

**MODELING A PROBLEM SOLVING APPROACH
THROUGH COMPUTATIONAL THINKING FOR
TEACHING PROGRAMMING**

ZEBEL AL TAREQ

**FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR**

2021

**MODELING A PROBLEM SOLVING APPROACH
THROUGH COMPUTATIONAL THINKING FOR
TEACHING PROGRAMMING**

ZEBEL AL TAREQ

**DISSERTATION SUBMITTED IN PARTIAL
FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SOFTWARE ENGINEERING
(SOFTWARE TECHNOLOGY)**

FACULTY OF COMPUTER SCIENCE AND

INFORMATION TECHNOLOGY

UNIVERSITY OF MALAYA

KUALA LUMPUR

2021

UNIVERSITY OF MALAYA
ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Zebel Al Tareq

Matric No: 17043515/1

Name of Degree: Master of Software Engineering

Title of Project Paper/Research Report/Dissertation/Thesis (“this Work”): Field of Study:

Modeling a Problem Solving Approach Through Computational Thinking for Teaching Programming

I do solemnly and sincerely declare that:

- (1) I am the sole author/writer of this Work.
- (2) This Work is original.
- (3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work.
- (4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work.
- (5) I hereby assign all and every right in the copyright to this Work to the University of Malaya (“UM”), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained.
- (6) I am fully aware that if in the course of making this Work, I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate’s Signature

Date: 16/ 08 / 2021

Subscribed and solemnly declared before,

Witness’s Signature

Date: 16/ 08 / 2021

Name:

Designation:

MODELING A PROBLEM SOLVING APPROACH THROUGH COMPUTATIONAL THINKING FOR TEACHING PROGRAMMING

ABSTRACT

Different teaching approaches for programming are widespread but what is essential for students is being able to computationally formulate an algorithmic solution at first and then transfer to code. A number of factors such as inefficient teaching approaches and lack of problem-solving skills are factors making this knowledge procedure difficult. This study aims to investigate teaching issues in solving programming problems and find the right approach to teach programming using a suitable problem solving approach method. Sorting algorithm as a concept for solving problems have been utilized to understand the effectiveness of the model in different teaching methods. After carrying out a thorough literature review on core concepts of the study, a pilot study was conducted, and it identified some difficulties faced in teaching programming and motivated the search for an approach to overcome the issues and design the workshops for the feasibility study. A problem-solving approach (PSA) model was formulated using computational thinking concepts based on the sorting problems. An experimental study was designed to evaluate the PSA model. The syntax-based programming workshop was the control group. The problem-based and the game-based programming workshops utilizing our problem-solving model using sorting algorithms were the experimental groups.

A one-way ANOVA test indicated that the mean score for syntax-based workshop post test scores ($M=6.99$, $SD=1.92$) was significantly different than the post test scores of activity-based workshop ($M=8.05$, $SD=1.96$) and the post test scores of game-based workshop ($M=8.62$, $SD=1.90$). However, the post test scores of activity-based workshop ($M=8.05$, $SD=1.96$) did not significantly differ from the post test scores of game-based workshop ($M=8.62$, $SD=1.90$).

The results suggested that students had improved their programming skills in all the workshops. However, participants had better acquisition of problem-solving skills and a better understanding of programming concepts with both the active learning skills compared to the syntax-based approach. Even though there was no significant difference between the scores of the active learning methods, a comparison between both the approaches from a teaching perspective suggested that game-based learning was more suitable due to its interactivity.

Keywords: Computational thinking, Problem-Solving Approach Model, Active learning, Problem-based learning, Game-based learning.

Universiti Malaysia

MEMODELKAN PENDEKATAN PENYELESAIAN MASALAH MELALUI PEMIKIRAN KOMPUTASIONAL UNTUK PENGAJARAN PROGRAM

ABSTRAK

Pendekatan pengajaran yang berbeza untuk pengaturcaraan **adalah semakin** meluas tetapi apa yang penting bagi pelajar adalah dapat merumuskan penyelesaian algoritma pada awalnya dan kemudian memindahkan ke kod. Beberapa faktor seperti pendekatan pengajaran yang tidak cekap, kekurangan kemahiran menyelesaikan masalah dan lain-lain adalah faktor yang menyukarkan prosedur pengetahuan ini. Kajian ini bertujuan untuk menyelidiki masalah pengajaran dalam menyelesaikan masalah pengaturcaraan dan mencari pendekatan yang tepat untuk mengajar pengaturcaraan menggunakan kaedah pendekatan penyelesaian masalah yang sesuai. Algoritma **isihan** sebagai konsep untuk menyelesaikan masalah telah digunakan untuk memahami keberkesanan model dalam kaedah pengajaran yang berbeza. Setelah melakukan tinjauan literatur yang menyeluruh mengenai konsep utamakajian ini, sebuah kajian rintis dilakukan, dan menunjukkan beberapa kesulitan yang dihadapi untuk mengajar pengaturcaraan dan memotivasi untuk mencari pendekatan untuk mengatasi masalah dan merancang bengkel untuk kajian kemungkinan. Model PSA dirumuskan menggunakan konsep pemikiran komputasional berdasarkan masalah isihan. Satu kajian eksperimental telah direka bentuk untuk menilai Model PSA. Bengkel pengaturcaraan berasaskan sintaks adalah sebagai kumpulan kawalan. Bengkel pengaturcaraan berasaskan masalah dan pembelajaran berasaskan permainan yang direka bentuk berdasarkan model PSA adalah kumpulan eksperimental.

Ujian ANOVA sehalu menunjukkan bahawa skor min untuk ujian pasca bengkel berasaskan sintaks ($M = 6.99$, $SD = 1.92$) berbeza dengan signifikan berbanding dengan skor ujian pasca bengkel berasaskan aktiviti ($M = 8.05$, $SD = 1.96$) dan skor ujian pasca bengkel berasaskan permainan ($M = 8.62$, $SD = 1.90$). Walau bagaimanapun, skor ujian pasca bengkel berasaskan aktiviti ($M = 8.05$, $SD = 1.96$) tidak berbeza dengan signifikan berbanding skor ujian pasca bengkel berasaskan permainan ($M = 8.62$, $SD = 1.90$). Hasil menunjukkan bahawa di semua bengkel pelajar telah meningkatkan kemahiran pengaturcaraan mereka. Walau bagaimanapun, para peserta memperoleh pemerolehan kemahiran menyelesaikan masalah dengan lebih baik dan pemahaman konsep pengaturcaraan yang lebih baik dengan kedua-dua pendekatan pembelajaran aktif berbanding dengan pendekatan berasaskan sintaks. Walaupun tidak ada perbezaan yang signifikan antara skor kaedah pembelajaran aktif, perbandingan antara kedua-dua pendekatan dari perspektif pengajaran menunjukkan bahawa pembelajaran berasaskan permainan lebih sesuai kerana interaktivitinya.

Kata kunci: Pemikiran komputasional, Model penyelesaian masalah, Pembelajaran aktif, Pembelajaran berasaskan penyelesaian masalah, Pembelajaran berasaskan permainan.

ACKNOWLEDGEMENTS

I would first like to thank my dissertation supervisor, Dr. Raja Jamilah Raja Yusof, lecturer at Faculty of Computer Science and Information Technology, University of Malaya. The door to Dr. Raja's office was always open whenever I ran into a troubled spot. Even during the Covid-19 pandemic, she has given all the necessary guidelines and support via online to keep the research process active and smooth. She consistently allowed this paper to be my own work but led me in the right direction whenever she thought I needed it.

I would like to thank all the participants who participated in this study. Their active and spontaneous participation has made it possible to achieve the most important goals of this research and made it an exciting experience.

I would also like to thank University of Malaya for giving me the chance to study here and enhance my knowledge for my future.

Finally, I must express my very profound gratitude to my parents, my sister, my newly wedded wife and my other family members, my classmates, my friends and my colleagues for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation. This accomplishment would not have been possible without them. Thank you.

Author

Zebel Al Tareq

TABLE OF CONTENTS

ABSTRACT.....	iii
ABSTRAK.....	v
Acknowledgements.....	vii
Table of Contents.....	viii
List of Figures.....	xiii
List of Tables.....	xvi
List of Symbols and Abbreviations.....	xvii
CHAPTER 1: INTRODUCTION.....	1
1.1 Background.....	1
1.2 Problem Statement.....	4
1.3 Research Objectives.....	4
1.4 Research Questions.....	5
1.5 Scope of the Study.....	6
1.6 Significance of the Study.....	7
CHAPTER 2: LITERATURE REVIEW.....	9
2.1 Why Computational Thinking as a Problem-Solving Skill.....	9
2.1.1 Critical thinking and its relationship with CT.....	10
2.1.2 Analytical thinking and its relationship with CT.....	11
2.1.3 Creative thinking and its relationship with CT.....	11
2.1.4 Justification.....	12
2.2 Problem-Solving in Programming through Computational Thinking.....	12
2.3 Computational Thinking in Solving Programming Problems.....	13
2.4 Computational Thinking Components for Solving Problems.....	16

2.4.1	Data	18
2.4.2	Decomposition.....	18
2.4.3	Pattern Recognition	18
2.4.4	Abstraction	18
2.4.5	Algorithmic	19
2.5	Discussing Teaching Approaches.....	19
2.5.1	Teacher-centred learning.....	20
2.5.1.1	Benefits and difficulties in the teacher-centred approach	20
2.5.1.2	Teacher-centred learning in programming.....	20
2.5.2	Student-centred active learning.....	21
2.5.2.1	Benefits and difficulties in student-centred active learning approach	23
2.5.2.2	Student-centred active learning in programming.....	23
2.5.3	Justification on the chosen pedagogy.....	24
2.6	PBL in Programming and CT.....	24
2.7	GBL in Programming and CT	25
2.8	Interactive Programming and Learning Environment	27
2.9	Game-Based Features and Components	29
2.10	Sorting Algorithms and its Application in Programming.....	30
2.11	Sorting Algorithm and Computational Thinking.....	34
2.12	Gap Analysis.....	34
CHAPTER 3: RESEARCH METHODOLOGY		36
3.1	Identifying the Literature.....	36
3.2	Pilot Study	37
3.3	Create PSA Model	38
3.4	Modelling and Notation.....	38

3.4.1	Model the problems	38
3.4.2	Notate the CT steps	39
3.4.3	Design the workshops	39
3.5	Experimental Design and Participants	39
3.5.1	Syntax-based learning	40
3.5.2	Problem-based learning	40
3.5.3	Game-based learning	40
3.6	Pre-Test and Post-Test	40
3.7	Analysis and Evaluation of the Pre-Test and Post-Test	41
CHAPTER 4: PILOT STUDY AND PSA MODEL DEVELOPMENT		42
4.1	Pilot Study to Evaluate Traditional Teaching Effectiveness	42
4.2	PSA Model and Modelling Sorting Problems	44
4.2.1	PSA MODEL	45
4.2.2	Sorting problems using PSA model	47
4.2.2.1	Bubble sort	47
4.2.2.2	Counting sort	49
4.2.2.3	Merge sort	51
4.2.2.4	Quick sort	55
4.2.2.5	Bucket sort	58
CHAPTER 5: EXPERIMENTAL PLANNING AND DESIGN		61
5.1	General Process for Feasibility Study	61
5.2	Syntax-Based Workshop	62
5.3	Problem-Based Learning Workshop	63
5.4	Game-Based Workshop	64
5.5	Design of the workshop materials	65

5.5.1	Teacher-centred syntax-based workshop design.....	66
5.5.2	Problem-based workshop design.....	67
5.5.2.1	Bubble sort in PBL.....	68
5.5.2.2	Counting sort in PBL.....	71
5.5.2.3	Merge sort in PBL.....	75
5.5.3	Game-based workshop design (interactive gamified system).....	79
5.5.3.1	Modules of the interactive gamified system.....	80
5.5.3.2	Bubble sort in GBL.....	81
5.5.3.3	Counting sort in GBL.....	85
5.5.3.4	Merge sort in GBL.....	89
CHAPTER 6: FINDINGS AND ANALYSIS		95
6.1	Comparison between Pre-Test and Post-Test.....	95
6.1.1	Syntax-based workshop.....	95
6.1.2	Problem-based learning workshop.....	96
6.1.3	Game-based learning workshop.....	96
6.2	Comparison between Pre-Tests of the Three Workshops.....	97
6.3	Comparison between Post-Tests of the Three Workshops.....	97
6.4	Analyse and Discuss the Results.....	98
CHAPTER 7: CONCLUSION.....		102
7.1	Research Contributions.....	102
7.2	Revisiting the Objectives.....	102
7.2.1	First objective and related question and answers.....	103
7.2.2	Second objective and related question and answers.....	103
7.2.3	Third objective and related question and answers.....	104
7.3	Significance of Study.....	104

7.4	Concluding Remarks	104
7.5	Research Challenges	105
7.6	Future Improvements.....	105
	References	106
	Appendix B	121
	Appendix C	124
	Appendix D	127
	Appendix E	129
	Appendix F	131
	Appendix G	133
	Appendix H	137

Universiti Malaysia

LIST OF FIGURES

Figure 1.1: Google Trend comparison of different teaching methods	1
Figure 2.1: Project-based learning subsets	22
Figure 3.1: Research Methodology Procedure	36
Figure 4.1: Problem-solving approach (PSA) model	45
Figure 4.2: Bubble sort using PSA model	47
Figure 4.3: Counting sort using PSA model	49
Figure 4.4: Merge sort using PSA model	51
Figure 4.5: Quick sort using PSA model	55
Figure 4.6: Bucket sort using PSA model	58
Figure 5.1: Experimental planning of the study	61
Figure 5.2: Bubble sort using PSA for PBL	68
Figure 5.3: Unsorted numbers with id 1 and id 2 chosen	69
Figure 5.4: id 1 > id 2 and swapped	69
Figure 5.5: Sort all numbers following this process	69
Figure 5.6: Define array with collection of numbers and index	70
Figure 5.7 Check condition to swap	70
Figure 5.8: How loops are used to iterate through the array	70
Figure 5.9: Counting sort using PSA for PBL	71
Figure 5.10: A collection of unsorted numbers in listToSort and count from 0 until highest index	72
Figure 5.11: Count occurrence of each number and add in correct index	72
Figure 5.12: Count of all available numbers in listToSort greater than 0	72
Figure 5.13: Adding to final list and ignore count 0	73
Figure 5.14: Initialize an array named data	73

Figure 5.15: Define function and create count array.....	74
Figure 5.16: Keep count of listToSort.....	74
Figure 5.17: How to append to final array	74
Figure 5.18: Merge sort using PSA for PBL.....	75
Figure 5.19: Collection of unsorted numbers in Padlet	76
Figure 5.20: Left and right ids are divided finding mid.....	76
Figure 5.21: Further divide between left side	76
Figure 5.22: Sort and merge left side	77
Figure 5.23: Further divide right, sort and merge	77
Figure 5.24: Sort and merge sorted left and right	77
Figure 5.25: Divide the array until length is greater than 1	78
Figure 5.26: Divide left and right further.....	78
Figure 5.27: Sort and merge last divided items.....	78
Figure 5.28: Sort and merge when no more items left in one side.....	79
Figure 5.29: Modules of the system.....	80
Figure 5.30: Levels to choose before starting the game	80
Figure 5.31: Bubble sort using PSA for GBL.....	81
Figure 5.32: Choose some random numbers.....	82
Figure 5.33: Two numbers highlighted.....	82
Figure 5.34: Swap or no swap based on condition.....	82
Figure 5.35: Game completed after all steps with score	83
Figure 5.36: Random numbers chosen initialized as array	83
Figure 5.37: If condition to swap numbers	83
Figure 5.38: A for loop for comparing the numbers	84

Figure 5.39 Nested loop to go (size - 1) rounds	84
Figure 5.40: Counting sort using PSA for GBL.....	85
Figure 5.41: Choose random numbers and find highest	86
Figure 5.42: Drag and drop highlighted number to respective index	86
Figure 5.43: Ignore for count = 0 and add for count > 0.....	86
Figure 5.44: Count index appended to final array	87
Figure 5.45: Define function and find highest number in list.....	87
Figure 5.46: How count array is created to keep count of numbers in data.....	88
Figure 5.47: How to append by ignoring count = 0	88
Figure 5.48: Merge sort using PSA in GBL.....	89
Figure 5.49: Choose a collection of unsorted numbers and find the highest	90
Figure 5.50: Left and right divided	90
Figure 5.51: Further divide left side.....	90
Figure 5.52: Drag and drop smaller number to highlighted final index.....	91
Figure 5.53: Further divide right side and merge.....	91
Figure 5.54: Drag and drop to merge sorted left and sorted right.....	91
Figure 5.55: Random chosen unsorted number as array and function defined.....	92
Figure 5.56: Break left and right side by inputting mid.....	92
Figure 5.57: Further right division calling recursive function	93
Figure 5.58: While loop for merging the unsorted divisions	93
Figure 5.59: Additional while loops if items are remaining in either side.....	93
Figure 5.60: Sorted left and right merging using while loops	93

LIST OF TABLES

Table 2.1: Research involving CT with other problem-solving skills	9
Table 2.2: Research involving different models of CT.....	12
Table 2.3: Findings from the studies related to computational thinking.....	14
Table 2.4: Usage of different CT concepts	16
Table 2.5: PBL in programming and CT	24
Table 2.6: Results from the studies of PBL and GBL with CT	25
Table 2.8: Findings from the studies of interactive programming learning studies.....	28
Table 2.10: Results from the studies of sorting algorithm and programming	30
Table 2.11: Classifications of sorting algorithm	33

Universiti Malaysia

LIST OF SYMBOLS AND ABBREVIATIONS

CT	:	Computational Thinking
PSA	:	Problem-solving Approach
CRIT	:	Critical Thinking
CRET	:	Creative Thinking
AT	:	Analytical Thinking
PBL	:	Problem-based learning
GBL	:	Game-based learning
PJBL	:	Project-based learning

Universiti Malaya

CHAPTER 1: INTRODUCTION

Learning and teaching programming has been an integral part of the modern world. Programming learning is very well related to computational thinking (CT) that enhances the problem-solving skills of a person (Voogt et al., 2015). Those who are programming in their everyday life are solving different problems through the representation and execution of codes. Since computational thinking enhances problem-solving skills, teaching programming has to be related to computational thinking (Kong et al., 2020).

1.1 Background

A challenging aspect of teaching a programming course is how to provide the right information in the right context at the right time (Adamchik & Gunawardena, 2003). Recently, many teaching approaches are being considered to teach programming. Active learning techniques have surpassed traditional teacher-centric approaches (Acharya & Gayana, 2021) and emphasized a collaborative approach between a student and a teacher, rather than one way of providing information. It is one of the most talked-about processes of learning, and a few notable ones under active learning are game-based learning, problem-based learning, project-based learning, etc.

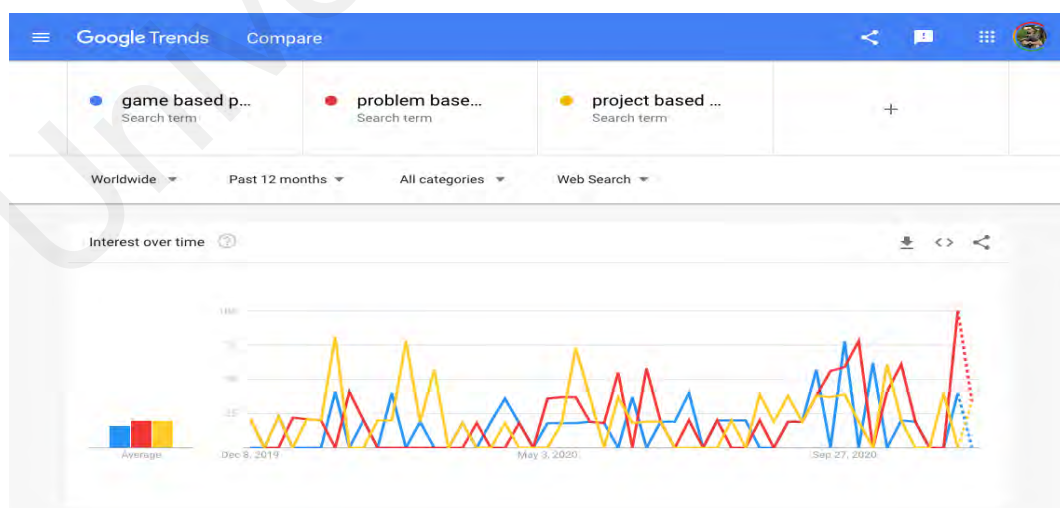


Figure 1.1: Google Trend comparison of different teaching methods

A comparison on Google Trends shows that project-based learning for programming has been a growing interest worldwide. Even though project-based learning has been trending, it is quite time consuming and often lead to assignment-based activities. Project-based learning is actually problem-based learning on a larger scale. Problem-based learning works with a problem to be solved and generates possible solutions. Problem-solving is an integral part of the game-based learning approach as well. Game-based learning is a gamified approach to problem-based learning. Despite all these different teaching procedures, it is still a matter of concern to teach programming effectively while engaging the students actively.

Teaching the concepts of programming and making it understandable for the learners has not been an easy task to do. It is known that programming learning is considered difficult and there have been many cases where learners have given up (Ala-Mutka, 2004). One of the main sources of programming difficulties is not being able to understand the fine line between programming knowledge and programming strategies. Several teaching methods have been used for teaching programming. It is a fact that teachers are not emphasizing a step-by-step problem-solving approach but only focusing on one line to another line (Siti Rosminah & Ahmad Zamzuri, 2012). The most common teaching processes are the common ways of using basic programming concepts and syntax (M., 2014). There is no doubt that it is necessary to have some knowledge of syntax. But what is more important is to be able to formulate a solution to a problem step-by-step so that the approach is clear and then transfer it to programming syntaxes and achieve the desired outcome.

Formulating the problems strategically is an essential approach, and it is undoubtedly important to be engaged in the required problem-solving process. While the lack of problem-solving capabilities are seen as a possible cause of failure in programming, it is also evident that non-viable strategies of key programming concepts held by students lead

to misconceptions in programming learning (Bubica & Boljat, 2014). In a problem-based learning environment, students have to approach a problem in a systematic manner. The use of a strategic approach in programming teaching enables students to participate in the process and learn the concepts of programming. Utilizing a strategic approach in teaching programming problems helps to cultivate procedural skills among students to perform complex programming tasks (Xie et al., 2019). Students need to identify key facts, check for any missing information, assimilate new knowledge, and apply the information individually or in a group. Similarly, for programming problems, based on the given data, a computational problem-solving process has to be involved where a problem is decomposed into smaller sub-problems, where students need to identify similarities or dissimilarities, abstract unnecessary characteristics, or avoid repetitions and processes in an algorithmic manner. It is very much inter-related to the teaching and learning of computer programming through the use of CT as a strategy to formulate problems based on the concepts of abstraction, decomposition, algorithms, logic, patterns and evaluation (Shute et al., 2019). CT is treated as a way of thinking just like the thought process of a computer which provides solutions to a specific task (Zaharin et al., 2018).

According to Wing (2012), it is the problem-solving process that is associated in formulating a problem and expressing a solution to that problem in such a way that can be carried out in an effective manner by a human or a machine. It helps to develop an analytical ability that enhances critical thinking and arithmetic capabilities. For someone to have a good base for programming, computational thinking is a rudimentary skill.

A fundamental approach towards learning to program and develop problem-solving skills is a considerable idea to develop fundamental skills. In order to teach programming to students, the teaching of sorting algorithm helps to cover the fundamentals of programming concepts (M. G. Voskoglou & Buckley, 2012a). **Sorting algorithm** implies defining a technique to arrange data in a specified order where most of the grouped orders

are in an arithmetic or lexicographical order (Rana et al., 2019a). Sorting problems have many algorithmic solutions and is convenient for beginners to be indulged in the process of it.

1.2 Problem Statement

As many students are finding programming to be difficult and disheartening (Ismail et al., 2010a), it is not yet clear to educators what could be an effective approach for teaching programming to the students and is becoming an area of concern. A number of issues arise when programming is being taught to students. Ineffective teaching approaches and lack of problem-solving skills are among the important factors that are contributing to the difficulties of teaching and learning of programming (Cheah, 2020).

The biggest mistake that many new programmers make is not focusing on how to formulate a solution to a problem but focus on learning the syntax (Spraul, 2013). Since problem-solving skills are important in order to learn and solve programming problems, it is essential to be cultivated in order to excel at programming. Problem-solving is a core concept in programming education and is regarded as a key skill to model the solution of a problem (Caeli & Yadav, 2020). There is a gap to understand the effective teaching approach that fully focuses on solving programming problems.

1.3 Research Objectives

With the inclusion of a proper problem-solving skill for solving programming problems, it is also necessary to understand the effective teaching method that is useful to develop CT based problem-solving skills. The main aim of the study is to learn how effective a problem-solving approach based on computational thinking is in teaching programming compared to the traditional way of teaching the subject. Therefore, the identified objectives of this study are as follows:

- To investigate teaching issues in solving programming problems.

- To design a problem-solving approach (PSA) model for teaching programming through the chosen problem-solving skill.
- To evaluate the effectiveness of the PSA model in relation to different programming teaching approaches.

1.4 Research Questions

The research questions that arise to achieve the above objectives are:

- To investigate teaching issues in solving programming problems.
 - Which problem-solving concept can be selected as a problem-solving skill among the notable ones?
 - Does the selected problem-solving concept enhance problem-solving skills among students while learning programming?
 - What is an effective teaching approach to develop problem-solving skills for solving programming problems, and why?
- To design a problem-solving approach (PSA) model for teaching programming through the chosen problem-solving skill.
 - How can CT concepts help to formulate a problem-solving approach (PSA) model in programming?
 - How does the PSA model help to solve problems for different algorithmic solutions?
 - How can the PSA model be translated into effective programming teaching methods to solve problems?
- To evaluate the effectiveness of the PSA model in relation to different programming teaching approaches.

- How to evaluate the effectiveness of different teaching approaches blended into the PSA model?
- Which teaching approach is more effective in utilizing the PSA model?

1.5 Scope of the Study

Several important works of literature on different pedagogical approaches, CT, and sorting algorithms related to programming studies have been discussed. There are many types of active learning approaches such as problem-based learning, experimental learning, flipped classroom etc. Since this study is focused on a problem-solving approach, the study has mainly focused on problem-based learning and teaching methods that are closely related to problem-based learning. Problem-based learning has been chosen as it emphasizes a balance in designing a course along with discussing problems(Xie et al., 2019). Game-Based learning focuses on problem solving as well by integrating problem-based learning in a gamified approach(Kazimoglu, Kiernan, Bacon, & Mackinnon, 2012). In addition, CT and its components and how they help in solving algorithmic problems have been an integral part of the discussion. Sorting algorithms are considered a problem-solving concept, and their importance in programming studies have been described in different studies.

This study focuses on a problem-solving approach model to solve problems and integrate them with teaching programming. It works with various sorting algorithms such as bubble sort, merge sort, counting sort, quick sort and bucket sort. These sorting algorithms are considered as the problems that need to be solved using the PSA model.

The participants in this study were mostly students from pre-university or first-year students from the undergraduate level. However, some professionals with little to no programming background were also allowed since this was an online workshop.

1.6 Significance of the Study

Since this study will be focusing on the problem-solving approach, while sorting problems have a number of algorithmic solutions, sorting algorithms along with CT has been considered as vital concepts to teach programming and learn the effectiveness of the teaching. Sorting algorithms are identified as the problems to be solved in this study during the process of problem-based learning.

If we take a look at the teacher-centred style of teaching programming across various institutions or websites, it is common practice that programming is taught using syntax from start to finish. Sorting algorithm is a very important part of programming and has many elements related to the syntax of programming. It might be effective to use sorting as a domain and to formulate their solution utilizing a problem-solving approach based on computational thinking. The study aims at getting a clearer insight of how effective the proposed problem-solving approach benefits students while teaching programming. Furthermore, the study has compared teacher-centred programming learning and problem-based learning to realize their impact on students.

Problem-solving skills such as analytical thinking (AT), computational thinking (CT), critical thinking (CRIT), creative thinking (CRET) etc. are being referred as problem-solving processes in this research. The strategies of these skills are referred as problem-solving approaches. In addition, Active learning and Teacher-centred approach is being referred as Teaching approach. Game-based learning, Problem-based learning, Project-based learning are being referred as active learning technique in the required cases. This research has been organised into the following chapters:

- I. Introduction
- II. Literature Review
- III. Research Methodology
- IV. Pilot Study and PSA Model Development

- V. Experimental Planning and Design
- VI. Findings and Analysis
- VII. Conclusion

Universiti Malaya

CHAPTER 2: LITERATURE REVIEW

It is important to know the desired problem-solving skill and know why it is the most necessary one. It is equally important to discuss the relation of the needed problem-solving skill and how it accumulates different teaching approaches.

2.1 Why Computational Thinking as a Problem-Solving Skill

Thinking is a cognitive activity utilized to process information, create a solution to problems, making decisions, and generating new ideas. Computational Thinking (CT) is the problem-solving process associated with formulating a problem and expressing a solution to that problem. Besides CT, there are several types of core problem-solving skills, and the notable ones include analytical thinking (AT), critical thinking (CRIT), and creative thinking (CRET). A study that has closely monitored programming and CT effectiveness on developing several problem-solving skills such as AT, CRIT, and CRET has suggested that there was a noticeable impact of CT and programming on these mentioned problem-solving skills even though further research is required to get clearer insights (Wong & Cheung, 2020).

Table 2.1: Research involving CT with other problem-solving skills

Author	AT	CRIT	CRET	Perception	Findings
(Wong & Cheung, 2020)	✓	✓	✓	Investigating the effects of programming and CT on problem-solving skills.	Noticeable impacts of programming and CT on the problem-solving skills.
(Van Dyne & Braun, 2014)	✓			Evaluate a CT course developed to improve the analytical skills of students.	Increase student analytical problem-solving skills.
(Atmatzidou & Demetriadis, 2014)		✓		Evaluation of CT skills in educational robotics activities to enhance critical thinking.	Students became familiar with CT skills and enhance critical skills.

Table 2.1 continued: Research involving CT with other problem-solving skills					
Author	AT	CRIT	CRET	Perception	Findings
(Kules, 2016)		✓		How CT and CRIT support each other and develop learning outcomes.	Suggests ways for better course design using the analysis of the relationship.
(M. G. Voskoglou & Buckley, 2012b)		✓		An experiment on problem-solving and discuss the relationship between CT and CRIT	Use of computers as a tool to enhance problem-solving capabilities.
(Avello-Martínez et al., 2020)			✓	Compressive review of CT and CRET's relationship with coding and educational robotics.	CT and CRET are related to find an efficient and good solution to problems in multiple ways.
(Hershkovitz et al., 2019)			✓	Trying to find association between general creativity and computational creativity.	Quite a few associations between the two creativity constructs.

In table 2.1, Research involving CT with other problem-solving skills have been illustrated.

2.1.1 Critical thinking and its relationship with CT

Critical thinking (CRIT) is the process of careful evaluation of a problem to be solved and determining the process of interpretation of that problem (Analytical Thinking and Critical Thinking, n.d.). To engage in problem-solving, we need to think at a deeper level and evaluate the problem using or adapting existing knowledge and skills, laying the groundwork for critical thinking (Doleck et al., 2017).

CRIT and CT complement each other as a way of solving problems, and their relationship suggests a better course structure in computer science (Kules, 2016). Furthermore, it enhances problem-solving capabilities (M. G. Voskoglou & Buckley, 2012b). An article by Laura says that CT is not that far afield from CRIT and the process of both of this problem-solving skills mirror each other (L. Lee, 2019). CT solves complex problems by integrating current knowledge and critical thinking (M. Voskoglou, 2013). Therefore, CT is a good approach chosen to solve problems because a vital concept of

problem-solving, which is CRIT, is connected with it and is nurtured during the process of it.

2.1.2 Analytical thinking and its relationship with CT

Analytical thinking (AT) is the mental process to break down a complex problem into smaller parts, and it is a component within critical thinking (Analytical Thinking and Critical Thinking, n.d.). Analytical thinking helps to distinguish and outline problems, extract key information, and develop feasible solutions.

A different study by Wing (2008) relates CT with AT by linking them to mathematical thinking in which a problem-solving approach might be initiated. One of the studies have proved that students' analytical problem-solving skills were improved by the association of CT concepts (Van Dyne & Braun, 2014). CT includes CRIT and AT for the development of an individual's problem-solving skills with the help of technology (Korucu et al., 2017). It is apparent that CT is a compact approach that integrates the other problem-solving skills such as AT and CRIT to better develop one's competencies.

2.1.3 Creative thinking and its relationship with CT

Creative thinking (CRET) is the ability to think in a new or different way. It refers to a new or different approach to solving a problem using a different angle (Tomaszewski, 2021).

When applying CT principles in problem-solving, a certain level of CRET is involved in it as a significant aspect of critical thinking (Doleck et al., 2017). CT and CRET have quite a few associations within their creativity constructs (Hershkovitz et al., 2019) and they are related to find an effective and good solution to problems in multiple ways (Avello-Martínez et al., 2020). In a number of aspects, CT problem-solving skills are found to be associated with CRET skills and is considered to be a reflection of logical and creative thinking along with problem-solving skills (Durak & Saritepeci, 2018). Therefore, CT is also a good problem-solving skill that relates to CRET to a certain extent.

2.1.4 Justification

The above discussion explains the relationship of CT with other core problem-solving skills that are associated with a problem-solving approach. It gives a clearer overview that computational thinking associates major problem-solving processes that are vital in solving problems. As a result, computational thinking fosters improvement in problem-solving skills by associating other thinking skills. Since other thinking skills for solving problems are associated with CT and can be enhanced through CT, it will be considered as the problem-solving approach for this paper.

2.2 Problem-Solving in Programming through Computational Thinking

Different teaching approaches are being utilized to make programming learning easier for students. Instructing and learning programming ideas and skills have been regarded as a huge challenge to both tutors and peers (Yang et al., 2015). Few problems that were identified are lack of skills in evaluating problems, ineffective use of problem representation processes to solve problems, and the unsuccessful application of teaching approaches for solving problems and programming (Ismail et al., 2010b). Some initiatives have been taken to enhance programming teaching and learning.

Table 2.2: Research involving different models of CT

Author	Perception	Problem-solving model used
(Szabó, 2020)	Interrelating concepts of programming tasks to enhance problem-solving.	A complete structure with programming concepts for algorithmic programming.
(Threekunprapa & Yasri, 2020)	Developing unplugged activities using flowcharts to enhance CT skills.	CT development model in different phases using flowcharts.
(Palts & Pedaste, 2020)	Common understanding of CT in CS and develop a model to describe three dimensions.	A model for developing CT in 3 stages consisting of 10 CT skills.

In table 2.2, Research involving different models of CT have been illustrated.

A conceptual structure has been created by connecting different programming concepts used in algorithmic thinking to enhance problem-solving skills among students (Szabó, 2020). This approach is useful provided the user has extensive knowledge in programming and already masters most of the well-known programming paradigms. In the case of a novice programmer, this is not an effective way to approach a solution to a problem. The use of algorithms proved to be useful for teaching programming but formulating through an effective approach is essential.

Another model for solving programming problems was illustrated, which displayed a model in 3 stages: a) defining the problem, b) solving the problem, and c) analysing the solution. The model also included 10 of the CT components (Palts & Pedaste, 2020). This model does not break down the computational thinking concepts step-by-step rather than just including them in different stages of the model. A limitation of the model is that there is not much information about the relationships between the CT elements.

In order for students to be genuinely involved in the process of programming learning based on computational thinking, a step-by-step approach is essential to illustrate the relationship between core CT elements. Even though unplugged activities resulted in finding student development of CT concepts in phases through flowcharts (Threekunprapa & Yasri, 2020), they cannot be the best approaches to foster programming learning using CT concepts until a relational problem-solving approach fully focused on CT has been modelled.

2.3 Computational Thinking in Solving Programming Problems

Computational thinking (CT) can be enhanced by participating in computational activities. Several studies have suggested that in the modern era of scientific and technological education, it is an important part. The effectiveness of CT in computer programming courses is widespread, and many studies have already concluded its

importance in programming. If computational activities are naturally integrated into the teaching process, then peers would be well prepared and more successful in solving programming problems as being computationally progressive (Hu, 2011). Another study suggests that greater efforts are needed to strengthen the foundation of CT long before the students participate in learning their first programming language (Lu & Fletcher, 2009). CT enables users to deal with complexity and open-ended problems and, as a result, persists in working with hard problems and solving them (D. Barr et al., 2011a). This helps students to get prepared for basic and extensive knowledge to get used to programming concepts.

Table 2.3: Findings from the studies related to computational thinking

Author Name	Perception	Findings
(Buitrago Flórez et al., 2017)	Highlight the necessity of learning programming at an early stage to develop CT skills.	Focus on CT for the CS and programming teaching and notice the adversities among different strategies.
(Nouri et al., 2020)	Understanding which CT related skills are developed among peers while working with programming.	Computational concepts, computational practices and computational perspectives are developed as CT perspectives.
(Yasmin et al., 2019)	Arguing CT framings, historize and situate CS to provide new directions for students to actively participate.	Propose a new direction to reframe CT by encompassing functional skills as well as socio-political and personal contexts to accompany youths' use.
(Kong & Wang, 2020)	Exploring CT perspectives development on programming learning and formation of computational identity (CI) components.	Questioning and connecting ability is developed and can foster CI formation, such as programming engagement.
(Zhang & Nouri, 2019)	A review to examine CT skills systematically through Scratch based on empirical evidence.	Students learn different CT skills and enhance their problem-solving skills.
(Kong et al., 2020)	Empirical evidence of design and evaluation of a teacher development program as there is a lack of high-quality research.	Better understand CT concepts and improved problem-solving skills.

Table 2.3 continued: Findings from the studies related to computational thinking		
Author Name	Perception	Findings
(García-Peñalvo & Mendes, 2018)	Exploring CT effects on pre-university students with a focus on CT to develop logical and problem-solving skills.	CT is an exemplary dimension to prepare students for the upcoming years.
(Voogt et al., 2015)	Advance discussion of what CT is and present instances of what is required to be taught and how.	Need to study development procedure of CT, improve the ability to deal with complexity, and study the role of programming.
(Marcelino et al., 2018)	Develop and run an education course for elementary school teachers to learn CT concepts and Scratch via e-learning courses.	It was possible for the trainees to acquire CT concepts and Scratch as well as develop useful products.

Findings from the studies related to computational thinking have been illustrated in table 2.3.

Programming and CT complement each other, and it has been suggested to teach programming at an early stage to develop CT skills gradually (Buitrago Flórez et al., 2017). A number of CT related skills such as computational concepts, computational practices, and computational perspectives are developed among students (Nouri et al., 2020) and encompass functional skills for active participation (Yasmin et al., 2019) while students are working with programming. Moreover, integration of CT concepts in programming studies develops questioning and connecting ability through fostering programming engagement (Kong & Wang, 2020) which results in a better computational approach. Zhang & Nouri (2019) carried out a review in order to systematically examine CT skills that can be achieved through the learning of Scratch programming. It was learned that students could acquire different CT skills through Scratch programming.

The above discussion suggests that CT needs to be at the centre of computer science and programming learning and teaching to improve problem-solving skills (Kong et al., 2020). It is good to know that CT is another entity in the teacher's toolbox for future years and a new silver bullet for 21st-century education (García-Peñalvo & Mendes, 2018).

Nevertheless, further research is required to understand the role of programming in CT to establish the claim that it enhances problem-solving abilities (Voogt et al., 2015).

2.4 Computational Thinking Components for Solving Problems

Computational thinking (CT) is a method for solving problems, and it has extensive usage in the field of computer science. It integrates critical thinking and current knowledge and relates those to resolve complex technological problems. It has already been mentioned in the [Background](#) from Jeannette M. Wing that it is the problem-solving process that is associated with formulating a problem and expressing a solution to that problem in such a way that can be carried out in an effective manner by a human or a machine. Computational thinking can also be considered as the mental activity for abstracting problems and formulating solutions (Yadav et al., 2014). Computational thinking has enabled and driven many technologies in this era of modern science.

Table 2.4: Usage of different CT concepts

CT CONCEPTS	USAGE	ACADEMIC AUTHORS
Data	Finding the data source for a problem, analyze and represent the data.	(V. Barr & Stephenson, 2011)
		(I. Lee et al., 2014)
Decomposition	Breaking the problems into smaller sub-problems to solve the problem.	(Ahsan Habib, 2019)
		(Sa, 2018)
Pattern Recognition	Recognizing the patterns in the process that looks for similarities or dissimilarities.	(Ahsan Habib, 2019)
		(Sa, 2018)
Abstraction	Encapsulating a set of often-repeated commands by filtering out unnecessary characteristics.	(V. Barr & Stephenson, 2011)
		(D. Barr et al., 2011b)
		(Hazzan & Kramer, 2008).
		(I. Lee et al., 2011).
Algorithmic		(Ahsan Habib, 2019)

	An algorithmic procedure for the problem to be solved.	(V. Barr & Stephenson, 2011)
		(Hu, 2011)

Usage of different CT concepts have been illustrated in Table 2.4.

Universiti Malaya

Research by Ahsan already discusses various elements of CT and maps the CT elements into the important and mostly used programming attributes. Table 2 has indicated several CT elements and some research that has been carried out based on those concepts.

2.4.1 Data

In order to solve a problem, it is necessary to find the data source of that problem (V. Barr & Stephenson, 2011). Data has to be logically analysed and organized, then represented through models or simulations (I. Lee et al., 2014).

2.4.2 Decomposition

Decomposition simply refers to breaking down a complex problem into smaller sub-problems, so it is decomposed into easily solvable parts (Sa, 2018). CT uses decomposition whenever it is trying to work on solving a complex task of designing a system that is complex (Wing, 2006).

2.4.3 Pattern Recognition

When we are working with a set of data that is hard to work on individually, we use some techniques to understand a common pattern across the problem-solving approach. Once a complex problem has been decomposed, CT paves the way to look into shared characteristics (Ahsan Habib, 2019). Looking into shared similarities or dissimilarities of the decomposed data using these programming concepts is coined after the term 'Pattern Recognition' (Sa, 2018).

2.4.4 Abstraction

In any basic or complex problem-solving approach, there will a need to ignore unnecessary characteristics by focusing on the general ones that are common to all or multiple elements (Sa, 2018). When it compresses a set of repetitive commands, that term is often referred to as abstraction (V. Barr & Stephenson, 2011). The precise role of abstraction is it is used whenever needed. In very simple words, the process of abstraction

can be perceived as a purpose of many-to-one mapping (Hazzan & Kramer, 2008). In programming, a model needs to be created with the general characteristics of the problem to be solved. It can take the form of stripping down a problem by capturing shared characteristics into a single set as a representative of other instances (I. Lee et al., 2011).

2.4.5 Algorithmic

Algorithms are step-by-step rules to follow when the problems are being solved (Sa, 2018). It is an idea to achieve the output by following the sequential statements correctly. A study indicates that algorithms are a precise specification of the functionality of a system that determines the quality of the computation process and effectively handles the complexity of models and representations (Hu, 2011). One study says that algorithmic thinking helps to automate solutions (D. Barr et al., 2011b).

The use of these different computational thinking concepts helps to create a step-by-step algorithmic solution. Once particular data or a group of data has been decomposed, they can be stripped down as a problem to be solved through abstraction by recognizing the pattern and eventually lead to an algorithmic solution.

2.5 Discussing Teaching Approaches

Teaching computer programming in computer science courses has been a difficult task for teachers so far. There have been many discussions to find the best pedagogical approach to teaching programming. In addition, computer programming is not actually an easy topic to master due to the nature of the subject (Lahtinen et al., 2005). Both teachers and students are still struggling for a better knowledge give and take process. High-quality research for programming education is important for a student to better understand the subject and gain quality knowledge (THE ROYAL SOCIETY, 2017). Most teaching practices fall into two kinds of pedagogical approaches, which are the a) teacher-centred approach and the b) student-based active learning approach (Duckworth, 2009) (Wohlfarth et al., 2008).

2.5.1 Teacher-centred learning

If a **teacher-centred pedagogy** is mentioned, then the name clearly indicates that it is mostly teacher-centric, and students are following the instructions from the teachers. This approach is more tutor-centred rather than being learner-centred. It can be referred to as the combination of an active teacher and a passive student (Mascolo, 2009). The teacher functions as the centre of the knowledge process as a classroom lecturer, present information to students and students are passively expected to obtain the knowledge as it is being presented (Faraon et al., 2020).

2.5.1.1 Benefits and difficulties in the teacher-centred approach

An article at the University of San Diego has discussed some potential benefits and drawbacks of the teacher-centred approach (Lathan, 2017). Teacher centred approaches in learning have a few benefits, and the notable ones are: i) the teacher maintains good order in class during the teacher-centred approach, ii) full responsibility is on the educator, iii) educators feel more comfortable while being in charge of class activities, and iv) students know that they need to focus on the teacher.

This article has also focused on the drawbacks of the teacher-centred approach, which include: i) the method will only be effective when a lesson is made interesting by the instructor, ii) lack of collaboration, iii) missing opportunities of discussion and sharing the discovery with classmates, and iv) lesser opportunities to develop problem-solving skills.

2.5.1.2 Teacher-centred learning in programming

A study indicated that teacher-centred education in programming and computational thinking propelled only the good students to excel in their performance, whereas the average or weaker students were left behind (Sahin & Abichandani, 2013). A lack of participation from students is an issue in the teacher-centred approach and results in a lack of attention to the learning of the students (Gelisli, 2009). Another issue that has

been addressed in the previous section amongst the drawbacks is that the teacher-centred approach offers lesser opportunities to develop problem-solving skills, whereas problem-solving is one of the core factors in programming. In addition, a lack of collaboration and discussion makes learning difficult and only ensures better control by the teacher in the classroom, whereas problem-solving skills and better learning has to be the main motive.

2.5.2 Student-centred active learning

- **Active learning** has been quite popular in the 21st century, and it involves students having the course materials through different approaches. Active learning can be in any form of engagement such as discussions and group tasks, problem-based approach, role plays, use of games in teaching etc. Active learning can also be called student-based active learning since the students are placed at the centre of a lecture's objectives and outcomes (Malhotra, 2019). Students are not only occupied in learning but also participate in the mental processes. It places a greater degree of responsibility on the learners and makes them active compared to a teacher-centric approach. Compared to traditional approaches of programming lectures, better learning outcomes have been witnessed by utilizing active learning (Park et al., 2020). **Problem-based learning (PBL)** can be considered as one of the very well-known approaches in programming studies. It is a pedagogical approach that helps students to learn while dynamically engaging with problems (Yew & Goh, 2016a). A number of studies have been carried out on the usability of PBL and its impact on programming studies. Problem-based learning is a method that makes the problem the centre of learning and is proved to be an effective learning method in programming courses (Bawamohiddin & Razali, 2017).

- On the other hand, PBL is a subset of **project-based learning (PJBL)** (Larmer, 2015) but PJBL is more focused on a long-term team-based approach by working in phases (Bell, 2010) to solve problems and enhance problem-solving skills. Students get to integrate and communicate while doing a project and can improve both individual and group experience throughout the process (Subramaniam et al., 2017). Activity-based problem-solving is an effective medium of learning for the students.
- **Game-based learning (GBL)** is a very interesting approach where educational games are utilized as a medium of teaching and learning programming combined with problem-solving strategies to attract students to the learning process (C. S. Chang et al., 2015). Game-based learning usually simulates solving problems in a game environment that helps students to develop their logical and formulation skills (Jesus & Silveira, 2021). Game-based learning is also a subset of PJBL since it stimulates building up problem-solving skills among students. Educational games are a combination of activity simulation and ideals of problem orientation that revolves around the problem as in problem-based learning. These games are used to develop objectives of learning and, as a result, obtain better learning outcomes (Malliarakis et al., 2014).

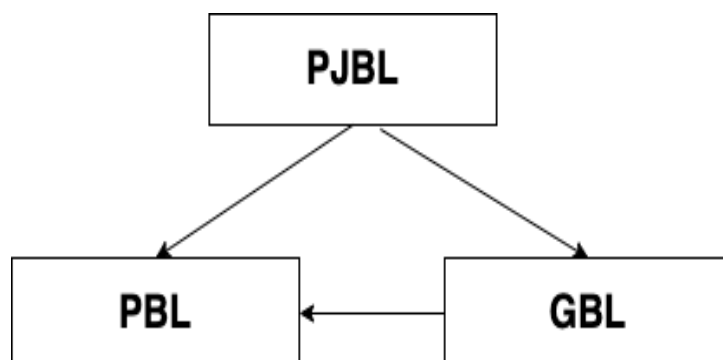


Figure 2.1: Project-based learning subsets

Project-based learning would be a suitable approach. But with the time constraints and enabling voluntary participation among students, problem-based learning would be a

more suitable approach to teach the students. Moreover, a gamified approach would be effectual as well to integrate problem-based learning in order to solve programming problems and develop CT skills.

2.5.2.1 Benefits and difficulties in student-centred active learning approach

The student-centred arrangement has been considered to be a dynamic process of learning (Price, 2019). This author has identified some potential benefits of the student-centred active learning approach. A few notable pros of student-centred active learning include: i) shared experience between educators and students, ii) facilitate critical thinking and further inquiry on solving problems, iii) learn to work independently and in a group, iv) peer-to-peer and peer-to-educator collaboration, and v) interact with each other during the learning process to solve problems. Some of the cons of student-centred active learning include: i) the class environment might be noisy or chaotic, ii) collaborative approach might not be beneficial for all students, iii) less focus on lectures might lead to missing out on important information.

2.5.2.2 Student-centred active learning in programming

Active learning concepts engage programming students in a group discussion to discuss the problem, analyse, and solve the problem (Tom, 2015). Programming is more about solving problems, and most of the active learning concepts focus on keeping the problems as a focus and to engage the students. Student-based learning enables enhanced participation and allows students to acquire knowledge and skills to solve programming problems that build design thinking, problem-solving, and the ability to analyse (Acharya & Gayana, 2021). Even though the class environment might feel a bit noisy and chaotic, the learning purpose is very well-served by utilizing a student-centred active learning approach.

2.5.3 Justification on the chosen pedagogy

The above discussion gives us a clear indication that student-based learning emphasizes more on solving problems and developing cognitive skills among students. In addition, students are more interested in learning activities that are collaborative and make them participate actively in the process (Otukile-Mongwaketse, 2018). As this study aims at solving problems within an effective pedagogy, student-centred learning will be very helpful since it focuses on the problem and engage the participants in the process.

2.6 PBL in Programming and CT

Table 2.5: PBL in programming and CT

Author Name	PBL	Perception	Findings
(Kale & Yuan, 2020)	✓	To know whether problem-solving skills are addressed through computational thinking.	The current study plans addressed some of the CT and problem-solving skills.
(Jonasen & Gram-Hansen, 2019)	✓	Highlight the benefits of PBL to develop CT skills.	CT concepts through PBL provide digital competencies and improve problem-solving.
(Yew & Goh, 2016b)	✓	Process of problem-based learning (PBL) and how its various components impact learning.	Longer-term knowledge retention and has been generally consistent and influence learning outcomes.
(Chen, 2018)	✓	Teaching programming based on CT and problem-based learning.	Better understanding of problem learning through discussion and improved test score.

Table 2.5 illustrates PBL in programming and CT. Even though some of the programming courses are integrating CT and problem-solving skills, it was evident in one of the studies that it is essential to focus on the other CT components to strengthen problem-solving skills (Kale & Yuan, 2020). Problem-based learning centralizes the problem by engaging students, and engaging CT concepts through PBL provides digital

competencies and foster problem-solving skills (Jonasen & Gram-Hansen, 2019). Problem-based learning and its various components are proven to be effective in longer-term knowledge retention and have been influencing learning outcomes on a consistent basis (Yew & Goh, 2016b). Discussion is an integral part of problem-based learning where a problem and the approach are discussed in steps. The discussion process assists students in understanding the problems better and have a greater achievement during academic learning (Chen, 2018).

2.7 GBL in Programming and CT

This part of the study will be focused specifically on how game-based learning blended in PBL has worked parallelly with computational thinking and has complemented each other. Since the study is focusing on using problem-based learning as the main pedagogical approach, and game-based learning also comes under problem-based learning, further discussion can provide a clearer overview.

Table 2.6: Results from the studies of PBL and GBL with CT

Author Name	PBL	GBL	Perception	Findings
(Kazimoglu, Kiernan, Bacon, & Mackinnon, 2012)	✓	✓	Design of an educational game framework to facilitate CT skill development in introductory programming.	Beneficial approach to help students to learn problem-solving skills.
(Zhao & Shute, 2019)	✓	✓	Evaluation of cognitive and attitudinal influence utilizing a video game for students CT skill development.	Improved CT skill and problem-solving significantly.
(Berland & Lee, 2011)	✓	✓	board games represent an informal and collaborative context to observe complex computational thinking.	Spontaneous board gameplay can contribute to developing complex computational thinking.
(Menon et al., 2019)	✓	✓	Evaluating tabletop escape games as a probable tool to cultivate CT among K-12 learners and	unplugged activities and a computer programming approach can help to remain

Table 2.6 continued: Results from the studies of PBL and GBL with CT				
Author Name	PBL	GBL	Perception	Findings
			assessing its effectiveness.	involved in the learning process.
(Turchi et al., 2019)	✓	✓	Focusing on playfulness and collaboration by introducing a game-based system to foster CT skills.	Playfulness can involve a wide audience to learn CT skills, whereas collaborative aspects are possibly effective to stimulate problem-solving formality on end-user reflection.
(Kazimoglu, Kiernan, Bacon, & MacKinnon, 2012)	✓	✓	Analysis of how a game supports CT concepts and mapping those to programming constructs.	The game was well designed for most students to understand introductory concepts.
(Malizia et al., 2020)			Introduces a game that combines GBL with tangible UI and virtual reality to foster CT skills.	Players will learn about their problem-solving abilities on the progress of CT.
(Tsarava et al., 2017)	✓	✓	A gamified instruction approach of plugged-in and unplugged activities to master specific CT processes.	Students can construct knowledge through playing and interacting with educational activities.
(Garneli & Chorianopoulos, 2019)	✓	✓	Effects of video game making (VGM) on CT skills development and peer performance.	VGM within science content brought in more CT skills and improved performance to acquire programming skills.
(C.-S. Chang et al., 2020)	✓	✓	An educational game utilized among students to improve their knowledge of programming.	Results show that PBL learning approach of the game can enhance learning satisfaction and motivation.

Table 2.6 illustrates Results from the studies of PBL and GBL with CT. Educational games do not only engage the students but also bring in important educational concepts for actively participating in learning. One study suggested a game that was well designed to support pupils to learn introductory programming concepts (Kazimoglu, Kiernan, Bacon, & MacKinnon, 2012). Educational games assist in strengthening the development of CT skills in introductory programming, and this approach benefitted students to acquire

problem-solving skills (Kazimoglu, Kiernan, Bacon, & Mackinnon, 2012). A problem-based learning approach enhances the learning satisfaction and motivation of a student. However, during the process, it is important that the students remain well connected in the process. Programming and unplugged activities can keep learners actively involved in the process (Menon et al., 2019). An active problem-solving approach can contribute to developing computational thinking among students (Berland & Lee, 2011). Whether via gamified or non-gamified problem-based approaches, playfulness is important to involve students to learn CT skills (Turchi et al., 2019). Furthermore, a problem-based approach with an interactive UI helps the instructor to measure the problem-solving abilities of the students as well as CT skills (Malizia et al., 2020).

Interactive video games are found to be interesting. A study by Zhao & Shute (2019) assessed that they improved the CT skills of users. Zhao and Shute evaluated attitudinal and cognitive impacts of a video game for CT skill development and assessed the effectiveness of the game by investigation of a particular game feature. An educational gamified instruction approach integrating a series of plugged-in and unplugged activities helped students construct knowledge through playing and interacting with educational activities (Zhao & Shute, 2019).

2.8 Interactive Programming and Learning Environment

In order to teach computational thinking skills, different technologies, including interactive learning environments (ILE), have been utilized to support teaching and learning (Barron-Estrada et al., 2020). An interactive programming environment or interactive development environment is a system that supports the basic tools which are needed to write and test software. A number of tools are used by developers across the globe for software code creation, building, and testing. Besides interactive development environments, there are learning tools that educators have been using over the years for making learning smooth for their students. Interactive programming learning

environments are gaining more popularity day by day among educators and pupils. It helps in greater student engagement and accelerates learning in a fun way rather than a conventional approach.

Table 2.7: Findings from the studies of interactive programming learning studies

Author Name	Game-Based Concepts	Findings
(Barron-Estrada et al., 2020)	An interactive learning environment focused on the development of CT skills using block-based visual programming.	Improved enjoyment and ease of use, natural UI provides learning motivation.
(Wang et al., 2020)	Introduce a self-paced programming environment that combines block-based and visual programming with structured practices for engagement.	Kept students engaged and progressive even though some struggled and required additional assistance.
(Tariq, 2020)	Exploring the usability of a shader mode tool for open-source software widely used in education and other sectors, discuss how it can aid shader for emerging computational artists.	Shader examples played an important part in providing inspiration and providing learning for less experienced users.
(Schwebel et al., 2012)	Evaluating “Blue Dog” to teach young children how to interact safely with dogs.	Children using “Blue Dog” had a greater change in recognition of risky dog situations than children learning fire safety.
(Smith et al., 2020)	Describe initial work towards solving the leveraging of narrative and computational thinking to engage students in a narrative-centred environment.	The environment successfully enabled students to engage in different CT concepts.
(Nguyen et al., 2020)	A programming environment to allow students producing an agent in a virtual world enabled to answer questions in spoken language from a user.	With an accuracy of 78 percent, students were capable of communicating intuitively with the environment.

Table 2.8 illustrates findings from the studies of interactive programming learning studies .An interactive learning environment (ILE) called CREA y JUEGA emphasized the enrichment of CT skills utilizing block-based programming, enhanced ease of participation, and motivated learners (Barron-Estrada et al., 2020). In order to enhance student engagement, another study introduced a self-paced coding environment that

combined block-based and visual programming. It kept most of the students engaged, and they progressed throughout the process. But some students still struggled with such a highly structured program (Wang et al., 2020). A study by Tariq (2020) found out the overall usability of a shader mode tool to be useful for teaching programming with CT for less experienced users, along with discussing the usefulness of the system. Taking a look for other research papers that uses ILE for general usages, such as teaching young children how to interact with dogs. A study described initial work towards solving the leveraging of narrative thinking and computational thinking through a narrative-centred interactive environment. The research found that the environment was successful in creating narratives as well as engaging peers in various CT concepts (Smith et al., 2020).

A research that evaluated “Blue Dog” had made children more aware of risky situations with dogs and how to interact safely (Schwebel et al., 2012). Another study proposed a VR programming environment to enable peers to produce an agent in a virtual world enabled to answer questions in spoken language from a user. With a high accuracy result of 78 percent, students were capable of intuitively communicating with the environment (Nguyen et al., 2020).

2.9 Game-Based Features and Components

There are a few things that need to be taken into consideration while designing a game. A study has confirmed that there are five crucial features to be focused on when building a game, including design, controls, social features, assets, and ease of navigation (Mohd et al., 2016). One of the best ways is by breaking the game into different levels. In addition, some rewards like points give a level of confidence to the user and ensure interesting engagement. It is better if the game is easy and has different content. However, the main goal of the game must be focused on each step of the development. The navigational features must be logical and easy. Assets such as fonts, look, and feel give a good impression to the user about the game. Social features are a good thing to be

implemented in a game but might not be very useful for the purpose of an educational game.

2.10 Sorting Algorithms and its Application in Programming

Sorting is a technique that is utilized for rearranging a set of unsorted items into a finite sequence or order, which might be lowest to highest, longest to shortest, alphabetical, and it is known as sorting algorithm (Margaret Rouse, 2017). It implies defining a technique to arrange data in a specified order where most of the grouped orders are in an arithmetic or a lexicographical order (Rana et al., 2019a). A very common example of sorting is when we buy products from e-commerce websites, we sort the items according to year or price, from lowest to highest. This simple example shows us how sorting is associated with our day-to-day life.

It is a very important operation in computer science. Sorting algorithms are often taught as a part of computer science education in the background of a programming language (Nasar, 2019). Good knowledge of sorting algorithm is an essential skill for computer science students or professionals. Sorting algorithm processes consist of well-defined steps for problem-solving, which is an undetachable part for driving the development and defining the discipline of computer science (Lui et al., 2019). Sorting algorithms enhance algorithmic thinking, which is more than necessary to solve programming problems. Few popular sorting algorithms are selection sort, bubble sort, insertion sort, merge sort, quick sort etc.

Table 2.8: Results from the studies of sorting algorithm and programming

Author Name	Perception	Findings
(Chuechote et al., 2020)	Provide theory-based explanation embracing a framework of cognitive development of how students ascertain sorting algorithms (SA) after digital gameplay.	Impact of the digital game on algorithmic thinking and for self-learning by discovering the relation between actions and schematic reasoning.
(Statter & Armoni, 2020)	Studying the effect of a framework for teaching abstraction with regards to	Indicate that the framework was highly effective to develop CS abstraction skills,

Table 2.10 continued: Results from the studies of sorting algorithm and programming		
Author Name	Perception	Findings
	algorithmic problem-solving by using Scratch for algorithmic solutions.	provide an explanation to a solution, use of initialization process and CS perception.
(Ward et al., 2010)	Examines implementing common CS problems through a parallel SA, another SA and a binary number conversion game in Scratch and Alice.	Both Scratch and Alice illustrated a skillful way to teach programming and resulted in programming significant CS concepts.
(Kohn & Komm, 2018)	Introduce programming as well as discuss algorithms by proposing a common computing agent as a notional machine.	Programming and algorithmic thinking need to be considered equally for explicit incorporation of underlying computing education.
(Bang, 2018)	Using the SortVR app to present a proof-of-concept that students can learn sorting algorithms using VR headsets.	The interaction was easy, and the app shows the potential of VR for sorting algorithm and CS concept teaching in the future.
(Muntean, 2019)	Introduces flipped classroom methodology to teach advance programming concepts such as sorting and searching in programming.	Results show that the methodology is useful in teaching the programming concepts with an enjoyable learning experience.

In table 2.10, Results from the studies of sorting algorithm and programming have been illustrated. Several studies have carried out research related to sorting and its application to programming courses. One of the notable aspects of sorting algorithm is that there are many ways of sorting, but still, it addresses the same problem. A study by Bang, (2018) presented a proof-of-concept using the SortVR app so that students can learn two sorting algorithms: bubble sort and selection sort, using VR headsets. It proved to provide better interaction and potentiality of VR technology for sorting algorithm as well as CS concept teaching in the future. In addition, the design of algorithms makes it more interesting for students. Another study embraced a framework to provide a theory-based framework of cognitive development of how students ascertain sorting algorithms (SA) after digital gameplay (Chuechote et al., 2020). It helped to find the impact of the

digital game on algorithmic thinking and for self-learning by discovering the relation between actions and schematic reasoning. The stability and adaptability of different algorithms can help to understand different CS problems and enhance problem-solving skills to a certain extent. A study examined implementing common CS problems through a parallel SA, another SA, and a binary number conversion game in Scratch and Alice (Ward et al., 2010). It was known that both Scratch and Alice illustrated a meaningful and creative way to engage while teaching programming as well as programming impactful CS concepts. The concept of abstraction and data is also a crucial CS concept which is a core part of sorting algorithms since it is basically about sorting numerous data. Statter and Armoni studied the effect of a framework for teaching abstraction to 7th grader novice students with regards to algorithmic problem-solving by using Scratch for algorithmic solutions (Statter & Armoni, 2020). The findings indicated that the framework was quite effective to develop abstraction skills, provide an explanation for a complicated solution, use of initialization process, and CS perception. In addition, the framework proved to be useful for improving the CS performance of students. Furthermore, the concept of sorting algorithm is very useful in computational education, and it is very rich in implementation. A study at the National College of Ireland introduced a flipped classroom-based methodology to teach advance programming concepts such as sorting and searching in programming for programming courses (Muntean, 2019). Results indicated that the approach was effective to teach concepts such as sorting and searching using the methodology for an enjoyable learning experience. In another paper, a framework using a mutual computing agent as a notional machine was proposed to introduce programming as well as discuss algorithms and their complexity (Kohn & Komm, 2018). It was learnt that programming and algorithmic thinking needs to be considered equally for explicit incorporation of underlying computing education.

Table 2.9: Classifications of sorting algorithm

Author	Year	Classifications of S.A.
(Rana et al., 2019b)	2019	In-place Sorting
		Not-in-place Sorting
		Stable Sorting
		Not Stable Sorting
		Adaptive Sorting
		Non-Adaptive Sorting

Table 2.11 shows classifications of different kinds of sorting algorithms according to a research. The paper has indicated six divisions of sorting algorithms (Rana et al., 2019b), and they are described below:

- **In-place Sorting:** Extra space for comparison is not required, e.g. selection sort, bubble sort.
- **Not-in-place Sorting:** More than or equal extra places needed to sort the elements, e.g., merge sort.
- **Stable Sorting:** Same items sequence not alternated in which they appear after being sorted, e.g., insertion sort, bubble sort.
- **Not Stable Sorting:** Same items sequence alternated in which they appear after being sorted, e.g., heap sort, quick sort.
- **Adaptive Sorting:** Uses already sorted items by not reordering them in sorted form, e.g., quick sort, insertion sort.
- **Non-Adaptive Sorting:** Force every single item to be reordered by confirming

the items have been sorted, e.g., heap sort, merge sort.

2.11 Sorting Algorithm and Computational Thinking

Programming is a process of learning that has multiple dimensions associated with it. Sorting algorithm and CT concepts are vital dimensions in learning programming. Along with sorting algorithm, CT concepts have a great contribution to both learning and applying problem-solving concepts (Nokkaew, 2019). Through sorting algorithms, students get to learn how to make comparison and rearrange items by using algorithmic thinking. Algorithmic thinking is one of the core concepts of CT. When sorting algorithm in programming, data are logically decomposed and represented through abstractions and solutions are automated through algorithmic thinking. On the other hand, it is also useful for educators because the basic concepts of sorting algorithms are easy to explain, and there is a large number of sorting algorithms that are useful in different circumstances (Anonymous, 2014).

2.12 Gap Analysis

The [problem statement](#) stated that ineffective pedagogy and the lack of problem-solving skills are a hindrance towards learning and teaching programming. A gap has been found which is to understand improving problem-solving skills through a problem solving approach and finding the effective teaching method to utilize the process. Some studies that have worked with problem-solving through computational thinking have been noted and explained the need for computational thinking skills besides other problem-solving skills. In addition, different teaching approaches and suitable teaching approaches for programming learning and computational thinking are discussed as well to address our problems. Problem-based learning and other active learning processes such as project-based learning and game-based learning and their relationships assimilate the need of GBL and PBL for this research. In addition, it is necessary to figure out the teaching

approaches along with the intended problem solving approach through computational thinking.

Universiti Malaya

CHAPTER 3: RESEARCH METHODOLOGY

This chapter focuses on the methodology on which this study has been carried out. It has identified the literature followed by a pilot study to understand students' basic level of understanding. A problem-solving approach (PSA) model has been created to model different sorting problems which was then used to design the teaching modules for the experimental group(s). Finally, the workshop was conducted, and the students were evaluated to establish the results of this study.

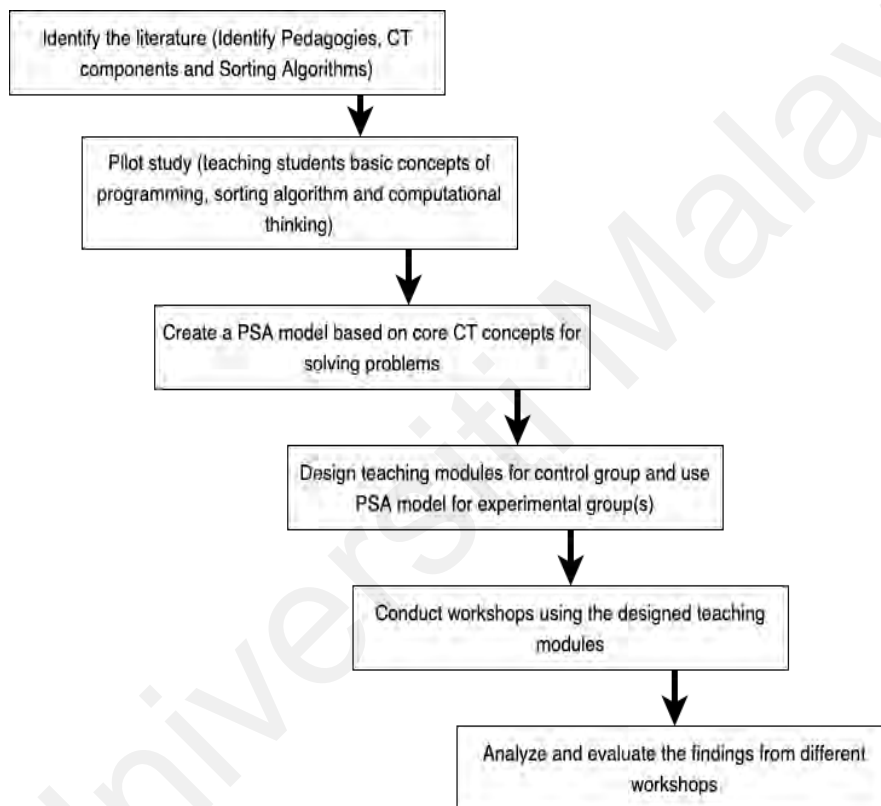


Figure 3.1: Research Methodology Procedure

3.1 Identifying the Literature

The literature review has identified the studies which have highlighted the need for a problem-solving approach. Our problem statement has pointed out computational thinking as a pivotal skill to solve programming problems, and further studies on the core problem-solving skills and their association with CT accelerated the fact that CT is necessary for solving complex programming problems. Then, the literature has identified

the computational thinking concepts which were used to formulate the solution of a problem. In addition, studies focusing on different kinds of teaching approaches and their findings have been discussed for a better insight into teaching techniques and establish why this study has chosen student-based active learning and its concepts such as PBL and GBL as its experimental teaching method. It was also identified that educational gamified approaches are utilizing problem-based learning as the teaching method. Furthermore, the discussions have added upon how sorting algorithm and CT support teaching programming. The additional discussion has been carried out for interactive learning environments and their significance in online learning and gamified approaches. Brief literature on crucial features of gamified approaches was discussed, which gave an idea for the design of the game-based approach for this study. Moreover, it was identified that sorting algorithm consists of the fundamentals of programming and can be quite effective with computational thinking for teaching various programming concepts.

3.2 Pilot Study

According to Doody & Doody (2014), a pilot study is a limited-scale version of an intended study and conducted within a small group of participants that are similar to the ones to be recruited later in the main study. The main objective of a pilot study is to enhance the prospect of success in the main study by testing the practicability to recruit the participants and determine the validity of the contents and materials (Fraser et al., 2018). A pilot study has been carried out in this study prior to aiming at the feasibility study. The study intends to investigate the issues in solving programming problems and enhance problem-solving skills among students through an effective teaching approach. Therefore, the pilot study has been carried out using a traditional teacher-centred approach and to understand its effectiveness among students through a pre-test and post-test. It raised some issues and shortcomings that needed to be rectified before conducting

the feasibility study. A detailed overview of the pilot study has been included in the [Modelling and Workshop Design](#) chapter.

3.3 Create PSA Model

A problem-solving approach (PSA) is a model that has utilized the core concepts of computational thinking to solve a programming problem and will later be integrated into the student-centred active learning approaches to teach the students. The discussions in the literature review established the use of various CT concepts in problem-solving approaches, and has guided towards establishing this model. The [modelling and design](#) chapter has illustrated the PSA model and explained the construction and procedure of the model.

3.4 Modelling and Notation

Different problems have been modelled in this step in accordance with the PSA model as well as notating the different steps mathematically. The design of the workshops has been done using these problem-specific models which were then used for designing the workshops.

3.4.1 Model the problems

The [Related Work](#) section mentioned that a problem-solving approach model that included the CT concepts was designed without much relationship between them (Palts & Pedaste, 2020). Another paper found the development of CT concepts through activities in phases through flowcharts (Threekunprapa & Yasri, 2020) but an overall CT process was not utilized but measured the concepts one by one. The necessity of a relational approach within the concepts of CT has already been mentioned. The PSA model has been designed using core CT concepts by connecting them to each other, resulting in an algorithmic solution. The PSA model helps us to formulate a solution to the algorithmic problems for different sorting algorithms. The model is used for some

important sorting techniques in this paper. How the model works, and its entire process has been further elaborated in the [Modelling and Workshop Design](#) chapter.

3.4.2 Notate the CT steps

Each step that has been carried out as the model has been notated mathematically for an easier representation so that the terms of the steps become more precise. Notating steps in an easy and clear format can make arguments and logics easy to realize (Bilech et al., 2015). It is more of a symbolic representation of what actually happens in the process and enables a better understanding.

3.4.3 Design the workshops

The design of the workshops has included these things when the PSA model has been utilized in the student-centred approach by using the PSA model of different algorithms and then utilizing those models in different active learning approaches. The PSA model of a specific algorithm was modified to fit in the procedure of the specific workshop. The modified model was then translated to the purposed activity of that algorithm to solve the problem (Please refer to [Modelling and Workshop Design](#)). Each step in the activity was later related to the required programming workable syntaxes and semantics (function calls and iterations).

3.5 Experimental Design and Participants

The research involves 90 participants which were divided into three groups, each consisting of at least 30 participants. The three groups are: syntax-based learning, problem-based learning, and game-based learning. A pre-test and post-test consisting of basic programming questions were given to the participants that have given a numerical evaluation of their performance and their progression. However, in all these cases, during participant recruitment, participants were encouraged to participate with little programming background rather than having zero knowledge in programming.

3.5.1 Syntax-based learning

The first group attended a workshop that was making use of a teacher-centred method of teaching programming using syntax. The topics that were covered during this workshop were indexing, operators, loops, data types etc., using python language. A pre-workshop test and post-workshop test were conducted to evaluate student progression.

3.5.2 Problem-based learning

The second group attended another workshop and was taught programming using online problem-based learning, and the problems involved in the workshop were outlined based on our PSA model. Students took part in activities for solving different sorting algorithms in a computational problem-solving process. The sorting algorithms included: bubble sort, counting sort, and merge sort. A pre-workshop test and post-workshop test were conducted to evaluate student progression.

3.5.3 Game-based learning

The third group attended another programming workshop and was taught programming using online game-based learning, and the course was outlined based on our PSA model. Students played games through a gamified approach for solving different sorting algorithms in a computational problem-solving process. The sorting algorithms included: bubble sort, counting sort, and merge sort. A pre-workshop test and post-workshop test were conducted to evaluate student progression.

A detailed discussion has been carried out in the [Experimental Design](#) chapter for all three modules designed for different workshops.

3.6 Pre-Test and Post-Test

A pre-test and post-test (refer to [Appendix C](#)) consisting of basic programming questions were given to participants that will give a numerical evaluation of their performance and their progression.

Marking criteria:

The marking criteria that were used for the tests are as follows:

Syntax – 10%

Decomposition, abstraction, pattern recognition – 50%

Algorithmic – 40%

3.7 Analysis and Evaluation of the Pre-Test and Post-Test

An analysis has been carried out for the syntax-based and problem-based group. A paired sample T-test to compare between the pre-test and post-test of the completed activity has been carried out. A one-way ANOVA test has been carried out to compare the pre-test results of the three outlined workshops. Another one-way ANOVA test has been carried out to compare between the T-test results of the three outlined workshops.

The above methodology gives us an overview of the methods on how this study has been conducted. The pilot study is the very first step that has been taken to understand the difficulties in programming teaching and solving programming problems.

CHAPTER 4: PILOT STUDY AND PSA MODEL DEVELOPMENT

First, a pilot study has been carried out to evaluate the traditional teacher-centred approach by giving an idea to basic CT concepts and then teach the programming concepts.

4.1 Pilot Study to Evaluate Traditional Teaching Effectiveness

Objective: Understanding potential effectiveness of traditional teacher-centred syntax-based approach in programming.

Material:

- Microsoft PowerPoint slides (samples in [Appendix B](#)) to lecture the topics that are covered in the study.
- A brief of CT concepts and different programming concepts using sorting algorithms.
- Google Meet video calling app to provide the lecture through an online platform.
- Utilizing the website repl.it to code python syntax with the students.

Demographic:

In total, there were 3 participants who were first-year undergraduate ICT students. Their ages were between 18 to 25, and all were male participants.

Procedure:

An interview was scheduled using Google Meet software for a video call on a specific date. At the start of the online lecturing session, students were given a question paper for the pre-test with an allotted time of 30 minutes. A brief overview was presented about the topics to be covered during the lecture. A 2-hour lecture session was conducted on those topics. First, the notable computational thinking concepts were presented using a slide and students were given a brief idea about how each concept works. Subsequently, some concepts of the sorting algorithm were presented. The concepts that were presented were: Brute forces (bubble sort), Multiple ways brute forces (counting sort two ways) and

Divide & Conquer (merge sort). The procedure for teaching the sorting algorithms was in this order: the students were given an overview of how the algorithms work through examples in PowerPoint slides. After the students had a brief idea of the process, they were presented the flowchart of the respective sorting algorithm. Then they were given a live demo on how to write the codes by sharing the screen of the instructor. The students were asked to write the code along with the instructor. Next, they were given an overview of the programming concepts that have been used in the respective sorting algorithm within the codes. Upon completion of all the lecture slides (samples in [Appendix B](#)), the students were asked to do a post-test for the study with a different set of questions. This helped to evaluate the differences before and after the lecture.

Marking criteria and results:

The marking criteria that were used for the test are as follows:

Syntax – 10%

Programming concepts – 50%

Algorithmic – 40%

Appendix A includes the pre-test and post-test numerical evaluation of this pilot study. The results display that there has not been sufficient improvement as per the scores of the pre-test and post-test. One student had scored better than the pre-test, and the other two students could not improve their post-test results. It indicates that the majority of the students need a more structured way for effective learning.

Difficulties/challenges encountered:

- Students have an idea about the syntax, but they are unsure of its usage at the right time.
- Formulating the problem-solving process was quite difficult for the students.

- Students were facing difficulties to transform the sorting algorithm process into a coding process.
- Providing instructions was challenging via online media. It was difficult to collaborate with the students, e.g., any code that has been written by the student or instructor was only visible via screen share, which was not intuitive.

Observations for improvement:

- The scores indicate that the improvement of the students need further enhancement in the course materials and improve the way of teaching.
- During the teacher-centred syntax-based approach in the feasibility study, focus more on the syntax and make sure that students get a better idea of the syntax in a structured manner.
- Other than the teacher-centred approach, computational thinking concepts need to be integrated as a process for solving problems rather than just giving an idea of how it works.
- The syntax and basic concepts need to be incorporated as a part of the CT process in due time so students can relate to the problem and be able to transfer it to code.
- The programming level of the participants are beginners, and therefore, a collaborative tool as an IDE would be easier to explain the concepts and for better understanding of the students.

4.2 PSA Model and Modelling Sorting Problems

The results of the pilot study have given the idea of integrating the CT concepts as a process to solve problems. This section focuses on creating a problem-solving approach (PSA) model and use it to model different sorting problems.

4.2.1 PSA MODEL

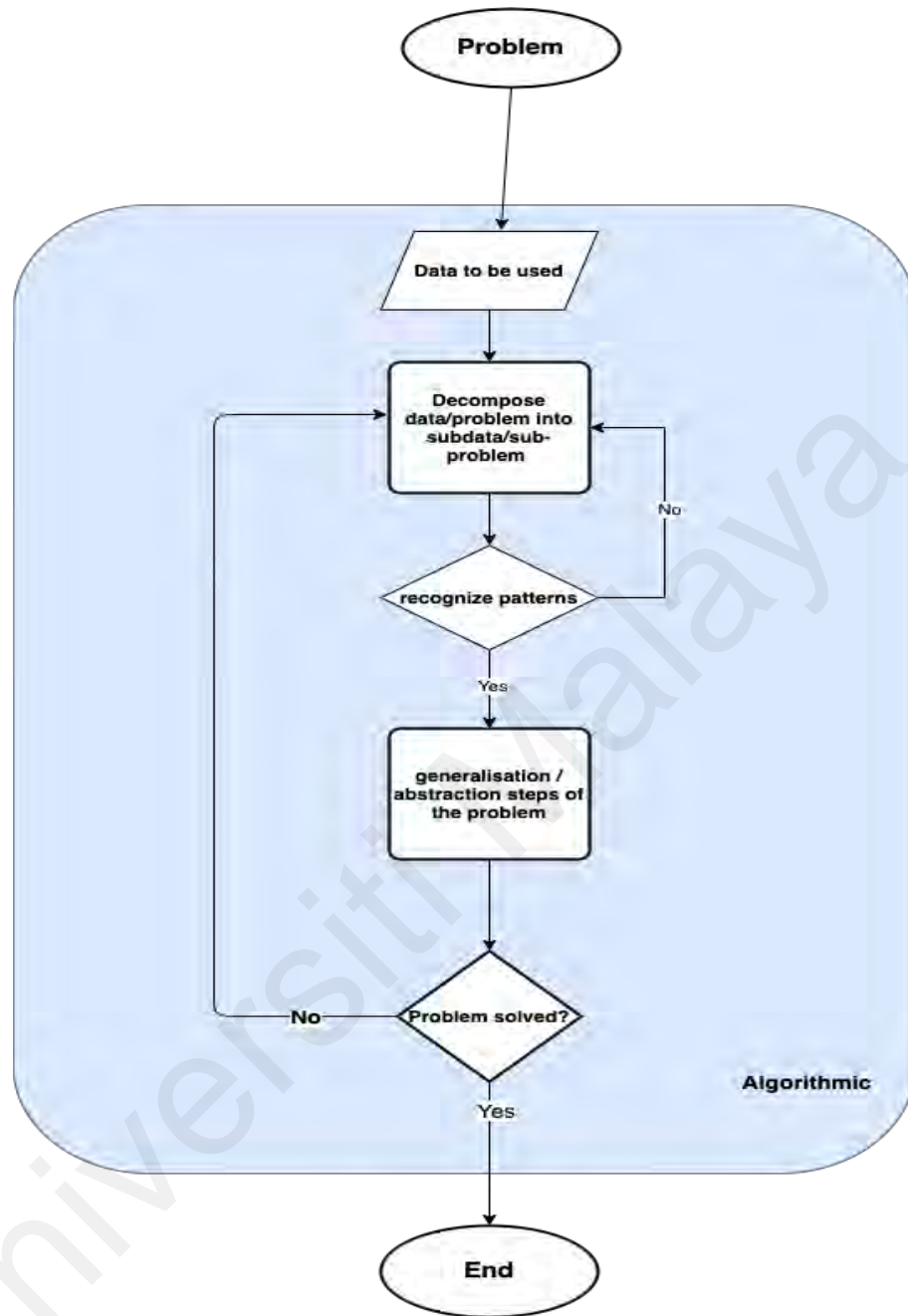


Figure 4.1: Problem-solving approach (PSA) model

From the literature review, the study has identified the core CT concepts to solve fundamental and complex problems. The five identified CT concepts are:

- i) Data
- ii) Decomposition
- iii) Pattern Recognition
- iv) Abstraction

v) Algorithmic

Figure 4.1 shows the problem-solving approach model for this research. The core computational thinking (CT) concepts are a major part of this model. First of all, it is a problem that has to be solved. So, we start with the problem. Since our aim is teaching and learning programming through this model, and it is obvious that computational thinking is a well-needed capability to be acquired, the first concept that needs to be considered is data. Data represents a problem as well as the instance of the problem (N. Miller & L. Ranum, 2006). Taking into consideration the data and the problem, we need to decompose the data into sub-data/sub-problems. Once the problem has been decomposed, we need to find the similarities or patterns among the decomposed problems. If the patterns have not been recognized or are unable to be recognized, then we need to look at our decomposition steps again. Once the pattern(s) have been recognized, then it is time to gather the general characteristics using the abstraction step. If the problem is not solved during the abstraction step, then we go back and check our decomposed sub-problems and proceed in solving the problem using the next step once the problem has been solved, then it is the end of the process. It is an algorithmic process that enables to generate an algorithmic solution from the beginning until the end. The algorithmic process comprises of decomposition, pattern recognition and abstraction through an interconnection. The iterative flow does not add another step for algorithm like a general CT approach but involves decomposition, pattern recognition, and abstraction as a whole algorithmic process. This iterative flow is based on the sequencing of the CT concepts where the data is identified first and then decomposed into smaller sub-problems followed by finding the similarities or dissimilarities, extract unnecessary characteristics, then avoid repetitions and processes in an algorithmic manner until the intended solution of the problem has been achieved.

4.2.2 Sorting problems using PSA model

A number of sorting algorithms, which is a domain for this research, and are used as problems to be solved are given below. A number of algorithms have been approached by applying the model above.

4.2.2.1 Bubble sort

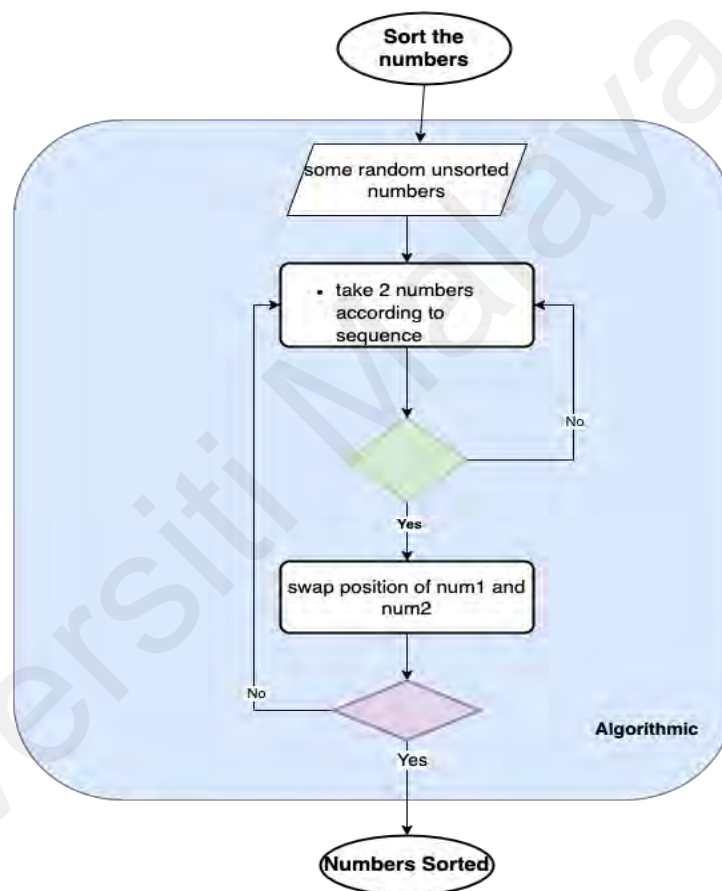
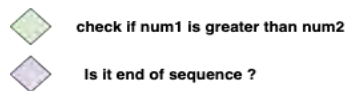


Figure 4.2: Bubble sort using PSA model

Mathematical Notation:

Data:

Let the sequence of numbers be denoted by, $X = \{ a_1 , a_2 , a_3 \dots a_n \}$

Decomposition:

And two of the sequential numbers from above denoted by,

$$\bar{a}_i = \{ a_i , a_{i+1} \} \text{ where } i \leq n - 1$$

Pattern recognition:

After decomposing the smallest sub-problem by taking 2 numbers and identifying pattern, if $a_i > a_{i+1}$

Abstraction:

$$\bar{a}_i = \begin{cases} \{a_{i+1}, a_i\} & \text{if } a_i > a_{i+1} \\ \{a_i, a_{i+1}\} & \text{else} \end{cases}$$

∴ To sort X into ascending order repeat $\bar{a}_i \forall X$ that can be denoted by a function S that takes value of X ; S(X)

$$S(X) = (\bar{a}_j)_{i,j}^n, \text{ for each } i \leq n - 1 \text{ there are values of } j \leq n - i - 1$$

Implementation:

```
def BubbleSort (Array):
```

```
size = len(Array)
```

```
for i in range(0,size-1) :
```

```
for j in range(0 , size - i - 1):
```

```
if Array[j] > Array[j+1] :
```

```
Array[j],Array[j+1] = Array[j+1],Array[j]
```

```
    endIF
```

```
    endFor
```

```
endFor
```

```
endFunction
```

```
Array = [10,7,2,1,3]
```

```
BubbleSort (Array);
```

4.2.2.2 Counting sort

◇ are all indexes placed as many times as count?

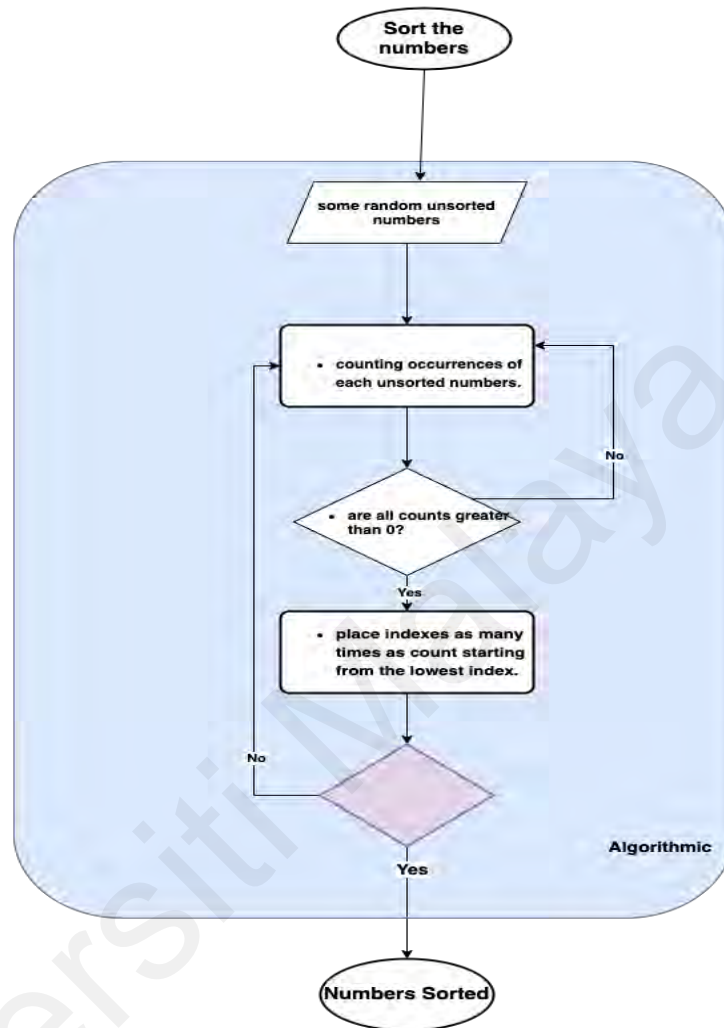


Figure 4.3: Counting sort using PSA model

Mathematical Notation:

Data:

Let the unsorted sequence of numbers be denoted by, $X = \{ a_1, a_2, a_3, \dots, a_n \}$

And biggest number in X is a_h where $0 \leq h \leq n$

Decomposition:

Let, decomposing each number by counting the occurrence in X, denoted by C_i ,
 where $i \leq h$

$$C_i = \{ c_i, c_{i+1}, c_{i+2}, \dots, c_h \}$$

Pattern recognition:

$$\forall C_i > 0$$

Abstraction:

Identifying pattern $\forall C_i > 0, y = (\bar{i})_0^e$

∴ In order to place the counts accordingly, repeat y until biggest number that can be denoted by a function S that takes value of C_i ; $S(C_i)$

$$S(C_i) = (\bar{y})_i^h$$

Implementation:



```
def CountSort (Array , max):
    size = len(Array)
    newArray = []
    m = max + 1
    count = [0] * m
    for i in range(0 , size):
        count[Array[i]] = count[Array[i]] + 1
    endfor
    for i in range(len(count)):
        if count[i] > 0:
            for j in range(count[i]):
                newArray.append(i)
            endfor
        endif
```

```

endfor
for i in range(0 , size):
Array [i] = newArray [i]
    endfor
endFunction
Array = [10,7,2,1,3]
CountSort(Arr, max(Array))

```

4.2.2.3 Merge sort

-  are there only one number in all the divided groups?
-  are all divided groups merged?

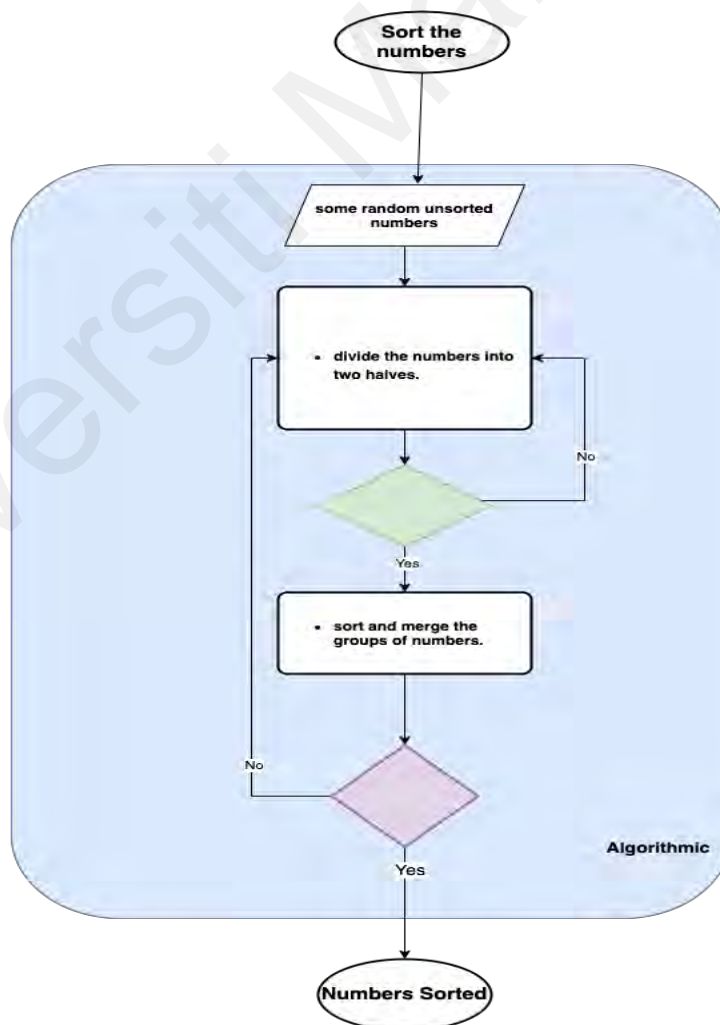


Figure 4.4: Merge sort using PSA model

Mathematical Notation:

Data:

Let the unsorted sequence of numbers be denoted by, $X = \{ a_1 , a_2 , a_3 \dots a_n \}$

Decomposition:

And $X(n)$ denotes the function with n elements.

$X(n) = X(n/2)$ divides the elements into 2 halves where values of n decrease until $n = 1$.

Repeatedly, dividing it into 2 halves decomposing to the smallest unit, denoted by X_s

$$X_s = (X(n))_n^1 = \{X_n, X_{n-1}, \dots, X_1\},$$

where each element represents a set of the smallest sub-division.

Pattern recognition:

Identifying Pattern, $\forall | X_s | = 1,$

Abstraction:

Let two groups to be compared from X_s denoted as X_i and X_{i+1} .

Lowest number to be placed first for sorting and can be denoted as,

$$X_{\min} = \{ \min(L_i + L_{i+1}) \}$$

Sorting the divided groups and merging them can be denoted as,

$$X_m = \bar{X}_{\min}$$

\therefore In order to finally merge all the groups, repeat X_m and finally sort them accordingly

and can be denoted by a function S that takes value of X_m ; $S(X_m)$

$$S(X_m) = (\bar{X}_m)_i^{n-1}$$

Implementation:

```
def MergeSort (Array):
```

```
if( len(Array) > 1 ):
```

```
mid = len(Array) // 2
```

```
L = Arr[0 : mid]
```

```
R = Arr[mid: ]
```

```
MergeSort (L)
```

```
MergeSort (R)
```

```
i = j = k = 0
```

```
while (i<len(L) and j<len(R)):
```

```
if (L[i] < R[j]):
```

```
    Arr[k] = L[i]
```

```
    i = i + 1
```

```
else
```

```
    Arr[k] = R[j]
```

```
    j = j + 1
```

```
    endif
```

```
    k = k + 1
```

```
    endwhile
```

```
while (i<len(L)) :
```

```
    Arr[k] = L[i]
```

```
    i = i + 1
```

```
    k = k + 1
```

```
    endwhile
```

```
while (j<len(R)) :
```

```
    Arr[k] = R[j]
```

```
    j = j + 1
```

```
    k = k + 1
```

```
    endwhile
```



```
endFunction
```

Array = [10,7,2,1,3]

MergeSort (Array)

Universiti Malaya

4.2.2.4 Quick sort

-  does pivot have smaller and greater number(s) in the sequence?
-  Is it end of sequence with the pivot in its rightful place?

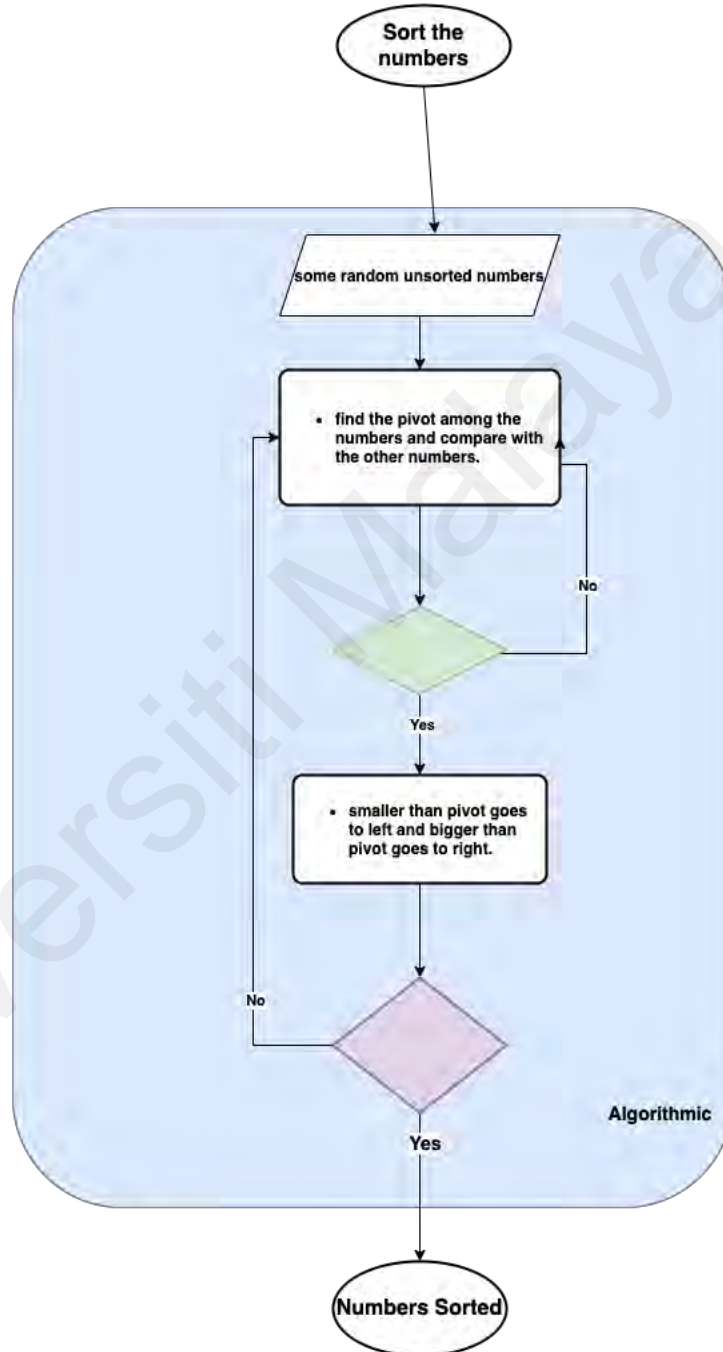


Figure 4.5: Quick sort using PSA model

Mathematical Notation:

Data:

Let the sequence of numbers be denoted by, $X = \{ a_1, a_2, a_3, \dots, a_n \}$

Decomposition:

Let a_p be the pivot of the sequence.

$X(n)$ denotes the function with n elements.

$X(n) = X(n - 1)$ denotes the number of comparisons against the pivot where values of n decreases until $n = 0$.

Let $X_L = \{ \forall X, a_i < a_p \}$ and $X_R = \{ \forall X, a_i > a_p \}$ where $i \leq n$

Pattern recognition:

Decomposing the numbers by comparing with the pivot and identifying pattern,

$\forall X, X_L < a_p < X_R$

Abstraction:

$$\bar{a}_i = \begin{cases} a_i \in X_L & \text{if } a_i < a_p \\ a_i \in X_R & \text{else if } a_i > a_p \end{cases}$$

\therefore To sort X into ascending order repeat $\bar{a}_i \forall X$ that can be denoted by a function S that

takes value of X ; $S(X)$

$$S(X) = (\bar{a}_i)_{i=n}^0$$

Implementation:

def Partition (array, low, high):

$i = \text{low} - 1$

pivot = array [high]

for j in range (low, high):

if array [j] < pivot :

$i = i + 1$

array [i + 1], array [j] = array [j] , array [i + 1]



endIF

array [i+1] , array [high] = array [high] , array [i + 1]

```
return (i+1)
    endFor
endFunction
def QuickSort (array, low, high):
if low < high:
pi = Partition (array, low, high)
QuickSort (array, low , pi - 1)
QuickSort (array, pi + 1 , high)
    endIF
endFunction
Array = [10,2,9,2,1]
QuickSort (Array, 0, len(Array) - 1)
```

Universiti Malaya

4.2.2.5 Bucket sort

-  are the numbers in respective buckets and empty buckets ignored?
-  are all sorted bucket elements added in correct sequence?

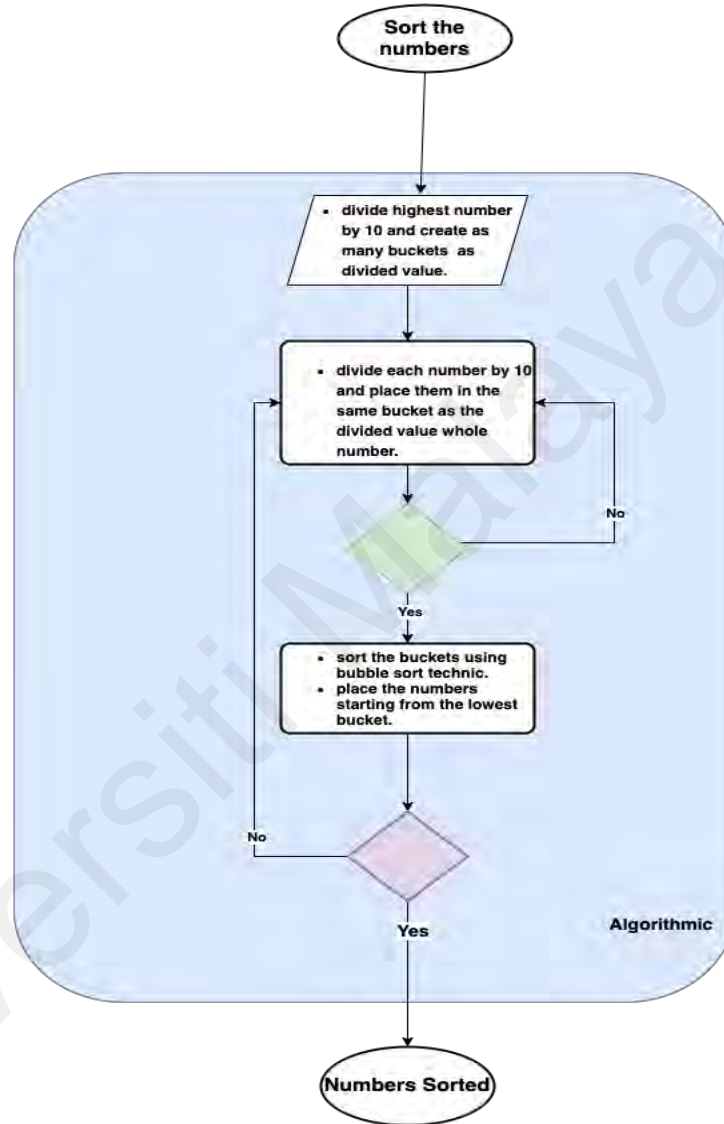


Figure 4.6: Bucket sort using PSA model

Mathematical Notation:

Data:

Let the sequence of numbers be denoted by, $X = \{ a_1 , a_2 , a_3 \dots , a_n \}$

Decomposition:

Let the maximum number in X be denoted by, a_m

Number of buckets required can be denoted as, $h = a_m / 10$

Let the sequence of Buckets be denoted by, $B = \{B_0, B_1, B_2, \dots, B_h\}$

Each number in the sequence X to be divided by 10 can be denoted by,

$X_d = \{i_1, i_2, i_3, \dots, i_n\}$ where i is the whole number before decimal point.

Each bucket in B can be denoted as $B_i = (\bar{i})_{i=0}^n$ where $i \in X_d$ and index of bucket.

$\forall X_d, y = (\bar{B}_i)_{i=0}^n$

Pattern recognition:

$\forall |B| > 0, B = (\bar{B}_s)_0^n$

Where each bucket will follow bubble sort method to sort the buckets individually and sorted buckets can be denoted as,

$B_s = \{s_1, s_2, s_3, \dots, s_n\}$

Abstraction:

∴ To sort X into ascending order add all elements of $\forall |B| > 0$ that can be denoted by a

function S that takes value of B ; $S(B)$

$S(B) = \{B_0 + B_1 + B_2 + \dots + B_n\}$

Implementation:

u₁₀ → def BucketSort (array):

u₁₁ → bucket = []

u₁₂ → for i in range(len(array)):

u₁₃ → bucket.append([])

endfor

u₁₄ → for j in array:

u₁₅ → bindex = int(10 * j)

u₁₆ → bucket[bindex].append(j)

endfor

u₁₇ → for i in range(len(array)):

u₁₈ → bucket[i] = BubbleSort (bucket[i])

```

    endfor
u19 → k = 0
u20 → for i in range(len(array)):
u21 → for j in range(len(bucket[i])):
u22 → array [k] = bucket [i][j]
    endFor
u23 → k = k + 1
    endFor
endFunction
u25 → Array = [10,7,2,1,3]
u26 → BucketSort (Array)

```

The above sorting problems have been modelled using the PSA model. The problem-solving process using the PSA model for different sorting problems has paved the way to solve programming problems in different active learning techniques such as PBL and GBL. The next chapter will focus on the experimental design and use PSA models for different sorting problems to design the workshops.

CHAPTER 5: EXPERIMENTAL PLANNING AND DESIGN

The research involves 90 participants that were divided into three groups, each consisting of at least 30 participants. The three groups are: syntax-based learning, problem-based learning and game-based learning. A pre-test and post-test consisting of basic programming questions were given to participants that have given a numerical evaluation of their performance and their progression. However, in all these cases, during participant recruitment, participants were encouraged to participate with little programming background rather than having zero knowledge in programming.

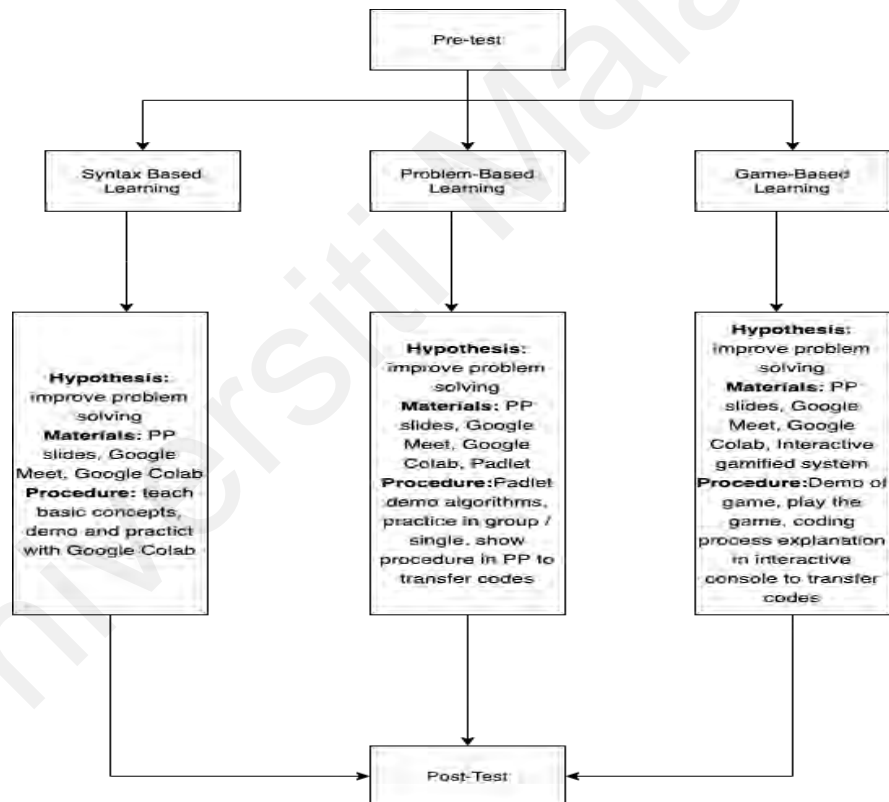


Figure 5.1: Experimental planning of the study

5.1 General Process for Feasibility Study

First of all, participants were recruited using a Google Form where they had to undergo a mandatory pre-test and solve questions related to programming. Among the workshops, syntax-based workshop was conducted first. Among the questions, basic programming

concepts has to be included and the marking criteria that has been followed is the same as in the pilot study. For each workshop, there were different Google Forms being handed out but with the same questions, so that it would be easy to keep track of the participants in each workshop. Once the students' registration due date finished, they were allocated a specific time and schedule to attend the workshops using Google Meet software. Once the workshop was successfully conducted, a post-test was conducted with the same pattern of questions as the pre-test but there were different problems to be solved. Same as the pre-test, the post-test questions were also handed out via different Google Forms but with the same questions in all of them so that it would be easier to check the answers of different groups later.

5.2 Syntax-Based Workshop

Hypothesis: Syntax-based teacher-centred programming learning is beneficiary to improve problem-solving skills.

Materials:

- Microsoft PowerPoint Slides to explain the basic programming concepts.
- Google Meet software for conducting a hassle-free online class.
- A collaborative online python compiler named "Google Colab" where code changes can be displayed in real time.

Demographic:

Number of people: 30

Age: 29 participants were aged between 18-25, 1 participant was above 40

Gender: male: 16, female: 14

Educational background: First year and pre-university ICT students, non-ICT related professionals

Workshop procedure: Basic concepts of programming were covered using Microsoft PowerPoint slides (refer to [Appendix C](#)). A small introduction was given about python and then topics that were covered are: Strings, operations, Functions, methods, numeric data, operators and mathematical functions, List/Array, Conditional & Logical operators, and Loops. Each topic was covered one by one and in the process, demonstrations were given to see how the syntax works and how to write them. An online python compiler named ‘Google Colab’ was utilized to give a demonstration to the students and they were encouraged to write chunks of code in the process. This is the most usual way of teaching programming, and this group is the control group of the study.

5.3 Problem-Based Learning Workshop

Initially the activity-based process was thought to be used in a classroom by using playing cards and combining students in a group. Since the data collection process has been undertaken during the Covid-19 pandemic, the process had to be changed. An online yet trackable approach had to be taken into consideration for this.

Hypothesis: Problem-based programming learning by utilizing the PSA model is beneficial to improve problem-solving skills.

Materials:

- Microsoft PowerPoint Slides explaining a step-by-step computational approach to transfer the problem-solving steps into programming concepts.
- Google Meet software for conducting a hassle-free online class.
- Padlet software to conduct sorting activities online and check how students were approaching the problem.
- A collaborative online python compiler named “Google Colab” was used where the sorting algorithms codes were displayed and run.

Demographic:

Number of people: 30

Age: 25 participants were aged between 18-25, 3 participants were aged between 25-30, and 2 participants were aged between 30-40

Gender: male: 22, female: 8

Educational background: First year and pre-university ICT students, non-ICT related professionals

Workshop procedure: An online-based tool called Padlet was chosen in order to enable the students to participate in this workshop. The students were recruited the same way as the syntax-based workshop. Three sorting algorithms that were introduced in this study were: bubble sort, counting sort, and merge sort. First of all, a demo was given on how to solve a sorting process using the padlets (refer to [Appendix B](#)). The algorithmic process was taught in accordance with the PSA model of the study. Then the students proceeded in doing the activity. The students were given the idea through the slides (refer to [Appendix F](#)) on how they have gone through the different stages of computational thinking in the process and at the same time, how to transfer the problem-solving process to the programming concepts. In each case, the programming concepts were explained in detail so that the students get used to the idea on how to transfer the computational thinking elements to programming concepts. The explanation proceeded as per the steps in the PSA model and the programming concepts were brought in accordance with it. Furthermore, Google Colab was used to give the working codes of the algorithms to the students. If students had to be taught any specific programming concepts separately, Google Colab was the go-to tool.

5.4 Game-Based Workshop

Hypothesis: Game-based programming learning by utilizing PSA is beneficial to improve problem-solving skills.

Materials:

- Microsoft PowerPoint Slides explaining different concepts of programming in case they were needed.
- Google Meet software for conducting a hassle-free online class.
- Interactive gamified system for sorting numbers which allowed users to solve algorithms in a gamified environment and illustrated a console-like behaviour to teach programming concepts.
- A collaborative online python compiler named “Google Colab” where the sorting algorithms codes were displayed and run.

Demographic:

Number of people: 30

Age: 18-25

Gender: male: 17, female: 13

Educational background: First year and pre-university ICT students.

Workshop procedure: The proposed game-based system was utilized in this procedure.

First of all, an algorithm was selected, and participants were instructed on how to play the game. Then they were given some time to play the games in 3 different levels. Following which, a demonstration was given on the interactive view about how the coding process works while they were playing the games. Furthermore, Google Colab was used to provide the working codes of the algorithms to the students. If students needed to be taught any specific programming concepts separately, Google Colab was the go-to tool.

5.5 Design of the workshop materials

A few issues that were addressed during the [pilot study](#) were taken into consideration, such as the difficulties faced and the observations of the study. The three workshops that have been identified in the [methodology](#) section included: syntax-based learning

workshop, problem-based learning workshop, and game-based learning workshop. The syntax-based learning workshop is actually the teacher-centred approach and focuses more on the syntax. The participants in this workshop fall under our control group. The other two groups were working in a student-centred active learning approach. The problem was the main focus and used our CT model, and the course materials have been designed. Furthermore, for the ease of the students and the workshop, python has been utilized as the programming language.

5.5.1 Teacher-centred syntax-based workshop design

This workshop focuses on the usual way of learning programming which is a syntax-based approach. Most of the design was done by focusing on the syntax approach from the basics up to the harder levels. First of all, a definition of the selected programming language has been given. Different data types such as string, integer, and float has been discussed. Basic concepts such as string and operations using string such as add, concatenate, and repetition was discussed. Following which, indexing and slicing in strings were discussed. In built functions such as length and type are explained with regards to string. Split, join, upper, lower, and other methods were discussed next. Once string concepts have been completed, harder concepts like numeric data and their functions were introduced along with operators and math functions. After all these concepts, students were brought into the topic of List/Array. A definition had been given about this topic. Then, topics like string were brought within List/Array such as indexing and slicing, operators, functions, and methods. Once this was done, conditional statements and logical operators were taught to students. Upon completion, loops were taught to students, e.g., for loop and while loop. Finally, the last topic that was covered in the workshop was how to define a function, pass arguments, and call the function. Appendix E has provided some of the slides that will give an overview on the design of this workshop through the PowerPoint slides.

5.5.2 Problem-based workshop design

This workshop was designed for one of our experimental groups that participated in a problem-based learning environment through an activity-oriented approach. The developed PSA model was our blueprint for this workshop. Students learned different concepts of programming by participating in solving sorting algorithmic problems. The sorting algorithms that were used in the workshop included: bubble sort, counting sort, and merge sort. A CT-based solution using our PSA model has already been formulated for all these algorithms (please refer to [bubble sort](#), [counting sort](#), [merge sort](#)). These formulated solutions were modified to fit in the activities that was used to design the activities of these algorithms in Padlet (refer to Appendix B). After a single activity was completed, the steps in the activity were related to the programming syntaxes associated to the sorting technique.

5.5.2.1 Bubble sort in PBL

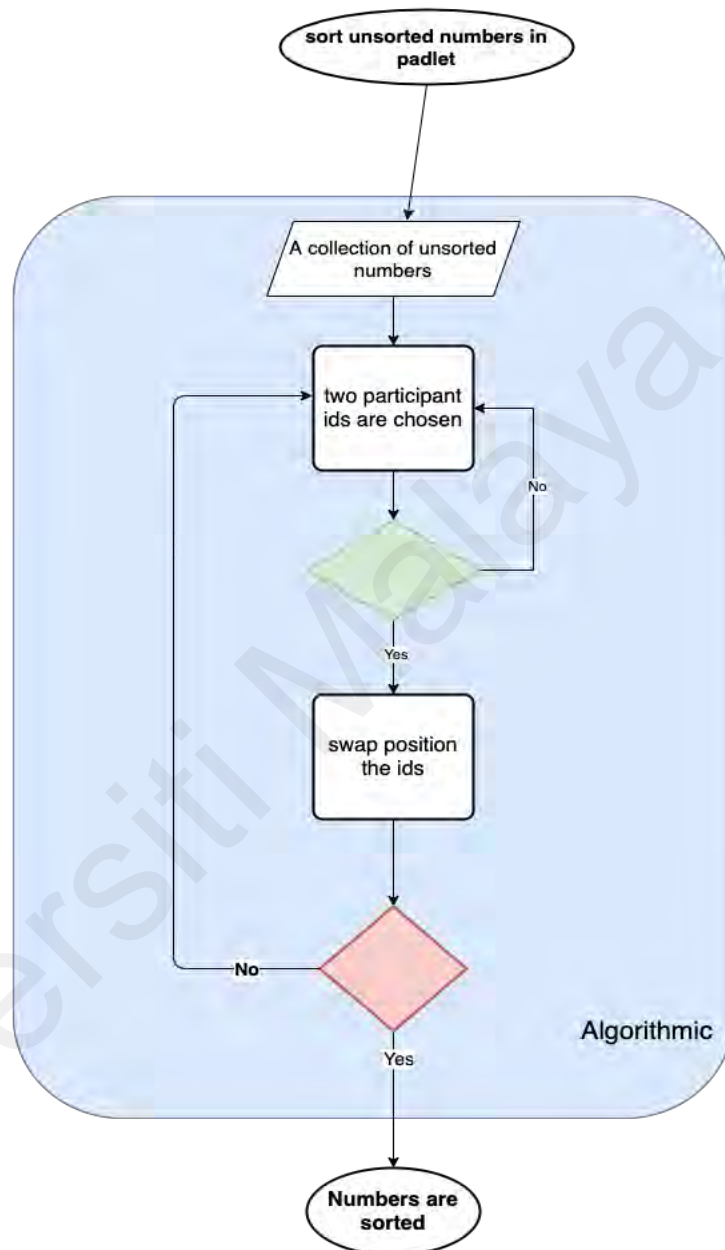
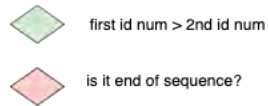


Figure 5.2: Bubble sort using PSA for PBL

The [Bubble sort problem-solving](#) process using the PSA model has been utilized to formulate the process of the bubble sort activity in this workshop. This model has been the guideline to design the activity in Padlet for bubble sort.

(a) *Transfer the model into activity:*



Figure 5.3: Unsorted numbers with id 1 and id 2 chosen



Figure 5.4: id 1 > id 2 and swapped



Figure 5.5: Sort all numbers following this process

The above figures illustrate how the PSA model was translated into activity. Id 1 and id 2 has been chosen and they have to identify if the first id is greater than the second id. If it is true, then they swap their places and if not, they do not swap. For example, id 1 and id 2 has been swapped. In the next step, id 1 and id 3 will be compared in the same fashion and this process keeps going on until the last number. If all numbers are not sorted,

they start from the beginning, choose two numbers, and follow the same process from the beginning.

(b) *From activity to code:*

How to initialize variables?

```

• a = 12
• b = 4
• array = [ 12, 4, 45, 1, 9,3]
#array[0] = 12
#array[1] = 4
•

```

Figure 5.6: Define array with collection of numbers and index

How to transfer this into code?

if num1 > num2:

num1, num2 = num2, num1



Figure 5.7 Check condition to swap

What is the next step to solve this problem?



size = len(array)

• Initialize length of array as size using len function.

for i in range(size-1):

• A loop to complete all the steps

for j in range(0, size - i - 1):

• Sorting the numbers using a nested loop.





Figure 5.8: How loops are used to iterate through the array

With each step of the activity, the codes are related with a motive to transfer the problem-solving approach into programming syntax. The concept of data is represented through an array. Two numbers are compared through an if else statement using the index of an array. To identify the index of an array and comparing all the numbers, the concept

of for loop is brought to light. Furthermore, how many rounds the number has to be compared is also explained by relating to the activity where the participants incremented the steps (Figure 5.5) and write completed whenever the local biggest number is in the right place.

5.5.2.2 Counting sort in PBL

-  available unsorted occurrences greater than 0?
-  are all index with count > 0 placed in final?

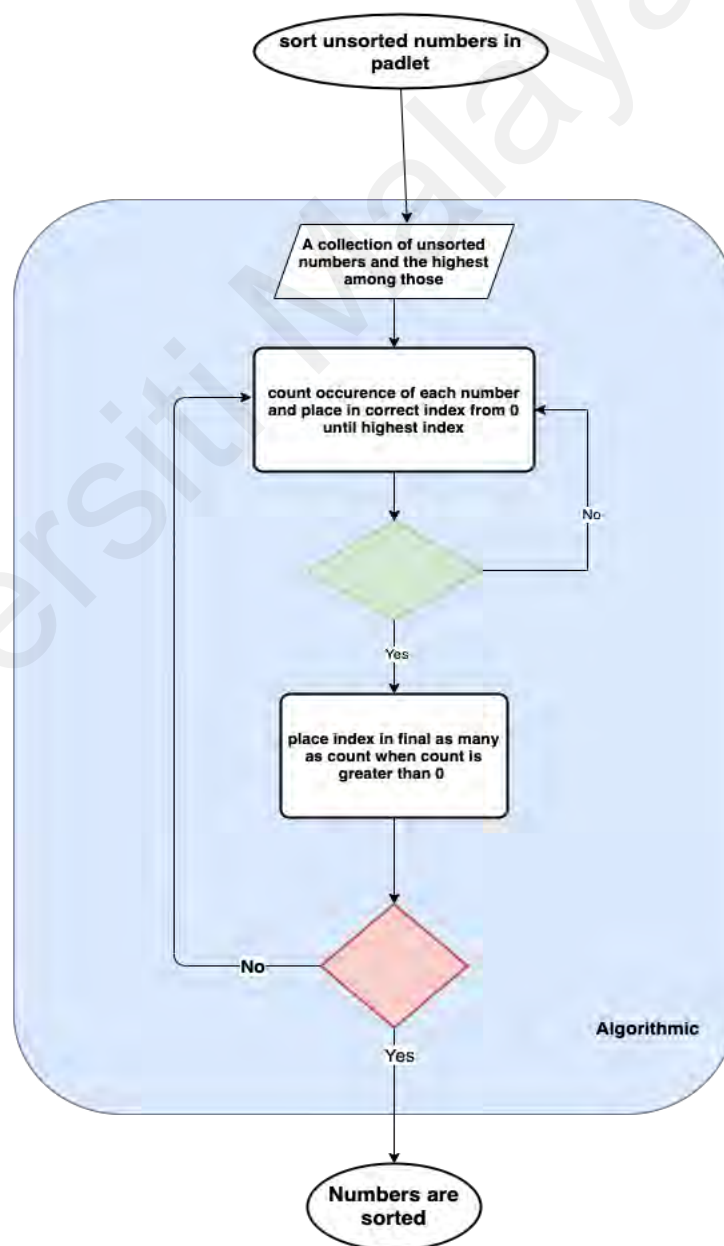


Figure 5.9: Counting sort using PSA for PBL

The [counting sort problem-solving](#) process using the PSA model has been utilized to formulate the process of the counting sort activity in this workshop. This model has been the guideline to design the activity in Padlet for counting sort.

(a) *Transfer the model into activity:*

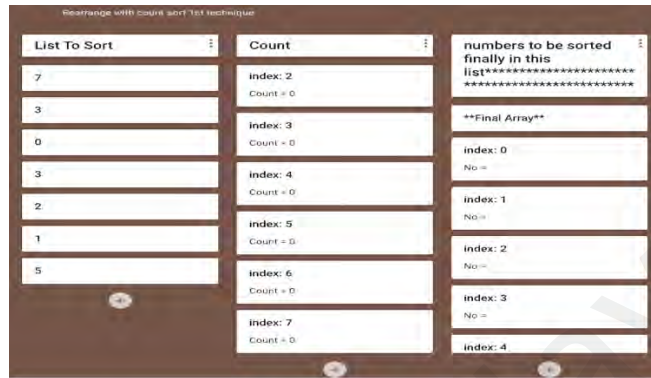


Figure 5.10: A collection of unsorted numbers in listToSort and count from 0 until highest index

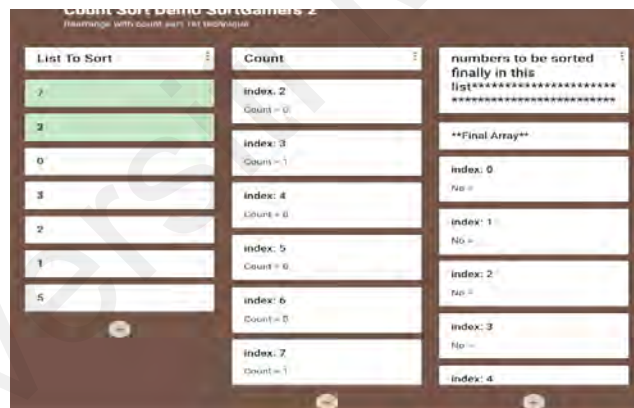


Figure 5.11: Count occurrence of each number and add in correct index

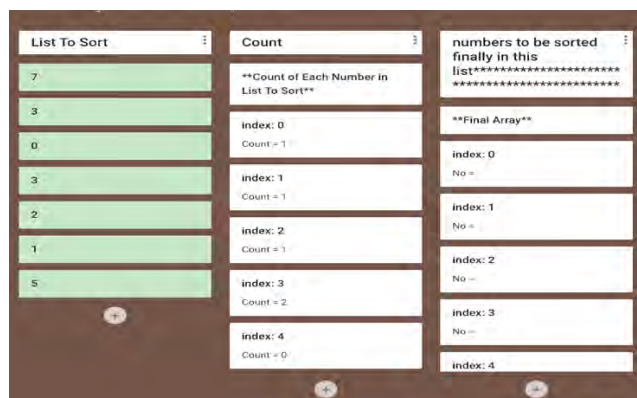


Figure 5.12: Count of all available numbers in listToSort greater than 0

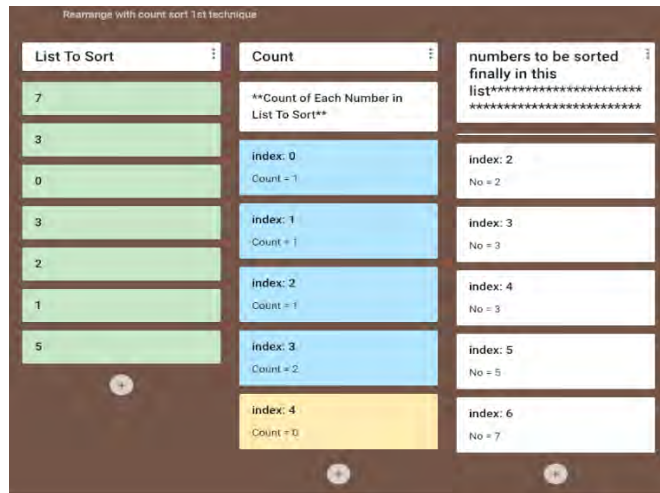


Figure 5.13: Adding to final list and ignore count 0

Figure 5.10 illustrates a collection of unsorted numbers in listToSort and using the highest number, a collection of count boxes has already been created so students understand that the count boxes are created according to the highest number. Figure 4.16 illustrates how the count is increased based on the numbers available in listToSort by increasing the count of an index. Upon counting one item, that item is colored. Once the count(s) of the listToSort numbers are all greater than 0, the numbers are then appended to final array as many times as the count (Figure 5.13).

(b) From activity to code:

How do you start to sort these numbers ?

```
data = [5, 0, 2, 3, 2, 5, 4]
```

- We need to store the numbers as a collection of elements such as an array named as data.
- Count occurrence of every number in the data.
- Storing the indexes of the count in another array.

Index	0	1	2	3	4	5
Count	1	0	2	1	1	2




Figure 5.14: Initialize an array named data

How to transfer this into code?

```
def countingSort(listToSort, max):
```

- Define a function countingSort that will receive the data as listToSort and max number in data.

```
count = [0] * (max + 1)
```

- Initialize count array which will have "max + 1" number of 0.

Index of Count	Index: 0	Index: 1	Index: 2	Index: 3	Index: 4	Index: 5
Count	0	0	0	0	0	0

Figure 5.15: Define function and create count array

So How do we populate count array?

listToSort	5	0	2	3	2	5	4
Count	0	0	0	0	0	0	0

i = 0

```
listToSort[0] = 5
Count[5] = 0 + 1 = 1 #Count[listToSort[0]]
```

i = 1

```
listToSort[1] = 0
Count[0] = 0 + 1 = 1 #Count[listToSort[1]]
```

i = 2

```
listToSort[2] = 2
Count[2] = 0 + 1 = 1 #Count[listToSort[2]]
```

Click to add text

Index of Count	Index: 0	Index: 1	Index: 2	Index: 3	Index: 4	Index: 5
Count before populate	0	0	0	0	0	0
count[listToSort[i]] += 1	1	0	2	1	1	2

```
size = len(listToSort)
Initialize size using len function to get length of listToSort.
for i in range(0, size):
    count[ listToSort[i] ] = count[ listToSort[i] ] + 1
Loop as many times as size and populate count array
```

Figure 5.16: Keep count of listToSort

Index of Count	Index: 0	Index: 1	Index: 2	Index: 3	Index: 4	Index: 5
Count	1	0	2	1	1	2

index = 2
count[index] = 2

```
for j in range(2):
    finalArray = [0, 2, 2]
```

index = 3
count[index] = 1

```
for j in range(1):
    finalArray = [0, 2, 3]
```

index = 4
count[index] = 1

```
for j in range(1):
    finalArray = [0, 2, 3, 4]
```

index = 5
count[index] = 2

```
for j in range(2):
    finalArray = [0, 2, 2, 3, 4, 5]
```

How do we append the indexes to final array?

Index of Count	Index: 0	Index: 1	Index: 2	Index: 3	Index: 4	Index: 5
Count after populate	1	0	2	1	1	2

```
finalArray = []
Initialize finalArray to arrange the items into ascending order.

if count[0] != 0:
    finalArray.append(0)
    Adds the element to the final Array using append method.



But what if count is more than 1?
if count[0] != 0:
    for j in range(count[0]):
        finalArray.append(0)
    A loop to add index as many times as count
```

Figure 5.17: How to append to final array

Figure 5.14 illustrates the concept of array by using the collection of elements and how to define a function to pass the highest number of the array. Figure 5.15 shows how the count array is produced based on the highest number. Figure 5.16 shows in code how to

keep count of the numbers available in listToSort, and Figure 5.17 illustrates how the count indexes are appended to array to finally sort the list. Finally, calling of a function is illustrated that passes the data to the defined function which is the listToSort inside the function and using max concept to find the highest number in an array.

5.5.2.3 Merge sort in PBL

-  all divided group has only one id?
-  are all divided groups merged back?

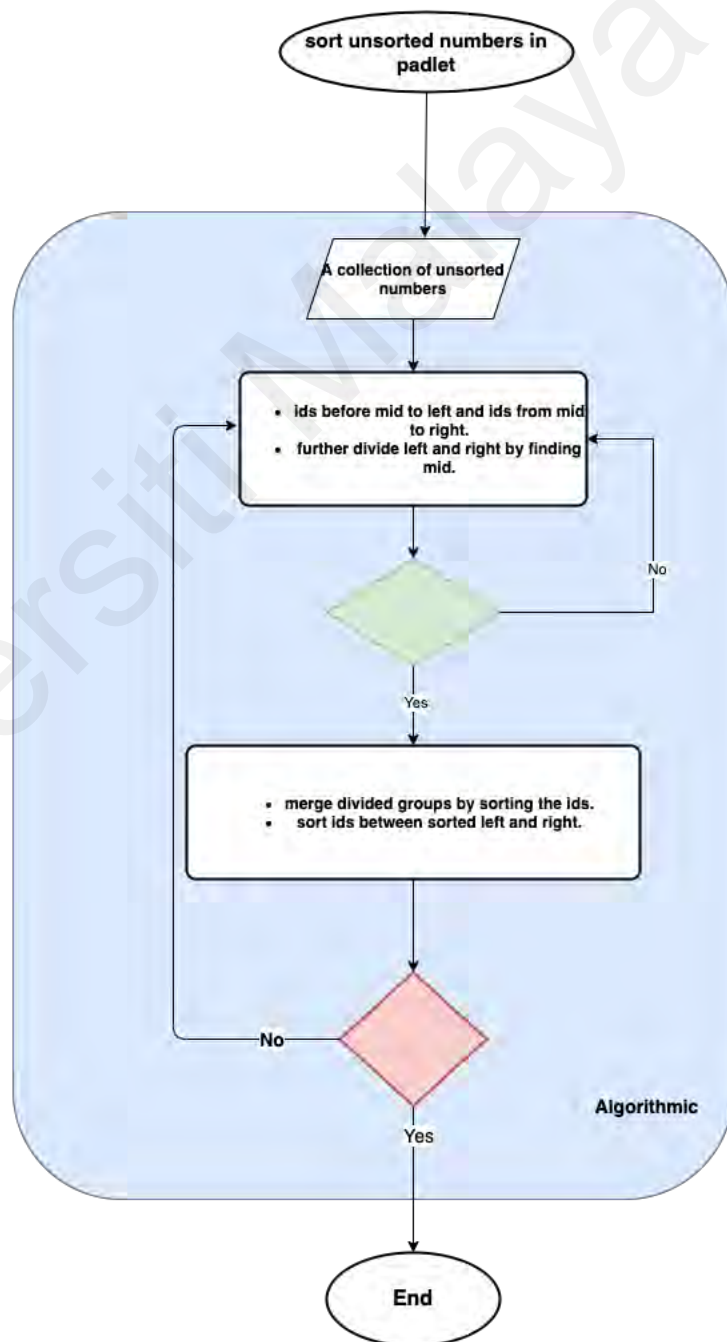


Figure 5.18: Merge sort using PSA for PBL

The [Merge sort problem-solving](#) process using the PSA model has been utilized to formulate the process of the merge sort activity in this workshop. This model has been the guideline to design the activity in Padlet for merge sort.

(a) *Transfer the model into activity:*



Figure 5.19: Collection of unsorted numbers in Padlet

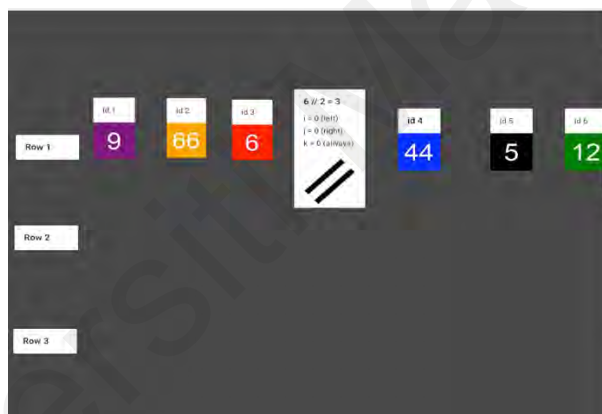


Figure 5.20: Left and right ids are divided finding mid

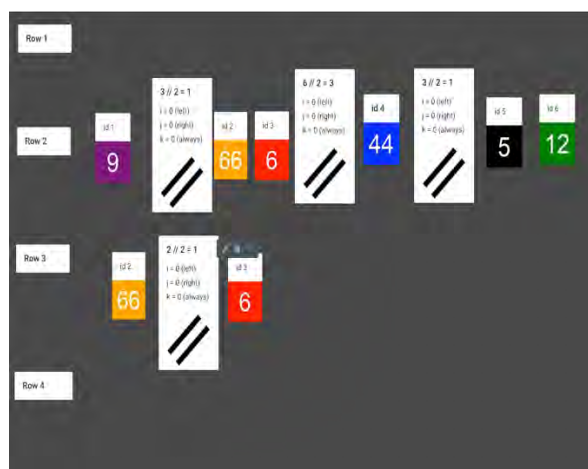


Figure 5.21: Further divide between left side

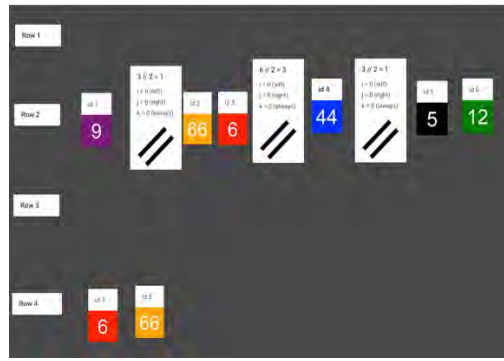


Figure 5.22: Sort and merge left side

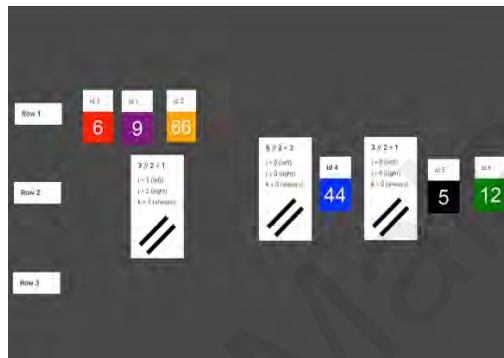


Figure 5.23: Further divide right, sort and merge

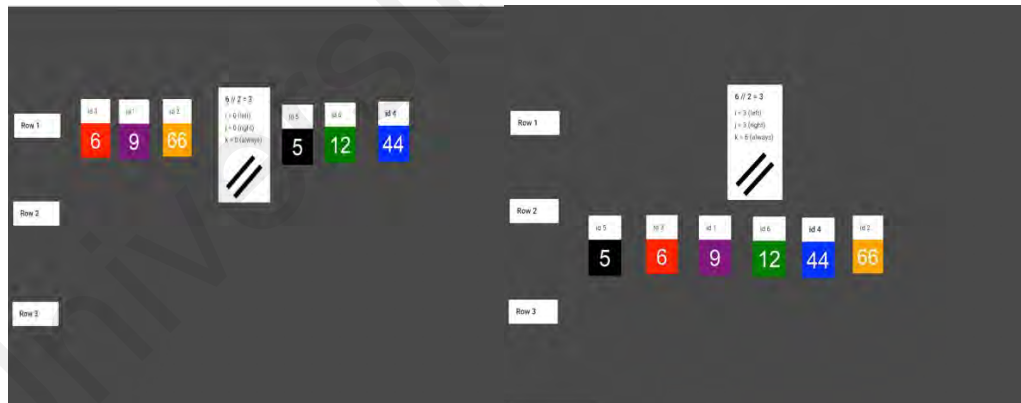


Figure 5.24: Sort and merge sorted left and right

Figure 5.19 illustrates a collection of unsorted numbers in Padlet. Mid index is found by floor technique and then divided left and right using the mid-point (Fig 5.20). Left side is further divided using same technique to find mid and then the divided groups are merged. The same divide and conquer technique happen for the right side as well. Finally,

sorting takes place between the sorted right and left side and the numbers are all in a sorted order (Fig 5.24)

(b) *From activity to code:*

But when do we get the middle point and divide?

`if len(array) > 1:`

- If length of array is greater than 1, then we will get the middle point and divide.
- If left and right side has no more than 1 element, then we do not divide any more.

How do we transfer this into code?

```
def mergeSort(array)
    mid = len(array) // 2
    Left = array[:mid]
    Right = array[mid:]
```

- Define a function mergeSort that will receive the data as array
- Getting mid of the array so we can split it
- Getting left side of array using middle point
- Getting right side of array using middle point

Figure 5.25: Divide the array until length is greater than 1

How do we divide further and further?

Click to add text

How do we divide further to the left side?

Recursively call mergeSort Function and pass Left array to divide further.

Figure 5.26: Divide left and right further

So how do we conquer and sort the whole left side?

- Starting from the last divided elements we start to conquer.

```
i = j = k = 0
```

- Initialize i(left), j(right), k(array) as 0 that will later keep count of left and right-side elements and organize sorting.

```
while i < len(Left) and j < len(Right):
    if Left[i] < Right[j]: # If Left item less than Right
        array[k] = Left[i] # Left item placed in array
        i += 1 # i = i + 1
    else:
        array[k] = Right[j] # Right item placed in array
        j += 1 # j = j + 1
    k += 1
```

`Left[0] < Right[0] => 66 < 6 = FALSE`

```
array[0] = 6
i = 0
j = 0 + 1 = 1
k = 0 + 1 = 1
```

Figure 5.27: Sort and merge last divided items

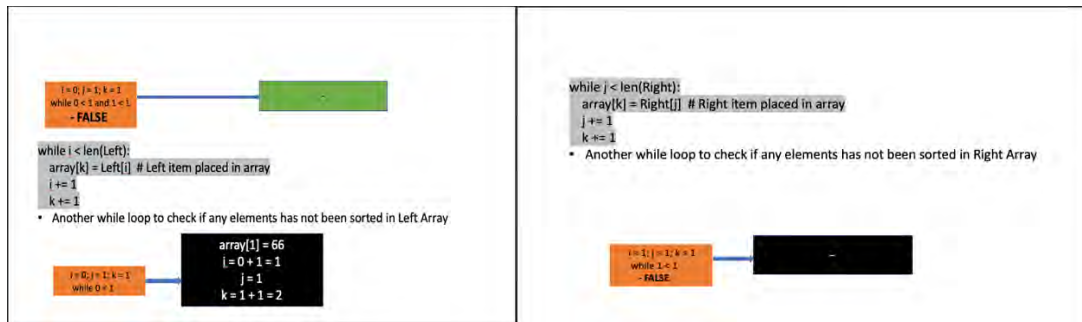


Figure 5.28: Sort and merge when no more items left in one side

When the numbers are being divided into groups, the division related syntaxes are displayed along with defining the function and how they work (Fig 5.25) using concepts of indexing and slicing if the array to be divided is greater than 1. Further divisions are explained with recursive function calls and how to re-use the functions in code. Fig 5.27 and 5.28 shows that while loops have been used to merge the divisions, and sort them. The variables initialized for increment within while loop (i, j, k) were conceptualized in Padlet during the activity so students can understand the increment and decrement operations within while loops.

5.5.3 Game-based workshop design (interactive gamified system)

Similar to the problem-based workshop, the sorting algorithms that were used in the workshop include: bubble sort, counting sort, and merge sort. A CT-based solution using our PSA model has already been formulated for all these algorithms (please refer to [bubble sort](#), [counting sort](#), [merge sort](#)). These formulated solutions were modified to fit into the games that were used to design the gamified approaches of these algorithms in an interactive gamified system. After each level has been played by the participants, they were provided with an idea about the programming syntaxes by relating them to the steps in the game. Users have to choose the level first. Bubble sort and counting sort consist of 3 levels which are easy, medium, and hard. Merge sort consists of 2 levels: easy and medium level. For each correct move, students get 1 point. For each wrong move, 1 point is deducted.

5.5.3.1 Modules of the interactive gamified system

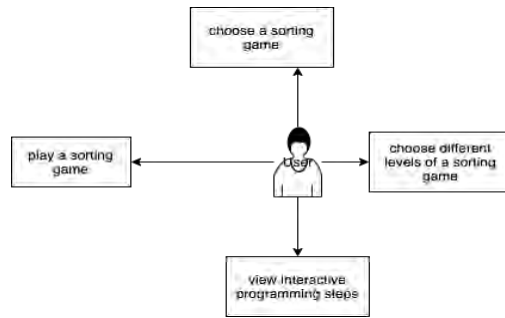


Figure 5.29: Modules of the system



Figure 5.30: Levels to choose before starting the game

5.5.3.2 Bubble sort in GBL

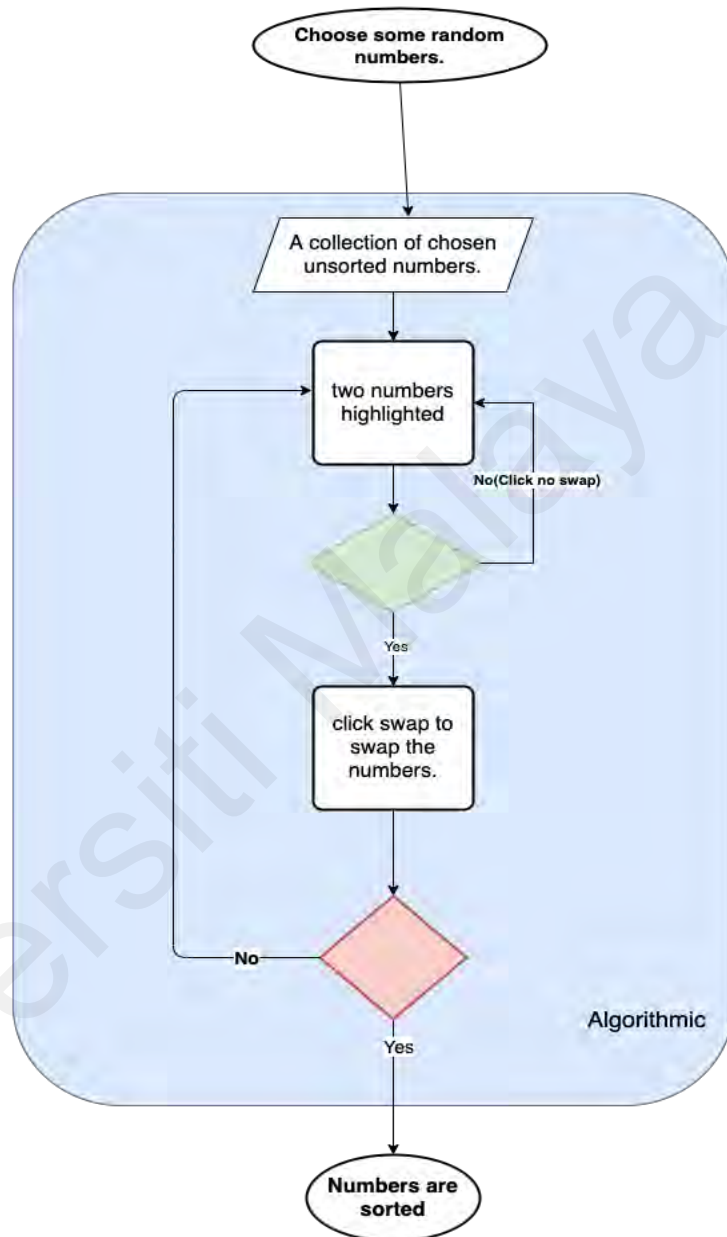
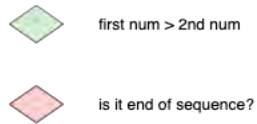


Figure 5.31: Bubble sort using PSA for GBL

The [Bubble sort problem-solving](#) process using the PSA model has been utilized to formulate the process of the bubble sort game in this workshop. This model has been the guideline to design the game for bubble sort.

(a) *Transfer the model into a gamified approach:*

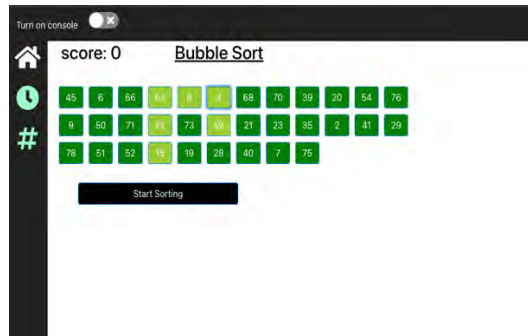


Figure 5.32: Choose some random numbers

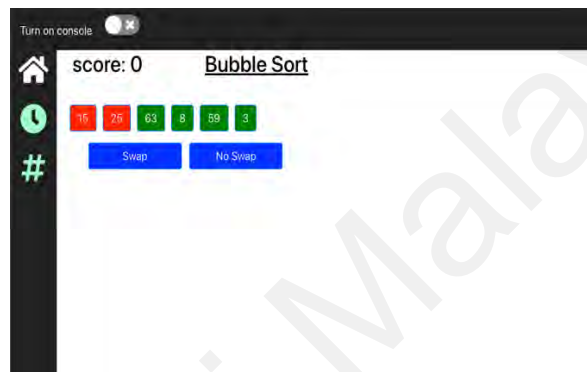


Figure 5.33: Two numbers highlighted

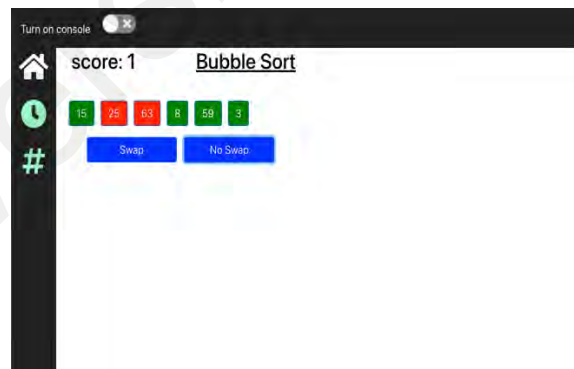


Figure 5.34: Swap or no swap based on condition

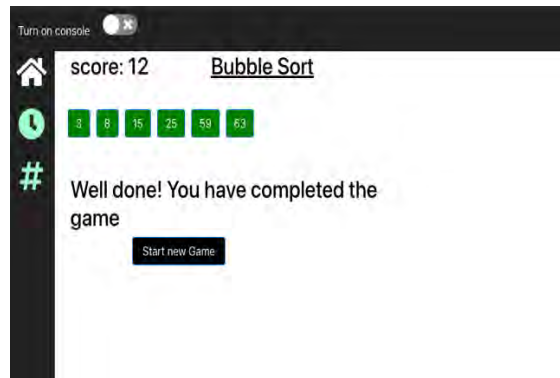


Figure 5.35: Game completed after all steps with score

The above figures illustrate how the PSA model was translated into a gamified approach. User selects the level and then selects numbers randomly and finalize a collection to sort the numbers. Figure 5.33 shows how two numbers are highlighted. Two buttons named ‘Swap’ and ‘No Swap’ are present for the user to decide the conditions based on the order of the selected numbers. After a correct move, next two numbers to be compared are highlighted.

(b) Interactive console to relate each step with code:



Figure 5.36: Random numbers chosen initialized as array



Figure 5.37: If condition to swap numbers





Figure 5.38: A for loop for comparing the numbers



Figure 5.39 Nested loop to go (size - 1) rounds

When the random numbers are being chosen, the interactive console display them as an array being initialized as shown in Figure 5.36. When two numbers are highlighted, it displays the if condition and the index of the highlighted numbers along with the condition to check if numbers are to be swapped or not. After the first swap, the for loop concept is shown to explain how to iterate within a collection of numbers. After completion of the first round of comparison among the numbers, the concept of a nested for loop is available for the participants to understand how many rounds the comparison has to be done.

5.5.3.3 Counting sort in GBL

-  available occurrences greater than 0?
-  are all numbers with count > 0 placed in final?

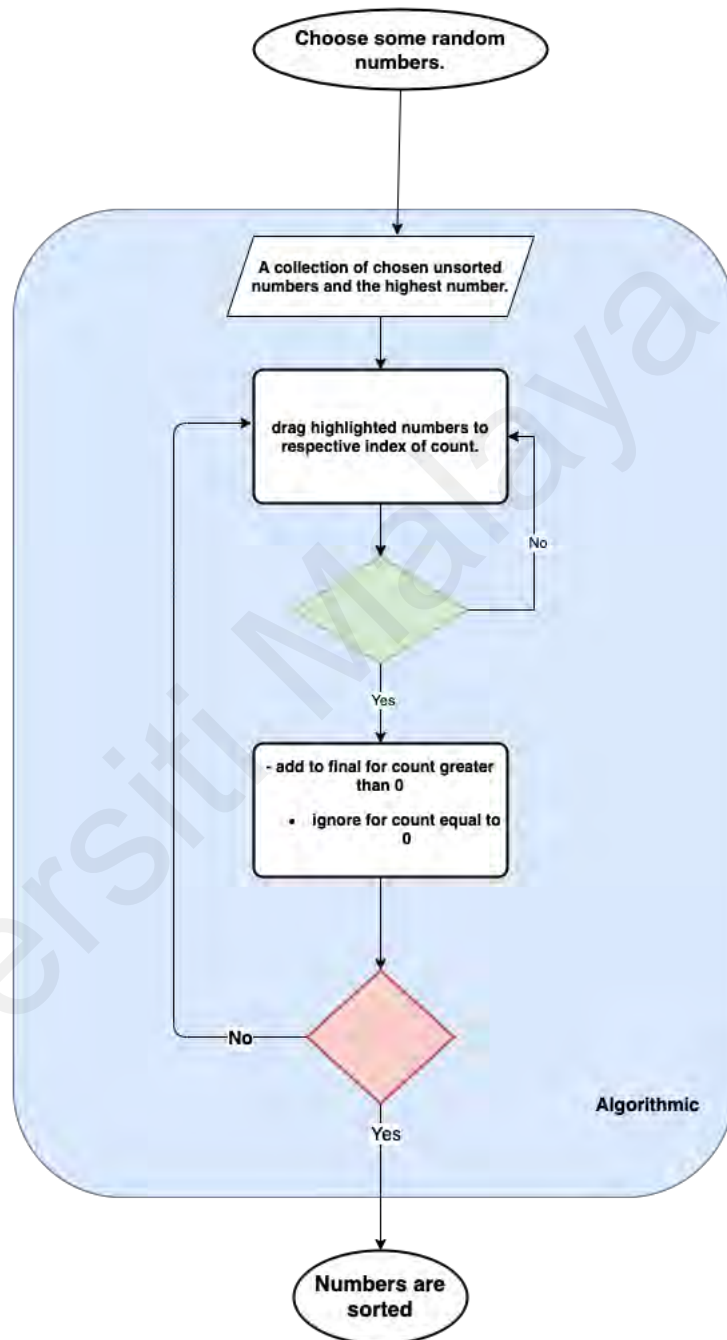


Figure 5.40: Counting sort using PSA for GBL

The [Counting sort problem-solving](#) process using the PSA model has been utilized to formulate the process of the counting sort gamified approach in this workshop. This model has been the guideline to design the game for Counting sort.

(a) *Transfer the model into a gamified approach:*

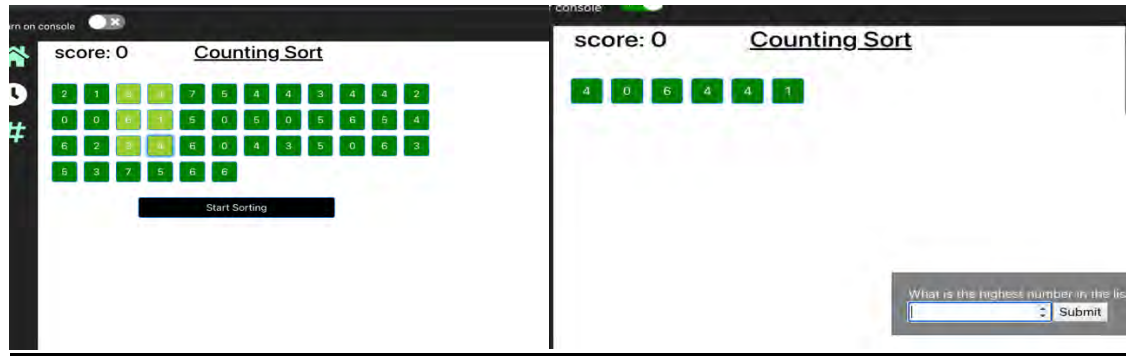


Figure 5.41: Choose random numbers and find highest

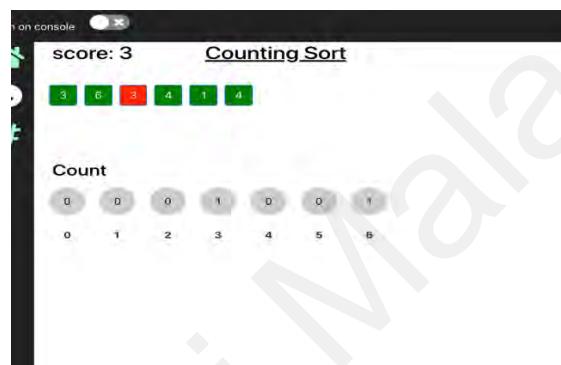


Figure 5.42: Drag and drop highlighted number to respective index

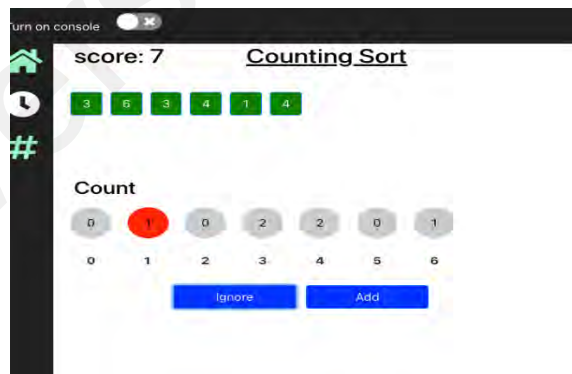


Figure 5.43: Ignore for count = 0 and add for count > 0

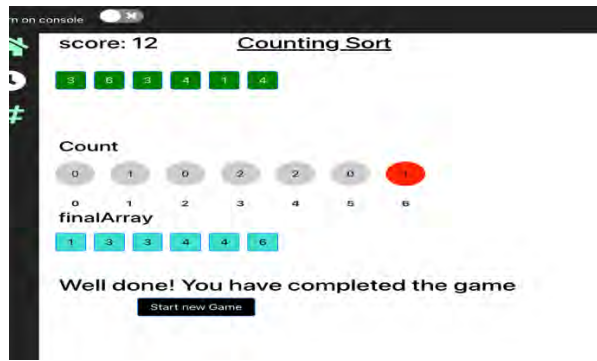


Figure 5.44: Count index appended to final array

After selecting the desired level from some random numbers, a collection of unsorted numbers is chosen to be sorted using counting sort. After correct input of the highest number, (highest + 1) number of count boxes are generated. Each time a highlighted number has to be dragged and dropped into the correct count index (Fig 5.42). Once all the numbers have finished counting from the list to be sorted, it is time to append them to a sorted list. Each count with 0 has to be ignored while those greater than 0 has to be added to final array. While adding the counted index to final array, the highlighted number has to be followed. Once all numbers have been appended successfully, the game is finished and the final score is displayed.

(b) Interactive console to relate each step with code:

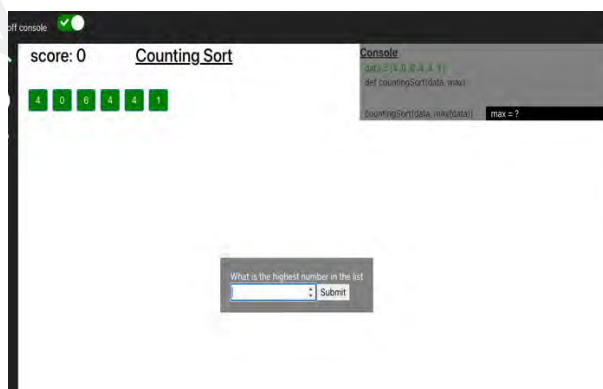


Figure 5.45: Define function and find highest number in list

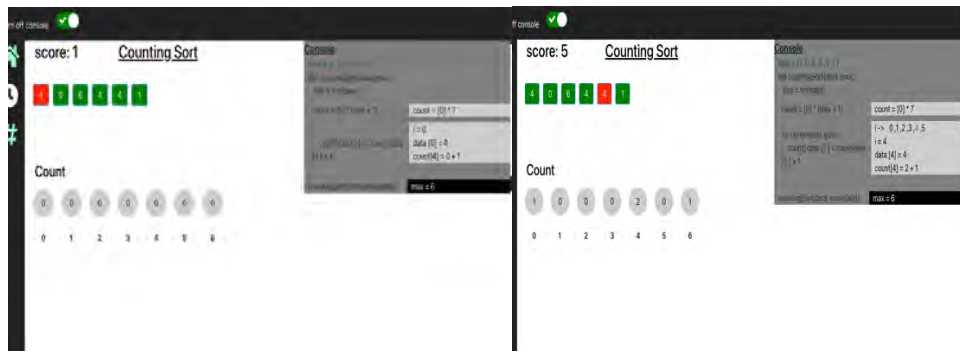


Figure 5.46: How count array is created to keep count of numbers in data

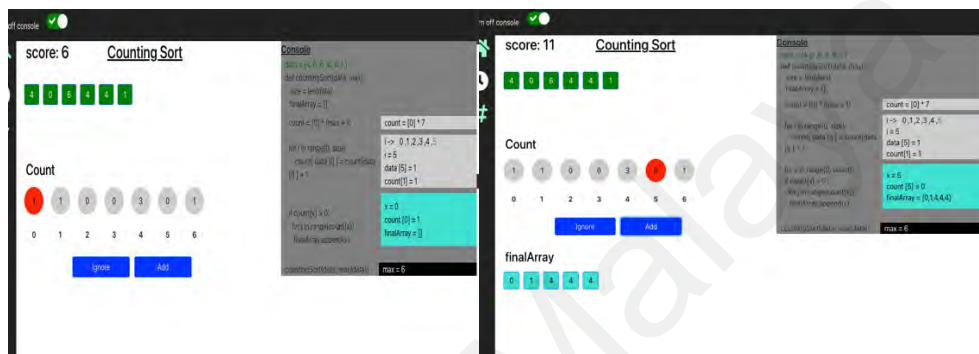




Figure 5.47: How to append by ignoring count = 0

Defining a function and calling a function – these two concepts will be included here. The highest number in the array will be displayed which will utilize the **max** function in python to get the highest number. (highest number + 1) array generation interactivity will be displayed at this step once the correct number has been input. How count is being added will be shown for the index 0 of count and the following index will be displaying the for loop concept. Once all the counts have been obtained, a for loop concept and ‘Append’ method with an if statement will be displayed, which will append the index as many times as count and ignore the count that is not greater than 0. A for loop will display how the count array is being iterated to append the counted index and finally sort the data.

5.5.3.4 Merge sort in GBL

-  all divided group has only one number?
-  are the highlighted numbers dragged to respective index ?

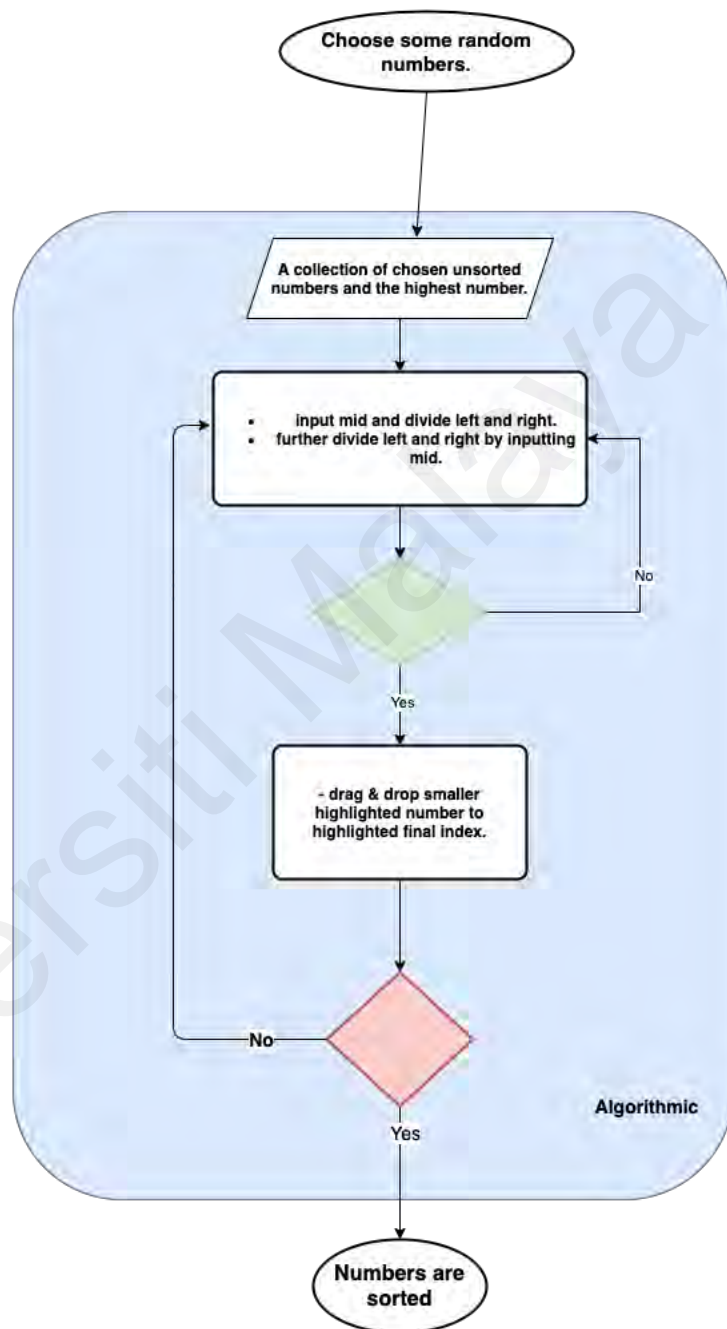


Figure 5.48: Merge sort using PSA in GBL

The [merge sort problem-solving](#) process using the PSA model has been utilized to formulate the process of the merge sort game in this workshop. This model has been the guideline to design the game for merge sort.

(a) *Transfer the model into a gamified approach:*

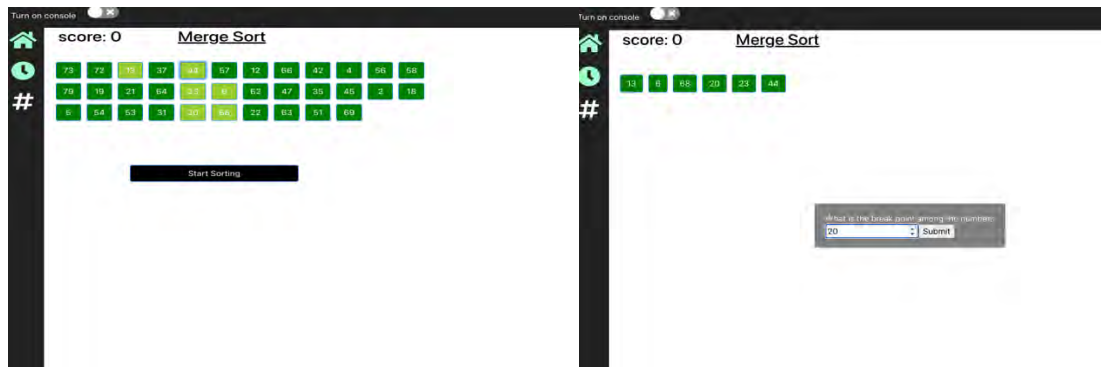


Figure 5.49: Choose a collection of unsorted numbers and find the highest

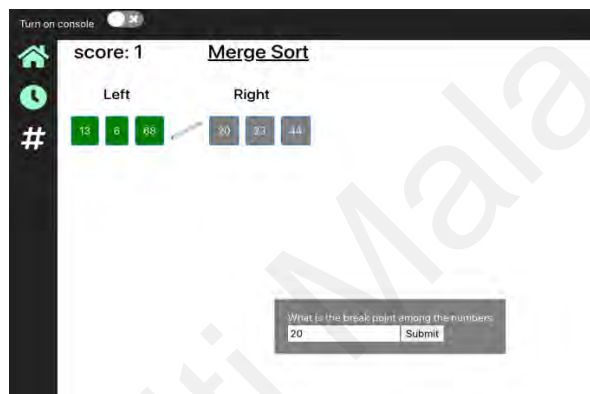


Figure 5.50: Left and right divided

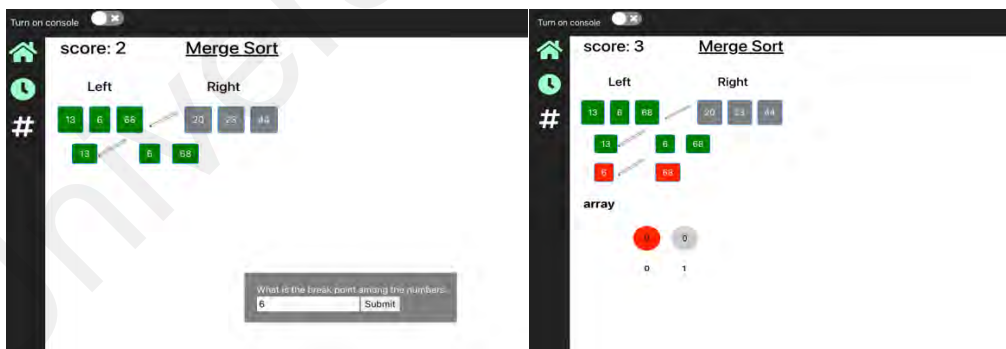


Figure 5.51: Further divide left side

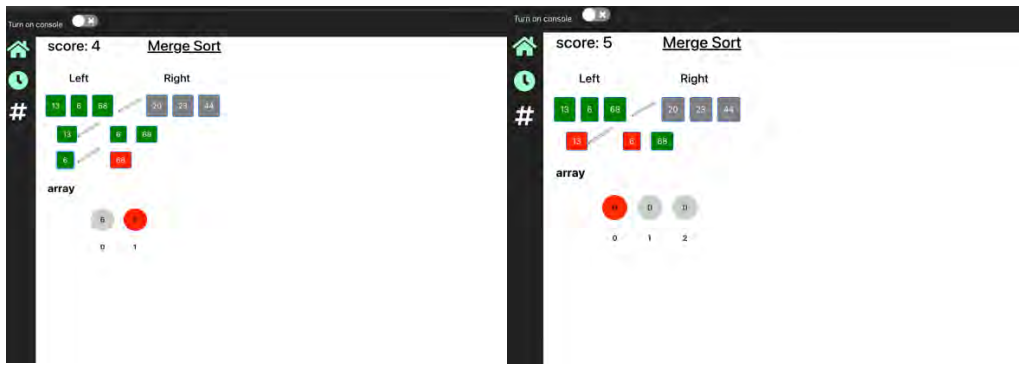


Figure 5.52: Drag and drop smaller number to highlighted final index

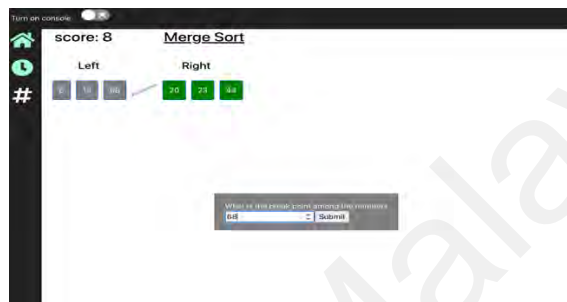


Figure 5.53: Further divide right side and merge

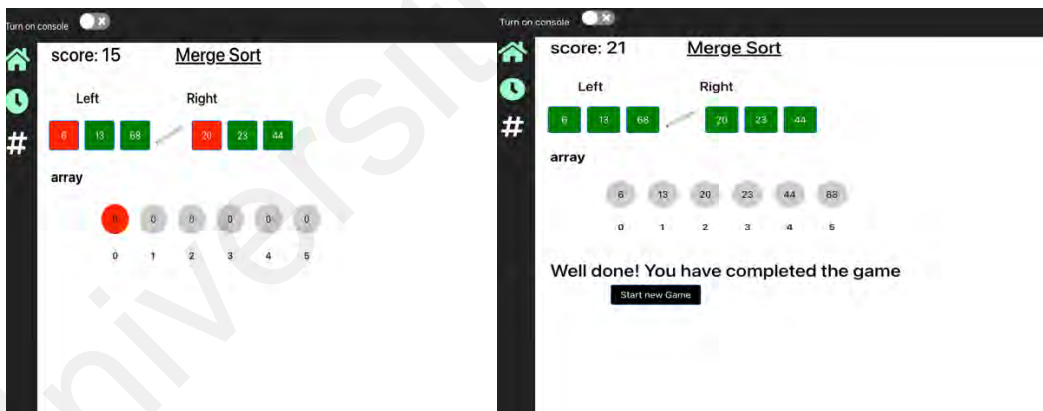


Figure 5.54: Drag and drop to merge sorted left and sorted right

Figure 5.49 illustrates a collection of unsorted numbers chosen by the user and then the user inputs the mid of the numbers using floor technique. If the correct number according to the mid index has been input, the numbers are divided into left and right. Utilizing the same procedure, the left side is divided first. Once all the divided groups have only one number, then the array displays as many boxes as available numbers in both sides of the last divided group. Among the selected numbers, the smallest number

needs to be dragged to the highlighted box of the array (Fig 5.52). If one side has already been dragged, then the highlighted number on the other side will be dragged to the highlighted box. In this process, the left side will be sorted. Follow the same procedure for the right side by inputting the mid and do all the things same as the left side. Once both the left and right sides are sorted, the final sorting takes place where the drag and drop takes place and sorts all the numbers.

(b) *Interactive console to relate each step with code:*

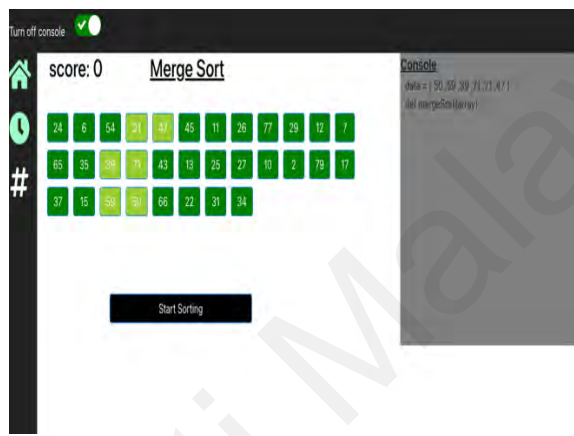


Figure 5.55: Random chosen unsorted number as array and function defined



Figure 5.56: Break left and right side by inputting mid

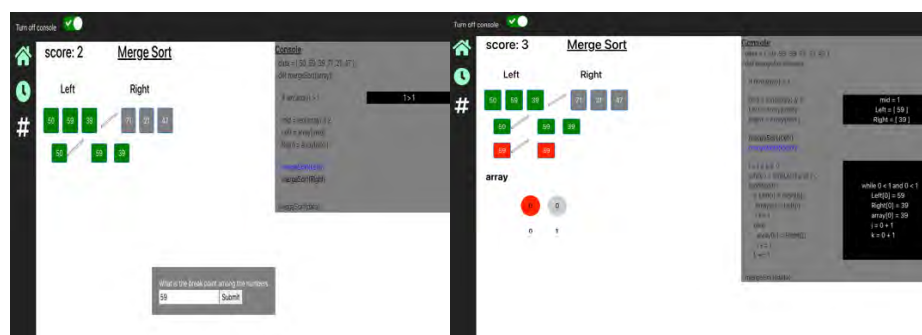


Figure 5.57: Further right division calling recursive function

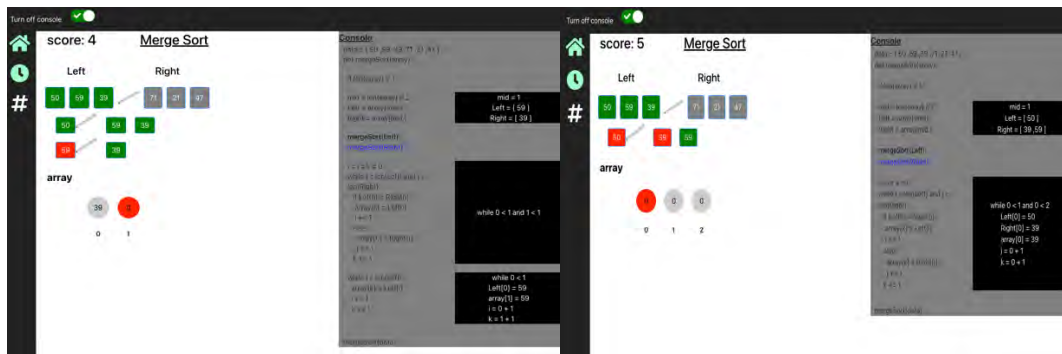


Figure 5.58: While loop for merging the unsorted divisions

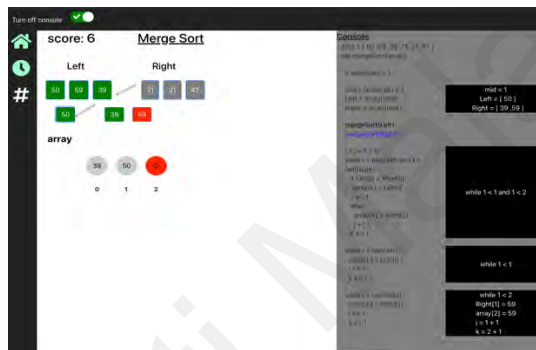


Figure 5.59: Additional while loops if items are remaining in either side

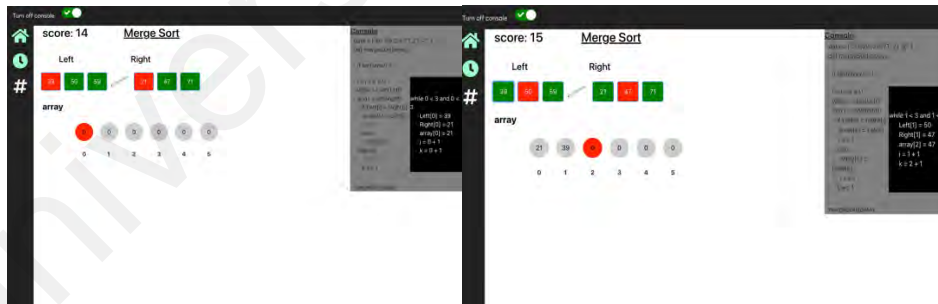


Figure 5.60: Sorted left and right merging using while loops

First of all, the interactive system will show that code checks in length of array is greater than 1. Once the array length is greater than 1, then the length is floored to divide the array into left and right. MergeSort function is called recursively so that the left and right can be divided further. The left and right further division is displayed through interactive design by reusing the previous logics for division. The merging process using

while loops are shown. First the two sides to be merged will be shown using the while loop as long as both sides have some elements. The merging process for the rest of the items and sorting them using another 2 while loops are displayed interactively and how they are placed in the sorted array.

The workshops have been conducted using the above experimental process for PBL and GBL. The students have actively participated in the process and the outcome of this workshops are evident in the next chapter.

Universiti Malaya

CHAPTER 6: FINDINGS AND ANALYSIS

In this section, the pre-test and post-test scores were included and analysed after they were marked based on our marking scheme. Paired sample T-test has been carried out to compare the pre-test and post-test of each workshop. A one-way ANOVA test has been carried out to compare the pre-tests and post-tests of all the workshops.

6.1 Comparison between Pre-Test and Post-Test

6.1.1 Syntax-based workshop

<i>Paired Samples Statistics</i>					
		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Test before workshop	5.9967	30	2.44452	.44631
	Test after workshop	6.9900	30	1.92750	.35191

<i>Paired Samples Test</i>									
		Paired Differences			95% Confidence Interval of the Difference		t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	Test before workshop - Test after workshop	-.99333	2.47197	.45132	-1.91638	-.07028	-2.201	29	.036

A paired sample T-test was conducted to compare the pre-test and post-test scores of the syntax-based workshop.

According to the above table, there was a significant difference (conditions $t(29) = -2.201$, $p = 0.036$) in the scores of pre-test ($M=5.99$, $SD=2.44$) and post-test ($M= 6.99$, $SD=1.92$) for 95% level of significance. These results suggest that the syntax-based programming course had a positive effect on students' programming learning. Specifically, the student's results were better after taking the course and achieved better marks in the post-test.

6.1.2 Problem-based learning workshop

<i>Paired Samples Statistics</i>					
		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	preTest	5.0967	30	2.51759	.45965
	postTest	8.0567	30	1.96814	.35933

<i>Paired Samples Test</i>									
		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	preTest - postTest	-2.96000	2.31064	.42186	-3.82281	-2.09719	-7.017	29	<.001

A paired sample T-test was conducted to compare the pre-test scores and post-test scores of the problem-based workshop.

According to the above table, there was a significant difference (conditions $t(29) = -7.017$, $p < .001$) in the scores of the pre-test ($M=5.09$, $SD=2.51$) and post-test ($M= 8.05$, $SD = 1.96$) for 95% level of significance. These results suggest that the problem-based programming course had a positive effect on students' programming learning. Specifically, the student's results were better after taking the course and achieved better marks in the post-test.

6.1.3 Game-based learning workshop

<i>Paired Samples Statistics</i>					
		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	preTest Scores	5.3600	30	3.30273	.60299
	postTest Scores	8.6200	30	1.90143	.34715

<i>Paired Samples Test</i>										
		Paired Differences					t	df	Significance	
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference					
					Lower	Upper			One-Sided p	Two-Sided p
Pair 1	preTest Scores - postTest Scores	-3.26000	1.77989	.32496	-3.92462	-2.59538	-10.032	29	<.001	<.001

A paired sample T-test was conducted to compare the pre-test scores and post-test scores of the game-based workshop.

According to the above table, there was a significant difference (conditions $t(29) = -10.032$, $p < .001$) in the scores of pre-test ($M=5.36$, $SD=3.3$) and post-test ($M=8.62$, $SD=1.90$) for 95% level of significance. These results suggest that the game-based programming course had a positive effect on students' programming learning. Specifically, the student's results were better after taking the course and achieved better marks in the post-test.

6.2 Comparison between Pre-Tests of the Three Workshops

ANOVA					
Dependent Variable: Participant marks					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	12.847	2	6.423	.830	.440
Within Groups	673.436	87	7.741		
Total	686.283	89			

A one-way ANOVA between groups was conducted to compare the test results and understand students' programming level based on the pre-test for the syntax-based workshop, problem-based workshop, and game-based workshop. There was not a significant difference in programming understanding level in the pre-test workshops at the $p=0.440$ level for the three conditions [$F(2,87)=0.83$, $p=0.44$]

6.3 Comparison between Post-Tests of the Three Workshops

ANOVA					
Dependent Variable: marks obtained					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	41.120	2	20.560	5.505	.006
Within Groups	324.924	87	3.735		
Total	366.044	89			

A one-way ANOVA between subjects was conducted to compare the test results and understand students' programming level based on the post-test for the syntax-based workshop, problem-based workshop, and game-based workshop. There was a significant difference of programming understanding level in the post-test programming workshops at the $p=0.006$ level for the three conditions [$F(2,87)=5.505, p=0.006$]. Since there was a significant difference, a required post hoc test was carried out to compare each condition to the other two conditions.

Descriptives

Dependent Variable: marks obtained

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
syntax	30	6.9900	1.92750	.35191	6.2703	7.7097	2.90	10.00
activity	30	8.0567	1.96814	.35933	7.3218	8.7916	4.30	10.00
game	30	8.6200	1.90143	.34715	7.9100	9.3300	4.90	10.00
Total	90	7.8889	2.02802	.21377	7.4641	8.3136	2.90	10.00

Multiple Comparisons

Dependent Variable: marks obtained

Test: LSD

(I) name of workshop	(J) name of workshop	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
syntax	activity	-1.06667*	.49898	.035	-2.0584	-.0749
	game	-1.63000*	.49898	.002	-2.6218	-.6382
activity	syntax	1.06667*	.49898	.035	.0749	2.0584
	game	-.56333	.49898	.262	-1.5551	.4284
game	syntax	1.63000*	.49898	.002	.6382	2.6218
	activity	.56333	.49898	.262	-.4284	1.5551

*. The mean difference is significant at the 0.05 level.

Post hoc comparisons using LSD test indicated that the mean score for the syntax-based workshop post-test scores ($M=6.99, SD=1.92$) was significantly different than the post-test scores of the activity-based workshop ($M=8.05, SD=1.96$) and the post-test scores of the game-based workshop ($M=8.62, SD=1.90$). However, the post-test scores of the activity-based workshop ($M=8.05, SD=1.96$) did not significantly differ from the post-test scores of the game-based workshop ($M=8.62, SD=1.90$).

6.4 Analyze and Discuss the Results

Our [problem statement](#) points out that the biggest mistake many new programmers make is not focusing on solving problems but on learning syntax. The PSA model was

developed to solve programming-related problems, solve the issues and teach students using this approach model. The first rule of learning programming language includes syntax and semantics. The second rule is how computers run a program (pointer, operation). The third is how to solve a specific problem (Chen, 2018). The workshops' design has included these things when the PSA model was utilized in the student-centred approach by using the PSA model of different algorithms and then using those models to different active learning approaches. The PSA model of a specific algorithm was modified to fit in the procedure of the particular workshop. The modified model was then translated to that algorithm's purposed workshop activity to solve the problem (Please refer to [Modelling and Workshop Design](#)). Each step in the activity was later related to the required programming workable syntaxes and semantics (function calls and iterations) and at the same time explained how different operations work. The syntax-based approach included syntax and semantics. The workshop has demonstrated the usage of those syntaxes that can be used to formulate a problem by using simple examples in a collaborative environment.

The above results indicate all the workshop pre-test and post-test result findings. The pre-test was conducted before the students joined their respective workshops, and the post-test was conducted upon completion of the workshop. The test results can be found in Appendix G. The pre-test results have been compared using one way ANOVA test, and no significant differences were found. It implies that the understanding level of the participants was almost similar in each workshop (refer to [6.1.4](#)). The pre-test and post-test results of the workshops through a paired sample T-test were compared (section [6.1.1](#), [6.1.2](#), [6.1.3](#)) and all the results have displayed that there was a significant difference between the pre-test and post-test results and students have improved their scores after the workshops. According to the marking criteria, as mentioned in the [Methodology](#) section, the students were evaluated mostly on their application of CT skills in the tests.

In addition, the questions in the test were open to any programming language as well as pseudocode because the main motive was to understand the problem-solving skill development of students by the workshops. The paired sample T-test results with the significant differences in each case show that students improved their problem-solving skills and acquired better programming knowledge by attending these workshops.

Ineffective teaching approaches and lack of problem-solving skills were identified as a problem in this research. Our PSA model was translated and utilized in problem-based workshops and game-based workshops. The control group was the participants in the teacher-centred workshop. Since all the workshops have shown student improvement, it was necessary to understand the level of improvement in different teaching approaches. A one-way ANOVA test (refer to [6.1.5](#)) was carried out, and a significant difference between the results paved the way to do a post hoc test for comparing each workshop with the other two. Results suggested that problem-based learning and game-based learning participants had better scores compared to the teacher-centred approach. This suggests that the effective teaching approach for programming is active learning since both these workshops had better results compared to the teacher-centred approach. Furthermore, these workshops encompass the PSA model and prove that the translation of the PSA model through these active learning approaches helped to strengthen the problem-solving skills.

The results (refer to [6.1.5](#)) also show that there was not a significant difference between the results of the problem-based workshop and the game-based workshop. So, the teaching aspect had to be compared for both approaches to understand the more convenient approach that fully focuses on solving problems based on CT skills. A few points to be noted from the instructors' perspective include:

- 1) The game-based workshop was more automated compared to the problem-based workshop due to an online gamified system, e.g., highlighting a number and

comparing the highlighted numbers were easier to find and more comfortable for the participants.

- 2) The scoring system of the game, along with the allotted time, created competitiveness among students as they were posting scores once they have finished playing all levels of the game (refer to Appendix H).
- 3) The interactivity of the game-based approach made it more distinct and helped to understand the programming steps while playing the sorting game. This was a real-time approach, whereas the problem-based approach was playing the game in an online software and then use PowerPoint slides to relate them with the codes.

The interactivity and automated features of the gamified system made instructions smoother and consumed less time. Therefore, it is suggested from a teaching perspective that the game-based approach would be a better approach to be utilized.

Universiti Malaysia

CHAPTER 7: CONCLUSION

7.1 Research Contributions

The PSA model has been derived from the basic concepts of computational thinking. This model has interconnected the CT concepts by deriving the data first, created an iterative flow between decomposition and pattern recognition, and then only it has been made possible for the decomposed part to be abstracted, bypassing the recognized pattern condition. The three major concepts of decomposition, pattern recognition, and abstraction are interconnected through an iterative approach, and this interconnection resulted in an algorithmic approach. This model has simplified the algorithmic process through the interconnection of these CT concepts. A general CT approach requires an algorithmic approach to be utilized after the initial three steps to solve a problem. But this model has enabled the algorithmic process to be automated by interconnecting the first three cornerstones of CT. This model is based on the sequencing of the CT concepts where the data is identified first and then decomposed into smaller sub-problems, identify similarities or dissimilarities, extract unnecessarily characteristics, avoid repetitions and process in an algorithmic manner until the intended solution of the problem has been achieved.

Another contribution in this research is that the PSA model has been utilized to formulate a solution to different sorting algorithms that have later paved the way to translate the sorting problems to effective active learning approaches to teach programming based on a problem.

7.2 Revisiting the Objectives

The objectives that were raised and the questions associated to them have been the motive to carry out this research.

7.2.1 First objective and related question and answers

Our first objective was to investigate teaching issues in solving programming problems. The first question was to know the reason behind selecting CT concepts as a problem-solving skill. The relationship of CT with other problem-solving skills in the literature provided an understanding that CT helps to develop the core thinking skills, and that is why it is the chosen problem-solving approach for this research. The second question about CT concepts enhancing problem-solving skills in programming was elaborated by researching various studies and finding out primarily that CT concepts are quite useful to enhance problem-solving skills and represent very important tools for teachers teaching programming in the 21st century. The third question was answered using a thorough investigation of the teacher-centred approach and a student-centred approach by discussing the positive and negative sides of both approaches. It was found that the student-centred active learning approach is effective in solving problems, and this claim was further established through enhanced research of active learning approaches such as problem-based and game-based learning and their role in developing CT skills.

7.2.2 Second objective and related question and answers

The second objective of our research was to design the PSA model for teaching programming. The CT concepts were utilized to develop the PSA model by interconnecting the major CT components and incorporating those to an iterative approach that already became an algorithmic approach. The PSA model helped to solve different sorting algorithm problems by utilizing this interconnected approach and solved five different algorithms. Later, the PSA model was translated into effective programming teaching methods by modelling the activity/game-based on the chosen sorting algorithm PSA model, and from that, programming concepts related to those steps were taught.

7.2.3 Third objective and related question and answers

The third and final objective was to evaluate the effectiveness of the PSA model in relation to different programming teaching approaches. The control group utilizing the teacher-centred teaching approach was compared with the active learning approaches such as the problem-based learning group and the game-based learning group. The workshop using a pre-test and post-test helped us to know that student-centred active learning approaches are better than teacher-centred approaches. From a teaching perspective, discussing the differences between both the active learning approaches, it was suggested that within active learning approaches, game-based learning is a better active learning approach than problem-based learning.

7.3 Significance of Study

The pilot study that was carried out helped us to understand the issues that were faced during the study and provided suggestions for improvement for the feasibility study. This study helped to overcome the raised issues by considering the suggestions to design the courses and bring the teaching into a computational process. The PSA model lets participants get into the process step-by-step and formulate the problems in an iterative problem-solving process. The novelty in this study is that this will help students to formulate solutions for sorting algorithm problems and enhance their problem-solving skills when they are approaching a programming question or task.

7.4 Concluding Remarks

Therefore, it can be concluded that the proposed model, based on the collected data and analysis of those, is good for solving problems and aids in teaching programming. Our execution of this model to formulate problems and using sorting algorithms to teach different programming concepts proved to be fruitful to instil interest among students for active participation and be better at solving problems in the classroom. Using this model,

educators and instructors might be able to get students more interested in learning programming and overcome the lacking.

7.5 Research Challenges

A limitation of the present study is that the workshops were conducted fully online. At the beginning of this research, face-to-face workshops were the real motive. But due to the pandemic situation, there were no other options rather than opting in for a complete online solution. It was initially intended to use in-person activities and games so that it could be more engaging.

A major challenge with the pre-test and post-test was that the time limitation was not possible to be tracked. Even though students were told not to use more than 30 minutes, but it was impossible to ensure other than just asking them. Some participants who did the pre-test did not attend the workshops and were hard to find later. Otherwise, there would be more participants in the study.

7.6 Future Improvements

The research has been limited to the core computational thinking concepts for the moment. In the future, more concepts can be integrated into the model to make it more detailed and effective. In addition, future researchers can use this model and come up with a guideline to map other problems so that other people can do mapping for different algorithms.

The study was conducted using Python programming language, and other languages were not used. Therefore, other languages such as Java, C/C++, Javascript, etc. languages can also be used to be taught using this model and transforming it into codes following this process. Any other suitable pedagogical approach can also be examined using this model, provided the teaching procedure is compatible with the model.

REFERENCES

- Acharya, S., & Gayana, M. N. (2021). *Enhanced Learning and Improved Productivity of Students ' using Project Based Learning Approaches for Programming Courses*. 34(January), 524–530.
- Adamchik, V., & Gunawardena, A. (2003). A learning objects approach to teaching programming. *Proceedings ITCC 2003, International Conference on Information Technology: Computers and Communications*, 96–99.
<https://doi.org/10.1109/ITCC.2003.1197507>
- Ahsan Habib, M. (2019). *A RECOMMENDER OF PHYSICAL GAMES FOR LEARNING PROGRAMMING AND COMPUTATIONAL THINKING*.
- Ala-Mutka, K. M. (2004). Problems in learning and teaching programming-a literature study for developing visualizations in the Codewitz-Minerva project. *Codewitz Needs Analysis*, 1–13.
<http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:PROBLEMS+I+N+LEARNING+AND+TEACHING+PROGRAMMING+-+a+literature+study+for+developing+visualizations+in+the+Codewitz-Minerva+project#0>
- Analytical Thinking and Critical Thinking. (n.d.). *Analytical Thinking and Critical Thinking*. <https://thepeakperformancecenter.com/educational-learning/thinking/critical-thinking/analytical-thinking-critical-thinking/>
- Anonymous. (2014). *Computational thinking: How do we think about problems so that computers can help?* 1–17. <http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/>
- Atmatzidou, S., & Demetriadis, S. (2014). How to Support Students ' Computational Thinking Skills in Educational Robotics Activities. *Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th*

International Conference Robotics in Education, July, 43–50.

Avello-Martínez, R., Lavonem, J., & Zapata-Ros, M. (2020). Coding and educational robotics and their relationship with computational and creative thinking. A compressive review. *Revista de Educacion a Distancia, 20*(63).

<https://doi.org/10.6018/RED.413021>

Bang, M. M. (2018). *Teaching Sorting Algorithms in an Interactive Virtual Reality Environment. July.*

Barr, D., Harrison, J., & Conery, L. (2011a). Computational Thinking: A Digital Age Skill for Everyone. *Learning and Leading with Technology, 38*(6), 20–23.

<http://quijote.biblio.iteso.mx/wardjan/proxy.aspx?url=https://search.ebscohost.com/login.aspx?direct=true&db=ehh&AN=59256559&lang=es&site=eds-live%5Cnhttps://content.ebscohost.com/ContentServer.asp?T=P&P=AN&K=59256559&S=R&D=ehh&EbscoContent=dGJyMMTo50Sep6>

Barr, D., Harrison, J., & Conery, L. (2011b). Computational Thinking: A Digital Age Skill for Everyone. *Learning and Leading with Technology, 38*(6), 20–23.

<http://quijote.biblio.iteso.mx/wardjan/proxy.aspx?url=https://search.ebscohost.com/login.aspx?direct=true&db=ehh&AN=59256559&lang=es&site=eds-live%5Cnhttps://content.ebscohost.com/ContentServer.asp?T=P&P=AN&K=59256559&S=R&D=ehh&EbscoContent=dGJyMMTo50Sep6>

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54. <https://doi.org/10.1145/1929887.1929905>

Barron-Estrada, M. L., Zatarain-Cabada, R., & Cardenas-Sainz, B. A. (2020). A natural user interface implementation for an interactive learning environment. *Proceedings - IEEE 20th International Conference on Advanced Learning Technologies, ICALT 2020, 341–343*. <https://doi.org/10.1109/ICALT49669.2020.00109>

- Bawamohiddin, A. B., & Razali, R. (2017). Problem-based learning for programming education. *International Journal on Advanced Science, Engineering and Information Technology*, 7(6), 2035–2050.
<https://doi.org/10.18517/ijaseit.7.6.2232>
- Bell, S. (2010). Project-Based Learning for the 21st Century: Skills for the Future. *The Clearing House: A Journal of Educational Strategies, Issues and Ideas*, 83(2), 39–43. <https://doi.org/10.1080/00098650903505415>
- Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning*, 1(2), 65–81. <https://doi.org/10.4018/ijgbl.2011040105>
- Bilech, B., Kay, K., & Yu, H. (2015). *An Analysis of Mathematical Notations: For Better or For Worse*. 1–54. <http://www.wpi.edu/academics/ugradstudies/project-learning.html>
- Bubica, N., & Boljat, I. (2014). Strategies for Teaching Programming to Meet New Challenges : State of the Art. *Ciet, September*, 1–6.
https://www.academia.edu/7689191/Strategies_for_Teaching_Programming_to_Meet_New_Challenges_State_of_the_Art
- Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation’s Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research*, 87(4), 834–860. <https://doi.org/10.3102/0034654317710096>
- Caeli, E. N., & Yadav, A. (2020). Unplugged Approaches to Computational Thinking: a Historical Perspective. *TechTrends*, 64(1), 29–36. <https://doi.org/10.1007/s11528-019-00410-5>
- Chang, C.-S., Chung, C.-H., & Chang, J. A. (2020). *Influence-of-problem-based-learning-games-on-effective-computer-programming-learning-in-higher-*

education-_-Enhanced-Reader.pdf.

- Chang, C. S., Chen, J. F., & Chen, F. L. (2015). Development and design of problem based learning game-based courseware. *Proceedings of the International Conference on E-Learning 2015, E-LEARNING 2015 - Part of the Multi Conference on Computer Science and Information Systems 2015*, 217–219.
- Cheah, C. S. (2020). Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review. *Contemporary Educational Technology*, 12(2), ep272. <https://doi.org/10.30935/cedtech/8247>
- Chen, G. (2018). *Programming Language Teaching Model Based on Computational Thinking and Problem-based Learning*. 156(Seiem), 128–131. <https://doi.org/10.2991/seiem-17.2018.31>
- Chuechote, S., Nokkaew, A., Phongsasithorn, A., & Laosinchai, P. (2020). A neo-piagetian analysis of algorithmic thinking development through the “sorted” digital game. *Contemporary Educational Technology*, 12(1), 1–15. <https://doi.org/10.30935/cet.685959>
- Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, 4(4), 355–369. <https://doi.org/10.1007/s40692-017-0090-9>
- Doody, O., & Doody, C. M. (2014). Conducting a pilot study: case study of a novice researcher Owen Doody and Catriona M Doody. *Case Study of Novice Researcher*, 4(1), 1–8.
- Duckworth, E. (2009). Helping Students Get to Where Ideas Can Find Them. *New Educator*, 5(3), 185–188. <https://doi.org/10.1080/1547688X.2009.10399573>
- Durak, H. Y., & Saritepeci, M. (2018). Analysis of the relation between computational

- thinking skills and various variables with the structural equation model. *Computers and Education*, 116, 191–202. <https://doi.org/10.1016/j.compedu.2017.09.004>
- Faraon, M., Rönkkö, K., Wiberg, M., & Ramberg, R. (2020). Learning by coding: A sociocultural approach to teaching web development in higher education. *Education and Information Technologies*, 25(3), 1759–1783. <https://doi.org/10.1007/s10639-019-10037-x>
- Fraser, J., Fahlman, D., Arscott, J., & Guillot, I. (2018). Pilot testing for feasibility in a study of student retention and attrition in online undergraduate programs. *International Review of Research in Open and Distance Learning*, 19(1), 260–278. <https://doi.org/10.19173/irrodl.v19i1.3326>
- García-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the computational thinking effects in pre-university education. *Computers in Human Behavior*, 80, 407–411. <https://doi.org/10.1016/j.chb.2017.12.005>
- Garneli, V., & Chorianopoulos, K. (2019). The effects of video game making within science content on student computational thinking skills and performance. *Interactive Technology and Smart Education*, 16(4), 301–318. <https://doi.org/10.1108/ITSE-11-2018-0097>
- Gelisli, Y. (2009). The effect of student centered instructional approaches on student success. *Procedia - Social and Behavioral Sciences*, 1(1), 469–473. <https://doi.org/10.1016/j.sbspro.2009.01.085>
- Hazzan, O., & Kramer, J. (2008). The role of abstraction in software engineering. *Proceedings - International Conference on Software Engineering, November*, 1045–1046. <https://doi.org/10.1145/1370175.1370239>
- Herskovitz, A., Sitman, R., Israel-Fishelson, R., Eguíluz, A., Garaizar, P., & Guenaga, M. (2019). Creativity in the acquisition of computational thinking. *Interactive Learning Environments*, 27(5–6), 628–644.

<https://doi.org/10.1080/10494820.2019.1610451>

Hu, C. (2011). Computational thinking - What it might mean and what we might do about it. *ITiCSE'11 - Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science*, 223–227.

<https://doi.org/10.1145/1999747.1999811>

Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010a). Instructional strategy in the teaching of computer programming: A need assessment analyses. *Turkish Online Journal of Educational Technology*, 9(2), 125–131.

Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010b). Instructional strategy in the teaching of computer programming: A need assessment analyses. *Turkish Online Journal of Educational Technology*, 9(2), 125–131.

Jesus, A. M. de, & Silveira, I. F. (2021). Game-based collaborative learning framework for computational thinking development. *Revista Facultad de Ingenieria*, 99(99), 113–123. <https://doi.org/10.17533/udea.redin.20200690>

Jonasen, T. S., & Gram-Hansen, S. B. (2019). Problem based learning: A facilitator of computational thinking. *Proceedings of the European Conference on E-Learning, ECEL, 2019-Novem*, 260–267. <https://doi.org/10.34190/EEL.19.150>

Kale, U., & Yuan, J. (2020). Still a New Kid on the Block? Computational Thinking as Problem Solving in Code.org. *Journal of Educational Computing Research*, 1–25. <https://doi.org/10.1177/0735633120972050>

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. *Procedia - Social and Behavioral Sciences*, 47, 1991–1999. <https://doi.org/10.1016/j.sbspro.2012.06.938>

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning programming at the computational thinking level via digital game-play. *Procedia*

- Computer Science*, 9(0), 522–531. <https://doi.org/10.1016/j.procs.2012.04.056>
- Kohn, T., & Komm, D. (2018). Teaching programming and algorithmic complexity with tangible machines. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11169 LNCS, 68–83. https://doi.org/10.1007/978-3-030-02750-6_6
- Kong, S. C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers and Education*, 151(March), 103872. <https://doi.org/10.1016/j.compedu.2020.103872>
- Kong, S. C., & Wang, Y. Q. (2020). Formation of computational identity through computational thinking perspectives development in programming learning: A mediation analysis among primary school students. *Computers in Human Behavior*, 106(December 2019), 106230. <https://doi.org/10.1016/j.chb.2019.106230>
- Korucu, A. T., Gencturk, A. T., & Gundogdu, M. M. (2017). Examination of the Computational Thinking Skills of Students. *Journal of Learning and Teaching in Digital Age (JOLTIDA)*, 2(1), 11–19.
- Kules, B. (2016). Computational thinking is critical thinking: Connecting to university discourse, goals, and learning outcomes. *Proceedings of the Association for Information Science and Technology*, 53(1), 1–6. <https://doi.org/10.1002/pra2.2016.14505301092>
- Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 14–18. <https://doi.org/10.1145/1067445.1067453>
- Larmer, J. (2015). *Project-Based Learning vs. Problem-Based Learning vs. X-BL*.

- Edutopia. <https://www.edutopia.org/blog/pbl-vs-pbl-vs-xbl-john-larmer#:~:text=We decided to call problem,in other types of projects.>
- Lathan, J. (2017). *Complete Guide to Teacher-Centered vs. Student-Centered Learning*. University of San Diego. <https://onlinedegrees.sandiego.edu/teacher-centered-vs-student-centered-learning/#:~:text=teacher-centered educational approach%2C the,%2C a student-centered vs.&text=In student-centered learning%2C the,role in their own learning.>
- Lee, I., Martin, F., & Apone, K. (2014). *Integrating computational thinking across the K–8 curriculum*. <https://dl.acm.org/doi/fullHtml/10.1145/2684721.2684736>
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1), 32–37. <https://doi.org/10.1145/1929887.1929902>
- Lee, L. (2019). *Computational Thinking is Critical Thinking—and Belongs in Every Subject*. Edutopia. <https://www.edutopia.org/article/computational-thinking-critical-thinking-and-belongs-every-subject>
- Lu, J. J., & Fletcher, G. H. L. (2009). Thinking about computational thinking. *SIGCSE Bulletin Inroads*, 41(1), 260–264. <https://doi.org/10.1145/1539024.1508959>
- Lui, A. K. F., Poon, M. H. M., & Wong, R. M. H. (2019). Automated generators of examples and problems for studying computer algorithms: A study on students' decisions. *Interactive Technology and Smart Education*, 16(3), 204–218. <https://doi.org/10.1108/ITSE-10-2018-0091>
- M., K. (2014). *Problems in Programming Education and Means of Their Improvement*. 459–470. <https://doi.org/10.2507/daaam.scibook.2014.37>
- Malhotra, N. (2019). *Implementing Active Learning and Student-Centered Pedagogy in Large Classes*. FACULTY FOCUS. <https://www.facultyfocus.com/articles/blended-flipped-learning/implementing->

active-learning-and-student-centered-pedagogy-in-large-classes/#:~:text=Active
learning places the student,such as comprehension and evaluation.

Malizia, A., Turchi, T., Danesi, F., Fogli, D., & Bell, D. (2020). *TAPASPlay: a Tangible Game-Based Learning Approach* (Issue 2017).

Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2014). Designing educational games for computer programming: A holistic framework. *Electronic Journal of E-Learning*, 12(3), 281–298.

Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., & Mendes, A. J. (2018). Learning Computational Thinking and scratch at distance. *Computers in Human Behavior*, 80, 470–477. <https://doi.org/10.1016/j.chb.2017.09.025>

Margaret Rouse. (2017). *sorting algorithm*. WhatIs.Com.
<https://whatis.techtarget.com/definition/sorting-algorithm>

Mascolo, M. (2009). Beyond Student-Centered and Teacher-Centered Pedagogy: Teaching and Learning as Guided Participation. *Pedagogy and the Human Sciences*, 1(1), 3.

Menon, D., Viéville, T., & Romero, M. (2019). Computational thinking development and assessment through tabletop escape games. *International Journal of Serious Games*, 6(4), 3–18. <https://doi.org/10.17083/ijsg.v6i4.319>

Mohd, F., Hielmi, E., & Daud, C. (2016). *MOBILE GAMES USABILITY*.

Muntean, C. H. (2019). *Teaching Tip : Flipping the Class to Engage Students in Learning Programming Algorithms*. 2320–2325.

N. Miller, B., & L. Ranum, D. (2006). *Problem Solving with Algorithms and Data Structures using Python*.

<https://runestone.academy/runestone/books/published/pythonds/Introduction/WhatIsProgramming.html#:~:text=Programming is the process of,be executed by a computer.&text=Algorithms describe the solution to,to produce the intended result.>

- Nasar, A. A. (2019). A Mathematical Analysis of student-generated sorting algorithms. *Mathematics Enthusiast*, 16(1–3), 315–330.
- Nguyen, V. T., Zhang, Y., Jung, K., Xing, W., & Dang, T. (2020). VRASP: A Virtual Reality Environment for Learning Answer Set Programming. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12007 LNCS(January), 82–91.
https://doi.org/10.1007/978-3-030-39197-3_6
- Nokkaew, A. (2019). *Sorted: An Educational Digital Game for Learning Sorting Algorithms*. April 2019.
- Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*, 11(1), 1–17.
<https://doi.org/10.1080/20004508.2019.1627844>
- Otukile-Mongwaketse, M. (2018). Teacher centered dominated approaches: Their implications for todays inclusive classrooms. *International Journal of Psychology and Counselling*, 10(2), 11–21. <https://doi.org/10.5897/ijpc2016.0393>
- Palts, T., & Pedaste, M. (2020). A model for developing computational thinking skills. *Informatics in Education*, 19(1), 113–128.
<https://doi.org/10.15388/INFEDU.2020.06>
- Park, H., Yang, S., & Choi, H. (2020). Scenario based active learning programming with unity 3d. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 56(4), 1283. <https://doi.org/10.1145/3328778.3372582>
- Price, I. (2019). *Student-Centered Vs. Teacher-Centered Classrooms: Which and Why?*. Iddblog. <https://www.iddblog.org/student-centered-vs-teacher-centered-classrooms-which-and-why/>
- Rana, M. S., Hossin, M. A., Mahmud, S. M. H., Jahan, H., Satter, A. K. M. Z., &

- Bhuiyan, T. (2019a). *MinFinder : A New Approach in Sorting Algorithm*.
- Rana, M. S., Hossin, M. A., Mahmud, S. M. H., Jahan, H., Satter, A. K. M. Z., & Bhuiyan, T. (2019b). *MinFinder : A New Approach in Sorting Algorithm*. 131, 2.1. Types of Sorting Algorithms, paragraph: 2.
<https://doi.org/10.1016/j.procs.2019.06.020>
- Sa, L. L. (2018). *The Basics of Computational Thinking*. <https://themekeeper.com/web-design/the-basics-of-computational-thinking>
- Sahin, C., & Abichandani, P. (2013). Should the first course in computational problem solving and programming be student-centered or teacher-centered? *Proceedings - Frontiers in Education Conference, FIE, May*, 748–754.
<https://doi.org/10.1109/FIE.2013.6684926>
- Schwebel, D. C., Morrongiello, B. A., Davis, A. L., Stewart, J., & Bell, M. (2012). The blue dog: Evaluation of an interactive software program to teach young children how to interact safely with dogs. *Journal of Pediatric Psychology*, 37(3), 272–281.
<https://doi.org/10.1093/jpepsy/jsr102>
- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2019). Demystifying Computational Thinking. *Researchgate*, 509, 1–21.
- Siti Rosminah, M. D., & Ahmad Zamzuri, M. A. (2012). Difficulties in learning Programming: Views of students. *1st International Conference on Current Issues in Education (ICCIE2012), September*, 74–78.
<https://doi.org/10.13140/2.1.1055.7441>
- Smith, A., Mott, B., & Taylor, S. (2020). *Toward a Block-Based Programming Approach to Interactive Storytelling for Upper Elementary Students*. 1–10.
- Spraul, V. A. (2013). THINK LIKE A PROGRAMMER. *Journal of Chemical Information and Modeling*, 53(9), 1689–1699.
- Statter, D., & Armoni, M. (2020). Teaching abstraction in computer science to 7th grade

students. *ACM Transactions on Computing Education*, 20(1), 8–837.

<https://doi.org/10.1145/3372143>

- Subramaniam, S., Chua, F. F., & Chan, G. Y. (2017). Project-based Learning for Software Engineering—An Implementation Framework. *Journal of Telecommunication, Electronic and Computer Engineering*, 9(3-4 Special Issue), 81–85.
- Szabó, Z. (2020). Problem solving and interrelation of concepts in teaching algorithmic thinking and programming. *CEUR Workshop Proceedings*, 2650, 318–327.
- Tariq, I. (2020). *An interactive interface for shader programming*. May.
- THE ROYAL SOCIETY. (2017). *After the reboot : computing education in UK schools Contents*.
- Threekunprapa, A., & Yasri, P. (2020). Patterns of computational thinking development while solving unplugged coding activities coupled with the 3s approach for self-directed learning. *European Journal of Educational Research*, 9(3), 1025–1045. <https://doi.org/10.12973/EU-JER.9.2.1025>
- Tom, M. (2015). Five Cs Framework: A Student-centered Approach for teaching programming courses to students with diverse disciplinary background. *Journal of Learning Design*, 8(1), 21–37. <https://doi.org/10.5204/jld.v8i1.193>
- Tomaszewski, M. (2021). *Creative Thinking: Definition, Examples & How to Boost Creativity*. <https://zety.com/blog/creative-thinking-skills>
- Tsarava, K., Moeller, K., Pinkwart, N., Butz, M., Trautwein, U., & Ninaus, M. (2017). Training computational thinking: Game-based unplugged and plugged-in activities in primary school. *Proceedings of the 11th European Conference on Games Based Learning, ECGBL 2017, October*, 687–695.
- Turchi, T., Fogli, D., & Malizia, A. (2019). Fostering computational thinking through collaborative game-based learning. *Multimedia Tools and Applications*, 78(10),

13649–13673. <https://doi.org/10.1007/s11042-019-7229-9>

- Van Dyne, M., & Braun, J. (2014). Effectiveness of a Computational Thinking (CS0) course on student analytical skills. *SIGCSE 2014 - Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 133–137. <https://doi.org/10.1145/2538862.2538956>
- Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4), 715–728. <https://doi.org/10.1007/s10639-015-9412-6>
- Voskoglou, M. (2013). Problem solving, fuzzy logic and computational thinking. *Egyptian Computer Science Journal*, 37(1), 131–145. <http://www.ecsjournal.org/Archive/Volume37/Issue1/8.pdf>
- Voskoglou, M. G., & Buckley, S. (2012a). *Problem Solving and Computational Thinking in a Learning Environment*. 36(4), 28–46.
- Voskoglou, M. G., & Buckley, S. (2012b). *Problem Solving and Computational Thinking in a Learning Environment*. 36(4), 28–46. <http://arxiv.org/abs/1212.0750>
- Wang, W., Zhi, R., Milliken, A., Lytle, N., & Price, V. W. (2020). Crescendo: Engaging students to self-paced programming practices. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 859–865. <https://doi.org/10.1145/3328778.3366919>
- Ward, B., Marghitu, D., Bell, T., & Lambert, L. (2010). Teaching computer science concepts in Scratch and Alice. *Journal of Computing Sciences in Colleges*, 26, 173–180. [https://www.dropbox.com/sh/628r06wzowu1py6/4bA9m7cJ3PWard et al 2010 Teaching CS concepts in Scratch and Alice-1546874624/Ward et al 2010 Teaching CS concepts in Scratch and Alice.pdf](https://www.dropbox.com/sh/628r06wzowu1py6/4bA9m7cJ3PWard%20et%20al%202010%20Teaching%20CS%20concepts%20in%20Scratch%20and%20Alice-1546874624/Ward%20et%20al%202010%20Teaching%20CS%20concepts%20in%20Scratch%20and%20Alice.pdf)
- Wing, J. M. (2006). Computational Thinking. *Computer Science Handbook, Second*

Edition, 49(3), 68-1-68–18. <https://doi.org/10.1201/b16812-43>

- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717–3725. <https://doi.org/10.1098/rsta.2008.0118>
- Wing, J. M. (2012). *Computational Thinking* (M. A. F. Summit (ed.)). Microsoft.
- Wohlfarth, D., Sheras, D., Bennett, J. L., Simon, B., Pimentel, J. H., & Gabel, L. E. (2008). Student Perceptions of Teaching Transparency. *The Journal of Effective Teaching*, 7(2), 36–50. <http://uncw.edu/cte/et/>
- Wong, G. K. W., & Cheung, H. Y. (2020). Exploring children’s perceptions of developing twenty-first century skills through computational thinking and programming. *Interactive Learning Environments*, 28(4), 438–450. <https://doi.org/10.1080/10494820.2018.1534245>
- Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., Tan, A. H., Hwa, L., Li, M., & Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education*, 29(2–3), 205–253. <https://doi.org/10.1080/08993408.2019.1565235>
- Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education*, 14(1). <https://doi.org/10.1145/2576872>
- Yang, T. C., Hwang, G. J., Yang, S. J. H., & Hwang, G. H. (2015). A two-tier test-based approach to improving students’ computer-programming skills in a web-based learning environment. *Educational Technology and Society*, 18(1), 198–210.
- Yasmin, B., Debora, A., Proctor, C., Kafai, Y. B., Proctor, C., & Lui, D. A. (2019). Framing Computational Thinking for Computational Literacies in K-12 Education. *Proceedings of the Weizenbaum Conference 2019 Challenges of Digital Inequality*,

7. <https://doi.org/10.34669/wi.cp/2.21>

Yew, E. H. J., & Goh, K. (2016a). Problem-Based Learning: An Overview of its Process and Impact on Learning. *Health Professions Education*, 2(2), 75–79. <https://doi.org/10.1016/j.hpe.2016.01.004>

Yew, E. H. J., & Goh, K. (2016b). Problem-Based Learning: An Overview of its Process and Impact on Learning. *Health Professions Education*, 2(2), 75–79. <https://doi.org/10.1016/j.hpe.2016.01.004>

Zaharin, N. L., Sharif, S., & Mariappan, M. (2018). Computational Thinking: A Strategy for Developing Problem Solving Skills and Higher Order Thinking Skills (HOTS). *International Journal of Academic Research in Business and Social Sciences*, 8(10). <https://doi.org/10.6007/ijarbss/v8-i10/5297>

Zhang, L. C., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers and Education*, 141(June), 103607. <https://doi.org/10.1016/j.compedu.2019.103607>

Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills? *Computers and Education*, 141(July), 103633. <https://doi.org/10.1016/j.compedu.2019.103633>

Appendix A

Student ID	Pre-Test	Post Test
01	4	3.5
02	5.1	5.8
03	6.1	6.1