Keyur Ashokbhai Barot (NU ID: 001568664)

Program Structure and Algorithms
Fall 2021
Assignment No. 5 (Parallel Sorting)

I. Task:

- 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.

- (Part 1) A cut-off (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 cut-off. If there are fewer elements to sort than the cut-off, then you should use the system sort instead.

- (Part 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).

- (Part 3) An appropriate combination of these.

- Show the results of your experiments and draws a conclusion (or more) about the efficacy of this method of parallelizing sort.

- 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 cut-off schemes.

II. Console Output:

```
Size of Array: 50000
Degree of parallelism: 2
cutoff : 5000 10times Time:196ms
cutoff : 10000 10times Time:82ms
cutoff : 15000 10times Time:36ms
cutoff : 20000 10times Time:37ms
cutoff : 25000 10times Time:35ms
cutoff : 30000 10times Time:70ms
cutoff : 35000 10times Time:68ms
cutoff : 40000 10times Time:26ms
cutoff : 45000 10times Time:25ms
cutoff : 50000 10times Time:25ms
Degree of parallelism: 4
cutoff : 5000 10times Time:76ms
```

cutoff : 10000 10times Time:35ms
cutoff : 15000 10times Time:27ms
cutoff : 20000 10times Time:31ms
cutoff : 25000 10times Time:35ms
cutoff : 30000 10times Time:40ms
cutoff : 35000 10times Time:54ms
cutoff : 40000 10times Time:26ms
cutoff : 45000 10times Time:25ms
cutoff : 50000 10times Time:29ms
Degree of parallelism: 8
cutoff : 5000 10times Time:37ms
cutoff : 10000 10times Time:25ms
cutoff : 15000 10times Time:13ms
cutoff : 20000 10times Time:31ms
cutoff : 25000 10times Time:21ms
cutoff : 30000 10times Time:16ms
cutoff : 35000 10times Time:31ms
cutoff : 40000 10times Time:31ms
cutoff : 45000 10times Time:23ms
cutoff : 50000 10times Time:31ms
Degree of parallelism: 16
cutoff : 5000 10times Time:15ms
cutoff : 10000 10times Time:38ms
cutoff : 15000 10times Time:16ms
cutoff : 20000 10times Time:15ms
cutoff : 25000 10times Time:32ms
cutoff : 30000 10times Time:31ms
cutoff : 35000 10times Time:22ms
cutoff : 40000 10times Time:31ms
cutoff : 45000 10times Time:32ms
cutoff : 50000 10times Time:22ms
Degree of parallelism: 32

cutoff : 5000 cutoff : 10000 cutoff : 15000 cutoff : 20000 cutoff : 25000 cutoff : 30000 cutoff : 35000 cutoff : 40000 cutoff : 45000 cutoff : 50000	10times Time:31ms 10times Time:23ms 10times Time:23ms 10times Time:24ms 10times Time:24ms 10times Time:28ms 10times Time:27ms 10times Time:27ms 10times Time:24ms 10times Time:26ms
Degree of parallelism: 64	
cutoff : 5000 cutoff : 10000	10times Time:33ms 10times Time:20ms 10times Time:24ms

cutoff : 15000 cutoff : 20000 cutoff : 25000 cutoff : 30000 cutoff : 35000 cutoff : 40000 cutoff : 45000 cutoff : 50000	10times Time:24ms 10times Time:23ms 10times Time:28ms 10times Time:28ms 10times Time:26ms 10times Time:28ms 10times Time:29ms

Process finished with exit code 0

Size of Array: 100000
Degree of parallelism: 2

cutoff : 5000 cutoff : 10000 cutoff : 15000 cutoff : 20000 cutoff : 25000 cutoff : 30000 cutoff : 35000 cutoff : 40000 cutoff : 45000 cutoff : 50000	10times Time:332ms 10times Time:178ms 10times Time:69ms 10times Time:69ms 10times Time:62ms 10times Time:101ms 10times Time:100ms 10times Time:53ms 10times Time:54ms 10times Time:62ms
Degree of parallelism: 4	
cutoff : 5000 cutoff : 10000 cutoff : 15000 cutoff : 20000 cutoff : 25000 cutoff : 30000 cutoff : 35000 cutoff : 40000 cutoff : 45000	10times Time:85ms 10times Time:53ms 10times Time:47ms 10times Time:54ms 10times Time:46ms 10times Time:47ms 10times Time:38ms 10times Time:47ms 10times Time:53ms

cutoff : 50000	10times Time:48ms
Degree of parallelism: 8	
cutoff : 5000 cutoff : 10000 cutoff : 15000 cutoff : 20000 cutoff : 25000 cutoff : 30000 cutoff : 35000 cutoff : 40000 cutoff : 45000 cutoff : 50000	10times Time:52ms 10times Time:47ms 10times Time:38ms 10times Time:46ms 10times Time:54ms 10times Time:47ms 10times Time:37ms 10times Time:47ms 10times Time:54ms 10times Time:47ms

Degree of parallelism: 16	
cutoff : 5000 cutoff : 10000 cutoff : 15000 cutoff : 20000 cutoff : 25000 cutoff : 30000 cutoff : 35000 cutoff : 40000 cutoff : 45000 cutoff : 50000	10times Time:69ms 10times Time:31ms 10times Time:53ms 10times Time:31ms 10times Time:47ms 10times Time:69ms 10times Time:54ms 10times Time:47ms 10times Time:46ms 10times Time:38ms
Degree of parallelism: 32	
cutoff : 5000 cutoff : 10000 cutoff : 15000 cutoff : 20000 cutoff : 25000 cutoff : 30000 cutoff : 35000 cutoff : 40000 cutoff : 45000 cutoff : 50000	10times Time:47ms 10times Time:53ms 10times Time:47ms 10times Time:38ms 10times Time:47ms 10times Time:38ms 10times Time:47ms 10times Time:53ms 10times Time:47ms 10times Time:53ms
Degree of parallelism: 64	
cutoff : 5000 cutoff : 10000 cutoff : 15000 cutoff : 20000 cutoff : 25000 cutoff : 30000 cutoff : 35000 cutoff : 40000 cutoff : 45000 cutoff : 50000	10times Time:63ms 10times Time:47ms 10times Time:51ms 10times Time:42ms 10times Time:48ms 10times Time:46ms 10times Time:47ms 10times Time:40ms 10times Time:50ms 10times Time:50ms

Process finished with exit code 0

Size of Array: 200000
Degree of parallelism: 2

cutoff : 5000 cutoff : 10000 cutoff : 15000 cutoff : 20000 cutoff : 25000 cutoff : 30000 cutoff : 35000 cutoff : 40000	10times Time:564ms 10times Time:416ms 10times Time:116ms 10times Time:207ms 10times Time:100ms 10times Time:116ms 10times Time:116ms 10times Time:116ms

cutoff : 45000	10times Time:100ms
cutoff : 50000	10times Time:116ms
Degree of parallelism: 4	
cutoff : 5000	10times Time:116ms
cutoff : 10000	10times Time:110ms
cutoff : 15000	10times Time:89ms
cutoff : 20000	10times Time:91ms
cutoff : 25000	10times Time:90ms
cutoff : 30000	10times Time:96ms
cutoff : 35000	10times Time:81ms
cutoff : 40000	10times Time:100ms
cutoff : 45000	10times Time:100ms
cutoff : 50000	10times Time:85ms
Degree of parallelism: 8	
cutoff : 5000	10times Time:116ms
cutoff : 10000	10times Time:84ms
cutoff : 15000	10times Time:85ms
cutoff : 20000	10times Time:78ms
cutoff : 25000	10times Time:85ms
cutoff : 30000	10times Time:84ms
cutoff : 35000	10times Time:69ms
cutoff : 40000	10times Time:93ms
cutoff : 45000	10times Time:92ms
cutoff : 50000	10times Time:82ms
Degree of parallelism: 16	
cutoff : 5000	10times Time:117ms
cutoff : 10000	10times Time:84ms
cutoff : 15000	10times Time:87ms
cutoff : 20000	10times Time:75ms
cutoff : 25000	10times Time:100ms
cutoff : 30000	10times Time:85ms
cutoff : 35000	10times Time:85ms
cutoff : 40000	10times Time:78ms
cutoff : 45000	10times Time:85ms
cutoff : 50000	10times Time:69ms

Degree of parallelism: 32

cutoff : 5000	10times Time:115ms
cutoff : 10000	10times Time:85ms
cutoff : 15000	10times Time:85ms
cutoff : 20000	10times Time:84ms
cutoff : 25000	10times Time:89ms
cutoff : 30000	10times Time:83ms
cutoff : 35000	10times Time:80ms
cutoff : 40000	10times Time:78ms

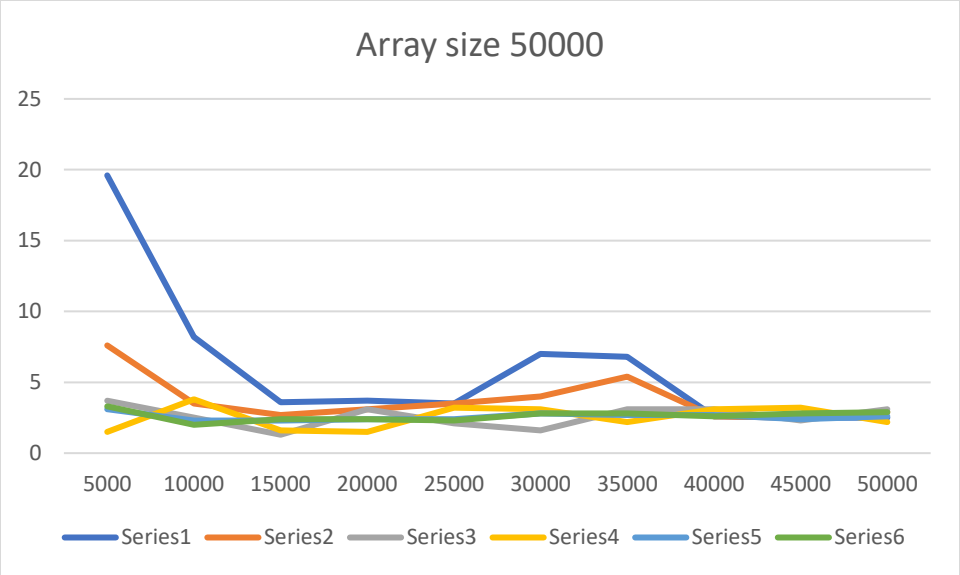
cutoff : 45000 cutoff : 50000	10times Time:87ms 10times Time:83ms
Degree of parallelism: 64	
cutoff : 5000 cutoff : 10000 cutoff : 15000 cutoff : 20000 cutoff : 25000 cutoff : 30000 cutoff : 35000 cutoff : 40000 cutoff : 45000 cutoff : 50000	10times Time:83ms 10times Time:94ms 10times Time:85ms 10times Time:84ms 10times Time:85ms 10times Time:84ms 10times Time:85ms 10times Time:78ms 10times Time:69ms 10times Time:85ms

Process finished with exit code 0

III. Comparison:

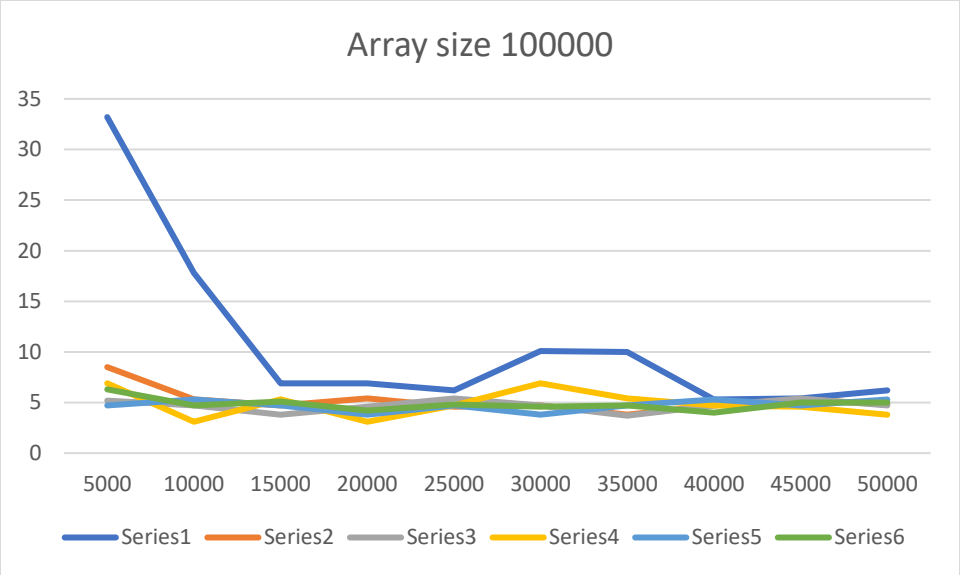
When the algorithm was tested against various array sizes at various cut-off times the following results were to be seen:

Array size = 50000						
Cut-off / Threads	**2**	**4**	**8**	**16**	**32**	**64**
5000	19.6	7.6	3.7	1.5	3.1	3.3
10000	8.2	3.5	2.5	3.8	2.3	2
15000	3.6	2.7	1.3	1.6	2.3	2.4
20000	3.7	3.1	3.1	1.5	2.4	2.4
25000	3.5	3.5	2.1	3.2	2.4	2.3
30000	7	4	1.6	3.1	2.8	2.8
35000	6.8	5.4	3.1	2.2	2.7	2.8
40000	2.6	2.6	3.1	3.1	2.7	2.6
45000	2.5	2.5	2.3	3.2	2.4	2.8
50000	2.5	2.9	3.1	2.2	2.6	2.9



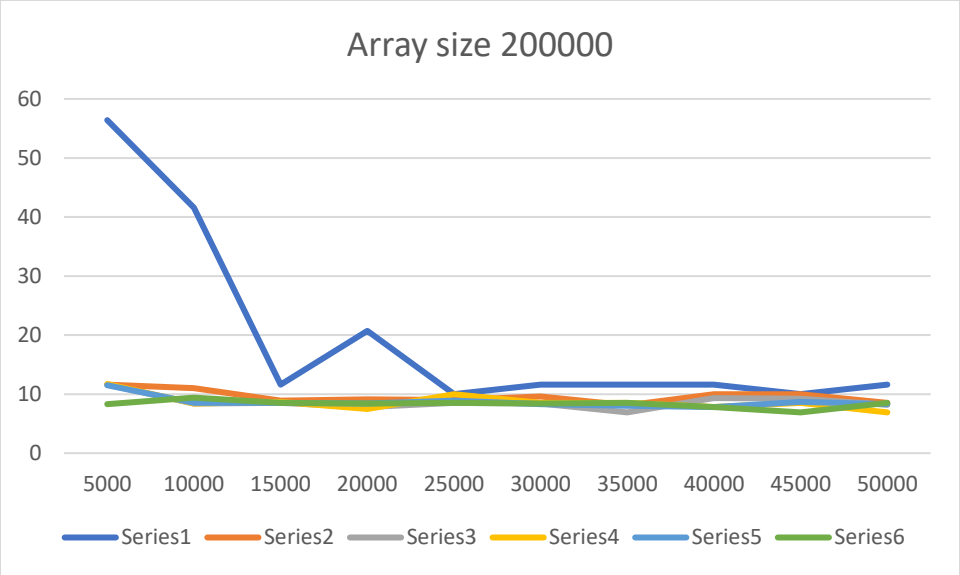
	Series1	Series2	Series3	Series4	Series5	Series6
No. of Threads	2	4	8	16	32	64

Array size = 100000						
Cut-off / Threads	**2**	**4**	**8**	**16**	**32**	**64**
5000	33.2	8.5	5.2	6.9	4.7	6.3
10000	17.8	5.3	4.7	3.1	5.3	4.7
15000	6.9	4.7	3.8	5.3	4.7	5.1
20000	6.9	5.4	4.6	3.1	3.8	4.2
25000	6.2	4.6	5.4	4.7	4.7	4.8
30000	10.1	4.7	4.7	6.9	3.8	4.6
35000	10	3.8	3.7	5.4	4.7	4.7
40000	5.3	4.7	4.7	4.7	5.3	4
45000	5.4	5.3	5.4	4.6	4.7	5
50000	6.2	4.8	4.7	3.8	5.3	5



	Series1	Series2	Series3	Series4	Series5	Series6
No. of Threads	2	4	8	16	32	64

Array size = 200000						
Cut-off / Threads	**2**	**4**	**8**	**16**	**32**	**64**
5000	56.4	11.6	11.6	11.7	11.5	8.3
10000	41.6	11	8.4	8.4	8.5	9.4
15000	11.6	8.9	8.5	8.7	8.5	8.5
20000	20.7	9.1	7.8	7.5	8.4	8.4
25000	10	9	8.5	10	8.9	8.5
30000	11.6	9.6	8.4	8.5	8.3	8.4
35000	11.6	8.1	6.9	8.5	8	8.5
40000	11.6	10	9.3	7.8	7.8	7.8
45000	10	10	9.2	8.5	8.7	6.9
50000	11.6	8.5	8.2	6.9	8.3	8.5



	Series1	Series2	Series3	Series4	Series5	Series6
No. of Threads	2	4	8	16	32	64

IV. Conclusion:

After running various experiments with different cut-off values and the different number of threads for various array sizes and then consequently comparing their respective graphs, I can conclude that 16 threaded programs are the optimal choice for my laptop, as beyond that there is no significant decrease or increase in performance.
The performance of Threads 4 and 8 is also not far behind, but to get the best performance out of my machine, 16 threads is the best option to opt for, as it seems to be the best at avoiding any bottlenecks and any kind of overhead delay which is expected to be there. Hence, we can now conclude that as we keep increasing the number of threads, i.e., the degree of parallelism in the program the time taken to execute the program is better, which in turn improves the efficiency of the program.