# PARALLEL SORTING REPORT:

## SETUP:

***Array Sizes:*** Three array sizes were tested: 2,000,000; 3,000,000; and 4,000,000 elements.

***Thread Counts:*** Sorting was performed using varying thread counts, specifically powers of two: 16, 32, 64, 128, and 256.

***Cutoff Values Scheme:*** A range of cutoff values were chosen to decide when to revert to sequential sorting. The values ranged from 10,000 to 500,000.

## OBSERVATIONS:

***Impact of Threads:*** The performance generally improved with more threads, but only to a certain threshold. Beyond this threshold, the increased overhead from thread management outweighed the performance benefits.

***Cutoff Strategy:*** Lower cutoff values tended to yield better performance in parallel sorting. However, excessively low cutoffs might cause excessive overhead due to the frequent switching between parallel and sequential sorting.

***Array Size***: Larger arrays showed more significant performance improvements with parallel sorting. This is likely because the overhead of parallel processing is more easily amortized over larger data sets.

## CONCLUSIONS:

1. The parallel sorting algorithm is most effective with a carefully chosen cutoff value and a moderate number of threads. An optimal cutoff value depends on the array size and the number of available threads.
2. There is a diminishing return on performance gains with an increase in the number of threads, especially for smaller array sizes.
3. The optimal configuration for parallel sorting involves balancing the granularity of the tasks (determined by the cutoff) with the computational overhead of managing multiple threads.