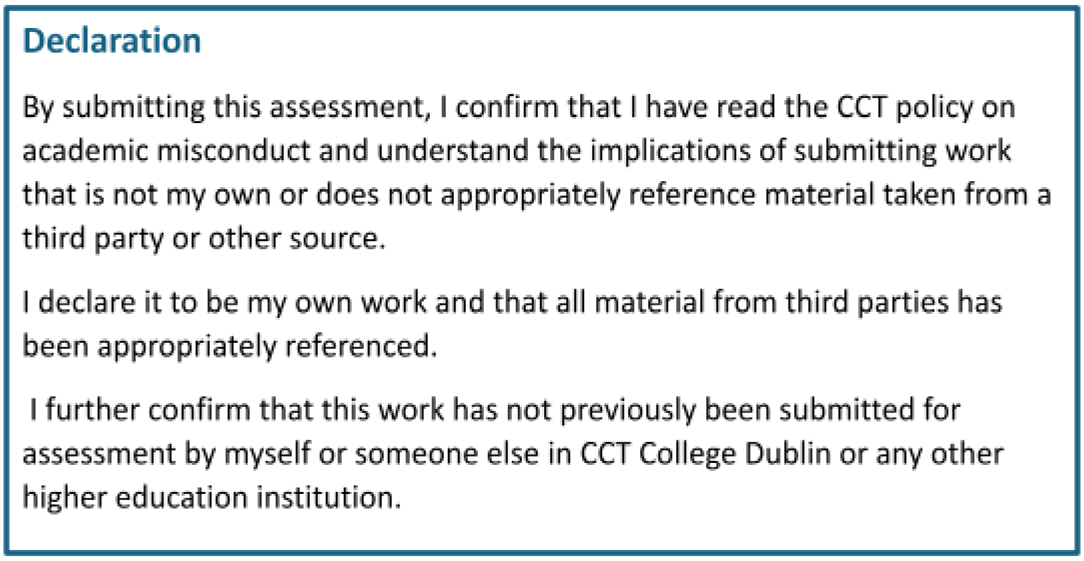
Assessment Cover Page

Student Full Name	Luiz Philipe de Albuquerque Moraes Ribeiro de Sena
Student Number	2024025
Module Title	Algorithms and Constructs
Assessment Title	System Modelling & Build - Search and Sort Algorithms in a Hospital Management System
Assessment Due Date	16th November 2024 - 23:59h
Date of Submission	16th November 2024

Use of AI Tools

I have not used any AI tools or technologies in the preparation of this assessment.



Declaration

By submitting this assessment, I confirm that I have read the CCT policy on academic misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source.

I declare it to be my own work and that all material from third parties has been appropriately referenced.

I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

Contents

Introduction

This Java project was developed with the intention of improving and organizing access to employee and patient data at Dublin Hospital. Over time, the volume of patient and employee data gradually increased, which required greater efficiency in inserting new data and searching for data already entered. With this increase in the volume of the database, it was necessary to develop a program so that the operations for searching, including and sorting data would be more assertive. For this to be actually carried out, defining the type of algorithm to be used is essential, because depending on the type chosen, the performance and speed of processing can be crucial for better system performance and, consequently, greater agility saving time for users.

Given the large number of algorithms currently available that perform these functions, I decided to implement the Merge Sort algorithm to perform the sorting and the Binary Search algorithm to search the data. In this report, I will explain the reason for this choice and make some comparisons with other available options and I will inform why these two were the most suitable for this type of project. Assuming that a hospital needs to make decisions with high agility, a program that is executed quickly can improve patient care and literally save lives, this is the importance of having an efficient and agile algorithm.

Merge Sort and Binary Search Algorithms in a Hospital Management System

To develop this hospital system, I chose to use the Merge Sort and Binary Search algorithms, used to sort data and search data, respectively. The reason for this choice is mainly based on the efficient execution of these algorithms, due to their high performance when we take into account the time complexity.

Merge Sort - Performing Data Sorting

The Merge algorithm is a sorting algorithm that has as its motto in its executions the idea of “divide and conquer”. This algorithm works by performing sequential divisions in each iteration of its loop until it reaches the last position, that is, when the Array is completely divided, leaving only unique elements. After these divisions, the algorithm then begins its purpose, which is to perform the sorting, separating the smallest value for the left position and its largest value for the right of the Array, as we can see in the following image:

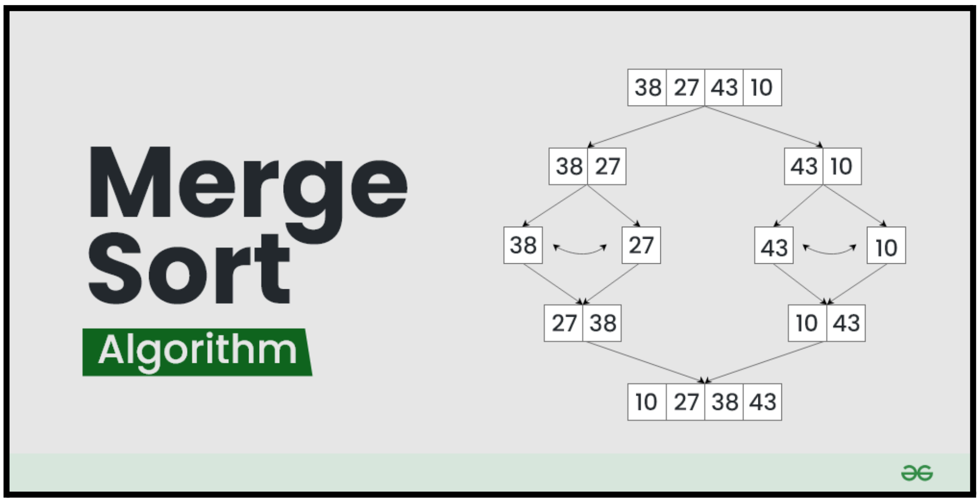


Fig. 01 - Merge Algorithm diagram (Geeks, 2024)

This algorithm performs efficiently even with a large volume of data ***and this was the main reason for choosing it,*** because as the years go by, the volumes of data entered into a hospital system's database will only tend to increase, with daily care for new patients and new employees being hired in a hypothetical situation of growth of the hospital structure. Or if we see some critical times like the COVID case, where hospital systems practically collapsed due to the number of patients being treated at all times, we have to think long-term and look for something that provides safety and efficiency. Applied to the hospital system, Merge Sort will work stably and due to the recursive model used, which will further optimize its efficiency by reducing the number of iterations in its application, and because it has a time complexity of O(n log n), it will provide a long-term solution, avoiding any type of overload with the increase in the amount of data.

Merge Sort works by dividing the main list into smaller sub-parts, until there is only one element. This process repeats recursively until the condition where “left side < right side” is true. At the moment when it can no longer be divided, this condition will then be false and the “left side will be equal to the right side" and from this moment the algorithm used in the program imports the method of the Sorter class called "mergeSortSubList" and begins the process of combining the sublists. At the moment of combination, the elements that were divided begin the process of coming together again in a list, but this time ordered.

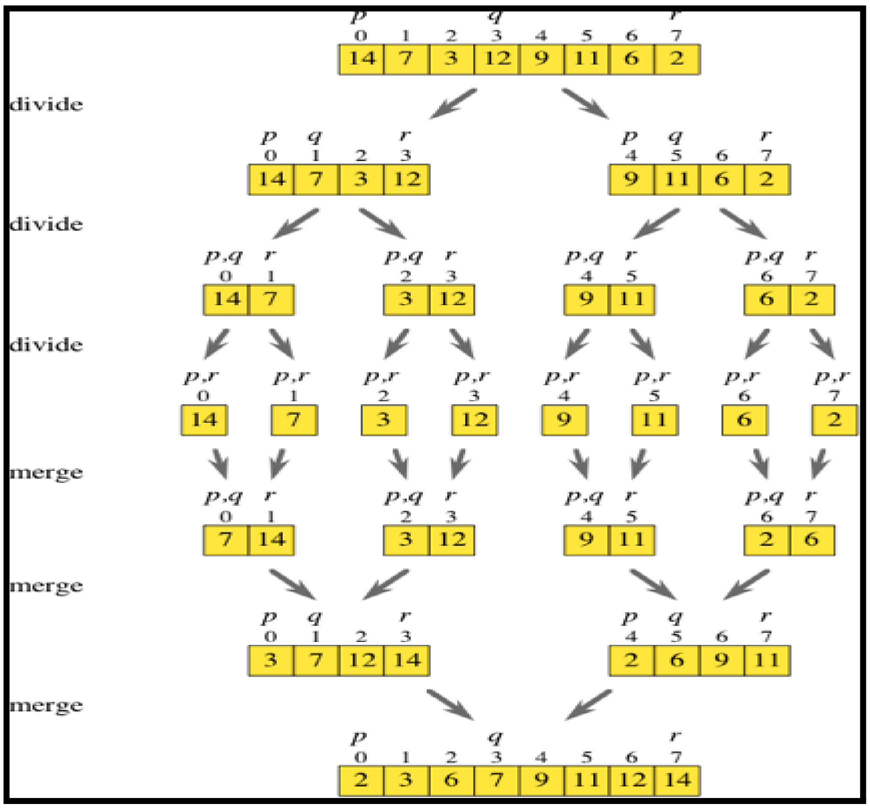
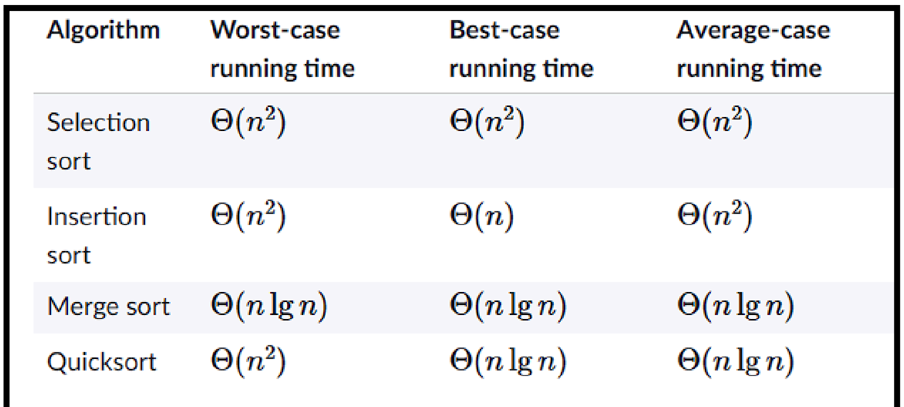


Fig. 02 - Khan Academy. (n.d.). Merge sort algorithm overview (article). [online] (2024)

Explaining The Merge Sort Time Complexity and its Advantages:

As previously mentioned, Merge Sort has a time complexity rating of O(n log n) and this basically means how long it will take for this algorithm to perform its sorting function on a list. This formula takes into account two "n's" O(n Log n) factors, the first 'n' being the number of elements contained in the list to be sorted. The second 'n' (Log n) is used to measure the number of divisions that were performed on the list into sub-lists. In all possible cases, Merge Sort will have the same time complexity, regardless of whether the list is already partially sorted or sorted in reverse order.

The algorithm will then behave exactly the same way, performing its divisions up to the last element and then merging the elements to form a new list. Regardless of the condition of the list, the performance will be the same. The index that can be changed is only the size of the list and consequently the number of divisions to be performed. The following image shows a comparative table of performance of algorithms that execute the Sort function.



Algorithm	Worst-case running time	Best-case running time	Average-case running time
Selection sort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$
Insertion sort	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n^2)$
Merge sort	$\Theta(n \lg n)$	$\Theta(n \lg n)$	$\Theta(n \lg n)$
Quicksort	$\Theta(n^2)$	$\Theta(n \lg n)$	$\Theta(n \lg n)$

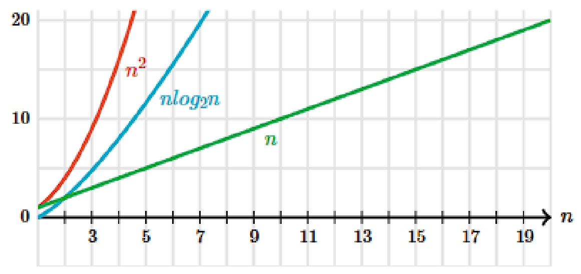


Fig. 03 - Sort Algorithms Performances (Divide and conquer algorithms (2018). Divide and conquer algorithms. [online] Khan Academy).

Therefore, the reasons for choosing these algorithms are mainly based on:

1) ***Stability***: as they have the same performance in all situations, including the worst case scenario where the list is sorted in reverse order.
2) ***Satisfactory performance:*** even when dealing with a list with a large volume of data.
3) ***Simplicity :*** when implemented in the program

GeeksforGeeks (2018). *Merge Sort - GeeksforGeeks*. [online] GeeksforGeeks.

Comparison between Merge Sort and Other Algorithms for Application in Large Volumes of Data:

Merge vs. Quick Sort: Although Quick Sort has a better performance in terms of processing speed, it is not considered a stable algorithm and in its worst case it will become quite inefficient. In the case of Inserted, a hospital system, the reliability of the algorithm must be taken into consideration, therefore… 1x0 Merge Sort.

Merge vs. Heap Sort: Despite having excellent performance and having the same time complexity O(n log n) as Merge, Heap Sort proves to be unstable when applied to a list of large volumes, returning to the idea of a hospital and dealing with lives, Merge Sort comes out ahead again due to its stability and reliability in ordering the list, therefore… 2x0 Merge Sort.

The other algorithms found perform better in a small volume quantity, which will not apply to the hospital system, therefore, compared to algorithms that work with large data volumes, Merge Sort performs better, in my opinion, because it has better reliability and stability than the others.

Chetan Lohkare (2021). *A Comparative Study on Sorting Algorithms - Chetan Lohkare - Medium*. [online] Medium. Available at:

Binary Search - Performing Data Searching

In the hospital management system, the Binary Search algorithm was used in a recursive model to efficiently search for patients and employees, using the full name provided by the user. Binary Search works on the principle of "divide and conquer", in which the list is repeatedly divided in half until the desired element is found. In the program, it is applied recursively, which simplifies the code structure by keeping the division process within successive function calls. Importantly, Binary Search requires a pre-sorted list, which is achieved through the Merge Sort algorithm. When the user sorts the list first using Merge Sort and then searches through the option menu to search a name, this action enhances the Binary Search performance.

Binary Search Efficiency: Binary Search has complexity O(1) in the best case when the key is exactly at the middle, O(log n) and O(log n), which means that the execution time increases logarithmically as the list size increases. In other words, for each new division of the list, the number of remaining elements to be checked is reduced by half.
ravikishor (2019). *Complexity Analysis of Binary Search*. [online] GeeksforGeeks.

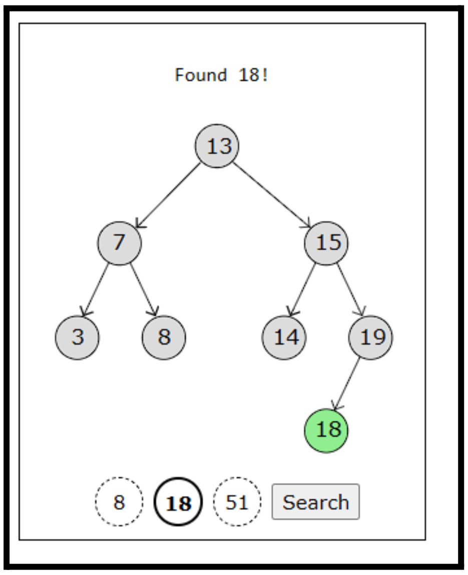


Fig. 04 - Binary Search Tree Finding the number 18 in the List. www.w3schools.com. (2024.). *DSA Binary Search Trees*. [online]

Compared to the linear search algorithm, where each element in the list would be checked until the desired one was found, which would be much less efficient for large lists of data such as in a hospital system, the Binary algorithm performs better.

Recursive Operation in the Hospital System: In the hospital context, binary search is implemented recursively, with two main functions:

- **binarySearchByName:** This function divides the list into two parts with each recursive call. A midpoint will then be defined by comparing the central value with the name provided and decides whether the search should continue in the left or right half of the list. The function repeats until the element is found or all divisions are verified, returning -1 if the person is not found and a message will be displayed to the user to indicate that the person to be searched is not on the list.

- **searchByName:** This function receives the patient's name as input from the user, processes the string (removing spaces and converting to lowercase letters), and then calls the binarySearchByName function to locate the patient in the list.

GeeksforGeeks (2014). Binary Search Algorithm Iterative and Recursive Implementation. [online] GeeksforGeeks

Comparison with Linear Search: In systems that work with highly important data, such as hospitals, where accuracy and speed of search are crucial, Binary Search offers a significant advantage over a linear search. A linear search, with complexity O(n), O(n), O(n), would require more checks as the list grows, while a binary search remains efficient. This factor makes binary search more scalable and practical for the system, especially considering a potential increase in data.

Therefore, the choice for Binary Search was due to the great need to perform fast and efficient searches. This algorithm guarantees this efficiency even with extensive lists, with large volumes of data. In an environment where data volumes have grown over time, the use of Binary becomes a viable option, since its performance will not have been affected as much as it would be in a linear search algorithm, for example.

Conclusion

In summary, this application of the Merge and Binary algorithms in the hospital management system brought great gains in performance, reliability and efficiency when searching for a specific person or ordering lists.

Both chosen algorithms have efficient scalability when it comes to increasing data volumes, which will occur in a hospital environment, where new data is entered daily, due to daily patient care and new employees being added to the hospital staff.

Having a long-term view, it can be said that the two chosen algorithms will perform well in the function they were assigned, processing data quickly and accurately even with a possible significant increase in data in the future.

And this speed in performance directly reflects on patient care, since having immediate information about a specific patient's care saves time and helps to assist a patient who needs emergency or extremely urgent care more quickly. This directly connects to the critical decision-making that is carried out by hospital staff, such as doctors, nurses and people responsible for providing a service to the patient directly.

References

Bharat (2011). Calculating mid in binary search. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/6735259/calculating-mid-in-binary-search. [Accessed 7 Nov. 2024].

Chetan Lohkare (2021). A Comparative Study on Sorting Algorithms - Chetan Lohkare - Medium. [online] Medium. Available at: https://chetan-187.medium.com/a-comparative-study-on-sorting-algorithms-208bdef85ebc [Accessed 6 Nov. 2024].

Divide and conquer algorithms (2018). Divide and conquer algorithms. [online] Khan Academy. Available at: https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms. [Accessed 6 Nov. 2024].

GeeksforGeeks (2014). Binary Search Algorithm Iterative and Recursive Implementation. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/binary-search [Accessed 7 Nov. 2024].

GeeksforGeeks (2018). Merge Sort - GeeksforGeeks. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/merge-sort/. [Accessed 6 Nov. 2024].

Khan Academy. (n.d.). Merge sort algorithm overview (article). [online] Available at: https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/overview-of-merge-sort. [Accessed 6 Nov. 2024].

ravikishor (2019). Complexity Analysis of Binary Search. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/complexity-analysis-of-binary-search/. [Accessed 7 Nov. 2024].

www.w3schools.com. (2024). DSA Binary Search Trees. [online] Available at: https://www.w3schools.com/dsa/dsa_data_binarysearchtrees.php. [Accessed 7 Nov. 2024].