BUBBLE SORT VISUALIZER

Presented by

RA2211003010959 - SRI SHAKTHI SARATH CHINTAPALLI

RA2211003010966 - SATWIK DONTHAMSETTY

RA2211003010994 - B VISHNUPRIYA

TABLE OF CONTENTS

01 Problem Statement

02 Objective

03 Application

04 Solution and Algorithm

05 Code Snippets

06 Output

07 References

PROBLEM STATEMENT

To Create a Bubble Sort Visualization Tool with an intuitive interface for users to input and observe sorting processes interactively. Use animations to illustrate element comparisons and swaps. Includes customizable speed options and performance metrics, making it an educational and user-friendly tool for understanding the Bubble Sort algorithm.

OBJECTIVE

The objective of the Bubble Sort Visualization Tool is to provide users with an interactive and visually engaging platform to comprehend the Bubble Sort algorithm. Through intuitive controls, real-time animations, and customizable features, the tool aims to enhance understanding and education about sorting processes in a user-friendly manner.

By offering step-by-step explanations, performance metrics, and customization options, the tool facilitates a deeper grasp of sorting principles. It caters to users with varying levels of programming knowledge, fostering a comprehensive understanding of Bubble Sort's efficiency and limitations. The objective is to create an accessible and effective educational resource for learning sorting algorithms.

APPLICATION

The Bubble Sort Visualization Tool finds application in educational settings, coding boot camps, and self-paced learning environments. It serves as a hands-on educational resource for individuals looking to understand sorting algorithms, making computer science concepts more accessible. The tool's interactive features make it valuable for both beginners and intermediate learners, fostering a practical understanding of algorithmic concepts in a visually intuitive manner.

Cont.

Additionally, the tool can be utilized in workshops and coding courses to supplement theoretical knowledge with practical demonstrations. It aids educators in illustrating algorithmic concepts effectively, fostering a dynamic and engaging learning experience. The application extends to programming enthusiasts and professionals seeking a quick, visual reference for sorting algorithms, allowing them to experiment with different datasets and gain insights into algorithmic efficiency.

Cont.

Additionally, the tool can be utilized in workshops and coding courses to supplement theoretical knowledge with practical demonstrations. It aids educators in illustrating algorithmic concepts effectively, fostering a dynamic and engaging learning experience. The application extends to programming enthusiasts and professionals seeking a quick, visual reference for sorting algorithms, allowing them to experiment with different datasets and gain insights into algorithmic efficiency.

SOLUTION AND ALGORITHM

To Create a Bubble Sort Visualization Tool using a graphical user interface (GUI) framework like Tkinter in Python. We design an interactive interface for user input and control, integrating real-time visualizations with animations to depict the Bubble Sort algorithm's comparisons and swaps. Implement customization options for speed and input data, and display performance metrics during sorting. Utilize a step-by-step explanation feature for educational value.

SOLUTION AND ALGORITHM

For the algorithm, We use a basic Bubble Sort implementation in Python:

```
def bubble_sort(arr):
    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]
```

CODE SNIPPETS

Header File:

```
#ifndef BUBBLE_SORT_VISUALIZER_H
#define BUBBLE_SORT_VISUALIZER_H

void startSorting(const char
    *input_text);
void generateRandomValues();
void resetInput();

#endif
```

CODE SNIPPETS

Main:

```c
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <stdio.h>

#define MAX_ARRAY_SIZE 100

int arr[MAX_ARRAY_SIZE];
int n;
int sorting_in_progress = 0;

void bubbleSort(int arr[], int n) {
    int temp, swapped;

    for (int i = 0; i < n - 1; i++)
        {
        swapped = 0;

        for (int j = 0; j < n - 1 -
            i; j++) {
            if (arr[j] > arr[j + 1]
                ) {
```

```c
                swapped = 1;
            }
        }

        if (swapped == 0) {
            break;
        }
    }

    sorting_in_progress = 0;
}

void startSorting(const char
    *input_text) {
    if (sorting_in_progress) {
        return;
    }

    sorting_in_progress = 1;

    int num;
    int count = 0;
    char input_text_copy[25
    strcpy(input_text_copy,
```

Run

CODE SNIPPETS

```c
    srand(time(NULL));
    char random_values[256] = "";

    for (int i = 0; i < 10; i++) {
        if (i > 0) {
            strcat(random_values, " ");
        }
        int random_value = rand() % 100;
        char temp_value[16];
        sprintf(temp_value, "%d", random_value);
        strcat(random_values, temp_value);
    }

    startSorting(random_values);
}

void resetInput() {
    sorting_in_progress = 0;
    memset(arr, 0, sizeof(a
```

Run

CODE SNIPPETS

```python
import ctypes
import tkinter as tk
import random
import time

# Load the shared C library
bubble_sort = ctypes.CDLL('./libbubblesort.so')  # Replace with the actual path to your shared library

# Define the function signature for C functions
bubble_sort.startSorting.argtypes = [ctypes.c_char_p]
bubble_sort.startSorting.restype = None
bubble_sort.generateRandomValues.argtypes = []
bubble_sort.generateRandomValues.restype = None
bubble_sort.resetInput.argtypes = []
```

```python
    None

def start_sorting():
    input_text = entry.get()
    bubble_sort.startSorting(input_text.encode('utf-8'))
    visualize_sorting(input_text)

def visualize_sorting(input_text):
    numbers = list(map(int, input_text.split()))
    n = len(numbers)

    for i in range(n):
        for j in range(0, n-i-1):
            if numbers[j] > numbers[j+1]:
                numbers[j], numbers[j+1] = numbers[j+1], numbers[j]
                entry.delete(0, tk
```

CODE SNIPPETS

```
                    '.join(map(str,
                    numbers)))  #
                    Update the input
                    field
            draw_bars(numbers,
                    j, j + 1)
            time.sleep(0.2)  #
                    Adjust the delay
                    for visualization
            draw_bars(numbers,
                    -1, -1)  # Reset
                    color to red

def draw_bars(numbers,
    highlight_index1,
    highlight_index2):
    canvas.delete('all')
    canvas_width = canvas
        .winfo_width()
    canvas_height = canvas
        .winfo_height()
    bar_width = canvas_width / len
```

```
        y0 = canvas_height
        x1 = (i + 1) * bar_width
        y1 = canvas_height -
            number *
            (canvas_height / 100)
             # Scale the bar
            heights for
            visualization
        if i == highlight_index1
            or i ==
            highlight_index2:
            canvas
                .create_rectangle
                (x0, y0, x1, y1,
                fill="green")
        else:
            canvas
                .create_rectangle
                (x0, y0, x1, y1,
                fill="red")
    canvas.update()

def generate_random_values():
    random_values = [rando
```

Run

CODE SNIPPETS

```python
    entry.insert(0, ' '.join(map(str, random_values)))
    bubble_sort.generateRandomValues()
    visualize_sorting(' '.join(map(str, random_values)))

def reset_input():
    bubble_sort.resetInput()
    entry.delete(0, tk.END)
    canvas.delete('all')

# Create the Tkinter window
window = tk.Tk()
window.title("Bubble Sort Visualizer")

# Canvas for visualization
canvas = tk.Canvas(window, bg='white')
canvas.grid(row=0, column=0, columnspan=4, sticky="nsew")
```

Run

```python
    text="Enter numbers separated by spaces:")
entry_label.grid(row=1, column=0, columnspan=4, pady=10)
entry = tk.Entry(window, width=30)  # Slightly larger input field
entry.insert(0, "54 32 12 89 45")  # Initial input
entry.grid(row=2, column=0, columnspan=4)

# Start Sorting button
start_button = tk.Button(window, text="Start Sorting", command=start_sorting)
start_button.grid(row=3, column=0, columnspan=4, pady=10)

# Generate Random Values button
generate_random_button = tk.Button(window, text="Generate Random", command=generate_random_values
generate_random_button.grid
```

Run

CODE SNIPPETS

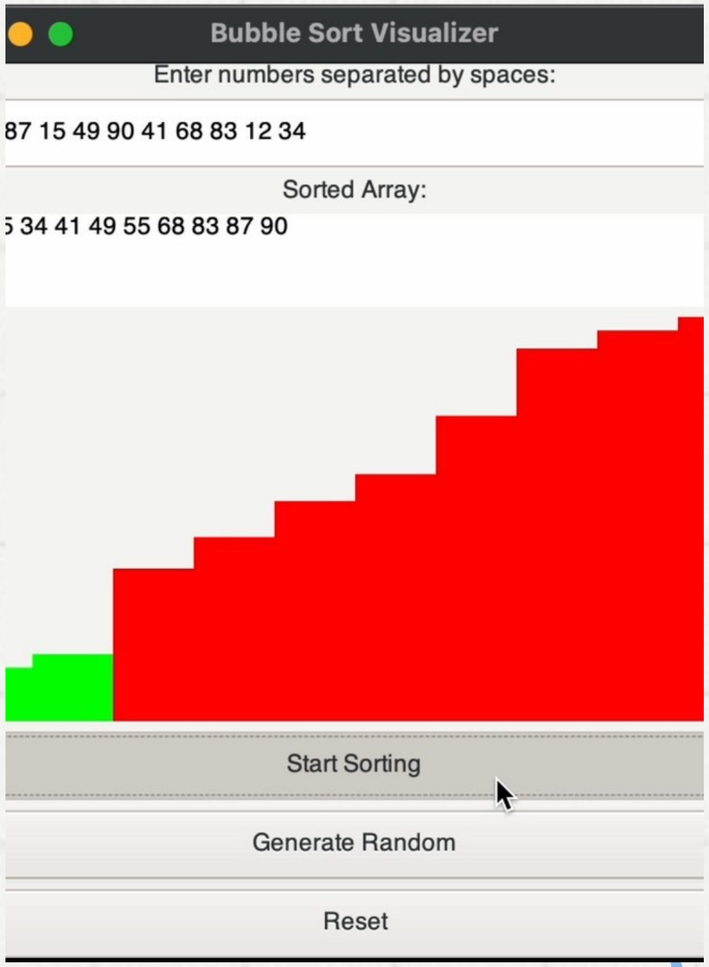
```
        column=0, columnspan=4, pady
        =10)

# Reset button
reset_button = tk.Button(window,
        text="Reset", command
        =reset_input)
reset_button.grid(row=5, column=0,
        columnspan=4, pady=10)

# Adjust row and column weights to
        allow resizing
for i in range(4):
    window.grid_columnconfigure(i,
            weight=1)
window.grid_rowconfigure(0, weight
        =1)

# Initially draw bars with current
        canvas size
draw_bars([int(num) for num in
        entry.get().split()], -1, -1)
```

Output


Bubble Sort Visualizer
Enter numbers separated by spaces:
87 15 49 90 41 68 83 12 34
Sorted Array:
5 34 41 49 55 68 83 87 90
Start Sorting
Generate Random
Reset

References

GUI Framework: Tkinter: Official documentation at https://docs.python.org/3/library/tkinter.html.

Bubble Sort Algorithm: Reference the algorithm on GeeksforGeeks: https://www.geeksforgeeks.org/bubble-sort/.

Real-time Visualization: Matplotlib for dynamic visualizations: https://matplotlib.org/.

Educational Resources: Khan Academy for algorithm explanations: https://www.khanacademy.org/computing/computer-science/algorithms/bubble-sort/a/bubble-sort.

Python Programming: Python official documentation: https://docs.python.org/3/.

W3Schools for Python: https://www.w3schools.com/python/.

Algorithm Visualization Techniques: "Algorithm Visualization" by H. Knut and M. Bykat: https://link.springer.com/book/10.1007/978-3-319-27261-1.

Coding Communities: Stack Overflow for community support: https://stackoverflow.com/.