Visualising Divide-and-Conquer Algorithms

Pradnesh Sanderan

Master of Science
Computer Science
School of Informatics
University of Edinburgh
2025

Abstract

Divide-and-conquer (D&C) algorithms represent a fundamental strategy in computer science, yet their recursive structure and multi-phase logic create significant learning challenges for students. Existing visualisation tools primarily focus on low-level code execution traces, which obscure the higher-level strategy of problem decomposition and solution recombination that defines the D&C approach.

This dissertation presents the design, implementation, and evaluation of an interactive web-based visualisation tool that addresses these educational challenges through a novel phase-explicit approach. The tool implements three visualisation modes: analogical views that ground abstract concepts in familiar real-world scenarios, standard views that provide guided algorithmic exploration, and explorative views that enable hands-on manipulation and hypothesis testing. These multi-modal representations support diverse learning preferences while maintaining cognitive load management principles derived from educational theory.

The tool was developed using React and JavaScript, and it implements algorithms including binary search, merge sort, and Strassen's matrix multiplication. The design process followed an iterative user-centred methodology, with multiple rounds of think-aloud sessions informing key design decisions such as the current-stage visualisation approach and phase-explicit labelling.

Evaluation through pre-post knowledge assessments with nine participants demonstrated measurable improvements in conceptual understanding, with strong gains in merge sort recursion comprehension (100% improvement) and base case recognition (133% improvement). Qualitative analysis revealed enhanced technical vocabulary usage and more structured responses. The tool achieved a System Usability Scale score of 85.0, exceeding the industry benchmark of 68 and indicating excellent usability.

This research shows that phase-explicit visualisation can improve D&C algorithm comprehension without compromising usability, providing a foundation for future educational technology development. The work demonstrates how learning theory, human-computer interaction principles, and algorithm visualisation can be integrated to create more effective educational tools for computer science concepts that traditionally challenge learners.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 778012
Date when approval was obtained:2022-10-24
The participants' information sheet and a consent form are included in the appendix.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Pradnesh Sanderan*)

Acknowledgements

I would like to thank my supervisor, Dr Murray Cole for his continuous guidance and support throughout this project. Many thanks to my parents and family for their advice and encouragement. Lastly, many thanks to my friends for helping me get through the long nights I spent in Appleton Tower.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Divide and conquer algorithms (D&C) form a cornerstone of computer science education as they represent important computational strategies used throughout this field. However, the recursive structure and multi-phase logic (divide, conquer, combine) create significant learning challenges that often confuse students.

Research has shown that recursion is one of the most difficult concepts for students to master [29][30]. D&C algorithms compound this difficulty by requiring students to simultaneously understand recursive decomposition, base case identification and solution combination across multiple abstraction levels. Students often struggle to visualise how problems are broken down and how partial solutions recombine to solve the original problem [21].

Although there are existing visualisation tools such as VisuAlgo and PyAlgoViz, which are useful for many algorithmic concepts, these tools primarily focus on low-level, line-by-line code execution. This approach visualises the flow but fails to convey the bigger picture of D&C algorithms, which is the decomposition of the problem into subproblems and the recombination of these partial solutions, thus forcing students to infer this from code. Hence, students may be able to trace algorithm execution without grasping the underlying divide-and-conquer strategy.

This gap between tool capabilities and learning needs motivates the development of visualisation approaches specifically designed for D&C algorithm education.

1.2 Research questions and approach

This dissertation addresses the abstraction gap in D&C algorithm education by developing an interactive web-based tool that emphasises algorithmic phases and enhances students' understanding through real-world analogies. The primary research question is: Can phase-explicit and exploratory visualisation improve students' understanding of D&C algorithms?

These specific questions guide this investigation:

1. How does explicit visualisation of divide, conquer and combine phases affect student comprehension of D&C algorithms?
2. Do real-world analogies play a role in helping students grasp abstract D&C concepts?
3. How does interactive exploration impact student engagement with recursive problem-solving?

The research uses an iterative design methodology with multiple rounds of think-aloud user sessions to develop and refine the tool. This approach emphasises making abstract recursive concepts easily understandable through interactive modes. Students can independently explore subproblem decomposition and recombination and relate it to real-world problems, all while maintaining algorithmic accuracy.

1.3 Contributions

This research makes two primary contributions to computer science education:

- A phase-explicit and exploratory visualisation framework that identifies and separates the divide, conquer and combine phases of a D&C algorithm while providing contextual explanations.
- Evidence that such visualisations can improve conceptual understanding of D&C algorithms without compromising usability.
- A mixed-method evaluation of phase-based learning effectiveness for D&C algorithm learning.

These contributions address gaps in educational technology research, offering evidence-based guidance for developing more effective algorithm learning tools.

1.4 Interactive Prototype

A working prototype of the visualisation tool developed in this research is available online at: [Prototype Link](https://shiny-cuchufli-7ce4c3.netlify.app/).Readers are encouraged to explore the tool and its interactive features to gain firsthand experience with it.

1.5 Dissertation Structure

The dissertation is divided into 6 chapters and is structured as follows:

- **Chapter 2** provides the background, covering D&C fundamentals, learning challenges with recursion, educational visualisation research, theoretical foundations, and analysis of existing tools to establish the research foundation and identify gaps.
- **Chapter 3** presents the design methodology and rationale, outlining the theoretical framework, project requirements, and key design decisions that inform the development of the tool.
- **Chapter 4** describes the technical implementation of the system, covering the architecture, development methodology, and algorithm-specific visualisation approaches.
- **Chapter 5** evaluates the tool's effectiveness through user studies, presenting both quantitative and qualitative findings from usability testing and learning assessments.
- **Chapter 6** concludes the research by synthesising key findings, discussing the contributions to D&C algorithm education, and identifying future research opportunities.

Chapter 2

Background

This chapter provides the background for the project. Section 2.1 introduces the D&C strategy, outlining its three-phase structure and educational relevance. Section 2.2 reviews the learning challenges associated with recursion. Section 2.3 surveys research on educational visualisation, highlighting what makes algorithm visualisation tools effective and how phase-based approaches can reduce cognitive overload. Section 2.4 presents key educational theories that inform the design of learning tools. Section 2.5 critically reviews existing algorithm visualisation tools. Finally, Section 2.5.1 provides a comparative analysis of these tools.

2.1 Introduction to Divide and Conquer

Divide-and-conquer (D&C) represents one of the most elegant and powerful algorithmic strategies in computer science, characterised by its recursive approach to problem-solving: break a problem into smaller subproblems, solve these subproblems independently, and combine their solutions to solve the original problem [7]. This three-phase strategy is the basis of fundamental algorithms taught in computer science curricula, such as binary search, merge sort and Strassen's matrix multiplication. Beyond their algorithmic efficiency, D&C algorithms provide a pedagogically valuable way to introduce recursion, abstraction, and problem decomposition, which are necessary skills in computer science education.

Despite its importance, D&C algorithms present significant pedagogical challenges that distinguish them from other algorithmic concepts. The primary difficulty lies in their inherently recursive nature, which requires students to simultaneously understand multiple levels of abstraction while maintaining awareness of both local operations and

the problem-solving strategy. Students often struggle to visualise how subproblems relate to the original problem and how partial solutions combine to produce the final result. Traditional teaching approaches that rely on code tracing can obscure the simplicity of the D&C strategy, leading students to successfully trace algorithm execution without understanding why it works or knowing how to apply it. Hence, the need for educational tools specifically designed to make the D&C structure and strategy more accessible to learners.

2.2 Learning Challenges with Recursion

Research shows that recursion is one of the most difficult concepts for computer science students to master, and these challenges are especially pronounced in D&C algorithms. Students often confuse recursion with iteration, leading to misconceptions about how recursive calls are generated and terminated [8][10]. Errors with base cases are common, with many assuming recursion will terminate itself without clearly defining when it should stop.

In the context of D&C, these difficulties are worsened by the need to understand not only the recursive calls but also the three-phase structure of divide, conquer, and combine. Studies have shown that students can often identify the divide step but struggle to grasp the combine stage, leaving them unable to see how subproblem results reassemble into a full solution [13][3].

A key reason for these struggles is the lack of appropriate mental models for recursion. Sorva (2013) [22] highlights that students often rely on flawed schemas, such as “magic” models that assume recursion just works, or “loop” models that treat it as disguised iteration. Additionally, while students may follow the descent into recursive calls, they often fail to understand the return flow, particularly how results propagate back in the combine phase.

2.3 Educational Visualisation Research

Research in algorithm visualisation (AV) shows that simply providing animations is not enough for effective learning. Hundhausen, Douglas, and Stasko (2002) [11] developed an “engagement taxonomy” which highlights that the educational effectiveness of AV depends strongly on the level of learner engagement: passive viewing leads to limited gains, while active learning leads to much deeper understanding. Later meta-analyses

confirm this pattern, showing that interactive visualisations consistently outperform passive ones in supporting conceptual understanding [22]. These findings suggest that for a visualisation to be pedagogically useful, it must encourage learners to actively explore and test their own understanding, rather than simply watch algorithm steps unfold.

As for recursion and D&C algorithms, several studies argue that structural and phase-based representations are more helpful than execution-based code traces. Research shows that explicitly separating algorithmic phases reduces cognitive overload and supports students in grasping the "big picture" of recursion. This highlights the need for tools that emphasise how problems are broken down and recombined, rather than focusing solely on the step-by-step details of function calls.

Finally, research in human-computer interaction (HCI) provides important design principles for educational tools. Nielsen's heuristics[17], such as recognition over recall, user control, and consistency, have been widely applied to learning tools to make them more usable and accessible. In educational tools, recognition over recall is important as labelling phases and showing visual cues reduce the cognitive burden on learners. Another relevant principle is progressive disclosure, where information is revealed gradually to prevent cognitive overload [23]. These insights suggest that visualisations should present information in manageable steps, while still giving them control to expand or collapse subproblems as needed.

2.4 Theoretical Foundations

Learning algorithms is tough, not just because they're technical, but because they're mentally challenging. Educational theories help us understand why ideas like recursion and divide-and-conquer are so hard to learn. Cognitive Load Theory (CLT) [23] distinguishes between intrinsic load (inherent task complexity), extraneous load (unnecessary distractions), and germane load (constructive processing that supports learning). In the case of D&C, students must simultaneously think about recursive calls, problem decomposition, and solution recombination, which is an inherently high intrinsic load that can overwhelm working memory. Research shows that structuring content into manageable "chunks" and reducing extraneous details helps lessen these difficulties [24]. Beyond cognitive load, other theories emphasise how students actively build their understanding. Constructivist theory highlights that learners develop knowledge through active engagement and scaffolding within their Zone of Proximal Develop-

ment (ZPD)[4]. This supports a staged approach, where students move from concrete analogies to abstract representations with appropriate guidance.

2.5 Existing Tools

There are several educational tools and platforms that support algorithm learning by differing in focus and effectiveness, especially for teaching D&C strategy.

VisuAlgo is the most comprehensive, offering step-by-step walkthroughs of dozens of algorithms. Its strength is that it has broad coverage and clear animations, but its passive presentation can limit deep learning: learners mainly watch algorithms execute, with limited opportunities for exploration or conceptual scaffolding.

Algorithm Visualizer provides a more modern, web-based interface with interactive controls and visual debugging. While it improves engagement, its design still emphasises line-by-line code tracing rather than highlighting the structural phases of D&C. This risks reinforcing procedural understanding without helping students grasp the recursive decomposition and recombination process[22].

PyAlgoViz is a Python-based visualisation library that allows instructors and students to generate custom algorithm visualisations. Its strength is its flexibility, but it requires programming knowledge to use effectively, and its default view remains tied to code executions.

2.5.1 Comparative analysis

Tool	Strengths	Limitations
VisuAlgo	Wide coverage, clear animations	Mostly passive, execution-focused, little scaffolding
Algorithm Visualizer	Web-based, interactive controls	Code-tracing focus, limited conceptual support
PyAlgoViz	Flexible, custom visualizations	Requires coding, still execution-centric, steep learning curve

Table 2.1: Comparison of existing algorithm visualisation tools

As summarised in Table 2.1, existing tools present complementary strengths but also notable shared weaknesses. visuAlgo excels in accessibility and breadth, Algorithm

Visualizer in interactivity and PyAlgoViz in customisability. However, none explicitly separates the phase structure of D&C or integrates analogies to help users build mental models. This highlights a critical gap, where existing tools can show recursion unfolding but rarely help users understand its structural phases.

Research gaps: Despite the strengths of the existing tools, several gaps remain that limit their effectiveness for teaching D&C algorithms. Theoretical gaps arise from the fact that most tools emphasise code execution rather than drawing on established learning theories such as Cognitive Load Theory and constructivism. This limits their ability to scaffold learners through high-intrinsic-load concepts like recursion and phase coordination. Empirically, few studies evaluate how these tools support student understanding of recursion or D&C specifically, leaving open questions about their impact beyond surface-level feedback. Technically, current tools tend to focus on execution traces and do not explicitly highlight the structural phases of divide, conquer, and combine. They also provide limited opportunities for exploratory interactivity, such as allowing students to manipulate subproblems, test hypotheses, or make and recover from mistakes. Finally, analogical grounding, which is a strategy to support beginner learners by linking abstract recursion to familiar contexts, is rarely incorporated into the tools. Together, these gaps highlight an opportunity for tools that combine theoretical grounding, empirical validation, and technical innovation to more effectively support the learning of D&C algorithms.

Chapter 3

Design

This chapter details the design of the tool. Section 3.1 outlines the design philosophy and theoretical framework, including the HCI principles and learning theories that guided the project. Section 3.2 identifies the project's goals and requirements. Section 3.3 presents the user-centred design process, and Section 3.4 discusses the core design decisions and their rationale, covering the visualisation approach, phase-explicit design, and multi-modal learning support. Finally, Section 3.5 concludes by detailing the interface design and user experience.

3.1 Theoretical Foundations

The design of this tool was guided by a blend of HCI principles and learning design principles, as detailed below:

3.1.1 HCI principles

The following HCI principles were applied to enhance usability, accessibility and engagement:

Recognitions over recall - clear labels and visually distinguishable phases for divide, conquer and combine phases ensured students could easily identify phases without memorising steps.

User control and freedom - next and previous step buttons, editable input controls and the exploratory mode encouraged interactive learning and self-directed exploration. However, this created tension with cognitive load management, where too much control can overwhelm novices. We solved this through progressive control revelation with

basic navigation in standard mode and full control in explorative mode.
Iterative refinement - feedback from think-aloud sessions helped refine the tool to better match student needs.
Visibility and feedback - Information panels provide instant feedback, clarifying the effects of user actions on algorithm behaviour. This addresses a gap in existing tools where students would have to infer algorithmic consequences from state changes.
Help & Documentation - on-page guides and onboarding prompts ensure that users can learn to operate the tool effectively with minimal external assistance. This principle became essential after observing that 100% of initial users struggled with feature discovery.

3.1.2 Cognitive load theory

Cognitive load theory [23] influenced both the phase separation strategy and how we presented the D&C algorithms. Presenting multiple interconnected concepts simultaneously can overwhelm the working memory. Hence, we explicitly separated, divide, conquer and combine phases (three-phase model) visually to allow the users to process each conceptual chunk independently. A stage-based approach was chosen over a binary tree representation to minimise extraneous load and present only essential details at any given time 3.4.1.

3.1.3 Constructivist learning theory

Constructivist learning theory [26][19] states that learners actively build knowledge through experience and reflection rather than passively receiving information. This theory influenced the design in 3 ways:
Active learning- The exploratory mode 3.4.3 allows students to manipulate the algorithm flow and observe real-time behaviour changes.
Scaffold discovery- Progression from real-world analogies to the standard view to the explorative view follows Vygotsky's Zone of proximal development[15]. Here, users would begin with familiar concepts before progressing to abstract representations with contextual support. Each transition point was validated through user testing to ensure appropriate challenge levels.
Collaborative learning- The tool supports collaborative learning through teacher-led demonstrations, enabling social knowledge construction through discussion and shared observation.

3.1.4 Design Philosophy

Our design philosophy prioritises educational effectiveness over technical sophistication. All technical implementations serve pedagogical goals, ensuring that cognitive load remains manageable and students engage as active participants in their learning process

3.2 Project goals and requirements

The goal of this project was to design and implement an interactive educational tool that helps students develop a better understanding of D&C algorithms. This project aimed to bridge the gap between algorithmic logic and conceptual understanding through phase-explicit visualisations and contextual learning approaches.
This project was guided by five goals that were derived from the gaps in current educational tools:
Educational Goals: Improve conceptual understanding of D&C algorithms by enabling users to develop robust mental models of problem decomposition and recombination independent of line-by-line code-tracing. Enable accurate phase identification to ensure students can correctly distinguish between divide, conquer and combine phases.
Pedagogical Goals: Support multiple representations by Providing both algorithmic(step-by-step execution) and analogical (real-world scenarios) views.Facilitate active learning by enabling independent exploration through interactive manipulation rather than passive observation.
Methodological Goals: Evidence-based iterative refinement by applying a systemic user-centred design process using think-aloud sessions and user studies to iteratively refine the tool.

3.2.1 Functional Requirements

1. **Step-by-step navigation with forward/backwards controls:** Navigation must maintain context across recursive levels, and controls must be consistently labelled (phase-based in analogical mode, directional in standard mode)
2. **Interactive visualisations with multi-level abstraction:** Visual highlighting of algorithmically significant elements and smooth animated transitions between algorithmic states
3. **User-modifiable algorithm flow:** Support user-modifiable algorithm flow by

providing input modification capabilities with validation and error handling, all of which trigger a real-time algorithm response.

4. **Phase classification with the visualisation, visual indicators, and contextual explanations:** Explicit visual separation of divide, conquer and combine phases and contextual information planes with phase-specific explanations
5. **Structured cycles of prototyping, testing and feedback integration:** Capability for rapid prototype iteration based on user feedback

3.2.2 Success Criteria and Evaluation Metrics

The success of this tool will be judged based on the following criteria)Post questionnaire average score $\geq 70\%$. 2)Improved conceptual understanding reflected in short-answer responses 3)average SUS score ≥ 68.0

3.3 User Centred Design Process

3.3.1 Target Audience Analysis

When developing the tool, three main target user groups were identified, each with distinct learning needs and interaction preferences:

- **Teachers or tutors :**Use the tool as a classroom aid and may integrate the visualisations into live demonstrations or assign them for independent student use. Require clarity, ease of navigation and pacing control.
- **Beginner users :** Users with limited to no prior exposure to D&C algorithms. They would require context-based explanations that introduce the core concepts without overwhelming details. Concrete analogies with proper labels and explanations would be required to help them build a foundational understanding before moving to abstract representation.
- **Advanced users :** Users who are already familiar with some D&C algorithms may seek deeper insights or may want to explore algorithms that are more conceptually difficult. For them, exploratory controls and more difficult algorithms would be needed to keep them engaged.

Recognising these distinct needs shaped our design approach. These design decisions are further discussed in the following sections.

3.3.2 Iterative Design Methodology

The project followed an iterative user-centred design methodology involving cycles of prototyping, user evaluation and refinement. Initial analysis of the existing tools (detailed in the background section), including VisuAlgo and PyAlgoViz, revealed key gaps:1)Lack of explicit differentiation of the divide, conquer and combine phase.s 2)Absence of real-world analogies. 3)Limited exploration capabilities.

The prototypes were tested in multiple informal think-aloud sessions with undergraduate students of varying prior computer science knowledge. Through these sessions and from the feedback gained, many gaps were identified in the tool and multiple features, such as the real-world analogies and the exploratory mode, were added to aid the users in their understanding of D&C algorithms. Iteration continued until the interaction patterns and learning outcomes converged on the intended educational objectives.

3.3.3 Algorithm Selection Rationale

Three D&C algorithms were chosen for the visualisation: merge sort, binary search and Strassen's matrix multiplication algorithm. These algorithms were chosen for their common occurrence in computer science curricula, clear visual representation opportunities, coverage of distinct application types (sorting, searching and mathematical computation) and progressive difficulty to accommodate both beginners and advanced learners.

3.4 Core Design Decisions and Rationale

3.4.1 Visualization Approach Selection

The primary design challenge for the visualisations was choosing how to represent the recursive structure in the tool in an effective manner. Two approaches were considered:

- **Tree diagram approach:** A hierarchical representation that shows the recursive structure with branching decomposition and merging recombination, hence forming a kind of diamond shape at the end.
- **Current Stage approach:** A state-based approach that shows the data structure (arrays or matrices) at each algorithmic phase with animated transitions to convey the operations and transitions between the phases.

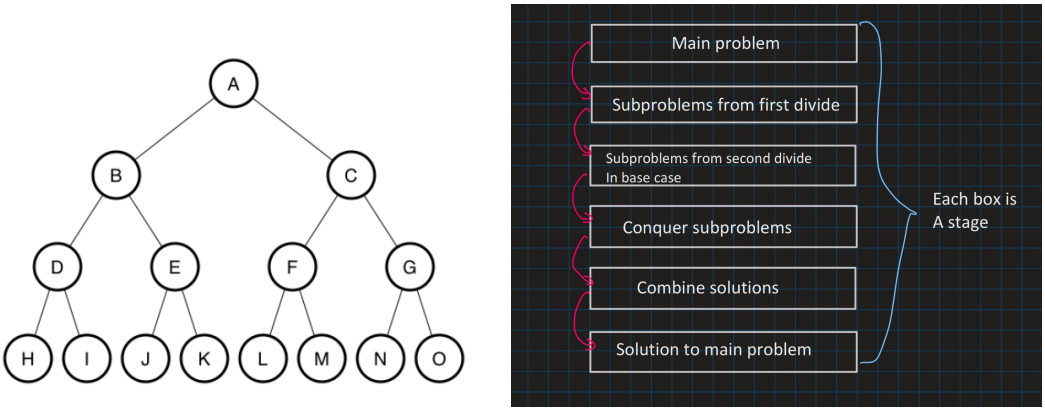

Figure 3.1: Binary Tree approach vs Current Stage approach

The current stage approach was chosen to reduce visual complexity for beginners while making phase transitions explicit. This decision involved significant trade-offs that required careful analysis.

Advantages of Tree approach: 1) Directly represents a recursive call structure. 2) Familiar to computer science educators and advanced students. 3) Shows complete problem decomposition simultaneously.

Disadvantages of the tree approach: 1) Cognitive overload for beginners as they may struggle with the tree complexity 2) The phase operations are embedded in tree relationships rather than being explicit and may be obscured. 3) Trees have visual scalability issues where they become unreadable for problems with depth > 4

Advantages of current stage approach: 1) Reduces cognitive load as it presents on the current level information 2) Explicit phase visualisation where each phase is represented through clear data structure transformations. 3) There is better conceptual alignment as it reflects the simultaneous nature of D&C operations.

Disadvantages of current stage approach: 1) By only showing the current level of the algorithm, users lose sight of the broader recursive structure. 2) While the approach simplifies transitions, it abstracts away the recursive call hierarchy

Resolution

While tree diagrams match algorithmic implementation more closely, research on representation effectiveness demonstrates that implementation fidelity does not guarantee learning effectiveness, and our approach prioritises conceptual fidelity over implementation fidelity. Hence, we maintained the current stage as the primary approach but added recursive context indicators to partially address the recursive understanding deficit. This approach better reflects the simultaneous nature of D&C. For example, in merge

sort, all sub-arrays are divided concurrently rather than sequentially, as suggested by code executions. This representation better aligns with the conceptual understanding of D&C as a parallel decomposition strategy while making phase transitions more visually apparent.

3.4.2 Phase-Explicit Design Framework

Early testing did not sufficiently distinguish phases, causing participants to view the algorithms as continuous processes. This represented a fundamental failure of our initial design philosophy, where, without explicit phase recognition, students could not develop the right mental model. To solve this issue, each algorithm visualisation was redesigned for clear phase separation. A design challenge arose in adapting binary search and Strassen's matrix multiplication to the three-phase model.

Binary Search adaptation

Binary search does not traditionally have a combine phase, so we designed one to preserve the uniformity across the visualisations. In this adaptation, the combine phase visually combines discarded portions of the array to show the target element's position relative to the original array.

Strassen's Matrix multiplication adaptation

For Strassen's Matrix multiplication, the common interpretation of the divide phase is simply splitting the matrices into quadrants. However, for consistency and to make the base case structure explicit, we designed the divide phase so that it includes both the quadrant splitting and the application of Strassen's operand formulas. This process repeats recursively until the base case is achieved (matrices are small enough to be multiplied directly). Similarly, the combine phase was defined as the repeated recombination of resulting matrices using Strassen's combination formulas until a final matrix of the original size is formed.

These refinements ensured that all three algorithms clearly demonstrated the three-phase model while still preserving the algorithm-specific operations. This approach also directly implements cognitive load theory's coherence principle, where extraneous differences that do not serve learning goals should be eliminated. The phases in each algorithm are as follows:

Merge-sort phase structure: Divide phase :arrays are split into halves until each element is in its own array group. **Conquer phase :**sorting of sub-arrays when combining two array groups. **Combine phase :**merging 2 array groups while sorting the elements

to form a single sorted array group.

Binary search phase structure: Divide phase :the pivot is selected and the array group is divided into half. **Conquer phase :**the pivot and the elements directly beside the pivot are compared to the target element to decide if the target is found or which side to keep and remove. The discarded side is removed. **Combine phase :**once the target is found, the discarded elements are recombined to show the position of the target relative to the original array.

Strassen's matrix multiplication phase structure: Divide phase :splitting the matrix into quadrants and applying Strassen's operands to the matrix quadrants. **Conquer phase :**computing the resulting matrices from applying the Strassen operands. **Combine phase :**recombination of resulting matrices with combination formulas to form the final result.

A limitation of this approach, however, is that it may create false expectations when students encounter D&C algorithms that do not fit the three-phase model exactly, such as the closest pair of points. Future work should investigate adaptive phase models that maintain core D&C concepts while accommodating algorithmic variation.

3.4.3 Multi-Modal Learning Support

The tool offers a multi-modal learning support whereby for merge sort and binary search, the standard view and explorative view were created, and for all three algorithms, real-world analogical visualisations were created. Each of these views was designed to serve different contexts and objectives, following Universal Design for Learning principles to accommodate diverse learning preferences and cognitive abilities.

Standard view: This view was intended primarily for teacher-led demonstrations or initial student exposure to the algorithm. The view presents the algorithm in its entirety and allows users to observe the "big picture" by observing how sub-problems are created, how they are solved and how their solutions are recombined at their respective recursive level. This view also incorporated the bottom panel and pop-up boxes to give a more in-depth explanation of the processes in the algorithm. In this view, the user is able to step forward or back through the different stages, showing how all sub-problems either get divided, solved or combined concurrently.

Explorative view: This view was designed for self-directed, hands-on learning. This mode gives students the freedom to step into and out of specific branches and experiment

with different scenarios. This trial-and-error interaction encourages active engagement, hypothesis testing and iterative refinement of the mental models. Prior research [26][19] suggests that exploratory, hands-on approaches can improve both understanding and long-term retention of algorithmic concepts.

Analogical view: A challenge in teaching D&C algorithms is helping students grasp the abstract flow and structure. Feedback from a think-aloud session showed that students who had not received a prior explanation of D&C found it more difficult to connect the algorithms to real-world scenarios compared to those who had. To solve this, we developed the introduction section of the tool, which gives real-world analogical visualisations for each algorithm to establish a clear understanding of the D&C structure. An explanation of the implementation of the analogical views is stated in 4.3.5.

Multi-Modal Integration Strategy: Our three-view approach follows progressive abstraction principles:

- **Analogical views:** Concrete, familiar contexts with direct D&C mapping
- **Standard views:** Abstract algorithmic representation with guided progression
- **Explorative views:** Independent manipulation and hypothesis testing

Critical Evaluation of Multi-Modal Approach:

Strengths: Accommodates diverse learning preferences, provides a scaffolded transition from abstract to concrete.Supports both guided and self-directed learning

Weaknesses: Having multiple views may overwhelm new users, particularly if they switch too frequently between modes. Students without prior explanation may struggle to map real-world analogies back to algorithmic concepts, suggesting that analogical views alone are insufficient without teacher or textual guidance.

3.4.4 Representing Recursive Structure

Another challenge in teaching divide and conquer algorithms is representing the recursive nature of it in a way that is easy for students to follow. This project was designed in such a way as to reduce cognitive load but still be a faithful representation of the algorithm's structure and flow. To do this, the recursive structure was represented using a stage-based recursion display and subproblem inspections.

Stage-based recursion display

As stated in 3.4.1, the visualisations use a current stage approach to visualise the recursion, where each stage shown is a new recursive level. To put this into perspective, if

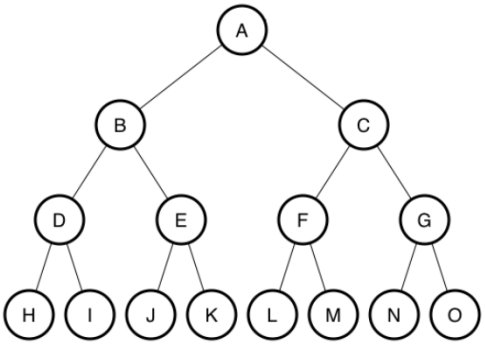

Figure 3.2: Standard Binary Tree

we take the binary tree below 3.2, the first stage would show the values of A and the next would show the values B and C and the following would show the values D, E, F and G.this way the user would get a better idea of the behaviour of every element when progressing through the recursive levels rather than just focusing on one element in one branch at a time.

As for the subproblems, animated transitions are used to convey the actions implemented on them, such as their splitting or recombination. This helps students better understand the behaviour of subproblems and the actions being implemented, as it is visually conveyed rather than being inferred from code.

Subproblem inspection: To inspect the subproblems, the tool allows users to bring up the bottom panel or pop-up boxes (for Strassen's matrix multiplication visualisation) to get an in-depth explanation of what is happening to the subproblems. This lets learners focus on a single subproblem when needed, reducing visual overload.

To support focus, highlighting and labelling are used extensively. For example, in the standard and exploratory modes of binary search, different colours are used to highlight the pivot, the array that is active and the array that has been discarded. Next in Strassen's matrix multiplication, different colours are used to show how a matrix is formed, along with labels to show the formulas used. This provides a clear indication of where the student should be focusing while also providing clarity on how the recursive level and subproblems were formed.

By combining these strategies, the visualisation provides a global view of the recursion while also providing a localised view of the individual subproblems, hence supporting understanding of high-level concepts to detailed execution steps.

3.5 Interface Design and User Experience

To enhance the user experience and to make the tool more beginner-friendly, certain features were added to the visualisation pages.

Bottom explanation panel and pop-ups Initially, in-depth explanations were placed in a side panel. However, feedback from think-aloud sessions showed that the side panel competed visually with the algorithm visualisations and reduced the available horizontal space. The design was revised to use the unused bottom area, allowing a panel to display more detailed explanations of what is occurring at the current stage of execution. For example, it would clarify what happens after a pivot is chosen or what happens when a user divides or combines an array. In the visualisation for Strassen's matrix multiplication, however, pop-up boxes were used instead, as the matrices formed during this visualisation would fill up the entire screen, and we did not want the panels to be competing with the screen for space. Hence, similar to the bottom panel, a pop-up would appear showing how a matrix is formed when it is clicked on.

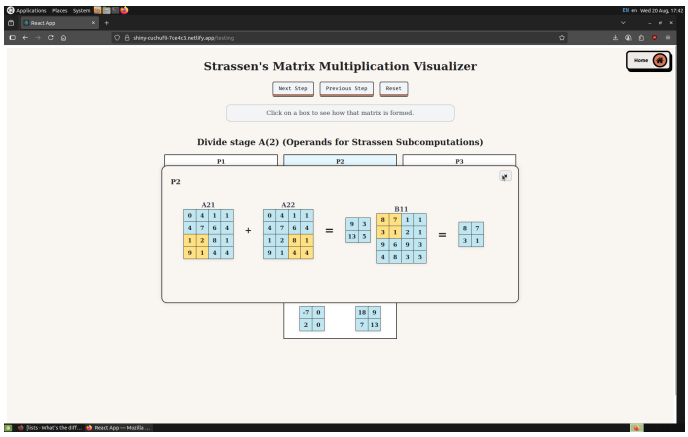

Figure 3.3: Pop-ups in Strassens visualisation

Improved guidance and description Early versions of the tool contained brief and vague textual explanations, which participants reported as insufficient for building understanding. These were rewritten to be descriptive and pedagogically supportive, helping users make sense of the algorithm's logic and purpose at each step.

Onboarding tutorial pop-up To address navigation and feature discovery issues, a pop-up guide was added to appear when a page loads for the first time. This interactive tutorial highlights interface elements, explains what each button does and shows how to access the explorative mode, ensuring that new users can quickly orient themselves without external instructions.

Chapter 4

Implementation

This chapter describes the technical implementation of the tool. Section 4.1 outlines the system's architecture and design decisions, covering architectural requirements and an evaluation of various architectural approaches. Section 4.2 details the methodology and validation framework, while Section 4.3 discusses the algorithm-specific implementations and the implementation of the exploratory and analogical views. Finally, Section 4.4 addresses the technical limitations and outlines directions for future work.

4.1 System Architecture and Design Decisions

4.1.1 Architectural Requirements Analysis

This visualisation tool presented its own unique architectural challenges that are different from general visualisation tools or standard web applications. Based on cognitive load theory and the specific pedagogical goals outlined in Chapter 3.2, we identified five critical architectural requirements:

Real-time Algorithm-Visualisation Synchronisation: The algorithm state must be consistent with the visual display when navigating forwards and backwards without re-computation.

Multi-Modal Consistency: The same algorithms must drive the three different views (standard, exploratory, analogical) without introducing inconsistencies that could impair mental model formation.

Educational State Preservation: Unlike traditional visualisers that only show execution steps, the tool must maintain educational context (learning phases, explanations, user progress) during complex interactions.

Performance Constraints for Learning: The system must remain responsive to maintain user engagement and cognitive flow, while being able to handle arrays up to 25 elements and matrices up to 8×8 for realistic educational scenarios.

4.1.2 Architecture Overview

The system is a browser-based visualisation tool that is implemented using React for user interface (UI) rendering, Framer Motion for animations and JavaScript logic for the algorithm executions. All the processing is client-side, so no server interactions are needed once the page loads, which improves the responsiveness. This architecture ensures fast responsiveness since user interactions and animations are handled locally without network latency.

React was selected over alternatives such as Angular and Vue due to its: **1) Component-based architecture:** The complex visualisations could be broken down into modular and reusable components. **2) Efficient state management:** hooks like `usestate` and `useffect` provide precise control over the UI state to enable smooth synchronisation between the algorithmic logics and visual changes. **3) Efficient rendering:** Reacts virtual DOM, selectively updates only modified components rather than re-rendering entire interfaces, making it suitable for visualisations that require frequent real-time updates. **4) Community and longevity:** React's comprehensive documentation and large developer community provide robust support and ensure the project remains maintainable and accessible for future work.

Framer Motion was chosen for its smooth animation capabilities, which are essential for phase transitions and the exploratory mode. The combination of React, Framer Motion and JavaScript provides a good balance of modularity, performance and educational clarity, which supports the project's core goal of creating an engaging learning tool.

4.1.3 Architectural Approach Evaluation

I evaluated three architectural approaches against the requirements:

Option A: Canvas-Based Monolithic Architecture

Approach: HTML5 canvas with a custom rendering engine and event handling.

Advantages: 1) Performance: Canvas provides direct pixel rendering with hardware acceleration, making it highly efficient for graphics-heavy applications such as games and simulations [9]. **2) Complete Visual Control**: Pixel-level rendering precision enables custom educational visualisations not constrained by HTML/CSS limitations.

Disadvantages: 1)Accessibility: By default, content drawn is invisible to assistive technologies like screen readers. Without additional ARIA (Accessible Rich Internet Applications) descriptions or fallback HTML, users with disabilities are excluded [2]. **2)Maintainability**: Monolithic canvas codebases tend to mix rendering logic and interaction handling, reducing modularity and reusability.
Suitability: Poor. High performance but unsuitable for inclusive educational tools.

Option B: Web Components with Custom State Management
Approach: Native Web Components with manually implemented state synchronisation.
Advantages: 1) Encapsulation: Shadow DOM allows strong component isolation, preventing style and DOM conflicts between components [31]. **2) Accessibility and Compatibility**: Web Components are natively supported in modern browsers without requiring external frameworks.
Disadvantages: 1) State Management Complexity: Unlike React or Vue, Web Components lack built-in state management. Manual implementation of reactive patterns is required. **2) Browser Compatibility**: Inconsistent behavior across browsers may require additional testing and workarounds.
Suitability: Moderate. Standards-compliant and modular, but high implementation risk for research projects requiring rapid iteration.

Option C: React-Based Component Architecture (Selected)
Approach: React application with hooks-based state management.
Advantages: 1) Rapid Iteration: React's Hot Module Replacement (HMR) allows developers to instantly preview UI changes without losing application state, significantly speeding development cycles [6, 28, 5]. **2) Accessibility Support**: React supports ARIA attributes and has strong accessibility tooling built into its ecosystem, improving inclusivity [1, 20]. **3) Component Reuse**: React's component-based architecture improves modularity and maintainability.
Disadvantages: Performance Overhead: Virtual DOM reconciliation increases rendering cost, but it is within acceptable bounds for an educational application [16].
Suitability: Optimal. The development speed advantage enables more design iteration cycles, crucial for educational technology research.

4.1.4 Architectural Design Patterns

Component Hierarchy Strategy

The system follows a component-based architecture optimised for visualisation clarity and reusability: **1) Page component:** Each visualisation and view is implemented as an independent page to make it easier to edit each visualisation. **2) Reusable elements:** UI elements like `Switch.jsx` and `ProgressButton.jsx` are modular and reused across multiple visualisations.

State Management Strategy: React's `useState` hooks were used to manage local state within each visualisation to avoid the complexity of global state solutions like Redux. This approach keeps logic close to the relevant UI, improves readability and prevents unnecessary re-renders.

Rationale: This architecture enables new algorithms and features to be added with minimal disruption, while maintaining a maintainable and readable codebase.

4.1.5 Cross-Cutting Concerns

The implementation addresses several cross-cutting concerns to ensure that the tool is robust, performs well, accessible and broadly compatible:

- **Error handling:** Educational tools require robustness since errors interrupt learning. We implemented a three-layer error handling strategy: 1)**Input Validation Layer:** If there is any error in the input, then a clear message is shown to the user. 2) **Algorithm Execution Layer:** Defensive programming is applied throughout with edge case checking and the usage of try/catch blocks to safeguard the input parsing. 3)**Consistency Layer:** The functions in the code validate user data and check for valid formats, non-empty array inputs and if a target is given for binary search.
- **Performance optimisation :** Performance lag and latency may impact the learning. This was solved by the following : 1)`useState` was used to minimise unnecessary re-renders. 2) Conditional rendering was used to ensure components like the bottom panels and guides only mount when required. 3) The algorithm steps are precomputed and stored to prevent redundant calculations.
- **Browser Compatibility :** To make the tool more accessible to a broader range of users, the following was done: 1) The tool was built on standard React and JavaScript to ensure compatibility with all modern browsers. 2) The build

toolchain compiles the code to ensure consistent behaviour across browsers without depending on proprietary APIs.

These measures ensure that the tool is reliable and performs efficiently despite the browsers used.

4.1.6 Limitations, Trade-offs, and Future Directions

While the current architecture meets the project's goals, several limitations and trade-offs were identified: **1) Limitations:** The system is primarily optimised for desktop use, but not for smartphones or smaller tablets. It also requires internet connectivity, as offline functionality has not yet been implemented. **2) Trade-offs:** React was chosen for its ecosystem and rapid development benefits, at the cost of framework independence and potential migration overheads. Similarly, fine-grained components increased initial complexity but allowed rapid iteration and flexible extension. **3) Future directions:** As the tool expands, a micro-frontend architecture could support independent development of new visualisations. For more computationally intensive algorithms, integrating WebAssembly could improve performance while retaining React's development advantages.

4.2 Methodology and Validation Framework

4.2.1 Methodology

The project followed an iterative,feedback-driven development process where features were prototyped quickly, tested with users and refined based on observations. Development was tracked with Git version control and a repository hosted on GitHub. This ensured traceability and allowed us to access the code from any device, which allowed us to work on it from anywhere.

4.2.2 Testing

Testing followed the Inverted test pyramid [14], which recommends a majority of testing to be focused on end-to-end testing and UI tests rather than unit tests. We conducted our testing as follows: 1) **Unit Tests :** We generated some unit tests to check the algorithm correctness (merge sort step generation, binary search edge cases) and to check the output of the formulas used in Strassen's matrix multiplication.

2) **User acceptance testing :** We conducted multiple think-aloud sessions where participants verbalised their reasoning and thoughts when using the tool. This helped us uncover gaps, strengths and weaknesses in the tool, such as explanation clarity issues that code-level tests could not detect. This layered approach helped us balance the technical reliability and educational effectiveness to ensure the system is both correct and usable.

4.2.3 Validation

Validation focused on ensuring accessibility and broad usability: 1) **Accessibility testing:** Checked compliance with WCAG 2.1 guidelines. [27] 2) **Cross-browser and device testing:** we checked and verified that the tool had consistent behaviour across modern browsers (Chrome, Firefox, Brave, Safari, Edge) and devices, including desktops and tablets.

4.3 Algorithm-Specific Implementations and Challenges

4.3.1 Binary Search (Standard view) Implementation

Implementation overview :
The binary search visualisation A.4 implements a reversible finite state machine style approach, addressing the fundamental challenge of maintaining algorithmic correctness while supporting bidirectional navigation.
The visualisation uses React hooks (`useState`, `useEffect`, `useRef`) paired with two primary step generation functions:

- `buildBinarySearchStepsRandom` - Generates random sorted arrays with recorded search steps
- `buildBinarySearchSteps` - Processes custom user input and generates the corresponding search steps

The UI uses the `getBoxVisuals()` function to dynamically style elements according to their role in the current step (the elements can either be in the active range, in the discarded elements or be a found target element). When an element is clicked, the bottom panel opens up with three sections: Pivot, Sides chosen and Explanation. The panel information is generated by the `getPanelInfoForStep()` function to ensure the explanations stay consistent throughout navigation.

Key Technical Innovation - Reversible Algorithm States: Unlike existing implementations that recompute state for backwards navigation, this approach precomputes all states at the start, enabling seamless transitions in both directions without recomputations. Each step is stored as an object with the array state, index markers, and discovery flags, eliminating the need for recalculation.
Educational Augmentation - The Combine Phase Challenge: Binary search naturally lacks a combine phase. As stated in 3.4.2, this visualisation introduces the recombining phase, where, after the target is found, the discarded elements are progressively shown again to recontextualise the target's position in the original array.
Implementation challenges and solutions: 1) State-UI synchronization - Centralised `getPanelInfoForStep()` function ensures consistent explanations across navigation states **2) Input validation complexity -** Regex-based parsing with educational error messaging handles invalid arrays while maintaining learning flow **3) Elements style management -** Use `getBoxVisuals` function to return the styling rules and overlay patterns to ensure consistency.

4.3.2 Merge Sort (standard view) Visualisation

Implementation overview:
The merge sort visualisation A.5 implements a recursive step-generation approach that records every split and merge operation, ensuring a consistent mapping between algorithm execution and visualisation.

Core Innovation - Level-wise Visualisation: Unlike tree-based or sequential representations, this approach uses the current stage approach and renders all subproblems at each recursive level simultaneously, better reflecting the conceptual nature of D&C as a parallel decomposition strategy. The visualisation has a two-phase approach: 1) **Divide phase** - Progressive array splitting until single-element subarrays. 2) **Combine phase** - Sorted merging of subarrays. In Merge sort, the conquer phase does not have an explicit use, but to maintain consistency with other algorithms, we assume that when the elements are in their own subarrays, they are in the conquer phase.
Technical Architecture - Metadata Tracking System: Each array group receives unique IDs describing its formation type (split, merge or carry) stored in `metaMap`, enabling dynamic reconstruction of both source and result array groups. This supports bidirectional explanation generation, where users can understand how arrays are formed (through splitting) or what they create (through merging).

Educational Design: Bottom panel content dynamically adapts based on metadata, presenting three contextual sections: 1) Original arrays: Shows formation sources 2) Result arrays: Shows operation outcomes 3) Explanation: Provides phase-specific explanations of what happened

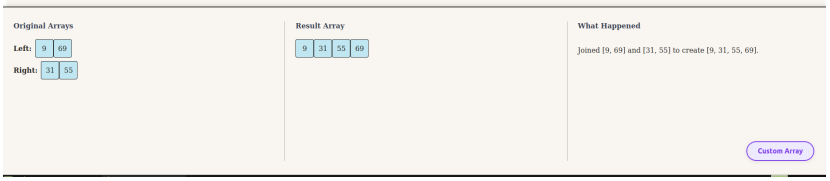

Figure 4.1: Bottom Panel

Animation Strategy: The visualisation uses directional splitting animations, guided by the getHalfDirection() function, to ensure that every motion has pedagogical meaning. Rather than relying on arbitrary transitions, array groups move left (-1) or right (+1) when splitting or merging based on the size of their neighbours. This spatial mapping reinforces the recursive structure of merge sort, helping learners intuitively associate the physical direction of movement with the logical division of the algorithm.
Implementation challenges and solutions: 1) Split/merge representation: A level-wise approach that clearly separates the split and merge phases was used. The metaMap records the formation history of every array group, including type and parent IDs, to enable accurate reconstruction in explanations **2) Array formation clarity:** Formation history tracking was implemented, where the metaMap records operation states, enabling dynamic explanation reconstruction that adapts based on operation type (split, merge, or carry). **4) User input validation:** A Robust parsing system with numeric validation was implemented that resets and rebuilds the visualisation state for valid inputs, while displaying clear error messages for invalid inputs without breaking the application. **3) Phase identification clarity:** Metadata-driven classification system automatically categorises subarrays into ”Dividing”, ”Conquering”, and ”Combining” phases, with an info box display that updates during step navigation to reinforce the D&C framework.

4.3.3 Strassen’s Matrix Multiplication

Implementation overview: Strassen’s algorithm visualisation A.6 presents the greatest technical and pedagogical complexity, requiring a multi-stage architecture to manage recursive matrix operations, formula derivations, and educational clarity simultaneously.

Just like the other visualisations, it is implemented as a React component and it uses a combination of `useState`,`useEffect` and `useRef` hooks to manage the algorithm's stages, pop-ups and matrix data.

Architectural Innovation - Stage-Based Recursive Display: Unlike the single-view approach used for previous algorithms, the Strassen implementation uses dedicated view components (OriginalView, DivideView, BoxView), enabling stage-specific layouts, interactions, and educational content. This addresses the algorithm's inherent complexity while maintaining pedagogical accessibility.

Educational Phase Adaptation: To maintain three-phase consistency while preserving mathematical accuracy, the three-phase model was applied as follows: **1) Divide phase:** Split into quadrants plus operand formula application (maintaining recursion until base cases). **2) Conquer phase:** M1-M7 matrix computations with explicit formula breakdowns. **3) Combine phase:** Result matrix reconstruction using Strassen combination formulas.

Critical Design Decision - Recursion Depth Management and labelling:
The implementation limits recursion depth to prevent cognitive overload while maintaining mathematical correctness. For educational contexts, depth limitation after two divide phases provides sufficient complexity demonstration without overwhelming beginners. To maintain consistency with the three-phase model, this implementation uses explicit stage labelling with paired divide/combine phases (Divide Stage A/B, Combine Stage B/A). This pairing helps learners understand which combine stage resolves each corresponding divide stage, reinforcing the algorithm's recursive symmetry.

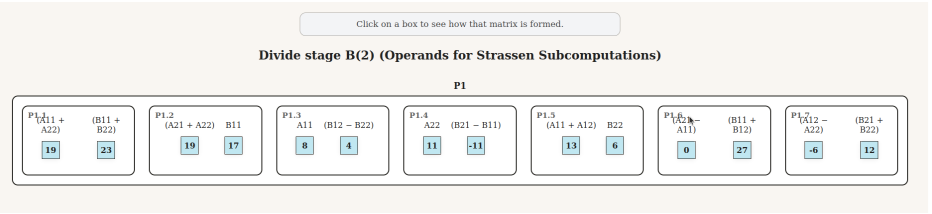

Figure 4.2: Labels for the current stage of the application

Interactive Learning Features:

- Clickable matrices reveal detailed formation breakdowns through pop-up modals
- Formula parsing using regex visualisation breaks down equations (for example C11 = M1 + M4 - M5 + M7) with corresponding quadrant highlighting

- Outside-click detection ensures purposeful popup interaction

Difference in architecture : Unlike the single dynamic view used for binary search and merge sort, this visualisation uses a stage-based architecture with dedicated components for each phase. This is to allow for multiple recursive operations, formula derivation and simultaneous matrix views while enabling stage-specific interactions like pop-ups and quadrant highlighting, which will not be applicable in the single dynamic view. This approach enabled a progressive disclosure essential for complex algorithms. Even though the implementation overhead significantly exceeds simpler visualisations, this trade-off prioritises educational accessibility over development efficiency.
Implementation challenges and solutions: 1) Deep recursive state management - Pre-computed matrix results with stage-based presentation prevent computational lag while maintaining educational clarity. **2) Complex stage coordination -** Encapsulated stage components with state-driven conditional rendering ensure synchronisation across multiple views. **3) Formula visualisation -** Regex-based parsing maps abstract formulas to interactive visual breakdowns, enabling exploration of mathematical relationships.

4.3.4 Explorative Views

Implementation Overview: The explorative viewsA.7A.8, implement student-driven algorithm manipulation, addressing the constructivist learning requirement for hands-on experimentation. Unlike passive observation modes, these views enable hypothesis testing through direct subproblem manipulation while maintaining algorithmic integrity.
Core Innovation - Selective Subproblem Management: In merge sort, the primary technical challenge was enabling users to manipulate individual subproblems without disrupting the broader algorithm state. This required a granular state management system where each subproblem maintains independent lifecycle control.
Merge Sort Explorative Architecture: Array groups are implemented as objects containing values and formation metadata, enabling bidirectional explanation generation. The system uses immutable state updates with unique ID assignment per group, preventing cross-contamination during selective operations.
Key Technical Innovation - Neighbour Detection System: To prevent invalid merging operations, the system implements size-compatible group detection. Users can only merge groups of equal or progressively smaller sizes, ensuring that groups being merged have reached the base case.
Binary Search Explorative Design: Implements a game-like progressive revelation

mechanism where array elements remain hidden until pivot selection. This simulates the information-hiding aspect of binary search while maintaining visual feedback through dynamic bound updates and colour-coded element states. Initially, all elements were shown, but think-aloud testing revealed that participants could immediately locate the target, undermining the learning experience. The design was therefore revised so that only the chosen pivot and its neighbours are revealed. This change aligns the interface with the algorithm's conceptual model by restricting visible information, mirroring the computer's perspective during binary search and promoting more authentic user decision-making.

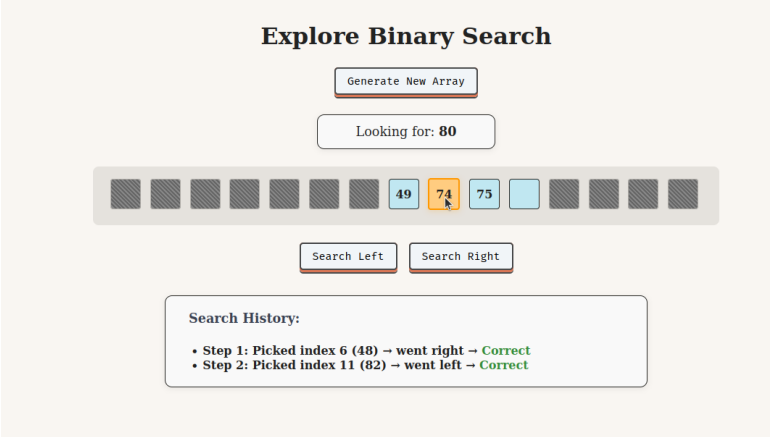

Figure 4.3: Explorative view for Binary Search

Educational Design Pattern: Both explorative views balance algorithmic correctness with learning flexibility through constrained freedom. In these views, users can deviate from optimal paths but cannot violate fundamental algorithmic principles. This operationalises constructivist learning theory within structured boundaries.

Implementation Challenges and Solutions:

1) State Synchronisation Complexity: Immutable state architecture with per-group lifecycle management and dependency isolation was implemented to manage nested state updates across multiple independent subproblems while preserving formation history. **2) Algorithmic Integrity vs. Exploration Freedom:** A constraint-based interaction system was implemented where merge sort allows flexible splitting/merging within size constraints, while binary search reveals information progressively while maintaining search space validity. **3) Progress Tracking and Recovery:** Undo functionality with state snapshots and validation feedback for invalid operations was implemented to support exploratory mistakes without losing learning context.

4.3.5 Analogical Views

Implementation Overview: The analogical views implemented concrete-to-abstract learning progression through real-world mappings. Each algorithm has an analogical representation designed to establish a basic understanding.

Core Innovation - Phase-Explicit Interaction Design:

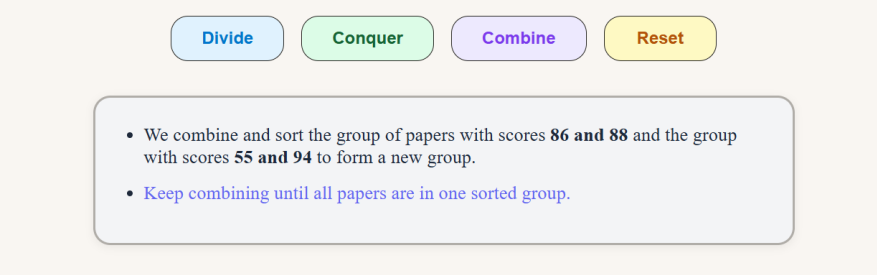

Figure 4.4: Phase labelled and informative bottom panel

Unlike traditional "Next/Previous" navigation, the analogical views use phase-labelled controls (Divide/Conquer/Combine) to establish direct cognitive mapping between concrete operations and algorithmic phases. This implements recognition-over-recall HCI principles while building phase identification skills.

Algorithmic Mapping Strategy: Each analogy maintains structural fidelity to algorithmic operations: **1) Binary Search - Student ID Lookup A.2:** Cards progressively black out and remove to simulate array halving. State machine synchronises card visibility with search bounds, while animations highlight pivot selection and elimination phases.

2)Merge Sort - Exam Paper Sorting A.1: Papers undergo splitting and recombination with score-based sorting visualisation. Sequential animations coordinate paper movement timing to prevent visual conflicts while maintaining educational flow.

3) Strassen's Matrix Multiplication - Canvas Painting A.3: Quadrant-based canvas division with sequential painting operations. Multiple image sets provide smooth visual transitions while representing recursive decomposition and recombination. **Technical Architecture - Synchronised State Management:** Phase-based state machines ensure visual elements, explanations, and user actions remain algorithmically consistent. Timing chains prevent overlap conflicts while maintaining educational pacing.

Implementation Challenges and Solutions: 1) Animation Timing Coordination: Timeout chain management with staged rendering was implemented where elements update only after animation completion to prevent visual overlaps during sequential operations (card removal, paper movement, canvas painting), ensuring smooth visual

flow. **2) Analogy-Algorithm Alignment:** Centralised state machines with phase-based validation were implemented to maintain synchronisation between concrete visual representations and abstract algorithmic states, ensuring visual elements, explanations, and algorithmic progress remain consistent throughout navigation. **3) Cognitive Load Management:** Progressive visual revelation with contextual explanation panels was implemented to balance analogical detail with educational clarity, breaking complex operations into digestible visual steps with immediate explanatory feedback.

4.4 Technical Limitations and Future Work

4.4.1 Current system limitations

Performance and Scalability Constraints: The current implementation limits the input size to 25 elements for arrays and 8x8 matrices due to worries of performance lag. Further optimisation and performance analysis would be required to extend the scalability for computationally intensive algorithms and larger inputs.

Educational design limitations: The three-phase model, while effective for the implemented algorithms, may not be applicable to other algorithms, such as closest pair of points that do not naturally conform to this pattern. This limitation could mislead users when encountering other D&C algorithms.

Technical architecture constraints: The tool requires a modern browser support and lack offline functionality, limiting accessibility to users with unreliable internet. Furthermore the tool is not optimised for mobile phone usage and the matrix visualisations are difficult to see on a smaller screen.

4.4.2 Technical enhancements:

Immediate extensions: WebAssembly could be integrated to solve the performance constraints while maintaining React's educational advantage. Responsive design improvements and Progressive Web App (PWA) implementation would enhance mobile learning and offline accessibility. Other algorithms such as quicksort, closest pair and karatsuba multiplication could be added to broaden the tool's educational applicability

Strategic development: An adaptive phase model that accommodates variation in algorithms while maintaining the core D&C concepts would solve the three-phase model limitations. This would enable broader algorithm coverage while preserving the educational benefits of explicit phase identification and structured learning progression.

Chapter 5

User Evaluations

This chapter presents the evaluation of the visualisation tool. Section 5.1 discusses the rationale and methodology behind the think-aloud sessions, which were used as the primary qualitative evaluation method. Section 5.2 describes the design of the survey, which employed a pre-post design to assess learning effectiveness and usability. Section 5.3 presents the quantitative insights from the study, and Section 5.4 presents the qualitative insights. Finally, Section 5.5 concludes with an analysis of the System Usability Scale (SUS) scores.

5.1 Think Aloud Sessions

Rationale and Methodology:
Think aloud sessions were chosen as the primary qualitative evaluation method for this project as it provides a direct, unfiltered window into the user's cognitive processes and understanding. Unlike interviews, think-aloud sessions allow for real-time observation of a user's mental model as they interact with the tool. This method was crucial for understanding not just when users were confused but why confusion occurred and what their expectations were. This method directly supported the project's iterative design methodology, enabling rapid cycles of prototyping, feedback and refinement. Feedback from each session directly informed the next design iteration, ensuring the tool's evolution was grounded in genuine user needs and behaviour rather than assumptions about educational effectiveness.

The specific aspects we aimed to evaluate were:

- **Phase separation effectiveness:** Whether explicit divide-conquer-combine labeling improved users' understanding of D&C strategy compared to traditional

sequential representations.

- **User experience and discoverability:** Navigation and controls clarity, and the effectiveness of on-screen feedback systems.
- **Cognitive load management:** The impact of visual recursion representation and information panel placement on working memory and comprehension.

Session Design and Participants

The user evaluation involved multiple think-aloud sessions rather than a single large study. This allowed for continuous and low-overhead feedback integration. Participants chosen were university students across undergraduate and postgraduate levels, intentionally excluding computer science backgrounds to better represent the target audience of users with minimal algorithmic knowledge. They were recruited via student WhatsApp groups. Each session involved 2-4 participants. To evaluate the necessity of scaffolding, participants were randomly divided into two conditions: half received brief introductions to D&C concepts before interaction, while the control group encountered the tool without prior explanation. This comparative approach informed decisions about the level of guidance and hand-holding required within the tool itself.

Key Findings and Design Iterations

The think-aloud sessions revealed critical usability and pedagogical gaps, leading to design evolutions across four major iterations:

- **Session 1:** The initial prototype failed to differentiate the algorithmic phases, with participants perceiving the algorithms as continuous processes rather than structured D&C operations. The breakdown in conceptual understanding lead to the development of the explicit three-phase model.
- **Session 2:** Participants struggled to form mental models from abstract visualisations alone and participants who received preliminary D&C explanations demonstrated significantly better comprehension, revealing the need for concrete conceptual foundations. This led to the development of analogical views.
- **Session 3:** The phase separation was not explicit enough in the analogical views, while the binary search explorative view undermined its educational purpose by revealing target elements immediately. These observations led to an information panel in the analogical view that explicitly connected the phases to the examples and the progressive revelation and initial hiding of element values in binary search.

- **Session 4:** The buttons' functionality was not implicit enough, and they were confused as to which button to click. This led to interface improvements including contextual popup guides on page load, enhanced explanations across all views and dynamic bottom panels providing step-specific guidance and next-action recommendations in analogical views.

5.2 Survey Design

The survey uses a pre-post design (Questions used are in the appendix) with mixed data collection to evaluate both the learning effectiveness and usability of the tool. This approached allowed us to observe the change in the participants understanding thereby assessing the extent to which our goals 3.2 were met while identifying areas for improvement.

Structure and rationale: The survey began by collecting essential background information to identify trends based on prior knowledge. This section captured the participants' educational background, if the participants had heard of "algorithms" and "divide-and-conquer", whether they had previously taken a course about algorithms and their confidence levels with D&C algorithms.

This demographic allowed us to categorise responses based on experience levels and evaluate whether the tool supported both beginner and advanced users, in line with our inclusivity goals.

Pre-post knowledge assessment: Participants completed identical tests before and after using the tool, which included questions about: 1)textbfConceptual recognition: Multiple-select questions identifying correct D&C characteristics. 2)**Algorithm familiarity:** Recognitions of specific algorithms. 3) **Open-ended explanations:** Free-response questions about D&C concepts and phase differentiation. 4) **Specific algorithm knowledge:** Targeted questions about the algorithms visualised. **Usability measurement:** Participants completed the standardised System Usability Scale (SUS) to provide industry benchmark usability metrics and enable comparison with other educational tools.

Methodological triangulation The survey design used methodological triangulation by combining pre-post knowledge assessment with SUS. This approach addresses the distinction between learning effectiveness and usability as interdependent but separate constructs [12][18]. In doing so, it ensured that improvements in understanding were not achieved at the expense of usability, hence validating our educational and usability

goals. **Relation to Research Question** The evaluations were deliberately crafted to directly address the central challenges mentioned in 2.2. To assess the impact of phase-explicit visualisation on comprehension, the study used a pre- and post-knowledge assessment with questions specifically targeting understanding of the ”divide, conquer, and combine” phases. The role of real-world analogies was explored through qualitative analysis from think-aloud sessions and open-ended responses. Lastly, the impact of interactive exploration on engagement was measured using the SUS. This approach ensured that the evaluation's design directly contributed to answering the research questions and provided a solid, evidence-based conclusion

5.3 Quantitative Insights

Nine participants completed the study, and results showed measurable improvements in understanding of divide-and-conquer algorithms. The strongest gains were in merge sort recursion (correct responses rose from 4 to 8) and base case recognition (3 to 7), directly supporting the effectiveness of the phase-explicit visualisation approach. However, Strassen's algorithm remained challenging for some participants, with only small improvements despite the visualisation. Similarly, the binary search explorative view caused some confusion when elements were hidden, with some participants stating in feedback that it initially felt confusing until explained. These issues highlight that while the tool was effective in reinforcing recursive structure, certain design choices require refinement to avoid introducing new barriers to understanding. While the small sample size limits statistical generalisation, these improvements provide evidence that phase-explicit and exploratory visualisations can strengthen learners' conceptual grasp of recursive structure and problem decomposition.

5.4 Qualitative Insights

The open-ended responses showed a clear improvement in conceptual articulation and use of technical vocabulary. Pre-survey answers were vague and uncertain, whereas post-survey answers showed structured thinking and accurate use of terms. For instance, one participant shifted from describing D&C as “splitting into smaller chunks” to “breaking a problem into smaller chunks to solve, then combining results to solve the original problem.” This evolution shows how the tool supported learners in forming more holistic mental models, particularly of the three recursive phases.

Question	Pre	Post	Increase
Divide-and-Conquer Algorithm Statements	6	9	50%
What is a major conceptual advantage of divide-and-conquer algorithms?	7	7	0%
In the binary search algorithm, what is the role of the pivot?	6	7	16.7%
Which of the following best describes the structure of merge sort's recursion?	4	8	100%
Why is it important to have a clear base case when designing a divide-and-conquer algorithm?	3	7	133.3%
How does Strassen's algorithm apply the "divide" step of divide-and-conquer?	4	6	50%

Table 5.1: Pre- and Post-Questionnaire Results.

5.5 System Usability Scale (SUS) Analysis

The tool achieved an average SUS score of 85.0, well above the industry benchmark of 68, indicating excellent usability. Individual scores ranged from 62.5 to 100, showing consistently positive user experiences. The highest-rated item was "Most people learning divide-and-conquer algorithms would benefit from this tool" (mean = 4.67/5), highlighting strong perceived educational value. The weak correlation between SUS scores and learning gains (r = 0.21) suggests that while the tool is highly usable, its usability does not have a direct, one-to-one relationship with how much the users learned. This is a finding consistent with prior research on educational software usability[25]. The full results from the study is included in a file along with this report.Based on the results, the methodological triangulation was effective. The qualitative analysis, quantitative analysis and SUS scores showed excellent results as discussed,but the weak correlation between usability and learning gains highlighted that a user-friendly interface alone does not guarantee learning. This approach validated both the tool's educational effectiveness and its usability, confirming that the study's goals were met.

Chapter 6

Conclusions

This dissertation set out to address a challenge in computer science education: helping students develop an understanding of D&C algorithms and better understand the bigger picture of it. Existing visualisation tools tend to emphasise execution traces and code-level details, which although is helpful, frequently obscures the higher-level strategy of dividing, conquering and combining. Hence the main aim of this work was to design, implement and evaluate an educational tool that makes the phases of D&C algorithms more explicit, while also ensuring usability and learner engagement.

The tool developed in this project introduced a phase-explicit and exploratory visualisation approach along with an analogical and standard visualisation, implemented as a web-based application. Key design choices included the use of a finite state machine style to enable reversible navigation, directional animations to convey problem subdivision, and multiple interaction modes.These design features were grounded in established learning theories such as CLT and constructivism, as well as learning design principles and HCI principles like recognition over recall and progressive disclosure. These decisions were made to reduce cognitive load, support active engagement, and scaffold learners toward building robust mental models of recursion.

The evaluation combined pre- and post-knowledge assessments with the system usability scale, adopting a methodologically triangulated approach to assess both learning effectiveness and usability. Despite the small sample size, the results showed concrete evidence of the tool's educational value. Quantitative findings showed substantial gains in participants' understanding of merge sort recursion and base case recognition, with correct responses nearly doubling in both areas. Recognition of general D&C characteristics also improved, suggesting that the phase-explicit design successfully reinforced conceptual understanding.

Qualitative feedback showed that the participants' free-response explanations shifted from vague and fragmented descriptions to more structured and technically accurate explanations. This shows a deepening in conceptual grasp. The SUS results validated the results as it scored an average of 85, which is well above the industry benchmark of 68, placing the tool in the "excellent" category. At the same time, the evaluation revealed some limitations. The small participant pool limits the generalisability of findings and the tool's coverage was restricted to 3 algorithms. While the usability scores were high, some participants expressed confusion when interacting with more complex algorithms. These limitations show that while the approach is promising, it is not a complete solution to the challenges of teaching recursion.

This work makes three key contributions. First, it demonstrates that a phase-explicit, multi-modal visualisation approach can improve contextual understanding of D&C algorithms. Second, it shows that this can be achieved without compromising usability. Third, it provides empirical evidence that learner background and algorithm complexity moderate the effectiveness of visualisation tools.

Some potential avenues for future work are:

- A larger-scale study with a more diverse participant group would strengthen the basis of the findings.
- Expanding the tool to cover more D&C algorithms and integrating richer explanatory scaffolding, such as adaptive hints and quizzes, could enhance the learning outcomes.
- Improvements in scalability and performance could allow for more complex recursive structures.

We are pleased with how the tool turned out in the end, particularly the interactive animations and graphics that make it more enticing for students. Considerable effort went into making the interface engaging as well as pedagogically sound and we are proud of the explorative views. These began as small experimental ideas but developed into one of the most distinctive features of the tool. Another major achievement was implementing Strassen's matrix multiplication. At first I struggled to understand the algorithm, yet was able to design and visualise it successfully which was the most challenging part of the project.

At the same time, there are aspects we would improve. We were not able to implement a fully recursive version of Strassen's algorithm that worked down to 1x1

matrices for all input sizes, as the complexity was too high. We would start participant recruitment earlier to achieve a larger sample size, strengthening the evaluation. Finally, we would run a comparative study against existing visualisation tools to more clearly assess the added value of the tool.

In conclusion, this dissertation has shown that making structural phases explicit through interactive visualisation can significantly improve the learners' understanding while maintaining excellent usability. While challenges remain in scaling and extending the approach, the results provide both theoretical and practical support for phase-explicit educational tools as an effective strategy for learning recursion.This work provides a foundation for future research on how algorithm education, learning sciences, and HCI can come together to improve the teaching of D&C algorithms.

Bibliography

[1] Adobe contributors. Accessibility, 2025.

[2] Paniz Alipour Aghdam and Reza Ravanmehr. A novel approach for canvas accessibility problem in html5, 2014.

[3] Michal Armoni and Judith Gal-Ezer. High-school computer science–its effect on the choice of higher education. *Informatics in Education*, 22(2):183–206, 2023.

[4] A. Faye Borthick, Donald R. Jones, and Sara Wakai. Designing learning experiences within learners' zones of proximal development (zpds): Enabling collaborative learning on-site and online. *Journal of Information Systems*, 17(1):107–134, 03 2003.

[5] Mallik Cheripally. What is react fast refresh?, 2020.

[6] DEV Community. Welcome!, 2025.

[7] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2022.

[8] Holger Danielsiek, Wolfgang Paul, and Jan Vahrenhold. Detecting and understanding students' misconceptions related to algorithms and data structures. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, page 21–26, New York, NY, USA, 2012. Association for Computing Machinery.

[9] Marcos Gonçalves. Understanding canvas api: A comprehensive guide, 2025.

[10] Sherif Hamouda, Stephen H Edwards, Hoda G Elmongui, Jack V Ernst, and Clifford A Shaffer. A basic recursion concept inventory. *Computer Science Education*, 27(2):121–148, 2017.

[11] Christopher D Hundhausen, Sarah A Douglas, and John T Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, 2002.

[12] Interaction Design Foundation - IxDF. What is triangulation in user research?, 2025.

[13] Panagiotis Kampylis, Valentina Dagienė, Stefania Bocconi, Augusto Chioccariello, Katja Engelhardt, Gabrielė Stupurienė, Vaida Masiulionytė-Dagienė, Eglė Jasutė, Chiara Malagoli, Milena Horvath, and Jeffrey Earp. Integrating computational thinking into primary and lower secondary education: A systematic review. *Educational Technology & Society*, 26(2):99–117, 2023.

[14] Ritesh Kapoor. Testing automation, what are pyramids and diamonds? *Medium*, 2022.

[15] Saul McLeod. Vygotsky's zone of proximal development, 2024.

[16] MDN contributors. Performance fundamentals, 2025.

[17] Jakob Nielsen. Ten usability heuristics, 1995.

[18] NN/g. Triangulation: Better research results using multiple ux methods, 2025.

[19] Jean Piaget. *The Development of Thought: Equilibration of Cognitive Structures*. Viking, 1977.

[20] React contributors. Accessibility, 2025.

[21] Raja Sooriamurthi. Problems in comprehending recursion and suggested solutions. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '01, page 25–28, New York, NY, USA, 2001. Association for Computing Machinery.

[22] Juha Sorva. Notional machines and introductory programming education. *ACM Transactions on Computing Education*, 13:8:1–8:31, 06 2013.

[23] John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285, 1988.

[24] Jeroen JG van Merriënboer and John Sweller. Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review*, 17(2):147–177, 2005.

[25] Prokopia Vlachogianni and Nikolaos Tselios. The relationship between perceived usability, personality traits and learning gain in an e-learning context. *International Journal of Information and Learning Technology*, 39(1):70–81, 01 2022.

[26] L. S. VYGOTSKY. *Mind in Society: Development of Higher Psychological Processes*. Harvard University Press, 1978.

[27] W3C. Web content accessibility guidelines (wcag) 2.1, 2025.

[28] webpack. Hot module replacement, 2025.

[29] Susan Wiedenbeck. Learning recursion as a concept and as a programming technique. In *Proceedings of the Nineteenth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '88, page 275–278, New York, NY, USA, 1988. Association for Computing Machinery.

[30] Susan Wiedenbeck. Learning recursion as a concept and as a programming technique. *SIGCSE Bull.*, 20(1):275–278, February 1988.

[31] Wikipedia contributors. Web content accessibility guidelines, 2025.

Appendix A

First appendix

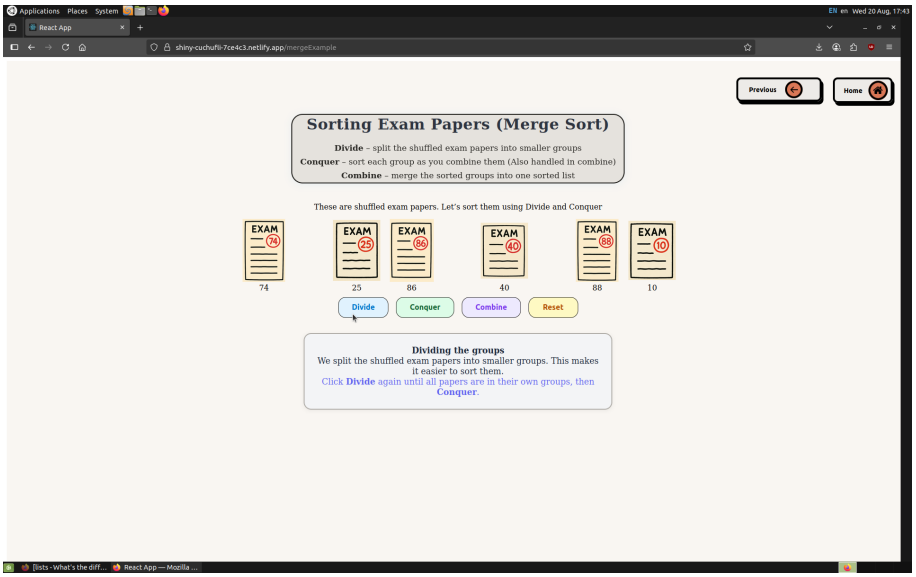

Figure A.1: Analogical View for Merge Sort

Visualising Divide-and-Conquer Algorithms

Thank you for participating in this study.The goal of this research is to evaluate how effective an interactive visualisation tool is in helping users understand the **concepts of divide-and-conquer algorithms**. **No prior algorithm knowledge is necessary.**
You will be asked to:

1. **Complete a short pre-questionnaire** to assess your current understanding of divide-and-conquer algorithms.
2. **Use the tool** to explore and interact with visualisations of these algorithms.
3. **Retake the same questionnaire** to measure any improvement in your understanding.
4. **Provide feedback using the System Usability Scale (SUS)** to help assess how easy and useful the tool is to use.

Your responses will remain anonymous and will only be used to evaluate the educational effectiveness and usability of the tool.This study has received ethics approval from the Informatics Research Ethics Process (RT #7045). Please review the Participant Information Sheet at the link below for details about the study and how your data will be handled:
https://drive.google.com/file/d/1EMLVW1nBQHf6WV-M6ot1iH4MCzSBQmL6/view?usp=sharing

We also ask that you read through the Participant Consent Form before continuing (**you do not have to fill and submit it**):
https://drive.google.com/file/d/1xacf9vjmH2Pl02Xwrj9rzVcb-h6BXkQp/view?usp=sharing

Please use a laptop or pc to complete this questionaire

* Indicates required question

1. How old are you *

2. What is your highest level of education? *

Mark only one oval.

- ◯ High school
- ◯ Undergraduate
- ◯ Masters
- ◯ PhD
- ◯ Other: _____

3. I have heard the term “algorithm” before *

Mark only one oval.

- ◯ Yes
- ◯ No

4. I have taken a course that covers algorithms and data structures: *

Mark only one oval.

- ◯ Yes
- ◯ No

5. I have heard the term “divide and conquer” in relation to algorithms *

Mark only one oval.

- ◯ Yes
- ◯ No

6. If you selected **Yes** for the question above, please give details (optional):
(e.g., examples you know, when or where you encountered it)

_____

7. How confident do you feel about your current understanding of divide-and-conquer algorithms? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very confident

8. Please tick all the statements below that you think are correct.

Divide-and-conquer algorithms usually... (tick all that apply)

Tick all that apply.

- [ ] Split a problem into smaller subproblems
- [ ] Can be parallelised in some cases
- [ ] Use a for-loop and a while-loop together
- [ ] Combine solutions of subproblems
- [ ] Always work faster than brute force
- [ ] Use recursion or iterative methods
- [ ] Sort elements one by one
- [ ] Solve a problem by solving only one subproblem
- [ ] Use the same method at each level of the problem
- [ ] Depend on whether the input size is even or odd
- [ ] Solve problems from bottom-up
- [ ] none of the above

9. Which of these algorithm names sound familiar to you? (tick all that apply)

Tick all that apply.

- [ ] Merge Sort
- [ ] Quick Sort
- [ ] Binary Search
- [ ] Karatsuba Multiplication
- [ ] Strassen's Matrix Multiplication
- [ ] Dijkstra's Algorithm
- [ ] Bubble Sort
- [ ] Dynamic Programming
- [ ] I have not heard of any of these

10. In one or two sentences, what do you think is the main idea of divide and conquer algorithms? *
(If you're unsure, give it your best guess.this is not a test.)

_____

11. *Can you briefly explain the difference between "divide", "conquer", and "combine" in your own words?*

(If you're unsure, give it your best guess.this is not a test.)

_____

12. What is a major conceptual advantage of divide-and-conquer algorithms compared to iterative approaches *

Mark only one oval.

- ◯ They use less memory
- ◯ They are always faster
- ◯ They break the problems into smaller problems which can be solved in parallel
- ◯ They are easier to code
- ◯ I am not sure

13. In the binary search algorithm, what is the role of the pivot? *

Mark only one oval.

- ◯ The pivot stores all the elements that are smaller than the target value.
- ◯ The pivot is used to randomly shuffle the array before searching.
- ◯ The pivot permanently removes half of the elements from memory to speed up the search.
- ◯ The pivot is the middle element used to decide whether to continue searching in the left or right half
- ◯ I am not sure

14. Which of the following best describes the structure of merge sort's recursion? *

Mark only one oval.

- ◯ Merge sort divides the list using a pivot element and sorts the partitions
- ◯ Merge sort merges two unsorted lists into a sorted one
- ◯ Merge sort divides the list into halves, recursively sorts them, then merges the sorted halves
- ◯ Merge sort swaps adjacent elements until the list is sorted
- ◯ I am not sure

15. Why is it important to have a clear base case when designing a divide-and-conquer algorithm? *

Mark only one oval.

- The base case stops recursion and prevents infinite calls.
- The base case stores all intermediate results permanently in memory.
- To ensure data structures are not mutated
- I am not sure

16. How does Strassen's algorithm apply the "divide" step of divide-and-conquer? *

Mark only one oval.

- By dividing matrices into rows and columns instead of blocks
- By splitting each matrix into four smaller submatrices (quadrants) and applying operands for subcomputations
- By converting matrices to one-dimensional arrays
- By factoring the matrix into diagonal and upper triangular components
- I am not sure

17. What do you think happens after the sub-problems are solved?(Write your understanding of how the overall solution is built.)

Try the tool

Please use the link below to access the tool. Explore each section, including the **Explorative Mode** (in binary search and merge sort)**,** to gain a deeper understanding of divide-and-conquer algorithms. You can experiment by entering your own arrays or simply use the randomly generated values provided by the tool.

Take your time to interact with the features before moving on to the next part of the study.
link to the tool:
shiny-cuchufli-7ce4c3.netlify.app

Post-Questionnaire

Now that you have explored and used the tool, please complete this questionnaire. Your responses will help us evaluate how your understanding of divide-and-conquer algorithms may have improved after using the tool.

Please answer all questions as accurately and honestly as possible.

18. After using the tool, how confident do you feel about your current understanding of divide-and-conquer algorithms? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very confident

19. Please tick all the statements below that you think are correct. *

Divide-and-conquer algorithms usually... (tick all that apply)

Tick all that apply.

- [ ] Split a problem into smaller subproblems
- [ ] Can be parallelised in some cases
- [ ] Use a for-loop and a while-loop together
- [ ] Combine solutions of subproblems
- [ ] Always work faster than brute force
- [ ] Use recursion or iterative methods
- [ ] Sort elements one by one
- [ ] Solve a problem by solving only one subproblem
- [ ] Use the same method at each level of the problem
- [ ] Depend on whether the input size is even or odd
- [ ] Solve problems from bottom-up

20. Which of these algorithm names sound familiar to you? (tick all that apply)

Tick all that apply.

- [ ] Merge Sort
- [ ] Quick Sort
- [ ] Binary Search
- [ ] Karatsuba Multiplication
- [ ] Strassen's Matrix Multiplication
- [ ] Dijkstra's Algorithm
- [ ] Bubble Sort
- [ ] Dynamic Programming

21. In one or two sentences, what do you think is the main idea of divide and conquer algorithms? *

22. *Can you briefly explain the difference between "divide", "conquer", and "combine" in your own words?*

(optional)

23. What is a major conceptual advantage of divide-and-conquer algorithms compared to iterative approaches *

Mark only one oval.

- ◯ They use less memory
- ◯ They are always faster
- ◯ They break the problems into smaller problems which can be solved in parallel
- ◯ They are easier to code

24. In the binary search algorithm, what is the role of the pivot? *

Mark only one oval.

- ◯ The pivot stores all the elements that are smaller than the target value.
- ◯ The pivot is used to randomly shuffle the array before searching.
- ◯ The pivot permanently removes half of the elements from memory to speed up the search.
- ◯ The pivot is the middle element used to decide whether to continue searching in the left or right half

25. Which of the following best describes the structure of merge sort's recursion? *

Mark only one oval.

- Merge sort divides the list using a pivot element and sorts the partitions
- Merge sort merges two unsorted lists into a sorted one
- Merge sort divides the list into halves, recursively sorts them, then merges the sorted halves
- Merge sort swaps adjacent elements until the list is sorted

26. Why is it important to have a clear base case when designing a divide-and-conquer algorithm? *

Mark only one oval.

- The base case stops recursion and prevents infinite calls.
- The base case stores all intermediate results permanently in memory.
- To ensure data structures are not mutated
- The base case speeds up the algorithm by skipping all recursive calls.

27. How does Strassen’s algorithm apply the “divide” step of divide-and-conquer *

Mark only one oval.

- By dividing matrices into rows and columns instead of blocks
- By splitting each matrix into four smaller submatrices (quadrants) and applying operands for subcomputations
- By converting matrices to one-dimensional arrays
- By factoring the matrix into diagonal and upper triangular components

28. What do you think happens after the sub-problems are solved?(Write your understanding of how the overall solution is built.) *

System Usability Scale Quiz

The following questions are part of the **System Usability Scale (SUS)**, a standard method used to evaluate how easy a system or tool is to use. Your feedback will help assess the usability of this divide-and-conquer learning tool and identify areas for improvement.

Please rate each statement based on your experience using the tool, from **1 (Strongly Disagree)** to **5 (Strongly Agree)**.

29. I would use this tool frequently to learn or revise divide-and-conquer algorithms. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

30. I found the tool unnecessarily complex. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

31. I thought the tool was easy to use and navigate. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

32. I found the tool's visualisations and features well integrated. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

33. I thought there were many inconsistencies in the tool (e.g., confusing interactions or unclear feedback). *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

34. I think most people learning divide-and-conquer algorithms would benefit from this tool. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

35. I found the tool very cumbersome to use. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

36. I felt confident using the tool without external help. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

37. I needed to learn a lot before I could use the tool effectively. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

38. I would recommend this tool to other students learning divide-and-conquer algorithms. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

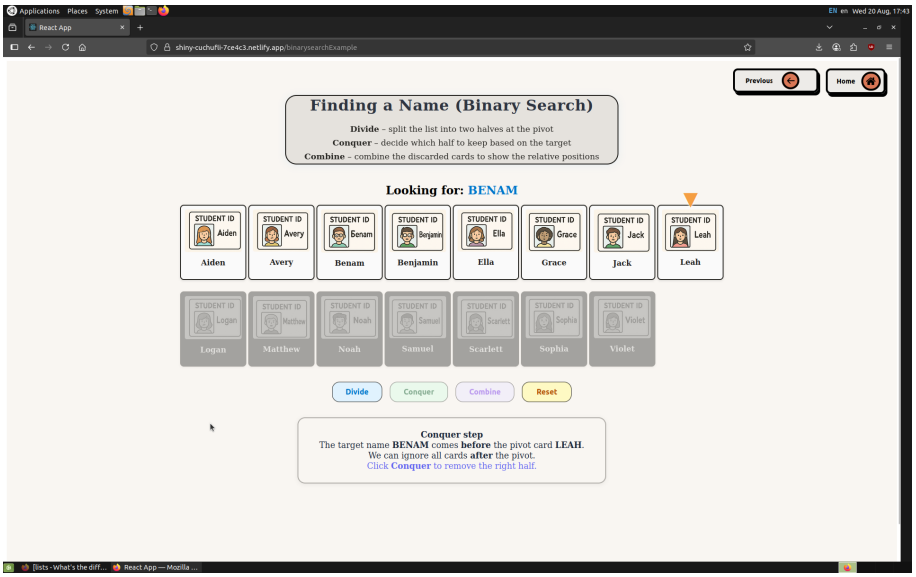

Figure A.2: Analogical View for Binary Search

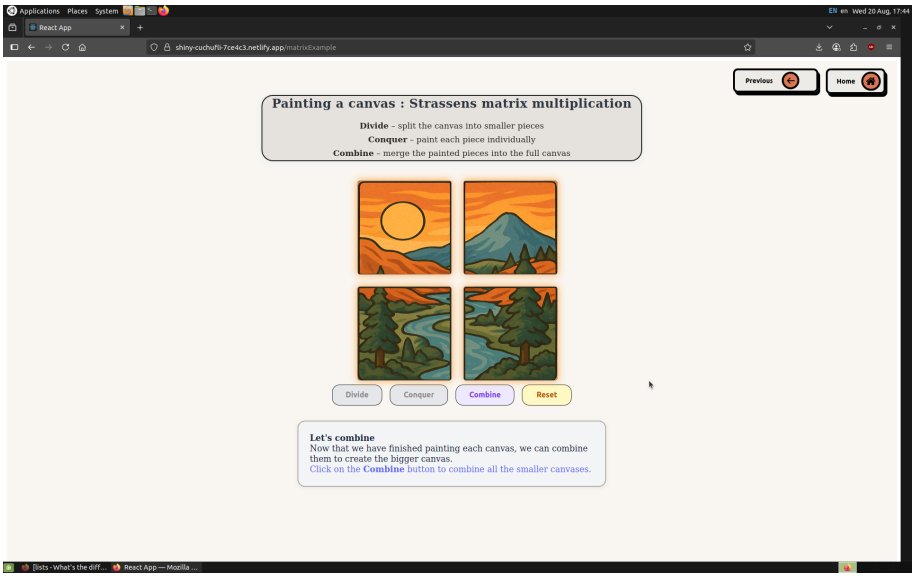

Figure A.3: Analogical View for Strassens matrix multiplication

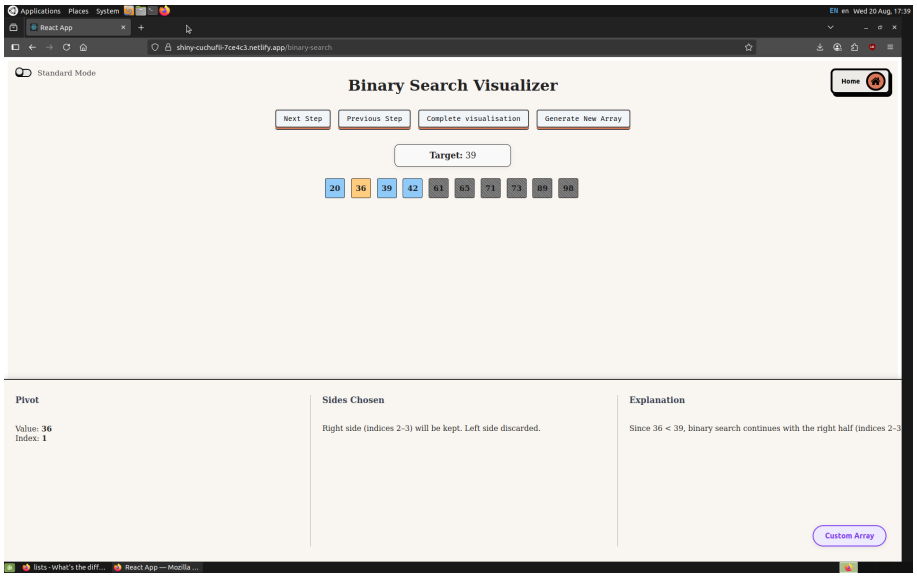

Figure A.4: Standard View for Binary Search

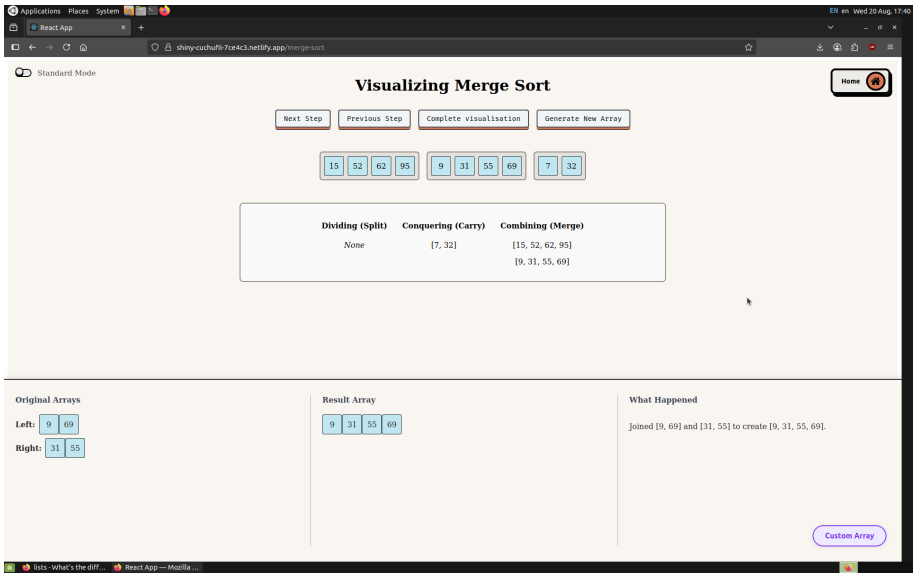

Figure A.5: Standard View for Merge Sort

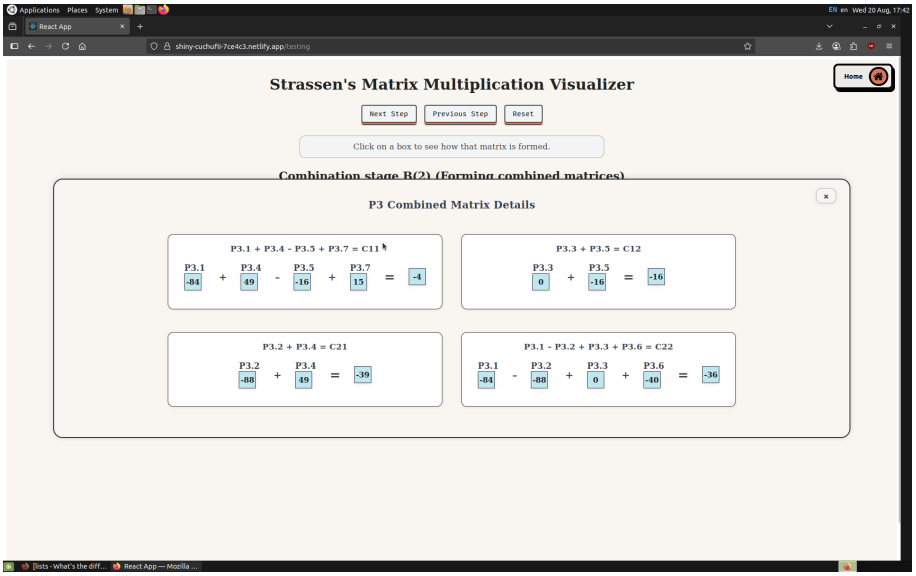

Figure A.6: Standard View for Strassens matrix multiplication

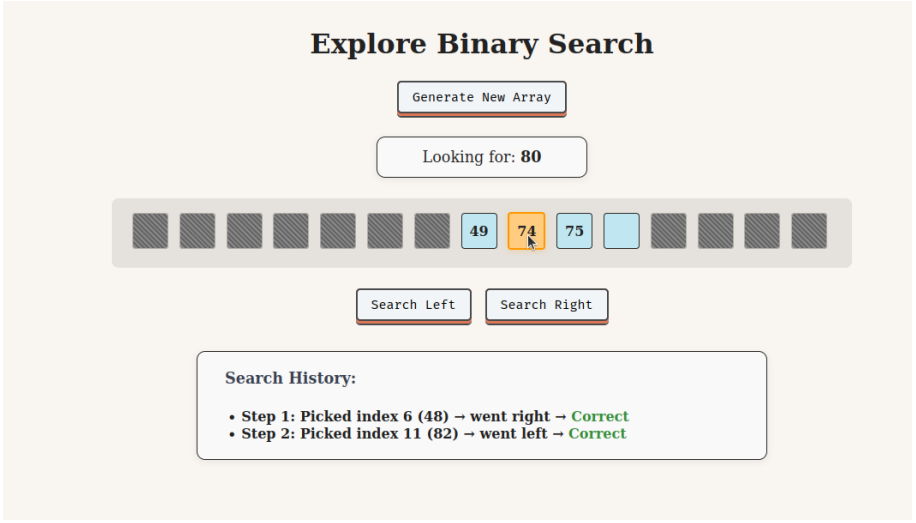

Figure A.7: Exploratory view for Binary Search

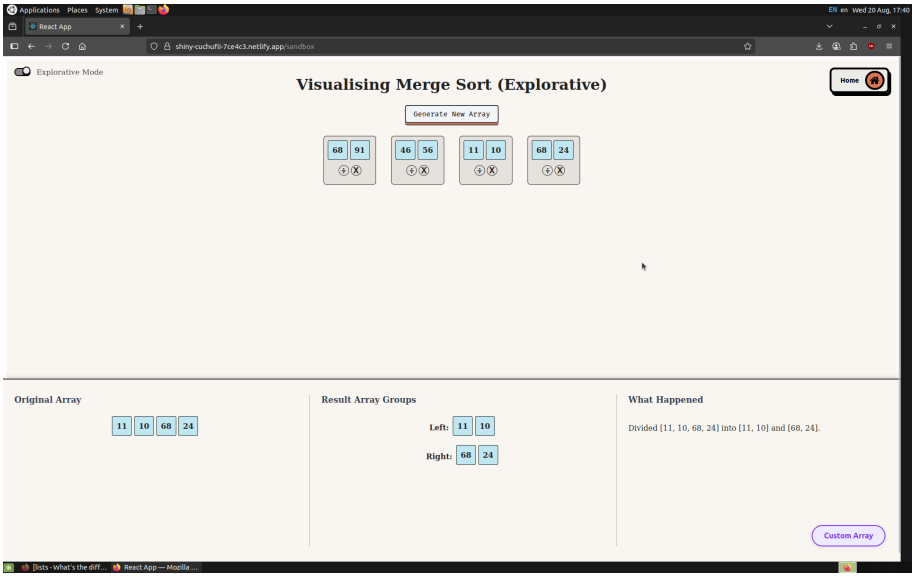

Figure A.8: Exploratory View for Merge Sort

Appendix B

Participants' information sheet

[h!]

Participant Information Sheet

Project title:	Visualization of Algorithms
Principal investigator:	Professor Murray Cole
Researcher collecting data:	Pradnesh Sanderan
Funder (if applicable):	N/A

This study was certified according to the Informatics Research Ethics Process, RT number **7045**. Please take time to read the following information carefully. You should keep this page for your records.

Who are the researchers?

Pradnesh Sanderan is an Undergraduate/MSc student in the School of Informatics, undertaking this study as part of their Honours/MSc project. The project is supervised by Professor Murray Cole.

What is the purpose of the study?

The project involves the design and evaluation of a set of interactive algorithm visualization tools, intended for use by junior Informatics students.

Why have I been asked to take part?

As an Informatics student, we seek your views on various aspects of the designed examples, and how they might be perceived by learners.

Do I have to take part?

No – participation in this study is entirely up to you. You can withdraw from the study at any time without giving a reason, until the project has been submitted. After this point, personal data will be deleted and anonymised data will be combined such that it is impossible to remove individual information from the analysis. Your rights will not be affected. If you wish to withdraw, contact the PI. We will keep copies of your original consent, and of your withdrawal request.

What will happen if I decide to take part?

You will complete an electronic (or paper) questionnaire, asking for your opinions on properties of an algorithm visualization system or systems (for example, attractiveness, difficulty, intuitiveness) as they might be perceived by junior informatics students. You will complete this in your own time, without supervision. You are not being asked to write the systems, merely to assess them.

Are there any risks associated with taking part?

There are no significant risks associated with participation.

Are there any benefits associated with taking part?

No, though we hope you will find the programs interesting and may learn something!

What will happen to the results of this study?

The results of this study will be summarised in the project report and presentation. Quotes or key findings will be anonymized: We will remove any information that could, in our assessment, allow anyone to identify you. Your data may be archived for a maximum of 1 year (in practice, only until the project report/dissertation that will be informed by the data collected, has been submitted). All potentially identifiable data will be deleted within this timeframe if it has not already been deleted as part of anonymization.

Data protection and confidentiality.

Your data will be processed in accordance with Data Protection Law. All information collected about you will be kept strictly confidential. Your data will be referred to by a unique participant number rather than by name. Your data will only be viewed by the researcher/research team: Pradnesh Sanderan and supervisor Professor Murray Cole.

All electronic data will be stored on any or all of a password-protected encrypted hard drive, on the School of Informatics' secure file servers, or on the University's secure encrypted cloud storage services (eg DataShare, ownCloud, or Sharepoint) and all paper records will be stored in a locked filing cabinet in the PI's office. Your consent information will be kept separately from your responses in order to minimise risk.

What are my data protection rights?

The University of Edinburgh is a Data Controller for the information you provide.You have the right to access information held about you. Your right of access can be exercised in accordance Data Protection Law. You also have other rights including rights of correction, erasure and objection. For more details, including the right to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk. Questions, comments and requests about your personal data can also be sent to the University Data Protection Officer at dpo@ed.ac.uk.

Who can I contact?

If you have any further questions about the study, please contact the lead researcher, Professor Murray Cole (mic@inf.ed.ac.uk).

If you wish to make a complaint about the study, please contact inf-ethics@inf.ed.ac.uk. When you contact us, please provide the study title and detail the nature of your complaint.

Updated information.

If the research project changes in any way, an updated Participant Information Sheet will be made available on http://web.inf.ed.ac.uk/infweb/research/study-updates.

Alternative formats.

To request this document in an alternative format, such as large print or on coloured paper, please contact Professor Murray Cole (mic@inf.ed.ac.uk).

General information.

For general information about how we use your data, go to: edin.ac/privacy-research

Appendix C

Participants' consent form

Participant number:_____

Participant Consent Form

Project title:	Visualization of Algorithms
Principal investigator (PI):	Professor Murray Cole
Researcher:	Pradnesh Sanderan
PI contact details:	mic@ed.ac.uk

By participating in the study you agree that:

I have read and understood the Participant Information Sheet for the above study, that I have had the opportunity to ask questions, and that any questions I had were answered to my satisfaction.

- My participation is voluntary, and that I can withdraw at any time without giving a reason. Withdrawing will not affect any of my rights.
- I consent to my anonymised data being used in academic publications and presentations.
- I understand that my anonymised data will be stored for the duration outlined in the Participant Information Sheet.

Please tick yes or no for each of these statements.

1. I agree to take part in this study.

Yes	**No**

Name of person giving consent	Date dd/mm/yy	Signature

Name of person taking consent	Date	Signature
Pradnesh Sanderan	06/08/2025	Pradnesh Sanderan

THE UNIVERSITY of EDINBURGH informatics