# Algorithm X: Competitive Programming as a Teaching Methodology for Algorithms and Data Structures in High School

**Odair Moreira de Souza**
Federal Institute of Education, Science and Technology of Paraná (IFPR), Cascavel, Brazil
Graduate Program in Informatics (PPGInf), Federal University of Paraná (UFPR), Curitiba, Brazil

**Clodis Boscarioli**
Graduate Program in Computer Science (PPGComp), Western Paraná State University (Unioeste), Cascavel, Brazil

**Letícia Mara Peres**
Graduate Program in Informatics (PPGInf), Federal University of Paraná (UFPR), Curitiba, Brazil

## Abstract

**Contextualization:** Teaching algorithms and data structures in technical high schools presents challenges related to abstraction, problem solving, and the persistence of non-student-centered teaching models. In this context, Competitive Programming and Competition-Based Learning emerge as promising pedagogical alternatives. **Objectives:** This study aims to analyze the educational impacts of the teaching project "Algorithm X" on students' learning of algorithms and data structures, as well as their motivation, autonomy, academic engagement, and perceptions regarding Competitive Programming in Technical High School. **Materials and methods:** The study is based on the experience of a teaching project developed with students from the Technical Course in Informatics Integrated with High School at Cascavel, Paraná, Brazil. The project was organized through weekly meetings structured around dialogic expository classes, problem solving, coding dojo, debates, simulated contests, and digital support resources. The data were obtained through an evaluative questionnaire by 37 students, and discursive responses were analyzed. **Results and discussions:** The results indicated a favorable evaluation of the project and recognition of its contribution to strengthening algorithmic thinking, developing problem-solving skills, motivating the study of programming, improving performance in programming-related subjects, and fostering learning autonomy. The project was understood as a space for learning and development that extended beyond the immediate context of competitions. **Considerations:** Competitive programming may constitute a viable pedagogical strategy for teaching algorithms in technical high schools when supported by structured didactic mediation, progressive practice, and digital resources. This approach consolidates technical skills and competencies and contributes to students' academic and professional development. Student participation in the project was not limited to the context of competitions but also involved a complementary formative process.

**2012 ACM Subject Classification** Applied computing → Education; Applied computing → Computer-assisted instruction; Applied computing → Collaborative learning; Social and professional topics → Computer science education; Social and professional topics → K-12 education

**Keywords and phrases** Competition-Based Learning, Algorithmic Thinking, Competitive Programming, Olympiad in Informatics, Algorithm Education in High School

**Digital Object Identifier** 10.4230/OASIcs.ICPEC.2026.6

**Acknowledgements** We thank the research participants, the Federal Institute of Education, Science and Technology of Paraná (IFPR), and the Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil, for their support of this research.



## 1 Introduction

The teaching of algorithms and data structures is a fundamental component of Computer Science education, recognised as a central topic in international reference curricula and as essential at all levels of Computer Science and Software Engineering [18, 15]. However, it is considered one of the most challenging subjects for novice students from both cognitive and educational perspectives, as the difficulty lies primarily in high-level concepts rather than low-level technical details [18, 22].

Empirical studies support this perception by showing that most first-year students struggle with abstraction and algorithmic thinking [2]. Survey data from first-year students indicate that 86% fall below the level considered satisfactory in abstraction skills, while 52.6% demonstrate insufficient performance in algorithmic thinking [2]. As abstraction is considered a key element in the early stages of learning, insufficient development of this competence directly hinders the transition from theory to code implementation [18].

In the Brazilian context, this challenge is accentuated by the predominance of traditional teaching practices in middle and high school education. According to Bandeira et al. [1], traditional teaching is understood here as a teacher-centered approach based on theoretical classes, content exposition, exercises on specific topics, written examinations, and final projects, with limited emphasis on self-directed learning and iterative problem solving. This background may make self-learning less common among students and create difficulties when they enter Computing courses, where they are often expected to learn more independently [1]. In this context, Competitive Programming can be used as an active methodology to revisit concepts that were insufficiently consolidated in previous educational experiences and to encourage students to seek solutions more autonomously [1].

As a pedagogical strategy, Competitive Programming extends beyond the concept of a tournament and can be understood as an instructional model in which contest problems structure the teaching and learning of algorithms. This approach fosters the development of observational skills, enabling students to reduce unfamiliar problems to known patterns and to make progress in implementing solutions through efficient coding and debugging [24, 15]. Furthermore, the use of automatic judging systems provides instant feedback, facilitates real-time error correction, and encourages independent learning, innovative thinking, and the development of logical reasoning and problem-solving skills [24, 1].

At the theoretical level, this proposal can be understood in light of Competition-Based Learning, defined as a student-centered strategy that uses tournaments as a stimulus to maximize learning outcomes [4]. From this perspective, competitive practice drives active learning, where knowledge is constructed through interaction, testing competencies, and confronting peers [11]. This process is not limited to the acquisition of technical skills, but also supports the development of socio-emotional competencies such as resilience, adaptability, confidence, and reasoning, associated with preparation for complex real-world challenges [6, 16].

Although the literature identifies Competitive Programming and Competition-Based Learning as promising strategies, there is still a lack of applied studies that systematically examine these models in real contexts of Technical Secondary Education. Most research focuses on higher education or short-term experiences, such as isolated marathons or specific educational projects. Therefore, it is necessary to investigate how the continuous integration of these practices into the technical curriculum can influence the learning of algorithms and students' perceptions of their academic and professional gains. Thus, this study is justified by the need to understand to what extent the use of competition as a teaching strategy can contribute to addressing the limitations of non-student-centered models and support formative trajectories in the field of Computing.

This article analyzes the educational impacts of students' participation in the teaching project "Algorithm X: Teaching Algorithms for Competitive Programming in Technical High School", developed with students from the Technical Course in Informatics Integrated with High School Education at Federal Institute of Education, Science and Technology of Paraná (IFPR), Cascavel, Paraná, Brazil. The objective is to examine, with emphasis on the learning of algorithms and data structures, students' perceptions of the project's contributions to the development of autonomy, motivation for study, and academic engagement, as well as to discuss how this experience relates to fostering interest in Computing and the construction of formative and professional trajectories.

The article is organized as follows. Section 2 discusses Competitive Programming as a teaching strategy in relation to Competition-Based Learning, Algorithmic Thinking, and Problem Solving. Section 3 presents the project analyzed and the research methodology. Section 4 presents the analysis of the project's results and students' perceptions. Finally, Section 5 presents the final considerations and perspectives for continuing the investigation.

## 2 Background

Competitive Programming has been explored as a non-traditional approach based on the Algorithmic Problem Solving method, serving as an active methodology in introductory programming courses. From this perspective, it contributes to increasing students' proactivity, familiarity with programming logic, and ability to abstract real-world problems into algorithms [1].

Competitive Programming can be understood from two complementary perspectives: (i) as an active methodology integrated into first-year formal curriculum classes, using timed contest-based activities and the practice of upsolving, where students solve unfinished problems outside the classroom to progressively develop algorithmic problem-solving skills; and (ii) as a motivational strategy inspired by the "mind sport" nature of contests, encouraging students to seek solutions independently through positive feedback, thereby increasing their enthusiasm, familiarity with programming, and technical confidence [1]. Case studies in introductory courses show that this practice stimulates interest in learning, fosters autonomous learning, and strengthens innovative thinking and problem-solving skills [24]. However, the fact that only 27% of novice competitors are able to solve at least one problem reveals a gap in the perception of difficulty between teachers and students, reinforcing the need for more appropriate questions organised in a progression of complexity [24].

In teaching algorithms, Competitive Programming uses contest problems as a structuring element to introduce, practice, and reinforce algorithmic techniques such as searching, sorting, basic data structures, and dynamic programming at introductory levels. These problems align with Computing curricular guidelines and have been associated with improvements in algorithmic thinking, problem solving, motivation, and, in some contexts, retention in courses in the field.

Luo (2025) organizes learning around three central outcomes: (i) the capacity for observation to reduce unknown problems to known problems; (ii) mastery of algorithmic techniques; and (iii) implementation through efficient coding and debugging [15]. To achieve these outcomes, the author uses a competition-based teaching model in which classes alternate between introducing new concepts, collective problem discussions, and formative assessments that simulate the time pressure of competitions. This format incorporates the time factor, bringing the experience closer to contexts such as contests, technical interviews, software sprints, and hackathons, while at the same time favouring the understanding of problem

statements, team organisation, strategic decision-making, and code debugging under pressure [15]. In addition, automatic judging systems provide instant feedback, enhancing the cycle of discovery and continuous improvement through multiple attempts after incorrect submissions [24]. Furthermore, cooperative learning strategies and team competitions encourage the strategic division of tasks and the exchange of technical knowledge among students, bringing the pedagogical dynamic closer to that observed in programming contests and real-world professional challenges.

A central benefit of this approach is its emphasis on knowledge transfer, as students must recognize the type of problem, relate it to known patterns, and adapt the solution. This process reinforces structural understanding rather than memorization. In this sense, adapting contests for beginners, with a focus on interpreting algorithms rather than full implementation, has proved to be an alternative for broadening the participation of less experienced students, while also improving logical reasoning, supporting the elaboration of test cases, and fostering the construction of pseudocode [17].

## 2.1 Competition-Based Learning

Competition-Based Learning is a methodology that uses structured competitions to stimulate motivation and improve student performance. As a student-centered instructional strategy, it aims to capture students' interest and encourage curiosity, allowing greater control over their own learning process. Additionally, the learning outcome is independent of the score or ranking in the competition, so the tournament serves as a pedagogical incentive without academic success depending strictly on competitive performance [4].

This methodological approach involves the development of competence, autonomy, and relatedness through challenges that make achievements visible and reinforce students' intrinsic motivation for accomplishment [11]. Previous research highlights that competition can foster a state of flow by establishing clear goals and providing immediate feedback, thereby intensifying engagement when the level of challenge remains compatible with participants' skills [5]. In addition, symbolic rewards such as badges and rankings can reinforce students' perception of progress and contribute to learning [5, 14, 21]. From a constructivist perspective, this dynamic supports active learning through interaction with problems and peers [5, 11].

Pedagogical planning should include guided challenges that extend beyond students' current capabilities [16], combined with real project-based activities that stimulate critical thinking, problem solving, scientific reasoning, collaboration, and communication [20]. Its success also depends on sustained participation across annual cycles or multiple semesters, allowing students to progressively consolidate scientific knowledge and skills [20]. When integrated into the formal curriculum and accompanied by mentors or specialists, this methodology can promote active learning, reflection, and reduced performance disparities [6, 16]. In the context of teaching and learning algorithms through Competitive Programming, this framework is realized in contests, mini-contests, and assessments with defined deadlines, often mediated by automatic judges, which encourage intensive practice and the application of algorithmic techniques under time constraints [15, 24]. Thus, Competition-Based Learning and Competitive Programming converge by creating authentic problem-solving environments, where competition structures complex tasks and supports progress monitoring through evidence of performance [6, 16].

In this arrangement, Algorithmic Thinking is directly encouraged, as competition-style problems require students to decompose statements, abstract relevant information, define systematic actions, and evaluate solution adequacy and efficiency [8, 12, 13]. These skills extend beyond coding, involving understanding why an algorithm works, generalizing strategies,

and recognizing recurring patterns, which supports structural understanding rather than memorization [10]. Thus, competitions and challenges can support the consolidation of algorithmic thinking as a central student competence, especially when aligned with curricular objectives and organized with progressive difficulty [15].

Problem solving is also a central component of this convergence, as competitive environments induce iterative cycles of formulation, implementation, testing, and debugging, often supported by automatic feedback. This process supports learning through trial and error and the continuous refinement of solutions [19, 24]. In particular, Cuba-Ricardo et al. [7] show that, when solving algorithmic programming problems, students tend to think and solve the problem while coding: they reinterpret the written code, compare it with the algorithmic solution conceived mentally, gradually correct errors, and refine partial solutions. This emphasizes the importance of pedagogical strategies that guide reasoning, planning, and quality control of solutions. In this context, explicitly elaborating implicit algorithmic notions can contribute to greater discipline in problem solving and foster students' computational thinking competence [9].

Even so, the effectiveness of competition-based learning depends on balancing the level of challenge with participants' competencies, as highly heterogeneous contexts can intensify anxiety and demotivation, especially among less experienced participants [5, 21]. In such cases, collaborative and cooperative learning dynamics, where teams compete externally and collaborate internally to achieve common objectives, can support the articulation of reasoning, negotiation of strategies, and collective correction [23]. Similarly, non-competitive contests have emerged as a relevant pedagogical alternative, as they retain formative elements of the contest format, such as problems, deadlines, and feedback, but moderate the adverse effects associated with rigid rankings and excessive pressure, shifting the focus to learning, cooperation, and reflection [17]. This adaptation tends to make the strategy more inclusive in Secondary Education, while maintaining the motivational and practical benefits of Competitive Programming without compromising educational objectives centred on algorithmic thinking and problem solving [3].

## 3 The Algorithm X teaching project in focus

The teaching project entitled "Algorithm X: Teaching Algorithms for Competitive Programming in Technical High School" is a complementary educational initiative to regular teaching, aimed at students in the Technical Course in Informatics Integrated with High School at IFPR. This project was conducted from March to November 2025, comprising 80 class hours through weekly meetings in a computer laboratory. It focused on developing competencies in logic, algorithms, data structures, and computational problem solving, articulated with preparation for the Brazilian Olympiad in Informatics (OBI[1]) and other competitions. In this context, students' participation in the OBI allowed the training developed in the meetings to be connected with concrete competition experiences, bringing the teaching process closer to real situations involving the application of knowledge constructed in the project.

The methodological proposal was structured based on Competitive Programming as an educational strategy, integrated with principles of active learning, collaborative learning, and problem-based learning. The course consisted of a sequence of didactic practices that combined moments of conceptual introduction, guided problem solving, collective discussion of solutions, and training in competition format. This approach was implemented through dialogic expository classes, problem solving, coding dojo, debates, simulated contests, and the use of progressive sets of exercises.

[1] OBI – `https://olimpiada.ic.unicamp.br`

The methodological teaching approaches were organized to address both conceptual development and intensive practice. Dialogic expository classes introduced the theoretical foundations necessary for competitive programming, especially topics in logic, discrete mathematics, algorithms, and data structures. These sessions aimed to present content interactively, with explanations, examples, simulations, and opportunities for student participation. Problem solving was the central methodology of the project, as the OBI and other competitions require analysis of problem statements, elaboration of strategies, and implementation of correct and efficient solutions. Problems were worked on individually and, at some moments, in pairs. This organization was determined by the teacher's formative assessment during meetings, considering the complexity of the topic, students' recurring questions, performance in previous exercises and submissions, and the autonomy observed during problem-solving activities. The class's level of mastery was not formally quantified but was inferred from classroom observation and students' difficulties during practice.

As a complementary collaborative learning strategy, coding dojo was used to promote collective problem-solving and discussion of algorithmic reasoning in the classroom. In this dynamic, participants alternated roles such as pilot and co-pilot, while others observed, suggested alternatives, and followed the process of constructing the solution. Debates on problems and solutions were incorporated into meetings at two distinct stages: before implementation, to discuss possible algorithmic approaches, and after resolution, to compare methods, data structures, and implementation decisions. This arrangement allowed exploration of the same task from different perspectives, encouraging reflection on the efficiency, clarity, and adequacy of the solutions.

Simulated contests formed another important dimension of the methodology, aiming to familiarize students with the real conditions of competitions. These activities were designed to reproduce the format of the OBI and similar events, involving problems of varying difficulty, limited time, and the need to define a resolution strategy. The goal was to develop not only technical mastery but also skills related to time management, interpretation of problem statements, and emotional control in testing situations. In parallel, sets of problems selected according to the content studied and the participants' level of development were used, with the main sources being the OBI website and the Beecrowd platform[2].

The problems used in the project were mainly implementation-oriented tasks, in which students had to read a statement, identify the underlying algorithmic idea, design a solution strategy, implement it, and test the submitted code using automatic judges. Some meetings also included complementary activities focused on tracing algorithms, discussing strategies in plain language, elaborating test cases, comparing alternative implementations, and analysing wrong submissions. Problem selection followed the OBI Programming syllabus and the topics planned for the project, such as logic, arithmetic reasoning, sorting, searching, basic data structures, graphs, trees, and dynamic programming. The task types and learning objectives are presented in the (Table 1). The problems were selected by the project teachers according to their alignment with the learning objectives, the difficulty level indicated by the source platforms, and the students' observed progress during the meetings. Thus, validation was pedagogical and curricular, based on the correspondence between the problems, the contents taught, and the skills expected in competitive programming activities.

Regarding the description of teaching practices, the project aimed to articulate the progression of content and the diversity of formative experiences. The planned content included foundations of mathematics, logic, and computing, followed by topics in algorithms

[2] Beecrowd – `https://www.beecrowd.com.br`

**Table 1** Examples of problem types used in the project.

| Topic | Type of task | Learning objective |
|---|---|---|
| Logic and arithmetic reasoning | Code-writing and test-case discussion | Interpret statements, identify input-output patterns, and construct basic algorithmic solutions. |
| Sorting and searching | Implementation, tracing, and comparison of strategies | Understand ordering, search strategies, and the relationship between algorithm choice and efficiency. |
| Basic data structures | Implementation and plain-language explanation | Apply stacks, queues, lists, or dictionaries to represent and manipulate problem data. |
| Graphs and trees | Modelling and implementation | Represent relationships between entities and solve connectivity, traversal, or hierarchy problems. |
| Dynamic programming | Strategy discussion and guided implementation | Identify subproblems, recurrence relations, and reuse of intermediate results. |

and data structures, such as algorithm analysis, Big O notation, algorithmic strategies, sorting, searching, dynamic programming, graphs, trees, and geometry. The selection of content also was based on the syllabus of the Programming modality of the OBI, with the possibility of including additional topics according to the class's progress and the availability of other competitions during the project.

Digital resources supporting teaching were used as an integral part of the methodology. The Beecrowd platform was employed in training sessions and simulated contests, as it provides a large collection of problems categorized by difficulty and offers automatic real-time feedback on submissions. The OBI website was used to access previous exams, challenges, and official materials, and also served as a submission channel during the competition phases. For visualizing the functioning of algorithms and data structures, Data Visualization[3] and VisuAlgo[4] were used as support tools for understanding processes such as sorting, searching, graph manipulation, trees, lists, stacks, and queues. As complementary study materials, W3Schools[5] was used for reviewing language foundations and syntax, and GeeksforGeeks[6] for deepening advanced concepts and typical competitive programming strategies.

Table 2 summarizes the main elements that structured the project methodology, organizing it into three complementary axes: methodological approaches, teaching practices, and digital support resources. It presents, in an integrated way, how the pedagogical proposal was operationalized throughout the meetings, linking the references of active, collaborative, and problem-based learning with the didactic strategies used and the digital tools adopted for practice, visualization, and autonomous study.

## 4 Results and Discussion

An evaluative questionnaire was developed, and the 38 students who participated in the project were invited to respond voluntarily, with anonymity guaranteed. Of these, 37 students completed the questionnaire and composed the study sample. All respondents were students in the Technical Course in Informatics Integrated with High School at IFPR – Campus Cascavel. The mean age was 16.41 years (standard deviation = 0.96). Most participants

[3] Data Visualization – `https://www.cs.usfca.edu/~galles/visualization/`
[4] VisuAlgo – `https://visualgo.net/en`
[5] W3Schools – `https://www.w3schools.com`
[6] GeeksforGeeks – `https://www.geeksforgeeks.org/competitive-programming-a-complete-guide/`

**Table 2** Pedagogical purposes of the strategies.

| Axis | Main elements | Pedagogical purpose |
|---|---|---|
| Methodological approaches | Active, collaborative, problem-based learning; competitive programming as a teaching strategy | Promote active participation, problem solving, and the development of algorithmic thinking |
| Teaching practices | Dialogic expository classes; problem solving; coding dojo; debates; simulated contests; problem sets | Articulate conceptual introduction, guided practice, collaboration, and training in a context similar to competitions |
| Digital resources | Beecrowd; OBI website; Data Visualisation; VisuAlgo; W3Schools; GeeksforGeeks | Support problem-based practice, algorithm visualisation, immediate feedback, and complementary study |

were male (75.7%), while 24.3% identified as female. Regarding distribution by school year, 27.03% were in the first year, 40.54% in the second year, and 32.43% in the third year. The third-year students had already participated in previous editions of the project, which should be considered when interpreting their responses. These data indicate that the sample included students at different stages of the technical high school programme, with a higher proportion of 2nd-year students. In 2026, 91,67% of the project alumni are pursuing undergraduate studies in Computing at public institutions.

Exploratory subgroup analyses were conducted to examine whether students' responses varied according to school year or previous participation in the project. Kruskal-Wallis tests did not indicate statistically significant differences among first-, second-, and third-year students in the main questionnaire dimensions ($p > 0.05$ for all dimensions). Similarly, Mann-Whitney tests comparing students with and without previous participation did not reveal significant differences in the main Likert-scale dimensions. A significant difference was observed only for the perceived contribution of the OBI to learning algorithms, measured through a single item on a 0–10 scale, with higher scores among students who had previously participated in the project ($p = 0.0231$). However, given the small size of the subgroups and the different scale of this item, these results should be interpreted as exploratory evidence.

Regarding the time dedicated to extracurricular study related to the project, 40.5% of participants studied exclusively during meetings, while 37.8% reported dedicating 1 to 2 additional hours per week, 18.9% dedicated 3 to 4 hours, and 2.7% dedicated 5 to 8 hours. This overview indicates that, although a significant portion of students maintain autonomous study practices, for a substantial part of the sample, the project remains the main learning space in competitive programming. Spearman's rank-order correlation indicated a moderate, positive, and statistically significant association between extracurricular study time and students' self-assessment in competitions ($\rho = 0.480$, p = 0.0026). A weaker positive association was also found with students' perception of the OBI's contribution to learning algorithms ($\rho = 0.330$, p = 0.0462). However, the associations with overall perception of the project ($\rho = 0.223$, p = 0.1843) and learning based on competitions ($\rho = 0.225$, p = 0.1813) were not statistically significant. Therefore, these results suggest that extracurricular study time was more clearly related to students' perceived competitive performance than to broader perceptions of the project. Taken together, these results suggest that study beyond formal meetings may be relevant to students' perceived competitive performance and that the project may provide conditions for developing autonomous study habits in technical education contexts. As these exploratory analyses are based on self-reported variables, they should be interpreted as associations rather than causal relationships.

The responses about the most difficult content areas focused on topics with greater abstraction and algorithmic complexity, emphasizing graph algorithms (70.3%), tree algorithms (64.9%), graph concepts (43.2%), and dynamic programming (37.8%). In contrast, the content areas identified as easiest to assimilate were fundamentals of logic (75.7%), sorting algorithms (67.6%), and search algorithms (45.9%). This pattern aligns with the expected progression in algorithms training, where introductory content generally presents a lower barrier to comprehension than topics that require greater capacity for abstraction, modeling, and generalization. This pattern suggests that the project exposed students to advanced content but also highlights the need for specific teaching strategies to support its assimilation, such as organizing progressive learning paths, using guided exercises, and systematically reviewing more difficult concepts.

For study resources outside project time, websites and online platforms were most prominent, especially online judges (67.6%), help from teachers or peers (54.1%), and online forums or communities (35.1%). Regarding the materials considered most useful for learning, higher frequencies were observed for online tutorials (59.5%), video lessons (51.4%), forums and communities (45.9%), and online judges (37.8%). These data indicate that participants' learning relies on digital resources and interpersonal support mechanisms. This suggests that the formative process is not restricted to the project space but encourages motivation for self-learning, complemented by digital learning resources and collaborative interactions.

In relation to students' perception of the project, based on statements rated on a five-point Likert scale, the overall average was 4.13 (standard deviation = 0.65), indicating a generally favorable evaluation. Furthermore, the scale also demonstrated internal consistency (Cronbach's $\alpha = 0.925$), indicating strong coherence among the items in this dimension. The lower internal consistency observed in the students' self-assessment dimension may be related to the heterogeneous nature of this construct. This dimension includes items on understanding problem statements, solving problems, developing strategies, following competitive activities, and recognizing the need for further learning. The last item may reflect metacognitive awareness rather than low performance, which can reduce the homogeneity of the scale. Therefore, this dimension was interpreted cautiously as an exploratory indicator of students' self-perception in competitions.

Figure 1 shows the distribution of responses regarding participants' perceptions of the project, highlighting a predominance of favorable evaluations for most items. Notably, aspects related to understanding the format and rules of the OBI, the appropriateness of the freedom to choose the programming language, the relevance of the content covered, and the contribution of the training to the development of problem-solving skills and preparation for the Olympiad stand out. Consistent with this pattern, the items with the highest averages were: "The meetings helped me to better understand the format and rules of the OBI" (M = 4.43), "The freedom to choose the programming language was an appropriate approach" (M = 4.35), and "The skills learned in the project facilitated my performance in academic subjects related to programming" (M = 4.35), reinforcing the positive evaluation of the pedagogical dimension of the experience.

Relatively lower averages, though still within a positive range, showed greater dispersion in responses. Among these, aspects related to the organization and productivity of the meetings stood out, as did the project's impact on defining future academic and professional choices. This result suggests that the positive evaluation of the project is shared, although its effects on vocational and certain operational aspects were perceived less uniformly.

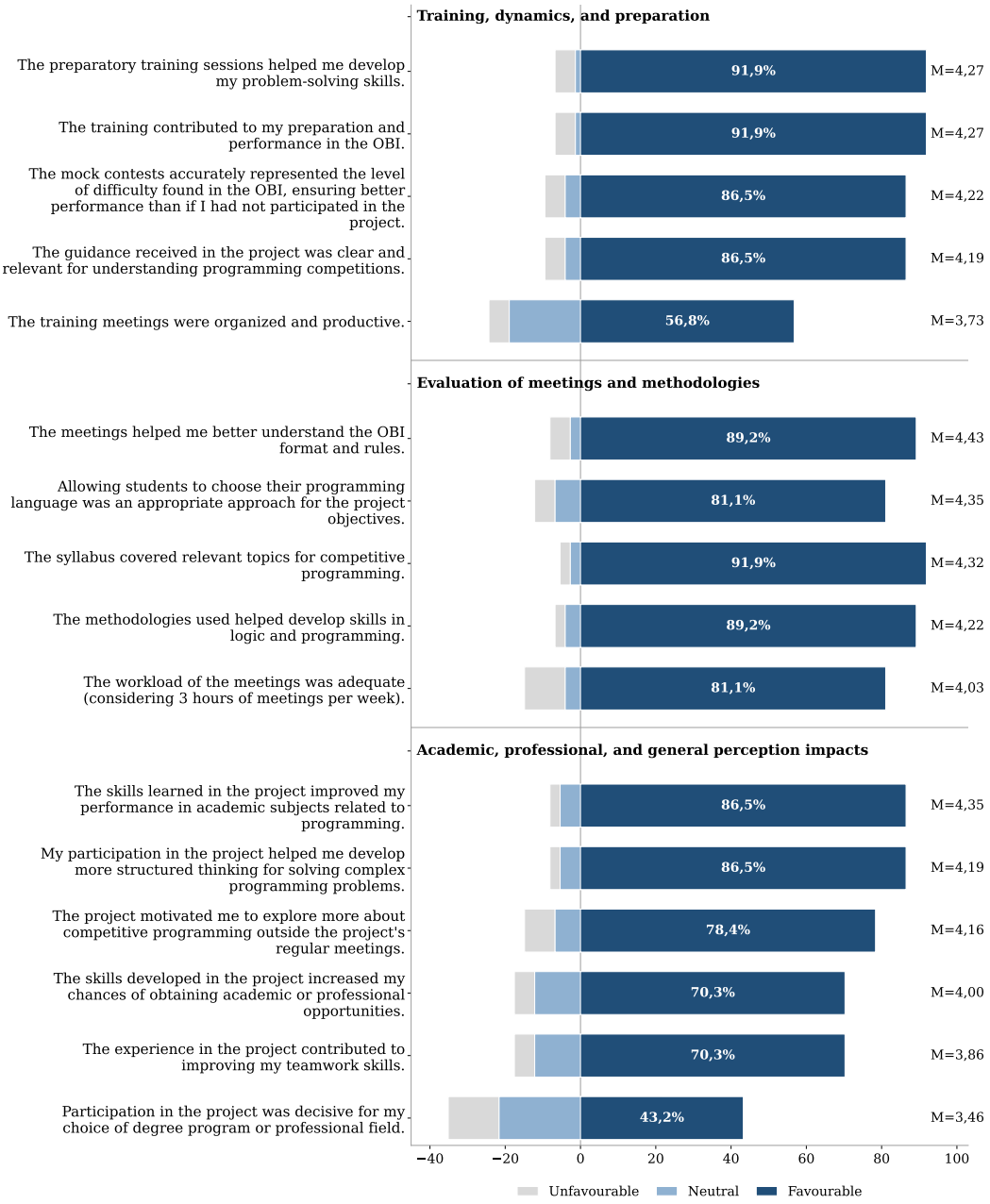


**Figure 1** Students' perception of the project.

Students were encouraged to participate in competitions beyond the OBI, such as the SBC Programming Marathon[7] – Phase Zero and Phase 1 ("Coffee with Milk[8]), as well as programming marathons organized by universities in the state. It should be noted that these competitions are at the undergraduate level; therefore, only a portion of the participants took part in the challenges. Before the competitions, students were familiarized with the exams through simulated contests and by solving problems from previous marathons.

[7] SBC Programming Marathon – `https://maratona.sbc.org.br`

[8] The "Coffee with Milk" category designates teams that participate outside of competition, as they are composed of high school students, while ICPC is for undergraduate students.

Students' self-assessment in the competitions presented a global mean of 4.01 (standard deviation = 0.45), indicating an overall favorable perception of their own performance. However, this dimension showed lower internal consistency ($\alpha = 0.548$), suggesting greater variation among the evaluated items.

Figure 2 shows the distribution of responses regarding students' self-assessment in the competitions. The statement with the highest average was "I believe I still need to learn more about algorithms and programming to perform better in competitions" (M = 4.68), indicating strong recognition among participants of the need for continued study. High averages were also observed for items related to the ability to solve at least part of the proposed problems (M = 4.11), understand the statements (M = 3.95), and develop strategies to address the challenges (M = 3.95).

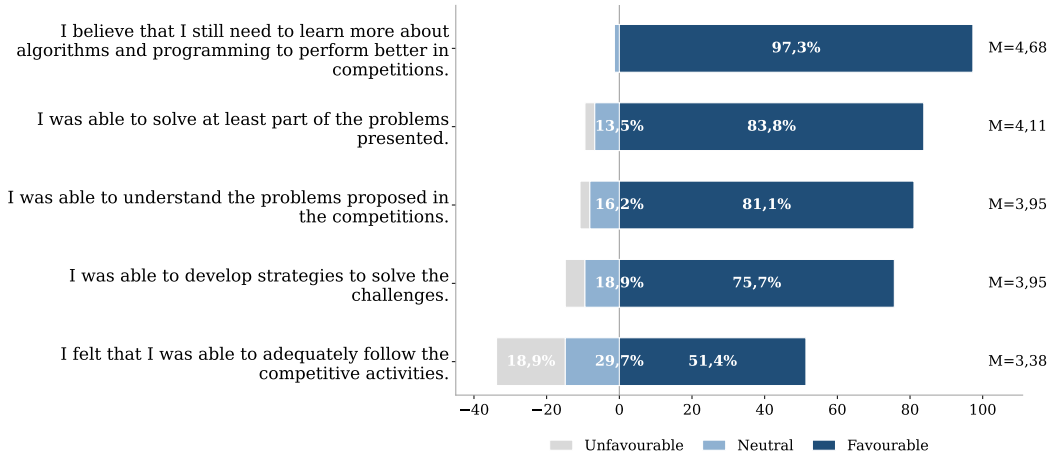


**Figure 2** Student self-assessment in competitions.

In contrast, the statement with the lowest average was "I felt that I was able to adequately follow the competitive activities" (M = 3.38), indicating that some students still do not feel fully adapted to the pace and demands of the competitive environment. This should be interpreted in light of the progressive nature of training in competitive programming, especially in contexts where participants have varying levels of familiarity with this type of activity. The students' perception of Competition-Based Learning showed an overall average of 4.26 (standard deviation = 0.54), with good internal consistency ($\alpha = 0.885$). This result indicates that participants recognize the pedagogical potential of competitive programming as a teaching strategy.

Figure 3 shows a predominance of favorable responses regarding competition-based learning. The item with the highest average score was "stimulates algorithmic thinking and problem-solving" (M = 4.65), followed by "motivates the habit of studying programming" (M = 4.41) and "deepens the understanding of programming concepts and practices" (M = 4.35). Positive evaluations were also observed with reference to interest in programming subjects, study autonomy, and immediate feedback from competitive activities, indicating that students see in this approach a formative contribution that extends beyond the strictly technical dimension. In line with this result, the data suggest that competitive programming is perceived not only as a competitive activity but also as an active learning strategy. The predominance of the "learning and development" category in the descriptive analysis reinforces this interpretation, highlighting an experience associated with the articulation of challenge, engagement, and knowledge construction.

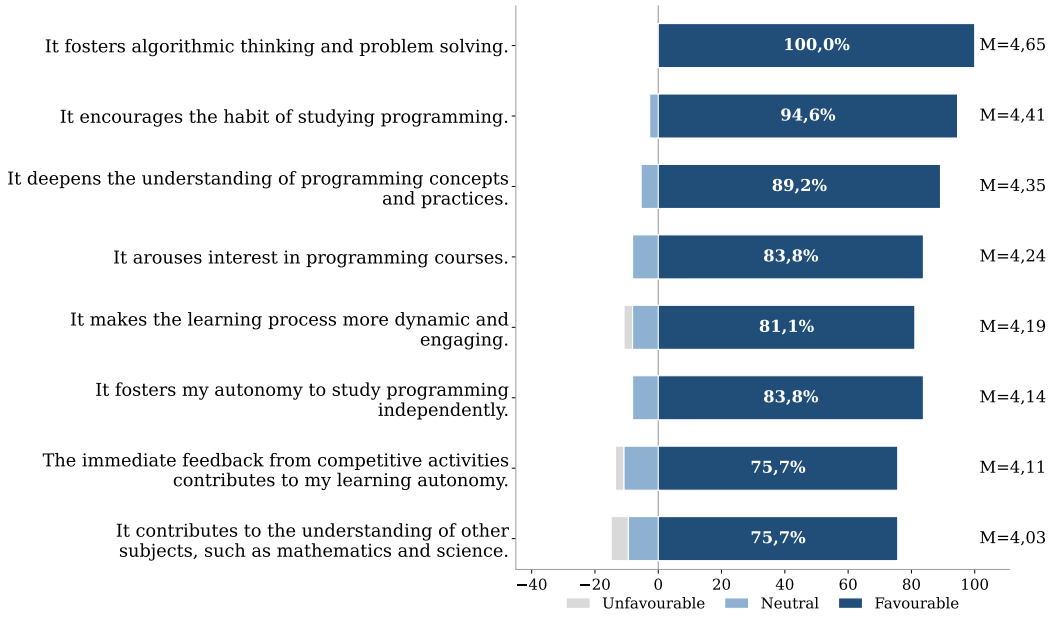


**Figure 3** Students' perception of competition-based learning.

Furthermore, items related to study autonomy and the role of immediate feedback also showed a predominantly favorable distribution, although there was a higher relative presence of neutral responses. On the other hand, the item regarding the contribution to understanding other subjects, such as Mathematics and Science, while positively evaluated, had the lowest mean among the items in this dimension and a smaller proportion of strongly favorable responses. This descriptive pattern suggests that students more clearly recognize the perceived contributions of competition-based learning to skills directly related to programming than to broader interdisciplinary impacts.

The open-ended responses were analyzed using exploratory thematic categorization. The thematic axes were defined deductively based on the questionnaire sections: perceptions of the project, self-assessment in competitions, competition-based learning, difficulties encountered, and suggestions for improvement. For the question about the aspect considered most important in the project, the category of learning and development predominated (75.7%), followed by competition and challenge (24.3%), and, less frequently, socialization and support network (5.4%). These data indicate that the project was interpreted mainly as a space for training rather than solely as preparation for competitive performance.

In the case of the influence of the experience on academic and professional interests, the most prominent categories were increased interest in computing and technology (56.8%) and confirmation or definition of career choice (40.5%). References also appeared to the development of logic and mathematics skills (21.6%) and to professional impact and future opportunities (10.8%). These responses suggest that the experience strengthened participants' connection to the field of computing.

For suggestions for improvement, the most common themes were expanding simulations and practice opportunities (24.3%), deepening advanced content (18.9%), and increasing group activities (10.8%). To a lesser extent, participants mentioned the need for more hours or time dedicated to the project (5.4%), while 13.5% stated that they did not see a need for relevant changes. Overall, these results indicate that the suggestions focus less on

criticizing the pedagogical approach and more on the expectation of expanding and deepening experiences already positively evaluated, providing concrete contributions for the future improvement of the project.

In the overall description of the experience, the following categories stood out: positive and motivating experience (43.2%), improved performance and results (40.5%), development of logic and strategy (32.4%), initial challenge and difficulty (35.1%), and teamwork (18.9%). These findings show that the project was perceived as both a challenging and formative experience, capable of engaging students. Furthermore, the open-ended responses suggest contributions that go beyond immediate learning, indicating increased interest in computing and technology and the consolidation of formative choices, which demonstrates a contribution to strengthening students' academic identity in the field.

Although the main focus of the project is not competitive performance, students demonstrated significant participation in subsequent stages of the Brazilian Informatics Olympiad (OBI). In 2025, 98 students from IFPR - Campus Cascavel participated in the programming category of the OBI. In the Programming (P2) category for students up to the third year of high school, 17 project students ranked among the top 1,000 out of 6,764 participants across Brazil. In the Programming (P1) category for first-year high school students, 13 project students ranked among the top 1,000 out of 3,241 participants. Of the 38 project participants, 30 qualified for the state phase, representing 71.42% of the 42 IFPR students who advanced to this stage, and 11 reached the national phase among the 12 from IFPR - Cascavel (91.66%). The project also included the participation of 10 female students in the OBI Women's Competition. In the first-year high school category, three students achieved 18th place among 243 participants nationwide. In the category for students up to the third year of high school, five students achieved 36th place among 518 participants. Additionally, two project students won medals, and one project student was admitted to a public university through an admission program for high school scientific Olympiad medalists. These outcomes provide contextual evidence of students' competitive engagement and subsequent academic trajectories.

## 5 Reflections and Future Work

The results of this study indicate that the Algorithm X teaching project, designed to teach algorithms for competitive programming in Technical High School, was positively evaluated by students with regard to its pedagogical organization and its effects on learning, motivation, and the development of problem-solving skills. The convergence of quantitative and descriptive data reinforces this interpretation, suggesting consistent recognition of its educational contribution and potential as a pedagogical strategy for teaching algorithms.

From a practical perspective, the findings highlight the importance of expanding simulated contests, exercise lists, and commented problem solving. It is also relevant to structure progressive learning pathways that differentiate between introductory and advanced content, especially in more complex topics such as graphs, trees, and dynamic programming. Additionally, the results reinforce the importance of encouraging autonomous extracurricular study and investing in specific preparation strategies for the competitive environment, such as time management, interpreting problem statements, and familiarization with submission platforms. The value attributed to group activities, in turn, suggests incorporating collaborative practices that articulate individual challenge and shared learning.

Some limitations should be considered when interpreting the findings. Participation in the project and questionnaire was voluntary, which may have introduced self-selection bias, as students more interested in Programming Competitions may have been more likely to

participate. In addition, the study did not include a control or comparison group, and the data are based on self-report rather than objective performance measures. Since the sample is restricted to a specific institutional context, the results should not be generalized broadly. Therefore, the findings should be interpreted as exploratory associations between students' perceptions. Even so, the combination of descriptive statistics, exploratory Spearman correlation analysis, and qualitative evidence provides analytical consistency to the study.

The results also allow for a more precise definition of the conditions under which competitive programming may support more consistent educational contributions. The difficulties reported in content involving greater abstraction indicate that this approach requires an appropriate progression of content, ongoing teacher mediation, and a balance between challenge and support. Thus, the data do not support the idea that simply adopting competitions is sufficient to produce educational improvements; instead, they show that competitive programming can be pedagogically relevant when systematically integrated into the teaching process within a structured proposal that includes guided practices, moments of collective discussions, simulated contests, and study support resources.

From this perspective, students' responses suggest that participation in the project was associated with perceived gains extending beyond the context of the competition, including broader aspects of academic education, the development of algorithmic thinking, autonomy, improved performance in programming-related subjects, and increased interest in further study in the field. In addition, participants received at least introductory exposure to undergraduate-level Computer Science content.

For future development, research may advance in at least three directions: tracking students over longer periods to verify the persistence of perceived effects; incorporating more objective performance measures by combining perception data with results from simulated contests, practical activities, and competitions; and improving the pedagogical approach by expanding progressive learning pathways, strengthening practical activities, and investigating formats that more effectively balance collaboration, autonomy, and technical challenge. Thus, the study contributes to consolidating the understanding that Competitive Programming can constitute an educational strategy in Technical High School, provided it is planned with pedagogical intentionality and analyzed with attention to the specific context and the students' formative needs.

## References


1 Ian Nery Bandeira, Thiago Veras Machado, Vitor F. Dullens, and Edna Dias Canedo. Competitive programming: A teaching methodology analysis applied to first-year programming classes. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–8, October 2019. ISSN: 2377-634X. doi:10.1109/FIE43999.2019.9028518.

2 Javier Bilbao, Eugenio Bravo, Olatz García, Carolina Rebollar, and Concepción Varela. Study to find out the perception that first year students in engineering have about the Computational Thinking skills, and to identify possible factors related to the ability of Abstraction. *Heliyon*, 7(2):e06135, February 2021. doi:10.1016/j.heliyon.2021.e06135.

3 Roberto Borchia, Antonella Carbonaro, Giorgio Casadei, Luca Forlizzi, Michael Lodi, and Simone Martini. Problem Solving Olympics: An Inclusive Education Model for Learning Informatics. In Sergei N. Pozdniakov and Valentina Dagienė, editors, *Informatics in Schools. Fundamentals of Computer Science and Software Engineering*, volume 11169, pages 319–335, Cham, 2018. Springer International Publishing. Series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-030-02750-6_25.

4 Juan C. Burguillo. Using game theory and Competition-based Learning to stimulate student motivation and performance. *Computers & Education*, 55(2):566–575, September 2010. doi:10.1016/j.compedu.2010.02.018.

5 Jordi Carenys. Competing to Learn: The Role of Competition in Students´ Flow, Cognitive Load, and Learning Gains in Game-Based Learning. *International Journal of Learning, Teaching and Educational Research*, 24(5):174–197, May 2025. doi:10.26803/ijlter.24.5.9.

6 Hui-Tzu Chang and Chia-Yu Lin. Applying Competition-Based Learning to Stimulate Students' Practical and Competitive AI Ability in a Machine Learning Curriculum. *IEEE Transactions on Education*, 67(2):256–265, April 2024. doi:10.1109/TE.2024.3350535.

7 Gilberto Cuba-Ricardo, María T. Serrano-Rodriguez, P. Alberto Leyva-Figueredo, and Laura L. Mendoza-Tauler. Methodology for Characterization of Cognitive Activities when Solving Programming Problems of an Algorithmic Nature. *Olympiads in Informatics*, 9:27–37, July 2015. doi:10.15388/ioi.2015.03.

8 Gerald Futschek. Algorithmic Thinking: The Key for Understanding Computer Science. In Roland T. Mittermeir, editor, *Informatics Education – The Bridge between Using and Understanding Computers*, volume 4226, pages 159–168, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. doi:10.1007/11915355_15.

9 David Ginat. On Implicit Means of Algorithmic Problem Solving. *Olympiads in Informatics*, 13:31–39, 2019. doi:10.15388/ioi.2019.03.

10 Dalibor Gonda, Viliam Ďuriš, Anna Tirpáková, and Gabriela Pavlovičová. Teaching Algorithms to Develop the Algorithmic Thinking of Informatics Students. *Mathematics*, 10(20):3857, October 2022. doi:10.3390/math10203857.

11 Omer Sami Kaya and Erinc Ercag. The impact of applying challenge-based gamification program on students' learning outcomes: Academic achievement, motivation and flow. *Education and Information Technologies*, 28(8):10053–10078, August 2023. doi:10.1007/s10639-023-11585-z.

12 Timothy H. Lehmann. Using algorithmic thinking to design algorithms: The case of critical path analysis. *The Journal of Mathematical Behavior*, 71:101079, September 2023. doi:10.1016/j.jmathb.2023.101079.

13 Timothy H. Lehmann. How current perspectives on algorithmic thinking can be applied to students' engagement in algorithmatizing tasks. *Mathematics Education Research Journal*, 36(3):609–643, September 2024. doi:10.1007/s13394-023-00462-0.

14 Yu-Cheng Lin and Huei-Tse Hou. The evaluation of a scaffolding-based augmented reality educational board game with competition-oriented and collaboration-oriented mechanisms: differences analysis of learning effectiveness, motivation, flow, and anxiety. *Interactive Learning Environments*, 32(2):502–521, February 2024. doi:10.1080/10494820.2022.2091606.

15 Zhongtang Luo. Curriculum Design of Competitive Programming: A Contest-based Approach, 2025. Version Number: 1. doi:10.48550/arXiv.2504.00533.

16 Michael McGuire. Impact of Competition-Based Learning on Student Engagement and Performance. *International Journal of Construction Education and Research*, pages 1–30, May 2025. doi:10.1080/15578771.2025.2512348.

17 Caroline Mendes, Péricles Gomes, Alessandro Brawerman, and Eduardo Alberti. Programming competition approach based on the algorithm interpretation. In *INTED2020 Proceedings*, 14th International Technology, Education and Development Conference, pages 4190–4194. IATED, 2-4 march, 2020 2020. doi:10.21125/inted.2020.1172.

18 Ogen Odisho, Mark Aziz, and Nasser Giacaman. Teaching and learning data structure concepts via Visual Kinesthetic Pseudocode with the aid of a constructively aligned app. *Computer Applications in Engineering Education*, 24(6):926–933, November 2016. doi:10.1002/cae.21768.

19 Sarantos Psycharis and Maria Kallia. The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science*, 45(5):583–602, 2017. doi:10.1007/s11251-017-9421-5.

20 Yan Ruan, Junlei Zhang, Jiali Wang, Rui Jian, Feng Mei, Hongli Li, Yun Zhang, Qiwen Hu, Lan Xiao, Yi Yang, Ming Li, Jiaxiang Xiong, and Yanping Tian. Academic competition-based learning cultivates scientific literacy to promote professional competitiveness in medical undergraduates. *Frontiers in Public Health*, 13:1590832, July 2025. doi:10.3389/fpubh.2025.1590832.

21 Michael Sailer and Lisa Homner. The Gamification of Learning: A Meta-analysis. *Educational Psychology Review*, 32(1):77–112, March 2020. doi:10.1007/s10648-019-09498-w.

22 Heera Gurubasavraj Wali, Vijaya Eligar, Venkatesh Mane, and Nalini C Iyer. A Channelizing Approach to teach Data Structures. *Procedia Computer Science*, 172:581–584, 2020. doi:10.1016/j.procs.2020.05.071.

23 Ding-Chau Wang and Yong-Ming Huang. Exploring the influence of competition and collaboration on learning performance in digital game-based learning. *Educational technology research and development*, 71(4):1547–1565, August 2023. doi:10.1007/s11423-023-10247-8.

24 Kevin K. F. Yuen, Dennis Y. W. Liu, and Hong Va Leong. Competitive programming in computational thinking and problem solving education. *Computer Applications in Engineering Education*, 31(4):850–866, July 2023. doi:10.1002/cae.22610.