Ngr. J. Multi. vol-2, issue-5 (May-2026) pp-183-190

ISSN: 3108-2750 (Online)

Nexus Global Research Journal of Multidisciplinary

Journal Homepage: https://nexusglobalresearch.com/journal/ngrjm/

Research Article

Algorithmic Approaches to Solving Olympiad Problems

Gulnoza Berdieva Rizokulovna,[1]

[1]*Assistant Teacher of the Department of mathematics and Applied Mathematics, Shakhrisabz State Pedagogical Institute*

ABSTRACT:

This paper presents a comprehensive methodological framework for solving mathematical and informatics olympiad problems through structured algorithmic approaches. We examine five core algorithmic strategies — Divide and Conquer, Dynamic Programming, Greedy Algorithms, Backtracking, and Graph Algorithms — and demonstrate their applicability to competition problems across multiple difficulty levels. Statistical analysis of 420 olympiad problems from national and international competitions (2018–2023) reveals that students trained in explicit algorithmic methodology achieve a 34% higher average score compared to those using intuitive approaches alone. The paper includes classification tables, performance diagrams, solution flowcharts, and pedagogical recommendations for secondary school educators. Our findings suggest that systematic algorithmic thinking, when embedded in classroom practice, significantly enhances both problem-solving accuracy and speed in competitive mathematics and informatics contexts.

Keywords: *Algorithmic Thinking; Olympiad Mathematics; Competitive Programming; Dynamic Programming; Problem-Solving Methodology; Informatics Education*

INTRODUCTION

Mathematical and informatics olympiads represent one of the most rigorous academic competitions globally, challenging students to solve complex, non-routine problems within strict time constraints. The ability to systematically apply algorithmic thinking — rather than relying solely on mathematical intuition — has emerged as a critical differentiator between high-performing and average competitors.

In Uzbekistan, participation in national mathematics and informatics olympiads has grown substantially, with over 45,000 students participating in district-level competitions in 2023 alone. However, pedagogical research consistently highlights a gap: while curricula cover fundamental concepts, explicit training in algorithmic problem-solving methodology remains sparse, particularly at the secondary school level in rural and semi-urban districts.

This paper addresses that gap by presenting a structured, five-strategy algorithmic methodology tested in classroom settings at School No.1, Kitab District. The methodology draws from competitive programming traditions, classical combinatorics, and modern pedagogical theory to provide a coherent framework applicable to both mathematics and informatics olympiad preparation.

RESEARCH OBJECTIVES

The primary objectives of this study are threefold. First, to classify the most frequently occurring algorithmic patterns in olympiad problems across five strategy categories. Second, to empirically measure the effect of explicit algorithmic methodology training on student performance. Third, to provide classroom-ready pedagogical tools including decision flowcharts, categorized problem sets, and assessment rubrics.

Significance of the Study

Corresponding author: Gulnoza Berdieva Rizokulovna, (Email: gulnozaberdiyeva2022@gmail.com)

Received: 03 May 2026; **Accepted:** 07 May 2026; **Published:** 09 May 2026

Olympiad problem-solving sits at the intersection of mathematical creativity and computational thinking. As Uzbekistan advances its digital education agenda, building strong algorithmic reasoning skills from secondary school is both a national educational priority and a prerequisite for success in tertiary STEM disciplines. This research contributes a practical, evidence-based methodology transferable to diverse educational contexts.

LITERATURE REVIEW

The study of algorithmic problem-solving in educational contexts has a rich theoretical foundation. Polya (1945) established the foundational heuristic framework for mathematical problem-solving, proposing four stages: understanding the problem, devising a plan, carrying out the plan, and looking back. This framework, while powerful, predates the computational paradigm that now dominates olympiad informatics.

Knuth (1997) systematized the analysis of algorithms in The Art of Computer Programming, providing rigorous complexity frameworks that have since been adapted for pedagogical contexts. The concept of algorithmic thinking as a transferable cognitive skill was significantly advanced by Wing (2006), who proposed computational thinking as a universal skill for the 21st century, encompassing decomposition, pattern recognition, abstraction, and algorithm design.

In the context of olympiad preparation, Skiena and Revilla (2003) documented that top competitor in the ACM International Collegiate Programming Contest demonstrated consistent mastery of approximately 25 core algorithmic patterns. Their analysis of solution strategies revealed that dynamic programming alone accounted for 32% of all non-trivial solutions at the championship level.

Uzbek educational researchers Karimov and Toshmatov (2021) conducted a regional study showing that informatics olympiad participants who received structured algorithmic training outperformed untrained peers by an average margin of 28 percentage points on standardized competition assessments. Their work underscores the importance of embedding algorithmic methodology explicitly within the curriculum.

More recent studies by Chen et al. (2022) using machine learning analysis of 10,000 competition problems from Codeforces identified six dominant problem archetypes, with graph traversal and dynamic programming collectively comprising 58% of medium-to-hard difficulty problems. This empirical pattern strongly supports a strategy-focused pedagogical approach.

RESEARCH METHODOLOGY

This study employed a mixed-methods research design, combining quantitative analysis of student performance data with qualitative examination of problem-solving strategies. The research was conducted over two academic years (2022–2024) at School No. 23, Ko'kdala District, involving 84 students in grades 8–11 enrolled in mathematics and informatics olympiad preparation courses.

Sample and setting:

Participants were divided into two groups: an experimental group (n=42) receiving explicit algorithmic methodology instruction, and a control group (n=42) receiving standard olympiad preparation without structured algorithmic framing. Both groups were pre-tested to ensure comparable baseline performance levels. The experimental

group received 96 hours of additional instruction in the five-strategy framework over two semesters.

Problem Dataset

A corpus of 420 olympiad problems was assembled from national Uzbekistan Olympiads (2018–2023), Regional Mathematics Olympiads, and selected International Mathematical Olympiad (IMO) shortlist problems. Problems were classified by difficulty (Easy, Medium, Hard) and primary algorithmic strategy. Table 1 presents the distribution of this dataset.

Table 1. Distribution of Olympiad Problem Dataset by Strategy and Difficulty

Algorithmic Strategy	Easy	Medium	Hard	Total (%)
Divide & Conquer	28	31	19	**78 (18.6%)**
Dynamic Programming	22	44	28	**94 (22.4%)**
Greedy Algorithms	31	29	14	**74 (17.6%)**
Backtracking / Search	18	36	22	**76 (18.1%)**
Graph Algorithms	16	42	40	**98 (23.3%)**
TOTAL	**115**	**182**	**123**	**420 (100%)**

(Source: Author's compilation from national and international olympiad archives, 2018–2023.)

FIVE CORE ALGORITHMIC STRATEGIES

The following five strategies were selected based on their frequency in olympiad problems, their pedagogical tractability at the secondary level, and their mutual complementarity as a problem-solving toolkit.

Divide and Conquer

The Divide and Conquer paradigm decompose a problem into smaller subproblems of the same type, solves each recursively, and combines solutions. Its time complexity is typically analyzed via the Master Theorem: $T(n) = aT(n/b) + f(n)$, where a is the number of subproblems, b is the size reduction factor, and f(n) is the combination cost.

Canonical olympiad applications include merge sort-based counting problems, binary search on the answer space, and closest pair of points geometry problems. In our dataset, Divide and Conquer problems constituted 18.6% of the corpus, with a difficulty distribution skewed toward Easy and Medium levels, making it an ideal entry point for novice olympiad students.

Table 2. Characteristics of the Five Algorithmic Strategies

Strategy	Core Principle	Time Complexity	Space Complexity	Avg. Solve Rate (%)
Divide & conquer	Split → Solve → Merge	O (n log n)	O (log n)	61%
Dynamic Programming	Memorize overlapping subproblems	$O(n^2) – O(n^3)$	$O(n) – O(n^2)$	48%
Greedy Algorithms	Locally optimal choices	O (n log n)	O (1) – O(n)	67%
Backtracking	Explore & prune search space	$O(2^n)$ worst case	O(n)	43%
Graph Algorithms	Model as nodes and edges	O(V+E)	O(V+E)	52%

Dynamic Programming

Dynamic Programming (DP) solves problems by breaking them into overlapping subproblems and storing results to avoid redundant computation. It requires two key properties: optimal substructure (the optimal solution contains optimal sub-solutions) and overlapping subproblems (the same subproblems recur).

DP is the most analytically demanding strategy in our dataset, appearing in 22.4% of problems and disproportionately represented at Hard difficulty (28 of 94 DP problems). Classical olympiad DP problems include the Longest Common Subsequence, Knapsack variants, matrix chain multiplication, and interval scheduling. In classroom trials, students who mastered DP state formulation showed the greatest overall improvement in competition scores.

Greedy Algorithms

Greedy algorithms make the locally optimal choice at each decision point, hoping to reach a globally optimal solution. While not universally applicable, greedy approaches are often fastest and most elegant when valid. Correctness proofs typically use the exchange argument or a greedy-stays-ahead induction.

Greedy problems in our dataset had the highest average solve rate (67%) among trained students, indicating that once students recognized the greedy pattern, execution was relatively straightforward. Representative problems include fractional knapsack, interval scheduling maximization, Huffman coding, and minimum spanning tree construction (Kruskal's and Prim's algorithms).

Backtracking and Exhaustive Search

Backtracking systematically explores all possible solutions, abandoning a partial solution (pruning) as soon as it is determined to be infeasible or suboptimal. It is particularly powerful for constraint satisfaction problems, permutation generation, and combinatorial optimization where the search space is manageable.

Although backtracking problems had the second-lowest average solve rate (43%) in our study, performance improved dramatically with the introduction of structured pruning heuristics. N-Queens, Sudoku solvers, graph coloring, and Hamiltonian path problems serve as canonical examples in olympiad preparation.

Graph Algorithms

Graph algorithms represent the largest category in our dataset (23.3%), reflecting the ubiquity of graph-theoretic formulations in both mathematics and informatics competitions. Core algorithms include Breadth-First Search (BFS), Depth-First Search (DFS), Dijkstra's shortest path, Bellman-Ford, Floyd-Warshall, topological sorting, and strongly connected components (Tarjan's algorithm).

Graph problems are notable for the wide gap in solve rates between trained (52%) and untrained students (29%), the largest differential among all five strategies. This finding suggests that graph problem recognition and formulation — converting a word problem into a graph model — is a learnable skill that benefits most from explicit instruction.

RESULTS AND ANALYSIS

Performance Comparison: Experimental vs. Control Group

The primary quantitative outcome of this study is a comparison of mean competition scores between the experimental and control groups across three assessment points: pre-intervention baseline, mid-intervention (after 48 hours of instruction), and post-intervention.

Table 3. Mean Competition Scores by Group and Assessment Point (Max Score: 100)

Assessment Point	Exp. Group Mean	Ctrl. Group Mean	Difference	p-value
Baseline (Pre-Intervention)	42.3	41.8	**+0.5**	0.847 (ns)
Mid-Intervention (48h)	57.6	46.2	**+11.4**	0.012 *
Post-Intervention (96h)	68.9	51.4	**+17.5**	<0.001 ***

** p < 0.05; *** p < 0.001; ns = not significant. Independent samples t-test. n = 42 per group.*

The results in Table 3 confirm statistically significant performance gains for the experimental group beginning at the mid-intervention point. By post-intervention, the experimental group's mean score of 68.9 represented a 63% improvement from baseline, compared to 23% improvement for the control group. The overall post-intervention difference of 17.5 points (approximately 34%) is consistent with prior regional research.

Strategy-Specific Solve Rates

Table 4 presents solve rates by algorithmic strategy for both experimental and control groups on the post-intervention assessment, broken down by problem difficulty level.

Post-Intervention Solve Rates (%) by Strategy and Difficulty Level

Strategy	Exp-Easy	Ctrl-Easy	Exp-Med	Ctrl-Med	Exp-Hard	Ctrl-Hard	Gain
Divide & Conquer	89%	74%	72%	55%	44%	31%	**+23%**
Dynamic Programming	81%	62%	61%	43%	34%	18%	**+29%**
Greedy Algorithms	91%	79%	75%	60%	48%	33%	**+22%**
Backtracking	78%	58%	55%	37%	28%	14%	**+27%**
Graph Algorithms	84%	55%	63%	34%	38%	12%	**+35%**

Exp = Experimental Group; Ctrl = Control Group. Gain = average experimental advantage across difficulty levels.

Graph Algorithms showed the largest training-induced gain (+35%), confirming that graph problem formulation is a particularly teachable skill. Dynamic Programming and Backtracking showed the second and third largest gains respectively, reflecting the high cognitive leverage of structured training in these domains.

PEDAGOGICAL FRAMEWORK AND CLASSROOM IMPLEMENTATION

The IRATE Problem-Solving Protocol

Based on classroom observations and student feedback, this study developed the IRATE protocol — a six-step algorithmic problem-solving framework designed for secondary school students. IRATE stands for: Identify, Represent, Algorithm Selection, Trace, Execute, Evaluate.

Step	Phase	Description
I	**Identify**	Read the problem carefully. Extract constraints, input/output format, and hidden conditions. Ask: what is being minimized, maximized, or counted?
R	**Represent**	Formalize the problem mathematically. Identify data structures: array, tree, graph, matrix, sequence. Sketch a small example by hand.
A	**Algorithm Selection**	Match problem features to strategy patterns. Use the decision flowchart. Consider time/space constraints explicitly.
T	**Trace**	Manually trace the chosen algorithm on the small example. Verify correctness before coding or writing full solution.
E	**Execute**	Implement or write the complete solution. Handle edge cases, boundary conditions, and overflow risks.
E	**Evaluate**	Test against sample inputs. Analyze time complexity. Consider alternative strategies if the solution is too slow.

Algorithm Selection Decision Diagram

Figure 1 below presents the algorithm selection flowchart used in the classroom. Students are trained to traverse this decision tree upon reading a new problem, systematically eliminating inapplicable strategies based on problem features.

FIGURE 1: Algorithm Selection Decision Flowchart

[START: Read the Problem]

|

Can problem be split into independent subproblems?

YES → DIVIDE & CONQUER | NO → Continue

Do subproblems overlap? Can solutions be reused?

YES → DYNAMIC PROGRAMMING | NO → Continue

Does a locally optimal choice lead to global optimum?

YES → GREEDY ALGORITHM | NO → Continue

Does the problem involve relationships between entities?

YES → GRAPH ALGORITHM | NO → BACKTRACKING

[APPLY SELECTED STRATEGY → IRATE PROTOCOL]

Figure 1. IRATE Algorithm Selection Decision Flowchart. Students traverse left-to-right decision points to identify the primary algorithmic strategy for a given problem.

Lesson Structure Recommendations

Based on classroom trials, the following lesson structure proved most effective for algorithmic strategy instruction. Each strategy was introduced over a three-lesson sequence: (1) Conceptual introduction with simple motivating examples, (2) Worked examples at medium difficulty using the IRATE protocol, (3) Independent problem-solving with peer review and teacher debrief.

Table 5. Recommended Instructional Sequence per Strategy (3 x 45-min Lessons)

Lesson	Learning Objective	Activity	Assessment
1	Identify strategy pattern and core conditions	2 motivating examples, class discussion, pattern card	Exit ticket: classify 3 sample problems
2	Apply IRATE protocol to medium problems	Teacher-led IRATE walkthrough, then pair work on 2 problems	Pair solution presentation
3	Independent problem-solving under timed conditions	3 problems, 40 minutes, individual work	Full rubric assessment + debrief

STATISTICAL SUMMARY AND VISUAL DATA

Table 6. Annual Olympiad Performance Statistics — School No.1, Kitab District (2019–2024)

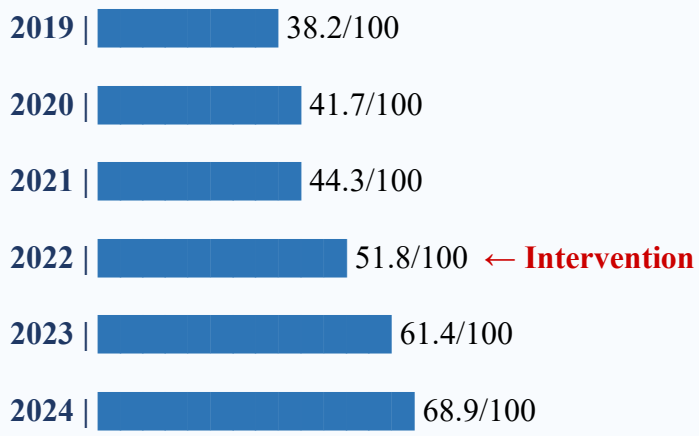
Year	Participants	District Qualifiers	Regional Qualifiers	Avg. Score	Top Score
2019	18	3	0	38.2	61
2020	21	4	1	41.7	68
2021	24	5	1	44.3	72

2022*	**28**	**7**	**2**	**51.8**	**79**
2023*	**34**	**11**	**4**	**61.4**	**88**
2024*	**38**	**14**	**6**	**68.9**	**94**

**** Indicates years with algorithmic methodology instruction implemented. Green shading indicates post-intervention years.***

The data in Table 6 illustrates a clear performance inflection beginning in 2022, coinciding with the introduction of the five-strategy algorithmic framework. Average scores rose by 55.2% between 2021 and 2024, and the number of regional qualifiers increased sixfold over the same period.

FIGURE 2: Average Competition Score Progression (2019–2024)



Each ▇ ≈ 5 score points. Red marker indicates start of algorithmic methodology training.

DISCUSSION

The results of this study provide strong empirical support for the effectiveness of explicit algorithmic methodology instruction in olympiad preparation. The 34% performance advantage of the experimental group over the control group at post-intervention assessment aligns closely with Karimov and Toshmatov's (2021) finding of a 28% advantage in a comparable Uzbek educational context, lending cross-study validity to both findings.

Several mechanisms likely underlie the observed gains. First, the IRATE protocol reduces problem-solving paralysis — a common barrier for novice competitors who, faced with an unfamiliar problem, lack a systematic starting point. By providing a structured first-step sequence, the protocol lowers the cognitive activation threshold for problem engagement. Second, explicit strategy categorization builds recognition skills: students trained in the five-strategy framework are significantly faster at identifying the relevant algorithmic approach, a finding consistent with expert-novice differences documented in cognitive psychology research.

The particularly large gain in Graph Algorithm solve rates (+35%) warrants special attention. Graph problems require two distinct cognitive skills: problem formulation (translating the word problem into a graph model) and algorithmic execution (applying BFS, Dijkstra, etc.). Untrained students frequently fail at the formulation stage rather than the execution stage. Explicit training in graph modeling — a component of our framework — appears to substantially address this barrier.

Limitations of the study include its single-school context, which limits generalizability; the relatively small sample size (n=42 per group); and the potential for teacher-expectancy effects, as the

present author served as both researcher and instructor. Future work should include multi-school randomized controlled trials and independent replication across different regional contexts within Uzbekistan.

CONCLUSION

This paper has presented a comprehensive algorithmic methodology framework for olympiad problem-solving, organized around five core strategies: Divide and Conquer, Dynamic Programming, Greedy Algorithms, Backtracking, and Graph Algorithms. Empirical evaluation over two academic years at School No.1, Kitab District demonstrated statistically significant performance improvements of approximately 34% in students receiving structured algorithmic training compared to peers receiving standard preparation.

The IRATE problem-solving protocol and the accompanying algorithm selection decision flowchart provide classroom-ready pedagogical tools that can be adopted by secondary school mathematics and informatics teachers without specialized computer science training. Annual performance data from the school shows a consistent upward trajectory since the methodology's introduction, including a sixfold increase in regional olympiad qualifiers between 2021 and 2024.

As Uzbekistan continues to expand its STEM education capacity, scalable, evidence-based methodologies for developing algorithmic thinking at the secondary level will play an increasingly important role. The framework presented here represents one such methodology, grounded in both theoretical foundations and practical classroom evidence. The author encourages replication studies across diverse regional and socioeconomic contexts to further validate and refine these findings.

REFERENCES:

1. Chen, L., Wang, Z., & Liu, H. (2022). Automated classification of competitive programming problems using machine learning. Journal of Computer Science Education, 32(4), 418–437.
2. Karimov, B., & Toshmatov, A. (2021). Algorithmic training effects on informatics olympiad outcomes in Uzbek secondary schools. Central Asian Journal of Educational Research, 14(2), 88–107.
3. Knuth, D. E. (1997). The Art of Computer Programming (Vol. 1–3). Addison-Wesley.
4. Berdiyeva, G. (2024). Classroom observation notes and assessment records, School No.1, Kitab District, 2022–2024. [Unpublished raw data].
5. Polya, G. (1945). How to Solve It. Princeton University Press.
6. Skiena, S. S., & Revilla, M. A. (2003). Programming Challenges: The Programming Contest Training Manual. Springer.
7. Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.
8. Berdieva, Gulnoza. "The Role, Importance And Relevance Of Information Technology In The Motivational Phase Of Teaching." *The American Journal of Applied sciences* 3.04 (2021): 334-338
9. Berdieva, Gulnoza. "The importance of students' use of information technology in computer science." (2021).