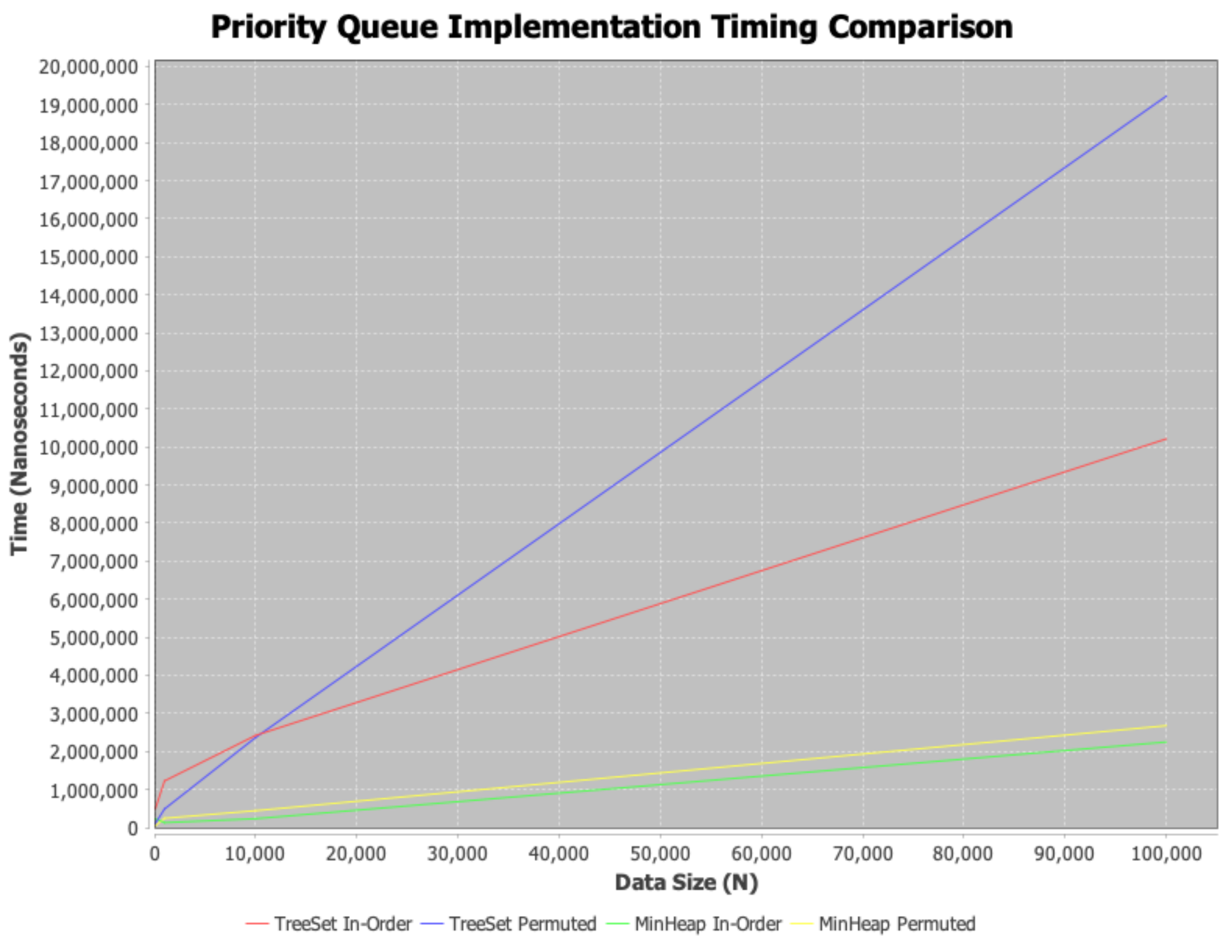
My partner is Yijun Zhan.

First plot



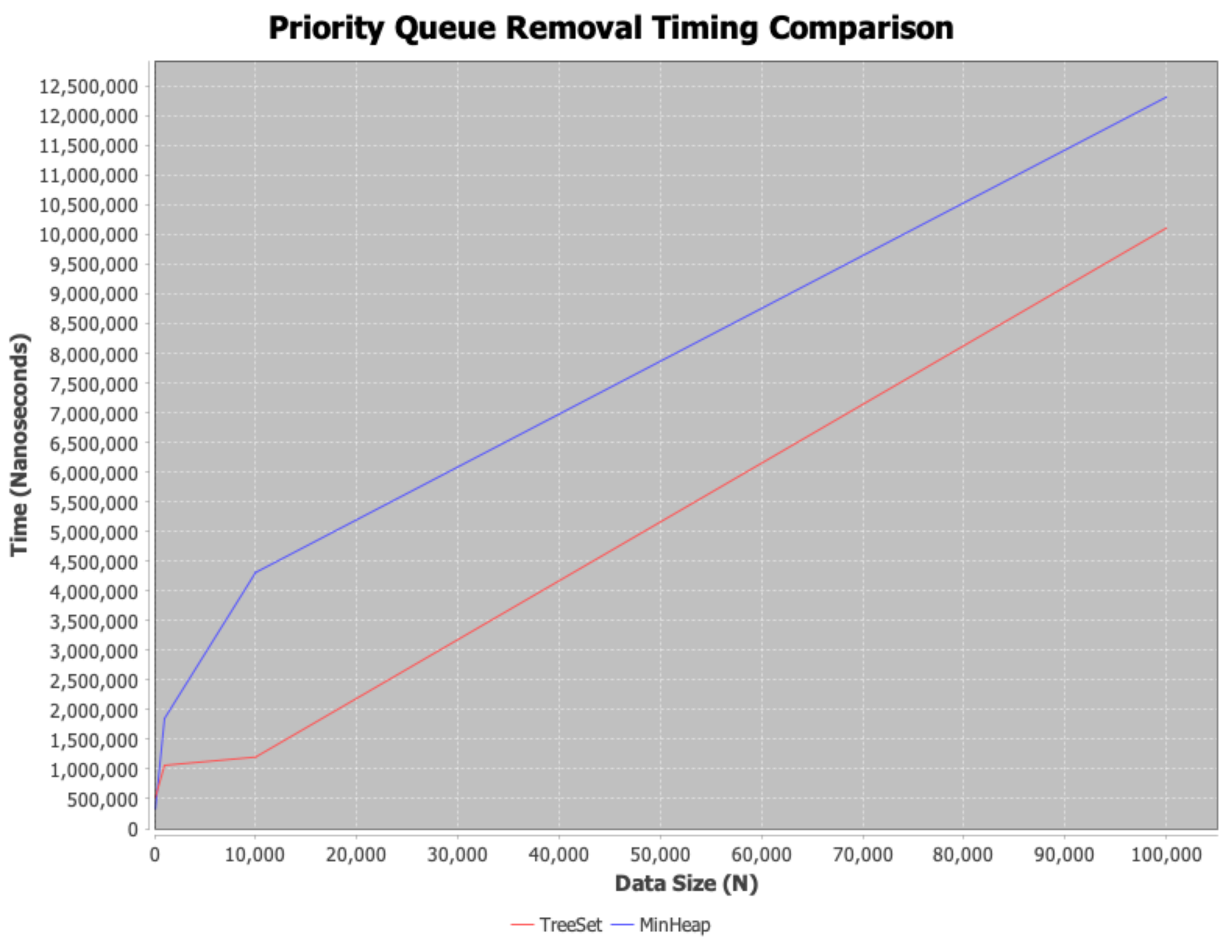
The chart shows the performance comparison of two priority queue implementations: a TreeSet-based queue and a MinHeap-based queue, each tested with in-order and permuted data sets.

- TreeSet In-Order: The performance appears to degrade linearly as the data size increases. This is expected as the TreeSet maintains order, so adding in-order data does not require rebalancing.
- TreeSet Permuted: With permuted data, the performance impact is greater than in-order, which suggests that the cost of rebalancing the tree for unordered data is significant.

- MinHeap In-Order: The MinHeap implementation shows excellent performance with in-order data, displaying nearly constant time across the tested range. This suggests that the MinHeap structure handles sequential data efficiently when constructed en masse.
- MinHeap Permuted: For permuted data, the MinHeap also performs well, although there is a slight performance penalty compared to in-order data. The impact of adding elements that require the heap to reorganize is minimal.

In summary, the MinHeap-based priority queue consistently outperforms the TreeSet-based implementation for both in-order and permuted datasets across all sizes tested. The TreeSet struggles with permuted data due to the overhead of tree rebalancing. The MinHeap's performance is robust with varying data orders, making it a better choice for a priority queue when the data is not guaranteed to be in any particular order.

Second plot



From the graph, we can observe that:

- The TreeSet (red line) consistently takes less time for removal operations than the MinHeap (blue line) across all data sizes presented.
- The time taken for both TreeSet and MinHeap increases linearly as the data size increases, but the rate of increase in time is less for the TreeSet. This indicates a more efficient removal operation for the TreeSet compared to the MinHeap.
- The performance gap between the two data structures widens as the data size grows, with the TreeSet maintaining a more efficient removal time than the MinHeap.

In summary, for the given data sizes, the TreeSet is more efficient for removal operations in a priority queue compared to the MinHeap, and this efficiency advantage becomes more noticeable as the number of elements increases.