**Nimish Sharma (001598648)**

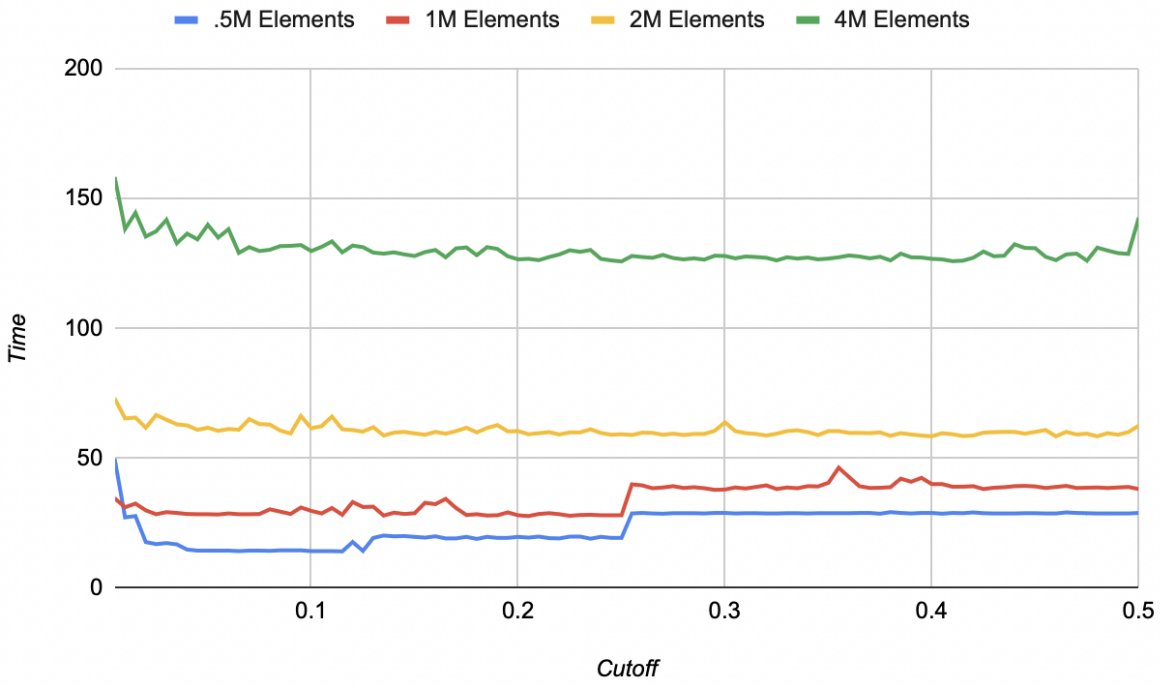
# Program Structures & Algorithms
# Fall 2021
# Assignment No. 5

- **Tasks performed in the assignment:**
  - Ran parallel sort on common parallelism pool with different array sizes, and benchmarked execution time corresponding to cutoff's.
  - Took 4 different arrays, and for each array of substantial size, ran parallel sort on multiple threads (1, 2, 4, 8, 16) and benchmarked the execution time corresponding to the cutoff's.
  - From the observations of the previous cases, picked a combination for most optimal cutoff, and the number of threads, and ran parallel sort again in order to benchmark the execution time.

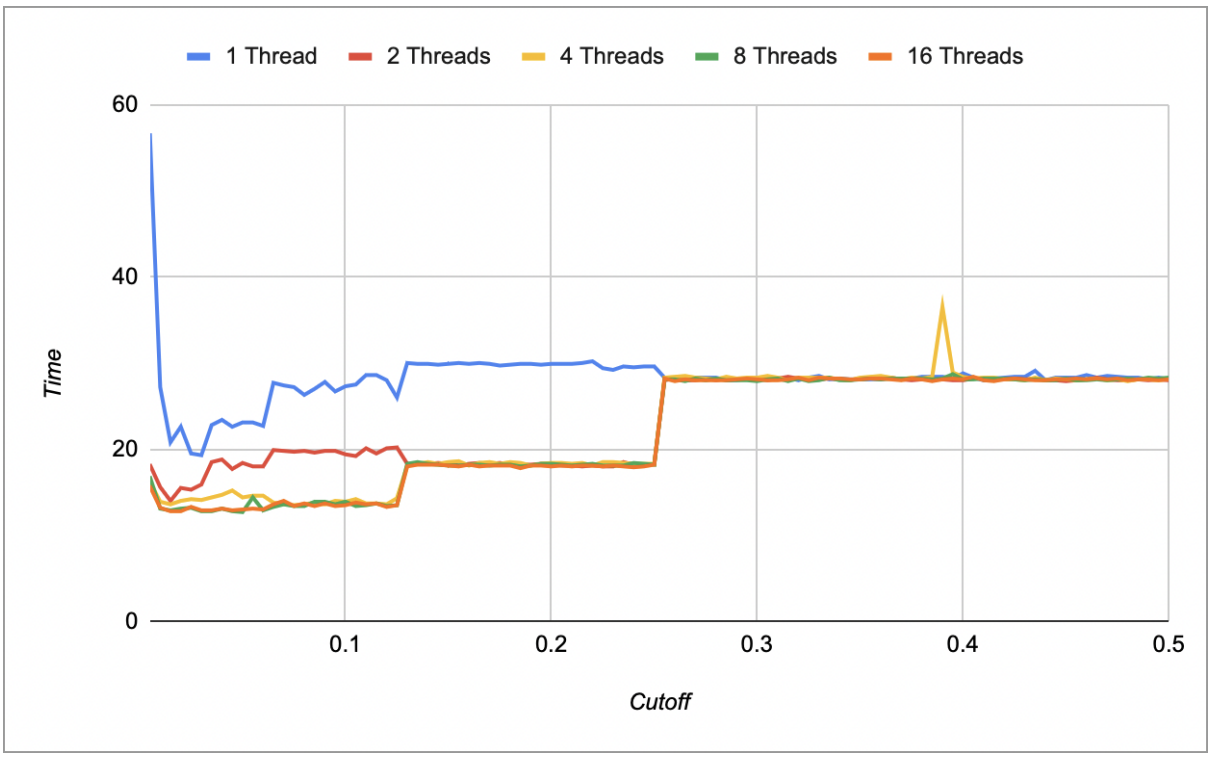
- **Conclusion:**
  - In case when no threads were manipulated, the performance across all array sizes (0.5M, 1M, 2M, 4M) was almost identical. However, with larger array sizes (> 1M), the performance was most consistent across all cutoffs.
  - When executing on multiple threads, less than 4 number of threads led to suboptimal performance. While executing on 16 threads had no advantage over executing on 8 threads (until the cutoff was not optimized). Hence, in general parallel sort on 8 threads gave the best performance across all cutoffs.
  - A cutoff between (20000...ArraySize/4), gave the most optimal benchmarks (shown in graphs).
  - So, in the final experiment, the cutoff was set to ArraySize/4 and it was observed that, if the cutoff is set such that each of the thread can have equal size (in power of 2) subarrays to work on, then we can have substantial gains in performance with a higher degree of parallelism.

- **Evidence to support conclusion (Graphical Representation):**

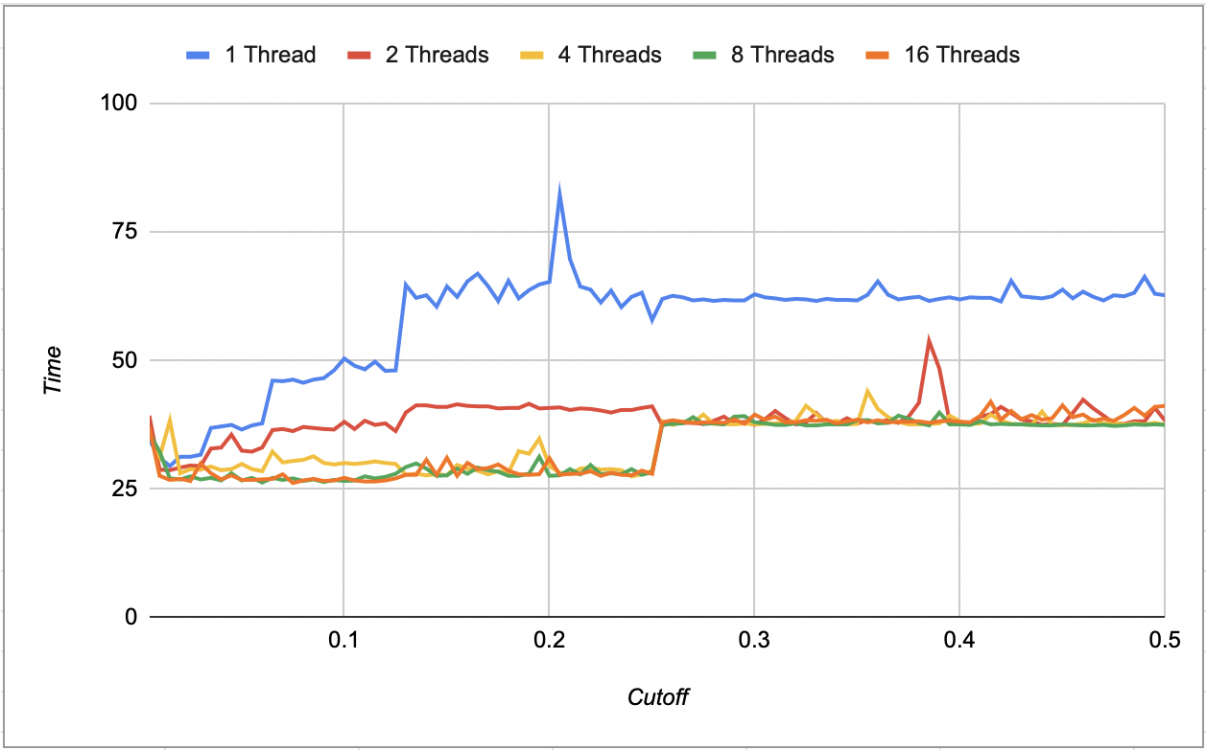
**Case 1: Experiment on Common Pool Parallelism.**



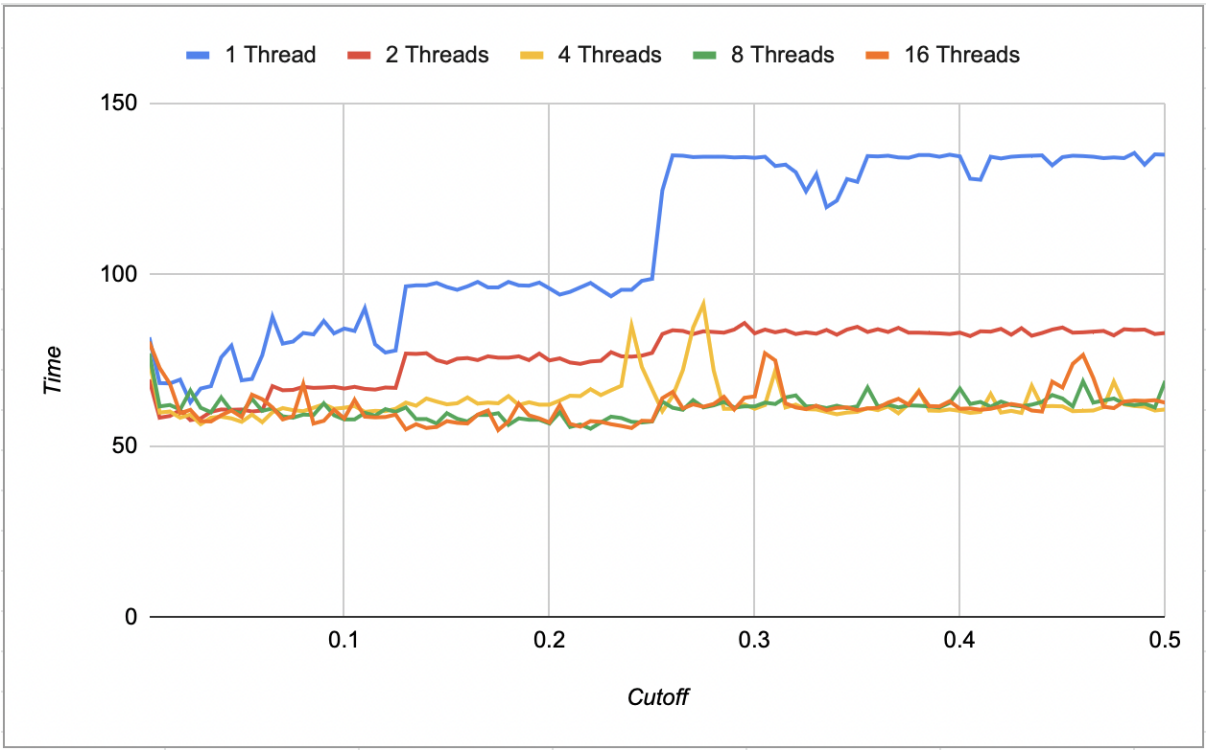

**Case 2.1: Experiment on 0.5M elements on multiple threads.**



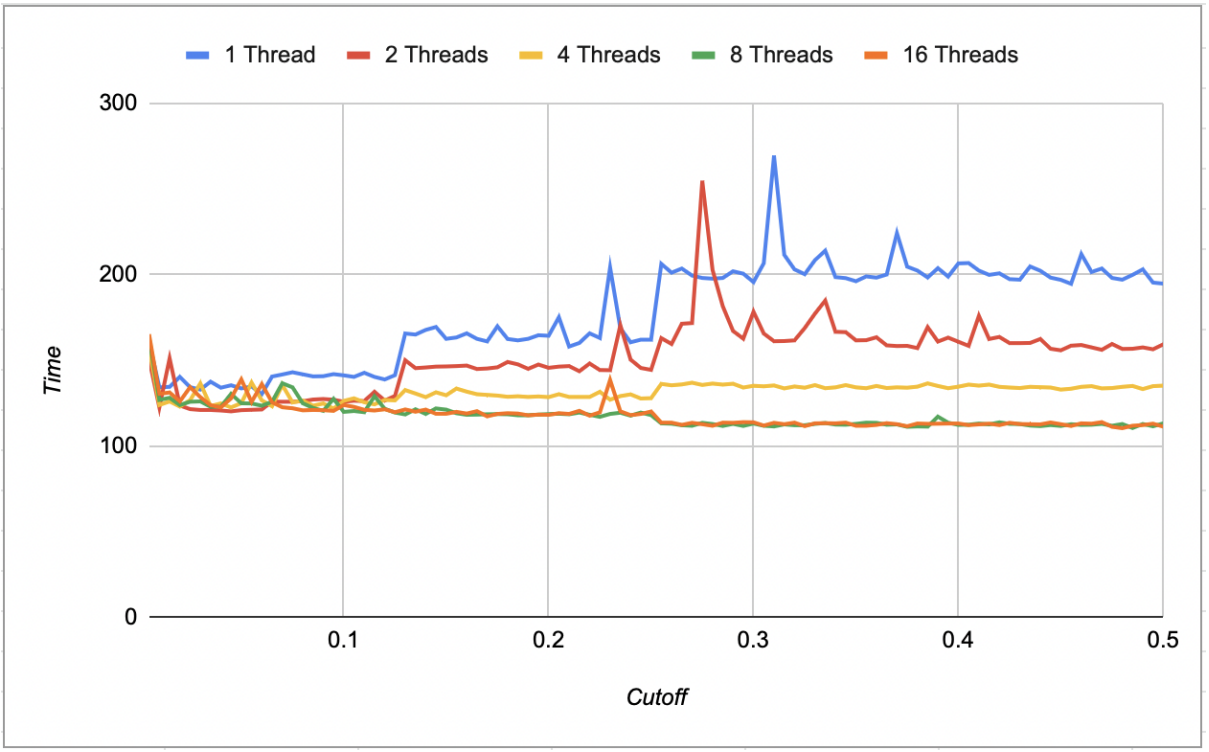

**Case 2.2: Experiment on 1M elements on multiple threads.**



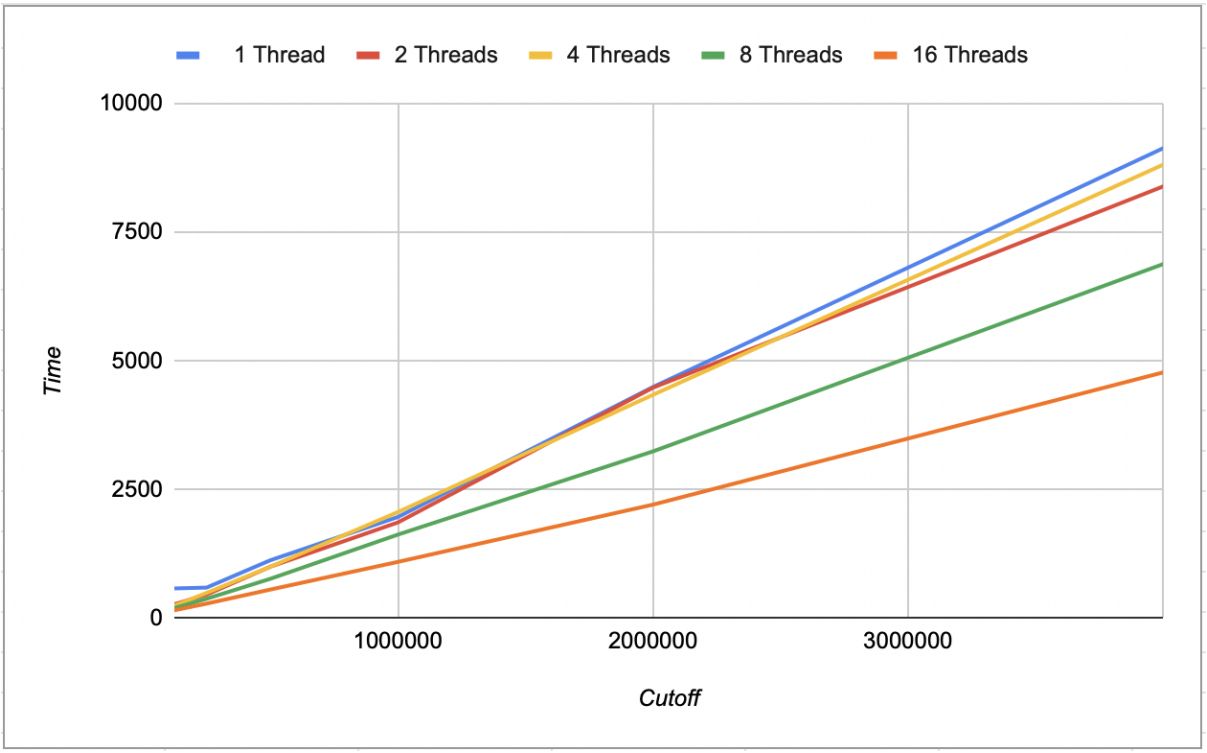

**Case 2.3: Experiment on 2M elements on multiple threads.**




**Case 2.4: Experiment on 4M elements on multiple threads.**




**Case 3: Experiment on 0.5M, 1M, 2M, 4M elements on multiple threads with cutoff ArraySize/4.**


1 Thread
2 Threads
4 Threads
8 Threads
16 Threads
10000
7500
5000
2500
0
Time
1000000
2000000
3000000
Cutoff