Zixiao Wang 001058840

# INFO 6205
## Program Structures & Algorithms
## Fall 2020
## Assignment 5

The report format follows this document



# Task

Assignment 4 (Parallel sort) Please see the presentation on Assignment on Parallel Sorting under Course Materials/Course Documents/Exams. etc.

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.

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. 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). 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..."

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.

# Experiments

Basically, I use different array length with different cutoff ratio to see when will the parallel sorting have the good performance

- `q time` is the time cost without parallel sorting
- `p time` is the time cost with parallel sorting

## Output

```
********Array Length: 2048
********Cutoff: 1024
q time:5
p time:61
********Cutoff: 512
q time:2
p time:16
********Cutoff: 256
q time:3
p time:81
********Cutoff: 128
q time:1
p time:105
```

```
********Array Length: 4096
********Cutoff: 2048
q time:2
p time:2
********Cutoff: 1024
q time:2
p time:3
********Cutoff: 512
q time:2
p time:65
********Cutoff: 256
q time:2
p time:60
```

```
********Array Length: 8192
********Cutoff: 4096
q time:4
p time:3
********Cutoff: 2048
q time:4
p time:4
********Cutoff: 1024
q time:4
p time:13
********Cutoff: 512
q time:5
p time:25
```

```
********Array Length: 16384
********Cutoff: 8192
q time:9
p time:11
********Cutoff: 4096
q time:21
p time:19
********Cutoff: 2048
q time:42
p time:19
********Cutoff: 1024
q time:9
p time:29
```

```
********Array Length: 32768
********Cutoff: 16384
q time:39
p time:11
********Cutoff: 8192
q time:22
p time:44
********Cutoff: 4096
q time:29
p time:86
********Cutoff: 2048
q time:21
p time:161
```

```
********Array Length: 65536
********Cutoff: 32768
q time:39
p time:25
********Cutoff: 16384
q time:45
p time:30
********Cutoff: 8192
q time:41
p time:47
********Cutoff: 4096
q time:40
p time:64
```

```
********Array Length: 131072
********Cutoff: 65536
q time:81
p time:46
********Cutoff: 32768
q time:80
p time:51
********Cutoff: 16384
q time:94
p time:72
********Cutoff: 8192
q time:82
p time:91
********Array Length: 262144
********Cutoff: 131072
q time:222
p time:95
********Cutoff: 65536
q time:182
p time:104
********Cutoff: 32768
q time:185
p time:165
********Cutoff: 16384
q time:184
p time:176
********Array Length: 524288
********Cutoff: 262144
q time:386
p time:201
********Cutoff: 131072
q time:379
p time:221
********Cutoff: 65536
q time:376
p time:307
********Cutoff: 32768
q time:377
p time:305
```

## Conclusion

1. Parallel sorting will have good performance when the array lenth is greater than $2^{13} = 8192$
2. Parallel sorting will have a good performance when the cutoff is approximate the quator of array length.
3. Parallel sorting do have better performace when the array is very large.