YanXuan NUID:001563047

Program Structures & Algorithms

Fall 2021

Assignment No. 5

- **Task (List down the tasks performed in the Assignment)**
  1. **Experiment and come up with a good value for cutoff.**
  2. **Decide on an ideal number of separate threads and arrange for that number of partitions to be parallelized.**
  3. **Prepare a report that shows the results of experiments and draw a conclusion (or more) about the efficacy of this method of parallelizing sort.**
  4. **Analyze and summarize the experimental data, and get the relationship between cutoff numbers, array's length and the number of threads.**

- **Relationship Conclusion: (For ex : z = a * b)**

  I. **The relationship between the best cutoff number, the array's length and the number of threads is:**

$$\boldsymbol{the\ best\ cutoff\ number = \frac{arrays.length}{threadNumber}}$$

  II. **The ParSort always does best (compared by the times the best cutoff numbers cost) when there are 4 threads.**

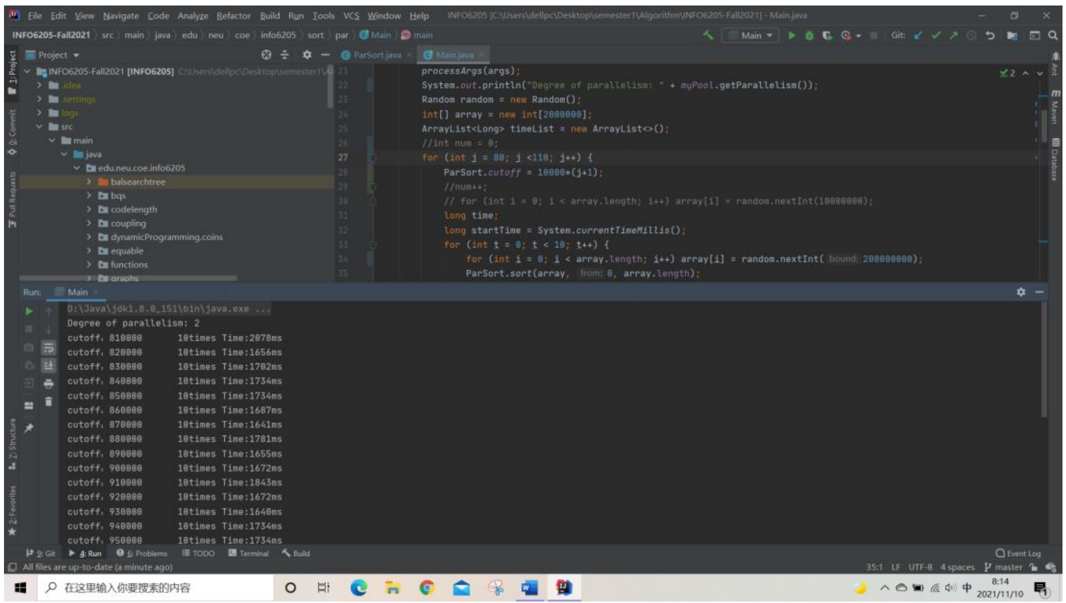
  III. **It is not that the more threads, the faster the ParSort speed.**

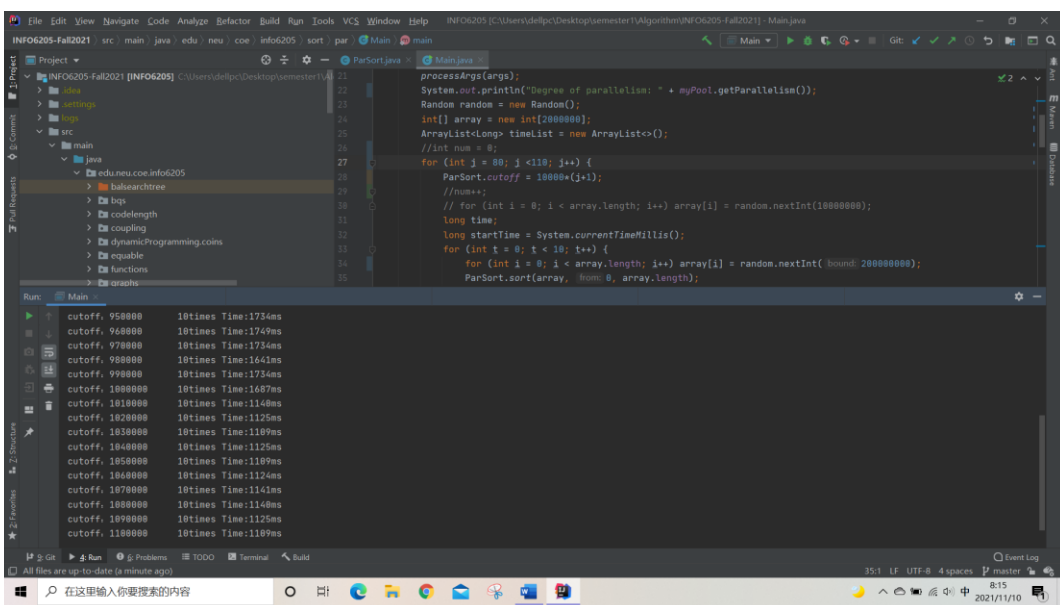
- **Evidence to support the conclusion:**

1. **Output (Snapshot of Code output in the terminal)**

   ① Keep the array's length and the number of threads un-changeable, search for the best cutoff number.

   Output when threadNumber = 2 and array's length = 2,000,000





   Output text:

Degree of parallelism: 2

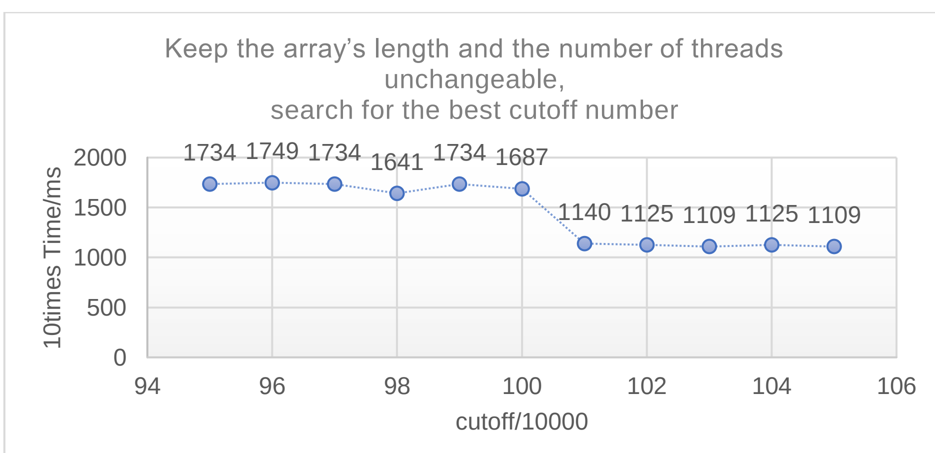
cutoff：950000	**10times Time:1734ms**
cutoff：960000	**10times Time:1749ms**
cutoff：970000	**10times Time:1734ms**
cutoff：980000	**10times Time:1641ms**
cutoff：990000	**10times Time:1734ms**
cutoff：1000000	**10times Time:1687ms**
cutoff：1010000	**10times Time:1140ms**
cutoff：1020000	**10times Time:1125ms**
cutoff：1030000	**10times Time:1109ms**
cutoff：1040000	**10times Time:1125ms**
cutoff：1050000	**10times Time:1109ms**

It seems that when the cutoff number is 1,010,000, the time significantly shortened, compared to the cutoff number is 1,000,000.

At which time, $\frac{arrays.length}{threadNumber} = \frac{2000000}{2} = 1,000,000$ **.**

Simulation:

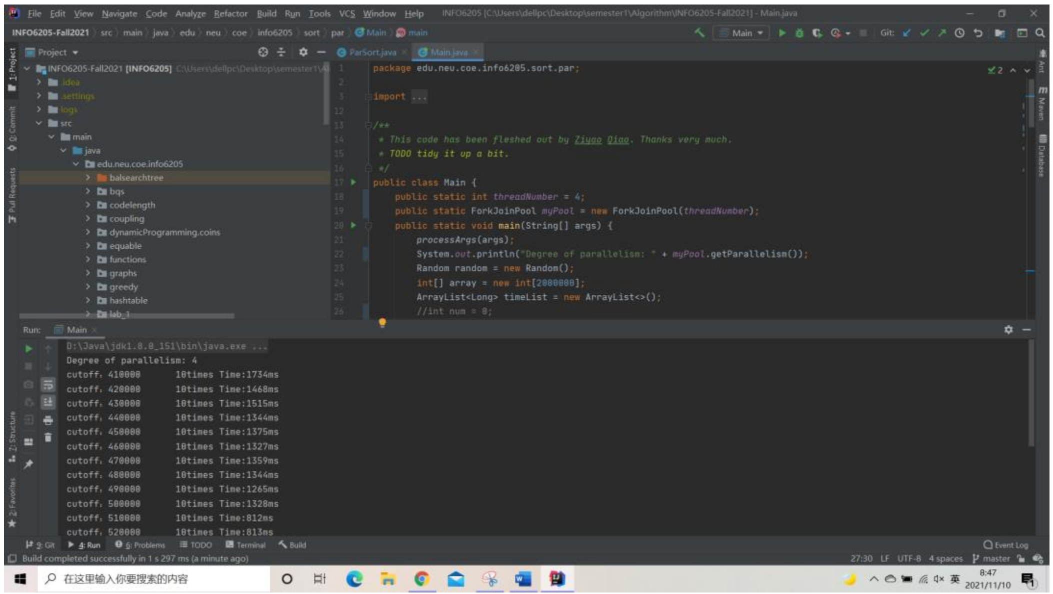
cutoff/10000	10times Time/ms
95	1734
96	1749
97	1734
98	1641
99	1734
100	1687
101	1140
102	1125
103	1109
104	1125
105	1109

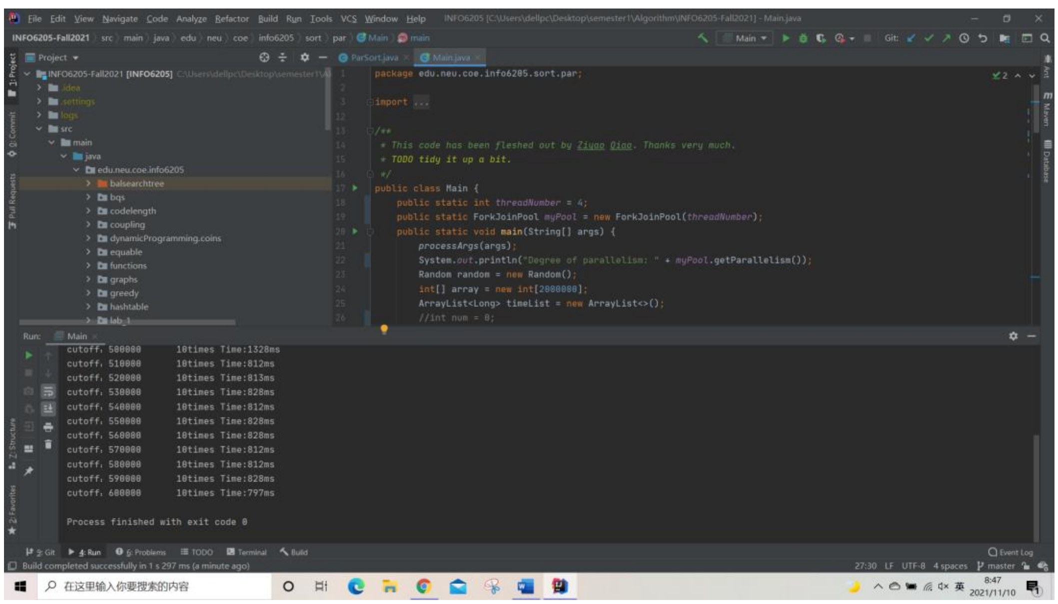


It seems that if the cutoff number is bigger than 1,000,000, which is equals to $\frac{arrays.length}{threadNumber}$, the 10times Time drops a lot.

② Keep the array's length unchangeable, change the number of threads, search for the best cutoff number.

(1)Output when threadNumber = 4 and array's length = 2,000,000





Output text:

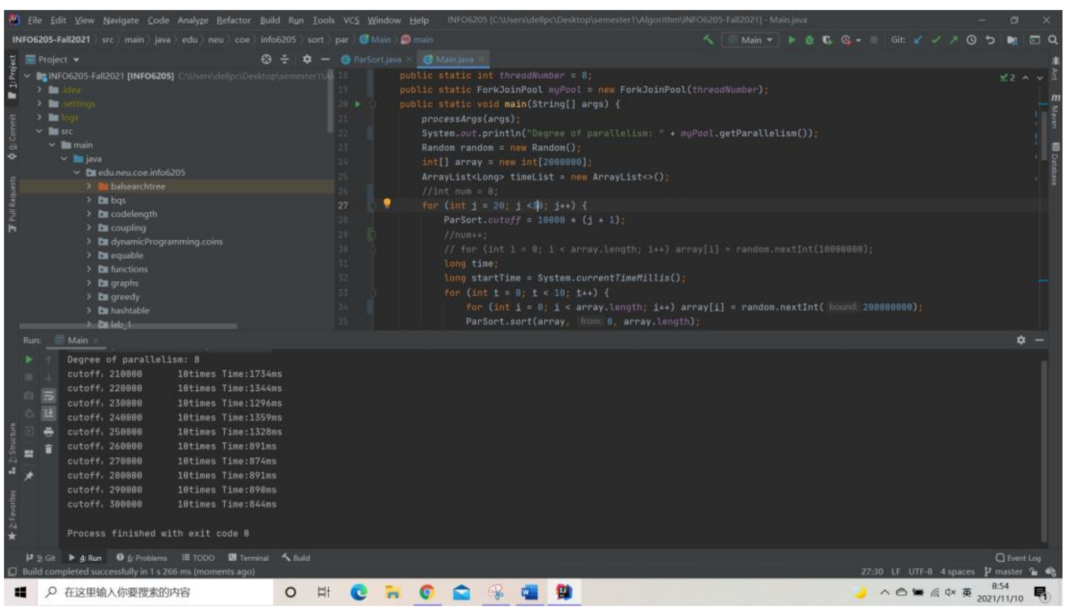
Degree of parallelism: 4

cutoff：450000	**10times Time:1375ms**
cutoff：460000	**10times Time:1327ms**
cutoff：470000	**10times Time:1359ms**
cutoff：480000	**10times Time:1344ms**
cutoff：490000	**10times Time:1265ms**
cutoff：500000	**10times Time:1328ms**
cutoff：510000	**10times Time:812ms**
cutoff：520000	**10times Time:813ms**
cutoff：530000	**10times Time:828ms**
cutoff：540000	**10times Time:812ms**
cutoff：550000	**10times Time:828ms**

We could find that in this case, time drops a lot after the cutoff is more than 500,000.

At which time, $\frac{arrays.length}{threadNumber} = \frac{2000000}{4} = 500{,}000$.

(2)Output when threadNumber = 8 and array's length = 2,000,000



Output text:

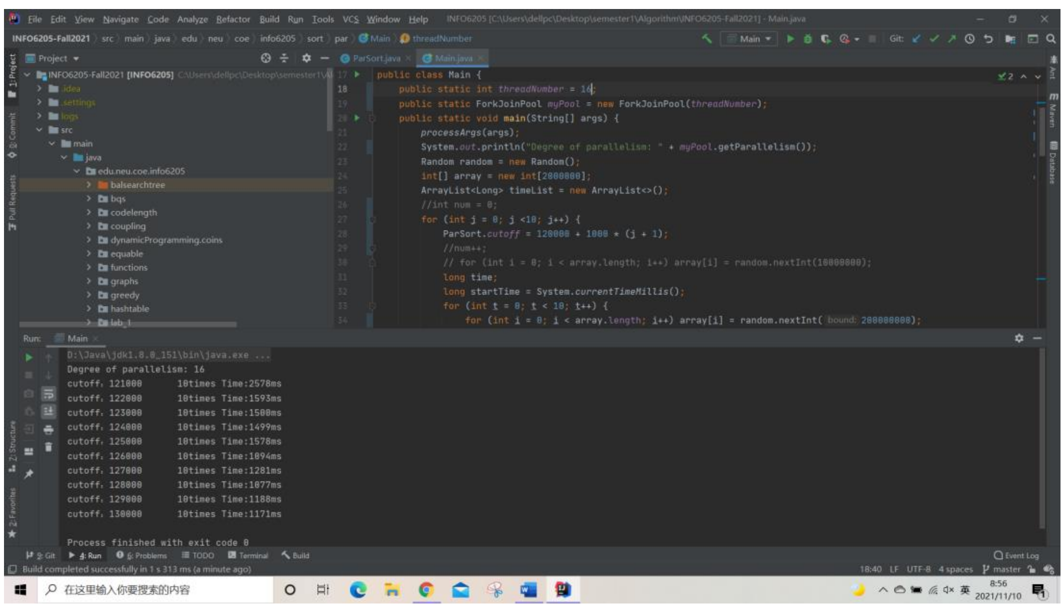
Degree of parallelism: 8	
cutoff：210000	**10times Time:1734ms**

cutoff：220000	**10times Time:1344ms**
cutoff：230000	**10times Time:1296ms**
cutoff：240000	**10times Time:1359ms**
cutoff：250000	**10times Time:1328ms**
cutoff：260000	**10times Time:891ms**
cutoff：270000	**10times Time:874ms**
cutoff：280000	**10times Time:891ms**
cutoff：290000	**10times Time:890ms**
cutoff：300000	**10times Time:844ms**

We could find that in this case, time drops a lot after the cutoff is more than 250,000.

At which time, $\frac{arrays.length}{threadNumber} = \frac{2000000}{8} = 250{,}000$.

(3)Output when threadNumber = 16 and array's length = 2,000,000



Output text:

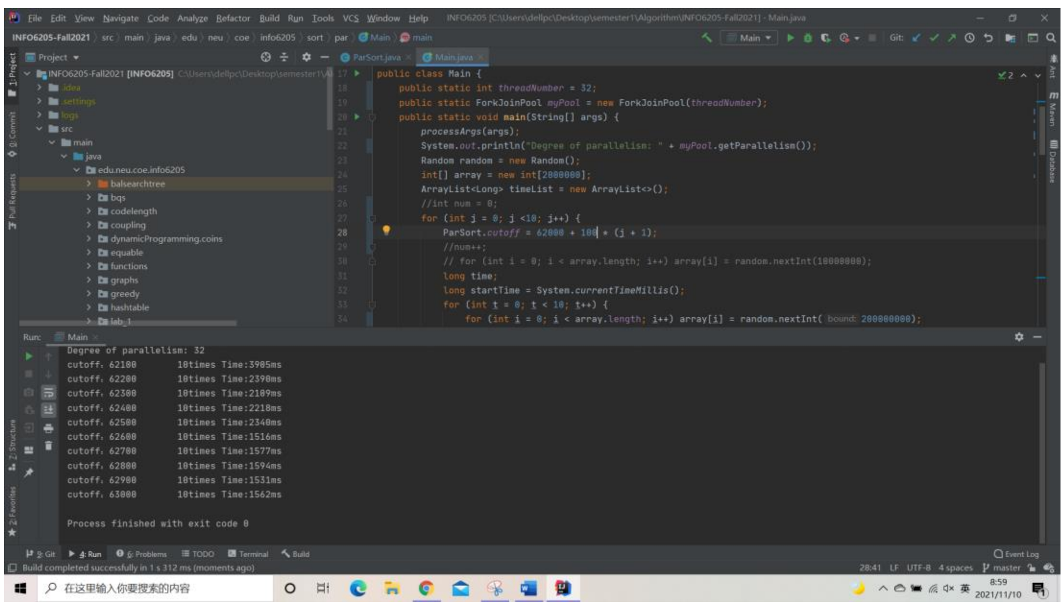
Degree of parallelism: 16	
cutoff：121000	**10times Time:2578ms**
cutoff：122000	**10times Time:1593ms**
cutoff：123000	**10times Time:1500ms**

cutoff：124000	**10times Time:1499ms**
cutoff：125000	**10times Time:1578ms**
cutoff：126000	**10times Time:1094ms**
cutoff：127000	**10times Time:1281ms**
cutoff：128000	**10times Time:1077ms**
cutoff：129000	**10times Time:1188ms**
cutoff：130000	**10times Time:1171ms**

We could find that in this case, time drops a lot after the cutoff is more than 125,000.

At which time, $\frac{arrays.length}{threadNumber} = \frac{2000000}{16} = 125{,}000.$

(4)Output when threadNumber = 32 and array's length = 2,000,000



Output text:

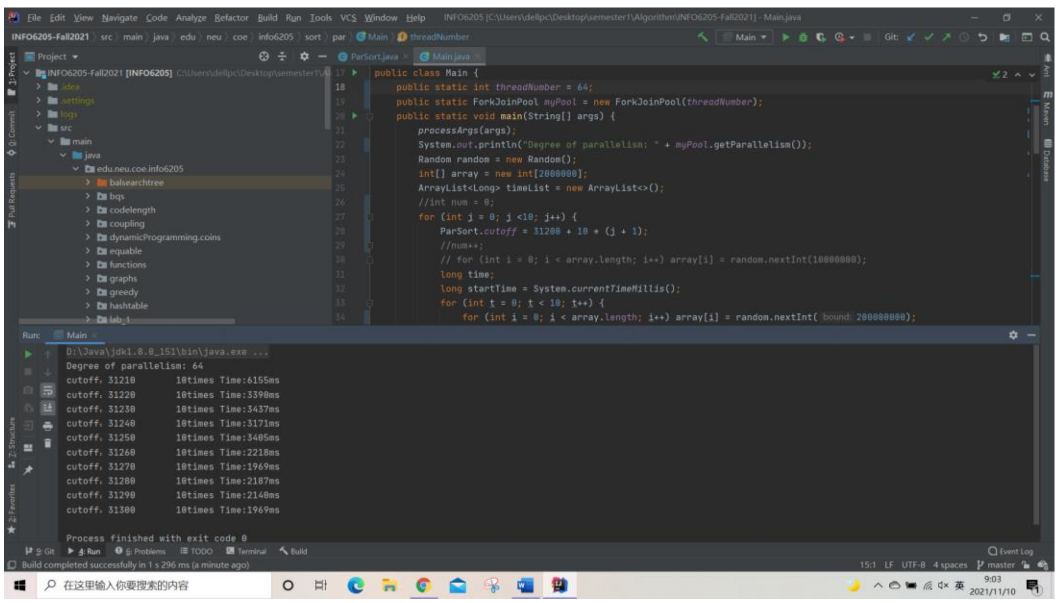
Degree of parallelism: 32	
cutoff：62100	**10times Time:3905ms**
cutoff：62200	**10times Time:2390ms**
cutoff：62300	**10times Time:2109ms**
cutoff：62400	**10times Time:2218ms**
cutoff：62500	**10times Time:2340ms**

cutoff：62600	**10times Time:1516ms**
cutoff：62700	**10times Time:1577ms**
cutoff：62800	**10times Time:1594ms**
cutoff：62900	**10times Time:1531ms**
cutoff：63000	**10times Time:1562ms**

We could find that in this case, time drops a lot after the cutoff is more than 62,500.

At which time, $\frac{arrays.length}{threadNumber} = \frac{2000000}{32} = 62{,}500$.

(5)Output when threadNumber = 64 and array's length = 2,000,000



Output text:

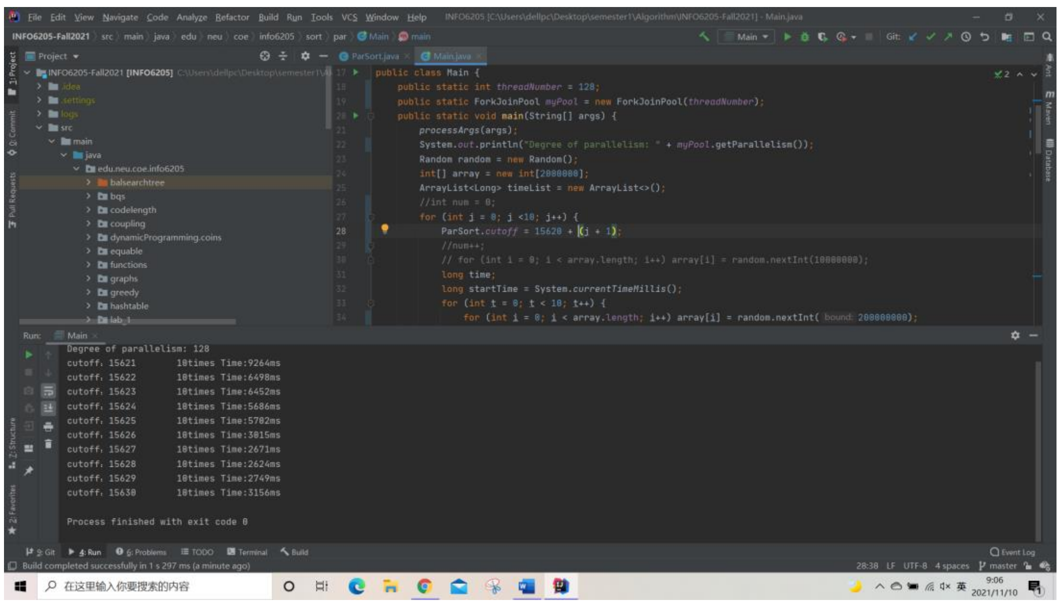
Degree of parallelism: 64	
cutoff：31210	**10times Time:6155ms**
cutoff：31220	**10times Time:3390ms**
cutoff：31230	**10times Time:3437ms**
cutoff：31240	**10times Time:3171ms**
cutoff：31250	**10times Time:3405ms**
cutoff：31260	**10times Time:2218ms**
cutoff：31270	**10times Time:1969ms**

cutoff：31280	10times Time:2187ms
cutoff：31290	10times Time:2140ms
cutoff：31300	10times Time:1969ms

We could find that in this case, time drops a lot after the cutoff is more than 31,250.

At which time, $\frac{arrays.length}{threadNumber} = \frac{2000000}{64} = 31{,}250$.

(6)Output when threadNumber = 128 and array's length = 2,000,000



Output text:

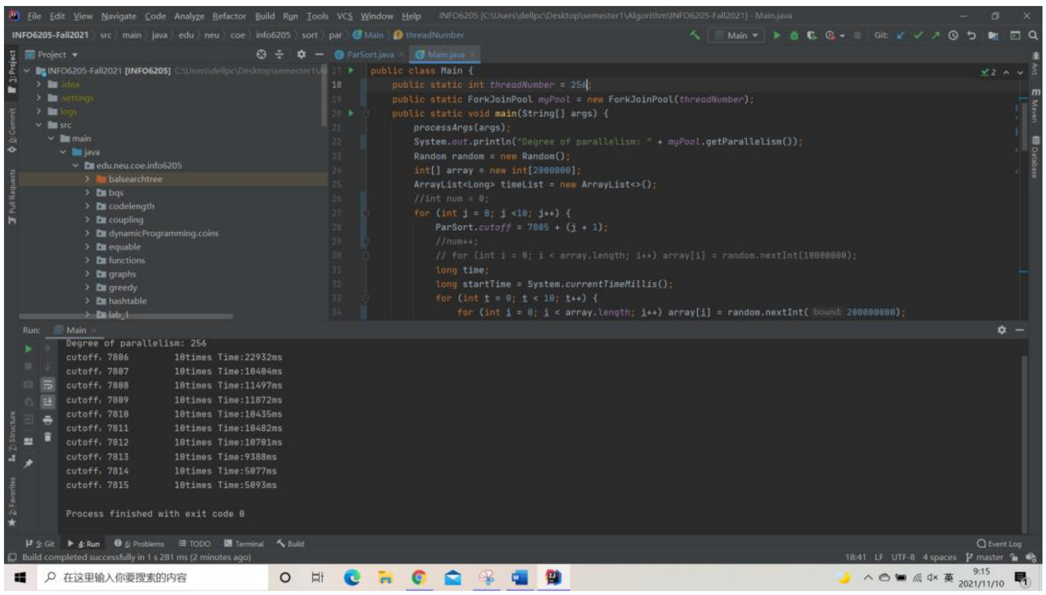
Degree of parallelism: 128	
cutoff：15621	10times Time:9264ms
cutoff：15622	10times Time:6498ms
cutoff：15623	10times Time:6452ms
cutoff：15624	10times Time:5686ms
cutoff：15625	10times Time:5702ms
cutoff：15626	10times Time:3015ms
cutoff：15627	10times Time:2671ms
cutoff：15628	10times Time:2624ms

cutoff：15629	**10times Time:2749ms**
cutoff：15630	**10times Time:3156ms**

We could find that in this case, time drops a lot after the cutoff is more than 15,625.

At which time, $\frac{arrays.length}{threadNumber} = \frac{2000000}{128} = 15{,}625$.

(7)Output when threadNumber = 256 and array's length = 2,000,000



Output text:

Degree of parallelism: 256	
cutoff：7806	**10times Time:22932ms**
cutoff：7807	**10times Time:10404ms**
cutoff：7808	**10times Time:11497ms**
cutoff：7809	**10times Time:11872ms**
cutoff：7810	**10times Time:10435ms**
cutoff：7811	**10times Time:10482ms**
cutoff：7812	**10times Time:10701ms**
cutoff：7813	**10times Time:9388ms**
cutoff：7814	**10times Time:5077ms**

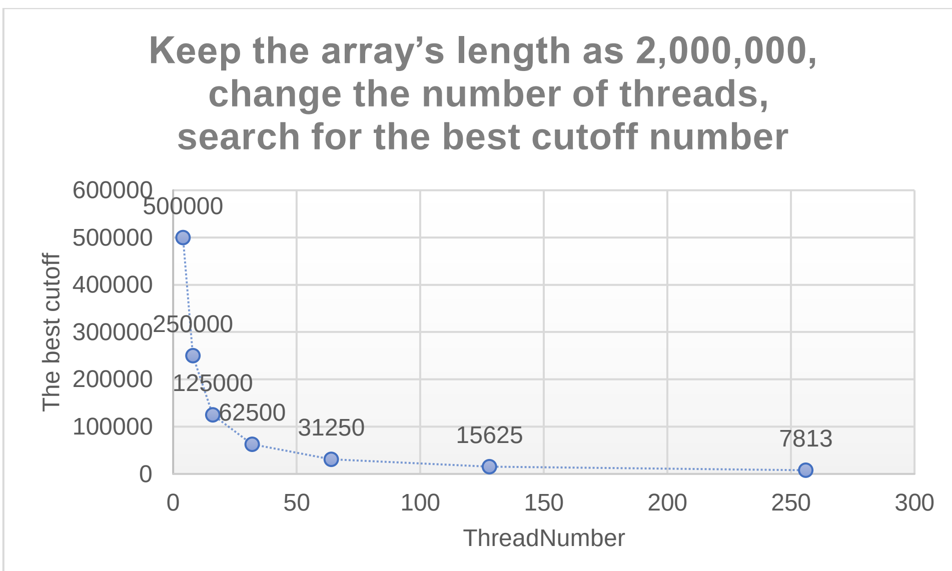
cutoff：7815	10times Time:5093ms

We could find that in this case, time drops a lot after the cutoff is more than 7,813.

At which time, $\frac{arrays.length}{threadNumber} = \frac{2000000}{256} = 7{,}812.5$.

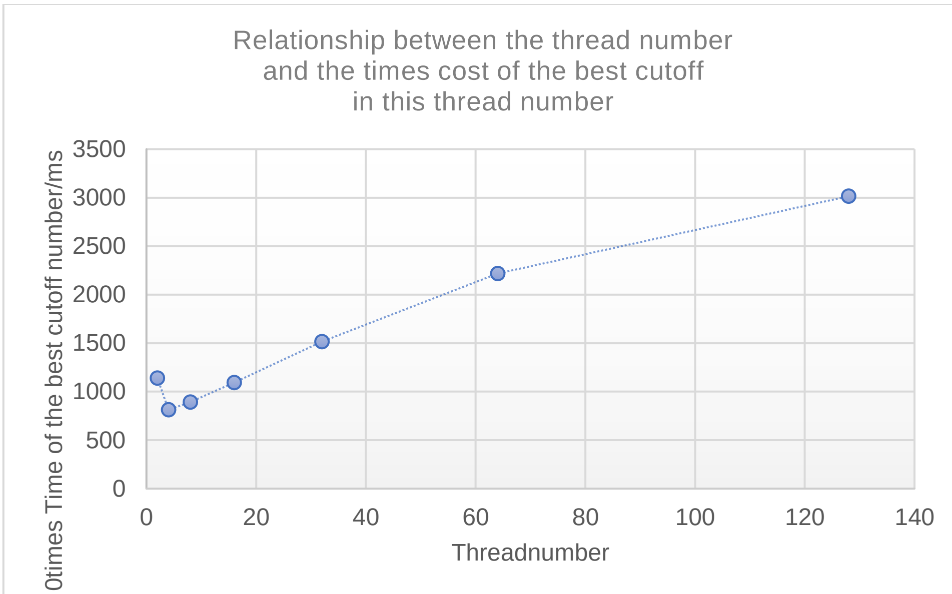
The simulation between the thread number and the best cutoff number when array’s length is unchangeable is showed below.

Threadnumber	The best cutoff
4	500000
8	250000
16	125000
32	62500
64	31250
128	15625
256	7813



It shows that y-axis is inversely proportional to x, which means with the growth of the thread numbers, the best cutoffs decrease. So in the function to get the best cutoff number, the thread number should be in the position of the denominator.

Threadnumber	10times Time of the best cutoff number/ms
2	1140
4	812
8	891
16	1094
32	1516
64	2218
128	3015
256	5077

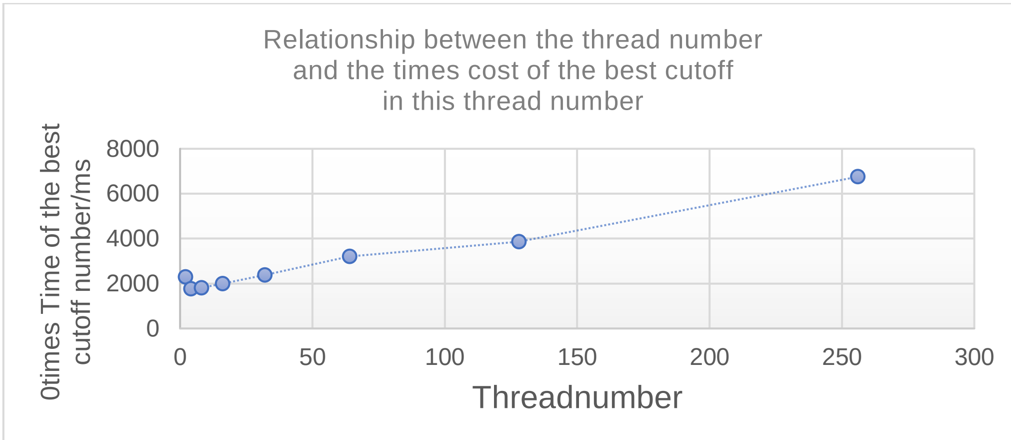


We can also make a conclusion that, though each thread number has its' own best cutoff number, the best cutoffs' time costs are different. In this simulation, we could find that when the thread number is 4, and the array's length is 2,000,000, it takes least time in the ParSort. In other words, the ParSort does best when there are 4 threads when the array's length is unchangeable.

To test if the thread number is 4 is best fits the ParSort, we could also make tests to test it. Because the steps are same as the above, I just give out the data and simulation directly.

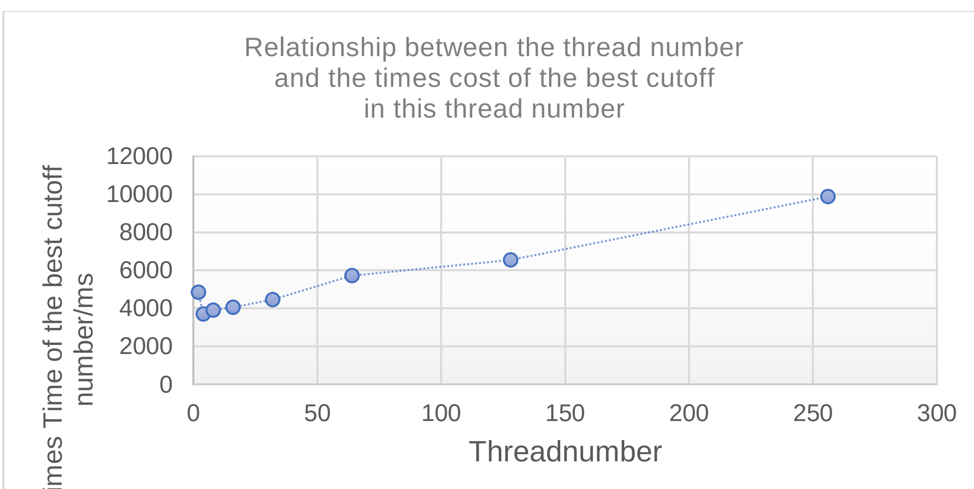
When the array's length = 4,000,000

Threadnumber	10times Time of the best cutoff number/ms
2	2296
4	1765
8	1812
16	1999
32	2374
64	3202
128	3859
256	6753



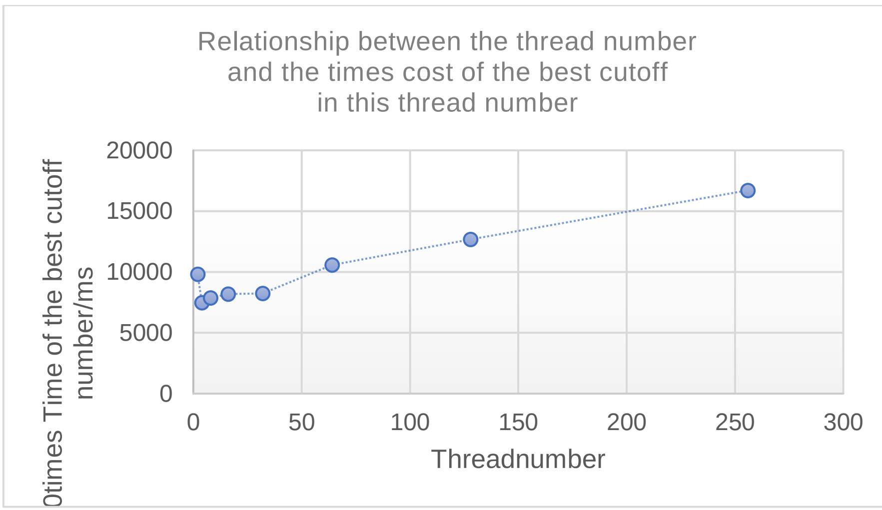
When the array’s length = 8,000,000

Threadnumber	10times Time of the best cutoff number/ms
2	4843
4	3702
8	3906
16	4046
32	4452
64	5717
128	6546
256	9873

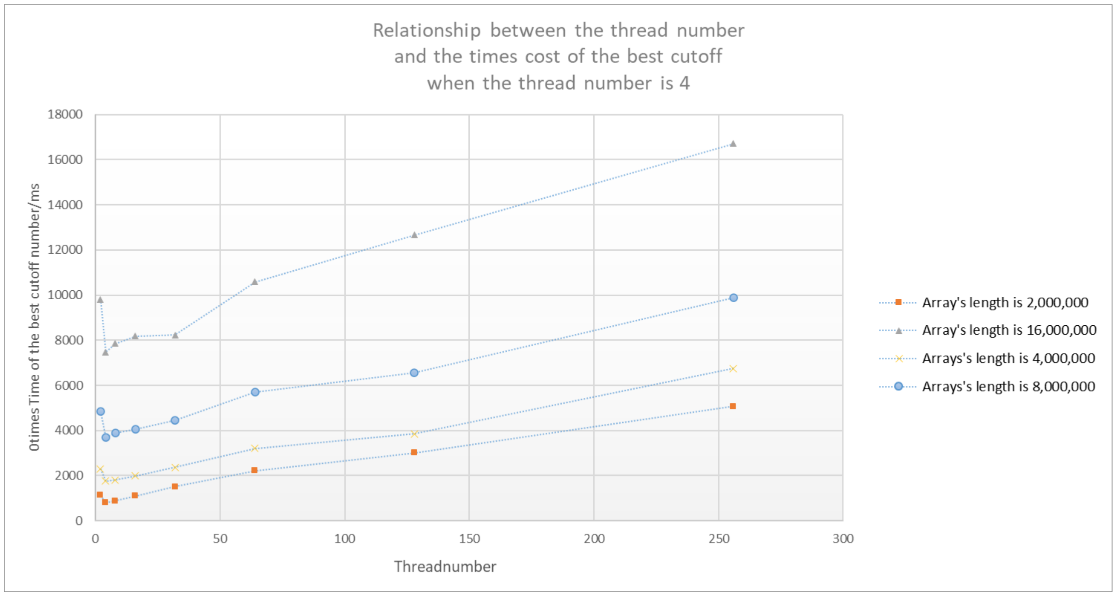


When the array's length = 16,000,000

Threadnumber	10times Time of the best cutoff number/ms
2	9810
4	7482
8	7858
16	8186
32	8233
64	10591
128	12669
256	16714



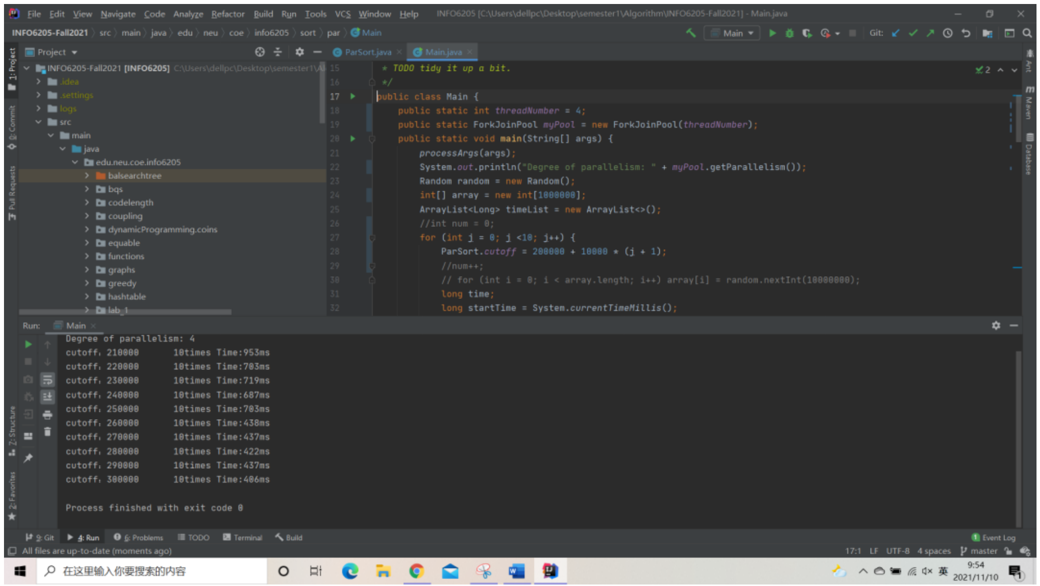
We could see that even the length of the array changes, the best cutoff numbers always cost the least time when the thread number is 4, which confirms the previous conclusion, that the ParSort does best when there are 4 threads when the array's length is unchangeable.



We could also find in the above diagram that the time costs of the best cutoff numbers increase with the number of threads when the number of threads is not 4, which means the increase of thread numbers doesn't improve the ParSort's performance. Why it happens? Personally, I think the main reason is, if multiple threads operate on the same atomic data synchronously, the resource may also be occupied easily.

③ Keep the number of threads unchangeable, change the array's length, search for the best cutoff number

(1) Output when threadNumber = 4 and array's length = 1,000,000



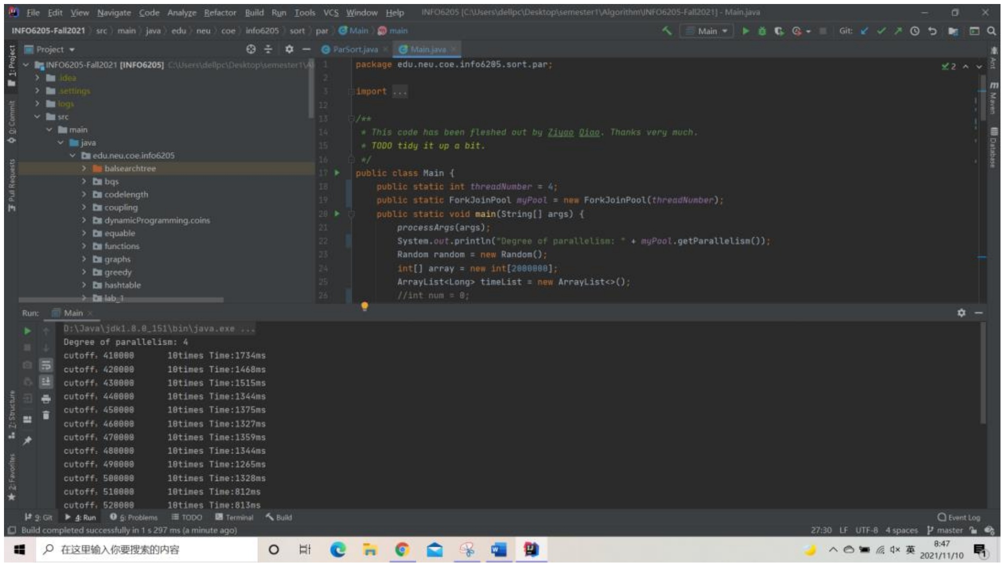
Output text:

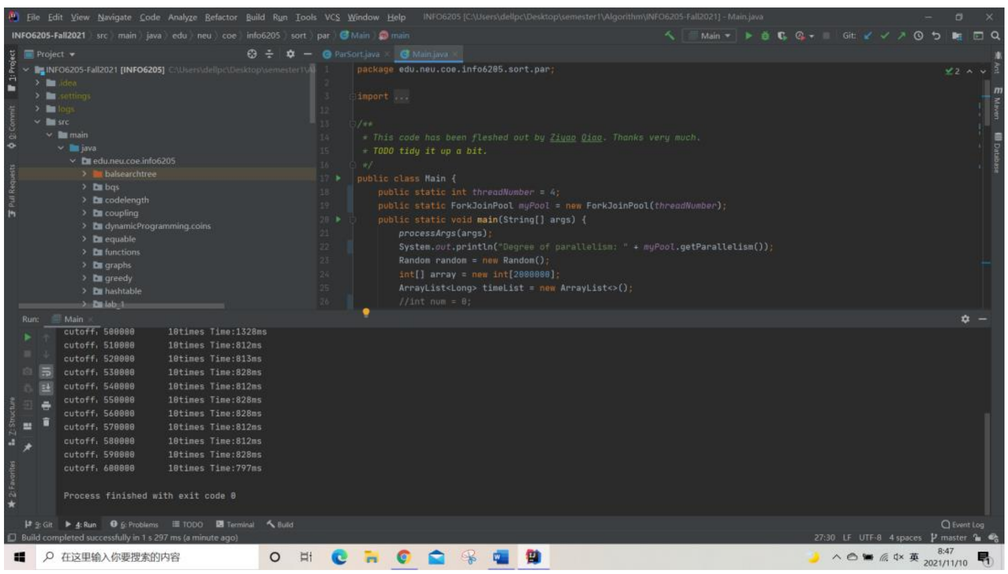
Degree of parallelism: 4	
cutoff：210000	**10times Time:953ms**
cutoff：220000	**10times Time:703ms**
cutoff：230000	**10times Time:719ms**
cutoff：240000	**10times Time:687ms**
cutoff：250000	**10times Time:703ms**
cutoff：260000	**10times Time:438ms**
cutoff：270000	**10times Time:437ms**
cutoff：280000	**10times Time:422ms**
cutoff：290000	**10times Time:437ms**
cutoff：300000	**10times Time:406ms**

We could find that in this case, time drops a lot after the cutoff is more than 250,000.

At which time, $\frac{arrays.length}{threadNumber} = \frac{1000000}{4} = 250{,}000$.

(2)Output when threadNumber = 4 and array's length = 2,000,000





Output text:

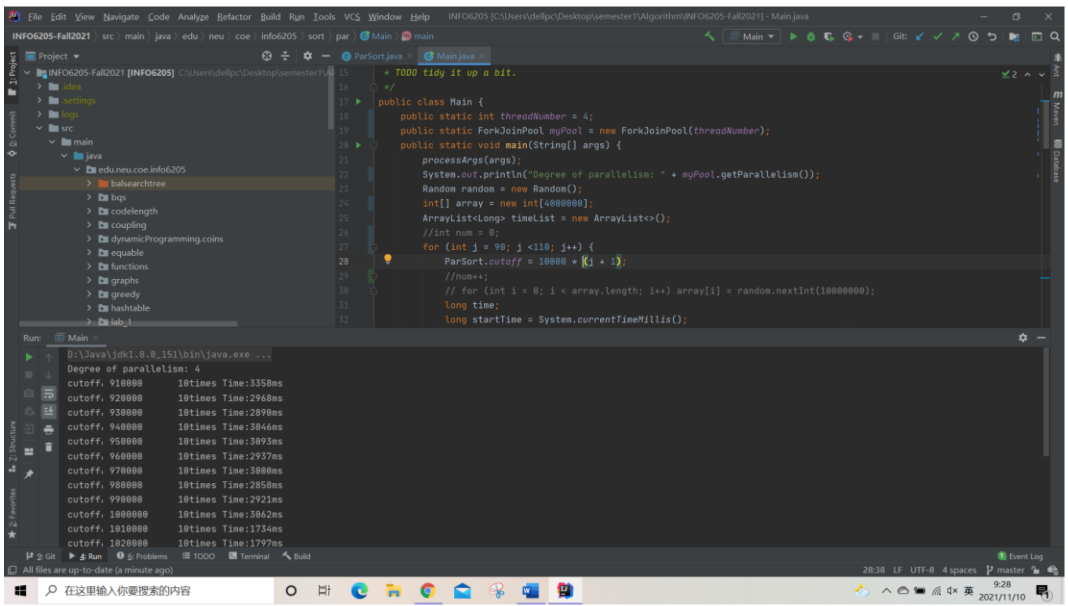
Degree of parallelism: 4	
cutoff： 450000	10times Time:1375ms
cutoff： 460000	10times Time:1327ms
cutoff： 470000	10times Time:1359ms
cutoff： 480000	10times Time:1344ms
cutoff： 490000	10times Time:1265ms
cutoff： 500000	10times Time:1328ms
cutoff： 510000	10times Time:812ms
cutoff： 520000	10times Time:813ms

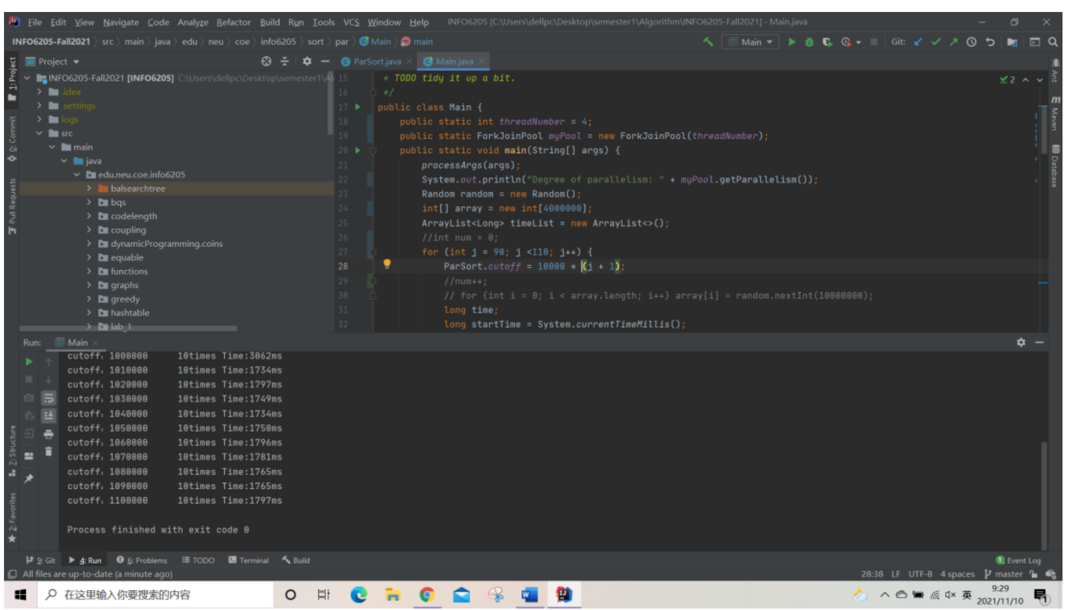
cutoff：530000	**10times Time:828ms**
cutoff：540000	**10times Time:812ms**
cutoff：550000	**10times Time:828ms**

We could find that in this case, time drops a lot after the cutoff is more than 500,000.

At which time, $\frac{arrays.length}{threadNumber} = \frac{2000000}{4} = 500{,}000$.

(3)Output when threadNumber = 4 and array's length = 4,000,000





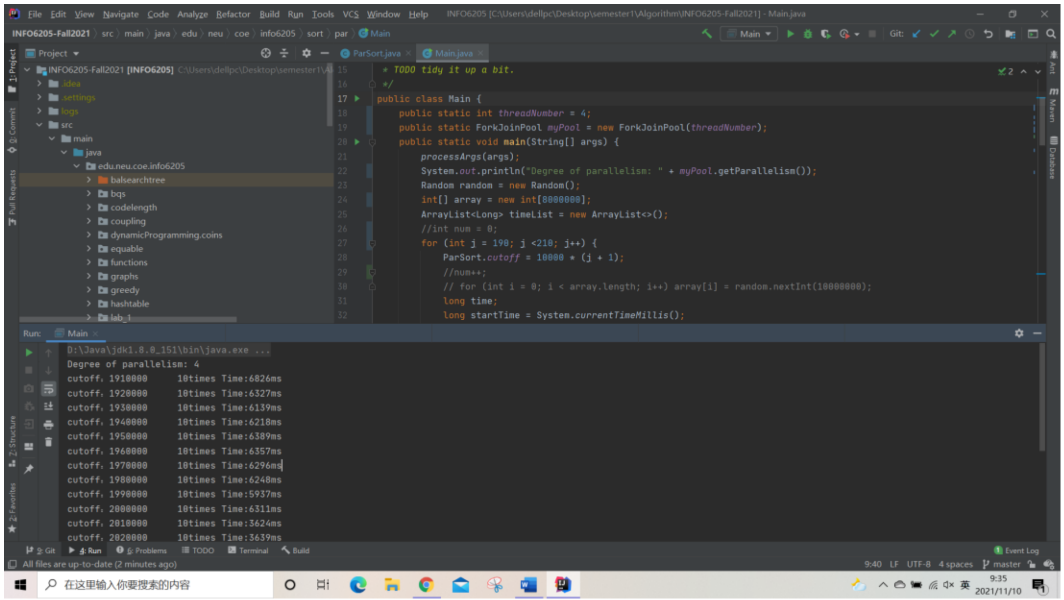
Output text:

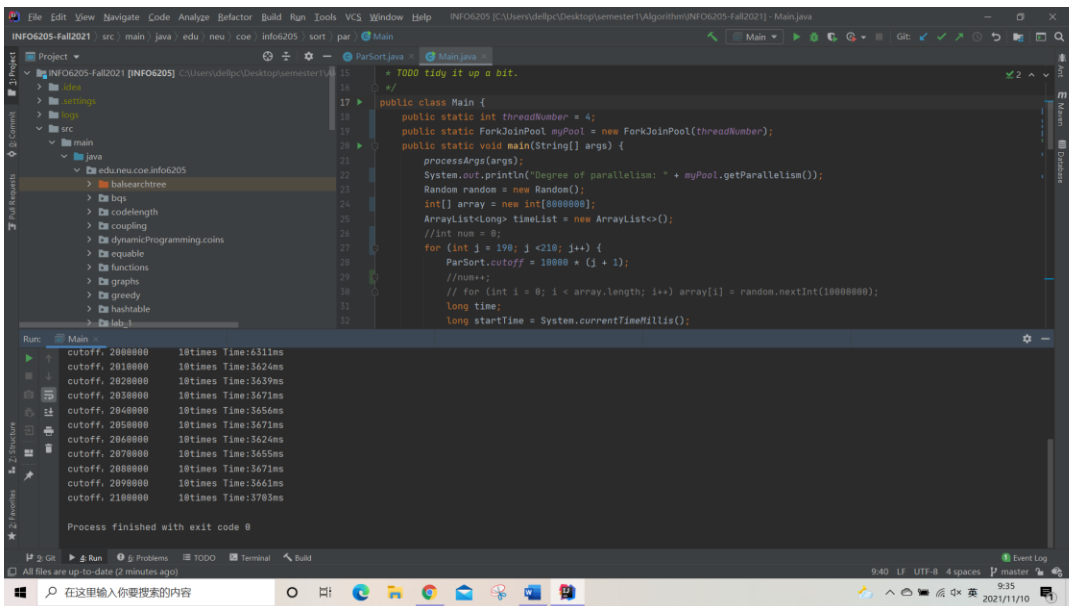
```
Degree of parallelism: 4
cutoff：910000          10times Time:3358ms
cutoff：920000          10times Time:2968ms
cutoff：930000          10times Time:2890ms
cutoff：940000          10times Time:3046ms
cutoff：950000          10times Time:3093ms
cutoff：960000          10times Time:2937ms
cutoff：970000          10times Time:3000ms
cutoff：980000          10times Time:2858ms
cutoff：990000          10times Time:2921ms
cutoff：1000000         10times Time:3062ms
cutoff：1010000         10times Time:1734ms
cutoff：1020000         10times Time:1797ms
cutoff：1030000         10times Time:1749ms
cutoff：1040000         10times Time:1734ms
cutoff：1050000         10times Time:1750ms
cutoff：1060000         10times Time:1796ms
cutoff：1070000         10times Time:1781ms
cutoff：1080000         10times Time:1765ms
cutoff：1090000         10times Time:1765ms
cutoff：1100000         10times Time:1797ms
```

We could find that in this case, time drops a lot after the cutoff is more than 1,000,000.

At which time, $\frac{arrays.length}{threadNumber} = \frac{4000000}{4} = 1{,}000{,}000$.

(4)Output when threadNumber = 4 and array's length = 8,000,000





Output text:

Degree of parallelism: 4

cutoff：1950000 10times Time:6389ms

cutoff：1960000 10times Time:6357ms

cutoff：1970000 10times Time:6296ms

cutoff：1980000 10times Time:6248ms

cutoff：1990000 10times Time:5937ms

cutoff：2000000 10times Time:6311ms

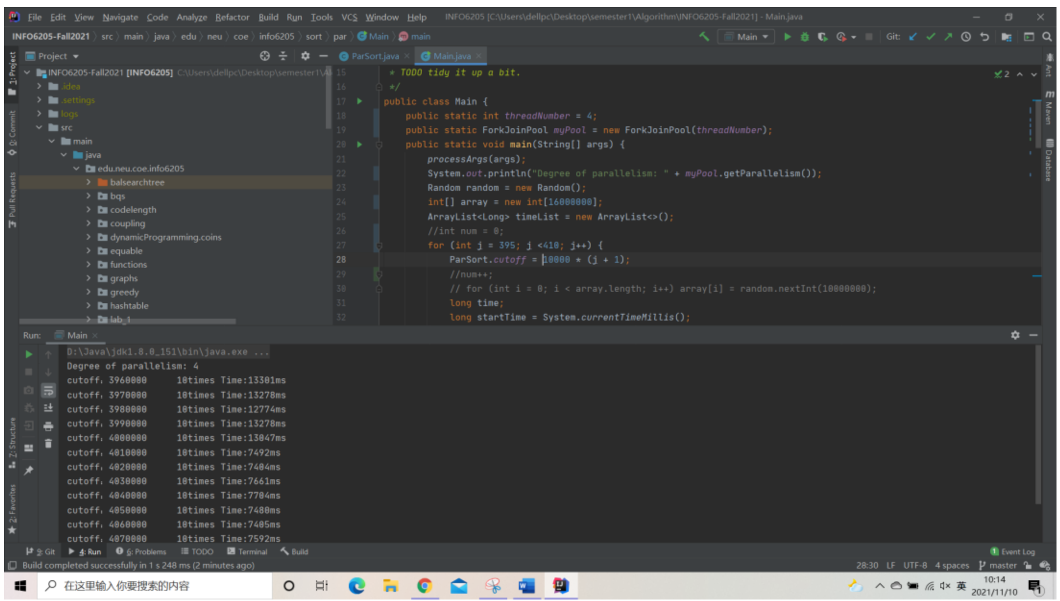
cutoff：2010000 10times Time:3624ms

cutoff：2020000	**10times Time:3639ms**
cutoff：2030000	**10times Time:3671ms**
cutoff：2040000	**10times Time:3656ms**
cutoff：2050000	**10times Time:3671ms**

We could find that in this case, time drops a lot after the cutoff is more than 2,000,000.

At which time, $\frac{arrays.length}{threadNumber} = \frac{8000000}{4} = 2{,}000{,}000$.

(5)Output when threadNumber = 4 and array's length = 16,000,000



Output text:

Degree of parallelism: 4	
cutoff：3960000	**10times Time:13301ms**
cutoff：3970000	**10times Time:13278ms**
cutoff：3980000	**10times Time:12774ms**
cutoff：3990000	**10times Time:13278ms**
cutoff：4000000	**10times Time:13047ms**
cutoff：4010000	**10times Time:7492ms**
cutoff：4020000	**10times Time:7404ms**
cutoff：4030000	**10times Time:7661ms**

cutoff：4040000 10times Time:7704ms

cutoff：4050000 10times Time:7480ms

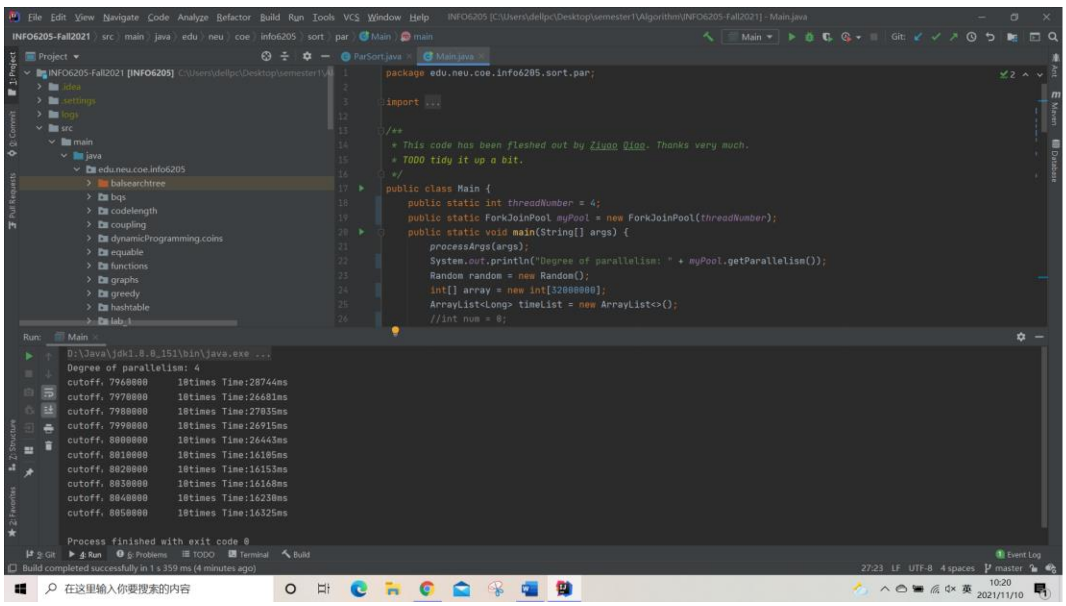
cutoff：4060000 10times Time:7405ms

cutoff：4070000 10times Time:7592ms

We could find that in this case, time drops a lot after the cutoff is more than 4,000,000.

At which time, $\frac{arrays.length}{threadNumber} = \frac{16000000}{4} = 4{,}000{,}000$.

(6)Output when threadNumber = 4 and array’s length = 32,000,000



Output text:

Degree of parallelism: 4

cutoff：7960000 10times Time:28744ms

cutoff：7970000 10times Time:26681ms

cutoff：7980000 10times Time:27035ms

cutoff：7990000 10times Time:26915ms

cutoff：8000000 10times Time:26443ms

cutoff：8010000 10times Time:16105ms

cutoff：8020000 10times Time:16153ms

cutoff：8030000 10times Time:16168ms

cutoff：8040000	**10times Time:16230ms**
cutoff：8050000	**10times Time:16325ms**

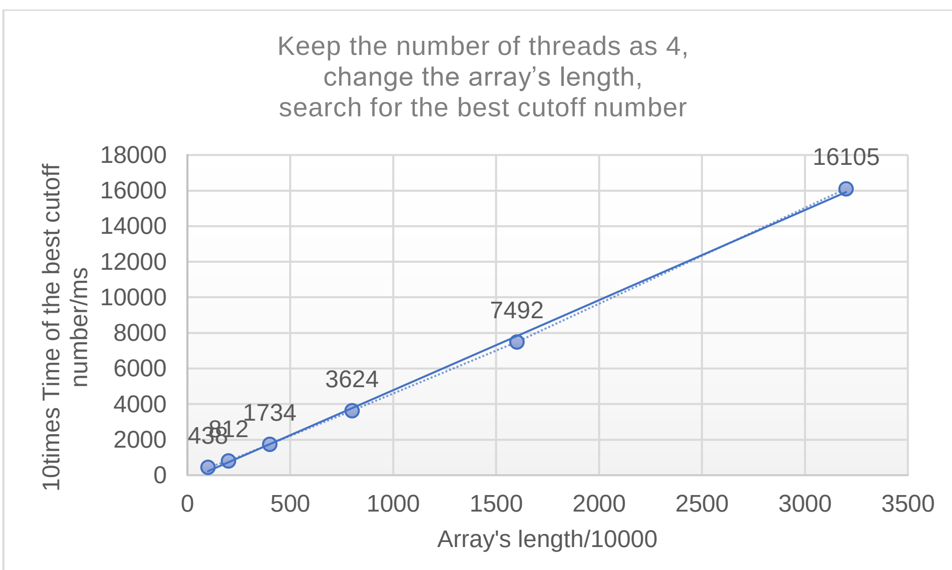
We could find that in this case, time drops a lot after the cutoff is more than 8,000,000.

At which time, $\frac{arrays.length}{threadNumber} = \frac{32000000}{4} = 8{,}000{,}000$.

It seems that when threadNumber = 4 and array’s length = 64,000,000, my computer didn’t work well so I stopped the simulation.

The simulation between the array’s length and the best cutoff number when the number of threads is unchangeable is showed below.

Array's length/10000	10times Time of the best cutoff number/ms	x/y
100	438	4.38
200	812	4.06
400	1734	4.36
800	3624	4.53
1600	7492	4.68
3200	16105	5.03



It shows that the 10times Time of the best cutoff is proportional to the array's length, and the proportionality factor is around 4, which is the same as the unchangeable thread number.

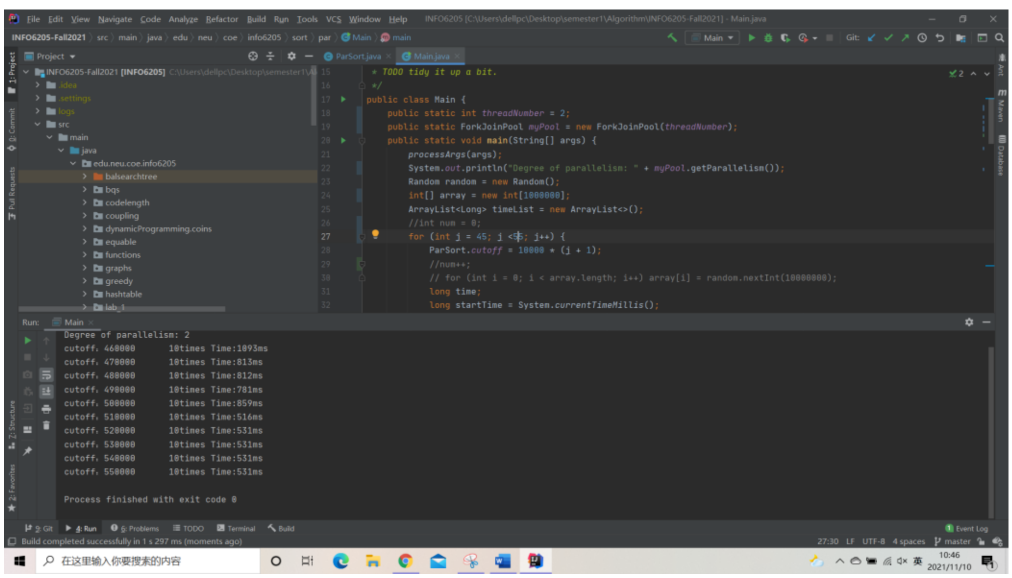
Combined with the analysis of the previous point ②, we could make a conclusion that the function between the number of threads, the array's length and the best cutoff number is as the following:

$$\textit{the best cutoff number} = \frac{\textit{arrays.length}}{\textit{threadNumber}}$$

We could make tests for our conclusion.

④ Keep the $\frac{arrays.length}{threadNumber}$ unchangeable, search for the best cutoff number.

(1) Output when threadNumber = 2 and array's length = 1,000,000



Output text:

Degree of parallelism: 2	
cutoff：460000	10times Time:1093ms

cutoff：470000 10times Time:813ms

cutoff：480000 10times Time:812ms

cutoff：490000 10times Time:781ms

cutoff：500000 10times Time:859ms

cutoff：510000 10times Time:516ms

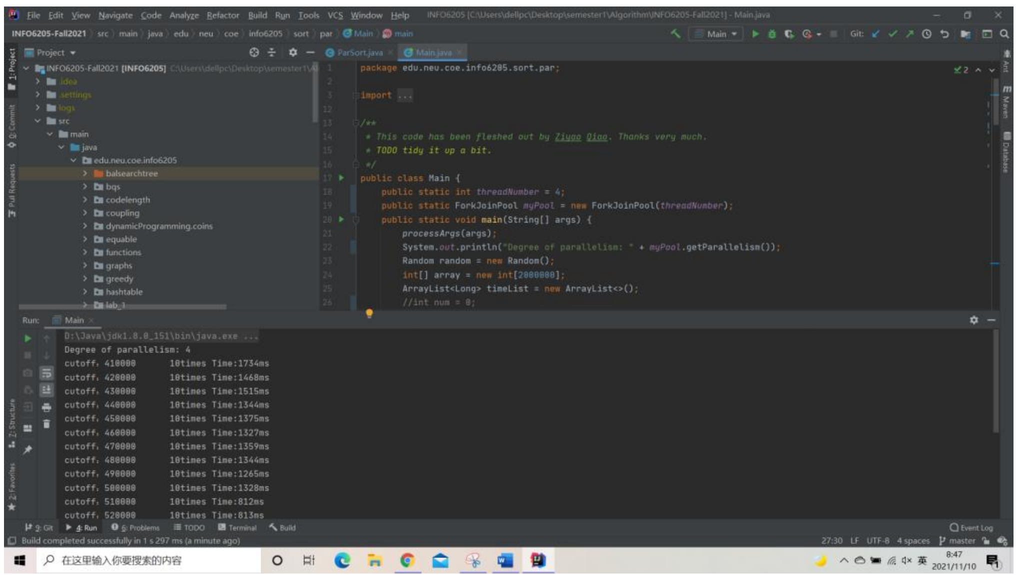
cutoff：520000 10times Time:531ms

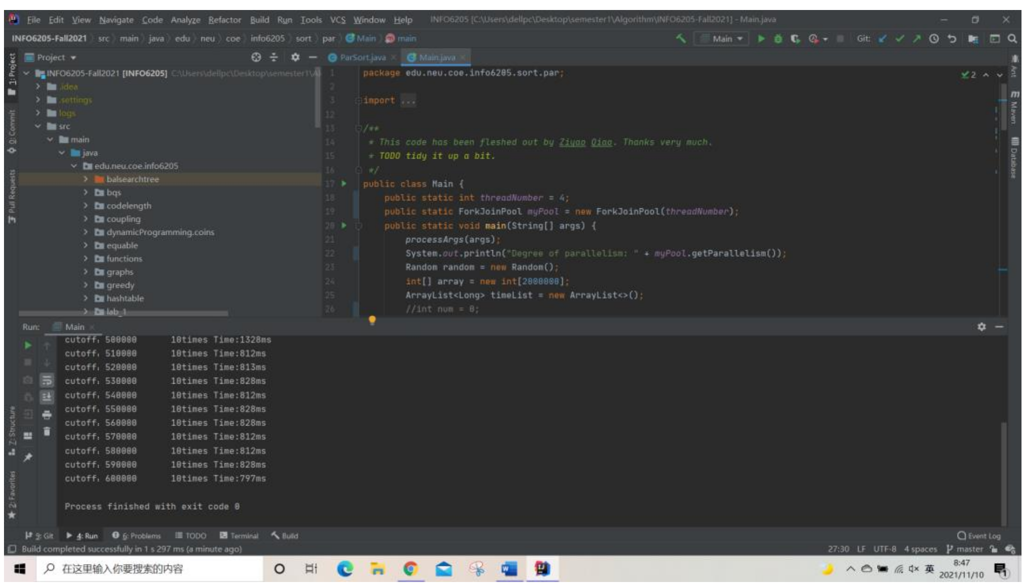
cutoff：530000 10times Time:531ms

cutoff：540000 10times Time:531ms

cutoff：550000 10times Time:531ms

(2)Output when threadNumber = 4 and array’s length = 2,000,000

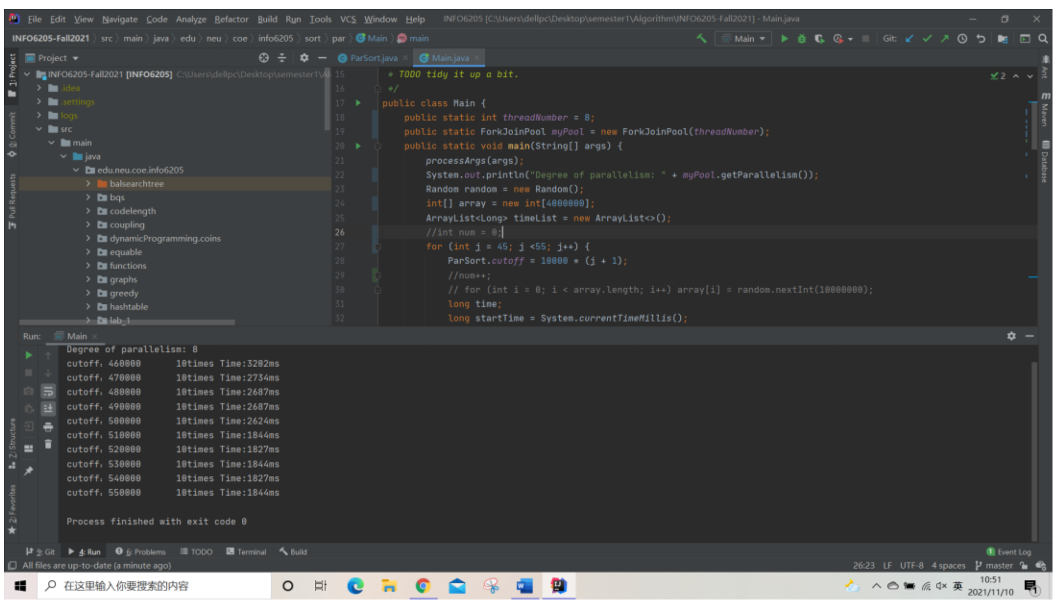




Output text:

Degree of parallelism: 4	
cutoff：450000	**10times Time:1375ms**
cutoff：460000	**10times Time:1327ms**
cutoff：470000	**10times Time:1359ms**
cutoff：480000	**10times Time:1344ms**
cutoff：490000	**10times Time:1265ms**
cutoff：500000	**10times Time:1328ms**
cutoff：510000	**10times Time:812ms**
cutoff：520000	**10times Time:813ms**
cutoff：530000	**10times Time:828ms**
cutoff：540000	**10times Time:812ms**
cutoff：550000	**10times Time:828ms**

(3) Output when threadNumber = 8 and array's length = 4,000,000

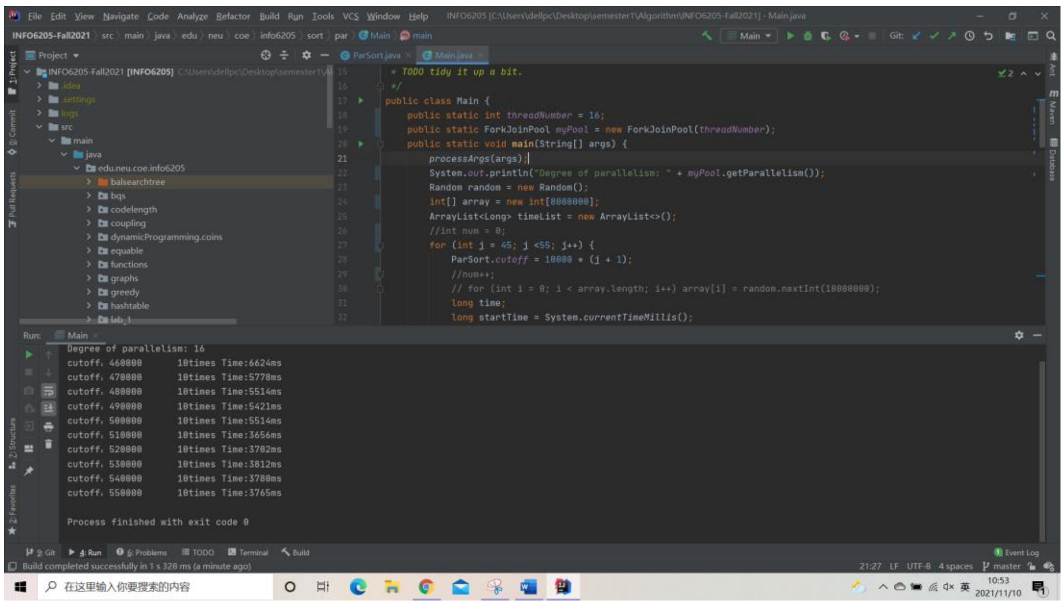


Output text:

Degree of parallelism: 8	
cutoff：460000	**10times Time:3202ms**
cutoff：470000	**10times Time:2734ms**
cutoff：480000	**10times Time:2687ms**
cutoff：490000	**10times Time:2687ms**

cutoff： 500000	10times Time:2624ms
cutoff： 510000	10times Time:1844ms
cutoff： 520000	10times Time:1827ms
cutoff： 530000	10times Time:1844ms
cutoff： 540000	10times Time:1827ms
cutoff： 550000	10times Time:1844ms

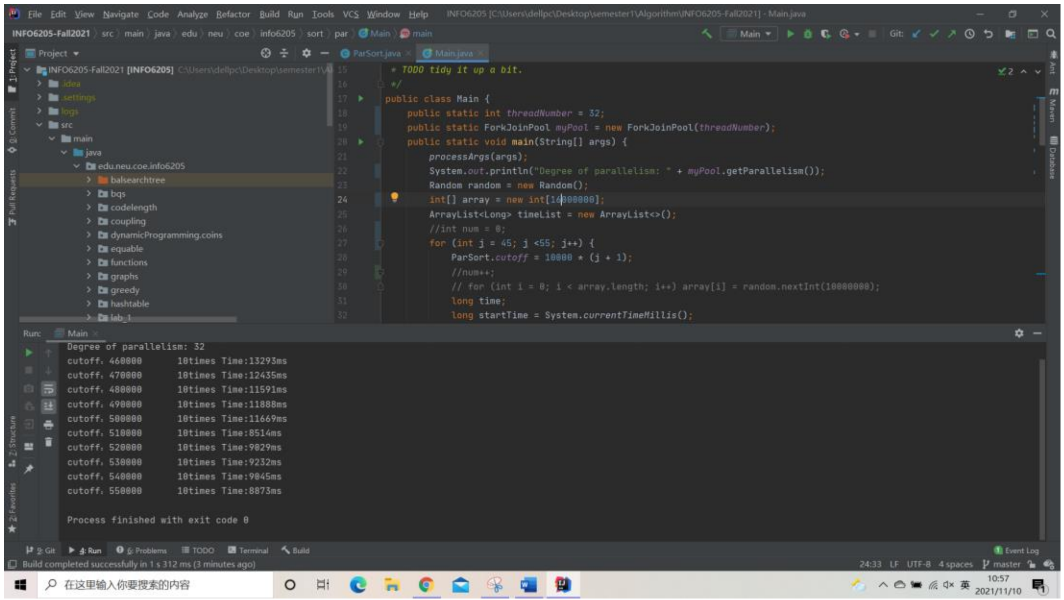
(4) Output when threadNumber = 16 and array's length = 8,000,000



Output text:

Degree of parallelism: 16	
cutoff： 460000	10times Time:6624ms
cutoff： 470000	10times Time:5778ms
cutoff： 480000	10times Time:5514ms
cutoff： 490000	10times Time:5421ms
cutoff： 500000	10times Time:5514ms
cutoff： 510000	10times Time:3656ms
cutoff： 520000	10times Time:3702ms
cutoff： 530000	10times Time:3812ms
cutoff： 540000	10times Time:3780ms
cutoff： 550000	10times Time:3765ms

(5) Output when threadNumber = 32 and array's length = 16,000,000



Output text:

Degree of parallelism: 32	
cutoff： 460000	10times Time:13293ms
cutoff： 470000	10times Time:12435ms
cutoff： 480000	10times Time:11591ms
cutoff： 490000	10times Time:11888ms
cutoff： 500000	10times Time:11669ms
cutoff： 510000	10times Time:8514ms
cutoff： 520000	10times Time:9029ms
cutoff： 530000	10times Time:9232ms
cutoff： 540000	10times Time:9045ms
cutoff： 550000	10times Time:8873ms

We could see in the tests, if the $\frac{arrays.length}{threadNumber}$ is unchangeable, the best cutoff number always occurs at the number of 500,000, which is equals to the number of $\frac{arrays.length}{threadNumber}$. The function seems right.