Discrete Algorithms - Programming and Collaborative Work: An Experience From the Virtual Classroom Environment

Carlos D. Colliard Schneider
Universidad Autónoma de Entre Ríos – Oro Verde

Miguel A. Fedonczuk
Universidad Nacional de Rosario – Oro Verde

Stella M. Vaira
Universidad Nacional de Rosario – Oro Verde
Universidad Nacional del Litoral

Collaborative learning understands learning as a social process of knowledge construction that goes beyond the individual instance of analysis, conceptualization and appropriation. Discrete Mathematics and Cryptography are two subjects of careers with a systemic and computational profile suitable for the early introduction of programming. With the aim of stimulating student participation and the development of transversal competencies, such as the ability to face a problem, teamwork, programming in a new language, improving oral communication; and in accordance with the line of research in Mathematics Education and training for the improvement of teaching and learning processes in the university classroom, a series of problems were activated in the virtual environment so that students could try to solve them and communicate the results to their peers. The students showed totally adequate resolution qualities, they were timid in their oral communications, but capable of defending the resolution of the problem. That is to say that there is benefit from this type of learning based on the possibilities offered by the new technologies.

Keywords: collaborative learning, programming, discrete mathematics, virtual environments

INTRODUCTION

Today, a science like cryptography, and at its base, discrete mathematics, has become essential knowledge for a society highly surrounded by technology and information. From paying with a bank card to surfing the Internet, cryptographic systems are present at all times. For this reason, the analysis of the available discrete algorithms becomes important in Discrete Mathematics and Cryptography courses. Concepts such as ring structure and modular arithmetic enhance their practicality when applied to everyday activities such as the ones that were mentioned before.

Discrete mathematics is the mathematical language of computer science, and as such, its importance has increased dramatically in recent decades. (Grimaldi, 2004; Kurgalin and Borzunov, 2018).

This paper aims to present a teaching and learning proposal for the subject of Discrete Mathematics of the Bachelor's Degree in Information Systems at the Faculty of Science and Technology in Universidad Autónoma de Entre Ríos, under the methodology of collaborative work and the use of the free programming language Python that was developed and implemented during the year 2019, with the aim of stimulating student participation and the development of transversal skills, such as the ability to face a problem, teamwork, programming in a new language and oral communication.

THEORETICAL FRAMEWORK

Collaborative Classroom Work

According to Maldonado Pérez (2007), "collaborative work, in an educational context, constitutes a model of interactive learning, which invites students to build together, for which it demands that they combine efforts, talents and skills through a series of transactions that allow them to achieve the goals established under consensus". For the author, "the collaborative work used in university classrooms is relevant and timely, because not only do students learn and generate knowledge about aspects of the discipline they are studying, but a great human learning also takes place; it also develops reflective thinking, stimulates the formulation of judgments, the identification of values, the development of respect and tolerance for the opinion of others".

According to Guitert (2000), "collaborative work is a process in which an individual learns more than he/she would learn on his/her own, as a result of the interaction with the members of a team, who know how to differentiate and contrast their points of view, in such a way that they generate a process of knowledge construction".

How do we understand it as a team of teacher-researchers? Collaborative learning understands learning as a social process of knowledge construction (beyond the individual instance of analysis, conceptualization and appropriation), as the need to share knowledge to achieve a goal that transcends individual possibilities. Interaction is emphasized, since learning takes place through the exchange of ideas in a synchronous manner in the first stage of the process where students sit down, agree, debate, solve the presented problem, execute the solution and then expose it to be evaluated by their peers and teachers.

We have new learning tools for the teaching process. The constructivist pedagogy on which collaborative learning is based, holds that knowledge is not received passively, but is actively constructed by the subject. The focus is not on the transmission of content, but on the construction of knowledge, which in turn is based on previous knowledge. That is why the role of the teacher changes, becoming a mediator between knowledge and the student.

Students are engaged in the construction of a meaningful product (Rovero et al., 2018; Panitz and Panitz, 2014); the learner has the possibility of self-managing their own learning.

A technological element, available to teachers, that promotes learning with these characteristics is Moodle (Module Object-Oriented Dynamic Learning Environment), a platform for online course management based on a constructivist pedagogy that complements the work in the classroom, actually augmenting the classroom from being only a physical space to being a combination of the physical and the virtual (Gros, 2000; Salinas, 2004).

On the other hand, this proposal is framed within problem-based learning, which is one of the most widely used techniques in the collaborative teaching-learning process. It takes place when students are confronted with an authentic problem as a starting point for learning. This type of learning has three fundamental axes: promoting student autonomy, fostering collaborative learning, and generating participatory evaluation processes.

Teaching Programming With Python

In Computer Science careers, the teaching of programming has always been a fundamental pillar and is one of the first courses that students entering these university careers must take (Matthíasdóttir, Á., 2006). Madoz et al. (2005), among others, argue that teaching and learning programming in these courses is a complex and difficult intellectual activity, both for students and for those who conduct the instruction; even

more so when its impact is very important in most of the successive subjects and in the professional field of the future graduate. The task of programming in the training of a computer systems professional cannot be limited only to knowing the syntax and semantics of a programming language, but as Meyer (2003) points out, its main purpose is to develop throughout the career a set of skills that will allow the resolution of complex computational problems.

Early incorporation of Python programming in subjects such as Discrete Mathematics and Cryptography is essential since the challenges that a graduate in communication systems must face (Computer Security, Digital Signature, Key Exchange, Intelligent Internet Searching) are based on algorithms, some simple and others complex, which are developed in these subjects.

Why Python? There are studies that point out the advantages of Python and state it as an ideal language to exemplify the implementation of cryptographic algorithms such as those proposed in this article to students. It is in a process of continuous development. Approximately every six months a new version is released.

Mazal and Garcia (2018) summarize the features of Python:

- Simple and expressive syntax.
- High readability: the syntax is very elegant.
- User-friendly interactive environment for easy testing.
- Free of charge. Available on almost any platform.
- Multiparadigm can be used as an imperative, functional or object-oriented language.
- Extensive set of data structures available.

These features make it relatively easy to translate mathematical algorithms into the Python language, and it lends itself as an ideal language to implement when exemplifying the algorithms used in the Discrete Mathematics and Cryptography dictation.

METHODOLOGY

The development of this work began in 2018 and was implemented in 2019, a research methodology with a qualitative approach has been used. Where the main objective is to interpret and understand how a teaching and learning strategy based on collaborative work and algorithm programming that involves an understanding of mathematical concepts, can enhance the learning and interest of students in the Discrete Mathematics and Cryptography subjects of the Bachelor's degree in Information Systems at the Faculty of Science and Technology in Universidad Autónoma de Entre Ríos.

Subjects and Algorithms: Development of the Pedagogical Proposal

The number of students of Discrete Mathematics in 2019 was 60 and they were invited to work under the mentioned methodology: collaborative work mediated by the virtual environment developing "discrete" algorithms; 20 students accepted to work and achieved a quantitative retribution in their grades. Divided into groups of no more than 3, and the problems were postulated in the environment and each group chose one to solve and it remained restricted for the rest of the students.

After about 20 days of time allowed for the resolution, development and subsequent presentation, the seminar was organized for the group to present and their peers to ask questions: the questions were about programming, about the presentation or about the exceptional cases that the algorithm could contemplate (large numbers or numbers with many digits, since it was a special interest, the execution time or response, and the work with prime numbers).

Algorithms that were postulated in the environment:

- Greatest common divisor and Euclidean algorithm
- Dispersion function: compression of a number under linear congruence
- Divide and conquer algorithms; bubble method
- Multiplicative inverse in congruence modeling
- Caesar cipher
- Generator of a U_n group from a Z_n ring

- Encoding and parity checking from matrices (Hamming metric)

Each of the problems was accompanied by a reference bibliography, and although they were not developed in Python, they could be found developed in other programming languages in the references.

RESULTS AND CONCLUSIONS

From the qualitative analysis carried out jointly by the professors and the student peers after each presentation, the following was agreed upon:

- Problem solution: all groups completely solved the chosen problem.
- Preparation of the exposition: all groups presented the expositions with slides and simultaneously analyzed and explained each programmed command line.
- Programming language: low level of difficulty and few previous consultations with the teacher, who in this case acted as tutor. The proposed material was sufficient for them.
- Oral communication: they were very timid in front of their peers, their voices were low in some cases and in response to some questions they answered *we did not test that because the problem did not ask for it.*

The implementation of the pedagogical proposal turned out to be innovative and enriching for the department, allowing to redefine the activities of Discrete Mathematics by combining a collaborative workspace, programming and communication for the development of group work, social and conflict resolution skills.

ACKNOWLEDGEMENT

Translated & edited by American Publishing Services (https://americanpublishingservices.com/).

REFERENCES

Grimaldi, R. (2004). *Discrete and combinatorial mathematics* (Fifth Ed.). Pearson Addison Wesley.

Gros, B. (2000). *The Invisible Computer: Towards the appropriation of the computer in education.* Barcelona, Spain: Gedisa Editorial.

Guitert, M., & Jiménez, F. (2000). *Learning to collaborate. Cooperating in class: Ideas and tools to work in the classroom*. Madrid: R. Rizzi Eds.

Kurgalin, S., & Borzunov, S. (2018). *The Discrete Math Workbook. A Companion Manual for Practical Study*. Cham: Springer International.

Madoz, M.C., Gorga, G., & Russo, C. (2005). *Analysis of the Impact of ICT's in the learning process of university students of initial level*. Congress on Information and Communication Technologies in Science Education. TICEC 05. La Plata. City of Buenos Aires.

Maldonado Pérez, M. (2007). Collaborative work in the university classroom. *Laurus Magazine*, *13*(23), 263–278

Marzal Varó, A., García Luengo, I., & García Sevilla, P. (2014). *Introduction to Python 3 programming.* http://dx.doi.org/10.6035/Sapientia93

Matthíasdóttir, Á. (2006). How to teach programming languages to novice students? Lecturing or not? *Proceedings of the International Conference on Computer Systems and Technologies*. Bulgaria.

Panitz, T., & Panitz, P. (2014). *Encouraging the Use of Collaborative Learning in Higher Education. University Teaching: International Perspectives*. In J.J.F. Forest (Ed.), (pp. 161–201). Taylor and Francis.

Rovero, O., Collazos-Ordoñez, C., & Jiménez-Toledo, J. (2018). Collaborative work as a didactic strategy for the teaching/learning of programming: A systematic literature review. *Medellín: Tecnológicas*, *21*(41).

Salinas, J. (2004). Teaching innovation and the use of ICTs in university education. *Universidad y Sociedad del Conocimiento Magazine*, *1*(1), 1–16.