Aidan Kiser
14 October 2023
COMP-3270

Programming Assignment 1 Results & Analysis

	n_{min}	t_{min} (milliseconds)	n_{max}	t_{max} (minutes)
SC	10^5	~ 819.21	10^7	~13.65
SS	10^6	~815.21	10^8	~13.59
SR	10^4	~32.40	10^7	~540.00
IC	10^8	~20.31	10^{11}	~338.50
IS	10^8	~20.39	10^{11}	~339.83
IR	10^8	~147.39	10^9	~24.57
MC	10^6	~23.47	10^8	~362.50
MS	10^6	~21.75	10^8	~391.17
MR	10^6	~34.46	10^8	~574.33
QC	10^5	~158.73	10^9	~26.29
QS	10^5	~148.42	10^9	~24.74
QR	10^4	~22.42	10^8	~37.37

	$t_{\text{max}}/t_{\text{min}}$	n ratio	$n \ln(n)$ ratio	n^2 ratio	Behavior
SC	$10^7 / 10^5$	100	140	10^{24}	Quadratic
SS	$10^8 / 10^6$	100	133.33	10^{28}	Quadratic
SR	$10^7 / 10^4$	1000	1750	10^{33}	Quadratic
IC	$10^{11} / 10^8$	1000	1375	10^{57}	Quadratic
IS	$10^{11} / 10^8$	1000	1375	10^{57}	Quadratic
IR	$10^9 / 10^8$	10	11.25	10^7	Quadratic
MC	$10^8 / 10^6$	100	133.33	10^{28}	$n \lg(n)$
MS	$10^8 / 10^6$	100	133.33	10^{28}	$n \lg(n)$
MR	$10^8 / 10^6$	100	133.33	10^{28}	$n \lg(n)$
QC	$10^9 / 10^5$	10000	18000	10^{56}	Quadratic
QS	$10^9 / 10^5$	10000	18000	10^{56}	Quadratic
QR	$10^8 / 10^4$	10000	20000	10^{48}	$n \lg(n)$

Report:

1. For the four algorithms that were being tested, selection, insertion, merge, and quick sort, they compare nicely to theoretical analysis of each algorithm respectively.
    - Selection sort is normally O(n^2) in every case, and we can see from my data that it matches theoretical analysis.
    - Insertion sort for the worst and average case is O(n^2), while the best case is O(n), however, we can see from my data that insertion sort follows the theoretical analysis of the worst and average cases.
    - Merge sort is almost always O(n(lg(n)), and we can see from my data that that is the case.
    - Quick sort for the best and average case is O(n(lg(n)), while the worst case is O(n^2), and we can see that this holds as shown in my last row, the quicksort ran an average case with O(n(lg(n)), but normally runs O(n^2).

2. My results for the most part match the theoretical analysis of each algorithm. The fact that your own selection, insertion, merge, and quicksort functions yielded results very similar to the theoretical analyses of each algorithm is a testament to the effectiveness of these sorting algorithms and the accuracy of my implementations. The only case in which the algorithm may not have run completely normally was when quicksort was run on a randomly sorted array, in which its behavior displayed O(n(lg(n)). However, this is only average case, so it is not extremely significant.