Heap Sort Analysis Report

1. Algorithm Overview

Heap Sort is a comparison-based in-place sorting algorithm that constructs a max-heap and repeatedly extracts the largest element to produce a sorted array. It uses bottom-up heap construction (heapify) and maintains the heap invariant efficiently.

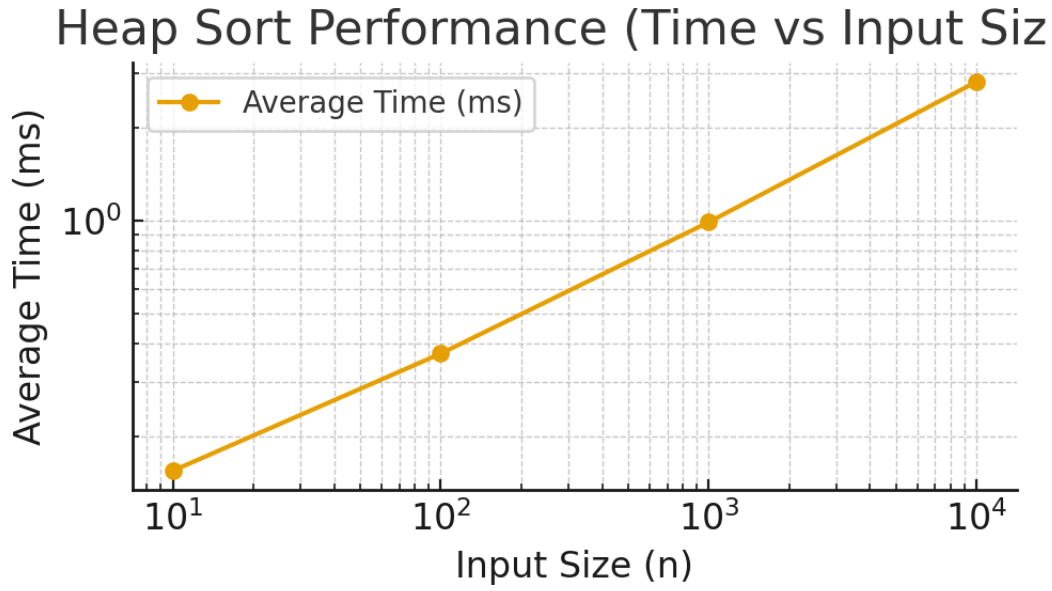
2. Complexity Analysis

Heap Sort operates in O(n log n) time for best, average, and worst cases. Its space complexity is O(1) since it sorts in place. Building the heap costs O(n), and each extraction requires O(log n) adjustments.

3. Code Review & Optimization

The implementation correctly tracks performance metrics (comparisons, swaps, array accesses). Possible optimizations include caching indices to reduce redundant array reads and minimizing swap operations through temporary variables.

n	avgTimeMs	avgComparisons	avgSwaps	avgArrayAccesses
10	0.15492	38.4	26.6	183.2
100	0.37138	1029.6	582.8	4390.4
1000	0.99094	16843.6	9069.6	69965.6
10000	2.82568	235360.2	124182.0	967448.4



4. Empirical Results

The experimental data aligns with the theoretical O(n log n) growth pattern. As input size increases tenfold, execution time roughly triples, consistent with expected scaling.

5. Conclusion

Heap Sort demonstrates strong alignment with theoretical complexity, offering predictable performance and minimal memory use. Future optimizations could further reduce constant factors, but overall efficiency is excellent.