*mathematics*

MDPI

*Article*

# Teaching Algorithms to Develop the Algorithmic Thinking of Informatics Students

**Dalibor Gonda [1], Viliam Ďuriš [2,*], Anna Tirpáková [2,3] and Gabriela Pavlovičová [2]**

[1] Department of Mathematical Methods and Operations Research, Faculty of Management Science and Informatics, University of Žilina, Univerzitná 1, 01001 Žilina, Slovakia
[2] Department of Mathematics, Faculty of Natural Sciences, Constantine The Philosopher University in Nitra, Tr. A. Hlinku 1, 94901 Nitra, Slovakia
[3] Department of School Education, Faculty of Humanities, Tomas Bata University in Zlín, Štefánikova 5670, 760 00 Zlín, Czech Republic
* Correspondence: vduris@ukf.sk; Tel.: +421-37-6408-708

**Abstract:** Modernization and the ever-increasing trend of introducing modern technologies into various areas of everyday life require school graduates with programming skills. The ability to program is closely related to computational thinking, which is based on algorithmic thinking. It is well known that algorithmic thinking is the ability of students to work with algorithms understood as a systematic description of problem-solving strategies. Algorithms can be considered as a fundamental phenomenon that forms a point of contact between mathematics and informatics. As part of an algorithmic graph theory seminar, we conducted an experiment where we solved the knight's tour problem using the backtracking method to observe the change in students' motivation to learn algorithms at a higher cognitive level. Seventy-four students participated in the experiment. Statistical analysis of the results of the experiment confirmed that the use of the algorithm with decision-making in teaching motivated students to learn algorithms with understanding.

**Keywords:** backtracking; computational thinking; heuristics; knight's tour problem; learning algorithms; problem solving

**MSC:** 68W01; 97C70



**Citation:** Gonda, D.; Ďuriš, V.; Tirpáková, A.; Pavlovičová, G. Teaching Algorithms to Develop the Algorithmic Thinking of Informatics Students. *Mathematics* **2022**, *10*, 3857. https://doi.org/10.3390/math10203857

Academic Editor: Michael Voskoglou

Received: 7 October 2022
Accepted: 17 October 2022
Published: 18 October 2022



## 1. Introduction

Programming, i.e., software development, is becoming increasingly important as it enables the integration of various technologies into an ever-widening range of human activities. According to Türker and Pala [1], computational thinking can be considered an ability that every individual should gradually have. For this reason, too, a considerable initiative is being developed to integrate programming into the teaching process from primary school onwards [2–4]. Along with the growing availability of personal computers, researchers have addressed the issue of linking the mathematics teaching and programming. Related to this, there was an effort to identify the way of thinking that students need to develop in mathematics teaching to increase their potential to program (compare: [5–7]). As early as the 1950s, the term "computational thinking" began to be used, which describes the use of structured thinking or algorithmic thinking to create a suitable output for a given input [8]. According to Wing [9], computational thinking is a type of analytical thinking that includes problem solving, system design and an understanding of human behavior based on informatics concepts. Computational thinking represents conceptualization rather than programming, basic skills rather than syntax skills, and human thinking based on creativity, not programmed computer thinking. Computational thinking is a way of thinking that is used to create systematic and repeatable problem-solving procedures and includes problem decomposition, pattern recognition, abstraction, and algorithmic thinking [9,10].

Indeed, research has shown that integrating the teaching of computational thinking into the teaching of mathematics subjects improves students' programming skills [11–14]. Given that algorithms form an intersection between mathematics and informatics [5], the teaching of algorithms can be used to develop algorithmic thinking, which can be considered a basic pillar of computational thinking. Algorithmic thinking is an important detail-oriented skill that is based on a person's cognitive ability to analyze problems, develop a sequence of steps leading to an appropriate solution, streamline an already known sequence of steps, and find alternative steps to solve a problem [15]. Algorithmic thinking is the ability of students to work with algorithms perceived as a systematic description of problem-solving strategies. The ability to think in algorithms and algorithmic procedures can be defined as one of the important educational goals in mathematics [16]. The importance of teaching algorithms is that students learn the logic of programming without any programming language. Using an algorithm, it is possible for a student to write the steps of processing a program in their own language, that is, they can create a flowchart of a program with so-called code [17]. Thus, algorithms form the basis of programming logic. As a result of this information, it can be stated that when teaching algorithms, the student acquires the logic of programming and begins to focus more on the problem-solving process. Knuth [5] pointed out that the attention paid to algorithms is common in algorithmic thinking, but not common in mathematics. Similarly, Kiss and Arki [18] found that the lack of algorithmic thinking disadvantages students in higher education. Their research has shown that traditional teaching strategies are unsuitable for developing algorithmic thinking. They emphasized the need for a strategic focus on algorithmic and computational thinking in teaching. To use the teaching of mathematics to develop students' algorithmic thinking, it is necessary to find suitable strategies to change the teaching and learning of algorithms. The aim of our research is to statistically verify whether solving the knight's tour problem using the backtracking method will cause a change in students' approach to learning algorithms.

## 2. Algorithms and Their Learning

An algorithm is a well-defined sequence of rules that provides guidance on how to create output information from input information using a finite number of steps [19]. Other authors refer to the algorithm as only some computational procedures, those that guarantee the solution of the problem, if the steps of the procedure are performed in the specified order and without errors [20,21]. The execution of the algorithm therefore has high reliability and speed, which is its strength, if the only goal is to solve the task. However, if it is the subject of teaching and learning, an algorithm executed without regard to its significance may lead to memorization. Algorithms are a basic and key part of mathematics. The problem is not the algorithms themselves, but the dominance of algorithmic solution templates in the teaching of mathematics [22]. This undesirable trend in teaching algorithms is because many teachers consider passing on finished algorithms to students as the main goal of teaching mathematics (e.g., [23–27]). With this goal, a mentality of mathematics teaching is formed, where students are "programmed" to solve a set of problems [28]. Students focus on memorizing algorithms because they give them a sense of security and believe that memorizing an algorithm and assigning it to a given task leads to success in mathematics [21]. However, those skilled in the art of mathematics agree that memorized algorithms are easy to forget, error-prone, and transmission-resistant (e.g., [29–31]). Already Brousseau's theory of didactic situations in mathematics [32] suggests that passing on finished algorithms to students robs them of their own intellectual work in learning mathematics, and that is why learning by imitating algorithms is ineffective. Fan and Bokhove [33] concluded in their literature review that this way of teaching mathematics leads students to a dichotomy between computational procedures and understanding. Two separate studies (Jonsson et al. and Wirebring et al. [34,35] have shown that learning mathematics through creative mathematical reasoning and constructing one's own solution methods can be more effective than if a teacher presents students with ready-made algorithms that they then try to emulate when solving tasks. Based on the above research, it is necessary to look for ways to teach

algorithms that would lead to a change in the way students learn algorithms. It is necessary to teach algorithms, especially for students of informatics study programs, in such a way that they develop algorithmic thinking, the ability to solve problems and creativity, which is a sub-dimension of computational thinking [36]. As with learning generally, learning the algorithms of students in school mathematics can take place at different cognitive levels. According to Fan and Bokhove [33], learning and learning algorithms can be implemented at three mutually supportive cognitive levels. At the first cognitive level, it is possible to memorize the algorithm and then execute it in a comparable situation, but without real understanding. At the second cognitive level, there is an understanding of why the algorithm works. Based on this understanding, the student can apply the algorithm in a relatively complex situation. At the third cognitive level, the student can compare the algorithm with other algorithms (for example in terms of efficiency), evaluate the algorithm and create their own algorithm, while generalization is also considered to create a new algorithm. We explicitly distinguish between levels two and three in terms of understanding one particular algorithm and comparing several algorithms, because assessment is commonly perceived at a higher level of cognition as understanding (e.g., [37]). Cognitive levels two and three can be considered as levels where algorithms are studied. It is through the study of algorithms as general procedures that students gain the knowledge that mathematics is well structured [38], and its study brings a lot of benefits to the programmer.

## 3. The Role of the Knight's Tour Problem and Its Didactic Potential

The knight's tour problem is a chess and math problem. The knight moves according to the chess rules on the chessboard and the task is to visit each square exactly once. On a typical $8 \times 8$ chessboard, this task has a large number of solutions, of which in exactly 26,534,728,821,064 cases the knight ends up in the field from which the knight endangers the starting field [39]. The simplest algorithm for solving the Knight's tour problem is backtracking. Backtracking is a way of solving algorithmic problems based on searching the depth of possible solutions. This algorithm is an improvement over brute force solutions, as a large number of potential solutions can be ruled out without direct testing (more detailed, e.g., [40,41]). Backtracking is one of the return search algorithms that solves a specific problem not according to fixed calculation rules but based on trial and error. It is the presence of the trial-and-error method that can be used to change the perception of algorithms and thus the way students acquire them. Another "didactic" benefit of this algorithm is the fact that the process of trial and error is broken down into several subtasks, which are expressed in recursive terms and consist in examining the final number of subtasks.

An algorithm to solve the knight's tour problem can be created by defining the task that the algorithm has to solve. If we have a chessboard $n \times n$, then the chessboard has $n^2$ squares and the knight must execute $n^2 - 1$ moves so that the knight enters each square exactly once. The task of the algorithm is to decide whether the next move can be performed. Its basic structure is an algorithm (Algorithm 1):

**Algorithm 1**: Algorithm to determine the possibility of making the next move

```
procedure try move;
begin initializing stroke selection;
    repeat select another candidate from a list of other moves;
        if acceptable then
        begin record move;
                if the chessboard is not full then
                begin try next move;
                        if unsuccessful then delete previous record
                end
        end
    until (move was successful) ∨ (there are no other candidates)
end.
```

In the next phase of algorithm creation, it is necessary to mathematize individual commands and conditions, so it would be possible to program the algorithm and then debug it. For example, we replace the condition "chessboard is not full" with the expression $i < n^2$; we write the position of the knight using local variables $u$, $v$; the predicate "acceptable" using a logical combination of $1 \leq u \leq n$ and $1 \leq v \leq n$, etc. The resulting algorithm for solving the knight's tour problem on the chessboard can be found e.g., in [42].

An essential characteristic of the above solution to the knight's tour problem on the chessboard is the fact that the individual steps, based on which we proceeded to solve the problem, were first examined, and recorded by error using the trial method. In the case of a "dead end" that does not lead to a result, it is possible to delete the wrong part of the solution and return to the position where we went in the wrong direction when looking for a solution. It is the use of the trial-and-error method that makes unexpected elements of the algorithm for students that can encourage them to change the way they learn algorithms. We consider the finding that even the right algorithm can lead to a "dead end" as an element that will strongly disrupt students' perception of algorithms as a safe and trouble-free way to solve the problem.

## 4. Method

In agreement with the students of informatics at the bachelor's degree level, we conducted an experiment within the seminar algorithmic graph theory. We devoted four lessons to the knight's tour problem, which were beyond the scope of the seminar's teaching hours. Experimental teaching took place in the fourth and fifth week of the semester without changing the teacher. We solved the problem based on a scheme established in 1910 by J. Dewey (Figure 1).

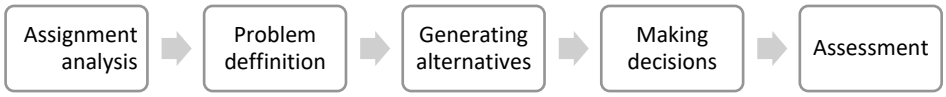


**Figure 1.** Basic stages of problem solving [43].

The use of this scheme allows students to be involved in creating a solution to the problem—creating an algorithm. During the individual phases, they were emphasized what solving skills are needed to master the individual stages of problem solving. At the last (fourth lesson) the students had the task to program the created algorithm.

When enrolling in the second year of bachelor's studies, we asked students if they would be willing to participate in the experiment. The course and goal of the experiment were presented to the students and they were assured of the anonymity of the obtained results. Out of 198 approached students, 74 students at the age of 20 voluntarily participated in the experiment. The aim of the experiment was to verify whether the use of the search strategy with return—backtracking to solve the problem of the so-called "fun math" the knight's tour problem motivates students to change the way they learn algorithms. Based on the stated goal of the research, we expressed the following working hypothesis:

**H1.** *The implementation of backtracking into the teaching of algorithms motivates students to learn algorithms at a higher cognitive level.*

At the beginning of the first lesson of experimental teaching of the algorithmic graph theory seminar, we asked students to fill in an anonymous questionnaire, which was created by members of the research team. The questionnaire contained eleven items (questions). The students filled in the questionnaire before the start of the experiment (pre-test). The same questionnaire was filled in by the participants of the experiment after the end of the last fourth lesson of the experiment (post-test). A total of 74 respondents participated in the experiment. Students answered the answers to the individual questions of the questionnaire from a scale of 1–5, where 1 means "least important" and 5 "most important". The aim of the questionnaire was for students to evaluate the importance of individual

activities in learning algorithms on the scale. The questions in the questionnaire are divided according to three cognitive levels of learning algorithms according to [33]. Three evaluated activities were dedicated to each cognitive level. Two questions of the questionnaire were aimed at finding out whether students prefer learning based on sample examples or the theoretical basis of the algorithm when learning algorithms.

We were interested in whether there were differences in the answers to individual items in the pre-test and in the post-test and if so, whether these differences are also statistically significant. We used the Stuart–Maxwell test to determine the statistical significance of differences in students' responses to individual items in the pre-test and post-test.

### *4.1. Description of the Method*

The Stuart–Maxwell test [44–47], a score-type test, is an extension of McNemar's test [48] to the situation where responses are allowed more than two response categories.

For the general matched-pair data summarized in an $r \times r$ square contingency table (Table 1) with total sample size $n$, we denote the probability of falling in to the $i$th row and $j$th column as $p_{ij}$, and we define the marginal probabilities $p_{i\cdot} = \sum_{j=1}^{r} p_{ij}$, $i = 1, 2, \ldots, r$, and $p_{\cdot j} = \sum_{i=1}^{r} p_{ij}$, $j = 1, 2, \ldots, r$; clearly $\sum_{i=1}^{r} \sum_{j=1}^{r} p_{ij} = 1$. We usually write by using the probability table (Table 2), which is generally in the form below.

**Table 1.** Contingency table.

| | | | | |
|---|---|---|---|---|
| $n_{11}$ | $n_{12}$ | ... | $n_{1r}$ | $n_{1\cdot}$ |
| $n_{21}$ | $n_{22}$ | ... | $n_{2r}$ | $n_{2\cdot}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $n_{r1}$ | $n_{r2}$ | ... | $n_{rr}$ | $n_{r\cdot}$ |
| $n_{\cdot 1}$ | $n_{\cdot 2}$ | ... | $n_{\cdot r}$ | $n$ |

**Table 2.** Probability table.

| | | | | |
|---|---|---|---|---|
| $p_{11}$ | $p_{12}$ | ... | $p_{1r}$ | $p_{1\cdot}$ |
| $p_{21}$ | $p_{22}$ | ... | $p_{2r}$ | $p_{2\cdot}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $p_{r1}$ | $p_{r2}$ | ... | $p_{rr}$ | $p_{r\cdot}$ |
| $p_{\cdot 1}$ | $p_{\cdot 2}$ | ... | $p_{\cdot r}$ | 1 |

Also, we denote the corresponding number of observations in the associated cases as $n_{ij}$, $n_{i\cdot}$, and $n_{\cdot j}$, respectively.

We have

$$n = \sum_{i=1}^{r} \sum_{j=1}^{r} n_{ij},\ n_{i\cdot} = \sum_{j=1}^{r} n_{ij},\ i = 1, 2, \ldots, r, n_{\cdot j} = \sum_{i=1}^{r} n_{ij},\ j = 1, 2, \ldots, r.$$

To test the marginal homogeneity of the two sets of probabilities, we are interested in assessing the hypothesis $H_0$: $p_{1\cdot} = p_{\cdot 1}$, $p_{2\cdot} = p_{\cdot 2}$, ... , $p_{r\cdot} = p_{\cdot r}$.

Let $d_i = n_{j\cdot} - n_{\cdot i}$ for $i = 1, 2, \ldots, r$ and $\mathbf{d} = (d_1, \ldots, d_r)'$. Denote $V_{ii} = n_{j\cdot} + n_{\cdot i} - 2n_{ii}$, $V_{ij} = -(n_{ij} + n_{ji})$ for $i \neq j$ and $V^* = (V_{ij})_{i,j=1}^{r-1}$, $d^* = (d_1, \ldots, d_{r-1})'$. Then, if the hypothesis $H_0$ is true, the Stuart-Maxwell test statistics $Q = d^{*\prime} V^{*^{-1}} d^*$ is asymptotically chi-squared with degree of freedom $r - 1$ for $n \to \infty$. Consequently, if $Q \geq \chi^2_{\alpha}(r-1)$, we reject the hypothesis $H_0$ at the significance level, which is asymptotically equal $\alpha$.

*4.2. Data Analysis and Results*

Using the Stuart–Maxwell test, we tested the null hypothesis $H_0$, which expresses that the probability of occurrence of the considered activity is the same before and after the experiment. We will test at a significance level of $\alpha = 0.05$. We performed the Stuart-Maxwell test using the STATISTICA program. After entering the input data in the output set of the computer, we received a contingency table for each item and the value of the test criterion of the Stuart-Maxwell test (Table 3). We will compare the value of the test criterion $Q$ with the critical value $\chi^2_{0.05}(4) = 9488$ We rejected the tested hypothesis $H_0$ at the significance level $\alpha = 0.05$, if the value of the test criterion $Q$ is greater than or equal to the critical table value (9488).

**Table 3.** Results of the Stuart-Maxwell test (pre-test and post-test).

| Cognitive Level | Question | *Q* |
|---|---|---|
| 1 | 1. question | 44.363 * |
| | 5. question | 2.787 |
| | 9. question | 44.224 * |
| 2 | 2. question | 20.941 * |
| | 6. question | 4.156 |
| | 10. question | 35.844 * |
| 3 | 3. question | 24.178 * |
| | 7. question | 2.313 |
| | 11. question | 37.926 * |

Note. Values exceeding the critical value are indicated * in the table.

Based on the results shown in Table 3, we can see that within cognitive level 1 there were statistically significant changes in the answers to questions no. 1 and no. 9. A statistically significant difference in the answers to no. 1 (when learning algorithms, I consider it important to remember the algorithm as a whole). in the pre-test as in the post-test illustrated in Figure 2.

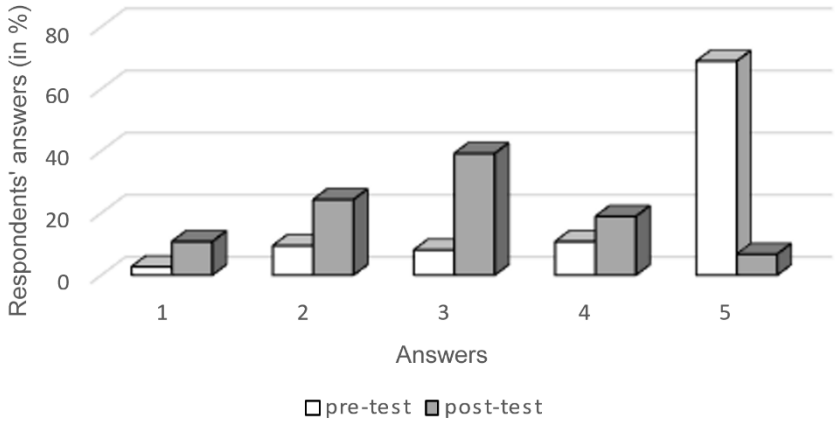


**Figure 2.** Respondents' answers to question no. 1 (when learning algorithms, I consider it important to remember the algorithm as a whole) in pre-test and post-test (in%).

In Figure 2 we can see that in the pre-test up to 80% of students considered remembering the algorithm as a whole to be very important, but in the post-test, this activity in learning algorithms was considered very important by only 28% of students. A similar decrease in the importance of the answer in the post-test compared to the answers in the pre-test was also recorded in the case of question no. 9 (when learning algorithms, I consider it important to solve more similar tasks to use a memorized algorithm). There was no change in importance in the ability to know how to assign an algorithm to a given

task (question no. 5). Students consider this ability to be very important in both pre-test and post-test.

Within cognitive level 2, there were statistically significant changes in the answers to questions no. 2 and no. 10. There was a statistically significant difference in the answers to question no. 10 (when learning algorithms, I consider it important to be able to use the whole algorithm in more complex tasks), in the pre-test as opposed to the post-test, illustrated in Figure 3.

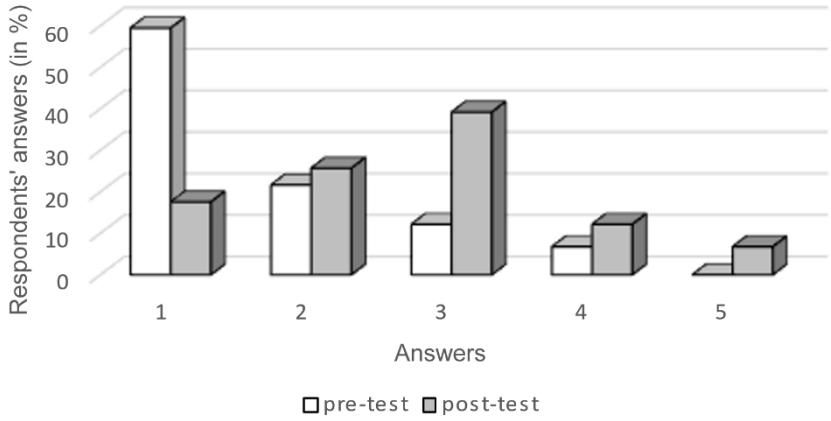


**Figure 3.** Respondents' answers to question no. 10 (when learning algorithms, I consider it important to be able to use the whole algorithm in more complex tasks) in pre-test and post-test (in%).

In Figure 3 we can see that in the pre-test, up to 81% of students consider the ability to know how to use the algorithm in more complex tasks to be insignificant. In the post-test, however, students were statistically significantly more inclined to believe that it is important to master the algorithm so that they can use it in solving more complex tasks. The same change was noted in the perception of the importance of the need to understand the individual steps of the algorithm (question 2). In the case of question no. 6 (when learning algorithms, I consider it important to analyze the algorithm to find out why it works), students consider this activity to be of little importance in the pre-test as well as in the post-test.

Within cognitive level 3, there were statistically significant changes in the answers to questions no. 3 and 11. There was a statistically significant difference in the answers to questions no. 3 (when learning algorithms, I consider it important to be able to adapt the algorithm to the problem) in the pre-test as opposed to the post-test, illustrated in Figure 4.

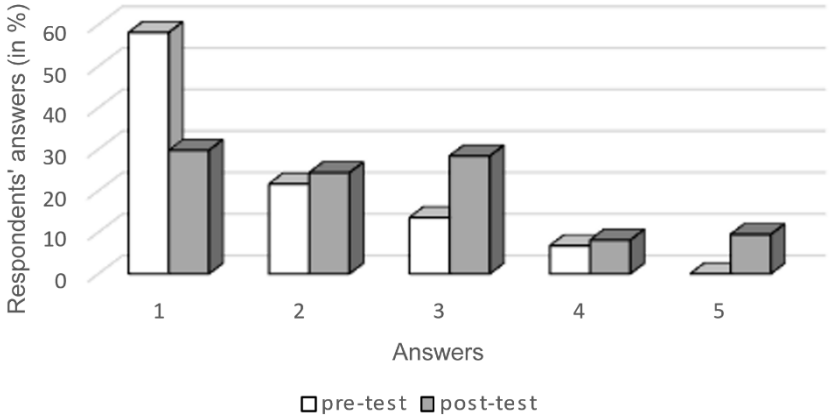


**Figure 4.** Respondents' answers to question no. 3 (when learning algorithms, I consider it important to be able to adapt the algorithm to the problem) in pre-test and post-test (in%).

In Figure 4 we can see that the ability to adapt the adopted algorithm to the solved problem was considered important only by 21% of students in the pre-test and up to 58% of them considered it to be of little importance. In the post-test, 45% considered this ability important and only 30% considered it unimportant. Even in the case of question no. 11 (When learning algorithms, I consider it important to be able to build my own algorithm.) there has been a similar change in the evaluation by students. In the pre-test, they rated this ability mostly as very unimportant, but in the post-test, they mostly rated it as important.

Through question no. 4 (when learning algorithms, I consider it important to have a model example, based on which I will learn the algorithm) and questions no. 8 (when learning algorithms, I consider it important to have a theoretical description of the algorithm), we looked at what resources students consider important when learning algorithms. Pre-test and post-test results for question no. 4 are listed in Table 4 and for questions no. 8 in Table 5 (Tables 4 and 5 are contingency tables expressed in %).

**Table 4.** Contingency table of answers to question no. 4 (in%).

| **Item 4** | **Post-Test (in%)** | | | | | **Sum** |
|---|---|---|---|---|---|---|
| **Pre-Test (in%)** | **1** | **2** | **3** | **4** | **5** | |
| 1 | 4 | 0 | 0 | 0 | 0 | 4 |
| 2 | 0 | 1 | 4 | 3 | 1 | 9 |
| 3 | 1 | 0 | 1 | 1 | 4 | 8 |
| 4 | 1 | 4 | 3 | 3 | 5 | 16 |
| 5 | 1 | 5 | 5 | 11 | 39 | 62 |
| sum | 8 | 11 | 14 | 18 | 50 | 100 |

$Q = 5913$.

**Table 5.** Contingency table of answers to question no. 8 (in%).

| **Item 8** | **Post-Test (in%)** | | | | | **Sum** |
|---|---|---|---|---|---|---|
| **Pre-Test (in%)** | **1** | **2** | **3** | **4** | **5** | |
| 1 | 50 | 16 | 8 | 1 | 1 | 77 |
| 2 | 5 | 5 | 5 | 0 | 0 | 16 |
| 3 | 1 | 3 | 0 | 0 | 0 | 4 |
| 4 | 1 | 0 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 0 | 0 | 1 |
| sum | 59 | 24 | 14 | 1 | 1 | 100 |

$Q = 8200$.

Given that the calculated value of the test criterion in both cases is less than the critical value $\chi^2_{0.05}(4) = 9488$, at the significance level $\alpha = 0.05$ we cannot reject the null hypothesis $H_0$. This means that the probability of occurrence of the considered learning sources is the same after the implementation of the experiment as the probability of occurrence before the implementation of the experiment. The observed differences are not statistically significant. From contingency table (Table 4), we see that students in both pre-test and post-test consider it very important that when learning algorithms, they have a sample example for the use of the algorithm. Based on contingency table (Table 5), we can state that students consider to have only a theoretical description of the algorithm to be of little importance.

## 5. Discussion

Due to great advances in science and technology, some algorithms are almost obsolete and in practice more attention is paid to the construction of new, often more complex algorithms. This progress requires the introduction of new concepts and ways of computational thinking [49], which also places new demands on the teaching of mathematics and informatics. According to Araya et al. [50] students may not be very skilled in using basic arithmetic algorithms to perform long divisions by large numbers. This frees up space in

teaching for teaching new concepts and algorithms, which are increasingly important for the development of computational thinking.

Based on the analysis of the results obtained in the pre-test, we can state that before solving the knight's tour problem, the students considered those activities that correspond to the first cognitive level of learning algorithms to be important when learning algorithms [51]. Activities that correspond to the second and third cognitive levels, according to [33], were considered to be of little importance. The results also show that students at the beginning of the seminar algorithmic graph theory preferred to master algorithms without understanding. This result corresponds to the research of [52], who found that students rely on memorized procedures and rules that they often learn without understanding. Several studies confirm that if a student has a template of solutions—learned algorithms, they try to use them [27,53,54]. This way of learning algorithms and their use in solving problems leads students to believe that mathematics is a subject where it is necessary to memorize a lot of definitions and algorithms [55].

By analyzing the results of the post-test, we concluded that the inclusion of a solution to a more demanding but didactically rich problem (in our case, the knight's tour problem) has the potential to encourage students to change the way they learn computational algorithms. Statistical analysis of the differences in students' answers to the same questions in the post-test and pre-test confirmed that there is a statistically significant decrease in the importance of activities that correspond to the first cognitive level. At the same time, the importance of activities corresponding to the second and third cognitive levels of learning algorithms has increased statistically significantly. In doing so, we came to the same conclusions as Laudano et al. [56] that the learning of algorithms may in individual cases take place at various levels simultaneously and not necessarily in a gradual manner from a lower cognitive level to a higher level. To achieve this goal, a suitable choice of tasks in the teaching of mathematics is needed [57].

In the post-test, the students considered remembering the algorithm as a whole (item 1) and practicing it (item 9) to be important, but not as important as it was in the pre-test. We think that this statistically significant decrease in the importance of these activities is related to the increasing importance of understanding the individual steps of the algorithm (item 2) and the ability to use the algorithm in more complex tasks (item 10). While in the pre-test the students rated these activities as unimportant, in the pre-test they evaluated them as important. The assessment of the ability to adapt the algorithm to a new task (item 3) and to be able to construct one's own algorithm (item 11) has also changed. If some of its steps are forgotten, they can reconstruct the algorithm [33]. We find it interesting to find out that students in both pre-test and post-test consider it very important to be able to assign an algorithm to a given task (item 5). This result points to the fact that students in learning mathematics are focused on solving problems, so they prefer problems that are solved using algorithmic methods. Their effort in solving it is reduced to the correct application of the algorithm leading definitely to the expected result [58,59]. This approach to problem solving is supported by the prevailing linearity of problem solving. Most school assignments are built in such a way as to encourage students to follow a structured, hierarchical, and linear path to the outcome. Therefore, it is not surprising that they feel less confident if they are to address a problem where this sequence is disrupted [60]. Therefore, we consider it appropriate that students have more experience with tasks whose solution procedures contain elements of heuristics. A trial and error "heuristic" was used to solve the knight's tour problem. For example, the trial-and-error strategy is a strategy without high cognitive demands, which is commonly used in mathematics lessons and in everyday life [61]. Therefore, this strategy is suitable for students becoming acquainted with the elements of heuristics, and its use requires problem analysis and understanding of the algorithm creation process [62]. By using heuristics within algorithms, students will gradually come to the realization that the certainty offered by the algorithm is stronger if they acquire the algorithm with understanding. Thus, it is possible to achieve the

integration of students procedural and conceptual knowledge, which brings an increase in their success in solving tasks [37,63,64].

Even though the knight's tour problem contains heuristics and the need to atomize the task, students in both pre-test and post-test consider it unimportant to analyze the algorithm and be able to evaluate the possibilities of its use in various problems. Although assessment and analysis are commonly perceived at a higher level of cognition than comprehension (e.g., [37]), there are probably more factors behind students' attitudes related to the way mathematics is taught. For example, Fuson [65] emphasizes the problem of an overemphasis on instrumental knowledge at the expense of relational knowledge in teaching. According to Benton [66] the obligation to follow a given method could hinder the development of children's thinking and discourage logical thinking. We consider students' lack of interest in solving long and complex tasks to be an equally key factor [67]. These factors cause students to have a strong concentration on mastering the algorithm in order to achieve success in solving problems. This concentration was also shown in the responses to items 4 and 8, where they consider it very important to have a model example, but for them it is of little importance to have a theoretical description of the algorithm. This can be interpreted as meaning that when students learn algorithms, they prefer to imitate the finished algorithm before thinking and creating their own algorithm. According to Araya et al. [49], it is appropriate to use more complex and advanced tasks in the teaching mathematics, which require creative procedures at the expense of the linearity of the solution. Because even more complex tasks within their analysis can be divided into simpler tasks—the task of atomizing. Atomization then leads to the solution of simple subtasks, which students prefer. This type of task is also the knight's tour problem on the chessboard. In our research, we focused on verifying the motivational potential of student involvement in the creation of a more complex computational algorithm, which is also a limiting factor in the interpretation of results. In further research, it would be necessary to investigate to what extent it is possible to implement more complex algorithms in the teaching of mathematics and what way will lead to the acquisition of computational algorithms at higher cognitive levels.

## 6. Conclusions

Based on the research, we can state in conclusion that the development of computational thinking of students is already possible in the teaching of mathematical subjects. The implementation of the solution to the knight's tour problem to the seminar and the subsequent analysis of the implementation confirmed that it is not necessary or appropriate to only pass on the finished algorithms to students. It was the students' involvement in creating the algorithm that encouraged students to change the way they learn algorithms. The decision-making and return algorithm have proven to be suitable to disrupt the prevailing linearity of algorithms that students are accustomed to in the normal teaching of mathematics. In the future, it might be appropriate to experimentally examine which elements of algorithms have an impact on changing the way algorithms are acquired. This could provide additional important data for the development of computational thinking in teaching mathematics.

**Author Contributions:** Data curation, G.P.; formal analysis, V.Ď.; investigation, D.G.; methodology, D.G.; project administration, A.T.; resources, G.P.; supervision, A.T.; validation, V.Ď. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** This work was supported by the Slovak Research and Development Agency under the contract No. APVV-14-0446, the Cultural and Educational Grant Agency of the Ministry of Education, Science, Research and Sports of the Slovak Republic No. KEGA 015UKF-4/2021 and the Scientific Grant Agency of the Ministry of Education, Science, Research and Sports of the Slovak Republic and the Slovak Academy of Sciences No. VEGA 1/0216/21.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Türker, P.M.; Pala, F.K. The effect of algorithm education on students' computer programming self-efficacy perceptions and computational thinking skills. *Int. J. Comput. Sci. Educ. Sch.* **2020**, *3*, 19–32.
2. Ching, Y.H.; Hsu, Y.C.; Baldwin, S. Developing computational thinking with educational technologies for young learners. *Tech. Trends* **2018**, *62*, 563–573. [CrossRef]
3. International Society for Technology in Education-ISTE. Computational Thinking for All. Available online: https://www.iste.org/explore/articleDetail?articleid=152 (accessed on 23 August 2021).
4. Chondrogiannis, E.; Symeonaki, E.; Papachristos, D.; Loukatos, D.; Arvanitis, K.G. Computational Thinking and STEM in Agriculture Vocational Training: A Case Study in a Greek Vocational Education Institution. *Eur. J. Investig. Health Psychol. Educ.* **2021**, *11*, 230–250. [CrossRef] [PubMed]
5. Knuth, D.E. Algorithmic thinking and mathematical thinking. *Am. Math. Mon.* **1985**, *92*, 170–181. [CrossRef]
6. Leron, U.; Dubinsky, E. An abstract algebra story. *Am. Math. Mon.* **1995**, *102*, 227–242. [CrossRef]
7. Misfeldt, M.; Ejsing-Duun, S. Learning mathematics through programming: An instrumental approach to potentials and pitfalls. In *CERME 9-Ninth Congress of the European Society for Research in Mathematics Education*; The European Society for Research in Mathematics Education (ERME): Prague, Czech Republic, 2015; pp. 2524–2530.
8. Denning, P.J. The profession of IT Beyond computational thinking. *Commun. ACM* **2009**, *52*, 28–30.
9. Wing, J.M. Computational thinking. *Commun. ACM* **2006**, *49*, 33–35. [CrossRef]
10. Rowe, E.; Almeda, M.V.; Asbell-Clarke, J.; Scruggs, R.; Baker, R.; Bardar, E.; Gasca, S. Assessing implicit computational thinking in Zoombinis puzzle gameplay. *Comput. Hum. Behav.* **2021**, *120*, 106707. [CrossRef]
11. Jona, K.; Wilensky, U.; Trouille, L.; Horn, M.S.; Orton, K.; Weintrop, D.; Beheshti, E. Embedding computational thinking in science, technology, engineering, and math (CT-STEM). In *Future Directions in Computer Science Education Summit Meeting*; Stanford University: Orlando, FL, USA, 2014.
12. Orton, K.; Weintrop, D.; Beheshti, E.; Horn, M.; Jona, K.; Wilensky, U. *Bringing Computational Thinking into High School Mathematics and Science Classrooms*; International Society of the Learning Sciences: Singapore, 2016.
13. Repenning, A.; Webb, D.C.; Koh, K.H.; Nickerson, H.; Miller, S.B.; Brand, C.; Repenning, N. Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Trans. Comput. Educ. (TOCE)* **2015**, *15*, 1–31. [CrossRef]
14. Tatar, D.; Harrison, S.; Stewart, M.; Frisina, C.; Musaeus, P. Proto-computational thinking: The uncomfortable underpinnings. In *Emerging Research, Practice, and Policy on Computational Thinking*; Springer: Cham, Switzerland, 2017; pp. 63–81.
15. Futschek, G. Algorithmic thinking: The key for understanding computer science. In *International Conference on Informatics in Secondary Schools-Evolution and Perspectives*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 159–168.
16. Broley, L.; Caron, F.; Saint-Aubin, Y. Levels of programming in mathematical research and university mathematics education. *Int. J. Res. Undergrad. Math. Educ.* **2018**, *4*, 38–55. [CrossRef]
17. Çamoğlu, K. *Algorithm*, 6th ed.; Kodlab: İstanbul, Turkey, 2018.
18. Kiss, G.; Arki, Z. The influence of game-based programming education on the algorithmic thinking. *Procedia-Soc. Behav. Sci.* **2017**, *237*, 613–617. [CrossRef]
19. Knuth, D.E. Computer science and its relation to mathematics. *Am. Math. Mon.* **1974**, *81*, 323–343. [CrossRef]
20. Anderson, J.R. Acquisition of cognitive skill. *Psychol. Rev.* **1982**, *89*, 369–406. [CrossRef]
21. Star, J.R. Reconceptualizing procedural knowledge. *J. Res. Math. Educ.* **2005**, *36*, 404–411.
22. Lithner, J. Principles for designing mathematical tasks that enhance imitative and creative reasoning. *ZDM* **2017**, *49*, 937–949. [CrossRef]
23. Lukhele, R.B.; Murray, H.; Olivier, A. Learners' understanding of the addition of fractions. In *Proceedings of the Fifth Annual Congress of the Association for Mathematics Education of South Africa*; Port Elizabeth Technikon: Port Elizabeth, South Africa, 1999; Volume 1, pp. 87–97.
24. Leung, F.K.S. Mathematics education in East Asia and the West: Does culture matter? In *Mathematics Education in Different Cultural Traditions: A Comparative Study of East Asia and the West*; Leung, F.K.S., Graf, K.-D., Lopez-Real, F.J., Eds.; Springer: New York, NY, USA, 2006; pp. 21–46.
25. Bergqvist, T.; Lithner, J. Mathematical reasoning in teachers' presentations. *J. Math. Behav.* **2012**, *31*, 252–269. [CrossRef]
26. Shield, M.; Dole, S. Assessing the potential of mathematics textbooks to promote deep learning. *Educ. Stud. Math.* **2013**, *82*, 183–199. [CrossRef]

27. Boesen, J.; Helenius, O.; Bergqvist, E.; Bergqvist, T.; Lithner, J.; Palm, T.; Palmberg, B. Developing mathematical competence: From the intended to the enacted curriculum. *J. Math. Behav.* **2014**, *33*, 72–87. [CrossRef]
28. Freudenthal, H. *Didactical Phenomenology of Mathematical Structures*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1986; Volume 1.
29. Hiebert, J.; Carpenter, T. Learning and teaching with understanding. In *Handbook of Research on Mathematics Teaching and Learning*; Grouws, D., Ed.; Simon Schuster Macmillan: New York, NY, USA, 1992; pp. 65–97.
30. Moreno, R. Decreasing cognitive load for novice students: Effects of explanatory versus corrective feedback in discovery-based multimedia. *Instr. Sci.* **2004**, *32*, 99–113. [CrossRef]
31. Oakes, J.; Lipton, M.; Anderson, L.; Stillman, J. *Teaching to Change the World*; Routledge: New York, NY, USA, 2018. [CrossRef]
32. Brousseau, G. *Theory of Didactical Situations in Mathematics: Didactique des Mathématiques, 1970–1990*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006; Volume 19.
33. Fan, L.; Bokhove, C. Rethinking the role of algorithms in school mathematics: A conceptual model with focus on cognitive development. *ZDM* **2014**, *46*, 481–492. [CrossRef]
34. Jonsson, B.; Norqvist, M.; Liljekvist, Y.; Lithner, J. Learning mathematics through algorithmic and creative reasoning. *J. Math. Behav.* **2014**, *36*, 20–32. [CrossRef]
35. Wirebring, L.K.; Lithner, J.; Jonsson, B.; Liljekvist, Y.; Norqvist, M.; Nyberg, L. Learning mathematics without a suggested solution method: Durable effects on performance and brain activity. *Trends Neurosci. Educ.* **2015**, *4*, 6–14. [CrossRef]
36. Naseer, M.; Zhang, W.; Zhu, W. Prediction of coding intricacy in a software engineering team through machine learning to ensure cooperative learning and sustainable education. *Sustainability* **2020**, *12*, 8986. [CrossRef]
37. Rittle-Johnson, B.; Star, J.R.; Durkin, K. Developing procedural flexibility: Are novices prepared to learn from comparing procedures? *Br. J. Educ. Psychol.* **2012**, *82*, 436–455. [CrossRef]
38. Kilpatrick, J.; Swafford, J.; Findell, B. *Adding it Up: Helping Children Learn Mathematics*; National Research Council, Ed.; National Academy Press: Washington, DC, USA, 2001; Volume 2101.
39. Wegener, I. *Branching Programs and Binary Decision Diagrams: Theory and Applications*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2000; ISBN 0-898-71458-3.
40. Kondrak, G.; Van Beek, P. A theoretical evaluation of selected backtracking algorithms. *Artif. Intell.* **1997**, *89*, 365–387. [CrossRef]
41. Van Beek, P. Backtracking search algorithms. In *Foundations of Artificial Intelligence*; Elsevier: Amsterdam, The Netherlands, 2006; Volume 2, pp. 85–134.
42. Wirth, N. *Algorithms and Data Structures*, 2nd ed.; Alfa: Bratislava, Slovakia, 1989; p. 488, ISBN 80-05-00153-3. (In Slovak)
43. Dewey, J. *How We Think*; D. C Heath Co Publishers: Chicago, IL, USA, 1910.
44. Stuart, A. A test for homogeneity of the marginal distributions in a two-way classification. *Biometrika* **1955**, *42*, 412–416. [CrossRef]
45. Maxwell, A.E. Comparing the classification of subjects by two independent judges. *Br. J. Psychiatry* **1970**, *116*, 651–655. [CrossRef]
46. Abbasi, N.; Dokoohaki, S.; Jamali, H. The Application of Stuart-Maxwell Test in Determining the Identically Distributed Correct Choice. *Appl. Math. Sci.* **2009**, *3*, 447–450.
47. Agresti, A. *Categorical Data Analysis*, 3rd ed.; John Wiley & Sons Inc: Hoboken, NJ, USA, 2013; p. 752. ISBN 0470463635.
48. McNemar, Q. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* **1947**, *12*, 153–157. [CrossRef]
49. Denning, P.J.; Tedre, M. *Computational Thinking*; Mit Press: Cambridge, MA, USA, 2019; ISBN 9780262536561.
50. Araya, R.; Isoda, M.; González, O. A Framework for Computational Thinking in Preparation for Transitioning to a Super Smart Society. *J. Southeast Asian Educ.* **2020**, *1*, 1–16.
51. Lisarelli, G.; Baccaglini-Frank, A.; Di Martino, P. From how to why: A quest for the common mathematical meanings behind two different division algorithms. *J. Math. Behav.* **2021**, *63*, 100897. [CrossRef]
52. Fuson, K.C.; Kalchman, M.; Bransford, J.D. Mathematical understanding: An introduction. In *How Students Learn: History, Mathematics, and Science in the Classroom*; National Research Council: Washington, DC, USA, 2005; pp. 217–256.
53. Hiebert, J. What research says about the NCTM standards. In *A Research Companion to Principles and Standards for School Mathematics*; National Council of Teachers of Mathematics: Reston, VA, USA, 2003; pp. 5–23.
54. Lithner, J. Students' mathematical reasoning in university textbook exercises. *Educ. Stud. Math.* **2003**, *52*, 29–55. [CrossRef]
55. Escalera Chávez, M.E.; Moreno García, E.; Rojas Kramer, C.A. Confirmatory Model to Measure Attitude towards Mathematics in Higher Education Students: Study Case in SLP Mexico. *Int. Electron. J. Math. Educ.* **2019**, *14*, 163–168.
56. Laudano, F.; Tortoriello, F.S.; Vincenzi, G. An experience of teaching algorithms using inquiry-based learning. *Int. J. Math. Educ. Sci. Technol.* **2020**, *51*, 344–353. [CrossRef]
57. Olteanu, C. Reflection-for-action and the choice or design of examples in the teaching of mathematics. *Math. Educ. Res. J.* **2017**, *29*, 349–367. [CrossRef]
58. Căprioară, D. Problem solving-purpose and means of learning mathematics in school. *Procedia-Soc. Behav. Sci.* **2015**, *191*, 1859–1864. [CrossRef]
59. Amar, G.I.; Suranto, S. The Use of Creative Problem Solving Based Genetic Mutation Module in Higher Education. *Int. J. High. Educ.* **2021**, *10*, 33–45. [CrossRef]
60. García, T.; Boom, J.; Kroesbergen, E.H.; Núñez, J.C.; Rodríguez, C. Planning, execution, and revision in mathematics problem solving: Does the order of the phases matter? *Stud. Educ. Eval.* **2019**, *61*, 83–93. [CrossRef]

61. Elia, I.; van den Heuvel-Panhuizen, M.; Kolovou, A. Exploring strategy use and strategy flexibility in non-routine problem solving by primary school high achievers in mathematics. *ZDM* **2009**, *41*, 605–618. [CrossRef]
62. Abdullah, A.H.; Rahman, S.N.S.A.; Hamzah, M.H. Metacognitive skills of Malaysian students in non-routine mathematical problem solving. *Bolema: Bol. De Educ. Matemática* **2017**, *31*, 310–322. [CrossRef]
63. Baroody, A.J. The development of adaptive expertise and flexibility: The integration of conceptual and procedural knowledge. In *The Development of Arithmetic Concepts and Skills: Constructing Adaptive Expertise*; Baroody, A.J., Dowker, A., Eds.; Erlbaum: Mahwah, NJ, USA, 2003; pp. 1–34.
64. Rittle-Johnson, B.; Schneider, M. Developing conceptual and procedural knowledge of mathematics. In *Oxford Handbook of Numerical Cognition*; Oxford University Press: Oxford, UK, 2015; pp. 1118–1134. [CrossRef]
65. Fuson, K.C. A Forum for Researchers: Issues in Place-Value and Multidigit Addition and Subtraction Learning and Teaching for Research on Mathematics Teaching. *J. Res. Math. Educ.* **1990**, *21*, 273–280. [CrossRef]
66. Benton, L.; Saunders, P.; Kalas, I.; Hoyles, C.; Noss, R. Designing for learning mathematics through programming: A case study of pupils engaging with place value. *Int. J. Child-Comput. Interact.* **2018**, *16*, 68–76. [CrossRef]
67. Phonapichat, P.; Wongwanich, S.; Sujiva, S. An analysis of elementary school students' difficulties in mathematical problem solving. *Procedia-Soc. Behav. Sci.* **2014**, *116*, 3169–3174. [CrossRef]