Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report
on
"Algolizer"

[Code No: COMP 206]

(For partial fulfillment of II Year/ I Semester in Computer Science/Engineering)

Submitted by

Suraj Bhattarai(06)
Ashitom Budhathoki (09)
Anmol Dahal (11)
Abhinav Lamsal(27)

Submitted to

Mr. Nabin Ghimire

Department of Computer Science and Engineering

Submission Date: 18th June ,2024

Bona fide Certificate

This project work on

“Algolizer”

is the bona fide work of

“Suraj Bhattarai,

Ashitom Budhathoki,

Anmol Dahal,

Abhinav Lamsal”

who carried out the project work under my supervision.

Project Supervisor

____

Dr. Rajani Chulyadyo
Assistant Professor
Department of Computer Science and Engineering

Date: 18th June ,2024

Acknowledgement

We would like to express our deepest appreciation to all those who provided with the possibility to complete this project. It has been a great pleasure to work with different individuals whose perspectives and ideas have assisted or motivated us.

We are highly indebted to out project supervisor, Dr. Rajini Chulyadyo, for her appreciable guidance, valuable input and productive criticism throughout the duration of the project. Her useful suggestions for this whole work provided with an enormous opportunity to lean the significance of collaboration and innovation. We would like to thank the Department of Computer Science and Engineering (DoCSE) and the whole university for providing us with a chance to work on the project.

Any Suggestions for the improvement of the project will be highly appreciated.

Sincerely,

Suraj Bhattarai (06)

Ashitom Budhathoki (09)

Anmol Dahal (11)

Abhinav Lamsal (27)

Abstract

The Algolizer project aims to simplify the learning of algorithms and data structures through a GUI-based visualization tool. This project addresses the difficulty many learners face in understanding complex concepts. We developed an interactive platform using C+, incorporating real-time visualizations and educational content. By offering a hands-on approach, the tool enhances comprehension and engagement. The expected outcome is an improved learning experience, fostering better retention and application of algorithmic concepts. Our work highlights the significance of visual aids and interactivity in educational tools, providing a valuable resource for students and professionals. The project anticipates positive feedback and adoption from the educational and professional community. It emphasizes the importance of interactive tools in making complex subjects accessible and engaging.

Keywords: *Interactive learning, educational tool, C++, Data structures.*

Table of Contents

List Of Figures

Abbreviations

GUI: Graphical User Interface

Ghz: Gigahertz

GB: Gigabyte

RAM: Random Access Memory

MB: Megabyte

UI: User Interface

Chapter 1 Introduction

This chapter introduces the developed system, providing a detailed overview of the project, including its background, objectives, motivation, and significance.

1.1 Background

The field of computer science, particularly algorithms and data structures, is foundational to software development and problem-solving. Understanding these concepts is crucial for students and professionals alike. Traditionally, algorithms are taught through theoretical lectures and static textbook examples, which can be challenging for many learners to grasp fully. Recent trends have seen the integration of technology in education, leading to the development of interactive and visual learning tools.

In recent years, there has been significant progress in the development of educational tools that leverage visual and interactive methods. Platforms like VisualAlgo and Algorithm Visualizer have made strides in making algorithms more accessible. These tools utilize animations and step-by-step visualizations to illustrate how algorithms work, significantly aiding in comprehension and retention. Despite these advancements, there remain several drawbacks in existing systems. Many of these tools lack a user-friendly interface or fail to provide an interactive learning experience that actively engages users.

The significance of improving these educational tools lies in their potential to transform how algorithms and data structures are taught and learned. By addressing the shortcomings of existing systems, we can create a more effective and engaging learning environment that caters to different learning styles and improves overall educational outcomes.

1.2 Objectives

The basic purpose of the Algolizer project is to develop a user-friendly and interactive tool for learning algorithms and data structures. The specific objectives of the project are:

- To create a GUI-based visualization tool for learning Algorithms and Data structures.
- To incorporate real-time visualizations and educational content for enhanced comprehension.
- To provide a valuable resource for students and professionals for mastering algorithmic concepts.

1.3 Motivation and Significance

The motivation to choose this topic stems from the recognized challenges students face in understanding algorithms and data structures through traditional teaching methods. The complexity of these concepts often leads to difficulties in comprehension and retention, which can hinder progress in computer science education. By developing an interactive visualization tool, we aim to address these challenges and provide an alternative learning approach that is both effective and engaging.

This project addresses the drawbacks of existing systems by offering a more user-friendly and interactive platform. Unlike many current tools, Algolizer focuses on providing real-time visualizations and an intuitive interface that facilitates active learning. The inclusion of interactive elements ensures that users are not just passive observers but active participants in the learning process.

The features of Algolizer include interactive visualizations, educational content, and a user-centric design that enhances the learning experience. By combining these elements, the project aims to set a new standard for algorithm education tools, making complex subjects more accessible and enjoyable for learners at all levels.

Chapter 2 Related Works

There are already similar projects tackling the objectives of Algolizer, focusing on visualizing algorithms and data structures for educational purposes. These existing projects served as valuable reference for Algolizer's development. These projects include:

2.1 VisualAlgo

VisualAlgo (VisualAlgo, 2011)is an innovative platform designed to revolutionize the way algorithms and data structures are learned and understood. By combining interactive visualizations with comprehensive explanations, VisualAlgo provides users with a unique and engaging learning experience. Whether you're a student delving into the basics or a seasoned professional exploring advanced concepts, VisualAlgo offers a visually immersive journey that simplifies complex ideas and fosters a deeper understanding of algorithms and data structures.

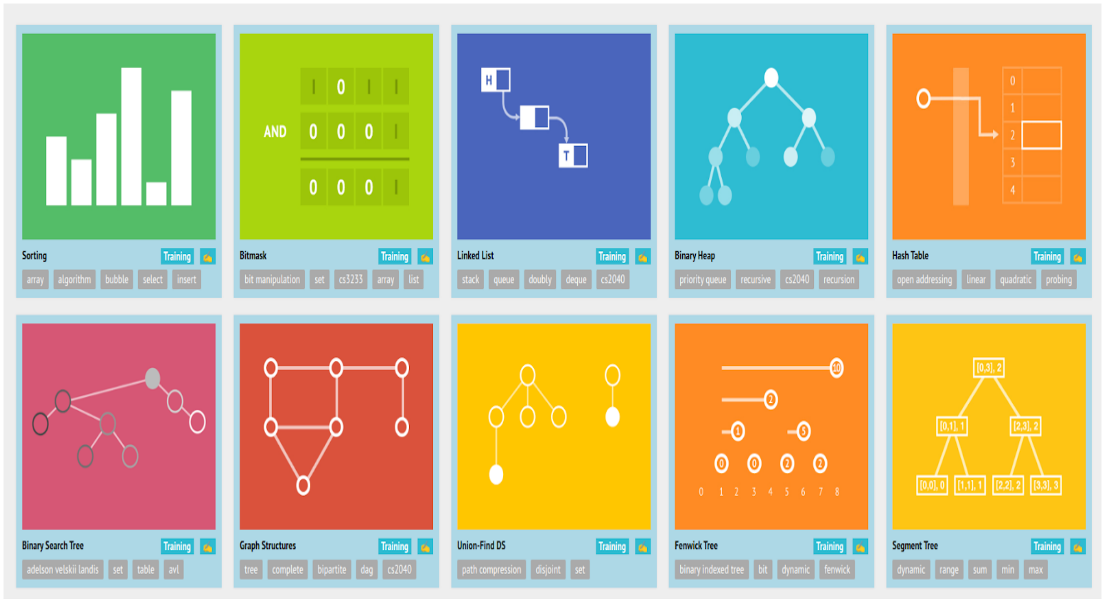
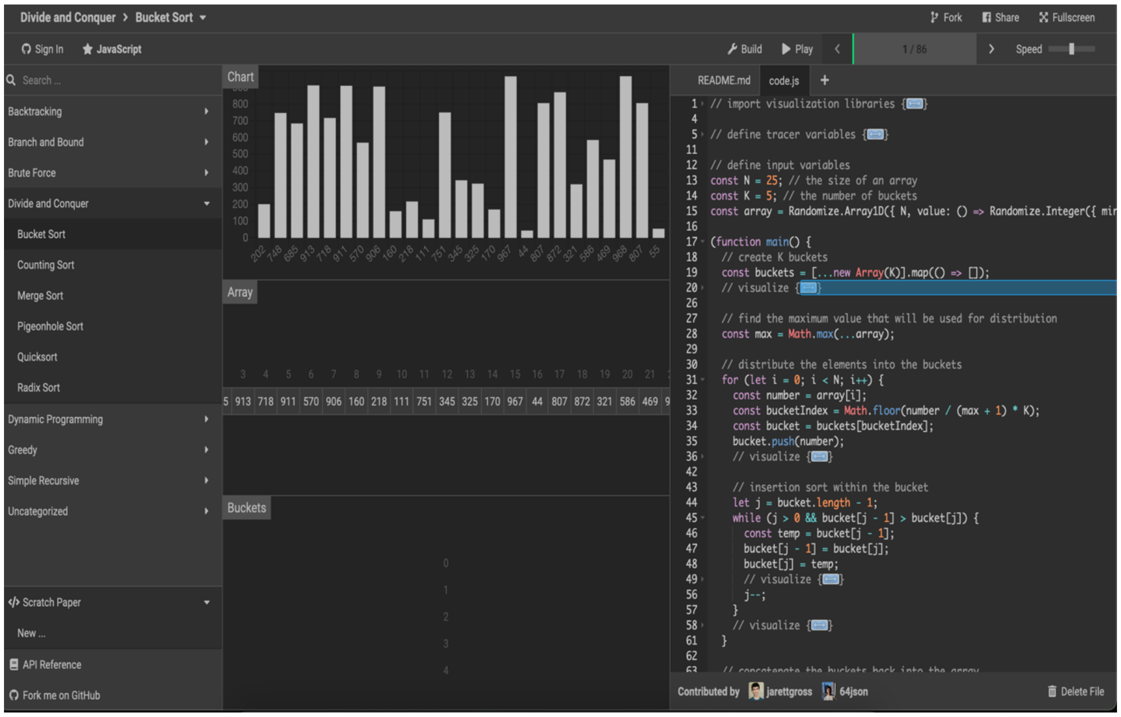

Sorting
Training
array algorithm bubble select insert
Bitmask
Training
bit manipulation set cs3233 array list
Linked List
Training
stack queue doubly deque cs2040
Binary Heap
Training
priority queue recursive cs2040 recursion
Hash Table
Training
open addressing linear quadratic probing
Binary Search Tree
Training
adelson velskii landis set table avl
Graph Structures
Training
tree complete bipartite dag cs2040
Union-Find DS
Training
path compression disjoint set
Fenwick Tree
Training
binary indexed tree bit dynamic fenwick
Segment Tree
Training
dynamic range sum min max

Figure 2.1.1 VisualAlgo

2.2 Algorithm-Visualizer.org

(Algorithm-Visualizer.org, n.d.)is a cutting-edge platform dedicated to enhancing algorithmic learning through interactive visualizations. It offers a wide range of algorithms and data structures, each accompanied by detailed explanations and step-by-step visualizations. Users can explore various sorting, searching, and graph algorithms, gaining valuable insights into their operations and complexities. Algorithm-Visualizer.org's intuitive interface and real-time animations make learning algorithms not only educational but also entertaining. Whether you're a student, educator, or professional, Algorithm-Visualizer.org provides a powerful tool to deepen your understanding of algorithms and sharpen your problem-solving skills.



```
// import visualization libraries {}

// define tracer variables {}

// define input variables
const N = 25; // the size of an array
const K = 5; // the number of buckets
const array = Randomize.Array1D({ N, value: () => Randomize.Integer({ mi

(function main() {
  // create K buckets
  const buckets = [...new Array(K)].map(() => []);
  // visualize {}

  // find the maximum value that will be used for distribution
  const max = Math.max(...array);

  // distribute the elements into the buckets
  for (let i = 0; i < N; i++) {
    const number = array[i];
    const bucketIndex = Math.floor(number / (max + 1) * K);
    const bucket = buckets[bucketIndex];
    bucket.push(number);
    // visualize {}

    // insertion sort within the bucket
    let j = bucket.length - 1;
    while (j > 0 && bucket[j - 1] > bucket[j]) {
      const temp = bucket[j - 1];
      bucket[j - 1] = bucket[j];
      bucket[j] = temp;
      // visualize {}
      j--;
    }
    // visualize {}
  }

  // concatenate the buckets back into the array
```

Figure 2.2.1 Algorithm-Visualizer.org

Chapter 3 Design and Implementation

This chapter details the design and implementation process of the Algolizer project. It includes the sequential procedures followed, the algorithms and flowcharts used, and the system diagrams that illustrate the overall design and functionality of the project.

3.1 System Overview

The Algolizer system is designed to provide a user-friendly interface for visualizing algorithms and data structures. The core components of the system include the user interface and the visualization engine. The system architecture is modular, allowing for easy updates and additions of new algorithms and features.

3.2 System Requirement Specifications

This section specifies the requirements of the developed system, including both software and hardware specifications.

3.2.1 Software Specifications

The software specifications include the functional and non-functional requirements, as well as software dependencies necessary for the Algolizer system.

3.2.1.1 Functional Requirements

- **User Interface**
  The main screen of the Algolizer project is designed with distinct sections for Algorithms and Data Structures. In the Algorithms section, users will find buttons dedicated to sorting and searching algorithms. The Data Structures section features buttons for various data structures, including trees, linked lists etc. providing a comprehensive and intuitive interface for exploring different aspects of algorithms and data structures. This layout ensures that users can effortlessly navigate the application

- **Visualization Engine**

  The visualization engine of the Algolizer project is designed to offer real-time visualizations of the execution of algorithms and the manipulation of data structures. This interactive feature enhances understanding by enabling users to observe the behavior of algorithms and data structures in detail, analyze each step, and gain a deeper insight into how these concepts work in practice, making learning more engaging and effective.

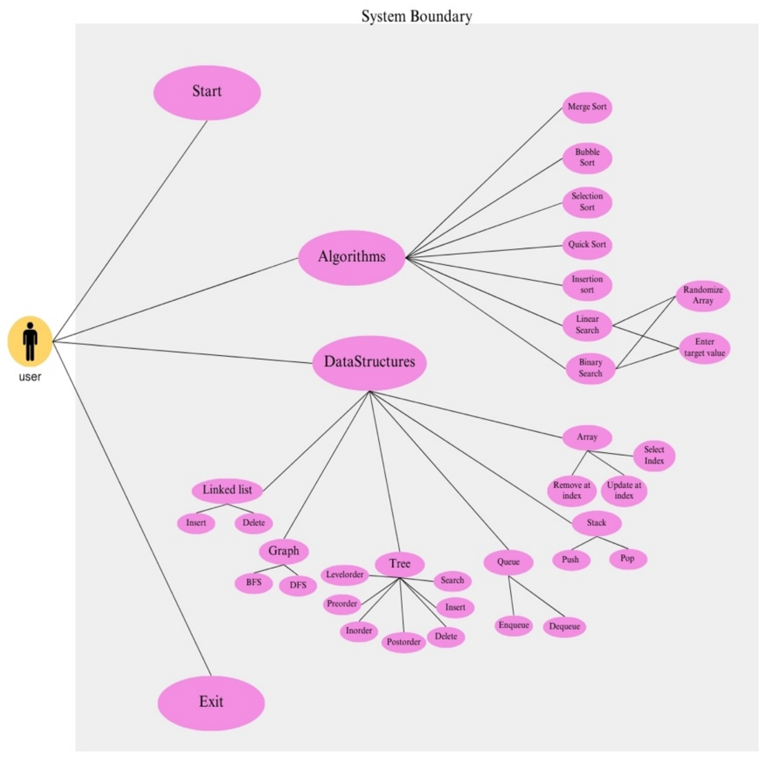


Figure 3.2.1.1 Use case Diagram

3.2.1.2 Non-Functional Requirements

- **Performance**

  The performance of the Algolizer project is optimized to deliver smooth and responsive visualizations with minimal latency. This ensures that users experience a seamless interaction with real-time visualizations of algorithms and data structures. By maintaining high performance and responsiveness, the system allows users to effectively observe and interact with the visualizations without any disruptions, facilitating a more productive and enjoyable learning experience.

- **Usability**

  The usability of the Algolizer project focuses on providing an intuitive and easy-to-navigate user interface. The design ensures that users can effortlessly access and interact with different sections and features, enhancing the overall user experience. This user-friendly approach enhances accessibility, allowing users to effectively utilize Algolizer and engage with its educational content seamlessly, making learning more enjoyable.
- **Compatibility**

  The software should be compatible with major operating systems, including Windows, macOS, and Linux, ensuring broad accessibility and usability across different platforms. This compatibility allows users to utilize Algolizer regardless of their preferred operating system, making it a versatile tool for learning algorithms and data structures.

3.2.2 Hardware Specifications

This section describes any additional hardware that may be required for the project and provides information about the required minimum configuration of the system to run the Algolizer smoothly.

Recommended Hardware Requirements:

- **Processor:** Dual-core processor (2.0 GHz or faster)
- **Memory:** 2 GB RAM
- **Storage:** 200 MB of free disk space for installation
- **Graphics:** Integrated graphics with support for OpenGL 3.3 or higher
- **Display:** 800x600 resolution or higher
- **Input Devices:** Keyboard and mouse

3.3 Design Procedure

The basic wireframe for Algolizer was designed to establish a clear and intuitive layout for user interaction. Initial sketches were created to map out the placement of key elements, such as buttons for Algorithms and Data Structures on the main screen, ensuring easy navigation The wireframes included placeholders for visualization panels and buttons. Feedback from potential users was gathered and incorporated to enhance usability. This iterative wireframing process ensured a solid foundation for the subsequent development stages.

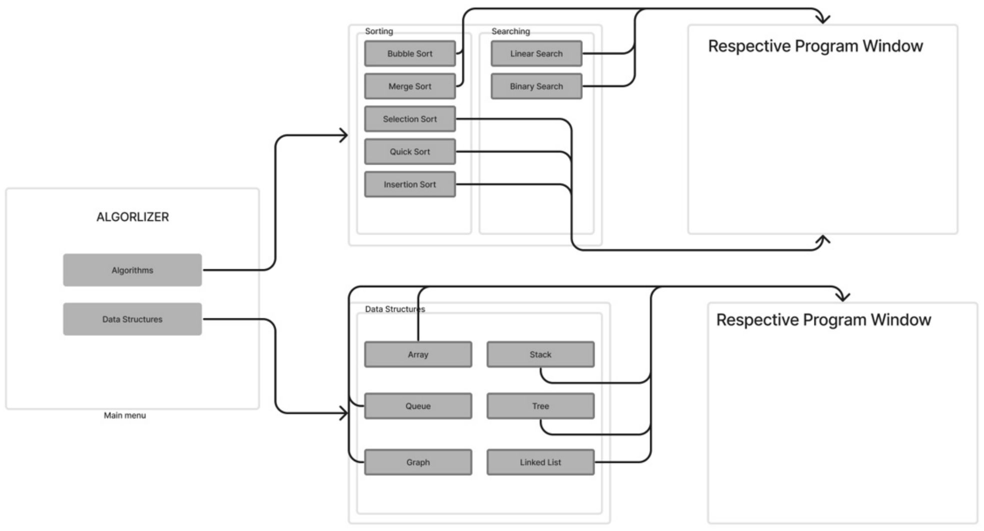


Figure 3.3.1 Wireframe Design

3.4 Implementation

3.4.1 User Interface Design

The user interface (UI) is designed using (Raylib, n.d.),a simple and easy-to-use library for creating graphical applications. The UI includes various buttons and input fields for selecting algorithms, adjusting parameters, and controlling the visualization. The main screen features buttons for Algorithms and Data Structures. The Algorithms section contains buttons for sorting and searching algorithms, while the Data Structures section includes buttons for various data structures such as Trees, Linked Lists etc.

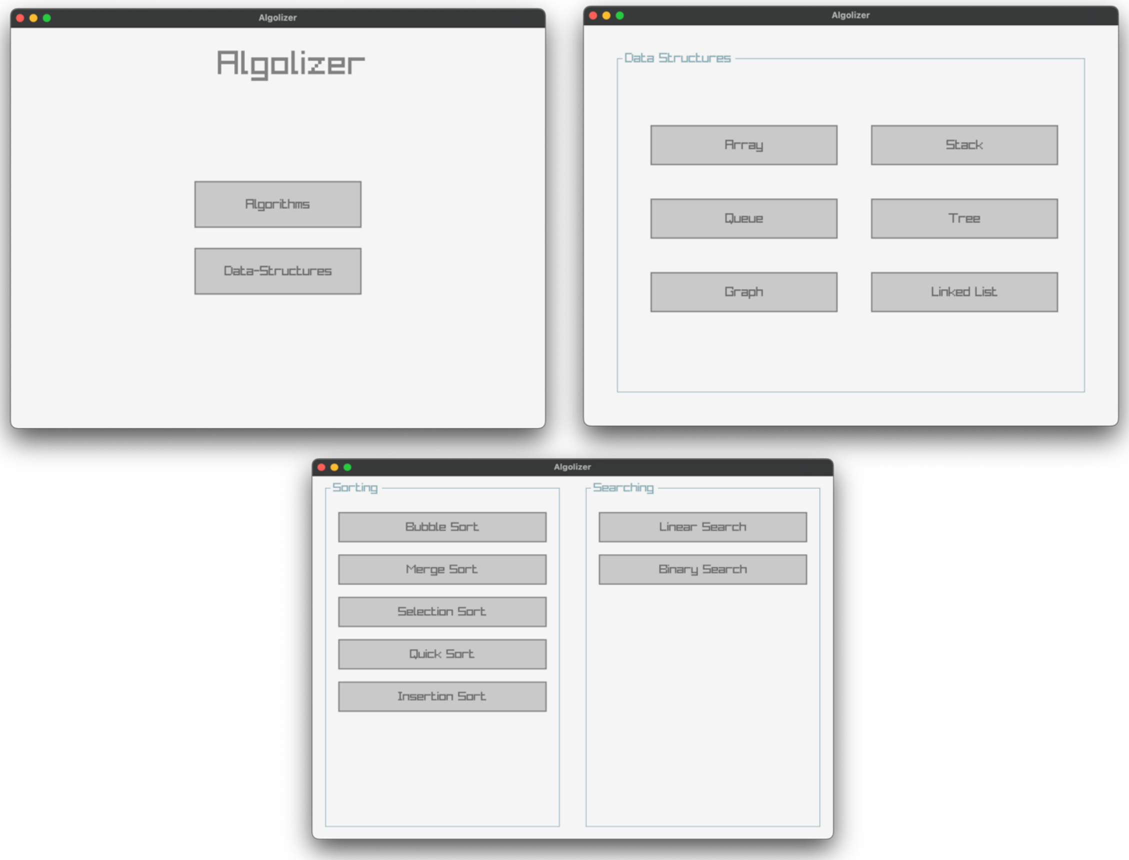


Figure 3.4.1.1 UI Design

3.4.2 Visualization Engine

The design of the visualization engine is central to the Algolizer system. This engine is tasked with rendering visual representations of algorithms and data structures. It supports real-time updates and interactions, enabling users to observe the step-by-step execution of algorithms. The design process involved meticulous planning to ensure smooth rendering and responsive interactions, providing an intuitive and educational experience for users.

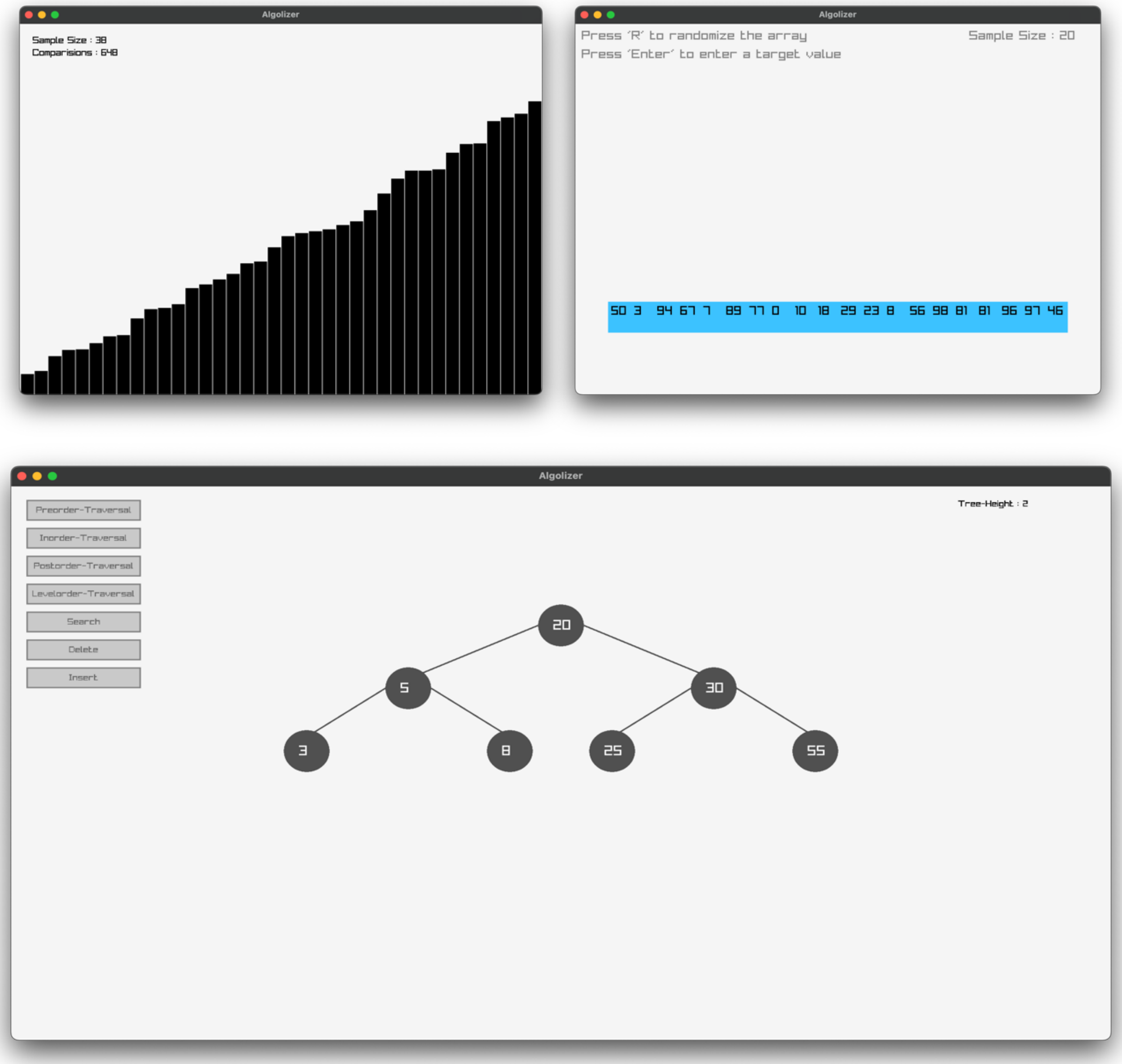


Figure 3.4.2.1 Visualization Engine

3.4.3 Testing and Feedback

During the testing phase, extensive tests were conducted to verify the accuracy and performance of Algolizer. This included testing various algorithms and data structures to ensure they functioned correctly and efficiently. Additionally, feedback was gathered from users, including students and professionals, to identify areas for improvement. Based on this feedback, necessary enhancements and adjustments were made to enhance the overall user experience and functionality of Algolizer.

3.4.4 Software Dependencies

- **Programming Language**
  C++ is the primary language for the Algolizer project due to its performance, flexibility, and extensive library support. Its ability to handle low-level memory management and support for object-oriented programming make it ideal for developing complex visualization tools.
- **Graphics Engine**
  (Raylib, n.d.) is a simple, easy-to-use library for graphics and visualization. It provides 2D and 3D rendering, input handling, and audio management, essential for creating interactive visual representations of algorithms and data structures. Raylib's minimalist API and compatibility with C++ make it a great fit for Algolizer.
- **User Interface**
  Raygui is a portable, immediate-mode GUI library that works seamlessly with (Raylib, n.d.). It offers essential GUI components like buttons, text boxes, and sliders, enabling the creation of a user-friendly interface. Its integration with (Raylib, n.d.) ensures a cohesive user experience, allowing for rapid development and prototyping of the UI.
- **Version Control**
  Git is the version control system used for managing the development of the Algolizer project. It allows multiple developers to collaborate efficiently by tracking changes, managing code versions, and facilitating seamless integration of contributions. Git's distributed nature ensures that every contributor has the full history of the project, providing resilience and flexibility in development.
  GitHub hosts the Algolizer project's repositories, providing features like pull requests, issue tracking, and project management tools. It facilitates efficient collaboration, code review, and task management among team members.

3.5 Flowchart

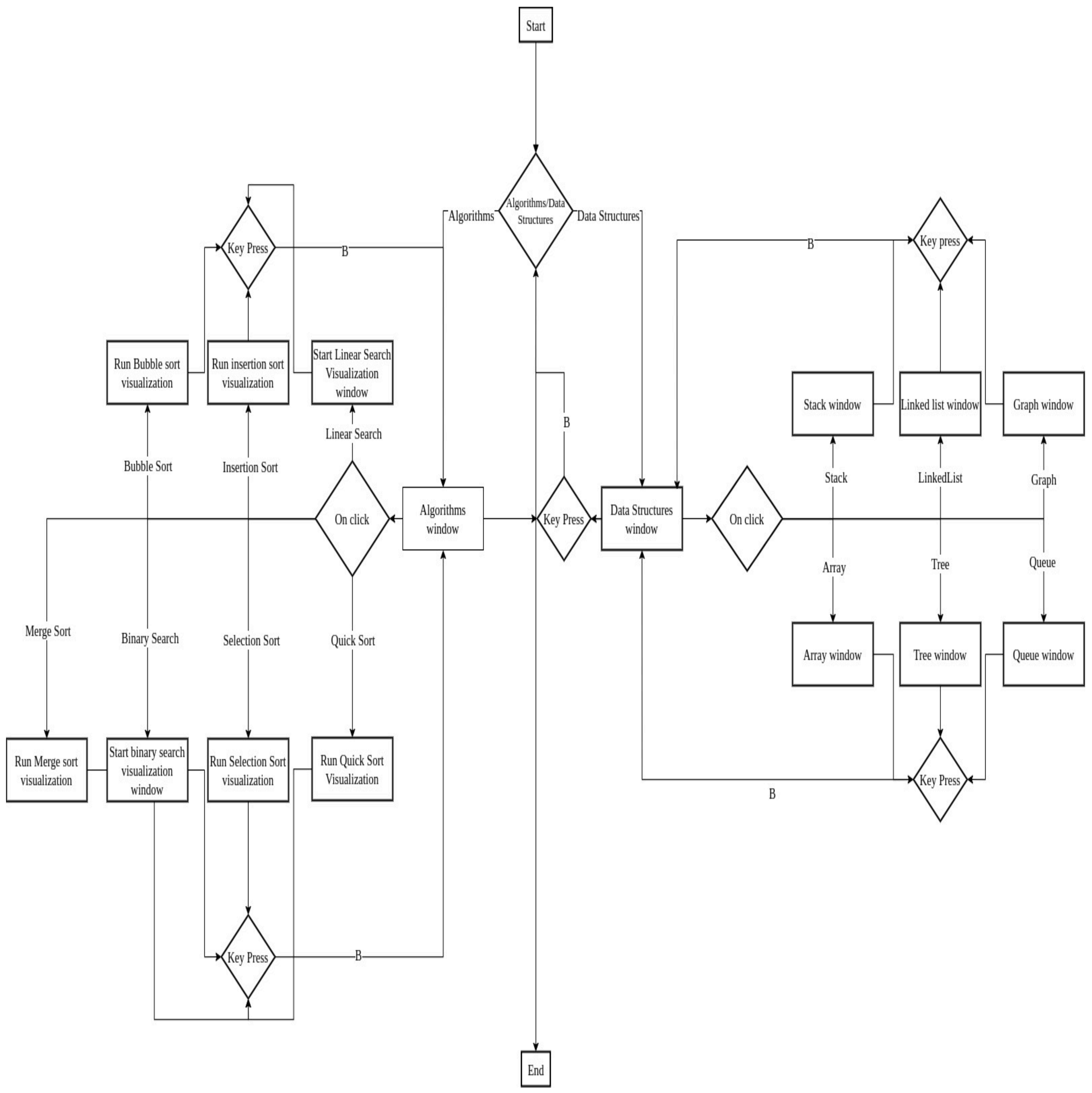


Figure 3.5.1 System Flowchart

Chapter 4 Discussion on the Achievements

In this chapter, we delve into the achievements of the Algolizer project, focusing on the challenges encountered and the implementation of new and vital features. The following section outlines the key features that have been developed as part of Algolizer's journey towards creating an accessible and engaging platform for learning algorithms and data structures.

4.1 Features

During the development of Algolizer, several new and vital features have been implemented to enhance the user experience and functionality of the application. These features include:

- **Interactive Visualizations**
  Algolizer offers interactive visualizations that allow users to witness algorithms and data structures in action, promoting a deeper understanding of their functionality.
- **Algorithm Selection**
  Users can choose from a wide range of sorting and searching algorithms, enabling them to compare different algorithms' performance and behavior.
- **Data Structure Representation**
  Algolizer provides visual representations of various data structures such as arrays, stacks, queues, trees, graphs and linked lists, aiding users in grasping complex data organization concepts.
- **Real-time Updates**
  The application provides real-time updates during algorithm execution and data structure operations, allowing users to track changes and outcomes as they occur.

4.2 Challenges

While developing Algolizer, several challenges were encountered that led to deviations from the initial objectives:

- **Performance Optimization**
  Initially, there were challenges in optimizing the performance of certain algorithms, especially when dealing with large datasets. This required fine-tuning and optimization techniques to ensure smooth and efficient visualizations.
- **Compatibility Issues**
  Integrating (Raylib, n.d.) and Raygui libraries posed compatibility issues with certain operating systems and hardware configurations. Addressing these issues required thorough testing and debugging.
- **Compiler Compatibility**
  Algolizer initially faced compatibility issues with different compilers used on Windows and Unix-based systems. Variations in compiler versions, standards compliance, and optimizations led to unexpected behavior and inconsistencies in application performance.
- **Data Visualization Scalability**
  Ensuring that Algolizer's visualizations remained clear and informative even with large datasets posed scalability challenges. Optimizing rendering performance and interaction responsiveness for varying data sizes were crucial.
- **Algorithm Complexity**
  Implementing and visualizing complex algorithms such as tree traversal or sorting required thorough understanding and careful implementation. Managing algorithmic complexity while maintaining code readability and efficiency was a challenge.

Chapter 5 Conclusion and Recommendation

In conclusion, the development journey of Algolizer has been marked by significant achievements and progress towards its objectives. However, like any project, there are certain limitations to consider. This section discusses the limitations encountered during the development of Algolizer and outlines possible future enhancements to address these limitations effectively

5.1 Limitations

While Algolizer has achieved several milestones and objectives, there are certain limitations to acknowledge:

- **Performance Optimization**
  Although efforts were made to optimize performance, there may still be room for further improvement, especially with larger datasets and complex algorithms.
- **Platform Compatibility**
  While Algolizer aims for cross-platform compatibility, there may be occasional compatibility issues or differences in behavior across different operating systems and environments.
- **Educational Depth**
  While Algolizer provides a solid foundation for understanding algorithms and data structures, it may not delve deeply into advanced or specialized topics within these domains. Users seeking advanced knowledge may need supplementary resources or materials.

5.1 Future Enhancements

To address the limitations and continue improving Algolizer, several future enhancements can be considered:

- **Advanced Algorithm Visualizations**

  Introduce visualizations for advanced algorithms, such as dynamic programming or graph algorithms, to provide a broader educational experience.
- **Performance Boost**

  Implement further performance optimizations to handle larger datasets and complex algorithms more efficiently, ensuring smooth user experience across platforms.
- **Interactive Learning Modules**

  Develop interactive learning modules or tutorials within Algolizer to provide step-by-step guidance and deepen users' understanding of algorithms and data structures.

By focusing on these future enhancements while building upon the strengths and achievements of Algolizer, the project can continue to evolve as a valuable tool for learning and exploring algorithms and data structures.

References

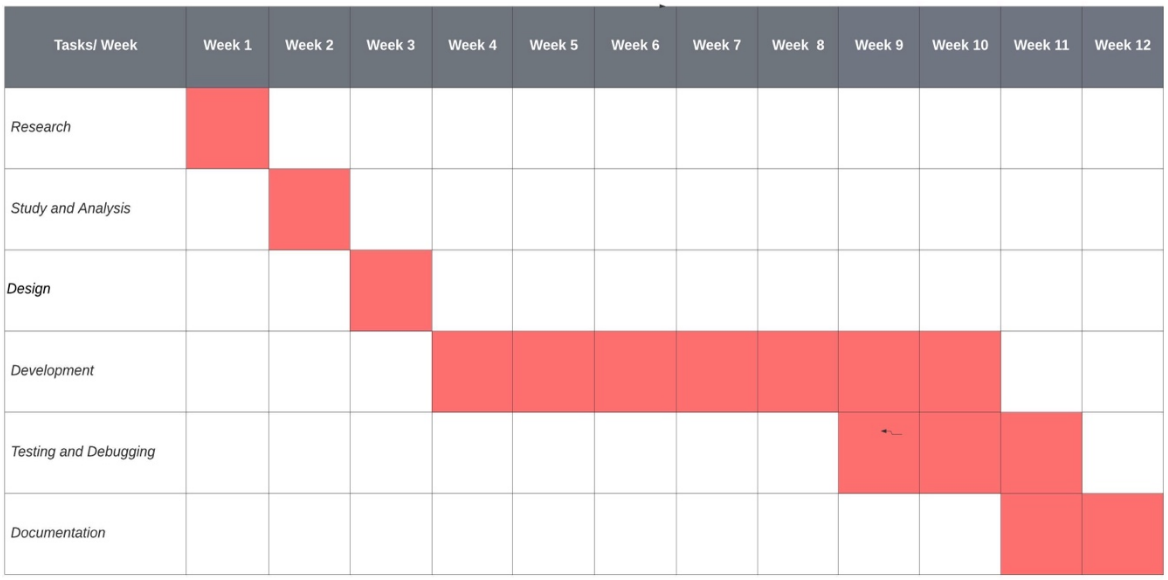
Algorithm-Visualizer.org. (n.d.). Retrieved from Algorithm-Visualizer.org: https://algorithm-visualizer.org/

Raylib. (n.d.). Retrieved from Raylib Website: https://www.raylib.com/

VisualAlgo. (2011). Retrieved from VisualAlgo: https://visualgo.net

Appendix

A.1 Gantt Chart



Tasks/ Week	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Research	🟥											
Study and Analysis		🟥										
Design			🟥									
Development				🟥	🟥	🟥	🟥	🟥	🟥	🟥		
Testing and Debugging									🟥	🟥	🟥	
Documentation											🟥	🟥

Figure A.1 Gantt Chart