Scott Mackenzie

Dr. Fontenot

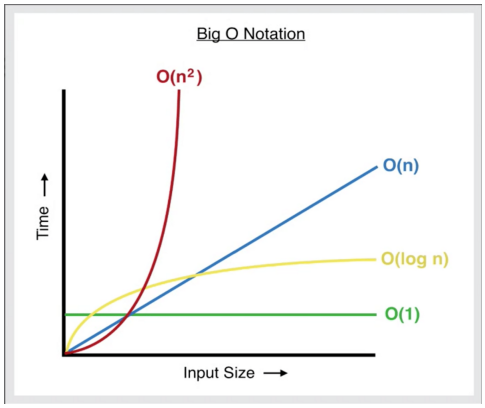
Mystery Sorter

October 10, 2021

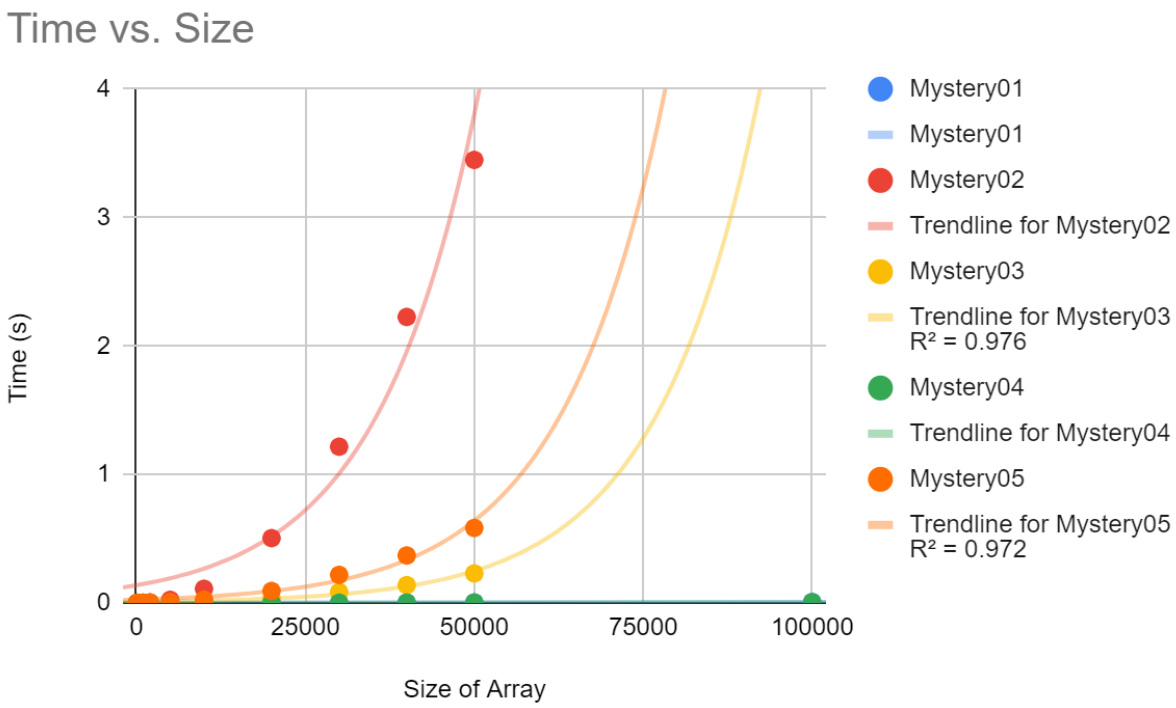
Mystery Sorter Essay

My name is Scott Mackenzie and I was given the task of using the *magic* of programming in C++ to attempt to lessen the havoc wreaked by the mischievous hackers on my humble professors project 3 repo. A daunting task indeed, it would be foolish to approach without a plan, therefore I concocted **this plan:**

1) **Separate into O(n^2) runtime functions and O(nlog(n)) (off average runtime)**
    - **First create a class to house my functions.**
    - **In this step, I plan to implement a void function that, when given an integer size, will randomly generate 15 arrays of that size, sort and time with each sorting algorithm, and average the 15 scores, then printing that average to the screen.**
    - **I will then graph the average time at a large range of sizes, and perform regression through google sheets, separating by growth rates into either category through comparison to this graph: (below)**



-Here is my graph:



-As you can see, quite clearly Mystery 02, 03, and 05 run at O(n^2) complexity, and Mystery 01 and 04 run at an O(nlogn) complexity.

-Link to data:

https://docs.google.com/spreadsheets/d/1P7273gvEH89omZNZbW3iyVBnb-7fid0oYPsjBg7oIGg/edit?usp=sharing

2) **Determine which function is Quicksort, which is Merge Sort, and which is insertion sort.**
    - **These can all be accomplished in the same step because both with quicksort and insertion sort, the sorting algorithm runs least optimally when passed an array in reverse order.**
    - **I am implementing a void function that will take an integer size and then create an array in reverse order of said size, I will then time each sorting algorithm given a reverse array, and print each.**
    - **Through comparing the sort speeds of each function given a random array of size 10000 to the results from a reverse array of 10000, the two functions showing the highest change in time would have to be insertion and quicksort, as they run least optimally in this case. Only one is nlogn regularly, therefore that one must be quicksort, and the other nlogn must be merge sort.**
    - **Therefore Mystery04 is quicksort, Mystery01 is merge sort, and Mystery03 is insertion sort**
3) **Determine which remaining sort is bubble sort and which is selection sort.**
    - **Bubble sort is significantly more inefficient at large sizes than selection sort, meaning that by passing each function a series of large arrays, the slower of the two would be bubble sort, and the faster would be selection sort.**
    - **Therefore Mystery02 is bubble sort, and Mystery05 is selection sort.**
4) **Answers**

- **Mystery01 = Merge Sort**
- **Mystery02 = Bubble Sort**
- **Mystery03 = Insertion Sort**
- **Mystery04 = Quicksort**
- **Mystery05 = Selection Sort**

(explanations above)