Tushar Kurhekar
NUID - 001521707

Program Structures & Algorithms INFO6205

Assignment 4

Task Performed :

Task (List down the tasks performed in the Assignment)

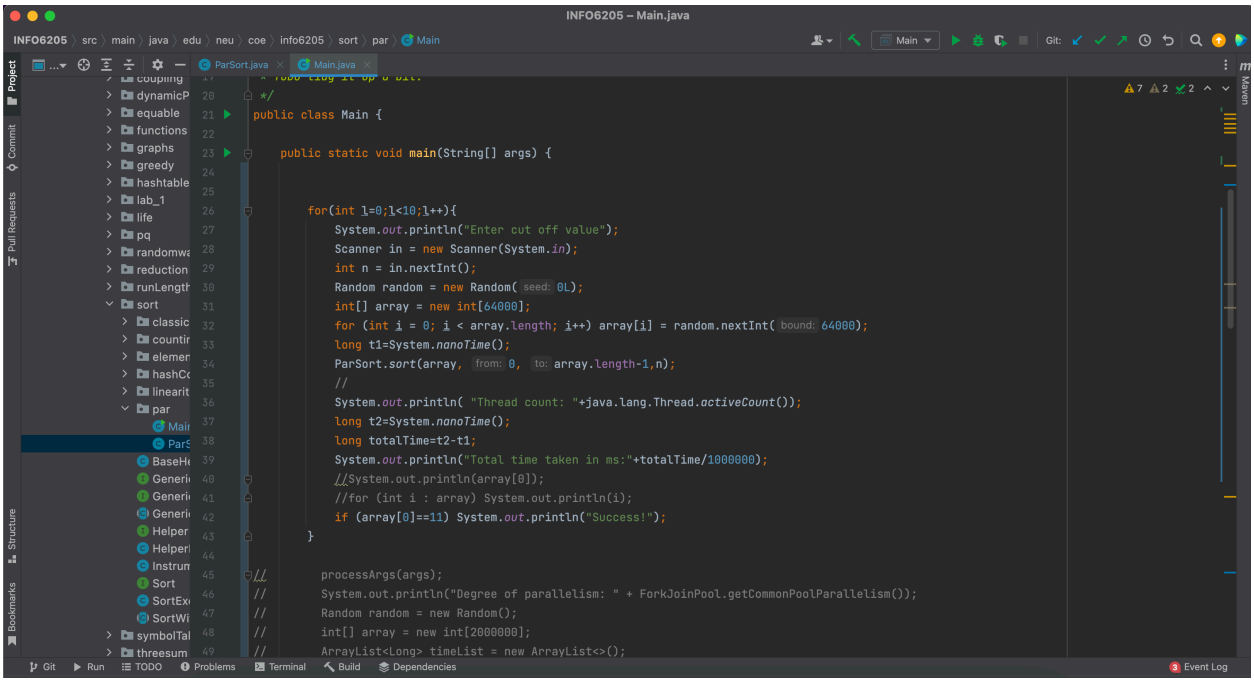
- 1. A cutoff (defaults to, say, 1000) which you will update according to the first argument in the command line when running. It's your job to experiment and come up with a good value for this cutoff. If there are fewer elements to sort than the cutoff, then you should use the system sort instead.
-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

Report Outcomes :

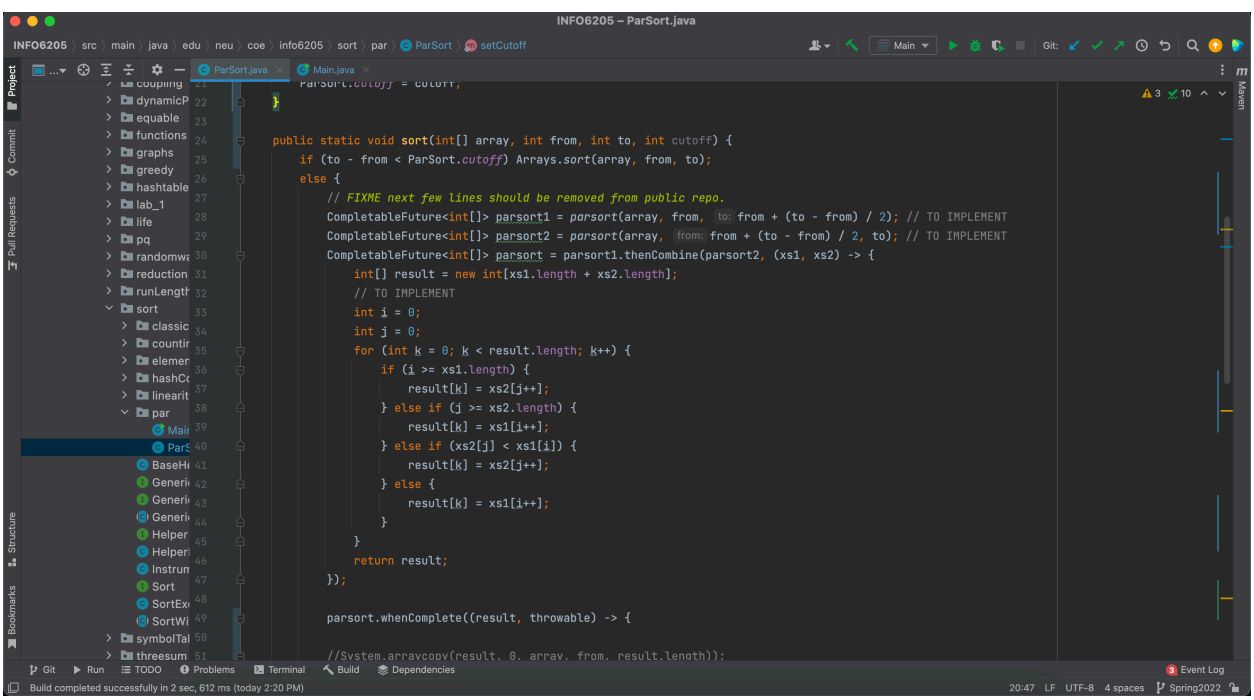
- It has been observed for lower cutoff values that System sort is more efficient than the parallel sort implemented which we observed by noting the timestamp.
- The ideal cutoff can be concluded as little more than 10 % of the array element which generates threads as well as takes less time in sorting
- Sorting becomes efficient as we increase the cut-off
- A good cutoff is array size/thread number, when the cutoff is similar to this number, the performance becomes flattened and better.
- The total number of cores of my laptop is 4, an ideal thread number is 4, when the thread number is larger than 4, the performance is similar.

Evidence to support the conclusion :

Main function :



partSort()



Results for the conclusion :

Output Terminal :

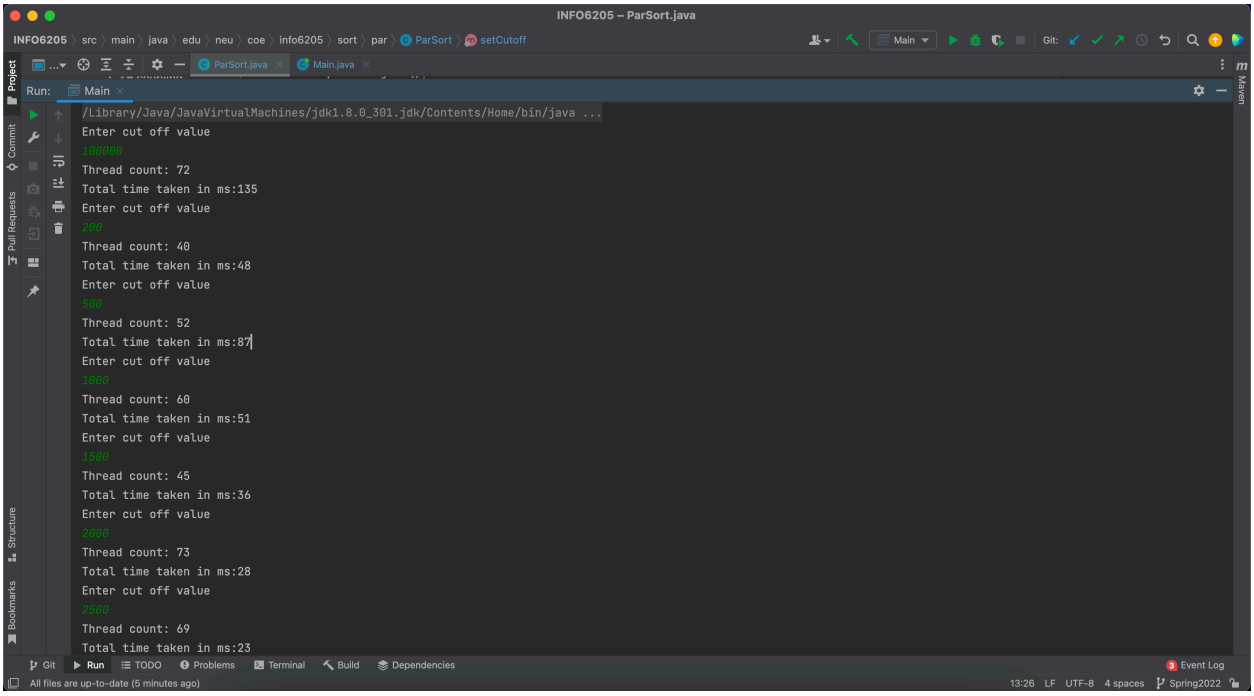
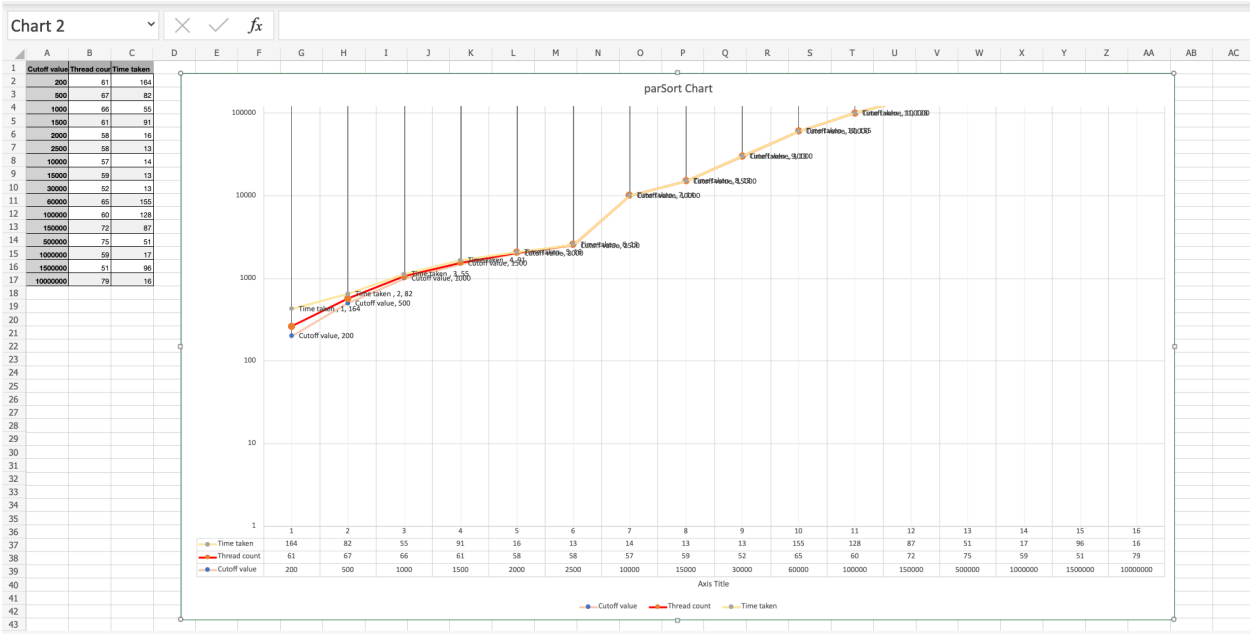


Table is to thread count is to time taken index

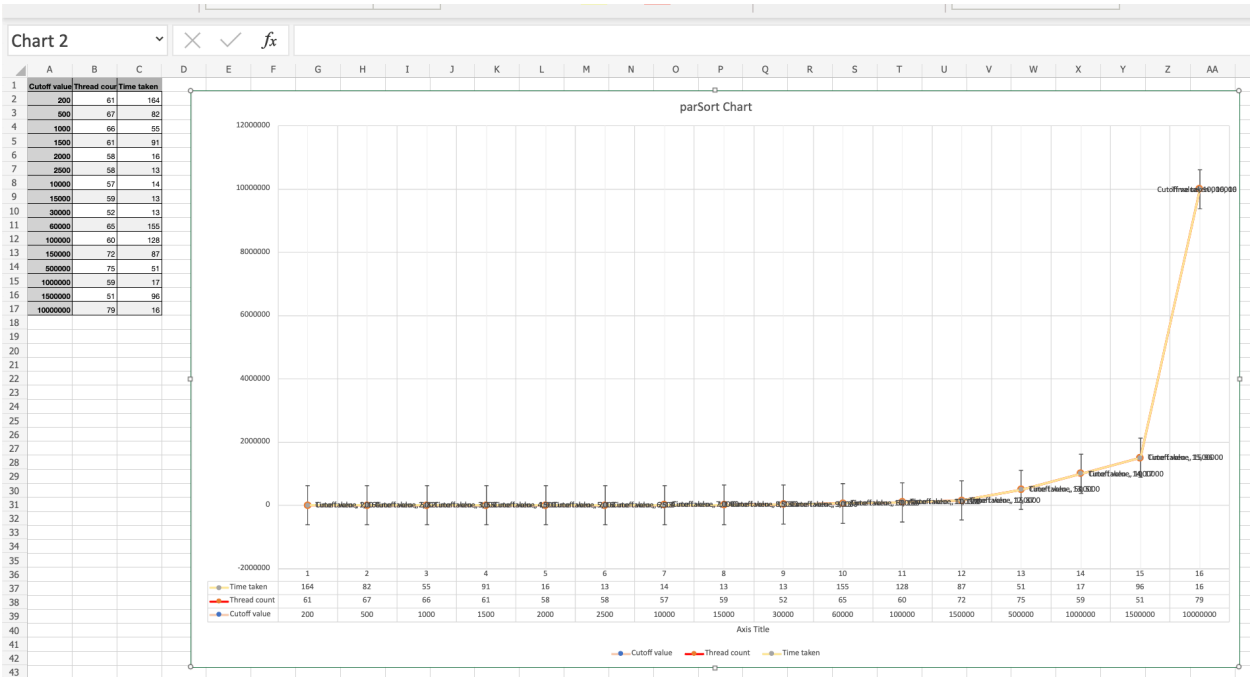
Cutoff value	Thread count (paralleism)	Time taken
200	61	164
500	67	82
1000	66	55
1500	61	91
2000	58	16
2500	58	13
10000	57	14
15000	59	13
30000	52	13
60000	65	155
100000	60	128
150000	72	87
500000	75	51
1000000	59	17
1500000	51	96
10000000	79	16

Logarithmic Scale :



Cutoff value	Thread cour	Time taken
200	61	164
500	67	82
1000	66	55
1500	61	91
2000	58	16
2500	58	13
10000	57	14
15000	59	13
30000	52	13
60000	65	155
100000	60	128
150000	72	87
500000	75	51
1000000	59	17
1500000	51	96
10000000	79	16

Actual Scale :



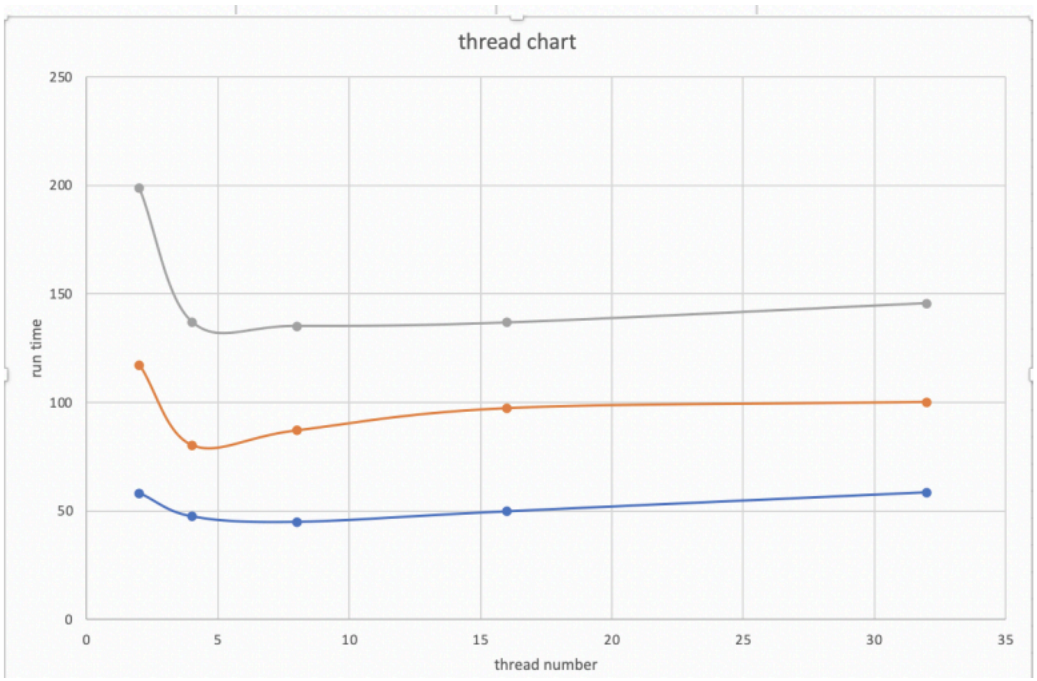
Cutoff value	Thread cour	Time taken
200	61	164
500	67	82
1000	66	55
1500	61	91
2000	58	16
2500	58	13
10000	57	14
15000	59	13
30000	52	13
60000	65	155
100000	60	128
150000	72	87
500000	75	51
1000000	59	17
1500000	51	96
10000000	79	16

Extended Conclusion :

From the above charts, we can conclude that a relatively better

CUTOFFF = array size / thread number. (middle value)

I have seen the time taken flatten by the cutoff values



Generalized observation of runtime and number of threads