International Journal of Instruction *April 2021 ● Vol.14, No.2*
e-ISSN: 1308-1470 ● www.e-iji.net *p-ISSN: 1694-609X*
pp. 27-44

Article submission code: 20200305042001
Received: 05/03/2020 Revision: 22/08/2020
Accepted: 14/09/2020 OnlineFirst: 03/01/2021

Improving AlgoRythmics Teaching-Learning Environment by Asking Questions

Zoltán Kátai
Sapientia Hungarian University of Transylvania, Romania, *katai_zoltan@ms.sapientia.ro*

Erika Osztián
Sapientia Hungarian University of Transylvania, Romania, *osztian@ms.sapientia.ro*

In this paper the challenge of promoting computational thinking for all by contextualized computing education is addressed. The two phases learning session we designed was implemented in the AlgoRythmics environment which includes ten algorithmic dance choreographies (and attached interactive computer animations). In addition to previous studies that had focused only on supporting students in assimilating the strategy of the studied algorithms, this study examined whether they are able to build on this knowledge by extracting from visualizations some algorithm efficiency related concepts too. To this end, the learning environment was complemented/supplemented with targeted questioning (with and without teacher guidance). Participants (first year undergraduate students) were grouped based on the number of years they had learned programming in high school (0, 1/2 or 4 years). We performed two analyses: (1) group-0 vs. group-1/2 and group-4 (N=181; self-paced setting: algorithm visualization complemented with teacher prepared questions); (2) group-0 (one-group pretest-posttest design, N=46; supplementary Socratic questioning). Research results revealed that the AlgoRythmics environment, if complemented/supplemented with tutorial question-asking, could be an effective instrument in introducing students with no prior knowledge in computing (group-0), even with deeper Computer Science concepts such as algorithm efficiency.

Keywords: instruction, learning environments, computational thinking, algorithm visualization, questioning

INTRODUCTION

Nowadays it is commonly accepted that computational thinking (CT) is an essential mindset for all students of the digital era. Although the phrase computational thinking was introduced by Seymour Papert (1980), it was brought to the forefront of the Computer Science (CS) education community only more than 25 years later by Jeannette

Citation: Kátai, Z., & Osztián, E. (2021). Improving AlgoRythmics Teaching-Learning Environment by Asking Questions. *International Journal of Instruction*, 14(2), 27-44. https://doi.org/10.29333/iji.2021.1423a

Wing (2006). Wing describes CT as a formative skill on a par with reading, writing and arithmetic, and emphasizes that everyone, not just computer scientists, would be eager to learn and use. Since the concept of CT can be defined as the thought process involved in formulating problems so that their solutions can be represented as computational steps and algorithms (Aho, 2012), a natural way to address the CT for all issue is computing education for all. For example, Echeverría et al. (2017) emphasize that computing is a skill required for any engineering field.

A possible approach to make computing education attractive for different category of learners (including K-12 learners and non-CS majors) is contextualization (Guzdial, 2010). For example, in the case of non-CS majors the context should be related to the major field of the students. Since developing differentiated teaching–learning strategies may involve substantial additional costs, some scholars have tried to find a context that is appealing to most students. A promising candidate for this "common denominator role" could be arts (Tew et al., 2005; Guzdial & Tew, 2006; Simon et al., 2010; Daily et al., 2014; Wood et al., 2016). The AlgoRythmics learning environment (Katai et al., 2020) was designed along this approach. Since music and dance are relatively close to most young people, this environment visualizes basic computer algorithms (searching and sorting) by professional dance choreographies. In addition, to each dance choreography an interactive computer animation was attached. The videos are also accessible by the AlgoRythmics YouTube channel and they have got millions of views (Katai & Toth, 2011; Katai et al., 2018).

In this paper we address the following issue: what is the best practice for incorporating the AlgoRythmics environment, and other similar environments, in the process of teaching and learning algorithms. Modern learning theories emphasize the superiority of those teaching approaches that facilitate student-centered learning (Tamim & Grant, 2013; Wijnen et al., 2017). All previous AlgoRythmics studies implemented this principle by using only self-paced learning settings (without any teacher intervention). On the other hand, our experience (as CS teachers) with this environment (and the feedback from many colleagues) made it clear to us that without teacher support the potential these visualizations incorporate, as CT promoter tools, can only be partially exploited. For example, definitions of CT often present the concept of algorithm efficiency as an important component of this ability (Shute et al., 2017; CSTE, 2020). It is hard to imagine that students without any prior experience in computing would be able to "discover" the best case and worst case behavior of the algorithms without teacher guidance.

An effective teacher practice that harmonizes with the principle of student-centered learning could be that of asking effective questions during student problem solving (Boyer et al., 2010). Accordingly, in this study we analyze a learning setting built around AlgoRythmics visualizations, where the principle of active involvement is implemented by question asking: with teacher guidance (teacher guided Socratic questioning) and without teacher guidance (students are asked to answer teacher prepared questions in a self-paced way).

The majority of previous studies that investigated question-asking from instructional perspective (in the context of CS education) focused on supporting students in coding (programming, data structures, etc.). For example, Boyer et al. (2010) analyzed the efficiency of one-on-one tutoring approach during a problem solving process that involved applying array data structures and for loops. In the study performed by Lane and Van Lehn (2005) the content to be assimilated was control flow (conditionals and loops) and how to write simple subprograms. These authors conclude that asking effective questions during the early phases of planning a solution can support the students' comprehension and decomposition of the problem at hand. Tenenberg and Murphy (2005) found that asking specific questions is useful for revealing knowledge gaps with novices, who are often unable to articulate their questions. The administered quiz aimed to assess students' data structures knowledge. Razmov and Anderson (2006) examined the value of open-ended questions in the context of an undergraduate course in software engineering. In addition to these studies, we have proposed to examine the effectiveness of question-asking in the context of algorithm visualization with respect to undergraduate students without any prior knowledge in programming.

Previous work

The AlgoRythmics project was initiated more than ten years ago. The first six videos (sorting strategies illustrated by folkdance performances) were posted on the AlgoRythmics YouTube channel in 2011. After this a web application was developed and to each dance choreography an expressive interactive computer animation was attached. The principle of active involvement was implemented by inviting students to reconstruct/orchestrate the studied algorithm for different inputs (to interactively predict the corresponding operation sequence). In 2018 four new videos were added to the AlgoRythmics collection (heap-sort illustrated by a new folk dance choreography; linear and binary search illustrated by flamenco dances; the backtracking solution for the four-queens problem illustrated by a ballet performance) and the web application was redesigned and extended with new modules.

Previous research on the AlgoRythmics environment (Katai, 2014a, 2014b, 2015, 2020) concentrate mostly on the potential incorporated in the dance choreographies (supplemented with computer animations) to support different categories of students in understanding the strategy of the algorithms. Research results confirmed this potential. For example, two studies addressed the issue of promoting the CT of both science and humanities oriented students. These investigations resulted in the following conclusions: (1) properly calibrated learning environments have the potential to effectively promote the CT of both sciences-oriented and humanities-oriented students (Katai, 2015); (2) although sciences-oriented students' motivational-scores were consistently superior to their humanities-oriented colleagues, there was strong correlation between them and differences diminished as both groups advanced with their learning tasks (Katai, 2020).

None of the previous research has examined if students are also able to build on the knowledge they have acquired (understanding the strategy of the studied algorithm). Consequently, in this study we investigate the following questions, with respect to a corresponding two phase (1: self-paced; 2: teacher guided) learning session:

- (RQ-1, phase-1) Are AlgoRythmics visualizations expressive enough, when complemented with targeted questions, to support students without prior knowledge in computing to imagine the best and worst case behaviour of algorithms?
- (RQ-2, phase-2) Can a supplementary teacher-guided, question-and-answer session (Socratic questioning based on the phase-1 questions) contribute significantly to students' phase-1 understanding?

Asking Questions to Support Learning with Visual Representations

The instructional method we designed and implemented is grounded in the Cognitive Apprenticeship Theory (Collins et al., 1991; Caspersen & Bennedsen, 2007). Cognitive Apprenticeship uses guided learning environments to support students in assimilating the learning content. An important characteristic of this approach is that support is reduced so that students can apply their acquired knowledge and skills as independent learners (Ghefaili, 2003). An effective method for implementing this moderate support could be the asking of targeted questions.

In line with prior research in the field, tutorial question asking could prove to be an efficient method of engaging students in meaningful learning (Graesser & Person, 1994). Boyer et al. (2010) reports on the positive results of asking effective questions to support students' learning in a CS educational context. These authors emphasise that the effective use of questions has the potential to prompt students to engage in valuable learning behaviours that they might not otherwise have undertaken. For example, it facilitates comprehension, encourages self-explanation, and reveals incomplete or incorrect knowledge. With respect to Socratic questioning, El-Zakhem (2016) draws attention to its role in encouraging critical thinking through rational arguments.

Tawfik et al. (2020), also emhasize that an important component of knowledge construction during problem-solving is the ability to ask meaningful questions. Boyer et al. (2010) suggest that studying student questions and investigating the impact of instructor questions should be considered complementary lines of research. After they have revised several existing theories in the field (role of questions in inquiry-based instruction), Tawfik et al. (2020) conclude that many models focus on how to elucidate and later replicate the expert reasoning process for novices within learning environments. Attaching a teacher prepared question sequence to a self-paced learning session can be seen as a subtle method for aligning the way of thinking of a novice to the reasoning of an expert.

In addition, as mentioned above, studying algorithms with the AlgoRythmics environment falls in the framework of contextualized computing education. Besides its evident advantages, Guzdial (2010) draws attention to possible negative side effects of contextualization. For example, overemphasizing a context can obstruct knowledge transfer. Although the AlgoRythmics environment involves the context only at the level of appealing decorative elements, the dangers of distraction still persist. Prior research shows that the effectiveness of learning with visual representations critically depends on

students' ability to make sense of the corresponding visual educational materials (Wu & Rau, 2018). Several studies conclude that effective instructional activities could support students in realizing how visual representations depict information about the content (Ainsworth, 2006; Rau, 2016). Without this support, students often focus on irrelevant surface features and fail to depict domain-relevant concepts (especially in the case of decorative illustrations) (Wu & Rau, 2018). According to Boyer et al. (2010), asking targeted questions could be an effective method to direct student's attention, for example, toward the relevant aspects of the visualizations.

Tofade et al. (2013) also argue that using questions is an effective way to stimulate the recall of prior knowledge, promote comprehension, and build critical-thinking skills. Well worded questions can stimulate students to think about a topic in a new way. Effective teachers are able to formulate questions to fit the cognitive level of students. Other characteristics of the effective use of questions are: careful phrasing and word clarity; creating a psychologically safe environment; appropriate sequencing and balance; properly calibrated wait time.

Accordingly, we sequenced the algorithms according to the principle of moderate-progressive challenge: linear search, binary search, bubble sort (although bubble sort is not the most intuitive sorting strategy, after being visualized, students usually realize its strategy quite easily). The questions attached to each algorithm were also sequenced according to this principle. In order to implement the suggestion regarding the careful phrasing and word clarity, expressions like "best/ worst case" and "comparing operation" were supplemented with synonyms like "happiest/ most unfortunate case" and "comparing scene", respectively.

METHOD

We designed the following three phase learning-testing session:

1. After a brief introduction participants were invited to watch and analyze the dynamic visualizations of three algorithms (linear and binary search: dance video played twice; bubble sort: dance video plus animation); After each visualization they had to answer algorithm complexity questions;

(The visualizations helped students in realizing the strategies the algorithms apply; The attached question-sequences guided them in imagining the best and worst case behaviour of the studied algorithms);

2. By questioning (using additional leading questions) the instructor helped students realize/discover the right answers to the phase-1 questions;

3. Students were presented with the phase-1 learning conditions again, but built around a new algorithm (selection sort: dance video plus animation followed by algorithm complexity questions).

We tested the effectiveness of this three phase learning unit on first year engineering students without any prior programming experience. We anticipated that

- (Hypothesis-1) participants would assimilate the algorithms at a satisfactory level, even during the first phase of the learning session (as reflected in students' answers to the phase-1 questions);
- (Hypothesis-2) phase-2 instructional intervention will contribute significantly to students' understanding (as reflected in their answers to the phase-3 questions).

We used a quasi-experimental (causal-comparative) research design. The experiment took place at the beginning of the academic year and all first year undergraduate students (181, 87% male) were invited to participate in the investigation. With respect to Hypothesis-1 (for the phase-1 part of the experiment), we identified as independent variable the number of years participants had learned programming in high school: group-0 (0 year, 27%); group-1/2 (1 or 2 years, 25%); group-4 (4 years, 48%). The high school curriculum for group-1/2 included searching and quadratic sorting algorithms at a basic level. Accordingly, these participants had already been introduced with the strategies these algorithms apply, but time complexity related concepts (like best and worst case behaviour) were new to them.

For this phase of the experiment group-0 was the experimental group and group-1/2 the control. We were wondering if AlgoRythmics environment (complemented with a targeted question sequence) has the potential to help group-0 students catch up with their group-1/2 counterparts (group-4 was involved in this analysis as a kind of secondary control group; their high school curriculum included both the algorithms and concepts of complexity). We considered as dependent variable participants' phase-1 performance (based on their answers to the attached questions).

In the second and third phases of the experiment only group-0 was involved. We implemented a one-group pretest-posttest design. Again, the dependent variable was participants' performance (pretest: phase-1 score, posttest: phase-3 score). According to our second hypothesis, we anticipated that after being invited to actively participate in the teacher guided classroom discussion (phase-2), group-0 students would score significantly higher on the third phase questions than they did on the first phase ones.

Materials

Because the particularity of the environment (its uniqueness) lies in the algorithmic dance-choreographies (supplemented with animations), we proposed to test, first of all, the potential these visualizations incorporate. Accordingly, we intentionally created a quite extreme in-class learning environment. Before starting the experiment, the teacher offered students only a minimal algorithmic intro (what an algorithm is and what a searching or comparison-based sorting algorithm/strategy is: succession of comparing or comparing/swapping operations). For the group-0 students this was their first contact with computer algorithms.

Because of their simplicity, searching algorithms were presented only as dance performances (played two times). In the case of the sorting algorithms, first the dance choreography and next the computer animation were played. During the first plays students became familiar with the environment and the algorithm (for example, how

comparing/swapping operations were danced). They were encouraged to focus on that content to which the questions related during the second play (or during the animation in the case of the sorting algorithms). The answers had to be provided only after they had finished viewing the visualization as a system-paced instructional material. An important but surprising finding reported in the (Berney & Bétrancourt, 2016) meta-analysis is that the positive effect of animation over static graphics was found only when learners did not control the pace of the display.

The visualizations illustrate the studied searching/sorting algorithms on given random sequences. As mentioned above, to stimulate participants to reflect on the strategies the algorithms apply, the attached question sequences requested subjects to imagine the best and worst case behaviour of the algorithms. In the description below, question-code Qx.y.z can be interpreted as follows: phase-x, algorithm-y, question-z.

The questions attached to the linear search (Q1.1.1-7) and binary search (Q1.2.1-7) algorithms were the followings:

- (1-3) How many comparison operations (comparing scenes) does the linear (or binary) search algorithm imply (for a list with 7/31/N elements) in the "best case" ("happiest case")?
- (4-6) How many comparison operations (comparing scenes) does the linear (or binary) search algorithm imply (for a list with 7/31/N elements) in the "worst case" ("most unfortunate case")?
- (7) What is the "worst case" with respect to the linear search algorithm?

The question groups appeared on students' questionnaires as individual questions (corresponding to lists with 7, 31 and N elements, respectively). We chose values 7 (2^3-1) and 31 (2^5-1) in order to have clear middle elements with respect to all current sub-sequences during the binary search algorithm.

To the bubble sort algorithm two questions were attached (Q1.3.1-2):

- (1-2) How many comparison operations does the bubble sort algorithm imply (for a list with 10 elements) in the "best case"/ "worst case" (the sequence is already sorted in ascending/ descending order)?

After students answered these questions, they were asked (via an extra synthesis question; Q1.4.1) to consider the relative effectiveness of the studied algorithms:

- (1) When is the "sorting + binary-search" strategy preferable to linear search? (a) for sequences with many elements; (b) if the searching operation has to be performed repeatedly (many times); (c) for sequences with large elements; (d) other?

Some questions and formulas that were analyzed during the second phase of the experiment are the followings:

- What are the worst case scenarios with respect to the linear search algorithm if we have or we do not have a guarantee that the searched element is included in the list?
- How many elements are eliminated from the list ("search space") by a single "you are not the one" dance in the case of the linear/binary search algorithm? After the corresponding binary tree was identified, students were helped to realize the following formulas (k denotes the height of the tree): $2^0+2^1+\ldots+2^{k-1} = N$; $k = \log_2(N+1)$.
- With respect to the worst case behaviour of the bubble sort algorithm the following formula was analyzed: $1+2+\ldots(N-1) = N(N-1)/2$.

Since during the phase-2 discussion we observed that for many students it was not implicitly evident that sorting algorithms in best cases do not perform any swapping operation, and in worst cases obligatorily perform a swap after each comparison operation, questions referring to these cases were also included in the selection sort analysis (phase-3) (Q3.1.1-4).

Procedure

During the first and third phases of the experiment the videos and the computer animations were presented in front of all students (in an amphitheater) from the laptop of the teacher (using a video projector). Students were asked to answer anonymously (indicating only the program where they were enrolled and the profile of their high school studies) the questions on a sheet of paper. The second phase was implemented in seminar rooms with subgroups.

Phase-1: After the above mentioned brief introduction students were invited

- to watch the dance choreography of the linear search algorithm (for a list with 7 elements) twice (see Figure 1.a) and to answer questions Q1.1.1-7;
- to watch the dance choreography of the binary search algorithm (for a list with 7 elements) twice (see Figure 1.b) and to answer questions Q1.2.1-7;
- to watch the dance choreography and the animation of the bubble sort algorithm (for a list with 10 elements) (see Figures 2.a and 2.b) and to answer questions Q1.3.1-2;
- to answer the synthesis question (Q1.4.1).

Phase-2: By discussion with the students (Socratic questioning), the teacher helped them (1) discover the correct answers for the previous day's questions and (2) realize why those answers are the right ones (teacher tried to confine only to ask questions).

Phase-3: Students were invited to watch the dance choreography and the animation of the selection sort algorithm (for a list with 10 elements) (see Figures 3.a and 3.b) and to answer questions Q3.1.1-4.

Figure 1
(a) Linear search with Flamenco dance; (b) Binary search with Flamenco dance.

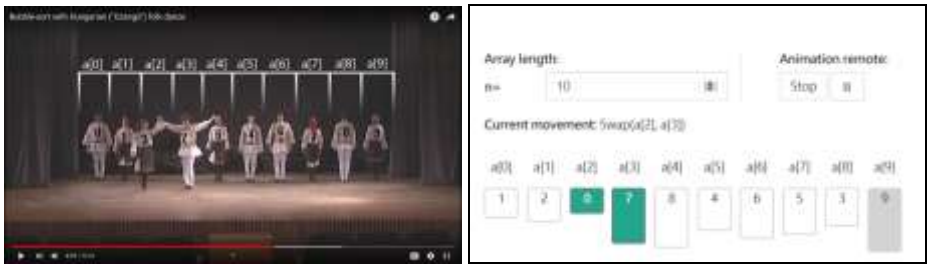

Figure 2
(a) Bubble sort with Hungarian folk dance; (b) Animating the Bubble sort algorithm.

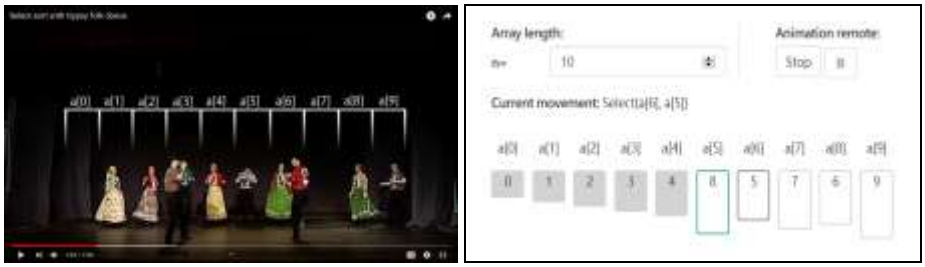

Figure 3
(a) Selection sort with Gipsy folk dance; (b) Animating the Selection sort algorithm.

FINDINGS AND DISCUSSION

The learning material that had to be assimilated with respect to each algorithm can roughly be divided in two parts: (1) students had to understand the strategy (or logic) of the algorithm and then, building on this knowledge, (2) they had to imagine its best/worst case behaviour. "Part-1 knowledge" had to be extracted mostly from the algorithm visualisation. To help students realize "part-2 knowledge" to each algorithm, a question sequence was attached. Group-0 had no prior knowledge with respect to the entire learning material. Group-1/2 had prior experience only in "part-1 knowledge". Group-4 was initiated in both "part-1 and part-2 knowledge". Although students' scores reflect their "part-2 knowledge", this knowledge assumed "part-1 knowledge".

Statistical analysis was performed in Excel, using the student t-test and with significance set to $p<0.05$. It included the phase-1 answers of all the three groups, and comparison of the corresponding phase-1 versus phase-3 answers of group-0. Table 1 shows students' results regarding phase-1 questions. For each group of questions, we computed the average performance. It can be noticed that group-0 students' scores are in line with how intuitive the corresponding algorithm was. In the case of binary search (logarithmic time complexity) the average performance was only 34%. It is reasonable to consider the complexity level of the synthesis question as fitting between the levels of questions regarding the quadratic bubble sort and logarithmic binary search algorithms.

Table 1
Performance results with respect to the phase-1 questions. (Since answering question Q1.2.6 assumed relatively advanced mathematical knowledge, we eliminated it from our analysis)

	Linear search (linear complexity) Q1.1.1-6	Binary search (logarithmic complexity) Q1.2.1-5	Bubble sort (quadratic complexity) Q1.3.1-2	Synthesis question Q1.4.1
Group-0	79%	34%	59%	51%
Group-1/2	86%	53%	49%	52%
Group-4	94%	76%	83%	65%

Interestingly, although students from group-1/2 had studied all the three algorithms previously, they did not outperform group-0 consequently. No significant differences were detected regarding the linear search and bubble sort algorithms and the synthesis question. Moreover, in the case of bubble sort, group-0 even outperformed group-1/2. The only significant difference (favouring group-1/2) was detected with respect to the binary search algorithm (t-test, group-0 vs. group-1/2: Q1.2.1-5 ($p=0.01<0.05$)), the least intuitive one.

Group-4 performed significantly better than group-1/2 in the case of all three algorithms (t-tests, group-4 vs. group-1/2: Q1.1.1-6 ($p=0.02<0.05$); Q1.2.1-5 ($p=0.000<0.05$); Q1.3.1-2 ($p=0.000<0.05$)). In the case of the synthesis question the difference was also substantial but not significant (Q1.4.1 ($p=0.07$)). The fact that group-4 scored consequently higher than the other two groups is a quite evident result. The questions were directly connected to a topic they had already studied: algorithm complexity analysis. As for the other two groups, this topic was unfamiliar.

On the other hand, group-1/2 students had a clear advantage over group-0 since "part-2 knowledge" had to be built on a knowledge that they already assimilated during their high school studies. While group-1/2 only had to refresh the strategies of the algorithms, group-0 students faced them for the first time. The fact that group-0 students (without any explicit help from the instructor) were able to perform on questions Q1.1, Q1.2 and Q1.4, shoulder-to-shoulder with their group-1/2 mates, partially supports our first hypothesis that the analysed learning environment (including algorithm visualization complemented with teacher prepared question sequence) has the potential to initiate non-majors (without prior knowledge in programing) in computer algorithms.

This conclusion is in line with prior research regarding Cognitive Apprenticeship Theory (Collins et al., 1991; Caspersen & Bennedsen, 2007). Two basic teaching methods related to this approach that we included in the phase-1 learning session were modeling and scaffolding. Inviting students to watch a teacher-created visualization can be seen as modelling since it allows students to observe how an expert implements the algorithm on given input. By the attached question sequence we provided students with a means of studying the algorithms from an algorithm efficiency perspective. Interestingly, Aulls (2002) reported (after he observed a number of teachers as they implemented constructivist activities in their classrooms) that the most effective teachers applied scaffolding when students failed to make learning progress in a discovery setting.

With respect to the effectiveness of the AlgoRythmics visualizations, other possible contributing factors are:

- The attached questions supported students in focusing on the relevant aspects of the dance choreography/animation. According to Rau, Michaelis and Fay (2015), in "textual + graphical" settings the text could guide learners' visual attention as they process the graphical representation.
- The algorithms are illustrated by human movement. Recent research results on the so-called human movement effect emphasize that observing human movements (or producing our own body movement) can be cognitively beneficial (Castro-Alonso et al., 2018). According to these authors, our cognitive systems are wired to observe human movements.
- Surprising science-art combination (Katai, 2014c). According to Keller (1983), providing novelty, incongruity and surprise are effective methods for arousing motivation.

On the other hand, the fact that group-0 scored significantly lower than gropup-1/2 on the questions attached to the least intuitive algorithm (binary search), and that both group-0 and group-1/2 performed poorly relative to group-4, support our second expectation that without explicit teacher guidance the potential this environment incorporates can be explored only partially.

Comparing group-0 students' phase-1 and phase-3 results strengthened us in this observation. As can be seen in Table 2, 57% of students realized that 45 (N(N-1)/2) swapping operations are needed for selection sort if the input sequence is sorted in descending order (worst case). This is a significantly better result (t-test: $p=0.05$) than in the case of the corresponding phase-1 question.

A "curious" particularity of the selection sort algorithm is that it implies N(N-1)/2 comparisons even in best case. Consequently, question Q3.1.1 was substantially more difficult (less intuitive) than its corresponding question from phase-1 (Q1.3.1). The fact that half of the students comprehended that the selection sort algorithm needs N(N-1)/2 comparisons and 0 swaps in the best case, is a noticeable result (taking into account that they did not study the algorithm previously). During the second and third phases of the

experiment group-0 students belonged to four sub-groups. A strange particularity of the Q1.3.1-dataset is that the performance of group-0 students belonging to sub-group-4 is only 14% and this value, in the case of the other three sub-groups, is 78%.

Table 2
Group-0 performance results with respect to the phase-1 and phase-3 sorting algorithms. The brackets contain the correct answers for the corresponding questions. (Phase-1 items did not include questions with respect to the number of swaps/comparisons in the best/worst case, respectively)

	Best case (comparison)	Best case (swap)	Worst case (comparison)	Worst case (swap)
Phase-1	Q1.3.1 (9)	- (0)	- (45)	Q1.3.2 (45)
Bubble-sort	78%	-	-	40%
Phase-3	Q3.1.1 (45)	Q3.1.2 (0)	Q3.1.3 (45)	Q3.1.4 (45)
Selection-sort	51%	95%	71%	57%

Phase-3 results emphasize the importance of the phase-2 discussion built around the phase-1 questions. Since the phase-3 learning environment was similar to the phase-1 setting, it is plausible to assume that phase-2 discussion contributed considerably to students' phase-3 performance. In addition, they performed better on the phase-3 algorithm, although the tutorial question-asking from phase-2 was limited to the phase-1 algorithms. The fact that students were able to extend their phase-1 comprehension to a new algorithm suggests that phase-2 discussion helped students see beyond the specific examples. Accordingly, phase-3 results confirm the hypothesis that the AlgoRythmics environment, if supplemented with tutorial question-asking, could be an effective instrument in introducing students with no prior knowledge in computing, even with deeper CS concepts such as algorithm efficiency.

This result is in line with previous findings regarding the possible benefit of Socratic questioning in CS education. For example, Wilson (1987) applied Socratic questioning during debugging processes. He observed that this kind of tutor-student dialogue helped students correct their misconceptions. We observed the same phenomenon: phase-2 Socratic questioning supported students in reanalyzing their phase-1 answers and, consequently, being more prepared for the phase-3 task. Lane and VanLehn (2005) also emphasize that Socratic questioning has the potential to help students in making important observations to improve their programming knowledge. More recently a chatbot was used (incorporated in an online learning environment) successfully to guide students through the Socratic method in a group discussion (Le and Huse, 2016).

Limitations

Definitions of CT emphasize that we are dealing with a multifaceted skill or ability. If we only consider the operational definition that ISTE (2020) attached to CT, then algorithms and their time complexity are just two elements of a complex skill set. Additionally, we did not follow the standard syllabus: usually, complexity aspects are analyzed only after the strategy of the algorithm had been thoroughly studied.

We concentrated on testing the efficiency of a particular algorithm visualization incorporated into a specific learning environment without comparing it with other possible approaches.

Another limitation could be that while all participants were involved in the first phase of the learning session, during the second and third phases we focused only on group-0 students. The other two groups could have provided us with additional data for our phase-3 analysis.

CONCLUSION

The AlgoRythmics YouTube channel (Katai & Toth, 2011; Katai et al., 2018) leads instructors to a collection of ten algorithmic dance choreographies: bubble-, insertion-, selection-, shell-, quick-, merge-, heap-sort, linear-, binary-search, and backtracking illustrated by folkdance/flamenco/ballet performances. The AlgoRythmics web application (Katai et al., 2020) supplements these videos with interactive computer animations. Important characteristics of this teaching-learning environment are its unified, artistically enhanced, human movement effect enriched, and CS free style. In this article we analyzed this environment (as CT promoter tool) from the perspective of contextualized computing education.

The basic question that motivated us in this research is as follows: what is the best practice for incorporating the AlgoRythmics environment in the teaching learning process of algorithms? In the research literature regarding the impact of instructional guidance during teaching, there is a dispute between those advocating the hypothesis that people learn best in an unguided or minimally guided environment (e.g., Papert, 1980) and those suggesting that novice learners should be provided with direct instructional guidance (e.g., Sweller, 2003; Mayer, 2004). Kirschner et al. (2006) argue that the superiority of guided instruction is grounded on our knowledge of human cognitive architecture, expert-novice differences and cognitive load.

Our experience with the AlgoRythmics environment also underlines the importance of instructional guidance in teaching learning algorithms. Research results reported in previous investigations and findings of the present study support this line of research. Even creating a visualization that illustrates the algorithm on different inputs can be considered as a kind of instructional guidance. How much additional guidance is needed depends on the on the complexity of the studied algorithm and the expressiveness of the visualization. In the previous AlgoRythmics studies, participants were invited to interactively orchestrate the studied algorithms and the environment assisted them in this learning task. The immediate feedback offered can also be seen as clear instructional guidance.

In this study this guidance was provided by targeted questioning (with and without teacher guidance). In addition to previous studies that focused only on supporting students to assimilate the strategy of the studied algorithms, we investigated if they were able to build on this knowledge by extracting from visualizations some algorithm efficiency related concepts too. Findings confirmed our expectation that AlgoRythmics visualizations are expressive enough, if supplemented with targeted questions, to

support students without prior knowledge in computing to assimilate the strategy of some basic computer algorithms and imagine the best and worst case behaviour of these algorithms.

An interesting further research topic would be to test the effectiveness of these methods in combination. We are planning to investigate the following four phase learning session: students are invited (1) to watch the dynamic visualization of the selected algorithm; (2) to interactively orchestrate the algorithm on random inputs (the software registers their activity); (3) to answer the attached question sequence regarding the best and worst case behaviour of the algorithm; (4) to participate in a Socratic questioning based on previous phase tasks.

Although the current intervention focused on first year engineering students, the findings of this study might be valuable in the case of other categories of learners too. Assimilating CT related concepts by analyzing dance choreographies could be an attractive approach at all levels of education. Such an initiative would be in line with those previously cited studies that have successfully combined computing education with arts at primary, secondary and high school level. Since AlgoRythmics videos can be used to introduce students to quite profound CS concepts too, these visualizations could prove to be valuable tools for CS-majors too. For example, the heap sort choreography displays how to perceive a unidimensional array as a binary tree, or the number of ballerinas in the Four-queens choreography reflects the number of calls that a recursive implementation of the algorithm implies, etc. Interestingly, the evaluation committee of the "2013 Best Practices in Education Award" (organized by Informatics Europe) points out that they "were impressed and appreciated this approach of abstracting away almost all details that might hinder understanding the idea or principle of an algorithm or a paradigm. The enactments thus not only can be used flexibly in teaching environments irrespective of a particular programming- or spoken-language but can be used as a starting point for the teacher to drill down into more technical concepts" (Informatics Europe, 2013).

ACKNOWLEDGEMENTS

The authors wish to acknowledge the help of the Professional Art Institute 'Muresul' and the András Lóránt Company for their artistic support. The authors would like to thank all subjects for participating in this research.

REFERENCES

Aho, A.V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835.

Ainsworth, S. E. (2006). DeFT: A conceptual framework for considering learning with multiple representations. *Learning and Instruction, 16*(3), 183-198.

Aulls, M. W. (2002). The contributions of co-occurring forms of classroom discourse and academic activities to curriculum events and instruction. *Journal of educational psychology, 94*(3), 520.

Berney, S., & Bétrancourt, M. (2016). Does animation enhance learning? A meta-analysis. *Computers & Education, 101*, 150-167.

Boyer, K. E., Lahti, W., Phillips, R., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2010, March). Principles of asking effective questions during student problem solving. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 460-464). ACM.

Caspersen, M. E., & Bennedsen, J. (2007). Instructional design of a programming course – A learning theoretic approach. In *Proceedings of the 3rd International Workshop on Computing Education Research* (pp. 111-122). ACM.

Castro-Alonso, J. C., Ayres, P., Wong, M., & Paas, F. (2018). Learning symbols from permanent and transient visual presentations: Don't overplay the hand. *Computers & Education, 116*, 1-13.

Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator 15*(3), 6-11.

Computer Science Teachers Association (CSTA). (2017). *CSTA K-12 Computer Science Standards*. http://www.csteachers.org/standards

Daily, S. B., Leonard, A. E., Jörg, S., Babu, S., & Gundersen, K. (2014, March). Dancing alice: exploring embodied pedagogical strategies for learning computational thinking. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 91-96). ACM.

Echeverría, L., Cobos, R., Machuca, L., & Claros, I. (2017). Using collaborative learning scenarios to teach programming to non-CS majors. *Computer applications in engineering education, 25*(5), 719-731.

El-Zakhem, I. H. (2016). Socratic programming: An innovative programming learning method. *International Journal of Information and Education Technology, 6*(3), 247.

Ghefaili, A. (2003). Cognitive apprenticeship, technology, and the contextualization of learning environments. *Journal of Educational Computing, Design & Online Learning, 4*(1), 1-27.

Graesser, A. C., & Person, N. K. (1994). Question asking during tutoring. *American educational research journal, 31*(1), 104-137.

Guzdial, M. (2010). Does contextualized computing education help?. *ACM Inroads, 1*(4), 4-6.

Guzdial, M., & Tew, A. E. (2006, September). Imagineering inauthentic legitimate peripheral participation: an instructional design approach for motivating computing education. In *Proceedings of the second international workshop on Computing education research* (pp. 51-58). ACM.

Informatics Europe. (2013). *Best Practices in Education Award*. http://www.informatics-europe.org/awards/education-award/2013.html

International Society for Technology in Education (ISTE). (2020). *ISTE Standards*. https://www.iste.org/standards

Katai, Z. (2014a, June). Selective hiding for improved algorithmic visualization. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 33-38). ACM.

Katai, Z. (2014b, June). Intercultural computer science education. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (pp. 183-18). ACM.

Katai, Z. (2014c, June). Algorithmic thinking for ALL: A motivational perspective. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 353-353). ACM.

Kátai, Z. (2015). The challenge of promoting algorithmic thinking of both sciences-and humanities-oriented learners. *Journal of Computer Assisted Learning, 31*(4), 287-299.

Katai, Z. (2020). Promoting computational thinking of both sciences- and humanities-oriented students: an instructional and motivational design perspective. *Educational Technology Research and Development*. https://doi.org/10.1007/s11423-020-09766-5

Katai, Z., Osztian, E., Osztian, P. R., Nagy, J. E., & Cosma, C. (2020). *AlgoRythmics* (Version 2.2.0). Sapientia Hungarian University of Transylvania. https://algorythmics.ms.sapientia.ro/

Katai, Z., Osztian, E., Osztian, P. R., & Vekov, G. K. (2018, January 30). *AlgoRythmics* [Video]. Youtube. https://www.youtube.com/user/AlgoRythmics/

Katai, Z., & Toth, L. (2011, March 29). *AlgoRythmics* [Video]. Youtube. https://www.youtube.com/user/AlgoRythmics

Keller, J. M. (1983). Motivational design of instruction. *Instructional Design Theories and Models: An Overview of Their Current Status, 1*(1983), 383-434.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational psychologist, 41*(2), 75-86.

Lane, H. C., & VanLehn, K. (2005). Teaching the tacit knowledge of programming to noviceswith natural language tutoring. *Computer Science Education, 15*(3), 183-201.

Le, N. T., & Huse, N. (2016). Evaluation of the formal methods for the Socratic method. In *Proceedings of the 16th International Conference on Intelligent Tutoring Systems* (pp. 68-78).

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning?. *American psychologist, 59*(1), 14.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..

Rau, M. A. (2016). Conditions for the effectiveness of multiple visual representations in enhancing STEM Learning. *Educational Psychology Review*, 1-45. https://doi.org/10.1007/s10648-016-9365-3.

Rau, M. A., Michaelis, J. E., & Fay, N. (2015). Connection making between multiple graphical representations: A multi-methods approach for domain-specific grounding of an intelligent tutoring system for chemistry. *Computers & Education, 82*, 460-485.

Razmov, V., & Anderson, R. (2006, March). Pedagogical techniques supported by the use of student devices in teaching software engineering. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education* (pp. 344-348).

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142-158.

Simon, B., Kinnunen, P., Porter, L., & Zazkis, D. (2010, June). Experience report: CS1 for majors with media computation. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 214-218). ACM.

Sweller, J. (2003). Evolution of human cognitive architecture. *Psychology of learning and motivation, 43*, 216-266.

Tamim, S. R., & Grant, M. M. (2013). Definitions and uses: Case study of teachers implementing project-based learning. *Interdisciplinary Journal of problem-based learning, 7*(2), 3.

Tawfik, A. A., Graesser, A., Gatewood, J., & Gishbaugher, J. (2020). Role of questions in inquiry-based instruction: towards a design taxonomy for question-asking and implications for design. *Educational Technology Research and Development*, 1-26.

Tenenberg, J., & Murphy, L. (2005). Knowing what I know: An investigation of undergraduate knowledge and self-knowledge of data structures. *Computer Science Education, 15*(4), 297-315.

Tew, A. E., McCracken, W. M., & Guzdial, M. (2005, October). Impact of alternative introductory courses on programming concept understanding. In *Proceedings of the first international workshop on Computing education research* (pp. 25-35). ACM.

Tofade, T., Elsner, J., & Haines, S. T. (2013). Best practice strategies for effective use of questions as a teaching tool. *American journal of pharmaceutical education, 77*(7), 155.

Wijnen, M., Loyens, S., Smeets, G., Kroeze, M., & van der Molen, H. (2017). Students' and teachers' experiences with the implementation of problem-based learning at a university law school. *Interdisciplinary Journal of Problem-Based Learning, 11*(2), 1-11.

Wilson, J. D. (1987). A Socratic approach to helping novice programmers debug programs. *ACM SIGCSE Bulletin, 19*(1), 179-182.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Wood, Z. J., Muhl, P., & Hicks, K. (2016, February). Computational Art: Introducing High School Students to Computing via Art. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 261-266). ACM.

Wu, S. P., & Rau, M. A. (2018). Effectiveness and efficiency of adding drawing prompts to an interactive educational technology when learning with visual representations. *Learning and Instruction, 55*, 93-104.