Assignment 4 - PSA

Name: Koovehithilu Shrikrishna Joisa

NUID: 002920963

Github Link: https://github.com/falcon-head/PSA-Assignments

Task

Please see the presentation on *Assignment on Parallel Sorting* under the *Exams. etc.* module.

Your task is 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.

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.

There is a *Main* class and the *ParSort* class in the *sort.par* package of the INFO6205 repository. The *Main* class can be used as is but the *ParSort* class needs to be implemented where you see "TODO..." [it turns out that these TODOs are already implemented].

Unless you have a good reason not to, you should just go along with the Java8-style future implementations provided for you in the class repository.

You must prepare a report that shows the results of your experiments and draws a conclusion (or more) about the efficacy of this method of parallelizing sort. Your 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 cutoff schemes.

For varying the number of threads available, you might want to consult the following resources:

- https://www.callicoder.com/java-8-completablefuture-tutorial/#a-note-about-executor-and-thread-pool (Links to an external site.)
- https://stackoverflow.com/questions/36569775/how-to-set-forkjoinpool-with-the-desired-number-of-worker-threads-in-completable (Links to an external site.)

Good luck and enjoy.

Output / Screenshots

Output array size with 2000

Degree of parallelism 2

Cut Off	Time
200	728
300	150
400	193
500	63
600	84
700	64
800	36
900	77
1000	62
1100	45
1200	41
1300	38
1400	239
1500	49
1600	40
1700	93
1800	35
1900	98
2000	38

Degree of parallelism 3

Cut Off	Time
200	245
300	191
400	65
500	75
600	53
700	53
800	78
900	102
1000	43
1100	36
1200	41
1300	38
1400	43
1500	40
1600	50
1700	115
1800	39
1900	32
2000	37

Degree of parallelism 4

Cut Off	Time
200	245
300	191
400	65
500	75
600	53
700	53
800	78
900	102
1000	43

Cut Off	Time
1100	36
1200	41
1300	38
1400	43
1500	40
1600	50
1700	115
1800	39
1900	32
2000	37

Degree of parallelism 5

Cut Off	Time
200	215
300	130
400	123
500	183
600	93
700	91
800	54
900	59
1000	58
1100	60
1200	54
1300	58
1400	62
1500	54
1600	41
1700	37
1800	41
1900	38
2000	37

Degree of parallelism 6

Cut Off	Time
200	232
300	156
400	107
500	99
600	98
700	102
800	61
900	66
1000	67
1100	61
1200	65
1300	61
1400	65
1500	61
1600	44
1700	39
1800	44
1900	41
2000	45

Degree of parallelism 7

Cut Off	Time
200	280
300	166
400	122
500	119
600	119
700	114
800	77
900	84
1000	95

Cut Off	Time
1100	94
1200	69
1300	79
1400	79
1500	79
1600	49
1700	44
1800	51
1900	47
2000	53

Degree of parallelism 8

Cut Off	Time
200	328
300	208
400	136
500	130
600	136
700	156
800	91
900	83
1000	91
1100	83
1200	78
1300	82
1400	91
1500	83
1600	59
1700	55
1800	48
1900	53
2000	49

Degree of parallelism 9

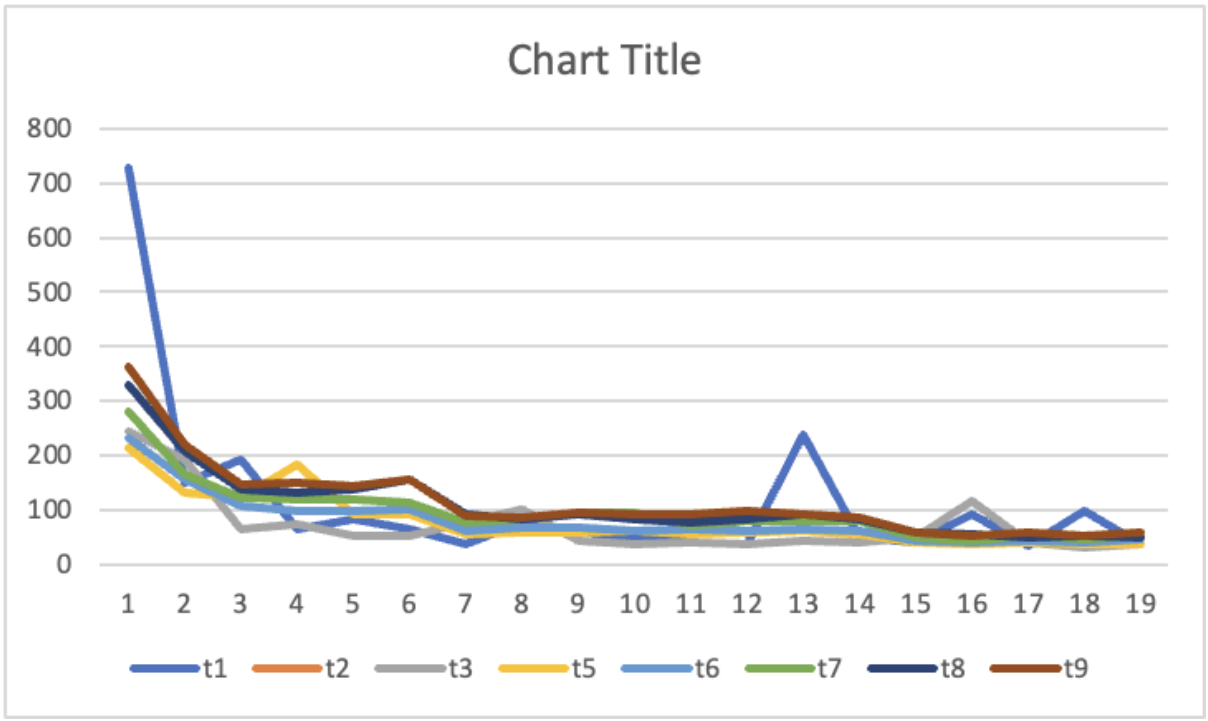
Cut Off	Time
200	364
300	219
400	146
500	151
600	145
700	155
800	90
900	87
1000	94
1100	91
1200	93
1300	98
1400	92
1500	86
1600	59
1700	51
1800	59
1900	53
2000	58

Degree of parallelism 10

Cut Off	Time
200	367
300	241
400	164
500	160
600	155
700	163
800	104
900	93
1000	102

Cut Off	Time
1100	94
1200	98
1300	97
1400	96
1500	100
1600	57
1700	64
1800	62
1900	61
2000	55

Output graph



Output with array size 100000

Degree of parallelism 2

Count off	Time
200	474

Count off	Time
300	217
400	133
500	207
600	182
700	57
800	50
900	130
1000	44
1100	42
1200	106
1300	37
1400	60
1500	55
1600	38
1700	40
1800	37
1900	138
2000	31

Degree of parallelism 3

Count off	Time
200	179
300	80
400	57
500	63
600	56
700	64
800	43
900	39
1000	86
1100	84
1200	44

Count off	Time
1300	46
1400	38
1500	43
1600	31
1700	42
1800	31
1900	38
2000	31

Degree of parallelism 4

Count off	Time
200	202
300	103
400	91
500	79
600	66
700	69
800	69
900	50
1000	75
1100	46
1200	54
1300	58
1400	55
1500	61
1600	40
1700	36
1800	37
1900	52
2000	34

Degree of parallelism 5

Count off	Time
200	254
300	137
400	89
500	93
600	85
700	88
800	54
900	59
1000	54
1100	53
1200	58
1300	53
1400	59
1500	54
1600	37
1700	43
1800	38
1900	45
2000	38

Degree of parallelism 6

Count off	Time
200	300
300	148
400	117
500	102
600	108
700	102
800	66
900	63
1000	65
1100	69

Count off	Time
1200	62
1300	67
1400	62
1500	62
1600	45
1700	44
1800	49
1900	41
2000	41

Degree of parallelism 7

Count off	Time
200	265
300	172
400	111
500	118
600	110
700	121
800	69
900	70
1000	74
1100	72
1200	75
1300	69
1400	82
1500	70
1600	44
1700	49
1800	44
1900	44
2000	49

Degree of parallelism 8

Count off	Time
200	305
300	198
400	143
500	129
600	132
700	126
800	88
900	107
1000	100
1100	100
1200	121
1300	136
1400	108
1500	98
1600	63
1700	54
1800	51
1900	58
2000	51

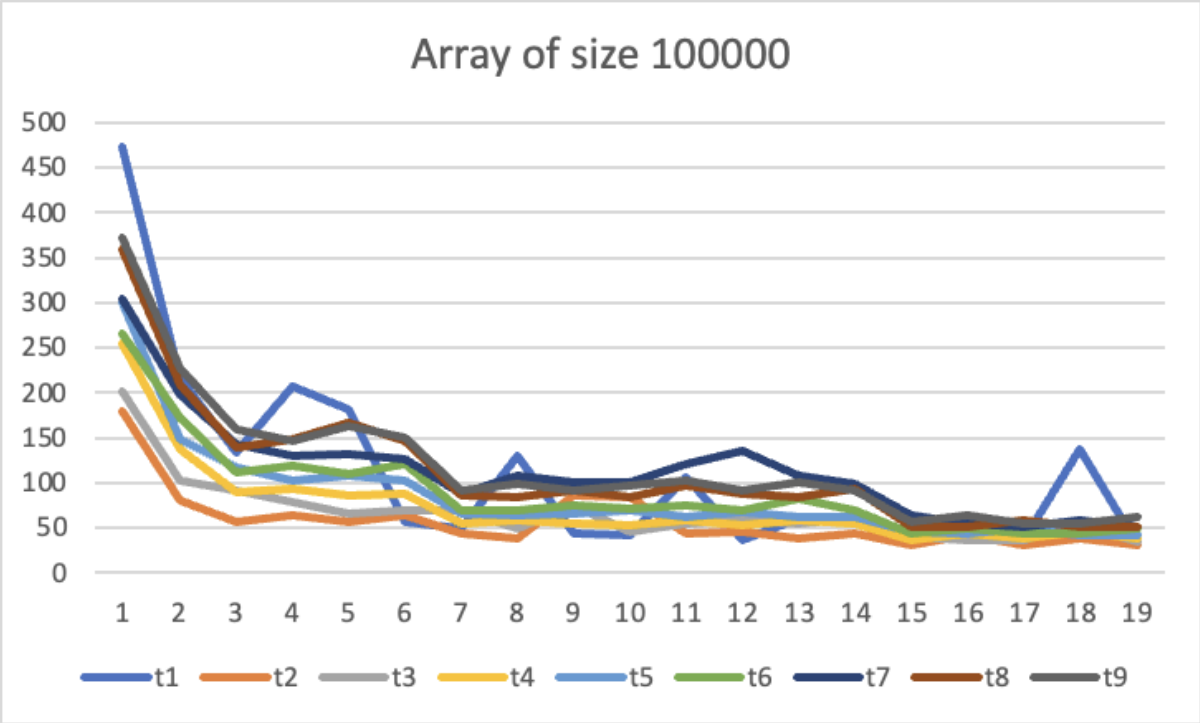
Degree of parallelism 9

Count off	Time
200	360
300	209
400	139
500	149
600	167
700	146
800	85
900	84
1000	92
1100	84

Count off	Time
1200	97
1300	88
1400	84
1500	93
1600	51
1700	51
1800	59
1900	51
2000	51

Degree of parallelism 10

Count off	Time
200	372
300	228
400	160
500	147
600	162
700	150
800	92
900	99
1000	92
1100	96
1200	102
1300	92
1400	100
1500	92
1600	57
1700	63
1800	55
1900	54
2000	62



Observation

- The above experimentation is for the array size of 200 & 10000
- From the above experiment I can say about 25-30% of the graphs are showing the same pattern.
- From the graph I would say the degree 5 is the ideal one & there is no remarkable performance improvement with the increase of degree in parallelism.