WAREHOUSE ESCAPE: AN INTERACTIVE GAME TO TEACH SEARCH AND SORT ALGORITHMS

MAIMOONA BASHIR
Bachelor of Science In Computer Science, Comsats University Islamabad, 2015

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

WAREHOUSE ESCAPE: AN INTERACTIVE GAME TO TEACH SEARCH AND SORT ALGORITHMS

MAIMOONA BASHIR

Date of Defence: May 16, 2022

Dr. J. Anvik Thesis Supervisor	Associate Professor	Ph.D.
Dr. W. Osborn Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. J. Zhang Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. Y. Chali Chair Thesis Examination Committee	Professor	Ph.D.

Dedication

I dedicate this thesis to my darling daughter *AYAT*. Thank you for shaping me into the person I am today. I feel your presence around me every day. You are the reason for my perseverance in tough times, and you are the one who have made me stronger than I could have ever imagined.

Abstract

In computer science, algorithms can be one of the most challenging concepts for beginners to learn. One approach to teaching challenging concepts is the use of game-based learning (GBL).

In this work, we investigate the use of game-based learning for teaching fundamental search and sort algorithms. Specifically, we created the Warehouse Escape game.

To assess the knowledge improvements from playing Warehouse escape, we compare playing the game to watching animations of the algorithms. We found that those who played the game showed a significant improvement in learning outcomes than those who watched the animations. Also, those who played the game did better on conceptual questions.

Based on this work, the more "hands-on" approach of game-based learning is more effective and engaging for learning algorithms than the more passive watching of animations.

Acknowledgments

Firstly, I would like to thank Almighty God for giving me strength and showering me with blessings. Without His blessing, I would not be able to complete this master's program.

I owe much to my supervisor Dr. John Anvik, and I would like to convey my deepest gratitude to him. He provided me with sincere advice, encouragement, and support not only during research but also during coursework. He guided me to the fullest extent of his ability even during the most challenging times and gave me the motivation to perform to my maximum ability. I was very fortunate to have been working under his supervision. Dr. Anvik, I thank you from the bottom of my heart.

I am extremely grateful to my M.Sc. supervisory committee members, Dr. Wendy Osborn and Dr. John Zhang, for their valuable feedback and time. I want to extend my appreciation to my outstanding 'Sibyl Lab' members for their support and encouragement throughout my program and especially for helping me review my thesis during job action disruption.

I am incredibly thankful to my mother (Salma) and father (Kh. Bashir Ahmed) for all the love, support, encouragement and prayers they have sent on my way during this journey. Thank you for believing in me, Dad, and for letting me spread my wings against all odds. I hope that I have made you proud. I want to thank my brother (Md. Abdullah) for always being there whenever I needed his assistance. To my sister (Masooma), niece (Duaa) and nephew (Azan), thank you for the comfort and emotional support during the stressful time.

I would also like to express my special thanks to my husband (Arslan) for moti-

vating me in my lows and supporting me throughout this journey. It would not have been possible without his support.

Finally, I am grateful to the School of Graduate Studies (SGS) for providing financial assistance due to COVID-19 extension for my graduate study and the Natural Science and Engineering Research Council of Canada (NSERC) for funding my research work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Games for education are widely available on the web and mobile devices, but they use basic programming or Computer Science (CS) terminology to teach algorithm concepts. Having no prior knowledge of coding or such terminologies makes it more difficult for the novice learner/student to understand the concepts [10, 13, 19]. Furthermore, different games use different languages to teach CS concepts. Some of them are C, Java, C++ and Python, all of which have different syntax. Therefore, a beginner may find it challenging to understand all the intricacies of a language; as a result, they lose the motivation to learn.

Although algorithms are one of the fundamental concepts in programming, it can be difficult for beginners to grasp the concept of algorithms. One of the reasons why algorithms are difficult to understand is because they use abstract concepts. [19]. As a result, they find algorithms challenging to learn and feel demotivated. Game-based learning (GBL) for algorithms can help understand the concepts, and applying them can help to address this issue.

Teachers constantly seek ways to improve their methodology, facilitate student learning, and keep students' attention. Byrne et al. concluded from their research that animation helps students solve algorithms' procedural problems [4]. Other research demonstrates that today's youth have more difficulty concentrating than students a few years ago. To elaborate on this statement, the researchers mentioned that students today are so accustomed to distraction and bombardment with media images that

they find it harder to concentrate than students in the past [20]. Students may find visualizations, interactive animations, and educational games helpful in focusing their attention on the visualized processes [9].

The growth of pervasive games[1] as an industry and research field shows that games and game technologies have surpassed the traditional limitations of their medium. In recent years, the term 'gamification' has emerged as a broader term to describe the use of elements of video games (instead of traditional games) to improve user engagement and experience with non-game services and applications [7].

In addition to entertainment, games can be used for educational purposes, learning, investigation, discovery, innovation, simulation of complex phenomena, and more. It includes game-based learning, which utilizes different games for learning and educational purposes. Students/learners gain knowledge and hands-on skills through serious games (i.e. games designed for a primary purpose other than pure entertainment) and exploration. In the context of game-based learning, edutainment refers to learning that is primarily aimed at educating and partially entertaining [18].

Some universities offer courses related to serious games or gamification, such as; Massachusetts Institute of Technology's Computer Games and Simulations for Education and Exploration course [6], the University of Southern California's Game Development course [8], and University of Pennsylvania's, Gamification course [12]. In addition to courses, many universities offer master's degree programs in gamification, such as Michigan State University, University of Barcelona, and University of Technology Sydney [17].

Our proposed work, the "Warehouse Escape" game, has interactive screens, enabling players to learn searching and sorting algorithms by actually devising creative solutions to the problems presented, allowing students to develop logical skills. In particular, this game teaches the concepts of linear search, binary search, insertion

[1] A game that uses real-world locations in gameplay. For example, people interact with virtual game characters in real-world locations using a mobile device.

GAME	SEARCH	SORT	Computational Complexity	QUIZ
WAREHOUSE ESCAPE	✓	✓	✓	✓
SORTED	X	Basic Sort	✓ Basic Level	X
INTERACTIVE ANIMATION	✓ Only Illustrations	✓ Only Illustrations	X	X
PROGRAM WARS	Basic Search	Basic Sort	X	X
PACMAN	✓ BFS, DFS A*	X	X	X

Table 1.1: Warehouse Escape Vs Other Algorithm Teaching-Games

sort, selection sort, quicksort and merge sort. The player interactively walks through these algorithms using operations such as *Select, Compare, Swap, Merge* and *Divide* with the intent of teaching them the logic behind search and sort algorithms. The game contains a quiz at the end of each level to assist the player in assessing their mastery of knowledge about an algorithm. After completing the quiz and completing all the tasks, two more quiz levels ask scenario-based questions to the player. The game's output provides each algorithm's number of moves, swaps, and comparisons to help the player think carefully about which search or sort to apply in different situations.

This research uses a game-based learning approach to examine how effective it is to learn search and sort algorithms. To determine the effectiveness of our approach, we conducted a user study. The effectiveness of Warehouse Escape will be determined by comparing the level of knowledge of the subject before and after playing the game or watching an animation.

Table 1.1 summarize the limitations of previous algorithm-teaching games and our work.

This research sets out to answer the following research questions:

RQ #1: Do the players' understanding of search and sort algorithms improve after playing the game?

RQ #2: What is more effective for learning algorithms, watching animations, or playing warehouse escape?

RQ #3: Does playing the game help more with answering practical or conceptual questions than watching animations?

The contributions of this work are:

- A mobile game for teaching basic sort and search algorithms.
- A user study of the game comparing the use of the game to that of watching algorithm animations.
- Results indicating that playing the game results in better learning outcomes than watching algorithm animations.

The rest of this thesis proceeds as follows. The related work is described in Chapter 2. A detailed description of the "Warehouse Escape" game is provided in Chapter 3. In Chapter 4, the details of the user study are explained. Chapter 5 analyzes the study results to provide answers to the research questions. Chapter 6 concludes this work.

Chapter 2

Related Work

Several research works have been conducted regarding games to teach computer programming, algorithms, and software engineering concepts.

The chapter begins with an overview of game-based learning and games for teaching algorithms, followed by reviewing previous work produced for teaching computer science concepts through games.

2.1 Game-based Learning

In recent years, a lot of work has been done on integrating game-based learning (GBL) in education. Game-based learning has been shown to result in a better understanding of difficult concepts [13, 19, 10]. It has been shown that educational computer games enhance the learning interest of the student and also increase their motivation to study deeper concepts [13]. With the advancement of technology, the traditional way of classroom teaching is slowly being replaced with e-learning and electronics like smartphones, e-books, tablets, and laptops.

In recent years, the interest in edutainment (any form of entertainment that is educational, which means education through games) has increased. There have been many research studies on the design and implementation of games for teaching the concepts of Computer Science, including online and offline games.

This section provides an overview of research work focusing on the teaching of algorithms.

2.2 Games for teaching Algorithms

2.2.1 Interactive Animation for Learning Sorting Algorithms

Interactive Animation for teaching and learning sorting algorithms [19] implemented the idea of teaching algorithms through visualization and animation. The focus was on two types of animations. The first was the collection of card sorting animations, and the second was the box sorting game. As shown in Figure 2.1[2] the right-side of the figure shows the box sorting game while the left-side shows the card sorting animation.

Card sorting animations contain five animations of sorting algorithms: Simple Exchange Sort, Bubble Sort, Insertion Sort, Minsort, and Maxsort. The task is to sort the playing cards in ascending order. The box sorting animation is a simple game where students are asked to sort three, five, or seven boxes in ascending order. The boxes have weight values hidden from the user, so the main goal is not to use unnecessary comparisons and sort the boxes in ascending order.

This game aims not to teach coding but to teach the basic concepts behind sorting algorithms. The evaluation of the results shows that the students recognized the essential aspects and the difference between sorting algorithms when they played the card sorting animation game. Furthermore, it was found that the number of unnecessary comparisons was also reduced after the students played the box sorting game. The result shows that visualization has a significant impact on learning and that game-based animations can be helpful in learning and teaching the concepts of algorithms.

How Warehouse Escape is different: The Interactive Animation game teaches the concepts of a simple exchange sort, such as the Bubble Sort, Insertion Sort, MinSort and MaxSort. It does not let the player explore the techniques for sorting;

[2]Images taken from `https://anim.ide.sk/sortingboxes.php` and `http://www.algoanim.ide.sk/`

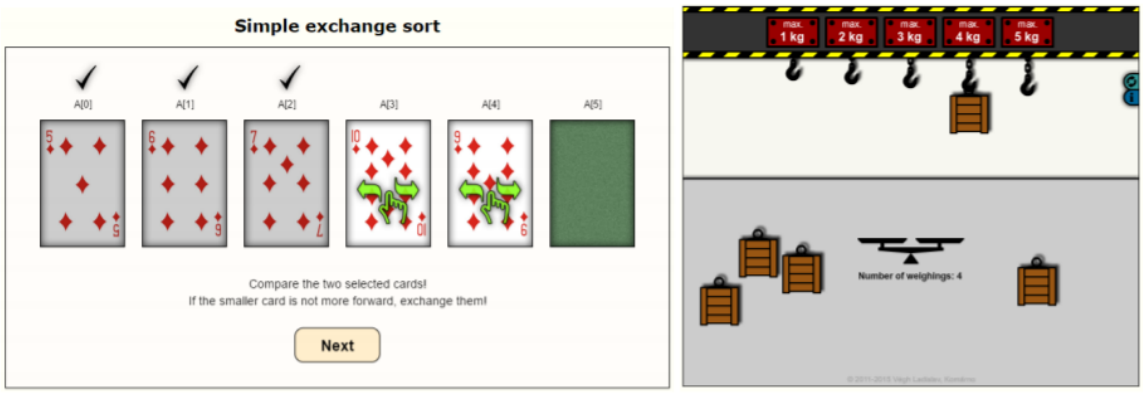


Figure 2.1: Animation for learning sorting algorithms

instead, it tells the step-by-step process to place the cards in sequence. Therefore, this game does not teach the time complexity of sorting algorithms, as there is no log or counter to check the number of moves played. Similarly, in the box sorting game, players are asked to sort boxes according to their weight by comparing only two boxes at a time. At the end of the game, players learn how to perform this task quickly, but they do not have any competition or quiz to check if there is any better technique than the one they performed in the game. The box sorting game also does not teach computational complexity.

In the Warehouse Escape, players explore different options by comparing and swapping. To gauge the student's understanding, there is a quiz after each task. The easy and medium scenarios in the game enable a competitive environment, and computational complexity is one of the objectives to learn in the Warehouse Escape. All these features are missing in the Interactive Animation game.

2.2.2 Pacman

Pacman [10] is an educational game[3], and it aims to assist and motivate students to learn and deeply understand Computational Intelligence search algorithms. This game provides an opportunity for the students to study and examine how search algorithms

[3]This is similar, but is not the same as the well-known arcade game.

operate. The game has two modes: an "Educational Mode" and a "Playing Mode." In "Educational mode", the student can select what type of algorithm they want to learn. There is a textual description, a graphical data flow and pseudo-code.

The game offers the student the opportunity to study the algorithm via visualizations. During visualization, Pacman can pause and ask the student what the next algorithm step is. The student can tell the game by selecting the next grid position. In the case of a correct answer, the game requests the student to justify their reasoning by offering a multiple-choice question.

Figure 2.2[4] shows the interface of the "Playing mode". The ghost is aiming to reach the Pacman to end the game. The student can move the Pacman in the maze to collect the dots and evade the ghosts and observe, while Pacman is moving, how the ghost applies a specific algorithm. Students can complete maze levels and proceed to the next ones that are more complex and challenging. Various levels are designed to necessitate students to apply a specific search algorithm. Students are requested to move the Pacman to reach the goal by moving Pacman based on a specific algorithm that the level specifies. The ghost would move faster towards the Pacman if an incorrect movement occurred. The student can only complete the level by correctly applying the algorithm and moving Pacman towards the goal.

As the student proceeds to the next level, the game gets more complex (i.e. maze characteristics increases, number of ghosts, goals to achieve, the complexity of the algorithm, and the parameters requested from students to apply).

How Warehouse Escape is different: The Pacman game teaches heuristic search algorithms such as *Breath-first search*, *Depth-first search*, and *A* search* (i.e. graph traversal algorithms). In contrast, the Warehouse Escape is designed to teach simpler search (i.e. linear and binary) and sort (i.e. bubble, insertion, selection, quick and merge) algorithms.

[4]Image taken from [10]

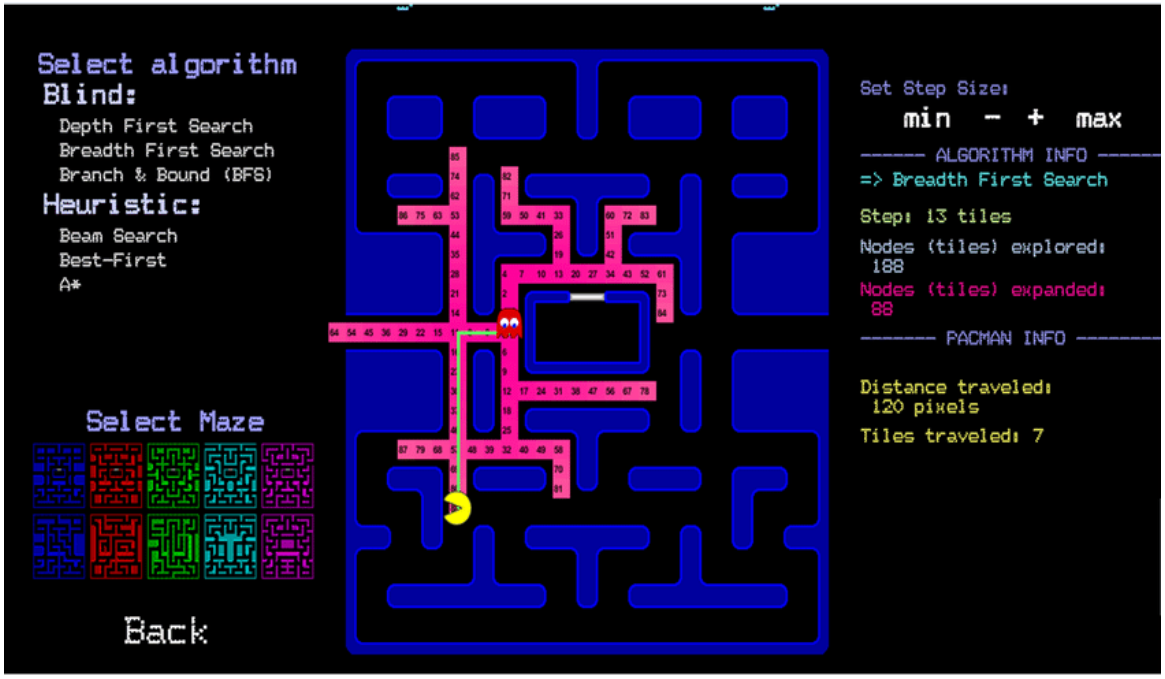


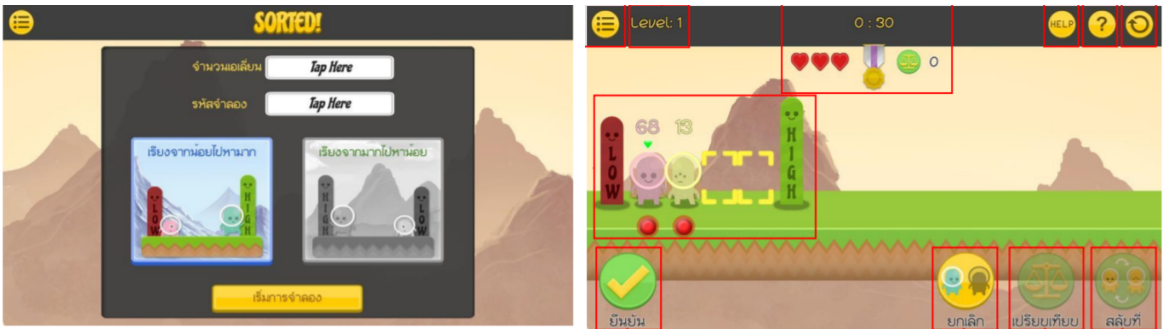
Figure 2.2: Pacman Educational Mode

2.2.3 Sorted

The objective of the Sorted game [13] is to facilitate a student-centred learning environment. The game mechanics used in designing the game are adopted from the execution processes of the sorting algorithms, including the objective of the algorithm, sorting orders (ascending and descending order), sorting operations (comparing and swapping), and the data structure (array). The player is assigned missions to rearrange disorderly standing aliens using comparing, swapping, and movement commands.

The game consists of the "Normal Mode" and "Simulation Mode." In "Normal Mode", everything is already given. The student has to follow the instructions and perform the task accordingly. It challenges the student with an adaptive difficulty level from easiest to hardest. The level of difficulty is determined by the total number of alien creatures generated. There are 15 levels, and each level contains two missions asking the player to sort aliens either in ascending or descending order. To unlock the next level, the player must finish at least one mission of the current level.

The main screen of the game is shown in Figure 2.3b [5]. The players score when they efficiently use comparing commands. There are three types of metal badges: gold, silver, and bronze. It is possible to achieve a gold medal badge if the number of times that comparison command is used is less than or equal to the total number of comparisons of the three-comparison based sorting algorithm (i.e. Bubble, Selection, and Insertion sort) that were executed on the same sequence of unsorted aliens.



(a) Simulation Mode (b) Sorted main screen

Figure 2.3: Game Sorted

The "Simulation Mode", as shown in Figure 2.3a [6], allows the players to customize the parameters like number the of aliens, sorting order (ascending or descending order) and seed number (a four-digit number that determines the characteristics of the sequence of the aliens including their values and positions).

How Warehouse Escape is different: The game Sorted is about sorting disorderly standing aliens in ascending or descending order by applying sorting operations, SWAP and COMPARE. The game's objective is to enable players to formulate their comparison-based or non-comparison-based sorting algorithm. However, in Warehouse Escape, players are instructed to apply a particular sort/ search algorithm and after completing a task, the player takes a quiz that tells the player's progress.

[5]Image taken from [13]
[6]Image taken from [13]

2.2.4 Map-Coloring Game

The Map-coloring game [11] teaches the basic concepts related to Constraint Satisfaction (CS), arc-consistency algorithms (AC-3 algorithms) and their combination with search algorithms (depth-first search with backtracking).

In this game, a map consisting of several regions is given, and the player needs to color each region of the map with one of a given set of colors so that no two adjacent regions have the same color. Figure 2.4 [7] shows the result of a map-coloring problem.

The game has two modes: "Educational Mode" and "Fun Mode." In "Educational Mode" the student can choose the number of colors and the complexity level of the game. They can also use help features to play the game. Whereas in "Fun Mode", the students can not select the difficulty level, as the game has predetermined levels to make the game more challenging. The game does not have any programming tasks, but it is designed in a way that students learn the concept of Constraint Satisfaction by completing the task according to the given constraint.

How Warehouse Escape is different: The Map-coloring game teaches how to solve Constraint Satisfaction problems. It is not directly linked with search and sorting algorithms, but a good search and sort technique can be helpful in solving the Constraint Satisfaction problems. Like in the Map-coloring game, searching for the best region for a color and then sorting the position of colored regions so that no adjacent vertices have the same color is a possible solution strategy.

2.2.5 Program Wars

The game Program Wars (PW) [2] is a web-based card game [1] for teaching concepts of programming and cybersecurity to beginners. It is a card game that teaches instructions, variables, loops, methods, malware and hacking. It can be played against a bot, or two human players can play each other.

[7] Image taken from [11]

Figure 2.4: Map-coloring final result

Figure 2.5[8] shows the interface of Program Wars, where a human player and a bot are playing the game. The player has to reach a score limit before an opponent and can win the game by applying good programming practices.

How Warehouse Escape is different: Program Wars teaches basic programming concepts through an online card game. Although it has search and sort cards included in the deck, it only teaches the effect of search and sort algorithms. In PW, a search card enables the player to search and pick any desired card from the deck and add it to their hand, while a sort card helps to rearrange the top five cards. In contrast, players learn to implement different search and sort algorithms in the Warehouse Escape. They also learn the time complexity of each algorithm.

[8]Image downloaded from `https://program-wars.firebaseapp.com/beginner` in February 2021

Figure 2.5: Program Wars interface

2.3 Games for Teaching Other CS Concepts

Board games are tabletop games where different pieces of the game are moved on the board according to the set of predefined rules of that particular game [3]. A few board games are discussed below, which teach different computer science concepts.

2.3.1 Code Master

Code Master [5] is a single-player board game that teaches logical concepts of programming, problem-solving, repeat statements (i.e. loops), conditional statements (i.e. if-else), unidirectional and bi-directional graph edges, and writing and executing a program. The game has 60 levels, with each level having six different patterns of difficulty from beginner and progressing through levels intermediate, advanced and expert.

The game's goal is to program your avatar (i.e. the player) to collect crystals (if any) and reach the exit portal. A player must finish the game within the limited number of the moves given on the scroll guide. To write the program, a colored action token (red, green, blue) and conditional tokens (for the advanced and expert levels) are placed on the scroll guide to move the avatar across similar lines on the map to reach the portal. Figure 2.6 shows a Code Master map, scroll guide, action tokens,

conditional token, crystals, portal (blue) and avatar (red).

Code Master teaches the logical thinking used to solve problems. Along with this, it introduces the concept of one-way and two-way paths (unidirectional and bi-directional edges on a graph), loops (by repeated statements) and conditional tokens (i.e. if-else statements). The player selects the scroll guide with an action token, and then "executes" their program on the map. This improves the writing of code and then simulates executing the code on a computer.

Figure 2.6: Board game Code Master

2.3.2 Potato Pirates

Potato Pirates [14] is a board game that teaches the basic concepts of computer programming. After playing the game, a player becomes familiar with variables, if-else statements, functions, for loops, while loops, nested loops, and switch-case statements. Also, some surprise cards teach about cybersecurity threats. Figure 2.7a shows all of

the cards, ships and a potato sack having a potato crew.

The game's goal is to gain all seven Potato King cards by drawing them from the deck or eliminating the potato crew of other players so that their ship sinks and their Potato King cards are seized. A player can win the game either by acquiring all of the Potato King cards or being the last player sailing.

The deck has action, control, surprise and Potato King cards. A player can play with action cards only, but to boost up the attack, control cards (i.e. loops and conditionals) are used with action cards. The game teaches for-loops, while loops and if-else statements by stacking the control and action cards. It also teaches the concept of nested loops by stacking two for-loop cards with an action card or by stacking a while card and a for-loop card with an action card, as shown in Figure 2.7b.

2.3.3 Robot Turtles

Robot Turtles [15] is a children's board game to teach how to write instructions (i.e. code) and how to execute them. This game also teaches the concept of functions, bugs and debugging.

The game consists of four robot turtle tiles in blue, green, pink and red colors. Each robot turtle has an identical-colored matching jewel and matching bug card. There are forty-four cards in the deck with eleven cards for each color (e.g. 11 blue cards for the blue robot turtle). For complex levels, some obstacles are introduced in the game (i.e. stone walls, ice walls and crates). To melt the ice wall, a laser card is introduced. A function frog card is introduced in complex levels to represent a move sequence (i.e. a function, procedure or method). Instead of using 3 or 4 cards for that sequence, the player can use one function frog card.

The game's main objective is to move the robot turtle to its matching colored jewel by using direction code cards (i.e. turn right, turn left and step forward code cards). Placing the code cards in the correct order teaches the concept of writing a

(a) Potato Pirates

(b) Potato Pirates loops, nested loops and conditional cards

Figure 2.7: Game Potato Pirate

code. Figure 2.8 shows the game in progress.

Figure 2.8: Robot turtle board game

If a player feels they have made a wrong move, they can tap the bug tile and shout "bug". Shouting a bug is the same as saying "do-over". The player can debug the card sequence (i.e. rewrite the code) by doing this. The game's complexity increases with each level, with the player having to overcome the obstacles by playing the laser card. The laser card melts the ice wall, while a crate can be pushed right, left or forward and a brick wall can not be moved or broken down.

2.4 Summary

Studies have shown that games and their components in education can enhance learning, engage students in the learning process, and allow the students to practice different software engineering techniques and apply them in various situations [16]. Results and evaluation from the research [10] show that game-based learning is effec-

tive for students and it enhances their problem-solving skills. Foteini Grivokostopoulou et al. [10] found that the We talked about games that teach algorithms and computer programming, and the research results show that the games positively impact students. They perform well with their motivation and engagement level being high, which helped them with learning the fundamental computer science concepts.

Chapter 3

Warehouse Escape

The Warehouse Escape game is designed so that it does not require prior knowledge of programming or algorithms. Anyone from a non-programming (could be CS) background can play the game and learn algorithm concepts. This game helps the player to understand the logic behind an algorithm, and they do not have to worry about the programming language syntax. In other words, the algorithm concepts a player learns from playing the game can be applied to any programming language.

3.1 Overview of Warehouse Escape

The Warehouse Escape Game is a mobile game[9] for teaching the fundamentals of search and sorting algorithms. The main screen of the game is shown in Figure 3.2. The game consists of two modes- one is to learn the algorithms called the "Learning," and the other is the "Play Game" mode where players complete given tasks. The rest of this chapter describes these two modes. The Figure 3.1 illustrates the game flow of the Warehouse Escape.

3.1.1 Learning Mode

The "Learning" mode contains information about the search and sort algorithms. When a player clicks on a button with the name of an algorithm, it will display a textual description along with visual examples of that algorithm. Players can read the text before playing the game or they can also move forward to the "Play Game"

[9]The game is developed in Xamarin, a cross-platform for Android and iOS.

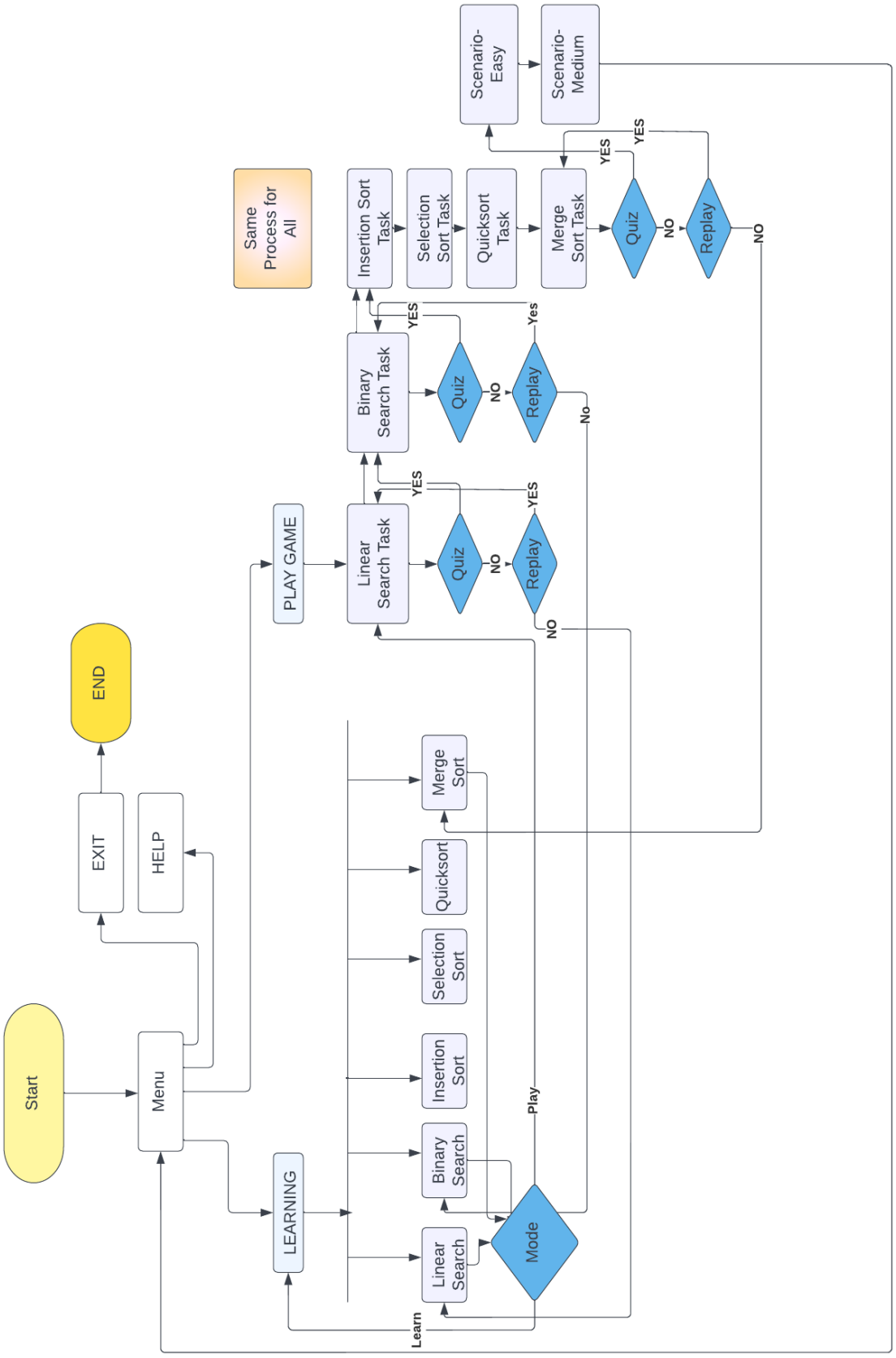


Figure 3.1: Warehouse Escape Flowchart

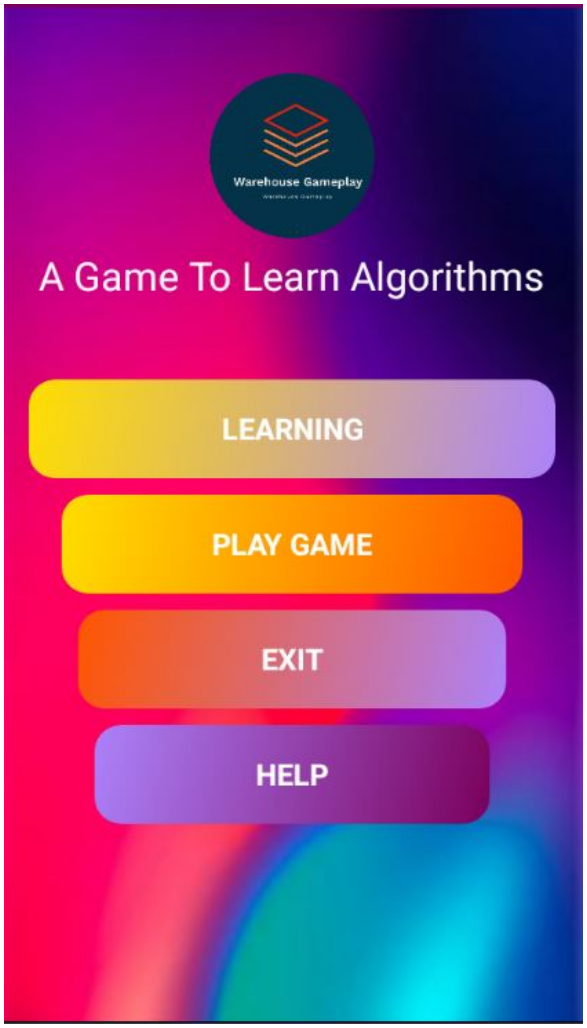


Figure 3.2: Warehouse Escape Main Screen

mode and start playing the game. During playing the game, they can go back to the "Learning" mode as every task has the button which takes the player to the "Learning" mode. The "Play Game" mode has a help button, which has the instructions about that specific search / sort algorithm which they are playing. If they need help understanding an algorithm during the game, they can switch to the "Learning" mode anytime. There is, however, no requirement to learn about the algorithm before starting the game. The player does not need to know the algorithm to play and can follow the instructions from the help button. Figure 3.3 is the main screen of learning mode.

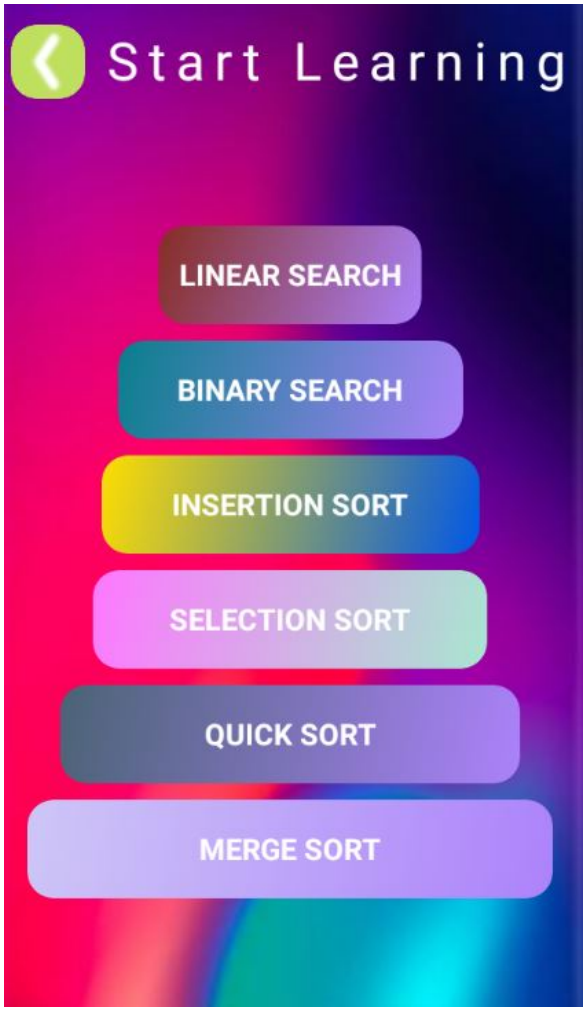


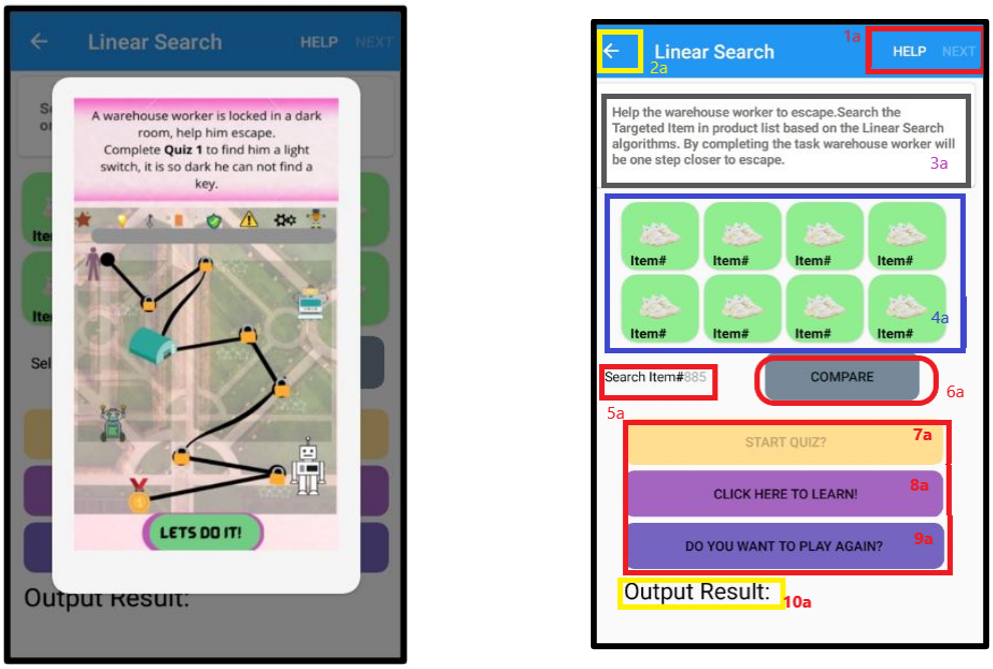
Figure 3.3: Learning Mode

3.1.2 Play Game Mode:

Different levels in "Play Game" correspond to each algorithm. At first, all the levels but the first one, are locked. Players can unlock each level by completing the tasks of the previous level. After completing a task, the number of comparisons, number of swaps and list of items of the search/sort algorithm will be displayed on the screen. This allows the player to compare different algorithms and see which algorithm is more efficient in terms of searching, sorting and computational complexity. Each level has a quiz at the end, allowing players to judge their ability to understand the algorithm.

3.2 Game Narrative

The game's storyline revolves around a worker who is trapped in a warehouse and must escape. A pop-up appears before every task and states the problem and what the player has to do. For example, the pop-up for the first task is shown in Figure 3.4a. By clicking on , "Lets do it" the game will take the player to the first task, which is the linear search. Figure 3.4b shows the basic controls for the linear search task, all of the other search and sort tasks have similar controls.



(a) Story Pop-up

(b) Game Controls

Figure 3.4: Warehouse Escape Game Controls

The following describes all the game controls highlighted in Figure 3.4b.

Help: The help button (1a) is present at the top right of the page and has instructions about the current task, how to complete the task and some basic concept about the algorithm. The help pop up is shown in Figure 3.5.

Back Arrow: The back arrow (2a) takes the player to the previous page.

Task Panel: Instructions for the task are provided in the task panel (3a), so players can understand their task and play accordingly.

Figure 3.5: Help Pop-up

TYPE	BUTTONS	BUTTONS	BUTTONS	BUTTONS
LINEAR	COMPARE	-	-	-
BINARY	LOW	MID	HIGH	-
INSERTION	COMPARE	SWAP	-	-
SELECTION	SELECT	COMPARE	DIVIDE	-
QUICK	PIVOT	COMPARE	SWAP	DIVIDE
MERGE	DIVIDE	MERGE	SWAP	-

Table 3.1: Operation Buttons for Search and Sort Algorithms

Play Area: In (4a), the game elements are provided. For each task, the items will differ.

Targeted Item: The label "Search Item" (5a) is the item to be found in the given list (4a). It will be different for each task.

Operation Buttons: The compare button (6a) is the main button of this task. Depending on the task, different buttons will appear here. As the Figure shows, an example is for linear search; there is only the compare button. Other algorithms will have other buttons as well. Table 3.1 provides a listing of the operation buttons that appear for each algorithm.

Quiz : The quiz button (7a) starts the quiz. After completing the task, the quiz button will be enabled, and the player can do the quiz. The quiz consists of 4 questions related to the task that the player just completed. Figure 3.6 shows one of the four questions of the linear search quiz. The goal of the quiz is to help the player assess how well the player knows the algorithm they just learned.

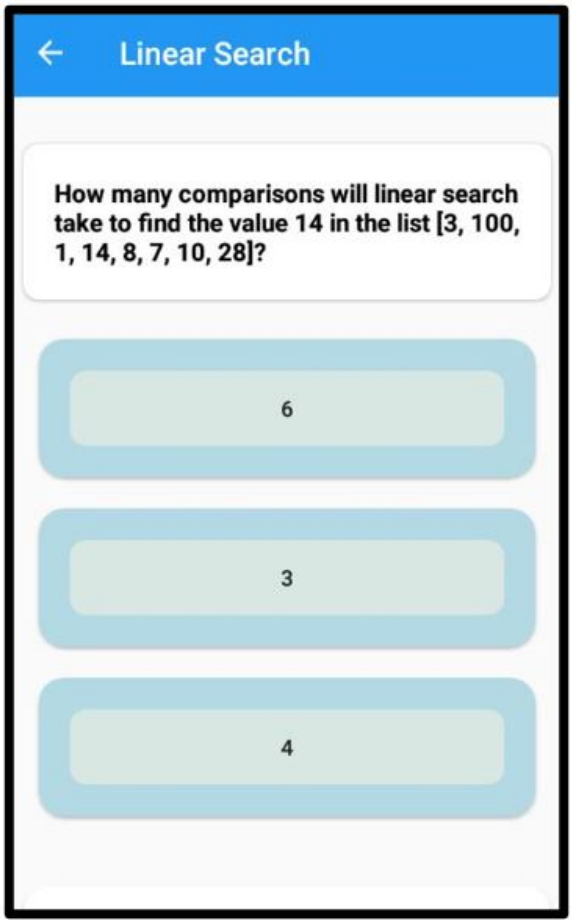
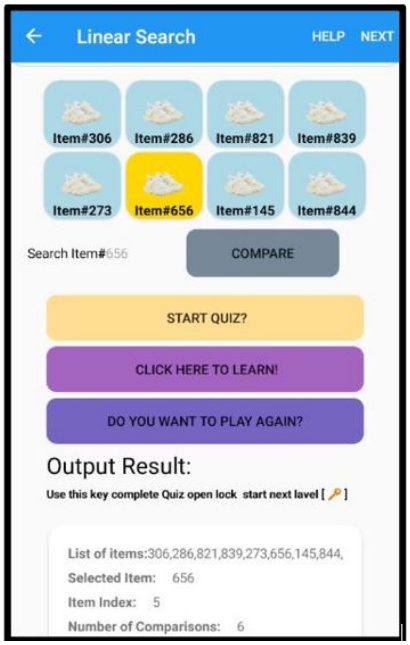


Figure 3.6: Linear Search Quiz

Learning Mode: The button 8a, takes the player to the learning page for the current search or sort algorithm.

Reload : The button (9a) reloads the task and enables the player to complete the task again with new values.

Output: Output (10a) displays the results of the current task. It shows the list of items, the numbers of swaps and the number of comparisons. After analyzing the results, the player can understand which search and sort algorithm takes fewer operations(number of swaps and comparisons) to perform a certain task.



(a) Linear Search Output

(b) Insertion Sort Output

Figure 3.7: Output Linear Search and Insertion Sort

Figure 3.7a shows an example of the linear search task along with the output screen. In the example, the player is asked to find Item #656 using the "Compare" operation. After the item #656 is found, all of the other items in the list became visible. The target item to be searched for, is highlighted in yellow. The output shows the list of items that were hidden at the start of the game. The selected item shows the item to be found. The item index is the index where the targeted item was

present, which is 5 [10]. The number of comparisons shows how many comparisons it took the player to find the targeted item.

Figure 3.7b shows the output screen for the insertion sort task. The list of items shows the original list before sorting. The number of swaps and comparisons varies depending on the item order and sorting algorithm. The last list shows the sorted items after applying the algorithm.

3.3 Warehouse Escape Walk-through:

The algorithms in the game consist of two searches (linear search and binary search) and four sorts (insertion sort, selection sort, merge sort and quicksort) algorithms. There is a quiz at the end of each task containing four questions. After completing all of the six algorithms, there are two more levels - Easy and Medium named "Scenarios - Easy" and "Scenarios - Medium." Scenarios - Easy has four scenario-based questions involving linear and binary search. The "Scenarios - Medium" has four scenario-based questions related to using sort and search together.

3.3.1 Linear Search

Linear search is the first algorithm taught in the game. The linear search gameplay area is shown in Figure 3.8. As we know, the linear search algorithm searches the data in sequential order. In other words, each item is examined starting from the first until a match is found. The *COMPARE* operation compares every item one by one until it finds the targeted item. After completing the task, the player has to complete the quiz as shown in Figure 3.6. It has four questions related to the algorithm the player just acted out. The *Next* button takes the player to the binary search algorithm.

[10] As the index starts from 0.

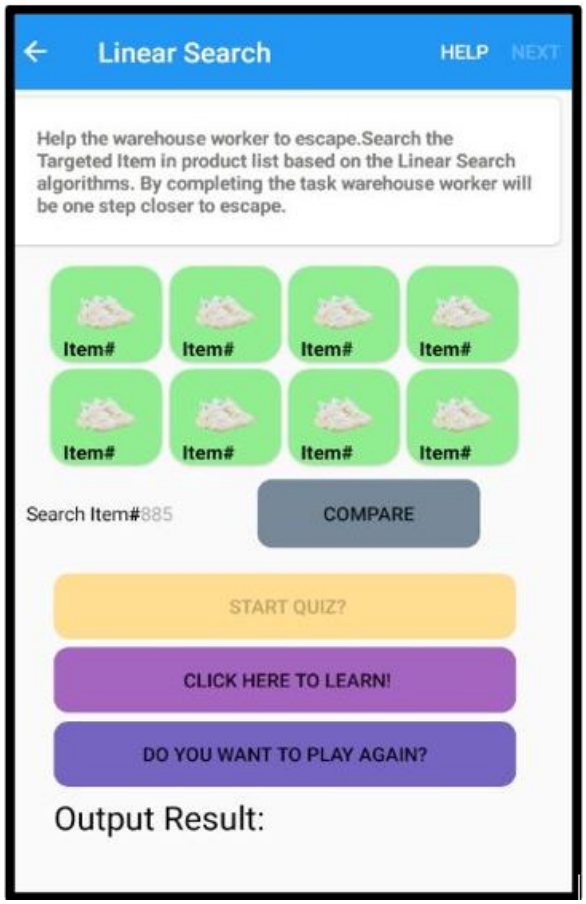


Figure 3.8: Linear Search

3.3.2 Binary Search

A binary search algorithm searches an item in a sorted list by repeatedly dividing the list in half. The Figure 3.9 shows the binary search game screen. By clicking on *LOW*, it will show the first element in the list. Clicking *HIGH* will select the last element in the currently searched (sub)list. Finally, clicking *MID* shows the middle element of the (sub)list. The next time, the buttons will work on either the first half (before midpoint) or the second half (after midpoint). The item number to be searched is given below the list of items.

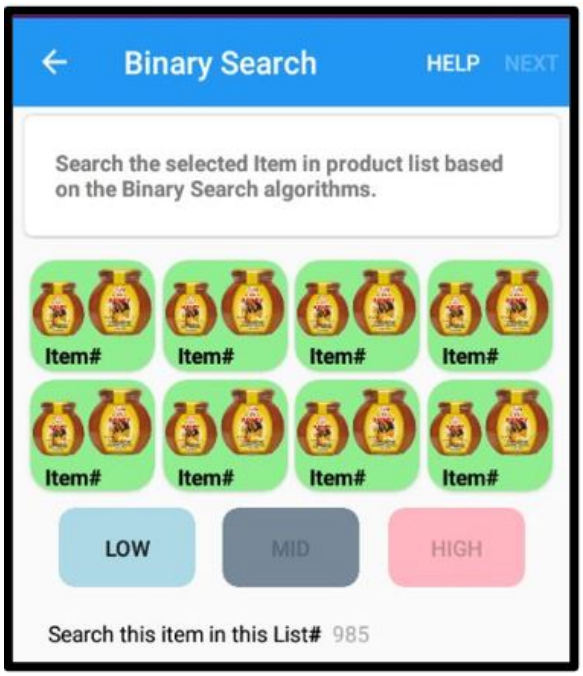


Figure 3.9: Binary Search

3.3.3 Insertion Sort

Insertion Sort has two buttons, *COMPARE* and *SWAP* to play the game and sort the list. The play screen is shown in Figure 3.10.

In insertion sort, the array is virtually split into a sorted and an unsorted part. The unsorted items are picked up and placed in the sorted part correctly.

The *COMPARE ITEM* button compares the currently selected item with its predecessor in the list. If the selected item is smaller than its predecessor, compare it with those items before it. The items are moved one position down in the list to make space for the swapped item.

Each time the *COMPARE ITEM* operation is run, two items are compared, starting from the first two items in the list. The *SWAP ITEM* operation will enable if the second item is smaller or larger than its predecessor. If the second item is smaller, it will move the predecessor item down one index and place the smaller item (i.e. currently selected item) at the front. The swap button will not work if the currently selected (second) item is greater, but pressing the compare button will check the next

and the predecessor items. This process will continue until the list is sorted. After completing the game, the results screen will display the number of swaps and the number of comparisons as shown in Figure 3.7b.

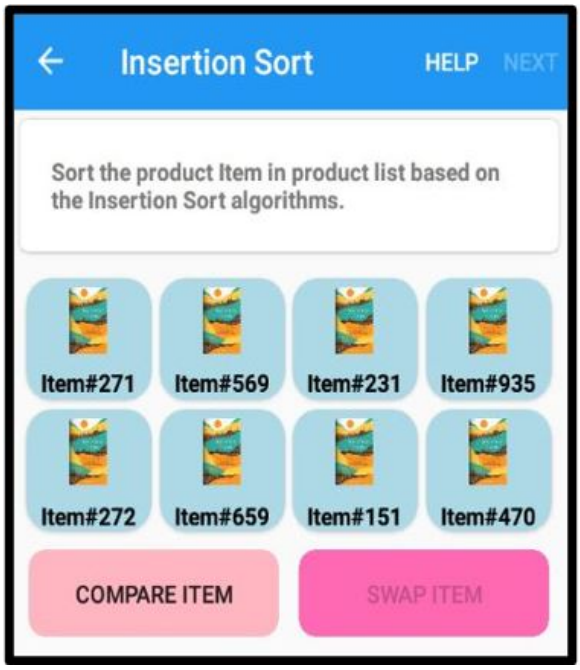


Figure 3.10: Insertion Sort

3.3.4 Selection Sort

Selection sort has three buttons for completing the task. *SELECT*, *COMPARE* and *SWAP* buttons as shown in Figure 3.11. The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning. The select button will select the minimum value. The compare button will compare that minimum value with the items in the list starting from the first item. Every time a new minimum value is selected, the position of the compare pointer will be incremented, and the pointer compares the smallest value to the next value. If the selected value is smaller than the compared value, the *SWAP* button will swap the compared value and selected values. This process will continue till the list is sorted.

After sorting is complete, the game result screen will show the number of swaps and comparisons. The player can analyze the result and compare swaps and comparisons of different sort algorithms.

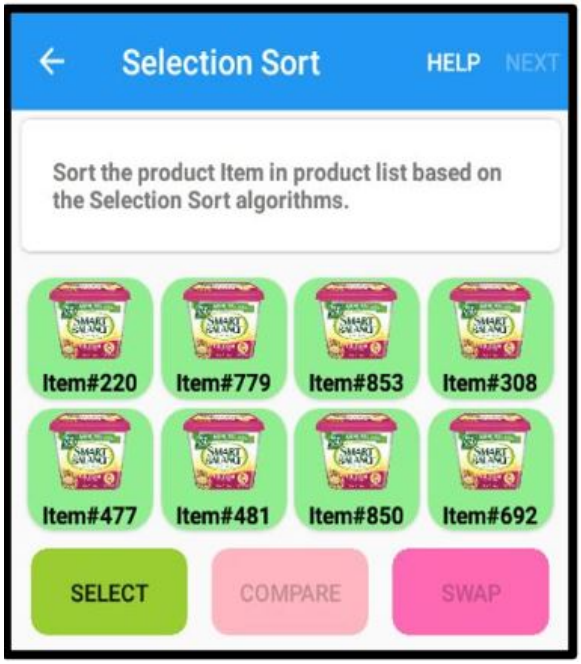


Figure 3.11: Selection Sort

3.3.5 Quicksort

Quicksort is a more complex algorithm than the other sort algorithms. There are four buttons for playing the task *PIVOT*, *COMPARE*, *SWAP* and *DIVIDE* as shown in Figure 3.12.

Quicksort is a Divide and Conquer algorithm. It picks an element as a pivot and partitions the given array around the picked pivot The *PIVOT* button will select the first element as a pivot. The *COMPARE* button will compare the elements in the list with pivot one by one, as the player clicks the *COMPARE* button. The *SWAP* button will swap the item to be compared with 'j' (element on the second index), which is initially the second element, but 'j' will increment one position forward every time an item next to the j is greater than the pivot item. After the first iteration and

the *DIVIDE* button will divide the list and places the pivot at the correct position. So, the elements on the left side of the pivot are less than the pivot, and the elements on the right side are greater than the pivot. After dividing the list again, the player clicks *SELECT* to select the pivot and continues dividing and swapping elements in the list until the list is sorted. The process repeats until all the elements in the list are in order. After completing the task, the results show the number of swaps and the number of comparisons.

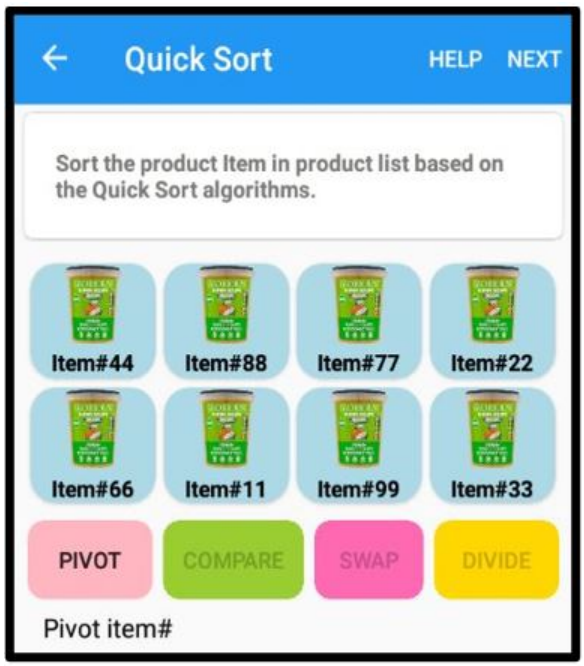


Figure 3.12: Quick Sort

3.3.6 Merge Sort

In this task, merge sort will be played with the two buttons *DIVIDE* and *MERGE* as shown in Figure 3.13.

The merge sort algorithm is a Divide and Conquer algorithm. It splits the list into two halves, repeatedly calls itself on each half until the list is one item, and then repeatedly merges the two sorted halves.

When clicking the *DIVIDE* button, the items in the list will be divided in half

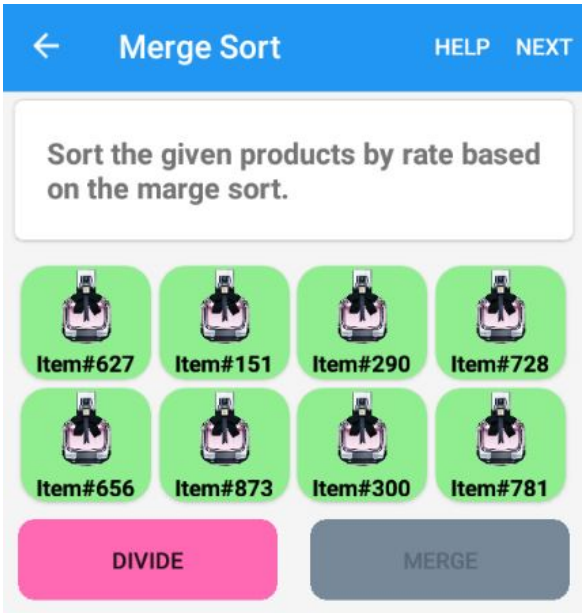


Figure 3.13: Merge Sort

until each sub-list has only a single element. For example, if a list has eight elements, the list will be divided into two sub-lists of four items after the first pass. After the second pass, the sub-lists containing four items each will form two new sub-lists containing two items each, and the last pass will further divide the list of two into a single item each. The entire process of the algorithm is shown in Figure 3.14.

Figure 3.14: Merge Sort Divide

By pressing the *MERGE*button, two list items will be merged and arranged in ascending order. For example, the list that was just divided and formed eight single items list, will be merged then. On the first call, *MERGE* will merge two items and

put them in order. On the second call, it will combine four items and then eight-item until the list is combined and sorted. The merge will combine all items in every iteration until a sorted list is formed. The entire process of merging back the list in order is shown in Figure 3.15. Players can visualize the list dividing and merging back as it will change the colour of the list every time the button merge or divide is pressed.

Figure 3.15: Merge Sort Process

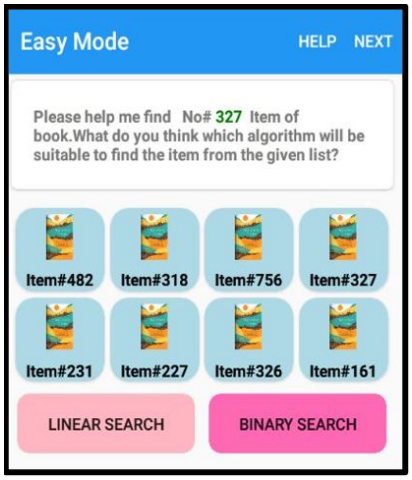
The results of this task, like those of the previous tasks, will be shown at the bottom of the page. After this sorting task has been completed, scenario-based questions will be given to check the improvement of the player's knowledge. The player must complete the task quiz before progressing to "Scenarios- Easy and Scenarios-Medium".

3.3.7 "Scenarios - Easy"

This level allows the player to test their knowledge of algorithms. The test beings with a pop-up after completing the merge sort, as shown in Figure 3.16a. The player then answers the scenario-based questions. There are four questions,about linear and binary search. An example of a question from "Scenarios - Easy" is shown in Figure 3.16b.

There are some questions in which the list is unordered. The player is asked to

(a) Scenarios- Easy Pop-up



(b) Easy mode Scenario-based Question

Figure 3.16: Scenarios- Easy

apply the search without mentioning that the list is out of order, so they are forced to analyze the visual representation of the list and respond accordingly. For example, in Figure 3.17 the list is not in order. As the player is asked to search for item 936, if the player does not look carefully at the list and then chooses to apply binary search, it will be the wrong answer, and the player will see a message that binary search cannot be applied in an unordered list.

On the other hand, in some questions, the item to be found is deliberately put in the middle, but the list is not in order. If the player selects binary search for the answer, the answer will be incorrect, and there will be a message saying that binary search cannot be applied to an unordered list.

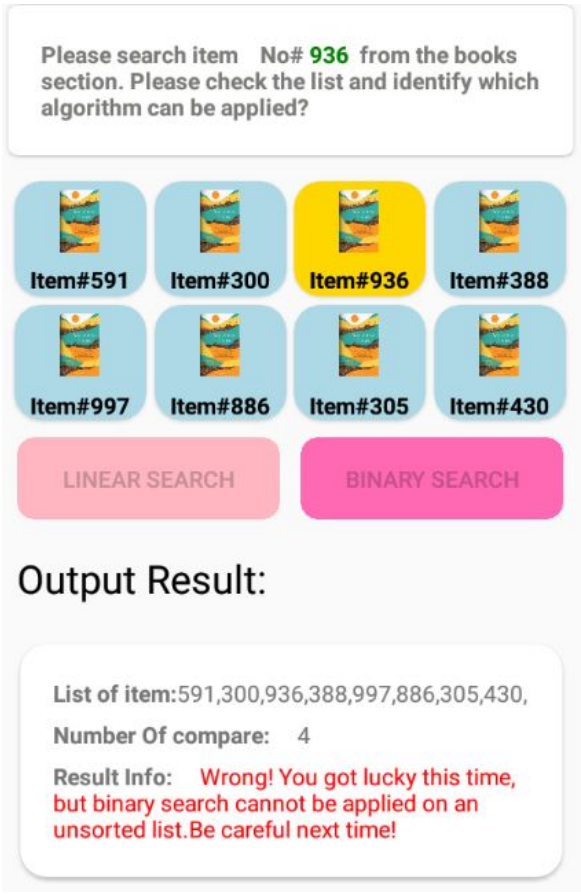


Figure 3.17: Easy Mode

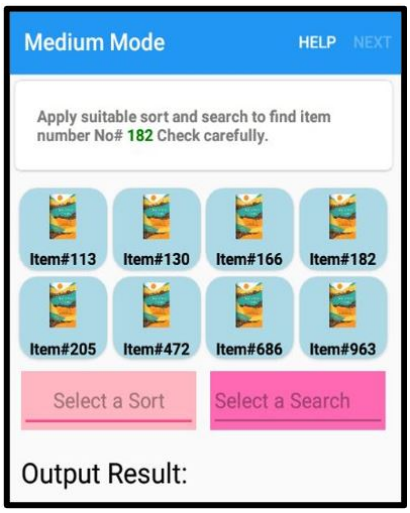
"Scenarios - Medium"

Similar to the "Scenarios - Easy", the "Scenarios - Medium" is also a knowledge test, but it is more challenging than the "Scenarios - Easy."

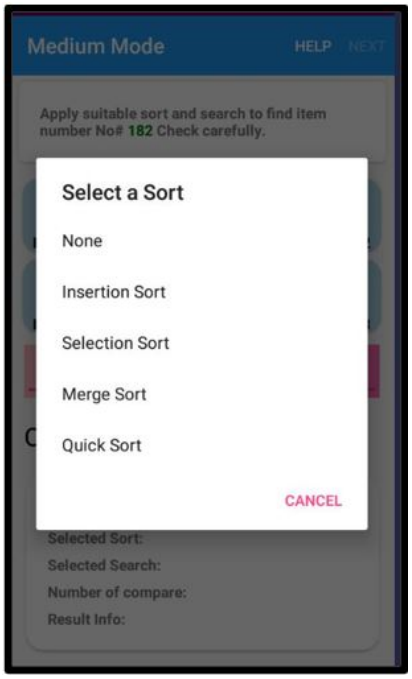
In "Scenarios - Medium", questions are asked based on combining search and sort algorithms as shown in Figure 3.18a. There are four questions in "Scenarios - Medium". After every question, two options are given in the form of drop-down menus, as shown in Figure 3.18b. A player can select both search and sort if applicable.

Some questions are designed so that they do not need a sort algorithm because the list given in the question is already sorted. If any search or sort is not applicable, the player can select "none". As shown in Figure 3.18b, it is the first option in the

drop-down menu.



(a) Medium Mode



(b) Medium Mode Drop-down

Figure 3.18: Medium Mode Images

When all four questions of the "Scenarios - Medium" have been answered, a message will appear stating that the player has completed the game and they are an "Algo Expert," as shown in Figure 3.19. At this point, the game is over, but the player can replay again if they want.

Figure 3.19: Game Completed

3.3.8 Summary

The Warehouse Escape game uses game-based learning (GBL) approach to teaching search and sort algorithms. This game is designed for novice students who may not have any prior knowledge about these algorithms.

Simple wording is used in the game to keep the player's focus on the algorithmic concepts. The game's purpose is to teach basic concepts of algorithms rather than coding. After the player grasps the concept, they can draw on this knowledge when programming algorithms in any programming language.

Computational complexity can be a difficult concept for novice students, therefore,

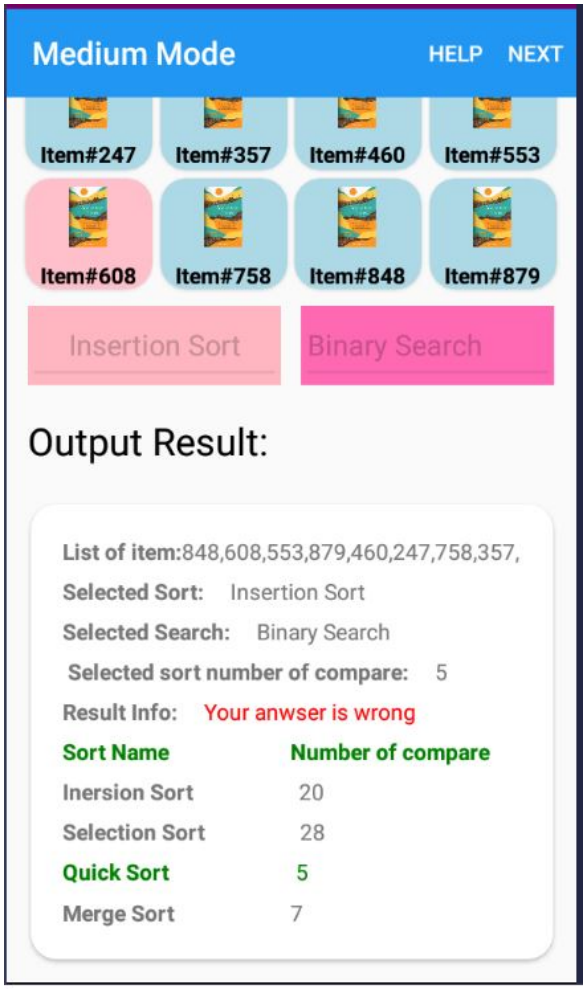


Figure 3.20: Output of Scenario-based Question

terms like swaps and comparisons are used in the Warehouse Escape Game. After every task has been completed, a result screen displays how many comparisons were required to complete the task using that specific algorithm. This concept is important for players as it allows them to decide which algorithm will be most efficient in different situations.

For example in Figure 3.20, the player answered the scenario-based question wrong and the output shows the correct answer with "the number of comparison" of all four sort algorithms.

The quiz feature helps players reinforce the information they learned while playing the game. They can forget what they learned if they don't repeat it, whereas

quizzes enable them to recall as much information as possible after completing a task. “Scenarios - Easy” and “Scenarios - Medium”assess player knowledge improvement.

Interactive screens make the game more engaging. The game play area highlights an element with each button press, so it is easy to understand what is happening in the task and how it works.

Students can learn the logical concepts of search and sort algorithms while enjoying playing the game with all these features.

Chapter 4

Evaluation Methodology of Warehouse Escape Game

To evaluate the game-based learning (GBL) approach used by the Warehouse Escape, we conducted a user study. The research methodology for the Warehouse Escape game, user study, is presented in this chapter.

4.1 Research Methodology

The selection criteria for participants in the study were those with little or no knowledge of searching and sorting algorithms. Participants booked an hour time slot for an online interview and observation session via Zoom[11]. The participants are split into two groups, the experimental and the control group. Participants in the experimental group played the game, and the control group watched various animations of search and sort algorithms.

First, the participants were asked to fill out a prior knowledge survey. Also, they were asked to complete a post-game/post animation survey. Based on the responses participants provided, the knowledge change of each participant was calculated. Later, the results were compared to see which group performed better and how their knowledge improved.

[11]Sessions were online due to COVID protocols in place at the time.

4.2 Details of User Study

The user study was conducted according to the Tri-Council Policy Guideline (TCPS2)[12] provided by the University of Alberta Research Ethics Committee. All materials from the user study, including prior and post-game survey questions, were analyzed and approved[13] by the committee. The survey was conducted using Qualtrics[14].

4.2.1 Participants

Participants involved in the study were recruited from a group of non-CS majors enrolled in CPSC 1000 course. To encourage participation, students who participated in the study were not required to complete some course activities related to searching and sorting algorithms.

Participants were recruited mainly through electronic communication. The students in the course were emailed an invitation (see Appendix B). Afterward, the principal investigator presented the study to the students in one of their lectures and encouraged them to participate in the study. Still, they were under no obligation to do so. The email sent to the class contained a link to a scheduler where participants could sign up for a 1-hour slot to participate in the study.

4.2.2 Study Procedure

The study session was conducted online over Zoom and was divided into five (05) parts as given below:

1. Introduction / Consent: At the start of a participant's session, they were given the consent form (see Appendix A), where a detailed description of the study was presented. Consent was obtained from the participant by clicking the appropriate button at the bottom of the form, as shown in Figure 4.1. Each participant was identified by a unique study ID, which replaced their full name and student ID. A

[12]https://ethics.gc.ca/eng/home.html
[13]Ethics Application Id: 2022-005
[14]https://uleth.qualtrics.com

unique study id was used to link the participants' pre and post-game responses. For those chosen to play the game, a demonstration of the mobile platform emulation software was given to run the game, and they were walked through the installation process.

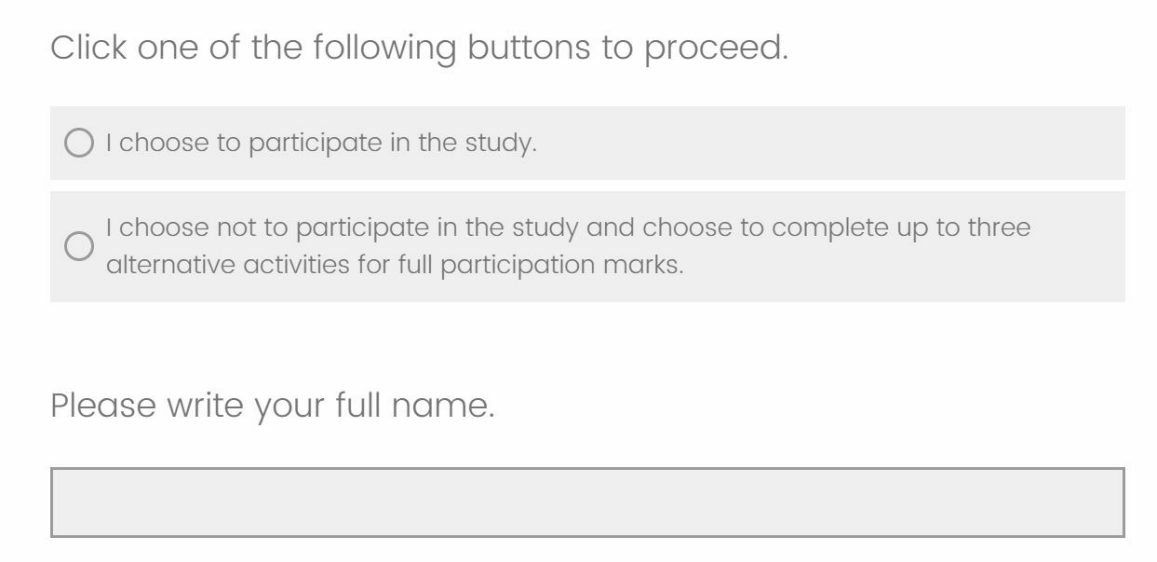


Figure 4.1: Consent Form

2. Prior Knowledge Survey: The participants were asked to complete a short Qualtrics survey. This questionnaire collected basic demographic information about the participant, their previous experience with algorithm concepts and some questions related to search and sort algorithms. The questions were presented as multiple-choice/true-false/fill-in-blank questions. The questions can be found in Appendix C.

If a participant did not complete the prior knowledge questionnaire or chose to end the interview and observation session, this was considered as the participant's decision to withdraw from the study. Their incomplete responses were not included in the final study results.

If the participant agreed to participate in the study, they were directed to a prior knowledge questionnaire. In the prior knowledge questionnaire, there was one (1) Demographic Question, three (3) Experience Questions and eighteen (18) Knowledge

Questions regarding the search and sort algorithm. Three questions were given for each search and sort algorithm, and there are two search algorithms and four sort algorithms.

3. Play game/watch animations: After completing the prior knowledge questionnaire, the participants were divided into two groups; one group played the Warehouse Escape game, and the other watched various animations of the searches and sorts. After every task, a quiz related to the completed task was given, regardless of if they played the game or watched the animation. Each player completed the quiz before moving forward. On average, it took 30 minutes to complete this stage.

4. Post knowledge survey: The participants answered some scenario-based algorithm questions to assess any changes in knowledge after completing the game or watching the animations. Both groups were given the same set of questions using the corresponding images from the game. The questionnaire had 18 questions, identical to those given in the prior knowledge questionnaire, three questions from the linear search, three questions from binary search, and three questions for each algorithm. After completing this stage, they were asked to provide feedback.

5. Debrief: Participants were asked to comment on their experience with playing the game or watching the animation and provide any additional feedback. Based on our observations, we also asked them some clarifying questions.

The full participation took around 60 minutes (5 minutes consent form, 10 minutes prior knowledge questionnaire, 30 minutes to play the Warehouse Escape and complete quizzes. Ten minutes to complete the post-game questionnaire and 5 minutes to debrief).

4.2.3 Assessing Knowledge Changes

To assess participant's knowledge gains about search and sort algorithms, we used the following rule:

PRE	POST	RULE
Correct	Incorrect	*Unclear*
Correct	Correct	*Knowledge same*
Incorrect	Correct	*Improved*
Incorrect	Incorrect	*No change*

Table 4.1: Categories for Knowledge Change

4.2.4 Knowledge Change Categories

To determine changes in knowledge about the algorithms, we compared the participants' answers to their prior knowledge and post-game or animation questions. Based on a comparison of their responses before and after, we categorized participants' knowledge changes into four categories as shown in Table 4.1.

a) Improved: If the pre-game/animation answer from a participant is 'Incorrect' and the post-game/animation answer is 'Correct,' it is assumed that their Knowledge has been improved. Accordingly, the knowledge change is considered 'Improved.'

b) No Change: It is assumed that a participant's Knowledge has not changed by playing the game or watching the animation if their pre-game/animation statement and post-game/animation statement are both 'Incorrect.' Hence, the knowledge change is recorded as 'No Change.'

c) Same Knowledge: If both of the pre-and post-game/animation answers are 'Correct,' then it will be assumed that the participant has previously known the concept, and their knowledge change will be marked as 'Same Knowledge.'

(d) Unclear: If a participant answers a pre-game/animation question correctly and then provides an incorrect response after playing a game or watching the animation, then it is unclear if the participant's Knowledge changed. Thus, we mark the knowledge change as 'Unclear.'

4.2.5 Survey Questions

The pre-game and post-game questionnaires of the user study asked several questions. The following are the details of the questions:

Demographic and Previous Experience Questions:

The participants were asked one (01) demographic question about their educational background and three (03) questions about their prior knowledge of search and sort algorithms. Refer to Appendix C for the specific demographic questions.

Descriptive Question

The participants were asked several descriptive questions in the prior knowledge and post-game questionnaire for each search and sort algorithm. Some examples of the descriptive questions for binary search, insertion sort and quicksort are shown in Figure 4.2, 4.3 and 4.4. The complete set of questions are found in Appendix C

Figure 4.2: Binary Search Descriptive Question

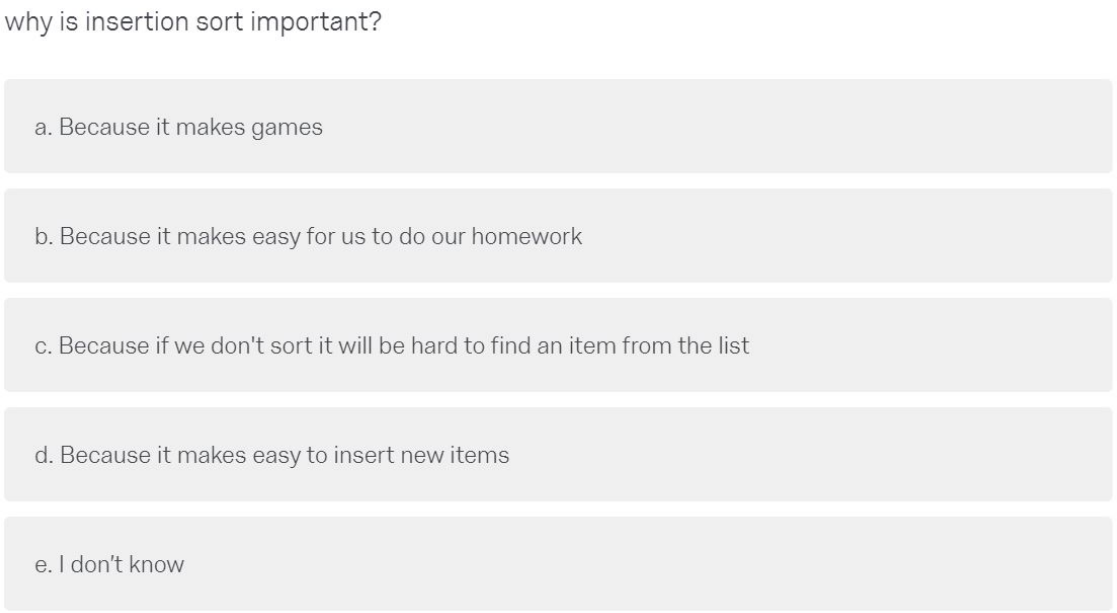


Figure 4.3: Insertion Sort Descriptive Question

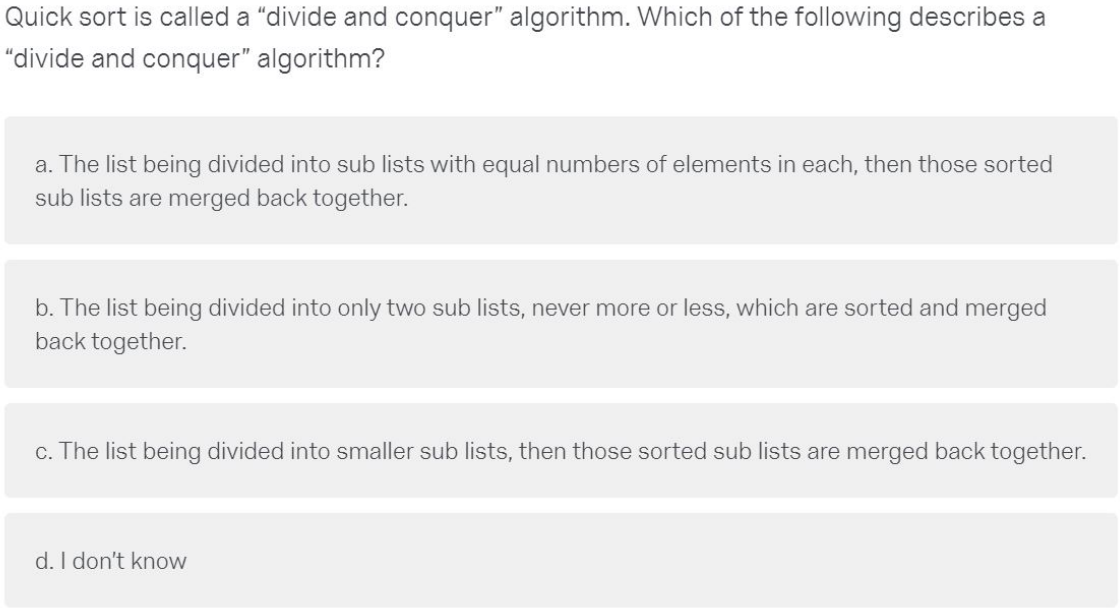


Figure 4.4: Quick Sort Descriptive Question

Problem-solving Questions

Problem-solving questions were among the types of questions asked in prior knowledge and post-game survey. A task is given in these questions, and the player is asked to solve it and select the correct answer. A list of numbers is given in most such questions, and the player is asked to search or sort it using a specific algorithm. Some of the examples of the problem-solving questions for binary search, insertion sort and selection sort are shown in Figure 4.5, 4.6 and 4.7

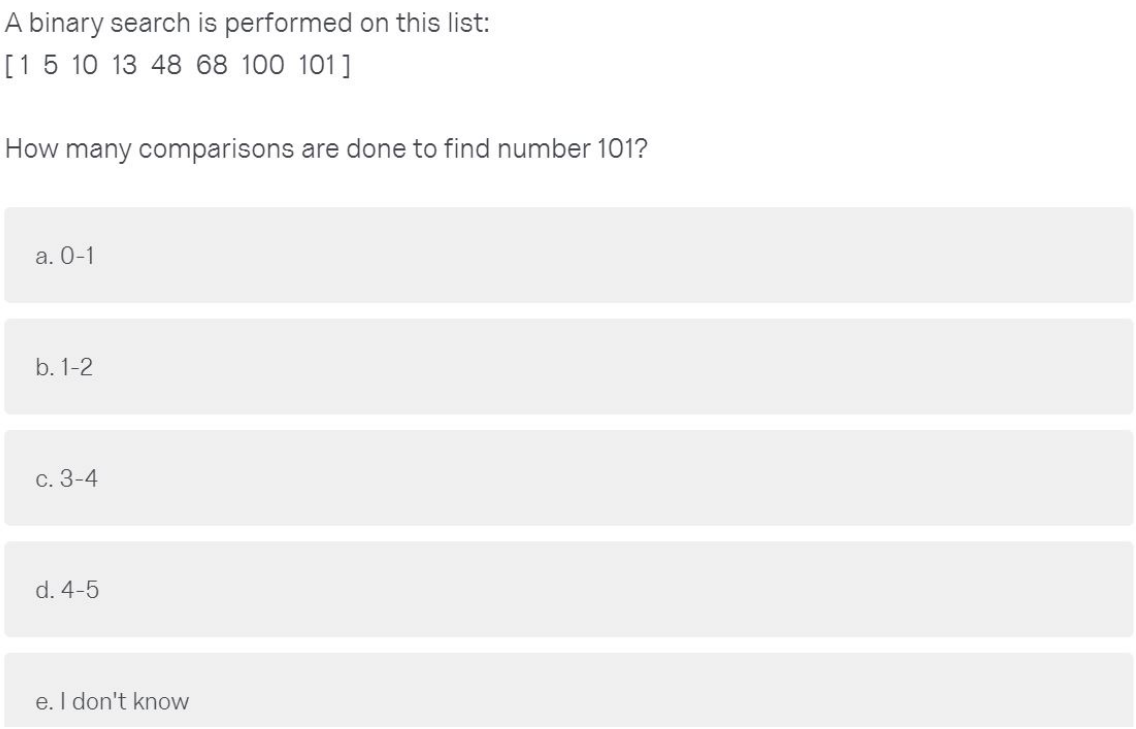


Figure 4.5: Binary Search Problem-Solution Question

Figure 4.6: Insertion Sort Sort Problem-Solution Question

Figure 4.7: Selection Sort Problem-Solution Question

Basic Knowledge Questions

The Basic Knowledge questions were "Yes/No" questions or general questions about search or sorting algorithms. Some of the examples of the basic knowledge questions of linear search, binary search, and merge sort are shown in Figure 4.8, 4.9 and 4.10

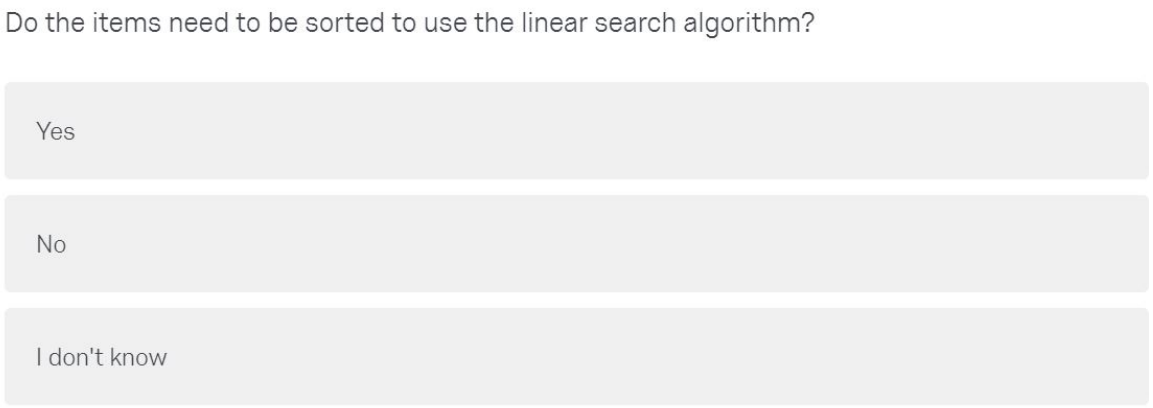


Figure 4.8: Linear search Basic Knowledge Question

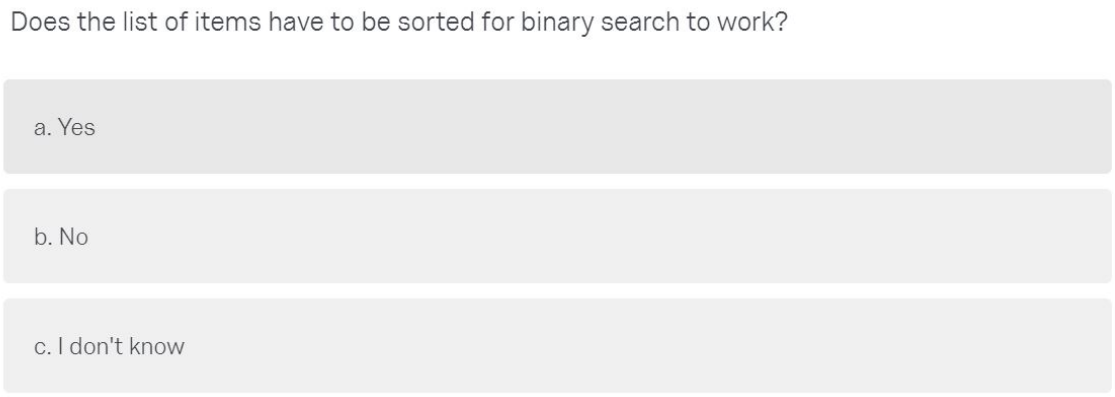


Figure 4.9: Binary search Basic Knowledge Question

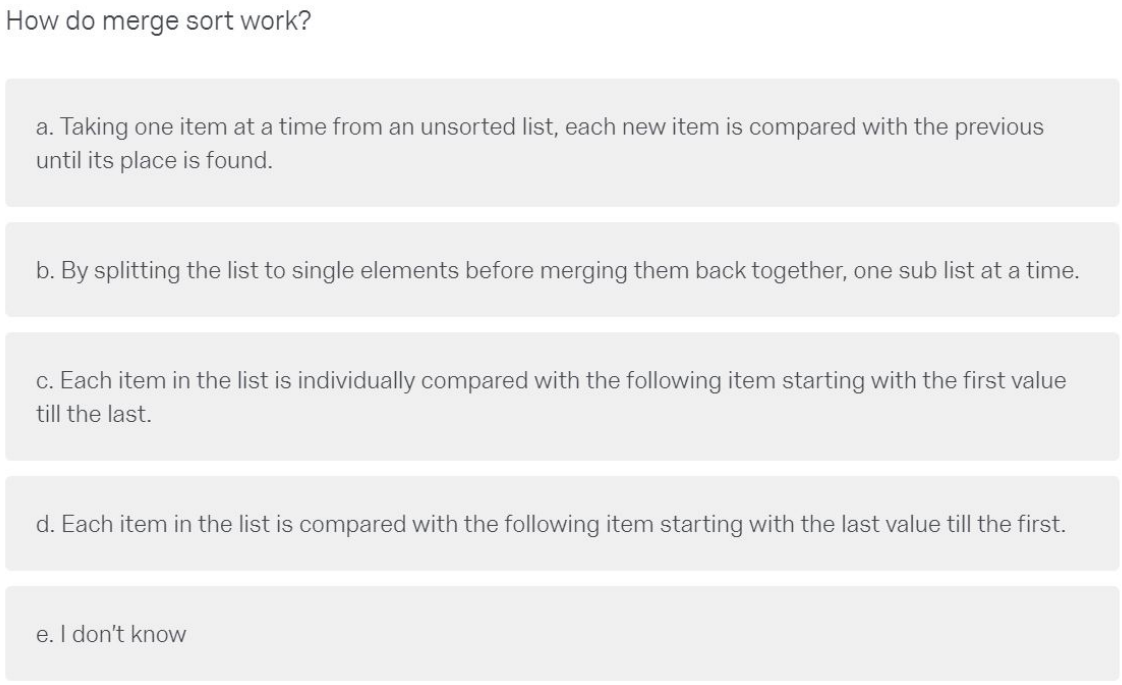


Figure 4.10: Merge Sort Basic Knowledge Question

4.2.6 Scenario Based Questions

There are scenario-based questions at the end of the game that the player has to answer after finishing all the tasks in the game. The questions are titled "Scenarios - Easy" and "Scenarios - Medium." There are four questions in each mode. The "Scenarios - Easy" has questions about linear search and binary search, whereas "Scenarios - Medium" has questions about search and sort. Two example questions from "Scenarios - Easy" is shown in Figure 4.11 and 4.12.

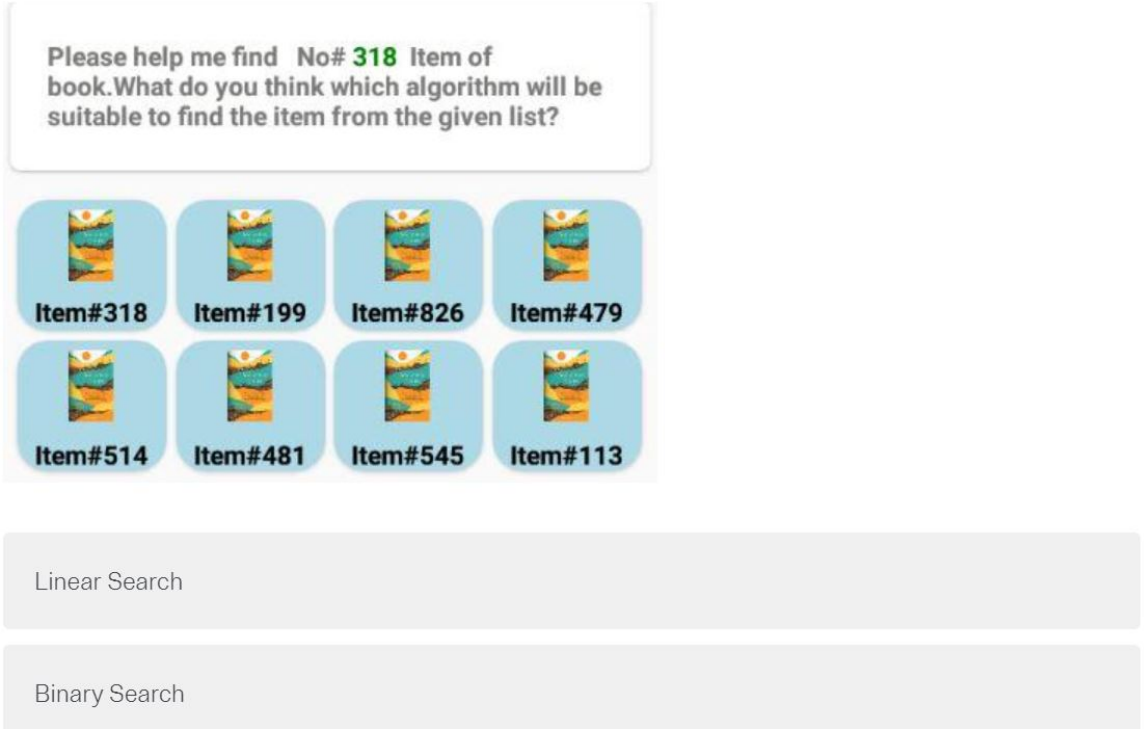


Figure 4.11: Scenario-based Question of Scenarios - Easy

The questions in "Scenarios - Medium" requires knowledge of both search and sort algorithms. The scenario-based questions are asked, and there are two drop-down boxes for each question offering a search option and a sort option. The player has to read the question and choose the right answer. Two out of four questions from "Scenarios - Medium" is shown in Figure 4.13 and 4.14. The drop-down has all six search and sort options along with a none option, as shown in Figure 4.15.

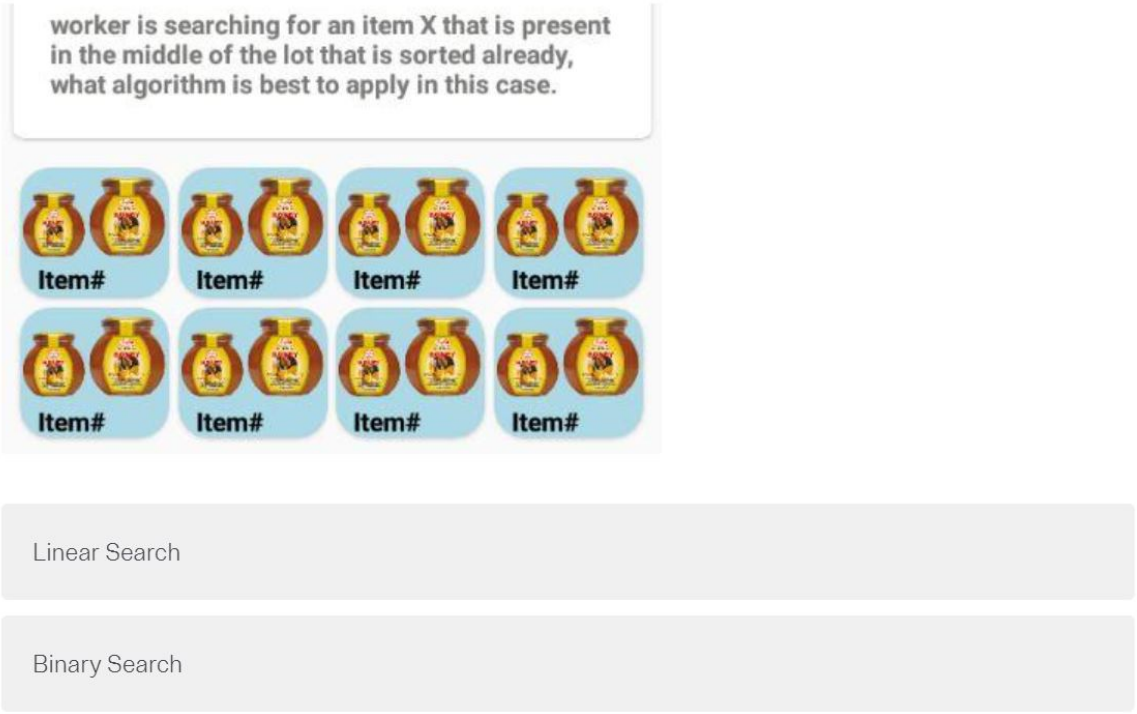


Figure 4.12: Scenario-based Question of Scenarios - Easy

4.2.7 Post Game Questions

The post-game questions were similar to those asked in prior knowledge questions so that we could assess any change in search and sort knowledge. The Figure 4.16 shows on of the binary search post-game questions.

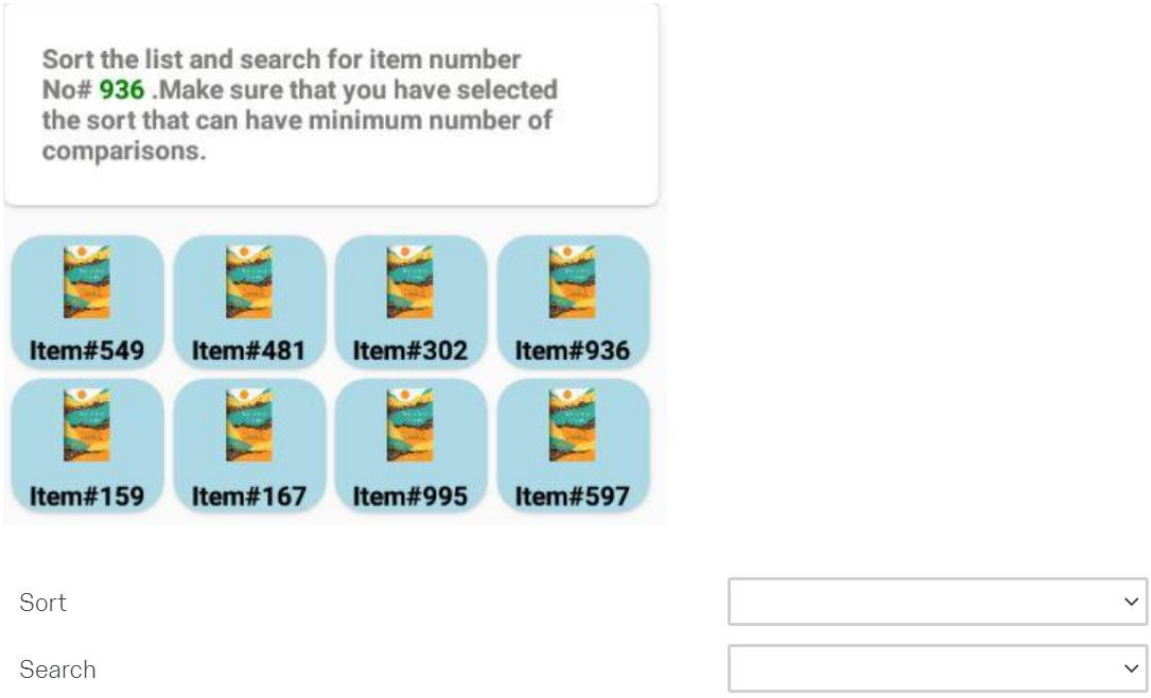


Figure 4.13: Scenario-based Question of Scenarios - Medium

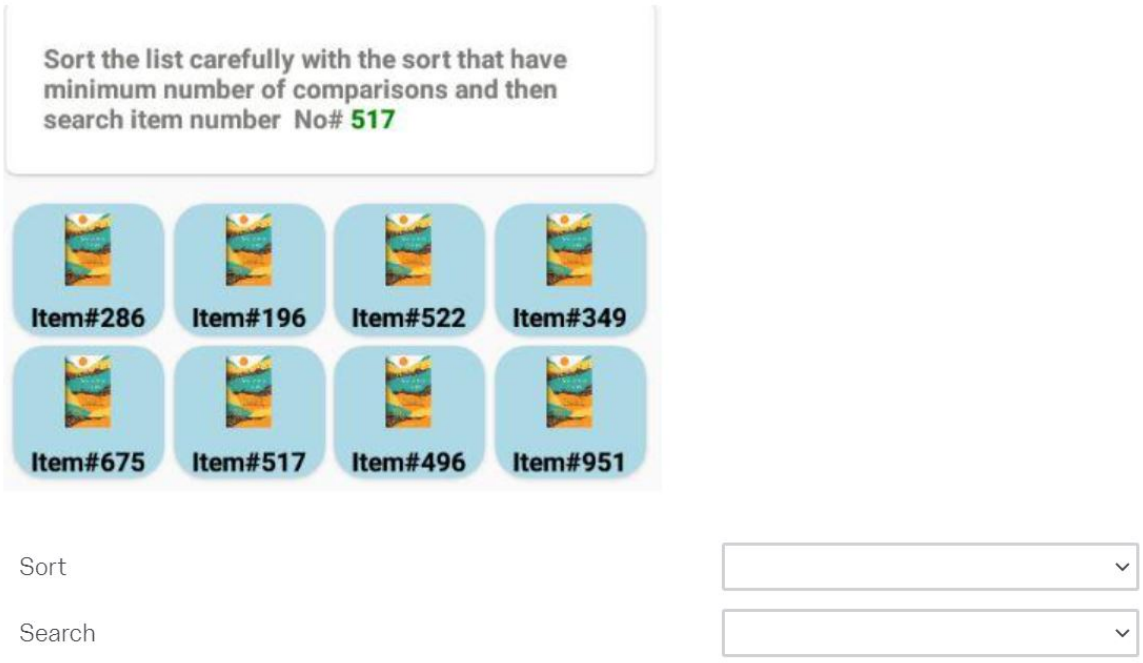


Figure 4.14: Scenario-based Question of Scenarios - Medium

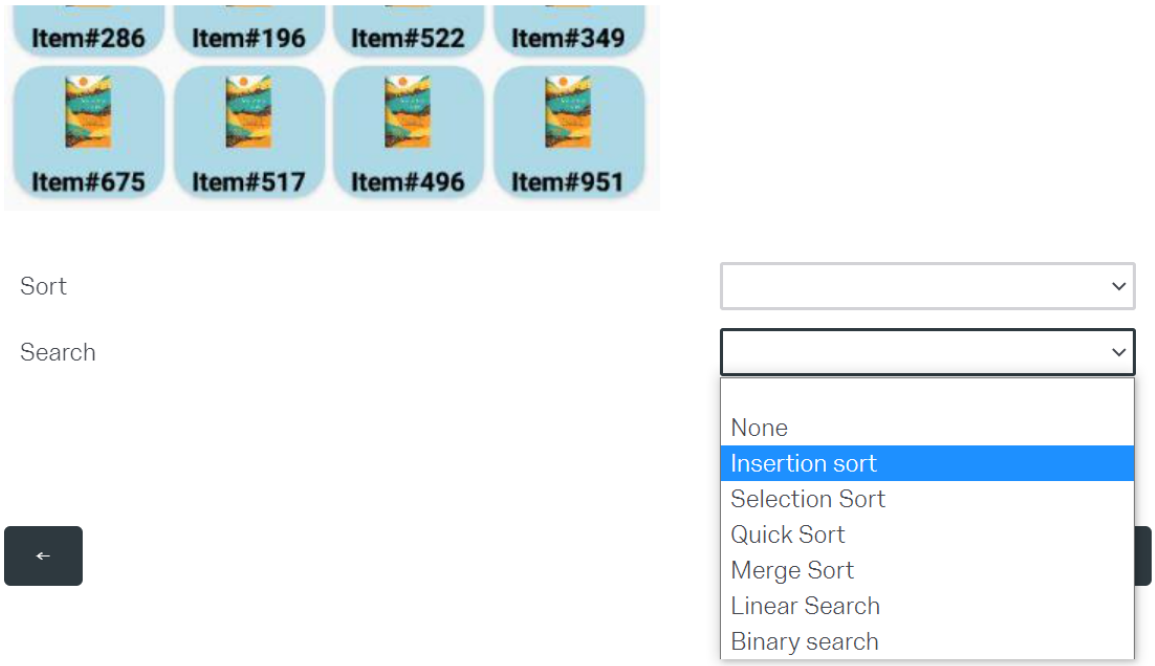


Figure 4.15: Drop-down from Scenarios - Medium Question

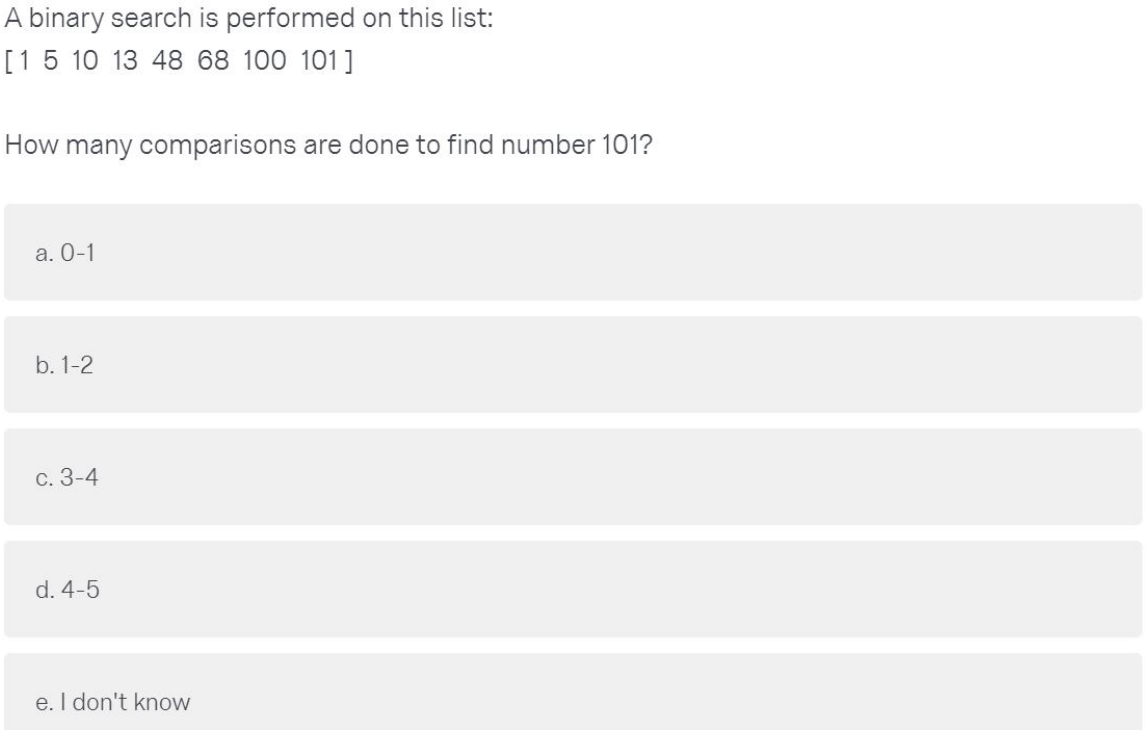


Figure 4.16: Binary Search Post-game Question

4.3 Animation Group

Animations of search and sort algorithms taken from the web were shown to participants to see whether gameplay improves algorithm knowledge or whether viewing animations is just as or more effective at improving algorithm knowledge.

In order to maintain fairness, the players were asked the same four quiz questions after watching each animation, just like players were asked after playing each task in the game. Screenshots of the algorithm animations for linear search [15], binary search[16] ,insertion sort[17] and selection sort[18] animations are shown in Figure 4.17, Figure 4.18, Figure 4.19 and, Figure 4.20.

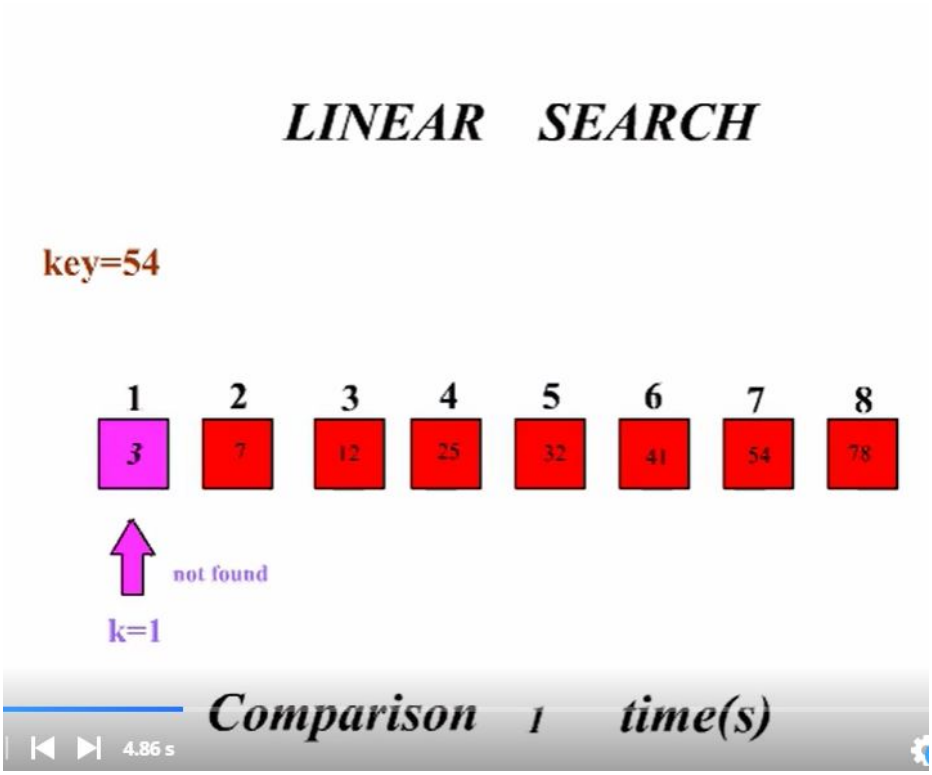


Figure 4.17: Linear Search Animation

[15] https://gfycat.com/vigoroussimplearawana
[16] https://gfycat.com/secondhandunevengelding
[17] https://visualgo.net/en/sorting?slide=9
[18] https://visualgo.net/en/sorting?slide=8

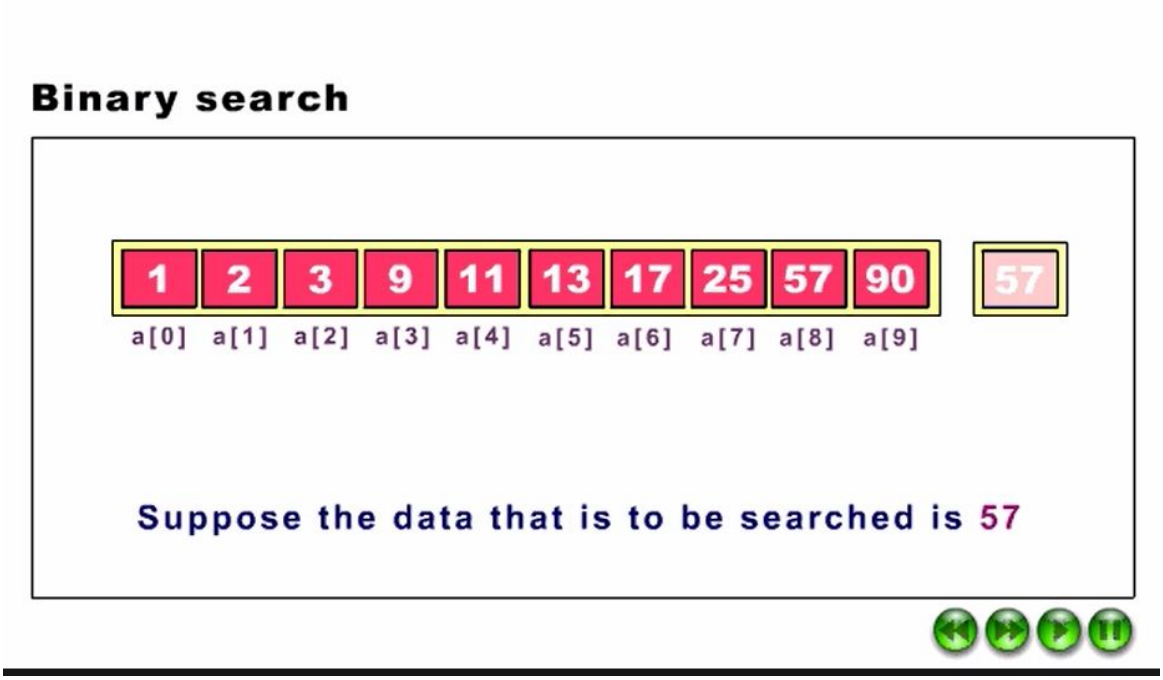


Figure 4.18: Binary Search Animation

Figure 4.19: Insertion Sort Animation

Figure 4.20: Selection Sort Animation

4.4 Summary

The methodology to evaluate the results of this research was a user study. Due to COVID-19 pandemic restrictions, it was conducted online, where participants answered a prior game survey questionnaire, and then they were divided into two groups. One group played the Warehouse Escape game, and the other group watched search and sort animations. After playing the game or watching the animations, participants answered a post-game survey. The learning outcomes of both groups were determined based on changes in correct answers between the two surveys.

Chapter 5

Results And Discussion

The Warehouse Escape user study results are presented in this chapter, along with a detailed discussion of our observations.

5.1 Results

This section presents the results of the user study. Based on the results, the following research questions were answered:

R.Q#1: Do the players' understanding of search and sort algorithms improve after playing the game?

R.Q. #2: What is more effective for learning algorithms, watching animations or Warehouse Escape?

R.Q. #3: Does playing the game help more with answering practical or conceptual questions than watching animations?

A total of seven students participated, with four (04) assigned to the animation group and four (04) assigned to the game group.[19] For demographic information about the participants, we asked about their educational background, and all of them reported being non-CS majors. The questionnaire responses from previous experience and demographic questions are summarized in Appendix D.

[19] A disruption to classes during the Spring 2022 semester likely prevented more students from participating in the research study.

5.1.1 Previous Knowledge

The results of the prior-knowledge questionnaire revealed that none of the participants appeared to have much experience with computer programming, with 25% of respondents reporting no experience with computer programming and 37% saying they are novices who have experience with small programs in a single file. However, 12.5% say they are interested in learning but have no experience. Thus, almost 62.5% of participants had little to no programming experience. The results chart in Figure 5.1 shows the level of computer programming knowledge among participants.

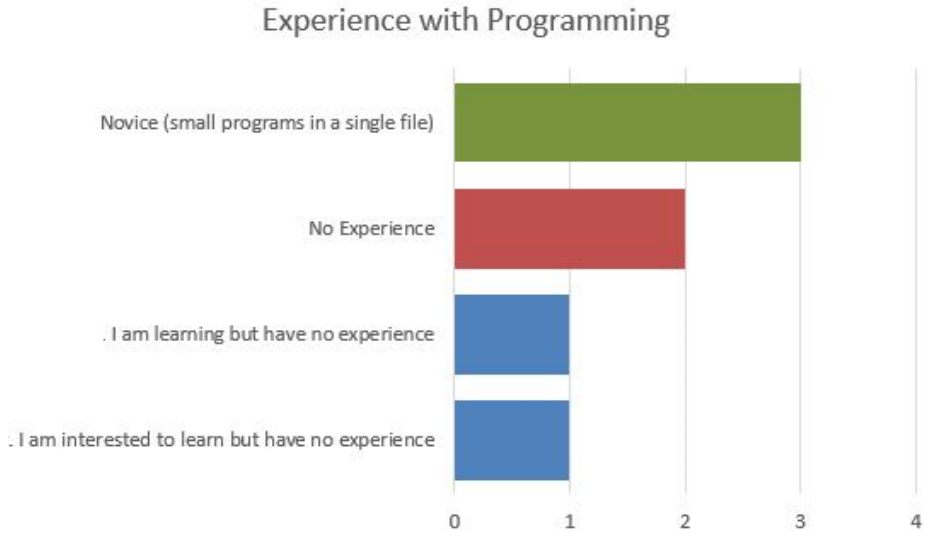


Figure 5.1: Experience with Computer Programming

In response to a question about search algorithms, 37.5% of participants said: "I don't know what an algorithm is," while 37.5% said "I have heard of search algorithms." Only 25% of respondents said they had used search algorithms in a program. Figure 5.2 shows the survey result of the knowledge level of participants about search algorithms.

In response to a question about sort algorithms, 62.5% of participants said "I don't know what an algorithm is," while 37.5% said "I have heard about sort algorithms." Figure 5.3 shows the survey result of the knowledge level of participants about sort algorithms.

The results showed that participants were not well versed in search and sort algorithms.

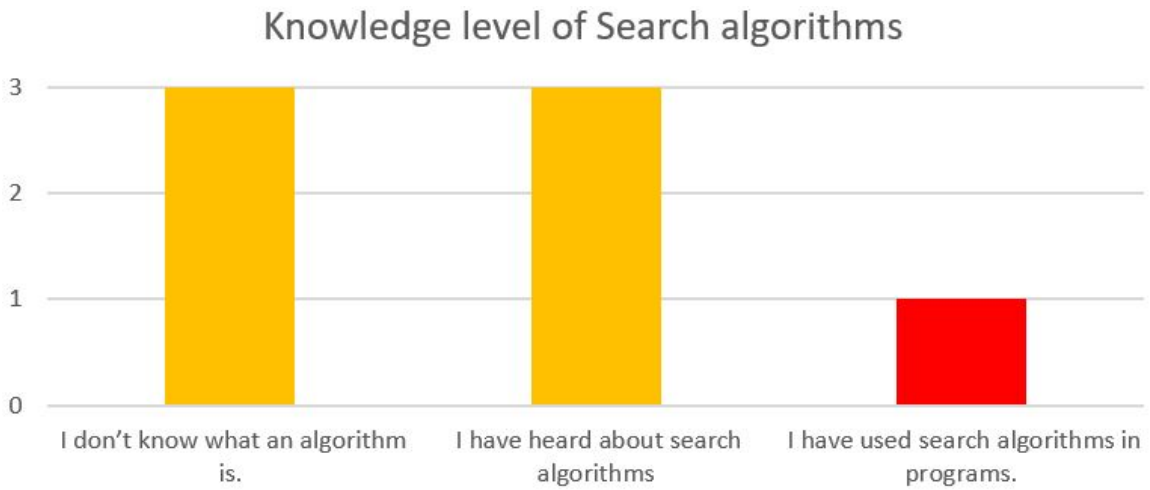


Figure 5.2: Previous Knowledge of Search Algorithms

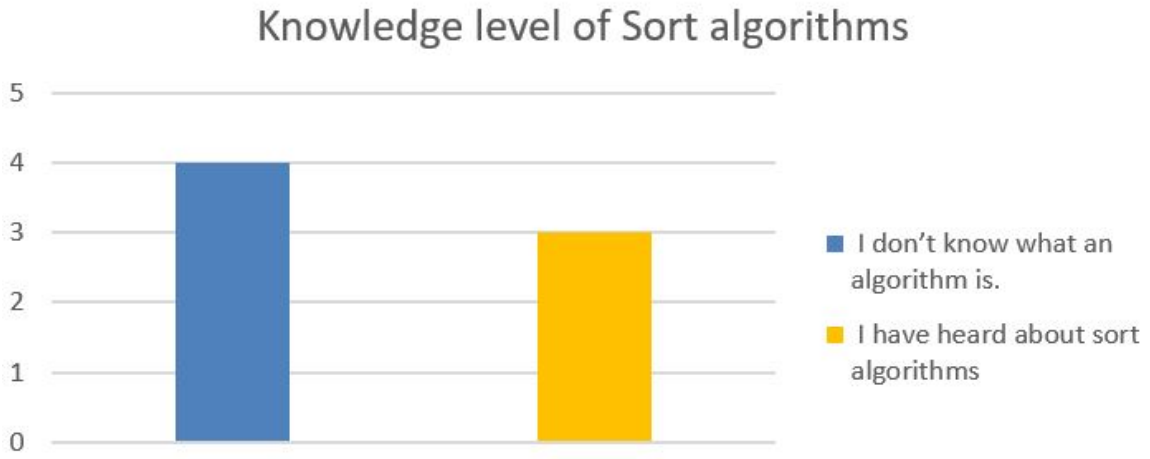


Figure 5.3: Previous Knowledge Sort Algorithms

5.1.2 Determining Learning Outcomes

The participant must show improvement for two out of three post-game questions for an algorithm to determine whether they gained Knowledge. The exception is *Quicksort*, where there were only two questions asked in the post-game/animation survey. The rule for *Quicksort* was that the participants must show improvement

TYPE	Q1	Q2	Q3	LEARNING
GAME	Improved	Improved	No change	YES
GAME	Improved	Improved	Improved	YES
GAME	No change	Improved	No change	NO
GAME	Improved	Improved	No change	YES
ANIMATION	Improved	Improved	No change	YES
ANIMATION	No change	Improved	Unclear	NO
ANIMATION	No change	Improved	No change	NO

Table 5.1: Learning Outcome of Binary search

for both questions. However, for all the other search and sort algorithms two out of three questions needed to show improvement to label the learning outcome as "Yes". As an example, the determination of learning outcomes of binary search is shown in Table 5.1. As we can see, the columns are the three questions, and each participant is listed in the rows. The first column indicates whether it was a game participant or an animation participant. Knowledge gain is divided into four categories: Improved, Unchanged, Unclear, and Same Knowledge.

These categories are already discussed in the above subsection **"Knowledge Change Categories"** We can look at the rows and see if the participant (rows) showed improvement for two out of three questions. If they did, we say that learning occurred (i.e. "yes"). We can determine whether games or animations were more effective for learning algorithms from these results.

The learning outcomes for *Quicksort* are shown in Table 5.2. We can look at the rows and see if the participant (rows) showed improvement for two out of two questions. If they did, we say that learning occurred (i.e. "yes"); otherwise, it is "No." Learning result tables for the other search and sort algorithms are found in Tables 5.3, 5.4, 5.5, 5.6 and 5.7

As can be seen in this Table 5.1, the learning outcome for binary search determined that 75% of the participants for the game and 33% for the animation showed improvement.

TYPE	Q1	Q2	LEARNING
GAME	Improved	No change	NO
GAME	Improved	Improved	YES
GAME	Improved	No change	NO
GAME	Improved	Improved	YES
ANIMATION	Improved	No change	NO
ANIMATION	No change	Improved	NO
ANIMATION	No change	Improved	NO

Table 5.2: Learning Outcome of Quicksort

TYPE	Q1	Q2	Q3	LEARNING
GAME	Same Knowledge	No change	No Change	NO
GAME	No change	No change	Improved	NO
GAME	Improved	No change	Improved	YES
GAME	No Change	Same Knowledge	Improved	YES
ANIMATION	Improved	Improved	Same Knowledge	YES
ANIMATION	Same Knowledge	Same Knowledge	Same Knowledge	NO
ANIMATION	No change	No change	Improved	NO

Table 5.3: Learning Outcome of Linear Search

TYPE	Q1	Q2	Q3	LEARNING
GAME	Improved	Improved	No Change	YES
GAME	Improved	Improved	Improved	YES
GAME	No change	Improved	No Change	NO
GAME	Improved	Improved	No Change	YES
ANIMATION	Improved	Improved	No Change	YES
ANIMATION	No change	Improved	Unclear	NO
ANIMATION	No change	Improved	No change	NO

Table 5.4: Learning Outcome of Binary Search

TYPE	Q1	Q2	Q3	LEARNING
GAME	No change	Same Knowledge	No Change	NO
GAME	No change	Improved	No change	NO
GAME	Improved	No change	No change	NO
GAME	Improved	Improved	Improved	YES
ANIMATION	No change	Improved	Improved	YES
ANIMATION	No change	Improved	Improved	YES
ANIMATION	Improved	No change	No change	NO

Table 5.5: Learning Outcome of Insertion Sort

TYPE	Q1	Q2	Q3	LEARNING
GAME	No change	No change	Improved	NO
GAME	Improved	Improved	No change	YES
GAME	Improved	Improved	No change	YES
GAME	No change	Improved	No change	NO
ANIMATION	Improved	Improved	Improved	YES
ANIMATION	No change	Improved	No change	NO
ANIMATION	No change	No change	Improved	NO

Table 5.6: Learning Outcome of Selection Sort

TYPE	Q1	Q2	Q3	LEARNING
GAME	Improved	Improved	Improved	YES
GAME	Improved	No change	Improved	YES
GAME	Improved	Improved	Improved	YES
GAME	No change	Improved	Improved	YES
ANIMATION	Improved	Improved	Improved	YES
ANIMATION	No Change	Improved	Improved	YES
ANIMATION	No change	No change	No Change	NO

Table 5.7: Learning Outcome of Merge Sort

5.2 Does playing the Warehouse Escape improve algorithm knowledge?

The research question#1 is further divided into six sub-questions to evaluate the learning outcomes more accurately. The first research question of this work is:

R.Q#1: Do the players' understanding of search and sort algorithms improve after playing the game?

This question was further divided into the following questions:

1. Did the participant's Knowledge of *Linear Search* improve by playing the game?

2. Did the participant's Knowledge of *Binary Search* improve by playing the game?

3. Did the participant's Knowledge of *Insertion Sort* improve by playing the game?

4. Did the participant's Knowledge of *Selection Sort* improve by playing the game?

5. Did the participant's Knowledge of *Quicksort* improve by playing the game?

6. Did the participant's Knowledge of *Merge Sort* improve by playing the game?

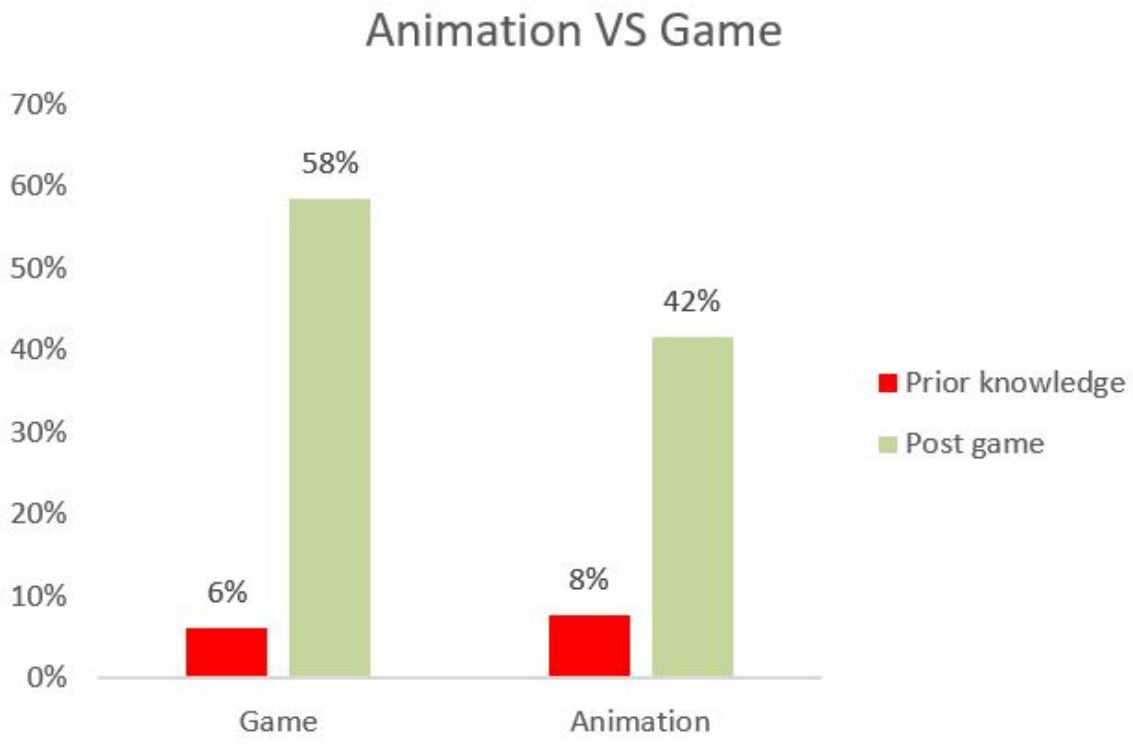


Figure 5.4: Correct Answers of Game and Animation from the Survey

In the prior knowledge survey, only two questions were correctly answered by two game participants, both for the linear search algorithm. All the other answers for the rest of the search and sort algorithms were incorrect. However, the Knowledge improved significantly after playing the game: Knowledge about merge sort improved to 100%, quicksort and binary search improved to 75%, linear search and selection sort increased to 50%, and insertion sort increased 25%. Insertion sort was much lower than the other algorithms, but overall it was increased to 25%, bar graph in Figure 5.5 shows the players' improvements for all the algorithms.

After evaluating the data gathered from the participants (see Appendix D), we concluded that after playing the "Warehouse Escape", participants improved their understanding of search and sort algorithms, as shown in Figure 5.4. Participants' Knowledge was considered 6% before playing the game and improved to 58% after

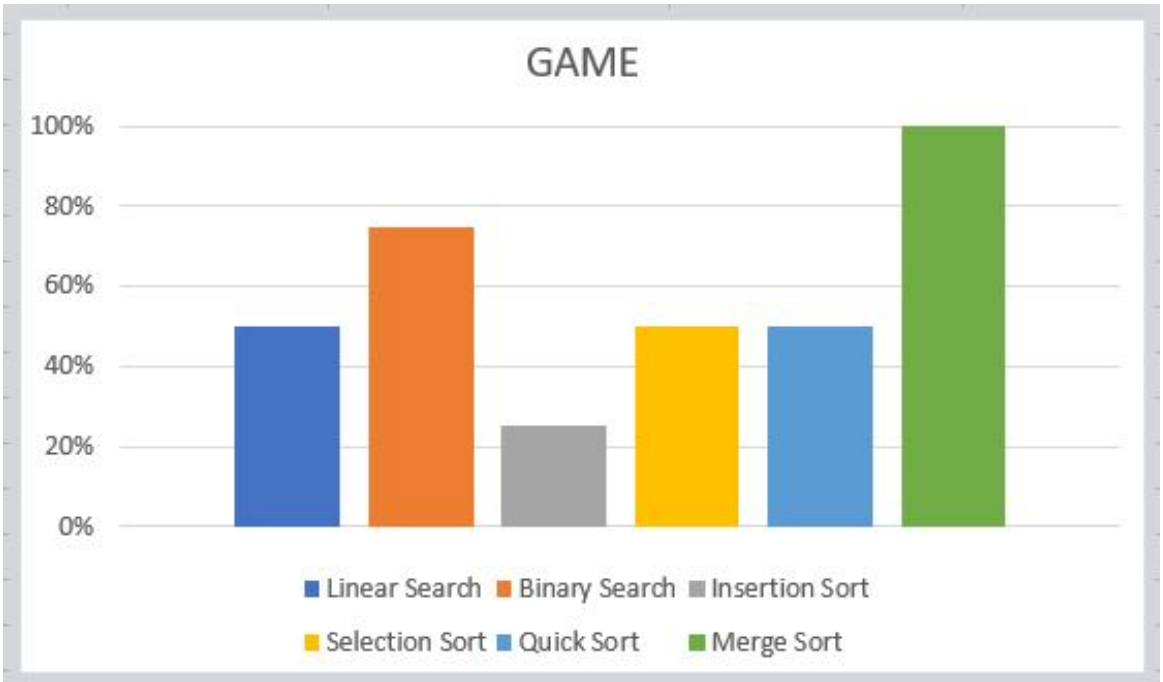


Figure 5.5: Search And Sort Knowledge improved by Playing Game

completing the game.

Game players only played the game once whereas participants from the animation group watched animations more than once. The knowledge improvement of game players could have been more if they would have played the game more than once.

The results were determined by individually calculating the percentage of correct answers for the game and the animation groups. In addition, the results of each search and the sort question have been assessed separately to answer the sub-questions of research question #1. Detailed graphs for each question are given in Appendix E.

Observations: There was an improvement for all of the algorithms, with merge sort being the most improved and insertion sort being the least improved. We believe that there are three reasons for insertion sort being the least improved.

First, when students played the search algorithm tasks, they found them easier than the sorting algorithm tasks. During the observation and interview sessions, participants stated that they were a bit familiar with the search algorithm, specifically linear search, as shown by the Figure 5.3. They found the sequential nature of linear

search easy to understand. Insertion sort was the first sort algorithm that players encountered after completing the binary search task, and students had a difficult time transitioning right away from search to sort. One of the participants said, "Sort algorithms are tougher than search. Playing them more than once would have been more beneficial".

The second reason is that one of the descriptive questions for insertion sort had a similar answer option in which students got confused and picked the wrong answer. The question was "why is insertion sort important?" and one of the answers was "Because it makes inserting new items easier," as shown in Figure 5.6. Most participants selected this option, which was incorrect.

Moreover, participants had difficulty sorting the list of practical questions. An example of a practical question is shown in Figure 5.7. A participant explained that he has numerical dyslexia, so he has trouble answering questions involving numbers. Another participant said that his math is weak, so he does not feel comfortable with number-related questions.

Lastly, and perhaps most importantly, all players played each task just once. It can be difficult to gain an exact understanding of the sort by playing just once. On the other hand, the participants in the animation group were able to go back to the animation when answering questions and so had aid in answering the insertion sort questions that those in the game group did not.

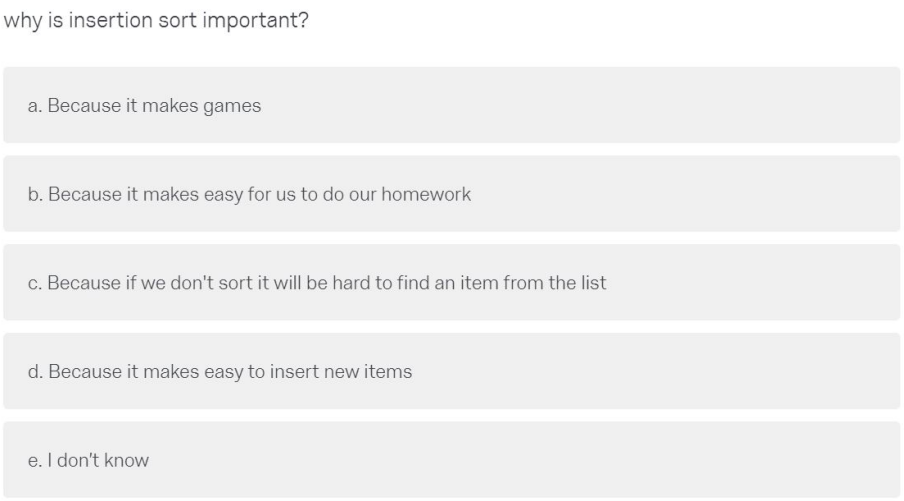


Figure 5.6: Insertion Sort Prior Knowledge Question (Descriptive)

Outcome: Based on our analysis, for research question #1 *"Do players' understanding of search and sort algorithms improve after playing the game?"*. We found the answer to be yes. Before playing the game, the pre-treatment surveys showed that participants had almost no understanding of the algorithms overall. The result indicates that, in general, their Knowledge improved from 6 to 58 percent, which is a significant improvement. We believe that their algorithm knowledge could have improved even more if they had played the game more than once.

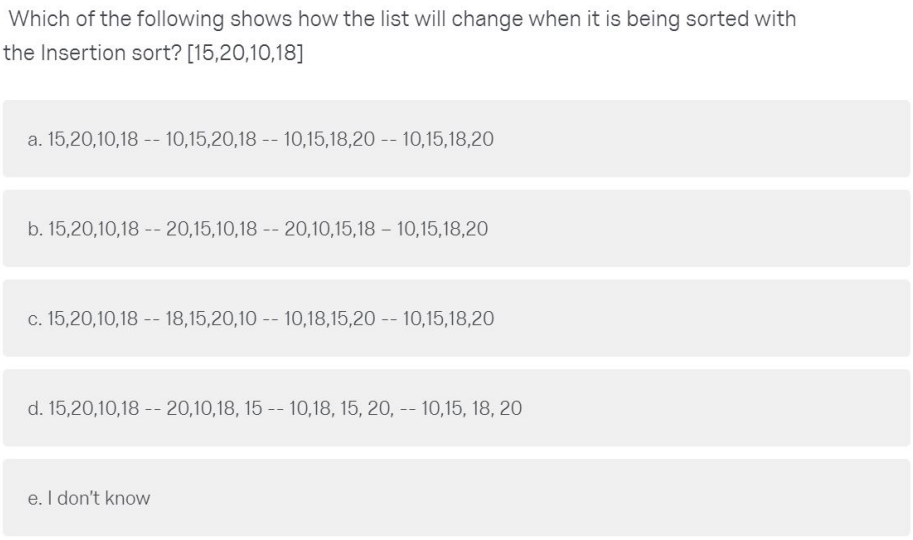


Figure 5.7: Insertion Sort Prior Knowledge Question (Practical)

5.3 Which is better at improving algorithm knowledge?

To answer this question, we formulated the rule presented in Table 4.1, which measured the learning outcomes for each question and subsequently accumulated the results for each search and sort algorithm.

R.Q. #2: What is more effective for learning algorithms, watching animations or playing Warehouse Escape?

Research question#2, was further divided into six sub-questions and then compared with the sub-questions of R.Q. #1 in order to determine which is the more effective way to teach algorithms. These sub-questions are:

1. Did the participant's Knowledge of *Linear Search* improve by watching animations?

2. Did the participant's Knowledge of *Binary Search* improve by watching animations?

3. Did the participant's Knowledge of *Insertion Sort* improve by watching animations?

4. Did the participant's Knowledge of *Selection Sort* improve by watching animations?

TYPE	GAME	ANIMATION
Linear Search	50%	33%
Binary Search	75%	33%
Insertion Sort	25%	66%
Selection Sort	50%	33%
Quicksort	50%	0%
Merge Sort	100%	66%

Table 5.8: Comparison of Game and Animation Learning Outcomes

5. Did the participant's Knowledge of *Quicksort* improve by watching animations?

6. Did the participant's Knowledge of *Merge Sort* improve by watching animations?

Figure 5.8 shows that the player's Knowledge improved for all the algorithms after watching animation with the exception of *Quicksort*. Table 5.8 shows the comparison of the learning outcomes for the game and animation groups. Based on this table, we can say that the game shows more learning improvement for all algorithms, except for the insertion sort.

Observations: The participants who watched the animation of insertion sort, watched it more than once. Watching animations more than once may have helped them to better solidify their Knowledge of insertion sort over those who only played the insertion sort level in the game once.

Outcome

When comparing the results of those participants who played the game or those who watched the animations, we can see that regardless of if they played the game or watched the animations, their Knowledge improved in animation. In contrast, *Quicksort* showed no improvement. This is because the post-game/animation survey questions asked for quicksort were two, unlike other algorithms. To be marked as Knowledge improved, participants must answer both questions correctly.

However, the participants who played the game showed a more significant knowl-

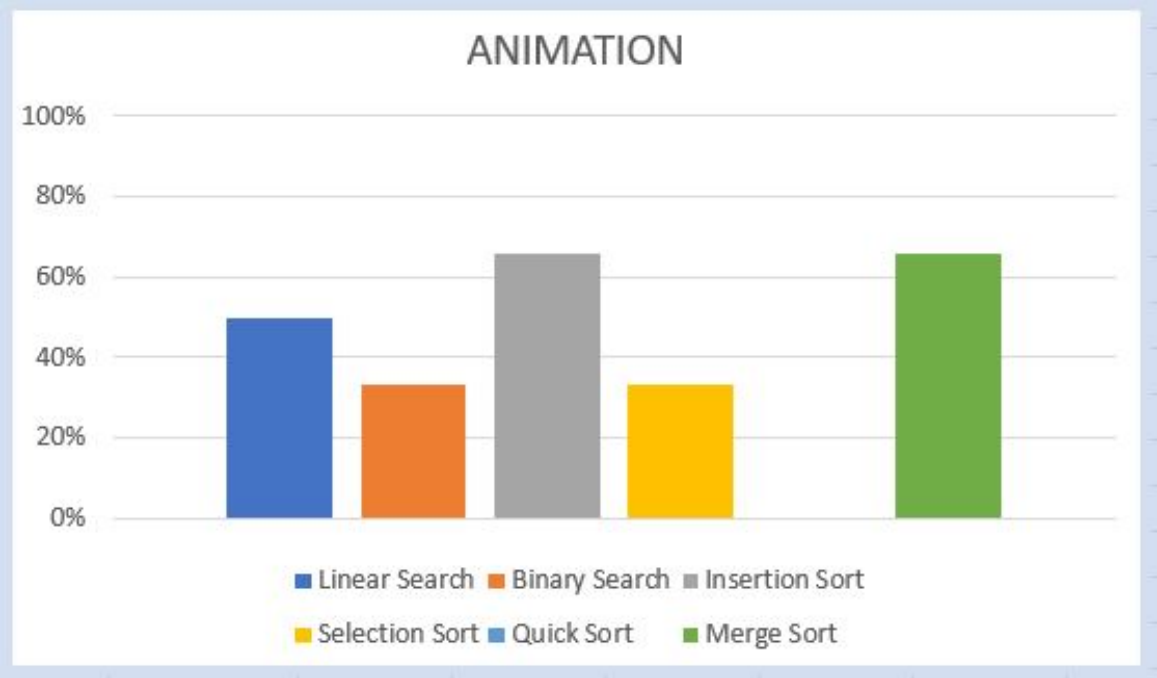


Figure 5.8: Search And Sort Knowledge improved by Watching Animation

edge gain than those who watched the animations. Students who watched animations improved their Knowledge by 42% This answers R.Q. #2; playing games was more effective than watching animations.

The answer to R.Q. #2 is that although all participants showed knowledge improvements regardless of interventions, more learning gains were observed for those who played the game. This was most evident with the merge sort algorithm.

Before playing the game or watching animations, participants demonstrated a general lack of understanding of the algorithms. The results indicate that those who played the game saw their Knowledge improved from 6 to 58 percent after playing the game and 8 to 42 percent after watching the animation. We believe that knowledge gains may have been more substantial if they had played the game more than once. Appendix E provides the learning improvement results for all the questions of the search and sort algorithms.

TYPE	LS	BS	IS	SS	QS	MS
QT	Q1 Q2 Q3	Q1 Q2 Q3	Q1 Q2 Q3	Q1 Q2 Q3	Q1 Q2	Q1 Q2 Q3
TYPE	C P C	C C P	C P P	C C P	C P	C P C
GAME	1 0 3	3 4 1	2 2 1	2 3 1	4 2	3 3 4
ANIMATION	0 1 2	1 3 0	1 2 2	1 2 2	1 2	1 2 2

Table 5.9: Correct Answers Comparison of conceptual vs. practical

5.4 Does playing the game help more with conceptual or practical questions?

As mentioned in Chapter 4, there were three types of questions in the user study: *Descriptive, Basic Knowledge* and *Problem-solution*. We grouped these three types of questions into two categories: *Conceptual Knowledge* and *Practical Knowledge*.

The *Conceptual Knowledge* group contained the descriptive and basic knowledge questions. In contrast, the *Practical Knowledge* group contained the problem-solving questions, such as giving unsorted lists to sort using a specific algorithm. Appendix F contains examples of conceptual and practical questions from the survey.

Table 5.9 shows a detailed analysis of the types of questions. The first row shows the question's number, and the second row shows the type of question asked for each algorithm, (i.e. conceptual (C) or practical (P)). And the last two rows show the number of participants who correctly answered the question in the post-game and animation survey. The other columns identify the specific algorithms: Linear Search (LS), Binary Search (BS), Insertion Sort (IS), Selection Sort (SS), Quicksort (QS), and Merge sort (MS).

Observations: The observation and interview sessions showed those in the game group did not play the game more than once, which is likely why they found the practical questions were more complex. Some participants mentioned that they could have done better if they had played the game or watched the animation more than once. Table 5.9 shows the comparison of correct answers for the *Practical* and *Conceptual* questions. It can be seen that the game group correctly answered the conceptual

questions more often than the animation group.

The *Practical* questions mostly involved numerical values and sorting lists. One of the responses we got from a participant was "I have numerical dyslexia, which is why I cannot do well on questions involving numbers." All the comments and feedback from the participants are listed in Appendix D. Based on our limited data, we found that the game helped the most with correctly answering the conceptual questions and watching the animations helped most with practical questions. However, this may be a result of those from the animation group being able to re-watch the animations when answering the questions.

Outcome: We found that participants did better overall on conceptual questions than on practical questions. As shown in Table 5.9 that the animation group mostly answered more practical questions correctly than the game group, with the *Merge Sort* being the exception. In contrast, the game group generally correctly answered more conceptual questions correct than the animation group. Therefore the answer to R.Q. #3 **"Does playing the game help more with answering practical or conceptual questions than watching animations?"** is that playing game helped more with answering conceptual questions.

5.5 Threats to validity

This section discusses threats to the external, internal and construct validity of our results.

5.5.1 External Validity

We could not gather as much data as we planned due to a job action disruption at the university. As a result, the sample size was small, which means the result might not generalized.

5.5.2 Internal Validity

It was observed that the animation group could re-watch the animations when answering questions. In fact, animation group participants watched almost all the search and sort algorithms more than once, and the game group could not do that while doing the quiz. This means that the difference in results between the game and animation groups may be larger in practice if the game group had been able to review their tasks when taking the quiz.

During the analysis of the results, the following item was noticed. In one of the linear search questions (Figure 5.9) where the list of numbers was [3, 100, 1, 14, 8, 7, 10, 28], participants had to count linearly until the item#14, which makes the count 4. During the observation session, it was observed that several students overlooked "1". This may have been due to poor placement within the list. Most students were found to have answered this question as "3" which was incorrect, instead of "4".

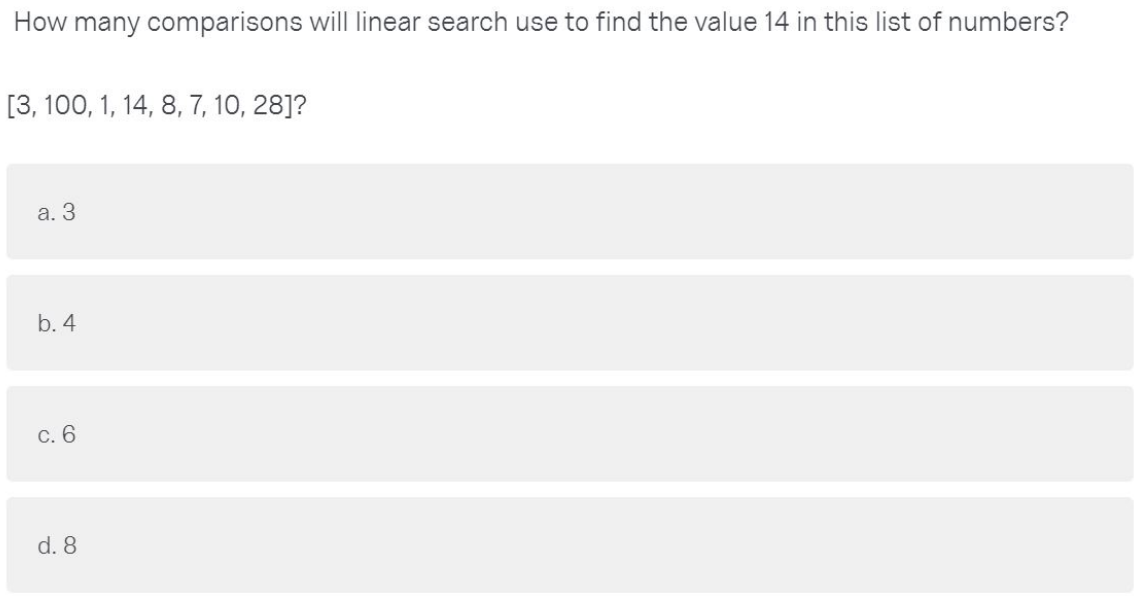


Figure 5.9: Linear Search Question(Internal Validity threat)

5.5.3 Construct Validity

To evaluate the learning outcomes, we designed a rule where two out of three post-game questions must show improvement in order to categorize learning improvement for that algorithm. All of the search and sort algorithms had three questions with the exception of *Quicksort* which had two questions. This means that the results for *Quicksort* may be an overestimation of the learning improvement.

5.6 Summary

According to our user study, after playing the Warehouse Escape game, participants' knowledge about search and sorting algorithms improved, answering R.Q. #1. The survey results of the game and animation group were compared, and we found that the knowledge improvements after playing the game were more significant than the knowledge improvements after watching the animations. More specifically, we found that algorithm correct response rates improved after playing the game from 6% to 58%, whereas for the animation watching group, the correct response rate grew from 8% to 42%. The result was evaluated by computing the overall percentage of correct answers to the game and animation from the survey. Therefore, we conclude that playing the game to improve algorithm knowledge is more effective.

Chapter 6

Conclusion

In this thesis, we aimed to identify the impact of game-based learning on the teaching of search and sort algorithms. To do this, we created the game Warehouse Escape.

Warehouse Escape is a mobile game for Android and iOS. It teaches the basic concepts of search and sort algorithms. The game is designed in a way, to give the players a theoretical explanation as well as practical knowledge of the algorithms. It consists of two modes; "Learning," where players read about the algorithms and the "Play Game" mode, where players complete given tasks and do quizzes. The game teaches two search and four sort algorithms. In each level, the player is asked to complete a task and then complete a quiz related to their completed task. The game ends with the scenario-based questions having "Scenario - Easy" and "Scenario - Medium" modes.

The three research questions for this work were:

1. Do the players' understanding of search and sort algorithms improve after playing the game?

2. What is more effective for learning algorithms, watching animations or Warehouse Escape?

3. Does playing the game help more with answering practical or conceptual questions than watching animations?

To answer these research questions, we conducted a user study. The participants were asked to fill out a prior knowledge survey. Then the participants were divided

into two groups. One group played "Warehouse Escape," and the other group watched animations. After that, they completed a post-game/post animation survey. Based on the responses participants provided, the knowledge change of each participant was calculated. Lastly, the results were compared to see which group performed better and how their knowledge improved.

Based on the analysis of the user study results, it is concluded that game-based learning proved to be an effective way of teaching search and sort algorithms. Participants improved their knowledge of algorithms after playing the game. The user study results from both animation and game groups were compared to determine whether playing a game improves knowledge or watching animation is more beneficial for understanding algorithms. Results showed that the knowledge of the search and sort algorithm improved by 58% among those who played the "Warehouse Escape" while the knowledge improved by 42% among those who watched animations. We found that participants did better overall on conceptual questions than on practical questions.

The contributions to this work are:

- A mobile game for teaching basic sort and search algorithms.
- A user study of the game comparing the use of the game to that of watching algorithm animations.
- Results indicating that playing the game results in better learning outcomes than watching algorithm animations.

6.1 Future Work

According to the feedback provided by the participants, they enjoyed playing the game as a way to learn about search and sort algorithms. They didn't find any difficulty in game design. However, the sample size was small for the user study due

to external circumstances beyond our control. In future, we will plan to re-conduct an improved user study.

Bibliography

[1] John Anvik, Vincent Cote, and Jace Riehl. Program wars. `https://program-wars-dev.firebaseapp.com/`, 2013. [Online; Accessed 15-February-2021].

[2] John Anvik, Vincent Cote, and Jace Riehl. Program wars: a card game for learning programming and cybersecurity concepts. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 393–399, 2019.

[3] Bruce leverett. Board games— Wikipedia, the free encyclopedia, 2020. [Online; accessed 04-April-2021].

[4] Michael D Byrne, Richard Catrambone, and John T Stasko. Evaluating animations as student aids in learning computer algorithms. *Computers & education*, 33(4):253–278, 1999.

[5] Code master. `https://www.thinkfun.com/products/code-master/`. [Online; Accessed 15-February-2021].

[6] Computer games and simulations for education and exploration. `https://ocw.mit.edu/courses/urban-studies-and-planning/11-127j-computer-games-and-simulations-for-education-and-exploration-spring-2` Accessed: 2022-03-03.

[7] Through In-Car Gaming. Chi 2011 workshop gamification: Using game design elements in non-game contexts. *Computers & education*, 1999.

[8] Game development. `https://www.cs.usc.edu/academic-programs/masters/game-development/`. Accessed: 2022-03-03.

[9] Scott Grissom, Myles F McNally, and Tom Naps. Algorithm visualization in cs education: comparing levels of student engagement. In *Proceedings of the 2003 ACM symposium on Software visualization*, pages 87–94, 2003.

[10] Foteini Grivokostopoulou, Isidoros Perikos, and Ioannis Hatzilygeroudis. An educational game for teaching search algorithms. In *CSEDU (2)*, pages 129–136, 2016.

[11] Ioannis Hatzilygeroudis, Foteini Grivokostopoulou, and Isidoros Perikos. Using game-based learning in teaching cs algorithms. In *Proceedings of IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE) 2012*, pages H2C–9. IEEE, 2012.

[12] Online gamification. https://platform.onlinelearning.upenn.edu/offering/gamification-a0Q2E00000JmMGSUA3. Accessed: 2022-03-03.

[13] Apichat Phongsasithorn, Artorn Nokkaew, and Parames Laosinchai. Sorted: An educational digital game for learning sorting algorithms. In *Sorted: An Educational Digital Game for Learning Sorting Algorithms*, 04 2019.

[14] Potato pirates. https://potatopirates.game/products/potato-pirates-coding-card-game. [Online; Accessed 15-February-2021].

[15] Robot turtles. https://www.thinkfun.com/products/robot-turtles/, 2013. [Online; Accessed 15-February-2021].

[16] Mauricio R A Souza, Lucas Veado, Renata Teles Moreira, Eduardo Figueiredo, and Heitor Costa. Games for learning: Bridging game-related education methods to software enginering knowledge areas. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*, ICSE-SEET '17, page 170–179. IEEE Press, 2017.

[17] Universities offer masters in gamification. https://petejenkins.com/list-of-masters-in-gamification-courses/. Accessed: 2022-04-02.

[18] Vladimir Uskov and Bhuvana Sekar. Gamification of software engineering curriculum. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–8. IEEE, 2014.

[19] Ladislav Végh et al. Using interactive game-based animations for teaching and learning sorting algorithms. In *Conference proceedings of eLearning and Software for Education*, pages 565–570. " Carol I" National Defence University Publishing House, 2016.

[20] Jeffrey R Young. Homework? what homework. *The Chronicle of Higher Education*, 49(15):A35–A37, 2002.

Appendix A

User Study Consent Form

Please read the following information carefully before beginning the survey.

Warehouse Escape: An Interactive Game to Teach Search and Sort Algorithms

What is this study about?
You have been invited to participate in a research study at the University of Lethbridge. This study is part of the master's thesis, that involves the creation of an educational mobile game. The game is designed to help the player learn and understand key concepts relating to search and sort algorithms.
Through your participation in this research study, we hope to understand better the effect of a mobile-based game on learning algorithms concepts. No programming or Computer Science background is required to participate in the study. In the study, we are researching whether a participant's knowledge of algorithm concepts increases after playing the game and finding ways to improve the educational nature of the game.

What are the benefits of participating?
You will receive participation credit(s) in CPSC 1000 for participating in this study. Also, it is hoped that you gain a better understanding of the basic concepts of search and sort algorithms. However, participating in this research is not expected to have any impact on tests or grades, other than for the credit you earn.

- If you are not interested in participating in the study and choose to do other activities, you will be given marks for completing the other activities.
- If you complete prior knowledge questions you will receive 1 participation mark and may then complete two activities to receive full participation marks.
- If you complete the study portion, you will receive an additional participation mark, and may then complete one activity to receive full participation marks.
- If you complete the post-game survey, you will receive 1 participation mark. You will not need to complete any activities to receive full participation marks.
- If you complete all three parts of the study and later wish to withdraw from the study, you can email the PI (email address given below). You have two weeks after participation to withdraw from the study. Withdrawal will not affect the participation credit in the course, and you will still receive the full participation marks.

What are the risks and benefits of participating?
There are no anticipated risks from participating in this study. As the study will be conducted using Zoom, there is no additional risk of exposure to COVID-19 or its variants

If you participate in this study, you can receive up to three (3) participation points as outlined above.

The instructor will not have access to information about who completed the study (either completely or partially) and who completed one or three of the alternative task(s). From the instructor's point of view all of the students may have participated, some may have participated, or none may have participated. The instructor will not be able to tell as the course gradebook on Moodle will have a single grade field that will be filled only by the marker.

What is expected of you?
Your participation in the research study is expected to take 60 minutes. Participating in this study involves three parts:

1. Completing an initial survey containing demographic questions. Also, you will be asked some questions to judge any prior knowledge of searching and sorting algorithms.

2. You will randomly be placed into one of two groups where you will either play the game or watch some animations. There are six tasks: two search and four sorting tasks. 3. Complete a quiz related to the task you just performed. There is no penalty for getting any wrong answers and are intended to help reinforce the knowledge that you gain from completing a task.4. After you have completed playing the game or watching animation, you will answer a set of scenario-based questions to assess if your knowledge has changed. You will also be asked about your experience playing the game.

What are the anticipated uses of the data collected?
The responses of the surveys will be aggregated and presented in one or more academic reports and presentations, including a master's thesis. At no time will personally identifying information be used in the reports or presentations.

How will your confidentiality and anonymity be protected?
Participation is voluntary. The instructor will not have access to information about who completed the study (either completely or partially) and who completed one to three of the alternative task(s).
However, as with any online survey, neither anonymity nor confidentiality can be guaranteed entirely. The survey is being conducted using Qualtrics, and
their privacy policy can be accessed at https://www.qualtrics.com/privacy- statement/.

You will be given a unique study_id You will enter your study id as part of the pre-and post-game questionnaires so that your responses can be linked. after the study for analysis. This is because the two questionnaires are separate in Qualtrics.

How can a participant withdraw?
Your participation is completely voluntary. You can withdraw from the study at any time during or after the session if you decide you no longer want to participate by informing the researcher. If you choose to withdraw, any responses you provided will not be included in the study. If you withdraw from the study, any data collected will be thrown out and not used in the study. However, once the data has been aggregated, we may not be able to remove the data.

1. If you complete prior knowledge questions you will receive 1 participation mark and may then complete two activities to receive full participation marks.
2. If you complete the study portion, you will receive an additional participation mark, and may then complete one activity to receive full participation marks.
3. If you complete the post-game survey, you will receive 1 participation mark. You will not need to complete any activities to receive full participation marks.
4. If you complete all three parts of the study and later wish to withdraw from the study, you can email the PI (email address given below). You have two weeks after participation to withdraw from the study. Withdrawal will not affect the participation credit in the course, and you will still receive the full participation marks.

Who is conducting this research?
If you require any additional information about this study, please contact Maimoona Bashir at maimoona.bashir@uleth.ca.

Questions regarding your rights as a participant in this research may be addressed to the University's Ethics Committee at ulethics@ualberta.ca. Please quote Ethics ID: 2022-005 This research project has been reviewed for ethical acceptability and approved by the University of Alberta Research Ethics Committee. Thank you for your consideration.

If you wish to participate in the study, please enter your full name, click " **I choose to participate in the study** ." We are collecting your names to record participation marks. A list of the names will be given to the marker of the course. The instructor will not be informed of who is on the list.

Thank you in advance for your participation.

Click one of the following buttons to proceed.

- **I choose to participate in the study.**
- **I choose not to participate in the study and choose to complete up to three alternative activities for full participation marks.**

Appendix B

Invitation to Participate in a Research Study

Hello everyone, my name is Maimoona Bashir. I am a second-year graduate student in Computer Science here at the University of Lethbridge. I am here inviting you to participate in a research study called "Learning Effects of Algorithm Concepts using a Mobile Game". This study is part of my thesis, which involves the creation of an educational mobile game. The game is designed to help the player learn and understand key concepts relating to search and sort algorithms.

The study we are conducting is to test whether the participant's knowledge increases after playing the game, as well as to get feedback on how we might improve the educational side of the game.

For participating in the study, you will be given up to **three participation marks in CPSC 1000**.
There are three parts to the study: Prior knowledge questions, playing the game or watching animations and a post-game survey.

1. If you complete prior knowledge questions you will receive 1 participation mark and may then complete two alternative activities to receive full participation marks.
2. If you complete the study portion, you will receive an additional participation mark, and may then complete one alternative activity to receive full participation marks.
3. If you complete the post-game survey, you will receive 1 participation mark. You will not need to complete any alternative activities to receive full participation marks.
4. If you complete all three parts of the study and later wish to withdraw from the study, you can email the me (email address given below). You have two weeks after participation to withdraw from the study. Withdrawal will not affect the participation credit in the course, and you will still receive the full participation marks.

Your participation in the study will require a maximum of 1-hour of your time. You will book a 1-hour session. A link to the scheduling application will be provided on Moodle. You will then be provided a Zoom link for the scheduled time.
Participating in this study involves five parts:

1. At the start of a session, you are asked to complete an informed consent form.
2. If you choose to continue with the study, you will then be asked to complete an initial survey containing demographic questions. You will also be given some questions to test your existing knowledge of searching and sorting algorithms. If you are interested in knowing your individual results, you can email me. Results can be sent once the study is complete. We will post the quiz and post knowledge answers on Moodle after the study finishes.
3. Next, you will be asked to either play the game or watch animations of the search and sort algorithms. Which task you do will be determined randomly before the session. After every search or sort task, a quiz will be given containing five (5) questions. All the answers will be accepted whether they are correct or incorrect as they are review questions.
4. After you have completed the game or watched all the animations you will complete a post-knowledge survey. The post-knowledge survey will ask you scenario-based questions to assess any changes in knowledge after completing the game or watching the animations.
5. At the end of the session, you will also be asked about your experience playing the game or watching the animations to learn about the algorithms. You might also be asked any clarifying questions based on my observations of you, like "Why did you make this decision?" or "What were you thinking when you were doing X?"

You will be asked to provide your name on the consent form so that you can receive the participation marks. A unique study id will be assigned to you to keep your data anonymous. Also, you will be given an option to opt-in to receiving the results of the study once the analysis is completed. If you choose to do so you will be asked to email me for the results. Your email address will be deleted as soon as the results are sent out to you.

Once the data is collected, I will match the pre-study and post-study surveys using your study id and make a list of names to give to a CPSC 1000 marker for entering the participation marks. Your name will be removed from the data, as it will no longer be needed. Your identity will be kept completely anonymous after this point and will never be used in any publication of the study or the results. Dr. John Anvik will not have access to information about who completed the study (either completely or partially) and who completed one, two or three of the alternative task(s). From the instructor's point of view all the students may have participated, some may have participated, or none may have participated. The instructor will not be able to tell as the Moodle grade book will only have a single field that will be filled by the marker.

I also want to mention that you can withdraw from the study at any time while taking the survey or playing the game if you decide you no longer want to participate. You would then need to complete the alternate activities to receive the full participation marks.

1. If you complete prior knowledge questions you will receive 1 participation mark and may then complete two alternative activities to receive full participation marks.
2. If you complete the study portion, you will receive an additional participation mark, and may then complete one alternative activity to receive full participation marks.
3. If you complete the post-game survey, you will receive 1 participation mark. You will not need to complete any alternative activities to receive full participation marks.
4. If you complete all three parts of the study and later wish to withdraw from the study, you can email me (email address given below). You have two weeks after participation to withdraw from the study. Withdrawal will not affect the participation credit in the course, and you will still receive the full participation marks.

An announcement will be posted on Moodle where you can book your 1-hour slot for the study.

Thank you for your time. If you have any questions about the study, please feel free to contact me at my email address miamoona.bashir@uleth.ca

Appendix C

User Study Demographic Questions

Please write your student id?

1. How would you rate your experience level with computer programming?

a. No Experience

b. Novice (small programs in a single file)

c. I am interested to learn but have no experience

d. I am learning but have no experience

e. Other

2. How would you rate your knowledge level with search algorithms?

a. I don't know what an algorithm is.

b. I have heard about search algorithms

c. I have **used** search algorithms in programs.

d. I have **implemented** search algorithms in programs.

3. How would you rate your knowledge level with sort algorithms?

a. I don't know what an algorithm is.

b. I have heard about sort algorithms

c. I have **used** sort algorithms in programs.

d. I have **implemented** sort algorithms in programs.

4. what is your educational background?

a. Computer Science

b. Non CS / Please write down

Appendix D

User Study Responses

Study_ID	Experience with Computer Programming	Knowledge of Search Algorithm	Knowledge of Sort Algorithm		Educational Background
101	b. Novice (small programs in a single file)	b. I have heard about search algorithms	a. I don't know what an algorithm	b. Non CS	New Media/ Education
102	c. I am interested to learn but have no experi	a. I don't know what an algorithm is.	a. I don't know what an algorithm	b. Non CS	Archaeology and Geography
103	d. I am learning but have no experience	b. I have heard about search algorithms	b. I have heard about sort algorith	b. Non CS	Geography
104	a. No Experience	a. I don't know what an algorithm is.	a. I don't know what an algorithm	b. Non CS	Psychology
105	b. Novice (small programs in a single file)	b. I have heard about search algorithms	b. I have heard about sort algorith	b. Non CS	New Media
106	a. No Experience	a. I don't know what an algorithm is.	a. I don't know what an algorithm	b. Non CS	History and Education
107	b. Novice (small programs in a single file)	c. I have used search algorithms in progr	b. I have heard about sort algorith	b. Non CS	BMus - DAA

Table D.1: Demographic Questions Responses

Study_ID	TYPE	COMMENTS
101	Game	Game-based learning is entertaining and interactive. It gives hands-on experience. However, I have numerical dyslexia it was difficult for me to answer number-related questions.
105	Game	I think game-based learning could be very effective. If I were able to go through the game a few more times I feel like I could understand sorting and ordering algorithms better. I like the visual and interactive demonstration of the algorithms.
106	Game	Yes, although I think more explanation of what was occurring and more practice would have helped me do better. I do think the game aspect helped me learn it more quickly
107	Game	I think so! Interaction with the material as I was encountering it helped me grasp the immediate concepts
102	Animation	Yes, I think game-based learning would provide a more hands-on approach, rather than watching videos.
103	Animation	For sure it can be beneficial, the game has both visual representation and written explanation allowing for different ways of collecting the knowledge.
104	Animation	I think it is fairly beneficial to learn about these algorithms through animations. One thing I would find helpful is for a review at the end of the functions each search and sort carry out. Overall, it was an interesting way to learn!

Table D.2: Participants' Comments

Type	Linear Q1	Linear Q2	Linear Q3	Learning
Game	same knowledge	no change	no change	No
Game	no change	no change	improved	No
Game	improved	no change	improved	Yes
Game	no change	same knowledge	improved	Yes
Animation	same knowledge	improved	improved	Yes
Animation	same knowledge	same knowledge	same knowledge	No
Animation	no change	no change	improved	No

Type	Binary Q1	Q2Binary	Q3Binary	Learning
Game	improved	improved	no change	Yes
Game	improved	improved	improved	Yes
Game	no change	improved	no change	No
Game	improved	improved	no change	Yes
Animation	improved	improved	no change	Yes
Animation	no change	improved	unclear	No
Animation	no change	improved	no change	No

Type	Insertion Q1	Insertion Q2	Insertion Q3	Learning
Game	no change	same knowledge	no change	No
Game	no change	Improved	no change	No
Game	improved	no change	no change	No
Game	improved	Improved	improved	Yes
Animation	no change	improved	improved	Yes
Animation	no change	improved	improved	Yes
Animation	improved	no change	no change	No

Type	Selection Q 1	Selection Q 2	Selection Q 3	Learning
Game	no change	no change	improved	No
Game	improved	improved	no change	Yes
Game	improved	improved	no change	Yes
Game	no change	improved	no change	No
Animation	improved	improved	improved	Yes
Animation	no change	improved	no change	No
Animation	no change	no change	improved	No

Type	Quick Q1	Quick Q2	Learning	
Game	improved	no change	No	
Game	improved	improved	Yes	
Game	improved	no change	No	
Game	improved	improved	Yes	
Animation	improved	no change	No	
Animation	no change	improved	No	
Animation	no change	improved	No	
Type	**Merge Q1**	**Merge Q2**	**Merge Q3**	**Learning**
Game	improved	improved	improved	Yes
Game	improved	no change	improved	Yes
Game	improved	improved	improved	Yes
Game	no change	improved	improved	Yes
Animation	improved	improved	improved	Yes
Animation	no change	improved	improved	Yes
Animation	no change	no change	no change	No

Table D.3: Final Results and Learning outcome

			PRIOR-KNOWLEDGE	
ID	Type	Linear search Q#1	Linear search Q#2	Linear search Q#3
101	Game	correct	Incorrect	Incorrect
105	Game	Incorrect	Incorrect	Incorrect
106	Game	Incorrect	Incorrect	Incorrect
107	Game	Incorrect	correct	Incorrect
102	Animation	correct	Incorrect	Incorrect
103	Animation	correct	correct	correct
104	Animation	Incorrect	Incorrect	Incorrect
ID	Type	Binary Search Q#1	Binary Search Q#2	Binary Search Q#3
101	Game	Incorrect	Incorrect	Incorrect
105	Game	Incorrect	Incorrect	Incorrect
106	Game	Incorrect	Incorrect	Incorrect
107	Game	Incorrect	Incorrect	Incorrect
102	Animation	Incorrect	Incorrect	Incorrect
103	Animation	Incorrect	Incorrect	correct
104	Animation	Incorrect	Incorrect	Incorrect
ID	Type	Insertion Sort Q#1	Insertion Sort Q#2	Insertion Sort Q#3
101	Game	Incorrect	Incorrect	Incorrect
105	Game	Incorrect	Incorrect	Incorrect
106	Game	Incorrect	Incorrect	Incorrect
107	Game	Incorrect	Incorrect	Incorrect
102	Animation	Incorrect	Incorrect	Incorrect
103	Animation	Incorrect	Incorrect	Incorrect
104	Animation	Incorrect	Incorrect	Incorrect

ID	Type	Selection Sort Q#1	Selection Sort Q#2	Selection Sort Q#3
101	Game	Incorrect	Incorrect	Incorrect
105	Game	Incorrect	Incorrect	Incorrect
106	Game	Incorrect	Incorrect	Incorrect
107	Game	Incorrect	Incorrect	Incorrect
102	Animation	Incorrect	Incorrect	Incorrect
103	Animation	Incorrect	Incorrect	Incorrect
104	Animation	Incorrect	Incorrect	Incorrect

ID	Type	Quick Sort Q#1	Quick Sort Q#2
101	Game	Incorrect	Incorrect
105	Game	Incorrect	Incorrect
106	Game	Incorrect	Incorrect
107	Game	Incorrect	Incorrect
102	Animation	Incorrect	Incorrect
103	Animation	Incorrect	Incorrect
104	Animation	Incorrect	Incorrect

ID	Type	Merge Sort Q#1	Merge Sort Q#2
101	Game	Incorrect	Incorrect
105	Game	Incorrect	Incorrect
106	Game	Incorrect	Incorrect
107	Game	Incorrect	Incorrect
102	Animation	Incorrect	Incorrect
103	Animation	Incorrect	Incorrect
104	Animation	Incorrect	Incorrect

Table D.4: Raw Data of Prior Knowledge Questions

			POST-GAME	
ID	**Type**	**Linear search Q#1**	**Linear search Q#2**	**Linear search Q#3**
101	Game	correct	Incorrect	Incorrect
105	Game	Incorrect	Incorrect	correct
106	Game	correct	Incorrect	correct
107	Game	Incorrect	correct	correct
102	Animation	correct	correct	correct
103	Animation	correct	correct	correct
104	Animation	Incorrect	Incorrect	correct
ID	**Type**	**Binary Search Q#1**	**Binary Search Q#2**	**Binary Search Q#3**
101	Game	correct	correct	Incorrect
105	Game	correct	correct	correct
106	Game	Incorrect	correct	Incorrect
107	Game	correct	correct	Incorrect
102	Animation	correct	correct	Incorrect
103	Animation	Incorrect	correct	Incorrect
104	Animation	Incorrect	correct	Incorrect
ID	**Type**	**Insertion Sort Q#1**	**Insertion Sort Q#2**	**Insertion Sort Q#3**
101	Game	Incorrect	correct	Incorrect
105	Game	Incorrect	correct	Incorrect
106	Game	correct	Incorrect	Incorrect
107	Game	correct	correct	correct
102	Animation	Incorrect	correct	correct
103	Animation	Incorrect	correct	correct
104	Animation	correct	Incorrect	Incorrect
ID	**Type**	**Selection Sort Q#1**	**Selection Sort Q#2**	**Selection Sort Q#3**
101	Game	Incorrect	Incorrect	correct
105	Game	correct	correct	Incorrect
106	Game	correct	correct	Incorrect
107	Game	Incorrect	correct	Incorrect
102	Animation	correct	correct	correct
103	Animation	Incorrect	correct	Incorrect
104	Animation	Incorrect	Incorrect	correct

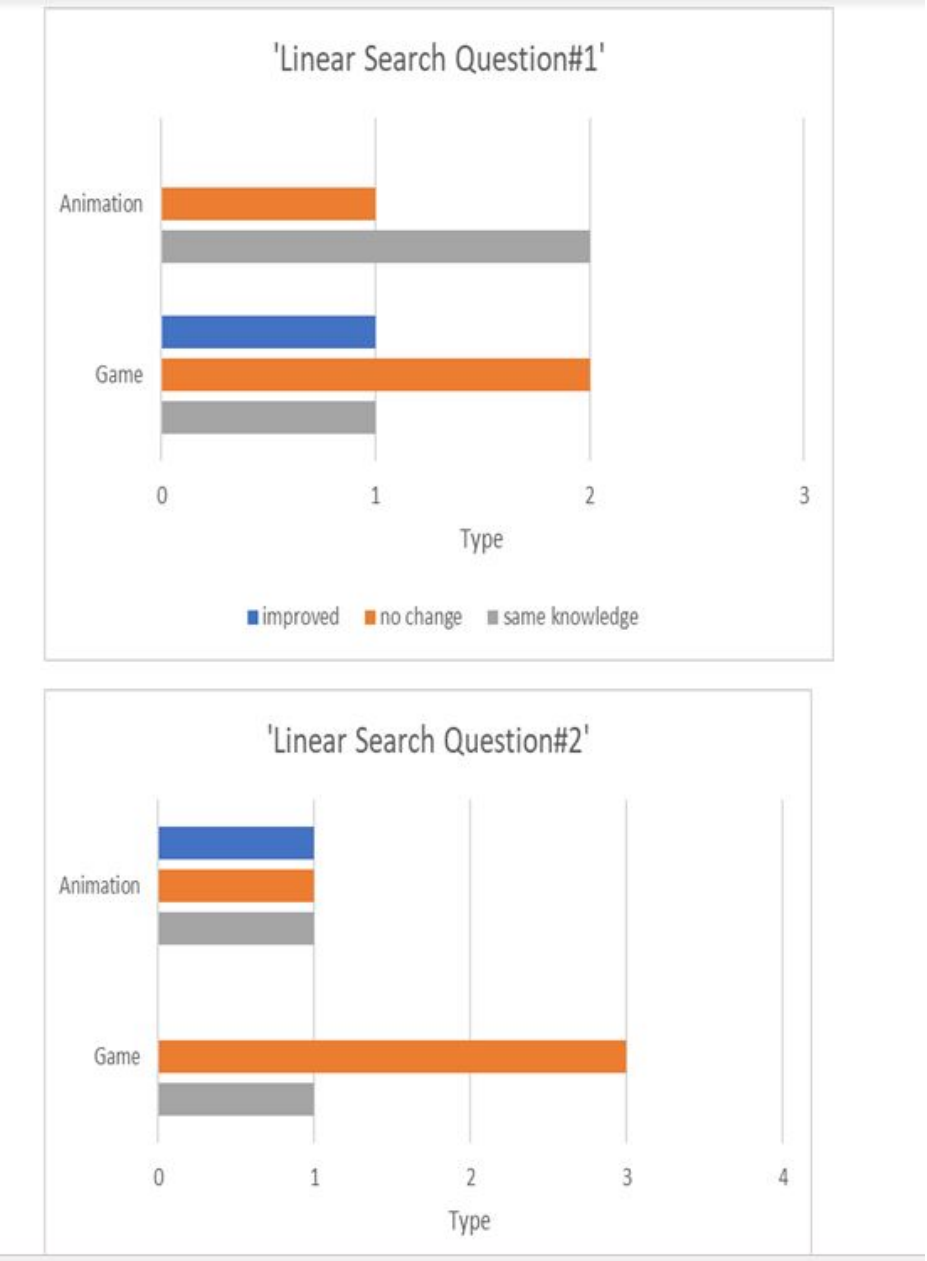
Table D.5: Raw Data of Post-Game Questions

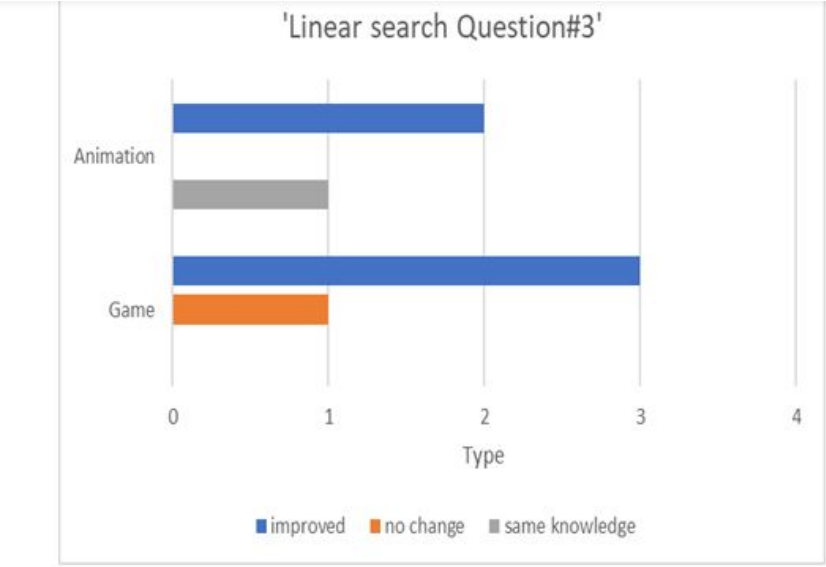
ID	Type	Quick Sort Q#1	Quick Sort Q#2
101	Game	correct	Incorrect
105	Game	correct	correct
106	Game	correct	Incorrect
107	Game	correct	correct
102	Animation	correct	Incorrect
103	Animation	Incorrect	correct
104	Animation	Incorrect	correct
ID	**Type**	**Merge Sort Q#1**	**Merge Sort Q#2**
101	Game	correct	correct
105	Game	correct	Incorrect
106	Game	correct	correct
107	Game	Incorrect	correct
102	Animation	correct	correct
103	Animation	Incorrect	correct
104	Animation	Incorrect	Incorrect

Table D.6: Raw Data of Post-Game Quick and Merge Sort Questions

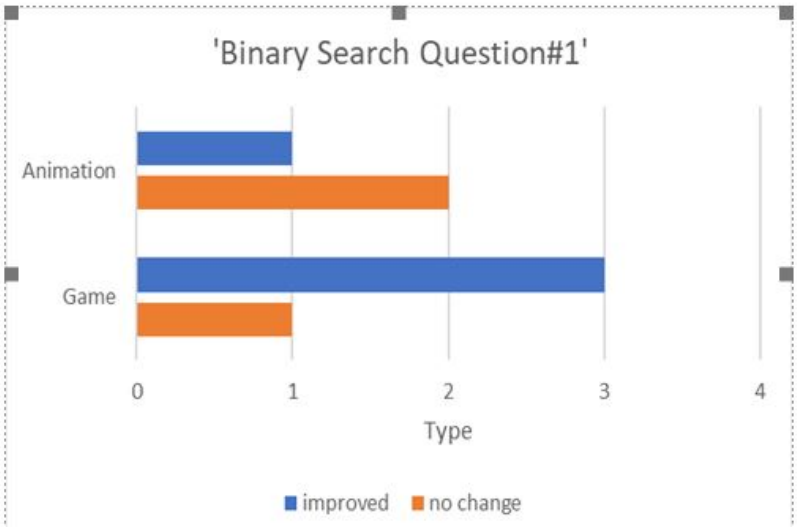
Appendix E

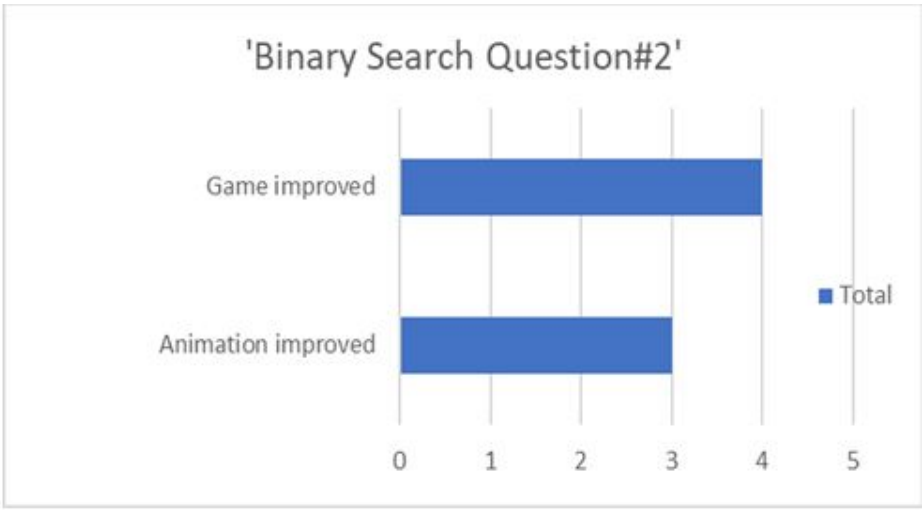
User Study Graphs

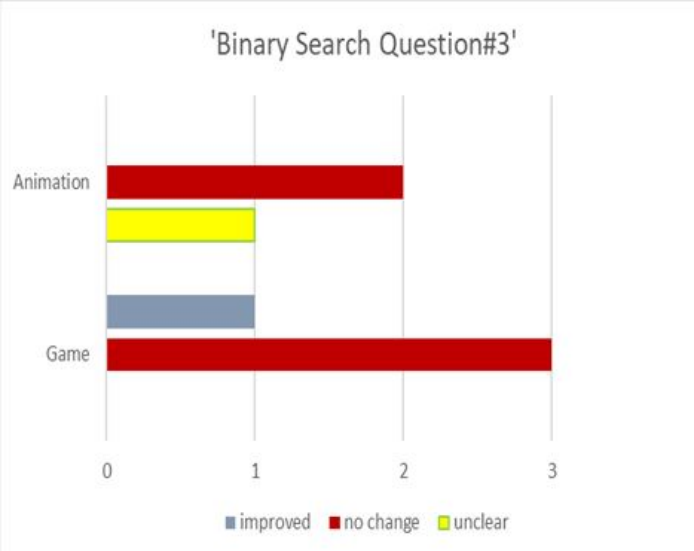




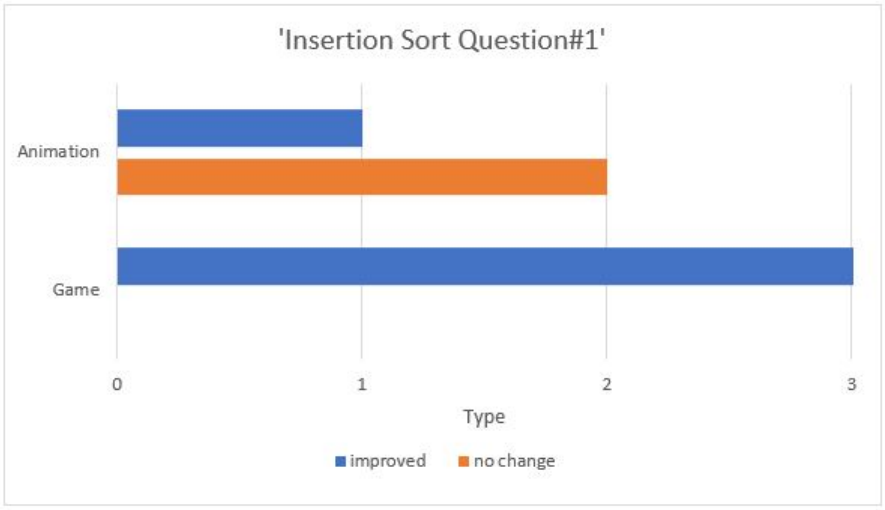
Binary Search Game VS Animation

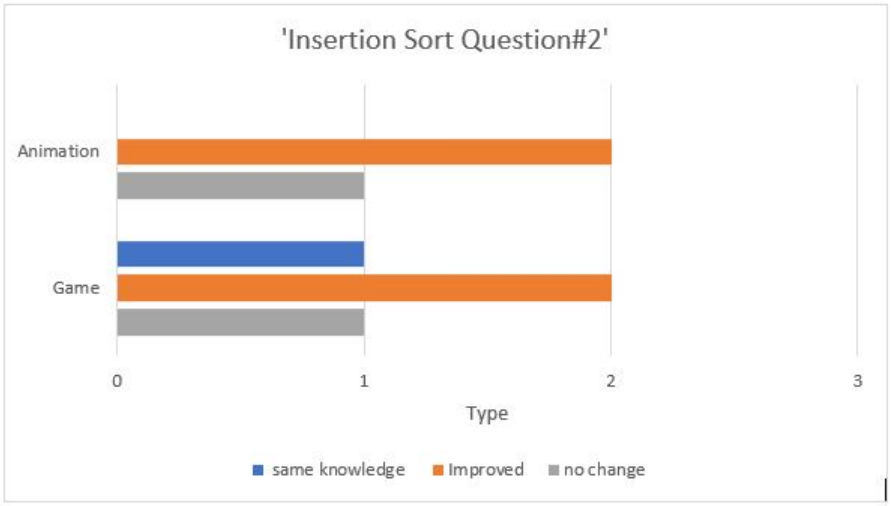


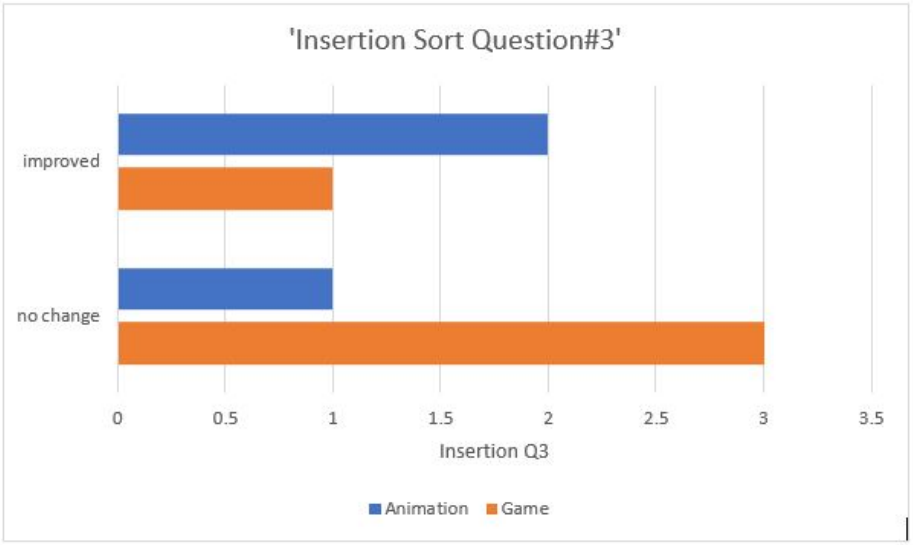




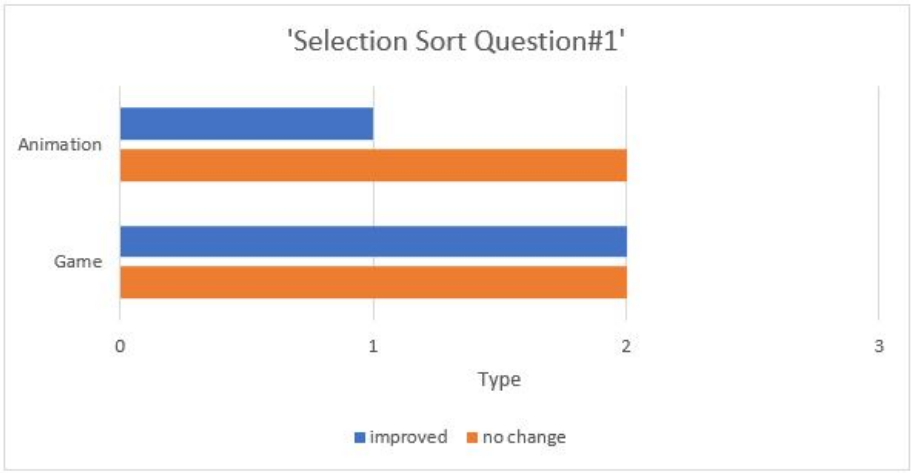
Insertion Sort Game VS Animation

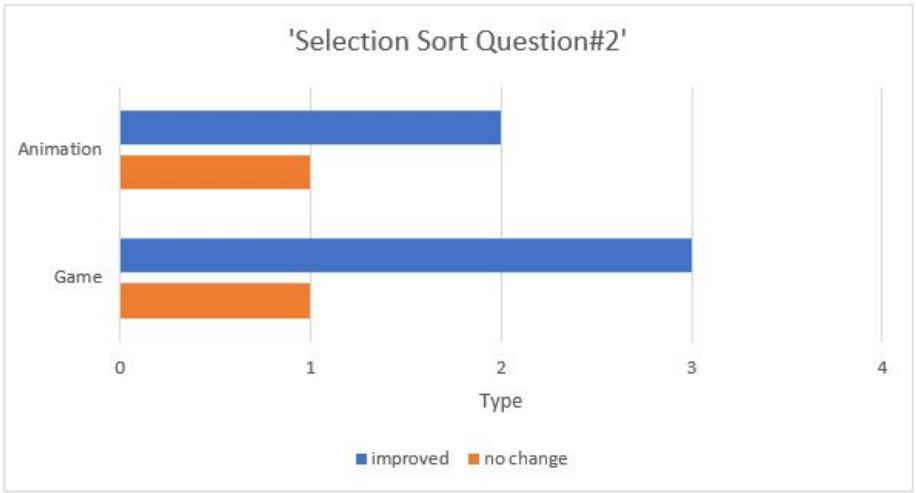


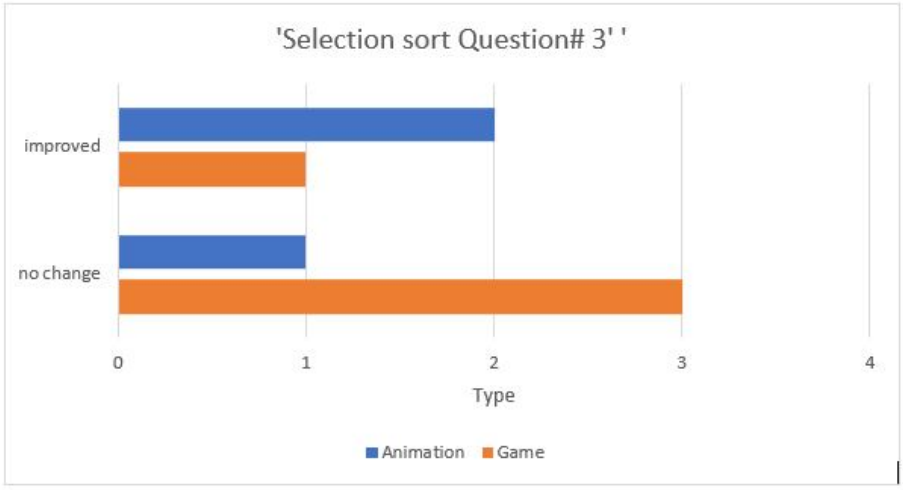




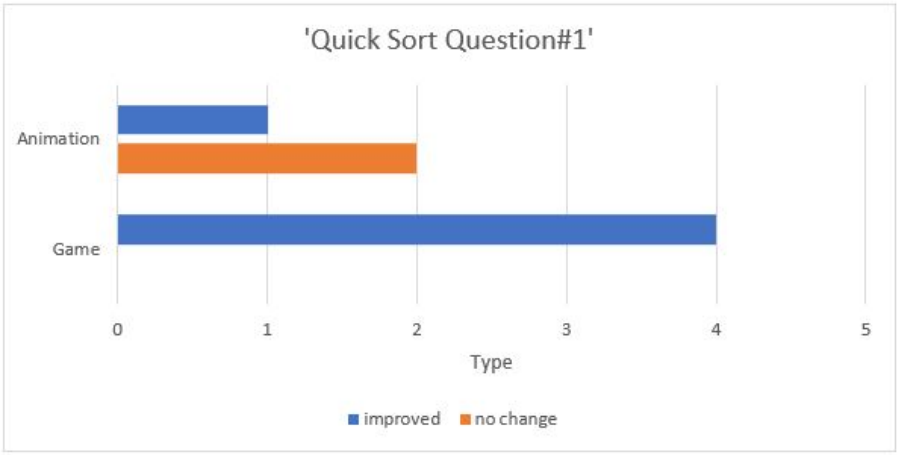
Selection Sort Game VS Animation

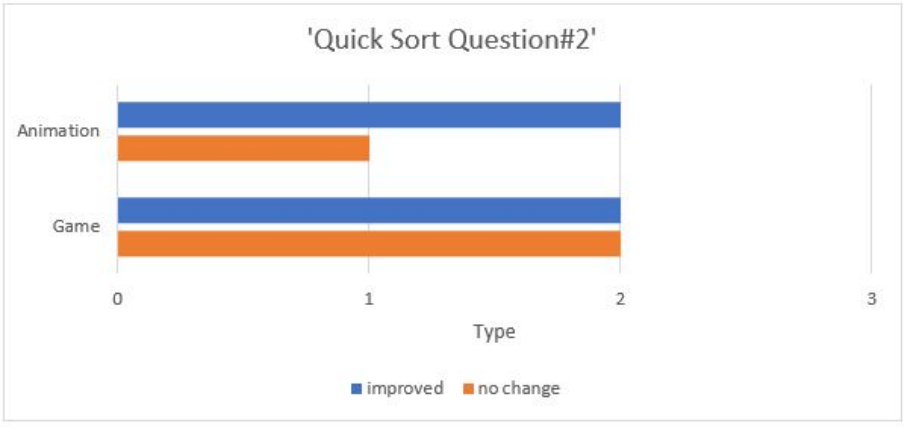




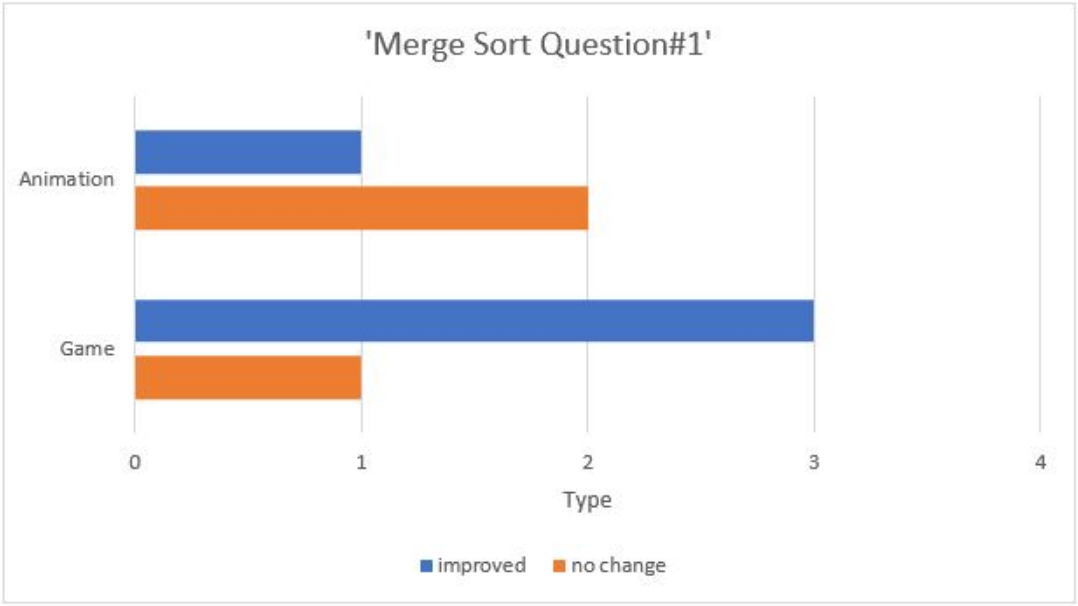


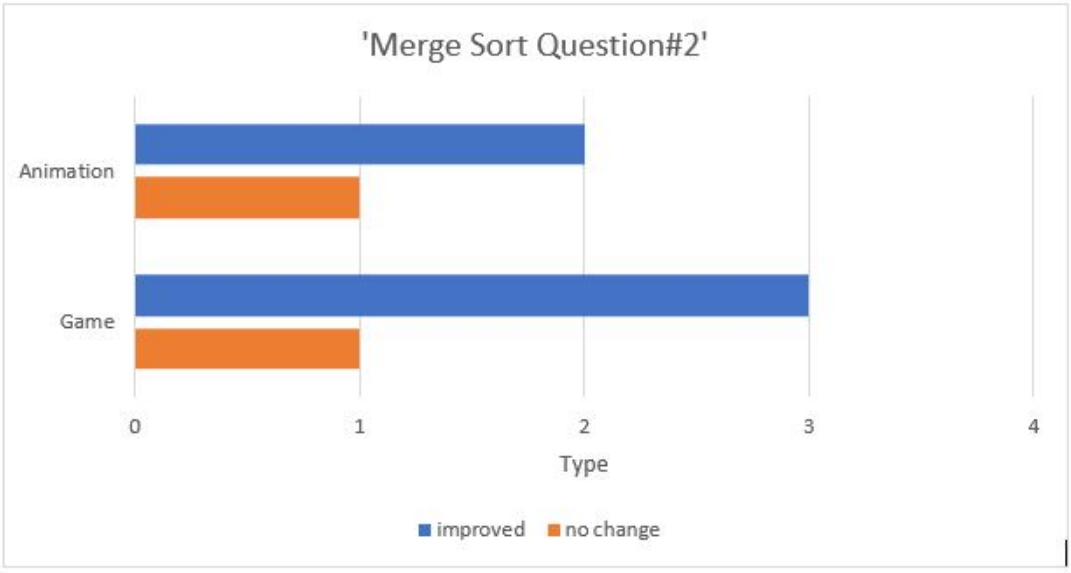
Quick Sort Game VS Animation





Merge Sort Game VS Animation





Appendix F

User Study Question Type

F.1 Conceptual Questions

How does selection sort work?

a. Check the item in the middle of the list , and put all of the items less than the middle to left and all the items bigger to right. Keep doing this for the lists to the left and right of the middle.

b. Compare each item in the list with its adjacent item and swap them if needed. Keep doing this to the list until no more swaps are made.

c. Find the smallest item in the list and exchange it with the left-most unsorted item in the list.

d. Keep dividing the list in two until there are only lists of one item. Combine the lists, putting the items in the correct order to create lists of two items. Keep doing this with larger and larger lists.

e. I don't know

What is the first step in the selection sort algorithm?

a. Find the largest item in the list

b. Swap adjacent elements

c. Find the smallest item in the list

d. Divide the list in half

e. I don't know

Quick sort is called a "divide and conquer" algorithm. Which of the following describes a "divide and conquer" algorithm?

a. The list being divided into sub lists with equal numbers of elements in each, then those sorted sub lists are merged back together.

b. The list being divided into only two sub lists, never more or less, which are sorted and merged back together.

c. The list being divided into smaller sub lists, then those sorted sub lists are merged back together.

d. I don't know

How do merge sort work?

a. Taking one item at a time from an unsorted list, each new item is compared with the previous until its place is found.

b. By splitting the list to single elements before merging them back together, one sub list at a time.

c. Each item in the list is individually compared with the following item starting with the first value till the last.

d. Each item in the list is compared with the following item starting with the last value till the first.

e. I don't know

Figure F.1: Conceptual Questions

F.2 Practical Questions

How many comparisons and swaps are needed for insertion sort to sort the next number which is 5?

[1 3 4 8 9 || 5 2]

a. 2 comparisons, 3 swaps

b. 3 comparisons, 2 swaps

c. 4 comparisons, 3 swaps

d. 3 comparisons, 3 swaps

e. I don't know

How many comparisons are required by selection sort to sort the following list into ascending order?
[5,4,3,2,1]

a. 5

b. 10

c. 20

d. 25

e. I don't know

Which of the following shows the correct list after the first iteration of quick sort?

[7, 11, 14, 6, 9, 4, 3, 12]

a. 7,6,4,3,9,14,11,12

b. 7,6,4,3,14,9,11,12

c. 7,6,14,11,9,4,3,12

d. 7,6,3,4,9,14,11,12

e. I don't know

What are the correct intermediate steps of the following item set when it is being sorted with the Merge sort?
[15,20,10,18]

a. 15,18,10,20 -- 10,18,15,20 -- 10,15,18,20

b. 15,20,10,18 -- 10,15,20,18 -- 10,15,18,20

c. 15,10,20,18 -- 15,10,18,20 -- 10,15,18,20

d. 15,20,10,18 -- 15,10,20,18 -- 10,15,20,18

e. I don't know

Figure F.2: Practical Questions

F.3 Post-game Questionnaire

Which of the following can be found using linear search?

a. Letters

b. Numbers

c. Words

d. All of the above

e. None of the above

f. I don't know

How many comparisons will linear search use to find the value 14 in this list of numbers?

[3, 100, 1, 14, 8, 7, 10, 28]?

a. 3

b. 4

c. 6

d. 8

e. I don't know

Do the items need to be sorted to use the linear search algorithm?

Yes

No

I don't know

Which of the following best describes binary search?

a. Looks at each item until it finds the item

b. Converts all the items into binary numbers

c. Splits the list in half repeatedly until it finds the item

d. None of the above

e. I don't know

Does the list of items have to be sorted for binary search to work?

a. Yes

b. No

c. I don't know

A binary search is performed on this list:
[1 5 10 13 48 68 100 101]

How many comparisons are done to find number 101?

a. 0-1

b. 1-2

c. 3-4

d. 4-5

e. I don't know

why is insertion sort important?

a. Because it makes games

b. Because it makes easy for us to do our homework

c. Because if we don't sort it will be hard to find an item from the list

d. Because it makes easy to insert new items

e. I don't know

Which of the following shows how the list will change when it is being sorted with the Insertion sort? [15,20,10,18]

a. 15,20,10,18 -- 10,15,20,18 -- 10,15,18,20 -- 10,15,18,20

b. 15,20,10,18 -- 20,15,10,18 -- 20,10,15,18 – 10,15,18,20

c. 15,20,10,18 -- 18,15,20,10 -- 10,18,15,20 -- 10,15,18,20

d. 15,20,10,18 -- 20,10,18, 15 -- 10,18, 15, 20, -- 10,15, 18, 20

e. I don't know

How many comparisons and swaps are needed for insertion sort to sort the next number which is 5?

[1 3 4 8 9 || 5 2]

a. 2 comparisons, 3 swaps

b. 3 comparisons, 2 swaps

c. 4 comparisons, 3 swaps

d. 3 comparisons, 3 swaps

e. I don't know

How does selection sort work?

a. Check the item in the middle of the list , and put all of the items less than the middle to left and all the items bigger to right. Keep doing this for the lists to the left and right of the middle.

b. Compare each item in the list with its adjacent item and swap them if needed. Keep doing this to the list until no more swaps are made.

c. Find the smallest item in the list and exchange it with the left-most unsorted item in the list.

d. Keep dividing the list in two until there are only lists of one item. Combine the lists, putting the items in the correct order to create lists of two items. Keep doing this with larger and larger lists.

e. I don't know

What is the first step in the selection sort algorithm?

a. Find the largest item in the list

b. Swap adjacent elements

c. Find the smallest item in the list

d. Divide the list in half

e. I don't know

How many comparisons are required by selection sort to sort the following list into ascending order?

[5,4,3,2,1]

a. 5

b. 10

c. 20

d. 25

e. I don't know

Quick sort is called a "divide and conquer" algorithm. Which of the following describes a "divide and conquer" algorithm?

a. The list being divided into sub lists with equal numbers of elements in each, then those sorted sub lists are merged back together.

b. The list being divided into only two sub lists, never more or less, which are sorted and merged back together.

c. The list being divided into smaller sub lists, then those sorted sub lists are merged back together.

d. I don't know

Which of the following shows the correct list after the first iteration of quick sort?

[7, 11, 14, 6, 9, 4, 3, 12]

a. 7,6,4,3,9,14,11,12

b. 7,6,4,3,14,9,11,12

c. 7,6,14,11,9,4,3,12

d. 7,6,3,4,9,14,11,12

e. I don't know

How do merge sort work?

a. Taking one item at a time from an unsorted list, each new item is compared with the previous until its place is found.

b. By splitting the list to single elements before merging them back together, one sub list at a time.

c. Each item in the list is individually compared with the following item starting with the first value till the last.

d. Each item in the list is compared with the following item starting with the last value till the first.

e. I don't know

The following two lists are to be merged, which element first goes into the new merged list?

List 1: 2, 5, 6, 7
List 2: 1, 9, 12, 15

a. 2

b. 5

c. 1

d. 9

For what purpose would a merge sort algorithm be used?

a. To reorder multiple lists into a singular ordered list.

b. To combine identical lists together.

c. To reorder a single list.

d. Both 1 & 3

e. I don't know