Rodrigo Pagán

Professor Laurence Liss

Project 1

11/17/2024

Project 1 | Sorting Algorithm Lab Report

This project consists of 6 sorting algorithms and arrays made of various sizes ranging from 10 to 100,000,000 with three different array characteristics: ascending, random, and descending. After having tested each of the algorithms, I found the most surprising result coming from Bogo sort when sorting through an ascending array. The algorithm itself is not all that reliable when sorting through most if not all the arrays because of how it only randomizes itself using the Fisher Yates Shuffle. However, when sorting an ascending array, it managed to identify the arrays as sorted much quicker than all the other algorithms. In the data I collected, Bogo Sort had a 0.03 time in seconds when sorting an array size of 100,000,000. This only beat out Bubble sort by 1 millisecond, while the other algorithms spent a longer time on their process starting around the 10,000 sizes with Selection sort. Selection sort had taken much longer to sort through larger arrays in comparison to the other algorithms around the 1,000,000 sizes. It would take close to if not 2 minutes, the time limit used, to complete its sort for each of the different characteristics. This is because Selection sort scans the array for the lowest element at the start before swapping the elements, making comparisons with the numbers inbetween until it finds the lowest element. The duplicated time for the data element in the chart for Selection sort with the Random array is from having ran the program twice; that first test-run took slightly less time than the one used in my collection of the data for the 1,000,000 size. During the first test-run of the program I had ommited any processes that took longer than 2 minutes after the first instance of the time limit being reached. Out of the six algorithms used, Quick sort and Merge sort are also known as divide-and-conquer algorithms[[1]](#footnote-1). These work by splitting the array into smaller arrays and sorting those before combining them into a fully sorted array. These are very effiecent and useful for the much larger arrays with Quick sort taking no longer than 15 seconds with the largest array in my data. When programming Quick sort, I ran into a minor issue with it because I had assigned the partition pivot point to the rightmost side of the array causing the algorithm to take much longer or crash when running, especially with larger arrays. After some research, I fixed this by instead assigning the pivot point to a random element in the partition[[2]](#footnote-2). Quick sort is my favorite of the 6 because of how effective it is in sorting larger arrays, and I like how it splits the arrays into chunks and swaps the elements. However, a close second for me would be Insertion sort because of how easy it is to implement despite it not being as efficient. I also like the visual representation of insertion sort being someone sorting through a standard deck of cards. Some other real-world implementations for the algorithms can be seen with virtual versions of cardgames like Yu-Gi-Oh, the leaderboards on competitive games, the stock market, library databases and other large databases can use these or combinations of these algorithms.

Charts

Works Cited

Bogo Sort - <https://en.wikipedia.org/wiki/Bogosort#Pseudocode>

Bubble Sort - <https://en.wikipedia.org/wiki/Bubble_sort#Pseudocode_implementation>

Selection Sort - <https://en.wikipedia.org/wiki/Selection_sort#Implementations>

Insertion Sort - <https://en.wikipedia.org/wiki/Insertion_sort#Algorithm>

Merge Sort - <https://en.wikipedia.org/wiki/Merge_sort#Algorithm>

Quicksort - <https://en.wikipedia.org/wiki/Quicksort#Algorithm>

Fisher Yates Shuffle - <https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle>

Divide-and-Conquer algorithm - <https://en.wikipedia.org/wiki/Divide-and-conquer_algorithm>

Choice of Pivot - <https://en.wikipedia.org/wiki/Quicksort#Choice_of_pivot>

1. Divide-and-Conquer algorithm - <https://en.wikipedia.org/wiki/Divide-and-conquer_algorithm> [↑](#footnote-ref-1)
2. Choice of Pivot - <https://en.wikipedia.org/wiki/Quicksort#Choice_of_pivot> [↑](#footnote-ref-2)