ACTA DIDACTICA NAPOCENSIA
Volume 17, Number 1, 2024 - DOI: 10.24193/adn.17.1.6

ALSORT – A SOFTWARE INSTRUMENT FOR LEARNING SORTING ALGORITHMS

Mariana-Ioana MAIER

Abstract: Finding methods to teach sorting algorithms is a challenge for most Computer Science teachers, but also for students who are open to learn and search for information. Because nowadays students are very receptive to educational software, we want to introduce our instrument named *ALSort* – a software designed for teachers and students interested in the instructional process of sorting algorithms. *ALSort* is an educational software based on gamification and modelling. It is built on six stages of challenges related to the six levels of learning from the Revised Bloom Taxonomy and the version from this paper is a prototype.

Key words: sorting algorithm, gamification, modelling, educational software

1. Introduction

Sorting is a fundamental concept that can be introduced to kids in a fun and engaging way (Bernát, 2014). It lays the foundation for developing logical thinking and problem-solving skills. Sorting for children is typically defined as the process of arranging items or objects into groups based on specific attributes or characteristics (Linder, Powers-Costello, & Stegelin, 2011). This foundational concept helps children develop early math and cognitive skills, including categorization, pattern recognition, and logical thinking (Platz, 2004). The primary goal is to teach children how to organize and make sense of the world around them by recognizing similarities and differences among objects (Fromboluti & Rinck, 1999). Sorting activities for children often involve hands-on experiences using toys, everyday objects, or specially designed materials. These activities aim to make learning enjoyable and interactive, helping children develop critical skills that form the basis for more advanced mathematical and analytical thinking in their later education. Introducing sorting algorithms to children can be a bit advanced, as these algorithms involve more abstract concepts and often require a level of mathematical and logical understanding that might be beyond the grasp of very young children. However, we can teach basic sorting concepts in a simplified and fun way (Bernát, 2014).

With many practical applications in our daily lives, data sorting acts as a fundamental activity in computer science. Consequently, a multitude of methods for improving the performance of sorting algorithms have been reported for both desktop computers (Hoare, 1962) and mobile devices (Lee, Lee, & Park, 2015). We want a thorough comprehension of the idea as well as soft abilities to create effective algorithms in this field in order to guarantee the ongoing advancement of these algorithms. These abilities are acquired at several learning stages (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956), and mastery of these phases is necessary for competent practitioners.

This paper aims to continue the work presented in Maier, Șerban, & Moisin (2022), where we have started to mine the sorting concept across curriculum levels in Romania.

The paper is organized as follows: Section 1 is an overview of the present article. In the Section 2, we present a brief incursion into the literature regarding pedagogical instruments used in learning sorting algorithms. Section 3 presents our work's background, whilst Section 4 describes our pedagogical instrument, named *ALSort*, with its stages of challenges. Section 5 presents our conclusions and future work.

Received April 2024.

Cite as: Maier M. I. (2024). ALSort – A Software Instrument for Learning Sorting Algorithms. *Acta Didactica Napocensia*, 17(1), 73-78, https://doi.org/10.24193/adn.17.1.6.

2. Related work

Learning taxonomies is used to guide the instructional design, because the information and competences are acquired in different stages. In the cognitive field, Revised Bloom Taxonomy (RBT) is very popular, with its six stages of learning: *remember, understand, apply, analyze, evaluate,* and *create* (Krathwohl, 2002).

Finding methods for teaching sorting algorithms is a concern for instructors both from Mathematics and Computer Science domains. The main focus is on the following algorithms: Selection Sort, Bubble Sort, Insertion Sort, Merge Sort and Quick Sort.

Bernát (2014) presents his strategies in this direction through demonstration method: with pictures, with animations, and with simulation programs. Kátai (2021) is concerned with multi-sensory computer science education, and he presents sorting algorithms through dances. In every dance, the dancers represent the objects to be sorted and the choreography follows the steps from the sorting algorithm.

For the mobile platforms, some software applications were introduced, like *Sortko* (Boticki, Barisic, Martin, & Drljevic, 2012), or the framework meant to demonstrate sorting algorithms introduced by (Meolic, 2013).

Visual Sorting is a software that illustrates the steps from a few sorting algorithms and the user has the opportunity of inter-active tracking of the performance for the presented algorithms (Mavrevski, Traykov, & Trenchev, 2019).

We notice that the educational methods based on presenting the algorithms directly by an instructor or through technology are very well represented in the literature. This valuable work covers the first level of learning from the RBT: *remember*. Also, we have some work in tracking the steps of different sorting algorithms, which covers the second level of learning from RBT: *understand*. For the next levels of learning, there are no relevant results.

We intend, in this paper, to introduce educational software which covers every level of learning from the RBT and briefly present its levels of challenges.

3. Background

Through an analysis of the sorting notion in the Romanian Computer Science curriculum, we have observed that the teaching of sorting algorithms is not uniform. While some secondary school students have the opportunity to learn about this subject in depth, others do not. This explains why the approach of sorting depends on the high school computer science class. Depending on their area of specialization, some students at this level enhance their skills, while other students simply have a surface-level understanding. Only students from Real Sciences classes learn sorting algorithms, but any high school graduate can attend a Computer Science program at the college. As a result, the college students' programming abilities varied greatly from one another.

In the paper “Mining sorting concept across curriculum levels: a cyclic learning-based approach” (Maier, Șerban, & Moisin, 2022), we presented: (1) how is the sorting problem approached in the Computer Science curriculum across three levels of study, (2) the perception of Computer Science teachers regarding the sorting’s teaching, and (3) an idea for a pedagogical instrument designed to help students in learning sorting algorithms. That pedagogical instrument was named *ALSort* and it was designed according with the six levels of learning from the RBT. *ALSort* is intended to fill the gap related to sorting algorithms for those who want to study Computer Science. This tool was built from a Computer Science perspective, but we also want to help children in developing logical and algorithmic skills while they learn sorting. In this regard, we present in the next section the challenges levels from *ALSort* both from mathematical and Computer Science perspectives.

4. *ALSort* – a pedagogical software for learning sorting algorithms

We concluded in Maier, Șerban, & Moisin (2022), that *educational software*, *didactic game* and *discovery learning* have important roles in teaching sorting algorithms at secondary and high school,

so there is a need for efficient tools in the instructional process. *ALSort* is a gamification and modelling-based teaching project, built in *Unity Editor 2020.3.32f1*. It attempts to cover the six RBT levels at various educational stages based on students' cognitive abilities. As a result, we have the opportunity to establish the foundation for sorting algorithms in middle school by addressing the RBT *remember* and *understand* levels. We can carry on our work throughout high school by assigning tasks that support learners in reaching *apply* and *analyze* levels. Students who complete the final two RBT levels (*evaluate* and *create*) - at the university have the chance to hone their sorting algorithms-related knowledge and abilities.

In what follows, the challenges levels from *ALSort* are presented.

4.1. *AL's Organizer* level

AL's Organizer is the first level of the application. It is based on discovery learning. The user can choose the type, number, and order for sorting a set of objects. Then, some items are generated on the screen and the user has to sort them, based on the chosen properties. Thus, children who don't know the sorting have the opportunity to discover how a sorted collection is obtained, and those who know the sorting can remember it, such that, at the end of this stage, the *remember* level from RBT is achieved. In the Figure 1 are presented three print screens from this level. In the first image, the user chooses the details for the sorting, in the second image we have a print screen from user's experience after choosing to sort 10 animals in alphabetical order, and the third image represents a print screen where the user failed to sort a collection of numbers, but and is encouraged to try again.

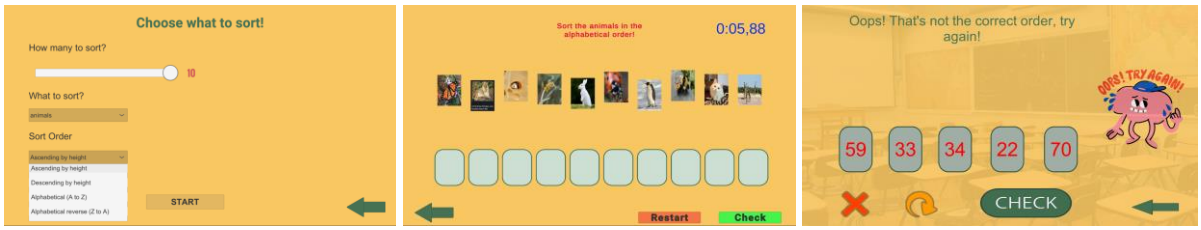

Figure 1. *Print screens from AL's Organizer level*

4.2. *AL's Friend* level

AL's Friend is built with the goal of understanding some sorting algorithms. For the moment, three of them are targeted: Bubble Sort, Selection Sort, and Insert Sort, but we work to have more sorting algorithms. For each of them, are provided hyperlinks to the documentation where the algorithm is presented through text or through dance (which represents the modelling of sorting algorithms), such that the user has easy access to the *remember* level for the studied algorithm. After the visualisation of the documentation, the user can practice what he/she learned by applying the algorithm on a set of numbers. Thus, he/she has to sort ascending a set of numbers. For every move, the user gets feedback from the application and, when the sorting is done, the number of mistakes is displayed.

From the Computer Science perspective, this level covers *remember* (through the documentation for each algorithm) and *understand* (through sorting the set of numbers) levels of learning from RBT.

From logical and mathematical perspective, *remember* level is covered by the text documentation, *understand* level is achieved by dance visualisation, and *apply* level is achieved by the challenge to sort the set of numbers. Figure 2 contains three images, representing print screens from this level. In the first image is presented the user's experience with Bubble Sort algorithm. The second image is a print screen while learning Insertion Sort algorithm. The third image presents the user's experience with Selection Sort algorithm. A video of this last experience is presented at (Maier, 2024a).

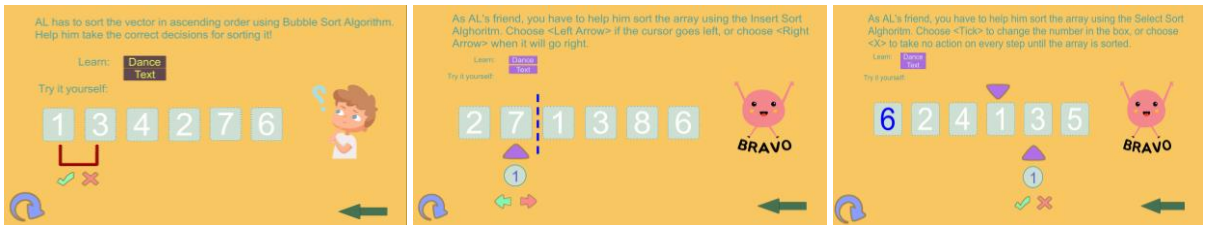

Figure 2. Print screens from *AL's Friend level*

4.3. *AL's Programmer* level

At this level, the user receive some blocks of instructions from pseudocode with the aim to design a given sorting algorithm.

From the Computer Science perspective, this stage is correlated with *apply* level from RBT, because the user has to use the information form the previous level to solve a task.

From logical and mathematical perspective, the user has to *analyse* the steps presented in the previous level and abstractize them, to be correctly translated in pseudocode. Figure 3 contains two images, representing print screens from this level. In the first image is presented the user's experience with Selection Sort. The second image presents the user's experience with Bubble Sort. A video from this last level is presented at (Maier, 2024b).

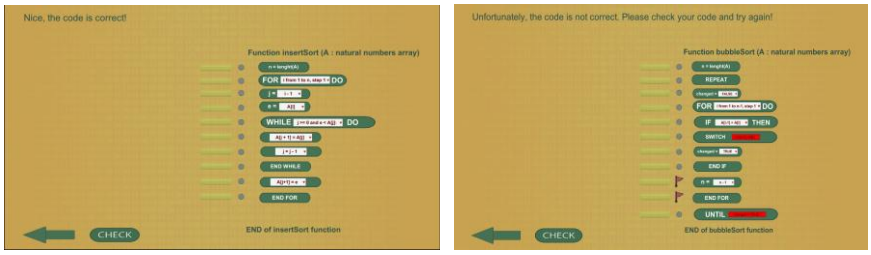

Figure 3. Print screens from *AL's Programmer* level

4.4. *AL's Debugger* level

At the fourth level, the user has to debug some given algorithms which contain sorting concept. These algorithms are prepared by teachers who have the rights to upload files in the application, and the users receive some blocks of pseudocode instructions which form an algorithm. When a student starts this level, he/she gets the challenge to debug a given algorithm, with the aim to find a correct form of it. For more interactivity, the user can choose how many mistakes will have the given algorithm. When the user thinks that the problem was solved, he/she presses the *Check* button. If the algorithm still has errors, the wrong instructions will be colored in red, like we can notice in the first image from the Figure 4. Otherwise, if the challenge is correctly solved, the user sees on the screen a correct form of that algorithm in pseudocode and how many times he/she pressed the *Check* button to verify the status of that algorithm, like we can notice in the second image from the Figure 4.

At this level, we have targeted the *analyse* level from RBT, because it's necessary to study how are connected the blocks of instruction and which are the factors that make them function together. Also, the *evaluate* level is reached when the users are able to monitor their work and take decisions to determine how to solve the problem. These RBT levels are reached both for Computer Science and mathematical perspectives.

Figure 4. Print screens from *AL's Debugger* level

4.5. *AL's Evaluator* level

The fifth level is focused on the time complexity of sorting algorithms. The user can study the complexity of each sorting algorithm from this stage by clicking the name of the algorithm. Also, general and useful information about time complexity is accessible by clicking on the text "Time complexity".

After algorithms complexity is studied, user has to choose the correct complexity for every sorting algorithm from a list. The complexity is chosen for best case, worst case, average case and total. This stage of *ALSort* is related with the *evaluate* level from RBT, because it's necessary to detect the number of steps made by each given algorithm and integrate this number in a complexity class.

Figure 5 represents a user's experience at this level for five sorting algorithms. After the user had chosen the complexity class, he/she can find how many right and wrong answers were given and which are these.

4.6. *AL's Assistant* level

The sixth level is related with the *create* level from RBT. An artificial intelligence component is integrated at this stage. At this level, the user has some interesting options, like: (a) create his/her own sorting algorithms, to test them and evaluate their complexity; (b) debug an algorithm; (c) ask questions, like a sorting problem statement with some tips for solving it. Having these options, the developing of the user's creativity is assured. Figure 6 presents a print screen from AL's Assistant level, where the user the last option.

Figure 5. Print screen from *AL's Evaluator* level

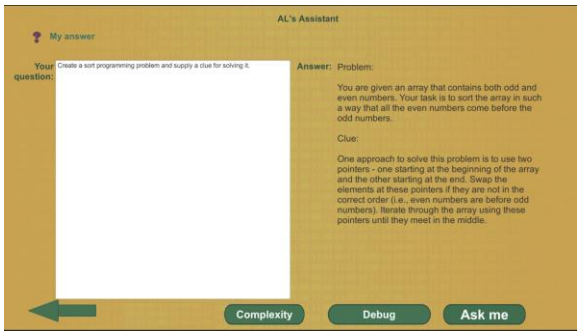

Figure 6. Print screen from *AL's Assistant* level

5. Conclusions and future work

In this work, we have presented a pedagogical instrument for learning sorting algorithms, named *ALSort*. It is implemented as an educational software based on gamification and modelling with six levels of challenges related with the six levels of learning from RBT. This instrument has advantages both for students and teachers, because the teachers can integrate activities related sorting algorithms for students, and students have the possibility to study and exercise sorting algorithms in their own pace.

As future work, we want to validate *ALSort* by students from different levels of learning and to enrich the list of sorting algorithms offered for learning. Also, we intend to implement an artificial

intelligence component, such that the software could fold on the user's profile. Thus, the instructional process would be customized for each user, which would streamline the learning. After we reach these targets, we intend to release our instrument as an open access application.

References

Bernát, P. (2014). The methods and goals of teaching sorting algorithms in public education. *Acta Didactica Napocensia*, 1-9.

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). Taxonomy of educational objectives: The classification of educational goals. În *Handbook I Cognitive domain.* New York, David McKay Company.

Boticki, I., Barisic, A., Martin, S., & Drljevic, N. (2012). Sortko: Learning Sorting Algorithms with Mobile Devices. *2012 IEEE Seventh International Conference on Wireless, Mobile and Ubiquitous Technology in Education*, (pg. 49-56).

COUNCIL RECOMMENDATION of 22 May 2018 on key competences for lifelong learning. (2018). *Official Journal of the European Union*.

European Commission and Directorate-General for Education, Y. S. (2019). *Key competences for lifelong learning.* Publications Office.

Fromboluti, C. S., & Rinck, N. (1999). *Early Childhood: Where Learning Begins: Mathematics: Mathematical Activities for Parents and Their 2-to 5-year-old Children.* US Department of Education, Office of Educational Research and Improvement, National Institute on Early Childhood Development and Education.

Hoare, C. A. (1962). Quicksort. *The Computer Journal*, 10-16.

Kátai, Z. (2021). *Algorythmics. Technologically and artistically enhanced computer science education.* Sapientia Könyvek.

Krathwohl, D. R. (2002). A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice*, 212-218.

Lee, S., Lee, E., & Park, J. (2015). An optimal sorting algorithm for mobile devices. *Indian Journal of Science and Technology*, 226–232.

Linder, S., Powers-Costello, B., & Stegelin, D. (2011). Mathematics in Early Childhood: Research-Based Rationale and Practical Strategies. *Early Childhood Educ*, 29–37.

Maier, M., Șerban, C., & Moisin, A. (2022). Mining sorting concept across curriculum levels: a cyclic learning based approach. *Proceedings of the 4th International Workshop on Education through Advanced Software Engineering and Artificial Intelligence*, (pg. 10-17).

Maier, M., (2024a). *Activity from level (2) AL's Friend of ALSort instrument.* https://drive.google.com/drive/u/0/folders/1LzbDx2loGIMQI_Z5SxjwvF5kHH-89EwC

Maier, M., (2024b). *Activity from the level (3) AL's Programmer in ALSort instrument.* https://drive.google.com/file/d/1HSELK_2rP_QA5ndxGH60epM6whF-odrz/view?usp=sharing

Mavrevski, R., Traykov, M., & Trenchev, I. (2019). Interactive Approach to Learning of Sorting. *International Journal of Online and Biomedical Engineering (iJOE)*, 120–134.

Meolic, R. (2013). Demonstration of Sorting Algorithms on Mobile Platforms. *CSEDU* , (pg. 36-141).

Platz, D. L. (2004). Challenging young children through simple sorting and classifying. *Education, 125*(1), 88-96.

Author

Mariana-Ioana MAIER, Babeș-Bolyai University, Cluj-Napoca (Romania). E-mail: mariana.maier@ubbcluj.ro, https://orcid.org/my-orcid?orcid=0000-0003-3905-5343