Program Structures and Algorithms
Spring 2023(SEC –01)
Assignment-5

NAME: Ashi Tyagi
NUID: 002706544

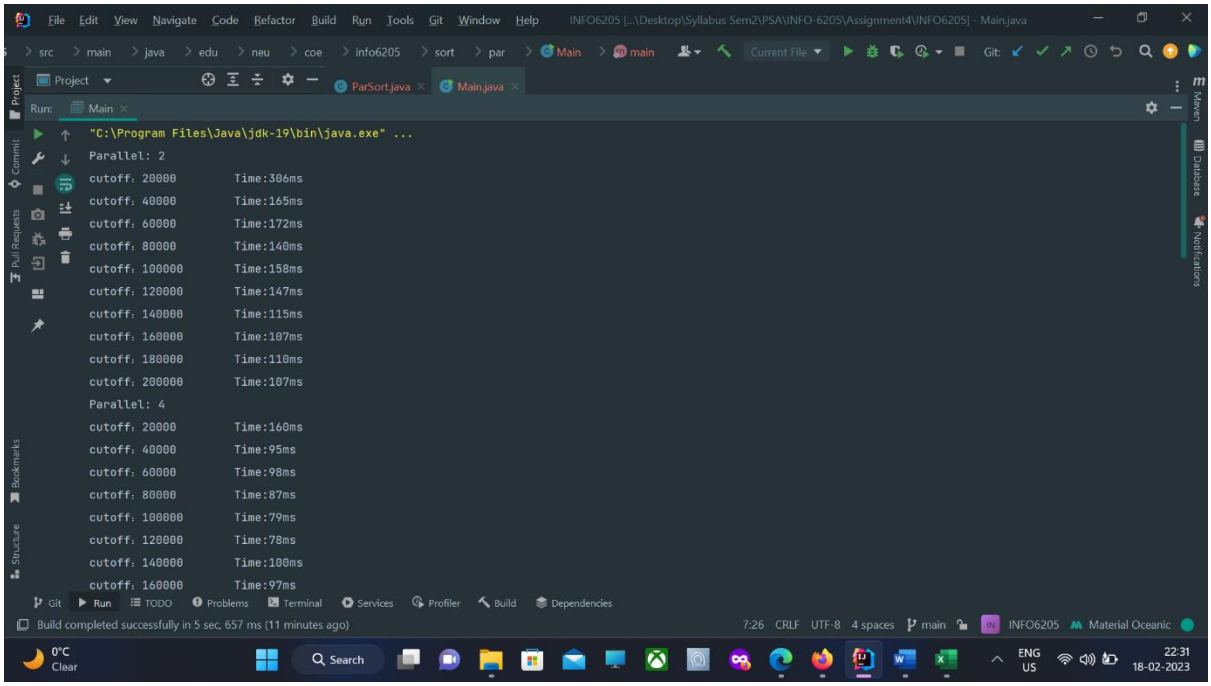
Task:

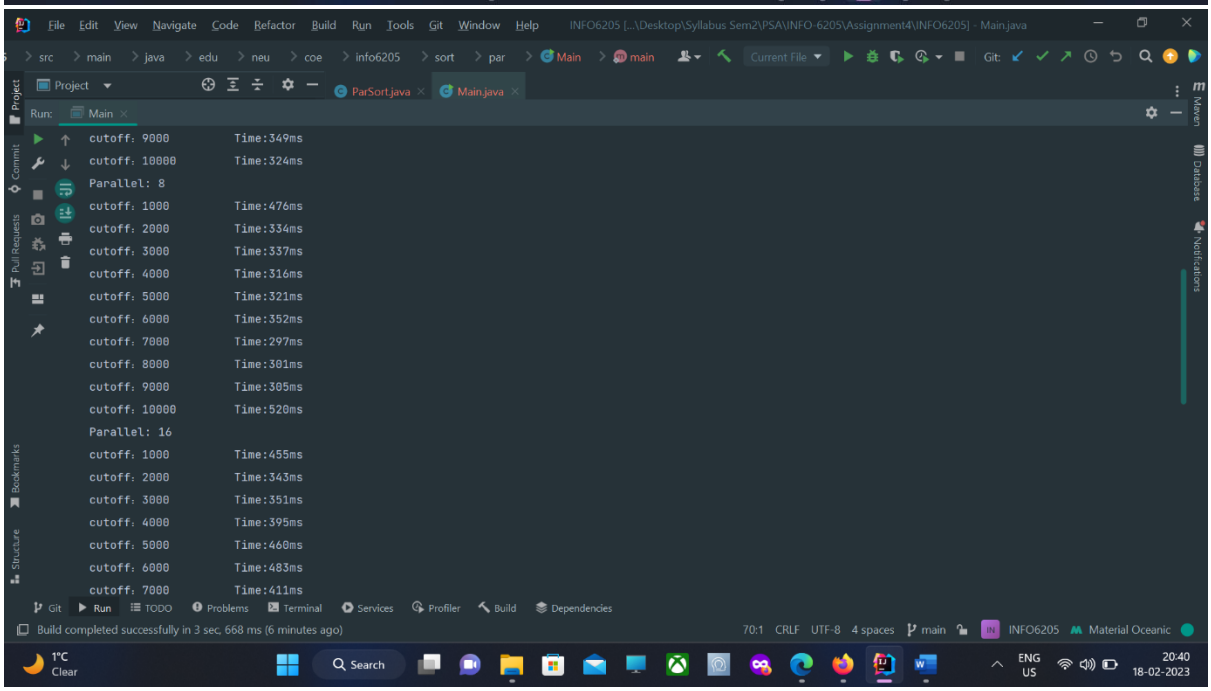
Please see the presentation on *Assignment on Parallel Sorting* under the *Exams. etc.* module. Your task is to implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. You will consider two different schemes for deciding whether to sort in parallel.

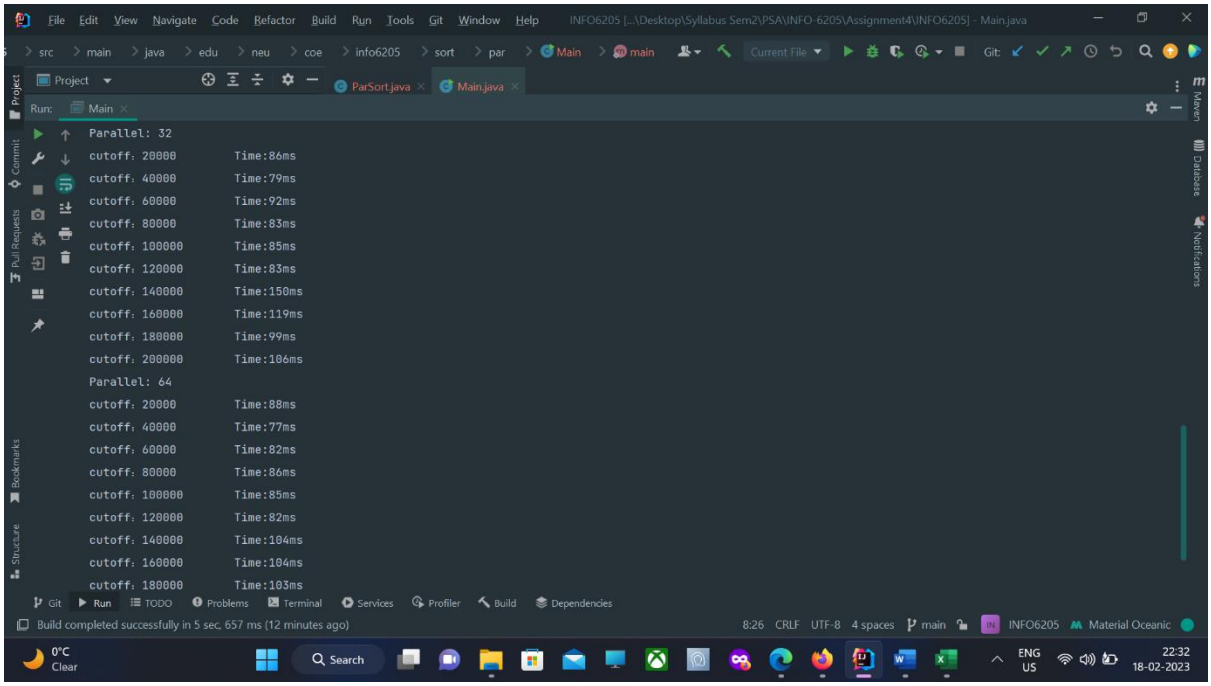
1. A cutoff (defaults to, say, 1000) which you will update according to the first argument in the command line when running. It's your job to experiment and come up with a good value for this cutoff. If there are fewer elements to sort than the cutoff, then you should use the system sort instead.
2. 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.

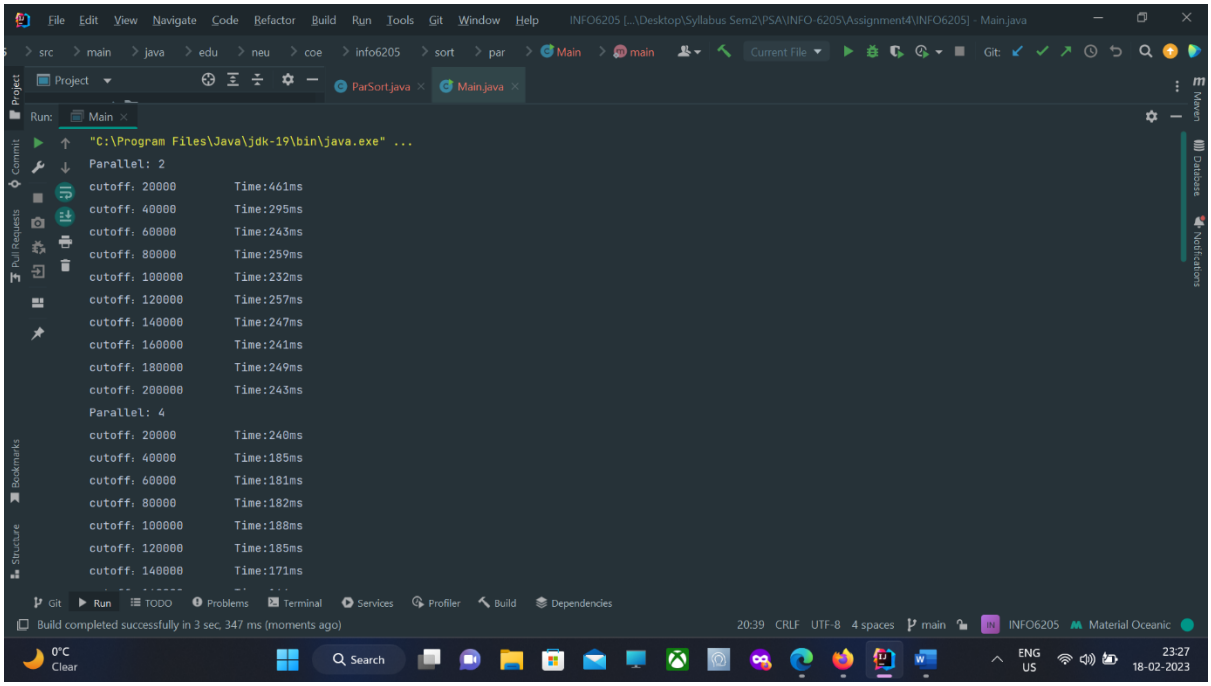
There is a *Main* class and the *ParSort* class in the *sort.par* package of the INFO6205 repository. The *Main* class can be used as is but the *ParSort* class needs to be implemented where you see "TODO..." [it turns out that these TODOs are already implemented].
Unless you have a good reason not to, 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 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.

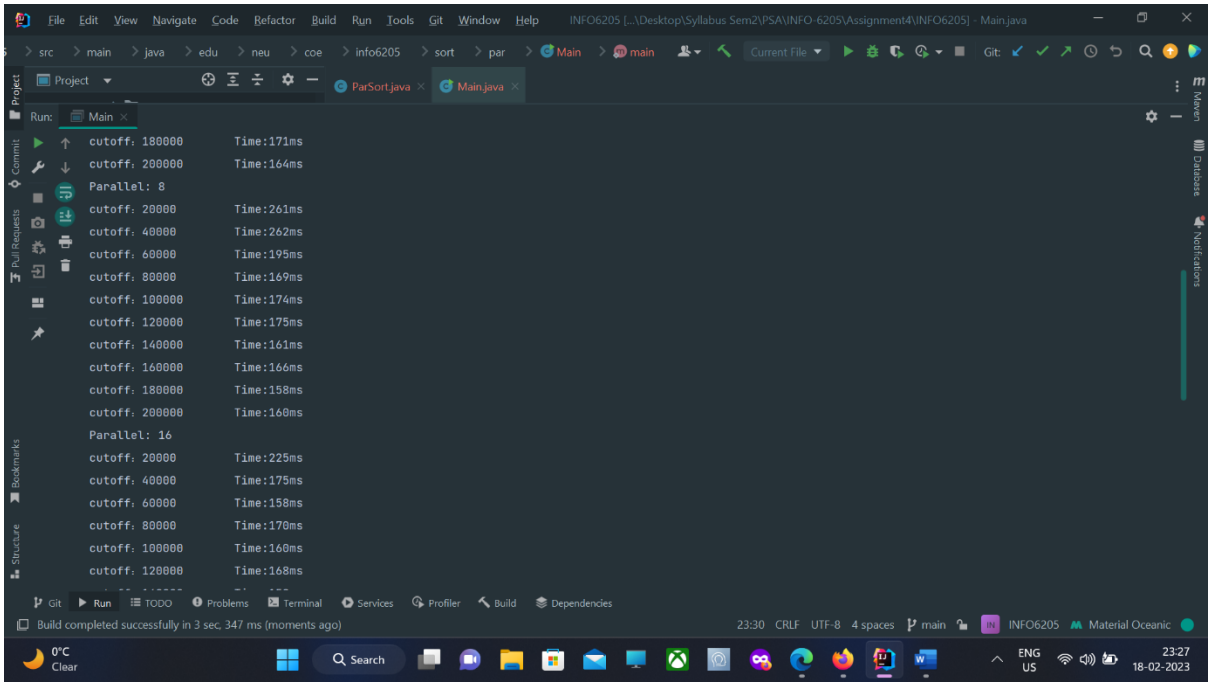
Outputs and Test cases:

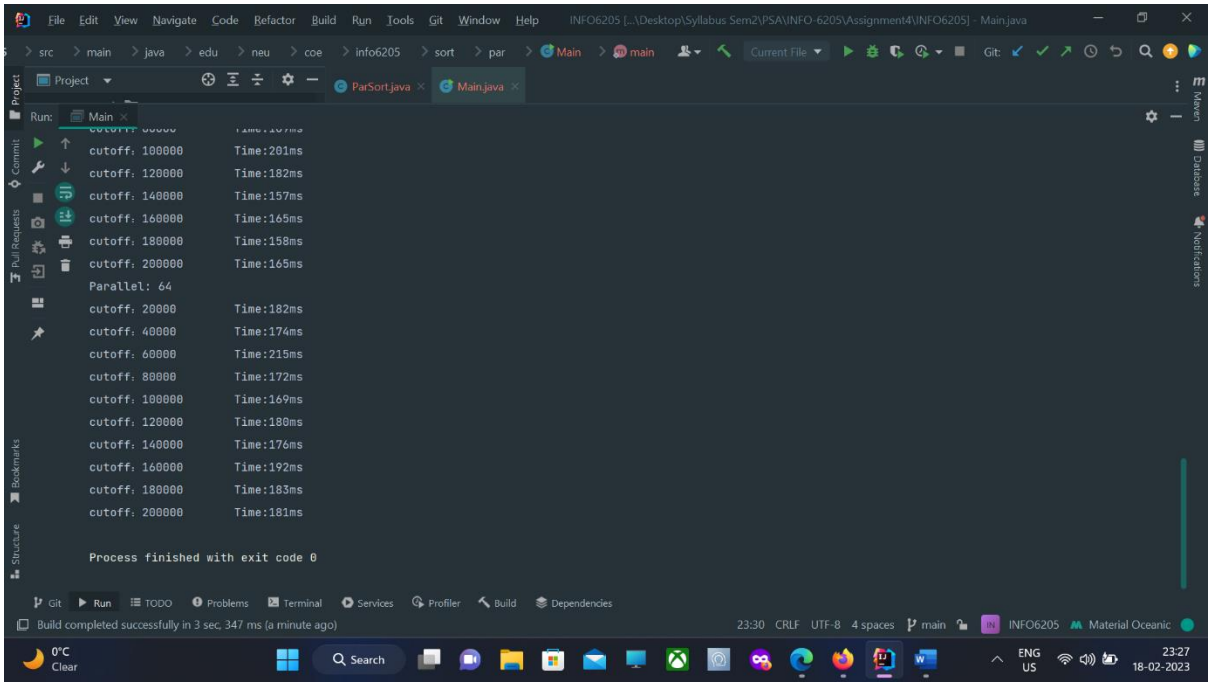

"C:\Program Files\Java\jdk-19\bin\java.exe" ...
Parallel: 2
cutoff: 20000 Time:306ms
cutoff: 40000 Time:165ms
cutoff: 60000 Time:172ms
cutoff: 80000 Time:140ms
cutoff: 100000 Time:158ms
cutoff: 120000 Time:147ms
cutoff: 140000 Time:115ms
cutoff: 160000 Time:107ms
cutoff: 180000 Time:110ms
cutoff: 200000 Time:107ms
Parallel: 4
cutoff: 20000 Time:160ms
cutoff: 40000 Time:95ms
cutoff: 60000 Time:98ms
cutoff: 80000 Time:87ms
cutoff: 100000 Time:79ms
cutoff: 120000 Time:78ms
cutoff: 140000 Time:100ms
cutoff: 160000 Time:97ms
Build completed successfully in 5 sec, 657 ms (11 minutes ago)

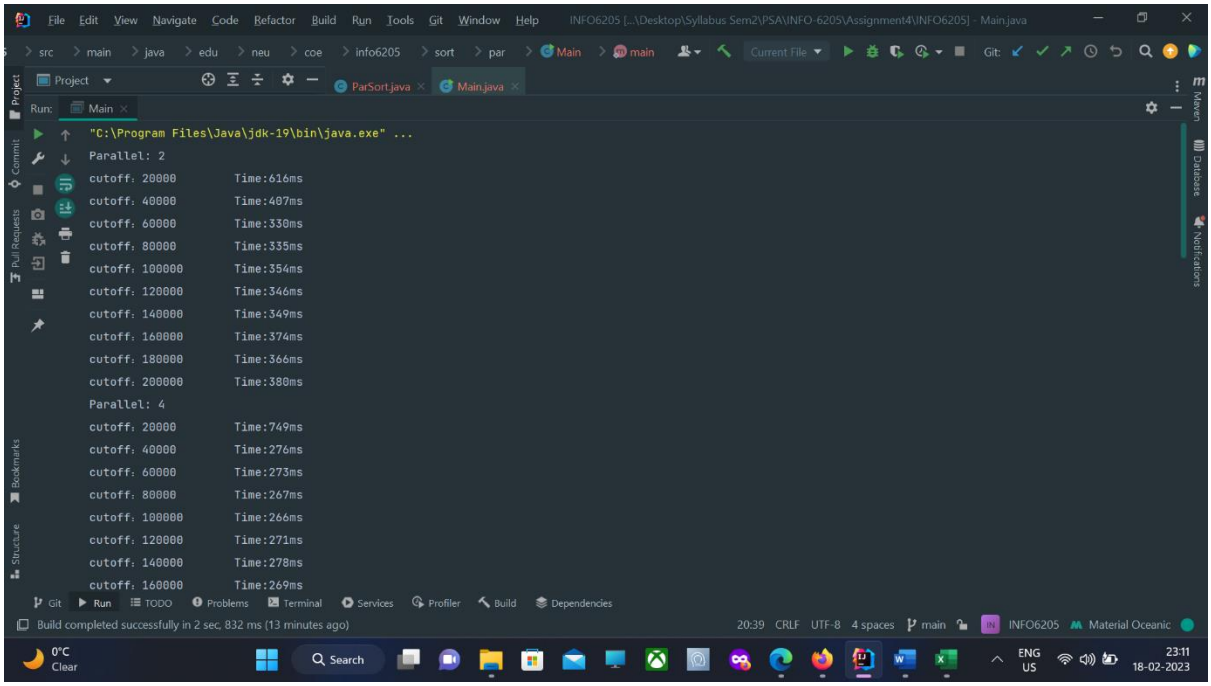

cutoff: 9000 Time:349ms
cutoff: 10000 Time:324ms
Parallel: 8
cutoff: 1000 Time:476ms
cutoff: 2000 Time:334ms
cutoff: 3000 Time:337ms
cutoff: 4000 Time:316ms
cutoff: 5000 Time:321ms
cutoff: 6000 Time:352ms
cutoff: 7000 Time:297ms
cutoff: 8000 Time:301ms
cutoff: 9000 Time:305ms
cutoff: 10000 Time:520ms
Parallel: 16
cutoff: 1000 Time:455ms
cutoff: 2000 Time:343ms
cutoff: 3000 Time:351ms
cutoff: 4000 Time:395ms
cutoff: 5000 Time:460ms
cutoff: 6000 Time:483ms
cutoff: 7000 Time:411ms
Build completed successfully in 3 sec, 668 ms (6 minutes ago)

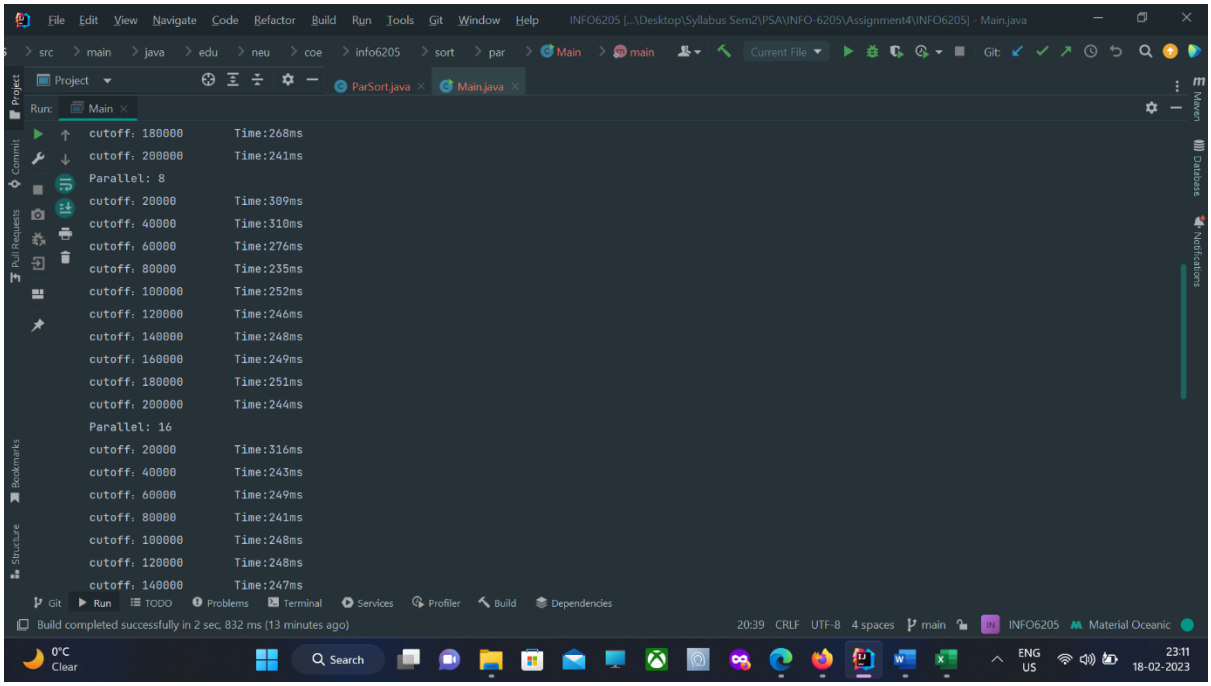

Parallel: 32
cutoff: 20000 Time:86ms
cutoff: 40000 Time:79ms
cutoff: 60000 Time:92ms
cutoff: 80000 Time:83ms
cutoff: 100000 Time:85ms
cutoff: 120000 Time:83ms
cutoff: 140000 Time:150ms
cutoff: 160000 Time:119ms
cutoff: 180000 Time:99ms
cutoff: 200000 Time:106ms
Parallel: 64
cutoff: 20000 Time:88ms
cutoff: 40000 Time:77ms
cutoff: 60000 Time:82ms
cutoff: 80000 Time:86ms
cutoff: 100000 Time:85ms
cutoff: 120000 Time:82ms
cutoff: 140000 Time:104ms
cutoff: 160000 Time:104ms
cutoff: 180000 Time:103ms
Build completed successfully in 5 sec, 657 ms (12 minutes ago)

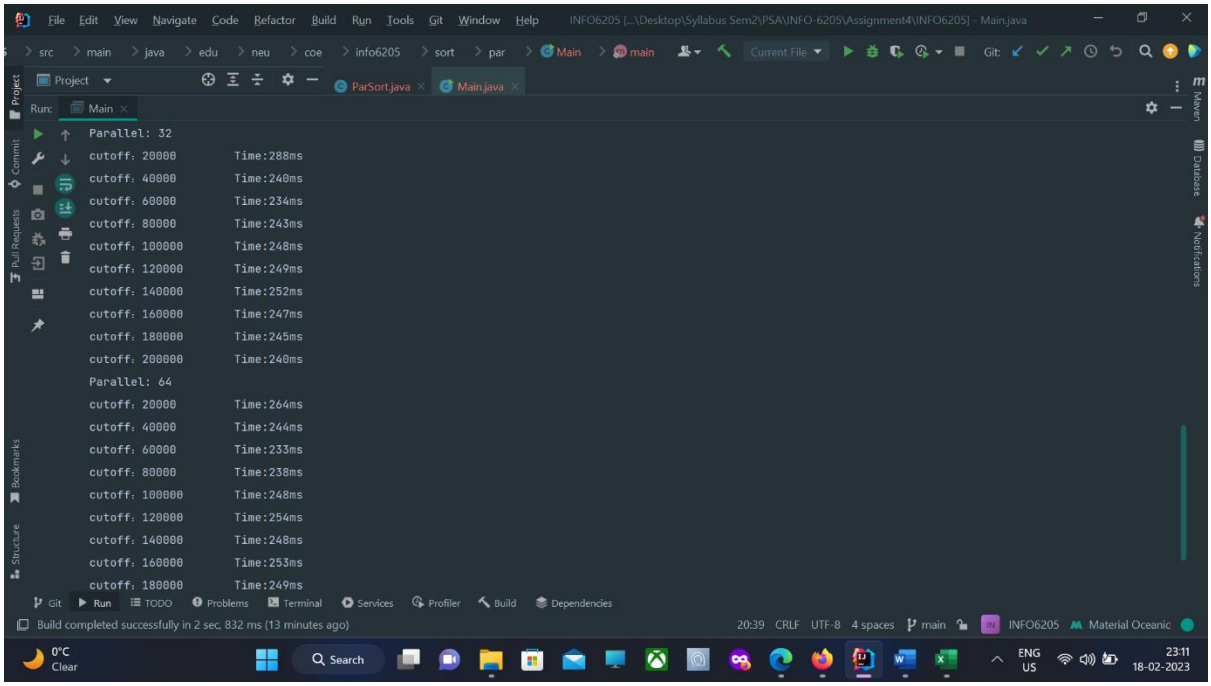

"C:\Program Files\Java\jdk-19\bin\java.exe" ...
Parallel: 2
cutoff: 20000 Time:461ms
cutoff: 40000 Time:295ms
cutoff: 60000 Time:243ms
cutoff: 80000 Time:259ms
cutoff: 100000 Time:232ms
cutoff: 120000 Time:257ms
cutoff: 140000 Time:247ms
cutoff: 160000 Time:241ms
cutoff: 180000 Time:249ms
cutoff: 200000 Time:243ms
Parallel: 4
cutoff: 20000 Time:240ms
cutoff: 40000 Time:185ms
cutoff: 60000 Time:181ms
cutoff: 80000 Time:182ms
cutoff: 100000 Time:188ms
cutoff: 120000 Time:185ms
cutoff: 140000 Time:171ms
Build completed successfully in 3 sec, 347 ms (moments ago)


cutoff: 180000 Time:171ms
cutoff: 200000 Time:164ms
Parallel: 8
cutoff: 20000 Time:261ms
cutoff: 40000 Time:262ms
cutoff: 60000 Time:195ms
cutoff: 80000 Time:169ms
cutoff: 100000 Time:174ms
cutoff: 120000 Time:175ms
cutoff: 140000 Time:161ms
cutoff: 160000 Time:166ms
cutoff: 180000 Time:158ms
cutoff: 200000 Time:160ms
Parallel: 16
cutoff: 20000 Time:225ms
cutoff: 40000 Time:175ms
cutoff: 60000 Time:158ms
cutoff: 80000 Time:170ms
cutoff: 100000 Time:160ms
cutoff: 120000 Time:168ms
Build completed successfully in 3 sec, 347 ms (moments ago)


cutoff: 100000 Time:201ms
cutoff: 120000 Time:182ms
cutoff: 140000 Time:157ms
cutoff: 160000 Time:165ms
cutoff: 180000 Time:158ms
cutoff: 200000 Time:165ms
Parallel: 64
cutoff: 20000 Time:182ms
cutoff: 40000 Time:174ms
cutoff: 60000 Time:215ms
cutoff: 80000 Time:172ms
cutoff: 100000 Time:169ms
cutoff: 120000 Time:180ms
cutoff: 140000 Time:176ms
cutoff: 160000 Time:192ms
cutoff: 180000 Time:183ms
cutoff: 200000 Time:181ms
Process finished with exit code 0
Build completed successfully in 3 sec, 347 ms (a minute ago)


"C:\Program Files\Java\jdk-19\bin\java.exe" ...
Parallel: 2
cutoff: 20000 Time:616ms
cutoff: 40000 Time:407ms
cutoff: 60000 Time:330ms
cutoff: 80000 Time:335ms
cutoff: 100000 Time:354ms
cutoff: 120000 Time:346ms
cutoff: 140000 Time:349ms
cutoff: 160000 Time:374ms
cutoff: 180000 Time:366ms
cutoff: 200000 Time:380ms
Parallel: 4
cutoff: 20000 Time:749ms
cutoff: 40000 Time:276ms
cutoff: 60000 Time:273ms
cutoff: 80000 Time:267ms
cutoff: 100000 Time:266ms
cutoff: 120000 Time:271ms
cutoff: 140000 Time:278ms
cutoff: 160000 Time:269ms


cutoff: 180000 Time:268ms
cutoff: 200000 Time:241ms
Parallel: 8
cutoff: 20000 Time:309ms
cutoff: 40000 Time:310ms
cutoff: 60000 Time:276ms
cutoff: 80000 Time:235ms
cutoff: 100000 Time:252ms
cutoff: 120000 Time:246ms
cutoff: 140000 Time:248ms
cutoff: 160000 Time:249ms
cutoff: 180000 Time:251ms
cutoff: 200000 Time:244ms
Parallel: 16
cutoff: 20000 Time:316ms
cutoff: 40000 Time:243ms
cutoff: 60000 Time:249ms
cutoff: 80000 Time:241ms
cutoff: 100000 Time:248ms
cutoff: 120000 Time:248ms
cutoff: 140000 Time:247ms



Relationship Conclusion/ Evidence Graph:

Cutoff values for array of size: 250,000

Cutoff	Thread:2	Thread:4	Thread:8	Thread:16	Thread:32	Thread:64
20000	64.2	52	13.2	12.7	13.1	14.5
40000	30.5	12.8	9.1	9.3	9.6	10.7
60000	25.7	12.1	9.1	10.5	10.4	9
80000	16.5	9.4	8.3	10	35.6	8.8
100000	17.6	8.6	8.5	8.8	12.5	8.9
120000	34.3	8.9	8.5	8.9	12.4	8.7
140000	17.5	9.4	8.8	8.9	9.4	8.7
160000	25.5	9.5	8.5	8.3	8.7	7.9
180000	19	8.7	8.7	8.5	8.5	7.9
200000	20.9	8.6	8.6	8.2	7.7	8.4

Cutoff values for array of size: 500,000

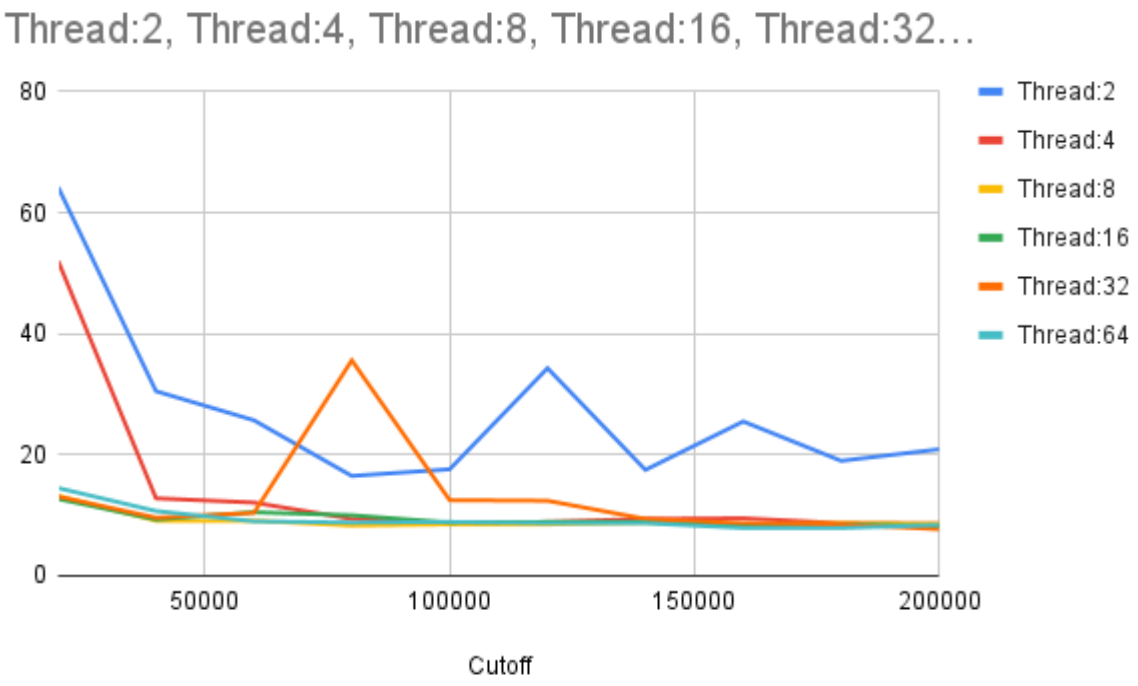
Cutoff	Thread:2	Thread:4	Thread:8	Thread:16	Thread:32	Thread:64
20000	42.7	25.5	24.7	22.9	16.4	17.2
40000	26.7	18.6	25.3	15.8	15.9	16.1
60000	22.8	18.5	18.1	15.6	16.4	15.9

80000	23.3	18.8	17.5	17	17.5	17.5
100000	24	18.3	17.8	16.6	17.1	17.3
120000	24.3	18.4	17.8	16.7	16.9	16.3
140000	23.5	17.2	19.7	17	16.4	16.4
160000	25.2	17.4	19.4	16.5	16.4	16.8
180000	25.1	17.1	16.4	16.4	16.7	16.9
200000	25.6	16.8	16.3	16.5	16.4	16.6

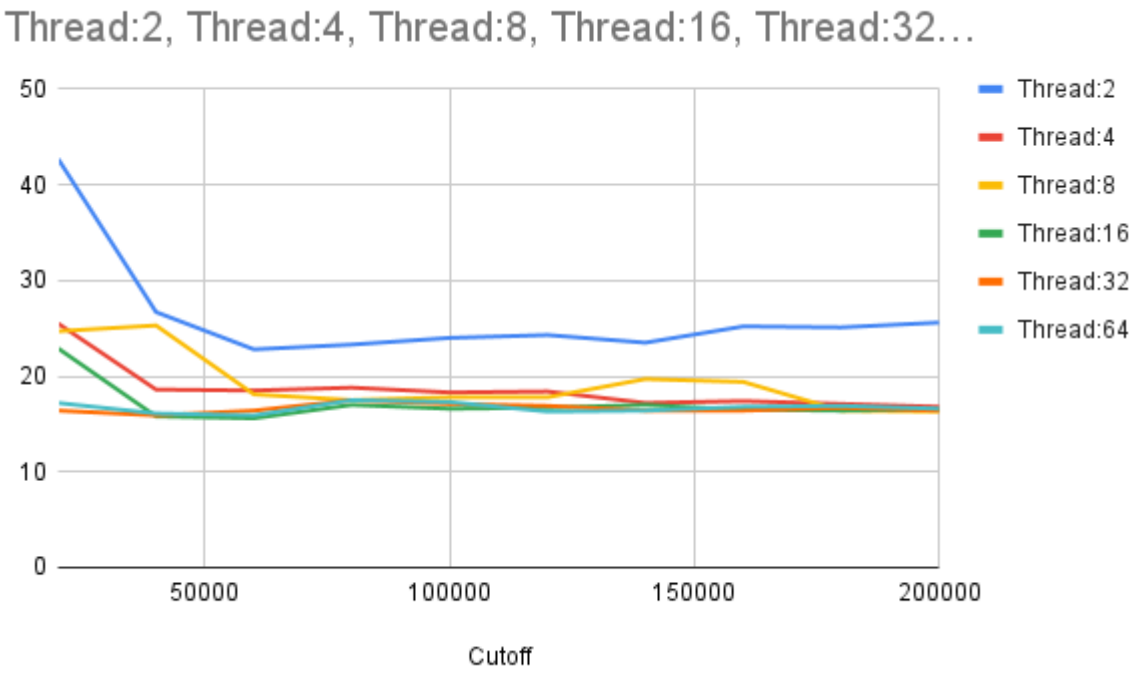
Cutoff values for array of size: 750,000

Cutoff	Thread:2	Thread:4	Thread:8	Thread:16	Thread:32	Thread:64
20000	61.6	74.9	30.9	31.6	28.8	26.4
40000	40.7	27.6	31	24.3	24	24.4
60000	33	27.3	27.6	24.9	23.4	23.3
80000	33.5	26.7	23.5	24.1	24.3	23.8
100000	35.4	26.6	25.2	24.8	24.8	24.8
120000	34.6	27.1	24.6	24.8	24.9	25.4
140000	34.9	27.8	24.8	24.7	25.2	24.8
160000	37.4	26.9	24.9	25	24.7	25.3
180000	36.6	26.8	25.1	24.8	24.5	24.9
200000	38	24.1	24.4	24.6	24	24.4

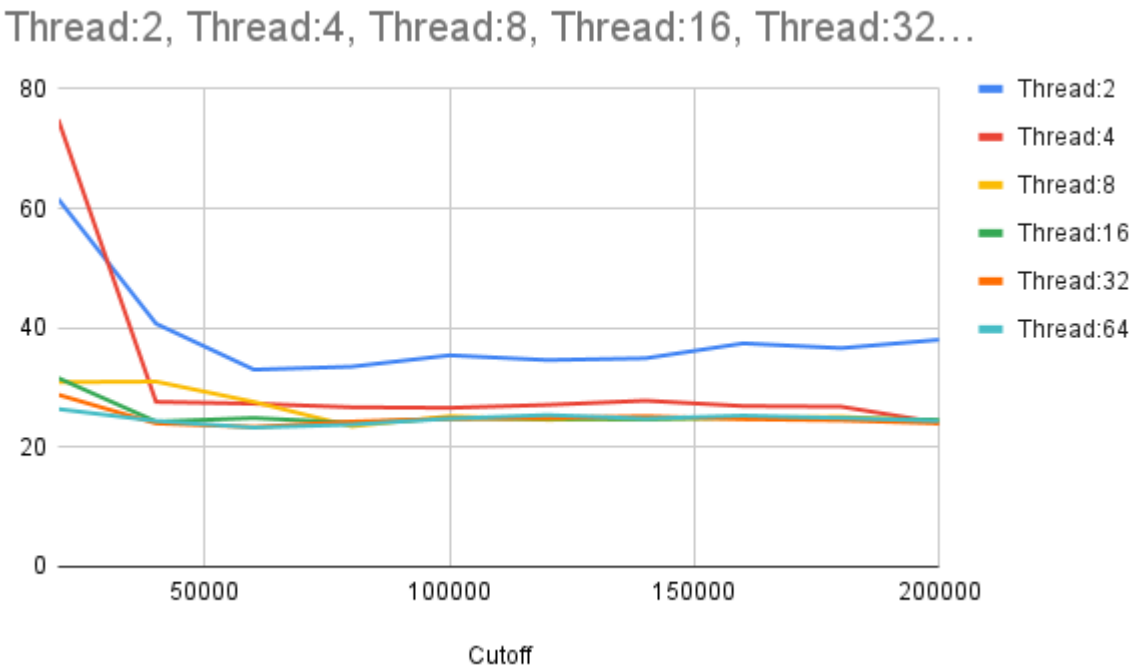
Graph for array size: 250,000



Graph for array size: 500,000



Graph for array size: 750,000



From the graph above we can conclude that:

After plotting the above three graphs from the different values tested for their cut-off limits. It can be understood that the cut-off ratio, defined as the ratio of cut off to the array size helps us understand the relationship between both. The ratio tends towards a constant time when there are sufficiently large gaps between the parallel sorting parameters and array size. The values plotted for the largest array size shows constant behaviour in plotting for the graph. Thus, it could be understood to keep the values of array size to be sufficiently large in comparison to the parallel sorting values to yield a comparable and appropriate cut-off value. The above values shows least constant behaviour when the array size is closest to parallel sorting parameters, there the graph plotted is erratic and not of much linearity.