CIS*3190 Software for Legacy Systems

Assignment 4 - Fun with Languages

Dobosort - Improved Bubblesort

Maneesh Wijewardhana (1125828)

- Running times of various sorting algorithms in **Fortran** (times in ms)

n	Quicksort	Bubblesort	Dobosort
1000	0.32	6.38	0.37
10000	2.39	264.22	4.32
50000	13.1	10297.33	19.83

- Running times of various sorting algorithms in **Ada** (times in ms)

n	Quicksort	Bubblesort	Dobosort
1000	0.22	5.96	0.4
10000	2.74	364.67	3.35
50000	9.82	9680.34	14.33

Note: These times were measured on a M1 CPU, when testing on the school server, times were significantly slower

- Implementing the algorithms in Fortran and Ada resulted in the same experience overall. They were both verbose languages and needed a lot of descriptions for each part of the program in order for it to function properly. Apart from the program structures being quite different, the underlying sorting algorithms were implemented in the same fashion. Also, all 3 sorting algorithms performed similarly whether it was implemented in Fortran or Ada, but it seemed to me that Ada performed slightly better in all scenarios. This could be the case due to many things, but I suspect it is just a matter of how my machine handles Fortran and Ada programs respectively.
- The improved Bubblesort also known as Dobosort resulted in a significant speed increase performing almost the same as Quicksort. The regular Bubblesort took more than 10 seconds to sort 50000 integers where as Dobosort did it in 14ms. Although it was not faster than Quicksort in any case, it was just as close for less code and less complexity in terms of the algorithm itself. Some limitations of Dobosort is that in the worst case, the algorithm is still $O(n)^2$. When it comes to Quicksort, one limitation would be if the partition process always picks the greatest or smallest element as the pivot which would result in a runtime of $O(n)^2$ but this can easily be mitigated and changed based on the data itself. Bubblesort has many limitations, the most prevalent one being comparing all adjacent values which would result in a very slow process for large amounts of data.
- In my opinion, if I just needed to quickly sort large amounts of data and wanted to do it relatively efficiently, I would resort to using Dobosort just because of its simplistic nature compared to Quicksort and various other sorting algorithms like Mergesort. Based on the data, it seems like the best approach without compromising in a lot of other areas. Despite all of this, in most cases I would simply use the built-in sort method provided in most languages.