Ethan Le

CS 2341

Prof. Fontenot

6th of October, 2021

Lab Report of Project Assignment 03: “Mystery Sort”

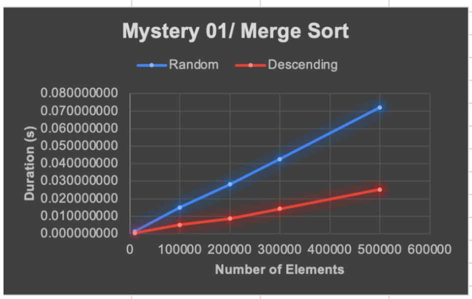
As an introduction, my name is Ethan Le, and I am a student at SMU studying for a bachelor's degree in computer science. My goal is to delve into the field of software engineering, because I want to explore the design and implementation of computer algorithms. This semester I took a course on Data Structure and came across project 3 “Mystery Sort”. The project assignment requires to determine the type of sorting in each mystery function that has been implemented. I would first separate the sorting function into two groups based on average run time then further study them when running in their best and worst case scenario.

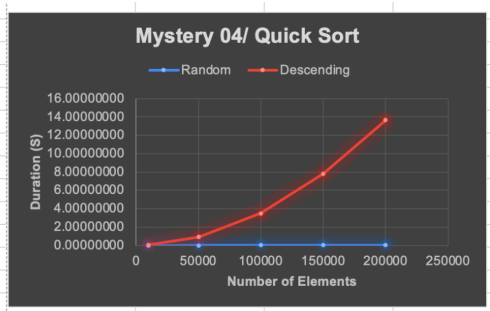
I started the distinguishing process by separating the mystery function into two groups using run time of O(nlogn) and O(n^2) in the average case. Merge sort and quicksort would be O(n Log n) as they are efficient sorting algorithms while selection sort, bubble sort, and insertion sort would be in place of O(n^2). I would first create a randomized array of integers with size ranging from 1000 to 500000 and then run those mystery functions on these arrays. To increase the accuracy of my test, I run the sorting function three times and take the average time in case there are variances in run time. Based on what I observe, the run time of mystery 1 and 4 are 0.028106s and 0.016059s, which are both less than 1 second in the case of 500000. Compared to the run time of the other mystery sorts, which are all several minutes long, I can easily identify that mystery sort 1 and 4 must be the group with run time of O(nlogn).

Mystery 01		
Sizes	Random	Descending
10000	0.001294315	0.000411183
100000	0.014985683	0.005031366
200000	0.028106890	0.008590262
300000	0.042574468	0.014167441
500000	0.072072206	0.025143856

Mystery 04		
Sizes	Random	Descending
10000	0.00056808	0.041067020
50000	0.00358493	0.899040658
100000	0.00795571	3.492967303
150000	0.01208633	7.822190262
200000	0.01605936	13.642814743

The next procedure was to determine the type of sorts between mystery 01 and mystery 04. In the worst case run time of quicksort, where the list is sorted in reverse order, the run time of the algorithm is very close to O(n^2). Therefore, I create an array of integers that is in reverse order and run these two mystery functions on this array. The Mystery 04 function is recognized as a quick sort because it has the slowest running time when sorting in this type of array. The data plot graph of Mystery 04 shows that it takes way more time to build and run through the number of elements than the Mystery 01. At a size of 200 000, mystery 04 had a significant amount of 13.64 seconds in descending order while the mystery 01 function took about 0.0086 seconds. Therefore, the mystery 01 function is identified as a merge sort Hence, mystery 01 represents the merge sort and mystery 04 represents the quick sort.

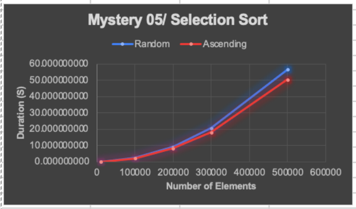




At this point, mystery 02, mystery 03, and mystery 05 had the slower run time. For the group O(n^2), there are two procedures to find the sorting algorithms. The first procedure is to determine the selection sort by running the number of elements in a sorted order. This is because selection sort is a consistent sorting algorithm, which means that even in the best case scenario of a sorted array, the run time is still O(n^2). Based on the observations of the graph, the ascending column of Mystery 05 shows a higher upward trend than the other two graphs. Consequently, mystery 05 function is the selection sort since it takes more time to run through the number of elements. Within almost sorted data, bubble sort and insertion sort need very few swaps. However, selection sort requires the same amount of search process even in almost sorted data.

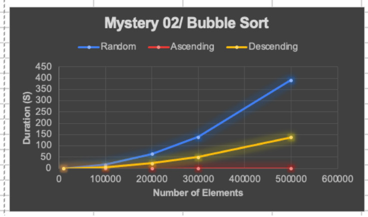
Mystery 05

Sizes	Random	Ascending
10000	0.027999301	0.027691584
100000	2.428134335	2.130611126
200000	9.179784043	8.271582271
300000	20.73234077	18.1733182
500000	56.52622908	50.434989686



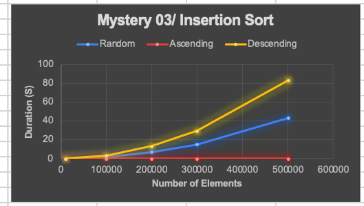
Mystery 02

Sizes	Random	Ascending	Descending
10000	0.12661107	0.000005636	0.063842871
100000	16.37790912	0.000064781	5.812633186
200000	63.517187965	0.000140571	22.47353246
300000	140.1344214	0.000183761	50.92603918
500000	391.106616348	0.000319813	137.9291341



Mystery 03

Sizes	Random	Ascending	Descending
10000	0.021541586	0.000012117	0.058664426
100000	1.79613073	0.000120955	3.512306902
200000	6.958663412	0.000353566	13.409657950
300000	15.0638765	0.000546211	29.633732395
500000	43.015400741	0.000658538	82.749315932

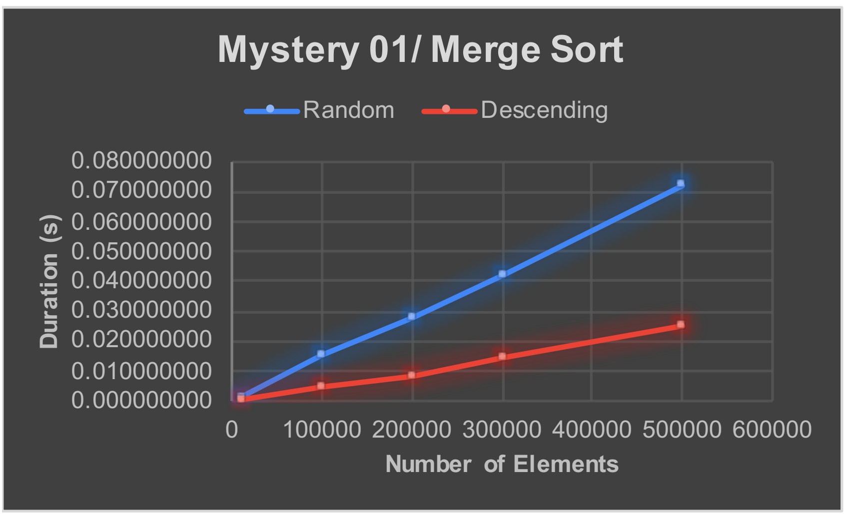


The second procedure is to determine the bubble sort and insertion sort by running the number of elements in reverse order. The chart of Mystery 02 shows that during the descent process, the running time is longer than that of Mystery 03. In addition, bubble sort performs more swap operations than the insertion sort. The higher number of swaps leads to a higher runtime for the bubble sort algorithm. Thus, the difference in runtime grows as the number of elements to be sorted increases on a random list. Therefore, the mystery 02 function is recognized as a bubble sort because it has the slower running time when sorted in the reverse order.

In conclusion, after, separate the sorting function into two groups based on average run time, then further study them when running in their best and worst case scenario. For O(nlogn), mystery 1 is identified as merge sort and mystery 4 is quick sort. Hence, for O(n^2), mystery 5 is identified as selection sort, mystery 2 is bubble sort and mystery 3 is insertion sort.

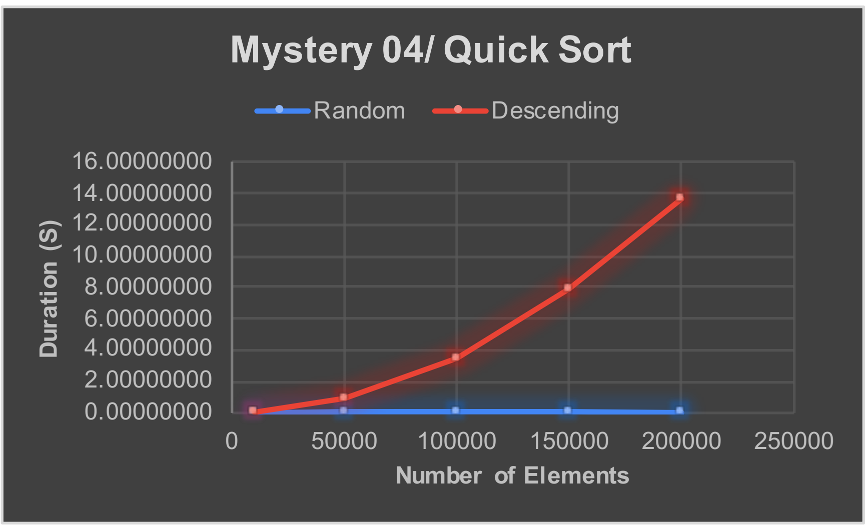
Mystery 01

Sizes	Random	Descending
10000	0.001294315	0.000411183
100000	0.014985683	0.005031366
200000	0.028106890	0.008590262
300000	0.042574468	0.014167441
500000	0.072072206	0.025143856



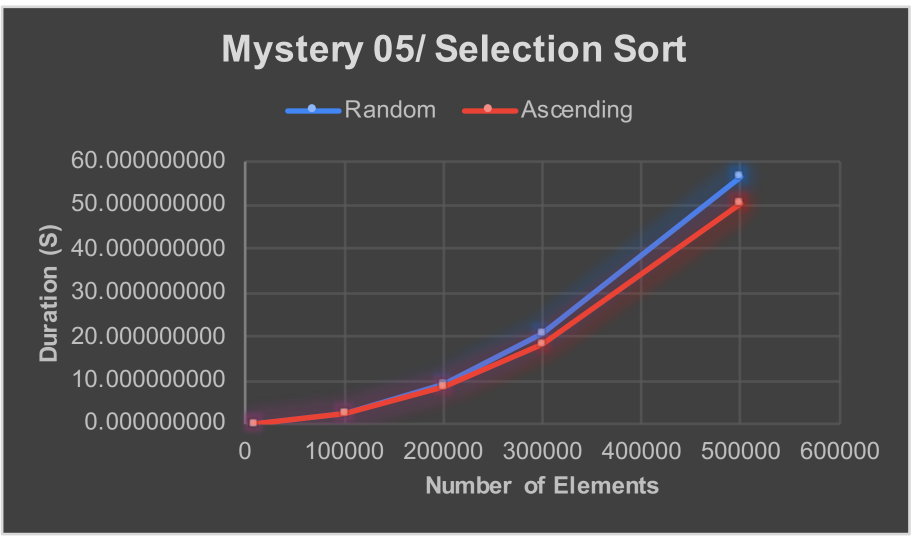
Mystery 04

Sizes	Random	Descending
10000	0.00056808	0.041067020
50000	0.00358493	0.899040658
100000	0.00795571	3.492967303
150000	0.01208633	7.822190262
200000	0.01605936	13.642814743



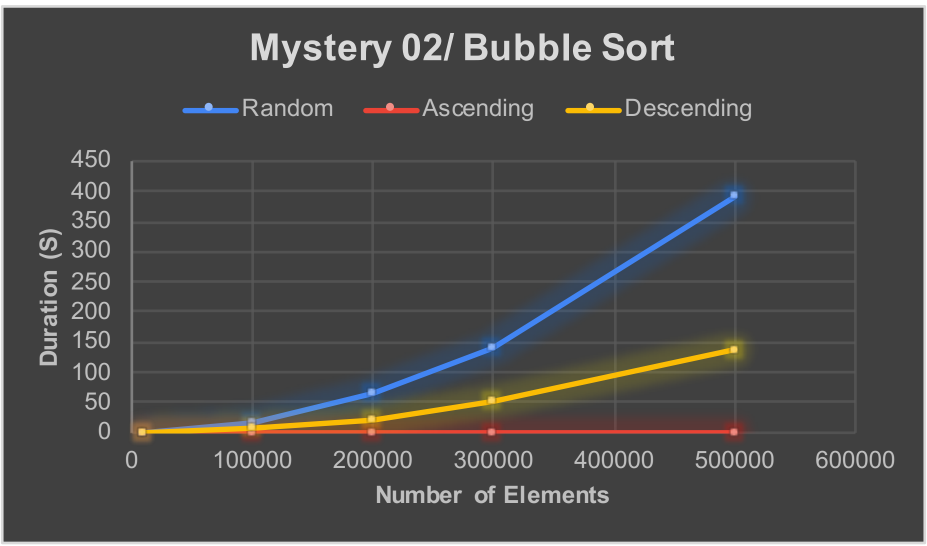
Mystery 05

Sizes	Random	Ascending
10000	0.027999301	0.027691584
100000	2.428134335	2.130611126
200000	9.179784043	8.271582271
300000	20.73234077	18.1733182
500000	56.52622908	50.434989686



Mystery 02

Sizes	Random	Ascending	Descending
10000	0.12661107	0.000005636	0.063842871
100000	16.37790912	0.000064781	5.812633186
200000	63.517187965	0.000140571	22.47353246
300000	140.1344214	0.000183761	50.92603918
500000	391.106616348	0.000319813	137.9291341



Mystery 03

Sizes	Random	Ascending	Descending
10000	0.021541586	0.000012117	0.058664426
100000	1.79613073	0.000120955	3.512306902
200000	6.958663412	0.000353566	13.409657950
300000	15.0638765	0.000546211	29.633732395
500000	43.015400741	0.000658538	82.749315932