Sorting Algorithm Analysis

Matthew Galitz

University of Virginia, Charlottesville, VA 22904

Abstract.
The purpose of the experiment is to compare the efficiencies of various sorting algorithms including: Bubble Sort, Insertion Sort, Merge Sort, Quick Sort, and a hybrid sorting algorithm called Tim Sort.

1 Introduction

1.1 Problem

The problem is to determine the relative efficiencies of the sorting algorithms.

1.2 Description

Bubble Sort compares adjacent elements and swaps them if they are in the wrong order. Insertion sort place an element in its final sorted position one element at a time. Merge Sort is a recursively divides an array into subarrays, sort the subarrays, and combines the subarrays to form the sorted array. Quick sort recursively selects a pivot point in arrays and organizes the elements less and greater than the partition into respective subarrays and combines the subarrays to form the sorted array. Tim Sort is a combination of Merge Sort and Insertion Sort.

2 Methods

Each algorithm is tested against three different types of arrays of 12,000 elements for 6 trials. The three types of arrays include arrays of random order, arrays in reverse order, and arrays that are "almost" sorted (containing 5-10 inversions).

Trials Run for Each Algorithm

	# of Trials
Bubble	6
Insertion	6
Merge	6
Quick	6
Tim	6

The average of the data for each algorithm is calculated using the following equation/

$$A = \frac{\sum_{n=1}^{N} x_n}{N}$$

Where x_n is the time elapsed for trial n, N is the total number of trials, and A is the average time for a particular algorithm.

3 Results

Time (ms) for Algorithm to Execute on Array of Random Order

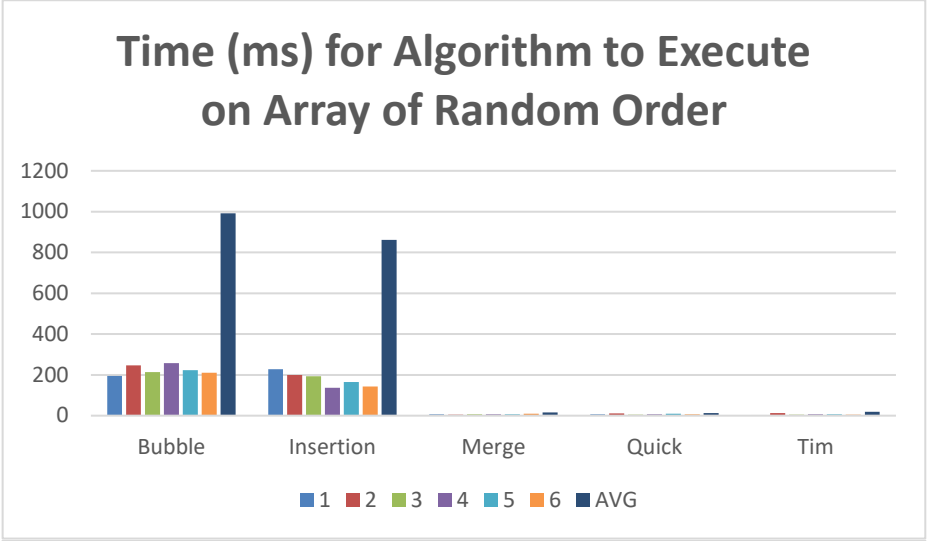
	1	2	3	4	5	6	AVG
Bubble	195	247	214	257	223	211	991
Insertion	227	200	193	136	165	143	861
Merge	6	5	6	7	6	10	15
Quick	6	11	5	6	9	6	12
Tim	4	13	5	7	7	5	19

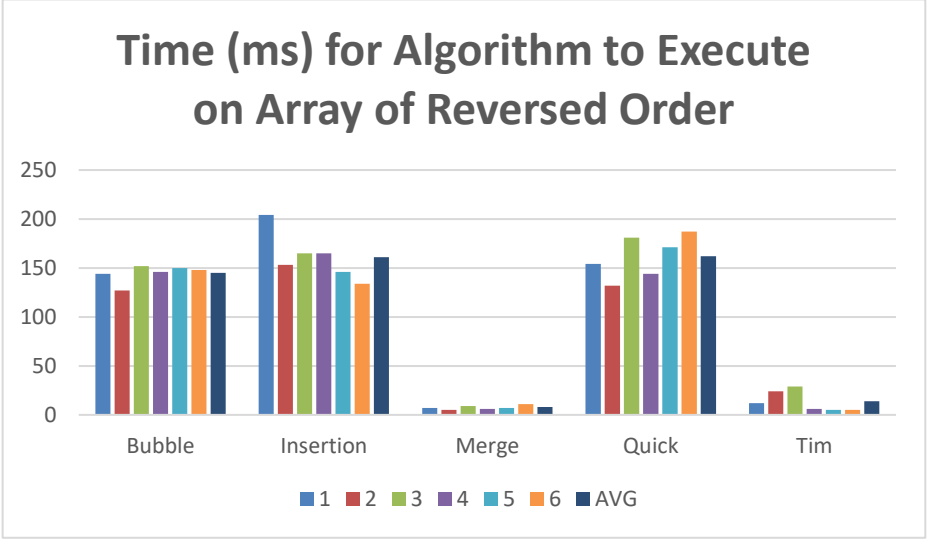
Time (ms) for Algorithm to Execute on Array of Reversed Order

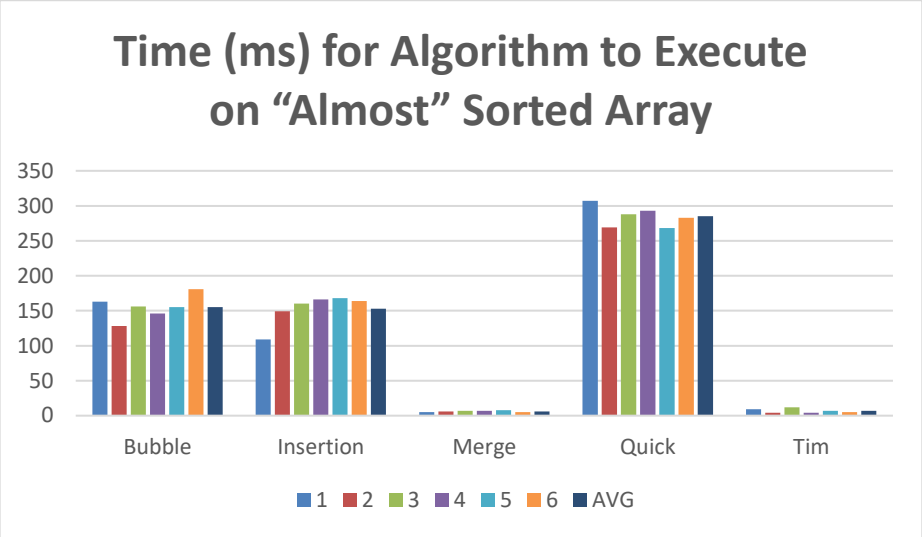
	1	2	3	4	5	6	AVG
Bubble	144	127	152	146	150	148	145
Insertion	204	153	165	165	146	134	161
Merge	7	5	9	6	7	11	8
Quick	154	132	181	144	171	187	162
Tim	12	24	29	6	5	5	14

Time (ms) for Algorithm to Execute on "Almost" Sorted Array

	1	2	3	4	5	6	AVG
Bubble	163	128	156	146	155	181	155
Insertion	109	149	160	166	168	164	153
Merge	5	6	7	7	8	5	6
Quick	307	269	288	293	268	283	285
Tim	9	4	12	4	7	5	7


Time (ms) for Algorithm to Execute on Array of Random Order
1200
1000
800
600
400
200
0
Bubble
Insertion
Merge
Quick
Tim
1 2 3 4 5 6 AVG


Time (ms) for Algorithm to Execute on Array of Reversed Order
250
200
150
100
50
0
Bubble
Insertion
Merge
Quick
Tim
1 2 3 4 5 6 AVG



4 Conclusion

For an array of randomly sorted elements, Merge Sort, Quick Sort, and Tim Sort are the optimal algorithms. For an array of reverse sorted Merge Sort and Tim Sort are optimal. Quicksort performs worse similarly . For an array of “almost” sorted elements, Merge Sort and Tim Sort are optimal. Quick Sort performs worse than both Bubble Sort and Insertion Sort.