IOI Syllabus 2023 vs CS Curriculum (9-12) 2009

IOI Syllabus 2023	CS Curriculum (9-12) 2009 (Covered topics)
6.1 Programming Fundamentals (PF) **PF1. Fundamental programming constructs**(for abstract machines) ✓ Basic syntax and semantics of a higher-level language (at least one of the specific languages available at an IOI, as announced in the *Competition Rules* for that IOI) ✓ Variables, types, expressions, and assignment ✓ Simple I/O ✓ Conditional and iterative control structures ✓ Functions and parameter passing ✓ Structured decomposition	**All topics are included except last one** **(✓ Structured decomposition)** **C and C++ programming languages are used, which are also recommended for IOI.**
PF2. Algorithms and problem-solving ✓ Problem-solving strategies (understand–plan–do–check, separation of concerns, generalization, specialization, case distinction, working backwards, etc.) ✓ The role of algorithms in the problem-solving process ✓ Implementation strategies for algorithms (also see 7 SE1) ✓ Debugging strategies (also see 7 SE3) ✓ The concept and properties of algorithms (correctness, efficiency)	**Basic level topics are included, covers about 20% weightage of IOI syllabus.**
PF3. Fundamental data structures ✓ Primitive types (boolean, signed/unsigned integer, character) ✓ Arrays (incl. multicolumn dimensional arrays) ✓ Strings and string processing ✓ Static and stack allocation (elementary automatic memory management) ✓ Linked structures ✓ Implementation strategies for graphs and trees ✓ Strategies for choosing the right data structure ✓ Elementary use of real numbers in numerically stable tasks ✓ The floating-point representation of real numbers, the existence of precision issues.[11] ✓ Pointers and references Data representation in memory,Heap allocation, Runtime storage management, ? Using fractions to perform exact calculations. ? ✗ Non-trivial calculations on floating point numbers, manipulating precision errors Regarding floating point numbers, there are well-known reasonswhy they should be, in general, avoided at the IOI.[12] However, the currently used interface removes some of those issues. In particular, it should now be safe to use floating point numbers insome types of tasks – e.g., to compute some Euclidean distancesand return the smallest one.	**Only Arrays and basic string processing topics are included, which covers about 10% syllabus of IOI of this section.**
PF4. Recursion ✓ The concept of recursion ✓ Recursive mathematical functions ✓ Simple recursive procedures (incl. mutual recursion)	**1%**

✓ Divide-and-conquer strategies ✓ Implementation of recursion ✓ Recursive backtracking	
PF5. Event-driven programming Some competition tasks may involve a dialog with a reactive environment. Implementing such an interaction with the provided environment is ✓. Everything not directly related to the implementation of reactivetasks is ?	**Not Included**
6.1 Algorithms and Complexity (AL) **AL1. Basic algorithmic analysis** ✓ Algorithm specification, precondition, post condition, correctness, invariants ✓ Asymptotic analysis of upper complexity bounds (informally if possible) ✓ Big O notation ✓ Standard complexity classes: constant, logarithmic, linear, $O(n \log n)$, quadratic, cubic, exponential, etc. ✓ Time and space tradeoffs in algorithms ✓ Empirical performance measurements. Identifying differences among best, average, and worst casebehaviors, ? Little o, Omega, and Theta notation, ? Tuning parameters to reduce running time, memory consumption or other measures of performance ? ✗ Asymptotic analysis of average complexity bounds ✗ Using recurrence relations to analyze recursive algorithms	**Basic level topics related to Algorithms (Algorithmic analysis) are included, which covers about 5% syllabus of IOI of this section.**
AL2. Algorithmic strategies ✓ Simple loop design strategies ✓ Brute-force algorithms (exhaustive search) ✓ Greedy algorithms ✓ Divide-and-conquer ✓ Backtracking (recursive and non-recursive), Branch-and- bound ✓ Dynamic programming Heuristics ? Finding good features for machine learning tasks[14] ? Discrete approximation algorithms Randomized algorithms. ? ✗ Clustering algorithms (e.g. k-means, k-nearest neighbor) ✗ Minimizing multi-variate functions using numerical approaches.	**2%**
AL3a. Algorithms ✓ Simple algorithms involving integers: radix conversion, Euclid's algorithm, primality test by O(√n) trial division, Sieve of Eratosthenes, factorization (by trial division or a sieve), efficient exponentiation	_

✓ Simple operations on arbitrary precision integers (addition, subtraction, simple multiplication)15 ✓ Simple array manipulation (filling, shifting, rotating, reversal, resizing, minimum/maximum, prefix sums, histogram, bucket sort) ✓ Simple string algorithms (e.g., naive substring search) ✓ sequential processing/search and binary search ✓ Quicksort and Quickselect to find the k-th smallest element. ✓ O(n log n) worst-case sorting algorithms (heap sort, merge sort) ✓ Traversals of ordered trees (pre-, in-, and post-order) ✓ Depth- and breadth-first traversals ✓ Applications of the depth-first traversal tree, such as topo- logical ordering and Euler paths/cycles ✓ Finding connected components and transitive closures. ✓ Shortest-path algorithms (Dijkstra, Bellman-Ford, Floyd Warshall) ✓ Minimum spanning tree (Jarn´ık-Prim and Kruskal algo- rithms) ✓ O(V E) time algorithm for computing maximum bipartite matching. ✓ Biconnectivity in undirected graphs (bridges, articulation points). ✓ Connectivity in directed graphs (strongly connected com- ponents). ✓ Basics of combinatorial game theory, winning and losing positions, minimax algorithm for optimal game playing ✗ Maximum flow. Flow/cut duality theorem. ✗ Optimization problems that are easiest to analyze using matroid theory. Problems based on matroid intersecions (except for bipartite matching). ✗ Lexicographical BFS, maximum adjacency search and their properties ✗ Fat nodes and other more complicated ways of implement- ing persistent data structures.	**Simple array manipulation is included for searching and sorting (Bubble sort), which covers about 0.5 % syllabus of IOI.**
AL3b. Data structures ✓ Stacks and queues ✓ Representations of graphs (adjacency lists, adjacency ma- trix) ✓ Binary heap data structures ✓ Representation of disjoint sets: the Union-Find data struc- ture. ✓ Statically balanced binary search trees. Instances of this in- clude binary index trees (also known as Fenwick trees) and segment trees (also known as interval trees and tournament trees).16 ✓ Balanced binary search trees17 ✓ Augmented binary search trees ✓ O(log n) time algorithms for answering lowest common an- cestor queries in a static rooted tree.18 ✓ Decomposition of static trees (heavy-light decomposition, separator structures such as centroid decomposition) ✓ Creating persistent data structures by path copying. ✓ Nesting of data structures, such as having a sequence of sets. ✓ Tries ✗ Data structures for dynamically changing trees and their use in graph algorithms. ✗ String algorithms and data structures (KMP, Rabin-Karp hashing, suffix arrays/trees, suffix automata, Aho-Corasick) ✗ Complex heap variants such as binomial and Fibonacci heaps, ✗ Using and implementing hash tables (incl. strategies to resolve collisions)	**Not Included**

✗ Two-dimensional tree-like data structures (such as a 2D statically balanced binary tree or a treap of treaps) used for 2D queries. ✗ Fat nodes and other more complicated ways of implement- ing persistent data structures.	
AL4. Distributed algorithms This entire section is . ?	**Not Included**
AL5. Basic computability All topics related to computability are ✗. This includes the following: Tractable and intractable problems; Unnomputable functions; The halting problem; Implications of uncomputability. However, see AL7 for basic computational models.	**Not Included**
AL6. The complexity classes P and NP Topics related to non-determinism, proofs of NP-hardness (reductions), and everything related is ✗. Note that this section only covers the results usually contained in undergraduate and graduate courses on formal languages and computational complexity. The classification of these topics as ✗ does not mean that an NP-hard problem cannot appear at an IOI.	**Not Included**
AL7. Automata and grammars ✓ Understanding a simple grammar in Backus-Naur form Formal definition and properties of finite-state machines, Context-free grammars and related rewriting systems, Regular expressions ? ✗ Properties other than the fact that automata are graphs and that grammars have parse trees. ?	**Not Included**
AL8. Advanced algorithmic analysis ✓ Amortized analysis. Online algorithms ? Randomized algorithms ? ✗ Alpha-beta pruning	**Not Included**
AL9. Cryptographic algorithms This entire section is . ?	**Not Included**
AL10. Geometric algorithms In general, the ISC has a strong preference towards problems that can be solved using integer arithmetic to avoid precision issues. This may include representing some computed values as exact fractions, but extensive use of such fractions in calculations is discouraged. Additionally, if a problem uses two-dimensional objects, the ISC prefers problems in which such objects are rectilinear. ✓ Representing points, vectors, lines, line segments. ✓ Checking for collinear points, parallel/orthogonal vectors and clockwise turns (for example, by using dot products and cross products).	**Not Included**

✓ Intersection of two lines. ✓ Computing the area of a polygon from the coordinates of its vertices.19 ✓ Checking whether a (general/convex) polygon contains a point. ✓ Coordinate compression. O(n log n) time algorithms for convex hull ✓ Sweeping line method ✗ Point-line duality ✗ Halfspace intersection, Voronoi diagrams, Delaunay triangulations. ✗ Computing coordinates of circle intersections against lines and circles. ✗ Linear programming in 3 or more dimensions and its geo- metric interpretations. ✗ Center of mass of a 2D object. ✗ Computing and representing the composition of geometric transformations if the knowledge of linear algebra gives an advantage.	
AL11. Parallel algorithms This entire section is . **?**	**Not Included**
6.2 Other Areas in Computing Science Except for GV (specified below), all areas are ✗. AR. Architecture and Organization OS. Operating Systems **NC. Net-Centric Computing** (a.k.a. cloud computing) PL. Programming Languages **HC. Human-Computer Interaction GV. Graphics and Visual Computing** Basic aspects of processing graphical data are , everything else (including the use of graphics libraries such as OpenGL) is ✗. IS. Intelligent Systems **IM. Information Management** SP. Social and Professional Issues CN. Computational Science Notes: AR is about digital systems, assembly language, instruction pipelining, cache memories, etc. OS is about the *design* of operating systems, not their usage. PL is about the *analysis and design* of programming languages, not their usage. HC is about the *design* of user interfaces. *Usage* of the operating system, GUIs and programming languages is covered in 8 and 6.1.	**About 10-15% IOI syllabus of this section is covered.** **Only usage of OS and Programming Language is included, design part is NOT included (which is the requirement of IOI).**
7 Software Engineering (SE) We quote from the IEEE-CS Curriculum: Software engineering is the discipline concerned with the application of theory, knowledge, and practice for effectively and efficiently building software systems that satisfy the requirements of users and customers.	

In the IOI competition, the application of software engineering concerns the use of light-weight techniques for small, one-off, single-developer projects under time pressure. All included top- ics are ✓. **SE1. Software design** ✓ Fundamental design concepts and principles ✓ Design patterns ✓ Structured design In particular, contestants may be expected to – Transform an abstract algorithm into a concrete, efficient program expressed in one of the allowed programming languages, possibly using standard or competition-specific libraries. – Make their programs read data from and write data to text files according to a prescribed simple format ✗ Software architecture, ✗ Design for reuse, ✗ Object-Oriented analysis and design, ✗ Component-level design	**Only basic level software design concepts are included. About 10-15% syllabus of IOI is covered.**
SE2. Using APIs ✓ API (Application Programming Interface) programmingIn particular, contestants may be expected to - Use competition-specific libraries according to the provided specification. ✗ Programming by example, ✗ Debugging in the API environment, ✗ Class browsers and related tools, ✗ Introduction to component-based computing	**Not Included**
SE3. Software tools and environments ✓ Programming environments, incl. IDE (Integrated Development Environment) In particular, contestants may be expected to – Write and edit program texts using one of the provided program editors. – Compile and execute their own programs. – Debug their own programs. ✗ Testing tools, ✗ Configuration management tools ✗ Requirements analysis and design modeling tools, ✗ Tool integration mechanisms	**Only basic level program editing, compiling and debugging is included. About 10-15% syllabus of IOI is covered.**
SE4. Software processes ✓ Software life-cycle and process models In particular, contestants may be expected to – Understand the various phases in the solution development process and select appropriate approaches. ✗ Process assessment models, ✗ Software process metrics	**Only theoretical concepts of Software life-cycle are included, no practical knowledge is given, which covers about 10-15 % syllabus of IOI.**
SE5. Software requirements and specification	

✓ Functional and nonfunctional requirements ✓ Basic concepts of formal specification techniquesIn particular, contestants may be expected to – Transform a precise natural-language description (with or without mathematical formalism) into a problem in terms of a computational model, including an understanding of the efficiency requirements. ✗ Prototyping, ✗ Requirements elicitation, ✗ Requirements analysis modeling techniques	**Only theoretical concepts are included, no practical knowledge is given, which covers about 10-15 % syllabus of IOI.**
SE6. Software validation ✓ Testing fundamentals, including test plan creation and testcase generation ✓ Black-box and white-box testing techniques ✓ Unit, integration, validation, and system testing ✓ Inspections In particular, contestants may be expected to – Apply techniques that maximize the opportunity to detect common errors (e.g. through well-structured code, code review, built-in tests, test execution). – Test (parts of) their own programs. ✗ Validation planning, ✗ Object-oriented testing	**Only theoretical concepts are included, no practical knowledge is given, which covers about 5-10 % syllabus of IOI.**
SE7. Software evolution ✗ Software maintenance, ✗ Characteristics of maintainable software, ✗ Re-engineering, ✗ Legacy systems, ✗ Software reuse	**Not Included**
SE8. Software project management ✓ Project scheduling (especially time management) ✓ Risk analysis ✓ Software configuration management In particular, contestants may be expected to – Manage time spent on various activities. – Weigh risks when choosing between alternative approaches. – Keep track of various versions and their status while developing solutions. ✗ Software quality assurance, ✗ Team management, ✗ Software measurement and estimation techniques, ✗ Project management tools	**Not Included**
SE9. Component-based computing This entire section is ✗.	**Not Included**
SE10. Formal methods ✓ Formal methods concepts (notion of correctness proof, in-variant) ✓ Pre and post assertions In particular, contestants may be expected to	

– Reason about the correctness and efficiency of algorithmsand programs. ✗ Formal verification, ✗ Formal specification languages, ✗ Executable and non-executable specifications	
SE11. Software reliability This entire section is ✗.	**Not Included**
SE12. Specialized systems development This entire section is ✗.	**Not Included**
8 Computer Literacy The text of this section is ✓.	
Contestants should know and understand the basic structure and operation of a computer (CPU, memory, I/O). They are expected to be able to use a standard computer with graphical user interface, its operating system with supporting applications, and the provided program development tools for the purpose ofsolving the competition tasks. In particular, some skill in file management is helpful (creating folders, copying and moving files). Details of these facilities will be stated in the *Competition Rules* of the particular IOI. Typically, some services are available througha standard web browser. Possibly, some competition-specific tools are made available, with separate documentation. It is often the case that a number of equivalent tools are made available. The contestants are not expected to know all the features of all these tools. They can make their own choice based on what they find most appropriate. The following topics are all: Calculator, Word-processors, Spread-sheet applications, Database management systems, E-mail clients, Graphics tools (drawing, painting).	**This section is fully covered.**

Submitted by:

1. **Mohammad Khalid**, Assistant Professor and Head of CS dept. OPF Boys College, Islamabad.
2. **Ms. Rozina Faheem**, Professor/Principal, FGHE&MS F-11/1, Islamabad