SORTING VISUALIZER

A PROJECT REPORT

Submitted by

Mrinal Mayank
UID : 19BCS1605
Semester 7

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

DECEMBER 2022

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

BONAFIDE CERTIFICATE

Certified that this project report **“SORTING VISUALIZER”** is the bonafide work of “**MRINAL MAYANK (19BCS1605)”** who carried out the project work under my/our supervision.

SIGNATURE

Dr. Navpreet Kaur Walia
HEAD OF THE DEPARTMENT
Computer Science & Engineering

SIGNATURE

Anup Lal Yadav(E12240)
SUPERVISOR (Assistant Professor)
Computer Science & Engineering

Submitted for the project viva-voce examination held on ____________________________

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

In the completion of the project based on algorithm analysis a lot of guidance , training and practice was required . First and foremost we would thank God for equipping me with such knowledge and desire to achieve the goal .
In the successful completion of this project and the work undertaken we would like to thank Chandigarh University for providing the opportunity to work on such projects under the guidance of skilled mentors and teachers .

A special thank of gratitude to our mentor *Anup Sir* for supporting and guiding us in technical as well as non-technical work which result in the final successful completion of the project . Mentors are just like a bridge between the students and the success to guide us to the right path.

At last we would like to thank all the engineers and researchers to develop such platform for us to conduct such tasks through their software and hardware . Once again we thank all the mentions above for helping me a lot in the final successful completion of the project “SORTING VISUALIZER”**.**

Mrinal Mayank (19BCS1605)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Visualization is the process of graphically or pictorially representing objects, concepts, or processes. Visualization is attention-grabbing, and generally more efficient than the verbal or numerical presentation of the same information or data in terms of understandability and user-friendliness.

Graphical representation of data helps in extracting, abstracting, and presenting meaningful, pertinent information from large volumes of complex data. Visual demonstrations and interactive visualization tools can be used effectively to convey complex ideas. Graphical display of sorting methods dynamically while in execution would help in better understanding and comparative appreciation of the sorting algorithms. The primary objective of this project was to develop a graphical tool for the visualization of a number of sorting methods.

Sorting algorithms are basically used to arrange a data structure according to a specific order such as ascending, descending, numerical order or may be in lexicographical order. This method is one of the most important, known and widely spread in the field of computer science. For a very long time, new methods has been developed in order to make this procedure as faster as possible. There are currently many sorting algorithms, each with its own particular characteristics. They are categorized as per the two metrics: space and time complexity.

The Sorting Visualizer entitled "*Logic Box*" will be designed as an Android NDK (Native Development Kit) Application in order to visualize different types of sorting algorithms including bubble sort visualizer, selection sort visualizer, insertion sort visualizer, master's theorem calculator in order to test the running time complexity of the algorithm and also an additional stack conversion module.

Keywords: Sorting Visualizer, Visualization, Time Complexity, Android

सार

विज़ुअलाइज़ेशन वस्तुओं, अवधारणाओं या प्रक्रियाओं को रेखांकन या चित्रमय रूप से प्रस्तुत करने की प्रक्रिया है। विज़ुअलाइज़ेशन ध्यान खींचने वाला है, और आम तौर पर समझ और उपयोगकर्ता-मित्रता के संदर्भ में समान जानकारी या डेटा की मौखिक या संख्यात्मक प्रस्तुति की तुलना में अधिक कुशल है।

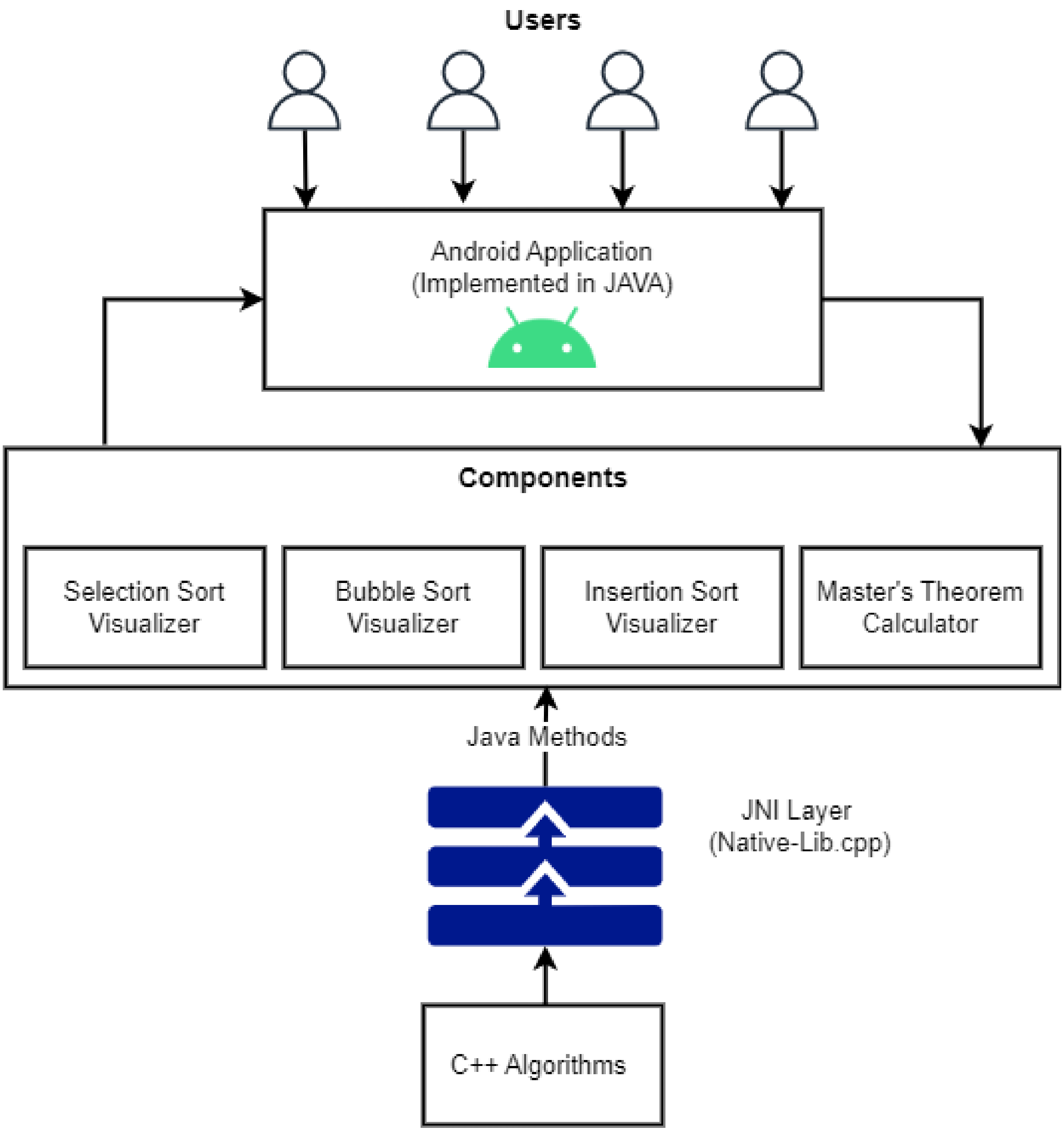
डेटा का ग्राफिकल प्रतिनिधित्व जटिल डेटा की बड़ी मात्रा से सार्थक, प्रासंगिक जानकारी निकालने, अमूर्त करने और प्रस्तुत करने में मदद करता है। जटिल विचारों को व्यक्त करने के लिए दृश्य प्रदर्शन और इंटरैक्टिव विज़ुअलाइज़ेशन टूल का प्रभावी ढंग से उपयोग किया जा सकता है। निष्पादन के दौरान गतिशील रूप से छँटाई विधियों का चित्रमय प्रदर्शन छँटाई एल्गोरिदम की बेहतर समझ और तुलनात्मक मूल्यांकन में मदद करेगा। इस परियोजना का प्राथमिक उद्देश्य कई छँटाई विधियों के दृश्य के लिए एक चित्रमय उपकरण विकसित करना था।

सॉर्टिंग एल्गोरिदम का उपयोग किसी विशिष्ट ऑर्डर संबंध के अनुसार डेटा संरचना को सॉर्ट करने के लिए किया जाता है, जैसे संख्यात्मक क्रम या लेक्सिकोग्राफ़िकल ऑर्डर। यह ऑपरेशन कंप्यूटर विज्ञान में सबसे महत्वपूर्ण और व्यापक में से एक है। लंबे समय से, इस प्रक्रिया को तेज और तेज बनाने के लिए नए तरीके विकसित किए गए हैं। वर्तमान में सैकड़ों अलग-अलग सॉर्टिंग एल्गोरिदम हैं, जिनमें से प्रत्येक की अपनी विशिष्ट विशेषताएं हैं। उन्हें दो मेट्रिक्स के अनुसार वर्गीकृत किया गया है: अंतरिक्ष जटिलता और समय जटिलता।

"लॉजिक बॉक्स" नामक सॉर्टिंग विज़ुअलाइज़र को एंड्रॉइड एनडीके (नेटिव डेवलपमेंट किट) एप्लिकेशन के रूप में डिज़ाइन किया जाएगा ताकि परीक्षण करने के लिए बबल सॉर्ट विज़ुअलाइज़र, सेलेक्शन सॉर्ट विज़ुअलाइज़र, इंसर्शन सॉर्ट विज़ुअलाइज़र, मास्टर के प्रमेय कैलकुलेटर सहित विभिन्न प्रकार के सॉर्टिंग एल्गोरिदम की कल्पना की जा सके। एल्गोरिथम की रनिंग टाइम जटिलता और एक अतिरिक्त स्टैक रूपांतरण मॉड्यूल भी।

खोजशब्द: छँटाई विज़ुअलाइज़र, विज़ुअलाइज़ेशन, समय जटिलता, एंड्रॉयड

GRAPHICAL ABSTRACT



ABBREVIATIONS

JNI	Java Native Interface
NDK	Native Development Kit
DSA	Data Structures & Algorithms
UI	User Interface
ANIMAL	A New Interactive Modeler for Animations in Lectures
JHAVE	Java-hosted Algorithm Visualization Environment
JAWAA	Java and Web-based Algorithm Animation
JSAV	Javascript Algorithm Visualization
DAVE	Dynamic Algorithm Visualization Environment
XML	Extensible Markup Language
DFD	Data Flow Diagram
SDK	Software Development Kit

SYMBOLS

O	Big Oh Notation
Θ	Big Theta Notation
Ω	Big Omega Notation

CHAPTER 1
INTRODUCTION

1.1. Identification of Relevant Contemporary Issues

Since the early ages of computing, the sorting techniques has attracted a great deal of research, but due to the complexity of solving it efficiently even its simple, familiar statement. It is the rearrangement of data in ascending or descending order. These algorithms are essential in the introductory computer science subject, where the abundance of sorting algrithm for the problem provides a precise introduction to a number of core algorithm concepts, such as big O notations, data structures, randomized algorithms, divide and conquer algorithms and case analysis.

A sorting visualizer is a useful education tool that helps students to visualize an algorithm by using transition effects and basic animations to understand how an algorithm is working at each step. Many researchers have concluded that the students who are involved in the visualization process of the data structure and algorithms to understand the key working of any algorithm yield a better result in comparison to others who use classical methods because of the *dynamic visual representation* and interactivity of the sorting visualizers. The product will be more focused on students & researchers, yet it can be useful to other users also. Below are the identified contemporary issues subject to the problem :

- Since researchers need a deep understanding of the solution they are doing research upon and visualizers can be the easiest way to visualize their solutions graphically and efficiently.
- *Data structures and algorithms (DSA)* are base and it plays a crucial role in computer science education and programming. Learning and teaching DSA are both challenging for students as well as teachers. Past researches have shown that the new learners find it difficult to understand the implementation of data structures (DS) and algorithms in programming. Instructors and educators until recently use physical means like pen-paper, textbook diagrams, etc. This requires a great deal of effort to convey the logic of an algorithm.
- In today's generation computer programming have become a very important skill for

students to acquire new age technologies. They need to be well equipped with the ability of critical thinking or algorithmic thinking and problem-solving skills apart from the basic usage of computers.

- Animation and graphical representation techniques have evolved and are often used to understand the step-by-step workflow of algorithms. Algorithm visualizations also one such technique. Sorting Visualizers will help them to understand the concepts and the working of the algorithm more *precisely*.
- Overall, sorting visualizers help learners to get a better understanding of DSA. It also increases the interest of the user by providing more interactive environment than simple text or classic animations. It solves the problem for teachers to recreate the same presentation for algorithms that uses a similar data structure, as it becomes very easy to change the existing data and base logic of the visualization according to the need.
- Sorting is a common algorithm studied in information technology, computer science, and engineering. In the field of computer science particularly in data structures & algorithm, selection sorting algorithm is for sorting a series of data. This concept is quiet hard to be understood for the students who are in the field of computer science, especially in creating coding & scripting in programming language.
- To understand the whole working process of an algorithm, it needs to be broken into steps and it is very difficult to present these steps with static tools like texts and pictures.
- There have been many solutions found by the researchers to overcome these barriers like graphical representations, classical animations, explanation videos, etc. These methods enable the learner to visualize the logic but do not provide the diversity to work with different data sets or reuse any existing asset for another similar task with a different approach. Apart from these approaches there evolves an approach that gains popularity amongst the educators in computer science education is Sorting Visualizer.

1.2. Identification of Problem

Sorting algorithms are generally used to sort a data structure according to a specific order, such as in ascending, descending or lexicographical order. This operation is one of the most essential and widely spread in computer science. For a very long time, new methods has been developed to make this process faster. In order to develop new techniques we need to first visualize and then test our algorithm. There are currently more than hundreds of different sorting techniques, with its own specific property.

Sorting algorithms can be hard to understand and it's quiet easy to get confused. We generally believe visualizing the sorting algorithms can be a good way to better understand their functionality while having fun with algorithms. Algorithm visualization opens the door to the field of Algorithm Analysis. Let us dive deep into this issue:

- There are currently ore than hundreds of different sorting algorithms, each with its own particular characteristics. They are grouped according to two metrics: space and time complexity. Those two kinds of metrics are represented with asymptotic notations which constitutes of symbols O, Θ, Ω, representing respectively the upper bound, the tight bound, and the last lower bound of complexity, specifying in brackets an expression in terms of n, the number of the elements of the data structure used for sorting.

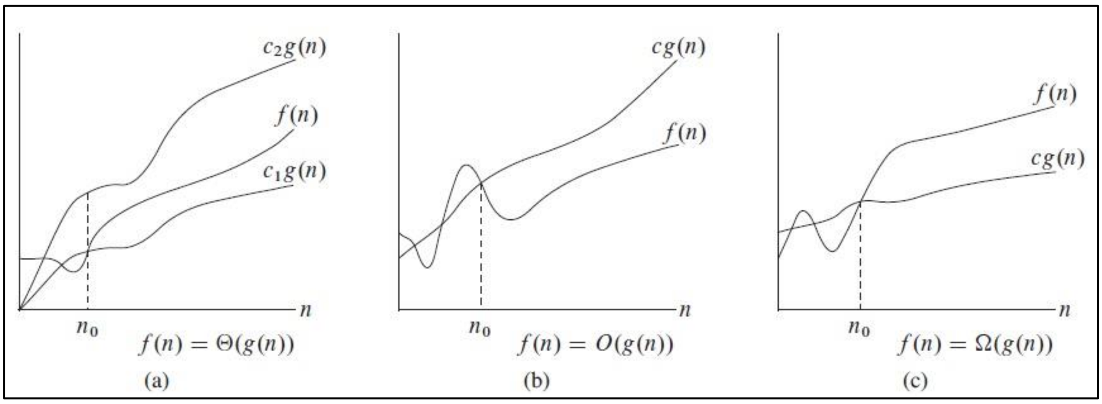


Figure 1.1. Asymptotic Bounds

- The calculation & analysis of the running time of a sorting algorithm is based on the number of procedural steps, i.e., the number of comparisons and swaps, required to sort n items. Programs access data in one of two ways: either keys are accessed for comparison, or the entire records are accessed to be shuffled. It is assumed that the implementation of the sorting algorithms here "avoids shuffling the records by doing an indirect type of sort, i.e., the records by themselves are not necessarily rearranged, in fact it is rather an array of pointers that is rearranged so that the first index pointer points to the smallest record in the data structure, etc.
- Another important thing to be considered is the space complexity of the algorithms.

All the sorting algorithms require an array or data structure of size n to hold n records to be arranged in sorted order. The amount of extra memory space required is an important metrics to be considered when we are doing analysis or visualization of the algorithms. The sorting methods could be categorized into three types on the basis of extra memory space required in the running time of the sorting algorithm :

1. Those that sort data in place and use no extra memory space, except perhaps for a small stack or table size.
2. Those that use a linked-list for the representation and thus uses n extra words of memory space for list pointers.
3. Those that need enough extra memory space in order to hold another copy of the array to be sorted.

1.3. Identification of Tasks

The android application is to be implemented in java and the algorithms are to be developed in C++. Since C++ is faster language as compared to java, therefore in order to improve the processor intensive task, it is better to integrate C++ algorithms with java. By using C++, it provide an abstraction layer that can be generalized for reusability, code that is abstract and generalized can be produced. It is followed by UI development, JNI Integration and development of android app. After the development the application is to be tested via various testing methodologies to identify the loopholes followed by the deployment of the application among the targeted users.

Table 1.1 : Tasks & Deadlines

Tasks	Deadline	Description
Research & Findings	8/10/2022	• We will identify the problem • Needs to solve the problem • Formulate modules and tasks to be completed in specified timeline.
Planning & Designing	20/10/2022	• Implementation Plan, Analysis of feature • Designing the framework

Implementing C++ Algorithms	25/10/2022	• Implementation of sorting visualizers. • Implementation of Time Complexity Calculator. • Implementation of Stacks Module.
JNI Layer Implementation	5/11/2022	• Integrating C++ Algorithms with Java Application. • Importing the Algorithms in Code.
Designing User Interface	5/11/2022	• Designing Activities, Buttons, UI for displaying the outputs, home page etc.
Android App Implementation	20/11/2022	• Implementation of logic for taking inputs from the User. • Implementation of MP Android Chart for Visualization.
Testing & Validation	30/11/2022	• Unit Testing of Modules • Integration Testing of Modules • Memory Testing.
Final App Deployment	30/11/2022	• Deployment & Release of the app among targeted users.

1.4. Timeline

The task are scheduled as per assumed timelines and the maximum timeline which include estimated delay which may occur in completion of the particular module . It helps in clearly visualizing project management, project tasks involved. The project is divided into different tasks including research phase , planning and designing , Implementation / development and finally testing of the app followed by deployment. Below is the project timeline depicted with the help of gannt chart (chart used for project scheduling)

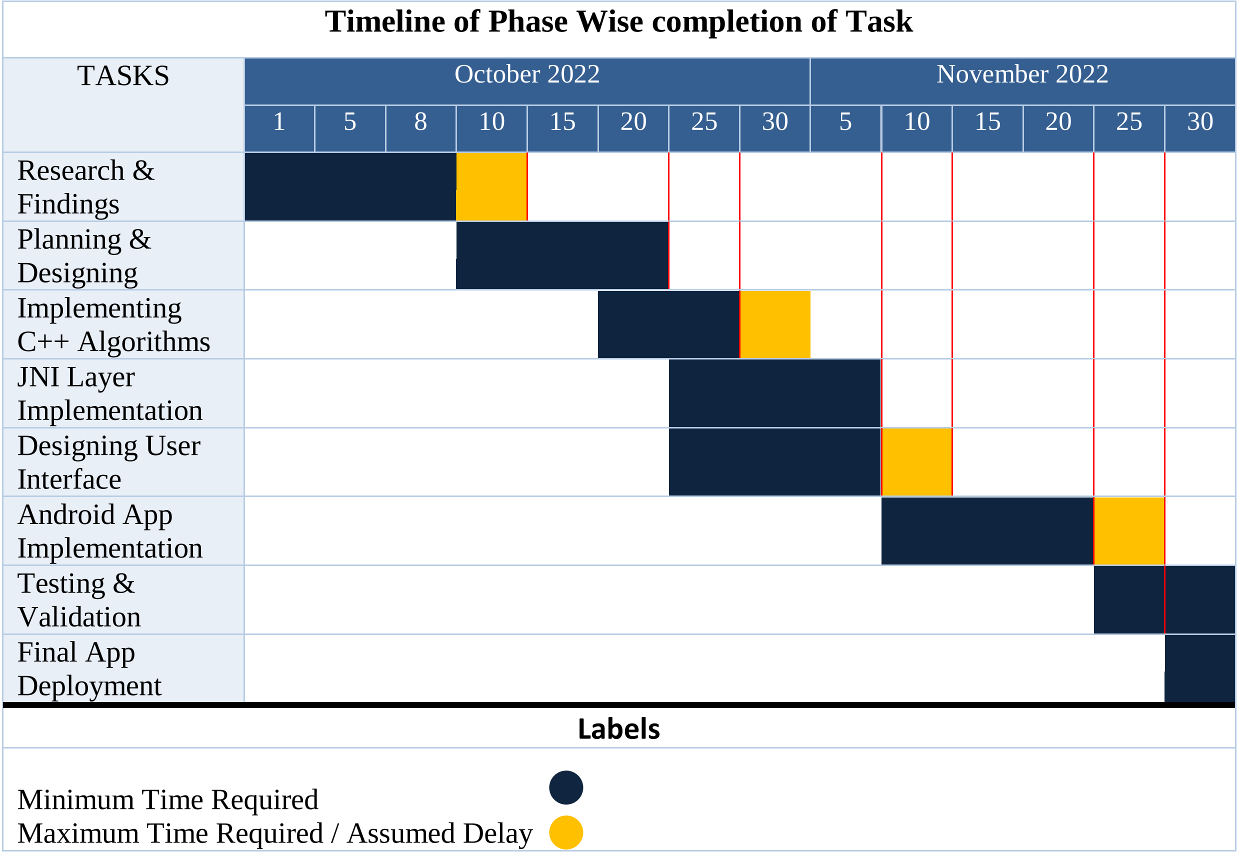


Figure 1.2. Gantt Chart

1.5. Organization of the Report

The following chapters will give an elaborate details of this project:

- **CHAPTER 2:** Literature review – This chapter will cover up the timeline of the reported problem, proposed solution, bibliometric analysis, review summary, problem definition and objective of the problem.
- **CHAPTER 3:** Design Flow or the design process - This chapter will cover up Evaluation & Selection of Specifications/Features, Design Constraints, Analysis and Feature finalization subject to constraints, Design flow, Design selection and implementation methodology.
- **CHAPTER 4:** Result analysis and validation - This chapter will cover up analysis, design drawings/schematics/ solid models, report preparation, project management, and communication, Testing/characterization/interpretation/data validation.
- **CHAPTER 5:** Conclusion and future work - This chapter will include the conclusion that should have expected results and the outcomes, derivation from the expected results and reason for the same, change in approach and suggestions for extending the solution.

CHAPTER 2
LITERATURE REVEW

2.1. Timeline of the reported problem

Learning is a basic necessity in life, since the birth till the end of the life. As a human, we learn to achieve the independence and learn to adapt to various changes in the environment. Nowadays, due to the phenomenon of the globalization and proliferation of the technology, education in the field of computer science in India becomes great demand for students. One of the favourite subjects in this field is Algorithm, which is now held courses as main part of the subject. The subject is basically studied by first grade students, as it as the basic knowledge to be implemented in this field. Students have to understand the algorithm clearly if they want to be a good coder. Visualization refers to the graphical representation of different aspects of the program which is specified in a conventional and textual manner.

Algorithms and data structures as an essential part of knowledge in a framework of computer science have their stable position in this field, since every computer scientist and every professional programmer must have the basic knowledge from this particualr area. With the increasing number of students in the field of computer science & engineering throughout the world in last decades, introduction of appropriate methods into the process of the education is also required. Our scope here is the higher education in the field of computer science. Appropriate methodologies can be used to attract students' attention during the lecture, explain concepts in visual terms, encourage a practical learning process, and facilitate better communication between students and instructors. Interactive algorithm visualizations allow students to experiment and explore the ideas with respect to their individual needs. Extensive studies on algorithm visualization effectiveness are available nowadays, and results are quite encouraging for those engaged in the field of computer science.

A significant number of systems were developed in the early 1990s for creating algorithm animations and visualizations. However, most of these are now no longer available, or so difficult to install and run due to changes in computer operating systems, that they are not currently a factor in education. If we consider the development of visualizations since the mid 1990s (i.e., Java), we have observed a decline over time in the creation of new visualizations, particularly after 2002.

Another *example* is the study with the goal to determine advantage of learning in interactive prediction facility which is provided by the courseware containing algorithm animations and visualizations of algorithms. Based on above mentioned aspects, results of studies are carried, as well as our own experiences and analysis, we consider algorithm visualization an important and needy area of further research and application of its results in today's computer science education. Sorting algorithm visualization, as part of the software visualization, could be described as "*graphical representation of an algorithm that dynamically changes as the algorithm iterates*". Even if the start of algorithm visualization date back into the era of 1940's, the greatest development in the area we could observe within the last span of 20-30 years.

Modern aspects to algorithm visualization were brought in the era of 1980's by the introduction of the BALSA (Brown & Sedgewick, Brown University, USA) system. Some of contemporary solutions include systems like ANIMAL, JAWAA, DAVE, JHAVÉ or JSAV.

2.2. Proposed solutions

This section composed of earlier proposed solutions that gained popularity among researchers & students in the field of computer science & engineering.

Algorithm Visualizers have a long history in computer science education as they are being used as an effective learning tool and have received increasing interest from both students and educators. There have been a lot of advancements in visualization. Since starting, hundreds of Algorithm Visualizers have been implemented and provided to the computer science education community.

ANIMAL, JAWAA, DAVE, JHAVÉ, JSAV and various other projects on smaller scales are previously built Algorithm Visualizers that gained popularity. As numerous visualization systems have been built over these years, the computer science education community has praised this form of learning and teaching. It is out of the scope of this paper to introduce each one. We will just get to know about some of the most popular and effective algorithm visualizer tools or systems.

W. C. Pierson et al. (1998) proposed JAWAA. One of the popular techniques used for creating and designing the algorithm visualizers in the early days is by annotating the algorithm code by using commands for scripting programming to generate the visualization. One of the popular Algorithm Visualizers that used this technique for visualization purposes is JAWAA which uses scripting language for generating animations of the data structure and algorithms by simply adding its commands in place of the output of the program. Since it is a scripting type language based on Java, a layman in the computer who has just started to learn may not be able to take the advantage it offers.

G. Rößling et al. (2000) proposed ANIMAL (A New Interactive Modeler for Animations in Lectures) , which is a general-purpose animation tool with a focus on algorithm animation. It is basically designed to overcome the drawbacks of JAWAA with animations and elimination of scripting.

T. L. Naps (2005) proposed JHAVE. JHAVE (Java-hosted Algorithm Visualization Environment), are designed and practiced for a very long time due to their capability to provide code changing of the algorithms. Although it is not a complete Algorithm Visualization tool, instead, it provides a supportive environment for various other visualizers to work. It is dubbed as an algorithm visualizer engine in the field of this work. Similar to ANIMAL, this tool is based on a desktop environment and needs the installation of software and different packages.

E. Vrachnos et al. (2008) proposed DAVE. Above discussed Visualizers had quirks and were not able to grow with the development of the industry. With the rise of browser and internet, Algorithm visualizers also started to adopt this trend and projects like DAVE, a web-based Algorithm Visualizer used to animate various algorithms that are applicable to the array data structure. It basically uses Java applets in order to run inside a browser that can support them.

V. Karavirta et al. (2015) proposed JSAV that uses Java and Java applets . The use of Java applets comes with its own challenges in modern web technologies. The support for Java applet has ended in major browsers and most of the new browsers does not allow the tools due to security and privacy. With modern technologies like HTML and JavaScript, it provides a system for programmers to design almost any kind of algorithm visualizer.

2.3. Bibliometric analysis

Analyzing research activity is an important aspect for the planning of future protective and adaptive the policies. This section is composed of analysis based on key features, effectiveness and drawback. Bibliometirc analysis can help business scholars to learn about the bibliometric methodology and to use that understanding to evaluate specific fields in the extant literature with large bibliometric data and corpus. It helps us to understand the limitations of existing solutions so that we can overcome these limitations & drawbacks in the new solution.

The following feature abbreviations is to be used for comparing features of various Visualizers which are highlighted in the above section. These features are useful for effective analysis and analyzing the drawbacks in the current solution. It will help us in formalizing the new solution based on the bibliometric analysis of the previously cited solutions.

Table 2.1 : Features for Bibliometric Analysis

Abbreviation	Feature
F1	Usability
F2	Platform Independence
F3	No Programming Limits
F4	Graphical Animations
F5	Fast & Responsive UI

We evaluated some of the existing and popular algorithm visualizers on the condition of having features that are stated in the previous sections. We have chosen and evaluated ANIMAL, JAWAA, DAVE, JHAVÉ & JSAV.

After evaluating these algorithm visualizers strictly on the condition of our list of features, we found some major problems that are necessary to overcome for us. Out of all these Algorithm Visualizers, only JSAV[5] is built with a modern technology framework. It uses the JavaScript programming language and runs on web browser. All of the others are using Java programming and run either on desktop environment or in browser using Java applets. In the past, Java was one of the best options available when comes to platform independence, but in modern computer era, we have more devices to support and the web browser is one such environment that is supported in most devices. Java applets are now not supported in many popular browsers that makes these Visualizers lack feature F2.

JAWAA (Java and Web-based Algorithm Animation) is also written in java and it provides an interface through which users can write their own animations as well as display these animations in a browser-based environment with the help of Java Applets. The commands are easy to understand but more sophisticated for the end-users as we need to use exact coordinates to plot the data in display frames to animate. It is old and not much work has been done to match the technological advancement since then. Therefore, it lacks features falling in category F1, F2, and F3.

The ANIMAL (animation in lectures) [6] is also a Java-based Algorithm Visualizer tool in which animations are generated using visual editors, scripting and API calls. It works using API calls which is a network-intensive task. This makes the process slow and delays the process of rendering the visualization. It is a visualizer system and needs scripting to perform visualization. Thus, because of the above-mentioned reasons, it lacks the features F2 and F5.

JHAVÉ (Java-hosted Algorithm Visualization Environment), works on Java in the form of plug-ins. It is mainly focused on increasing user engagement in the process of visualization. It has an animation engine that provides VCR-like controls to control the animation, information and pseudocode windows to view the needed information and actual code that is being rendered currently. In-between the user gets some questions as an exercise regarding the current algorithm from the education and learning perspective. It lacks features falling in category F1, F2, F3, and F4.

DAVE is one of the most updated from the pool of visualizers. It utilizes the features provided by the modern-day browser by integrating the tool with Web technologies. It gives the facility to experiment by doing modifications in the code and data of existing array algorithms in the project. It is written in Java and uses Java Applets to render visualizations in a web browser. One of the major drawbacks is that it provides too much control over the interactive options for a novice programmer.

The control over each bit of the visualizer serves as a challenge for a novice programmer who is not familiar with the basics of an algorithm, making the learning curve difficult.

DAVE heavily incorporates the idea of interactive exercises to learn a particular algorithm. It lacks features F1 and F2.

JSAV is built using modern web technologies like JavaScript and HTML5. It runs on modern age browsers, making it platform-independent. It lacks feature F3, as it is a visualizer system that is used by developers to build a visualizer but not by students. These researched visualizers focus on one or two features from the mentioned features list instead of implementing all the features to boost the process of learning through Algorithm Visualizers. Having listed features lacked by popular algorithm visualizers, we decided to create our own visualizer with an attempt to include all the listed features above.

2.4. Review Summary

Based on our findings, algorithm recognition can be seen as a useful supportive tool, used in addition to conventional educational methods in the field of computer science. Algorithms are an exciting way to be used in viewing. To visualize the algorithm, we simply do not enter the data into the chart; no primary database. Instead, there are sound rules about behavior. This may be the reason why algorithm recognition is rare, as designers try novel forms to better communicate. This is a good reason to read them. But algorithms are also a reminder that visualization is more than just a tool for finding patterns in data. Visualization enhances one's visual system to enhance one's intelligence: we can use it to better understand these important invisible processes, and perhaps other things. Graphics have been used for centuries to communicate information effectively among people and aid in the comprehension of complex information.

While many good algorithm visualizations are available, the need for more and higher quality visualizations continues. There are many topics for which no satisfactory visualizations are available. Yet, there seems to be less activity in terms of creating new visualizations now than at any time within the past ten years. On the other hand, the theoretical foundations for creating effective visualizations appear to be steadily improving. Collectively, the community is learning how to improve. While more fundamental research on how to develop and use algorithm visualizations is necessary, we also simply need more implementers to produce more quality visualizations.

2.5. Problem Definition

Based on analysis of existing solutions, we decided to start developing our own algorithm visualization platform which is intended to be used as a support tool within the subject Data structures and algorithms. The selection of topics within the scope of the subject is quite wide and it could probably be changed over the time. To cover the scope of the subject, probably more tools would be used, or quite big interventions to selected tool would be required. Considering the possible changes to the subject's design into account, we decided it will be better to start designing & developing our own solution. To reach the wide number of users we can select the language which is platform independent, fast and effective.

A running time complexity calculator is also needed in order to analyze the running time complexity of the algorithm, master's theorem is suitable for development of time complexity calculator. Many researchers have concluded that the students who are involved in the visualization process of the data structure and algorithms to understand the key working of any algorithm yield a better result in comparison to others who use classical methods because of the dynamic visual representation and interactivity of the sorting visualizers. The product will be more focused on students & researchers, yet it can be useful to other users also.

Apart from the functionality, the solution need to be implemented in such a way that it can reach a vast number of audience, thus it is important to analyze requirements carefully before building the final solution. In today's world more than 6 billion users have a smartphone, thus implementation of the solution as a mobile application would be a great choice to reach a wide number of audience. In Chapter 3 we will discuss in detail the features require to build the solution based on existing technologies and demand of the users & finally implement the solution based on the feature finalization.

2.6. Goals & Objectives

We drafted some main features that a Sorting Visualizer should have in order to achieve the goal effectively. These features are selected based on various parameters which include the currently existing technologies, user's perspective, cost, availability of resources etc. Those features are the following:

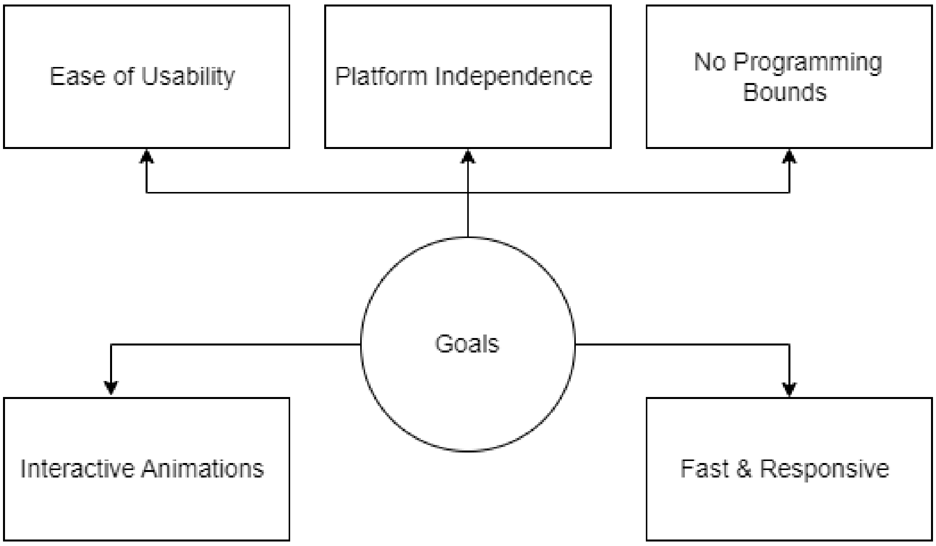


Figure 2.1. Goals

Ease of usability for novice programmer

It should be easy to use even for a layman in the computer field, thus the design should be made as per the users point of view i.e. easy to navigate, use, directed with instructions, clean user interface etc.

Platform Independence

In the modern world, the software has to be platform independent in order to reach a mass audience.

No programming bounds

An ideal sorting visualizer should not require programming from the end-user to visualize a particular algorithm. Thus the user must be presented with a clean User Interface which is accessible and operational without any manual coding.

Interactive animation

Users should be given control of the animation for better engagement. For example different users take variable amount of time to understand and gain the concept via visualization.

Fast and responsive

It must not lag or slow in the working. It should also be small in package size to load up faster. The target application size is 5 MB. In order to make application faster we are using C++ in the backend, integrated with Java Native Interface (JNI).

CHAPTER 3.

DESIGN FLOW

3.1. Evaluation & Selection of Specifications/Features

3.1.1. Selection of Application & System Programming Language

In order to make our solution efficient and Fast we need to first decide which programming language is to be used for system programming and which language is to be used for application programming. In order to provide code reusability and faster execution of code C++ is the better option and for the application programming side java is suitable. Java is an interpreted language thus it takes more time to compile as compared to C++ as shown in the *figure 4*, therefore it is advised to use C++ for system programming. Since we have developed our application using java it's quiet a challenging task to integrate both the languages as C++ is platform dependent & java is platform independent.

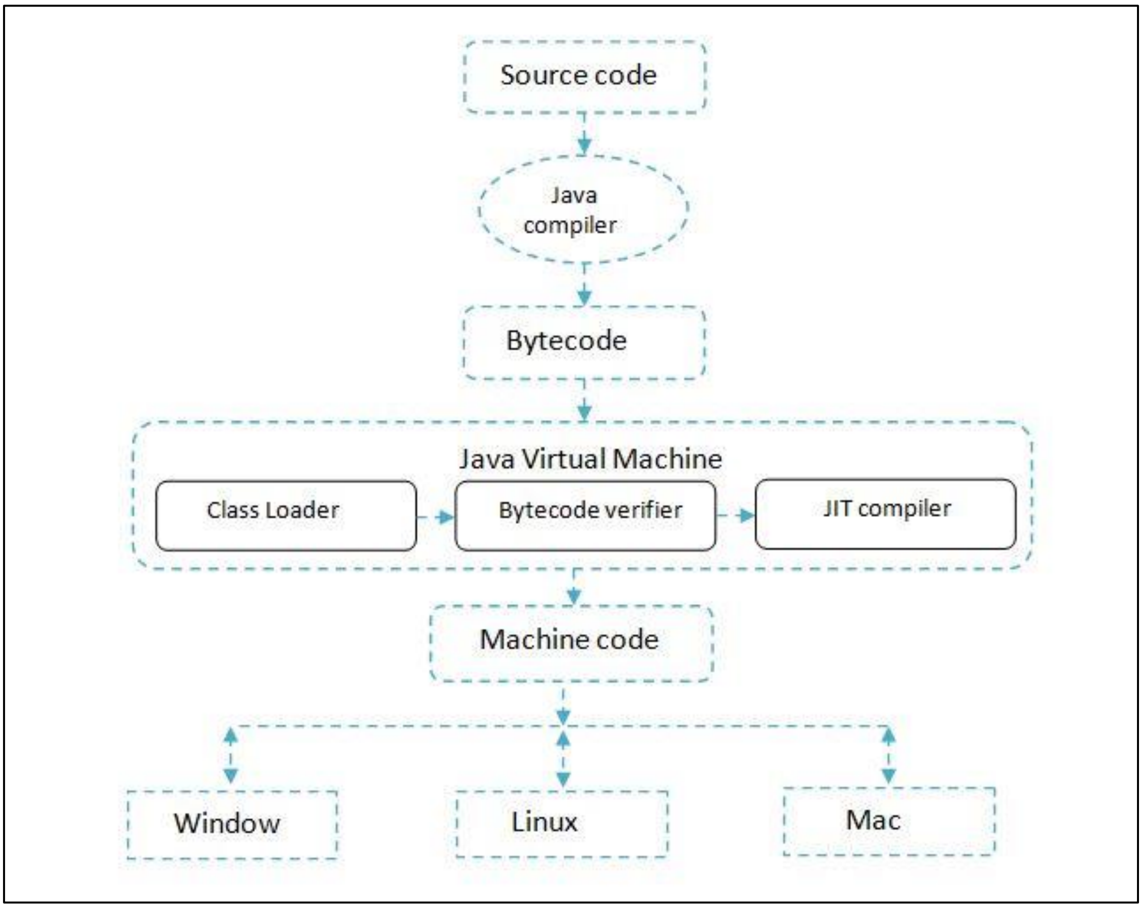


Figure 3.1. Java Code Execution

3.1.2. Features to support C++ & Java Integration

Development of the Android Application using both C++ and java is quiet tricky as both are opposite in terms of platform independency. Java is platform independent and executed when the source code is converted into the byte code whereas C++ does not need all these stuffs to compile. In order to achieve easy and clean integration of both the languages we need to identify the most optimal features. Thus Android NDK (Native development kit) & JNI (Java Native Interface) will be the most optimal feature in order to achieve this task.

- **Android NDK**

  The Android Native Development Kit (NDK) is a collection of tools that allows you to use C++ programming with Android, and provides libraries which you can use to manage the native activities and control physical device components, such as sensors and the touch input. The NDK tools may not be appropriate for most of the novice Android programmers who want to use only Java code in order to develop their applications. However, the NDK can be very useful for many cases in which you need to do the following things: Squeeze out the extra performance out to achieve low latency or to run computationally intensive applications, Reuse your own or other developers' C or C++ libraries.

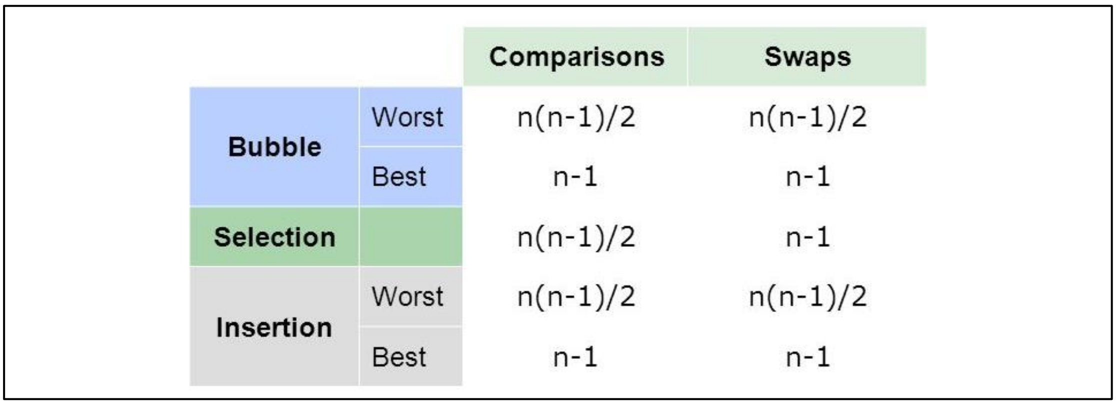
- **JNI (Java Native Interface)**

  Android NDK is a just collection of tools such as debugger or CMake, and Java Native Interface does all the things on the interaction between the Java & the native code. It act as an interface between Java, applications and the libraries that are written in other languages. The mostly interaction with the native code is calling functions written in C++ from Java & vice versa.

3.1.3. Sorting Algorithms for Visualization

In order to demonstrate the sorting visualization we have used three in place sorting algorithms : selection sort, insertion sort & bubble sort. An in-place sorting algorithm is an algorithm that do not need an extra memory space and produces an output in the same allocated memory that contains the data by transforming the input data ‘in-place’. However, a very small constant extra memory space used for variables are allowed.

- **Selection Sort :** Sort an array by repeating pick up the minimum element then swap to the current index. It has Time Complexity: O(n^2) & Auxiliary Space: O(1)
- **Insertion Sort :** Starting from the second element of the array, compare the current element (key) with all the previous elements. If the key element is smaller, then move the larger element up one position till there is no larger element, & then put the key element in that location.
- **Bubble Sort:** Starting from the first element, the two groups check whether they need to be exchanged until all the elements are moved to the correct positions, no more exchanges are needed. As it continuously moves the larger one towards the next position and then make comparison with the next element, for every loop, it puts the maximum element at the end of the array data structure, and so on.



		Comparisons	Swaps
Bubble	Worst	n(n-1)/2	n(n-1)/2
	Best	n-1	n-1
Selection		n(n-1)/2	n-1
Insertion	Worst	n(n-1)/2	n(n-1)/2
	Best	n-1	n-1

Figure 3.2. Comparisons & Swaps in Sorting

Table 3.1: Time Complexity of Sorting Algorithms

Sorting Algorithm	Time Complexity			Space
	Best	**Average**	**Worst**	**Worst**
Bubble Sort	O(N)	O(N2)	O(N2)	O(1)
Selection Sort	O(N2)	O(N2)	O(N2)	O(1)
Insertion Sort	O(N)	O(N2)	O(N2)	O(1)

3.1.4. Feature to calculate running time complexity

There are basically three cases for evaluation of the efficiency of an algorithm, i.e., the average case, worst case and best case. We included these scenarios for the algorithms to work upon. A worst-case occurs for an algorithm when it takes maximum time to complete in a situation or on a set of data. The Average Case complexity is the closest to the expected result in the real world. It is calculated by taking all the possible inputs and taking the average of the time and space taken. Like the unsorted array is given input to the bubble sort to implement. Bubble and Insertion sorting algorithms behave alike with their worst-case scenarios. In best case situations, they tend to process the array faster. This verifies the behavior of insertion sort for sorting sorted elements. Just like this, we can conclude that the Selection sort performance is similar in each of the three cases as shown in the comparison graphs.

- Thus with the help of time complexity calculator we can easily calculate the time complexity of the algorithm using the time function.
- This can be achieved with the help of masters theorem. Master's Theorem is one of the best method to find the algorithm's time complexity quickly from its recurrence equation. This theorem can be applied to the decreasing functions as well as the dividing functions, each of which we'll be looking into detail ahead.
- Further, it covers the important scenarios that cover the calculation being involved in the Master's Theorem Algorithm. Master's Theorem is made in such a way that it is easy for the reader by explaining the proofs and solving examples for both dividing functions and decreasing functions.
- Every Theorem or technique comes with its underlying drawbacks and limitations, which are important to understand so that the loopholes can be avoided.

3.1.5. Interface and functionality

As simplicity and ease of use are one of the most necessitated features for an algorithm visualizer. We tried to develop an application which is user-friendly by having modern looking design and easy to understand controls. On the home screen of the application, there is one button which routes us to the main page. Both the pages have similar user interfaces for a unified experience.

The user interface consists of a visual representation of data stored in the array data set. This data is represented in form of bars with their height corresponding to the value of the element. Height is one of the most easily recognized factor for differentiating. This representation section is called Animation Arena and it is the main section of the application to visualize the process of sorting. We will achieve this graphical visualization using MP Android Chart. Animation is updated frame by frame, in which each frame represents the current state of data in the array after the current step of processing in an algorithm. The different algorithms available are Bubble sort, Insertion sort & Selection sort. The number of elements can be changed by the user by changing the value in the input field. After completion of sorting, the animation will stop and animation area will contain sorted bars with increasing order of height. Each bar will represent an element in the array that is used internally for sorting. By this simulation, a learner can understand how each algorithm has different processing for an equal amount of random data and also understand about its time and space complexity using the masters theorem calculator.

MP Android Chart : If you are looking for a User Interface component to represent a huge form of data in readable formats then you can think of displaying this data in the form of interactive bar charts or bar graphs. It makes it easy to visualize and read the data with the help of animated bar graphs. MP Android Chart is a powerful Android chart view library, supporting line, bar, pie, radar, bubble and candlestick charts as well as scaling, panning and animations.

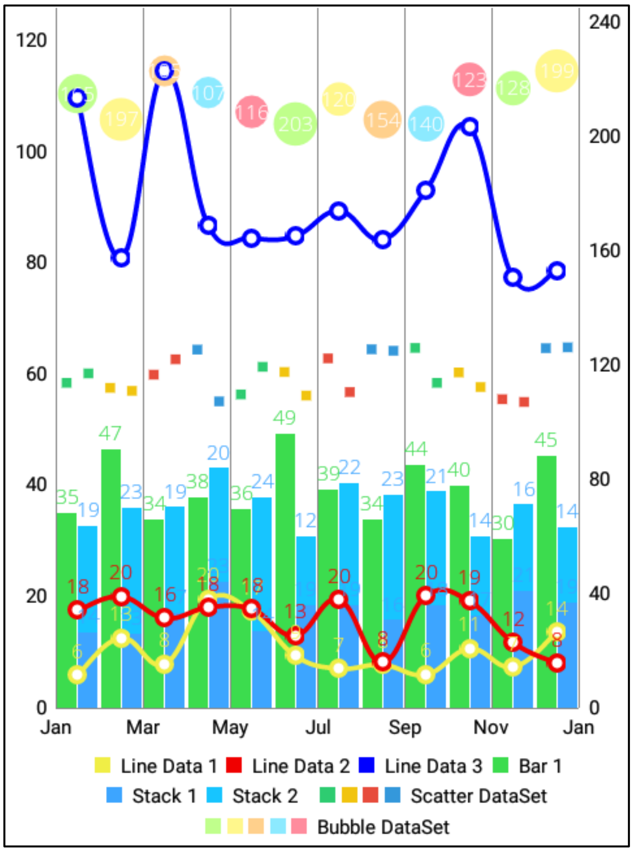


Figure 3.3. MP Android Chart Example for Bubble Sort

3.2. Design Constraints

Before the analysis of the design constraint we first need to identify the requirement specification based on which we formulate our design constrains. Below is the requirement specification which contains all the software and hardware requirements required to build the solution. After that we can identify the design constraints.

Design constraints are basically imposed on the design of the solution. These constraints are typically imposed by the client, by the organization, or by the external regulations. The constraints may be imposed on the software or hardware, on the data, operational procedures, user interfaces, or any other component of the system. In the table 6 design constraints are highlighted with respect to the parameters: Cost, Reliability, Functionality, Usability, Safety & Ethical constraints.

Table 3.2 : Requirements

Requirement Specification		
Software Requirements	**Dev C++** To implement Sorting Algorithms, Complexity Calculator, Stack Modules in C++ **Android Studio and NDK(Native Development Kit)** To build an android application that privatises users' data by adding noise to it. **Others** Any suitable command prompt, git, android emulator, JDK (Java Development Kit), rustup 1.24.3	
Hardware Requirements	Operating System	Windows 7 or above
	Processor	Intel(R) Core(TM) i3-6006U
	CPU	2.00 GHz
	System type	64-bit operating system, , x64-based processor
	RAM	4 GB

Table 3.3: Design Constraints

Parameters	Constrains Applied
Cost & Economic Constraints	System overall budget must not go beyond 1k excluding the application deployment charges. Overall size of the software should not go beyond 10MB to make sure the applications does not consume too much of RAM.
Reliability	System should be able to withstand 1000 request daily since on an average there could be a 1000 request in a medium sized organization.
Functionality	Interface should be user friendly to ensure that even a novoice user can easily use it.
Usability	System should be deployable in any type of environment & must be supported in the android version which is used by more than 90% of the customers all over the world.
Safety & Ethical Constraints	No personal details of the user to be recorded and stored in any manner, other data should be well hidden from regular employees and should be accessible upon request only since we value and respect our customer's privacy.

The following standards will be used for developing the application:

- **ISO/IEC 14882:2020** is a standard which specifies the requirements for implementations of the programming language, specifically C++.
- **ISO 16757-1:2015** is based on the provision of data structures for electronic products which is used to build the services and applications.
- **ISO/IEC 19503:2005** is a standard to enable easy interchange of metadata using XML(Extensible Markup Language). It is also a W3C Standard.

3.3. Analysis and Feature finalization subject to constraints

3.3.1. Minimizing footprint of JNI layer

We must try to minimize the footprint of your JNI layer. There are several things to consider here. Your Java Native Interface (JNI) solution should try to follow these guidelines in order, where the first one is the most important following the next.

You must following these policies before the implementation of JNI in order to minimize the JNI footprints and make your layer tightly coupled as much as possible with the application which allows for more faster execution:

- **Minimize the marshalling of the resources in the JNI layer**
  Marshalling across JNI layer has non-trivial expenses. Try to develop an interface that minimizes the amount of data you need in order to marshall and frequency with which you must marshall the data.
- **Avoid asynchronous syncing between code written in a managed language and code written in C++ language when possible**
  This will help to keep your JNI interface easier to maintain. You can basically simplify asynchronous User interface updates by keeping the async update in the same language as the User Interface. Instead of invoking a C++ function from the User Interface thread in the Java code via JNI, it is better to do a callback between two types of threads in the Java language, with one of them making a blocking C++ call and then notifying the User Interface thread when the blocking call is complete.
- **Minimize the number of threads that need to touch or be touched by JNI**
  If you do need to utilize thread pools in both the Java and C++ languages, try to keep JNI communication between the pool owners rather than between individual worker threads.
- **Keep your interface code in a low number of easily identified C++ and Java source locations to facilitate future refactors**
  Consider using a JNI auto-generation library as appropriate and required in your application, thus it will help in lowering your interface code.

3.3.2. Analysis

This application is developed to adapt all the listed features in previous sections. We have used a new and different approach by using modern visual design curated for the user of the modern era. As a development platform for the project was selected Java as a system programming language as it is platform independent, ensuring high portability, popular in mobile development and very good support by available tools, libraries, etc. We have used bars to represent the different data values present in the data structures. This helps in visually differentiating each element present in the data set. The application is made easy for the end-user to use.

It has been toned down to an extent where a user who has little or no knowledge can also operate and understand the visualization process. It can be easily controlled by save and iterate buttons from the control panel. It uses modern mobile technologies for platform independence by using Java and Android NDK. The uprising of NDK and mobile technologies with increasing capabilities of modern device graphics have enabled the developers of these algorithm visualizers to build sophisticated applications or tools for mobile devices.

In the modern world, smart devices like mobiles and tablets are also becoming capable enough to execute such complex tasks as graphics rendering and running applications. Our Visualizer provides an interactive visualization of different sorting algorithms using MP Android Chart with manual control by the user using iteration buttons. We have introduced a time complexity calculator which is to be implemented using master theorem to calculate the running time complexity.

We have to first calculate the time function of the algorithm and provide the time function as input in the calculator and the user will be presented with the desired complexity. Our Solution does not require a user to code in any form. It presents a graphical user interface with control buttons to interact. This makes it easy to use and free from code anxiety for novice programmers. It completes the features F1, F2, F3and F4.

Our solution tends to have a very small package size and requires very small amount of RAM to execute which completes the feature F5. Our solution also allows the users to control the amount of delay between two animation frames or algorithm steps. This helps users to visualize the animation at a speed that suits to their preference i.e. manual control over the animation.

3.3.3. Feature Finalization

Finalized features/ specification required to implement the solution is highlighted in the Table 7 along with the desired tech stack used to build the solution. It is necessary to clearly identify the features & requirements before the solution implementation as it helps us to design the overall design of the system with approximate cost and stipulated timeline. These features are identified based on the analysis of previously discussed solution in order to overcome their limitations and develop the new solution.

Table 3.4: Feature Finalization

Feature	Technology Used
System Programming Language	C++
Application Programming Language	Java
Supported features for integration of java & C++	JNI(Java Native Interface) , NDK (Native development kit)
Sorting Visualizers	C++, MP Android Chart
Time Complexity Calculator	C++, Master's Theorem Implementation
User Interface & Functionality	Android Studio, Java, XML, C++

3.4. Design Flow

3.4.1. Data Flow Diagram

In Software engineering DFD(data flow diagram) can be drawn to represent the design of different levels of data abstraction. Higher-level DFDs are being partitioned into low level of abstraction.

It maps out the flow of data for any process of the system. It uses defined symbols like rectangles for input and output, circles for functionality and arrows, plus short text labels, storage points and the routes between each destination. Data flow graphs can range from simple, even hand-drawn design overviews, to in-depth, multiple level of DFDs that dig progressively deeper into how the data is being handled.

DFD levels are numbered as 0, 1 or 2 level, and sometimes go to even Level 3 or beyond based on the complexity of the software design. The necessary level of detail depends on the project scope of what you are trying to achieve. Here, we will see mainly 3 levels in the data flow diagram based on the complexity of the project, which are: **0-level DFD, 1-level DFD, and 2-level DFD.**

DFD Level 0 (Context Diagram):

DFD initial level is also called a Context Diagram. It is a basic overview of the whole system being analysed or modelled. It is designed to be a low level view, showing the system as a single high-level process, with its inter-relationship to external entities. It must be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers. Basically it is represented using one function block with a circle with basic input and output.

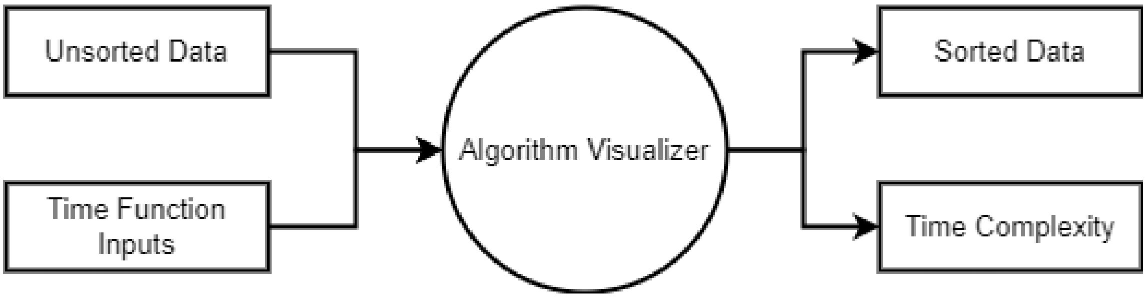


Figure 3.4. DFD Level 0

DFD Level 1 :

DFD Level 1 abstraction provide a more detailed division of pieces of the Context Level Diagram. You will show the main functions carried out by the software, as you break down the higher-level process of the Context Diagram into its sub component processes. The objective of a DFD is to show boundaries of a system/ process as a whole.

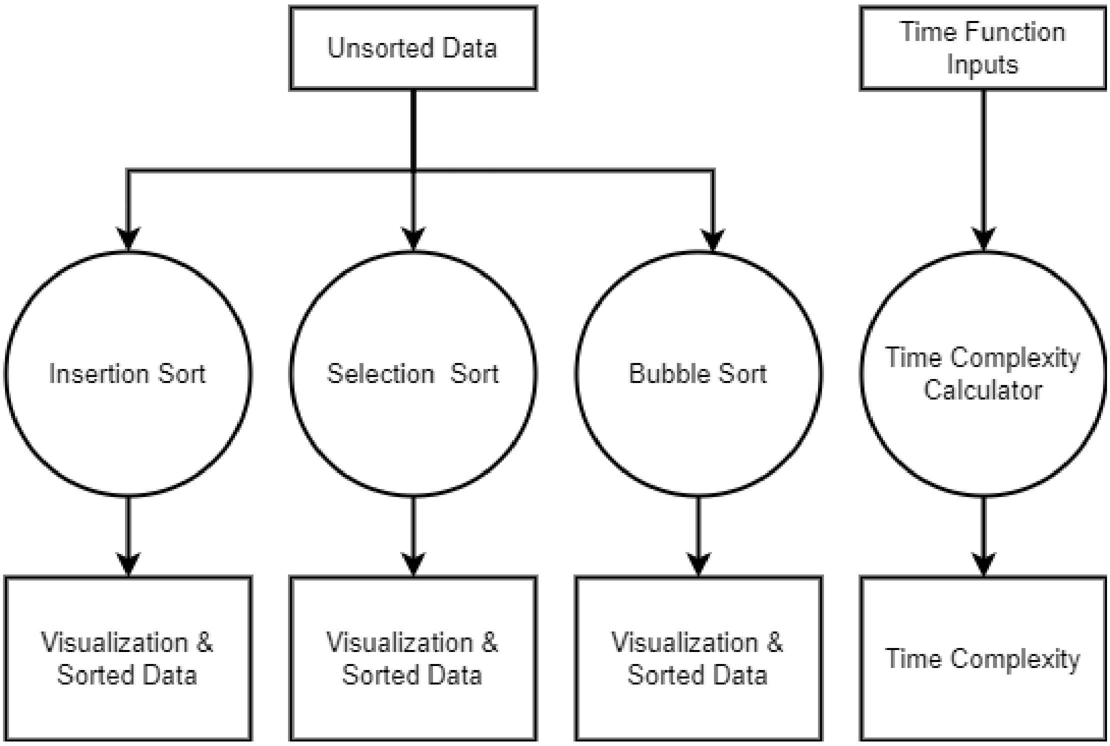

Unsorted Data
Time Function Inputs
Insertion Sort
Selection Sort
Bubble Sort
Time Complexity Calculator
Visualization & Sorted Data
Visualization & Sorted Data
Visualization & Sorted Data
Time Complexity

Figure 3.5. DFD Level 1

DFD Level 2 :

Second level DFD goes one step deeper into parts of first-level DFD. It can be used to specify or record the necessary detail about the system's functionality. As you can see in the Figure 15 how level 1 is being decomposed into further level of complexity in the level 2 diagram. Data flow shows the transfer of data from one part of the software component to another. The flow should have a name identifier that determines what information is being moved.

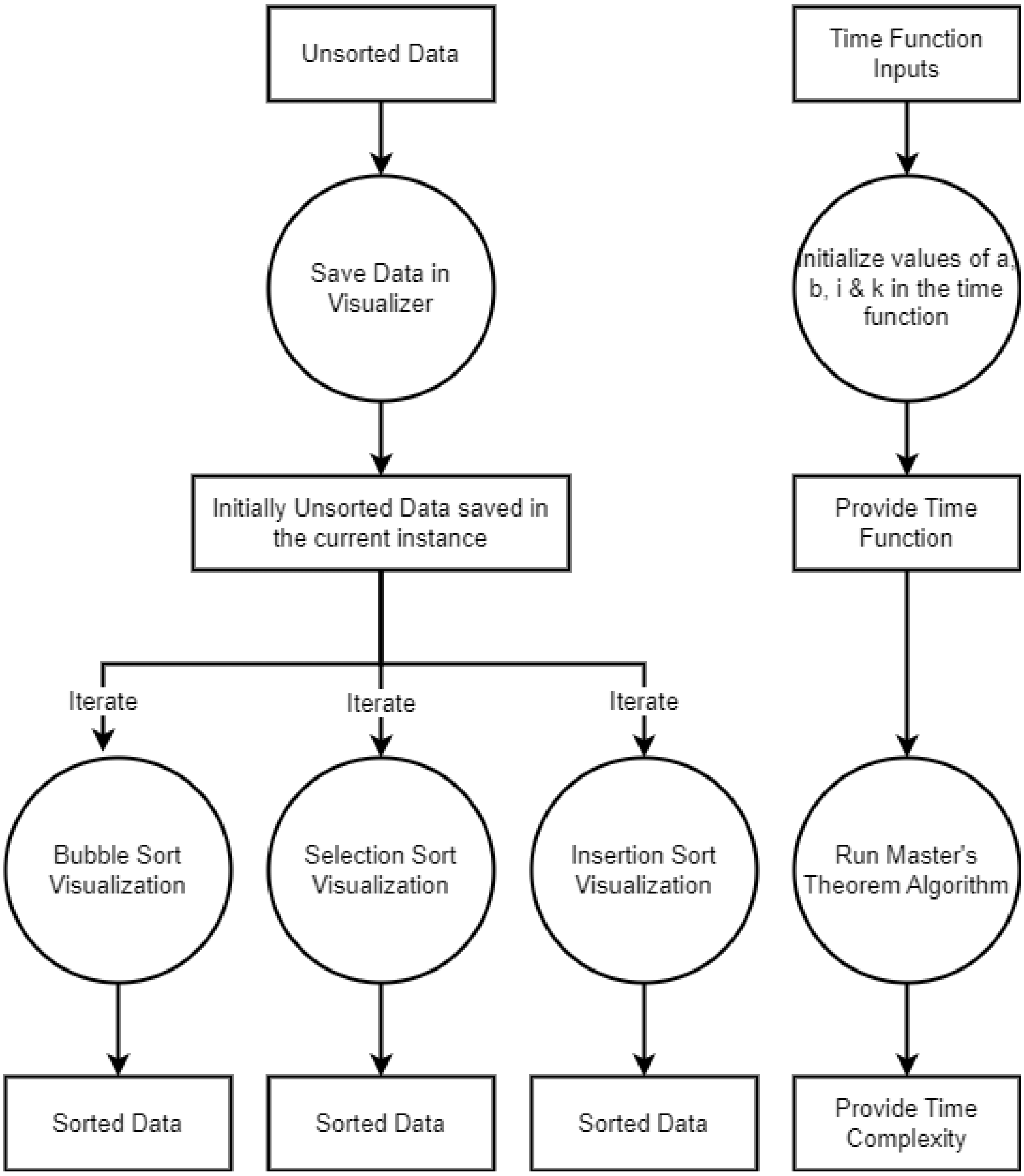

Unsorted Data
Save Data in Visualizer
Initially Unsorted Data saved in the current instance
Iterate
Iterate
Iterate
Bubble Sort Visualization
Selection Sort Visualization
Insertion Sort Visualization
Sorted Data
Sorted Data
Sorted Data
Time Function Inputs
Initialize values of a, b, i & k in the time function
Provide Time Function
Run Master's Theorem Algorithm
Provide Time Complexity

Figure 3.6. DFD Level 2

3.4.2. Activity Diagram

Activity diagram is a UML diagram suitable for modelling the activity flow of the process/ system. Activity diagram also captures these processes and describes the flow from one process to another. This specific usage is not available in other type of diagrams. These systems can be a database, external stacks/queues, or any other type of system. Typically, an activity needs to be achieved by some operations, particularly where the desired operation is intended to achieve a no. of different things that require coordination, or how the activity in a single use case relate to one other.

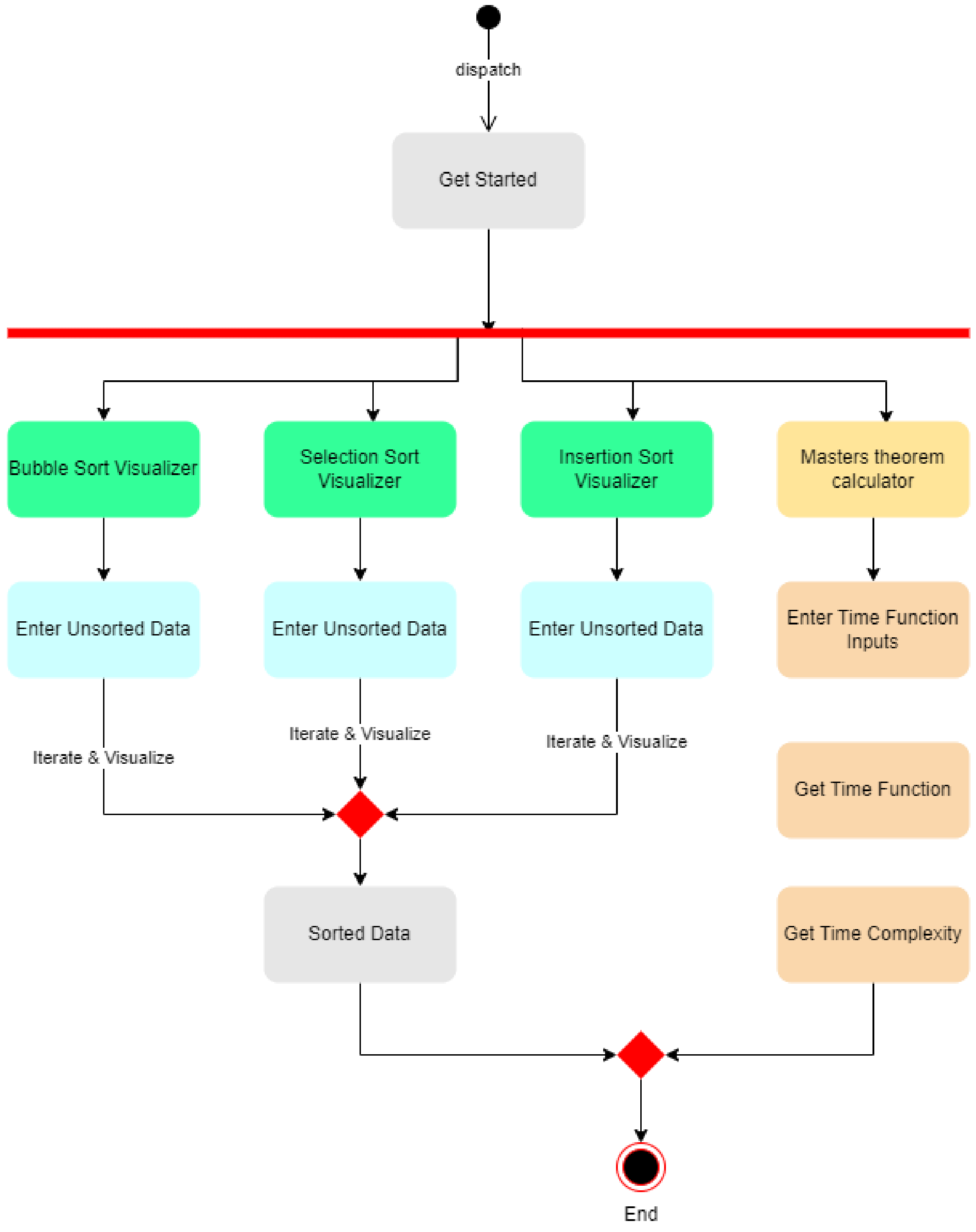


Figure 3.7. Activity Diagram

3.5. Design selection

From the initial R&D phase to the final launching stage, there are many things that simultaneously occur while building a product. And one of the major decisions that need to be taken care before the actual development is what model to use. When it comes to **android app development**, the way you approach an application idea has a great impact on how the outcome would be. Different types of application development methodologies are existing in the market, prevailing based on different factors such as application requirements, features, & the direction of the workflow.

We have discussed various designs in the above section, based on the design and analysis of the system we have to select the model for the development. In this section we will compare various software development models and based on the project design we will finalize our model. When it comes to selecting the best android app development method, the two best approaches in the market are **Agile and Waterfall.**

- **Waterfall Mobile App Development Approach**
  Waterfall methodology is one of the earliest and sequential app development lifecycle model. In this application development approach, the project is being divided into different phases which include initiation, analysis, design, development, testing, and deployment, such that one cannot move to the next phase without completion of the existing one. Meanwhile, there's no possibility of the overlapping of phases in this model. In this development approach we don't have ambiguous requirements i.e. requirements are fixed. This model is not suitable for the development of large applications and suitable for only small scale applications.

- **Agile Application Development Approach**
  Agile method of development is an iterative, rapid application designing approach that involves a more 'time-bound', team-based and a sprint action style. According to the top app development businesses, this strategy emphasizes on being lean and creating MVPs over a stipulated period of time while to enhance each particular iteration. This model is suitable for the large scale application development where the requirements keeps changing based on user's demand, and we keep iterating the product.

3.5.1. Comparison

In this section we will compare the benefits and disadvantages of the waterfall and the agile model and based on the comparison we will finalize our model which obeys to the previously formulated system designs.

Table 3.5: Benefits of agile & waterfall model

Waterfall Model	Agile Model
• The first benefit of the Waterfall model approach is that it is quite simple and easy to develop and implement, without much changes in the requirements of the product. • Since application development phases are processed and completed one at a time, therefore it is quite easier to determine the deliverables & manage the rigidity of the design. • As explained in the last point, it is not possible to move to the next level of development phase without completing the current existing one. • Waterfall model approach enables one to easily get an estimate of application development cost structure and timeline. Thus it concludes that this model is best suited for small scale mobile application with fixed number of requirements of the users.	• The agile app development approach emphasizes on regular communication and teamwork, which maintains the workflow & enables the development team in delivering a result-oriented application. • Testing is being done at the end of each phase that means an earlier encounter with the bugs & solving them. At last, you get a high-quality application with the agile development process. • The app goes in the hands of the users much earlier, even before the launch of the product. This is helpful in order to determine their response towards the app , which reduces the chances of building an app nobody wants to try. • The short app. development cycles increase the flexibility of the process but if the requirements keeps changing the no. of cycles increases which increases reiteration. This reiteration may cause delay in the development of the desired product.

Table 3.6: Disadvantages of agile & waterfall model

Waterfall Model	Agile Model
• One of the biggest drawback of Waterfall mobile app development method is that you cannot return to the previous phase till the whole life cycle is completed. That means, if the market or customer needs changes in between, you won't be able to update an application until it's fully ready. • Since the testing phase is being performed at the end of the development, it is likely that you might find it difficult to tackle bugs that would have been easily removed if found at the initial level. • Without a working prototype with functioning, the users may not be able to figure out what exactly they want in the application. Since the requirement gathering is the first step of the methodology, this results in the risk of missing some crucial details.	• As Agile model focuses on active team involvement and direct interactions, it is necessary that the whole team is committed to the product Otherwise, the project will take longer than estimated time. • This technique prefers working software over a detailed design documentation. This is good to some extent, but, the developers have to maintain the right balance between code and documentation. • Since the Agile development methodology is associated with frequent reprioritization, the project might not get delivered within the estimated time frame, unless the right mobile app development organizations you are working with has strong experience with the methodology.

3.5.2. Selection of Appropriate model based on analysis

Both the application development approaches are efficcient to consider for one or the other type of situation. So, the best way is to decide the right option and to look into the requirements first. Based on that we had selected **Waterfall Model for development** because of the following reasons.

- App requirements are clear, definite, and well-documented. Waterfall relies on teams following a sequence of steps & not moving forward until the previous phase have been completed.

- The concerned technique is well-understood & is not dynamic. Waterfall's approach is highly methodical approach, therefore it should come as of no surprise that the methodology emphasizes a clear and clean transfer of information at every step.
- There must not be any type of ambiguous requirements in the project.
- Ample number of resources with requisite expertise are available. When applied in a software setting, each new step involves a new group, & though that might not be the case at your company, you still should aim to document the information throughout the project's lifecycle.
- The project is short with finite number of modules.
- One of the definite steps of Waterfall model is committing to an end product objective, or deliverable at the beginning, & teams must avoid deviating from that type of commitment.
- In Waterfall model development, very less interaction of the end users / customers is involved during the development of the product. When the product is completely ready then only it can be demonstrated to the customers/ end users.

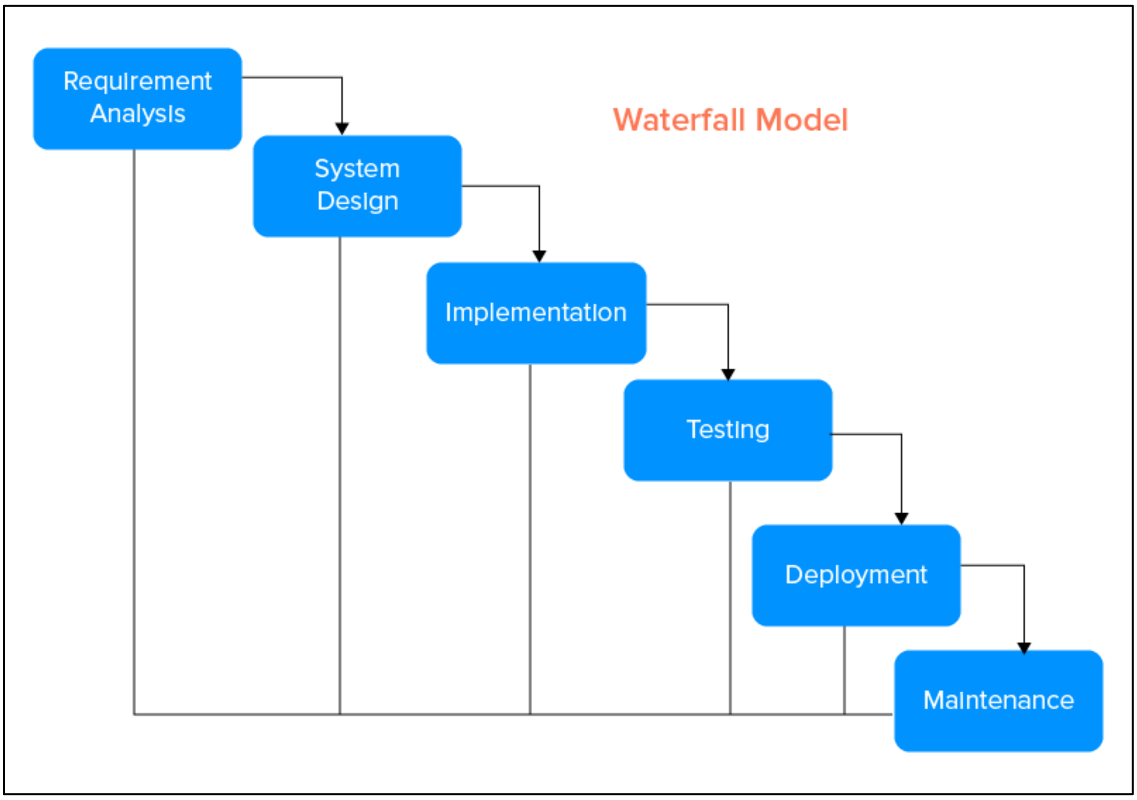


Figure 3.8. Waterfall Model

3.6. Implementation plan/methodology

The architecture design of a system shows how the system is used & how it interacts with other system components and the outside world. It describes the interdependence of all the system's components and how the data is linked with those components.

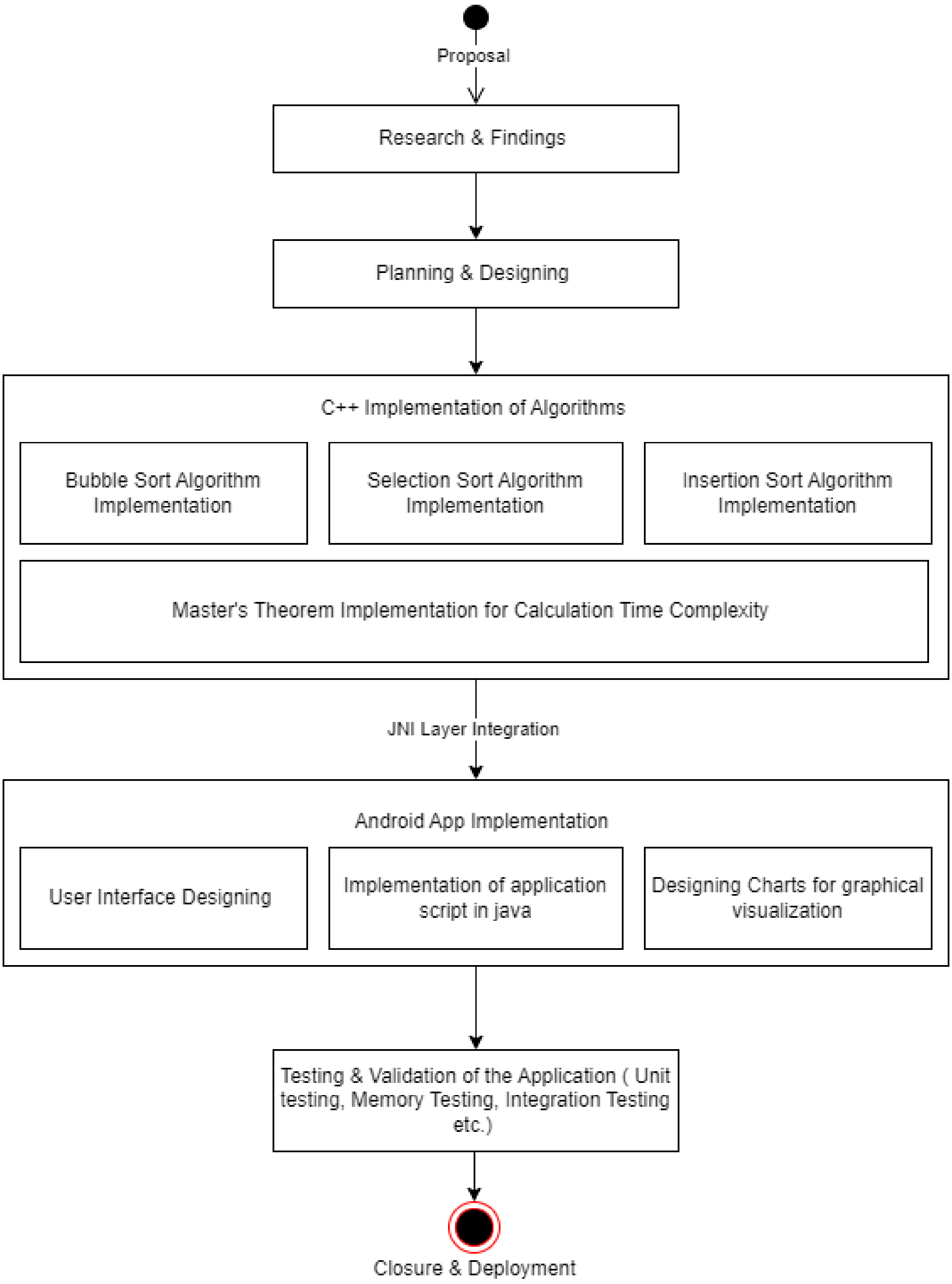

Proposal
Research & Findings
Planning & Designing
C++ Implementation of Algorithms
Bubble Sort Algorithm Implementation
Selection Sort Algorithm Implementation
Insertion Sort Algorithm Implementation
Master's Theorem Implementation for Calculation Time Complexity
JNI Layer Integration
Android App Implementation
User Interface Designing
Implementation of application script in java
Designing Charts for graphical visualization
Testing & Validation of the Application (Unit testing, Memory Testing, Integration Testing etc.)
Closure & Deployment

Figure 3.9. System Framework

3.6.1. Bubble Sort Algorithm

Bubble sort is an algorithm which repeatedly compares & swaps the adjacent data in every pass. In **i-th pass** of Bubble Sort, **last (i-1) elements are already sorted**, and i-th largest element being is placed at (N-i)-th position, i.e. i-th last position. If the input is previously sorted, no swaps are ever needed & Step 3(Arr [J] > Arr[J+1]) is executed n times. Below is the bubble sort algorithm which require n-1 comparisons:

Algorithm:

```
procedure bubbleSorting( list : array of items )
  n = list.count;

  for i = 0 to size of array-1 do:
    swapped = false

    for j = 0 to size of array-1 do:

      /* compare  adjacent elements */
      if list[j] > list[j+1] then
        /* swap them */
        swap( list[j], list[j+1] )
        swapped = true

      end if
    end for

    /*if no no. was swapped that means
    array is sorted now,  then break the loop.*/
    if(not swapped) then
      break

    end if
  end for

end procedure
return list
```

Optimization of Algorithm:

Look if there occurs any swapping operation in the inner loop or not. If there is no swapping in any of the pass, it means the array is now fully sorted, theefore no need to continue, and stop the sorting operation. So now we can optimize the no. of passes when the array gets sorted before the completion of all the passes. And, it can be detected that the given input array is sorted or not, in the first pass only.

Time Complexity:

- **Best Case** Sorted array as input. Or almost all elements are in proper place. [**O(N)**]. **O(1)** swaps.
- **Worst Case**: Sorted Reversely/ A few no. of elements are in the proper place. [**O(N^2)**] . **O(N^2)** swaps.
- **Average Case**: [**O(N^2)**] . **O(N^2)** swaps.

Space Complexity:

Since a temp. variable is used for swapping [auxiliary, **O(1)**]. Hence it's In-Place sort.

Advantage:

1. It is one of the simplest sorting approach.
2. Best case time complexity is of **O(N)** [for the optimized approach] while the array is almost sorted.
3. Using this **optimized approach**, it **can detect the if the array is already sorted in first pass** with time complexity of **O(N)**.
4. Stable soring do not change the relative order of elements with equal key elements.

Disadvantage:

1. Bubble sort is comparatively slower algorithm as compared to others.

3.6.2. Selection Sort

Selection sort algorithm selects the i-th smallest element & places at i-th position. This method divides the array into 2 parts: sorted (left part) & unsorted (right part) subarray. It selects the smallest element from right part and places in the first position of that subarray (in ascending order). It selects the next smallest element repeatedly. This method is not suitable for large array as its average & worst case complexities are of O(n2), where n is the number of items. Steps:

Step 1 − Set MIN to location= 0

Step 2 − Search the min. element in the arry

Step 3 − Swap with key at location MIN

Step 4 − Increment MIN to point to next key element

Step 5 − Repeat till array is sorted

Algorithm:

```
procedure selection sort
   list  : array of items
   sizelist     : size of list

   for i = 1 to sizelist – 1 do:
   /* set current element as min*/
      min = i

      /* check the element to be min */

      for j = i+1 to sizelist
         if list[j] < list[min] then
            min = j;
         end if
      end for

      /* swap min element with  current element*/

      if indexMin != i  then
         swap list[min] and list[i]

      end if
   end for

end procedure
```

Time Complexity:

- **Best Case** [**O(N^2)**].& **O(1)** swaps.
- **Worst Case**: Reversely sorted, & when the inner loop makes a maximum number of comparison. [**O(N^2)**] . Also, **O(N)** swaps.
- **Average Case**: [**O(N^2)**] . Also **O(N)** swaps.

Space Complexity: [auxiliary, **O(1)**]. In-Place sorting(where elements are shifted instead of being swapped (that is temp= a[min], which results in shifting elements from a[i] to a[min-1] one place up & then declaring a[i]=temp).

Advantage:

1. It can also be used on data structures that make add & remove efficient, such as a linked list data structure. Just remove the smallest element of unsorted list & end at the end of sorted list.

2. The number of swaps reduced. **O(N)** swaps in all cases.
3. In-Place sort.

Disadvantage:

1. Time complexity in all cases(best, average, worst) is **O(N^2)**

3.6.3. Insertion Sort

Insertion Sort algorithm is a simple comparison based sorting method. It inserts every array element into the proper location. In the i-th iteration, previous (i-1) elements (i.e. sub-array A[1:(i-1)]) are already sorted, and the i-th element (A[i]) is bring inserted into the proper place in the previously sorted sub-array. This method is not suitable for large data structure sets as its average and worst case complexity are of O(n2), where n is the no. of items.

Algorithm:

```
procedure insertionSort( A : array of integers )
  int holePos
  int valueToInsert

  for i = 1 to length of (A) do:

    /* select value to inserted */
    valueToInsert = A[i]
    holePos = i

    /*locate hole position for the element to be inserted */

    while holePos> 0 and A[holePos-1] > valueToInsert do:
      A[holePos] = A[holePos-1]
      holePos= holePos -1
    end while

    /* insert the number at hole position */

    A[holePos] = valueToInsert

  end for

end procedure
}
```

Time Complexity:

- **Best Case** [**O(N)**] & **O(1)** swaps.
- **Worst Case**: Reversely sorted, & when inner loop makes maximum no. of comparison, [**O(N^2)**] & **O(N^2)** swaps.
- **Average Case**: [**O(N^2)**] & **O(N^2)** swaps.

Space Complexity:

[auxiliary, **O(1)**]. In-Place sort.

Advantage:

1. It can be easily computed.
2. Best case time complexity is **O(N)** when the array is already sorted.
3. Number of swaps reduced as compared to bubble sort.
4. For smaller values of N, insertion sort performs more efficiently like other quadratic sorting methods.
5. Stable sort.
6. Adaptive: total no. of steps is reduced for partially sorted array.
7. In-Place sort.

Disadvantage:

1. It is used when the value of N(array size) is small. For **larger values of N**, it is **inefficient**.

3.6.4. Master's Theorem Algorithm

Masters theorem is the most widely used method to analyse or compare various algorithms by computing their recurrence & time complexities.

Recursive functions call themselves in their function body. It may get complex if we start computing its time complexity function by other commonly used simpler methodology. Master's Theorem method is the most useful & easy methodology to calculate the time complexity function of recurrence equations. We can apply Master's Theorem for:

1. Dividing type of functions
2. Decreasing type of functions

Equation for Master's Theorem : ***T(n) = aT(n/b) + f(n),*** where

- n = size of input
- a = no. of sub-problems in recursion
- n/b = size of each sub-problem. All sub-problems are assumed to have same size.
- f(n) = estimate cost of the work done outside recursive call, including the cost of dividing the problem & cost of merging the solutions

Here, $a \geq 1$ & $b > 1$ are constants, and f(n) is an asymptotically +ve function of the theorem . If $a \geq 1$ & $b > 1$ are constants and f(n) is an asymptotically +ve function, where T(n) has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a-\epsilon})$, then the function $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then the function $T(n) = \Theta(n^{\log_b a} * \log n)$.
3. If $f(n) = \Omega(n^{\log_b a+\epsilon})$, then the function $T(n) = \Theta(f(n))$.

$\epsilon > 0$ is a constant.

3.6.5. Initial Steps (Project Setup)

Before you start Java programming for Android, you will need certain tools installed. Make sure that you download the SDK Bundle(includes the Android SDK & the Integrated Development Env. (IDE)).

If you don't have a desired project opened, Android Studio will show you the Welcome screen, here you can create a new projects by just clicking Start a new Android Studio project. If a project is opened, then create a new project by selecting File > then New > New Project from the main menu Steps for the setup:

Install Android Studio.
Task: To Create your first project.
Step 1: Create a new project.
Step 2: Get your screen set up.
Step 3: Explore the structure of the project and layout.
Step 4: Create a virtual device in the studio (emulator) .
Step 5: Run your application on your new emulator.
Step 6: Your setup is now ready for the development.

CHAPTER 4.

RESULTS ANALYSIS AND VALIDATION

4.1. Analysis

The application is developed using java. In the backend C++ is used for algorithm implementation. This also promotes code reusability and faster execution of tasks. Since Java is an interpreted language thus as compared to java C++ is faster in execution, therefore it is to be used for system programming and algorithm implementation. This task is to be carried out with the help of Android NDK tool and JNI Layer. JNI Layer is the interface between Java Application & C++ Algorithms which is used to call C++ classes & methods in java program. The size of the application is 5MB with maximum RAM Usage of 200MB. Android MP Chart is to for visualization of the sorting algorithms using gradle plugin.

Let’s say you want to add a critical function or a specific processor-dependent code to your own app which will require high performance, using Java only. It would not succeed so in order to get the following requirements, you have to use C or C++ which will make it run faster. But even if we know C or C++, how would we implement it in our app? So this is where we require Native Development Kit(NDK) to integrate the native languages to make it run on the application.

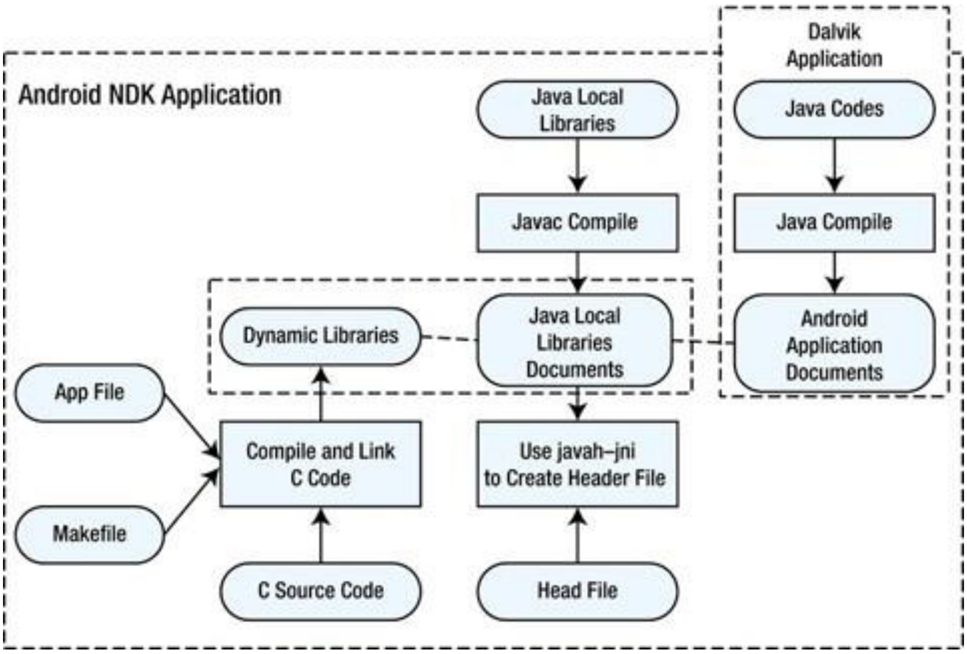


Figure 4.1. Application Blueprint

Applications projects developed by the Native Development Kit have components, as shown in Figure 10. In contrast to typical applications developed using the Android SDK, projects developed in the Native Development Kit add the Dalvik class code, manifest files, common resources, and also the JNI and a shared library generated by the Native Development Kit

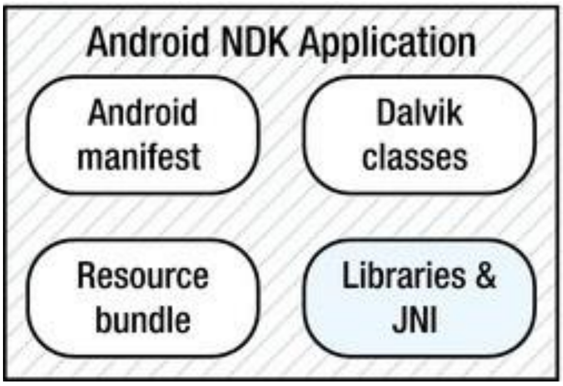


Figure 4.2. Application Components

JNI general workflow is as follows: Java initiates calls so that the local function's side code (such as a function written in C++) runs. Now the object is passed over from the Java, & run at a local function's completion. Here Java Native Interface is an adapter, mapping the variables and functions (Java methods) between the Java language and the native compiled languages (such as C++). We know that Java and C++ are very different in function prototype definitions and variable types.

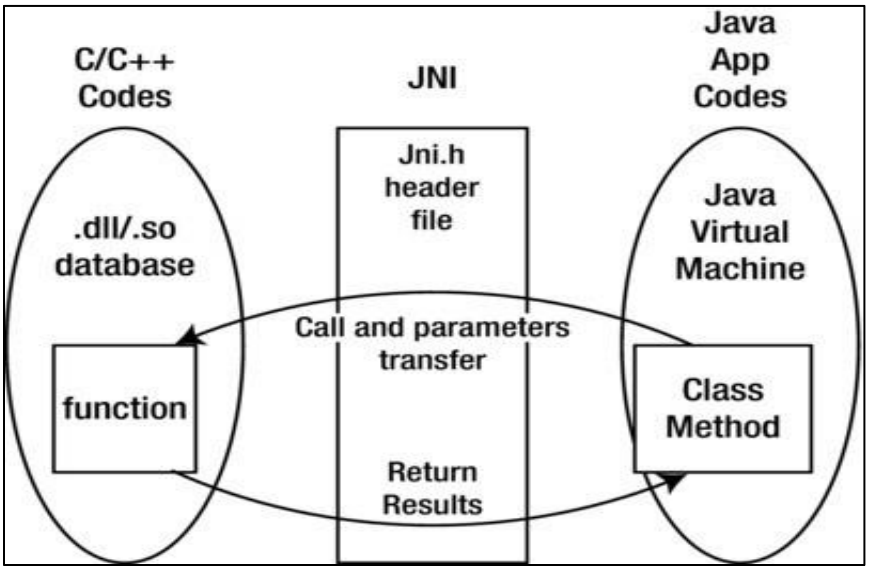


Figure 4.3. JNI Workflow

JNI Threat Model: The Java runtime environment safely manages memory space: It performs automatic checks on access within bounds of the array, & it has no explicit pointer arithmetic. When you compare Java with C & C++programs, you find that C and C++ programs can fail because of corruption in the memory that is caused by heap & buffer overflows.

JNI Runtime Behavior: A specific behavior of the runtime environment is considered a vulnerability(weakness) if it allows an attacker to analyse the checking mechanisms that protect the confidentiality, integrity, & availability of the JNI environment. This applies to bypassing the security checks in context with java. Native functions are capable of bypassing the Java security system architecture. The action of native code is not checked against the security policies or visibility declarations (for example, public or protected). For example, Java Native Interace code might alter the values of normally immutable classes such as java.lang.Integer, & it also might allow read and modify (for example, private key data) from arbitrary private fields. If attackers can inject data into a native function they can trigger buffer or heap overflows that could lead to arbitrary code execution on behalf of the adversary.

Implementing logic using Java: After the front-end elements are being finalized, the most important step remains implementing logic for all the activities in order to work well. The logic needs to be implemented in the MainActivity.java file. The code snippet for bubble sort visualizer which uses bubble sort algorithm and MP Android Chart for Visualization is provided in the appendix.

Android XML: Basically in Android XML standard is used to implement the User Interface-related data. So understanding the core part of the User Interface with respect to XML standard is very important. The User Interface for an Android Application is built as a group of main layouts & widgets. The layouts can be ViewGroup objects or containers that control how the children view should be positioned on the device screen. Widgets here are the view objects, such as Buttons & the text boxes. XML tags define the data & used to store data. It's easily scalable & very simple to develop. XML is used to implement User Interface related data, and it's a lightweight language that doesn't make layout heavy.

Gradle File: Gradle is a open source build system that is used to automate testing, deployment, building etc. These are scripts where one can automate the desired tasks.

The task to copy some files from one directory to other are performed by the Gradle build script before the actual build process can happen.

Every project developed in android needs a Gradle for generating an APK file from the .java & .xml files in the development project. Gradle takes all the source files including java & XML & applies appropriate tools, for example converts the java files into dex files, then compresses all of them into a single file (apk) that is actually used.

4.2. Testing

4.2.1. Testing Model

Testing is the process of verification and validation of the software in terms of efficiency, usability and accuracy . The testing process of a software product is based on different parameters including the functional testing , compatibility testing , system testing , regression testing depending on the particular type of software which is being developed and the model which is used for the development . For the testing of the project the approach used is of a V model. It depends on the association of a testing phase respective to the development stage. Development of each step is directly linked with the testing phase. The other phase starts only after the previous phase i.e. for each development event, there is a testing action corresponding to it.

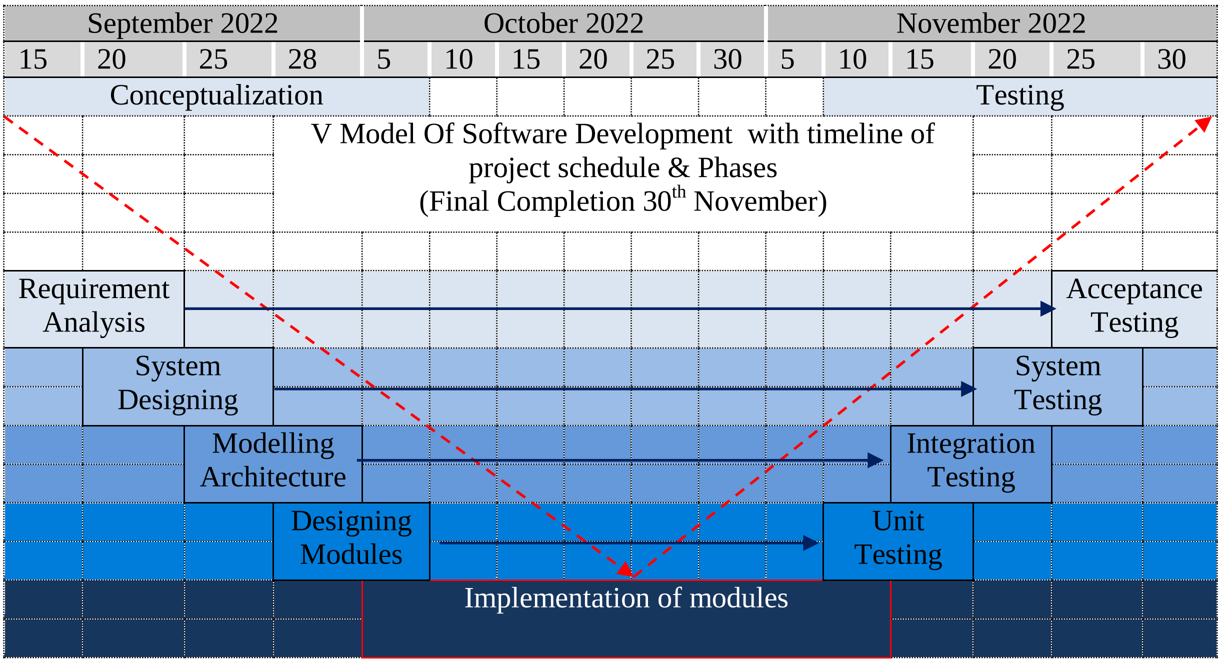


Figure 4.4 V Model for Testing

Verification: Verification phase involves the static analysis technique (review) which is done without executing the code. It is the process of analysing the product development phase to find whether requirements meet or not

Validation: Validation phase involves dynamic analysis method including functional & non-functional testing done by executing code. It is the process to evaluate the software after its completion of the development / implementation phase to determine whether software meets the customer / end users expectations as well as the requirements.

Main Testing Phases to be included in the project:

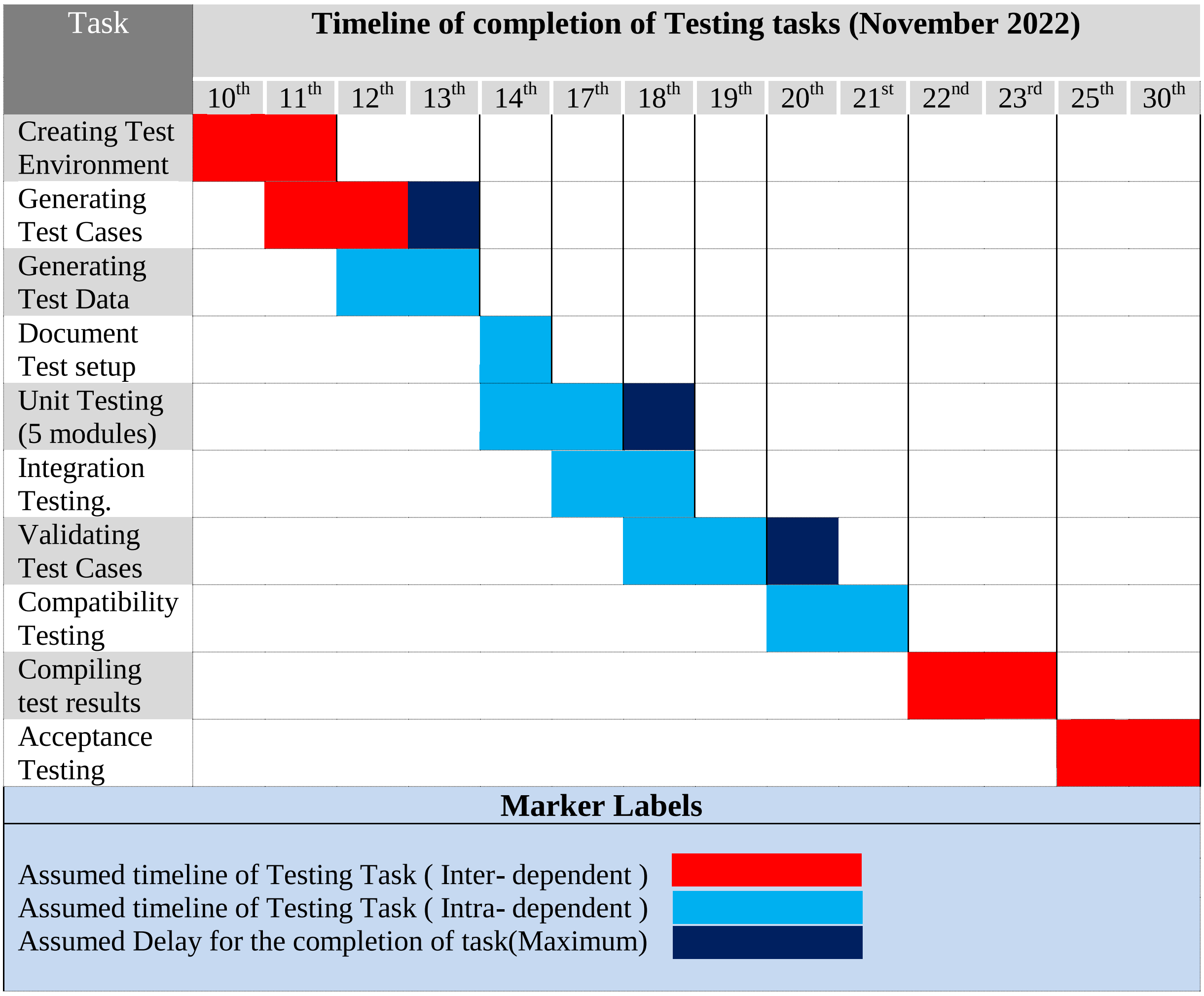
- **Unit Testing:** These are developed during module design phase. These Unit Test are executed independently to eliminate bugs at lowest level of the code.
- **Integration testing:** After completion of unit tests, Integration testing is being performed. In this phase of testing, the components/ units are integrated and the system is tested. It is performed on the Architecture design phase. Integration test verifies the communication of components among themselves.
- **System Testing:** It test the complete application with its desired functionality, total memory usage, CPU & its size. It tests the functional & non-functional requirements of the completely tested & developed application.
- **User Acceptance Testing (UAT):** It is performed in the customer environment that resembles the properties of the production environment. User acceptance testing verifies that the delivered system meets user's requirement .
- **The high-level design (HLD)** phase focuses on system architecture & design. HLD provide overview of solution, platform used , system configuration, product and service/process. An integration test plan is also created in this phase as well in order to test the software systems ability to work together. **The low-level design (LLD)** phase is where the real software components are designed. It defines the exact logic for each & every module of the system. Class diagram(UML diagram) with all the methods & relation between the classes comes under Low Level Design. Component tests are created in this phase as well.

4.2.2. Gantt Chart for Testing Cycle

We have used gannt chart for the depiction of the testing cycle for the application. The testing is starts on 10th November 2022 and ends on 30th November 2022 with User

Acceptance testing. In the V-Model, the respective testing phae of the development phase is planned parallely. So, there are phases for verification on one side of the ‘V’ and phases for validation on the other side. The Implementation Phase joins the two sides of the V-Model.

Below is the testing schedule of the model in a ***gantt chart*** (represent the project schedule over time) :



Task	**Timeline of completion of Testing tasks (November 2022)**													
	10th	11th	12th	13th	14th	17th	18th	19th	20th	21st	22nd	23rd	25th	30th
Creating Test Environment	🟥	🟥												
Generating Test Cases		🟥	🟥	⬛										
Generating Test Data			🟦	🟦										
Document Test setup					🟦									
Unit Testing (5 modules)					🟦	🟦	⬛							
Integration Testing.						🟦	🟦							
Validating Test Cases							🟦	🟦	⬛					
Compatibility Testing									🟦	🟦				
Compiling test results											🟥	🟥		
Acceptance Testing													🟥	🟥

Marker Labels

Assumed timeline of Testing Task (Inter- dependent)	🟥
Assumed timeline of Testing Task (Intra- dependent)	🟦
Assumed Delay for the completion of task(Maximum)	⬛

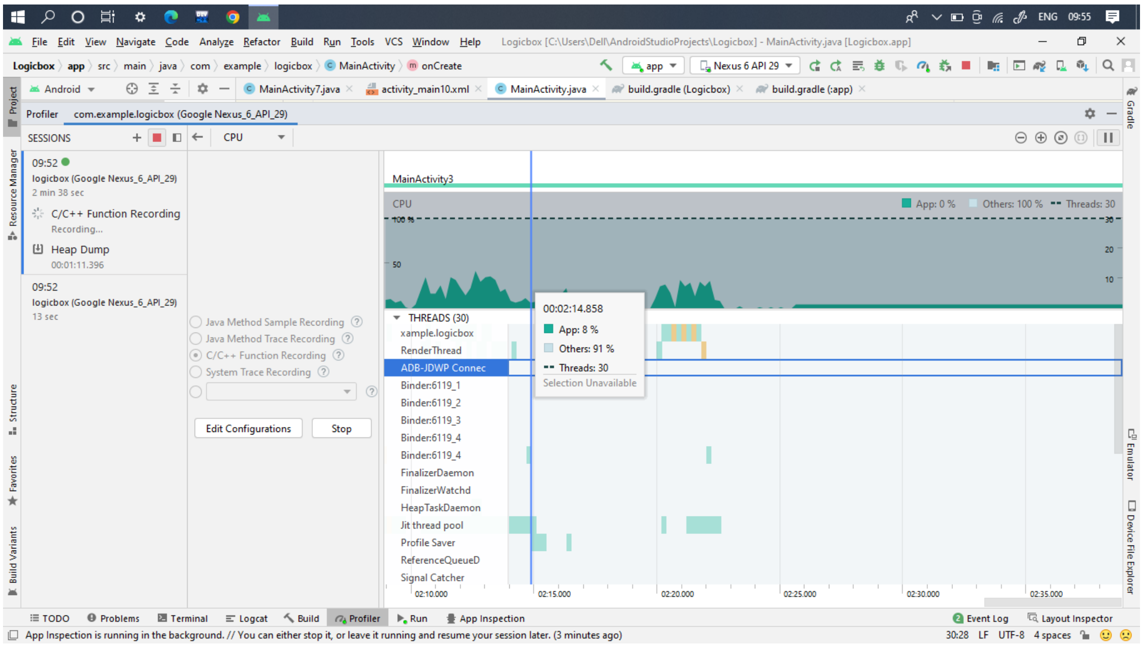
Figure 4.5. Gantt Chart for testing

4.2.3. Executing Tests

Creating Test Environment : Before testing , we have to create a proper test environment to verify the software with real life scenarios . Since it involve the manipulation with the actual model so we will be working with a clone of the software so that any changes will not be reflected in between the testing in the actual model .

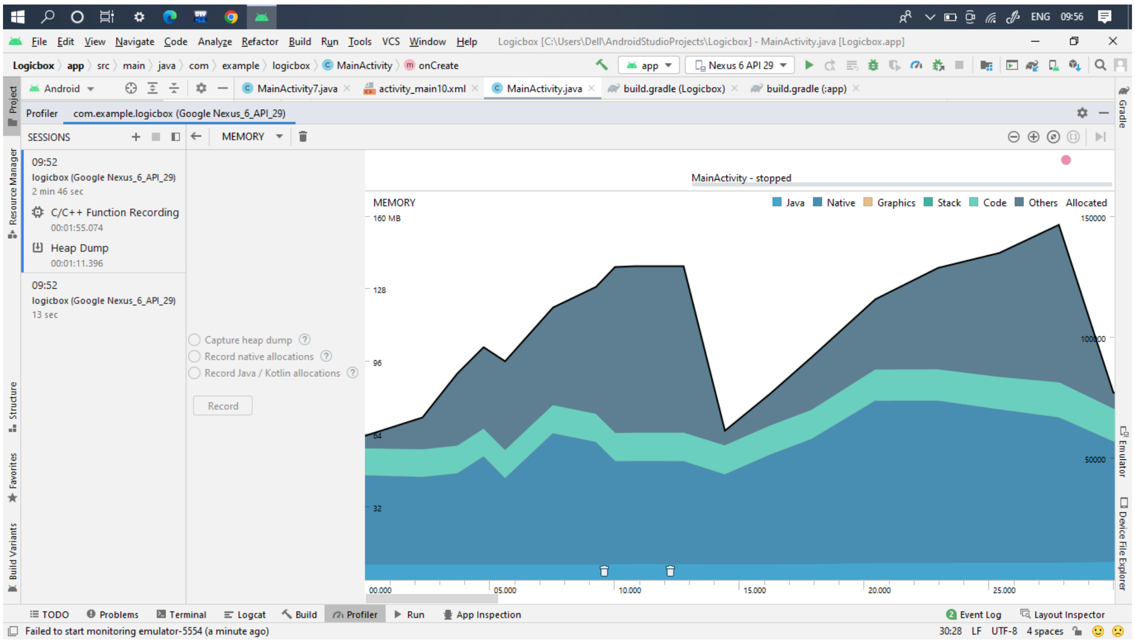
Compatibility/ System Testing : We have done the compatibility testing with the help of android profiler. Below are the snapshots of the compatibility testing. To open the Android Profiler, you can follow these steps:

1. Click on View > Tool Windows > Profiler
2. Select the device & app process you want to profile from the Profiler.
3. Click anywhere in the MEMORY timeline in order to open the Memory Profiler.



App Inspection is running in the background. // You can either stop it, or leave it running and resume your session later. (3 minutes ago)

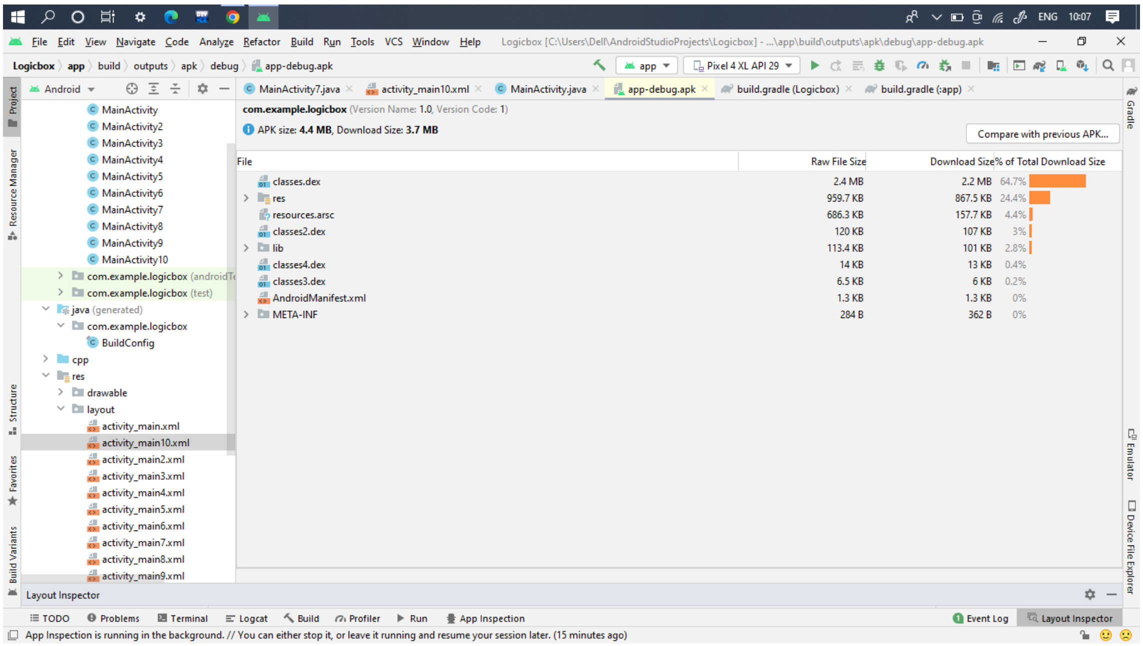
Figure 4.6. CPU Usage



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Logicbox [C:\Users\Dell\AndroidStudioProjects\Logicbox] - MainActivity.java [Logicbox.app]
Logicbox app src main java com example logicbox MainActivity onCreate
app
Nexus 6 API 29
Android
MainActivity7.java
activity_main10.xml
MainActivity.java
build.gradle (Logicbox)
build.gradle (:app)
Profiler com.example.logicbox (Google Nexus_6_API_29)
SESSIONS
MEMORY
09:52
logicbox (Google Nexus_6_API_29)
2 min 46 sec
C/C++ Function Recording
00:01:55.074
Heap Dump
00:01:11.396
09:52
logicbox (Google Nexus_6_API_29)
13 sec
Capture heap dump
Record native allocations
Record Java / Kotlin allocations
Record
MainActivity - stopped
MEMORY
160 MB
128
96
64
32
Java Native Graphics Stack Code Others Allocated
150000
100000
50000
00.000
05.000
10.000
15.000
20.000
25.000
TODO Problems Terminal Logcat Build Profiler Run App Inspection
Event Log Layout Inspector
30:28 LF UTF-8 4 spaces
ENG 09:56

Failed to start monitoring emulator-5554 (a minute ago)

Figure 4.7. Memory Usage



Logicbox [C:\Users\Dell\AndroidStudioProjects\Logicbox] - ...\app\build\outputs\apk\debug\app-debug.apk

com.example.logicbox (Version Name: 1.0, Version Code: 1)

APK size: **4.4 MB**, Download Size: **3.7 MB**

File	Raw File Size	Download Size	% of Total Download Size
classes.dex	2.4 MB	2.2 MB	64.7%
res	959.7 KB	867.5 KB	24.4%
resources.arsc	686.3 KB	157.7 KB	4.4%
classes2.dex	120 KB	107 KB	3%
lib	113.4 KB	101 KB	2.8%
classes4.dex	14 KB	13 KB	0.4%
classes3.dex	6.5 KB	6 KB	0.2%
AndroidManifest.xml	1.3 KB	1.3 KB	0%
META-INF	284 B	362 B	0%

App Inspection is running in the background. // You can either stop it, or leave it running and resume your session later. (15 minutes ago)

Figure 4.8. Occupied Space

APK Analyzer lets you open and inspect the contents of any APK file you have on your computer, either built from your local Android Studio project or acquired from your build server or other artifact repository.

APK Analyzer shows raw file size and download file size values for each entity, as shown in the above figure. Raw File Size depicts the unzipped size of the entity on disk but Download Size depicts the estimated compressed size as it would be delivered by Google Play Store. The percentage of Total Download Size depicts the percentage of the APK's total download size the entity represents. The application uses a small size of 4.4 MB and small memory of maximum 160 MB.

After all the completion of test a proper documentation is provided to the review team to formally review the software for the final deployment or release in the market following marketing release strategies .

Special Note:

APK Analyzer with release builds works best. If you need to visualize the debug build of your app, assure you are using an APK which is not instrumented for the Instant Run. In order to obtain it, you may use the Build → then Build APK command. You can see if you have opened an Instant Run by checking the presence of the instant-run.zip file inside the archives.

4.3. Results

4.3.1. Working of the Application

The project SORTING VISUALIZER entitled "*LogicBox*" is developed as an android application using Java as an application programming language & C++ as a system programming language. All the sorting algorithms & time complexity calculator is being developed in C++ and integrated with the android application with the help of Android's Native Development Kit (NDK) and JNI (Java Native Interface). The project is configured using gradle plugins and various libraries has been utilized for the visualization of the android application (for example MP Android Chart).

The Application is divided into 6 activities termed as Main Activity 1, Main Activity 2 and so on. The first two activities is the home page of the application which contains the main menu. The main menu contains all the buttons which routes us to the desired visualizer. The UI Frontend is developed in XML in the android studio and the backend is encoded in java.

All the sorting visualizer takes the unsorted data in the form of input. The user have to provide comma separated numerical values for the sorting visualization. Once the user input the data then it need to be saved in the current buffer using the save button in the user interface. After the data is saved in the buffer the user can visualize the sorting algorithm using iterate button as per his convenience & timing. The Visualization is shown with the help of bars and its height which is developed with the help of MP Android Chart. The data which is being sorted is also shown in the user interface for each iteration. When the array is sorted the iteration button is disabled.

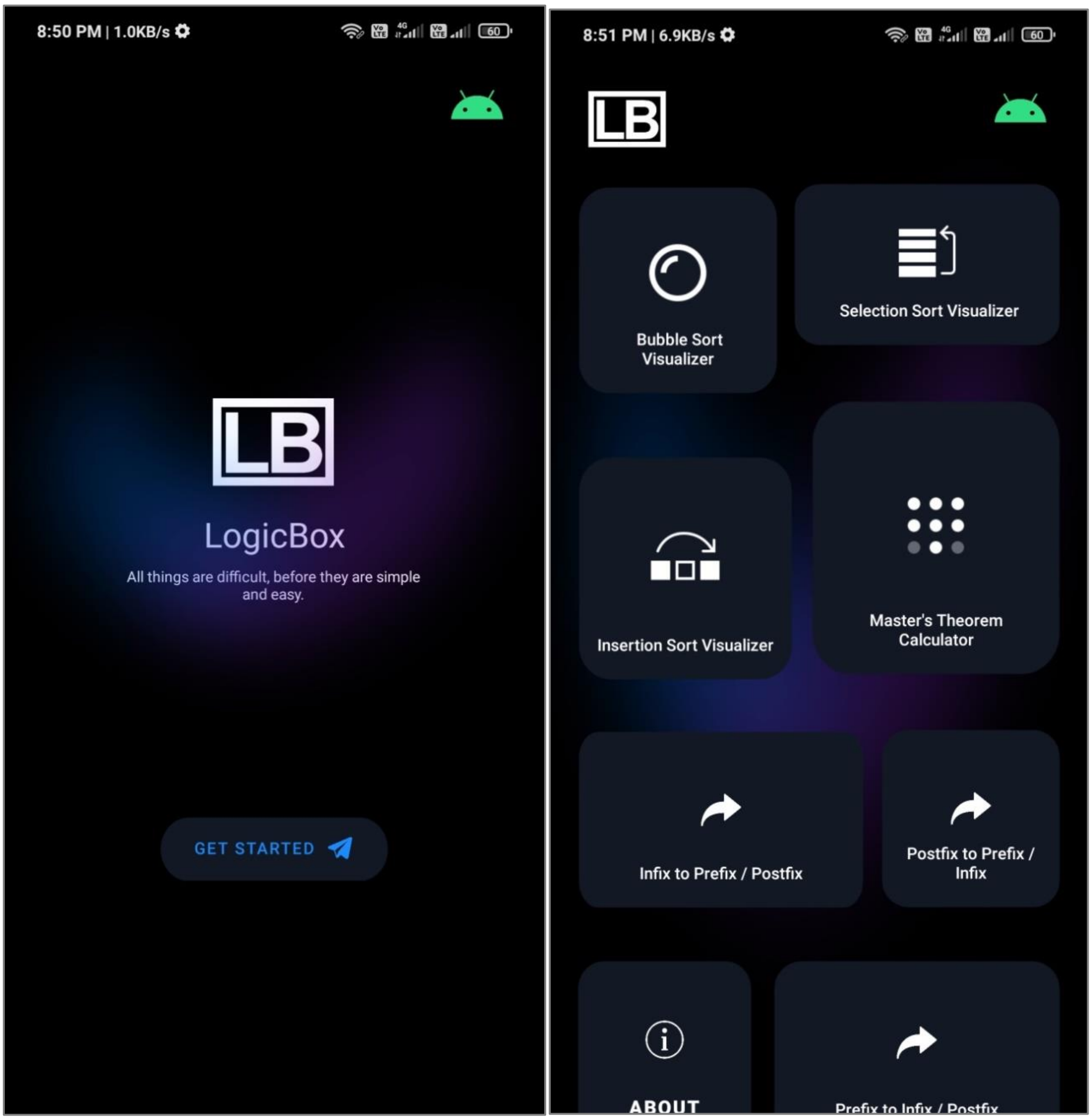
You can navigate to the master's theorem calculator for analysing the time complexity of any decreasing time function. You have to provide the input as stated in the UI for the given time function format.

4.3.2. Main Activity & Menu Bar

The application opens in the main activity where there is the button "Get Started" which takes you to the main menu of the application. The main menu contains all the activities.

An activity in android provides the window which the app draws its User Interface. This window basically fills the screen, but can be smaller than the screen & float on top of other windows. Most of the times, one activity implements one screen in an application. For instance, one of an application’s activities may implement as a Preferences screen, where as another activity implements a Select Photo screen.

Most application contain multiple screens, that means they comprise multiple activities. Meanwhile, one activity in an application is specified as the main activity, which is the first screen to appear when the user launches the application. Each activity can then start another activity in order to perform different types of actions.

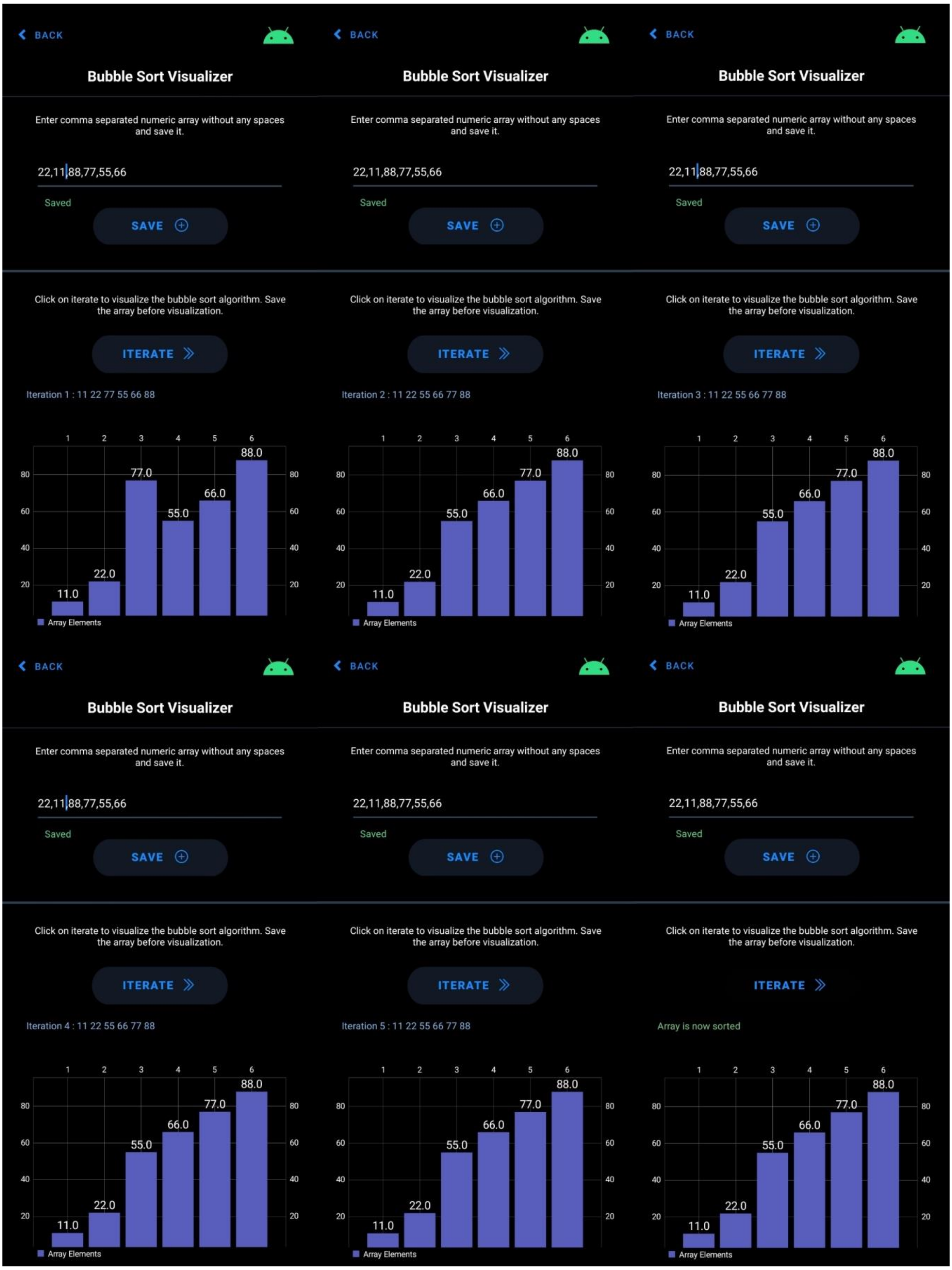

8:50 PM | 1.0KB/s
LB
LogicBox
GET STARTED
8:51 PM | 6.9KB/s
LB
Bubble Sort Visualizer
Selection Sort Visualizer
Insertion Sort Visualizer
Master's Theorem Calculator
Infix to Prefix / Postfix
Postfix to Prefix / Infix
ABOUT
Prefix to Infix / Postfix

All things are difficult, before they are simple and easy.

Figure 4.9. Main Activity

4.3.3. Bubble Sort Visualizer

The below figure shows the demonstration how bubble sort works with the help of MP android chart and integrated algorithm.



Bubble Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the bubble sort algorithm. Save the array before visualization.

Iteration 1 : 11 22 77 55 66 88

Bubble Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the bubble sort algorithm. Save the array before visualization.

Iteration 2 : 11 22 55 66 77 88

Bubble Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the bubble sort algorithm. Save the array before visualization.

Iteration 3 : 11 22 55 66 77 88

Bubble Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the bubble sort algorithm. Save the array before visualization.

Iteration 4 : 11 22 55 66 77 88

Bubble Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the bubble sort algorithm. Save the array before visualization.

Iteration 5 : 11 22 55 66 77 88

Bubble Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

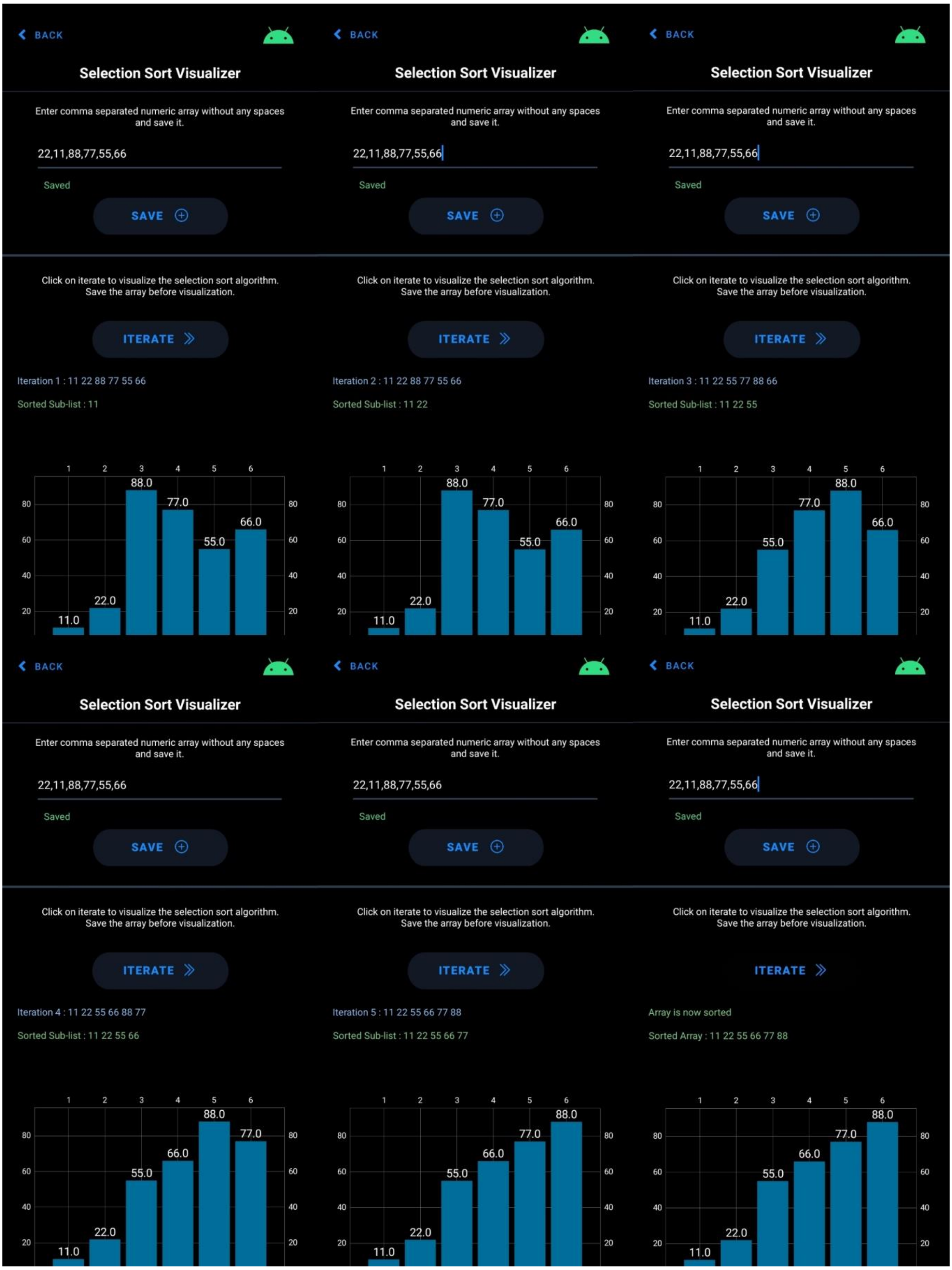
Click on iterate to visualize the bubble sort algorithm. Save the array before visualization.

Array is now sorted

Figure 4.10 Bubble Sort Visualization

4.3.4. Selection Sort Visualizer

The below figure shows the demonstration how selection sort works with the help of MP android chart and integrated algorithm.



Selection Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the selection sort algorithm. Save the array before visualization.

Iteration 1 : 11 22 88 77 55 66

Sorted Sub-list : 11

Selection Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the selection sort algorithm. Save the array before visualization.

Iteration 2 : 11 22 88 77 55 66

Sorted Sub-list : 11 22

Selection Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the selection sort algorithm. Save the array before visualization.

Iteration 3 : 11 22 55 77 88 66

Sorted Sub-list : 11 22 55

Selection Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the selection sort algorithm. Save the array before visualization.

Iteration 4 : 11 22 55 66 88 77

Sorted Sub-list : 11 22 55 66

Selection Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the selection sort algorithm. Save the array before visualization.

Iteration 5 : 11 22 55 66 77 88

Sorted Sub-list : 11 22 55 66 77

Selection Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the selection sort algorithm. Save the array before visualization.

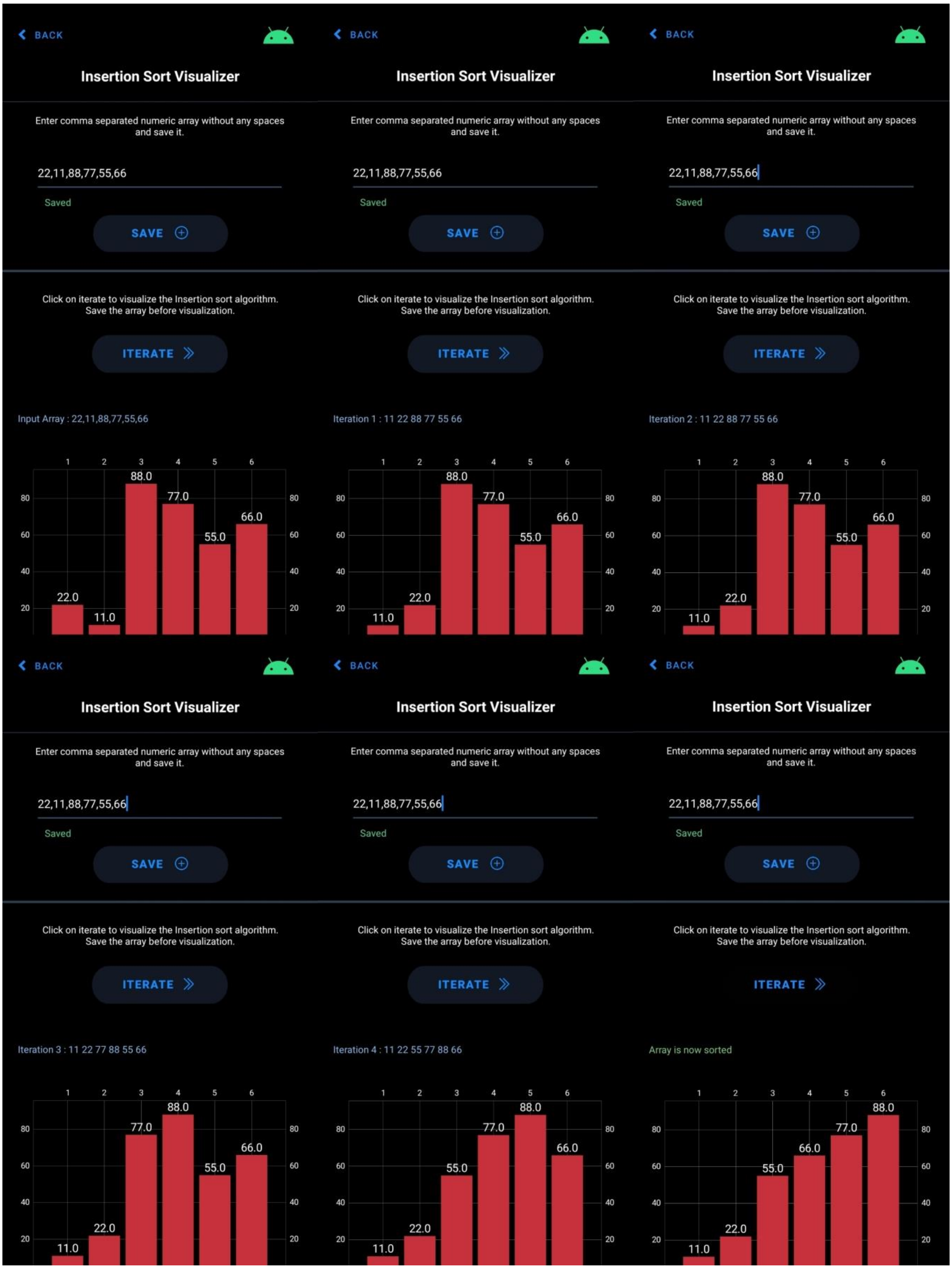
Array is now sorted

Sorted Array : 11 22 55 66 77 88

Figure 4.11. Selection Sort Visualization

4.3.5. Insertion Sort Visualizer

The below figure shows the demonstration how bubble sort works with the help of MP android chart and integrated algorithm.



Insertion Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the Insertion sort algorithm. Save the array before visualization.

Input Array : 22,11,88,77,55,66

Insertion Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the Insertion sort algorithm. Save the array before visualization.

Iteration 1 : 11 22 88 77 55 66

Insertion Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the Insertion sort algorithm. Save the array before visualization.

Iteration 2 : 11 22 88 77 55 66

Insertion Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the Insertion sort algorithm. Save the array before visualization.

Iteration 3 : 11 22 77 88 55 66

Insertion Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the Insertion sort algorithm. Save the array before visualization.

Iteration 4 : 11 22 55 77 88 66

Insertion Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

22,11,88,77,55,66

Saved

Click on iterate to visualize the Insertion sort algorithm. Save the array before visualization.

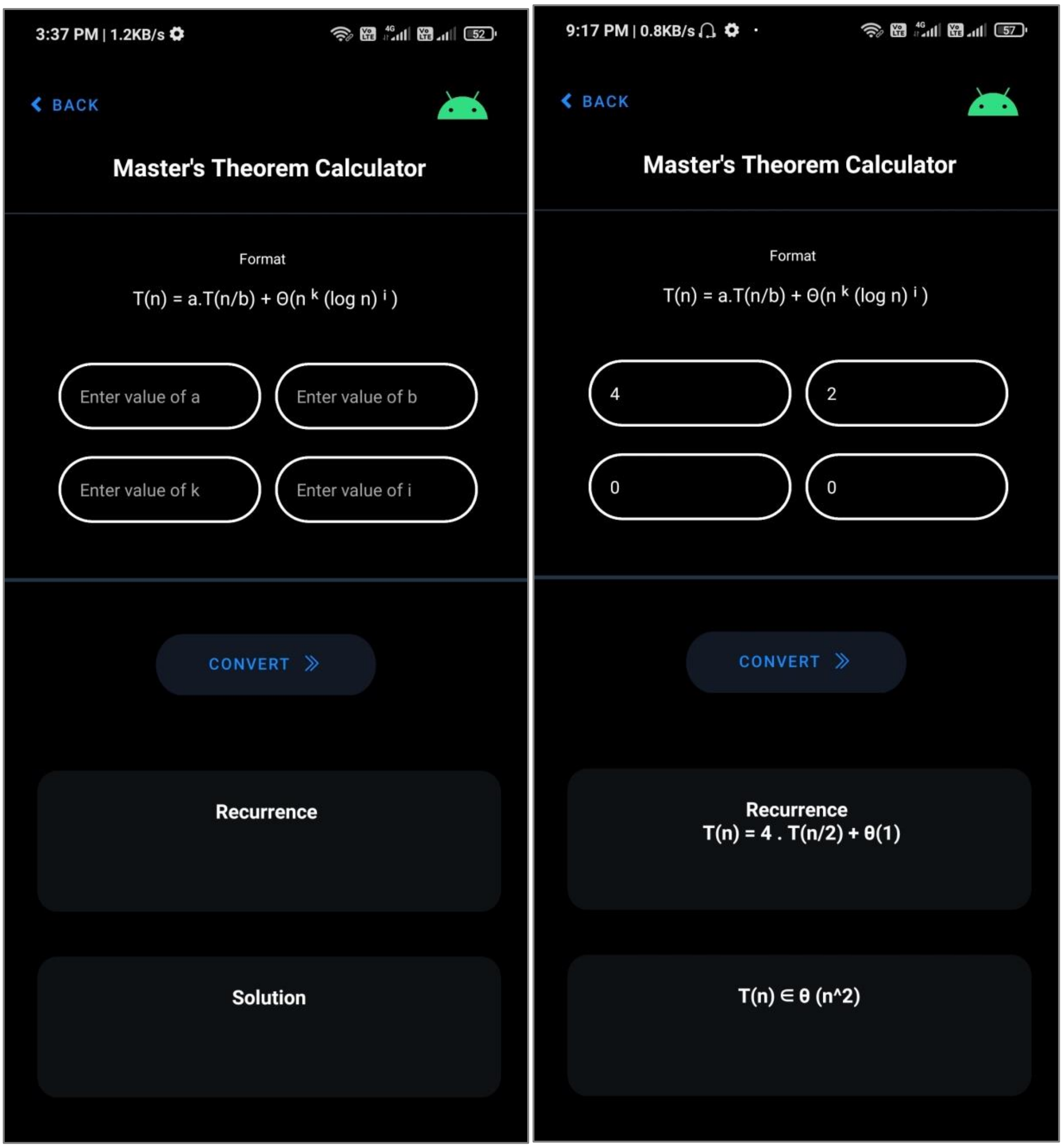
Array is now sorted

Figure 4.12 Insertion Sort Visualization

4.3.6. Master’s Theorem Calculator

Master’s Theorem is one of the most widely used method in order to analyze or compare various types of algorithms by computing their recurrence relations and time complexities.

Recursive functions call themselves in their function body. It may get complex if we start computing its time complexity function by other used simpler methods. It is the most useful & easy technique to calculate the time complexity function of the recurrence equations. The application shows the default syntax of master’s method where you have to specify the desired inputs in order to calculate running time complexity.



Master's Theorem Calculator

Format

T(n) = a.T(n/b) + Θ(n ^{k} (log n) ^{i})

Enter value of a

Enter value of b

Enter value of k

Enter value of i

Recurrence

Solution

Master's Theorem Calculator

Format

T(n) = a.T(n/b) + Θ(n ^{k} (log n) ^{i})

4

2

0

0

Recurrence
T(n) = 4 . T(n/2) + θ(1)

T(n) ∈ θ (n^2)

Figure 4.13. Master’s Theorem Caclulator

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. Conclusion

According to our research, algorithm visualization can be seen as an essential supporting tool in the field of computer science and engineering as a additional standard way for education. Within the project we provided an overview of the Logic Box algorithm visualization platform as well as our practical experiences with the whole system. We believe that it helps to improve the quality of education and also contribute to the solution for some of the problems in higher field of education mentioned at the beginning of the report . We had implemented all the in place sorting algorithms including selection sort, insertion sort and bubble sort. We had also implemented time complexity calculator for the decreasing type functions with the help of master's theorem. All modules are independent of each other providing module independency. The application is developed using java as a system programming language and C++ for application programming.

There are still open issues and problems with using algorithm visualizations. Sorting algorithm visualizations can help us to understand the principles, but do not replace the urge need to implement algorithms by students is a chosen language.

Another drawback of using these algorithm visualizations within our subject is the lack of tools offering the required visualizations in one package with a unified interface. The Logic Box platform can also be considered as a step in this direction. Meanwhile, more systematic evaluation of algorithm visualization packages and tools is required, as there is rather informal proofs available that the applications of algorithm visualizations are useful.

Our intentions here is to include the development of new visualization modules from the area of sorting algorithms & more complex data structures. Some of the proposed core-related features are on the list too (like graphically visualizations with MP Android Chart etc.), but some of them will probably not be implemented in a near future (for example undo/step back in the running visualization), as they will require more fundamental changes in the algorithm which may alter the complexities and time bound.

5.2. Future work

Although sorting of data structures could be done on any well-ordered set which includes integers, real no’s , structures, or programs, however for the convenience & ease of representation and visualization, only the sorting of non-negative/positive integers was considered in the application. This thesis involved developing a new graphical tool for the visualization of three different types of sorting algorithms. In our current solution we have highlighted the visualization of in place sorting algorithms such as bubble sort, selection sort and insertion sort. We have also highlighted the importance of running time complexity and implemented master’s theorem for the same. But in our current solution masters theorem is only applicable to decreasing type functions so in the near future we will try to improve this major current limitation in our solution.

Future versions of the tool could include the following modifications in the application:

- If the system can support sound or voice support, a narrative module to the dynamic display would make the display more interesting & helpful and the graphical representation of the sorting algorithms could be dynamic based on the sorting information obtained from each new iteration or run.
- Logic Box can have more features which will enhance the learning of algorithms even better. We can also customize the data as well as add more sorting algorithms to enhance the software scalability and reliability.
- More features i.e., greedy and dynamic programming algorithms, can also be added there thus it will provide a better understanding to researchers, programmers and students (the end users of the application).
- As stated we have implemented in place sorting algorithms, in the future work we will try to implement out place sorting algorithms as well as stable and unstable sorting algorithms.
- The current solution is limited to mobile devices where it is supported only on android devices, but we will try to extend the device supportability in the future where our application can run on any device including IOS as well as android devices.
- In future we will try to implement more calculators to calculate the running time complexity beyond decreasing functions. Also we tend to improve the current sorting algorithm visualization which lacks the undo or reverse operation.

REFERENCES

[1] M.H. Brown, R. Sedgewick, A system for algorithm animation, Proceedings of the 11th annual conference on Computer graphics and interactive techniques, SIGGRAPH’84 (ACM New York, NY, USA, 1984)

[2] W. C. Pierson and S. H. Rodger, Web-based animation of data structures using JAWAA, ACM SIGCSE Bulletin 30(1) (1998), 267-271.

[3] G. Rößling, M. Schüler and B. Freisleben, The ANIMAL algorithm animation tool, 5th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education (2000), 37-40.

[4] S. Khuri, Designing Effective Algorithm Visualizations, In proceedings of: First International Program Visualization Workshop, ITiCSE, Porvoo, Finland, July 7 - 8, 2000, Available: http://www.cs.sjsu.edu/~khuri/invited.html

[5] T. L. Naps, Jhavé: Supporting algorithm visualization, IEEE Computer Graphics and Applications 25(5) (2005), 49-55.

[6] E. Vrachnos and A. Jimoyiannis, Dave: A dynamic algorithm visualization environment for novice learners, 2008 Eighth IEEE International Conference on Advanced Learning Technologies (2008), 319-323

[7] Krajci, I., Cummings, D. (2013). Creating and Porting NDK-Based Android Applications. In: Android on x86. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4302-6131-5_7

[8] V. Karavirta and C. A. Shaffer, Creating engaging online learning material with the JSAV javascript algorithm visualization library, IEEE Transactions on Learning Technologies 9(2) (2015), 171-183.

[9] Saurabh Singh, Agile or Waterfall: Which App Development Approach to Consider, https://appinventiv.com/ (May 23, 2022)

[10] Alfred V. Aho, John E. Hopcroft, and Jeffrey Ullman, Data Structures and Algorithms, Addison - Wesley Publishing Company, Reading, MA, 1983

[11] Daniel Bernstein, Using Motif C++, SIGS Books, New York, NY, 1995.

[12] Tomas Gerald and Christoph W. Ueberhuber, Visualization of Scientific Parallel Programs, Springer-Verlag, Berlin, Germany, 1994.

[13] Nahum D. Gershon, "From Perception to Visualization", Computer Graphics, Vol. 26, No.2, pp. 414 - 417, July 1992.

[14] William C. House, Interactive Computer Graphics Systems, Petrocelli Books, Inc., New York, NY, 1982.

[15] Donald E. Knuth. The Art of Computer Programming, Vol. 3: Sorting and Searching, Addison-Wesley Publishing Company, Inc., Reading, MA, 1973.

[16] Bobby S. Koshy, Towards a Graphical Deadlock Analysis Tool, Master of Science Thesis, Computer Science Department Oklahoma State University, Stillwater, OK, 1995.

[17] Wilson Lee, An Implementation of a Data Structure Display System, Master of Science Thesis, Computing and Infonnation Science Department, Oklahoma State University, Stillwater, OK, 1988.

[18] Udi Manber, Introduction to Algorithms: A Creative Approach, AddisonWesley Publishing Company, Inc., Reading MA, 1989.

[19] Niall Mansfield, The X Window System: A User 's Guide, Addison-Wesley Publishing Company, Reading, MA, 1991.

[20] Andrian Nye,X Protocol Reference Manualfor Version 11, O'Reilly & Associates, Inc., Sebastapol, CA, 1990.

[21] John K. Ousterhout, Tel and the Tk Toolkit, Addison-Wesley Publishing Company, Reading, MA, 1994.

[22] Robert P. Rich, Internal Sorting Methods illustrated with PLI1 Programs, Prentice-Hall, EngleWood Cliffs, NJ, 1972.

[23] Randi J. Rost, X and Motif Quick Reference Guide, Digital Press, X and Motif Series, Bedford, MA, 1990.

[24] Robert Sedgewick, Algorithms in C, Addison-Wesley Publishing Company, Inc., Reading, MA, 1990.

[25] Donald L. Shell, "A High-Speed Sorting Procedure", Communications of the ACM, Vol. 2, No.7, pp. 30-32, July 1959.

[26] Ben Shneiderman, Designing the User Interface: Strategies for Effective Human-Computer Interaction, Addison-Wesley Publishing Company, Reading, MA, 1987.

[27] Jerry D. Smith, Object Oriented Programming with the X Window System Toolkits, John Wiley & Sons, Inc., New York, NY, 1991.

[28] Sorting out Sorting, The Dynamic Graphics Project Computer Systems Research Group, University of Toronto, Toronto, Canada, 1981.

[29] Yuh-Ching Su, An Empirical Study of Comb sort and Ways TO Improve It, Master of Science Thesis, Computer Science Department, Oklahoma State University, Stillwater, OK, 1993.

[30] Mark A. Weiss, "Empirical study of the expected running time of Shell sort", The Computer Journal, Vol. 34, No.1, pp. 88 - 91, February 1991.

[31] Mark A. Weiss, Data Structures and Algorithm Analysis in C, Benjamin/Cummins Publishing Company, Inc., Redwood City, CA, 1993.

[32] Douglas A. Young, Object-Oriented Programming with C++ and OSF, Prentice Hall, Upper Saddle River, NJ, 1995.

[33] T. Bingmann. "The Sound of Sorting - 'Audibilization' and Visualization of Sorting Algorithms." Panthemanet Weblog. Impressum, 22 May 2013.

[34] Bubble-sort with Hungarian ("Cs´ang´o") Folk Dance. Dir. K´atai Zolt´an and T´oth L´aszl´o. YouTube. Sapientia University, 29 Mar. 2011. Web. 29 Mar. 2017.

[35] A. Kerren and J. T. Stasko. (2002) Chapter 1 Algorithm Animation. In: Diehl S.(eds) Software Visualization. Lecture Notes in Computer Science, vol 2269. Springer, Berlin, Heidelberg.

[36] A. Moreno, E. Sutinen, R. Bednarik, and N. Myller. Conflictive animations as engaging learning tools. Proceedings of the Koli Calling '07 Proceedings of the Seventh Baltic Sea Conference on Computing Education Research - Volume 88, Koli '07 (Koli National Park, Finland), pages 203-206.

[37] J. Stasko. Using Student-built Algorithm Animations As Learning Aids. Proceedings of the Twenty-eighth SIGCSE Technical Symposium on Computer Science Education. SIGCSE '97 (San Jose, California), pages 25-29.

[38] J. Stasko, A. Badre, and C. Lewis. Do Algorithm Animations Assist Learning?: An Empirical Study and Analysis. Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems, CHI-93 (Amsterdam, the Netherlands), pages 61-66.

[39] Bubble-sort with Hungarian ("Csángó") folk dance YouTube video, created at Sapientia University, Tirgu Mures (Marosvásárhely), Romania.

[40] Sorting Out Sorting, Ronald M. Baecker with the assistance of David Sherman, 30 minute color sound film, Dynamic Graphics Project, University of Toronto, 1981. Excerpted and reprinted in SIGGRAPH Video Review 7, 1983. Distributed by Morgan Kaufmann, Publishers.

APPENDIX

GRADLE PLUGIN SETUP

```
plugins {
    id 'com.android.application'
}

android {
    compileSdk 32

    defaultConfig {
        applicationId "com.example.logicbox"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        externalNativeBuild {
            cmake {
                cppFlags ''
            }
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    externalNativeBuild {
        cmake {
            path file('src/main/cpp/CMakeLists.txt')
            version '3.10.2'
        }
    }
    buildFeatures {
        viewBinding true
    }
}

dependencies {
    implementation 'androidx.appcompat:appcompat:1.4.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.2'
    implementation 'androidx.annotation:annotation:1.2.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'
}
```

NATIVE-LIB.CPP SAMPLE

```
#include <jni.h>
#include <string.h>
#include "CPPClasses/infixToPrefix.cpp"
#include "CPPClasses/postfixToPrefix.cpp"
#include "CPPClasses/postfixToInfix.cpp"
#include "CPPClasses/prefixToInfix.cpp"
#include "CPPClasses/prefixToPostfix.cpp"

extern "C" JNIEXPORT jstring JNICALL
Java_com_example_logicbox_MainActivity_stringFromJNI( JNIEnv* env,
jobject){
    std::string hello = "By Mrinal";
    return env->NewStringUTF(hello.c_str());
}
extern "C" JNIEXPORT jstring JNICALL
Java_com_example_logicbox_MainActivity6_infToPostFromJNI(JNIEnv* env,
jobject, jstring s) {
    infixToPostfix o;
    std::string str =  env->GetStringUTFChars(s,0);
    std::string infixconversion=o.infToPost(str);
    return env->NewStringUTF(infixconversion.c_str());
}
extern "C" JNIEXPORT jstring JNICALL
Java_com_example_logicbox_MainActivity6_infToPreFromJNI(JNIEnv* env,jobject
,jstring s) {
    infixToPrefix o;
    std::string str =  env->GetStringUTFChars(s,0);
    std::string infixconversion=o.infToPre(str);
    return env->NewStringUTF(infixconversion.c_str());
}
extern "C" JNIEXPORT jstring JNICALL
Java_com_example_logicbox_MainActivity7_postToInfFromJNI(JNIEnv* env,
jobject, jstring s) {
    postfixToInfix o;
    std::string str =  env->GetStringUTFChars(s,0);
    std::string postfixconversion=o.postToInf(str);
    return env->NewStringUTF(postfixconversion.c_str());
}
extern "C" JNIEXPORT jstring JNICALL
Java_com_example_logicbox_MainActivity7_postToPreFromJNI(JNIEnv* env,
jobject,jstring s) {
    postfixToPrefix o;
    std::string str =  env->GetStringUTFChars(s,0);
    std::string postfixconversion=o.postToPre(str);
    return env->NewStringUTF(postfixconversion.c_str());
}
extern "C" JNIEXPORT jstring JNICALL
Java_com_example_logicbox_MainActivity8_preToInfFromJNI(JNIEnv* env,jobject
/* this */ ,jstring s) {
    prefixToInfix o;
    std::string str =  env->GetStringUTFChars(s,0);
    std::string prefixconversion=o.preToInfix(str);
    return env->NewStringUTF(prefixconversion.c_str());
}
```

ANDROIDMANIFEST.XML FILE

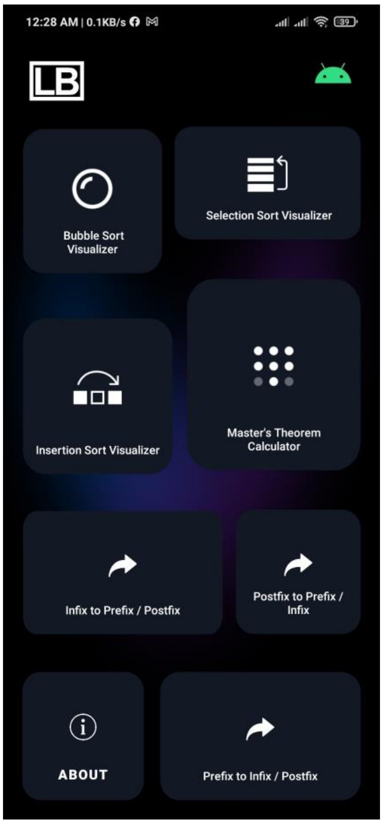
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.logicbox">

    <application
        android:allowBackup="true"
        android:icon="@drawable/lbblack"
        android:label="@string/app_name"
        android:roundIcon="@drawable/lbblack"
        android:supportsRtl="true"
        android:theme="@style/Theme.Logicbox">
        <activity
            android:name=".MainActivity10"
            android:exported="false"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name=".MainActivity9"
            android:exported="false"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name=".MainActivity8"
            android:exported="false"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name=".MainActivity7"
            android:exported="false"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name=".MainActivity6"
            android:exported="false"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name=".MainActivity5"
            android:exported="false"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name=".MainActivity4"
            android:exported="false"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name=".MainActivity3"
            android:exported="false"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name=".MainActivity2"
            android:exported="false"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name=".MainActivity"
            android:configChanges="orientation"
            android:screenOrientation="portrait"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

MAIN ACTIVITY SAMPLE CODE

```
package com.example.logicbox;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.example.logicbox.databinding.ActivityMainBinding;

public class MainActivity extends AppCompatActivity {

    static {
        System.loadLibrary("logicbox");
    }

    private ActivityMainBinding binding;
    Button ac2btn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        TextView tv = binding.sampleText;
        tv.setText(stringFromJNI());
        ac2btn=(Button)findViewById(R.id.ac2btn);
        ac2btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                openMainActivity2();
            }
        });
    }
    public void openMainActivity2(){
        Intent intent = new Intent(this,MainActivity2.class);
        startActivity(intent);
    }

    public native String stringFromJNI();
}
```

USER MANUAL

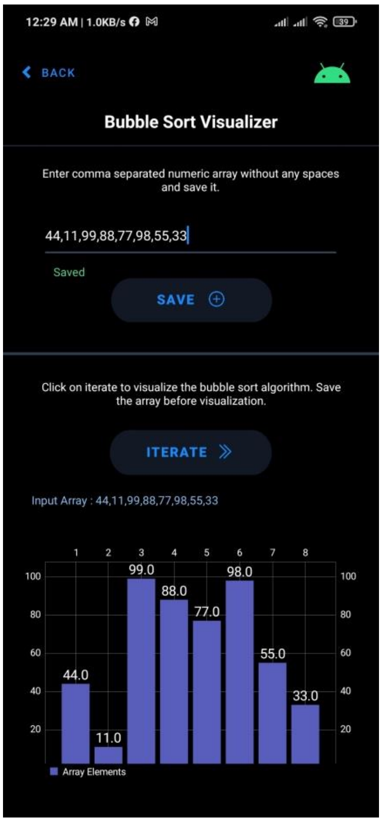
Menu Bar


12:28 AM | 0.1KB/s
LB
Bubble Sort Visualizer
Selection Sort Visualizer
Insertion Sort Visualizer
Master's Theorem Calculator
Infix to Prefix / Postfix
Postfix to Prefix / Infix
ABOUT
Prefix to Infix / Postfix

This is the page where you can get access to all the tools present in the application. As you can see in the dashboard, all the buttons are present which will take you to the respective tool where you can visualize the algorithm.
In this page there are 8 buttons :

- Bubble Sort Visualizer
- Selection Sort Visualizer
- Insertion Sort Visualizer
- Master’s Theorem Calculator
- Infix to other conversions
- Postfix to other conversions
- Prefix to other conversions

Bubble Sort Visualizer



Enter comma separated numeric array without any spaces and save it.

44,11,99,88,77,98,55,33

Saved

Click on iterate to visualize the bubble sort algorithm. Save the array before visualization.

Input Array : 44,11,99,88,77,98,55,33

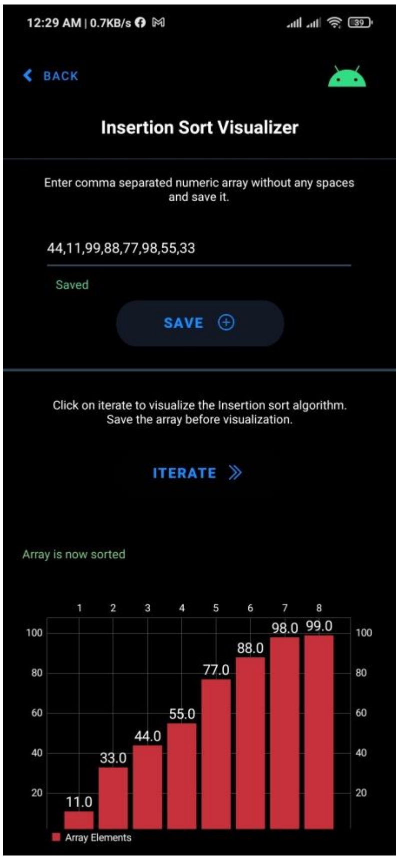
In this page you can visualize the respective bubble sort algorithm. Here are multiple buttons present. First of all you have to enter the unsorted array values in the input field. It must be comma separated values and non-negative.
After that you have to click on save button to save the array in the current temporary buffer.
When the unsorted array is saved you can start your visualization by clicking on iterate button. In the screen you can see the graphical visualization of the sorting algorithm along with the initial iteration manipulations. When the array is sorted the iterate button will be disabled.

Insertion Sort Visualizer

In this page you can visualize the respective Insertion sort algorithm. Here are multiple buttons present. First of all you have to enter the unsorted array values in the input field. It must be comma separated values and non-negative.

After that you have to click on save button to save the array in the current temporary buffer.

When the unsorted array is saved you can start your visualization by clicking on iterate button. In the screen you can see the graphical visualization of the sorting algorithm along with the initial iteration manipulations. When the array is sorted the iterate button will be disabled.



Insertion Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

44,11,99,88,77,98,55,33

Saved

Click on iterate to visualize the Insertion sort algorithm. Save the array before visualization.

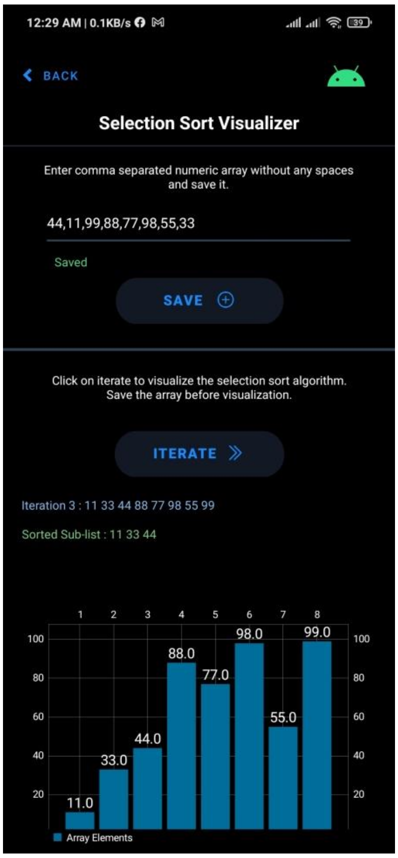
Array is now sorted

Selection Sort Visualizer

In this page you can visualize the respective Selection sort algorithm. Here are multiple buttons present. First of all you have to enter the unsorted array values in the input field. It must be comma separated values and non-negative.

After that you have to click on save button to save the array in the current temporary buffer.

When the unsorted array is saved you can start your visualization by clicking on iterate button. In the screen you can see the graphical visualization of the sorting algorithm along with the initial iteration manipulations. When the array is sorted the iterate button will be disabled.



Selection Sort Visualizer

Enter comma separated numeric array without any spaces and save it.

44,11,99,88,77,98,55,33

Saved

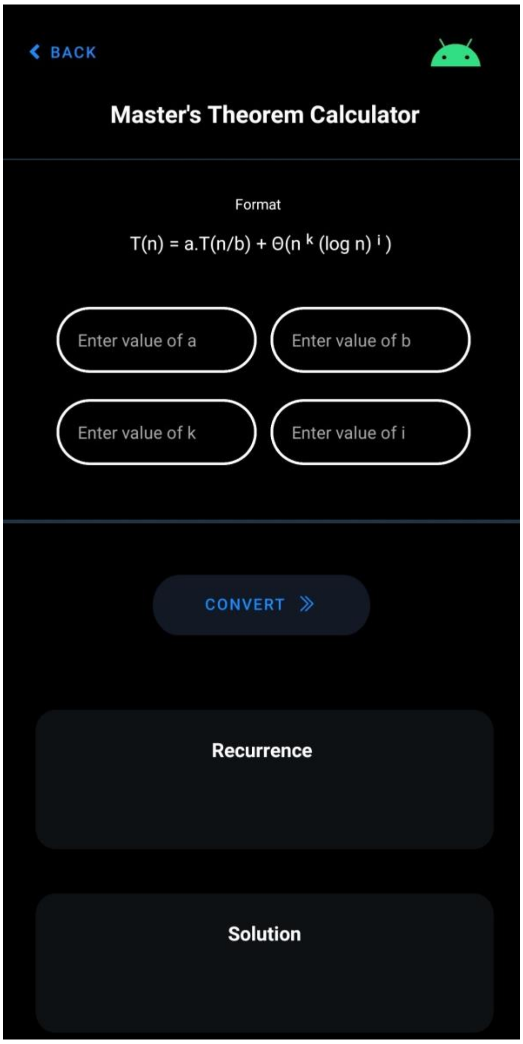
Click on iterate to visualize the selection sort algorithm. Save the array before visualization.

Iteration 3 : 11 33 44 88 77 98 55 99

Sorted Sub-list : 11 33 44

Masters Theorem Calculator

This calculator is basically used to calculate and analyse the time complexity of the decreasing time function. The sample of the time function is given on the screen.
After analysis of the algorithm when the user has formulated the time function then they can fed the input parameters in the form and will be provided with the desired complexity. It is only valid for decreasing time functions.



Master's Theorem Calculator

Format

$T(n) = a.T(n/b) + \Theta(n^{k} (\log n)^{i})$