Alan Rodriguez

Project 1

2/20/2016

Project 1 Analysis

For Selection Sort algorithm, I expected the constant and sorted arrays to be the best-case scenarios, and they seemed to be at first. I also noticed that all the selection sort arrays seemed to take the longest out of all the algorithms and this makes sense considering that for all cases, the algorithm compares the minimum value of the array to every value in the array every iteration. The random array seemed to follow the same complexity as the other constant and sorted arrays with a worst-case of O(n^2).

For Insertion Sort, again I expected the constant and sorted arrays to be best-case scenarios, which they were. They ran at a time complexity of Ω(n), while random had a worst-case scenario of O(n^2). All variations of the arrays ran at expected time complexities.

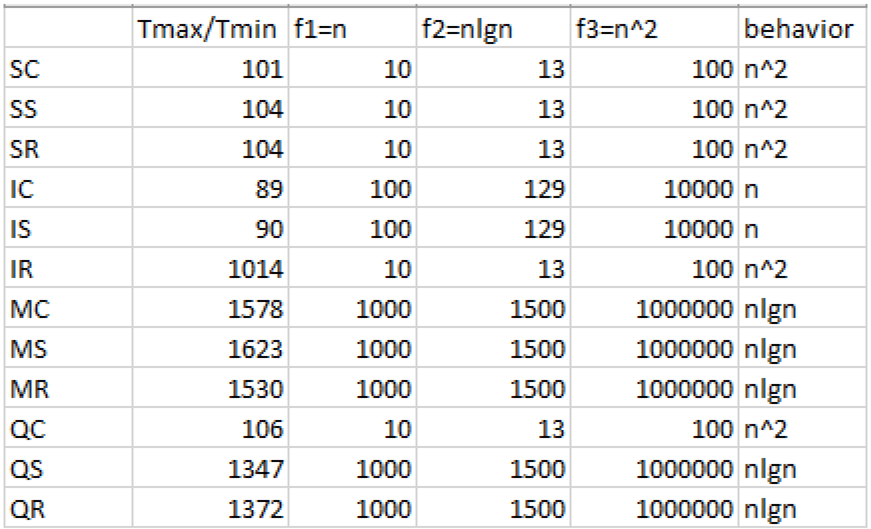
For the Merge Sort algorithm, constant, sorted, and random ran at Θ(nlgn), which is the expected rate of Merge Sort algorithm, Ω = Θ = O(nlgn). Although I expected random to run at a slower time complexity as it did before in Insertion sort, I was surprised that merge sort's random array ran at the same rate of constant and sorted. That implies that for larger arrays, merge sort would be more favorable then insertion sort, and selection sort.

For the Quick Sort algorithm I expected that the sorted array and constant array would be the best-case scenarios and random the worst, but constant seemed to have a worst case of O(n^2). I suppose this makes sense based on the fact that pivot is the partition of the array, but when the pivot is selected as the first element every iteration, it would be very inefficient resulting in a worst case scenario runtime of O(n^2).

Alan Rodriguez

Project 1

2/20/2016



	Tmax/Tmin	f1=n	f2=nlgn	f3=n^2	behavior
SC	101	10	13	100	n^2
SS	104	10	13	100	n^2
SR	104	10	13	100	n^2
IC	89	100	129	10000	n
IS	90	100	129	10000	n
IR	1014	10	13	100	n^2
MC	1578	1000	1500	1000000	nlgn
MS	1623	1000	1500	1000000	nlgn
MR	1530	1000	1500	1000000	nlgn
QC	106	10	13	100	n^2
QS	1347	1000	1500	1000000	nlgn
QR	1372	1000	1500	1000000	nlgn