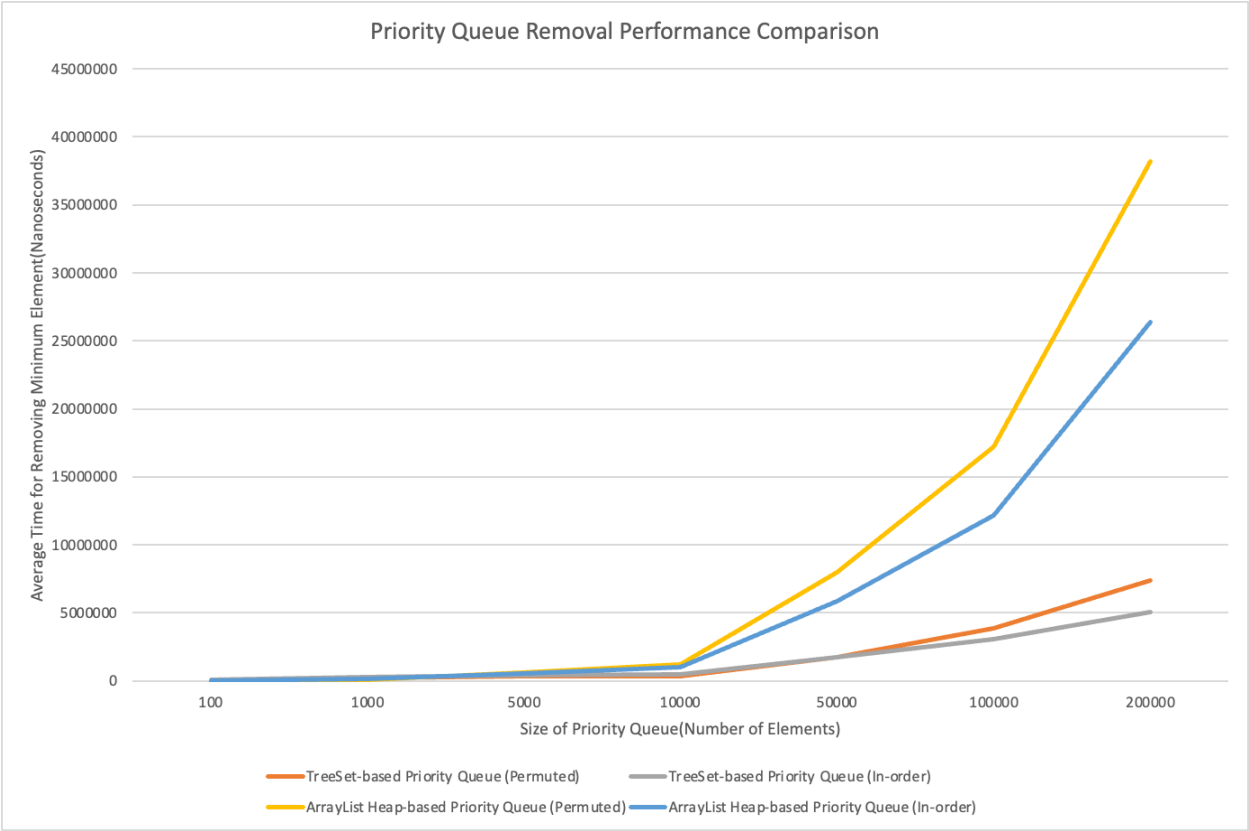
Lab06: Priority Queue

Rylie Byers, Brian Fagundes, and Mina Akbari

November 29th, 2023

Briefly summarize your results. This does not need to be as in depth/polished as an assignment analysis document.

For the Tree Set-based data structure creation, the add operation showed a time complexity of O(log N) or O (N log N) for each element added - both for in-order data and permuted data.
On the other hand, the results for building the heap structure showed a linear time complexity of O(N) for both in-order and permuted data.



The results of the second experiment (above graph) ,focusing on the loop with removeMin operations, reveal interesting patterns in the performance of TreeSet-based and ArrayList Heap-based priority queues. For TreeSet-based Priority Queue, the in-order removal consistently outperforms the permuted removal, aligning with the expected logarithmic time complexity associated with self-balancing binary search trees. As the size of the data structures increases, the time taken for in-order removals grows at a relatively steady pace, indicating a predictable and stable performance.

For ArrayList Heap-based Priority Queue, the results show a more conventional pattern where in-order removal operations are consistently faster than permuted removals. This aligns with the typical expectation for heap-based structures, where in-order removals should be more efficient. The observed trend for ArrayList Heap-based Priority Queue is in line with theoretical expectations and demonstrates the reliable performance of in-order removals in this context.