Qinyun Lin (NUID: 001582464)

Program Structures & Algorithms

Fall 2021

Assignment No. 5

Task (List down the tasks performed in the Assignment)

- Test the parallelizing Merge Sort performance with different array size, cutoff size and threads number.
- Draws a conclusion (or more) about the efficacy of this method of parallelizing sort

Conclusion:

- The best cutoff size in my computer is range from 1/16 to 1/4 of the array size. The larger array should use the larger cutoff size.
- The best available threads number in my computer is range from 16 to 128.

Evidence to support the conclusion:

1. Output

I used 6 different array size(100w, 200w, 400w, 600w, 800w, 1000w), 10 different cutoff percent(1/2, 1/4, ..., 1/1024) and 11 different threads number(1, 2, 4, ..., 1024) to test the performance.

i. part of output - 1

```
Array length: 800w
Degree of parallelism: 8
threads num: 8    cutoff percent: 1/1024    cutoff size: 7812    average time: 337.500
threads num: 8    cutoff percent: 1/512    cutoff size: 15625    average time: 311.300
threads num: 8    cutoff percent: 1/256    cutoff size: 31250    average time: 311.500
threads num: 8    cutoff percent: 1/128    cutoff size: 62500    average time: 305.300
threads num: 8    cutoff percent: 1/64    cutoff size: 125000    average time: 303.900
threads num: 8    cutoff percent: 1/32    cutoff size: 250000    average time: 271.300
threads num: 8    cutoff percent: 1/16    cutoff size: 500000    average time: 270.600
threads num: 8    cutoff percent: 1/8    cutoff size: 1000000    average time: 272.200
threads num: 8    cutoff percent: 1/4    cutoff size: 2000000    average time: 274.700
threads num: 8    cutoff percent: 1/2    cutoff size: 4000000    average time: 307.000
threads num: 8    cutoff percent: 1/1    cutoff size: 8000000    average time: 392.500

Array length: 800w
Degree of parallelism: 16
threads num: 16    cutoff percent: 1/1024    cutoff size: 7812    average time: 352.300
threads num: 16    cutoff percent: 1/512    cutoff size: 15625    average time: 318.100
threads num: 16    cutoff percent: 1/256    cutoff size: 31250    average time: 311.600
threads num: 16    cutoff percent: 1/128    cutoff size: 62500    average time: 299.100
threads num: 16    cutoff percent: 1/64    cutoff size: 125000    average time: 292.100
threads num: 16    cutoff percent: 1/32    cutoff size: 250000    average time: 269.400
threads num: 16    cutoff percent: 1/16    cutoff size: 500000    average time: 267.600
threads num: 16    cutoff percent: 1/8    cutoff size: 1000000    average time: 262.700
threads num: 16    cutoff percent: 1/4    cutoff size: 2000000    average time: 259.800
threads num: 16    cutoff percent: 1/2    cutoff size: 4000000    average time: 303.000
threads num: 16    cutoff percent: 1/1    cutoff size: 8000000    average time: 388.200
```

ii. part of output - 2

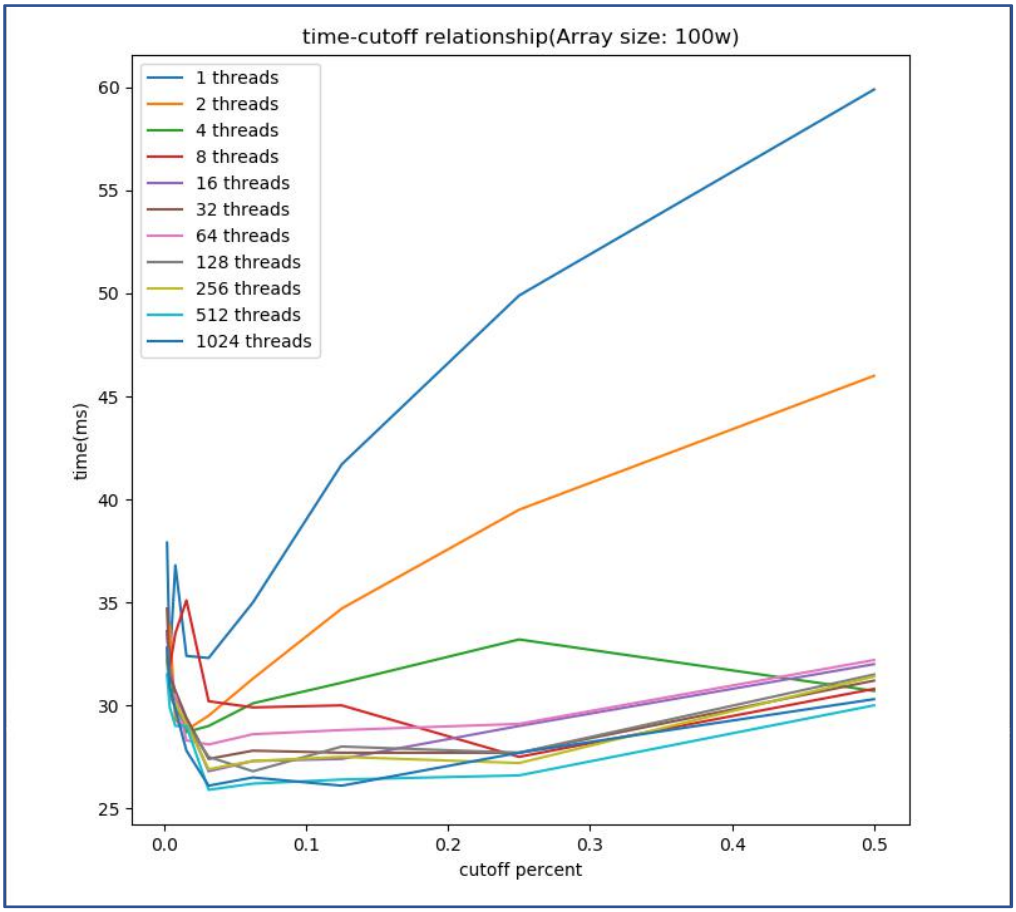
```
Array length: 1000w
Degree of parallelism: 512
threads num: 512    cutoff percent: 1/1024    cutoff size: 9765    average time: 458.800
threads num: 512    cutoff percent: 1/512    cutoff size: 19531    average time: 444.000
threads num: 512    cutoff percent: 1/256    cutoff size: 39062    average time: 432.500
threads num: 512    cutoff percent: 1/128    cutoff size: 78125    average time: 420.100
threads num: 512    cutoff percent: 1/64    cutoff size: 156250    average time: 411.100
threads num: 512    cutoff percent: 1/32    cutoff size: 312500    average time: 400.600
threads num: 512    cutoff percent: 1/16    cutoff size: 625000    average time: 375.000
threads num: 512    cutoff percent: 1/8    cutoff size: 1250000    average time: 381.500
threads num: 512    cutoff percent: 1/4    cutoff size: 2500000    average time: 380.500
threads num: 512    cutoff percent: 1/2    cutoff size: 5000000    average time: 435.700
threads num: 512    cutoff percent: 1/1    cutoff size: 10000000    average time: 548.800

Array length: 1000w
Degree of parallelism: 1024
threads num: 1024    cutoff percent: 1/1024    cutoff size: 9765    average time: 495.200
threads num: 1024    cutoff percent: 1/512    cutoff size: 19531    average time: 445.900
threads num: 1024    cutoff percent: 1/256    cutoff size: 39062    average time: 444.400
threads num: 1024    cutoff percent: 1/128    cutoff size: 78125    average time: 417.400
threads num: 1024    cutoff percent: 1/64    cutoff size: 156250    average time: 409.600
threads num: 1024    cutoff percent: 1/32    cutoff size: 312500    average time: 405.300
threads num: 1024    cutoff percent: 1/16    cutoff size: 625000    average time: 404.900
threads num: 1024    cutoff percent: 1/8    cutoff size: 1250000    average time: 394.200
threads num: 1024    cutoff percent: 1/4    cutoff size: 2500000    average time: 323.400
threads num: 1024    cutoff percent: 1/2    cutoff size: 5000000    average time: 380.300
threads num: 1024    cutoff percent: 1/1    cutoff size: 10000000    average time: 481.300
```

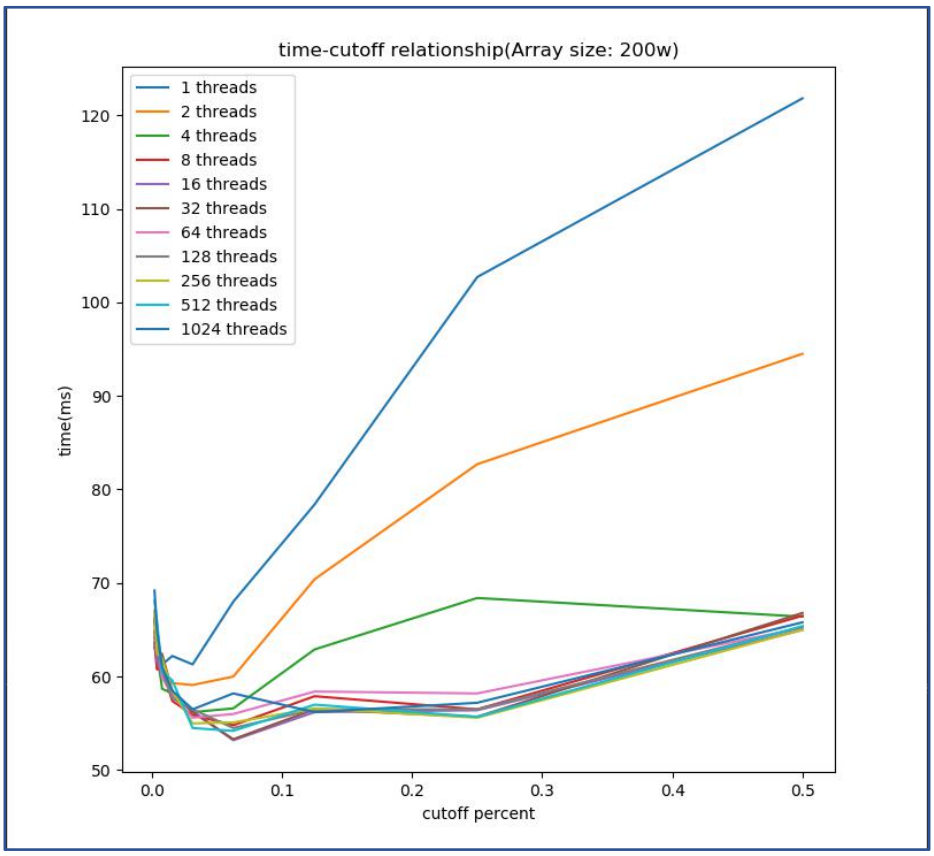
All the output have been saved to .csv files what will be processed by python to make visualization.

2. **Graphical Representation**

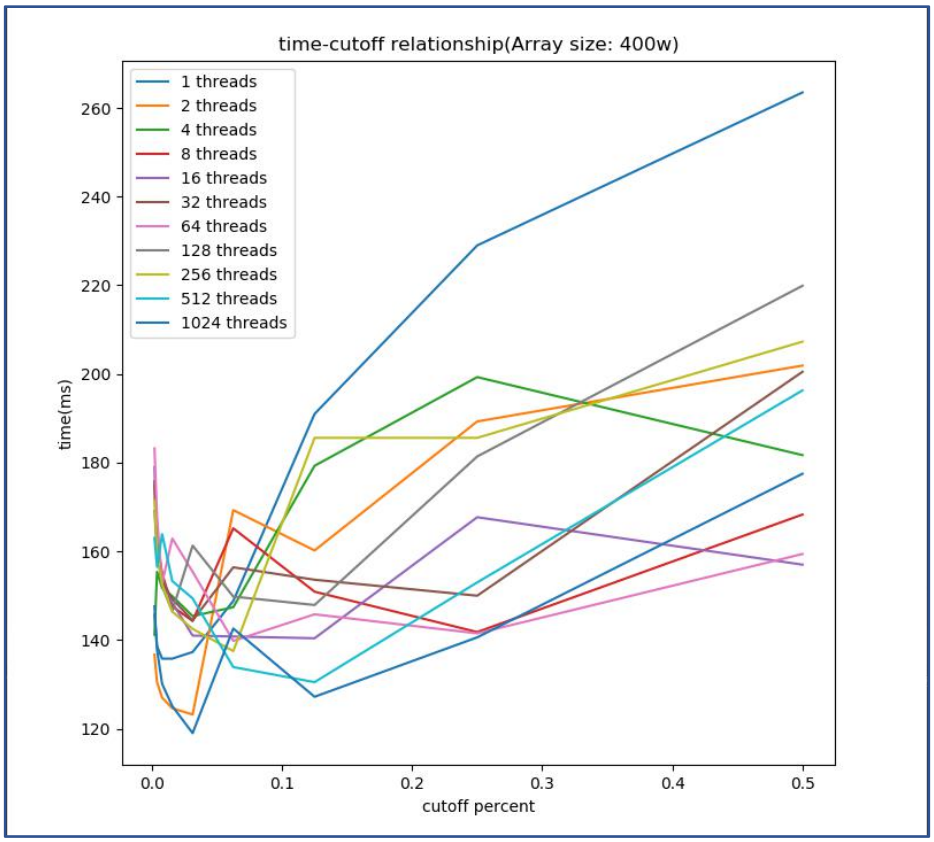
i. result: array size - 100w



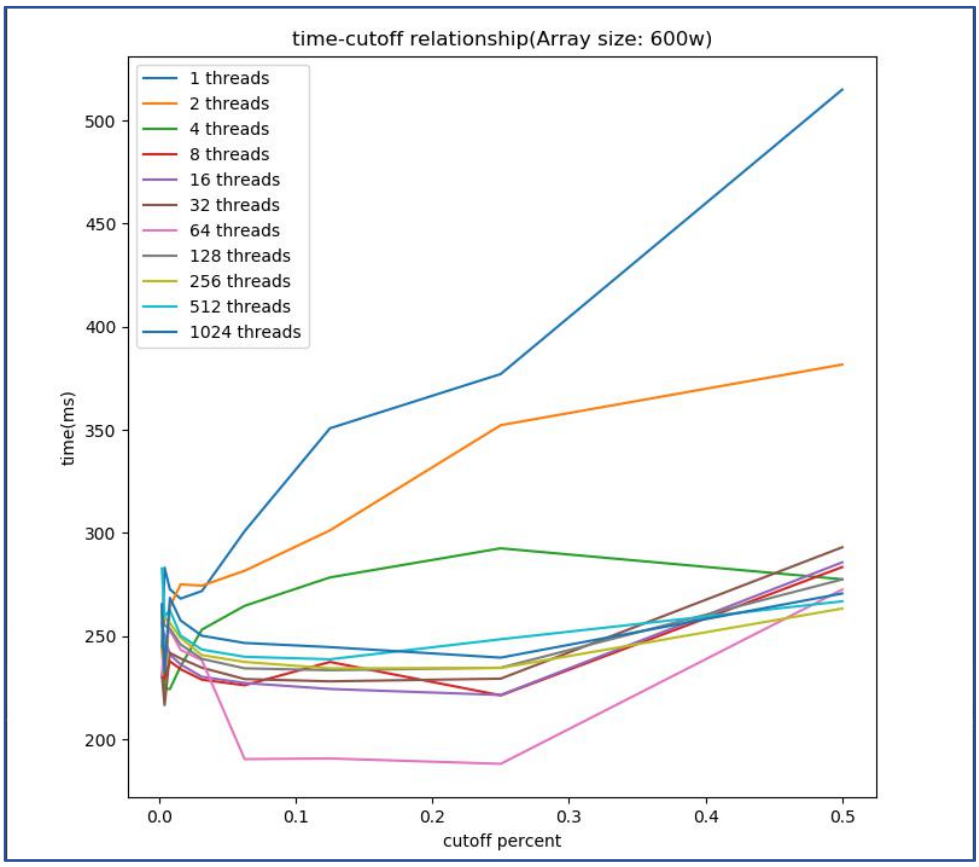
ii. result: array size - 200w



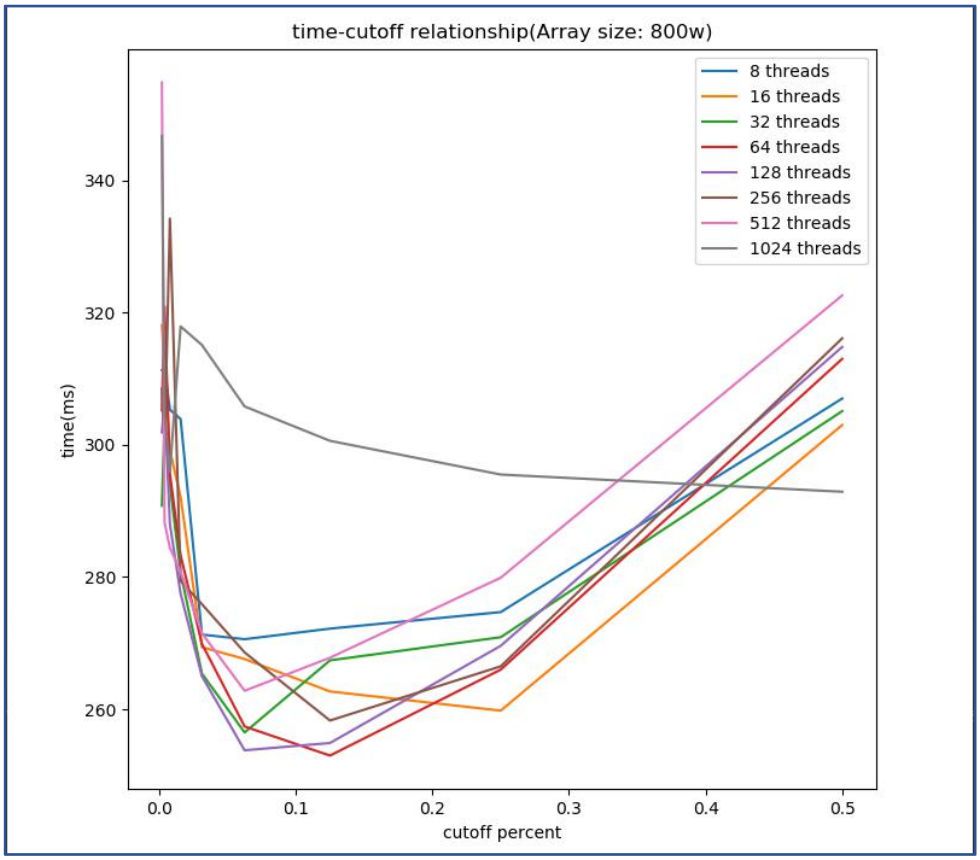
iii. result: array size - 400w



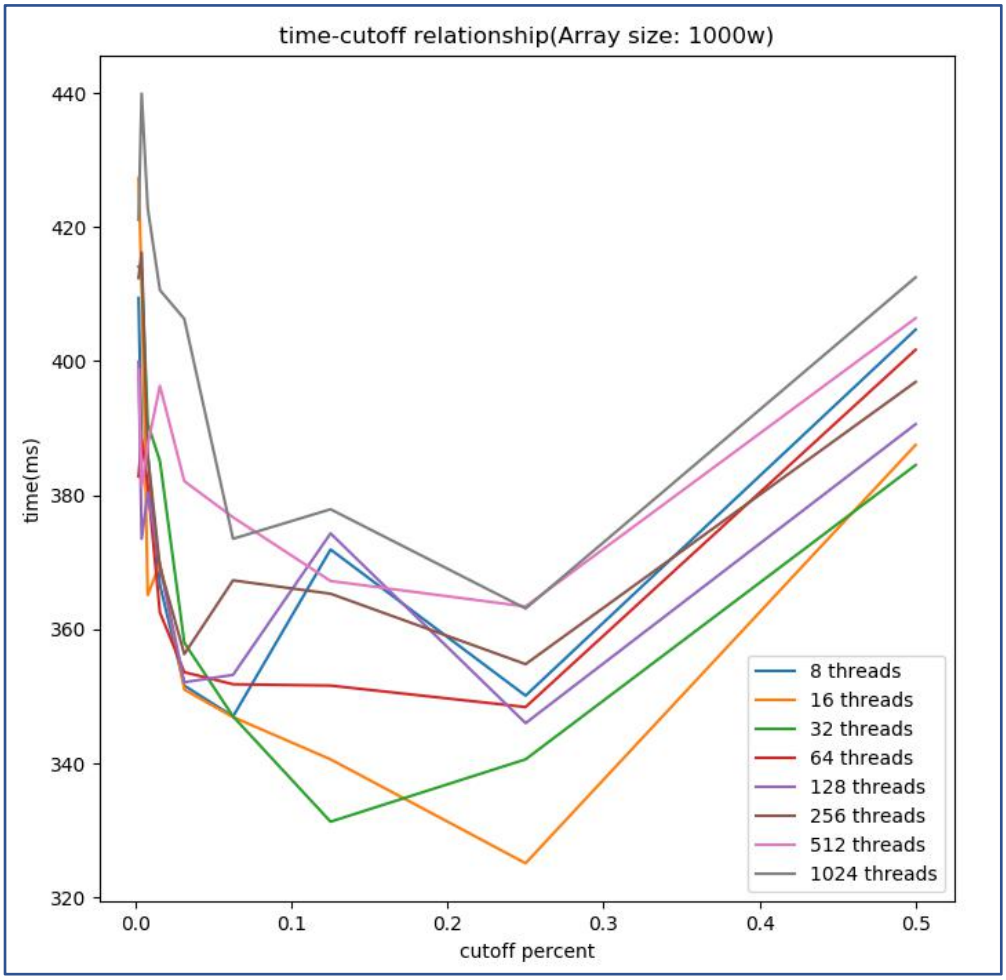
iv. result: array size - 600w



v. result: array size - 800w



vi. result: array size - 1000w



According to the diagrams, in most cases, the best cutoff percent is from 1/16 to 1/4, and as the array size goes up, the appropriate cutoff percent should increase slightly.

As for threads number, it seems not that stable, but when the threads number between 16 and 128 will be better.