A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions

Agbo, Friday Joseph

ACM Press
A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions

Friday Joseph Agbo
School of Computing
University of Eastern Finland
Joensuu, Finland
fridaya@uef.fi

Solomon Sunday Oyelere
School of Computing
University of Eastern Finland
Joensuu, Finland
solomon.oyelere@uef.fi

Jarkko Suhonen
School of Computing
University of Eastern Finland
Joensuu, Finland
jarkko.suhonen@uef.fi

Sunday Adewumi
Department of Computer Science
Federal University Lokoja
Lokoja, Kogi State, Nigeria
sunday.adewumi@fulokoja.edu.ng

ABSTRACT
This study examined how computational thinking (CT) has been used to teach problem-solving skills and programming education in the recent past. This study specifically (i) identified articles that discussed CT approach for programming education at higher education institutions (HEIs), (ii) classified the different CT approaches and tools employed for programming education at HEIs, (iii) synthesised and discussed results that are reported by relevant studies that utilized CT for teaching programming at HEIs. A systematic literature review methodology was adopted in this study. Out of 161 articles retrieved, 33 of them that met the inclusion criteria were reviewed. Our study revealed that the use of CT at HEIs for programming education began in 2010; many studies did not specify the context of use, but the use of CT is found to be gaining grounds in many contexts, especially the developed countries; course design approach was mostly employed by educators to introduce CT at HEIs for programming education. Furthermore, this study pointed out how CT approach can be explored for designing a smart learning environment to support students in learning computer programming.

CCS CONCEPTS
• Social and professional topics → computational thinking

KEYWORDS
computational thinking, problem-solving, programming, higher education, undergraduates, smart learning

INTRODUCTION
Computational thinking (CT) has the capacity to improve programming education by building the problem-solving and algorithmic skills of learners [1]. CT involves having thinking skills, i.e., “ways of thinking and practicing that are sharpened and honed through practice”—Denning and Tedre [2] (pp.6). Among other skills, CT are envisioned to make the developing of programming and problem-solving skills a flexible experience. Learning of computer programming remains one of the challenges facing students and educators [3] [4] [5]. While CT is the fundamental approach towards understanding, abstracting and modelling a problem with the intention of providing a solution, computer programming is the actual transcription of the abstraction into a computer understandable language. The stages of computer programming involve writing, testing, debugging and running a set of codes using different programming languages [6] [7]. These stages can be tasking and complex, especially for novices. Researchers [6] [8] [9] have shown that often, programming is relatively considered difficult among other science-related courses, and building the skill takes time and commitment [3]. However, efforts to ease the task of learning to develop one’s problem-solving skills and programming education exist [4]. Some authors applied approaches such as visualization, games, puzzles, and computational thinking to motivate students and increase interactions between learners and educators [8] [9] [10]. For instance, Oyelere et al. [9], Li [11] designed a system/approach for teaching computing education with puzzles to motivate students’ learning.

Computational thinking was made popular by Wing [12] in 2006, stating that CT “involves solving problems, designing systems, and understanding human behaviours, by drawing on the concepts fundamental to computer science” [12] (pp.33). Since then, varied definitions of CT have been provided.

In this paper, we discussed the various definitions of CT by authors from different perspectives and presented a definition that is focused on using CT for problem-solving as a fundamental step towards teaching and learning programming education. The study also investigated how CT is being utilized for teaching and learning
at higher education institutions (HEIs). Furthermore, this study investigates the use of CT approach by educators and scholars in teaching programming education at the HEIs. As far as we know, there is no systematic literature review of CT as a programming teaching approach conducted within the premise of HEIs. Hence, this study aims to investigate different CT approaches used for teaching and learning programming at HEIs to explore the opportunity for ingrain CT feature into the design of a smart learning environment (SLE). This study is important because it helps to give direction to the design of a SLE to support programming education based on CT approach. Besides, CT skills are fundamental knowledge for students at the HEIs [1]. Furthermore, this study will provide insight to researchers and developers of contemporary learning environment, specifically in the area of computing education, on ways CT can enhance the development of a SLE. To achieve the aim of this paper, the following objectives were outlined for this study: (i) identify articles that discussed CT approach for programming education at HEIs, (ii) classify the different CT approaches and tools used for programming education at HEIs, (iii) discuss results that are reported by scholars that used CT approaches for programming education at HEIs. Based on the objectives, this study seeks to provide answers to the following research questions: 

**RQ1.** How has the use of CT been explored for teaching programming in HEIs? 

**RQ2.** What are the ways CT approach has been used to teach programming education in HEIs? 

**RQ3.** How has the use of CT approach for teaching programming impacted students learning experience? 

## 2 BACKGROUND

### 2.1 Definition of CT

According to Denning and Tedre [2] (pp.10), the historical perspective of computational thinking exists even before the emergence of electronic computers. However, since the discussion of CT by Wing [12]—centred on the fundamental knowledge for computer programming, it has fast become popular in the computer science domain [11, 1]. Although there exists no consensus definition of CT, and as presented by Velázquez [13], most of the definitions are ambiguous. For instance, Barr and Stephenson [14] provided what they called “operational definition” of CT as a problem-solving process that includes a set of characteristics. These characteristics are: i) formulating a problem in a way that one can use a computer or other tools to solve them; ii) organising and analysing the data logically; iii) abstracting the data in form of models and simulations; iv) automating the solution through algorithmic steps; v) identifying, analysing, and implementing a possible solution with the goal of achieving the most efficient combination of steps and resources; vi) generalizing and applying the problem-solving steps to varieties of problems in other areas of endeavours. In addition, Mannila et al. [15] have defined CT as “a term meant to encompass a set of concepts and thought processes that aid in formulating problems and their solutions in different fields in a way that could involve computers.” [15] (pp. 1). One of the points stressed by many authors is the fact that the concept of CT is a fundamental background in computer science [16], which deals with scientific thinking in the technological age. Also, in many definitions of CT, the word ‘problem’ is mostly the focus. For instance, it is common to encounter words like problem-solving, problem decomposition, and problem abstraction. From these various definitions of CT, some keywords emerge; problem-solving, systems design, and human behaviour [12, 17, 18]. These keywords are relevant when discussing the fundamental concepts of computer science.

In trying to solve the very many human problems, CT promises to be a good approach by applying human-computer interaction and relationship perspectives [11]. According to Wing [12], CT is beyond computer programming or the ability to write codes, CT involves more thinking about multiple levels of abstractions. Problem decomposition and abstraction is the first approach towards problem-solving; next is the algorithm design, which eventually leads to computer programming, and then, a concrete solution (See Figure 1).

One of the ways to help the students develop programming knowledge is to build problem-solving skills. Problem-solving skills involved a critical thinking process about the understanding of the problem on which to develop concrete steps towards its solution. This skill is needed to develop students programming experience since an explicit path to the solution can be defined. The application of CT to problem-solving is not limited to computer science alone, but every field of science and indeed, all human endeavours. However, this paper discussed CT as a fundamental step towards building problem-solving skills that can aid programming education using the CT approach to achieve this objective.

![Figure 1: Definition of computational thinking and problem-solving steps](image)

### 2.2 CT for problem-solving and programming education

Every day, humans are faced with real-world problems that require some thinking and logic in order to solve them. People apply the concept of CT consciously or unconsciously. CT’s concept may not be necessarily complex but attempts to solve every problem by defining some simple computational steps. These steps, however, can involve abstraction of the problems, creating models, designing algorithms, and verifying results, and ascertaining the viability of the solution [11]. Since the discussion on CT emerged, many researches demonstrate the practical application in problem-solving, algorithm, system modelling, and automation. Most of these researches presented results of CT application in teaching K-12 and STEM [14, 19]. Although studies that investigated the approaches for integrating CT in teaching specific disciplines exists [20], there seems to be limited attempt to study the application and
impact of CT at HEI to teach programming education and problem-solving. For example, Araujo et al., [21] studied the techniques and approaches used by researchers and teachers to assess the development of CT abilities in students. Their work explored previously proposed approaches to promote CT and classified them based on their capability to assess CT in education. Similarly, Moreno-León [22] reviewed recent investigations that study CT from different perspectives. The study further classified technologies to learn and teach CT into five, which includes i) unplugged activities; ii) arrow-based visual environment; iii) block-based visual environment; iv) textual programming language; v) connected with the physical world. In an attempt to also investigate how CT in schools is assessed, [22] highlighted three main approaches used by educators—CT-test, Bebras, and Dr. Scratch. Another recent study by [23] reviewed literature on CT and reported on the diversity in definitions, interventions, assessments, and models. In computing education, for example, Repenning et al. [24] employed the concept of designing 3D computer graphics as a CT approach in the teaching of computer science education. This approach, beyond motivating the learners, presents the visualization of computer programming to aid learners’ understanding.

Similarly, Philip et al. [25] studied the relationship between CT, problem-solving and programming education with students of HEI. They tried to present a pragmatic approach towards developing CT and problem-solving skills for novices of computer programming. This study [25] showed that CT is fundamental in introductory programming; it demonstrates that in problem-solving, defining the abstraction and modelling precedes both programming logic and constructs of the solution. The two levels of problem-solving are problem analysis and design, and implementation (see Figure 1).

The role of programming in computing education, according to [26] has been a major debate. Nowadays, programming education has grown into a discipline and further into a professional career. The traditional ways of learning programming using languages such as Java and C++ can be challenging and boring. However, using visualization technique in supporting programming education can improve computational thinking and problem-solving skills of learners [19]. For example, when trying to teach freshmen the basics of programming education, it is better to use visual programming language such as Scratch rather than the traditional approach, which may be difficult to understand [25]. Visual programming has been researched recently to aid the students’ understanding of computer programming [19] and simplify their learning experience. A block programming concept is used to present visualization of the traditional programming syntaxes, so that the students do not necessarily have to understand its syntax and semantics that may appear abstract. A practical example in Figure 2a-b depicts a simple program for the number system.

A study to investigate the use of CT and its adoption at the HEI was conducted by Czerkawski and Lyman [27]. Their study revealed that efforts are being made to design and introduce CT courses at the HEI and to implement CT in computer science curricula. All these studies play a significant role in advancing the research contribution in CT and provide more insight into the level of efforts being made by scholars regarding the current situation. However, they do not provide knowledge regarding the state-of-the-art tools and interventions of CT utilized at HEIs and their impact on students at HEIs.

Besides, our study intends to explore the opportunity of ingraining CT features into the design of SLE for programming education, since CT is believed to be a foundation for teaching and learning of programming by novices and freshmen.

**Figure 2a. Traditional coding in JavaScript**

3 RESEARCH METHODOLOGY

3.1 Procedure

We began by collecting relevant literature on CT from selected multidisciplinary databases following the systematic review approach [23]. During the data collection, we considered only articles that were peer-reviewed and published in scholarly journals and conferences. Our search obtained relevant articles on studies regarding computational thinking, problem-solving, and programming education at the HEIs. Thus, our search keywords were computational thinking, problem-solving, programming, computer science, and undergraduate. The search was conducted in five databases—IEEE Xplore, ACM, ScienceDirect, and Springer Link. The decision to select these databases for our search was because they warehouse articles that are published in the field of education, STEM (science technology engineering and mathematics), and computer science [28].

In each of the databases, the same keywords were used to conduct an advanced search. The search was targeted at metadata aspects of the literature. The metadata for an article contains important information about that article, for example, the title of an article, keywords, and abstracts are part of metadata. An example of the combination of search string is (((computational thinking) AND problem solving) AND programming) OR computer science). These strings were modified when necessary to suit the search pattern of each databases. The results returned by the search were subjected to list of criteria in order to select only articles that are relevant to our study.
3.2 Paper screening

Screening of the paper was conducted by applying some sets of inclusion and exclusion criteria. The process for selection started by scanning through the metadata before downloading the document for further screening. For instance, the title, keywords and abstract were first skimmed through to ensure that the article is focused on our study. The inclusion and exclusion criteria are presented in Table 1 while the results returned after running the queries in each database are shown in Table 2.

Table 1. Inclusion and exclusion criteria

<table>
<thead>
<tr>
<th>Inclusion criteria</th>
<th>Exclusion criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Articles that focuses on computational thinking, problem-solving, and programming</td>
<td>Articles that do not focus on the keywords or not written in English Language are</td>
</tr>
<tr>
<td>education</td>
<td>excluded</td>
</tr>
<tr>
<td>Articles that are published in a peer-reviewed journals or conferences</td>
<td>Materials that are not peer-reviewed (audio/video files, PPT, e.t.c) are left out</td>
</tr>
<tr>
<td>Articles that either presented a concrete programming artefacts, evaluated a</td>
<td>Theoretical and conceptual studies are removed</td>
</tr>
<tr>
<td>solution for programming or design a study to explore CT</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Summary of search results

<table>
<thead>
<tr>
<th>Database</th>
<th>Search Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>47</td>
</tr>
<tr>
<td>IEEE Xplore</td>
<td>64</td>
</tr>
<tr>
<td>ScienceDirect</td>
<td>22</td>
</tr>
<tr>
<td>SpringerLink</td>
<td>28</td>
</tr>
<tr>
<td>Total</td>
<td>161</td>
</tr>
</tbody>
</table>

The next stage of the selection process considered only studies that are peer-reviewed and published in a journal or conferences. The last major selection stage examined whether the paper utilized a tool, artefact or designed a course to apply in a real-life teaching/learning scenario and presented empirical evaluation of the study outcome.

4 RESULTS

This section presents the findings by first examining the trends in the use of CT for programming education at the higher education institutions (HEIs). Further, we investigate the different CT approaches and tools used for programming education at HEIs.

4.1 CT for programming education in HEIs

4.1.1 Studies distribution by year of publication. Figure 4 presents the studies distribution according to the year of publication. Although we do not delimit the study by defining date range, but data gathered revealed the trends when the topic “Computational Thinking” began to gain scholar’s attention. Regarding the use of CT for programming education at HEIs, we found 3 studies that were conducted in 2010. More studies were found in 2017 and 2018.

4.1.2 Studies distribution by outlets of publication. Our study was focused on collecting data from two major outlets (i.e., conferences and journals) as explained in section 2.1. Table 3 delineates the distribution of studies by the publication outlet.

<table>
<thead>
<tr>
<th>Publication Outlet</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Journals</td>
<td>10</td>
</tr>
<tr>
<td>Conferences</td>
<td>23</td>
</tr>
</tbody>
</table>

4.1.3. Context of study. Table 4 shows the context in terms of country where each study was conducted. Although many of the study do not explicitly mentioned the country where it was conducted, our study shows that United States had more frequency among the countries.

<table>
<thead>
<tr>
<th>Country</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canada</td>
<td>1</td>
</tr>
<tr>
<td>India</td>
<td>1</td>
</tr>
<tr>
<td>United States</td>
<td>6</td>
</tr>
<tr>
<td>China</td>
<td>1</td>
</tr>
<tr>
<td>Ireland</td>
<td>1</td>
</tr>
<tr>
<td>Japan</td>
<td>1</td>
</tr>
<tr>
<td>Unspecified</td>
<td>22</td>
</tr>
<tr>
<td>Paper title</td>
<td>Author(s)</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Connecting undergraduate programs to high school students: teacher workshops on computational thinking and computer science</td>
<td>Morreale [29]</td>
</tr>
<tr>
<td>Using video to explore programming thinking among undergraduate students</td>
<td>Wellington and Ward [30]</td>
</tr>
<tr>
<td>Promoting Computational Thinking with Programming</td>
<td>Selby [31]</td>
</tr>
<tr>
<td>A Qualitative Study of Students’ Computational Thinking Skills in a Data-Driven Computing Class</td>
<td>Yuen and Robbins [34]</td>
</tr>
<tr>
<td>A Case Study of Computer-based Problem-Solving Skill Development by Using Spreadsheet Software</td>
<td>Chatvichienchai [36]</td>
</tr>
<tr>
<td>Teaching Inclusive Thinking in Undergraduate Computing</td>
<td>Palan et al. [37]</td>
</tr>
<tr>
<td>A Novel Interdisciplinary Course in Gerontology for Disseminating Computational Thinking</td>
<td>Yang, et al. [39]</td>
</tr>
<tr>
<td>A Pragmatic Approach to Develop Computational Thinking Skills in Novices in Computing Education</td>
<td>Philip et al [25]</td>
</tr>
<tr>
<td>A Fun-Learning Approach to Programming: An Adaptive Virtual Reality (VR) Platform to Teach Programming to Engineering Students</td>
<td>Chandramouli et al. [40]</td>
</tr>
<tr>
<td>Teaching and Learning Computational Thinking through Solving Problems in Artificial Intelligence on Designing Introductory Engineering and Computing Courses</td>
<td>Silapachote and Srisuphab [41]</td>
</tr>
<tr>
<td>Design Thinking and Computational Modeling to Stop Illegal Poaching</td>
<td>Padmanabhan et al. [42]</td>
</tr>
<tr>
<td>Teaching Computational Thinking to Entry-level Undergraduate Engineering Students at Amrita University</td>
<td>Shyamala et al. [45]</td>
</tr>
<tr>
<td>CS for ALL: Introducing Computational Thinking with Hands-on Experience in College</td>
<td>Jung et al. [44]</td>
</tr>
<tr>
<td>The effect of simulation games on the learning of computational problem solving</td>
<td>Liu [45] et al.</td>
</tr>
<tr>
<td>A serious game for developing computational thinking and learning introductory computer programming</td>
<td>Kazimoglou et al. [46]</td>
</tr>
<tr>
<td>Learning Programming at the Computational Thinking Level via Digital Game-Play</td>
<td>Kazimoglou et al. [47]</td>
</tr>
<tr>
<td>A validity and reliability study of the computational thinking scales</td>
<td>Kormaz et al. [48]</td>
</tr>
<tr>
<td>Computational thinking development through creative programming in higher education</td>
<td>Romero et al. [49]</td>
</tr>
<tr>
<td>A Case Study to Promote Computational Thinking: The Lab Rotation Approach</td>
<td>Cai et al. [50]</td>
</tr>
<tr>
<td>From Computational Thinking to Constructive Design with Simple Models</td>
<td>Margaria [18]</td>
</tr>
<tr>
<td>Applying online externally-facilitated regulated learning and computational thinking to improve students’ learning</td>
<td>Tsai and Tsai [51]</td>
</tr>
<tr>
<td>Supporting Undergraduate Computer Science Education Using Educational Robots</td>
<td>Saad et al. [52]</td>
</tr>
<tr>
<td>Infusing Computational Thinking across Disciplines: Reflections &amp; Lessons Learned</td>
<td>Pollock et al. [53]</td>
</tr>
<tr>
<td>Computational Thinking in a Game Design Course</td>
<td>Settle [53]</td>
</tr>
<tr>
<td>MPCT – Media Propelled Computational Thinking</td>
<td>Freudenthal et al. [54]</td>
</tr>
<tr>
<td>Comparing Block-Basedand Text-BasedProgramming in High School Computer Science Classrooms</td>
<td>Weintrop and Wilensky [55]</td>
</tr>
<tr>
<td>Teaching Computational Thinking to Non-computing Majors Using Spreadsheet Functions</td>
<td>Yeh et al. [56]</td>
</tr>
<tr>
<td>Learning to think like a trainer: bringing Scratch for Educational Sciences professional’s formation</td>
<td>Almeida and Pessoa [57]</td>
</tr>
</tbody>
</table>
4.2 Tools, interventions and design approach to CT for programming education at HEIs

In order to provide an answer to the research objectives highlighted in section 1, the study investigated the tools and research design employed for teaching/learning of computational thinking at HEIs. Analysis shows that the majority of the study reviewed utilized course design approach to teach computational thinking at HEIs. Although some studies did not explicitly mention the approach used, however, workshop, seminar, and exploratory study forms part of the strategies adopted in introducing CT for computing education. A few studies introduced computational thinking at higher education institutions by using a concrete tool and technology. These tools are developed/adopted to build problem-solving, algorithmic, and programming knowledge among the freshmen. Some of the tools leveraged on the common software such as Microsoft Excel, while some are designed to be game/puzzle-based approach. However, many of the studies did not specify the name of the tool or technology utilized.

5 DISCUSSIONS

In this review, we analyze scholarly articles on the use of computational thinking approach for programming education at higher education institutions. By doing so, we investigated the study design, tools, and technology adopted, and the impact of these studies. This section discusses the findings presented in section 3 regarding the adoption of CT for programming education at HEIs and the impacts of such studies on the students within the context of discussion. Although studies on the use of CT approach for teaching programming education at lower education exists [58] [59] [60], our study has shown that CT courses have been designed and administered to undergraduate entry-level students to prepare them for introductory programming courses (IPCs) that requires rigorous logical and algorithmic skills. For example, Ref [46] [33] [34] [36] [49] [44] [39] utilized CT approaches and tools such as Wiki-based project, MATLAB, puzzles, spreadsheet micro, Scratch, and robot to introduce new students to IPCs and problem-solving.

5.1 Reflections from the use of CT approach for programming education at HEIs and its impact

Educators and researchers have reported their experiences from the conduction, analysis, and evaluation of the CT approach introduced among the freshers undergraduate at HEIs. While some results showed a positive outcome, others have mixed feelings about the way different categories of students reacted to the CT courses [39, 61, 25]. On the positive side, the introduction of CT as a prerequisite for introductory programming courses was shown to bridge the gap that exists between students with and without programming background [25]. For instance, feedback from students in a study shows that “the course has helped them to improve their problem solving skills...and they performed well in their regular programming sessions” [25](pp.202). In another study, the use of games to teach students CT and problem-solving skills showed that some students had flow experience; some were bored and anxious during the simulation; some demonstrated analytic reasoning strategies to learn computational problem-solving skills, while others applied the trial-and-error approach to problem-solving [33] [45]. In addition, a study that teaches programming within a data-driven context in a university and allowed students to learn computing concepts and computational thinking by writing programs in MATLAB was conducted [34]. This study shows that computational and visualization tasks appear to be closely linked, and the visualization component appears to provide valuable feedback for students in the programming tasks accomplishment. However, a few of the articles did not present a concise result about the impact of CT on students at HEIs; fundamental concern raised about the distinctive boundary-line between computer science and CT; curriculum design that compliments the field of computer science also requires further discussion. Overall, the introduction of CT at undergraduate level have been reported to impact positively on the students by motivating interest in programming, enhancing problem-solving skills and cognitive capabilities, and improving interdisciplinary teamwork participation (see Table 5).

5.2 CT approach for programming education in a smart learning environment

Nowadays, there is high interest on adopting a smart learning approach for teaching and learning programming education [62]. Smart learning environment (SLE) is a new paradigm in the learning ecosystem that seeks to enhance the learning experience. The aim of SLE is to allow for learner-centred approach by personalizing, adapting, and contextualizing learning. The use of CT for problem-solving and programming education is a viable approach towards designing an interactive SLE that can aide the learning process for especially novices. This section discusses the use of CT for teaching/learning of programming education as a potential approach towards designing a SLE at HEIs. In our previous work Agbo et al. [62] had identified four components that are relevant when designing a SLE. These components, which consists of user, context, pedagogy, and tools, are investigated in this study and are linked to the current findings. This study revealed specific context of use, for instance, undergraduate students in China [50] [25], University students in Ireland [18], computer science students in a Canadian University [49], and US high school teachers [29]. Similarly, the pedagogical component of these studies that we have reviewed are presented in Table 2. The result showed that the majority of the articles utilized course design approach as their pedagogy, and a few studies employed workshop/seminar approach. Others such as [36] have used project design approach and integrated into the curriculum of a semester or section to evaluate the students collaborative and cooperative problem-solving skills. For instance, some authors designed courses that were focused on
computer programming concepts such as variables, data types, algorithmic thinking, functions, recursion, and problem-solving techniques [39, 25, 42, 41, 43]. Besides, puzzle-based techniques, games, and creative exercises were other strategies adopted for introducing the concept of CT and problem formulation at the undergraduate level of education [33, 61, 17]. In addition, our analysis based on the type of tools and technology deployed to conduct the studies showed that majority did not specify the tool. However, tools such as Microsoft Excel Spreadsheet [36], MATLAB [34], Puzzles [33], Scratch and mBot [49] [44], were employed at different context of HEIs. The integration of CT in a smart learning environment can create a more supportive platform for teaching/learning of programming. While the CT is a pedagogical approach adopted to present learning contents, the SLE allows the modeling of users to enhance learning process by personalizing, adapting, and motivating learning.

5.3 Sample cases of CT approach for programming concepts

In this section, we present some of the CT approaches used to teach students on how to develop their thinking ability, problem-solving skills, and improve their programming knowledge. These cases are adapted from Lagama [34]. The first case of CT approach used to teach programming concept is about an aged long historical problem with recreational mathematics. This problem is called ‘river crossing puzzle.’ This puzzle teaches the students how to think algorithmically. The storyline to the puzzle consists a Wolf, a Goat, and a Cabbage. “A farmer has to take all these items across a river, and the only boat available can transport the farmer and at most one of his possessions at a time” (see Figure 7 left). A second case has to do with a mathematical logic, which is an important aspect of building one’s computational thinking skill.

![Figure 7: Case 1 & 2 of CT approach. adapted from [34]](image)

Most real-world problems that require computer programming require logic expressions and operators. A puzzle called ‘a fork in the road’ was used to introduce logic. The situation is such that you’ve been out alone for a long walk on the island, and unfortunately, you got lost. You are very tired, and it’s already getting dark, at this point you come to a fork in the road with a conspicuous signpost on the ground, it is written ‘one road leads back home and the other road leads to den of venomous snakes.’ Fortunately, at the fork, there’s a native who will only answer a yes/no question for you. But, you do not know whether the native is a Truth teller or Liar. What question should you ask to determine with certainty the way back home?

In building problem-solving skills for freshmen, the use of common and real world-problems enhances their understanding, to creatively think of computational steps towards a solution. Among other cases, the examples we adapted shows that CT can be used to teach programming concepts. Hence, it is important to design a learning tool that allow users teach/learn programming concept through CT approach while abstracting problems that are familiar within a context.

6 CONCLUSIONS

Computational thinking has become a fundamental approach to building problem-solving skills. The use of CT approach to introduce freshmen to introductory programming courses at the higher education institutions has been gaining research interest in the recent past. This study has identified trends and approaches employed by educators to introduce CT at HEIs and its impact on students. It was discovered that there is increasing number of research publications regarding the use of CT to promote programming education at HEIs since 2017. The research publication regarding the use of CT for programming education seems to be more in the US Universities. The possible explanation for this trend can be linked to the presence of Computer Science Teachers Association (CSTA) annual conference1 that holds in the US and has existed for about two decades. The penetration of CT approach into the higher education institutions indicates good sign for computing education as it develops student’s cognitive ability and prepares the freshmen for the core programming courses. This way, the existing gap between the students who have programming background and those without the background can become narrow since CT teaches the fundamentals of programming concepts. Also, the educational technologists can use the CT approach to design a smart learning solution that enhances the students to have a better learning experience. In fact, we believe that the future of teaching/learning of computer programming is a SLE with the CT approach; that provides the opportunity for a more impactful learning experience. Our future research will be in the direction of exploring models that employed CT approaches to implement a SLE to aide teaching and learning of programming education.

7 REFERENCES


[38] K. Boom, M. Bower and A. Arguel, “Relationship between Computational Thinking and a Measure of Intelligence as a General Problem-Solving Ability,” in Proceedings of the 23rd Annual ACM Conference on Innovation and Technology, Cyprus.


