OPEN ACCESS

CONTEMPORARY EDUCATIONAL TECHNOLOGY

ISSN: 1309-517X (Online) 2020, 12(1), ep261, https://doi.org/10.30935/cet.685959

Research Article

A Neo-Piagetian Analysis of Algorithmic Thinking Development through the "Sorted" Digital Game

Suparat Chuechote
Naresuan University, Thailand
ORCID: 0000-0003-1862-6426

Artorn Nokkaew
Mahidol University, Thailand
ORCID: 0000-0002-5325-1271

Apichat Phongsasithorn
Nakhonsawan School, Thailand
ORCID: 0000-0002-7369-8775

Parames Laosinchai
Mahidol University, Thailand
ORCID: 0000-0002-8535-9834

Received: 4 Jan 2020 Accepted: 1 Feb 2020

Abstract

Sorting is a fundamental computing concept. As for today, it is taught at the secondary school level. However, this kind of algorithm is an obstacle for some students due to its high level of abstraction. To prevent discouragement as well as to incorporate a fun and challenging algorithmic task, a novel tablet-based digital game, *Sorted*, was created to serve the purpose. This research article embraces the neo-Piagetian framework of cognitive development and provides the theoretical-based explanation of how high school students establish sorting algorithms as a result of the digital gameplay. Twenty-three tenth-grade students, who have no proper knowledge of sorting algorithms, participated voluntarily in this study. They played the game with a multi-level design involving multiple unknowns. To later reflect on their operational reasoning and hence decision-making, the series of game actions were logged for individual empirical data. The sorting algorithm formation can be deduced from the logged sequential actions. They were coded and analyzed according to the neo-Piagetian framework to elicit the students' operational reasoning. The discovery of the relations between actions and schematic reasoning to solve sorting problems suggests the impact of a digital game on algorithmic thinking development, and, in general, the use of a game for self-learning of computing concepts.

Keywords: algorithmic thinking, digital game, neo-Piagetian theory, sorting algorithm

INTRODUCTION

For teaching and learning in the 21st century, digital media and technology have become teachers' assistants in various aspects. Teaching media are available in different forms, for example, web application, video, digital game, mobile application, virtual reality (VR) and augmented reality (AR). They have created an impact on the way teachers teach and perceive education. Although some people are still conservative and do not agree with the substitution of teachers by computers, the use of ICT educational tools have been employed

to increase self-directed knowledge acquisition and the motivation for learning (Fancovicova et al., 2010). For example, a digital game offers a learning environment that does not only incorporates the learning concept but also sparks joy for students. To elevate learning experience, a game can be used to teach history by providing virtual scenes and mimicked historical events according to the game timeline. This virtual game with narrative content helped students learn history as well as encourages students' reflection and discussion (Shiue & Hsu, 2017). Gamification with interactivity enhances game-based learning and triggers the ability of ones to learn specific skills or concepts. However, for an abstract concept, not only can gaming keep students engaged while learning and dealing with difficult content, but it can also dynamically scaffold them.

Gamification becomes handy to many teaching areas, including computer science. To teach a fundamental computer science concept such as sorting, teachers look for tools to facilitate learning and to provide a meaningful learning experience. Nowadays, there are several educational media for sorting problems (Baecker, 1998; Meolic, 2013; Yohannis & Prabowo, 2015). Some apply visualization techniques embedded in animations with explanations to teach sorting methods. Learning with visualization can support the perception of sorting algorithms (Baecker, 1998; Byrne et al., 1999). Visualization for education has evolved from being static, dynamic, interactive, into gamified respectively (Baecker, 1998; Boticki et al., 2013; Yohannis & Prabowo, 2015). On the other hand, the lecture-based approach includes only verbal and static visual explanations. The level of interaction and engagement depends solely on the teachers who lead the lesson. Thus, interactive games and media are designed to fill this gap. The purpose of visualization in games and media is to support learning in several ways, such as capturing concepts, noticing patterns from game actions and feedback, and experiencing success/failure of prospective solutions. Besides cognitive supports, gamified visualization also offers emotional support. Challenges and competitive feelings, as a result of multi-level design, can engage students to keep playing, and, at the same time, learning.

In computer science education, the use of digital games is increasing. Some games provide an interactive programming environment, allowing learners to perform programming. The way learners explore through games suggests autonomous learning. With a series of game actions, where programming syntax is none of players' concern, their thinking and logic in programming are more expressed (Malan & Leitner, 2007). As a digital game offers a learning environment for individuals to construct knowledge, it follows the constructivist theory, which associates experiential learning via active engagement (Butler, 1997). Existing knowledge and meaningful experience will determine how ones deal with new experience and environment and hence develop conceptual understanding.

To comprehend cognitive development, Piaget's original theory explains that mental schemas change when one learns. It describes cognitive development as cumulative developmental stages and is age-related (Piaget, 1964). Therefore, age is a factor in one's comprehension of abstract content knowledge. In contrast with the original Piaget's theory, neo-Piagetian theory instead focuses on the developmental stages of a knowledge domain (Knight & Sutton, 2004; Solaz-Portolés & Sanjosé, 2008; Teague et al., 2013). Knight and Sutton adopted neo-Piagetian theory to analyze how mathematical concepts were formed. In their study, the participants were asked to solve a set of arithmetic problems (Knight & Sutton, 2004). Similar to a mathematical concept, the computer science concept of sorting involves abstraction and systematic thinking, which are inherently the roots of algorithmic thinking development. Therefore, the idea of the neo-Piagetian framework is applicable to the investigation of algorithmic thinking (Lister, 2011).

Since the importance of computing and programming skills have been recognized and the skills have been widely integrated into compulsory curricula in many countries, teachers carefully design lessons with various approaches; for example, unplugged activities, augmented reality application, and several forms of game-based approach (Bell et al., 2012, 2014). Algorithmic thinking consequently becomes essential. It relates the ability to think in steps or sequential rules to problem-solving (Csizmadia et al., 2015). Through computer programming assignments, teachers can observe and assess this skill easily by checking programming codes. However, the codes do not reveal much about how students construct the programs, the boundary between existing knowledge and new knowledge, or any debugging experience they might have. Here comes a digital game, called *Sorted*, that reveals all (Phongsasithorn et al., 2019). It allows the thinking and reasoning of

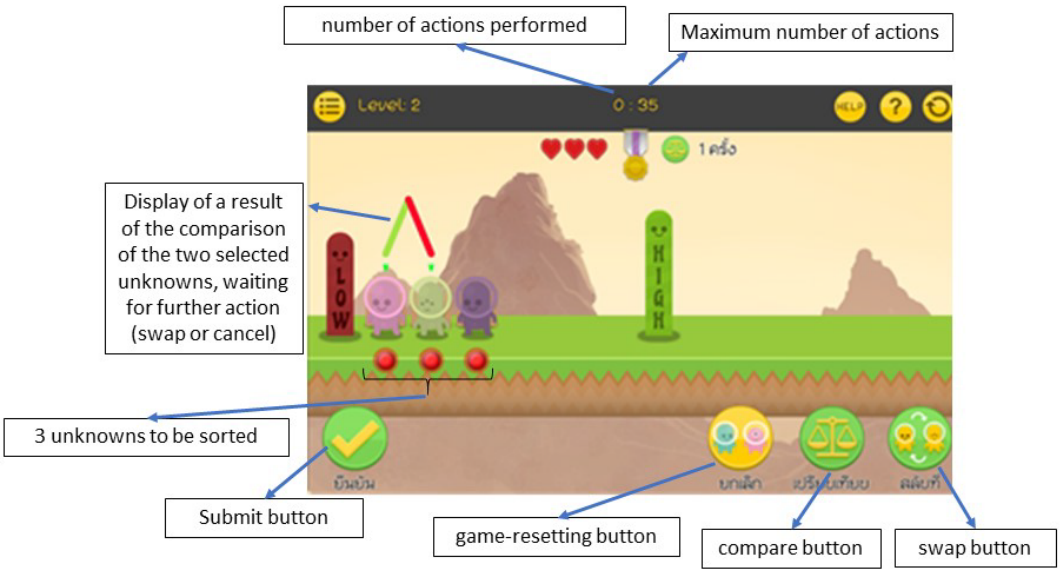


Figure 1. Screen capture of the Sorted Digital Game (Phongsasithorn et al., 2019)

game players to implicitly show during the gameplay through a series of actions to achieve the game's mission.

Centering around an authentic mission-based approach, the *Sorted* digital game has promising game features that facilitate sorting-algorithm development (Phongsasithorn et al., 2019). This study presents the neo-Piagetian analysis toward algorithmic thinking and algorithm formation. The game serves as the purposive learning tool with well-designed components, namely mechanics, dynamics and aesthetics. Through playing, students get to self-learn. As game players proceed to more difficult levels, they get the opportunities to apply their algorithmic thinking schemes to confirm their conceptual construction. It can be a failure or success. However, the game experience would shape how they think for the next step or towards the mission of the game. The series of actions are logged and served as empirical data to be analyzed according to the neo-Piagetian framework. Not only does the analysis shows the effectiveness of the sorting game but it also suggests the feasibility of non-formal autonomous learning through this well-designed game.

THEORETICAL FRAMEWORK

The integral parts of this study are the digital game, called *Sorted*, and the neo-Piagetian framework as a means for the data analysis. Therefore, this section starts with the elaboration of the game features and design, followed by the data analytical framework.

Features and Design of the *Sorted* Digital Game

The game called *Sorted* is an offline single-player digital game (Phongsasithorn et al., 2019). It provides a learning environment that allows and motivates students to construct sorting algorithms. At the start of the game, players receive missions to sort random unknowns, from two unknowns and subsequently more, depending on the difficulty levels (see **Figure 1**). The numeric unknowns are disguised as alien characters-players do not know the values to be sorted. They can only pair two unknowns to be compared or to be swapped. Manipulating the unknowns can be done by using two key actions*, compare* and *swap* actions. For the *compare* action, when the two unknowns are paired, the display shows which one is greater. For the *swap* action, the positions of the two paired unknowns get swapped. Once they are confident that the unknowns are ordered, they can press the *submit* button.

After submission, the game will perform the correctness checking of the sorted list. If it is correct and the number of actions does not exceed the maximum for that level, the players can then proceed to the next level. With a multi-level design, the difficulty of the game increases as the level increases. The higher the

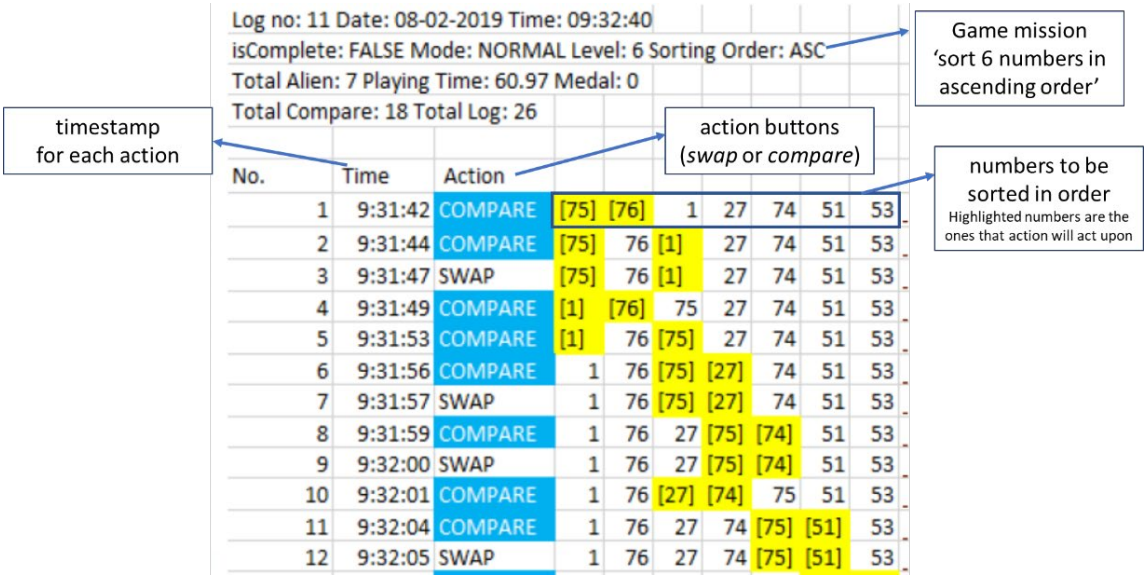
level, the more unknowns are to be sorted. Interestingly, the game will not guide or lead to a certain sorting algorithm. The game only gives two types of feedback, which are the correctness of the submitted list of unknowns and the efficiency of the sorting algorithm-the number of actions used in comparison with the maximum number for each mission. In each level, players have only three chances to submit their sorting. The score of the game depends on the number of actions performed to pass each game level.

Overall, the game design covers the significant features of a purposive game for learning. They are mechanic, dynamic, and aesthetic components according to the MDA framework (Hunicke et al., 2004). The mechanics of the game guide players with rules, obstacles, challenges, and goals. The game dynamics create real-time interaction between a player and the game. Different players generate different dynamics. That would include game feedback corresponding to the players' actions and the difficulty levels that the players experience. The last components, which are aesthetics, serve as both cognitive and emotional supports. The game's aesthetics provide sensation and fantasy to motivate players to play and learn. In addition, the action options and dynamic feedback capture and provoke a player's thought, as a result of objective-oriented visualization.

Neo-Piagetian Analytical Framework

Piaget's work has influenced the education industry significantly. Instructional design has to concern cognitive development. Based on his clinical research, Piaget (1964) explained that children and adults were different in perceptual structures and processes of thoughts. Younger learners and older ones think differently, not just because the latter are brighter. In his theory, the structure of thinking will develop gradually by age. He came up with a view of intellectual development classified into four periods: sensorimotor (birth-2 years), preoperational (2-7 years), concrete operational (7-11 years), formal operational (11 years and above) (Ginsburg & Opper, 1988). Learning is an active process. Learners encompass interpretation, construction, and assimilation to give meaning to the experience. Knowledge must be discovered and constructed through the activities of learners. Although Piaget's theory contributes significantly, some aspects of the classical Piagetian theory have come under criticism. Neo-Piagetian theory was born in response to that criticism in order to preserve, to extend, and yet to alter some aspects of the classical Piagetian. Neo-Piagetian preserves four principles of the classical Piaget theory: (i) children assimilate experiences to existing cognitive structures; (ii) children create their own cognitive structures; (iii) children pass through a universal sequence of structural levels; and (iv) earlier structures are included in later ones (Beilin & Pufall, 2013; Case, 1992). Although the sequence of structural levels and the active roles of learners are the same as those in the classical theory, the development from domain to domain may become a variable (Case, 1992; Case & Sowder, 1990). Due to the capacity of working memory and past knowledge, cognitive behaviors reflect learning stages in the learning domain, in this case the algorithmic thinking to form sorting algorithms. The stages used in the analysis of this study will follow the neo-Piagetian framework that comprises four stages, namely sensorimotor, pre-operational, concrete operational, and formal operational stages.

Based on neo-Piagetian theory, the data analysis for this study aims to investigate how the digital game can provide a learning experience and cause meaningful operational actions. The neo-Piagetian framework sets the deductive thematic analysis for the empirical data retrieved from the game logs. With key achievement for each stage of the neo-Piagetian frame, the investigation uncovers organized patterns of actions and conceptual formation of sorting algorithms. The qualitative data is classified into stages of cognitive development. Behavioral patterns from the game logs identify learning and thinking schemes for the game. According to Piaget's theory, changes in the schemes imply the accommodation process. Therefore, a series of game actions fit well with neo-Piagetian thematic analysis. Operational reasoning hidden in the developmental stage will underlie how a player forms sequential actions to solve sorting problems. Thus, algorithmic thinking can be inferred from the player's actions.



No.	Time	Action							
1	9:31:42	COMPARE	[75]	[76]	1	27	74	51	53
2	9:31:44	COMPARE	[75]	76	[1]	27	74	51	53
3	9:31:47	SWAP	[75]	76	[1]	27	74	51	53
4	9:31:49	COMPARE	[1]	[76]	75	27	74	51	53
5	9:31:53	COMPARE	[1]	76	[75]	27	74	51	53
6	9:31:56	COMPARE	1	76	[75]	[27]	74	51	53
7	9:31:57	SWAP	1	76	[75]	[27]	74	51	53
8	9:31:59	COMPARE	1	76	27	[75]	[74]	51	53
9	9:32:00	SWAP	1	76	27	[75]	[74]	51	53
10	9:32:01	COMPARE	1	76	[27]	[74]	75	51	53
11	9:32:04	COMPARE	1	76	27	74	[75]	[51]	53
12	9:32:05	SWAP	1	76	27	74	[75]	[51]	53

Figure 2. Example of a log file of a game player

METHODOLOGY

Research Design

This study explored how students learned to solve sorting problems by playing the digital game. We aimed to investigate cognitive development in a self-learning situation. The research design responded two research questions.

- RQ1: How did students learn to solve sorting problems through the educational digital game?
- RQ2: How did game actions relate to cognitive development based on neo-Piagetian analysis?

The participants were 23 tenth graders from a public school in Thailand. According to the Thailand Core Curriculum, students are expected to possess computational thinking and capable of designing an algorithm for solving a real problem with computers. Also, they are expected to be mature in learning autonomously.

The implementation process involved game introduction and game playing. The introduction went over the mission in the game, the game's characters and features, how each button worked, how to accumulate scores and proceed to the higher levels of the game. Then the participants got to play freely and individually on the tablets with the game installed and ready to play. During the game playing, the researchers observed and stood by for prompt inquiries about the game-only for game instruction or some helps to fix technical problems. No additional clues and guidance were provided. After finishing the game playing session, which took 60 minutes, there were three open-ended questions for the participants to express: (i) their satisfaction; (ii) their dissatisfaction; and (iii) suggestions for the game. This was to verify that the game had a user-friendly and engaging design and did not cause any unintentional reactions.

Data Collection

This study contained both qualitative and quantitative data. The game actions from the participants were logged by the data-logging system embedded in the *Sorted* digital game. **Figure 2** shows the log data from each device played by each individual player. The data format was composed of five significant components: a log number (for individual identification), actions (*compare*, *swap*, *submit*), timestamps of actions, numeric unknowns, and the highlights of the unknowns assigned to called actions. These were used for thematic data analysis and interpretation of cognitive development based on the neo-Piagetian framework.

Data Analysis

The analysis in the quantitative part covered a number of thinking patterns or action schemes. On the other hand, the qualitative part contained the identification of action schemes and the relations between cognitive developmental stages and actions in the game. This was where the neo-Piagetian framework played a role in the theme setting. Following investigator triangulation, the three researchers read through the log files and looked for action patterns to code and to create themes of schematic thinking. To achieve this, several meetings were conducted to find a consensus in the interpretation of the log files. Then, the categories of themes and schemes were used to explain the acquisition of sorting algorithms through the gameplay.

RESULTS AND DISCUSSION

In this result section, we present a behavioral description of the participants that would lead to the cognitive-developmental stages: sensorimotor, pre-operational, concrete operational, and formal operational stages. We identified changes in the solving schemes to mark the learning. Resulting from data analysis, the schemes were used repeatedly. In this study, we defined that a scheme for solving a sorting problem was an organized pattern of actions. Schemes were extracted when the participants performed them often and purposely. We found multiple schemes that the participants developed and used throughout the game. Relating to algorithmic thinking, schemes were the temporary solving abstraction that the participants came up based on their play experience. We also noticed that some schemes were present more frequently than others.

Scheme Extraction

To investigate algorithmic thinking, we first extracted schemes from the log files. As actions were performed in sequence, dynamic feedback helped the participants to decide what to perform next. This marked the thinking moment and could reflect how the participants thought toward the solution of the game mission. From the series of actions to pattern extraction, we discovered six prominent schemes from the 23 participants.

(a) Extremum Flushing Scheme. **Figure 3a** shows how this scheme works. The pattern of actions for this scheme suggests that the participants performed the actions in one direction to place the highest or lowest

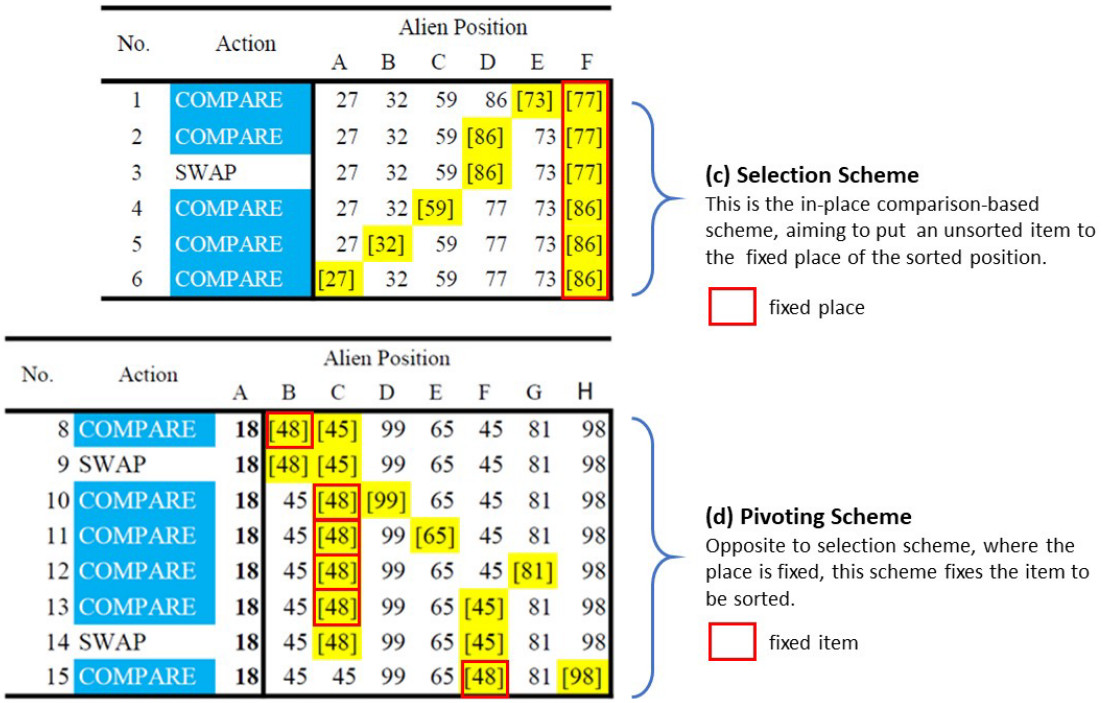
No.	Action	Alien Position A	B	C	D	E	F
1	COMPARE	[11]	[77]	47	36	10	68
2	COMPARE	11	[77]	[47]	36	10	68
3	SWAP	11	[77]	[47]	36	10	68
4	COMPARE	11	47	[77]	[36]	10	68
5	SWAP	11	47	[77]	[36]	10	68
6	COMPARE	11	47	36	[77]	[10]	68
7	SWAP	11	47	36	[77]	[10]	68
8	COMPARE	11	47	36	10	[77]	[68]
9	SWAP	11	47	36	10	[77]	[68]

(a) Extremum Flushing Scheme
Comparing and swapping are performed in order to put maximum or minimum value toward one end.

No.	Action	Alien Position A	B	C	D	E	F	G
13	COMPARE	49	[1]	[76]	90	36	69	95
14	COMPARE	49	1	[76]	[90]	36	69	95
15	COMPARE	49	1	76	[90]	[36]	69	95
16	SWAP	49	1	76	[90]	[36]	69	95
17	COMPARE	49	1	[76]	[36]	90	69	95
18	SWAP	49	1	[76]	[36]	90	69	95
19	COMPARE	49	[1]	[36]	76	90	69	95

(b) Insertion Scheme
Finding the place for unsorted item by comparing the unsorted item to the adjacent item in the sorted list until finding the place to insert.

Figure 3. Example of log files showing (a) Extremum Flushing Scheme and (b) Insertion Scheme



No.	Action	Alien Position A	B	C	D	E	F
1	COMPARE	27	32	59	86	[73]	[77]
2	COMPARE	27	32	59	[86]	73	[77]
3	SWAP	27	32	59	[86]	73	[77]
4	COMPARE	27	32	[59]	77	73	[86]
5	COMPARE	27	[32]	59	77	73	[86]
6	COMPARE	[27]	32	59	77	73	[86]

(c) Selection Scheme
This is the in-place comparison-based scheme, aiming to put an unsorted item to the fixed place of the sorted position.

fixed place

No.	Action	Alien Position A	B	C	D	E	F	G	H
8	COMPARE	18	[48]	[45]	99	65	45	81	98
9	SWAP	18	[48]	[45]	99	65	45	81	98
10	COMPARE	18	45	[48]	[99]	65	45	81	98
11	COMPARE	18	45	[48]	99	[65]	45	81	98
12	COMPARE	18	45	[48]	99	65	45	[81]	98
13	COMPARE	18	45	[48]	99	65	[45]	81	98
14	SWAP	18	45	[48]	99	65	[45]	81	98
15	COMPARE	18	45	45	99	65	[48]	81	[98]

(d) Pivoting Scheme
Opposite to selection scheme, where the place is fixed, this scheme fixes the item to be sorted.

fixed item

Figure 4. Example of log files showing (c) Selection Scheme and (d) Pivoting Scheme

value at one end. As in **Figure 3a**, this participant wanted to achieve a sorted list in ascending order. He had his window pair of 2 unknowns and moved this window one step to the right. Before each move, he compared or swapped the unknowns to make the current maximum stay rightward.

(b) Insertion Scheme. As the name suggested, this scheme aims to insert the unsorted item in the sorted list. In **Figure 3b**, 36 is an unsorted item and the sorted list is the list of 1, 76, 90. The participant compared 36 to adjacent numbers in the sorted list until finding the place for 36. After insertion, the sorted list became larger with 1, 36, 76, 90.

(c) Selection Scheme. This scheme starts with selecting the place and looks for the right item for that place. In **Figure 4c**, the comparison in the six-item frame, the first position was fixed. Thus, this participant looked for the greatest number among 27, 32, 59, 86, 73, 77 to be put in the last position.

(d) Pivoting Scheme. Opposite to the selection scheme, this scheme works with a fixed item or a pivot and looks for its correct place. In **Figure 4d**, the participant looked for the place for the fixed item, 48.

(e) Shell Scheme. This scheme works on the shells of the *n*-item comparison frames. The outer items of a frame get compared, then the *n*-item frame slides to the next frame. In the case of the three-item frame, each item got to be the middle of the frame, as in **Figure 5e**.

(f) Subgroup Scheme. This scheme starts by dividing the comparison frame into mutually exclusive two-item subgroups. As in **Figure 5f**, each subgroup got sorted, followed by a comparison of items from different groups.

No.	Action	Alien Position					
		A	B	C	D	E	F
23	COMPARE	34	44	61	[46]	70	[90]
24	COMPARE	34	44	[61]	46	[70]	90
25	COMPARE	34	[44]	61	[46]	70	90
26	COMPARE	[34]	44	[61]	46	70	90

(e) Shell Scheme
This scheme acts upon 3-item frame, whose shell defines the outer items.

No.	Action	Alien Position			
		A	B	C	D
1	COMPARE	16	2	[92]	[59]
2	SWAP	16	2	[92]	[59]
3	COMPARE	[16]	[2]	59	92
4	SWAP	[16]	[2]	59	92
5	COMPARE	2	[16]	[59]	92

(f) Subgroup Scheme
In the comparison frame, subgroups are formed and sorted first.

Figure 5. Example of log files showing (e) Shell Scheme and (f) Subgroup Scheme

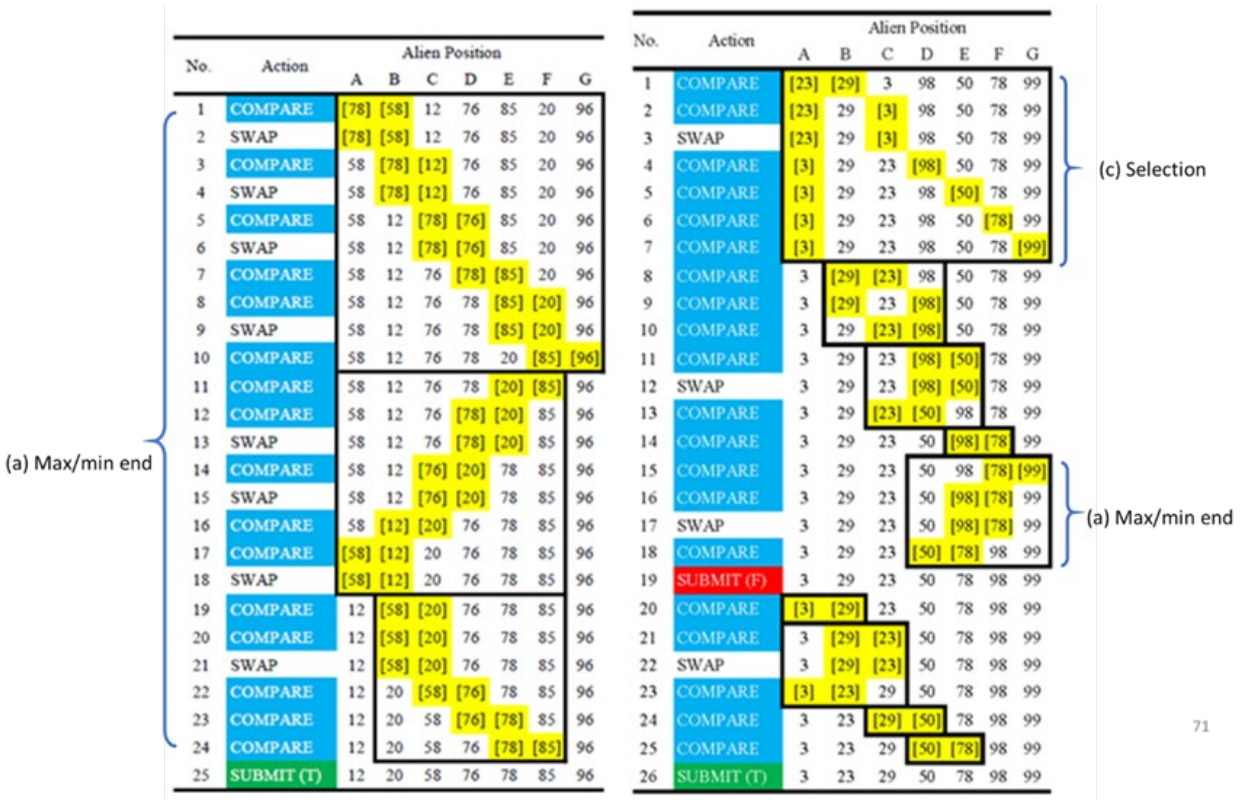


Figure 6. (left) Extremum flushing scheme–based algorithm; (right) Mixed scheme–based algorithm

The schemes provided the fundamental elements for some students to develop successful sorting algorithms as shown in **Figure 6**. The left figure shows the serial of only the extremum flushing schemes to solve the sorting of seven unknowns, whereas the right figure combines the extremum flushing scheme and the selection scheme to handle the similar sorting problem.

Neo-Piagetian Analysis

The neo-Piagetian mental model was employed to investigate algorithmic thinking developed through the digital game playing. In the scheme discovery process, we observed that several participants changed their schemes when their submission failed. This infers how experience entails algorithm formation. The series of

Unreasonable actions

No.	Action	Alien Position A	B	C	D
1	SWAP	[43]	37	34	[92]
2	SWAP	92	37	[34]	[43]
3	COMPARE	92	37	[43]	[34]
4	SWAP	92	37	[43]	[34]
5	SUBMIT (F)	92	37	34	43
6	COMPARE	92	[37]	[34]	43
7	SWAP	92	[37]	[34]	43
8	SUBMIT (F)	92	34	37	43
9	SUBMIT (F)	92	34	37	43

No.	Action	Alien Position A	B	C
1	COMPARE	68	[13]	[21]
2	COMPARE	68	[13]	[21]
3	COMPARE	[68]	[13]	21
4	COMPARE	[68]	13	[21]
5	SWAP	[68]	13	[21]
6	COMPARE	21	[13]	[68]
7	SUBMIT (F)	21	13	68

Uncertain purpose

Figure 7. (left) Example of unreasonable actions and (right) actions with an uncertain purpose

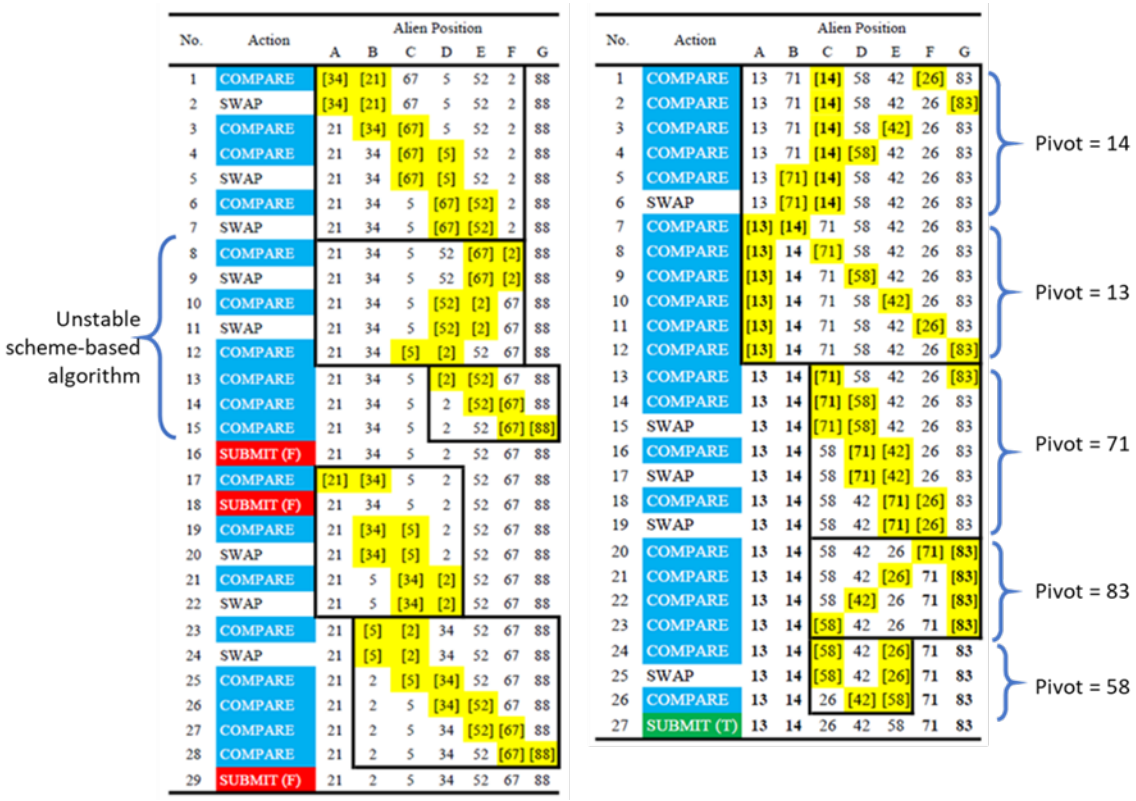
actions in each individual log gave the qualitative data form that could reflect how one managed to pass the game mission, how actions were related to schemes for sorting, and how they were related to cognitive development. Therefore, this result section describes actions and related behaviors corresponding to stages of cognitive development.

1) Sensorimotor Stage. This is the first stage of neo-Piagetian theory. As the digital game was introduced, the participants took some time to explore as it was new to them. They interacted with the available buttons with curiosity. They clicked on the unknowns to test their selected actions. After a couple of playtimes, some got used to this new environment, but some others could not get passed and struggled as we observed that the actions were placed randomly. Associated with the neo-Piagetian frame, this stage represented the on-boarding phase, where the participants tried to get familiar with the game mechanism. Hence, there were neither patterns of actions nor schematic moves.

2) Pre-operational Stage. The pre-operational stage occurs when learners have adapted to the environment and show their readiness. In the game, the participants started to use the given actions to sort the unknowns. However, the actions were unorganized. Some actions contradicted others. Some actions were unreasonably performed, for example, swapping without comparison, see **Figure 7** (left), lines 1-2. **Figure 7** (right) shows actions with an uncertain purpose. Working memory and past knowledge affect cognitive behaviors as described in neo-Piagetian theory. The participants focused on what was in hand, instead of the whole picture. Therefore, they used up all actions to complete the mission or failed to sort the unknowns. The distinct behaviors in this stage were unidirectional comparing and swapping, without a definite purpose. As for the sensorimotor stage, we could not identify any schemes of sorting.

3) Concrete Operational Stage. At this stage, there are two noticeable phases, the scheme-emerging phase and the algorithm-emerging phase. Scheme emergence occurs in the early phase of this stage, where schemes of sorting are expected to be found. The six prominent schemes were identified: (a) Extremum flushing, (b) Insertion, (c) Selection, (d) Pivoting, (e) Shell, and (f) Subgroup schemes, as described in the former section. However, the schemes were not proof of the participants' ability to form successful sorting algorithms. The schemes emerged singly and unrelatedly. Some schemes were used repeatedly. Some were used in combinations with other schemes. By chance, some schematic combinations resulted in a successful sort. However, some combinations did not work for general sorting problems with more numbers to be managed. Some schemes appeared for a short time and disappeared. This could mark the lesson learned. In this stage, trials and errors were dominant and were to shape the learning experience of the participants.

The emergence of scheme mixing was a remarkable and promising cognitive growth in this stage. The repeatable and viable combinations occurred because the participants succeeded and learned. Therefore, the successful repeatable schematic combinations, so-called *emerging algorithms*, were mostly made from the emerging schemes. However, the algorithms formed in this stage were unstable. They might work well just for small sets of unknowns. The players later encountered a failure when applying them for larger sets of sorting problems. When the participants were overwhelmed with a lot of data, they could not keep the



No.	Action	Alien Position A	B	C	D	E	F	G
1	COMPARE	[34]	[21]	67	5	52	2	88
2	SWAP	[34]	[21]	67	5	52	2	88
3	COMPARE	21	[34]	[67]	5	52	2	88
4	COMPARE	21	34	[67]	[5]	52	2	88
5	SWAP	21	34	[67]	[5]	52	2	88
6	COMPARE	21	34	5	[67]	[52]	2	88
7	SWAP	21	34	5	[67]	[52]	2	88
8	COMPARE	21	34	5	52	[67]	[2]	88
9	SWAP	21	34	5	52	[67]	[2]	88
10	COMPARE	21	34	5	[52]	[2]	67	88
11	SWAP	21	34	5	[52]	[2]	67	88
12	COMPARE	21	34	[5]	[2]	52	67	88
13	COMPARE	21	34	5	[2]	[52]	67	88
14	COMPARE	21	34	5	2	[52]	[67]	88
15	COMPARE	21	34	5	2	52	[67]	[88]
16	SUBMIT (F)	21	34	5	2	52	67	88
17	COMPARE	[21]	[34]	5	2	52	67	88
18	SUBMIT (F)	21	34	5	2	52	67	88
19	COMPARE	21	[34]	[5]	2	52	67	88
20	SWAP	21	[34]	[5]	2	52	67	88
21	COMPARE	21	5	[34]	[2]	52	67	88
22	SWAP	21	5	[34]	[2]	52	67	88
23	COMPARE	21	[5]	[2]	34	52	67	88
24	SWAP	21	[5]	[2]	34	52	67	88
25	COMPARE	21	2	[5]	[34]	52	67	88
26	COMPARE	21	2	5	[34]	[52]	67	88
27	COMPARE	21	2	5	34	[52]	[67]	88
28	COMPARE	21	2	5	34	52	[67]	[88]
29	SUBMIT (F)	21	2	5	34	52	67	88

No.	Action	Alien Position A	B	C	D	E	F	G
1	COMPARE	13	71	[14]	58	42	[26]	83
2	COMPARE	13	71	[14]	58	42	26	[83]
3	COMPARE	13	71	[14]	58	[42]	26	83
4	COMPARE	13	71	[14]	[58]	42	26	83
5	COMPARE	13	[71]	[14]	58	42	26	83
6	SWAP	13	[71]	[14]	58	42	26	83
7	COMPARE	[13]	[14]	71	58	42	26	83
8	COMPARE	[13]	14	[71]	58	42	26	83
9	COMPARE	[13]	14	71	[58]	42	26	83
10	COMPARE	[13]	14	71	58	[42]	26	83
11	COMPARE	[13]	14	71	58	42	[26]	83
12	COMPARE	[13]	14	71	58	42	26	[83]
13	COMPARE	13	14	[71]	58	42	26	[83]
14	COMPARE	13	14	[71]	[58]	42	26	83
15	SWAP	13	14	[71]	[58]	42	26	83
16	COMPARE	13	14	58	[71]	[42]	26	83
17	SWAP	13	14	58	[71]	[42]	26	83
18	COMPARE	13	14	58	42	[71]	[26]	83
19	SWAP	13	14	58	42	[71]	[26]	83
20	COMPARE	13	14	58	42	26	[71]	[83]
21	COMPARE	13	14	58	42	[26]	71	[83]
22	COMPARE	13	14	58	[42]	26	71	[83]
23	COMPARE	13	14	[58]	42	26	71	[83]
24	COMPARE	13	14	[58]	42	[26]	71	83
25	SWAP	13	14	[58]	42	[26]	71	83
26	COMPARE	13	14	26	[42]	[58]	71	83
27	SUBMIT (T)	13	14	26	42	58	71	83

Figure 8. Examples of emerging algorithms; (left) unstable scheme-based algorithm, (right) Pivoting scheme–based algorithm

combination of schemes consistent, see **Figure 8** (left). Thus, in this stage, although the patterns of actions were discovered, some were not successful. Interestingly, in the algorithm-emerging phase, some schemes, like the *Shell* and *Subgroup* schemes, disappeared because the participants could not use them to construct a viable algorithm. In **Figure 8** (right), a *Pivoting* scheme–based algorithm was found and produced a successful sorting result. However, it was counted as a concrete operational stage because it could not be applied to a larger problem or a different set of data. Another remark for this phase was that the players, though discovered the algorithms, were unconfident about them. Their logged actions included solution checking, back comparison, or unnecessary comparison. According to the neo-Piagetian analysis, we found the concrete, prospective algorithms in this concrete operational stage, where logics played a role in decision-making.

4) *Formal Operational Stage.* Formal operational is the most advanced developmental stage. In this stage, fully systematic sorting algorithms were found. The thinking expressed through confident actions was consistent and efficient. One common behavior was that the comparison checking was not needed before submitting. The constructed algorithms could be applied to a large number of unknowns or in high-level tasks. Based on our empirical data, only two types of formal algorithms were found: the *Insertion* scheme–based type and the *Selection* scheme–based type. These algorithms were found in the participants' log data while working with the 21-unknown sorting problem. Note that the most advanced participant stopped because of time limitation. He used his solving method consistently and efficiently, and successfully submitted with confidence; without checking before submitting. This could confirm that the constructed algorithms were found to be applicable to any sorting problems in a similar context. Hence, the participant was in the formal operational stage, where formal operation allowed the participant to deal with a load of data in a sorting task with better working memory management. The examples of formal algorithms are shown in **Figure 9**. Note that the formal algorithms constructed through the game are similar to the sorting algorithms taught in the standard curriculum, namely insertion sort and selection sort (Bell et al., 2012, 2014).

No.	Action	Alien Position							
		A	B	C	D	E	F	G	H
1	COMPARE	[46]	[8]	45	40	64	90	64	70
2	SWAP	[46]	[8]	45	40	64	90	64	70
3	COMPARE	8	[46]	[45]	40	64	90	64	70
4	SWAP	8	[46]	[45]	40	64	90	64	70
5	COMPARE	[8]	[45]	46	40	64	90	64	70
6	COMPARE	8	45	[46]	[40]	64	90	64	70
7	SWAP	8	45	[46]	[40]	64	90	64	70
8	COMPARE	8	[45]	[40]	46	64	90	64	70
9	SWAP	8	[45]	[40]	46	64	90	64	70
10	COMPARE	[8]	[40]	45	46	64	90	64	70
11	COMPARE	8	40	[45]	[46]	64	90	64	70
12	COMPARE	8	40	45	[46]	[64]	90	64	70
13	COMPARE	8	40	45	46	[64]	[90]	64	70
14	COMPARE	8	40	45	46	64	[90]	[64]	70
15	SWAP	8	40	45	46	64	[90]	[64]	70
16	COMPARE	8	40	45	46	[64]	[64]	90	70
17	COMPARE	8	40	45	[46]	64	[64]	90	70
18	COMPARE	8	40	45	46	64	[64]	[90]	70
19	COMPARE	8	40	45	46	64	64	[90]	[70]
20	SWAP	8	40	45	46	64	64	[90]	[70]
21	COMPARE	8	40	45	46	64	[64]	[70]	90
22	SUBMIT (T)	8	40	45	46	64	64	70	90

No.	Action	Alien Position										
		A	B	C	D	E	F	G	H	I	J	K
43	COMPARE	3	27	49	50	[71]	99	56	66	72	77	[82]
44	COMPARE	3	27	49	50	[71]	99	56	66	72	[77]	82
45	COMPARE	3	27	49	50	[71]	99	56	66	[72]	77	82
46	COMPARE	3	27	49	50	[71]	99	56	[66]	72	77	82
47	SWAP	3	27	49	50	[71]	99	56	[66]	72	77	82
48	COMPARE	3	27	49	50	[66]	99	[56]	71	72	77	82
49	SWAP	3	27	49	50	[66]	99	[56]	71	72	77	82
50	COMPARE	3	27	49	50	[56]	[99]	66	71	72	77	82
51	COMPARE	3	27	49	50	56	[99]	66	71	72	77	[82]
52	SWAP	3	27	49	50	56	[99]	66	71	72	77	[82]
53	COMPARE	3	27	49	50	56	[82]	66	71	72	[77]	99
54	SWAP	3	27	49	50	56	[82]	66	71	72	[77]	99
55	COMPARE	3	27	49	50	56	[77]	66	71	[72]	82	99
56	SWAP	3	27	49	50	56	[77]	66	71	[72]	82	99
57	COMPARE	3	27	49	50	56	[72]	66	[71]	77	82	99
58	SWAP	3	27	49	50	56	[72]	66	[71]	77	82	99
59	COMPARE	3	27	49	50	56	[71]	[66]	72	77	82	99
60	SWAP	3	27	49	50	56	[71]	[66]	72	77	82	99
61	COMPARE	3	27	49	50	56	66	[71]	72	77	82	[99]
62	COMPARE	3	27	49	50	56	66	[71]	72	77	[82]	99
63	COMPARE	3	27	49	50	56	66	[71]	72	[77]	82	99
64	COMPARE	3	27	49	50	56	66	[71]	[72]	77	82	99
65	COMPARE	3	27	49	50	56	66	71	[72]	77	82	[99]
66	COMPARE	3	27	49	50	56	66	71	[72]	77	[82]	99
67	COMPARE	3	27	49	50	56	66	71	[72]	[77]	82	99
68	COMPARE	3	27	49	50	56	66	71	72	[77]	82	[99]
69	COMPARE	3	27	49	50	56	66	71	72	[77]	[82]	99
70	COMPARE	3	27	49	50	56	66	71	72	77	[82]	[99]
71	SUBMIT (T)	3	27	49	50	56	66	71	72	77	82	99

Figure 9. Examples of formal algorithms; Insertion (left) and Selection (right) scheme–based algorithms

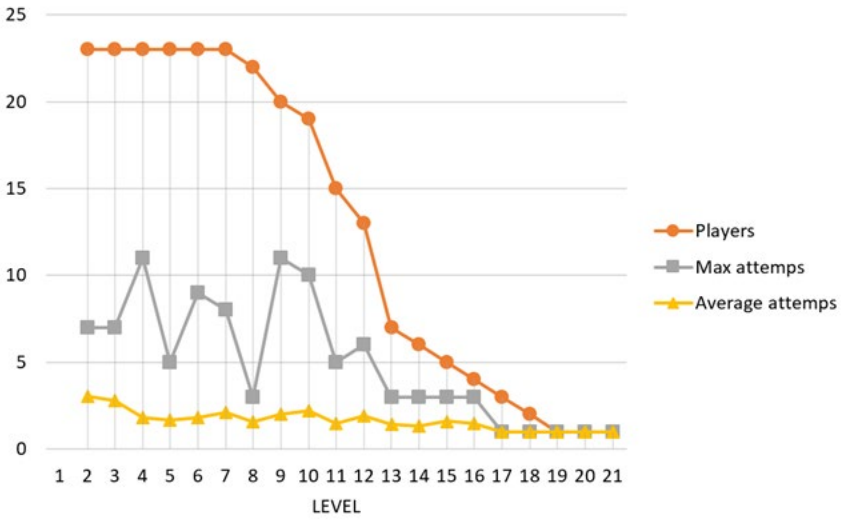


Figure 10. Playing attempts for each level (of data collected within the 60-minute play)

Gameplay-Data Summary

The digital gameplay created empirical data that led to cognitive development in the sorting domain. This section presents the summary of quantitative data and introspection related to the qualitative analysis. We got 23 participants boarded the game. Level 2 was the starting own-play level, where two unknowns were given to be sorted. This was a low-level problem, requiring only one compare action and one swap action, if necessary, to get the unknown sorted. If the first submission was wrong, swapped and submitted again. Therefore, only two attempts or submissions would suffice for this mission. However, there was a participant who submitted the wrong sorting seven times (see **Figure 10**). This shows the on-boarding phase still

persisted and that participant was still in the *sensorimotor* stage. Some participants could not go beyond level 7 within 60 minutes, as we shall see in **Figure 10**, where the number of participants (students) started to decrease from 23 after level 7. As the level went up, the number of participants decreased. Noticeably, the number of maximum submissions decreased significantly after level 10. From level 11 onward, the average number of submissions was within two. This was because the participants who could play fast with efficient solutions were the participants who developed their cognition in the sorting domain to the concrete operational stage or the formal operational stage. Therefore, they had their working schemes and more tendency to submit the right answers quickly without spending too many attempts. However, within the playtime, only one participant could achieve more-than-18-unknown sorting problems.

From both quantitative and qualitative empirical data, algorithmic thinking development involved many stages of cognitive development. The time spent in each stage varied, depending on the learning experience and working memory as stated in Neo-Piagetian theory (Case, 1992; Case & Sowder, 1990; Solaz-Portolés & Sanjosé, 2008). All participants developed schemes. Some could develop only one scheme, but some came up and used six different schemes. The most common scheme was the *Extremum flushing* scheme, used by 22 participants. However, four participants could not construct algorithms in the concrete operational stage. Not surprisingly, this agrees with the quantitative data in **Figure 10**, where, at level 10, only 19 participants who had working algorithms could achieve. The interesting point here is that the schemes that can be extended to form algorithms are *Extremum flushing, Insertion, Selection,* and *Pivoting* schemes. The four participants who could not construct algorithms were found to struggle with non-viable schemes or multi-directional sorting patterns. The most common algorithms found in the emerging-algorithm phase of the concrete operational stage were the *Insertion* scheme–based algorithms. However, the development of formal algorithms was first found at level 6. Six participants (26%) found the formal sorting algorithms, i.e., their cognitive development was in the formal operational stage. Note that this happened within the 60-minute gameplay only. In the most advanced stage, three of them discovered the *Selection* scheme–based algorithms while three others discovered the *Insertion* scheme–based algorithms. Surprisingly, one participant discovered the formal *Insertion* algorithm without showing a trace of the *Insertion* scheme. His abstract thinking allowed him to solve the problems without trying.

CONCLUSION

The algorithmic thinking analysis is what we have demonstrated throughout this article. To respond to the research questions, we investigated the logged data, which spoke to us that all participants developed sorting schemes through the gameplay. Some schemes only worked for low-level problems, but some could be applied to any levels. The scheme-changing events confirmed the learning process of the participants. The game truly provided the learning experience, where we could attach the meanings to the changes in thinking schemes. The temporary sorting schemes were developed and got tested in the next higher-level tasks. Some of them failed once and disappeared from the series of actions. Some were tested several times before being ditched. Based on the process of learning, the thinking schemes often used in different problems marked the assimilation process in learning. On the other hand, the scheme changing due to the wrong solution marked the event of accommodation. From this learning experience, we observed the evolution of complete algorithms, starting from trials and errors to fractional schemes, to working schemes, to algorithmic schematic combinations, and to complete algorithms. The game provided sorting experience whereby the participants themselves learned at their own pace. The algorithm formation without traces of relating schemes was a rare phenomenon. Scaffolding was hidden in the game elements. The distinct characteristics of unknowns, displayed as alien characters, helped memorization. The limitation of actions, only swap, compare, or submit, created a thinking frame, which was geared to systematic thinking for scheme organization. The instant feedback after submission formed the concrete learning experience. Fast learners showed the ability to sort large numbers of unknowns. However, we could not conclude that the slower learners would not be able to solve higher-level problems.

The formal algorithms allowed the participants to deal with a large number of unknowns in sorting problems. It was evident that the ability to manage working memory got improved as the play went on. This promising

result of algorithmic thinking development is in accordance with the previous study on the educational game. It supports the claim that a game that offers information organization and logics has high potential to develop algorithmic thinking (Yildiz et al., 2017). Here, the *Sorted* digital game that contains these two elements has successfully shown the indication of algorithm formation.

To integrate technology for effective learning, the engagement is as important as the subject content. The products that can provide high engagement have more tendency to see achievement (Harris et al., 2016). As we would like to show constructive cognitive growth through the digital game, we believe that autonomous learning can be effective and sustainable. From the open-ended questions asked after the game period, 20 participants found themselves absorbed in the gameplay and mentioned that the time spent in the game was just slipped away while 21 participants mentioned that the game was inviting. Most of them were satisfied with the game design and challenging mechanism, whereas only three participants mentioned that they got frustrated by the incremental number of unknowns. The dissatisfactory areas and suggestions were mostly related to the number of unknowns and the lack of sound effects for the game. The game design overall was friendly for learners to practice or develop sorting schemes at a satisfactory level. This led to an enjoyable and deliberate learning experience as they felt like time went by quickly while playing the game. With the utilization of effective engagement and autonomous learning (playing), we have a promising approach to handle the difficult abstraction for sorting problems.

The neo-Piagetian theory has enlightened us about the cognitive differences of individuals. The learning experience in combination with the management of working memory is the major factor in cognitive development. Based on the digital game context, the empirical data here reveals a variety of cognitive development in sorting problems. In addition, it suggests a constructive instructional design for teachers who teach sorting algorithms. Teachers could adapt the game to be unplugged activities or raise a challenging mission in class. Thinking development of students could be classified into four different thinking abilities: ones who are not on board yet; ones who generate scattered schemes without logical thought; ones who can combine schemes to form an algorithm but struggle with abstract level thought; and ones who are able to think in abstraction without trying. Thus, teachers should be aware of cognitive differences in algorithmic thinking of students. Even in a technology integrated class, the past research also recommends teachers to concern with individual profiles of students and suggests the handson technology-integrated approach that demonstrates theoretical concepts that students can explore and test by themselves (Almeida & Simoes, 2019). Hence, teachers should facilitate and scaffold learning according to individual thinking abilities.

ACKNOWLEDGEMENT

This research was partially supported by The Institute for the Promotion of Teaching Science and Technology (IPST) and the Thailand Research Fund (TRG5880196).

REFERENCES

Almeida, F., & Simoes, J. (2019). The Role of Serious Games, Gamification and Industry 4.0 Tools in the Education 4.0 Paradigm. *Contemporary Educational Technology, 10*(2), 120-136. https://doi.org/10.30935/cet.554469a

Baecker, R. (1998). Sorting out sorting: A case study of software visualization for teaching computer science. *Software visualization: Programming as a multimedia experience, 1*, 369381.

Beilin, H., & Pufall, P. B. (2013). P*iaget's theory: prospects and possibilities.* Psychology Press.

Bell, T., Duncan, C., Jarman, S., & Newton, H. (2014). Presenting computer science concepts to high school students. *Olympiads in Informatics, 8*, 3-19.

Bell, T., Newton, H., Andreae, P., & Robins, A. (2012). The introduction of computer science to NZ high schools: an analysis of student work. Paper presented at the *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*.

Boticki, I., Barisic, A., Martin, S., & Drljevic, N. (2013). Teaching and learning computer science sorting algorithms with mobile devices: A case study. *Computer Applications in Engineering Education, 21*(S1), E41-E50. https://doi.org/10.1002/cae.21561

Butler, J. (1997). How would Socrates teach games? A constructivist approach. *Journal of Physical Education, Recreation & Dance, 68*(9), 42-47. https://doi.org/10.1080/07303084.1997.10605029

Byrne, M. D., Catrambone, R., & Stasko, J. T. (1999). Evaluating animations as student aids in learning computer algorithms. *Computers & education, 33*(4), 253-278. https://doi.org/10.1016/S0360-1315(99)00023-8

Case, R. (1992). Neo-Piagetian theories of child development. In R. J. Sternberg & C. A. Berg (Eds.), *Intellectual development* (p. 161-196). Cambridge, UK: Cambridge University Press. Retrieved from https://assets.cambridge.org/97805213/97698/toc/9780521397698_toc.pdf

Case, R., & Sowder, J. T. (1990). The development of computational estimation: A neo-Piagetian analysis. *Cognition and Instruction, 7*(2), 79-104. https://doi.org/10.1207/s1532690xci0702_1

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). *Computational thinking-A guide for teachers*. https://eprints.soton.ac.uk/424545/

Fancovicova, J., Prokop, P., & Usak, M. (2010). Web-Site as an educational tool in biology education: a case of nutrition issue. *Educational Sciences: Theory and Practice, 10*(2), 907-921.

Ginsburg, H. P., & Opper, S. (1988). *Piaget's theory of intellectual development*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Harris, J. L., Al-Bataineh, M. T., & Al-Bataineh, A. (2016). One to One Technology and its Effect on Student Academic Achievement and Motivation. *Contemporary Educational Technology, 7*(4), 368-381.

Hunicke, R., LeBlanc, M., & Zubek, R. (2004). MDA: A formal approach to game design and game research. Paper presented at the *Proceedings of the AAAI Workshop on Challenges in Game AI*.

Knight, C. C., & Sutton, R. E. (2004). Neo-Piagetian theory and research: Enhancing pedagogical practice for educators of adults. *London Review of Education, 2*(1), 47-60. https://doi.org/10.1080/1474846042000177474

Lister, R. (2011). Concrete and other neo-Piagetian forms of reasoning in the novice programmer. Paper presented at the *Proceedings of the Thirteenth Australasian Computing Education Conference*-Volume 114.

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM Sigcse Bulletin, 39*(1), 223-227. https://doi.org/10.1145/1227310.1227388

Meolic, R. (2013). Demonstration of sorting algorithms on mobile platforms. Paper presented at the *CSEDU*.

Phongsasithorn, A., Laosinchai, P., & Nokkaew, A. (2019). Sorted: An educational digital game for learning sorting algorithms. Paper presented at the *International Symposium on Education and Psychology*.

Piaget, J. (1964). Part I: Cognitive development in children: Piaget development and learning. *Journal of Research in Science Teaching, 2*(3), 176-186.

Shiue, Y., & Hsu, Y. (2017). Understanding factors that affecting continuance usage intention of game-based learning in the context of collaborative learning. *Eurasia Journal of Mathematics, Science and Technology Education, 13*(10), 6445-6455. https://doi.org/10.12973/ejmste/77949

Solaz-Portolés, J. J., & Sanjosé, V. (2008). Piagetian and Neo-Piagetian variables in science problem solving: directions for practice. *Ciências & Cognição, 13*(2), 192-200.

Teague, D., Corney, M., Ahadi, A., & Lister, R. (2013). A qualitative think aloud study of the early neo-piagetian stages of reasoning in novice programmers. Paper presented at the *Proceedings of the Fifteenth Australasian Computing Education Conference*-Volume 136.

Yildiz, H. D., Yilmaz, F. G. K., & Yilmaz, R. (2017). Examining the relationship between digital game preferences and computational thinking skills. *Contemporary Educational Technology*, *8*(3), 359-369.

Yohannis, A., & Prabowo, Y. (2015). Sort attack: Visualization and gamification of sorting algorithm learning. Paper presented at the *2015 7th international conference on games and virtual worlds for serious applications (vs-games)*.

Correspondence: Artorn Nokkaew, Institute for Innovative Learning, Mahidol University, Thailand.
E-mail: art.nokkaew@gmail.com