Program Structures & Algorithms
Spring 2022

Assignment No - 4 (Parallel Sorting)

Name: Naina Rajan
NUID: 002922398

Task

Please see the presentation on *Assignment on Parallel Sorting* under the *Exams. etc.* module.
Your task is to implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. You will consider two different schemes for deciding whether to sort in parallel.

1. A cutoff (defaults to, say, 1000) which you will update according to the first argument in the command line when running. It's your job to experiment and come up with a good value for this cutoff. If there are fewer elements to sort than the cutoff, then you should use the system sort instead.
2. Recursion depth or the number of available threads. Using this determination, you might decide on an ideal number (*t*) of separate threads (stick to powers of 2) and arrange for that number of partitions to be parallelized (by preventing recursion after the depth of *lg t* is reached).
3. An appropriate combination of these.

There is the *Main* class and the *ParSort* class in the *sort.par* package of the INFO6205 repository. The *Main* class can be used as is but the *ParSort* class needs to be implemented where you see "TODO..." [it turns out that these TODOs are already implemented].
Unless you have a good reason not to, you should just go along with the Java8-style future implementations provided for you in the class repository.
You must prepare a report that shows the results of your experiments and draws a conclusion (or more) about the efficacy of this method of the parallelizing sort. Your experiments should involve sorting arrays of sufficient size for the parallel sort to make a difference. You should run with many different array sizes (they must be sufficiently large to make parallel sorting worthwhile, obviously) and different cutoff schemes.

For varying the number of threads available, you might want to consult the following resources:

- https://www.callicoder.com/java-8-completablefuture-tutorial/#a-note-about-executor-and-thread-pool
- https://stackoverflow.com/questions/36569775/how-to-set-forkjoinpool-with-the-desired-number-of-worker-threads-in-completable

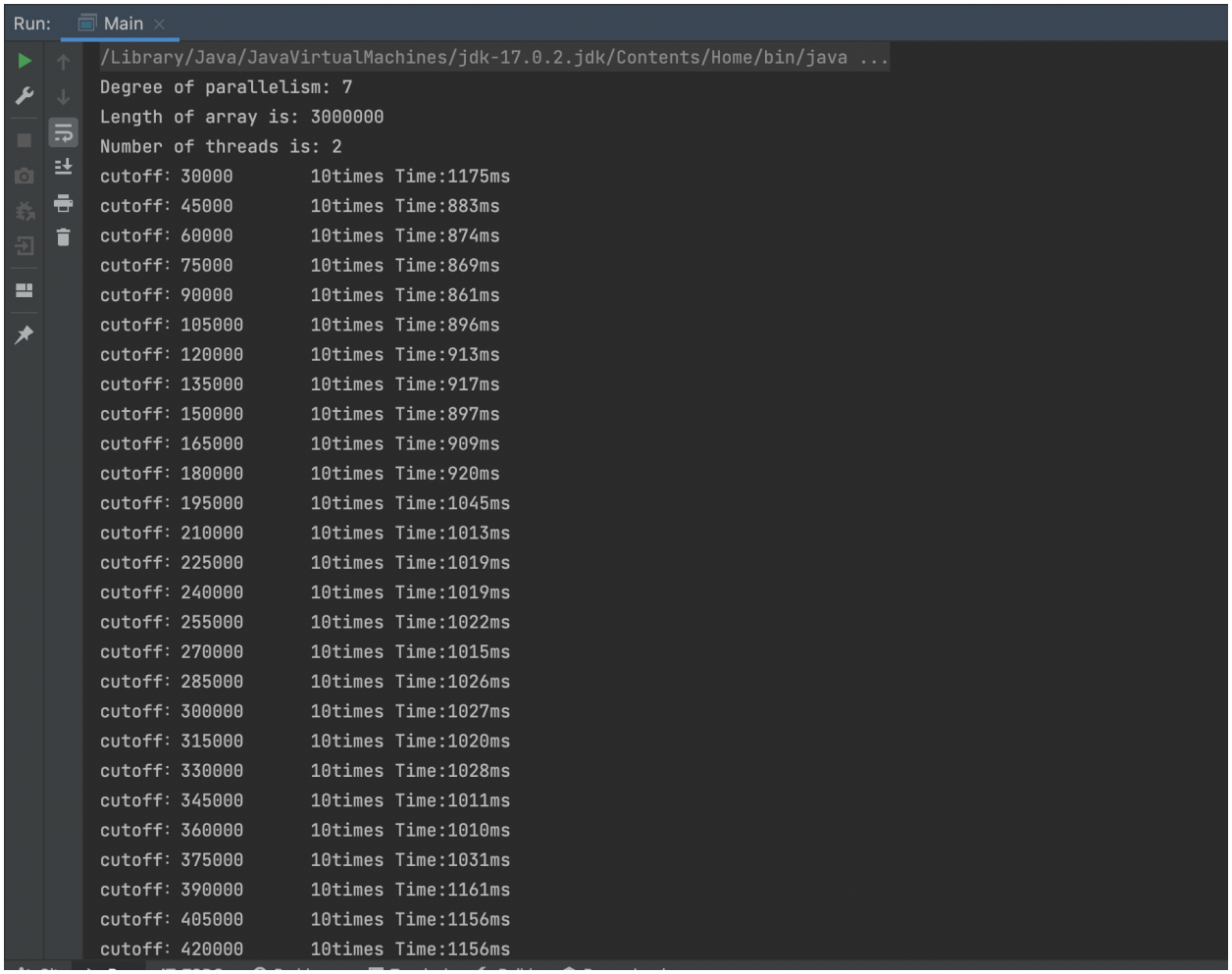
Output Screenshot

Degree of parallelism: 7

Array Length: 3000000

Thread count: 2

Number of core in test machine: 8

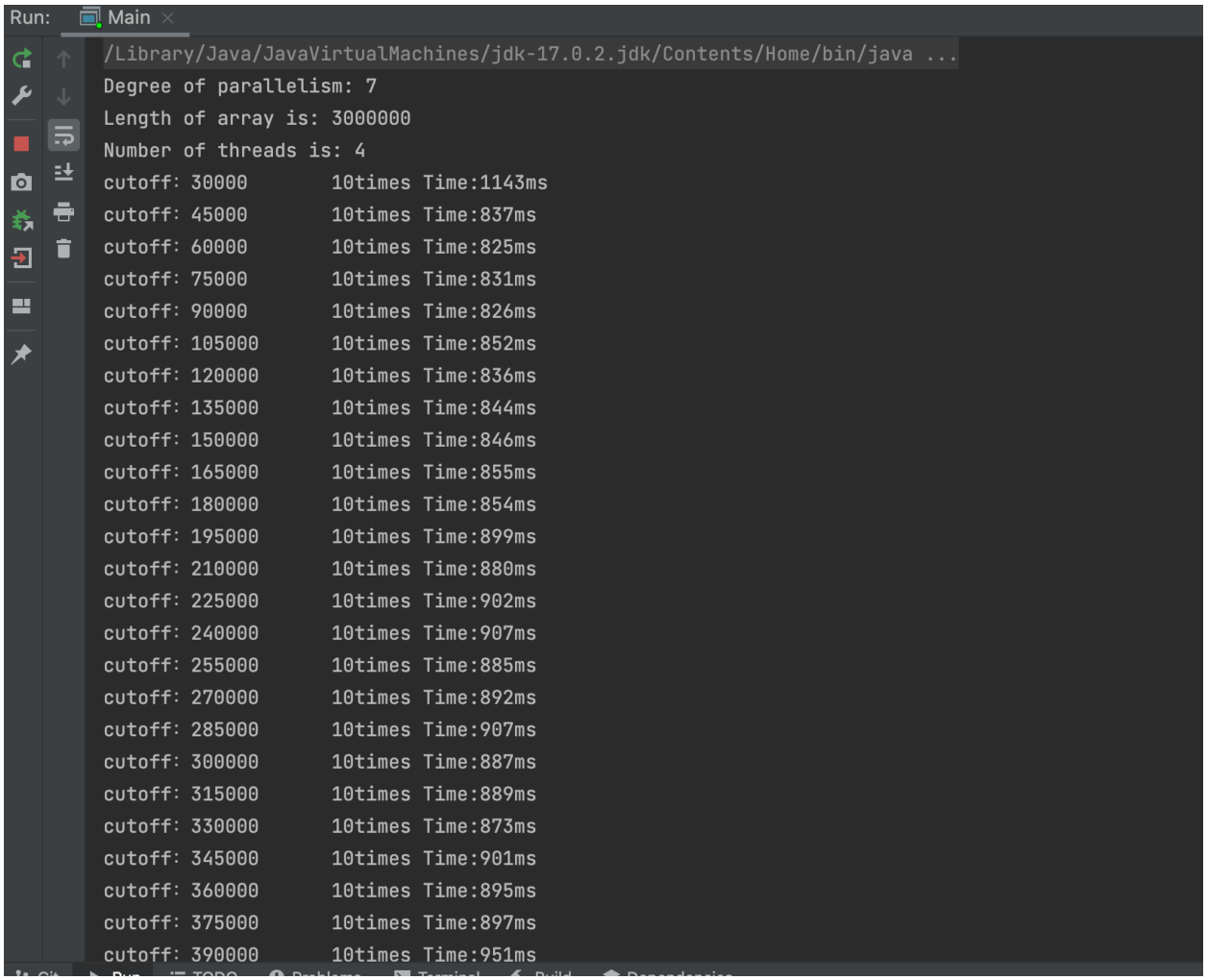


Degree of parallelism: 7

Array Length: 3000000

Thread count: 4

Number of cores in test machine: 8

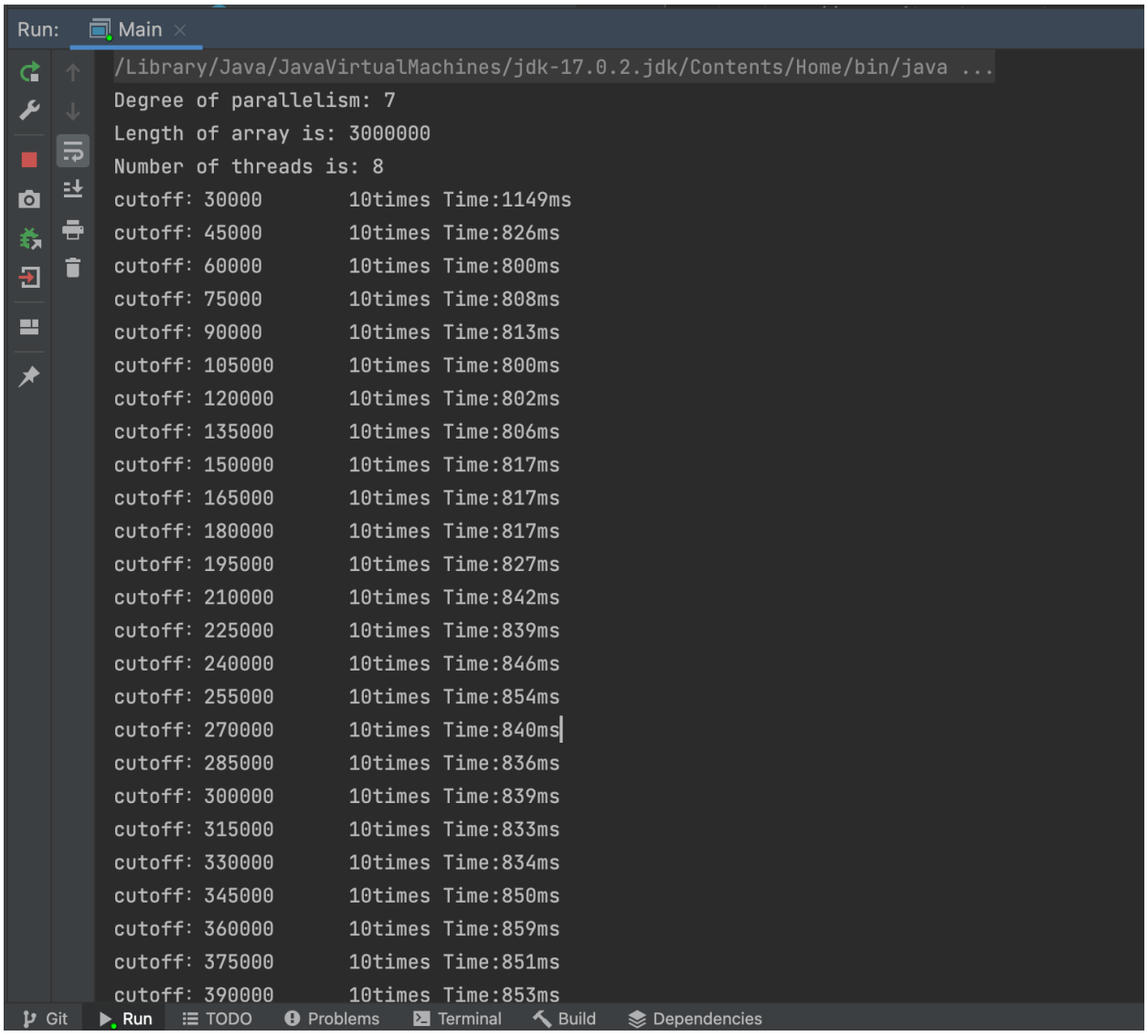


Degree of parallelism: 7

Array Length: 3000000

Thread count: 8

Number of cores in test machine: 8

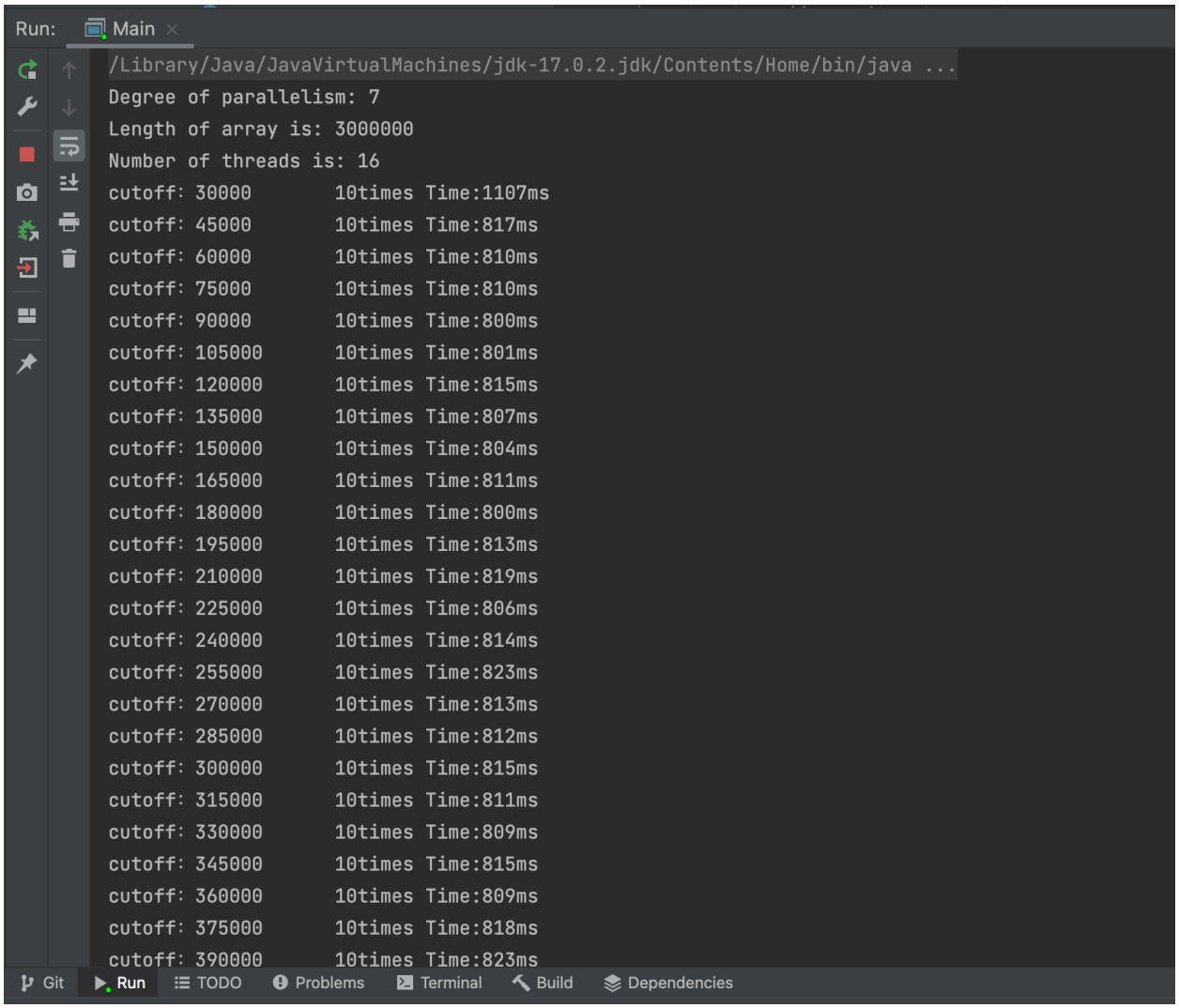


Degree of parallelism: 7

Array Length: 3000000

Thread count: 16

Number of cores in test machine: 8

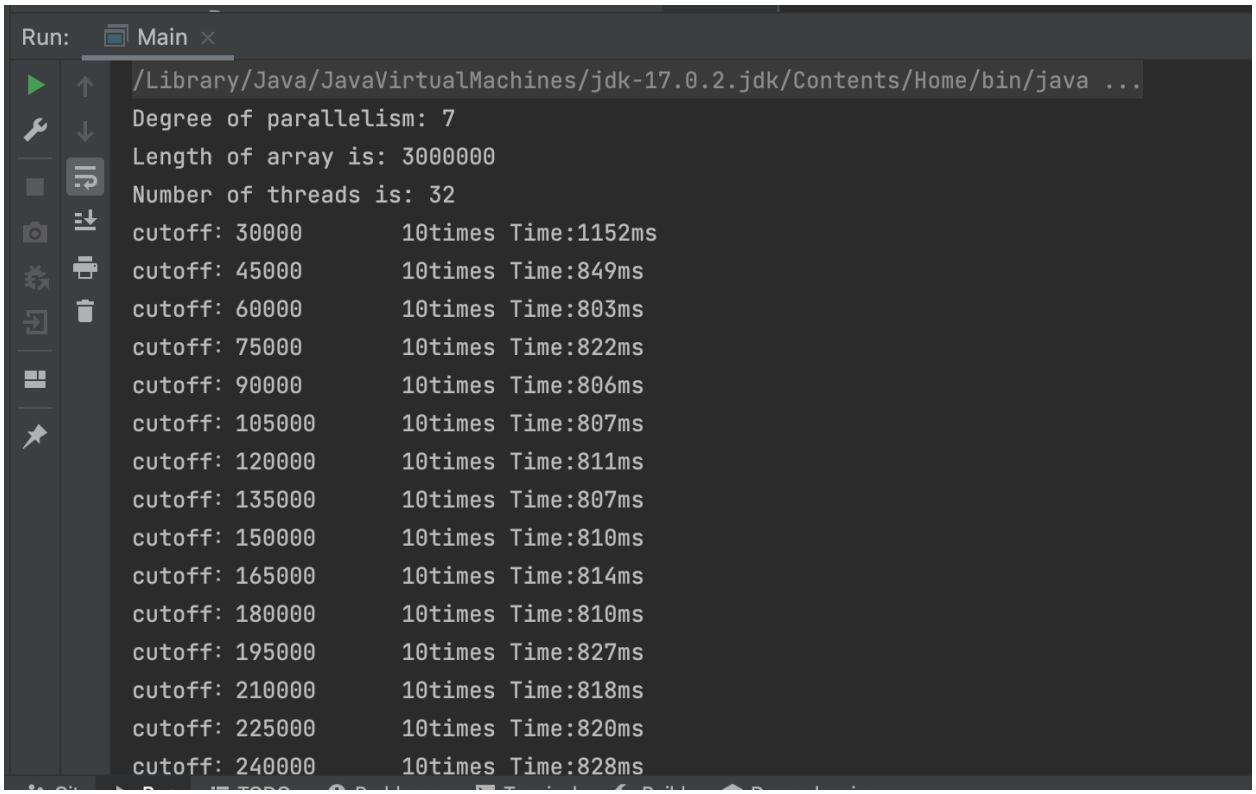


Degree of parallelism: 7

Array Length: 3000000

Thread count: 32

Number of cores in test machine: 8

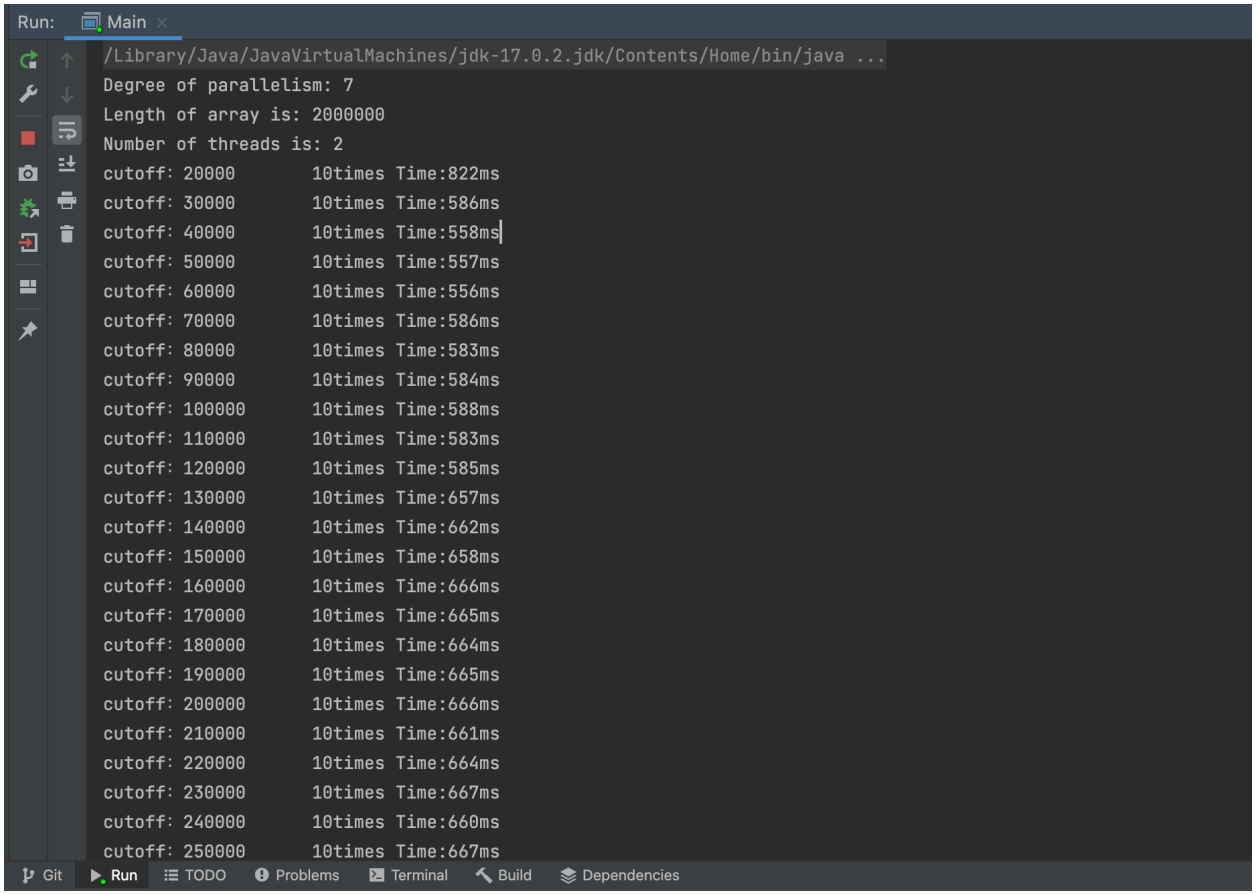


Degree of parallelism: 7

Array Length: 2000000

Thread count: 2

Number of cores in test machine: 8

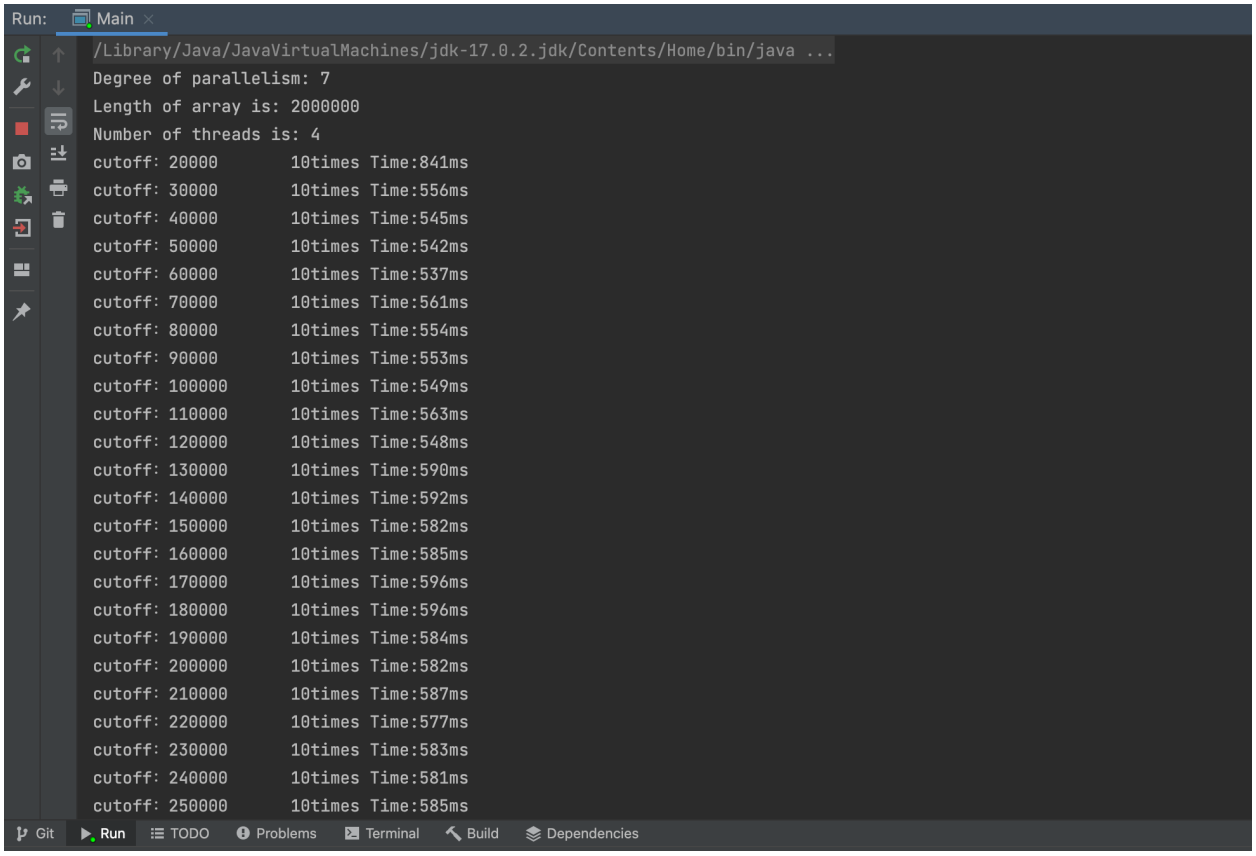


Degree of parallelism: 7

Array Length: 2000000

Thread count: 4

Number of cores in test machine: 8

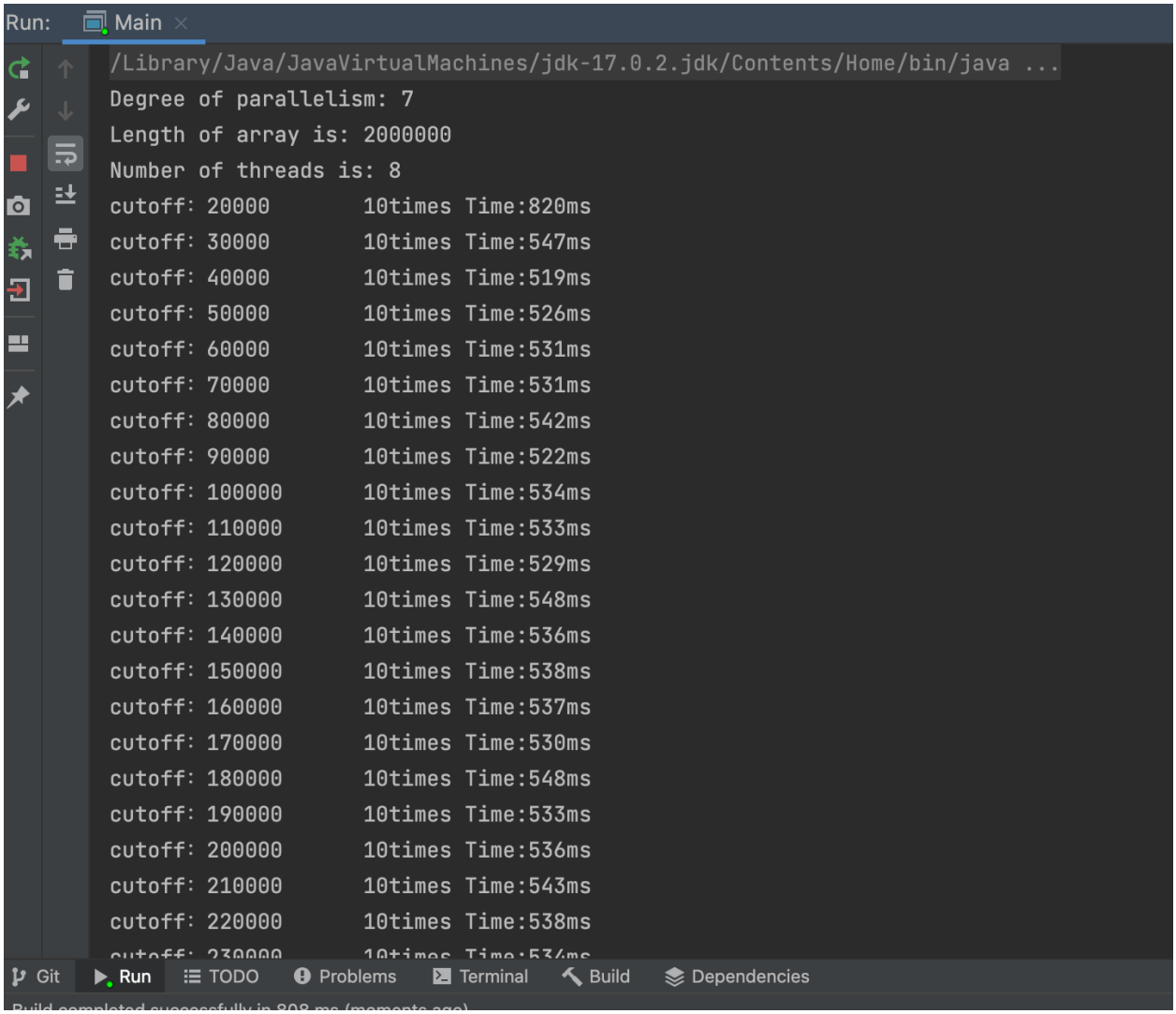


Degree of parallelism: 7

Array Length: 2000000

Thread count: 8

Number of cores in test machine: 8

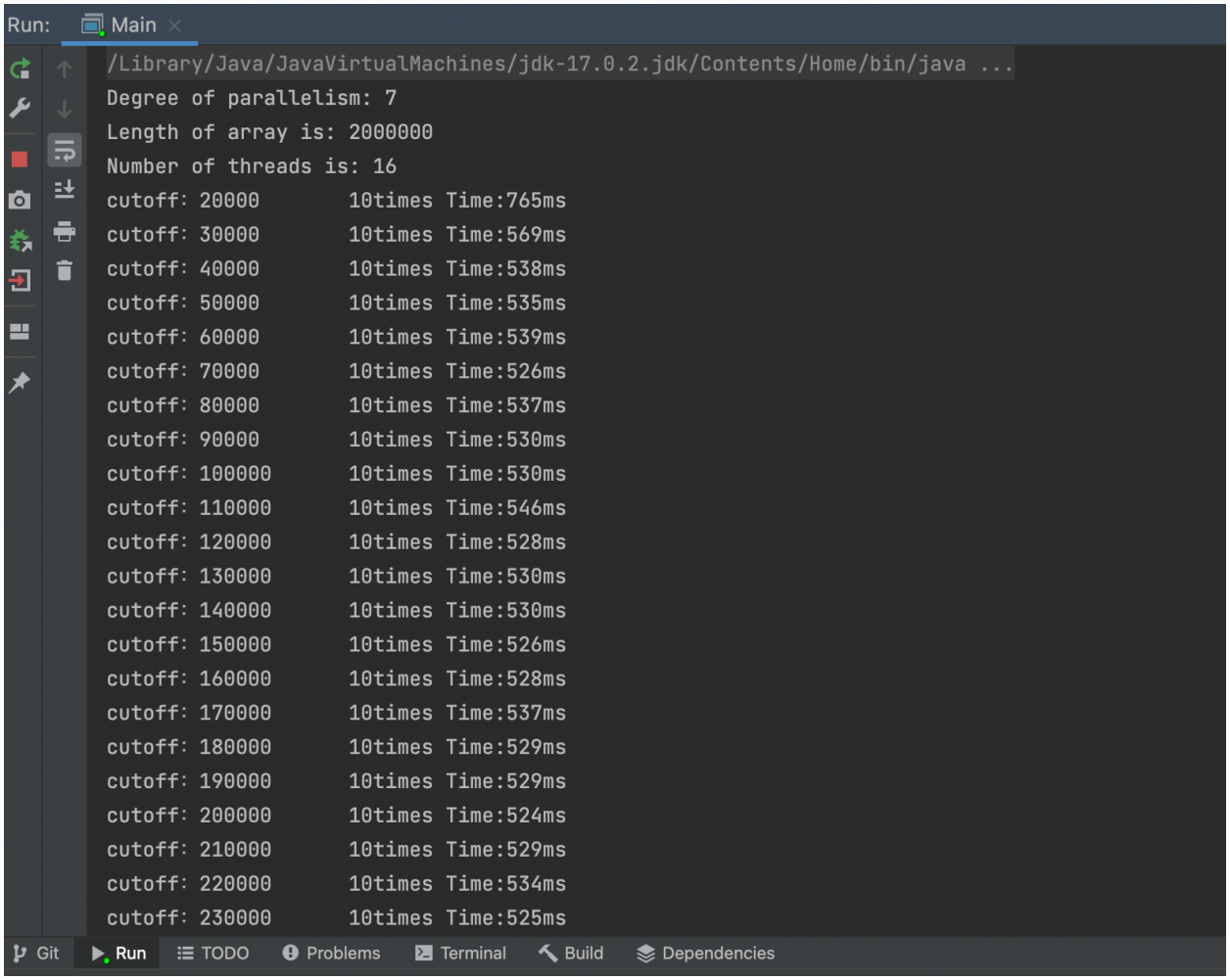


Degree of parallelism: 7

Array Length: 2000000

Thread count: 16

Number of cores in test machine: 8

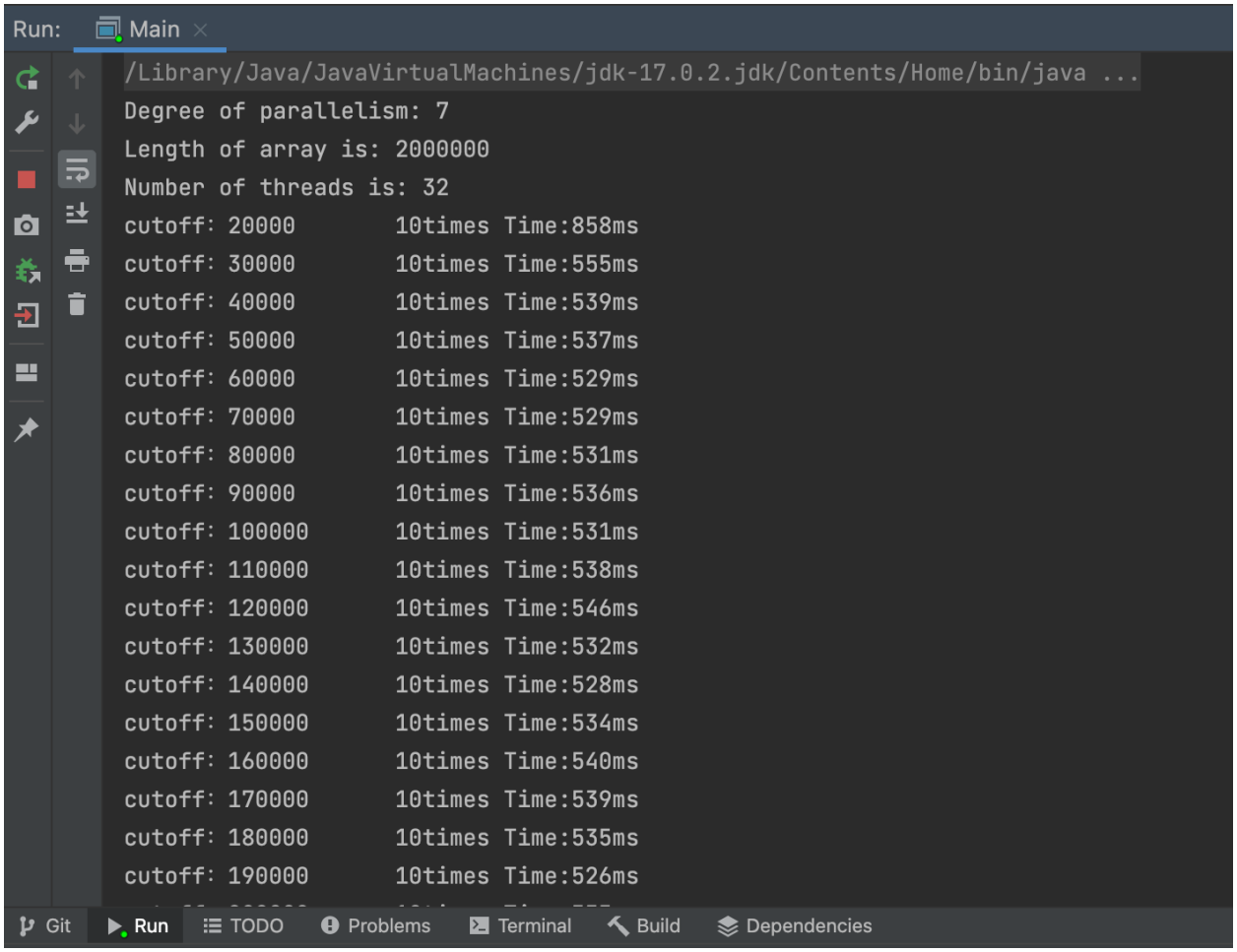


Degree of parallelism: 7

Array Length: 2000000

Thread count: 32

Number of cores in test machine: 8



Relationship Conclusion

We can conclude -

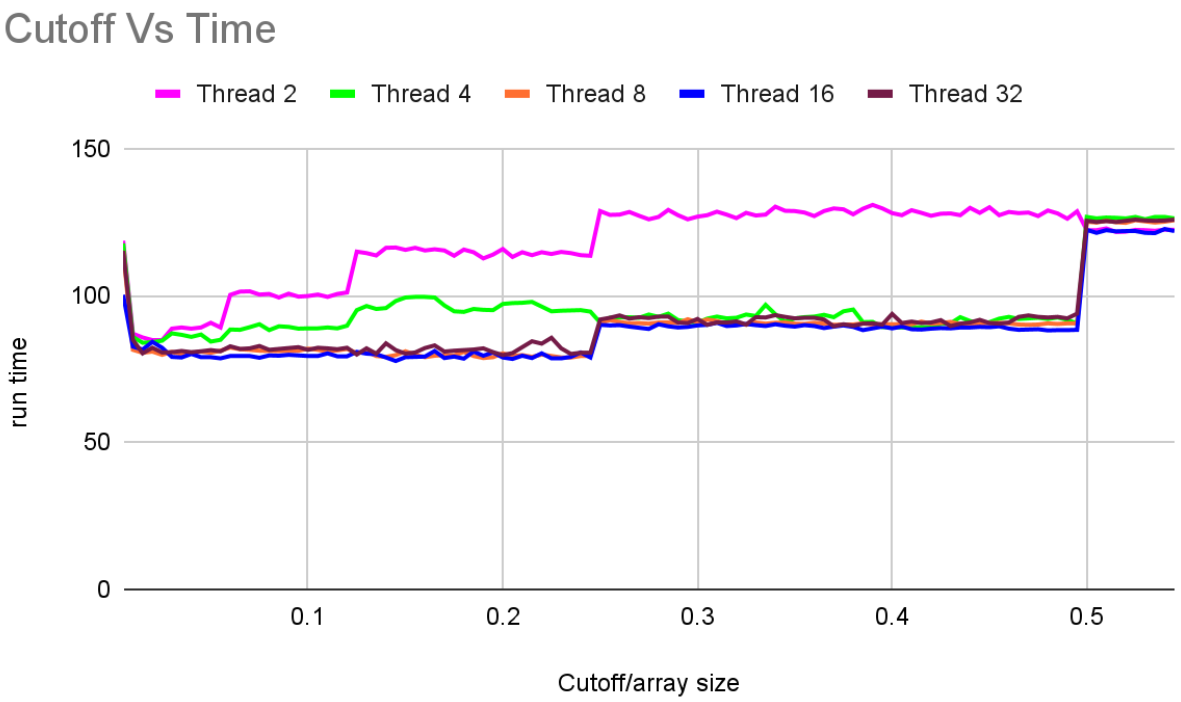
Best cutoff = array size / no. of threads

From the above data and chart where with different array sizes ranging from 1000000 and 2000000 and degree of parallelism is constant and thread count being 2, 4, 8, 16 and 32 for all the experiments, it can be observed that there is a decrease of time at first, reach the lowest point and then increase a bit where cutoff value is around 20% of the array size in all the experiments. Thus, concluding that 20% of the array size is the best cutoff value

Evidence/Graph

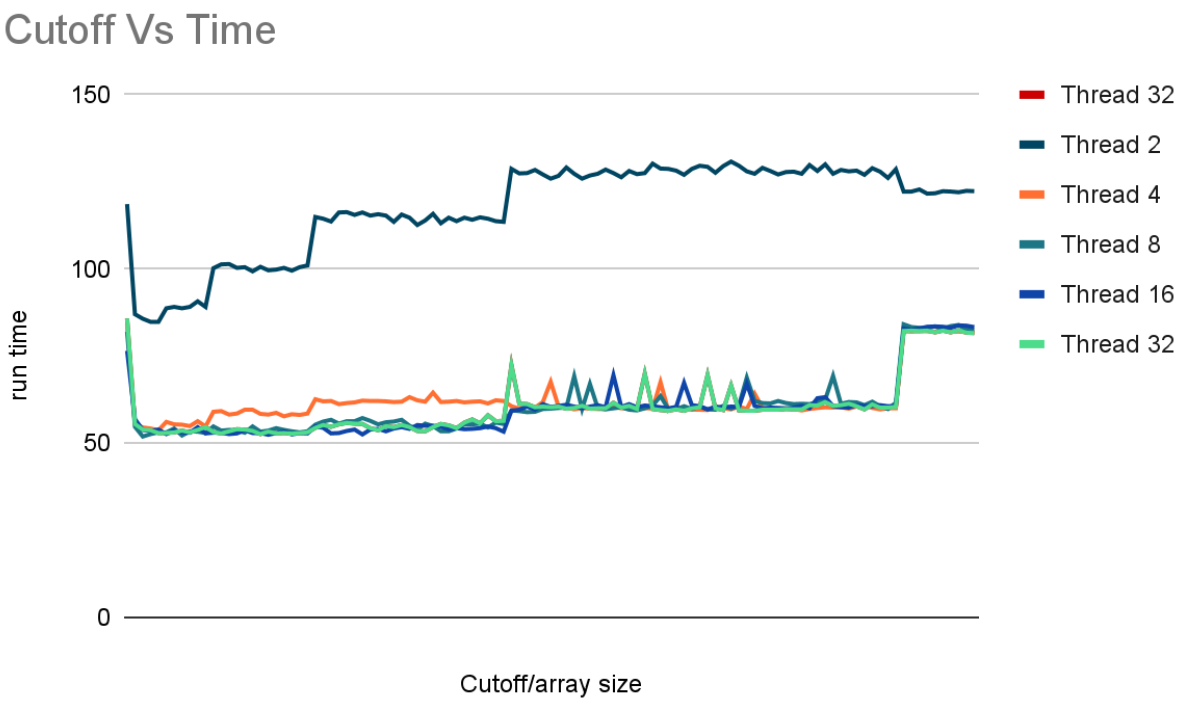
Degree of parallelism: 7

Array Length: 3000000



Degree of parallelism: 7

Array Length: 2000000



Git Repository -

https://github.com/Naina-NEU/INFO6205/commit/97850a26d50f9efdc1571f16639dbaeae12bff0c