

BUBBLE SORTING
A PROJECT REPORT

Submitted by

MOHIT SIDDHARTH BAGGU [Reg No: RA2211031010108]

RAGHUKARTHIKEYAN[Reg No: RA2211031010095]

PRANOV[Reg No:RA2211031010127]

LOKESH S[Reg No:RA2211031010094]

Under the Guidance of

DR. Praveena Akki

Associate Professor, Department of Network and Communications

In partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

with a specialization in INFORMATION TECHNOLOGY



DEPARTMENT OF NETWORK AND COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGYSRM

INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

NOV 2023



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603

203

BONAFIDE CERTIFICATE

Certified that this B.Tech project report titled “**BUBBLE SORTING**” is the bonafide work of MOHIT SIDDHARTH BAGGU [Reg. No.: RA2211031010108] , RAGHUKARTHIKEYAN [Reg. No. RA2211031010095] , LOKESH S [RA2211031010094] and PRANOV [Reg. No: RA2211031010127] who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

DR. PRAVEENA AKKI

SUPERVISOR

Associate Professor

Department of Network and communications

DR. K. ANNAPOORNI

HEAD OF THE DEPARTMENT

Department of Network and Communications

**SIGNATURE OF INTERNAL
EXAMINER**

**SIGNATURE OF EXTERNAL
EXAMINER**



Department of Network and Communications

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/ Course : B.Tech in Computer Science and Engineering with a
specialization in Information Technology

Student Names : Mohit Siddharth Baggu, Raghukarthikeyan, Lokesh S, Pranov

Registration Number: RA2211031010108, RA2211031010095, RA2211031010094,
RA2211031010127

Title of Work : Bubble Sorting

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own

- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T.V.Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. Annapoorni**, Professor, Department of Network and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinator, **Dr. Praveenaa Akki**, Assistant Professor, Department of Network and Communications, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. P.Gowthaman**, Assistant Professor, Department of Network and Communications, SRM Institute of Science and Technology, for leading and helping us to complete our course.

We sincerely thank the Network and communications Department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Mohit Siddharth Baggu [RA2211031010108]

Raghukarthikeyan [RA2211031010095]

Pranov [RA2211031010127]

Lokesh S[RA2211031010094]

CONTENT TABLE:

S.NO	TOPIC	PG NO.
1	Abstract	1
2	Project Statement	2
3	Algorithm	3
3	Complexity Analysis	5
4	Advantages and Disadvantages	6
5	Code and Explanation	7
6	Code	8
7	Output	12
8	Conclusion	18

ABSTRACT

This Python code is a graphical user interface (GUI) program built using the Tkinter library. It is a "Bubble Sorting Visualizer" that allows users to input a list of numbers separated by spaces, then sorts and visualizes the sorting process using the bubble sort algorithm. Here's a more detailed breakdown of the code:

1. ****Import Libraries****:

- The code imports the necessary libraries, including Tkinter for GUI, random for generating random data, and time for adding delays in the visualization.

2. ****Bubble Sort Algorithm****:

- The ``bubble_sort`` function takes an array, a canvas for visualization, and a label for displaying the sorted array. It sorts the input array using the bubble sort algorithm while updating the canvas to visualize the sorting process and the label to show the sorted array.

3. ****Visualization Functions****:

- ``drawBars``: This function is used to draw bars on the canvas to represent the elements in the array. It clears the canvas and then draws colored bars for each element.
- ``update_label``: This function updates the label to display the sorted array.

4. ****start_sorting Function****:

- This function is called when the "Start Sorting" button is clicked. It reads the input from the user, parses it into an array of integers, and calls the ``bubble_sort`` function to sort and visualize the array. It also updates the `info_label` to indicate the completion of the sorting process.

5. ****GUI Setup****:

- The code creates a Tkinter window and sets its title and dimensions.
- It creates a canvas for drawing the bars representing the array elements.
- It includes an entry field where the user can input a space-separated list of numbers.
- The "Start Sorting" button triggers the ``start_sorting`` function.
- Labels are used to display information and the sorted array.

6. ****Main Loop****:

- The ``root.mainloop()`` call starts the Tkinter main event loop, allowing the GUI to interact with the user.

Project Statement

This Python code implements a Bubble Sorting Visualizer using the Tkinter library. The program is designed to allow users to input a list of numbers separated by spaces and then visualize the sorting process using the bubble sort algorithm. Here's a brief project statement:

The "Bubble Sorting Visualizer" is a simple interactive program that serves as an educational tool to help users understand the bubble sort algorithm. Bubble sort is a straightforward sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This visualization tool provides a user-friendly interface for observing the sorting process step by step

The program provides a graphical user interface (GUI) created with Tkinter. Users can input a list of integers separated by spaces in the entry field. Upon clicking the "Start Sorting" button, the algorithm kicks in, and the sorting process is visually represented on a canvas.

The canvas displays bars representing the elements of the input array. The height and position of each bar correspond to the values in the array. As the sorting algorithm progresses, the bars move into their sorted positions. The visual representation makes it easy for users to follow how the bubble sort algorithm works and understand the concept of sorting.

The program provides real-time feedback by updating a label to display the sorted array and an information label to indicate when the sorting process is completed. If the user provides an invalid input (e.g., non-numeric characters), the tool informs them of the error.

This Bubble Sorting Visualizer is an educational resource for individuals learning about sorting algorithms, programming, or data structures. It offers a hands-on experience of a sorting algorithm in action, making it a valuable tool for teaching and learning the concept of sorting.

The Bubble Sorting Visualizer project combines programming, user interface design, and educational value. It is a useful tool for both beginners and educators to explore and understand the bubble sort algorithm in a visually intuitive way. The program's simplicity and interactivity make it a valuable addition to the learning resources for computer science and programming enthusiasts.

ALGORITHM

The provided code implements a Bubble Sorting Visualizer using the Tkinter library, allowing users to input a list of numbers and visualize the bubble sort algorithm. Here's a high-level algorithmic description of the code in six brief paragraphs:

1. **Initialization**:

- The program initializes a Tkinter GUI window, creates a canvas for visualization, and sets up labels for user feedback.

2. **Input and Validation**:

- Users enter a list of integers separated by spaces in the input field. The `start_sorting` function reads this input, attempting to convert it into a list of integers. It handles any potential `ValueError` (invalid input) by displaying an error message if the input is not a valid list of numbers.

3. **Bubble Sort**:

- The core of the program is the `bubble_sort` function. This function sorts the input list using the bubble sort algorithm. It iterates through the list, comparing adjacent elements and swapping them if they are out of order. After each comparison and swap, it calls the `drawBars` and `update_label` functions to update the visualization and display the current state of the array.

4. **Visualization**:

- The `drawBars` function clears the canvas and then visually represents the elements in the list as bars of varying heights, with the height and color of each bar reflecting the corresponding value in the array. This dynamic visualization updates in real-time as the sorting algorithm progresses.

5. **User Feedback**:

- The program provides feedback through labels. The `info_label` informs the user when the sorting process is completed. The `sorted_label` displays the sorted array as it gets updated with each step of the sorting algorithm, allowing users to track the sorting progress.

6. **Main Loop**:

- Finally, the code enters the Tkinter main event loop (`root.mainloop()`), making the GUI interactive and responsive to user actions. Users can input their data, initiate the sorting process, and observe the bubble sort algorithm in action through the graphical visualization.

Complexity Analysis

1. ****Bubble Sort Time Complexity****:

- The primary algorithm in this code is the bubble sort. In the worst case, bubble sort has a time complexity of $O(n^2)$ because it involves nested loops. The outer loop runs for 'n' iterations, and for each iteration, the inner loop runs 'n-i' times, where 'i' is the current iteration. This results in a quadratic time complexity. However, in the best case, if the input is already sorted, the time complexity is $O(n)$ because no swaps are needed.

2. ****Canvas Updates and Sleep****:

- The `drawBars` function is used to update the canvas with the current state of the array. The time complexity of this function is $O(n)$ because it iterates through the array once. The `canvas.updateIdletasks()` operation has a negligible time complexity and is mainly for responsiveness.

- The `time.sleep(0.1)` statement introduces a fixed delay, but it doesn't significantly impact the overall time complexity, as it's a constant factor.

3. ****Space Complexity****:

- The space complexity of the code is primarily determined by the data structures used. The array 'arr' and the canvas are the primary memory consumers. Since the code doesn't use any additional data structures or recursive function calls, the space complexity is $O(n)$, where 'n' is the number of elements in the input array.

4. ****User Interface Overhead****:

- The space and time complexity associated with the user interface (UI) components (e.g., labels, buttons, and the entry field) is generally low and constant because the number of UI elements doesn't depend on the size of the input array.

5. ****Interactive Delay****:

- The time complexity of the interactive part (mainloop) largely depends on user interaction. It is an event-driven system that responds to user input and redraws the UI as necessary. The responsiveness of the UI is essential for a positive user experience.

6. ****Educational Use****:

- While the code itself doesn't have a high computational complexity, its primary purpose is educational. It provides a hands-on way for users to understand and visualize how the bubble sort algorithm works. The educational value of the tool is more significant than its computational complexity, making it a valuable resource for learning about sorting algorithms and programming.

Advantages and Disadvantages

Advantages:

1. **Educational Tool**: The code serves as an educational tool, helping users understand the bubble sort algorithm through interactive visualization. It offers a practical way to learn about sorting algorithms.
2. **User-Friendly Interface**: The graphical user interface (GUI) is user-friendly, making it accessible for individuals who may not have a programming background. Users can input data, start the sorting process, and observe the algorithm in action with ease.
3. **Real-Time Feedback**: The program provides real-time feedback through the canvas and labels. Users can see the sorting progress and observe how the array is changing step by step. This immediate visual feedback enhances the learning experience.
4. **Customization**: The code is easily customizable. Users can change the delay time between steps, the colors used for visualization, and the canvas dimensions. This flexibility allows for a personalized learning experience.

Disadvantages:

1. **Limited Sorting Algorithm**: The code specifically focuses on the bubble sort algorithm. While it's excellent for educational purposes, it doesn't offer the versatility to explore and visualize other sorting algorithms. Adding more algorithms would require significant modifications.
2. **Inefficient for Large Datasets**: Bubble sort is not an efficient sorting algorithm for large datasets. It has a worst-case time complexity of $O(n^2)$, which can result in slow sorting for sizable input arrays. Users might encounter performance issues when dealing with larger datasets.
3. **Blocking User Interface**: The program's use of `time.sleep(0.1)` in the sorting process introduces a fixed delay. While this is useful for visualization, it can block the user interface during the sorting operation. Users may experience unresponsiveness during the sorting process, especially for large arrays.
4. **Simplified User Interface**: While the user interface is straightforward and functional, it may lack advanced features found in more robust data visualization tools.

Code and Explanation

This function takes three parameters: an array (`arr`), a canvas (`canvas`), and a label (`label`). It implements the bubble sort algorithm to sort the input array. Bubble sort repeatedly iterates through the array, comparing adjacent elements, and swapping them if they are out of order. After each comparison and swap, it calls the `drawBars` and `update_label` functions to visually update the canvas and display the current state of the array. It introduces a small time delay (0.1 seconds) between each step for visualization.

This function takes the array and the canvas as parameters. It clears the canvas and draws bars to represent the elements of the array. The height and color of each bar correspond to the values in the array, creating a visual representation of the data. This function takes the sorted array and a label as parameters. It updates the label to display the current state of the sorted array, providing real-time feedback to the user.

This function is called when the "Start Sorting" button is pressed. It retrieves the input array from the user's input, splits it into integers, and performs error handling to handle invalid input. It then initiates the bubble sort algorithm with the input array and updates labels to indicate the sorting process's completion.

The code initializes a Tkinter window, sets its title, and defines its dimensions. It creates a canvas within the window where the sorting visualization is displayed. There's an entry field for users to input a space-separated list of numbers, a "Start Sorting" button to initiate the sorting process, and labels to provide information and display the sorted array.

This line of code starts the main event loop, allowing the GUI to be interactive and responsive to user actions. Users can input data, trigger the sorting process, and watch the bubble sort algorithm in action through the graphical visualization.

Code

```
import tkinter as tk
import random
import time

def bubble_sort(arr, canvas, label):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                drawBars(arr, canvas)
                update_label(arr, label)
                canvas.update_idletasks()
                time.sleep(0.1)

def drawBars(arr, canvas):
    canvas.delete("all")
    canvas_width = 400
    canvas_height = 200
    bar_width = canvas_width / len(arr)
    max_val = max(arr)

    for i, val in enumerate(arr):
        bar_height = (val / max_val) * (canvas_height - 20)
        x1 = i * bar_width
        y1 = canvas_height - bar_height
        x2 = (i + 1) * bar_width
        y2 = canvas_height
        canvas.create_rectangle(x1, y1, x2, y2, fill="cyan")

def update_label(arr, label):
    label.config(text="Sorted Array: " + " ".join(map(str, arr)))

def start_sorting():
    input_arr = entry.get()
    try:
        arr = [int(x) for x in input_arr.split()]
```

```
except ValueError:
    info_label.config(text="Invalid input.")
    return
bubble_sort(arr, canvas, sorted_label)
info_label.config(text="Sorting completed")

root = tk.Tk()
root.title("Bubble Sorting Visualizer")
root.geometry("500x400")

canvas = tk.Canvas(root, width=400, height=200)
canvas.pack()

entry_label = tk.Label(root, text="Enter numbers separated by spaces:")
entry_label.pack()
entry = tk.Entry(root)
entry.pack()

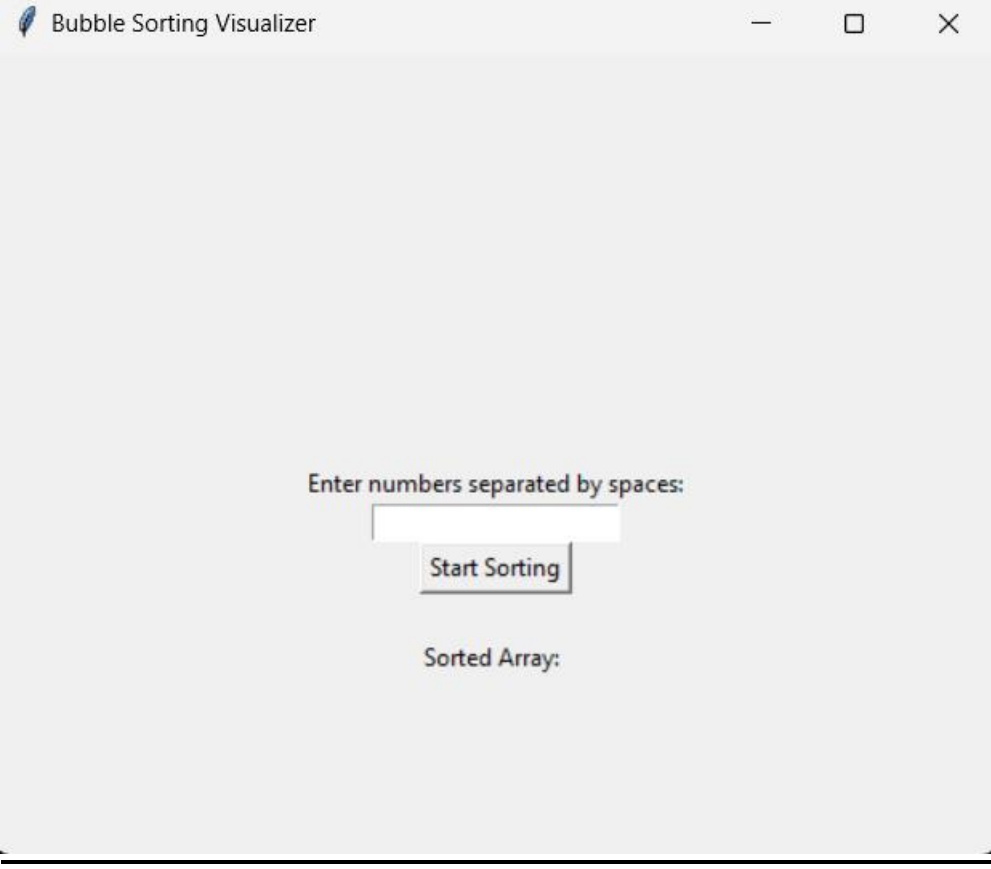
start_button = tk.Button(root, text="Start Sorting", command=start_sorting)
start_button.pack()

info_label = tk.Label(root, text="")
info_label.pack()

sorted_label = tk.Label(root, text="Sorted Array: ")
sorted_label.pack()

root.mainloop()
```

Output



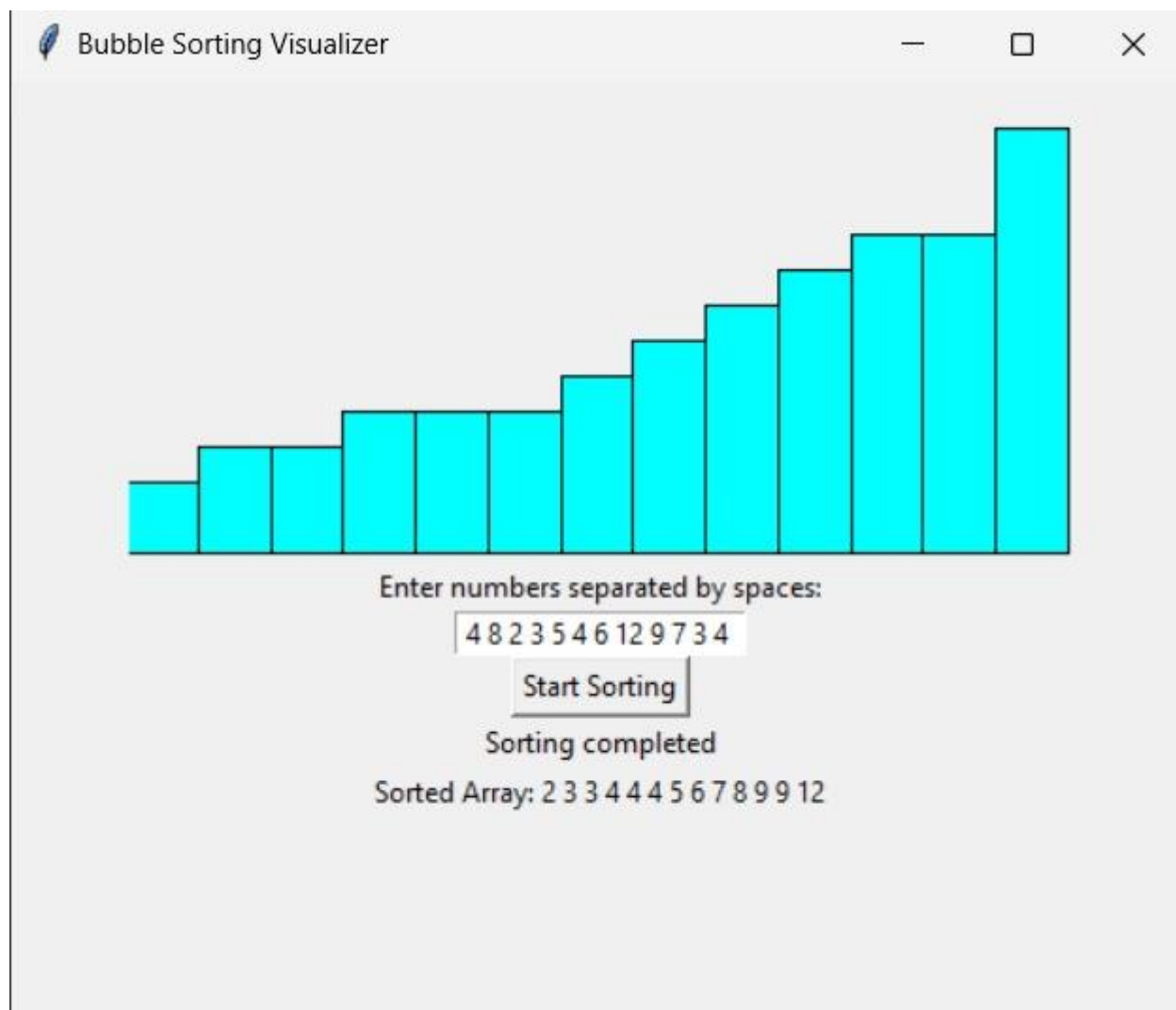
A screenshot of a web application window titled "Bubble Sorting Visualizer". The window has a light gray background and standard window controls (minimize, maximize, close) in the top right corner. In the center of the window, there is a text prompt "Enter numbers separated by spaces:" followed by a white input field. Below the input field is a button labeled "Start Sorting". Further down, the text "Sorted Array:" is displayed.

Bubble Sorting Visualizer

Enter numbers separated by spaces:

Start Sorting

Sorted Array:



Conclusion

In conclusion, this code presents a practical and educational tool for visualizing the bubble sort algorithm using a Tkinter-based graphical user interface (GUI). The program allows users to input a list of numbers, initiates the sorting process, and provides a real-time visualization of how the bubble sort algorithm works. Here are the key points to emphasize:

The code is designed with education in mind. It enables individuals, particularly those new to programming and sorting algorithms, to interactively learn how the bubble sort algorithm functions. The dynamic visualization of the sorting process fosters a deep understanding of the algorithm's step-by-step operation. The GUI is user-friendly, making it accessible to a wide range of users. Users can input data easily, start the sorting process with a single click, and observe the sorting progress through the visual representation of the data.

The code provides real-time feedback by updating labels and the canvas to reflect the sorting process. Users can monitor how the array changes throughout the sorting operation, enhancing their comprehension of the algorithm. The program offers customization options for the delay time between steps and visual elements such as colors and canvas dimensions. This flexibility allows users to tailor their learning experience.

It's essential to recognize that the code primarily focuses on the bubble sort algorithm. While suitable for educational purposes, it may not be the best choice for large datasets due to the algorithm's inefficiency for such cases. Additionally, the interface is relatively simple, lacking advanced features found in more comprehensive data visualization tools.