Program Structures and Algorithms

Spring 2023(SEC 3)

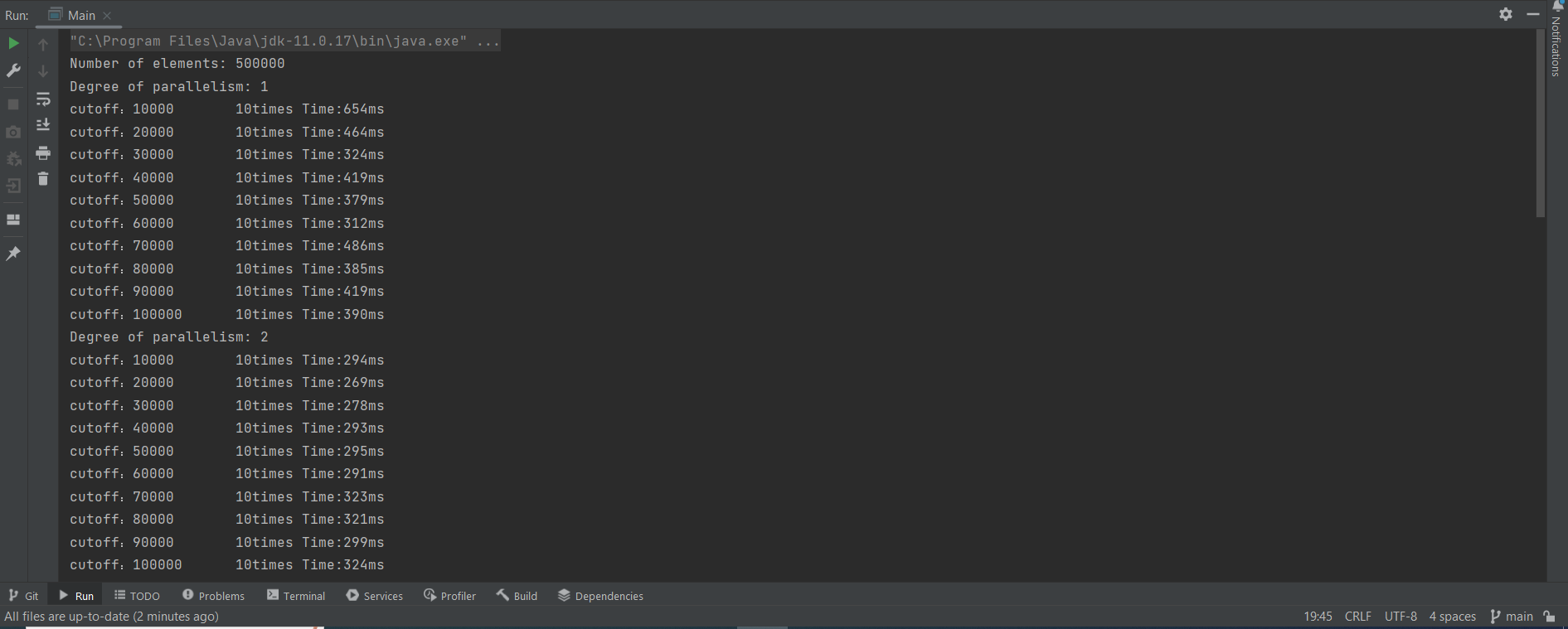
NAME: Subbu Manickam

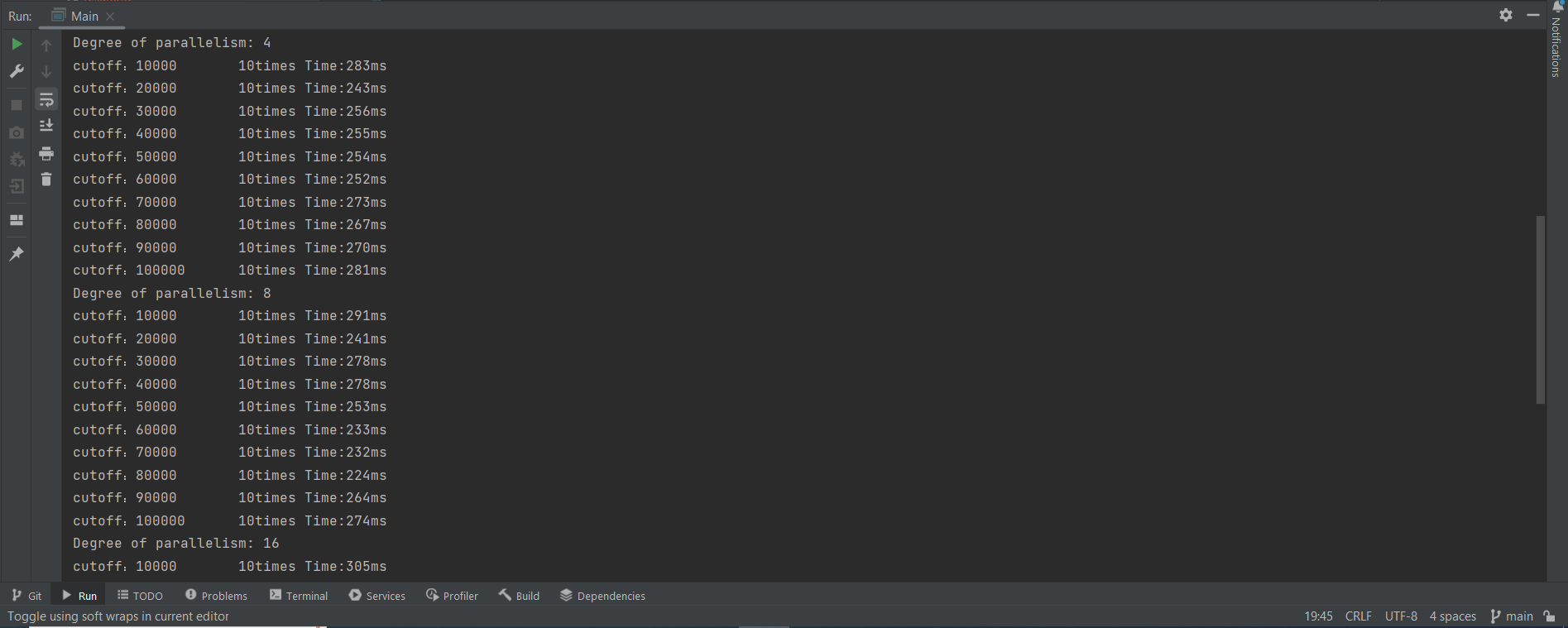
NUID: 002768764

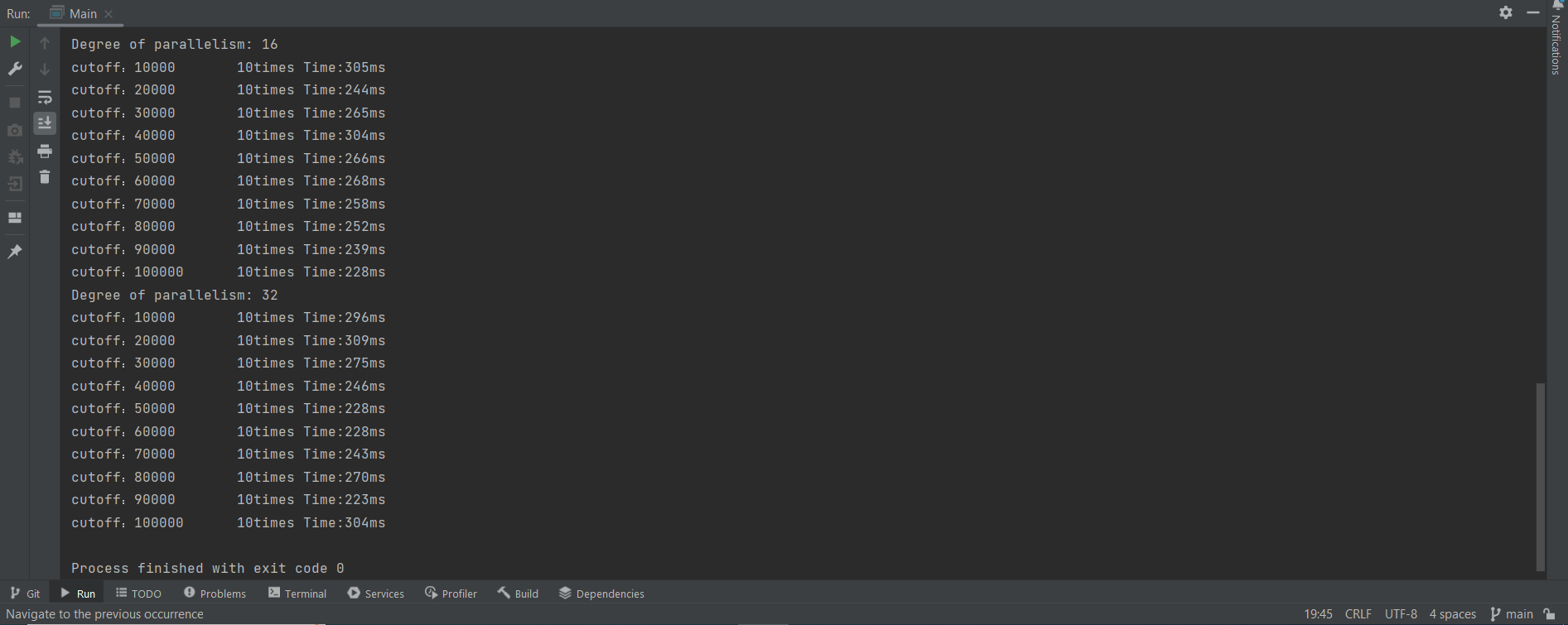
**Task:** Analysis of Parallel Sorting and its efficiency based on thread count and cut-off value

**Relationship Conclusion:** Parallelism increases the efficiency as long as the number of threads do not affect the allocation of resources. If the system cannot handle the extra threads, then parallelism acts destructively thereby increasing the time rather than decreasing it.

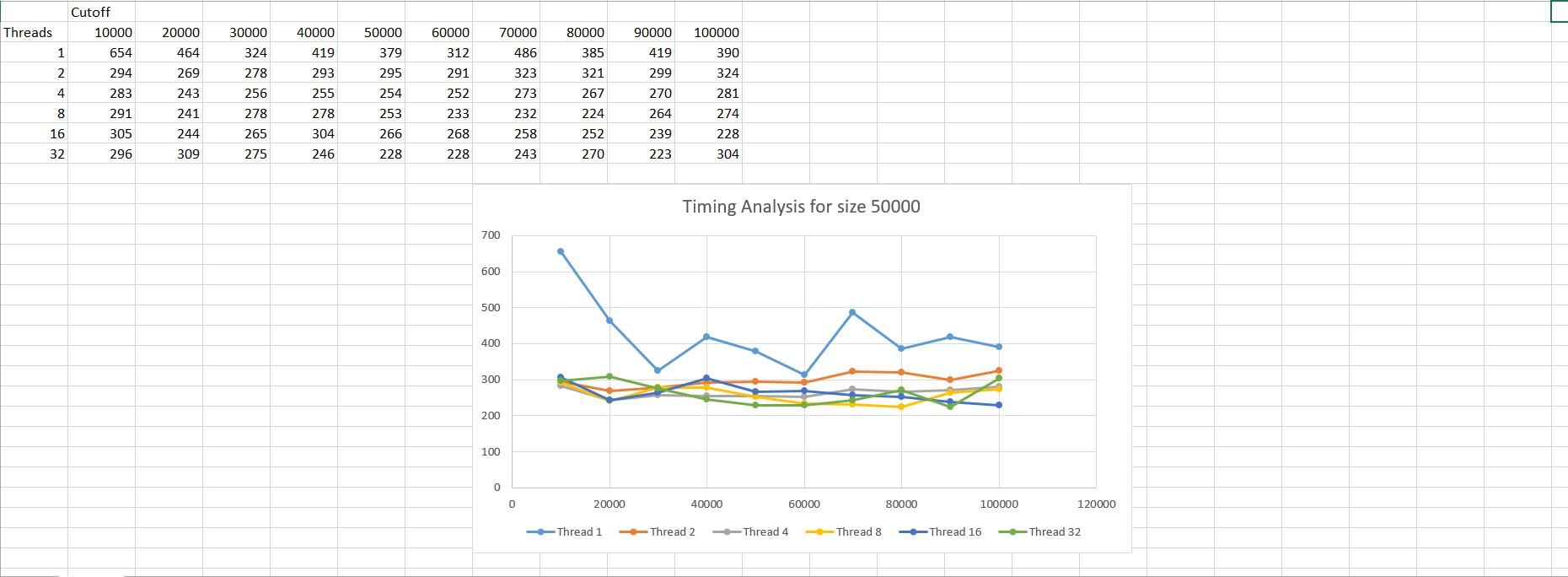
**Evidence to support that conclusion:** Based on the below screenshots of the execution time of different degrees of parallelism (threads), we can conclude that beyond a thread count of 2 the execution time becomes saturated and there isn’t much of an improvement in performance or efficiency. Moreover as we increase the thread count to 16 and 32, we can visualise that the execution time is actually higher than as compared to thread counts of 2 and 4.







**Graphical Representation:**

****

**Working of Parallel Sort and Analysis:** Based on our analysis by varying the number of threads and cut-off values for finding the timing analysis due to the effect of parallelism on merge sort, we can conclude that parallelism doesn’t always increase the execution time and will be productive as long as the number of threads do not compete for the available resources. From the analysis above, the optimal number of threads for my system configuration seem to be 2 threads, beyond which there isn’t any significant increase in the performance of the sorting algorithm. Similarly, the optimal cut-off value seems to fall around the 30% mark of the entire array size.