Table of Contents

Abstract

Algorithms are a fundamental concept in computer science, used to solve problems and automate processes in a variety of fields. Sorting algorithms are an important concept in computer science and the Bubble Sort algorithm is one of the simplest and most commonly taught sorting algorithms. In this project, we aimed to visualize the Bubble Sort algorithm using Tkinter and Python. Our visualization provides an animated representation of the algorithm in action, highlighting the steps involved in the algorithm and the elements being compared and swapped.

Our project aimed to provide a visual representation of the Bubble Sort algorithm that would make it easier to understand and learn. We achieved this by creating an engaging and interactive visualization with easy to use user interface that demonstrates the workings of the Bubble Sort algorithm. Our project also provides a tool that can be used by students and educators to teach and learn about the Bubble Sort algorithm.

To create our visualization, we used Tkinter as our graphical user interface and Python as our programming language. We developed an algorithm that animates the Bubble Sort algorithm step by step, highlighting the elements being compared and swapped. Our project also includes a feature that allows users to interact with the visualization and see the changes in the sorting algorithm in real-time. Furthermore, it also includes the use of different searching methods to understand the full potential of the sorting algorithm by seeing how it benefits and can be used in real time problems.

Our visualization of the Bubble Sort algorithm provides a unique and effective way to teach and learn about sorting algorithms. We believe that our project has the potential to help students and educators better understand the Bubble Sort algorithm and its applications. In this report, we will describe our methodology for visualizing the Bubble Sort algorithm, present the results of our project and discuss the strengths and limitations of our visualization.

Keywords: Algorithms, Sorting algorithms, Bubble sort, Searching, Linear search, Binary search, Python, Tkinter, Visualization, User interface.

Introduction

- **Algorithms and Sorting algorithms**

Algorithms are a fundamental concept in computer science, used to solve problems and automate processes in a variety of fields. At its simplest, an algorithm is a set of instructions that a computer program follows to complete a task. These instructions can range from simple, straightforward steps to complex decision-making processes that involve multiple variables and conditions.

The importance of algorithms cannot be overstated, as they are used in virtually every aspect of computer science and technology. From simple arithmetic operations to complex machine learning algorithms, algorithms are essential for solving problems and automating tasks. They are used in a wide range of applications, including web search engines, image recognition software, financial modelling and more.

In addition to their practical applications, algorithms are also a fascinating and intellectually stimulating subject in their own right. Studying algorithms provides insights into the underlying principles of computer science and how computers can be used to solve problems in innovative and creative ways.

One of the most common type of algorithm used all the time is Sorting algorithm. Sorting algorithms are used to arrange data and information in a logical and efficient manner. A sorting algorithm is a set of instructions that a computer program follows to rearrange a collection of data in a specific order. There are many types of sorting algorithms, each with their own strengths and weaknesses, making them suitable for different use cases and applications.

Sorting algorithms are used in a wide range of applications, from simple tasks such as sorting a list of names alphabetically to more complex tasks such as sorting large datasets for analysis. They are also used in a variety of fields, including computer science, finance and data analysis.

Sorting algorithms can be broadly classified into two categories: comparison-based sorting algorithms and non-comparison-based sorting algorithms. Comparison-based sorting algorithms compare elements of the collection being sorted and determine their relative order. Examples of comparison-based sorting algorithms include Bubble Sort, Quick Sort and Merge Sort. Non-comparison-based sorting algorithms do not compare elements but instead make assumptions about the data being sorted. Examples of non-comparison-based sorting algorithms include Counting Sort and Radix Sort.

In this project we will be discussing about Bubble sort algorithm, one of the simplest sorting algorithms out there and take you through the process how we achieved its visualization through code implementations.

- **Bubble sort**

Bubble Sort is a simple sorting algorithm that works by repeatedly swapping adjacent elements if they are in the wrong order. It is named as "Bubble" because the smaller or larger elements (depending on the need) "bubble" to the top of the list during each pass. Bubble Sort is a comparison-based algorithm, meaning that it compares pairs of elements to determine their relative order and then swaps them if necessary.

Bubble Sort is not an efficient sorting algorithm compared to other sorting algorithms like Quick Sort and Merge Sort, which have better time complexity in the average case. Bubble Sort has a worst-case and average-case time complexity of O(n^2), where n is the number of elements in the list to be sorted. This means that as the number of elements to be sorted increases, the time taken to sort them using Bubble Sort increases exponentially.

Despite its inefficiency, Bubble Sort is still commonly taught in computer science courses and used in practice for small lists or as a teaching tool. Its simplicity and ease of implementation make it an excellent choice for learning about sorting algorithms and understanding their basic concepts.

- **Why choose Python and Tkinter?**

Python is a popular and versatile programming language that is used in a wide range of applications, from web development to scientific computing. It is known for its ease of use and readability, making it a great choice for beginners and experienced developers alike. Additionally, Python has a large and supportive community, with many resources and libraries available to developers.

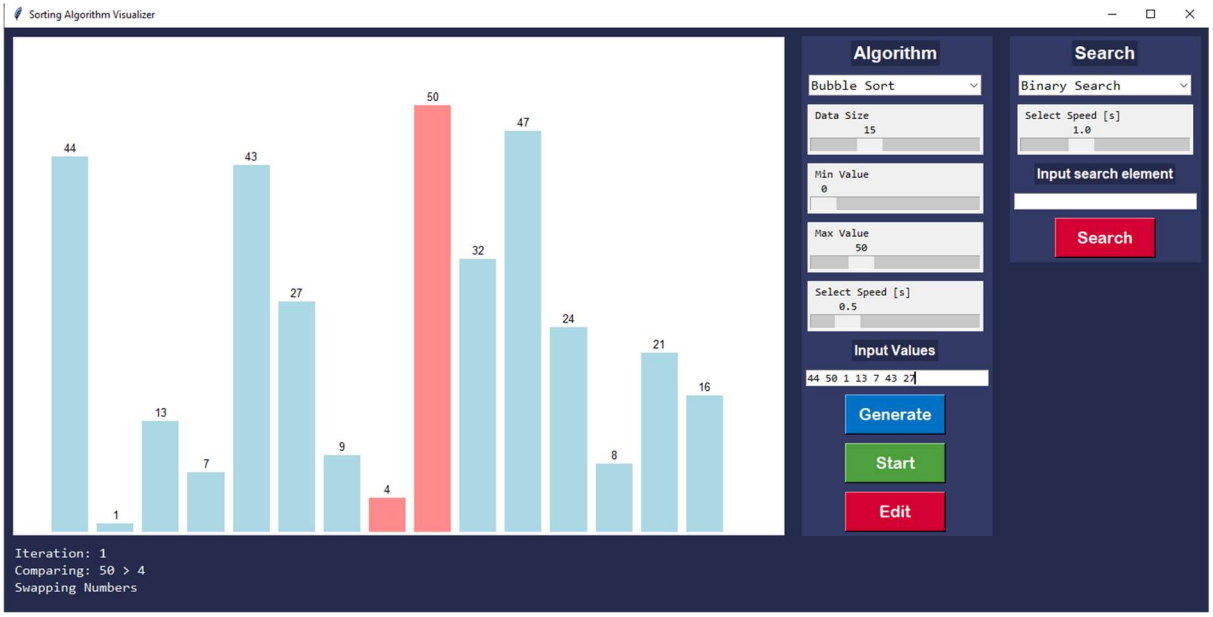
Tkinter is a Python library for creating graphical user interfaces (GUIs). It is easy to use and comes bundled with Python, so there is no need for additional installation. Tkinter provides a variety of widgets, such as buttons, labels and canvas, that can be used to create interactive and dynamic interfaces. It also allows for easy integration with other Python libraries, such as NumPy and Matplotlib, which can be used for data analysis and visualization.

For the Bubble sort visualization project, Python and Tkinter were a good choice because they allowed for easy creation of an interactive GUI that could display the sorting process in real-time. Tkinter's canvas widget was particularly useful for drawing and updating the graphical representation of the sorting algorithm. Creation of different widgets based on our needs was very much shorten and simplified. Additionally, Python's simplicity and readability made it easy for the team to collaborate and write code that was easy to understand and maintain.

- **Project Overview**

The Bubble sort visualization project is a Python program that uses the Tkinter library to create a graphical user interface (GUI) for the Bubble sort algorithm. The program consists of a canvas, which is the main visual component of the project and two widget sections on the right side of the canvas.

The canvas displays a set of boxes representing the data to be sorted. The algorithm is visualized by moving the boxes around to their correct position during the sorting process. The boxes are color-coded to show their current state during the sorting process, such as being swapped or already sorted. The canvas also displays text annotations that describe the current state of the sorting process.



The first widget section on the right side of the canvas is used to modify the algorithm and data. It contains sliders that allow the user to adjust the data size, minimum value, maximum value and speed of the sorting process. The user can also manually input data into an input section, or generate random data using a generate button. The generate button also creates the boxes on the canvas to represent the generated data. Once the data is generated, the user can start the sorting process by clicking on a start button. An edit button also appears after the generate button is clicked, which opens a new window allowing the user to add, delete, or modify the current generated data.

The second widget section is a search widget that allows the user to choose from several different search algorithms. The user can adjust the speed of the search using a slider and enter the search element using an input section. Once the search is started using a button, the algorithm is visualized on the canvas and text annotations describe the current state of the search process.

At the bottom of the canvas, there is a listbox that displays the sorting or searching process happening in text. The listbox shows the current state of the data and the actions being taken by the algorithm during the sorting or searching process. This provides an additional way for the user to follow the algorithmic process.

Overall, the Bubble sort visualization project provides an interactive and intuitive way to understand and visualize the Bubble sort algorithm. It allows the user to modify the data and the algorithm parameters, search for specific elements within the data and follow the algorithmic process in real-time using visual and text-based feedback.

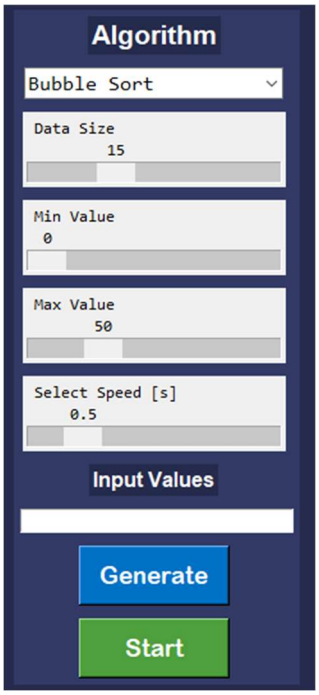
Methodology

- **Initial setups**

First, we needed to understand how the Bubble Sort algorithm works. We researched and studied the algorithm and identified the steps involved in it. We also learned about the complexities and limitations of the algorithm.

Then we had to start setting up our environment. We set up the environment by installing Python and the library that includes Tkinter. These libraries provided the tools we needed to create the graphical user interface and handle the data manipulation and visualization. We researched from various resources to understand how the initial program needs to be started and what is need to be done accordingly.

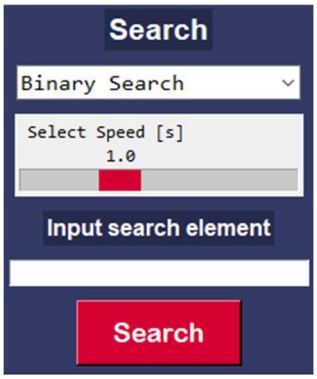
First, we started the project by simply creating a canvas and simple UI. We created a blank white canvas and added some buttons to generate random data. Then we added the necessary code to draw the rectangles in the canvas. Creating canvas and buttons is very much easy in Tkinter as all we had to do was to call the pre-build functions with the necessary parameters and Tkinter almost build the widgets for us. Initially There was no input section and we didn't add the sorting algorithm yet.



- **Handling user input**

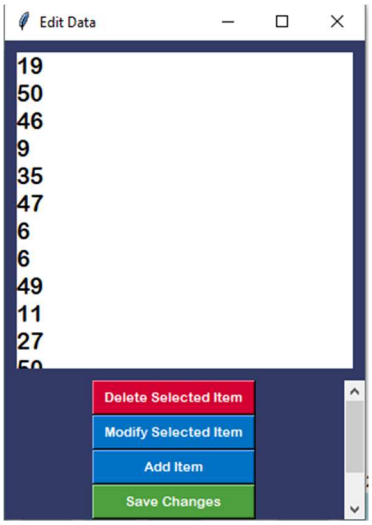
Though our project could create data, it was totally at random. We didn't have any user input section on it. And user input being one of the main necessities of the project we created a text box that takes manual input. We also added sliders that can change the data size, maximum value and minimum value. We didn't want the user to be bound just to use the random or the manual input. So, we mixed them both together and created an algorithm that uses both the functionalities together. We did have a section to choose which sorting algorithm they want to use so it can be later expanded. But for our project's sake we stuck to our bubble sort algorithm.

Coding the bubble sort algorithm itself wasn't very hard because of its simplicity. We created separate file for the different algorithms for making it easier to debug. We created a drawData function that that updates and draws the data and their boxes with their respective colours in the canvas.



- **Search, Edit and More**

We added search algorithms in the program. We created a separate widget section for the search algorithms, where users could choose the algorithm they want to use, set the speed of the search and input the search element. Adding the search section wasn't that much hassle because the resources it used was mostly already created beforehand.



The edit section in our program allows users to modify the generated data for sorting. After generating the data using the "Generate" button, the user can click on the "Edit" button, which opens a new window that displays the data in a list format.

From there, the user can add, delete, or modify individual data elements. The user can also choose to save the modified data or discard the changes.

This feature provides flexibility for the user and enables them to work with data that is relevant to their needs. It also allows them to experiment with different data sets and see how the Bubble Sort algorithm performs with different inputs.

One of the other necessary elements needed for our project was the real time text box that shows the iterations and the comparisons in a text-based scenario. So, we used the listbox function of the Tkinter to create a separate portion to show the process of the algorithm. It used the same technique as edit portion but without the buttons and stuff and it has to update in real time with according to the algorithm. For example, during the Bubble Sort algorithm, the listbox displays the current state of the data, such as which two elements are being compared and which one is being swapped. The listbox also displays the current step of the algorithm, such as "Comparing elements" or "Swapping elements."

This feature helps users to understand how the algorithm works and what is happening at each step. It also provides feedback to the user, letting them know that the program is running and working as expected.

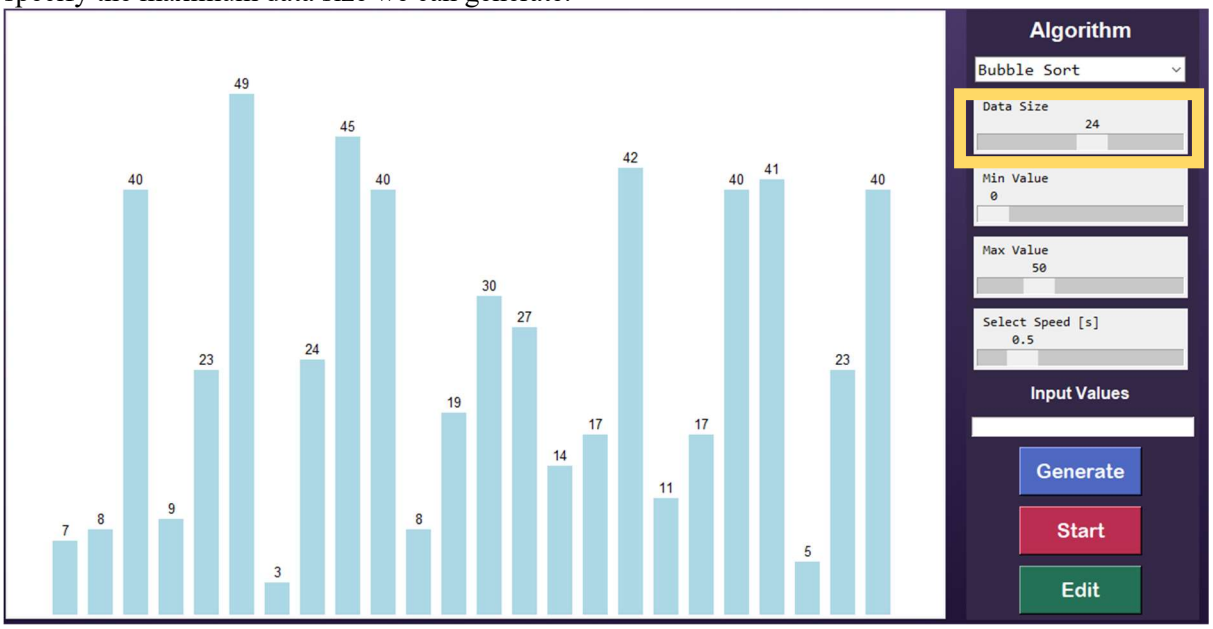
```
Iteration: 3
Comparing: 31 > 20
Swapping Numbers
```

Result and Discussion

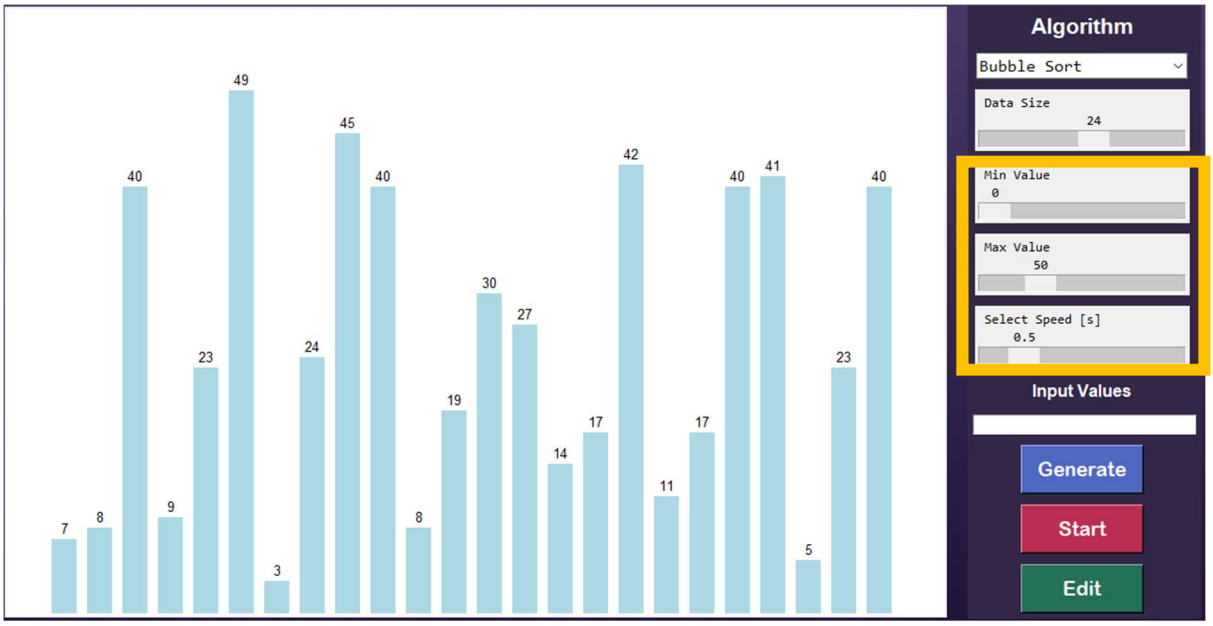
- **Input and Output**

Our project successfully visualizes the Bubble Sort algorithm using Tkinter and Python. Here are some screenshots of our visualization in action:

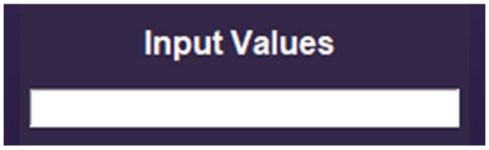
This is our landing page for our project. As you can see there are several options to choose from in this project. There are various ways to generate the data also. Using the Data size slider, we can specify the maximum data size we can generate.



The Min Value and Max Value Determines the limit of the data and the speed scale determines how fast or slow the animation works.

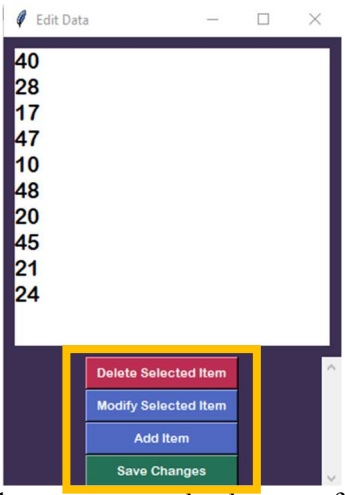


We also have an option to manually input values in the algorithm.

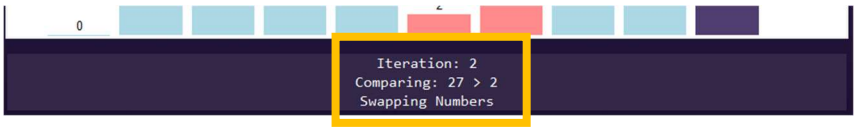


The Generate button generates random values and the Start button starts the sorting algorithm. We also have an Edit button that opens up a new window.

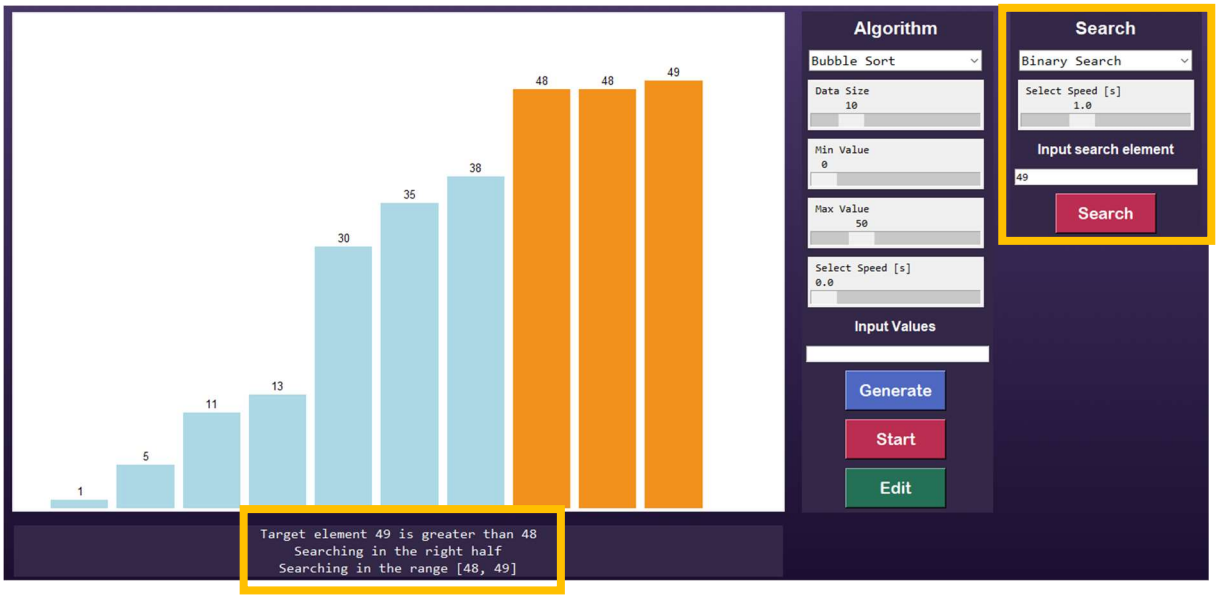
This edit window lets the user to add, modify or delete any data in the list. It also has an option to save the changes or discard the changes depending on the needs of the users.



There is also a listbox underneath the canvas to make the users follow along the algorithm using real time text. It shows the number of iterations and also the data which are being swapped and compared at every part of the algorithm.



There is also a section to search values based on the sorted elements. We added both linear and binary search to the application and added a different widget section for them.



Using all these widgets and modifications we can run the visualizations according to the user's needs.

As shown in the screenshots, the program generates a set of data as boxes on a canvas and the Bubble Sort algorithm is applied to sort the data. The listbox at the bottom of the screen displays the current step of the algorithm and the state of the data. The program also allows users to search elements using the sorted data and binary search.

- **Discussion**

Our visualization of the Bubble Sort algorithm provides several benefits in helping users to understand how the algorithm works.

Firstly, the use of visual aids, such as boxes representing data elements and animation of swapping and comparing, makes it easier for users to follow the sorting process. This can be particularly helpful for users who are new to programming or algorithms and might struggle to understand abstract concepts.

Secondly, the listbox at the bottom of the screen provides a log of what is happening at each step, making it easier for users to see the progress of the algorithm and identify any issues or errors. The log can also be useful for debugging or troubleshooting purposes.

Lastly, the ability to modify the data and adjust the speed of the algorithm allows users to experiment with different scenarios and see how the algorithm behaves under different conditions. This can be helpful in gaining a deeper understanding of the algorithm and its limitations.

- **Challenges**

During the course of our project, we encountered several challenges that required us to think creatively and problem-solve in order to overcome them.

First challenge we faced was working with the Tkinter library, which we had limited experience with prior to the project. We overcame this challenge by conducting research and seeking guidance from more experienced programmers. We also collaborated closely as a team and worked together to solve problems and debug the code.

Another challenge we faced was to optimize everything together and making sure that none of the different algorithms collided with each other and made issues along the way. We added a lot of customization option and making sure that every option works together perfectly is no less feat.

Lastly, we faced a challenge in ensuring that our program was user-friendly and accessible. We overcame this challenge by soliciting feedback from other team members and our lecturer and iterating on our design to make it more intuitive and easier to use. We also added features, such as the ability to modify data and adjust the speed of the algorithm, that increased the flexibility and usefulness of the program.

Conclusion

Our project successfully visualized the Bubble Sort algorithm using Tkinter and Python and provided users with a helpful tool for understanding how the algorithm works. By breaking down the algorithm into smaller steps and creating visual aids, we were able to make it easier for users to follow along and gain a deeper understanding of the sorting process. Our project also demonstrated the importance of visualizing algorithms for better understanding and provided a framework for future research and improvement.

For now, the visualizer only includes the bubble sort algorithm. But our team has worked hard to make such a framework that any sorting algorithm can be added and modified according to its need. And we will make sure to improve it further down the road as we add more complex algorithm into the mix. By building on our findings and continuing to innovate in this area, we can help to make complex algorithms more accessible and understandable to a wider audience.

Throughout the course of this project, we gained a number of valuable insights and learned several important lessons. One of the most significant lessons we learned was the importance of collaboration and communication when working on a team-based project. By working closely together, we were able to identify each team member's strengths and weaknesses and assign tasks accordingly. This helped us to stay organized and work more efficiently towards our shared goal.

Another important lesson we learned was the importance of breaking down complex algorithms into smaller, more manageable steps. By creating visual aids and breaking down the Bubble Sort algorithm into individual steps, we were able to help users better understand the sorting process and follow along more easily. This approach could be applied to other complex algorithms or programming concepts to help make them more accessible to learners.

Finally, we learned the importance of flexibility and adaptability when working on a programming project. As we encountered challenges or obstacles, we had to be willing to pivot and adjust our approach in order to find a solution that worked for us. This required us to be creative and to think outside the box, which helped us to become better problem-solvers and more versatile programmers.

Overall, our project taught us a number of valuable lessons about collaboration, communication, algorithm visualization and adaptability. These lessons will be valuable not just in our future programming projects, but also in our professional and personal lives more broadly.

Developed by

- MD. Nahid Hasan

ID: 20212041010

- Suchana Biswas Cheity

ID: 20212003010

- Limon Majumdar

ID: 20212011010

Department Of CSE,

North Western University Khulna, Bangladesh