Research Article

Effect Of Execution Time Analysis Epl Program For Computational Thinking Of Elementary School Students

Woojong Moon, Jonghoon Kim
Jeju National University, Jeju, Republic of Korea

***Article History**: Received:11 January 2021; Accepted: 27 February 2021; Published online: 5 April 2021*

***Abstract*—** Software education has emerged as a hot topic around the world, with the goal to raise interests on computational thinking. However, assessments on the computational thinking have not been actively conducted thus far. According to a study by Lee (2019), which analyzed 138 papers on computational thinking published in Korean journals from 2015 to 2018, software education has been introduced and studies on computational thinking are being conducted, but studies on teaching methods that2 improve computational thinking are needed. In this study, we developed and applied a primary educational programming language(EPL) program focused on execution time analysis aimed at improving computational thinking. By using the "Bebras Challenge" as an assessment tool and SPSS as a statistical tool, educational effects were analyzed through the results of pre- and post-computational thinking assessments. The analysis outcomes showed that the EPL education focused on execution time analysis was effective in improving the computational thinking of elementary school students. Putting execution time analysis EPL into primary software education as an educational topic will be effective in improving computational thinking.

***Keywords*—**Algorithm, Execution time, EPL education, Computational thinking

1. INTRODUCTION

Software education has been widely introduced and provided to elementary and middle schools in various approaches. Most elementary schools currently offer basic programming education through block-based programming languages as these languages provide audiovisual elements that are suitable for the cognitive level of elementary school students and can easily attract their interests [1]. However, assessments on the computational thinking, which students develop through the programming learning process, have not been actively conducted. According to a study by Lee (2019), which analyzed 138 papers on computing thinking published in Korean journals from 2015 to 2018, software education has been introduced and studies on computing thinking skills are being conducted, but studies on teaching methods that improve computing thinking are needed [2]. Some of the reasons for this include lack of awareness regarding the importance of computational thinking assessments, absence of assessment tools that are easily accessible to instructors and learners to evaluate computational thinking, and complexity of computational thinking assessment methods [3]. The goal of software education should involve "cultivating creative convergent talents having computational thinking"; therefore, research on improving the computational thinking of students is necessary. Accordingly, this study developed an elementary educational program using execution time-oriented educational programming language (EPL) to help improve the computational thinking of elementary school students.

2. THEORETICAL BACKGROUND

2.1. ALGORITHM

Algorithm refers to a logical process of procedure that indicates a resolution procedure to solve a problem in an easy way. In other words, various methods of trying to solve problems are the concepts of algorithms. It is the core concept of computer and programming, and it is possible to increase problem-solving power through algorithm education. As the goal of software education is generally represented by computational thinking, it is very important to have a problem-solving method expressed in algorithms [4]. This study focused on finding two algorithms that solve the same problem and comparing the execution time of each algorithm.

2.2. EXECUTION TIME

Execution time refers to the time the algorithm takes to generate an output based on a given input. The analysis result of the algorithm execution time is referred to as time complexity, which enables algorithm's performance assessment by counting the number of computational operations required to execute the algorithm. As the

number of computational operations in the algorithm primarily varies on the algorithm execution time, the algorithm execution time should be carefully handled to enable efficient problem solving [17]. According to a study by Kim (2020), even though one may make the same result by solving the same problem, the time taken to solve the problem may differ by each algorithm, which is directly linked to efficient problem solving. By analyzing algorithm's execution time, the capability is enhanced to understand the operational mechanism of algorithm and choose an adequate and efficient algorithm to solve a problem. Through education, which analyzes algorithm execution time and chooses and applies an algorithm suitable for problem solving, it can build computational thinking ability to solve problems efficiently [5]. In this study, to allow students to easily compare the execution times and to simplify the education process by teaching students according to their capability levels, the number of computational operations is counted directly in the Scratch program. Further, the educational program was designed to encourage the students to try various approaches to find an algorithm that minimizes the execution time.

2.3. EPL EDUCATION

EPL is a computer language designed to prevent students from feeling overwhelmed when learning to program. A programming language generally requires learners to spend a large amount of time in grasping basic grammar or structure, causing cognitive burdens. By providing simple structures, the EPL is designed to allow young learners and beginners to reduce cognitive burdens and to further engender students' interest in programming learning [6]. According to a study by Kim (2019), which analyzed 39 EPL papers published in Korea over the past decade, education using EPL has been found to be useful in greatly improving elementary school students' thinking skills [7]. In this study, the EPL education method was adopted to improve the computational thinking ability of elementary school students by focusing on algorithm execution time. Scratch, which is the most widely used EPL method and has many advantages, was used to implement the proposed educational program.

2.4. COMPUTATIONAL THINKING

The phrase computational thinking first emerged in the computer science education community in 2006, after Wing defined it as "thinking like a computer scientist when tackling a problem to be solved". Table I show the core factors of computational thinking [8].

Table I. The Core Factors of Computational Thinking

Components	Definitions
Data Collection	The process of gathering appropriate information.
Data Analysis	Making sense of data, finding patterns, and drawing conclusions.
Data Representatio	Depicting and organizing data in appropriate graphs, charts, words, or im
Problem Decomposit	Breaking down tasks into smaller manageable parts.
Abstraction	Reducing complexity to define the main idea.
Algorithms & Proced	Series of ordered steps taken to solve a problem or achieve some end.
Simulation	Representation or model of a process. Simulation also involves running experiments using models.
Parallelization	Organize resources to simultaneously carry out tasks to reach a common

Although different scholars assign slightly different definitions to computational thinking, all the definitions share a common idea given that it is a thinking process of collecting problems and finding solutions to effectively take actions through a computing system. Further, computational thinking can be referred to as a procedural thinking ability for solving problems through abstraction and automation. In recent years, computational thinking has begun to be considered as the ability that every learner of the twenty-first century should acquire in addition to the 3Rs (Read, Record, and Recite). Hence, developing computational thinking has become the main goal of software education [9]. In this study, the main goal was to promote the development of

students' computational thinking through the process of comparing and analyzing the execution time needed to solve a given problem and to find a more efficient algorithm using EPL.

2.5. PREVIOUS STUDIES

A study by Shin (2015) analyzed the impact of Scratch-based education on improving computational thinking. As a result, students' computational thinking improved in areas where students understood and analyzed problems. In addition, it was suggested that further research is needed on which approach of teachers is more helpful for improving computational thinking. Based on this, this study analyzed the impact of Scratch-based education on the performance time on improving students' computational thinking skills [10].

A study by Yoon (2018) analyzed the effects of EPL on programming ability, computational thinking, and problem solving in terms of programming education. In the said study, among various EPLs, Scratch was used to provide programming language education, which helped students improve their programming abilities, computational thinking, and problem-solving skills while achieving a high student satisfaction [11]. However, since the subject of education is university students, it is necessary to study whether the results of this study will be effective in primary education. Thus, this study designed EPL education programs for elementary school and sought to improve students' computational thinking skills.

In a study by Lim (2017), an algorithm-based teaching and learning method was developed to enhance computational thinking. According to Lim, computational thinking skills were improved through the process of solving problems by using algorithm-based teaching and by learning method to solve computer science problems in the information subject [12]. However, this study did not go as far as to apply the teaching method developed to students and analyze the changes in computational thinking pre- and post- education. Thus, the change in computational thinking was measured in this study by applying the program to improve the computational thinking based on algorithms and analyzing the degree of computational thinking on improvement pre- and post-education.

Yang (2019) has developed and applied an educational program that was focused on analyzing the execution time of sorting algorithms for elementary school students. By analyzing the educational effects of that program, he determined that education that featured comparing and analyzing algorithm execution time has provided a positive effect on improving the logical thinking of elementary school students. Further, it was suggested that execution time can be an effective programming education topic to help improve thinking skills [13].

A study by Kim (2018), has develop a Python search algorithm educational program based on execution time to improve the logical thinking of elementary school students. He focused on algorithm education through which students ponder about how to solve a problem in minimal time and its importance, beyond simply solving the problem. The results of the analysis showed that this study can positively affect the logic of elementary school student [14].

Kim (2020) has developed and applied an unplugged educational program that was focused on execution time analysis for third grade elementary school students. Focused educations were given during six days in vacation period and then, a comparative analysis was made between a control group taken a board game education and a group taken an education program developed in this study. The result of the pre and post-tests on computational thinking and creativity showed that the education program developed in this study is effective in enhancing computational thinking ability of elementary school students [5].

As described previously, to promote and enhance computational thinking, which is an essential goal of software education, various studies have been actively conducted in the education field by using the EPLs. In this study, the execution time was selected as a tool for improving the computational thinking of students. To present the concept of algorithm execution time to students, the concept was simplified in accordance with the students' cognitive level. Further, after conducting a demand analysis by reflecting the difficulty of the topic among various EPLs, Scratch, which the students are familiar with, was selected as the education tool. Finally, we verified the effectiveness of the developed educational program on improving the computational thinking of the elementary school students.

3. RESEARCH METHODS

3.1. RESEARCH HYPOTHESIS

Null hypothesis: No difference exists between the learner's pre- and post-computational thinking after participating in algorithm execution time-oriented primary EPL education.

Alternative hypothesis: A difference exists between the learner's pre- and post-computational thinking after participating in algorithm execution time-oriented primary EPL education.

3.2. STUDY SUBJECTS

In this study, 25 volunteers participated for the education donation program conducted by Jeju National University. The overall content of the program, including the subject and content of the program, was released online in advance. The participants, who were minors, gave their guardians' consents to join the study. Table II summarizes the grade and gender of the participants.

Table II. Grade and Gender of the Subject

Class	Male	Female	Total
Grade 4	11	6	17
Grade 5	4	4	8
Total	15	10	25

3.3. EDUCATIONAL PROGRAM

The educational program was developed in accordance with the ADDIE model process of Dick& Carey, which is the most widely used instructional design model. In the analysis stage, the pre-requisite analysis was conducted among the elementary school students, and in the design stage, the educational goals were specified based on the results of the demand analysis. In the development stage, the education was developed based on the materials needed for education and the teaching process plan was developed during the implementation stage, the education was conducted based on the development. In the assessment phase, post-tests were conducted using the evaluation tool, and the Bebras Challenge was used to conduct post-tests and measure changes in students' computational thinking.

3.3.1. PRELIMINARY DEMAND ANALYSIS

The demand analysis was conducted online using Google forms to 48 senior elementary school students in Jeju, Republic of Korea. Table III shows the list of teaching methods that the students mainly experienced in their software education. According to the survey results, physical computing was the most common teaching method, followed by EPL, unplugged, and computer language.

Table III. The Experience of SW Education

	Unplugg	Physica Computi	EPL	Comput Langua
Studen	11(23%	20(42%	16(33%	1(2%)

Table IV shows the results of the survey on whether the students understood the software education concept of algorithm (procedural problem solving). The results show that more than 80% of the students discovered the term algorithm through software education and understood its meaning.

Table IV. Understanding Algorithms in SW Education

	Not well aw	Be averag	Well-inform
Student	9(18.8%)	21(45.8%	17(35.4%

As shown in Table V, 96% of the students answered that algorithm education is essential in software education.

Table V. The Need of Algorithms in SW Education

	Do not nee	be Norma	Essential
Student	4(4%)	19(19%)	77(77%)

The demand analysis results identified the below demands for this study. First, the software education methods that students most often experience are physical computing, EPL, and unplugged. As the students are

rather unfamiliar to the algorithm execution time topic, among the tools that students frequently used, Scratch, which is one of the EPLs and enables convenient comparative analysis of execution time, was selected as the tool for the software education. Second, most students answered that they had previously encountered algorithms (procedural problem solving) through software education and that they understood the concept to some extent. Additionally, the students agreed that learning about algorithm execution time (finding out if the problem-solving method is effective) is necessary. Accordingly, we aimed to improve the computational thinking of students by allowing them, through Scratch, to analyze and compare various algorithms by solving the same problem and finding out which algorithm is the most efficient.

3.3.2. EDUCATIONAL PROGRAM DESIGN AND DEVELOPMENT

In this study, we constructed an execution time-oriented primary EPL education content based on learner levels and Scratch characteristics. The early sessions of the course were focused on the basic content required for comparing and analyzing the algorithm execution time, such as the basic Scratch UI, repetitive statements and conditional statements, and variables. Furthermore, we included separate practice sessions to help students familiarize themselves with the corresponding content. Subsequently, the course' mid-sessions were designed for the students to solve problems using various algorithms, such as a pedometer, calorie calculator, prime number detector, and sieve of Eratosthenes, and to compare each execution time to locate more efficient algorithms. Finally, in the final course sessions, the students were asked to conduct individual projects by selecting their own topics on which they solve a problem by using two or more algorithms and comparing execution times. Table VI summarizes the learning themes of the class sessions.

Table VI. Algorithm Execution Time Analysis Education Program

Hou	Step	Topic
1~	Orientation and foun	- Orientation and pre-test paper input - Learn basic functions and UI of the Scratch - View to create a simple program (to move a sprite, change the shape of sprite)
5~1	Conditional and repe statements Variable and Lis	- Learn the concept of conditional statement and setting of exit con (press the keyboard to move the fish, shut down when sharks touc - Learn how to use variables and lists (create score using variables, find the divisor and a common divis
13~	Learning the conce execution time	- Concept of execution time (compare the two algorithms of finding the maximum of three nur - Learn how to compare execution times in a Scratch program (create a variable to count the number of executions of repetitive statement)
19~	Execution time ana	- To compare and analyze the execution time of two algorithms (obtain the maximum, find the maximum committee, distinguish p number, find a decimal number between 1 and 100, etc.)
31~	Preparing the work individual proje	- Create a work plan of individual project - To produce individual project work (solving one problem with two algorithms and comparing executi with Scratch)
37~	Announce project v	- Post-test paper input - Announce the work of individual projects (comparing analysis of algorithm execution time)

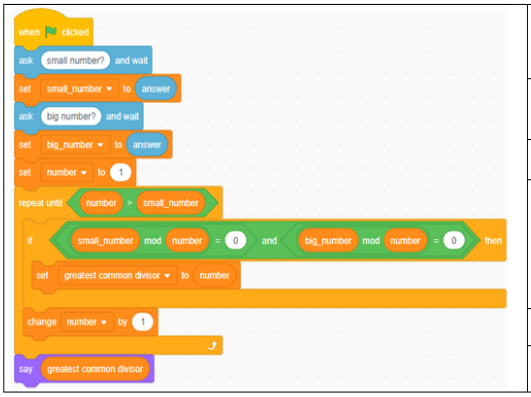
Figure 1 shows part of the textbook on the process of how to calculate the execution time of the algorithm in Scratch. The textbook first introduces the process of obtaining two maximum pledged numbers, creating four variables: 'small number', 'big number' and 'number', 'greatest common divisor'. When 'small' and 'big' are divided into 'number', both find the remaining 0 and store 'number' in greatest common divisor, and repeat 'number' until it is bigger than the small number. Through this course, students can find greatest common divisor, calculating how many operations are performed within the algorithm and calculating the execution time manually. It also guides them to realize that they can calculate the execution time by creating another variable within the scratch called 'execution time' and increasing it by one for each calculation, such as 'number'. In this way, students can create these "execution time" counters within algorithms to compare the efficiency of two or more algorithms.

Execution Time Analysis EPL Program for Computational Thinking

Chapter 9. Finding Greatest Common divisor

1. Greatest Common divisor: the largest of two or more common divisor

2. Find Greatest Common divisor

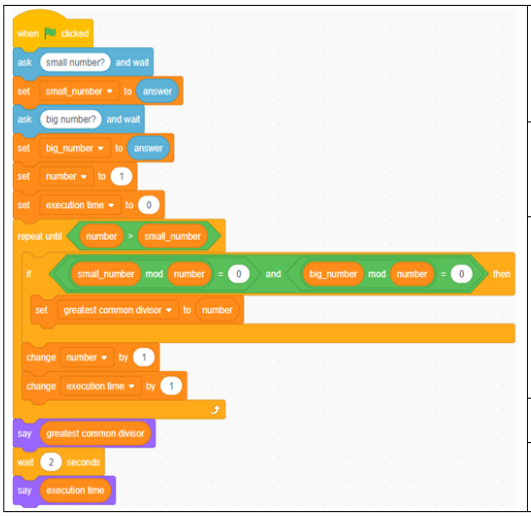


1) Enter a small number and save it as a variable
2) Enter a big number and save it as a variable
3) Store 1 in variable number
4) Divide 'small' and 'big' into 'number' until 'number' is greater than 'small'. If the remainder is 0, store 'number' in 'greatest common divisor'.
5) Make 'number' one bigger.
6) say 'greatest common divisor'

3. What is execution time analysis?

Execution time analysis is the time it takes for an algorithm or program to receive input, output, and shutdown, and the shorter the execution, the shorter the processing time and the more can be solved. You can count the number of times you do the calculation in the repetitive statement.

4. How can I get the execution time in the repetitive statement?

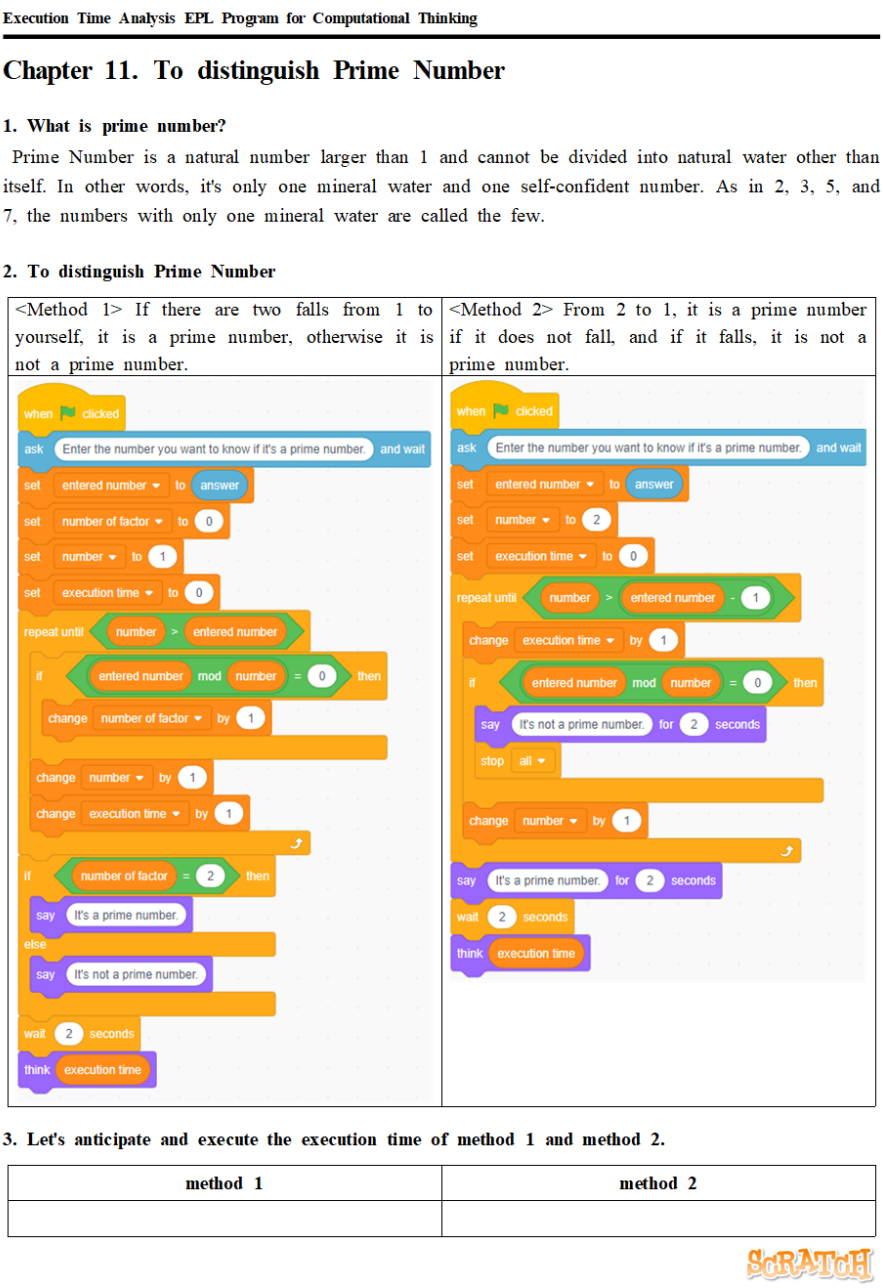


<What to add for 'execution time' anlysis>
1) Make a variable called 'execution time' to store zero.
2) Each calculation of a repetitive statement results in a larger number of times.
3) Say 'execution time'
<Let's think> How does this algorithm change its execution time?

SCRATCH

Fig. 1 Part of a textbook introducing methods for calculating execution time through examples of finding the greatest common divisor

Figure 2 shows part of the textbook that puts execution time counters into two algorithms to solve the same problem to distinguish whether the number is a prime number and to compare the execution time. Method 1 is to divide the number of entries from one to the number of entries, and to see if there is a number of zeros in the remainder. This method repeats the division from 1 to the number entered, so the execution time is the number of entries for both the maximum and maximum values. Method 2 starts with 2, divides by one, but stops immediately when the remaining number is zero and determines that it is not a minority. Therefore, the maximum value of execution time is 1 and the maximum value is input number-2. As the number of inputs increases, method 2's execution time becomes smaller compared to method 1, resulting in a big difference. In this educational program, students tried to measure the execution time of algorithms using these execution time counters and improve whether they could change the algorithm execution time of their own program more efficiently.



Execution Time Analysis EPL Program for Computational Thinking

Chapter 11. To distinguish Prime Number

1. What is prime number?

Prime Number is a natural number larger than 1 and cannot be divided into natural water other than itself. In other words, it's only one mineral water and one self-confident number. As in 2, 3, 5, and 7, the numbers with only one mineral water are called the few.

2. To distinguish Prime Number

<Method 1> If there are two falls from 1 to yourself, it is a prime number, otherwise it is not a prime number.	<Method 2> From 2 to 1, it is a prime number if it does not fall, and if it falls, it is not a prime number.

3. Let's anticipate and execute the execution time of method 1 and method 2.

method 1	method 2

Fig. 2 Part of a textbook comparing execution time with two algorithms that distinguish whether the entered number is a prime number

3.4. SCRATCH

Scratch is a free programming language tool developed by the MIT Media Lab in the US in 2007 as a means of developing the intelligence and creativity of children ages 8–16. The Scratch programming language combines the way children think of algorithms with built-in logical programming called blocks to alleviate the difficulties presented by the existing text input methods for understanding structure and grammar [15]. Scratch can be programmed by dragging and dropping blocks and provides 10 types of block categories including of action, form, sound, pen, data, event, control, observation, computation, and additional block. Each block category has several blocks, and because each category has a different color, the color of the block alone can tell which category it belongs to. Block-based language like Scratch is a suitable programming language for programmers especially among elementary school and middle school students because it is easier to visually simplify, structure, and utilize the structure or contents of a program when written in text-based language [16].

3.5. EXPERIMENT TOOLS

As for the experiment tool, we used the assessment sheets from the Korea Bebras Challenge hosted by the Korea Information Science Education Federation to evaluate the computational thinking improvement. The Bebras Challenge is an assessment tool that measures computational thinking elements, such as algorithm and programming analysis, data analysis and representation, problem analysis, problem decomposition, and modeling based on informatics concepts. For this study, in particular, the Bebras Challenge group III assessment sheets (designed for 5th and 6th graders) were selected based on the age and level of the students participating in the experiment. Additionally, pre- and post-test sheets were used to measure the changes in computational thinking. Table VII shows the experimental design.

Table VII. Experimental Design

	Pre-test	**Treatmen**	**Post-test**
G(N=25)	O_1	X	O_2
X: Input Execution time analysis education program G: Experimental group O_1, O_2: Paired sample T-test(pre-test, post-test)			

4. RESULTS

4.1. EVALUATION OF EDUCATIONAL PROGRAM EFFECTIVENESS

An evaluation was conducted to analyze the effects of execution time-oriented primary EPL education on the computational thinking of students. Since the sample size was 25 students and fell in the range of $10 \leq n < 30$, a normality test was conducted to verify that the computational thinking test results of the experimental group showed normality. The Shapiro-Wilks test was used for the normality test, and the results are shown in Table VIII.

Table VIII. Normality Test(*p<.05)

Subsca	**Descriptive Statistics(N=25)**				**stat**	**p**
	M	**SD**	**Max**	**Min**		
CT	1.240	1.200	3	-1	.920	.282

From the normality test, a p-value of 0.282 was obtained, which was greater than 0.05; thus, the null hypothesis was accepted to assume that the normality was secured. As the computational thinking assessment results were verified to show normality through the Shapiro-Wilks test, a paired t-test was used for comparison of pre- and post-test results.

Table IX. Change in Computational Thinking (Paired T-test, *p<0.5)

Subsca	**N**	**Pre-Test**		**Post-Test**		**t**	**p**
		M	**SD**	**M**	**SD**		
CT	25	5.28	.363	6.32	.298	-3.43	.002

As shown in Table IX, in the paired t-test, the average score increased by 1.08, from pre-test score of 5.28 to post-test score of 6.32, and the p-value of 0.002 was obtained, indicating a statistical significance in the improvement of the computational thinking.

4.2. EXPERIMENT RESULTS ANALYSIS

According to the paired t-test results, a higher computational thinking average score was observed in the post-test, and a statistically significant improvement was obtained in the p-value. Therefore, the experiment in this study successfully managed to prove that the execution time-oriented primary EPL education can improve the

computational thinking ability of elementary school students and that the topic of algorithm execution time can be implemented as a useful software education topic for elementary school students.

5. CONCLUSIONS

In this study, we attempted to analyze the effects of algorithm execution time-oriented primary EPL education on the computational thinking of elementary school students. To achieve this, we developed an educational program based on a preliminary demand analysis by following the ADDIE model process. Subsequently, educational program was provided to students as a 6-day intensive course, and changes in computational thinking were evaluated through the pre- and post-tests results.

Based on these results, the proposed educational program was found to be effective in enhancing the computational thinking ability of the elementary school students. Additionally, students showed high satisfaction with the educational program by trying various approaches in the process of comparing execution times and improving their programming skills. Students continued to think and work hard to reduce their execution time, not just to complete the program. Previously, the program ended with completion, but after studying the execution time through the program, students became interested in reducing the execution time of the program, not just completing the program. Based on these findings, we can conclude that the algorithm execution time comparison and analysis subject is useful in developing the computational thinking of learners, which is the main goal of software education.

However, generalizing this study presents some limitations, as the sample size of the experimental group did not exceed 30 participants, which is the minimum number required for a general correlational study. Additionally, since the effectiveness of the developed educational program was verified only with the pre- and post-test results, without having a comparison group, there is a possibility that the correlation of the program effectiveness may not have been clearly expressed in detail. In a follow-up study, we intend to secure a larger number of participants and to construct experimental and comparison groups to analyze each study result factors more systematically.

ACKNOWLEDGMENTS

This study was supported by the research grant of Jeju National University in 2020.

REFERENCES

JOURNAL ARTICLE

1. Shim, J., Chae, J., “Development of On-line Judge System based on Block Programming Environment,” The Journal of Korean association of computer education, vol. 21, no. 4, pp.1-10, (2018)
2. C. Kurinchi Vanan & R. Subramani, “Digital Divide: Rural and Urban College Students’ Attitude Towards Technology Acceptance”, International Journal of Communication and Media Studies (IJCMS) , Vol. 5, Issue 4, pp, 1-8
3. Lee, A., “Domestic Research Trend Analysis of Computing Thinking”, The Journal of Korea Contents Society, vol. 19, no. 8, pp.214-223, (2019) DOI: 10.5392/jkca.2019.19.08.214
4. Rahma Nasir, “Identifying the Students’ Proportional Reasoning”, International Journal of Educational Science and Research (IJESR), Vol. 8, Issue 2, pp, 71-78
5. Park, J., “Evaluation of Computational Thinking through Code Analysis of Elementary School Students’ Scratch Projects,” Journal of The Korean Association of Information Education, vol. 23, no. 3, pp.207-217, (2019) DOI: 10.14352/jkaie.2019.23.3.207
6. Ali A. AL-Bakhrani, Abdulnaser A. Hagar, Husam H. Abdulmughni, Ahmed A. Hamoud, Bharti W. Gawali, Ratnadeep R. Deshmukh, Ramesh Manza, Manasi Baheti & Sunil Nimbhore, “The Utilization of Google Classroom in the Learning Procedure and Implement in Yemen”, International Journal of Mechanical and Production Engineering Research and Development (IJMPERD), Vol. 10, Issue 3, pp, 1539–1544
7. Jeong, I., “Software Battle for Algorithm Education Focused on Sorting Algorithm”, Journal of The Korean Association of Information Education, vol. 22. no. 2, pp.223-230, (2018), DOI:10.14352/jkaie.2018.22.2.223
8. Garima Sharma, “A Critical Study of the Biology Curriculum at Senior Secondary Stage With Respect to Life Skills Education and the HIV/AIDS Education”, IASET: International Journal of Library & Educational Science (IASET: IJLES), Vol. 2, Issue 3, pp ; 1-10
9. Kim, J., Oh, M., Kim, J., “Effect of analysis of algorithm execution time and adopting unplugged method on third grade elementary students’ computational thinking ability”, The Asian Internatilanl Journal of Life Science, vol. 29, no. 1, pp.269-279, (2020), DOI: 10.21742/IJCWPM.2019.3.1.02

10. Kiran Dalal, “Learning Enhancements Programmes in Schools”, International Journal of Linguistics and Literature (IJLL), Vol. 5, Issue 5,pp; 15-18
11. Jeon, S., “Art based STEAM Education Program using EPL,” Journal of The Korea Society of Computer and Information, vol. 19, no. 4, pp.149-158, (2014) DOI: 10.9708/jksci.2014.19.4.149
12. Dirgha Raj Joshi, “Useful Applications/Software for Mathematics Teaching in School Education”, IMPACT: Journal of Computational Sciences and Information Technology (IMPACT: JCSIT), Vol. 1, Issue 1, pp, 29-34
13. Kim, D., “A Meta-Analysis on the Effects of Software Education on Computational Thinking.” Journal of The Korean Society of Computer Information, vol. 23, no.6, pp.81-89, (2018) DOI: 10.9708/jksci.2018.23.06.081
14. Choi, S., “An Analysis of "Informatics" Curriculum from the Perspective of 21st Century Skills and Computational Thinking,” Journal of The Korean association of computer education, vol. 14, no. 6, pp. 19-30, (2011)
15. Lee, S., “The Effect of Self-regulating Learning-Based Unplugged Activities on Computing Thought in Primary Room and SW Education,” M.S. thesis, Industrial Technology Education, Chungnam National University, Chungnam, Republic of Korea, (2019)
16. Shin, S., “The Improvement Effectiveness of Computational Thinking through Scratch Education”, Journal of The Korea Society of Computer and information, vol. 20, no. 11, pp. 191-197, (2015) DOI:10.9708/jksci.2015.20.11.191
17. Yoon, S., “A Study on the Effect of EPL on Programing, Computing Thinking and Problem Solving Ability of Programing Education,” The Journal of the Convergence on Culture Technology vol. 4, no. 4, pp.287-294, (2018) DOI:10.17703/jcct.2018.4.4.287
18. Lim, S., “Development of Teaching and Learning Methods Based on Algorithms for Improving Computational Thinking,” Journal of The Korean Association of Information Education, vol. 21, no. 6, pp.629-638, (2017) DOI: 10.14352/jkaie.2017.21.6.629
19. Yang, Y., Moon, W., “Effect of Execution Time-oriented Python Sort Algorithm Training on Logical Thinking Ability of Elementary School Students,” Journal of The Korean Association of Information Education, vol. 23, no. 2, pp.107-116, (2019) DOI:10.14352/jkaie.2019.23.2.107
20. Kim, B., Kong, G., Kim, J., “Effect of Search Algorithm Execution Time Analysis Education on Logical Thinking of Elementary School Student”, International Journal of Computer Science and Information Technology for Education, vol. 4, no. 2, pp.9-16, (2019), DOI: 10.21742/ijcsite.2019.4.2.02
21. Moon, W., “Development and Application of STEAM Education Model using Scratch Programming and Sensor Board in Class of Elementary School Students,” Journal of The Korean Association of Information Education, vol. 18, no. 2, pp.213-224, (2014) DOI:10.14352/jkaie.2014.18.2.213
22. Hwang, S., Hwang, Z., “A Meta-Analysis Study on The Effect of Software Education on Computational Thinking”, Asia-pacific Journal of Convergent Research Interchange, vol. 6, no. 10, pp.191-202, (2020), DOI: 10.47116/apjcri.2020.10.15

BOOK

1. Hwang, J., “Computer Internet IT Dictionary” Iljin Publishers, Seoul (2011)

CONFERENCE PROCEEDINGS

2. Moon, W., Kim, J., "Effects of Running time-Oriented EPL Program on Computational Thinking of Elementary School Students", International Journal of Education and Learning (IJEL), ISSN: 2234-8034(Print); 2207-6352(Online), NADIA, (2020), Vol. 9, No. 2, pp. 19-26