Samantha Lane
Data Structures – Lab 4 Analysis
8/12/2025

For this lab, I implemented and evaluated five sorting algorithms: four variants of Quicksort and one linked-list–based Natural Merge Sort. The Quicksort variants differed by pivot selection strategy—either the first element or a median-of-three—and by whether small partitions switched to insertion sort, using thresholds of 50 and 100 elements. The Natural Merge Sort used a linked-list representation to allow merges to be performed via pointer relinking rather than allocating temporary arrays.

The data structure choices were deliberate. Quicksort benefits from array-based storage due to constant-time random access, enabling fast partitioning and in-place swaps. In contrast, the linked-list version of Natural Merge Sort exploits its structure for space efficiency in merging—eliminating the need for large auxiliary arrays. To facilitate performance measurement, each run tracked comparisons and exchanges using a dedicated counter object.

Experimental Setup
I tested each algorithm on integer sequences of sizes 50, 2,000, 5,000, and 10,000, with three input orders: ascending, descending, and random. The smallest file served to check correctness, while larger files revealed scaling behavior.

For example:

- qsort_median3_stop12 on the 2,000-element random file performed 30,580 comparisons and 7,897 exchanges.
- nat_merge_linked on the 10,000-element ascending file performed 9,999 comparisons and 0 exchanges, approaching the ideal O(n) case.

In the worst-case scenario for first-pivot Quicksort—10,000 elements in descending order, qsort_first_stop12 required 186,992 comparisons and 93,494 exchanges. These statistics provided a direct view into the operational cost differences between algorithms.

Results and Observations
Impact of Input Order:
Natural Merge Sort excelled on sorted and nearly sorted data, where comparisons approached n and exchanges were negligible. Quicksort's sensitivity to order was evident:

- First-pivot selection degraded performance sharply on ordered inputs.
- Median-of-three pivoting mitigated this by producing more balanced partitions, cutting comparisons by over 35% for the 10,000-element descending case compared to first-pivot.

Effect of Insertion Sort Cutoff:
Hybrid Quicksort with insertion sort for small partitions consistently reduced operation counts on large, random datasets:

- For 2,000 random elements, first pivot, insertion at 100 yielded 25,448 comparisons and 6,774 exchanges, versus 30,580 and 7,897 for the no-cutoff version.
- The 100-element cutoff slightly outperformed the 50-element cutoff in most large datasets.

Relative Performance:
On random data, Quicksort variants generally performed fewer comparisons than Natural Merge Sort, especially for larger sizes. However, for ordered inputs, the linked-list merge sort dominated in both comparisons and exchanges. The merge sort also maintained stable O(n log n) growth for random and descending data, while first-pivot Quicksort risked $O(n^2)$ in adverse cases.

Key Takeaways

- Pivot selection (especially median-of-three) is critical to avoiding worst-case Quicksort performance.
- Insertion sort cutoffs reduce the overhead of recursive calls on small partitions.
- Natural Merge Sort is highly competitive, especially when input data has partial order, and benefits from low memory overhead in its linked implementation.
- No single algorithm is universally optimal; selecting the right approach depends on expected data characteristics.

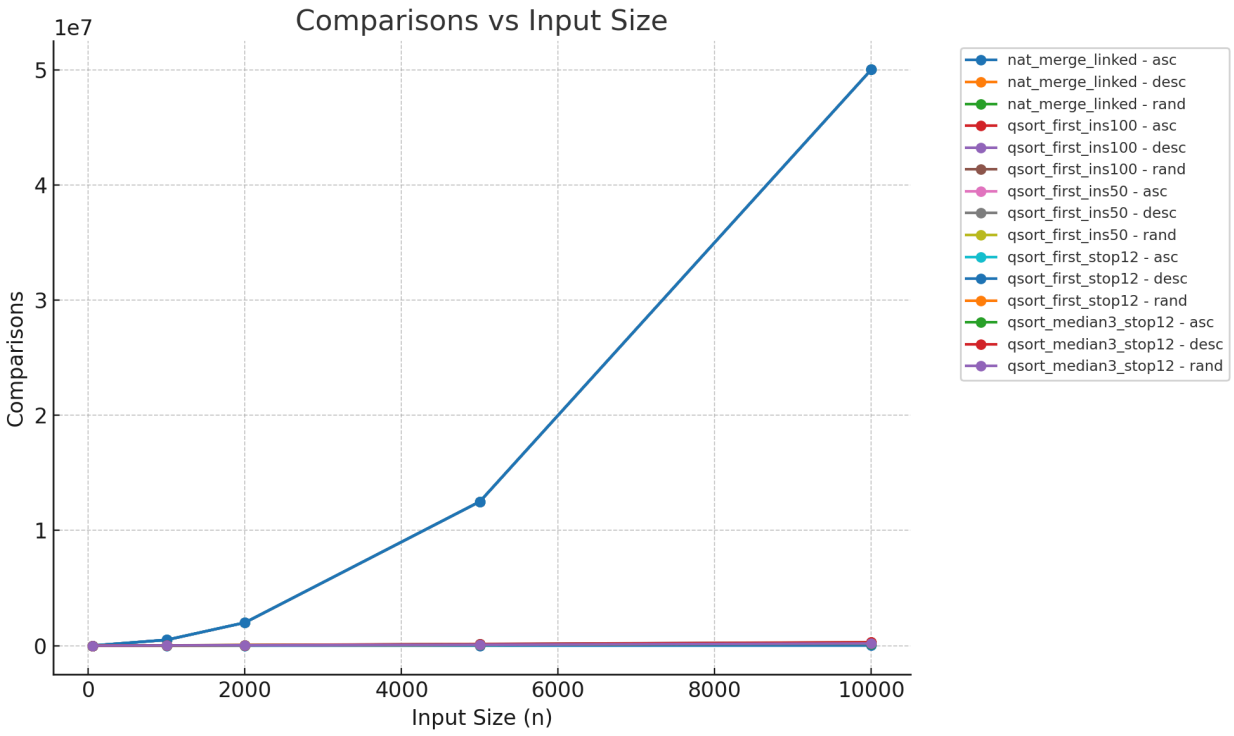
Sorting Results With Comparisons And Exchanges

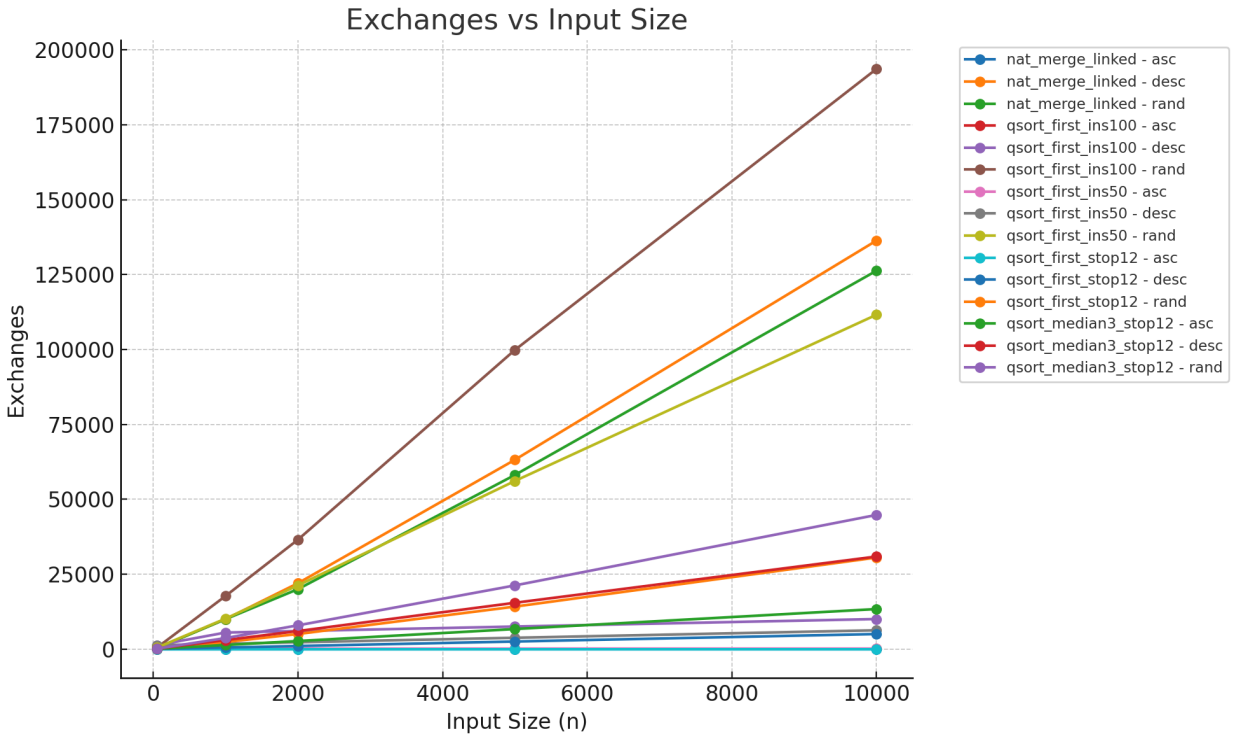
Algorithm	Input File	Comparisons	Exchanges	Size	Order
nat_merge_linked	50_asc.txt	49	0	50	asc
nat_merge_linked	50_desc.txt	476	294	50	desc
nat_merge_linked	50_rand.txt	494	247	50	rand
nat_merge_linked	1000_asc.txt	999	0	1000	asc
nat_merge_linked	1000_desc.txt	15921	9984	1000	desc
nat_merge_linked	1000_rand.txt	19753	9976	1000	rand
nat_merge_linked	2000_asc.txt	1999	0	2000	asc
nat_merge_linked	2000_desc.txt	34852	21968	2000	desc
nat_merge_linked	2000_rand.txt	40475	19982	2000	rand
nat_merge_linked	5000_asc.txt	4999	0	5000	asc

nat_merge_linked	5000_desc.txt	99790	63160	5000	desc
nat_merge_linked	5000_rand.txt	119202	58092	5000	rand
nat_merge_linked	10000_asc.txt	9999	0	10000	asc
nat_merge_linked	10000_desc.txt	214593	136320	10000	desc
nat_merge_linked	10000_rand.txt	258741	126256	10000	rand
qsort_first_ins100	50_asc.txt	49	49	50	asc
qsort_first_ins100	50_desc.txt	1225	1274	50	desc
qsort_first_ins100	50_rand.txt	461	465	50	rand
qsort_first_ins100	1000_asc.txt	496449	99	1000	asc
qsort_first_ins100	1000_desc.txt	501750	5499	1000	desc
qsort_first_ins100	1000_rand.txt	22545	17727	1000	rand
qsort_first_ins100	2000_asc.txt	1997949	99	2000	asc
qsort_first_ins100	2000_desc.txt	2003750	5999	2000	desc
qsort_first_ins100	2000_rand.txt	49310	36509	2000	rand
qsort_first_ins100	5000_asc.txt	12502449	99	5000	asc
qsort_first_ins100	5000_desc.txt	12509750	7499	5000	desc
qsort_first_ins100	5000_rand.txt	142214	99818	5000	rand
qsort_first_ins100	10000_asc.txt	50009949	99	10000	asc
qsort_first_ins100	10000_desc.txt	50019750	9999	10000	desc
qsort_first_ins100	10000_rand.txt	297598	193608	10000	rand
qsort_first_ins50	50_asc.txt	49	49	50	asc
qsort_first_ins50	50_desc.txt	1225	1274	50	desc
qsort_first_ins50	50_rand.txt	461	465	50	rand
qsort_first_ins50	1000_asc.txt	500224	49	1000	asc
qsort_first_ins50	1000_desc.txt	501875	1749	1000	desc
qsort_first_ins50	1000_rand.txt	16124	10160	1000	rand
qsort_first_ins50	2000_asc.txt	2001724	49	2000	asc
qsort_first_ins50	2000_desc.txt	2003875	2249	2000	desc

qsort_first_ins50	2000_rand.txt	36791	21235	2000	rand
qsort_first_ins50	5000_asc.txt	12506224	49	5000	asc
qsort_first_ins50	5000_desc.txt	12509875	3749	5000	desc
qsort_first_ins50	5000_rand.txt	105068	56090	5000	rand
qsort_first_ins50	10000_asc.txt	50013724	49	10000	asc
qsort_first_ins50	10000_desc.txt	50019875	6249	10000	desc
qsort_first_ins50	10000_rand.txt	228687	111566	10000	rand
qsort_first_stop12	50_asc.txt	1321	0	50	asc
qsort_first_stop12	50_desc.txt	1345	25	50	desc
qsort_first_stop12	50_rand.txt	397	64	50	rand
qsort_first_stop12	1000_asc.txt	501496	0	1000	asc
qsort_first_stop12	1000_desc.txt	501995	500	1000	desc
qsort_first_stop12	1000_rand.txt	14646	2330	1000	rand
qsort_first_stop12	2000_asc.txt	2002996	0	2000	asc
qsort_first_stop12	2000_desc.txt	2003995	1000	2000	desc
qsort_first_stop12	2000_rand.txt	33389	5009	2000	rand
qsort_first_stop12	5000_asc.txt	12507496	0	5000	asc
qsort_first_stop12	5000_desc.txt	12509995	2500	5000	desc
qsort_first_stop12	5000_rand.txt	95144	14150	5000	rand
qsort_first_stop12	10000_asc.txt	50014996	0	10000	asc
qsort_first_stop12	10000_desc.txt	50019995	5000	10000	desc
qsort_first_stop12	10000_rand.txt	209633	30527	10000	rand
qsort_median3_stop12	50_asc.txt	435	64	50	asc
qsort_median3_stop12	50_desc.txt	542	141	50	desc
qsort_median3_stop12	50_rand.txt	462	138	50	rand
qsort_median3_stop12	1000_asc.txt	13136	1332	1000	asc
qsort_median3_stop12	1000_desc.txt	19838	2986	1000	desc
qsort_median3_stop12	1000_rand.txt	14262	3666	1000	rand

qsort_median3_stop12	2000_asc.txt	28270	2664	2000	asc
qsort_median3_stop12	2000_desc.txt	44303	5994	2000	desc
qsort_median3_stop12	2000_rand.txt	30580	7897	2000	rand
qsort_median3_stop12	5000_asc.txt	77483	6664	5000	asc
qsort_median3_stop12	5000_desc.txt	128585	15426	5000	desc
qsort_median3_stop12	5000_rand.txt	84641	21220	5000	rand
qsort_median3_stop12	10000_asc.txt	164979	13332	10000	asc
qsort_median3_stop12	10000_desc.txt	280922	30876	10000	desc
qsort_median3_stop12	10000_rand.txt	184976	44749	10000	rand


Comparisons vs Input Size
1e7
Comparisons
Input Size (n)
0
1
2
3
4
5
0
2000
4000
6000
8000
10000
nat_merge_linked - asc
nat_merge_linked - desc
nat_merge_linked - rand
qsort_first_ins100 - asc
qsort_first_ins100 - desc
qsort_first_ins100 - rand
qsort_first_ins50 - asc
qsort_first_ins50 - desc
qsort_first_ins50 - rand
qsort_first_stop12 - asc
qsort_first_stop12 - desc
qsort_first_stop12 - rand
qsort_median3_stop12 - asc
qsort_median3_stop12 - desc
qsort_median3_stop12 - rand


Exchanges vs Input Size
Exchanges
Input Size (n)
0
25000
50000
75000
100000
125000
150000
175000
200000
0
2000
4000
6000
8000
10000
nat_merge_linked - asc
nat_merge_linked - desc
nat_merge_linked - rand
qsort_first_ins100 - asc
qsort_first_ins100 - desc
qsort_first_ins100 - rand
qsort_first_ins50 - asc
qsort_first_ins50 - desc
qsort_first_ins50 - rand
qsort_first_stop12 - asc
qsort_first_stop12 - desc
qsort_first_stop12 - rand
qsort_median3_stop12 - asc
qsort_median3_stop12 - desc
qsort_median3_stop12 - rand