Module 3 - Critical Thinking

Miles Exner

Colorado State University Global

Course Code: CS506-2

Instructor: Jonathan Vanover

Due Date: 06/02/2024

Module 3 - Critical Thinking

This assignment demonstrates the sorting of hospital patient records using two sorting algorithms: Bubble Sort and Merge Sort. By analyzing their time complexities and performance under various conditions, we aim to determine the most suitable algorithm for sorting patient records, considering the volume and nature of the data. Additionally, the importance of stable sorting algorithms in maintaining the integrity of patient records will be highlighted. The data being sorted is an array of random dictionaries containing a few properties (name, age, and height). Stable sorting ensures that the data can be sorted by one key and then another while preserving the order. In this demonstration, both keys (age and height) will be used to show sorting stability. The sorting algorithms are timed, and the visualization speed can be adjusted. For simplicity, we use 60 cycles per second, which allows us to observe the dataset and compare the sorting times, even though the actual sorting process is much faster.

# Definitions

## Bubble Sort

Bubble Sort is a basic sorting algorithm that operates by repeatedly swapping neighboring elements if they are in the incorrect order. This algorithm is inefficient for large datasets due to its high average and worst-case time complexity. Specifically, its best-case time complexity is O(n) when the array is already sorted, and its worst-case time complexity is O(n^2).The advantage is that it does not create additional arrays (GeeksforGeeks, 2024).

## Merge Sort

Merge sort is known for its efficiency as a sorting algorithm. It operates on the principle of Divide and Conquer, which involves breaking a list down into several sub-lists until each sublist contains a single element, and then merging these sublists to form a sorted list. The process can be outlined in three main steps: dividing the problem into multiple subproblems, solving these subproblems by breaking them into atomic subproblems, and combining the solutions of the subproblems to solve the original problem. Specifically, merge sort works by dividing an unsorted array into subarrays, each containing a single element. These single-element subarrays are then merged into arrays of two elements, and this merging process is repeated until a single sorted array is obtained. Initially, an array of size 'N' is divided into two parts of size 'N/2'. This division continues until single elements are reached. Upon reaching this base case, the left and right parts are merged to form a sorted array (Pankaj, 2022). Merge Sort has a consistent time complexity of O(n log ⁡n) in all cases, making it suitable for larger datasets. It is also a stable sorting algorithm, which means it maintains the relative order of equal elements.

# Analysis

## Time Complexity and Optimal Conditions

The time complexity of Bubble Sort is O(n2)O(n^2)O(n2) in both average and worst-case scenarios, making it unsuitable for large datasets. It performs optimally only when the dataset is small and nearly sorted, with a best-case time complexity of O(n)O(n)O(n). In the context of patient records, Bubble Sort's inefficiency becomes evident as the number of records increases, leading to slower sorting times and potential delays in accessing critical patient information.

On the other hand, Merge Sort consistently performs with a time complexity of O(nlog⁡n)O(n \log n)O(nlogn), regardless of the dataset's initial order. This makes it highly efficient for sorting large volumes of patient records. Merge Sort's stability ensures that records with identical keys (e.g., patients with the same age) remain in their original relative order, which is crucial for maintaining the integrity of patient data.

## Critical Problem of Sorting in Healthcare

Efficient sorting algorithms are vital in healthcare for managing and accessing patient records swiftly. Delays in retrieving patient information can lead to suboptimal medical decisions and compromised patient care. Therefore, choosing the right sorting algorithm directly impacts the hospital's operational efficiency and the quality of care provided to patients.

## Justification of Data Structures

For this analysis, arrays of patient records were used. Each record is a dictionary containing fields such as name, age, and height. This data structure is well-suited for both Bubble Sort and Merge Sort, allowing for easy access and manipulation of individual records. The choice of dictionaries ensures that the sorting algorithms can compare multiple fields, facilitating stable sorting.

## External Factors Affecting Efficiency

External factors such as the hardware specifications of the hospital's computer systems, the size of the dataset, and the frequency of sorting operations can influence the efficiency of the sorting algorithms. In a real-world healthcare setting, where large datasets are common and timely access to records is vital, Merge Sort's superior performance and stability make it the preferred choice.

# Findings

The performance of Bubble Sort and Merge Sort was tested on a dataset of 20 patient records. The results are as follows:

* **Bubble Sort**: 3.2609345999953803 seconds
* **Merge Sort**: 1.1130728999560233 seconds

# Conclusion

In conclusion, Merge Sort significantly outperforms Bubble Sort in terms of efficiency, making it the better choice for sorting patient records in a hospital setting. Its consistent O(n log n) time complexity ensures that it can handle large datasets effectively, while its stability preserves the relative order of records with identical keys. In contrast, Bubble Sort's inefficiency and high time complexity render it unsuitable for large datasets. The importance of choosing a stable sorting algorithm cannot be overstated, as it ensures the integrity and reliability of patient records, ultimately contributing to better patient care and operational efficiency in healthcare facilities.

# References

GeeksforGeeks. (2024, April 12). *Bubble Sort*.<https://www.geeksforgeeks.org/bubble-sort/>

Pankaj. (2022, August 3). *Merge Sort Algorithm - Java, C, and Python Implementation*. Data Structure and Algorithms. DigitalOcean.<https://www.digitalocean.com/community/tutorials/merge-sort-algorithm-java-c-python>