Administrative

Team name: Team Seb Team: Sebastian Sosa (CakeCrusher) GitHub repo: https://github.com/CakeCrusher/DSA_P3
Video demo: https://www.youtube.com/watch?v=cSww_7tjOAg&feature=youtu.be

COVID-19 Case Analysis and Visualization Project

Problem Statement

Analyzing the countries with the highest COVID-19 cases at any given time presents a significant challenge. The task involves processing vast amounts of daily data to identify trends and hotspots, which is crucial for informed decision-making.

Motivation

Understanding the most affected countries can drive critical policy decisions, such as implementing travel bans or allocating resources effectively. This project aims to provide a tool for proactive pandemic management.

Data Description

The project uses a JSON file containing daily COVID-19 case data for countries worldwide. Key features include: - Date (year, month, day) - Country - Number of cases The data's daily granularity presented challenges when aiming for monthly trend analysis.

Tools and Technologies

- Python: For data processing and analysis
- React: Frontend development
- react-spring: Smooth transitions in data visualization
- Axios: Data fetching

Implemented Algorithms

1. Bubble Sort: A simple, intuitive algorithm that repeatedly steps through the list, comparing and swapping adjacent elements.
2. Merge Sort: An efficient, divide-and-conquer algorithm that splits the list, sorts smaller lists, and merges them back.

Data Structures

- Python lists for initial data processing

- Hashmaps (Python dictionaries) for tracking peak cases per country and efficient data retrieval in React

Project Evolution

The project underwent significant changes when both teammates dropped the class. This necessitated a complete overhaul, the project was simplified and used simpler algorithms.

Complexity Analysis

Grouping:

Grouping each entry by year-month - Time Complexity: O(n), Space Complexity: O(n)

Sorting:

sorting both the year-month groups and the items in the groups. - Bubble Sort: Time O(n^2), Space O(1) - Merge Sort: Time O(n log n), Space O(n)

Overall:

m represents the groups and n the items - Bubble Sort: O(m * n^2) time - Merge Sort: O(m * n log n) time

Performance logs showed Merge Sort (23.82s) significantly outperforming Bubble Sort (139.58s) in sorting time.

Final logs for performance:

```
2024-08-05 20:53:35,442 - INFO - Bubble sort metrics:
2024-08-05 20:53:35,442 - INFO -   grouping_time: 0.09735894203186035
2024-08-05 20:53:35,443 - INFO -   sorting_time: 139.5772614479065
2024-08-05 20:53:35,443 - INFO -   month_sorting_time: 0.0
2024-08-05 20:53:35,444 - INFO -   total_time: 139.67462038993835
2024-08-05 20:53:35,444 - INFO -   memory_usage: {'input_size': 10906970, 'output_size': 109
2024-08-05 20:53:35,444 - INFO - Merge sort metrics:
2024-08-05 20:53:35,444 - INFO -   grouping_time: 0.11149454116821289
2024-08-05 20:53:35,444 - INFO -   sorting_time: 23.81571936607361
2024-08-05 20:53:35,444 - INFO -   month_sorting_time: 0.0073795318603515625
2024-08-05 20:53:35,444 - INFO -   total_time: 23.92721390724182
2024-08-05 20:53:35,444 - INFO -   memory_usage: {'input_size': 10906970, 'output_size': 109
2024-08-05 20:53:35,444 - INFO - Data processing complete
```

Reflections

Detailed logging proved invaluable for performance analysis and debugging. The development process was relatively smooth, with only minor adjustments needed

for the merge sort implementation.

A major oversight in the initial data analysis led to a mismatch between the preprocessing design and the actual data structure (daily vs. monthly data). This was discovered late in the visualization phase, necessitating additional grouping and sorting in the frontend.

Despite challenges, the project resulted in a functional system for tracking and visualizing COVID-19 hotspots over time. While not perfect, it serves as a solid foundation for potential use by policymakers and health officials.

References

- Merge Sort: https://www.geeksforgeeks.org/python-program-for-merge-sort/
- react-spring: https://react-spring.dev/docs/components/use-transition
- python logging: https://docs.python.org/3/library/logging.html