Project One: Asymptotic Analysis

Analysis

	t_{max}/t_{min}	F_1 ratio	F_2 ratio	F_3 ratio	Behavior
IC	206	**208**	281	43,403	Linear
IS	208	**213**	287	45,269	Linear
IR	29,945	178	286	**31,524**	Quadratic
MC	5,350	3,571	**5,901**	12,755,102	n · ln (n)
MS	6,118	3,690	**6,113**	13,616,372	n · ln (n)
MR	14,148	8,065	**14,250**	65,036,420	n · ln (n)
QC	30,000	172	275	**29,727**	Quadratic
QS	4,063	2,725	**4,407**	7,424,511	n · ln (n)
QR	13,894	8,333	**14,766**	69,444,444	n · ln (n)

Summary

Insertion Sort

For constant and sorted arrays, my testing resulted in **linear** behavior which is the **best-case** complexity for this algorithm. This makes sense due to the way in which Insertion Sort operates: selecting an unsorted element and determining its position in the sorted array by iteratively comparing it to already sorted elements. For random arrays, I concluded that the algorithm was **quadratic** in behavior, or **average/worst-case** complexity according to theoretical analysis of the algorithm.

Merge Sort

For each type of array - constant, sorted, and random - Merge Sort resulted in **logarithmic** behavior. This is entirely accurate as this algorithm has the same behavior for **best-**, **average-**, and **worst-case** complexity. These complexities are a result of the fact that Merge Sort is a divide-and-conquer algorithm which excels at sorting by splitting the array into smaller sub-arrays that are easier to sort.

Quick Sort

According to my results, Quick Sort has **logarithmic** behavior for sorted and random arrays but **quadratic** behavior for constant arrays. According to theoretical analysis, this indicates that sorted and random arrays are **best-** or **average-case** complexity while constant arrays are **worst-case** complexity. This result makes sense because the selected pivot is the same as every element in the array. In this case, Quick Sort can only divide the array into subarrays of size 0 and *n - 1*, where *n* is the number of elements in the array.