

Recognizing Algorithmic Concepts in New Contexts: An Analysis of Students' Reasoning

Jacqueline NIJENHUIS-VOOGT¹, Durdane BAYRAM-JACOBS²,
Paulien C. MEIJER³, Erik BARENDSSEN⁴

¹*Institute for Science Education, Radboud University, Nijmegen, The Netherlands*

²*Eindhoven School of Education, Eindhoven University of Technology, Eindhoven, The Netherlands*

³*Radboud Teachers Academy, Radboud University, Nijmegen, The Netherlands*

⁴*Institute for Science Education, Radboud University, Nijmegen and Department of Computer Science, Open University, The Netherlands*
e-mail: jacqueline.nijenhuis@ru.nl, d.bayram.jacobs@tue.nl, p.meijer@docentenacademie.ru.nl, erik.barendsen@ru.nl

Received: May 2021

Abstract. Teaching algorithmic thinking enables students to use their knowledge in various contexts to reuse existing solutions to algorithmic problems. The aim of this study is to examine how students recognize which algorithmic concepts can be used in a new situation. We developed a card sorting task and investigated the ways in which secondary school students arranged algorithmic problems (Bebras tasks) into groups using algorithm as a criterion. Furthermore, we examined the students' explanations for their groupings. The results of this qualitative study indicate that students may recognize underlying algorithmic concepts directly or by identifying similarities with a previously solved problem; however, the direct recognition was more successful. Our findings also include the factors that play a role in students' recognition of algorithmic concepts, such as the degree of similarity to problems discussed during lessons. Our study highlights the significance of teaching students how to recognize the structure of algorithmic problems.

Keywords: computer science education, algorithms, secondary education, card sorting.

1. Introduction

Algorithms are at the heart of computer science (CS) (Harel and Feldman, 2004), and are a central concept for CS education (Schwill, 1994; Zender and Spannagel, 2008). Algorithmic thinking is regarded as a special problem-solving competence (Futschek and Moschitz, 2010) and is one of the main skills that students acquire in CS education. Learning algorithmic thinking is not only important for learning CS but these skills are also perceived to be useful in other domains as well (Gal-Ezer and Stephenson, 2014).

CS curricula in secondary education often involve the teaching of algorithmic problem-solving and standard (or ‘classic’) algorithms (e.g., CSTA, 2017; Barendsen *et al.*, 2016). To be able to apply this knowledge when solving new problems, students are required to recognize the underlying algorithmic concepts in a new problem. When students encounter a new problem, they may directly recognize the underlying algorithmic concepts; for example, when students are asked to find an item in a sorted list, they may directly recognize that they can use a binary search algorithm. On the other hand, students may recognize a similarity with a previously seen problem and realize that the algorithm used to solve that solved problem can be used again; for instance, when students are asked to find the shortest path on a map from city X to city Y, students may remember a similar problem they solved previously where they also had to find a shortest path, which may remind them that they could use Dijkstra’s algorithm for finding the shortest path. These two routes for recognizing algorithmic concepts are illustrated in Figure 1.

To recognize algorithmic concepts directly (route A in Fig. 1), students need skills such as decomposition and abstraction (Armoni, 2013; Muller and Haberman, 2008; Soloway, 1986). Students are required to learn that a problem can be decomposed into parts for which algorithms have already been developed. The ability to identify the essential aspects of a problem can help in this process.

Another way of recognizing underlying algorithmic concepts is by identifying similarities with a previously solved problem (route B, existing of B1 followed by B2). Previous research has studied this “problem-solving transfer” (Bassok, 2003; Catrambone and Holyoak, 1989; Schmid *et al.*, 2003), where students use (or modify) an existing solution when encountering a new problem. These studies are often based on the degree of similarity between the base (previously solved) problem and the target problem (in the new context; see B1 in Fig. 1); for example, Bassok (2003) distinguished between surface and structural similarities between problems.

Both routes (direct or via similar problems) are viable for solving new problems, and it will be interesting to examine how secondary school students recognize underlying algorithmic concepts when they are presented with new algorithmic problems. To the best of our knowledge, no studies have been found that focus on students’ recognition of algorithmic concepts or examine whether students prefer the direct route or if they more often identify similarities with previous problems. The contextualization of algo-

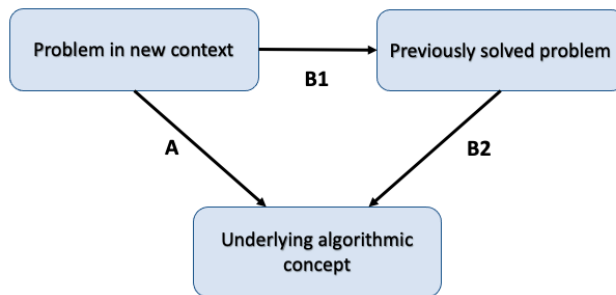


Fig. 1. Recognition of algorithmic concepts.

gorithms is a key characteristic of problem-solving in CS. Problems occur in a specific context, but a solution is developed at a conceptual level. The main aim of this study is to examine the way in which students handle this contextualization. By investigating how students match algorithmic problems to underlying algorithms, we aim to provide insight into the recognition of algorithmic concepts. Consequently, this study addresses the following research questions:

1. How do students recognize algorithmic concepts in new problems?
2. What factors play a role in students' recognition of algorithmic concepts?

The data for this study were collected using a card sorting task, for which students were asked to sort algorithmic problems based on the underlying algorithm. Other studies have used such categorization assignments to characterize students' conceptual knowledge (Chi *et al.*, 1981; Muller and Haberman, 2008; Smith *et al.*, 2013) but no prior research has used the sorting of Bebras tasks or similar algorithmic problems to examine students' recognition of algorithmic concepts.

In this paper, we first present a conceptual framework of the recognition of algorithmic concepts. We then describe the method used in this study, with the development of the card sorting task. This is followed by a description of our findings. Finally, our conclusions and a discussion are presented, including a description of some of the potential implications.

2. Conceptual Framework

2.1. Recognition of Algorithms in the Problem-Solving Process

At the core of algorithmic thinking lies the problem-solving process. Learning algorithmic thinking includes learning how to define a problem, how to design and develop an algorithm that solves this problem, and how to evaluate an algorithmic solution in terms of correctness and complexity (Futschek, 2006; Selby and Woollard, 2013).

Algorithmic thinking is one of the elements of *computational thinking* (CT), which is regarded as a key 21st century skill and is included in K–12 education worldwide (Grover and Pea, 2018; Selby and Woollard, 2013). CT skills are considered essential for all students (Wing, 2006; Yadav *et al.*, 2016). Based on a review of existing definitions and models, Shute *et al.* (2017) defined CT as “the conceptual foundation required to solve problems effectively and efficiently (i.e., algorithmically, with or without the assistance of computers) with solutions that are reusable in different contexts” (p. 151). CT is concerned with algorithmic problem-solving in every domain (Barr and Stephenson, 2011), hence the importance of reusing solutions in different contexts.

The analysis and specification of the problem to be solved are significant steps in algorithmic thinking (Futschek, 2006). Especially for problem-solving in different domains, as is the aim of CT, the specification of the problem includes the translation of the problem into computational elements (Kallia *et al.*, 2021). In a study investigating CT in mathematics education, Kallia *et al.* (2021) described the ‘*contextualization*’ of

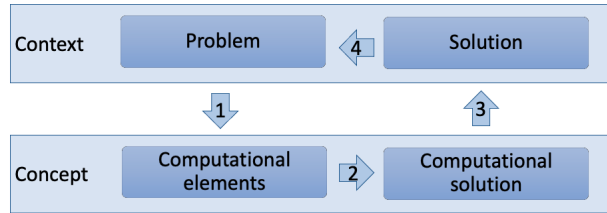


Fig. 2. Contextualization activities (Adapted from Kallia *et al.* (2021)).

the problem-solving discipline and distinguished four categories of cognitive activities (see Fig. 2): 1) translation of the problem into computational elements; 2) construction of a solution by using or developing algorithms; 3) translation of this solution in terms of the specific context or domain; 4) evaluation of whether the solution solves the real-world problem.

Consequently, when encountering a new problem, it would be helpful for students to realize when the underlying concepts are similar to the algorithmic concepts of a previously solved problem. Students may recognize the underlying concepts directly, on the concept level (see Fig. 2), which is comparable to route A in Fig. 1, or they may identify similarities with a previously solved problem (the context level in Fig. 2), enabling them to recognize the underlying algorithmic concepts via the previous solution (route B in Fig. 1).

2.2. Direct Recognition of Algorithmic Concepts

To recognize the algorithmic concepts in a new problem, it is essential that students understand how a problem is structured. In his study about learning to program, Soloway (1986) described the necessity of teaching effective problem-solving skills and highlighted the importance of teaching students strategies for using common solutions. He proposed a set of design strategies, the first of which is ‘stepwise refinement’: “Break down a problem into subproblems, on the basis of problems that you have already solved and for which you have canned (or almost canned) solutions” (Soloway, 1986, p. 855). These ‘canned solutions’ consist of programming plans or algorithms to achieve a specific goal. To recognize common subproblems and to decompose the problem, students need to learn about these ‘canned solutions’ and need to know the algorithms that can be used to solve common subproblems. In addition, attention should be devoted to ‘plan composition methods’ (Soloway, 1986), allowing students to learn how these common solutions should be woven together by, for example, nesting and merging.

In addition to learning decomposition skills, students also may need abstraction skills to recognize algorithmic concepts. To support the abstraction processes in problem-solving, Muller and Haberman (2008) proposed a pattern-oriented instruction. They complemented Soloway’s (1986) ‘plan composition methods’ by stating that algorithmic patterns can be identified, and subsequently need to be nested or merged to generate an

algorithmic solution. Muller and Haberman (2008) described three facets of abstraction that support focusing on the essence of a problem. One of these facets is pattern recognition, a process that will support students in understanding the significant elements of a problem while ignoring the context of the problem.

Armoni (2013) described a framework for teaching abstraction and emphasized the importance of representing only the essential aspects of a problem to reduce the complexity. Likewise, Izu *et al.* (2019) highlighted the role of using abstraction to identify ‘plans’ as an important part of program comprehension. These plans represent a high-level schema for solving a problem and are comparable to Soloway’s (1986) ‘canned solutions’.

Together these studies provide insights into the importance of decomposition and abstraction for the direct recognition of algorithmic concepts. Both skills are also characteristics of computational thinking, described by Wing (2006) as “using abstraction and decomposition when attacking a large complex task or designing a large complex system” (p. 33).

2.3. Identification of Similarity with a Previously Solved Problem

When encountering a new algorithmic problem, students may recognize similarities with a previously solved problem. If the algorithmic concepts of the previously solved problem are similar to the algorithmic concepts of the new problem, the previously developed solution can be applied to the new problem. This problem-solving transfer involves the use (or modification) of existing solutions when encountering new problems (Bassok, 2003).

There is a considerable amount of research highlighting how the process of transfer is supported by the acquisition of a solid knowledge base. Haskell (2001) defined the prerequisites that enable transfer, listing as the first prerequisite that learners need to acquire a high level of expertise in the area where transfer is needed. Bransford *et al.* (2000) highlighted the importance of “learning with understanding” (p. 55), and stated that merely memorizing sets of facts does not prepare students to transfer their knowledge when solving problems. Students’ understanding of concepts as an influencing factor in the transfer of knowledge has been confirmed by a recent study in the field of mathematics. In an investigation of transfer of algebraic skills from mathematics into physics (Tursucu *et al.*, 2020), the students lacked sufficient symbol sense behavior and basic algebraic skills and therefore encountered difficulties during the transfer.

Besides the factors related to students (their prior knowledge or skills), other factors can also play a role in the process of transfer. Kershaw *et al.* (2013) examined the constraints that impede transfer and made a distinction between constraints that are related to the ‘problem’ and those related to the ‘individual’. Constraints related to the problem may be caused by the complexity of the insight problems. In their article, Kershaw *et al.* (2013) presented a problem that requires a three-dimensional representation, where most participants in the experiment only thought of two-dimensional solutions. For CS, the representation of a problem may play a similar role; for example, The Knight’ tour

puzzle (included in the Appendix as one of the problems used in our research) is easier to solve if the problem is represented as a graph (Curzon, 2015).

Bransford *et al.* (2000) identified other factors that influence transfer, including ‘context’ (p. 62). The significance of the context is reiterated in the literature regarding context-based learning; for example, Guzdial (2010) argued for the use of multiple contexts to prevent a situation where new knowledge is only connected to a single context and cannot be transferred to other contexts or situations. Likewise, in a study regarding the Chemie im Kontext (ChiK) Project, Nentwig *et al.* (2007) described how a variety of contexts are essential for enabling a ‘step of abstraction’ (p. 1442), which is needed to acquire a systematic concept understanding that can be used in new situations.

Furthermore, case-based reasoning (CBR), a model of learning from experience (Kolodner, 1993), may contribute to the process of transfer. CBR promotes the use of previous cases to support the problem-solving process by adapting an old solution or blending parts of a few old solutions. CBR is developed to empower students to learn from previous experiences; they may remember something relevant from an earlier experience, assess whether that knowledge is applicable in a new situation, and apply what they have recalled. In this way, CBR and the reuse of lessons learned from previous experiences may promote the transfer of algorithmic knowledge (Kolodner *et al.*, 2003a).

In all the studies reviewed here, various factors are described that influence the process of transfer. These factors may also play a role in the identification of a similarity with a previously solved problem, as is examined in our study.

3. Method

To collect student data for this study, we used a card sorting task inspired by the works of Chi *et al.* (1981) and Smith *et al.* (2013). We investigated the ways in which upper secondary school students arranged algorithmic problems into groups and how they explained their grouping to their student peers.

3.1. Card Sorting Task to Examine Students’ Recognition

Previous studies have used card sorting tasks to characterize how conceptual knowledge is organized and connected (Smith *et al.*, 2013). Since the seminal work of Chi *et al.* (1981), sorting tasks are often used to investigate differences in the categories used by experts and novices. For sorting tasks, participants may be asked to sort cards with either terms (Brinda *et al.*, 2019) or problems (Chi *et al.*, 1981; Smith *et al.*, 2013). Entities to be sorted “need to be at the same semantic level as each other” (Rugg and McGeorge, 2005). Card sorting tasks have been used in physics (Chi *et al.*, 1981), biology (Smith *et al.*, 2013), chemistry (Irby *et al.*, 2016), and computer science (Brinda *et al.*, 2019; McCauley *et al.*, 2005).

A similar categorization assignment was used by Muller and Haberman (2008) to examine students’ abstraction skills. To investigate the influence of pattern-oriented in-

struction on students' problem-solving competences, Muller and Haberman (2008) asked students to sort and group algorithmic problems according to different types of student choices, revealing that students taught with the pattern-oriented instruction performed better in identifying structural similarities between problems.

Taken together, these studies support the notion that a categorization assignment is supportive in research into student reasoning; thus, a card sorting task seemed a reasonable choice to investigate students' recognition of underlying algorithms in new problems.

3.2. Development of Card Sorting Task for Algorithmic Problem-Solving

For the development of the card sorting task, we selected various algorithmic problems, conducted a pilot study with experts, and piloted the task with students. These different steps are described below.

We selected a set of algorithmic problems from three sources: a) the Bebras tasks (Bebras, 2021; Dagienè and Futschek, 2008) which were used in an earlier study regarding concepts in K-9 CS education (Barendsen *et al.*, 2015); b) the tasks found on the website of the Dutch Bebras contest (Beverwedstrijd, 2018); and c) the puzzle-based activities of 'Teaching London Computing' (CAS & CS4FN, 2018). Bebras tasks are particularly suitable because a wide majority of these tasks focus on algorithms and procedures (Izu *et al.*, 2017). The activities of 'Teaching London Computing' focus on various themes, including algorithmic thinking (Curzon, 2014). After studying the teaching material that would be used to teach the participating students, we selected problems that were related to one of the algorithms they had been taught.

In order to test the card sorting task, we performed two subsequent pilot studies, one with experts and another with students. The pilot study with six experts was carried out during a meeting of the research group 'Computer science didactics' in the Netherlands. We asked the experts to sort the algorithmic problems into groups using algorithm as the criterion. Participants were handed 16 cards and were asked to sort them into more than one and fewer than 16 groups, and to provide a name for each group (i.e., an unframed task condition). We tested two different settings: in the first group, all participants read the problem descriptions together and decided whether this problem was defined by a new category or could be added to an existing category. In the second group, participants read the problems individually and made their own sorting. When all had finished, each participant explained the groups he or she had identified, and they discussed the different groupings. The explanations and discussions of both groups were audiotaped.

This pilot revealed that the second setting (individual sorting followed by explanation and discussion) provided more data regarding the reasoning of participants when they explained and discussed the different groupings. Based on the results of the pilot, the procedure to start with individual sorting was adopted and the problem descriptions were modified. Furthermore, we decided to change the unframed task condition into a framed task condition and to give students the different categories for sorting, because

Table 1
Problems used in the card sorting task (see Appendix 1 for problem descriptions)

Title of the problem	Underlying algorithm
Shopping at Shoes World	Binary search
Hospitals	Intractable
Track the thief	Binary search
Give me the change!	Intractable
Mega WoodLand discount	Intractable
Signal Fire	Shortest path algorithm
No turning left!	Shortest path algorithm
Apple in the basket	(Partial) Sorting algorithm
Putting people in line	Sorting algorithm
Collecting candies	Shortest path algorithm
A postman	Intractable
Knight's Tour	Intractable

participants in the pilot study indicated that the current task required a well-developed power of abstraction and might hinder our efforts to examine students' reasoning.

In the next phase of developing the card sorting task, we conducted a pilot study with ten secondary school students who took CS classes taught by the first author. We used the 16 problem cards with the modified descriptions. This time we used a framed task condition protocol and gave students four different categories: sorting algorithms, searching algorithms, shortest path algorithms, and intractable algorithms (i.e., no efficient algorithm known). These categories correspond to the algorithms that students have learned in the previous lessons and the selected problems could therefore be matched to one of those categories. In case students thought a card did not fit in any of the given categories, they could come up with a new category. First, students were offered the problem cards individually, and were asked to sort them into groups using algorithm as the criterion. The next step was a discussion in a focus group, during which the students were asked to come up with a final grouping. They explained to each other the grouping they had made and were asked to discuss the underlying reasons. We tested specifically whether all problem descriptions were clear, and the time needed by students to read all the problem cards. This pilot showed that it took the students a long time to read all cards properly and that they became tired of it. The cards with descriptions that were not clear or that did not contribute to the real discussions were therefore discarded, resulting in a set of 12 problem cards (see Table 1). All problem cards are included in Appendix 1.

3.3. Participants

Twelve students were asked to participate in the main study, including six students from a HAVO 4 class (senior general secondary education) and six students from a VWO 4 class (pre-university education), which is comparable to grade 10 (students aged 15–16 years). Both classes were taught by the same teacher. In the weeks before the study,

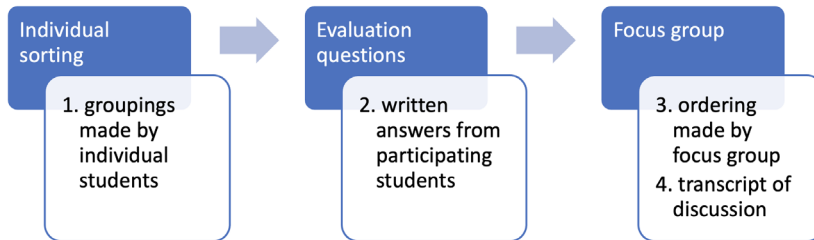


Fig. 3. Phases in a session for data collection and the resulting data sources.

students had attended lessons regarding standard algorithms such as Dijkstra's shortest path algorithm and intractable algorithms such as the knapsack problem. Students were selected to participate by their teacher, who knew whether these students have a basic knowledge of the algorithms taught. The teacher was asked to select students who, in his opinion, understood the teaching material and were able to express themselves well. In addition, the teacher was asked to form four groups of three students who perceive each other as students with equal abilities to avoid the students turning to the 'smartest kid' for the correct answer. Our research proposal was approved by the local Ethics Committee and the data were collected, stored, and analyzed in accordance with the ethical conduct of research.

3.4. Data Collection

The data were collected during sessions consisting of three phases, resulting in various data sources (see Fig. 3). Three students participated in each session; thus, we conducted four sessions for the twelve participants. The sessions lasted approximately 45 -60 min. The session started with an introduction to explain that this study was aimed at examining how students use their algorithmic knowledge in new contexts. The slides that were used during lessons about algorithms were shown to refresh students' memory. Students were asked to individually read the problem cards and sort them into four categories (searching, sorting, shortest path, and intractable algorithms), or a new category if needed. They were instructed that we did not require a correct solution for the algorithmic problem, only the identification of the underlying algorithm. To create a safe environment, the students were told that the task was not a test and that results would not be graded. The students could use as much time as needed to sort the cards.

After the individual sorting, the result was recorded, and each student was given a sheet with two evaluation questions to answer:

1. For which problem(s) did you immediately recognize the category to which it belongs? Why?
2. For which problem(s) did you find it difficult to determine the correct category? Why?

Subsequently, the students formed a focus group and were asked to come up with a joint grouping. For each algorithmic problem, they compared their individual results,

explained to each other why they had matched a problem to a certain category, and discussed these reasons until they reached agreement. The results of the focus group sorting were captured, and the discussions and conversations in the focus groups were audio- and video-recorded. The audiotapes of the discussion have been transcribed verbatim.

3.5. Data Analysis

We analyzed the data to examine how students recognize algorithmic concepts and what factors play a role in their reasoning. The data analysis consisted of three consecutive steps. During the first step, we investigated how students recognized algorithmic concepts in the given problems. We examined the results of both the individual groupings and the ordering by the focus groups (data sources 1 and 3, see Fig. 3). The results of the individual sorts are described in Table 2. An asterisk indicates the correct category. Results of the focus group sorts are described in Table 3.

We studied the extent to which the students had correctly matched the problem cards to the algorithmic concepts and investigated the variation in answers for each problem card. In addition, we analyzed the answers to the evaluation questions (data source 2, see Fig. 3) and prepared a table listing the problems that were either mentioned as ‘easy to match’ or ‘difficult to determine where it belongs’. Each of the lists was sorted by the number of times a problem card was mentioned (see Table 4).

The results of this first step in the data analysis revealed widespread agreement for some problem cards and a large variation in answers for other problem cards, along with a list of problems considered the most ‘easy’ or ‘difficult’ to match. These results were beneficial for the next steps, which were aimed at delving deeper to examine students’ recognition of algorithmic concepts and the factors that play a role in this recognition. For example, we expected that for a problem with a variety of answers and listed as ‘dif-

Table 2
Results of individual sorts

Title of the problem	Searching algorithm	Sorting algorithm	Shortest path algorithm	Intract-able algorithm	New category	Unknown
Shopping at Shoes World	8 *	3	1			
Track the thief	8 *	3		1		
Apple in the basket	1	11 *				
Putting people in line		12 *				
No turning left!	1		11 *			
Collecting candies	2		7 *	2		1
Signal fire		1	7 *	1		3
Give me the change!	2	3	1	4 *	1	1
Mega WoodLand discount	3	2		7 *		
A postman	3		4	4 *		1
Knights’ Tour	1		6	5 *		
Hospitals	1		8	2 *		1

Table 3
Results of focus group sorts

Title of the problem	Searching algorithm	Sorting algorithm	Shortest path algorithm	Intract-able algorithm	New category	Unknown
Shopping at Shoes World	4 *					
Track the thief	3 *	1				
Apple in the basket	1	3 *				
Putting people in line		4 *				
No turning left!			4 *			
Collecting candies			3 *		1	
Signal fire			2 *	1	1	
Give me the change!				2 *	2	
Mega WoodLand discount				4 *		
A postman	1		2	1 *		1
Knights' Tour			1	2 *	1	
Hospitals		1	1	2 *		

Table 4
'Easy' or 'Difficult' to match problems, according to students

Title of the problem	Frequency of 'easy'	Title of the problem	Frequency of 'difficult'
Putting people in line	6	Signal fire	9
Mega WoodLand discount	5	A postman	7
Apple in the basket	4	Give me the change!	6
Track the thief	4	Hospitals	5
Shopping at Shoes World	2	Mega WoodLand discount	4
No turning left!	2	Collecting candies	4
Collecting candies	2	No turning left!	3
Give me the change!	1	Knights' Tour	3
A postman	1	Shopping at Shoes World	2
Knights' Tour	1	Track the thief	1
Signal fire	0	Apple in the basket	1
Hospitals	0	Putting people in line	1

difficult to match' we would find more factors, because students would likely use different reasons to explain the choices they made during the sorting.

In the second step, we examined how students recognized algorithmic concepts in more detail by investigating whether students recognized the underlying algorithmic concepts directly (route A, Fig. 1) or whether they recognized similarities with a previously solved problem (route B, Fig. 1). In addition, we analyzed whether these criteria led to a match or a mismatch: did the recognition lead to the student grouping the problem card to a corresponding algorithm, or to a non-corresponding algorithm. The combination of these two lenses to examine students' reasoning about their groupings resulted in a framework for the analysis of the focus group discussions (see Table 5).

Table 5
 Framework for analysis of focus group discussions

	Match	Mismatch
Direct recognition of underlying algorithmic concepts (route A)	Problem matched to corresponding algorithm with recognition of algorithmic concepts	Problem matched to non-corresponding algorithm with recognition of algorithmic concepts
Similarity with other problem (route B)	Problem matched to corresponding algorithm with use of similarity to other problem	Problem matched to non-corresponding algorithm with use of similarity to other problem

We reread the transcripts of the focus group discussions (data source 4, see Fig. 3) and coded these transcripts using Atlas.ti qualitative data analysis software. A combination of deductive and inductive coding strategies was used to develop a code list for the data analysis. We started with four ‘master codes’ for the type of recognition (direct or via other problem; route A or B, Fig. 1) and the result of the recognition (match or mismatch). Subcodes to describe the character of the types of recognition were developed inductively using in Vivo coding (Miles *et al.*, 2014). We coded each segment in the transcript where a student explained or reasoned about his or her match of a problem card and an algorithm, labeling such a segment with a ‘result code’ and a ‘recognition code’. For example, a reasoning such as “*this problem is like the knapsack -problem and therefore it should be matched with ‘intractable algorithms’ because it is about something that you put in your knapsack*” would be coded as ‘match’ and ‘similarity with problem -wording’ (the student matched the problem to a corresponding algorithm based on specific words of the problem). The query tool of the Atlas.ti software permitted the analysis of all coded segments with a combination of specific recognition and result codes, allowing us to interpret and gain insight into these combinations.

As a third step, we analyzed the data to investigate what factors play a role in students’ recognition of algorithmic concepts (the second research question). We repeatedly read the responses to the evaluation questions (data source 2, see Fig. 3) and the transcripts of the focus group discussions (data source 4). All statements that indicated an aspect that played a role in their reasoning have been coded. We used descriptive coding (Miles *et al.*, 2014) to summarize the basic topic of the segments in a word or short phrase, e.g., the ‘visual elements’. This analysis revealed five categories: ‘Prior knowledge of algorithmic concepts’, ‘Understanding of the problem’, ‘Representation of the problem’, ‘Degree of similarity to problems discussed during lessons’ and ‘Algorithm partially solves the problem’. We investigated whether each category was related to the problem (e.g., the representation of the problem) or to the student (e.g., prior knowledge).

The resulting factors are described in more detail in section 4.2. *Factors that Play a Role in Students’ Recognition.*

To ensure the trustworthiness of the qualitative analysis, the first and second authors met almost every week to discuss the analysis and the coding process. The first author was the main coder. The second author checked the codes and patterns for the

appropriateness of the first author's interpretation. During meetings with all authors (one per every six weeks), the process and the preliminary results were discussed until consensus was reached.

4. Results

In this section, we first report the results for our first research question: How do students recognize algorithmic concepts in new problems? Furthermore, the results of the analysis of the factors that play a role in students' reasoning (research question 2) are reported. The students' quotes were translated from Dutch to English. We have indicated whether each quote was taken from the focus group discussion (FGx, where x is the number of the focus group) or from the individual answers to the evaluation questions (EQ). The student names are pseudonyms.

4.1. Recognition of Algorithmic Concepts

As described in the *Method* section, we used a framework that offers two lenses for the analysis of the data: type of recognition and result of the recognition (see Table 5). The type of recognition describes whether students recognized the algorithmic concepts directly (route A in Fig. 1) or whether they observed a similarity with a previously solved problem (route B in Fig. 1). The result of the recognition refers to matching problems to a corresponding algorithm (match) or non-corresponding algorithms (mismatch). Combining these lenses yields four different combinations, which are described below.

Direct recognition & Match. In this situation, students matched algorithmic problems to analogous algorithms based on their recognition of the underlying algorithmic concepts. For example, a problem that concerns sorting (here, either apples or people) is recognized as 'sorting'. The combination of 'direct recognition' and 'match' was found predominantly in the discussion regarding *searching* or *sorting* problems. Students were able to recognize the sorting character of such problems. Alyssa, for example, commented on 'Apple in the basket':

"You put an apple in the middle and examine whether "[the other apples]" are bigger or smaller, and that way you can sort easily"
(Alyssa, FG1)

The concept of the binary search algorithm was also transferred to matching problems based on the recognition of algorithmic concepts (such as 'works only when items are sorted', 'start in the middle', etc.). Frank commented on 'Shopping at Shoes World', a searching problem about finding shoes of the right size in a sorted pile of shoes without the sizes written on the shoe boxes:

"Since the boxes are in the right order, I thought it would be most useful if you start in the middle, pick up the box, and check 'is my size

bigger or smaller than the one I picked up?’. And when it is bigger, yeah, you only have to search in the half that is left...then eventually you’ll get to the right one most quickly.” (Frank, FG2)

Direct recognition & Mismatch. Only a few cases were found where recognizing the underlying algorithmic concept was combined with a mismatch. These cases reveal that, to a certain extent, students seem to observe the concepts (such as searching in a sorted list and starting in the middle for the binary search algorithm) but still match the problem to a non-analogous algorithm. For example, Alyssa matched the abovementioned shoe boxes problem to the *sorting* algorithm, although she recognized the concept of the binary search algorithm:

“I thought ‘sorting’. They are in order so you start in the middle and check if your shoe size is smaller or bigger, then you get there faster than if you had to go through everything.” (Alyssa, FG1)

Similarity with previously solved problem & Match. In this case, the students assigned a problem to the matching algorithm by looking mostly at similarities with a previously solved problem, such as similar words or phrases. This is clearly noticeable in the discussion of two students regarding ‘Mega WoodLand discount’, a problem comparable to the ‘knapsack problem’. The knapsack problem was discussed during class:

“You have a set of boxes that differ in weight and value, and you have a knapsack that can hold 12 kg. Which boxes do you put in the knapsack in order to take the highest value with you?”

During our research, students were handed a card with a similar problem, but instead of boxes the problem concerned products, and instead of the values of the boxes, the value of discount for each product was listed. In focus group 3, the following discussion took place:

Isabel: *“this one looked the most like the example. It looked like the **knapsack problem** in every way ...”*

Hanna: *“there is also a ‘knapsack’ in it”*

Isabel: *“oh, I did not even see that, but it looks like it ...I did not explain my reasoning well”*

Likewise, the students in focus group 4 discussed the ‘Collecting candies’ problem, and one of the students mentioned a similarity to a problem discussed during the lesson:

“Instead of kilometers as we did during the lesson, it’s just candies.” (Kevin, FG4)

Similarity with previously solved problem & Mismatch. Students recognized several similarities with previously solved problems which in the end led to a mismatch. For example, the phrasing of a problem could suggest a similarity with a non-analogous problem. Several problems with the words ‘shortest’ or ‘fastest’ were matched to ‘shortest path algorithms’. In this way, Benjamin matched the abovementioned shoe boxes problem to ‘shortest path algorithms’:

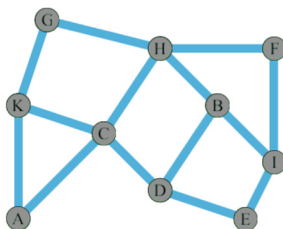


Fig. 4. Map from one of the problem cards.

*“I chose ‘shortest path’ because I thought you were looking for the **shortest** way to find a shoe size.” (Benjamin, FG1)*

Likewise, the fact that you had to search for a solution caused a match with searching algorithms:

*“I chose ‘searching’ because you are **searching** for it.” (Daniel, FG2)*

The illustration of a problem might also lead to a mismatch; for example, when presented with an intractable problem that was illustrated with a map (example in Fig. 4), the students in focus group 3 commented:

Gabrielle: *“yeah, I chose ‘shortest path’ ...I saw a map and I thought:”*

Isabel: *“that belongs to ‘shortest path’ ”*

Gabrielle: *“exactly”*

4.2. Factors that Play a Role in Students’ Recognition

The data analysis revealed various factors that play a role in students’ recognition of algorithmic concepts: ‘Prior knowledge of algorithmic concepts’, ‘Understanding of the problem’, ‘Representation of the problem’, ‘Degree of similarity to problems discussed during lessons’, and ‘Algorithm partially solves the problem’. The factors have been categorized into those which are related to the student and those that may be linked to the problem (see Table 6, in which student quotes are listed as illustrative examples).

Regarding the factors related to the student, it is apparent that students’ understanding of the problem enables them to recognize algorithmic concepts. In addition, prior knowledge of algorithmic concepts is required for this recognition. Likewise, we found that students’ incomprehension or misunderstanding hinders their recognition of algorithmic concepts.

The factors that may be linked to the problem can sometimes hinder and sometimes enable the recognition of underlying algorithmic concepts; for example, the problem description may indicate similarities and facilitate the recognition, but it may also cause confusion. In addition, the degree of similarity with a previously seen problem determines the extent to which it is supportive for the recognition of similarities.

Table 6
Factors that play a role in students' recognition of algorithmic concepts

Type	Factor	Illustrative examples
Student	Understanding of the problem	Because you compare these two things ...and that way you sort from low to high
	Prior knowledge of algorithmic concepts	I thought 'shortest path'; it is actually just the same concept
Problem	Representation of the problem	All points are connected
	Degree of similarity to problems discussed during lessons	We had discussed a similar example in class
	Algorithm partially solves the problem	So then you have to use the shortest path algorithm plus something else

The last factor, *algorithm partially solves the problem*, describes a common aspect of solving problems, since a new problem may be more comprehensive than a previously solved one. Despite this, the existing solution may solve part of the new problem and might therefore be useful; for example, for problems that are quite dissimilar to those that have been discussed before (far transfer), it might very well be that a standard algorithm only partially solves the new problem, which is apparent in the case described below.

4.2.1. Description of the 'Signal Fire' Problem to Illustrate Factors

In this section, we describe a case to demonstrate the factors that play a role in student reasoning. We selected the problem that ranked highest on the list of problems that were difficult to categorize: 'Signal Fire' (see Fig. 5). This problem was reported as 'difficult to determine' by nine of 12 students. In addition, this problem card was matched to a wide variety of algorithms during the sorting task. Because of the apparent difficulty of the problem, we expected a comprehensive exchange of thoughts regarding this problem.

Signal fire

A long time ago in Japan, some Ninjas served the beaver community to help them defend their beaver lodge. In case of emergency, they used smoke signals to communicate with each other.

In the figure at the right, the red point is the location of the beaver lodge. Each blue point is a location where a smoke signal could be lit. Also, two points are joined by a line if their smoke signals can be seen from each other.

At every point, there are some beavers who stand on all day long. They fire a smoke when they see a signal from a point joined to theirs, just 1 minute after this signal was fired.

When a smoke signal is lit at the beaver lodge, how much later will there be a signal lit at all points?

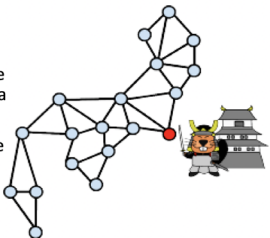


Fig. 5. Signal Fire: An example of a Bebras task used in the study.

During the lessons before this study, the students learned to work with Dijkstra's shortest path algorithm to find the shortest route between two cities. Another example was discussed in class where traveling time instead of distance between cities is used, demonstrating that the algorithm can also be used to find the fastest route. Some students used this **prior knowledge of algorithmic concepts** when they explained to each other why they had matched the problem card 'Signal Fire' to the 'shortest path' algorithm:

"It is just the same method you use for a shortest route: you check every path and you continue until you have dealt with the whole map"
(Isabel, FG3)

The 'Signal Fire' problem may be characterized as far transfer because it is rather different from the original context in which the 'shortest path' algorithms were learned. Students were challenged in their **understanding of the problem**. In the answers to the evaluation questions, we found various reasons why students listed the 'Signal Fire' problem as 'difficult to determine':

"Signal fire: I do not understand it very well." (Kevin, EQ)

Another student individually categorized this problem as 'shortest path' but during the focus group he first stated that "that's not correct". He then reread the problem and exclaimed:

"Oh, now I get it! This is a shortest path algorithm but reversed; you have to find the longest path because you assume that everyone lives somewhere, right? Of course, this is a shortest path; you find the one that has the longest path from the red point and that is how long it takes." (Lucas, FG4)

The **representation of the problem** (i.e., the description, the use of signal words, or the visual elements) also appeared to be a factor that played a role in student reasoning; for example, the words "a signal lit at all points" caused confusion to Benjamin, who reported during the discussion in focus group 1:

"Because, in my opinion, the signal has to go everywhere so you do not need just the shortest path; you need to reach every point, I think."
(Benjamin, FG1)

However, Caleb replied:

"but you have to look for the furthest ..."

Then Alyssa read the problem description out loud: "how much later will there be a signal lit at all points?" and Caleb added:

"so you only need to know the furthest because in the meantime all the other points are lit as well, so it is the shortest path to the furthest point."

The description of the problem apparently caused confusion for Benjamin but was clear to Caleb. Reference was also made to missing signal words in the problem description; for example, Gabrielle noted in the evaluation questions that ‘Signal Fire’ was difficult to determine because:

“it has a route but no distances.” (Gabrielle, EQ)

In focus group 2, there was a lot of discussion regarding this problem. Ezra noticed a **similarity to a problem discussed during lessons** when he commented:

“It looks like shortest path, because you have to calculate how long it takes, similar to shortest path.”

Frank contradicted that it could not be ‘shortest path’:

“It is not really ‘shortest path’ ...the fact is that the red point is lit and the three adjacent blue points will be lit next, and then the ones that surround these. So ...you cannot search for a specific path.”

But when Ezra claimed that it should be ‘shortest path’, Frank responded:

“Or you have to find the shortest path to the furthest point, but you don’t know which one is the furthest.”

The students kept discussing that the ‘shortest path’ algorithm would only **partially solve the problem**, by allowing you to find the distance to each location. But instead of realizing that the maximum length of these various paths would be the correct answer, they proposed using a ‘sorting algorithm’:

“You need to know when the furthest point is lit, because then all points will be lit, so you will need a sorting algorithm ...because as Frank just said, you need to sort which one is further and which is less far.” (Ezra)

In focus group 3, the students commented that an algorithm may not solve the problem completely because the problem requires the opposite outcome to that which the algorithm is intended to solve:

“It is actually exactly the opposite; we have to find the longest path.”
(Hannah, FG3)

5. Conclusions and Discussion

Teaching algorithmic thinking, whether in CS or in the broader field of CT, is aimed at enabling students to use their knowledge in various contexts and to reuse existing solutions to algorithmic problems. When encountering a new problem, students are therefore required to recognize algorithmic concepts, either directly or via a previously solved problem (see Fig. 1). In this study, we examined how students recognize the underlying

ing algorithm in a new problem. We developed a card sorting task to investigate how students arranged algorithmic problems into groups, and we examined their reasoning when they explained their groupings to student peers. Our findings reveal that students may recognize the underlying algorithmic concepts directly, which nearly always leads to the students connecting the problem to the correct algorithm (match). In addition, our results show that students used similarities with previously solved problems to connect problems to underlying algorithms. We found that this identification of similarities may lead to a match or a mismatch (connection to a correct or incorrect algorithm). Furthermore, by examining students' reasoning, we identified the factors that play a role in the recognition of algorithmic concepts, such as the representation of the problem or the degree of similarity to problems discussed during lessons.

As described before, contextualization is an essential characteristic of problem solving in CS or CT. The problem occurs in a specific context, but the solution is developed on a conceptual level. Our findings suggest that the recognition of algorithmic concepts is more successful when students recognize these concepts directly at the conceptual level. Similarities at the context level may support students in recognizing the corresponding concepts, but may also distract students and lead them to link the problem to a non-analogous algorithmic concept.

In the following sections, we describe the insights gained through this study regarding students recognizing algorithmic concepts and influencing factors. We address possible implications, and conclude with the limitations of this study and our recommendations for future work.

5.1. *Recognizing Algorithmic Concepts*

When encountering new algorithmic problems, the participating students were more able to classify the underlying concepts if they could recognize similarities with a problem or an underlying algorithm that had been previously discussed in class. Particularly when they recognized algorithmic concepts directly, they were predominantly able to connect a problem to a matching algorithm. Most references for algorithmic concepts were made regarding sorting and searching algorithms, suggesting that algorithmic concepts were easier to recognize for new sorting or searching problems than for problems where shortest path algorithms or intractable algorithms could be used. This might be related to the fact that the structure of sorting or searching problems is easier to understand, and therefore may be easier to recognize. Intractable problems are known to be difficult for students (Gal-Ezer and Trakhtenbrot, 2016). That might clarify why, for some of the intractable problems, it seemed to be of minor importance whether a new problem is closely similar or quite dissimilar to problems that have been discussed in class. It is entirely possible that students did not know how to handle these problems and were therefore not able to use prior knowledge or experiences, which is consistent with previous research (e.g., Bransford *et al.*, 2000; Bassok, 2003).

It is noteworthy that sometimes students' reasoning points to the recognition of algorithmic concepts, but still leads to a mismatch. It seems that students have difficulty connecting the right 'term' to the insight they have.

Furthermore, our results reveal that students hesitate in expressing the similarities they have found (e.g., “it is just similar ...I did not explain my reasoning well” during the discussion in focus group 3 about ‘Mega WoodLand discount’). This might be due to the difficulty of correctly explaining the choices made. Students might even have recognized the concept, but as algorithmic concepts may be more difficult to clarify, students might prefer to point to similarities with other problems.

It is expected that the recognition of similarities with earlier solved problems may support students in finding a solution for a new problem. This is, for instance, the essence of the model of learning as suggested by case-based reasoning (Kolodner *et al.*, 2003b). However, we found no evidence that this route is successful for recognizing algorithmic concepts in difficult problems, such as intractable problems.

5.2. Factors that Play a Role in Students’ Recognition

Our study revealed several factors that play a role in students’ recognition of algorithmic concepts. In the Introduction, we described how students, when confronted with a new problem, may see the similarity with a previously solved problem or with the underlying algorithms (see Fig. 1). From this perspective, it is interesting to consider whether the factors that play a role in students’ recognition relate to the problem (or context) or the concepts. This is related to the discussion about context-based learning. When students learn a new concept in only one specific context, they may connect the new knowledge to that specific context (Guzdial, 2010). When confronted with a new problem with the same underlying concept, they might recognize the similarity only when the new problem is presented in the same context. Our findings indicate that both levels (concepts and contexts) can be found in the factors that play a role in students’ reasoning. The factors ‘prior knowledge of algorithmic concepts’ and ‘algorithm partially solves the problem’ appear to be related to the concept level. On the other hand, the factors ‘understanding of the problem’, ‘representation of the problem’, and ‘degree of similarity of problems discussed during lessons’ may be linked to the context of the problem.

The factors that we found may be connected to the student or to the problem (see Table 6). This is in line with the study of Kershaw *et al.* (2013), who distinguished the properties of the problem and student prior knowledge or experiences as constraints that both may impede the application of prior knowledge. In addition, our findings demonstrate that the factors that enable transfer, such as ‘understanding of the problem’ and ‘representation of the problem’, may also be distinguished as factors relating to the student and factors relating to the problem.

5.3. Implications

This study provides insight into the recognition of algorithmic concepts in new situations or contexts. Our findings show that both the direct recognition of underlying algorithmic concepts or the identification of similarities with previously solved problems

will support students in applying what they learn in different contexts; however, direct recognition more often leads to connecting a problem to a matching algorithm. These findings may help us to understand what skills students need to develop to recognize algorithmic concepts and reuse existing solutions for algorithmic problems. This exploratory study suggests that recognizing underlying algorithmic concepts may be a learning goal that needs more attention in CS education. The analysis of students' reasoning points to the significance of teaching students how to identify the structure of the algorithmic problem. Improving students' abstraction and decompositions skills may contribute to this, as proposed by Armoni (2013) and Muller and Haberman (2008). By teaching students to focus on the underlying algorithmic concepts when encountering a new problem, they may realize that what is learned in a specific context may be useful in another context.

In addition, the used research method may be applied as a learning activity, where teachers show how to recognize these concepts and where students are encouraged to share their reasoning with their peers. The card sorting task developed for this study proved to be useful for examining students' reasoning regarding the matching of the problem cards with standard algorithms. Although we did not monitor the thinking of students while matching the cards, the recorded discussions of students explaining their results to peers in the focus groups provided ample opportunity to gain insight into their reasoning. This type of card sorting task could be included in teaching materials for teaching algorithms because it appeared to be an interesting activity that helped students practice with recognizing algorithmic concepts and applying what they had learned in a different situation. The results of this study suggest that it is important for students to explain their choices and reasoning to peers because the active discussions seemed to contribute to a clear focus on the underlying algorithmic concepts.

Our findings also indicate several factors related to new problems that play a role in students' reasoning, such as 'representation of the problem' and 'degree of similarity to problems discussed during lessons'. We therefore recommend that, to ensure well-structured learning trajectories, attention should be paid to the representation of the problems and that tasks should build up from near transfer to far transfer with regard to applying what is learned.

5.4. Limitations and Future Work

In this study, a limited number of students participated, and we investigated how they recognized algorithmic concepts in new situations after receiving only a first introduction into the topic of algorithms. It is therefore obvious that we should not make any generalizations; however, the detailed research of this qualitative study has revealed more insights into the important role of the recognition of algorithmic concepts. It would be interesting to further examine this topic in connection with the education provided, exploring how well students recognize algorithmic concepts after receiving more lessons on this subject.

In this article, we focused on the recognition of algorithmic concepts and the factors that play a role in this process within the domain of CS education. Further research is

needed to examine whether the same factors play a role when students apply their algorithmic knowledge to other domains.

This study focused on enabling students to use their algorithmic knowledge and skills in various contexts by examining how they recognize underlying algorithmic concepts. Further work is needed to determine the effects of students using their creativity when designing relevant contexts for the algorithmic concepts they have learned. A future study regarding students developing contexts that contribute to learning algorithms would be worthwhile.

Acknowledgments

We would like to thank all the students who participated in this study.

Funding

This research received funding from the Dutch Ministry of Education, Culture, and Science under the Dudoc program.

References

- Armoni, M. (2013). On teaching abstraction in CS to novices. *Journal of Computers in Mathematics and Science Teaching*, 32(3), 265–284.
- Barendsen, E., Grgurina, N., Tolboom, J. (2016). A new informatics curriculum for secondary education in the Netherlands. In: Brodnik, A., Tort, F. (Eds.), *Informatics in Schools: Improvement of Informatics Knowledge and Perceptions*. Springer, Cham, pp. 105–117.
- Barendsen, E., Mannila, L., Demo, B., Grgurina, N., Izu, C., Mirole, C., Sentance, S., Settle, A., Stupuriene, G. (2015). Concepts in K-9 computer science education. In: *Proceedings of the 2015 ITiCSE on Working Group Reports*. Association for Computing Machinery, New York, NY, USA, pp. 85–116.
- Barr, V., Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1), 48–54.
- Bassok, M. (2003). Analogical transfer in problem solving. In: Davidson, J.E., Sternberg, R.J. (Eds.), *The Psychology of Problem Solving*. Cambridge University Press, Cambridge, MA, pp. 343–369.
- Bebras (2021). Bebras -International challenge on informatics and computational thinking. <https://bebras.org>
- Beverwedstrijd (2018). Beverwedstrijd -The Dutch Bebras contest. www.beverwedstrijd.nl
- Bransford, J.D., Brown, A.L., Cocking, R.R. (2000). *How People Learn (Expanded ed.)*. National Academy, Washington, DC.
- Brinda, T., Napierala, S., Tobinski, D., Diethelm, I. (2019). Student strategies for categorizing IT-related terms. *Education and Information Technologies*, 24(3), 2095–2125.
- CAS & CS4FN (2018). Teaching London Computing: A resource hub from CAS London & CS4FN. <https://teachinglondoncomputing.org>
- Catrambone, R., Holyoak, K.J. (1989). Overcoming contextual limitations on problem-solving transfer. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(6), 1147–1156.
- Chi, M.T.H., Feltovich, P.J., Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5(2), 121–152.
- CSTA (2017). Computer Science Teachers Association K-12 computer science standards, Revised 2017. <http://www.csteachers.org/standards>

- Curzon, P. (2014). Unplugged computational thinking for fun. In: Brinda, T., Reynolds, N., Romeike, R., Schwill, A. (Eds.), *KEYCIT 2014 -Key Competencies in Informatics and ICT*. Universitätsverlag Potsdam, Postdam, Germany, pp. 15–28.
- Curzon, P. (2015). Computational Thinking: Puzzling tours. Computer Science for Fun, CS4FN, London. <http://www.cs4fn.org/graphs/puzzlingtours>
- Dagiënė, V., Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In: Mittermeir, R.T., Sysło, M.M. (Eds.), *Informatics education -Supporting computational thinking*. Springer, Berlin, pp. 19–30.
- Futschek, G. (2006). Algorithmic thinking: The key for understanding computer science. In: Mittermeir, R.T. (Ed.), *Informatics Education – The Bridge between Using and Understanding Computers*. Springer, Berlin, pp. 159–168.
- Futschek, G., Moschitz, J. (2010). Developing algorithmic thinking by inventing and playing algorithms. In: *Proceedings of Constructionist Approaches to Creative Learning, Thinking and Education*.
- Gal-Ezer, J., Stephenson, C. (2014). A tale of two countries: Successes and challenges in K-12 computer science education in Israel and the United States. *ACM Transactions on Computing Education*, 14(2).
- Gal-Ezer, J., Trakhtenbrot, M. (2016). Identification and addressing reduction-related misconceptions. *Computer Science Education*, 26(2–3), 89–103.
- Grover, S., Pea, R. (2018). Computational thinking: A competency whose time has come. In: Sentance, S., Barendsen, E., Schulte, C. (Eds.), *Computer Science Education: Perspectives on Teaching and Learning in School*. Bloomsbury Publishing, London, pp. 19–38.
- Guzdial, M. (2010). Does contextualized computing education help? *ACM Inroads*, 1(4), 4–6.
- Harel, D., Feldman, Y. (2004). *Algorithmics: The Spirit of Computing* (3rd ed.). Pearson Education Limited, Harlow, England.
- Haskell, R.E. (2001). *Transfer of Learning*. Academic Press, San Diego, CA, USA.
- Irby, S.M., Phu, A.L., Borda, E.J., Haskell, T.R., Steed, N., Meyer, Z. (2016). Use of a card sort task to assess students' ability to coordinate three levels of representation in chemistry. *Chemistry Education Research and Practice*, 17(2), 337–352.
- Izu, C., Mirole, C., Settle, A., Mannila, L., Stupuriene, G. (2017). Exploring Bebras tasks content and performance: A multinational study. *Informatics in Education*, 16(1), 39–59.
- Izu, C., Schulte, C., Aggarwal, A., Cutts, Q., Duran, R., Gutica, M., Heinemann, B., Kraemer, E., Lonati, V., Mirolo, C., Weeda, R. (2019). Fostering program comprehension in novice programmers -Learning activities and learning trajectories. In: *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '19)*. ITiCSE-WGR '19. Association for Computing Machinery, New York, NY, USA, pp. 27–52.
- Kallia, M., van Borkulo, S.P., Drijvers, P., Barendsen, E., Tolboom, J. (2021). Characterising computational thinking in mathematics education: A literature-informed Delphi study. *Research in Mathematics Education*, 0(0), 1–29.
- Kershaw, T.C., Flynn, C.K., Gordon, L.T. (2013). Multiple paths to transfer and constraint relaxation in insight problem solving. *Thinking & Reasoning*, 19(1), 96–136.
- Kolodner, J.L. (1993). *Case-based Reasoning*. Morgan Kaufmann, San Mateo, CA, USA.
- Kolodner, J.L., Gray, J.T., Fasse, B.B. (2003a). Promoting transfer through case-based reasoning: Rituals and practices in Learning by Design™ classrooms. *Cognitive Science Quarterly*, 3, 119–170.
- Kolodner, J.L., Camp, P.J., Crismond, D., Fasse, B., Gray, J., Holbrook, J., Puntambekar, S., Ryan, M. (2003b). Problem-based learning meets case-based reasoning in the middle school science classroom: Putting learning by design into practice. *Journal of the Learning Sciences*, 12(4), 495–547.
- McCauley, R., Murphy, L., Westbrook, S., Haller, S., Zander, C., Fossum, T., Sanders, K., Morrison, B., Richards, B., Anderson, R. (2005). What do successful computer science students know? An integrative analysis using card sort measures and content analysis to evaluate graduating students' knowledge of programming concepts. *Expert Systems*, 22(3), 147–159.
- Miles, M.B., Huberman, A.M., Saldana, J. (2014). *Qualitative Data Analysis: A Method Sourcebook*. Sage Publications, Los Angeles, CA, USA.
- Muller, O., Haberman, B. (2008). Supporting abstraction processes in problem solving through pattern-oriented instruction. *Computer Science Education*, 18(3), 187–212.
- Nentwig, P.M., Demuth, R., Parchmann, I., Ralle, B., Gräsel, C. (2007). Chemie im Kontext: Situating learning in relevant contexts while systematically developing basic chemical concepts. *Journal of Chemical Education*, 84(9), 1439–1444.
- Rugg, G., McGeorge, P. (2005). The sorting techniques: A tutorial paper on card sorts, picture sorts and item sorts. *Expert Systems*, 22(3), 94–107.

- Schmid, U., Wirth, J., Polkeh, K. (2003). A closer look at structural similarity in analogical transfer. *Cognitive Science Quarterly*, 3(1), 57–89.
- Schwill, A. (1994). Fundamental ideas of computer science. *Bulletin -European Association for Theoretical Computer Science*, 53, 274.
- Selby, C., Woollard, J. (2013). Computational thinking: the developing definition. <https://eprints.soton.ac.uk/356481/>
- Shute, V.J., Sun, C., Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142–158.
- Smith, J.I., Combs, E.D., Nagami, P.H., Alto, V.M., Goh, H.G., Gourdet, M.A.A., Hough, C.M., Nickell, A.E., Peer, A.G., Coley, J.D., Others (2013). Development of the biology card sorting task to measure conceptual expertise in biology. *CBE-Life Sciences Education*, 12(4), 628–644.
- Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM*, 29(9), 850–858.
- Tursucu, S., Spandaw, J., de Vries, M.J. (2020). Search for symbol sense behavior: Students in upper secondary education solving algebraic physics problems. *Research in Science Education*, 50, 2131–2157.
- Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35.
- Yadav, A., Hong, H., Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, 60(6), 565–568.
- Zendler, A., Spannagel, C. (2008). Empirical foundation of central concepts for computer science education. *Journal on Educational Resources in Computing*, 8(2), 1–15.

J. Nijenhuis-Voogt is a PhD candidate at the Institute for Science Education at Radboud University (Nijmegen, the Netherlands) and a computer science teacher in secondary education. Her research focuses on computer science education, specifically on teaching algorithms and algorithmic thinking and on the use of a context-based approach for CS education.

D. Bayram-Jacobs is an assistant professor at the Eindhoven School of Education (ESoE), Eindhoven University of Technology (TU/e). Her research interests are: Socioscientific Issues (SSI), science education for citizenship, Pedagogical Content Knowledge (PCK) of science teachers, formative evaluation of SSI lessons, and innovation in education.

P.C. Meijer is a professor of Teacher Learning and Development at the Radboud Teachers Academy at Radboud University (Nijmegen, the Netherlands). Her research focuses on teacher education, the development of teachers' professional identity, workplace learning and teaching for creative learning.

E. Barendsen is a professor of Science Education at Radboud University (Nijmegen, the Netherlands) and a professor of Computing Education at Open University of the Netherlands. His scientific interests include design-based and context-based teaching and learning in computer science and STEM subjects, computational thinking and its integration into the school curriculum, digital literacy, and teachers' practical knowledge, in particular Pedagogical Content Knowledge (PCK).

A. Appendix 1 -Problem cards

Unless indicated otherwise, all problems are based on Bebras tasks.

Bebras – International Challenge on Informatics and Computational Thinking, <https://www.bebas.org/>

Shopping at Shoes world

Beaver Sébastien is at *Shoes World*. He would like to buy a new pair of shoes. After searching for a long time, he finally found the model he wants. Unfortunately, the sizes are not written on the boxes, but only on shoes. In order not to make a mess in the store, he will only check one box at a time. He also knows the boxes are sorted in ascending order of size.



There are 20 boxes of the model he likes, and we assume that the size he is searching for is available.

How many boxes must Sébastien open at least to be sure to find his size?

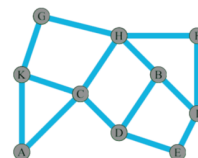
Hospitals

The beaver doctor wants to build a hospital on three islands. The islands are indicated on the right and have names from A to K.



He wants to build the hospitals in such a way that the beavers only have to swim through one channel to get to a hospital, no matter which island the beaver is on.

Choose three places for the three hospitals so that the condition is met.



Based on: a task of the *Beverwedstrijd* (Dutch Bebras challenge); *Stichting Informatica Olympiade*

Mega WoodLand discount

Mega WoodLand is making discount on all of their products. Beaver Dylan is going to buy new stuff for his home, but he cannot carry more than 15 kg in his backpack.

Here is a list of all the available products, with their weight and the value of the discount:

Product	Weight	Discount
Log	10 kg	€ 11
Statuette	8 kg	€ 10
Book	3 kg	€ 3



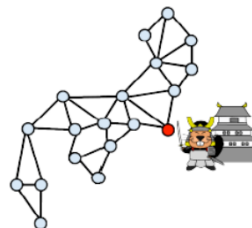
Which choice of products should Beaver Dylan purchase to get the most discount?

Signal fire

A long time ago in Japan, some Ninjas served the beaver community to help them defend their beaver lodge. In case of emergency, they used smoke signals to communicate with each other.

In the figure at the right, the red point is the location of the beaver lodge. Each blue point is a location where a smoke signal could be lit. Also, two points are joined by a line if their smoke signals can be seen from each other.

At every point, there are some beavers who stand on all day long. They fire a smoke when they see a signal from a point joined to theirs, just 1 minute after this signal was fired.



When a smoke signal is lit at the beaver lodge, how much later will there be a signal lit at all points?

Apple in the basket

Next to you there is a basket full of apples of different sizes.

Step 1: You take an apple out of the basket and put it on the table in front of you.

Step 2: You reach for the next apple out of the basket

- If the apple in your hand is smaller than the one on the table, then you put the apple from your hand into the other basket.
- If the apple in your hand is larger than the one on the table, then you put the apple on the table into the other basket and put the apple from your hand on the table.

You repeat the 2nd step until the initial basket is empty.

Which apple remains on the table at the end?



Note from the authors: this problem might be seen as a searching problem (searching for the largest apple) but also as a step in a sorting problem (the first step of the selection sort algorithm). This problem reminded most of the participants during both pilots to the selection sort algorithm, we therefore regarded this problem as matching to sorting algorithms.

Collecting candies

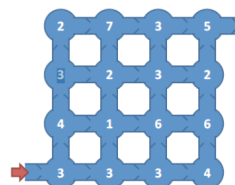
Beaver Bruno enters a cave consisting of several rooms connected by passages.

The passages are one-way only, Bruno can move from left to right and from bottom to top, but not in the two other directions.

There are some candies in each room (the numbers shown in white on the figure).

Bruno wants to collect as many candies as possible, but he is allowed to enter the cave just once.

How many candies can he collect?



Track the thief

The famous "Blue Diamond" was stolen from the museum today. A thief swapped the blue diamond with a cheap fake diamond with a green color.

Today 2000 people visited the exhibition with the diamond. They entered the room with the diamond one by one.



Inspector Bévar has to find the thief by questioning a number of visitors. He has a list of all the visitors in the order in which they visited the room with the diamond.

He asks each person the same question: "Have you seen a green or a blue diamond?"

Every visitor will answer truthfully except the thief, who will say he saw a green diamond.

Inspector Bévar wants to interrogate as few people as possible, but he wants to find the thief. Is it necessary that he interrogates all visitors? If not, how many?

Based on: a task of the Beverwedstrijd (Dutch Bebras challenge); Stichting Informatica Olympiade

Knight's tour

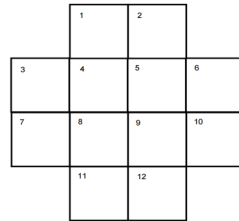


A chess knight is able to move like this:
 2 spaces forward and 1 sideways, or
 1 space forward and 2 spaces sideways.

A single chess knight is supposed to jump on a small cross-shaped board, see figure.

You must find a sequence of moves that starts from square 1 and finishes again in square 1.

Make sure the knight visits every square exactly once.



Based on "Computational Thinking: Puzzling tours: The Knight's Tour Puzzle"
 Created by Paul Curzon, Queen Mary University of London with support from the Mayor of London for Teaching London Computing: <http://teachinglondoncomputing.org>

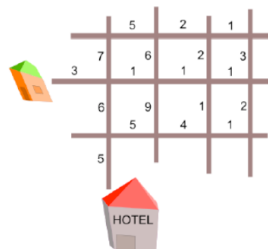
No turning left!

Traffic jam in a city! It is impossible to turn left in such a traffic density.

Beaver Floris is hurrying up home by car from the hotel he works. In the picture to the right, the streets are represented by the brown lines. The numbers indicate the travel duration for each street in minutes.

Floris uses his GPS navigation to find the shortest way home in minutes.

How long will it take at least to go from the hotel to his home when it is not possible to turn left?



Putting people in line

You are in charge of arranging a group of people in the correct order by the number on individual's shirts. The initial ordering is:

7 3 2 9 8 5 1 4 6

You will arrange individuals using the following technique:

Look at two consecutive people at a time, starting from the left.

If the person on the left has a number which is larger than that of the person on the right, switch the positions of those two people; otherwise, leave them in the order they are in

Move to the right one position, so that you are comparing one new person with one of the people just compared, and repeat the above comparison and potential swap.

Repeat step 3 until you have come to the end of the list.

How often do you have to go through the list until the list is in the order 1 2 3 4 5 6 7 8 9?

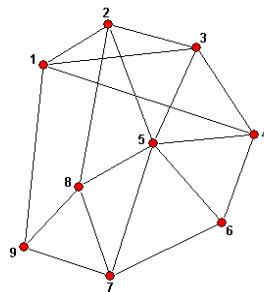


A postman

Beaver Polo is a postman. He delivers letters and parcels driving his car through different villages.

He has a map with all the roads between the villages, see the figure to the right.

Is it possible to deliver all letters and parcels passing each road only once? In which village should the beaver start his trip and in which village should he finish?



Give me the change!

Beaver Anthony bought a great European oak wood plank for his lunch. The plank costs 79 beavlars and he only has one 100 beavlars note. The seller has to give him back the change, but only has the following coins denomination (he has an unlimited number of each coin):

- 14 Beavlar
- 9 Beavlar
- 3 Beavlar
- 1 Beavlar

How many coins will Anthony receive back, at least?

