Mini Project Report
On

Visualizing

BUBBLE SORT ALGORITHM USING PYGAME

Submitted By:

V. AAKANKSHA

Submitted To:

ExternsClub

ABSTRACT

The project utilizes the Pygame library to create an interactive visualization of the Bubble Sort algorithm. The program generates a random array of integers, represented as colored rectangles, and sorts them using the Bubble Sort method. The visualization displays the step-by-step process of the algorithm, allowing users to observe how the array is sorted. The project aims to provide an engaging and educational tool for understanding the Bubble Sort algorithm, making it accessible to a wider audience. By using Pygame, the project creates a dynamic and interactive experience, enabling users to pause, resume, and restart the sorting process.

CONTENTS

INTRODUCTION

Bubble Sort is a fundamental sorting algorithm that works by repeatedly comparing and swapping adjacent elements if they are in the wrong order. Although simple, its step-by-step nature makes it an excellent candidate for visualization. Using Pygame, this project visually represents the sorting process, helping users grasp the algorithm's mechanics dynamically.

SYSTEM REQUIREMENTS

Python Version: 3.7 or higher

Libraries: Pygame (Install via pip install pygame)

Hardware Requirements: Any system capable of running Python and rendering simple graphics.

FEATURES

Real-Time Visualization: Displays the sorting process step-by-step.

Dynamic Bar Heights: Bars represent data values, varying in height based on their value.

Color Indicators:

Red: Bars being compared.

Green: Sorted section.

Customizable Parameters: Users can modify data size, speed, and colors.

ARCHITECTURE OVERVIEW

Components:

1. Pygame Setup: Initializes the display and handles the rendering of bars.

2. Data Representation: Uses an array of integers to represent unsorted data.

3. Sorting Logic: Implements Bubble Sort with embedded visualization steps.

4. User Interaction: Allows the user to trigger sorting via keyboard input.

Flow:

1. Generate random data and draw unsorted bars.

2. Start sorting on user input.

3. Highlight comparisons and swaps dynamically.

4. Render the final sorted array.

IMPLEMENTATION

5.1 Setup Environment

Install Python and the Pygame library.

pip install pygame

5.2 Code Implementation

5.2.1 Import Libraries

import pygame

import random

import time

5.2.2 Initialize Pygame

pygame.init()

Screen dimensions

SCREEN_WIDTH = 800

SCREEN_HEIGHT = 600

screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))

pygame.display.set_caption("Bubble Sort Visualization")

5.2.3 Define Constants and Variables

```python
# Colors

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

RED = (255, 0, 0)

GREEN = (0, 255, 0)

# Data and visualization parameters

NUM_BARS = 50

bar_width = SCREEN_WIDTH // NUM_BARS

data = [random.randint(10, SCREEN_HEIGHT - 10) for _ in range(NUM_BARS)]

delay = 0.05   # Delay between steps for visualization
```

5.2.4 Draw Bars Function

This function visualizes the current state of the data array.

```python
def draw_bars(data, color_positions=None):

    screen.fill(WHITE)  # Clear the screen

    for i, value in enumerate(data):

        color = RED if color_positions and i in color_positions else BLACK

 pygame.draw.rect(screen, color, (i * bar_width, SCREEN_HEIGHT - value, bar_width - 2, value))

    pygame.display.update()
```

5.2.5 Bubble Sort Algorithm with Visualization

```python
def bubble_sort_visualization(data):

    n = len(data)
```

```python
    for i in range(n - 1):

        for j in range(n - i - 1):

            # Highlight bars being compared

            draw_bars(data, color_positions=[j, j + 1])

            time.sleep(delay)

# Swap if necessary

            if data[j] > data[j + 1]:

                data[j], data[j + 1] = data[j + 1], data[j]

                # Highlight the sorted section

        draw_bars(data, color_positions=list(range(n - i, n)))

    draw_bars(data)  # Final render of sorted data
```

5.2.6 Main Loop

```python
running = True

sorting = False

while running:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

running = False

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K_SPACE:  # Start sorting when SPACE is
pressed

                sorting = True

if sorting:

    bubble_sort_visualization(data)
```

```
        sorting = False

    draw_bars(data)

pygame.quit()
```

USAGE INSTRUCTIONS

1. Run the script in any Python IDE or terminal.

2. A window will appear displaying unsorted bars.

3. Press SPACE to begin sorting.

4. Observe the step-by-step sorting process.

CONCLUSION

This project demonstrates how to visualize sorting algorithms dynamically, enhancing understanding through graphical representation. Using Pygame, the Bubble Sort algorithm is brought to life, offering an engaging and educational experience. This framework can be expanded to visualize other algorithms, making it a versatile tool for algorithm learning.