Assignment 3

Sorting your Affairs in Order

WRITEUP.pdf

Reuben T. Chavez

January 27, 2022

Introduction

For this write-up, we'll focus on the trends and the following program sorting.c. Which implements four different sorting algorithms to organize the same array of random integers whenever called, outputting the number of elements in the array, along with the number of moves and comparisons that it took for each algorithm to get the ordered array, and by understanding how each sorting algorithm work we get to understand the time complexity in effect for these algorithms.

The algorithms include the: Insertion Algorithms, the Heap Sort Algorithms, Batcher's Algorithm, and the Quick Sort Algorithm.

The Insertion algorithm orders an array by determining if going through each element in the array should be placed in front of the current element or behind the current element.

The heapsort algorithm works with binary trees, meaning the algorithm order everything with the greatest/lowest element at the top called a parent followed by two nodes called children that contain lesser or greater values – depending on whether it's a max heap or min-heap –, which is then followed by two more nodes until the last node contains the last element in the list. Using a max heap, the algorithm orders the array by first building a heap and then fixing the heap. Meaning the array elements are collected and ordered like the heap structure, and then the array is rearranged where the largest element is placed at the beginning of the array.

The Batcher Sort is described more as a sorting network where it actually sorts a set number of in two's. Where depending on the number of elements of k, it first sorts the array in indexes of k/2, then in k/4, and so on until it compares each value element individually if they are greater or lower than the previous value.

The Quicksort Algorithm is noted to be a dived and conquer algorithm that actually breaks the array in two and selects an element as a pivot. Depending on the pivot, the element is moved to either the left or right if they are smaller or of greater value than the pivot, respectively. As well, utiles a recursive algorithm to accomplish this for every element in the array.

Graph Anyalsis

The first two graphs note how the number of moves changes as the number of elements increases, the second following two look for how the number of comparison change as the number of elements is increased.

Upon graphing the moves, there are three notable features in both graphs on how many moves it takes for an ordered pair. With Insertion being the highest, it deduces that it takes more time to compile and collect the information. As Insertion is denoted as an O(n) it remains consistent but is much slower with larger arrays. Quicksort, though more erratic as the elements increase, takes fewer moves to solve the ordered pair. As mentioned to have time complexity of $O(n^2), itslowlybutsurelystartstocurveasthedown.WhileBatcherandHeapstarttoint$

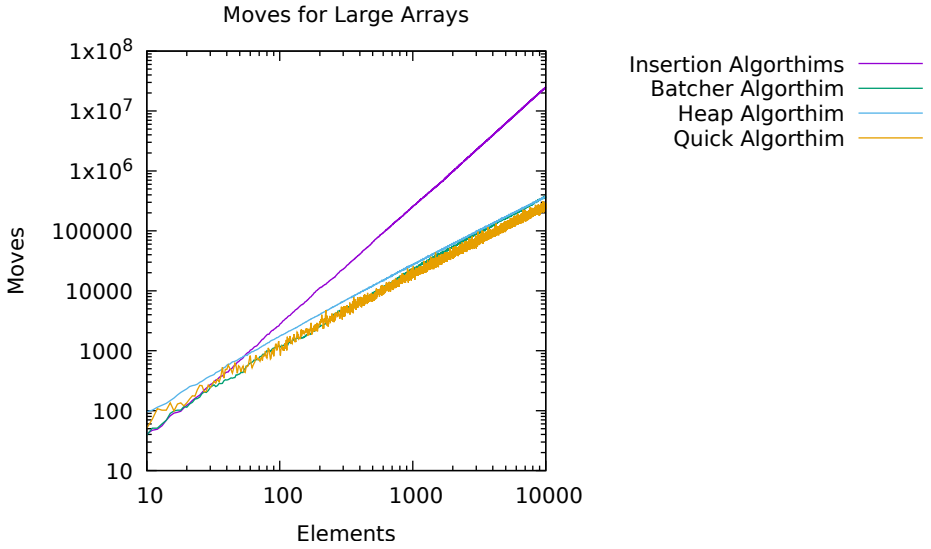

Figure 1:

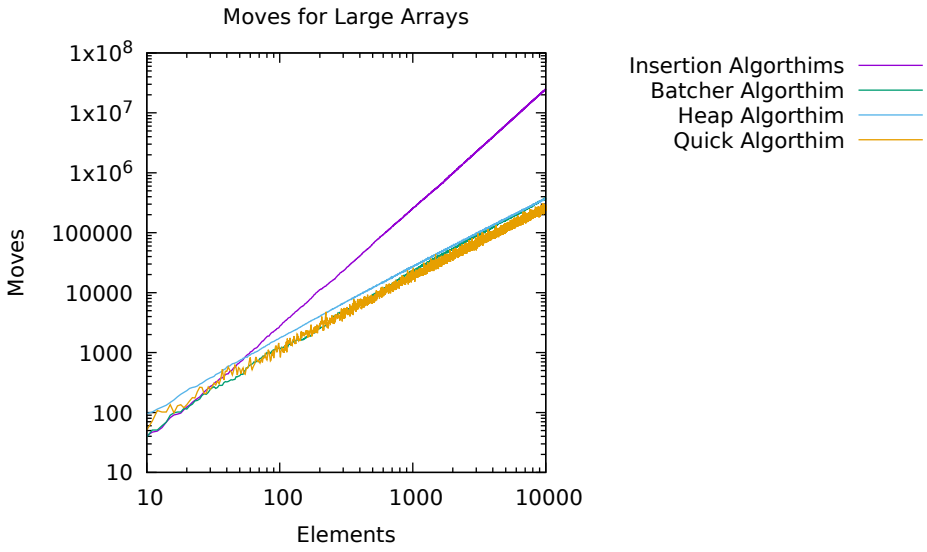
Moves for Large Arrays
Moves
Elements
1x10^8
1x10^7
1x10^6
100000
10000
1000
100
10
10
100
1000
10000
Insertion Algorthims
Batcher Algorthim
Heap Algorthim
Quick Algorthim

Figure 2:

Number of Comparisons as Array Increases

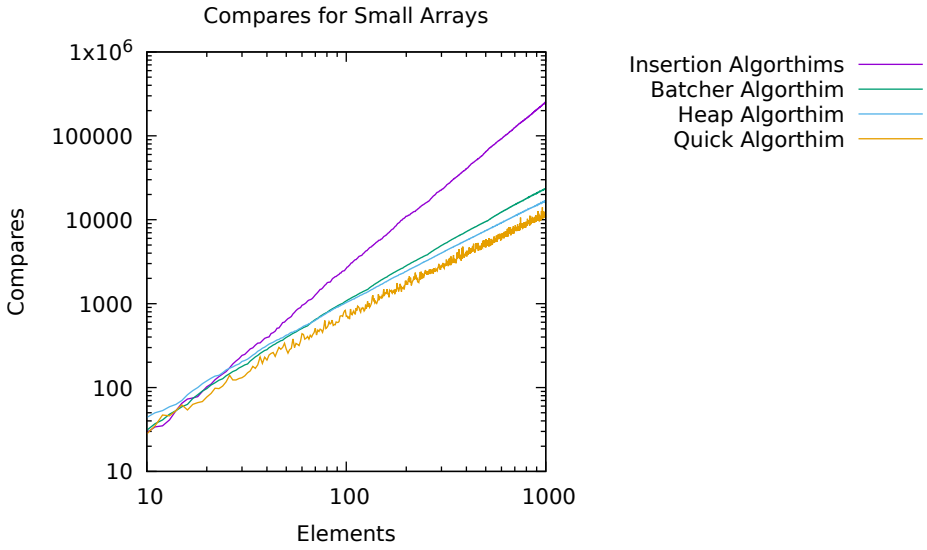

Figure 3:

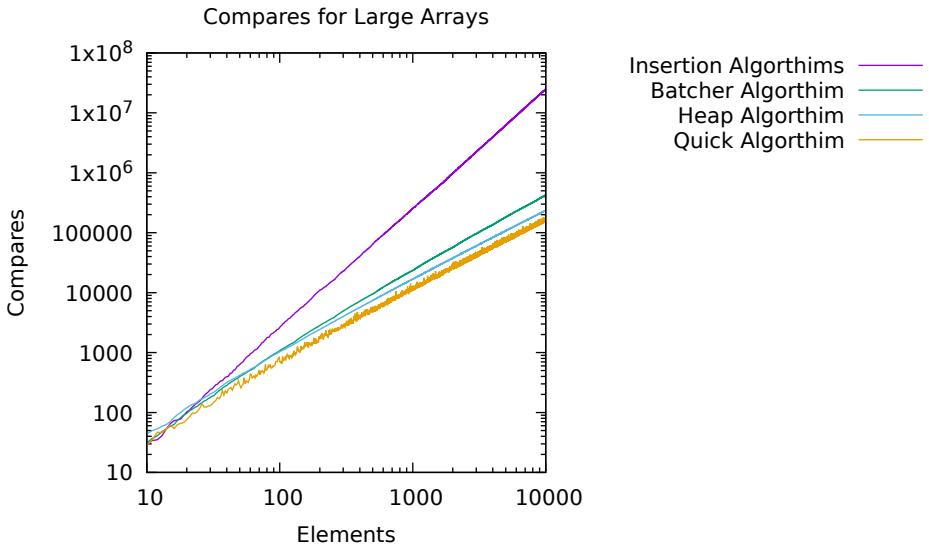
Compares for Large Arrays
Compares
Elements
1x10^8
1x10^7
1x10^6
100000
10000
1000
100
10
10
100
1000
10000
Insertion Algorthims
Batcher Algorthim
Heap Algorthim
Quick Algorthim

Figure 4: