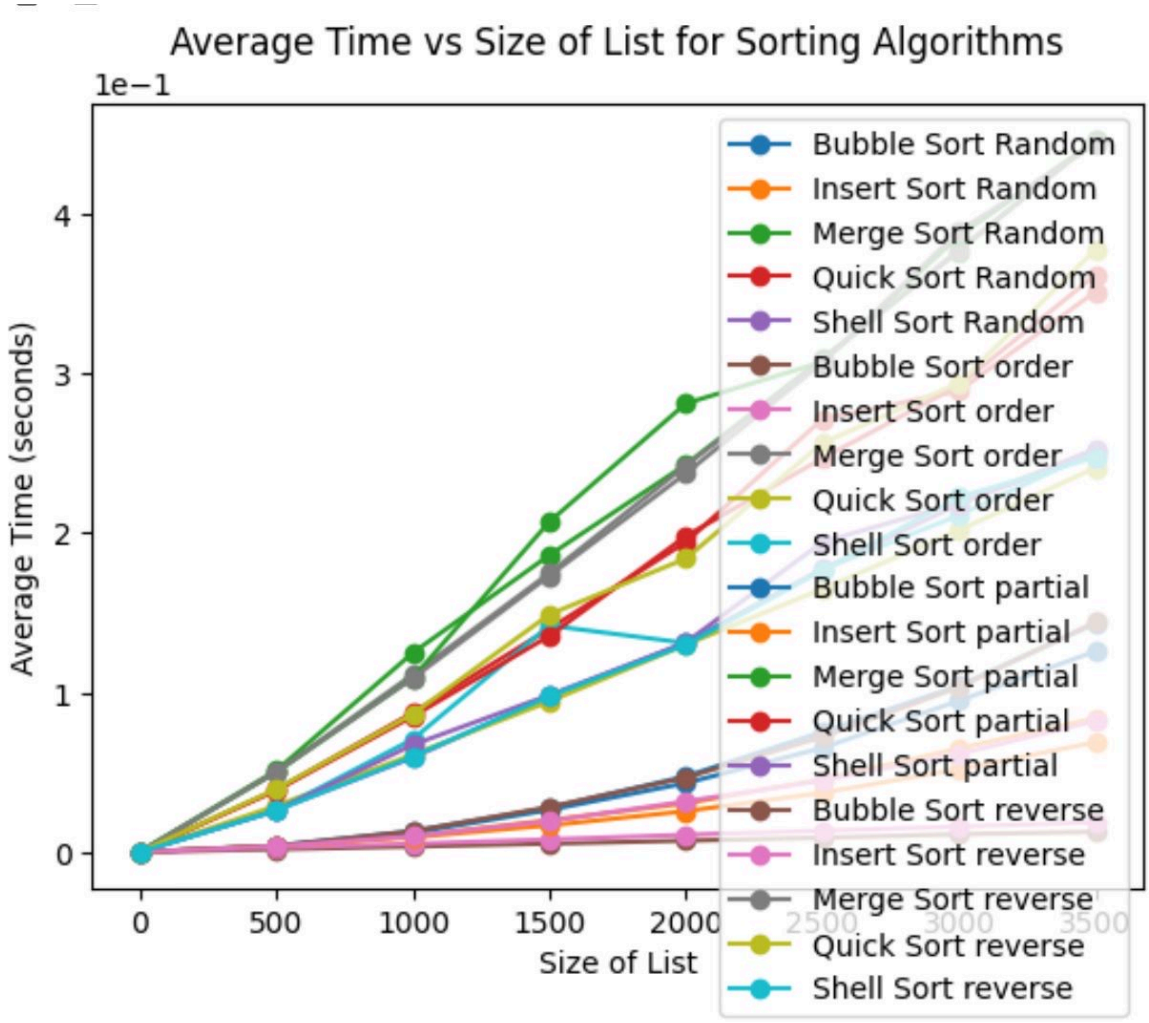
Report



gm

Observations

Bubble Sort

- Best Case: O(n) when the list is already sorted (order).
- Worst Case: O(n^2) when the list is in reverse order.
- Average Case O(n^2) for random lists.
- Observations: The graph should show bubble sort performing best for ordered lists and worst for reverse sorted lists. It is generally inefficient for large random lists causing my computer to crash multiple times.

Insertion Sorting:

- Best Case: O(n) for ordered lists.

- Worst Case: O(n^2) for reverse ordered lists.
- Average Case: O(n^2) for random lists.

Observations: Similar to bubble sort, insertion sort should show quick runtimes for nearly sorted lists and slowest for reverse sorted lists.

. **Merge Sort**
- Best Case: O(n log n) for all cases.
- Worst Case O(n log n) for all cases.
- Average Case: O(n log n) for all cases.
- Observations: Merge sort should show consistent performance out of all the other cases

Quick Sort:
- Best Case: O(n log n) when the pivot selection consistently divides the list in half.
- Worst Case: O(n^2) when the pivot selection results in one side being much larger than the other (usually when the list is already ordered or in reverse order).
- Average Case: O(n log n) for random lists.
- Observations: Quick sort should perform well on random lists but may slow down in performance for ordered or reverse ordered lists if the pivot selection is poor i have noticed.

Shell Sort
- Best Case: Depends on the gap sequence, but generally better than O(n^2).
- Worst Case: Depends on the gap sequence, can be as bad as O(n^2) but typically better.
- Average Case: Between O(n) and O(n^2), depending on the gap sequence.
- Observations: It should generally perform better than simple quadratic sorts (bubble, insertion) but is not as consistent as merge or quicksort; it's a bit of the middle.

Let Use Calculation Tables:

Ordered List:

Index	Short Bubble Sort	Insertion Sort	Merge Sort	Quick Sort	Shell Sort	Size
0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0	0.0	100.0
2.0	0.0	0.0	0.005205710728963216	0.001506487528483073	0.0036789576212565103	1000.0
3.0	0.0	0.0	0.010702768961588541	0.015631675720214844	0.00521699587504069	5000.0
4.0	0.0	0.0	0.02593843142191569	0.021013975143432617	0.0210570494333903	10000.0
5.0	0.004502296447753906	0.0052153269449869795	0.18203210830688477	0.17929871877034506	0.16287382443745932	50000.0

Bubble and Insertion Sort: They're practically instant up to 10,000 items, but at 50,000, they start to lag. This shows they're super quick for smaller, neat lists but not for bigger ones. Horrible crash your computer

Merge Sort and Quicksort: These two are the steady ones, handling both small and large lists with the same grace due to their O(n log n) time complexity.

Shell Sort: bit of a middle child, doing better than Bubble and Insertion, but not quite keeping up with Merge and Quicksort.

Merge Sort has a time of 0.18203210830688477 seconds.
Quick Sort has a time of 0.17929871877034506 seconds.
Quick Sort is performing slightly better than Merge Sort for this particular dataset. Both algorithms are O(n log n). A reason for this may be different pivot choices in Quick Sort

Random List:

Short Bubble Sort	Insertion Sort	Merge Sort	Quick Sort	Shell Sort	Size
0.0	0.0	0.0029687881469726562	0.005933443705240886	0.0	0.0
0.005236387252807617	0.0	0.0	0.0008477369944254557	0.0	100.0
0.03703467051188151	0.045876105626424156	0.002338409423828125	0.008883396784464518	0.005181630452473958	1000.0
1.1338259379069011	1.04459547996521	0.020841519037882488	0.0155946413675944	0.013087749481201172	5000.0
6.185863892237346	4.106320301691691	0.043665568033854164	0.012092510859171549	0.03183786074320475	10000.0
856.7915905316671	138.1670219898224	0.21570165952046713	0.06307133038838704	0.2785561879475911	50000.0

Quadratic Sorts (Bubble, Insertion): Their O(n^2) complexity leads to slow performance on large lists; only efficient for small data.

Merge Sort: O(n log n) complexity means it handles random lists well, remaining efficient regardless of size.

Quick Sort: Also O(n log n) complexity; generally efficient, but performance can vary with pivot choice.

Shell Sort: Better than quadratic sorts due to less than O(n^2) complexity on average but varies based on the gap sequence used

Merge Sort is a bit slower than Quicksort here because it uses extra memory to hold data while sorting, which can slow things down. Quick Sort, on the other hand, sorts right where the data is, saving time for a large list like the 50,000 items we've got. That's why Quicksort took in about 0.063 seconds, while Merge Sort took 0.2157 seconds.

Partially Ordered

Short Bubble Sort	Insertion Sort	Merge Sort	Quick Sort	Shell Sort	Size
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.002341111501057943	0.0	100.0
0.03283540407816569	0.021076202392578125	0.004505713780721028	0.007202466328938802	0.005216121673583984	1000.0
1.1461742719014485	1.2496728897094727	0.0210873285929362	0.02080424626668294	0.015580495198567709	5000.0
4.935295979181926	4.960421323776245	0.047200918197631836	0.04204026858011881	0.0527338981628418	10000.0
905.6285235881805	214.86098766326904	0.23677897453308105	0.2971532344818115	0.2684026559193929	50000.0

Bubble and Insertion Sort: Greatly improved efficiency; perform close to O(n), much faster than usual.

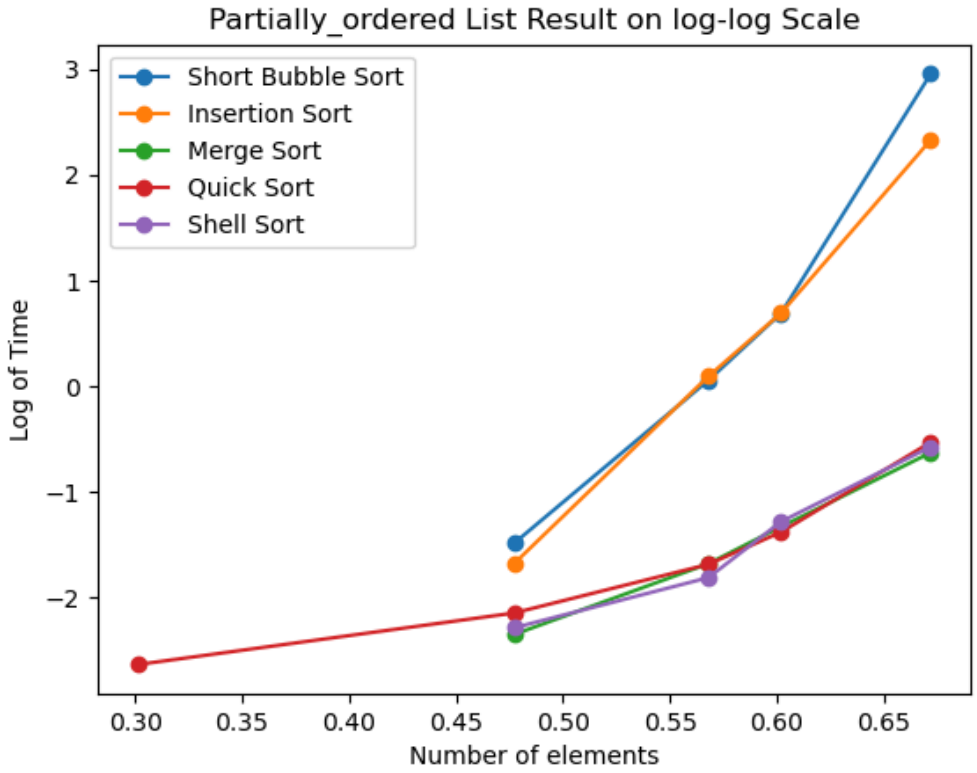
Merge and Quicksort: Slightly more efficient, but still around O(n log n); minor improvement.

Shell Sort : Benefits noticeably; fewer steps needed, faster than on unsorted lists.

Overall: Partially sorted lists boost performance of simpler sorts significantly and offer modest gains for more complex sorts.

Merge Sort outperforms Quicksort when the list size is 50,000: Merge Sort takes approximately 0.2367 seconds while Quicksort takes about 0.2971 seconds. A reason this happened on a partially ordered list is Quicksort speed can swing a bit because it depends a lot on picking a good pivot point. If the pivot's not spot-on, especially with a list that's already kinda sorted, it might do extra work. Meanwhile, Merge Sort just steadily divides and conquers, no matter what the list looks like, which can make it faster in this case.

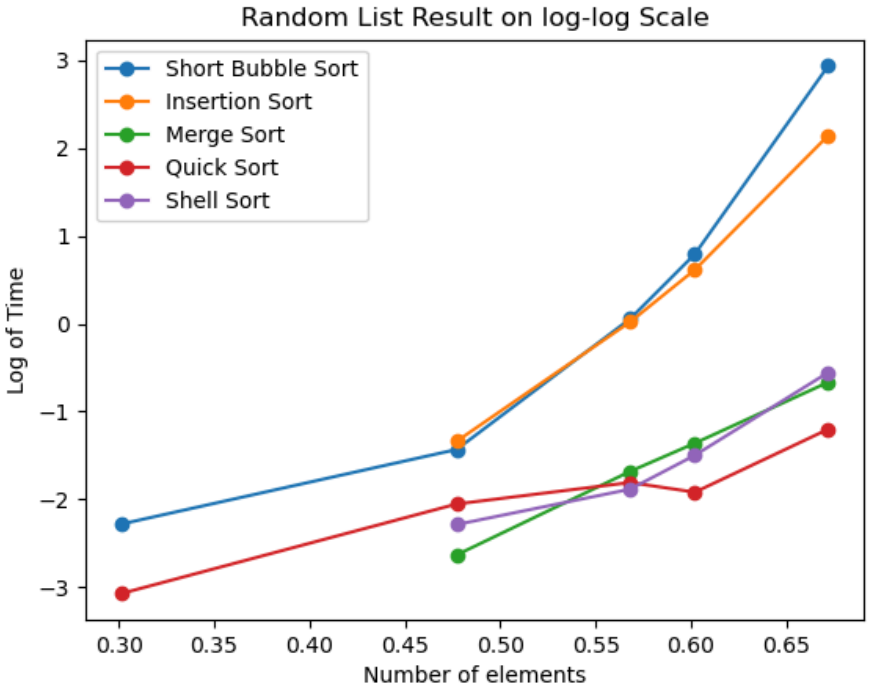
Let's Analyze the Log- Log: Used Base 10



Short Bubble Sort and Insertion Sort: Have steep curves, implying a high time complexity (likely O(n^2)) for larger lists.

Merge Sort and Quicksort: Show flatter curves, indicating a time complexity of O(n log n), which is efficient for larger lists.

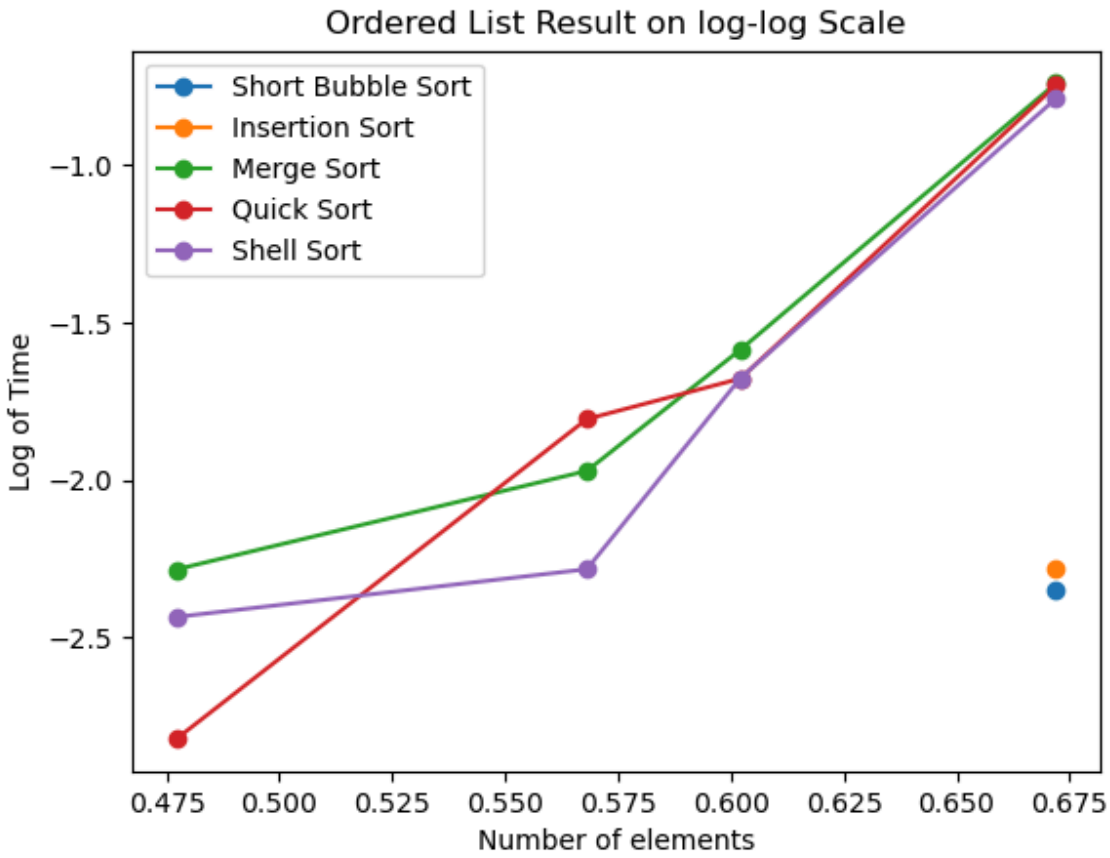
Shell Sort: Falls between the two, suggesting its time complexity is between O(n) and O(n^2)



For the random list we can see Short Bubble Sort and Insertion Sort (steep blue and orange lines): These have a steep upward trajectory, indicating a higher time complexity, likely O(n^2). Bubble sort out performed Insertion sort here

We can see the green line (Quick Sort) to be below the red line (Merge Sort) towards the right end of the graph, suggesting that Quicksort was more efficient than Merge Sort for this random list. This shows that Quicksort can often be faster than Merge Sort even thou they both have the time complexity of O(n log n).

The line for Shell Sort rises faster than Merge and Quick Sort but not as sharply as Bubble and Insertion Sort, implying its time complexity is between O(n) and O(n^2) not O(n log n)



On an ordered list, Short Bubble Sort and Insertion Sort demonstrate a significant increase in efficiency, approaching O(n) complexity, we can see by their lower positioning on the log-log scale. This might be because its an ordered pair so its easier for the sort algorithm to be more effective than its ussually showing 0(n^2)

The steep curves for Bubble and Insertion Sort hint that their sorting times shoot up fast as the list gets bigger. This matches their usual O(n^2) time complexity when they're dealing with random lists.

Shell Sort is the slowest on the graph, much slower than Bubble or Insertion Sort.

Overall Short Bubble Sort and Insertion Sort is the fastest, and Shell Sort is the slowest