# Lempel: Developing the pattern recognition skill in computational thinking through an online educational game

Ekaitz Polledo[1[0000-0002-9928-7667]] and Pablo Garaizar[1[0000-0001-8160-9130]] and Mariluz Guenaga[1[0000-0002-0311-2150]]

[1] Faculty of Engineering, University of Deusto, 48007 Bilbao, Spain

[epolledo, garaizar, mlguenaga]@deusto.com

**Abstract.** Computational thinking is a key set of skills for the 21st century's digital literacy. Taking advantage of computers to solve complex problems automatically will be helpful in most future jobs. Among the skills that comprise Computational Thinking, pattern recognition plays an important role in managing and compressing information. To foster the development of this skill among primary and secondary school students, we have developed Lempel. In this game, we propose a set of challenges of increasing complexity in which players have to provide a compressed version of the information presented. Lempel's fine-grained interaction data logging system allows us to use Learning Analytics techniques to better understand how the learning of this skill takes place.

**Keywords:** Computational Thinking, Pattern Recognition, Educational Games, Text Compression.



## 1 Introduction

Due to business needs and the importance of technology in our society, the concept of Computational Thinking has emerged in recent years, especially focused on compulsory education. STEM (Science, Technology, Engineering and Mathematics) are priority areas in education in Europe and basic skills in arithmetic, mathematics and science are considered fundamental foundations for further learning [1]. This goes beyond programming by enabling problem solving, system design and understanding of human behavior by making use of the fundamental concepts of computer science [2]. Everyone can benefit from applying these concepts to their daily lives, based on a spiral that includes society, science and technology in which all affect and enrich each other [3]. Computational Thinking main skills are decomposition, pattern recognition, algorithm solving and abstraction.

Computational Thinking has become one of the topics of global attention as part of the efforts to bring computer science to all K-12 schools [7]. In addition, initiatives such as Hour of Code or CodeWeek have boosted the development of this competence, making it accessible to millions of students through free digital platforms.

The increased use of digital tools for learning has resulted in the use of Learning Analytics to be able to make decisions on a large amount of user interaction data. The Society for Learning Analytics and Research (SoLAR) defined learning analytics as "the measurement, collection, analysis and reporting of data about learners and their

contexts in order to understand and optimize learning and the environments in which it occurs." [10]. Numerous studies support it and demonstrate its potential to improve engagement and motivation, to support teachers and even to predict results [11][12]. In addition, in combination with other models [13] it can categorize learners according to their way of processing information and can help to personalize their learning path.

In section 2 we present the fundamentals of computational thinking and some tools used for its development. Section 3 describes the "Lempel" game itself, its design and development phases. Section 4 describes the experimental approach used and preliminary results. Finally, conclusions and future lines of work are presented in section 4.

## 2 Computational Thinking

Many applications can be found for the development of Computational Thinking [8]. Code.org is a non-profit organization that has several games for learning programming and creating new challenges. Scratch is a tool for creating games, animations and interactive resources using a visual programming language. Blockly is a game of successive challenges for learning programming based on a visual programming library. Finally, MakeWorld is a platform that provides a methodology and innovative educational resources for learning STEM while developing Computational Thinking.

Currently, several models have been defined to understand how students develop Computational Thinking. Werner et al. have followed an analysis to describe how middle school students program in Alice [4]. Piech and collaborators have used a Markov model to describe how students reach solutions [5]. Seiter and Foreman developed a progression model that was used to relate good programming practices and the age of the authors [6]. All the mentioned studies are based on the analysis of algorithm solving and there are not numerous models based on the other 3 skills of Computational Thinking. This trend has led to consider coding as the core of Computational Thinking [7].

In this case, we are focusing on pattern recognition. Some authors highlight the importance of the analysis of this competence and the lack of studies on it [14]. Previous studies have examined some aspects of pattern recognition such as the identification and completion of patterns with kindergarten students [15][16]. Moreover, this is one of the most complete CT competencies associated with other competencies such as abstraction [17].

Therefore, we present Lempel, a tool for the development of computational thinking based on pattern recognition. Our objective with this tool is, through the application of Learning Analytics, to analyze the development of this computational thinking competence in learners taking into account their personal characteristics and the faced challenges. To perform this analysis, we anonymously collect user interactions following the best practices of other authors in similar experiments using Learning Analytics and Computational Thinking [18][19][20].

# 3 Lempel

Lempel is an online educational game developed by the Deusto LearningLab group at the University of Deusto. It has been created to be an educational resource to help the development of Computational Thinking in a classroom or stand-alone environment. Participants must compress a text string displayed on the screen composed of different characters. To achieve this, they must recognize the pattern or patterns in the string and insert them into containers called "registers". The game consists of a series of blocks that represent the different characters of the strings or calls to the different registers. As the blocks are inserted into the different registers and the different patterns are composed, they will be replaced in the initial character chain, giving a visual response to the participant's activity (Fig. 1).

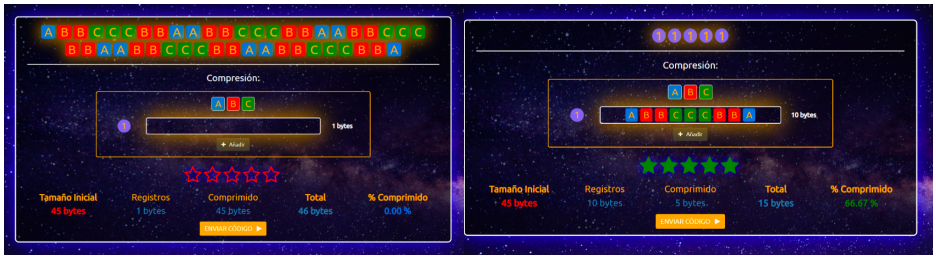


**Fig. 1.** Lempel game main interface

Therefore, this game focuses on the development of Pattern Recognition, which is one of the main skills of Computational Thinking and thus, it goes beyond programming and algorithm-oriented applications by putting the focus on a data-oriented format and its analysis and processing. This game also works on skills such as abstraction to be developed while the user focuses on the patterns to be compressed and forgets about the characters around him. It is designed to suitable for everyone, whether they have previous knowledge or not. So, it is not necessary to have previously used Computational Thinking tools.

## 3.1 Game design.

Lempel is based on a space game theme. In this one, a ship going to the moon runs out of space for the processing of all its data, therefore, its crew members must compress them to be able to make space for the new ones and be able to arrive successfully. Through a series of incremental difficulty levels, participants encounter a series of text strings in which they have to recognize the available patterns and thus reduce their size.

### *3.1.1 Game mechanics & GUI*

The interface of Lempel consists of three parts: the chain string to be compressed, the registers to introduce the patterns and the progress indicators. Before starting every block of levels, participants are introduced with tutorials about how to use the game.

In the upper area of the game, you will find the string to be compressed. Each of the characters in the string is represented by a letter and a color to make the game visually more friendly, so the patterns will be easier to recognize. These blocks will be replaced by circles representing the registers as you advance in the level and enter the patterns.

In the middle of the screen, you will find the registers. These are represented by a number and the player will have the possibility to add up to 4 depending on the patterns detected in the proposed chain. The player will have to drag to each of these the different characters available forming a chain that represents a pattern.

Finally, in the lower area you will find the progress indicators. These represent the degree of compression reached in the level by means of the size of the string and the compression percentage, in text format, and the efficiency of the solution through a 5-star scale.

When the player believes that the level is complete and his solution is correct, he must confirm by pressing the "Send Code" button and the game will show them whether it is correct, partially correct (it can still be compressed further) or incorrect.

The different gamification elements such as the stars or the compression limit to pass the level are parameterizable and can be activated or deactivated depending on the desired game mode.

### *3.1.2 Levels*

The current version of LEMPEL is composed of a total of 61 levels, 5 of which are tutorials distributed throughout the game. The levels have been organized in groups of levels of the same category and ordered based on their difficulty calculated through a heuristic formed by the following variables: length of the string to be compressed, different characters available, size of the solution, letters not belonging to patterns, patterns that must call other patterns, number of registers to be used (patterns), size of the registers and patterns composed by equal letters.

The different levels can be classified into the categories listed in Table 1. Each of the initial character chains has an initial size and the solutions indicate the sum of the resulting string and the size of the registers.

**Table 1.** Level types included in Lempel.

| Level Type | Description | Level Example |
|---|---|---|
| Levels with 1 pattern | These levels are composed of patterns with different letters. | ABCDABCDABCDABCD (16)<br>Solution: 1111 (9)<br>1: ABCD |
| Levels with repetitive letters | These levels include patterns formed by the same letter repeatedly. | BBBBBBBBBBBB (12)<br>Solution: 1111 (8)<br>1: BBB |
| Levels with letters out of pattern | These levels contain some letters that are not part of any pattern, therefore they should not be entered in the registers. | ABCDCDCDCDCDCDCD (16)<br>Solution: AB1111111 (12)<br>1: CD |

| | | |
|---|---|---|
| Levels with 2 patterns | These levels include 2 different patterns and can be combined with patterns from the previous categories. | CBCBCBCBADADADAD (16)<br>Solution: 11112222 (14)<br>1: CB 2: AD |
| Levels with 2 registers, using one as a multiplier | These levels use one of the registers to call another one. The first pattern multiplies its content the number of times it is called from the second one. | ABCABCABCABCABCABCABCABC (27)<br>Solution: 222 (11)<br>1: ABC 2: 111 |
| Levels with 2 patterns and registers with characters and calls to other registers | These levels contain 2 different patterns. The first one is simple and the 2nd one is made up of characters and calls to the first of the registers | ABCABCDDABCABCABCABCDD (22)<br>Solution: 2112 (13)<br>1: ABC 2: 11DD |
| Levels with 3 and 4 registers | These levels combine previous categories using up to 4 registers increasing the complexity of the registers. | AAABBBAAAAAABBBBBBAAAAAAB BBBBBAAAAAABBBBBBBBAAA (48)<br>Solution: 1233321 (11)<br>1: AAA 2: BBB 3: 1122 |

### *3.1.3 Logging System*

One of the main keys for the analysis of this game for the development of Computational Thinking is its event fine-grained logging system. Each of the triggered events contains the following information:

- User information obtained at the beginning of the activity through a brief questionnaire. Fields included: user, username, age, gender, group ref.
- Event timestamp. Fields included: user timestamp, server timestamp, level delta time, log order (incremental number).
- Level information. It is the information about the level and challenge of which the event is being registered. Fields included: challenge code (game version), level reference.
- Game information. Fields included: action container (orig./dest.), action object, action position (orig./dest.), action, code, dictionaries (code on each of them), size, size of solution, score.

All these events are logged into the following scenarios: level start, result check, error, partial and success solutions, dictionary add/remove and drag and drop or click actions over blocks.

## 3.2 Technological Implementation

Nowadays there are many technological alternatives for game development. One of the most widespread options for this type of platform is web development including technologies such as TypeScript and HTML5.

In this case, as it is a text-processing oriented game and does not require very complex graphic processing, web technology has been chosen for the development using the Angular framework. Furthermore, as this technology is accessible from any browser, it facilitates access from any computer available in educational centers.

This web application communicates in real-time with an Apache server that implements a REST API through the Symfony framework, and the data is stored in a MySQL database. Customized implementation of the logging-storage system has been chosen due to the positive previous experiences on similar projects and the possibility of data integration between Computational Thinking tools for the personalization of the learning process.

### 3.3 Game evolution

The Lempel platform has gone through 3 different versions until it reached the currently available one.

First version (40 levels, 2 tutorials). This version started with no real-time feedback about the results of the game, participants had to validate their solutions for it. After the first experimentations (136 participants), only 58% of the participants reached level 20. Moreover, the ratio of correct solutions only reached 30% at that level. Therefore, it was concluded that the levels were not well designed to follow an increasing difficulty path and therefore, some concepts were not being correctly understood by the participants.

Second version (40 levels, 4 tutorials). This version of the game started with 2 new tutorials from level 20 onwards. After piloting this version (114 participants), it was observed that the results of players reaching level 20 had improved substantially, reaching 93%. This was due to a better understanding of the higher levels through the tutorials and the redesign of the predecessor levels. Even so, we detected that there was still a gap in players reaching the higher levels (53% at level 26).

Third version (56 levels, 5 tutorials). This version introduces new levels between 20 and 40. In addition, levels are reordered complying with the heuristic of leveling previously mentioned. This version includes real-time scoring of game status and changes in the initial chain to observe what is happening in each move. It includes also compression efficiency limits, in which the user must compress a minimum of half of the best solution, and scoring stars, in which the user can see how he is performing the level in real-time and thus be able to rectify if his solution is not the most appropriate.

## 4 Experimentation methodology

The Learning Analytics process, as mentioned above, consists of the following phases: Measurement, Collection, Analysis and Reporting of the data. Once the development of the platform has been completed, the experimentation phase begins in which its usage data are collected and analyzed for subsequent decision making.

### 4.1 Materials & Tools

The aforementioned tool Lempel has been integrated into Kodetu platform (https://kodetu.org) - a platform that integrates several tools related to Computational Thinking. It allows the management of groups for the different experimentations, employing unique access codes for each group, and a common access/registry for different tools.

The different game levels are organized as follows. Each group of levels is preceded by a tutorial. Levels 1-10 are composed of simple patterns and introduce the user to the game. Next, levels 11-18 combine the patterns of the previous levels with characters that do not form a pattern. In levels 19-26 we find two patterns in each chain and as in the previous ones, in levels 27-30 we find these combined with characters that are not part of the pattern. Levels 31-36 introduce the registers that are used to call other ones (recursion). Finally, levels 37-40 introduce 2 patterns and registers with a combination of characters and calls to other registers. At this point, the player will have worked through all the Pattern Recognition techniques and will find the more complex levels 41-56 with up to 3 and 4 patterns.

### 4.2 Participants

By June 2021, 16 experiments have been carried out by inviting secondary and high school students to activities organized by the Faculty of Engineering of the University of Deusto. After performing the cleaning of test data and erroneous users, a total of 337,231 interactions have been recorded from 393 participants between the ages of 13 to 16 years old (Mean: 14.46, SD: 1.16, Girls: 47.07%, Boys: 47.84%, Others: 5.09%, Workshops: 16).

Participants have been divided into 4 different groups: A) Participants with neither limit on compression nor stars during the game (65 participants, Mean: 14.32, SD: 1.19, Workshops: 3), B) Participants with 50% limit on compression but no stars during the game (107 participants, Mean: 14.65, SD: 1. 09, Workshops: 4), C) Participants with no limit in compression but with stars during the game (106 participants, Mean: 14.58, SD: 1.06, Workshops: 4), D) Participants with 50% limit in compression and stars during the game (115 participants, Mean: 14.27, SD: 1.25, Workshops: 5).

### 4.3 Experimentation procedure

At the beginning of each session, each group of participants was assigned to an experimental group and was informed about the objective and their voluntary participation in the session. To ensure that all players correctly accessed their game session, they were given a 5-letter group code that would show them only their game version. Before showing the game, participants were asked about their demographics (age, gender, education level), whether they knew how to program before the workshop (yes or no), whether they have played Kodetu before (yes or no), and their like for technology (1-min to 10-max).

Once the initial questionnaire is completed, the player is introduced to the game and the game procedure is explained. Upon completion of each level, if it is partially correct the player has the opportunity to improve it or continue, and if it is perfect, the next level is shown until the end of the game.

The experimentations lasted 60 minutes, of which 15 minutes were used for the general explanation of the game and 45 minutes for playing on their own.

### 4.4 Preliminary results

After obtaining the logs generated by the application, a preliminary comparison has been made to observe the performance of the participants in the different versions.

First, we analyzed the achievement level of participants. All participants achieved level 18, which is the last level with one register. In the following levels, as difficulty increases, participants were dropping out accordingly (Fig. 2).

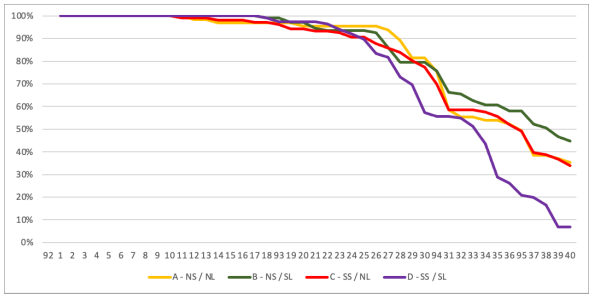


**Fig. 2** Permanence rate through levels

We analyzed the success percentage of participants (Fig. 3). A level is successful if the participant achieves the best possible solution. They have the opportunity to improve their solutions if they are not introducing the best one, so that result is the last solution proposed. As it can be observed, the group with limits and stars (D) has the best performance maintaining its success rate always upper than 85%.

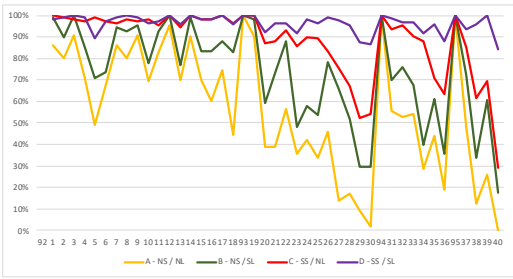


**Fig. 3** Success rate on 1-40 levels

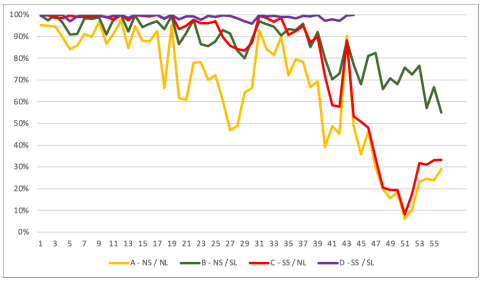


**Fig. 4** Quality of the solution (avg.)

Concerning this analysis, we have represented in Figure 4 the quality of the given solutions by each user. The quality is the percentage of compression being 0% the worst correct solution and 100% the best solution. We found that on levels 1-18 percentages are maintained in 99% on groups with stars (C, D), group D continues on this trend till level 44 (where the last participant arrived). On the contrary, group C continues a similar approach to group B from level 19, where 2 register levels start. Group A has an average quality on those levels of 89% and group B of 96%. Taking into account levels 1-40, group A, with no limits neither stars, is the worst one with an average of 79%, continued by group B with 93%, group C with 95%, and on top group D with 99%.

In addition, we observed that there are levels (like 5 and 10) where there are quality decreases in the easiest levels. This matches with the levels at which new concepts, such as patterns with repeated characters, characters that do not belong to patterns, etc. are introduced.

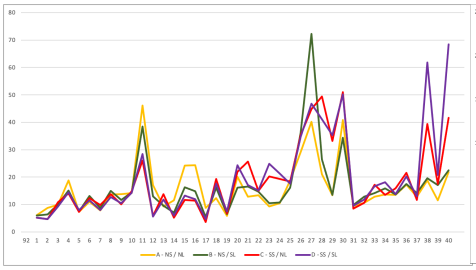

**Fig. 5** Interactions per user (avg.)

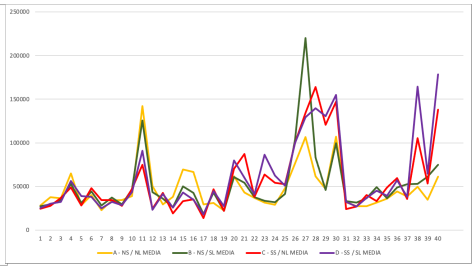

**Fig. 6** Time to resolve (avg.)

We also analyzed the interactions per user (Fig. 5) and the time they need to resolve each level (Fig. 6). As it can be observed, both indicators follow a similar correlation. We found that all groups follow a similar average of interactions and time. The difference is remarkable on most difficult levels, or in those where new concepts are introduced. On levels 1-18, the worst performing group is A with an interaction average of 14,72 and level completion average of 46 seconds, continued by group B with 12,72 interactions and 41 seconds, group C with 11,09 interactions and 36 seconds, and finally group D with 10,97 and 37 seconds. Groups that include stars (C, D) have similar results both on interactions and level time.

## 5 Conclusions

The present work provides an educational tool, LEMPEL, which allows us to understand and analyze how learners acquire knowledge about certain computational thinking skills, such as pattern recognition. This platform integrates into a data compression game a fine-grained logging system, which allows us to register each of the events that students trigger on the platform. The information captured allows us to apply learning analytics techniques to evaluate the development of the different competencies.

An exhaustive analysis of the interactions logged on the platform is currently underway. Preliminary results indicate that the stars and solution quality limit included in the game and the improvements in the leveling are an aid to improve performance during the learning path. In addition, as future work we will carry out statistical analysis to analyze the performance in pattern recognition taking into account the user's characteristics, level types, etc.

## References

1. Anderson,L.W., Krathwohl, D.R.,Airasian, P.W.,Cruikshank, K.A., Mayer, R.E., Pintrich, P.R., Raths, J., Wittrock, M.C. (2001). A Taxonomy for Learning, Teaching, and Assessing: A revision of Bloom's Taxonomy of Educational Objectives.
2. J. M. Wing, “Computational thinking,” Communications of the ACM, vol. 49, no. 3, pp. 33–35, 2006.
3. J. M. Wing, “Computational thinking and thinking about computing,” Philosophical Transactions of the Royal Society A, vol. 36, no. 1881, pp. 3717–3725, 2008.

4. Werner, L., McDowell, C., & Denner, J. (2013, March). Middle school students using Alice: what can we learn from logging data? In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 507-512). ACM.
5. Piech, C., Sahami, M., Koller, D., Cooper, S., & Blikstein, P. (2012, February). Modeling how students learn to program. In Proceedings of the 43rd ACM technical symposium on Computer Science Education (pp. 153-160). ACM.
6. Seiter, L., & Foreman, B. (2013, August). Modeling the learning progressions of computational thinking of primary grade students. In Proceedings of the ninth annual international ACM conference on International computing education research (pp. 59-66). ACM.
7. Tedre, Matti, and Peter J. Denning. 'The Long Quest for Computational Thinking'. In Proceedings of the 16th Koli Calling International Conference on Computing Education Research, 120–129.
8. Eguiluz A, Garaizar P, Guenaga M (February 14th 2018). An Evaluation of Open Digital Gaming Platforms for Developing Computational Thinking Skills,
9. Dragan Cvetković. Simulation and Gaming, IntechOpen,
10. Siemens, G., Gašević, D., Haythornthwaite, C., Dawson, S., Buckingham Shum, S., Ferguson, R., Duval, E., Verbert, K., & Baker, R. S. (2011). *Open Learning Analytics: an integrated modularized platform*.
11. U. K. Mothukuri et al., "Improvisation of learning experience using learning analytics in eLearning," 2017 5th National Conference on E-Learning & E-Learning Technologies, Hyderabad, 2017, pp. 1-6.
12. Paulo Blikstein. 2011. Using learning analytics to assess students' behavior in open- ended programming tasks. In Proceedings of the 1st Int. Conf. on Learning Analytics and Knowledge (LAK '11). ACM, New York, NY, USA, 110-116.
13. Felder, R. M. and Brent, R. (2005), Understanding Student Differences. Journal of Engineering Education, 94: 57-72.
14. A. Dasgupta and S. Purzer, "No patterns in pattern recognition: A systematic literature review," 2016 IEEE Frontiers in Education Conference (FIE), 2016, pp. 1-3,
15. Bennett, J., & Müller, U. (2010). The development of flexibility and abstraction in preschool children. Merrill-Palmer Quarterly, 56(4), 455-473.
16. Dasgupta, Annwesa & Rynearson, Anastasia & Purzer, Senay & Ehsan, Hoda & Cardella, Monica. (2017). Computational Thinking in K-2 Classrooms: Evidence from Student Artifacts (Fundamental).
17. Gentner, D., & Lowenstein, J. (2002). Relational language and relational thought. In E. Amsel & J. P. Byrnes (Eds.), Language, literacy, and cognitive development: The development and consequences of symbolic language (pp. 87–120). Mahwah, NJ: Erlbaum. 14
18. Eguiluz A, Guenaga M, Garaizar P, Olivares-Rodriguez C (2017) Exploring the progression of early programmers in a set of computational thinking challenges via clickstream analysis. IEEE Transactions on Emerging Topics in Computing PP(99):1–1.
19. I. Kanellopoulou, P. Garaizar and M. Guenaga, "First Steps Towards Automatically Defining the Difficulty of Maze-Based Programming Challenges," in IEEE Access, vol. 9, pp. 64211-64223, 2021.
20. Grover S, Basu S (2017) Measuring Student Learning in Introductory Block-Based Programming: Examining Misconceptions of Loops, Variables, and Boolean Logic. In: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. ACM, New York, NY, USA, pp 267–272