**CCT College Dublin**

**Assessment Cover Page**

|  |  |
| --- | --- |
| **Module Title:** | HDIP in Science in Computing |
| **Assessment Title:** | Algorithms and Constructs CA-1 |
| **Lecturer Name:** | Dr Muhammad Iqbal |
| **Student Full Name:** | Anudari Bat-ochir. |
| **Student Number:** | 2024636 |
| **Assessment Due Date:** | 07th April 2025 |
| **Date of Submission:** | 10th May 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. |

Table of Contents

[Why I Picked Merge Sort for Sorting 3](#_Toc197676486)

[Why I Picked Binary Search for Searching 3](#_Toc197676487)

[GITHUB LINK 4](#_Toc197676488)

# **Why I Picked Merge Sort for Sorting**

Honestly, when I started this project, I just needed a way to sort and search through a list of hospital employees, and I didn’t want to overcomplicate things. I picked Merge Sort for sorting because it’s reliable. It always works pretty fast, even if the list is big or messy. I didn’t want to take the risk with something like Quick Sort, because even though it can be fast, it can also slow down if the data’s already kind of sorted, and I didn’t feel like dealing with that.

Plus, the assignment said we had to use something recursive, and Merge Sort is naturally recursive, so it just made sense to me. Also, it’s stable, which basically means if two people had the same name (even though that didn’t really happen much), it would keep them in the right order, and that’s kind of important when you’re dealing with hospital records.

At first, I thought about just using something simple like Bubble Sort or even Insertion Sort because they’re easy, but honestly, they’re way too slow for bigger lists. I wanted the system to feel fast and smooth, not like it was struggling if there were a lot of employees. Merge Sort was a bit more work to set up, but once it was done, it worked perfectly and handled everything really well. I thought about Quick Sort for a while because it can be faster sometimes, but the problem with Quick Sort is that if the data is already nearly sorted, it can actually get slower.  
I didn’t want to risk it and add extra work trying to make it better, so I just stuck with Merge Sort, which is more consistent.

# **Why I Picked Binary Search for Searching**

For searching, I first thought about using a basic linear search — just checking each name one by one — because it's easy to code and I knew it would work no matter what. But when I realized my list was already sorted because of Merge Sort, it felt stupid not to use Binary Search. Binary Search is way faster because instead of checking every name, it just keeps splitting the list in half until it finds the right one. It's like looking for a word in a dictionary instead of reading every single page. I mean, who has time to scroll through a thousand names one by one? Binary Search made everything a lot quicker and made the system feel way more professional.

Another reason I liked Binary Search is because it kind of forces you to keep your data organized (sorted), which is just a good habit anyway when you’re building systems like this.  
  
At first, I thought about just using a simple Linear Search because it’s easy to write, but after thinking about it, I realized it would be too slow if the employee list got bigger.

Looking back, I’m really glad I didn’t take the lazy route and just throw in some basic sorting and searching. Merge Sort and Binary Search made the system faster, cleaner, and more reliable. It wasn’t just about ticking the boxes for the assignment — it actually made a difference when using the program. The system feels solid now, and if it were a real hospital system with loads of employees, it could still handle everything without a problem. So yeah, I’m really happy with the choices I made.

# GITHUB LINK

<https://github.com/Anudari906/CA2_HospitalManagement.git>