# Assignment 5

# the results of your experiments

## **Implementation:**

We first execute the task for different array sizes. The initial array size is 2 mils and we perform tasks with 2 mils and 4 mils. For different thread counts, we achieve this by externally implementing the thread count and passing it to the ForkJoinPool.java executor, which in turn calls the CompletableFuture.java supplyAsync method with the thread count and supply.

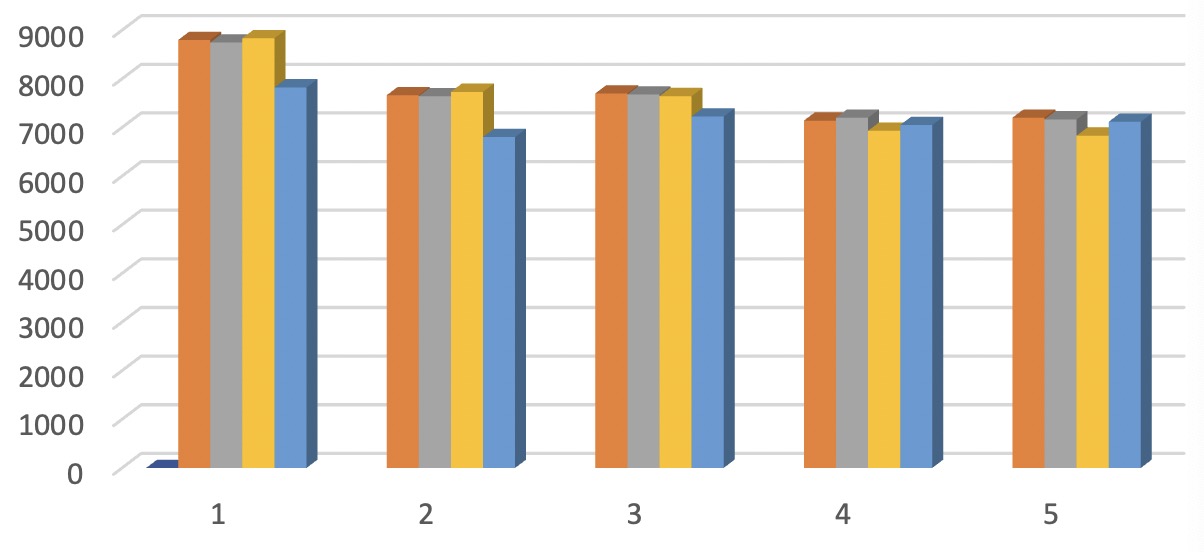
Random random = new Random();

int[] array = new int[2000000];  
 ArrayList<Long> timeList = new ArrayList<>();

## **Execution time against the number of threads:**

For the experiments, we kept the random array cutoff (from 20000 to 100000) and the input size fixed (10 million raw integer) and records the execution time against the number of threads provided in the pool.

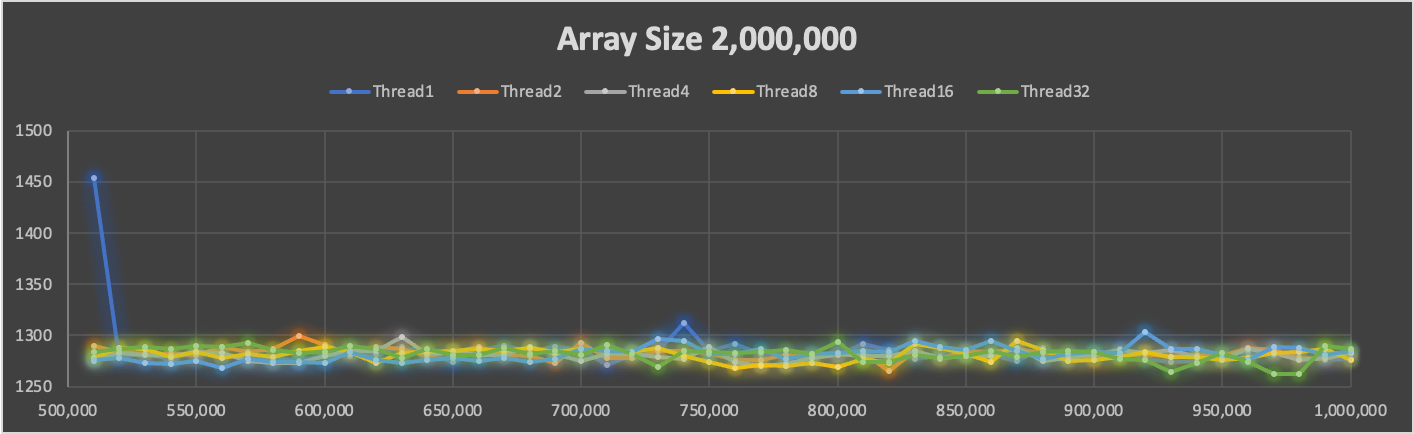
It is observed that the execution time of sorting 10 million primitive integers is almost the same on threads 1, 2, 4performance, while it degrades when the thread pool is increased to 8. However, when the thread pool was further increased to accommodate 16 threads, it showed no significant gain.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number | Number of threads | Time/ | | | | |
| 1 | 1 | 8799 | 7664 | 7700 | 7141 | 7201 |
| 2 | 2 | 8747.5 | 7643 | 7676 | 7200 | 7166 |
| 3 | 4 | 8835.5 | 7732 | 7644 | 6932 | 6832 |
| 4 | 8 | 7823.5 | 6804 | 7226 | 7050 | 7118 |
| 5 | 16 | 7610 | 7245 | 7320 | 7043 | 7085 |
| 6 | cutoff | 20000 | 40000 | 60000 | 80000 | 100000 |

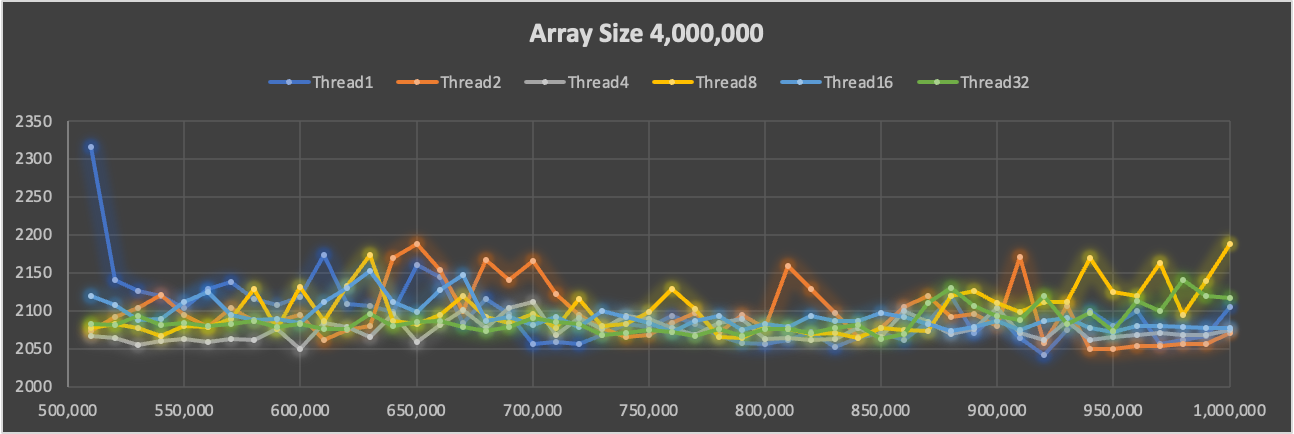
## **Data for different threads and call off for 2 mil elements**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cutoff | Thread1 | Thread2 | Thread4 | Thread8 | Thread16 | Thread32 |
| 510,000 | 1454 | 1290 | 1275 | 1278 | 1276 | 1284 |
| 520,000 | 1279 | 1284 | 1283 | 1288 | 1278 | 1287 |
| 530,000 | 1282 | 1282 | 1281 | 1287 | 1273 | 1289 |
| 540,000 | 1277 | 1282 | 1280 | 1279 | 1272 | 1287 |
| 550,000 | 1284 | 1281 | 1284 | 1284 | 1275 | 1290 |
| 560,000 | 1282 | 1289 | 1283 | 1278 | 1268 | 1289 |
| 570,000 | 1278 | 1284 | 1275 | 1282 | 1277 | 1293 |
| 580,000 | 1278 | 1287 | 1273 | 1279 | 1274 | 1286 |
| 590,000 | 1275 | 1299 | 1273 | 1285 | 1274 | 1283 |
| 600,000 | 1281 | 1291 | 1280 | 1289 | 1273 | 1284 |
| 610,000 | 1280 | 1282 | 1285 | 1284 | 1284 | 1290 |
| 620,000 | 1282 | 1289 | 1285 | 1273 | 1276 | 1287 |
| 630,000 | 1289 | 1287 | 1298 | 1283 | 1273 | 1278 |
| 640,000 | 1283 | 1280 | 1282 | 1284 | 1276 | 1287 |
| 650,000 | 1274 | 1284 | 1284 | 1285 | 1278 | 1281 |
| 660,000 | 1286 | 1289 | 1281 | 1287 | 1275 | 1281 |
| 670,000 | 1290 | 1281 | 1281 | 1285 | 1278 | 1288 |
| 680,000 | 1285 | 1280 | 1289 | 1288 | 1274 | 1282 |
| 690,000 | 1289 | 1273 | 1282 | 1284 | 1277 | 1285 |
| 700,000 | 1287 | 1293 | 1275 | 1287 | 1287 | 1281 |
| 710,000 | 1271 | 1278 | 1282 | 1285 | 1285 | 1291 |
| 720,000 | 1281 | 1278 | 1282 | 1284 | 1284 | 1282 |
| 730,000 | 1285 | 1285 | 1279 | 1288 | 1297 | 1269 |
| 740,000 | 1312 | 1285 | 1277 | 1280 | 1295 | 1285 |
| 750,000 | 1285 | 1283 | 1289 | 1274 | 1282 | 1284 |
| 760,000 | 1292 | 1276 | 1274 | 1268 | 1282 | 1283 |
| 770,000 | 1282 | 1276 | 1270 | 1271 | 1287 | 1284 |
| 780,000 | 1283 | 1281 | 1272 | 1270 | 1277 | 1286 |
| 790,000 | 1280 | 1278 | 1278 | 1273 | 1282 | 1282 |
| 800,000 | 1281 | 1281 | 1282 | 1269 | 1283 | 1294 |
| 810,000 | 1292 | 1283 | 1280 | 1277 | 1285 | 1274 |
| 820,000 | 1286 | 1265 | 1280 | 1273 | 1284 | 1274 |
| 830,000 | 1277 | 1285 | 1285 | 1292 | 1295 | 1281 |
| 840,000 | 1280 | 1277 | 1278 | 1288 | 1289 | 1278 |
| 850,000 | 1285 | 1285 | 1280 | 1282 | 1286 | 1280 |
| 860,000 | 1281 | 1284 | 1280 | 1274 | 1295 | 1285 |
| 870,000 | 1275 | 1287 | 1288 | 1295 | 1285 | 1279 |
| 880,000 | 1288 | 1283 | 1275 | 1286 | 1277 | 1284 |
| 890,000 | 1279 | 1283 | 1278 | 1275 | 1282 | 1285 |
| 900,000 | 1281 | 1277 | 1281 | 1276 | 1284 | 1284 |
| 910,000 | 1287 | 1280 | 1286 | 1279 | 1283 | 1277 |
| 920,000 | 1277 | 1282 | 1283 | 1284 | 1303 | 1276 |
| 930,000 | 1274 | 1279 | 1286 | 1279 | 1287 | 1264 |
| 940,000 | 1275 | 1281 | 1279 | 1279 | 1287 | 1273 |
| 950,000 | 1281 | 1280 | 1277 | 1276 | 1281 | 1283 |
| 960,000 | 1284 | 1288 | 1287 | 1278 | 1276 | 1274 |
| 970,000 | 1281 | 1286 | 1283 | 1283 | 1289 | 1262 |
| 980,000 | 1279 | 1278 | 1276 | 1284 | 1288 | 1262 |
| 990,000 | 1276 | 1277 | 1278 | 1287 | 1281 | 1290 |
| 1,000,000 | 1277 | 1283 | 1283 | 1276 | 1285 | 1287 |
| Avg | 1281 | 1278 | 1276 | 1275 | 1277 | 1278 |
| Max | 1454 | 1299 | 1298 | 1295 | 1303 | 1294 |
| Min | 1270 | 1260 | 1270 | 1260 | 1260 | 1260 |

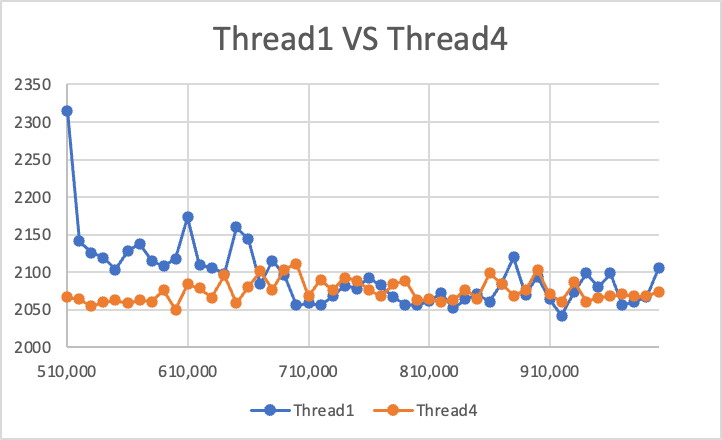


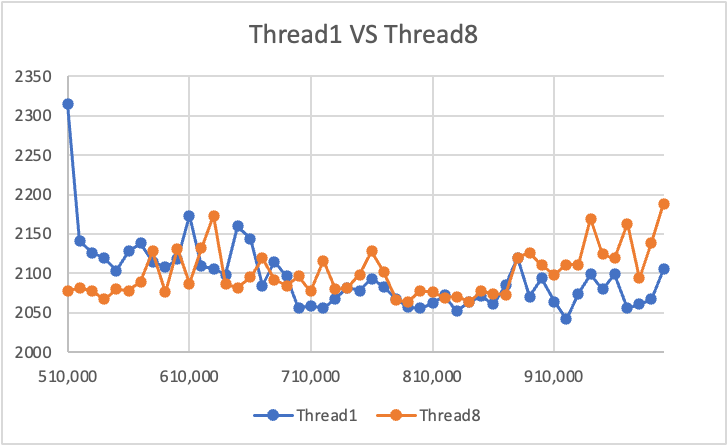
## Data for different threads and call off for 4 mil elements

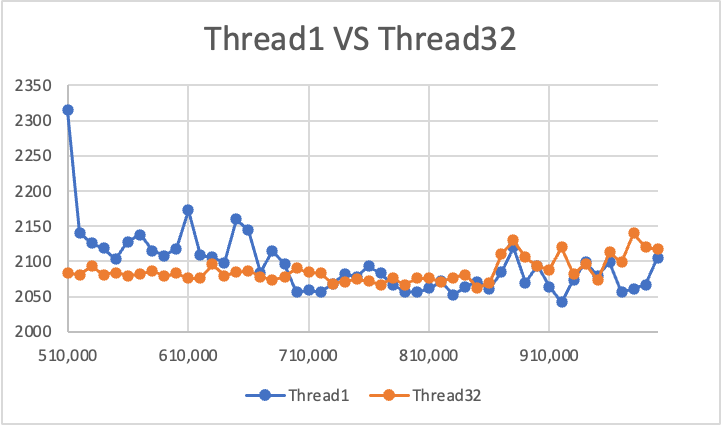
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Cutoff** | **Thread1** | **Thread2** | **Thread4** | **Thread8** | **Thread16** | **Thread32** |
| 510,000 | 1454 | 1290 | 1275 | 1278 | 1276 | 1284 |
| 520,000 | 1279 | 1284 | 1283 | 1288 | 1278 | 1287 |
| 530,000 | 1282 | 1282 | 1281 | 1287 | 1273 | 1289 |
| 540,000 | 1277 | 1282 | 1280 | 1279 | 1272 | 1287 |
| 550,000 | 1284 | 1281 | 1284 | 1284 | 1275 | 1290 |
| 560,000 | 1282 | 1289 | 1283 | 1278 | 1268 | 1289 |
| 570,000 | 1278 | 1284 | 1275 | 1282 | 1277 | 1293 |
| 580,000 | 1278 | 1287 | 1273 | 1279 | 1274 | 1286 |
| 590,000 | 1275 | 1299 | 1273 | 1285 | 1274 | 1283 |
| 600,000 | 1281 | 1291 | 1280 | 1289 | 1273 | 1284 |
| 610,000 | 1280 | 1282 | 1285 | 1284 | 1284 | 1290 |
| 620,000 | 1282 | 1289 | 1285 | 1273 | 1276 | 1287 |
| 630,000 | 1289 | 1287 | 1298 | 1283 | 1273 | 1278 |
| 640,000 | 1283 | 1280 | 1282 | 1284 | 1276 | 1287 |
| 650,000 | 1274 | 1284 | 1284 | 1285 | 1278 | 1281 |
| 660,000 | 1286 | 1289 | 1281 | 1287 | 1275 | 1281 |
| 670,000 | 1290 | 1281 | 1281 | 1285 | 1278 | 1288 |
| 680,000 | 1285 | 1280 | 1289 | 1288 | 1274 | 1282 |
| 690,000 | 1289 | 1273 | 1282 | 1284 | 1277 | 1285 |
| 700,000 | 1287 | 1293 | 1275 | 1287 | 1287 | 1281 |
| 710,000 | 1271 | 1278 | 1282 | 1285 | 1285 | 1291 |
| 720,000 | 1281 | 1278 | 1282 | 1284 | 1284 | 1282 |
| 730,000 | 1285 | 1285 | 1279 | 1288 | 1297 | 1269 |
| 740,000 | 1312 | 1285 | 1277 | 1280 | 1295 | 1285 |
| 750,000 | 1285 | 1283 | 1289 | 1274 | 1282 | 1284 |
| 760,000 | 1292 | 1276 | 1274 | 1268 | 1282 | 1283 |
| 770,000 | 1282 | 1276 | 1270 | 1271 | 1287 | 1284 |
| 780,000 | 1283 | 1281 | 1272 | 1270 | 1277 | 1286 |
| 790,000 | 1280 | 1278 | 1278 | 1273 | 1282 | 1282 |
| 800,000 | 1281 | 1281 | 1282 | 1269 | 1283 | 1294 |
| 810,000 | 1292 | 1283 | 1280 | 1277 | 1285 | 1274 |
| 820,000 | 1286 | 1265 | 1280 | 1273 | 1284 | 1274 |
| 830,000 | 1277 | 1285 | 1285 | 1292 | 1295 | 1281 |
| 840,000 | 1280 | 1277 | 1278 | 1288 | 1289 | 1278 |
| 850,000 | 1285 | 1285 | 1280 | 1282 | 1286 | 1280 |
| 860,000 | 1281 | 1284 | 1280 | 1274 | 1295 | 1285 |
| 870,000 | 1275 | 1287 | 1288 | 1295 | 1285 | 1279 |
| 880,000 | 1288 | 1283 | 1275 | 1286 | 1277 | 1284 |
| 890,000 | 1279 | 1283 | 1278 | 1275 | 1282 | 1285 |
| 900,000 | 1281 | 1277 | 1281 | 1276 | 1284 | 1284 |
| 910,000 | 1287 | 1280 | 1286 | 1279 | 1283 | 1277 |
| 920,000 | 1277 | 1282 | 1283 | 1284 | 1303 | 1276 |
| 930,000 | 1274 | 1279 | 1286 | 1279 | 1287 | 1264 |
| 940,000 | 1275 | 1281 | 1279 | 1279 | 1287 | 1273 |
| 950,000 | 1281 | 1280 | 1277 | 1276 | 1281 | 1283 |
| 960,000 | 1284 | 1288 | 1287 | 1278 | 1276 | 1274 |
| 970,000 | 1281 | 1286 | 1283 | 1283 | 1289 | 1262 |
| 980,000 | 1279 | 1278 | 1276 | 1284 | 1288 | 1262 |
| 990,000 | 1276 | 1277 | 1278 | 1287 | 1281 | 1290 |
| 1,000,000 | 1277 | 1283 | 1283 | 1276 | 1285 | 1287 |
| Avg | 1281 | 1278 | 1276 | 1275 | 1277 | 1278 |
| Max | 1454 | 1299 | 1298 | 1295 | 1303 | 1294 |
| Min | 1270 | 1260 | 1270 | 1260 | 1260 | 1260 |



# **Conclusion**







When it comes to relatively small cutoff sizes relative to the size of the array, the running time does decrease considerably with the number of threads. However, as the cutoff size is not an optimal value with the respect to the size of the array, say the cutoff size is not a multiple of size of the array, we will see performance penalty which is independent to the number of threads. As a result, sometimes we may see that 1 threads have better performance than 4 threads, or even 32 threads.

After 4 threads, we have diminishing returns and benefits. We can infer that the quickest cut-off range is from 700,000 to 850,000.When the cutoff size is too small, single-threaded sort is the in multithreaded sorting algorithms, the overhead of managing so many partitions becomes too large.