David Hatcher

U74842093

Project 1

Analysis

	Tmax/Tmin	n ratio	nln(n) ratio	n^2 ratio	Behavior
SC	11210.67	110.00	170.71	12100.00	n^2
SS	11211.67	110.00	170.71	12100.00	n^2
SR	11214.00	110.00	170.71	12100.00	n^2
IC	111.33	100.00	128.57	10000.00	n
IS	111.67	100.00	128.57	10000.00	n
IR	10175.67	100.00	152.01	10000.00	n^2
MC	3020.00	2000.00	3158.47	4000000.00	nln(n)
MS	3282.67	2000.00	3158.47	4000000.00	nln(n)
MR	8056.73	5000.00	8488.92	25000000.00	nln(n)
QC	15653.00	125.00	192.16	15625.00	n^2
QS	2066.33	1666.67	2595.99	2777777.78	nln(n)
QR	7834.00	5000.00	8488.92	25000000.00	nln(n)

Selection Sort

With Selection Sort the algorithm will look through the entire array to find the smallest value on the first loop then swap it with the item in the first index, on the next loop it will do the same thing but starting from and swapping with the second index, it will continue this until it reaches the end. As the swaps and comparisons are constant time operations it means that the entire sort time should be O(n^2). Thus, it makes sense that in all cases of a constant array, a sorted array, and a random array it will always be n^2.

Insertion Sort

With Insertion Sort the algorithm will look through the entire array and checking if the current index value is less than any of the previous values. Because of this trait it allows it to skip searching if the current value, a, is greater than the previous value, b, as everything before b is less than b thus a must also be greater than them as well. Following this logic, it means that an array that is already sorted in ascending order would result in the Best-case complexity and an array sorted in descending order would result in the Worst-case complexity. Since both, a Sorted array and a Constant array are sorted it makes it makes sense that running insertion sort on both would result in the Best-case complexity. While the random comes out to an Average-case complexity which makes sense as Insertion Sorts Average-case complexity is a randomly sorted array.

As a side note on this experiment the values for n ratio and nln(n) ratio were both very close to the value for Tmax/Tmin. However, it is a closer to the ratio for n and the Best-case for Insertion Sort is Omega(n), these two facts are the reason I selected n as the behavior for IC and IS as these are both bound below by the n ratio.

Merge Sort

With Merge Sort as it is a divide and conquer algorithm it will split the array in half until you have a single item array and start sorting then putting them back together, since it'll take a total of lgn steps to break it down and n comparisons it will always been nlgn. Thus, it makes sense that all of these came out to be very close, within 5.1%, of the nln(n) value.

Quick Sort

For Quick Sort, the Worst-case complexity happens whenever the algorithm picks either the min or max value as its pivot. As this algorithm chooses the middle value in the array as its pivot it is unlikely for that to happen in a random array, and impossible to happen in a sorted array. However, this will always happen in an array that has only one value. Based on this it makes sense that the constant array for Quick Sort to match the Worst-case complexity and for the Sorted and Random arrays to match the average and best case complexities of algorithm.