### **CCT College Dublin**

**Assessment Cover Page**

| **Module Title:** | Algorithms and Constructs |
| --- | --- |
| **Assessment Title:** | System Modelling & Build |
| **Lecturer Name:** | Dr. Muhammad Iqbal |
| **Student Full Name:** | Mariana Casarin Battistin |
| **Student Number:** | 2024276 |
| **Assessment Due Date:** | 10/05/2025 |
| **Date of Submission:** | 10/05/2025 |

**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. |
| --- |

## 

## **Version Control**

This project is available in the GitHub repository: <https://github.com/mbattistin/CA_2>. It is public to all viewers. This documentation is also in the link for version control.

## **Sorting: Insertion Sort vs Bubble Sort**

The hospital system needs to sort the applicants list when requested by the user. I choose insertion sort instead of bubble sort because it performs better, specially in small lists. In practice, insertion sort needs fewer operations. The first sort is going to be less efficient, but once the list is sorted for the first time, insertion sort goes faster and does less swaps than bubble sort. For instance, one study found bubble sort to be twice times slower than insertion sort on typical data. It is an ideal algorithm for those who are still learning about the subject, like me, because it is not too complex and still is better than Bubble sort (Agha and Nawaz, 2021) .

## **Searching: Linear vs Binary Search**

The system also needs to search for the applicants through the list. One key point for the implementation is if the list is ordered or not. As the user can decide when he wants to sort the list, I chose a different approach in each situation to try to increase performance. If a list is unsorted, I implement the linear search, checking the records one by one. Even if it is the slower option, it is the available one when the list is not ordered. If the user already sorted the list, I used binary search, which repeatedly divides the search by two. For instance, searching 1000 records takes about 1000 comparisons with linear search but only around 10 with binary search (P.Parmar and Kumbharana, 2015). I understand that the binary search is the best option for the search, but I decided not to interfere in the user's decision of sort or not, due to the fact that if I start the sort before the search, it would consume time as well and may be noticed by the user.

## **References**

P.Parmar, V. and Kumbharana, C. (2015). Comparing Linear Search and Binary Search Algorithms to Search an Element from a Linear List Implemented through Static Array, Dynamic Array and Linked List. International Journal of Computer Applications, 121(3), pp.13–17. doi:https://doi.org/10.5120/21519-4495.

Agha, Fanila and Nawaz, Haque. (2021). International Journal of Advanced Trends in Computer Science and Engineering, 10(2), pp.1020–1025. doi:https://doi.org/10.30534/ijatcse/2021/761022021.

‌