**This project was inspired by a similar one by Julie Zelenski and Colleagues at Stanford University*

Name: Ethan Chou, freshman at SMU majoring in Computer Science

Problem:

I am tasked with determining which sorting algorithm is which out of Bubble Sort, Quick Sort, Merge Sort, Selection Sort, and Insertion Sort.

Strategy:

My strategy is to use the chrono library to time each sorting algorithm with a sorted array in ascending and descending order, and a random order, and plotting the data on a line graph. The idea is that by using the line graph, it should line up with a big O notation line graph such as O(n^2), O(n log n) and O(n).

What I figured out:

mysterysort01 = Merge Sort

mysterysort02 = Bubble Sort

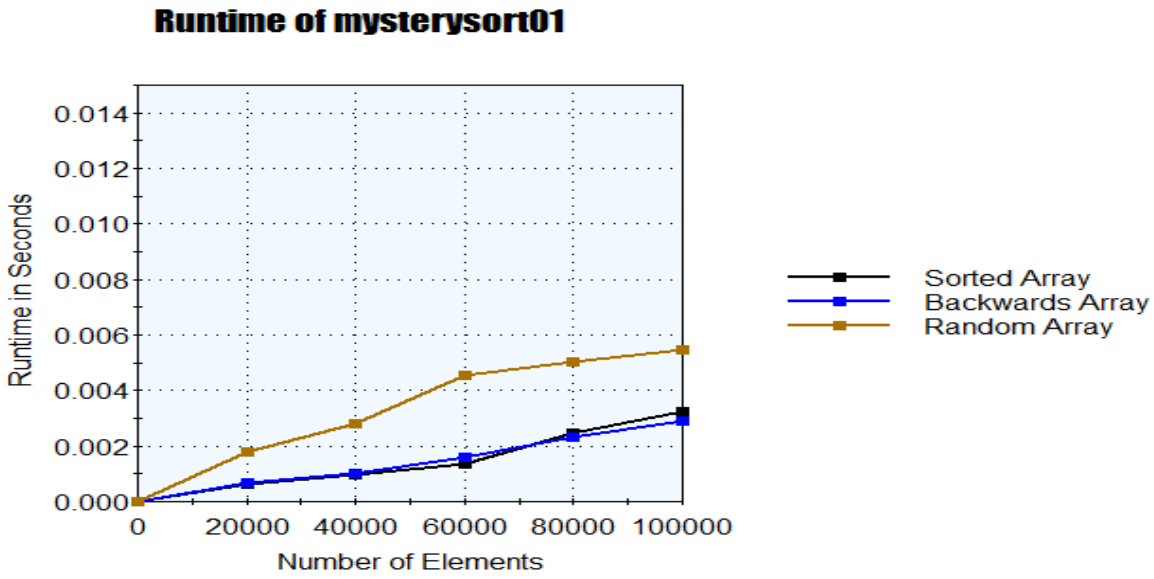
mysterysort03 = Insertion Sort

mysterysort04 = Quick Sort

mysterysort05 = Selection Sort

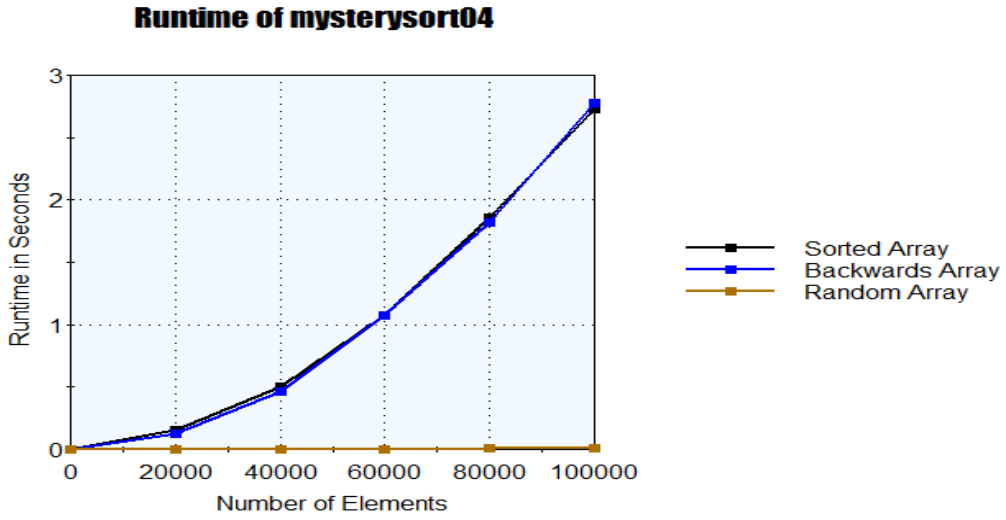
How I figured it out:

I know that the time complexity of Merge Sort is always O(n log n), no matter what, so it should always have a fast runtime as well as similar runtimes regardless of the order of the array, which is what mysterysort01 has. This is supported by the graph below, where all three arrays were sorted in less than 0.006 seconds.

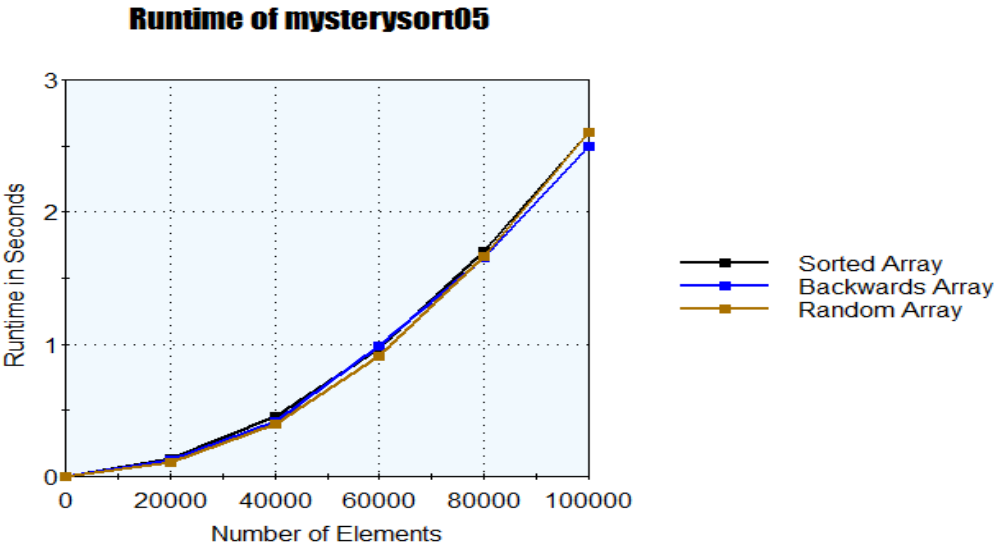


**This project was inspired by a similar one by Julie Zelenski and Colleagues at Stanford University*

I know that Quick Sort's worst case complexity is O(n^2) when the pivot element is either the smallest or biggest in an array, and because it has been specified that the pivot element is the last, that means that when I tested with both ascending and descending order of 100000 elements, it was dramatically longer than the random order. This also means that ascending and descending should have similar runtimes, as supported by the graph below. Quick Sort's best case complexity is O(n log n) when the pivot is near the middle in the array, which is what the Random Array is, and is supported in the graph.



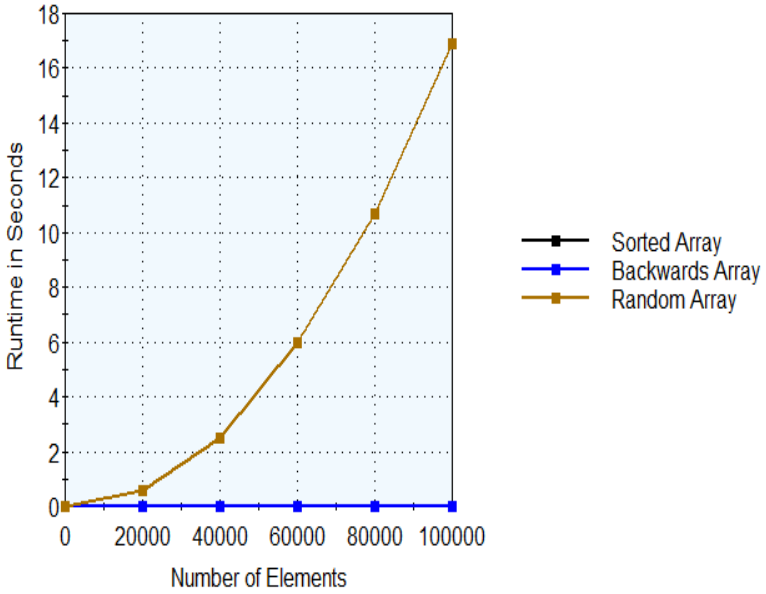
I know that Selection Sort always has a complexity of O(n^2), which means that no matter the order, it should always have a similar runtime. This means that mysterysort05 is Selection Sort because all three array runtimes have a parabolic curve, which means that the runtime complexity is O(n^2), which is supported by the graph below.



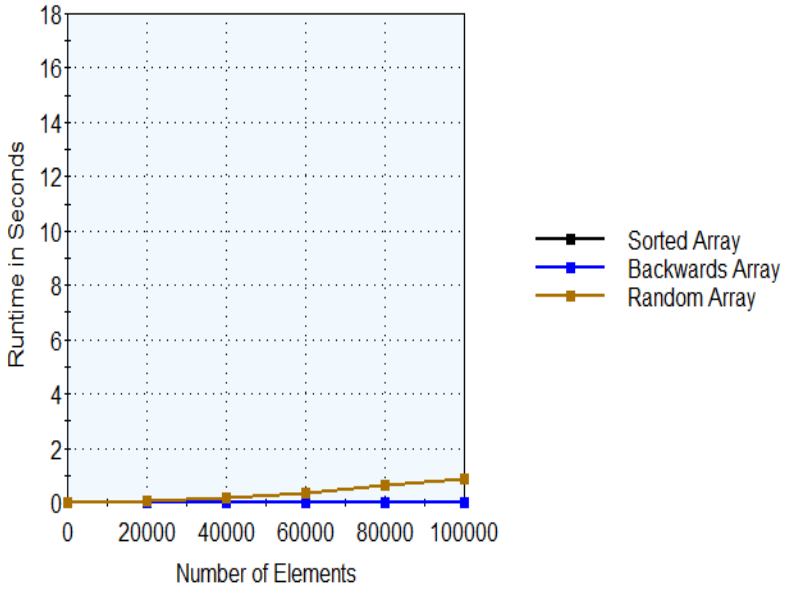
**This project was inspired by a similar one by Julie Zelenski and Colleagues at Stanford University*

I know that Bubble Sort and Insertion Sort have the same best, average, and worst case complexities; however, when dealing with sufficiently large enough elements, Bubble Sort becomes much slower than insertion Sort. When the array is already sorted, the runtime is O(n), which is supported by the two graphs below, so that leaves the random array, which is O(n^2). The graph shows that for random arrays mysterysort02 becomes much slower compared to mysterysort03 the more elements there are. The two graphs have the same scale for the y-axis in order to show the difference in the random sorted array times more clearly. Another graph was created to show that they have the same type of curve, being O(n^2) for random array, just at a different scale.

Runtime of mysterysort02



Runtime of mysterysort03



Runtime of mysterysort03 (Smaller Scale)