cct | College Dublin
Computing • IT • Business

Assessment Cover Page

Student Full Name	Tungalagtamir Begzsuren
Student Number	2025168
Module Title	Algorithms & Constructs
Assessment Title	System Modelling & Build
Assessment Due Date	Saturday 29th November 2025
Date of Submission	Saturday 29th November 2025

I agree and confirm

Contents

Explanation of Algorithm Choices

Why I Chose Merge Sort

When choosing between sorting algorithms for the employee list, I considered what would suit our bank's organisational system. We needed something fast and reliable, especially since the number of employees will keep growing. I selected **Merge Sort** because it works like organising a large stack of papers: you split them into smaller piles, sort those piles, and then merge them back together carefully.

It guarantees **O(n log n)** performance in all cases—worst, average, and best—making it predictable for business use. Some other methods slow down with certain name arrangements, but Merge Sort stays consistently efficient. Another benefit is that it preserves the original order of employees with identical names, which is useful for maintaining stability in the records.

The assignment also required us to provide a recursive algorithm, and Merge Sort naturally fits this requirement by breaking the problem into smaller parts and solving them recursively. For all these reasons, it was an ideal choice.

Why Binary Search Works Best in Finding Employees

For searching through employee data, the clear winner was **Binary Search**. Since we maintain the list in sorted order—thanks to merge sort—binary search can take full advantage of that structure. It can locate any employee in only a few steps, even when the system holds thousands of names. Using a simple linear search would become slow and inefficient as the bank grows.

Binary search is predictable, offering **O(log n)** time complexity. This means that searching through 1,000 employees requires about 10 comparisons instead of 1,000, giving fast results each time a lookup is done. The code sorts the list using Merge Sort first and then performs Binary Search on the sorted data, ensuring consistent efficiency.

Building the Employee Family Tree

I implemented a **Binary Tree** to represent how people in the bank are connected. In the structure I created, the tree grows in a balanced way, filling positions from left to right at each level as employees are added. This makes it easy to see who reports to whom and to understand the bank's hierarchy. The tree shows different levels similar to an actual organisational chart, and I included features to count how many employees there are and how "tall" the structure is.

How These Choices Work Together

All these components work together like an organised office. Merge Sort keeps the employee list ordered, Binary Search helps us find people quickly, the binary tree shows how everyone fits in the organisational structure, and the validation ensures that our information remains accurate. This setup works well now and will scale effectively as the bank continues to grow.

GitHub URL

https://github.com/2025168/algorithm-and-construct-ca-2