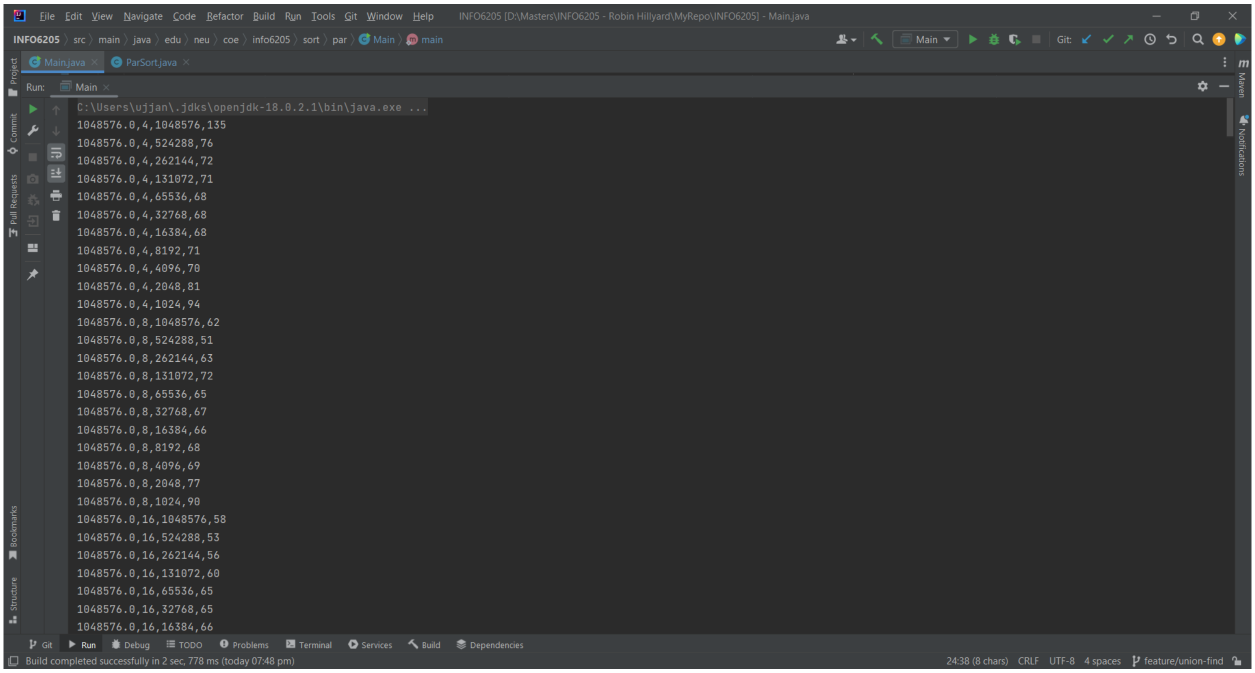
The experiment of parallelisation was to compare timings obtained from comparison of parallel sorting results obtained as a combination of 3 different factors:

a. Array size - Size of the array on which comparison was being done on
b. Degree of parallelisation - Indicates the number of threads being created for the run
c. Cutoff value - The value below which the system sort is run instead of the parallel sort

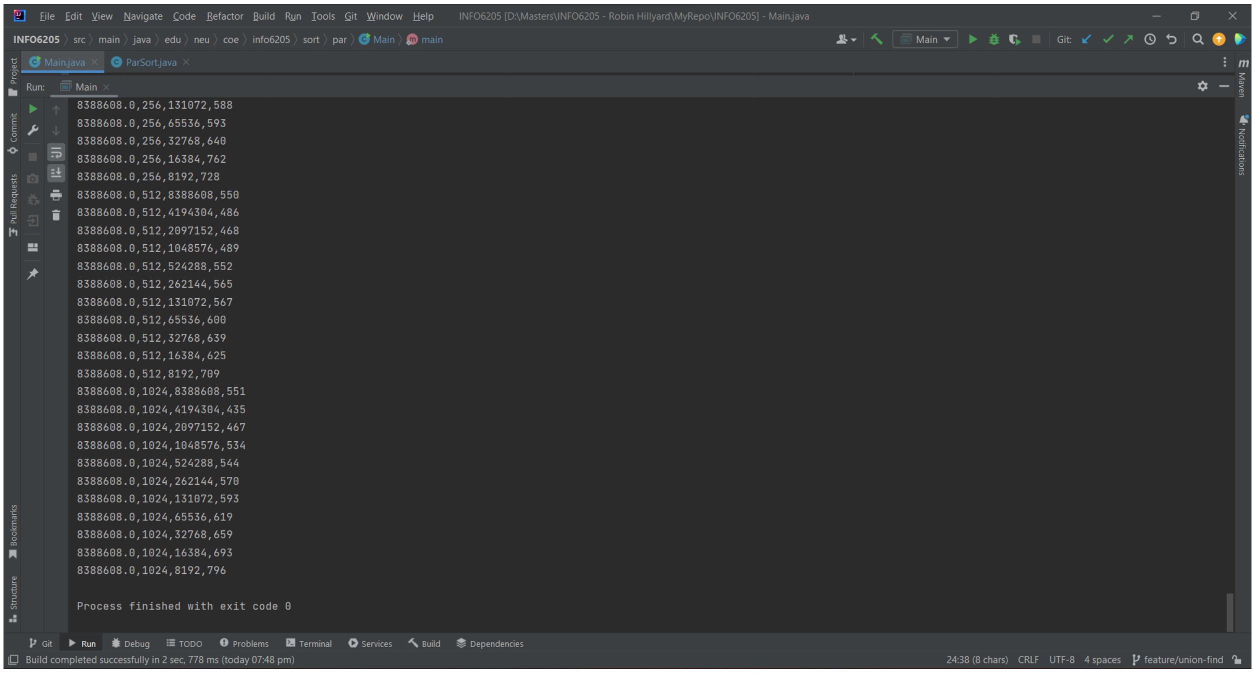
The program functions as a combination of those values. For each size of array being considered, the amount of time taken for the execution of parallel sort for various degrees of parallelism is recorded by keeping a cut off value, below which system sort is applied and a parallel sort in all other cases.

The first and last screenshot of the values obtained are shown below.


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
INFO6205 [D:\Masters\INFO6205 - Robin Hillyard\MyRepo\INFO6205] - Main.java
INFO6205 › src › main › java › edu › neu › coe › info6205 › sort › par › Main › main
Main.java
ParSort.java
Run: Main
Git Run Debug TODO Problems Terminal Services Build Dependencies
24:38 (8 chars) CRLF UTF-8 4 spaces feature/union-find

```
C:\Users\ujjan\.jdks\openjdk-18.0.2.1\bin\java.exe ...
1048576.0,4,1048576,135
1048576.0,4,524288,76
1048576.0,4,262144,72
1048576.0,4,131072,71
1048576.0,4,65536,68
1048576.0,4,32768,68
1048576.0,4,16384,68
1048576.0,4,8192,71
1048576.0,4,4096,70
1048576.0,4,2048,81
1048576.0,4,1024,94
1048576.0,8,1048576,62
1048576.0,8,524288,51
1048576.0,8,262144,63
1048576.0,8,131072,72
1048576.0,8,65536,65
1048576.0,8,32768,67
1048576.0,8,16384,66
1048576.0,8,8192,68
1048576.0,8,4096,69
1048576.0,8,2048,77
1048576.0,8,1024,90
1048576.0,16,1048576,58
1048576.0,16,524288,53
1048576.0,16,262144,56
1048576.0,16,131072,60
1048576.0,16,65536,65
1048576.0,16,32768,65
1048576.0,16,16384,66
```

Build completed successfully in 2 sec, 778 ms (today 07:48 pm)



File Edit View Navigate Code Refactor Build Run Tools Git Window Help
INFO6205 [D:\Masters\INFO6205 - Robin Hillyard\MyRepo\INFO6205] - Main.java
INFO6205 〉 src 〉 main 〉 java 〉 edu 〉 neu 〉 coe 〉 info6205 〉 sort 〉 par 〉 Main 〉 main
Main.java
ParSort.java
Run:
Main

```
8388608.0,256,131072,588
8388608.0,256,65536,593
8388608.0,256,32768,640
8388608.0,256,16384,762
8388608.0,256,8192,728
8388608.0,512,8388608,550
8388608.0,512,4194304,486
8388608.0,512,2097152,468
8388608.0,512,1048576,489
8388608.0,512,524288,552
8388608.0,512,262144,565
8388608.0,512,131072,567
8388608.0,512,65536,600
8388608.0,512,32768,639
8388608.0,512,16384,625
8388608.0,512,8192,709
8388608.0,1024,8388608,551
8388608.0,1024,4194304,435
8388608.0,1024,2097152,467
8388608.0,1024,1048576,534
8388608.0,1024,524288,544
8388608.0,1024,262144,570
8388608.0,1024,131072,593
8388608.0,1024,65536,619
8388608.0,1024,32768,659
8388608.0,1024,16384,693
8388608.0,1024,8192,796

Process finished with exit code 0
```

Build completed successfully in 2 sec, 778 ms (today 07:48 pm)

The table values for all the combinations of sizes, degree of parallelism, cutoff values and time for them is shown. The various array sizes are obtained as a power of 2 from 20 to 23 (inclusive), the degree of parallelism values are obtained as a power of 2 from 2 to 10 (inclusive) and the cutoff values are obtained as a power of 2 from 0 to 10.

Size 1048576		
Degree of Parallelism	**Cutoff size**	**Time**
4	1048576	135
4	524288	76
4	262144	72
4	131072	71
4	65536	68
4	32768	68
4	16384	68
4	8192	71
4	4096	70
4	2048	81
4	1024	94
8	1048576	62
8	524288	51
8	262144	63

8	131072	72
8	65536	65
8	32768	67
8	16384	66
8	8192	68
8	4096	69
8	2048	77
8	1024	90
16	1048576	58
16	524288	53
16	262144	56
16	131072	60
16	65536	65
16	32768	65
16	16384	66
16	8192	69
16	4096	70
16	2048	77
16	1024	86
32	1048576	65
32	524288	56
32	262144	57
32	131072	58
32	65536	59
32	32768	65
32	16384	66
32	8192	70
32	4096	73
32	2048	77
32	1024	86
64	1048576	61
64	524288	54
64	262144	55
64	131072	57
64	65536	59

64	32768	63
64	16384	64
64	8192	70
64	4096	72
64	2048	76
64	1024	89
128	1048576	63
128	524288	53
128	262144	63
128	131072	58
128	65536	59
128	32768	61
128	16384	63
128	8192	65
128	4096	74
128	2048	81
128	1024	89
256	1048576	58
256	524288	52
256	262144	58
256	131072	64
256	65536	60
256	32768	61
256	16384	63
256	8192	65
256	4096	67
256	2048	77
256	1024	91
512	1048576	63
512	524288	52
512	262144	60
512	131072	59
512	65536	62
512	32768	62
512	16384	64

512	8192	66
512	4096	71
512	2048	74
512	1024	96
1024	1048576	58
1024	524288	53
1024	262144	56
1024	131072	60
1024	65536	61
1024	32768	69
1024	16384	65
1024	8192	73
1024	4096	76
1024	2048	80
1024	1024	94

Size 2097152		
Degree of Parallelism	**Cutoff size**	**Time**
4	2097152	122
4	1048576	131
4	524288	151
4	262144	139
4	131072	138
4	65536	133
4	32768	131
4	16384	133
4	8192	138
4	4096	149
4	2048	168
8	2097152	128
8	1048576	109
8	524288	130
8	262144	139

8	131072	131
8	65536	141
8	32768	136
8	16384	138
8	8192	142
8	4096	152
8	2048	166
16	2097152	121
16	1048576	106
16	524288	119
16	262144	121
16	131072	138
16	65536	135
16	32768	135
16	16384	137
16	8192	140
16	4096	152
16	2048	170
32	2097152	123
32	1048576	103
32	524288	116
32	262144	120
32	131072	121
32	65536	127
32	32768	134
32	16384	137
32	8192	143
32	4096	155
32	2048	168
64	2097152	129
64	1048576	107
64	524288	112
64	262144	120
64	131072	118
64	65536	123

64	32768	131
64	16384	138
64	8192	146
64	4096	152
64	2048	169
128	2097152	123
128	1048576	110
128	524288	112
128	262144	117
128	131072	127
128	65536	124
128	32768	134
128	16384	131
128	8192	145
128	4096	155
128	2048	170
256	2097152	129
256	1048576	106
256	524288	115
256	262144	121
256	131072	121
256	65536	129
256	32768	128
256	16384	133
256	8192	135
256	4096	153
256	2048	169
512	2097152	125
512	1048576	107
512	524288	130
512	262144	123
512	131072	122
512	65536	122
512	32768	136
512	16384	135

512	8192	143
512	4096	143
512	2048	163
1024	2097152	124
1024	1048576	102
1024	524288	114
1024	262144	123
1024	131072	125
1024	65536	141
1024	32768	132
1024	16384	139
1024	8192	149
1024	4096	151
1024	2048	163

Size 4194304		
Degree of Parallelism	**Cutoff size**	**Time**
4	4194304	244
4	2097152	262
4	1048576	303
4	524288	257
4	262144	263
4	131072	271
4	65536	282
4	32768	278
4	16384	288
4	8192	301
4	4096	320
8	4194304	249
8	2097152	213
8	1048576	250
8	524288	273
8	262144	251
8	131072	270

8	65536	266
8	32768	284
8	16384	292
8	8192	300
8	4096	339
16	4194304	293
16	2097152	244
16	1048576	231
16	524288	255
16	262144	274
16	131072	287
16	65536	271
16	32768	284
16	16384	295
16	8192	298
16	4096	339
32	4194304	242
32	2097152	207
32	1048576	224
32	524288	246
32	262144	236
32	131072	273
32	65536	293
32	32768	288
32	16384	286
32	8192	306
32	4096	330
64	4194304	288
64	2097152	217
64	1048576	298
64	524288	258
64	262144	250
64	131072	253
64	65536	265
64	32768	289

64	16384	292
64	8192	309
64	4096	337
128	4194304	263
128	2097152	220
128	1048576	240
128	524288	241
128	262144	250
128	131072	252
128	65536	263
128	32768	269
128	16384	297
128	8192	306
128	4096	333
256	4194304	263
256	2097152	221
256	1048576	232
256	524288	244
256	262144	246
256	131072	250
256	65536	263
256	32768	270
256	16384	282
256	8192	314
256	4096	337
512	4194304	265
512	2097152	220
512	1048576	237
512	524288	239
512	262144	253
512	131072	263
512	65536	275
512	32768	278
512	16384	288
512	8192	300

512	4096	337
1024	4194304	264
1024	2097152	222
1024	1048576	232
1024	524288	246
1024	262144	253
1024	131072	279
1024	65536	273
1024	32768	278
1024	16384	294
1024	8192	303
1024	4096	315

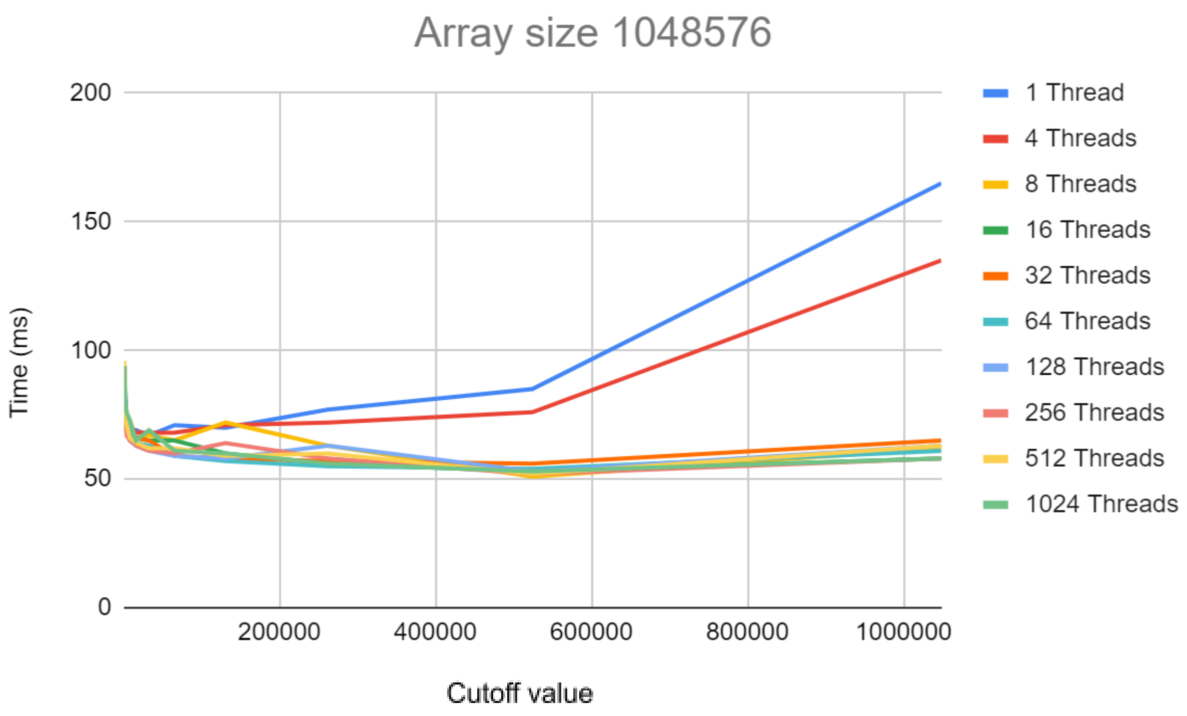
Size 8388608		
Degree of Parallelism	**Cutoff size**	**Time**
4	8388608	549
4	4194304	599
4	2097152	653
4	1048576	593
4	524288	531
4	262144	563
4	131072	559
4	65536	637
4	32768	606
4	16384	635
4	8192	662
8	8388608	548
8	4194304	445
8	2097152	527
8	1048576	595
8	524288	579
8	262144	582
8	131072	584
8	65536	597

8	32768	634
8	16384	651
8	8192	695
16	8388608	535
16	4194304	444
16	2097152	451
16	1048576	495
16	524288	649
16	262144	562
16	131072	571
16	65536	610
16	32768	607
16	16384	628
16	8192	683
32	8388608	531
32	4194304	440
32	2097152	464
32	1048576	567
32	524288	657
32	262144	570
32	131072	588
32	65536	633
32	32768	625
32	16384	665
32	8192	697
64	8388608	551
64	4194304	484
64	2097152	458
64	1048576	501
64	524288	593
64	262144	634
64	131072	610
64	65536	618
64	32768	642
64	16384	651

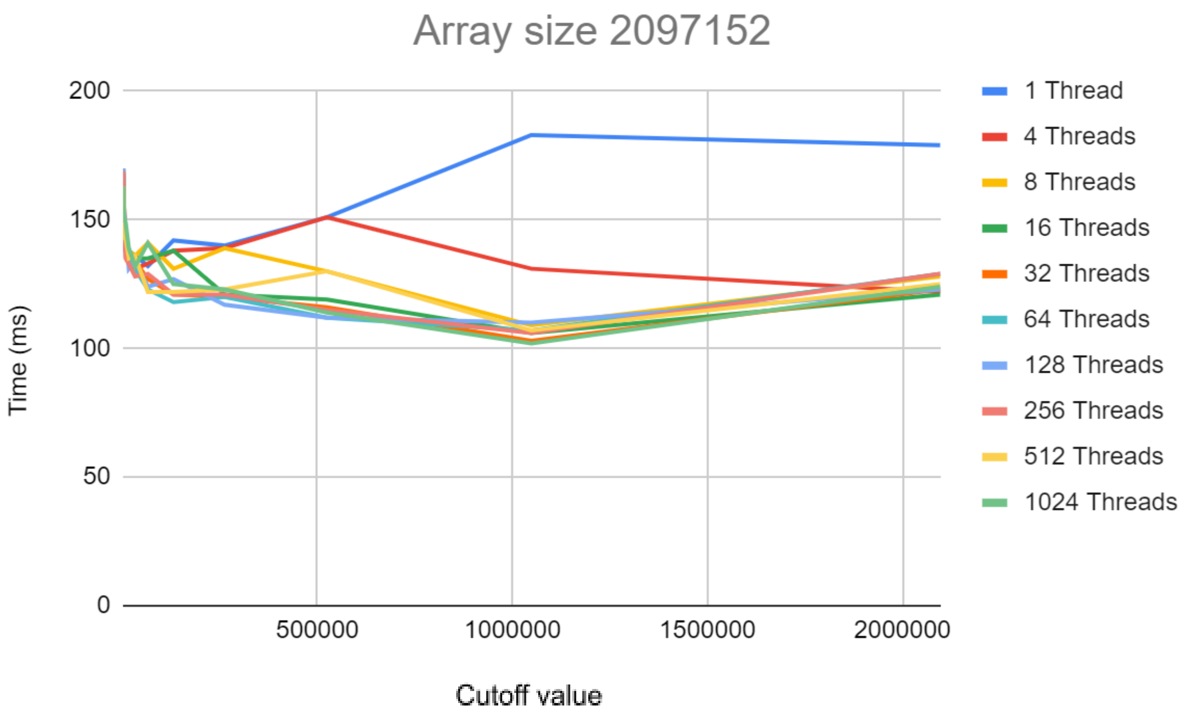
64	8192	678
128	8388608	561
128	4194304	496
128	2097152	545
128	1048576	545
128	524288	499
128	262144	602
128	131072	682
128	65536	639
128	32768	636
128	16384	677
128	8192	699
256	8388608	588
256	4194304	478
256	2097152	459
256	1048576	537
256	524288	499
256	262144	537
256	131072	588
256	65536	593
256	32768	640
256	16384	762
256	8192	728
512	8388608	550
512	4194304	486
512	2097152	468
512	1048576	489
512	524288	552
512	262144	565
512	131072	567
512	65536	600
512	32768	639
512	16384	625
512	8192	709
1024	8388608	551

1024	4194304	435
1024	2097152	467
1024	1048576	534
1024	524288	544
1024	262144	570
1024	131072	593
1024	65536	619
1024	32768	659
1024	16384	693
1024	8192	796

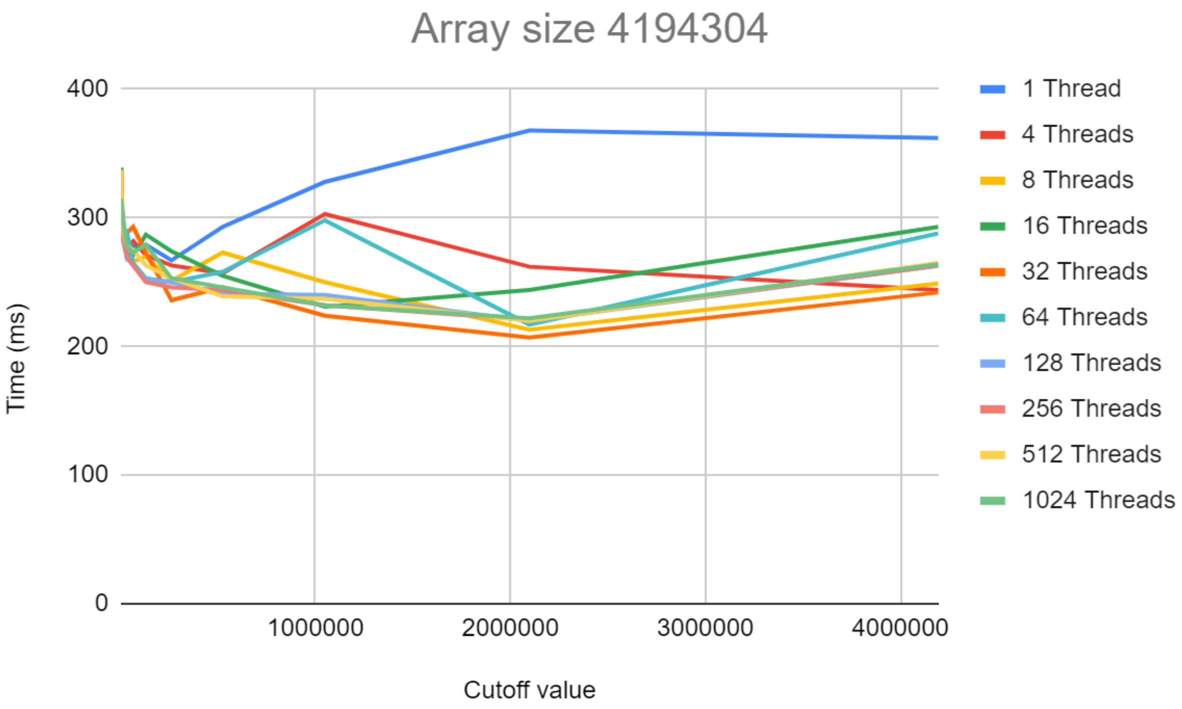
The time analysis for each array size is as follows with analysis based on each graph.



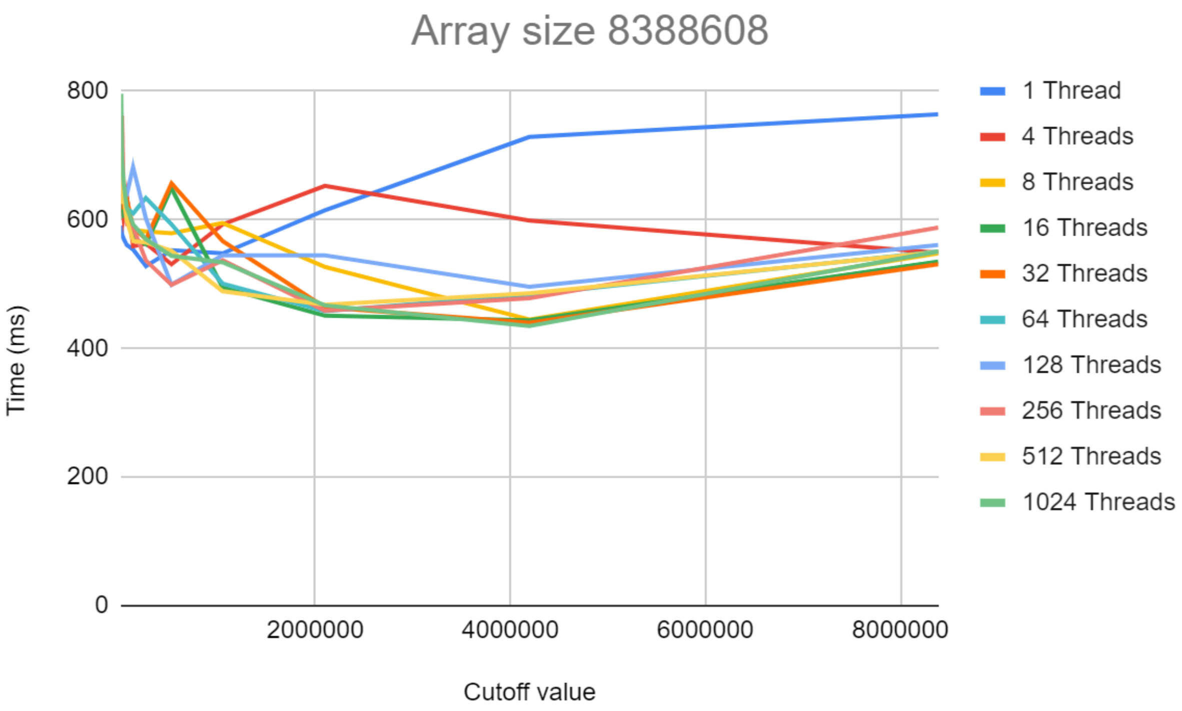
In the above graph, we can see that the general trend is initial increase, followed by a decrease in time followed by a slow increase as the size of the array increases. The only exception to this behavior is that of 4 threads that tend to increase rapidly instead of slowly increasing. This shows that increasing the number of threads has a huge improvement in performance. Just increasing by a factor of 2. For 1024 threads, there is an initial time spike after which it gives a very smooth performance for mid and large values. For small values, of n, all the remaining threads seem to perform well.



In the above graph, we can see that the general trend is initial increase, followed by a decrease in time followed by a slow increase as the size of the array increases. The only exception to this behavior is that of 4 threads that show similar behavior but vary by a much larger magnitude. The one that varies 2nd most is the 8 threads one but at around the 1000000 mark, it generalizes like the rest of the threads. This again shows that increasing the cutoff value has a huge improvement in performance for mid values. We can see that for all threads, in general we can see large levels of variations for small cutoff values. This smooths out for mid level range.



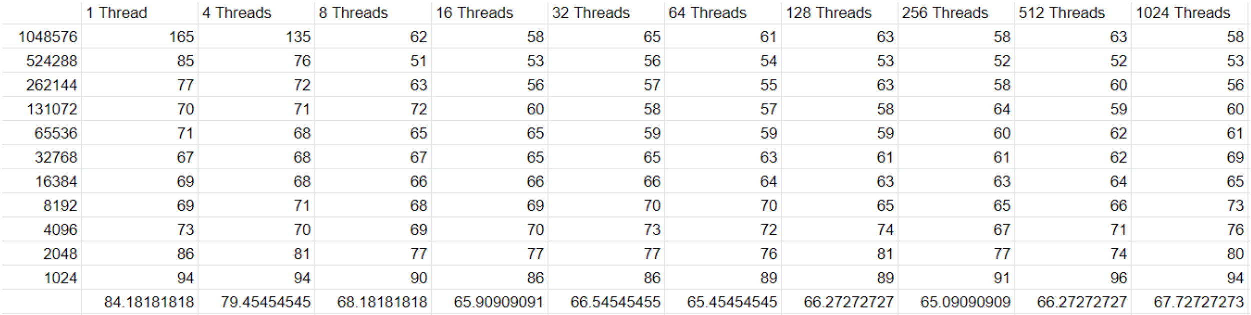
In the above graph, we can see that the previously maintained general trend gives way. The 4 and 8 thread values tend to increase rapidly for small and mid values and start leveling off by the time it reaches the 2000000 cutoff value. We also notice that for small thread values there is significant variation for small cutoff values but eventually smooths off.



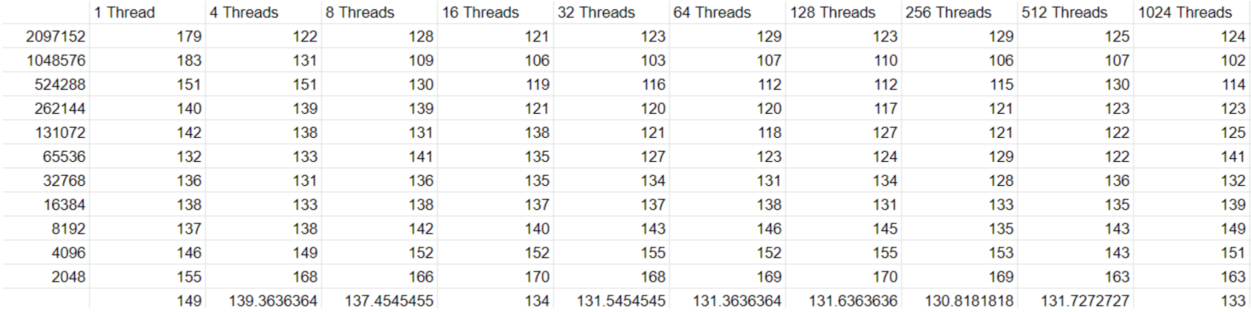
In the above graph, we can see that for almost all threads for low cutoff regions there is a huge variation. The lower regions have spikes in values of time that do not tend to smooth out quickly with increase in cutoff value. For these large sizes of arrays, we notice that the increase in the number of threads ends up playing a very significant role along with the cutoff values.

From all the graphs that we observed, both the cutoff values and the number of threads play a huge role to ensure that less time is taken to sort an array.
The cutoff value plays a huge role for medium(relatively) array sizes and the increase in the number of threads play a huge role for large array sizes.
To be benefitted, a good combination of cutoff values usually at around the middle region and the large number of threads will ensure a generally good level of performance that is less amount of time taken to sort the array.

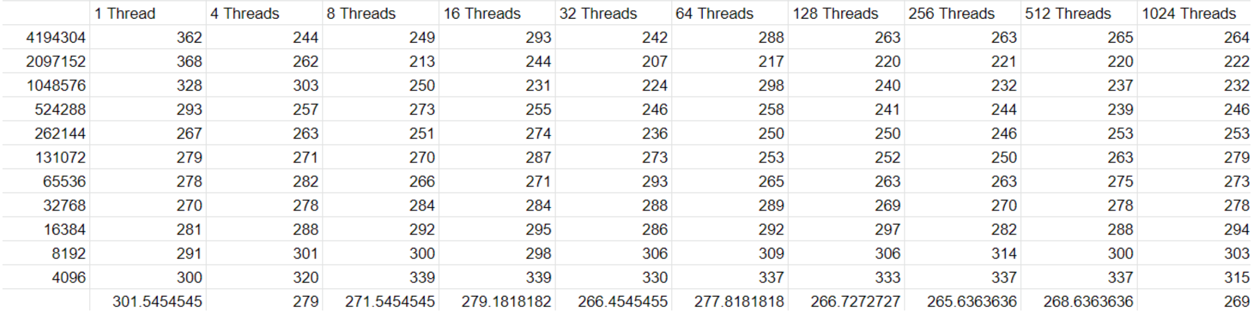
The table with average values are shown for the corresponding sizes below:



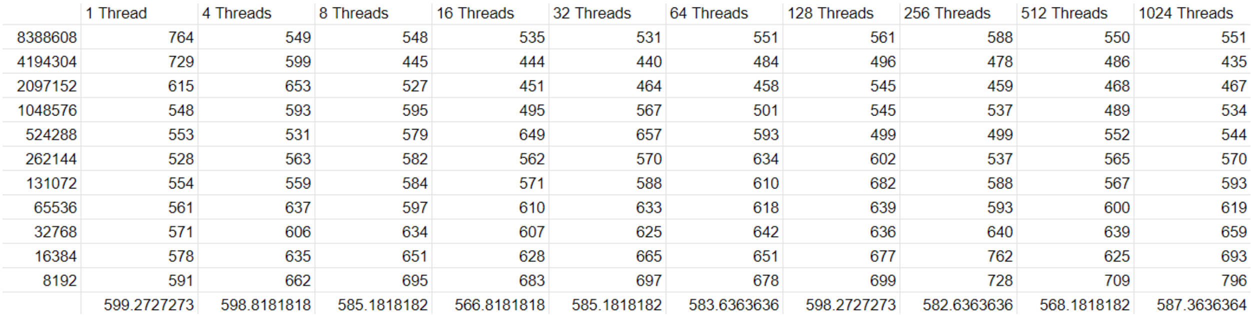
	1 Thread	4 Threads	8 Threads	16 Threads	32 Threads	64 Threads	128 Threads	256 Threads	512 Threads	1024 Threads
1048576	165	135	62	58	65	61	63	58	63	58
524288	85	76	51	53	56	54	53	52	52	53
262144	77	72	63	56	57	55	63	58	60	56
131072	70	71	72	60	58	57	58	64	59	60
65536	71	68	65	65	59	59	59	60	62	61
32768	67	68	67	65	65	63	61	61	62	69
16384	69	68	66	66	66	64	63	63	64	65
8192	69	71	68	69	70	70	65	65	66	73
4096	73	70	69	70	73	72	74	67	71	76
2048	86	81	77	77	77	76	81	77	74	80
1024	94	94	90	86	86	89	89	91	96	94
	84.18181818	79.45454545	68.18181818	65.90909091	66.54545455	65.45454545	66.27272727	65.09090909	66.27272727	67.72727273



	1 Thread	4 Threads	8 Threads	16 Threads	32 Threads	64 Threads	128 Threads	256 Threads	512 Threads	1024 Threads
2097152	179	122	128	121	123	129	123	129	125	124
1048576	183	131	109	106	103	107	110	106	107	102
524288	151	151	130	119	116	112	112	115	130	114
262144	140	139	139	121	120	120	117	121	123	123
131072	142	138	131	138	121	118	127	121	122	125
65536	132	133	141	135	127	123	124	129	122	141
32768	136	131	136	135	134	131	134	128	136	132
16384	138	133	138	137	137	138	131	133	135	139
8192	137	138	142	140	143	146	145	135	143	149
4096	146	149	152	152	155	152	155	153	143	151
2048	155	168	166	170	168	169	170	169	163	163
	149	139.3636364	137.4545455	134	131.5454545	131.3636364	131.6363636	130.8181818	131.7272727	133



	1 Thread	4 Threads	8 Threads	16 Threads	32 Threads	64 Threads	128 Threads	256 Threads	512 Threads	1024 Threads
4194304	362	244	249	293	242	288	263	263	265	264
2097152	368	262	213	244	207	217	220	221	220	222
1048576	328	303	250	231	224	298	240	232	237	232
524288	293	257	273	255	246	258	241	244	239	246
262144	267	263	251	274	236	250	250	246	253	253
131072	279	271	270	287	273	253	252	250	263	279
65536	278	282	266	271	293	265	263	263	275	273
32768	270	278	284	284	288	289	269	270	278	278
16384	281	288	292	295	286	292	297	282	288	294
8192	291	301	300	298	306	309	306	314	300	303
4096	300	320	339	339	330	337	333	337	337	315
	301.5454545	279	271.5454545	279.1818182	266.4545455	277.8181818	266.7272727	265.6363636	268.6363636	269



	1 Thread	4 Threads	8 Threads	16 Threads	32 Threads	64 Threads	128 Threads	256 Threads	512 Threads	1024 Threads
8388608	764	549	548	535	531	551	561	588	550	551
4194304	729	599	445	444	440	484	496	478	486	435
2097152	615	653	527	451	464	458	545	459	468	467
1048576	548	593	595	495	567	501	545	537	489	534
524288	553	531	579	649	657	593	499	499	552	544
262144	528	563	582	562	570	634	602	537	565	570
131072	554	559	584	571	588	610	682	588	567	593
65536	561	637	597	610	633	618	639	593	600	619
32768	571	606	634	607	625	642	636	640	639	659
16384	578	635	651	628	665	651	677	762	625	693
8192	591	662	695	683	697	678	699	728	709	796
	599.2727273	598.8181818	585.1818182	566.8181818	585.1818182	583.6363636	598.2727273	582.6363636	568.1818182	587.3636364

Conclusion: For a laptop that has a core i3 10th gen processor, with 2 physical cores and 2 virtual cores, the analysis has been done. The number of threads that will be a good metric for any size array is 8 threads, for the cutoff value of about 1/4th the length of the array to get the least amount of time. Threads beyond 8 seem to produce identical results as that of 8.