Competitive Programming as a Source for Problems and Tools for CS1 (Short paper)

Ofer Wald
The Open University of Israel
oferwald@openu.ac.il

Judith Gal-Ezer
The Open University of Israel
galezer@openu.ac.il

Nezer Zaidenberg
Ariel University
nezerjz@ariel.ac.il

תכנות תחרותי במקור לבעיות וכלים בהוראת מבוא למדעי המחשב (מאמר קצר)

עופר ולד
האוניברסיטה הפתוחה
oferwald@openu.ac.il

יהודית גל-עזר
האוניברסיטה הפתוחה
galezer@openu.ac.il

נצר זיידנברג
אוניברסיטת אריאל
nezerjz@ariel.ac.il

Abstract

Teaching introductory computer science (CS1) courses in colleges poses significant challenges. Students enroll in classes with diverse backgrounds and varying levels of experience. College constraints, including budget limitations and limited teaching forces, reduce the number of practice assignments, thus limiting opportunities for students to deepen their understanding of the material.

In this article, we discuss the integration of competitive programming (CP) methods and problems into an introductory computer science course for first year college students. We specifically focus on a single group of 83 students taught by the first author out of 555 who began the academic year at the college, serving as a control group.

The authors provide a brief overview of the competitive programming landscape and discuss the rationale for its integration, along with the development of tools aimed at facilitating this approach. Ultimately, based on our teaching experiences, we illustrate a positive impact of employing competitive programming on students' performance compared to those who followed traditional teaching methods.

Keywords: CS1, CS Teaching, CP, Competitive Programming, Hands-on Teaching, CS Education Tools.

Introduction

A substantial body of research addresses the teaching of CS1, particularly in light of the growing presence of large language models (LLMs) and the increasing challenges faced by instructors (e.g., (Vadaparty, 2024); (Pang, 2024)). CS1 is often students' first encounter with academic computer science and algorithmic problem-solving. This role is especially demanding in college settings, where student cohorts are highly heterogeneous. Students differ widely in mathematical and scientific preparation.

Core concepts such as abstraction and recursion are known to be difficult for novice learners (Armoni, 2013), particularly when combined with learning a first programming language, such as C in our course. Instruction consisted of weekly four hour lectures (recorded for later viewing) and practical workshops emphasizing hands-on learning.

Typically, students complete a small number of assignments that are manually graded, often with substantial delays due to limited instructional resources. To address this issue, we explored the integration of techniques inspired by competitive programming (CP), focusing not on competition itself but on its rich ecosystem of problems, automatic grading, and immediate feedback.

We curated a collection of CP style problems suitable for beginners, designed to require meaningful effort while remaining accessible. A simple example is identifying a “lonely” number in a list where all others appear in pairs. The full set of problems may be found on https://cpe3.islands.co.il .

To support this approach, we developed a web based submission system that provides immediate feedback. While many CS1 problems can be solved by LLMs, problems drawn from CP platforms are significantly more challenging for such models (Shuvo, 2025), reducing the temptation to rely solely on AI tools.

The paper introduces competitive programming in an educational context, describes the developed tool, and reports on its use in a CS1 course.

Competitive Programming

Competitive programming (CP) encompasses algorithmic problem-solving, optimization, and related domains such as cybersecurity and software engineering. Its roots lie in the ACM International Collegiate Programming Contest (ICPC), which has played a central role in shaping the field.

Today, numerous international contests and platforms such as ICPC , IOI (IOI), IEEEXtreme (IEEE), Codeforces (Mirzayanov, 2020), LeetCode, and HackerRank (HackerRank), support millions of participants worldwide. These platforms offer extensive problem repositories, detailed solutions, and automated evaluation systems.

From an educational perspective, CP platforms provide high quality problems with robust test cases that emphasize correctness and efficiency. While complexity was not emphasized in CS1, such platforms offer motivated students opportunities for deeper engagement.

Our goal in adopting CP inspired methods was to streamline the learning process: helping students read and understand problem statements, develop solutions, and receive immediate feedback. Prior studies (e.g., (Coore, 2019); (Yuen, 2023); (Zheng, 2022)) report similar benefits. We view CP not as a competitive framework but as a pedagogical resource that supports engagement, practice, and skill development.

The Website

To integrate CP into our course, we required a suitable technological platform. Existing CP platforms such as HackerRank or Codeforces have been used successfully by others (Nishanov, 2024), but none met our institutional requirements, particularly seamless integration with the college’s single sign-on (SSO) system.

We evaluated several systems, including CMS (Maggiolo, 2012) and Virtual Judge (Han), but found them unsuitable for managing a curated problem collection in an educational setting. Consequently, we designed and implemented our own web-based system.

The main technical challenges security, sandboxing, and execution timing, are beyond the scope of this paper. Our focus here is on the pedagogical affordances of the system: structured problem sets, automated checking, and immediate feedback.

CS1 – Introduction to Computer Science

The system was used in a first semester CS1 course taught in C at a public science-and-technology oriented institution. The course enrolled 555 students across seven classes, taught by six instructors. Student backgrounds varied considerably, particularly between daytime and evening classes.

This semester was exceptional due to external circumstances: the semester started two months late, was shortened by four weeks, and included no mandatory assignments. As a result, the original plan to use the site for required coursework was not fully implemented. Some instructors were reluctant to adopt a new pedagogy under these conditions, and concerns were raised about problem difficulty.

Nevertheless, the first author, who taught one of the classes, encouraged all students to use the site and was more persistent with students in their own group. Problems were released in alignment with course progress, and students could view peer solutions after submitting correct ones. A small bonus (3-5 points) was promised for site participation, though this bonus was excluded from all analyses.

Grades were determined solely by a final exam, with results summarized in Tables 1 and 2.

Table 1. Student results – first exam sitting

Class Number	#Students	Fail %	Average grade	Median grade	STD DEV
1	76	35.5	65.7	71	29.0
2	85	34.1	68.4	75	27.8
3	75	20	74.7	80	27.6
4	88	25	70.8	77.5	27.5
5	53	28.3	70.6	80	28
6	66	16.7	75.5	86	27.4
7	59	33.9	64.3	70	27.8
total	502	27.7	70	77	27.9

Table 2. Student results – second exam sitting

Class Number	#Students	Fail %	Average grade	Median grade	STD DEV
1	44	52.3	52.5	46.5	27.4
2	44	52.3	56.9	54	27.6
3	36	38.9	57.8	68.5	32.5
4	42	33.3	64.0	70.5	25.7
5	21	61.9	52.2	40	30.1
6	28	39.3	57.9	60	30.8
7	38	47.4	53.4	60	23.7
total	253	45.8	56.6	60	28

Analysis of the Author's Group (#3)

Given voluntary participation, analysis focuses on group #3, taught by the first author. Student assignment to groups was pseudo-random, and the author had no influence over selection.

The site was introduced as a supplementary resource, with problems integrated into lectures and released in sync with course topics. Near the end of the semester, a "secret" bonus code was offered after solving several problems, both to encourage participation and to identify users without affecting grading anonymity.

Group #3 included 83 students; 79 took at least one exam. Of these, 55 created site accounts, 53 submitted at least one solution, and 51 solved at least one problem successfully.

Figure 1 shows that site users outperformed the overall student population by 12.76 points in the first exam and 16.2 points in the second.

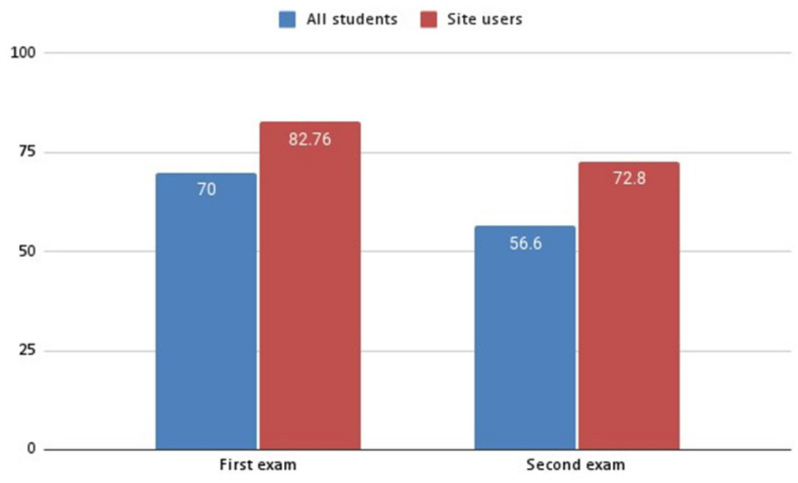

Fig. 1. All students vs Site users in both exams

Figure 2 compares users and non-users within group #3. Non-users averaged 52.04, while users averaged 86.69 - a difference of 34.65 points. This result is interpreted cautiously (see Limitations).

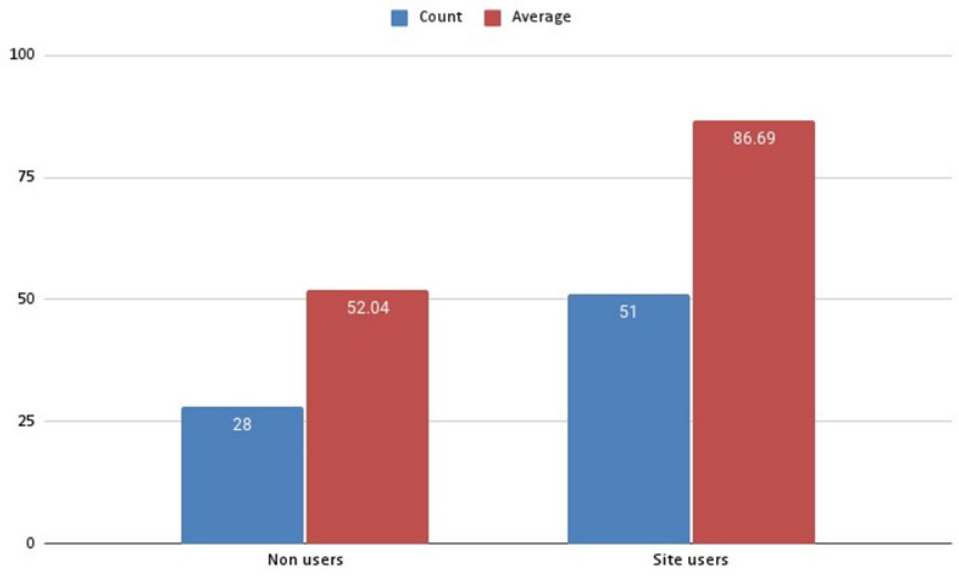

Fig. 2. Users vs Non-users in group #3, count and average

Figure 3 demonstrates a clear trend: students who solved more problems achieved higher exam scores. Anomalies are explained by small group sizes and the bonus threshold.

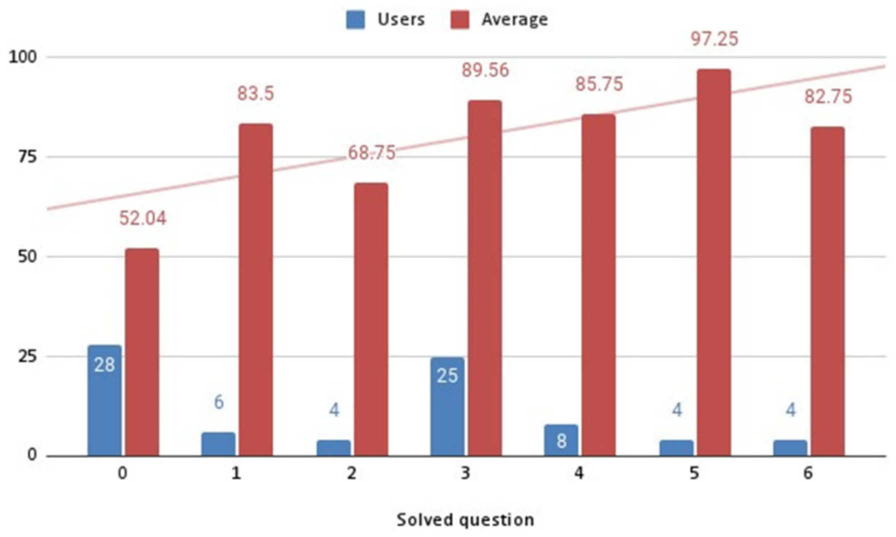

Fig. 3. Users in group #3, by number of successful submissions

Figure 4 compares early adopters with students who joined after the bonus announcement. The difference between the groups was under four points, suggesting that even late engagement provided benefits, though sustained use appeared more advantageous.

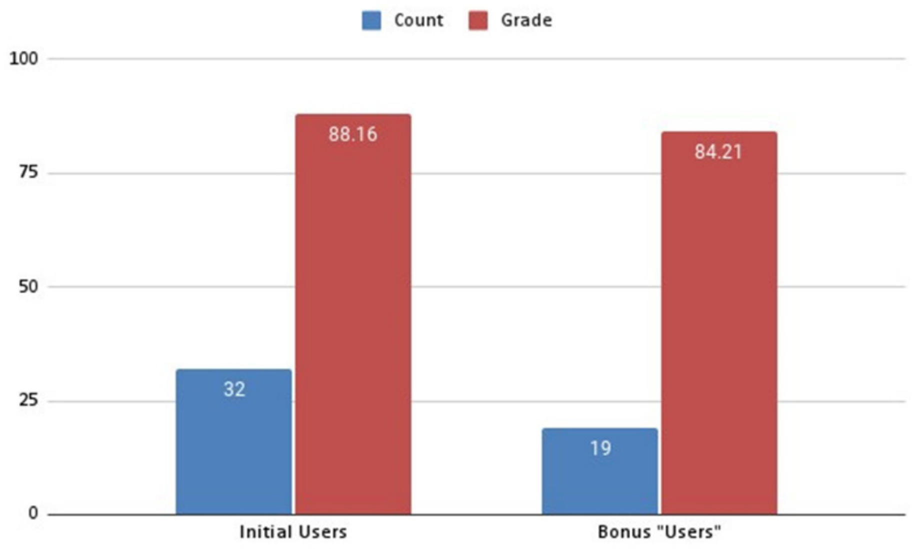

Fig. 4. Initial users vs bonus users

Limitations

This study reports on a single course iteration and a single class. Participation was voluntary, and external factors, most notably a shortened semester, affected implementation. Additionally, the authors' direct involvement in teaching could introduce potential experimental bias.

Discussion and Future Trends

While definitive conclusions cannot be drawn from an experience report, the results are striking. Students who engaged with the site performed substantially better than their peers, and increased effort correlated with improved outcomes.

Our experience suggests that integrating CP style problems and automated feedback into CS1 can enhance understanding and performance, even without emphasizing competition. Expanding the problem set is expected to further benefit students.

We encourage replication of this approach in other institutions and contexts to assess its generalizability, particularly under less constrained conditions.

References

Armoni, M. (2013). On teaching abstraction in CS to novices. *32*(3), 265-284.

Coore, D. (2019). Facilitating course assessment with a competitive programming platform. 449-455.

HackerRank. (n.d.). About HackerRank. Retrieved https://www.hackerrank.com/about-us/

Han, X. (n.d.). Virtual Judge. Retrieved https://vjudge.net/

IEEE. (n.d.). IEEEXtreme – 24 Hour Programming Competition. Retrieved https://ieeextreme.org/

IOI. (n.d.). International Olympiad in Informatics (IOI). Retrieved https://ioinformatics.org/

Maggiolo, S. (2012). Introducing CMS: a contest management system. *6*, 86-99.

Mirzayanov, M. (2020). Codeforces as an educational platform for learning programming in digitalization. *14*(133-142), 14.

Nishanov, A. (2024). Methodology of Teaching Programming Science Through Online Platforms. 1410-1413. doi:10.1109/PIERE62470.2024.10804934

Pang, A. (2024). ChatGPT and Cheat Detection in CS1 Using a Program Autograding System. 367-373. Milan, Italy: Association for Computing Machinery. doi:10.1145/3649217.3653558

Shuvo, U. (2025). Assessing ChatGPT's Code Generation Capabilities with Short vs Long Context Programming Problems. 32-40. Association for Computing Machinery. doi:10.1145/3704522.3704535

Vadaparty, A. (2024). CS1-LLM: Integrating LLMs into CS1 Instruction. 297-303. Milan, Italy: Association for Computing Machinery. doi:10.1145/3649217.3653584

Yuen, K. (2023). Competitive programming in computational thinking and problem solving education. *31*(4), 850-866.

Zheng, Y. (2022). C++ Teaching Reform and Exploration Based on ACM/ICPC and Live Code. 281-286.