# Cross-Review Summary: Min-Heap vs Max-Heap

This document presents a comparative analysis between the Min-Heap and Max-Heap implementations. Both data structures are based on binary heap principles but differ in their ordering properties and operations.

## 1. Overview

• The Min-Heap maintains the smallest element at the root. It supports decrease-key and merge operations.  
• The Max-Heap maintains the largest element at the root. It supports increase-key and extract-max operations.

## 2. Performance Comparison

The following summarizes the performance metrics measured for both implementations:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Heap Type | Input Size (n) | Comparisons | Swaps | Elapsed Time (ms) |
| Min-Heap | 100 | 680 | 517 | 1 |
| Min-Heap | 1000 | 12009 | 8603 | 16 |
| Min-Heap | 10000 | 169812 | 119582 | 56 |
| Max-Heap | 100 | 700 | 333 | 0.55 |
| Max-Heap | 1000 | 16215 | 11976 | 2.75 |
| Max-Heap | 10000 | 229749 | 170440 | 10.27 |

## 3. Conclusion

The Max-Heap generally performs more comparisons and swaps than the Min-Heap for similar input sizes. Its elapsed time also increases faster with larger inputs. The Min-Heap shows more consistent scaling, possibly due to simpler key comparisons in the decrease-key operation. Both structures exhibit the expected O(log n) performance per operation.

Overall, the Min-Heap implementation demonstrates slightly better efficiency for large datasets, while the Max-Heap shows higher costs due to repeated reordering during extract-max operations.