/Grid DP, knapsack, LIS/LCS, tree DP, bitmask DP, optimization
Advanced DS	Fenwick, segment tree, lazy propagation, LCA, HLD, persistent structures
Math/ Geometry	GCD, primes, modular arithmetic, combinatorics, cross product, convex hull
Techniques	Modeling, invariants, greedy proofs, offline processing, state compression

The Actual Progression & Conclusion

For other Olympiads, the progression is mostly content depth. For Informatics, the ladder is fundamentally different:

- District: Can you code?
- Province: Can you recognize standard algorithms?
- National: Can you invent and combine algorithms?
- IOI: Can you solve a genuinely unfamiliar computational problem efficiently under severe time constraints?

Do not confuse being good at general programming (React, Next.js, Python, AI frameworks, Web Dev) with being good at the IOI. The cleanest possible target is: C++ -> Complexity -> Basic Algorithms -> Data Structures -> Graphs -> Greedy -> DP -> Advanced Data Structures -> Advanced Graphs -> Geometry -> Contest Problem Solving. By systematically mastering this roadmap, a dedicated NEB student can develop the algorithmic maturity required to excel on the global stage.

References and Recommended Literature

[1] International Olympiad in Informatics (IOI). Official IOI 2025 Syllabus and Getting Started Guidelines. IOI Secretariat.

[2] NPLCoder. Nepal Olympiad in Informatics (NOI) Structure and NPLAlgo Initiative. NPLCoder, 2025.

[3] Halim, S., & Halim, F. Competitive Programming 4. Lulu Press, 2020.

[4] Laaksonen, A. Competitive Programmer's Handbook. 2018.

[5] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. Introduction to Algorithms. MIT Press, 2009.

[6] Ministry of Education, Curriculum Development Centre (CDC). Secondary Level Computer Science Curriculum. Government of Nepal, 2078 (2021).