For lab 2 we where task to find the time between several sorting algorithms, bubble sort, merge sort, and quick sort. These algorithms are easy to explain but alot more difficult to code.

Easy yet tidious the assignment started off with bubble sort and coding it in Idecided to use a similar if not the same code as the professor showed us, wich was rather tidous, as Itried and run it my computer seemed to not be able to run all of it, but Inever got the message saying it was going for ever, so Iproceeded with what Ibelived was right since Icouldnt find a problem with the code, second was merge sort, and well then Ileanerd to hate merge sort, looked up how to do it, how to its supposed to work and well Igot a version going in which Ihad three def one for merge sort that would call on the next wich Icalled split, that one would call it self till there was one left then call merge to merge them back together. Lastly quick sort, had to get a pivot have three new lists pivot it self would be a list then greater or less would have their own, it would call it self over untill there was 1 element in each list then it would merge in order. Last one was the same but only doing it once.

My Experimental data went from a random generating numbers from 1 to a hundred to me telling the machine to appened five numbers , because of my trials with merge sort , Ideleted all my code in frustation and wasn’t able to get it back.

Seeing how unefficient the easiest sorting algorithm to code is was rather disappointing due to the fact that Iknow Iwill be using the others more due to their efficently, what Ido need to do Iperfect these codes and keep them safe so that later on Ican just copy and past my code into what I need.