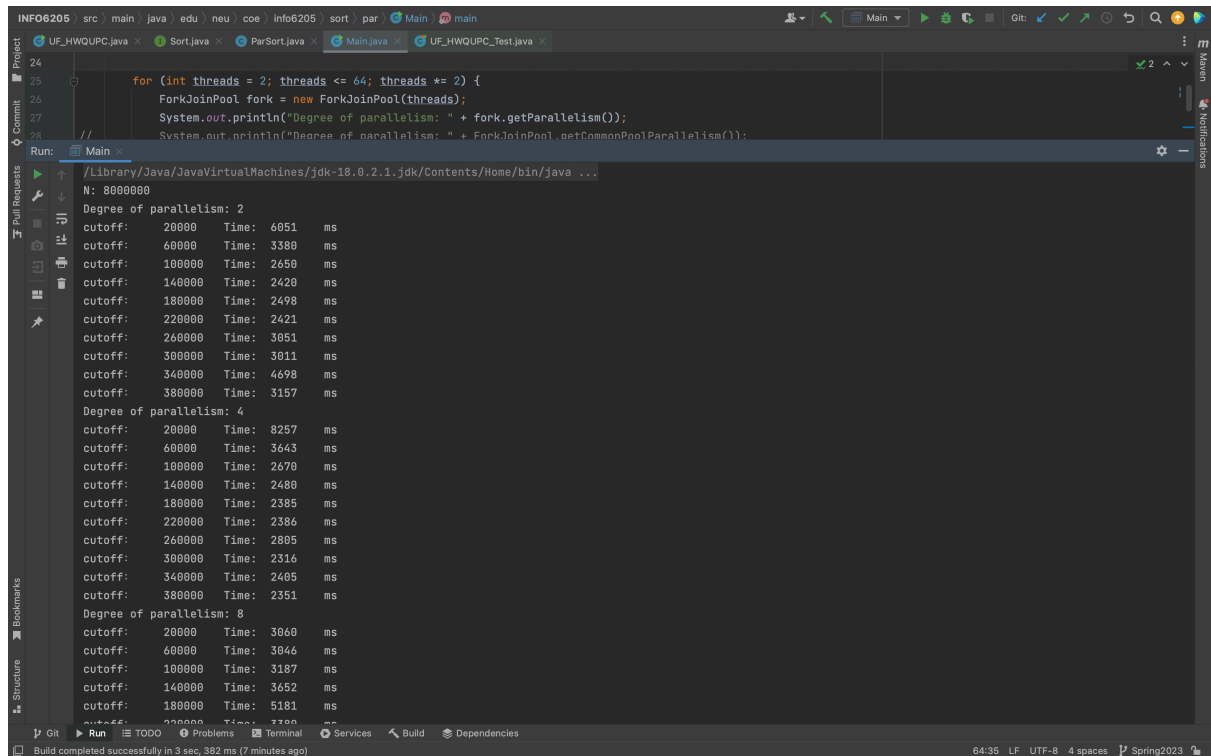


Assignment 5

1. Program Output

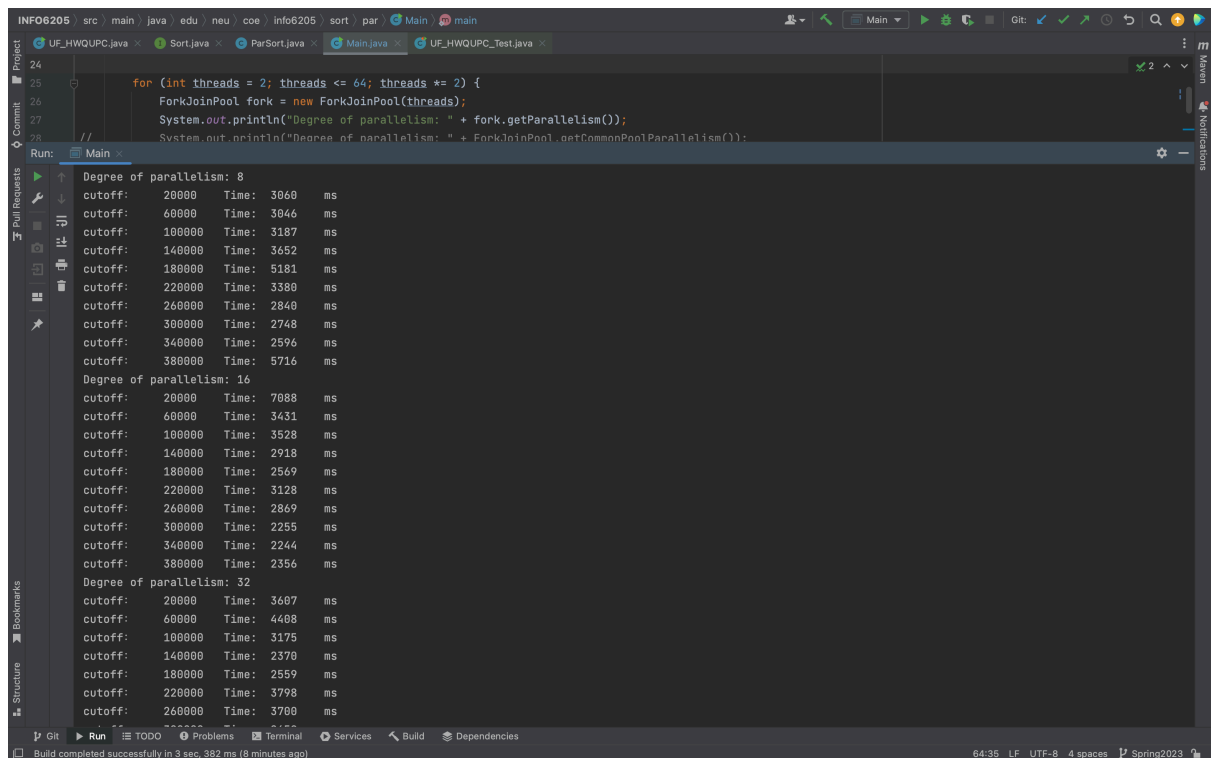


The screenshot shows an IDE with the following code in `Main.java`:

```
24
25     for (int threads = 2; threads <= 64; threads *= 2) {
26         ForkJoinPool fork = new ForkJoinPool(threads);
27         System.out.println("Degree of parallelism: " + fork.getParallelism());
28         // System.out.println("Degree of parallelism: " + ForkJoinPool.getCommonPoolParallelism());
```

The Run console output is as follows:

```
/Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...
N: 8000000
Degree of parallelism: 2
cutoff: 20000 Time: 6051 ms
cutoff: 60000 Time: 3380 ms
cutoff: 100000 Time: 2650 ms
cutoff: 140000 Time: 2420 ms
cutoff: 180000 Time: 2498 ms
cutoff: 220000 Time: 2421 ms
cutoff: 260000 Time: 3051 ms
cutoff: 300000 Time: 3011 ms
cutoff: 340000 Time: 4698 ms
cutoff: 380000 Time: 3157 ms
Degree of parallelism: 4
cutoff: 20000 Time: 8257 ms
cutoff: 60000 Time: 3643 ms
cutoff: 100000 Time: 2670 ms
cutoff: 140000 Time: 2480 ms
cutoff: 180000 Time: 2385 ms
cutoff: 220000 Time: 2386 ms
cutoff: 260000 Time: 2805 ms
cutoff: 300000 Time: 2316 ms
cutoff: 340000 Time: 2405 ms
cutoff: 380000 Time: 2351 ms
Degree of parallelism: 8
cutoff: 20000 Time: 3060 ms
cutoff: 60000 Time: 3046 ms
cutoff: 100000 Time: 3187 ms
cutoff: 140000 Time: 3652 ms
cutoff: 180000 Time: 5181 ms
```



The screenshot shows the same IDE with the same code as above. The Run console output continues with the following data:

```

cutoff: 20000 Time: 3700 ms
Degree of parallelism: 16
cutoff: 20000 Time: 7088 ms
cutoff: 60000 Time: 3431 ms
cutoff: 100000 Time: 3528 ms
cutoff: 140000 Time: 2918 ms
cutoff: 180000 Time: 2569 ms
cutoff: 220000 Time: 3128 ms
cutoff: 260000 Time: 2869 ms
cutoff: 300000 Time: 2255 ms
cutoff: 340000 Time: 2244 ms
cutoff: 380000 Time: 2356 ms
Degree of parallelism: 32
cutoff: 20000 Time: 3607 ms
cutoff: 60000 Time: 4408 ms
cutoff: 100000 Time: 3175 ms
cutoff: 140000 Time: 2370 ms
cutoff: 180000 Time: 2559 ms
cutoff: 220000 Time: 3798 ms
cutoff: 260000 Time: 3700 ms
```

```

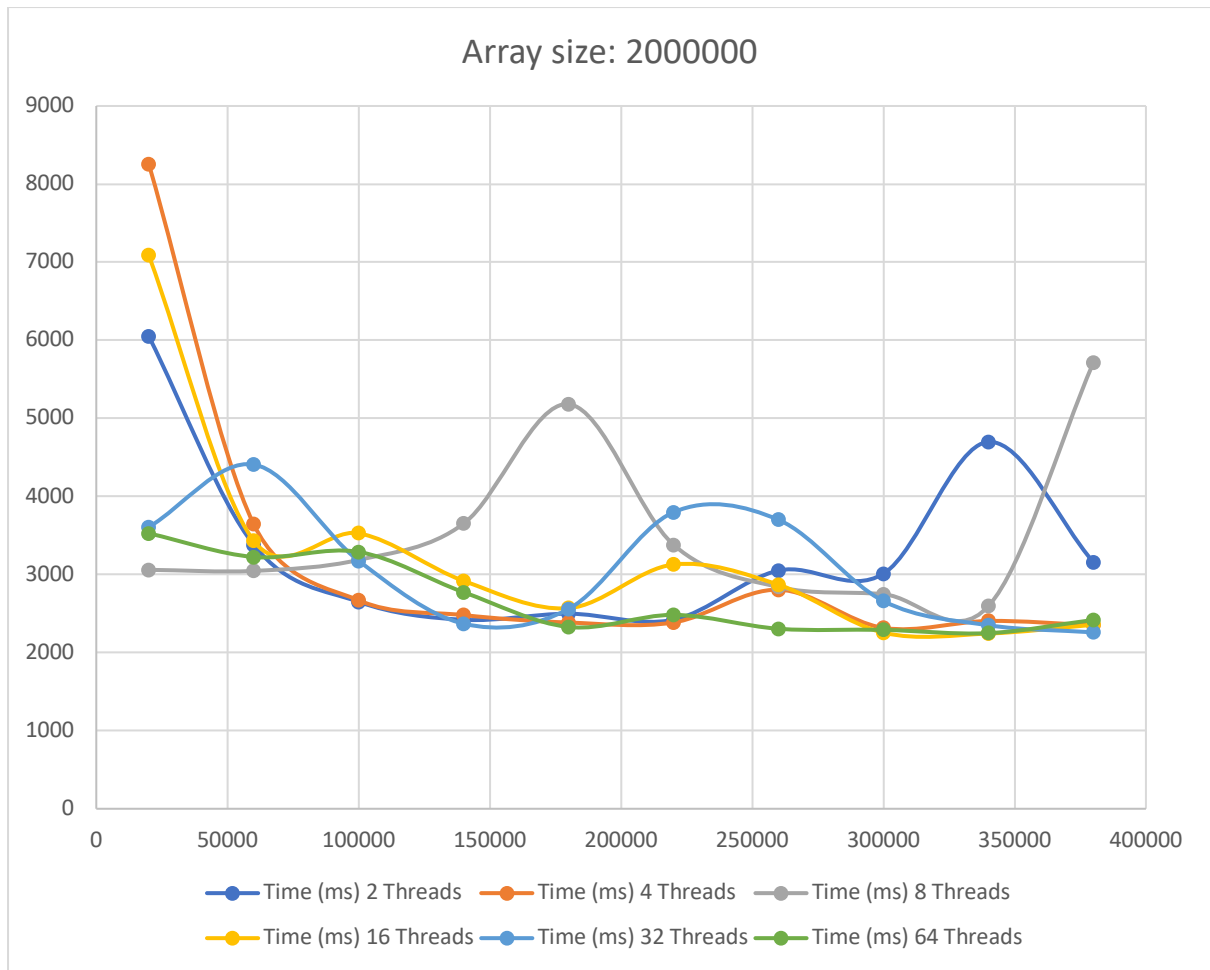
INFO6205 src \ main \ java \ edu \ neu \ coe \ info6205 \ sort \ par \ Main \ main
UF_HWQUPC.java Sort.java ParSort.java Main.java UF_HWQUPC_Test.java
24
25 for (int threads = 2; threads <= 64; threads *= 2) {
26 ForkJoinPool fork = new ForkJoinPool(threads);
27 System.out.println("Degree of parallelism: " + fork.getParallelism());
28 // System.out.println("Degree of parallelism: " + ForkJoinPool.getCommonPoolParallelism());
Run: Main
cutoff: 20000 Time: 4307 ms
cutoff: 220000 Time: 3128 ms
cutoff: 260000 Time: 2869 ms
cutoff: 300000 Time: 2255 ms
cutoff: 340000 Time: 2244 ms
cutoff: 380000 Time: 2356 ms
Degree of parallelism: 32
cutoff: 20000 Time: 3607 ms
cutoff: 60000 Time: 4408 ms
cutoff: 100000 Time: 3175 ms
cutoff: 140000 Time: 2370 ms
cutoff: 180000 Time: 2559 ms
cutoff: 220000 Time: 3798 ms
cutoff: 260000 Time: 3700 ms
cutoff: 300000 Time: 2659 ms
cutoff: 340000 Time: 2350 ms
cutoff: 380000 Time: 2259 ms
Degree of parallelism: 64
cutoff: 20000 Time: 3527 ms
cutoff: 60000 Time: 3224 ms
cutoff: 100000 Time: 3289 ms
cutoff: 140000 Time: 2773 ms
cutoff: 180000 Time: 2328 ms
cutoff: 220000 Time: 2483 ms
cutoff: 260000 Time: 2305 ms
cutoff: 300000 Time: 2291 ms
cutoff: 340000 Time: 2249 ms
cutoff: 380000 Time: 2416 ms
Process finished with exit code 0
Build completed successfully in 3 sec, 382 ms (8 minutes ago)
71:1 LF UTF-8 4 spaces Spring2023

```

Observations:

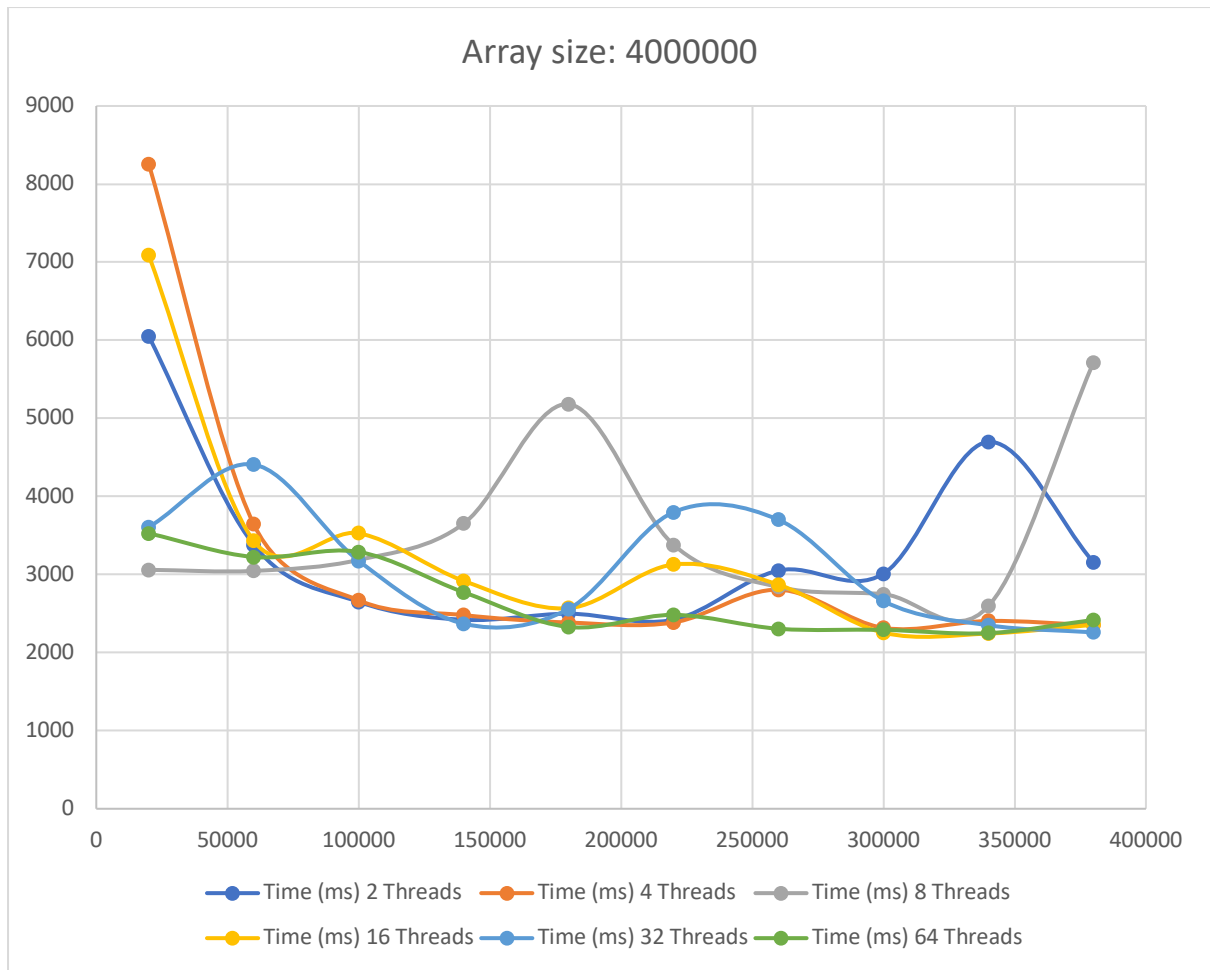
Array size: 2000000

Cut off	Time (ms) 2 Threads	Time (ms) 4 Threads	Time (ms) 8 Threads	Time (ms) 16 Threads	Time (ms) 32 Threads	Time (ms) 64 Threads
20000	1092	1179	1562	1203	795	1636
60000	655	836	1590	1205	618	966
100000	740	563	1368	897	549	585
140000	690	596	1448	589	567	566
180000	668	598	563	555	608	587
220000	660	746	628	557	607	565
260000	793	790	605	560	650	578
300000	785	720	619	582	753	662
340000	798	687	724	730	676	992
380000	1000	816	946	765	952	747



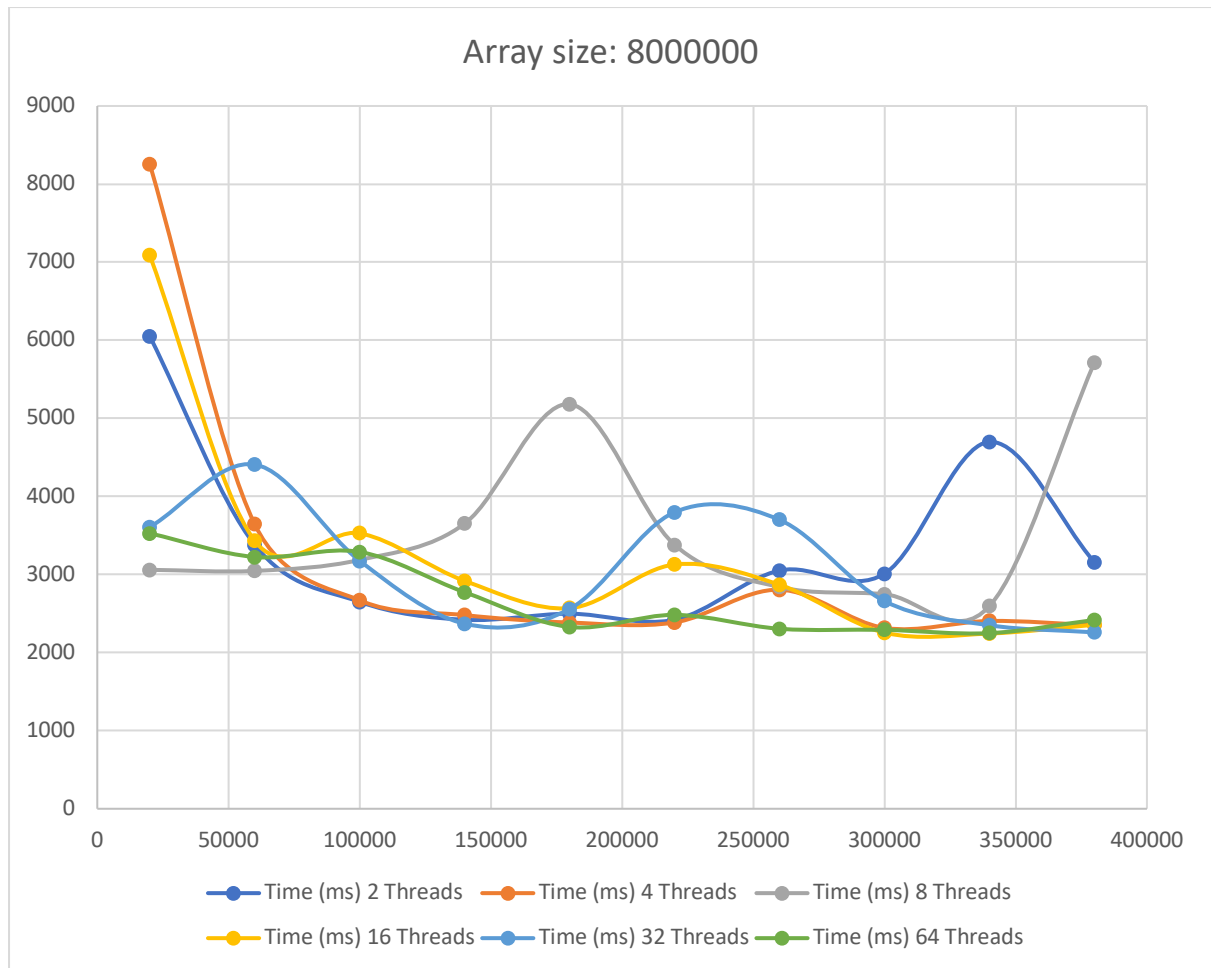
Array size: 4000000

Cut off	Time (ms) 2 Threads	Time (ms) 4 Threads	Time (ms) 8 Threads	Time (ms) 16 Threads	Time (ms) 32 Threads	Time (ms) 64 Threads
20000	1968	1641	2050	1396	1498	1436
60000	1966	1356	3941	1184	1185	1186
100000	1340	1191	1128	1126	1149	1193
140000	1265	1157	1093	1127	1129	1189
180000	1445	1157	1168	1085	1110	1245
220000	1264	1151	1127	1181	1099	1225
260000	1575	1347	1228	1109	1109	1138
300000	2707	2320	1821	1098	1110	1128
340000	2336	2037	2307	1142	1093	1159
380000	1563	1654	1112	1190	1111	1469



Array size: 8000000

Cut off	Time (ms) 2 Threads	Time (ms) 4 Threads	Time (ms) 8 Threads	Time (ms) 16 Threads	Time (ms) 32 Threads	Time (ms) 64 Threads
20000	6051	8257	3060	7088	3607	3527
60000	3380	3643	3046	3431	4408	3224
100000	2650	2670	3187	3528	3175	3289
140000	2420	2480	3652	2918	2370	2773
180000	2498	2385	5181	2569	2559	2328
220000	2421	2386	3380	3128	3798	2483
260000	3051	2805	2840	2869	3700	2305
300000	3011	2316	2748	2255	2659	2291
340000	4698	2405	2596	2244	2350	2249
380000	3157	2351	5716	2356	2259	2416



2. Conclusions:

1. Based on the all the graphs, when the cut off is 2%-3% of input size array, the running time is faster.
2. Based on all graphs number of threads = 64 takes less time than other threads.
3. The time remains similar and no significant difference when the array size increase.

Hence as per the analysis, the ideal cut off value is 2% of the input size array and ideal thread count is 64.