

How can unplugged approach facilitate novice students' understanding of computational thinking? An exploratory study from a Nigerian university

Friday Joseph Agbo^{a,b}, Linda Ojone Okpanachi^c, Patrick Ocheja^d, Solomon Sunday Oyelere^{e,*}, Godwin Sani^f

^a School of Computing and Information Sciences, Willamette University, Salem, USA

^b School of Computing, University of Eastern Finland, Joensuu, Finland

^c Department Computer Science, Federal University Lokoja, Nigeria

^d Graduate School of Informatics, Kyoto University, Kyoto, Japan

^e Department of Computer Science, University of Exeter, Exeter EX4 4PY, UK

^f Department of Computer Science, Federal University Lokoja, Nigeria

ARTICLE INFO

Keywords:

Computational thinking
Problem-solving
Unplugged activities
Programming education
CS students, Nigeria university

ABSTRACT

Computational thinking (CT) has been relevant in teaching STEM courses, and educators are incorporating both plugged and unplugged CT activities in their modules to prepare new entrants for advanced knowledge. However, additional empirical data are needed to understand how an unplugged activities approach can lead to an improved understanding of CT concepts. Therefore, this study investigates the impact of unplugged activities on novice students' CT skills and unravels the characteristics of students' CT competency from performing unplugged activities. The experimental aspect of this study was conducted with a cohort of 210 students participating in an introductory programming course at a University in Nigeria. A mixed-method research approach and a quasi-experimental design were applied in this study to understand how the unplugged approach in teaching CT could help students improve their CT competency. A paired-sample *t*-test of familiarity score for each CT concept showed a significant increase in the score before and after the intervention. Furthermore, Latent Dirichlet Allocation (LDA) was used for topic modeling of the text describing students' solutions. Topic modeling allows for more insights into the steps taken by the students to solve CT problems. Only 27.62 % of students' solution descriptions contained CT concepts, 56.90 % of the students solved the puzzle with an optimal score, and 72.41 % of students who applied CT concepts of problem abstraction and decomposition were able to solve the puzzle correctly. Indeed, CT unplugged activities significantly increased the number of students who gained CT competency. In addition, this study provides recommendation for computer science educators and highlight future direction.

* Corresponding author.

E-mail address: s.oyelere@exeter.ac.uk (S.S. Oyelere).

1. Introduction

Computational thinking (CT) skills have been recognized as a potential tool fostering problem-solving, creative thinking, and analytical thinking (Moreno-León et al., 2018). The development of CT in the higher education institutions context has recently gained traction (Agbo et al., 2019; Lyon & Magana, 2020), which indicates CT relevance beyond K-12 education (Weese, 2016). Indeed, the relevance of CT becomes evident as it relates to building the foundations for computer science (CS) education and facilitating students' understanding of programming concepts (Agbo et al., 2023; García-Peñalvo, 2018). Over the years, studies that examined students' performance in CS1 courses showed that the failure rate of introductory programming had dropped from 33 % in 2007 (Bennedsen & Caspersen, 2007) to 28 % in 2017 (Bennedsen & Caspersen, 2019). This finding may suggest that efforts in fostering CS education yield positive outcomes even though a 5 % progress rate within 12 years may seem marginal. On the other hand, in the case of a developing country – particularly in Nigeria – a study has shown that the failure rate in introductory programming is over 60 % (Sunday et al., 2020), which seems alarming and requires concerted efforts. Some of the reasons causing the difficulty in understanding CS courses have been linked to the traditional teaching approaches (Vihavainen et al., 2014), lack of prior CS experience (Salguero et al., 2021), or apprehension about coding (Alhazbi, 2016).

Most Nigerian CS students do not have computing experience as they are not taught CS courses in secondary schools. Consequently, these students are exposed to programming first at the university level. This situation has caused the majority of the students to become anxious and frightened to the extent that it may affect their understanding and motivation in the course (Agbo et al., 2019). Not only the students that major in computer science degree are expected to enroll for introductory programming courses. At junior levels in the university - mainly year 1 and 2, students whose primary course is Statistics, Computer Science Education, Mathematics, Mathematics Education, and Geography also enroll for introductory programming. Over the years, our experience in teaching introductory programming in a Nigerian university has shown that only a few students grasp the concept of programming. Therefore, we identified a problem that requires an innovative approach to help these students understand the basics of problem-solving through CT.

Because CT involves thought processes to solve problems through concepts that provide foundational knowledge of programming (Wing, 2008; Aho, 2012), it is essential to demonstrate how several CT concepts can support students' understanding in the classroom. CT concepts includes algorithmic thinking, problem abstraction, problem decomposition, pattern recognition, and recursive thinking (Agbo, 2022). Besides, other CT concepts according to literature includes debugging, logical reasoning, and simulation (Yadav et al., 2011; Araujo et al., 2019). One way to teach CT at school is to visualize learning objects using concrete tools. Since CT in Africa is relatively new (Agbo et al., 2021), the authors decided to explore the opportunity to demonstrate it in a CS course through CT unplugged activities. The focus of this study is to explore how to foster novices' understanding of programming concepts using CT unplugged approach in the context of Africa, which was motivated by similar recent studies (Conde et al., 2017; Anthony et al., 2022; Looi et al., 2018). The study examined the effect of this approach on students by assessing their performance in the class and their self-reported responses to the questionnaires provided before and after the course. According to Basso et al. (Basso et al., 2018), CT learning can be assessed by analyzing data collected at the beginning and end of interventions to measure initial and final conditions (milestones).

2. Theoretical background

This section provides theoretical background as the foundation for this study.

2.1. Computer science unplugged

Computer science or "CS unplugged" has existed for a long time. Early researchers have argued that it was initially meant to be an outreach program that allows primary students to gain knowledge of computer science other than programming (Bell & Vahrenhold, 2018). However, unplugged activities have gone beyond an outreach program to become a pedagogical approach with numerous applications and are integrated into the curriculum in different contexts (Salguero et al., 2021). Indeed, Bell and Vahrenhold refer to "unplugged" as a term that is mainly used for CS "teaching activities that do not involve programming" (Bell & Vahrenhold, 2018).

Furthermore, while most unplugged activities may be predominantly in CS courses, other disciplines utilize the same approach in their classes (Carruthers et al., 2009). Notwithstanding, the use of unplugged activities has strived in the CS field, particularly in the K-12 context (Brackmann et al., 2017; Bell & Vahrenhold, 2018). Because of its impact, CS experts have been developing unplugged activities and publishing them in academic forums and even on websites as free learning activities that teachers could use. For example, the CSunplugged website contains relevant resources from contributors advancing CS education. Nowadays, unplugged activities are also explored by CS educators in colleges (Anthony et al., 2022), with promising outcomes in fostering advanced computing.

As good as the CS unplugged approach seems, some researchers have expressed negative views about its impact on students' learning outcomes. For example, Grover and Pea (Grover & Pea, 2013) assert that one should be mindful of the unplugged approach utilized in CS classes. It "may be keeping learners from the crucial computational experiences involved in CT's common practice" (Grover & Pea, 2013). Consequently, this raises a need for some principles that guide the use of unplugged approaches and activities in classrooms. According to Nishida et al. (Nishida et al., 2009), principles that could guide the use of CS unplugged approach include the following, which particularly emphasized that the approach should i) avoid using computers and programming, ii) contain a sense of play or challenge (game) for the student to explore, iii) be highly kinesthetic, iv) be a constructivist approach, v) contains short and simple explanations, and vi) have a sense of story.

Following the six principles of Nishida et al. (Nishida et al., 2009), this study carefully selected the unplugged activities that could fit the novice students in a programming class from the context of Africa where there is no or little computing education at high school.

2.2. Computational thinking approaches in higher education

Various definition of CT, its concepts, and principles have been provided in the literature (Cansu & Cansu, 2019). Currently, there is no consensus definition, and rather than focusing on defining these terms, this study identified five main concepts of CT to develop an intervention to foster students' problem-solving skills. This section briefly presents these concepts. Algorithmic thinking refers to thought processes for creating innovative steps to solve a problem through understanding, executing, and evaluating (Oyelere et al., 2023); problem abstraction is the process of making problem more understandable by reducing the unnecessary details and focusing on essential aspects, which will lead to more straightforward solutions (Cansu & Cansu, 2019); problem decomposition refers to breaking a problem into smaller sets of problems for more understandable constituents (Fried et al., 2018); pattern recognition deals with identifying the arrangements and relationships within and between problems; recursive thinking relates to finding ways of solving a smaller portion of a problem and iteratively applying the same solution to the remaining part of the problem. These strategies have been used teach computational thinking and problem-solving skills to programming novices at higher education institutions (Kuhail et al., 2021).

Various studies have investigated CT approaches in the higher education context (Agbo et al., 2023; Lyon & Magana, 2020; de Jong & Jeuring, 2020). These approaches include robotics-based approach, block-based programming, and non-programming unplugged activities. For example, García-Peñalvo et al. (García-Peñalvo et al., 2021) emphasizes that educational robotics is a valuable approach to teaching CT skills regardless of age, grade, or previous knowledge. Similarly, (Corral et al., 2021) appraised that block-based programming inspires students to gain CT knowledge that can be transferred into understanding computing topics such as software engineering.

In a paper by Eben et al. (Witherspoon et al., 2017), an online robotics curriculum involving a sequence of lessons in robotics programming using ROBOTC Graphical language was adopted to teach CT constructs. The curriculum consisted of four main teaching units like introduction to programming, basic movement sensors, and program flow. Each curriculum unit required the students to pass through a sequence of virtual robotics problem-solving tasks. To ensure that students were fully engaged, they were allowed to use a robot's virtual world, a simulated 3D game-like virtual environment. Students could test programmed solutions with simulated robots in a three-dimensional virtual platform.

Similarly, Caballero and García-Valcárcel (Caballero González & García-Valcárcel Muñoz-Repiso, 2018) studied how to foster computational skills using a pilot experience play and program with Bee-Bot. In their approach, activities were developed using a framework that allowed students to solve challenges in programming using the Bee-Bot floor robot. Furthermore, Jormanainen and Tukiainen (Jormanainen & Tukiainen, 2020) focused on training fundamental CT skills via the use of a programmable and Interactive Teddy Bear toy where a hardware set is embedded in the Teddy Bear. The students were asked to program the Teddy Bear to complete specific tasks. Jormanainen and Tukiainen (Jormanainen & Tukiainen, 2020) showed that students gained significant CT skills from this approach.

Even though educational robots in CT education showed significant results, a paper by Fronza et al. (Fronza et al., 2019) argued a limitation in the complexity and cost of implementing such robots. Instead, they proffered a different strategy to create an affordable alternative via the use of hardware/software. To physically demonstrate their technique, Fronza et al. (Fronza et al., 2019) used a coding Bootcamp that required students to design and implement a simple algorithm using block-based programming tools, not only that but to also create a simple hardware prototype using electronic components. Thus, Scratch block-based programming language was used, and it was run on Raspberry Pi computer while trying to implement the traffic light algorithm. According to Fronza et al. (Fronza et al., 2019), this algorithm was chosen because it was sequential, linear, and repeatable. Their study showed that students were able to complete phases of a software development life cycle and a practical example of the application of CT. Furthermore, a previous study by LeChen et al. (Zhang et al., 2020) validates the approach used by Fronza et al. (2019). However, there are slight differences as the research by Zhang et al. (2020) presented a choice amongst block-based programming tools, analog tools, and text-based.

Out of the three approaches for introducing CT to students discussed above, studies have shown that the robotics and block-based approaches have been the mainstream. They have been used in several contexts with varying learning outcomes (Kert et al., 2020). However, the third approach, that is, the CT unplugged activities, have shown to be more prominent among CS educators than the other approaches (Anthony et al., 2022; Nishida et al., 2009). Another study by Conde et al. (Conde et al., 2017) showed significant differences from an experiment on students divided into two groups. The first group was taught the concept of CT with the robotics approach, while the second group was taught using unplugged activities. The group taught using unplugged activities had better results than those taught using the robotics approach. Unplugged activities have been deployed in many settings incredibly informal to introduce the concept of CT in a very competent and engaging manner. Most activities are kinesthetic, engaging, and typically accessible for students who lack a computer and may not have typing skills (Salguero et al., 2021).

Furthermore, Rodriguez et al. (Rodriguez et al., 2017) developed a CS unplugged curriculum that engages students via assessment mapped from questions relating to a CT skills and bloom taxonomy project. Their results show that about 50 % of students who had not encountered such activities before demonstrated remarkable performance, as revealed by their posttest result. Also, approximately 80 % of students achieved higher proficiency. Curzon (Curzon, 2013) developed 'Computer Science for Fun' - cs4fn, a public project engagement aimed at embedding cs4fn unplugged activities in stories and illustrative motion pictures. cs4fn "unplugged activities" resources are widely used in the UK and have positive feedback from teachers and students.

Similarly, Ouyang et al. (Tonbuloglu & Tonbuloglu, 2019) studied how to integrate CT in formal classroom settings by helping teachers connect these concepts with skills necessary for student success using certain unplugged activities. This led to the development and implementation of CT integrated lessons which provided CT experiences for students. Since the teachers were involved, it made it easier for them to find avenues to deepen their students' understanding of CT. Hence, the study presented in Tonbuloglu and Tonbuloglu, (2019) showed a significant increase in scores relating to CT.

2.3. NLP in education research

Natural Language Processing (NLP), an interdisciplinary field combining computer science, artificial intelligence, and linguistics, enables machines to understand and derive meaning from human language (Cambria & White, 2014). One of the key areas where NLP has been applied is educational research, in order to analyze the large amounts of textual data generated in educational settings, ranging from student assignments and essays to transcripts of classroom discussions (Yang et al., 2022). A critical method employed within NLP is Latent Dirichlet Allocation (LDA), a topic modeling technique that identifies abstract topics across multiple documents (Blei et al., 2003). In essence, LDA assumes that each document (e.g., a student's response to a problem) is a mixture of several topics, and it calculates the likelihood of a particular word being associated with a specific topic.

LDA has proven beneficial in various educational research contexts, such as identifying students' learning strategies from their written reflections (Chen et al., 2016) and understanding online discussion dynamics in massive open online courses (MOOCs) (Wen et al., 2014). By analyzing students' written responses, LDA helps researchers unravel underlying patterns and themes that would be challenging to identify manually due to the volume of data. In qualitative research, LDA can be used as a form of text mining to analyze and interpret large volumes of textual data (Maier et al., 2021). For instance, students' written responses to a particular task can be seen as a corpus of text to be analyzed. LDA can then be applied to this corpus to identify the main topics present across the responses. These topics can provide a high-level view of the main themes emerging from the students' understanding, enabling the qualitative researcher to identify patterns or commonalities.

To apply LDA, researchers first need to preprocess the text, including steps like tokenization (breaking up the text into individual words), removing stop words (common words like 'the', 'and', 'is', that usually do not add much meaning), and stemming (reducing words to their root form). After preprocessing, the text is converted into a document-term matrix, which forms the input to the LDA algorithm. The output of LDA is a set of topics, each represented by a distribution over words. While LDA can be a powerful tool for educational researchers, it is essential to interpret the topics with care and relate them back to the research questions at hand (Roberts et al., 2019). It's also crucial to validate the results through triangulation with other data sources or methods, to ensure the robustness of the findings (Oztok et al., 2015).

In the context of this study, LDA serves as a powerful tool for delving deeper into the process through which novice students engage with CT concepts. As we seek to understand how unplugged activities facilitate students' comprehension of CT, the textual data generated from students' descriptions of their solution strategies offers rich insights. LDA allows us to analyze this corpus of textual data, identifying latent themes and patterns that speak to the students' grasp of CT concepts and their application in problem-solving scenarios (Blei et al., 2003). The use of this advanced NLP technique helps us reveal the cognitive processes involved in students' engagement with CT, and the extent to which unplugged activities aid their understanding. The insights gained from the LDA analysis significantly contribute to our understanding of effective teaching strategies for CT, providing valuable recommendations for computer science educators aiming to improve students' CT competency. Thus, our study serves as an example of how NLP techniques, specifically LDA, can provide nuanced understanding of learning processes in STEM education, extending beyond traditional quantitative measures to offer a richer picture of student learning dynamics.

Following this background, this study formulated two research questions to guide its exploratory nature. To address the research questions, this study present evidence on how CT was demonstrated in the classroom and analyzed the data collected from the students through a quasi-experimental process implemented during the class sessions, their scores from the CT unplugged puzzle, and the explanation of their solution were also presented.

RQ1. Do unplugged activities significantly improve novice students' computational thinking skills after the intervention?

RQ2. What characteristics of students' computational thinking competency can be unraveled from performing unplugged activities?

3. Study design

3.1. Description of context

A short course was designed to introduce students to a computer science introductory programming course tagged 'Computer Programming I' (CSC 211) taught in a federal university located in north-central Nigeria. The course CSC 211 originally introduces students to problem-solving techniques using a C++ programming language. The course attracts students from different backgrounds where the majority have no programming experience. The goal of the course is to allow students to learn specific C++ constructs and concepts. After completing the course, students were expected to be familiar with the basics of C++ programming language and use concepts such as variables, data types, arrays, pointers, functions, and classes to solve problems. This introductory programming course is meant for students in year two from the Computer Science, Mathematics, Computer Science Education, Mathematics Education, Statistics, Geography, and Biotechnology departments.

In fall semester 2021, 370 students enrolled in the CSC 211 course, with most students having little to no programming experience. The lack of programming experience among students in this context is mainly because students are not introduced to computing at secondary school and programming is not part of the subjects being taught in Nigeria public secondary school yet. Consequently, majority of the students are confronted with the challenge of understanding programming concept necessary for solving problems at the university level. A few of the students who possess limited programming knowledge either attended private secondary schools where computing was taught or engaged in self-learning. This situation creates a lot of anxiety among students and majority fail the course over the years.

To enhance students' understanding of the concept of the C++ programming language, the course lecturer – also an author in this study - introduced the concept of CT unplugged activities held at the beginning of the course. The reason for introducing CT unplugged activities at the beginning of the semester is to explore how students can gain the fundamentals of problem-solving necessary to comprehend the course objective. In addition, we envisaged that students could understand the basics of problem-solving through simulations and visualization of unplugged activities and before learning how to construct C++ programs to solve problems. Previously, the lecturer has observed that the majority of the students lack the basics of CT and problem-solving skills and hence, struggle to comprehend the programming concepts covered within the course.

3.2. Participants and ethics

Although 370 students enrolled for the course, only 210 students actively participated in this study by completing the tasks and responding to the survey. The remaining students either did not show up or could not complete the activities. Hence, they were not included in this study. Since the study emerged from a short lecture designed to introduce students to problem-solving, it was challenging to get all students to actively participate at the same level of engagement with class activities. However, the majority were seen to be engaged based on our observations. Table 1 presents the descriptive statistics of the participants by gender and course of study.

This study conforms to the ethical principles and guidelines of the Finnish national board on research integrity regarding responsible conduct of research (RCR). Compliance with these guidelines is required by the first author's affiliation when this study was conducted, however the first author's affiliation has changed afterwards. All the students in this study voluntarily participated without any form of persuasion. Although, the general condition where students would attain seventy percent of class attendance before being allowed to sit for the course exams was upheld. This condition suggests that students may naturally participate to satisfy the requirement for taking the exams. Therefore, class attendance was typically submitted to the lecturer after each session, which is the usual practice.

3.3. The design of CT unplugged activities and data collection

According to experts, CT is a widely utilized approach in demonstrating how to facilitate students' understanding of programming concepts (Rojas-López & García-Peñalvo, 2019; Figueiredo & García-Peñalvo, 2021), however, this approach has not been explored extensively in the context of Nigeria. In this study, authors designed lessons as intervention to foster CT skills. The intervention was designed and implemented within two weeks as shown in Fig. 1. Within these weeks, several classes with lessons and activities were implemented. A similar approach was implemented by Yadav et al. (Yadav et al., 2014) where they developed a one-week module course to introduce preservice teachers to fundamentals of CT. This section provides an overview of the lesson, unplugged activities, and CT problems developed and used in this study.

CT lesson: First, the authors designed an introductory lesson consisting of the definition of CT and its concepts – algorithmic thinking, problem abstraction, problem decomposition, pattern recognition, and recursive thinking – which was implemented in the week 1. This introductory lesson is necessary to provide students with foundational knowledge of CT since they were presumably new to these concepts in general. Although other CT concepts such as debugging and simulation were showcased in the data collection items adapted from previous study, the introductory lesson designed and administered in week 1 only contain the five concepts presented earlier. Consequently, our analysis was limited to these five CT concepts.

3.3.1. Examples of CT activities

Additionally, the authors also introduced examples as CT mini games (Tower of Hanoi and River Crossing puzzles) to demonstrate how tangible artefacts can broaden the understanding of problem-solving and computing. For example, the Tower of Hanoi (ToH) is one of the puzzles acknowledged to provide concrete ways to teach recursive thinking (Butgereit, 2016; Agbo, 2022). In this study, the

Table 1
Descriptive statistics of the study participants.

		Frequency	Percentage (%)
Students	Male	139	66.2
	Female	71	33.8
Course of study	Computer Science	96	45.7
	Mathematics	26	12.4
	Geography	58	27.6
	Others	30	14.3

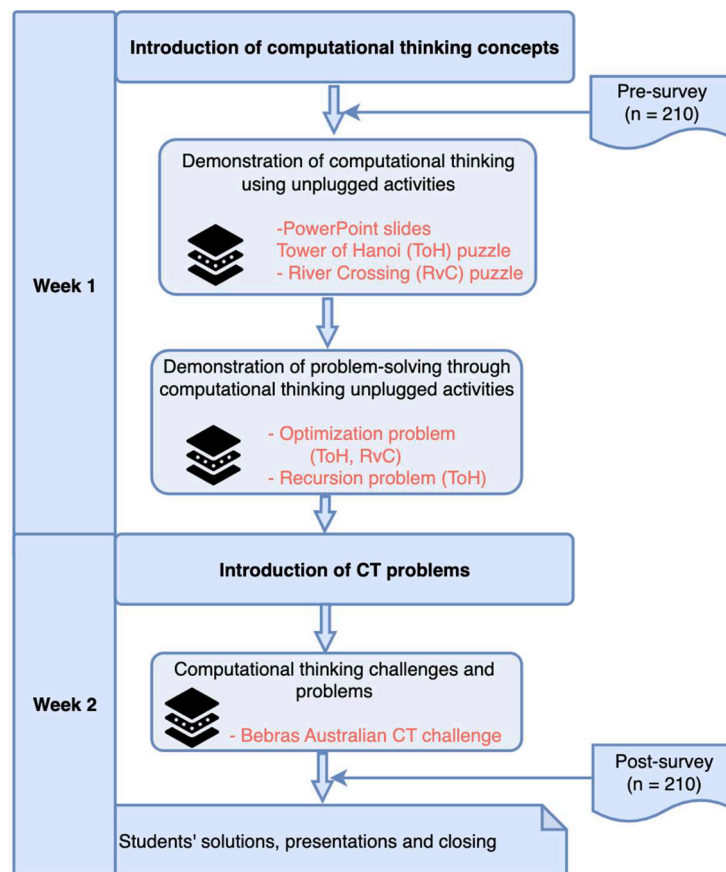


Fig. 1. Plan and implementation of computational thinking lessons and activities.

students experienced a table-top ToH where they learnt techniques behind unravelling optimal arrangement of n -discs from a ToH peg 1 to peg 3 ($n = 1, 2, 3, \dots$).

3.3.2. CT problem

Furthermore, the second week was planned to expose the students to several CT unplugged problems adapted from widely recognized sources such as Bebras Australian CT challenge 2021. The goal of this intervention is to allow students to demonstrate skills in several CT concepts such as problem decomposition, abstraction, algorithmic thinking, pattern recognition, recursive thinking, debugging, and simulation. Thus, this study followed the Nishida et al. (2009) principles to select the CT problems presented in Appendix 1. The activities were adapted so that students could learn through demonstration and simulation to understand some of the programming concepts by starting from simple Bebras problems (for ages 5–9) to complex problems (for ages 11–12). The rationale for adapting this approach is to allow novices to gain problem-solving skills without necessarily overloading the contents that may make it difficult to comprehend since they do not have prior programming experience. Moreover, a previous study argued that unplugged activities must be carefully selected to deliver the learning objective (Grover & Pea, 2013).

3.3.3. How students engaged with CT problems

The students were given four problems to solve on their own. The CT problems selected for this study consists of puzzles such as Tower of blocks, Matching socks, Marbles and boxes, and Electric cars. Each of these problems has a storyline and narrations with accompanying challenge that students must solve. In addition, the learning objectives and connections to the CT concepts highlighted earlier are shown in Appendix 1. Moreover, the students were asked to solve the Tower of blocks first and any other problems they found interesting. They could build the towers by drawing them on the paper, and perform the removal (Pop) or placing (Push) operation on the blocks. Besides, the students were encouraged to write down the steps they were taking to arrive at the answer and count how many Pop and Push operations - in total - they did to solve the problem. Fig. 1 presents an overview of the intervention showing the timeframe, planned CT lessons, unplugged activities, and CT problems utilized in the study.

Regarding the data collection, the study adapted the instruments on fundamental knowledge of CT used in previous study and also developed contextual quizzes related to the puzzles used in the study. In other words, *computational thinking quiz* adapted from Yadav et al. (Yadav et al., 2011) and authors' self-developed questions were used to elicit students' knowledge of CT, familiarity with CT

concepts, their experience in applying CT concepts to solve the problems, and their confidence with CT after the intervention. The pre-post survey consists of multi-choice, Likert scale of 1 to 4 (from 1-> not familiar/confident to 4-> very familiar/confident), and open-ended questions as shown in [Appendix 2](#). Before the pre-post data collection, the authors explained to students how the terms ‘familiarity’ and ‘confidence’ were defined in the study. For example, the term ‘familiar’ was used during the pre-survey since the students were not taught CT prior to the study. On the other hand, post-survey replaced the term ‘familiar’ with ‘confident’ since they already underwent the intervention. Therefore, both familiarity and confidence with CT concepts measures students’ level of understanding and use of the CT concepts introduced in the study.

Consequently, the same survey with slight modifications were implemented during the pre and post surveys to understand students’ experience and whether the intervention has impacted their knowledge, thinking skills, and self-confidence of CT concepts. Aside from the survey, other qualitative data such as hand written solutions were collected from the students. Thus, a mixed-method approach following a quasi-experimental design was applied in this study to obtain data from participants in a pre and post intervention surveys. Hence, a single treatment cohort was implemented where the same students were recruited for both pre and post surveys. Regarding the data analysis, both quantitative and qualitative techniques were applied to address the questions raised in this study.

3.4. LDA analysis process

In this study, we employ LDA technique to analyse students’ solutions and detect the presence of CT concepts. [Fig. 2](#) shows an overview of this process. First, we ask students to describe their solution to the given puzzle and use the text data for qualitative analysis. In the data pre-processing phase, we clean the data to ensure proper tokenization, stop words removal and lemmatization using WordNet ([Bird et al., 2009](#)). We use the processed data as input to the LDA model to extract topics from students’ responses. To ensure that there is a good topic coherence, we use the Coherence model by genism ([Srinivasa-Desikan, 2018](#)) and recursively generate different models by varying the number of topics in starting from 2 to 50 topics in steps of 6 at each iteration. A plot of coherence scores from the different runs can be used to determine the best number of topics (model). In the next phase, we use the selected model to determine which of the topic(s) best fit each CT concept using a match threshold of ≥ 0.4 . Similarly, we use the model to also determine which topic(s) a student’s solution falls under. Finally, we check for intersections in topics found for both the CT concepts and the students’ solutions. An intersection between a CT concept and student’s solution is interpreted as the presence or use of the matching CT concept by the student to solve the puzzle.

4. Result

This study seeks to address two main research questions. Therefore, the findings are presented in this section under the RQs.

4.1. Impact of unplugged activities on novice students’ computational thinking skills

In this study, 210 students participated, consisting of 139 males (66.2 %) and 71 females (33.8 %). The characteristics of the students that took the course show that the majority were not familiar with the term ‘computational thinking’ and the self-rating of their experience of using CT concepts were low, as revealed in the result in [Table 3](#) that showcased their mean score before class intervention.

To better understand whether the CT unplugged approach - an intervention to facilitate students’ problem-solving skills in an introductory programming class - had any significant impact, we conducted a paired-sample *t*-test of students’ familiarity-confidence scores of the CT concepts before and after the educational intervention. The analysis of the paired-sample *t*-test for each of the CT concepts considered in this study revealed useful results shown in [Table 3](#). For example, students’ algorithmic thinking showed a significant increase in the score before intervention ($M = 2.26$, $SD = 0.71$) to after intervention ($M = 3.88$, $SD = 0.75$), $t(209) = -22.49$, $p < 0.001$ (two-tailed). The mean increase in the score was -1.62 with a 95 % confidence interval ranging from -1.761 to -1.477 .

Furthermore, students’ problem abstraction showed a significant increase in the score before the intervention ($M = 2.14$, $SD = 0.68$) to after intervention ($M = 3.74$, $SD = 0.63$), $t(209) = -25.62$, $p < 0.001$ (two-tailed). Similarly, results were obtained for the paired-sample *t*-test for problem decomposition, pattern recognition, and recursive thinking, which all showed a significant increase in

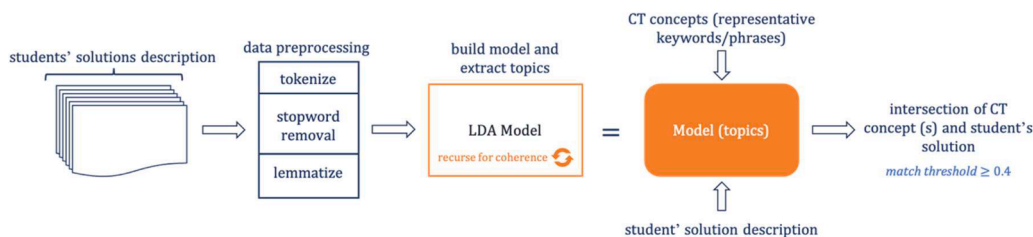


Fig. 2. Process of qualitative analysis using LDA.

the score before and after intervention. For the problem decomposition, there is a significant increase in students' score before ($M = 2.10$, $SD = 0.67$) to after ($M = 3.90$, $SD = 0.61$), $t(209) = -28.26$, $p < 0.001$ (two-tailed). For pattern recognition, there is a significant increase in students' score before ($M = 2.15$, $SD = 0.69$) to after ($M = 3.87$, $SD = 0.64$), $t(209) = -25.52$, $p < 0.001$ (two-tailed). For recursive thinking, there is also a significant increase in students' score before intervention ($M = 2.15$, $SD = 0.66$) to after intervention ($M = 3.79$, $SD = 0.67$), $t(209) = -25.83$, $p < 0.001$ (two-tailed).

In addition, effect sizes were computed for each CT concept to ascertain the extent of statistical significance. Concerning CT's algorithmic thinking scores, the differences in favor of the after-class condition can be considered large-sized, according to Cohen (Cohen, 1988), as $d = 1.04$. Besides, a similar trend ensued in the other CT concepts, problem abstraction ($d = 0.91$), problem decomposition ($d = 0.93$), pattern recognition ($d = 0.97$), and recursive thinking ($d = 0.92$).

In the second week of the intervention, the authors introduced CT unplugged problems described in Appendix 1. The students individually solved the problems and recorded the answers in an online form. For example, Table 2 shows raw scores from Tower of block puzzle. Afterwards, the authors analyzed students' scores from solving the puzzle in order to gain insight into how their solution corresponds to the CT concepts demonstrated in the classroom. We analyzed students' scores for the Tower of blocks puzzle because most of the students ($n = 198$) completed the task compared to other puzzles. Moreover, the learning objectives consist of the five CT concepts we intend to investigate in this study. For example, the Tower of blocks puzzle requires that students should find the 'minimum number of moves that make all towers the same height. This is typically an optimization problem where there could be more than one solution, but only a single key is most accurate in cost and efficiency. The Tower of blocks can be solved with 8 moves in the activity we presented to obtain an optimal solution. However, 9 moves are also a possibility when the towers are made of the same height of three blocks, but 9 moves are, unfortunately, not the optimal solution.

As shown in the pie chart in Fig. 3, students' confidence in CT skills which elicit their responses on whether they solved the puzzle or not, indicates that the majority were confident to have solved the puzzle correctly. In other words, 67 % claimed to solve the puzzle while 33 % failed to solve the puzzle. Since this finding emerged from their self-assessment report, we further analyzed their scores from solving the Tower of block puzzle and found that 47 % obtained an excellent score, 8 % of the students gained a satisfactory score. In comparison, 45 % performed poorly, as presented in the bar chart in Fig. 3. This finding indicates a disparity between students' self-assessments and their actual performance.

4.2. Characteristics of students' computational thinking competency from unplugged activities

Having analyzed students' perceived confidence in CT and their actual performance from one of the unplugged activities, we went further to investigate their explanation to the solution to gain insight into whether students have used the CT concepts, how they have demonstrated these concepts in their solution, and other characteristics that showcased their CT competency. Therefore, we addressed the RQ2 carrying out text mining on the students' description of their solution to the Tower of blocks puzzle.

First, we performed topic modeling (Nkhoma et al., 2020) with all the text describing students' solutions using LDA (Hoffman et al., 2010). LDA is a type of probabilistic topic model which identifies potential topics based on the frequency and co-occurrence of words in documents. In essence, it assumes that each document (in this case, each student's solution description) is a mixture of a small number of topics, and each word's presence is attributable to one of the document's topics. We used this approach to examine whether students' solution descriptions contained one or more CT concepts. This method allows us to classify student responses into coherent topics that reflect different CT concepts. To obtain the optimal number of topics to classify the solution description data, we calculated the topic coherence score (Röder et al., 2015) for various models, varying the number of topics from 2 to 50 in steps of 6. From a plot of topic coherence score of each model against the number of topics, we obtained an optimal number of topics 8, which had the best coherence score just before the curve dips. We then created keywords that represented each CT concept, as shown in Table 4.

The LDA model was then used to determine if a student's solution description contains one or more CT concepts. To improve the accuracy of the analysis, we only considered a student's report of their solution to match a CT concept when the topic coherence score is ≥ 0.4 . Recall that in the study design section, we describe the qualitative analysis used to detect the presence or use of CT concepts in students' solutions as shown in Fig. 2. Table 5 presents matrix plot of various CT concepts and how many students' solutions were found in the same topic as that of the CT concepts. For example, 21 students' solutions were in the same topic as CT concepts of algorithm design and recursion. Similarly, 29 students' solutions consist of the same topics as CT concepts of problem abstraction and problem decomposition. Further analysis of the students presented in the result shown in Table 5 indicates that only 58 out of 210 (27.62 %) students' descriptions of their solution contained CT concepts. Out of the 58 students whose solution consisted of at least one CT concept, 56.90 % of them were able to solve the puzzle with optimal moves which was higher than the number of students who got the solution correctly without CT concepts (40.13 %) mentioned in their solution. Also, 72.41 % of students who applied CT concepts of problem abstraction and decomposition were able to solve the puzzle correctly with the optimal number of moves.

Table 2
Sample of students' raw scores from solving Tower of block puzzle ($n = 198$).

Prompt	How many moves did you used to solve the puzzle?							
Number of moves (options)	3	4	5	6	7	8	9	10
Frequency of scores.	9	16	12	17	12	96	9	27

Note: excellent score = 8 moves; satisfactory score = 9 moves; poor score = others.

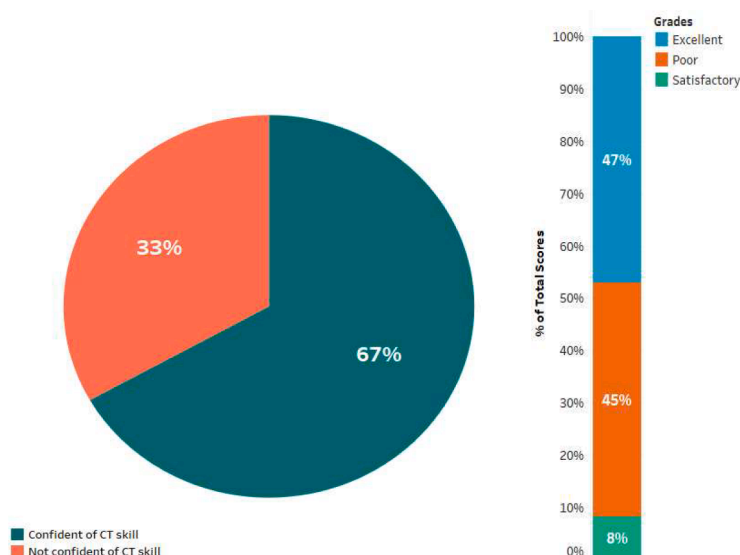


Fig. 3. Students' confidence in CT knowledge versus performance score.

Indeed, this finding correlates with the required solution for the Tower of blocks puzzle. Thus, we summarize our findings as follow:

1. Students who reported to have understood at least one CT component (as expressed in their description of how the problem was solved) are more likely to solve the puzzle correctly than students who cannot.
2. Students who claimed to understand the CT components of algorithm design and recursion were more likely to solve the puzzle correctly than students who solved the puzzle correctly but did not express CT components in their description of the solution.
3. Students who mentioned that they understood CT components of problem abstraction and problem decomposition were more likely to solve the puzzle correctly.

Furthermore, we obtain the overall reaction of students to the CT unplugged activities by asking them to provide a short reflection regarding their experience. Part of the reflection feedback we asked that students should describe were their experiences with the CT unplugged approach. We analyzed their written responses thematically by first aggregating the keywords through word cloud plot using Atlas.ti. Several keywords emerged including “interesting, engaging, challenging, enlightening, educative, logical, critical, and insightful.

Indeed, students' feedback generally shows positive experience. For example, the students reported that the intervention and activities were interesting to them, thus some asserted: “Interesting....It involves thinking and this made it interesting”. Another comment reads: “It was interesting to face something easy but challenging”. In addition, the students found the intervention educative with relevant challenges that engages them to learn. Some excerpts from students' feedback reads: “It opened my mind to a lot of possibilities. after solving the puzzle and checked the answer, I was happy to know that I was correct”: “While solving the puzzle, I was able to relate it to some other things. It was more like I was modelling and working on an algorithm”. Although these feedbacks were self-reported, it provides a general overview of students' experience from the intervention.

5. Discussion

CT competency has been the focus of recent research (Grover & Pea, 2018; Lai, 2021). Understanding CT competency and how students gain those competencies has the potential to support how teachers adapt and apply teaching strategies for better uptake of learning in diverse contexts. For example, teachers are usually confronted with concerns about what kind of activities would lead to the development of CT competencies. The application of the unplugged approach in teaching CT concepts is not new (Bell & Vahrenhold, 2018). Several studies have recommended using unplugged activities to measure CT competency (Bell & Vahrenhold, 2018). However, understanding what impact those unplugged activities have on CT competency would provide adequate complementary information to teachers (Bell & Vahrenhold, 2018).

Moreover, concerns have been raised by an earlier study regarding how unplugged activities support students or otherwise (Grover & Pea, 2013). To contribute to scholarly findings on the use of unplugged activities in formal classrooms, this study developed a short course that explored CT unplugged activities to foster students' problem-solving skills in an introductory programming course. This study aimed at understanding how the use of unplugged activities facilitates novice computer science students' CT skills. In addition, we studied the characteristics of students' perceived CT confidence to gain insights into their actual performance and CT competence demonstrated through the unplugged CT activities.

Table 3
Paired-sample *t*-test of students' familiarity-confidence score of the computational thinking concepts before and after intervention.

		Paired differences										
				Mean	Std. Deviation	Std. Error Mean	95 % Confidence Interval of the Difference		t	df	Sig. (2-tailed)	Cohen's d
		Mean	Std Dev				Lower	Upper				
Algorithmic thinking	Before class	2.26	0.71	−1.62	1.04	0.072	−1.761	−1.477	−22.49	209	0.000*	1.04
	After class	3.88	0.75									
Problem abstraction	Before class	2.14	0.68	−1.61	0.91	0.063	−1.728	−1.481	−25.62	209	0.000*	0.91
	After class	3.74	0.63									
Problem Decomposition	Before class	2.10	0.67	−1.81	0.93	0.064	−1.931	−1.679	−28.26	209	0.000*	0.93
	After class	3.90	0.61									
Pattern recognition	Before class	2.15	0.69	−1.72	0.97	0.067	−1.854	−1.588	−25.52	209	0.000*	0.97
	After class	3.87	0.64									
Recursive thinking	Before class	2.15	0.66	−1.64	0.92	0.063	−1.763	−1.513	−25.83	209	0.000*	0.92
	After class	3.79	0.67									

**p* < 0.001.

Table 4

CT concepts and associated keywords and phrases used for topic mining by LDA.

CT concepts	Representative keywords and phrases
Algorithm design	<i>think, reason, compare, arrange, select, first step, next step, from top, count, arrangement, ordering</i>
Problem abstraction	<i>remove from, takeoff, put away, drop out, exclude, removed, reduced</i>
Problem decomposition	<i>break down, divide, divided, add to, remove from, scatter, broke, take from, taking from, collect, drop off</i>
Pattern recognition	<i>identify, recognize, observe, show, similar, same</i>
Recursion	<i>repeat, iterate, redo, do again, return, place it again, count from</i>

Table 5

Result of how students' description of their solution fit into various CT concepts.

CT concepts	Algorithm design	Problem abstraction	Problem decomposition	Pattern recognition	Recursion
Algorithm design	21				
Problem abstraction	0	29			
Problem decomposition	0	29	29		
Pattern recognition	0	0	0	8	
Recursion	21	0	0	0	21

Regarding the first research question, we compared the perceived knowledge of CT before and after the intervention to understand whether the CT unplugged approach had any significant impact on problem-solving and programming skills. The results presented in Table 3 indicated a significant increase in all the CT competencies measured. The unplugged approach increased students' algorithmic thinking, problem abstraction, problem decomposition, pattern recognition, and recursion. This result is similar to study (Tonbuloglu & Tonbuloglu, 2019; Montes-León et al., 2020; Sun et al., 2021; del Olmo-Muñoz et al., 2020), in which the unplugged approach deepened students' understanding of CT concepts and led to an improved CT competency. Unplugged activities positively affected the improvement of students' CT skills such as creativity, algorithmic thinking, collaboration, problem-solving, and thinking skills in middle high school (Gardeli & Vosinakis, 2017). There is a large effect size in the before and after intervention results, which means that students' earlier knowledge of CT concepts has significantly improved due to the unplugged activities intervention (Tonbuloglu & Tonbuloglu, 2019; Merino-Armero et al., 2022).

Furthermore, this study revealed that students gained higher confidence in applying CT skills to solving problems. The majority of the students (67 %) affirmed that their CT confidence improved after the intervention. This finding aligns with a previous study investigating how to introduce CT to teachers using unplugged activities where the approach was found to inspire and build confidence among the subjects (Curzon et al., 2014). Besides, this study's outcome emphasizes the potential of using unplugged activities as a pedagogical approach (Bell & Vahrenhold, 2018) and improves CS students' learning outcomes (Curzon, 2013).

In answering research question 2, we applied a data mining approach. A text mining with topic modeling using LDA to detect what CT concepts students applied when solving the unplugged puzzle revealed relevant insight on students' performance after the intervention. A recent study has proposed data mining techniques as a practical approach to investigating properties of CT competency from the extract of learners' performance patterns (Lai, 2020). Furthermore, the data mining technique using the LDA algorithm has been used in another study to determine the reasons for students' academic failure (Nkhoma et al., 2020). Our analysis reveals that more students can optimally solve the puzzle when they apply at least one CT concept in their solution. More so, more students solved the puzzle correctly when the CT concepts of problem abstraction and decomposition were applied.

This finding further reinforces the importance of CT concepts in understanding and solving computational problems. Besides, the result also presented a unique approach to examining the impact of the CT unplugged approach in any context. As revealed in previous studies (Peteranetz et al., 2017; Soh et al., 2015), this present study shows that unplugged activities are capable of improving computer science education in the context of developing nations' higher education institutions where students lack programming background. In addition, we argue that the assessment of users-perceived computational thinking confidence score is insufficient to understand the impact of an intervention (Basso et al., 2018). For example, this study discovered a disparity between students' claims regarding their CT confidence compared to their actual performance. We assumed that one reason for this disparity could be because this is the students' first attempt to learn about CT, which could mean that they gain insubstantial understanding of the concepts. Therefore, they may have limited experience on how to self-report their solution to the CT problems that reflects the use of these concepts. Besides, the students could potentially have bias in their responses if they thought that this could impact their grades in the CS course. While this study provides useful outcomes on the integration of CT unplugged activities in programming classes from the context of the developing country, it is important to conduct several rounds of similar interventions and apply other method of assessments to ascertain its efficacy. In other words, further analysis of how students performed during the unplugged activities will provide a deeper understanding of the intervention's impact. Also, we recommend that teachers and researchers use this text mining technique of topic modeling by LDA to evaluate students' understanding of CT concepts. The results of such evaluation can inform the teacher on what areas students lack knowledge or require additional clarification.

The LDA used in our study was critical in classifying student responses into coherent topics, shedding light on their CT competencies. Our analysis indicated the following:

- a) **Relevance of CT Concepts to the Puzzle:** An optimal number of 8 topics were identified, which encompass various CT concepts like algorithm design, problem abstraction, and recursion. This not only showcases the breadth of CT concepts utilized by students but also the depth and richness of their problem-solving strategies. The alignment of CT concepts with the Tower of Blocks puzzle suggests that the puzzle inherently demands the application of these CT concepts, emphasizing its appropriateness as a tool for gauging CT abilities.
- b) **Efficacy of Expressing CT in Solutions:** A significant number of students (27.62 %) employed at least one CT concept in their solution descriptions. Notably, there was a clear correlation between recognizing and employing CT concepts and success rates in solving the puzzle. Students who expressed understanding and applied CT concepts were more successful, especially those using algorithm design, recursion, problem abstraction, and decomposition.
- c) **Correlation with Performance:** The students who could articulate their CT strategies (like problem abstraction or recursion) had higher chances of solving the puzzle optimally. This suggests that the conscious recognition and articulation of CT processes might aid in refining problem-solving strategies.

Moreover, it is important to discuss other relevant approaches for measuring learners' CT competency as shown in literature. For example, Yadav et al. (Yadav et al., 2022) recently developed survey that measured preservice teachers' CT competency which is analyzed using their ranked responses. Their aim is to find patterns in the self-reported survey by the participants. Similarly, research has shown other frameworks for measuring and assessing CT competency which include the Computer Science Teachers Association's (CSTA) curriculum, Barefoot's computational thinking model, and the Dr. Scratch analytical tool (Romero et al., 2017). The strengths and limitations of these approaches have been discussed in literature, which is not the focus of this study. However, this study explored the use of survey and learning activity recorded from the intervention to assess learners' CT competency in the context of a developing nation's higher education.

5.1. Study limitation

This study is not without some limitations. Mainly, the study is limited in the experimental design, which is evident in its lack of internal validity. The survey adapted in this study allowed students to provide their perceived understanding and confidence in using CT concepts to unravel problems, which could cause bias in responses regarding the intervention. In addition, this study's intervention was conducted within two-weeks, which may be insufficient to measure students' learning outcomes. Compared to several iterations, this study conducted one circle of intervention and reported students' learning outcomes, which could limit the findings' generalizability. Thus, it is difficult to crystalize students learning process in depth by using unplugged activities to facilitate CT and problem-solving skills. Notwithstanding, the analysis of students' explanation of their solution to the unplugged problems revealed preliminary insights on their learning process that could guide future study.

6. Conclusion

Through this study, we have shown how CT unplugged activities can support novice students in computer science. The study demonstrated that understanding CT concepts are a valuable step towards facilitating problem-solving skills, which can help students' understanding of programming concepts. Evidence from this study supports the argument that unplugged CT is a potential pedagogy for CS education not only in K-12 settings but also in colleges where students without prior experience of computing are struggling to understand programming. Beyond the CT competency, students gained confidence in solving unplugged puzzles, motivating their interest in programming courses. Since this was the first time this CT unplugged activity was introduced into a college introductory programming course, our follow-up study will investigate students' performance in the exams to see whether there is any impact on the exam score of the current academic year compared to the previous years. In other words, our future study will examine actual students' performance in programming using their exam scores to ascertain if the intervention impacts their overall performance in the course.

Beyond providing evidence on how CT unplugged activities are a useful intervention to facilitate novices problem-solving skills, this study advances the understanding of learners' perceived learning achievement in relation to actual performance that test their knowledge by applying a unique NLP method. One of the implications of this study is to demonstrate CT unplugged activities to students with similar contextual background as showcased in this study, and explore NLP to investigate actual learning outcomes.

Funding

Not applicable.



Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Anonymised data for this research are available upon request.

Appendix 1. Computational thinking challenge and problems.

CT activities & level	Description of the problem	How the problem was adapted to fit our intervention	CT learning objective
#1. Matching Socks	<p>You are about to leave for holiday, but you forgot your socks! You race back to your room, but the power is off so you can't see sock colors. Never mind, because you remember that in your drawer there are ten pairs of green socks, ten pairs of black socks, and eleven pairs of blue socks, but they are all mixed up.</p> <p>How many of your socks do you need to take before you can be sure to have at least one matching pair?</p>	The Matching Socks activity was adapted to fit into college-entry students' standard by using it to teach how to debug a problem through tracing and abstraction	Problem abstraction and debugging.
#2. Towers of blocks	<p>"Sam, the little beaver, is playing with his toy blocks. He built seven beautiful towers, each one made with blocks of the same size. There are two ways to change the height of a tower: adding blocks to the top or removing blocks from the top. Adding or removing a block count as a move.</p> <p>For instance, if he changes the height of the leftmost tower to 2, it takes 3 moves (removing 3 blocks), and if he changes it to 7, it takes 2 moves (adding 2 blocks).</p> <p>Moving a block from one tower to another is considered as 2 moves.</p> <p>Sam wants all towers to be the same height, and he wants to make as few moves as possible."</p> <p>In</p>  <p>total, what is the minimum amount of moves that it will cost Sam to make all towers the same height?</p>	<p>The Tower of blocks activity presents a suitable approach to teach novices the concept of Stack in computer science with two possible operations- Push and Pop. We demonstrated these two operations to visualize Stacks as a very useful technique in many algorithms that solves more complex problems than block building. In addition, searching for the best solution such as the minimum cost is an optimization problem, which we taught the students using the Tower of Hanoi unplugged</p>	Problem decomposition, problem abstraction, Pattern recognition, algorithmic thinking, and recursive thinking.
#3. Marbles and Boxes	<p>"Marbles and boxes challenge presents a problem where a student has a box with 9 trays, as well as 9 marbles. Student can choose between 0 and 9 marbles and places them in the box according to the following rules:</p> <ul style="list-style-type: none"> Each marble is in a different tray. The total number of marbles in each row is even. The total number of marbles in each column is even."  <p>The task is to determine how many different ways can the student place the marbles in the box?</p>	The marble and boxes activity provided the opportunity to teach students about data representation, transmission, and data structure - arrays. Students can develop their thinking skills by observing the pattern and computational placing the marble by following the conditions	Algorithmic thinking, problem-solving, and simulation.
#4. Electric cars	<p>"The blue car can travel 4 km before it needs charging and needs 3 min to charge. The green car can travel 5 km before it needs charging and needs 4 min to charge. The purple car can travel 6 km before it needs charging and needs 5 min to charge.</p> <p>A car does not need to stop at every charging station, but when a car stops to charge, it must wait for the full charging time, no matter how much charge the car has left. All cars travel at the same speed, travelling 1 km every minute and start at node A fully charged.</p>	The electric cars present a CT problem that requires the students to find a shortest path between two nodes. The problem is similar to the famous Travelling Salesman (TS) problem used in teaching graphs and data structure. We are more interested in how to	Problem decomposition, problem abstraction, algorithmic thinking, and simulation.

(continued on next page)

(continued)

CT activities & level	Description of the problem	How the problem was adapted to fit our intervention	CT learning objective
	<p>The map below shows all the roads and charging stations between A and B. The numbers indicate the distance (in km) between each charging station.”.</p>	demonstrate these programming concepts and to explain optimization concepts in computer science.	
Based on the map, which car can travel from A to B in the fastest time?			

Appendix 2. Pre and post survey questionnaire (adapted from Yadav et al., 2011))

Pre-survey	<p>Open Ended Question:</p> <ul style="list-style-type: none"> - What is computational thinking? <p>Likert Scale Question:</p> <ul style="list-style-type: none"> - Indicate how familiar are you with the following computational thinking concepts? very familiar (4), familiar (3), somewhat familiar (2) not familiar (1) - algorithmic thinking - problem decomposition - problem abstraction - pattern recognition - recursive thinking - debugging - simulation <p>Multi-Choice Question:</p> <ul style="list-style-type: none"> - What are the two main concepts of computational thinking? - [A] Abstraction and Automation - [B] Algorithm and Analysis - [C] Debugging and Logical Thinking - [D] Problem decomposition - [E] Simulation - [F] All of the above
Post-survey	<p>ItemsLikert Scale Question:</p> <ul style="list-style-type: none"> - Indicate how confident are you with the following computational thinking concepts? very confident (4), confident (3), somewhat confident (2) not confident (1) - algorithmic thinking - problem decomposition - problem abstraction - pattern recognition - recursive thinking - debugging - simulation <p>Multi-Choice Question:</p> <ul style="list-style-type: none"> - How many moves did you used to complete the Tower of blocks problem? (Multi-choice) - [A] 3 moves - [B] 4 moves - [C] 5 moves - [D] 6 moves - [E] 7 moves - [F] 8 moves - [G] 9 moves - [H] 10 moves <p>Open Ended Question:</p> <ul style="list-style-type: none"> - What is computational thinking? - Describe the steps you have taken to solve the Tower of blocks problem - Describe your experience solving the puzzle

References

- Agbo, F. J., Olaleye, S. A., Bower, M., & Oyelere, S. S. (2023). Examining the relationships between students' perceptions of technology, pedagogy, and cognition: The case of immersive virtual reality mini games to foster computational thinking in higher education. *Smart Learning Environments*, 10(1), 16.
- Agbo, F. J. (2022). *Co-designing a smart learning environment to facilitate computational thinking education in the Nigerian context*. (Doctoral dissertation, Itä-Suomen yliopisto).
- Agbo, F. J., Oyelere, S. S., Suhonen, J., & Laine, T. H. (2021). Co-design of mini games for learning computational thinking in an online environment. *Education and Information Technologies*, 26(5), 5815–5849.
- Agbo, F. J., Oyelere, S. S., Suhonen, J., & Adewumi, S. (2019). A systematic review of computational thinking approach for programming education in higher education institutions. *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* (pp. 1–10).
- Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, 55(7), 832–835.
- Alhazbi, S. (2016). Active blended learning to improve students' motivation in computer programming courses: A case study. *Advances in engineering education in the Middle East and North Africa* (pp. 187–204). Cham: Springer.
- Anthony, B. M., Erdil, D. C., Glebova, O., & Montante, R. (2022). Unplugged parallelism for first-year CS majors. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, 2, 1079–1079.
- Araujo, A. L. S. O., Andrade, W. L., Guerrero, D. D. S., & Melo, M. R. A. (2019). How many abilities can we measure in computational thinking? A study on Bebras challenge. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 545–551).
- Basso, D., Fronza, I., Colombi, A., & Pahl, C. (2018). Improving assessment of computational thinking through a comprehensive framework. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research* (pp. 1–5).
- Bell, T., & Vahrenhold, J. (2018). CS unplugged—How is it used, and does it work?. *Adventures between lower bounds and higher altitudes* (pp. 497–521).
- Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin*, 39(2), 32–36.
- Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming: 12 years later. *ACM Inroads*, 10(2), 30–36.
- Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: Analyzing text with the natural language toolkit*. O'Reilly Media, Inc.
- Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(January), 993–1022.
- Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education, November* (pp. 65–72).
- Butgereit, L. (2016). Teaching recursion through games, songs, stories, directories and hacking. In *2016 International Conference on Advances in Computing and Communication Engineering (ICACCE)* (pp. 401–407). IEEE.
- Caballero González, Y. A., & García-Valcárcel Muñoz-Repiso, A. (2018). A robotics-based approach to foster programming skills and computational thinking: Pilot experience in the classroom of early childhood education. In *Proceedings of the 6th International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM 2018)*.
- Cambria, E., & White, B. (2014). Jumping NLP curves: A review of natural language processing research. *IEEE Computational Intelligence Magazine*, 9(2), 48–57.
- Cansu, F. K., & Cansu, S. K. (2019). An overview of computational thinking. *International Journal of Computer Science Education in Schools*, 3(1), 17–30.
- Carruthers, S., Gunion, K., & Stege, U. (2009). Computational biology unplugged!. In *Proceedings of the 14th Western Canadian Conference on Computing Education* (p. 126).
- Chen, Y., Yu, B., Zhang, X., & Yu, Y. (2016). Topic modeling for evaluating students' reflective writing: A case study of pre-service teachers' journals. In *Proceedings of the Ninth International Conference on Learning Analytics & Knowledge, April* (pp. 1–5).
- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed., p. 1988). New York, NY, USA: Lawrence Erlbaum Associates.
- Conde, M.Á., Fernández-Llamas, C., Rodríguez-Sedano, F. J., Guerrero-Higueras, A. M., Matellán-Olivera, V., & García-Peñalvo, F. J. (2017). Promoting Computational Thinking in K-12 students by using unplugged methods and robotics. In *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality* (pp. 1–6).
- Corral, L., Fronza, I., & Pahl, C. (2021). Block-based programming enabling students to gain and transfer knowledge with a no-code approach. In *Proceedings of the 22nd Annual Conference on Information Technology Education* (pp. 55–56).
- Curzon, P. (2013). cs4fn and computational thinking unplugged. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (pp. 47–50).
- Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014). Introducing teachers to computational thinking using unplugged storytelling. In *Proceedings of the 9th workshop in Primary and Secondary Computing Education* (pp. 89–92).
- de Jong, I., & Jeuring, J. (2020). Computational thinking interventions in higher education: A scoping literature review of interventions used to teach computational thinking. In *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research* (pp. 1–10).
- del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of Primary Education. *Computers and Education*, 150, Article 103832.
- Figureiredo, J., & García-Peñalvo, F. J. (2021). Teaching and learning strategies for introductory programming in university courses. In *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21)* (pp. 746–751).
- Fried, D., Legay, A., Ouaknine, J., & Vardi, M. Y. (2018). Sequential relational decomposition. In *Proceedings of the 33rd annual ACM/IEEE Symposium on Logic in computer science* (pp. 432–441).
- Fronza, I., Corral, L., & Pahl, C. (2019). Combining block-based programming and hardware prototyping to foster computational thinking. In *The 20th Annual Conference on Information Technology Education. (SIGITE'19)*.
- García-Peñalvo, F. J. (2018). Computational thinking and programming education principles. In *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*.
- García-Peñalvo, F. J., Conde, M.Á., Gonçalves, J., & Lima, J. (2021). Current trends in robotics in education and computational thinking. In *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality. TEEM'21* (pp. 9–12).
- Gardeli, A., & Vosinakis, S. (2017). Creating the computer player: An engaging and collaborative approach to introduce computational thinking by combining 'unplugged' activities with visual programming. *Italian Journal of Educational Technology*, 25, 36–50.
- Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, 42(1), 38–43.
- Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. *Computer science education*. Bloomsbury Academic.
- Hoffman, M., Bach, F., & Blei, D. (2010). Online learning for latent Dirichlet allocation. *Advances in Neural information Processing Systems*, 23.
- Jormanainen, Ilkka, & Tukiainen, Markku (2020). Attractive educational robotics motivates younger students to learn programming and computational thinking. In *Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality*.
- Kert, S. B., Erkoç, M. F., & Yeni, S. (2020). The effect of robotics on six graders' academic achievement, computational thinking skills and conceptual knowledge levels. *Thinking Skills and Creativity*, 38, Article 100714.
- Kuhail, A., Negreiros, M., & Seffah, J. (2021). Teaching recursive thinking using unplugged activities. *World Transactions on Engineering and Technology Education*, 19(2), 160–175.
- Lai, R. P. (2020). The design, development, and evaluation of a novel computer-based competency assessment of computational thinking. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 573–574).
- Lai, R. P. (2021). Beyond programming: A computer-based assessment of computational thinking competency. *ACM Transactions on Computing Education (TOCE)*, 22(2), 1–27.
- Looi, C. K., How, M. L., Longkai, W., Seow, P., & Liu, L. (2018). Analysis of linkages between an unplugged activity and the development of computational thinking. *Computer Science Education*, 28(3), 255–279.
- Lyon, J. A., & Magana, A. J. (2020). Computational thinking in higher education: A review of the literature. *Computer Applications in Engineering Education*, 28(5), 1174–1189.

- Maier, D., Waldherr, A., Miltner, P., Wiedemann, G., Niekler, A., Keinert, A., et al. (2021). Applying LDA topic modeling in communication research: Toward a valid and reliable methodology. *Computational methods for communication science* (pp. 13–38). Routledge.
- Merino-Armero, J. M., González-Calero, J. A., Cózar-Gutiérrez, R., & del Olmo-Muñoz, J. (2022). Unplugged activities in cross-curricular teaching: Effect on sixth graders' computational thinking and learning outcomes. *Multimodal Technologies and Interaction*, 6(2), 13.
- Montes-León, H., Hijón-Neira, R., Pérez-Marín, D., & Montes-León, S. R. (2020). Improving computational thinking in secondary students with unplugged tasks. *Education in the Knowledge Society*, 21, 12.
- Moreno-León, J., Román-González, M., & Robles, G. (2018). On computational thinking as a universal skill: A review of the latest research on this ability. In *IEEE Global Engineering Education Conference (EDUCON)*.
- Nishida, T., Kanemune, S., Idosaka, Namiki, M., Bell, T., & Kuno, Y. (2009). A CS unplugged design pattern. *ACM SIGCSE Bulletin*, 41(1), 231–235.
- Nkhoma, C., Dang-Pham, D., Hoang, A. P., Nkhoma, M., Le-Hoai, T., & Thomas, S. (2020). Learning analytics techniques and visualization with textual data for determining causes of academic failure. *Behaviour & Information Technology*, 39(7), 808–823.
- Oyelere, A. S., Agbo, F. J., & Oyelere, S. S. (2023). Formative evaluation of immersive virtual reality expedition mini-games to facilitate computational thinking. *Computers & Education: X Reality*, 2, 100016.
- Oztok, M., Zingaro, D., Makos, A., Brett, C., & Hewitt, J. (2015). Capitalizing on social presence: The relationship between social capital and social presence. *The Internet and Higher Education*, 26, 19–24.
- Peteranetz, M. S., Flanigan, A. E., Shell, D. F., & Soh, L. K. (2017). Computational creativity exercises: An avenue for promoting learning in computer science. *IEEE Transactions on Education*, 60(4), 305–313.
- Roberts, M. E., Stewart, B. M., & Tingley, D. (2019). Stm: An R package for structural topic models. *Journal of Statistical Software*, 91, 1–40.
- Röder, M., Both, A., & Hinneburg, A. (2015). Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* (pp. 399–408).
- Rodríguez, B., Kennicutt, S., Rader, C., & Camp, T. (2017). Assessing computational thinking in CS unplugged activities. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 501–506).
- Rojas-López, A. and García-Peñalvo, F.J. 2019. Initial learning scenarios based on the computational thinking evaluation for the course Programming fundamentals at INACAP. In *Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality*. 6–12.
- Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, 14(1), 1–15.
- Salguero, A., Griswold, W. G., Alvarado, C., & Porter, L. (2021). Understanding Sources of Student Struggle in Early Computer Science Courses. In *Proceedings of the 17th ACM Conference on International Computing Education Research* (pp. 319–333).
- Soh, L. K., Shell, D. F., Ingraham, E., Ramsay, S., & Moore, B. (2015). Learning through computational creativity. *Communications of the ACM*, 58(8), 33–35.
- Srinivasa-Desikan, B. (2018). *Natural language processing and computational linguistics: A practical guide to text analysis with python, gensim, spaCy, and keras*. Packt Publishing Ltd.
- Sun, L., Hu, L., & Zhou, D. (2021). Which way of design programming activities is more effective to promote K-12 students' computational thinking skills? A meta-analysis. *Journal of Computer Assisted Learning*, 37, 1048–1062.
- Sunday, K., Ocheja, P., Hussain, S., Oyelere, S., Samson, B., & Agbo, F. J. (2020). Analyzing student performance in programming education using classification techniques. *International Journal of Emerging Technologies in Learning (IJET)*, 15(2), 127–144.
- Tonbuloglu, B., & Tonbuloglu, İ. (2019). The effect of unplugged coding activities on computational thinking skills of middle school students. *Informatics in Education*, 18(2), 403–426. <https://doi.org/10.15388/infedu.2019.19>
- Vihavainen, A., Airaksinen, J., & Watson, C. (2014). A systematic review of approaches for teaching introductory programming and their influence on success. In *Proceedings of the tenth annual Conference on International Computing Education Research*.
- Weese, J. L. (2016). Mixed methods for the assessment and incorporation of computational thinking in K-12 and higher education. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*.
- Wen, M., Yang, D., & Rose, C. (2014). Sentiment Analysis in MOOC Discussion Forums: What does it tell us? *Educational Data Mining 2014*.
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717–3725.
- Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Shoop, R., & Baehr, E. C. (2017). Developing computational thinking through a virtual, robotics programming curriculum. In , 18. *The Robotics Institute, Carnegie Mellon University ACM Transactions on Computing Education* (p. 1).
- Yadav, A., Caeli, E. N., Ocak, C., & Macann, V. (2022). Teacher education and computational thinking: Measuring pre-service teacher conceptions and attitudes. In , 1. *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education* (pp. 547–553).
- Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). *Computational thinking in elementary and secondary teacher education*. In . 14 pp. 1–16).
- Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J.T. 2011. Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 465–470).
- Yang, L., Xin, T., Zhang, S., & Yu, Y. (2022). Predication of writing originality based on computational linguistics. *Journal of Intelligence*, 10(4), 124.
- Zhang, L. C., Nouri, J., & Rolandsson, L. (2020). Progression of computational thinking skills in Swedish compulsory schools with block-based programming. In *Proceedings of Twenty-second Australasian Computing Education Conference. (ACE'2020)*.