observed discrepancies between theoretical analysis and practical performance

Provided graphs at the end of this document reveal interesting observations in the comparison between Quick Sort and Merge Sort regarding both execution time and memory usage. Let's break down the theoretical expectations and contrast them with the practical data from the graphs.

1. Execution Time:

- **Theoretical Analysis:**
  - Quick Sort has an average time complexity of $O(n \log n)$, but in the worst-case scenario (when the pivot selection is poor), it can degrade to $O(n^2)$.
  - Merge Sort, on the other hand, consistently runs in $O(n \log n)$ regardless of the input.
- **Practical Performance:**
  - On sorted and reverse-sorted data, Merge Sort's performance seems to be consistently slower compared to Quick Sort. This is somewhat counterintuitive, as Quick Sort is typically expected to perform poorly on sorted data due to its pivot selection strategy.
  - One possible explanation is that Quick Sort's implementation in our tests uses a strategy like the median-of-three pivot, which avoids the worst-case scenario on sorted or reverse-sorted data. In contrast, Merge Sort's consistent overhead due to copying data between arrays or memory allocation can make it slower in practice, even if its theoretical worst-case scenario is better.

2. Memory Usage:

- **Theoretical Analysis:**
  - Quick Sort is an in-place sorting algorithm, meaning it uses $O(n \log n)$ *addi*tional space due to recursive calls, but otherwise it doesn't require significant extra memory.
  - Merge Sort, by contrast, requires *O(n)*extra space, as it needs to allocate additional arrays to merge sorted subarrays.
- **Practical Performance:**
  - The graphs for memory usage suggest a different story. Quick Sort's memory usage seems significantly higher than Merge Sort, particularly on random and reverse-sorted data. This may indicate that the Quick Sort implementation is not purely in-place and may be using additional auxiliary data structures or has inefficient recursion, increasing its memory consumption.

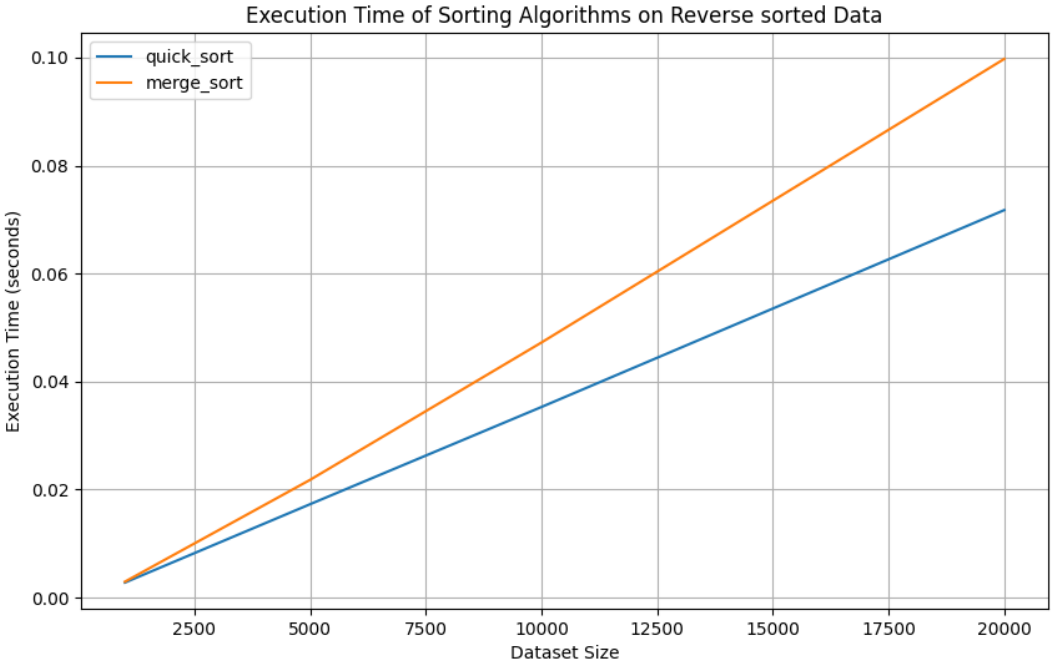
    - Merge Sort, while requiring more memory in theory due to its need for extra arrays, shows lower memory usage. This could be due to a highly optimized implementation that minimizes overhead.

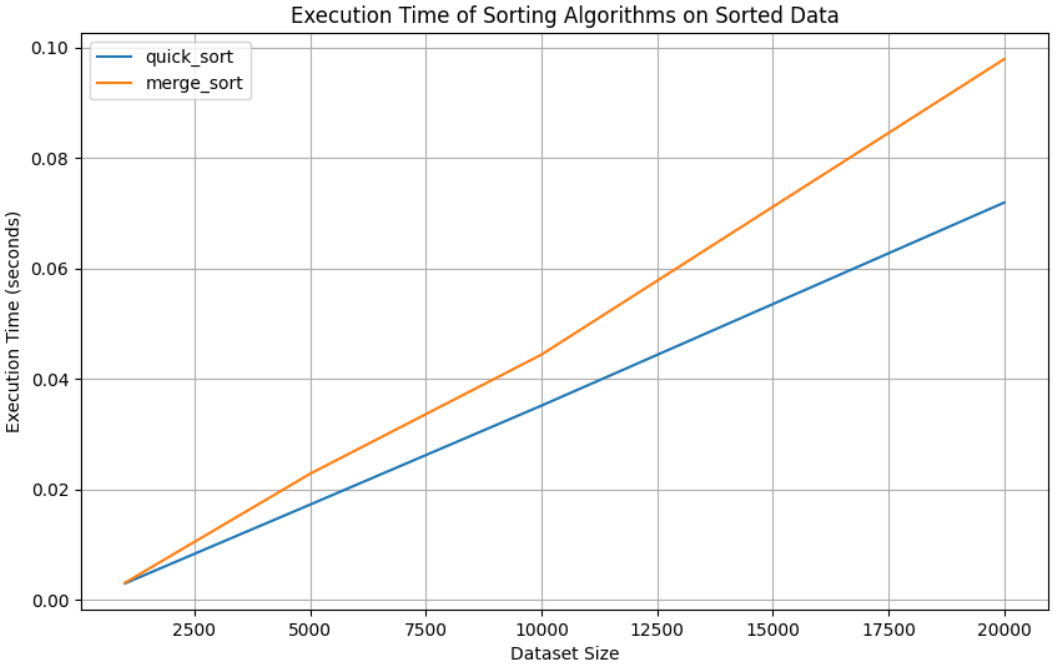
3. Random Data:

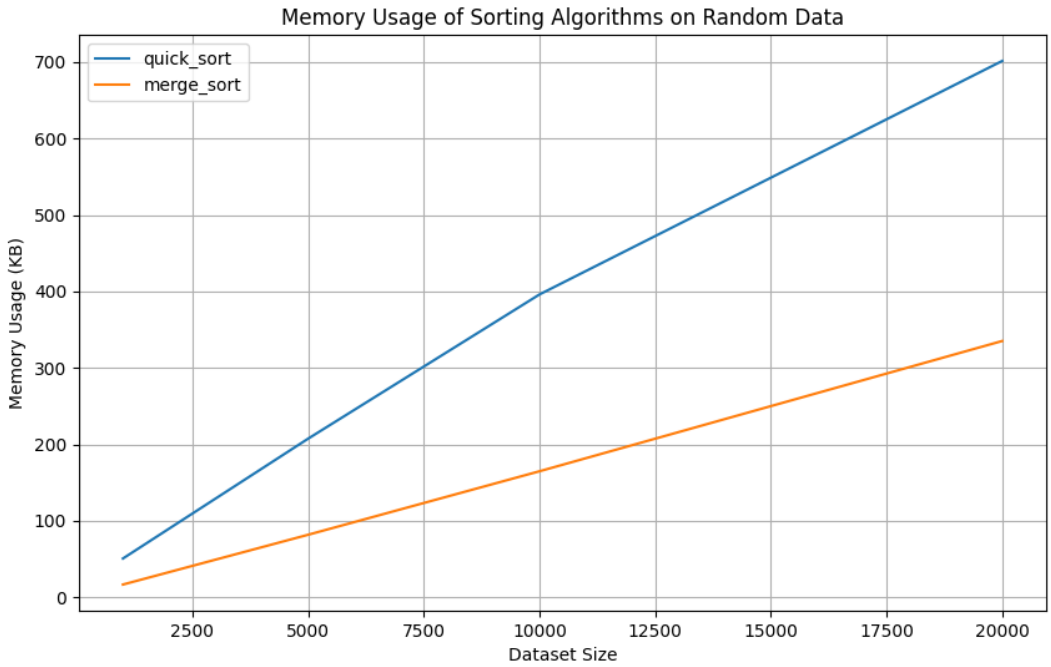
- On random data, Quick Sort appears to have a faster execution time than Merge Sort, aligning with its expected average-case performance. However, the significant discrepancy in memory usage, where Quick Sort uses considerably more memory than expected, suggests that memory efficiency in Quick Sort might be a concern in our specific implementation.

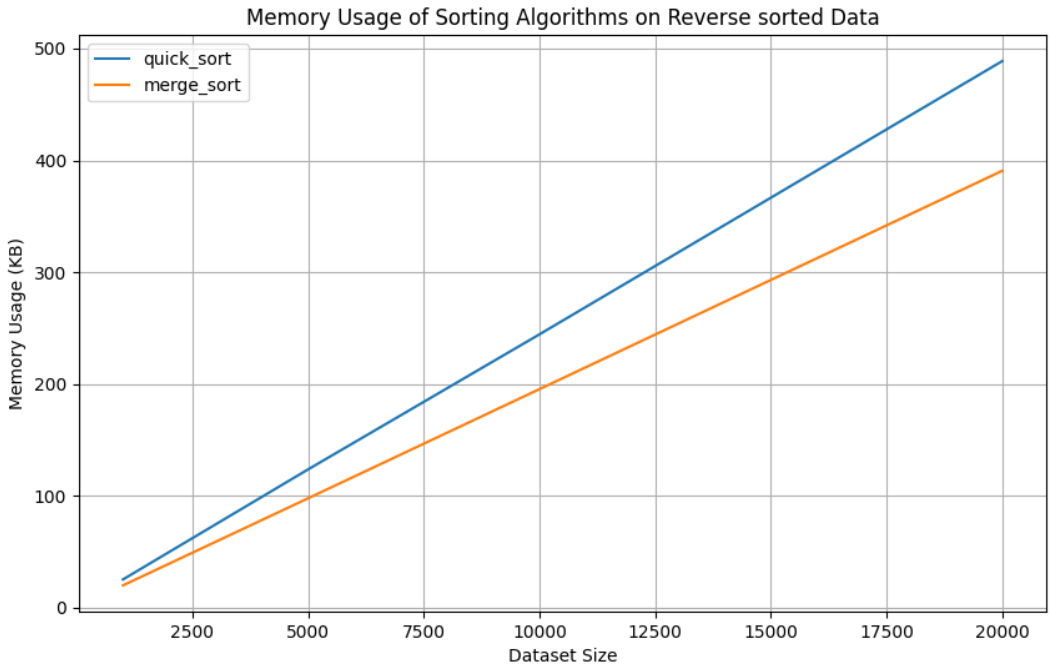
Conclusion:

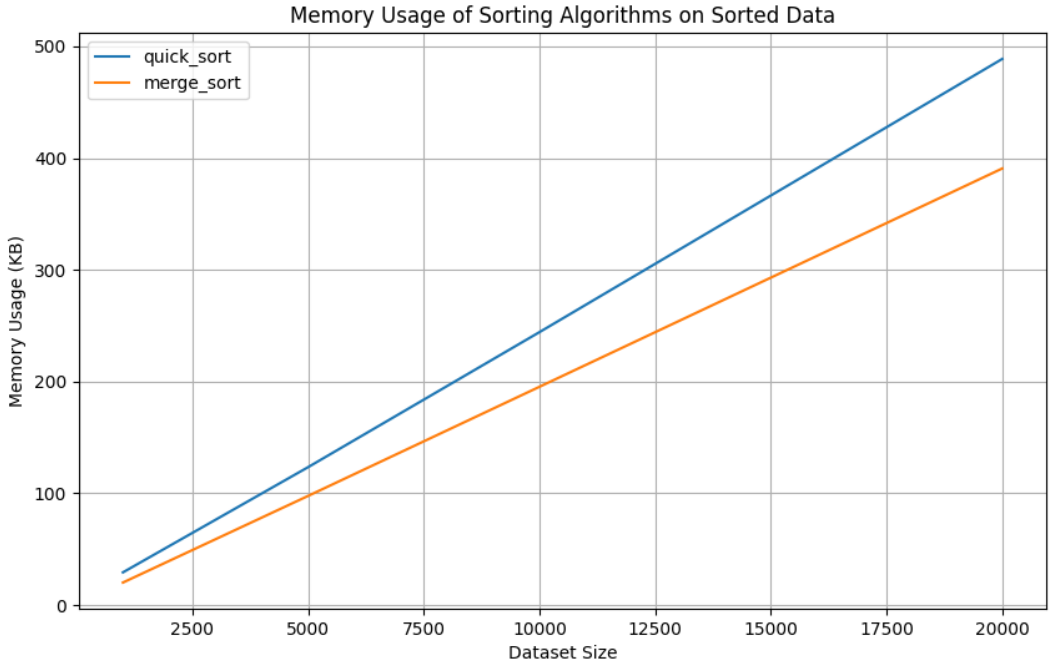
- In practice, Quick Sort outperforms Merge Sort in terms of speed across the board in our results, but with higher memory usage than anticipated. Theoretically, Quick Sort's time complexity should deteriorate with sorted data, but our implementation likely avoids this through a better pivot strategy.
- The memory usage results show that Merge Sort is more memory-efficient than expected. These discrepancies suggest that optimizations in implementations play a significant role in practical performance and can deviate from theoretical predictions.

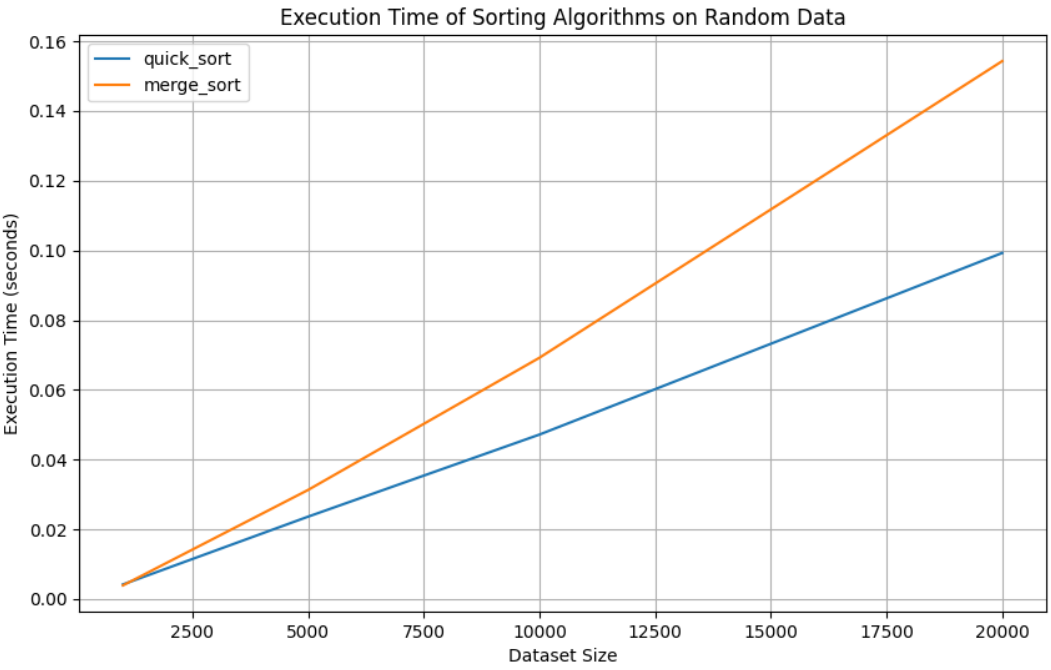

Execution Time of Sorting Algorithms on Reverse sorted Data
quick_sort
merge_sort
Execution Time (seconds)
0.00
0.02
0.04
0.06
0.08
0.10
2500
5000
7500
10000
12500
15000
17500
20000
Dataset Size


Execution Time of Sorting Algorithms on Sorted Data
quick_sort
merge_sort
Execution Time (seconds)
0.00
0.02
0.04
0.06
0.08
0.10
2500
5000
7500
10000
12500
15000
17500
20000
Dataset Size


Memory Usage of Sorting Algorithms on Random Data
quick_sort
merge_sort
Memory Usage (KB)
0
100
200
300
400
500
600
700
2500
5000
7500
10000
12500
15000
17500
20000
Dataset Size


Memory Usage of Sorting Algorithms on Reverse sorted Data
quick_sort
merge_sort
Memory Usage (KB)
0
100
200
300
400
500
2500
5000
7500
10000
12500
15000
17500
20000
Dataset Size


Memory Usage of Sorting Algorithms on Sorted Data
quick_sort
merge_sort
Memory Usage (KB)
0
100
200
300
400
500
2500
5000
7500
10000
12500
15000
17500
20000
Dataset Size


Execution Time of Sorting Algorithms on Random Data
quick_sort
merge_sort
Execution Time (seconds)
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
2500
5000
7500
10000
12500
15000
17500
20000
Dataset Size