Varun Venkatesh Gowda (002126161)

Program Structures & Algorithms

Fall 2021

Assignment No. 5

Tasks:

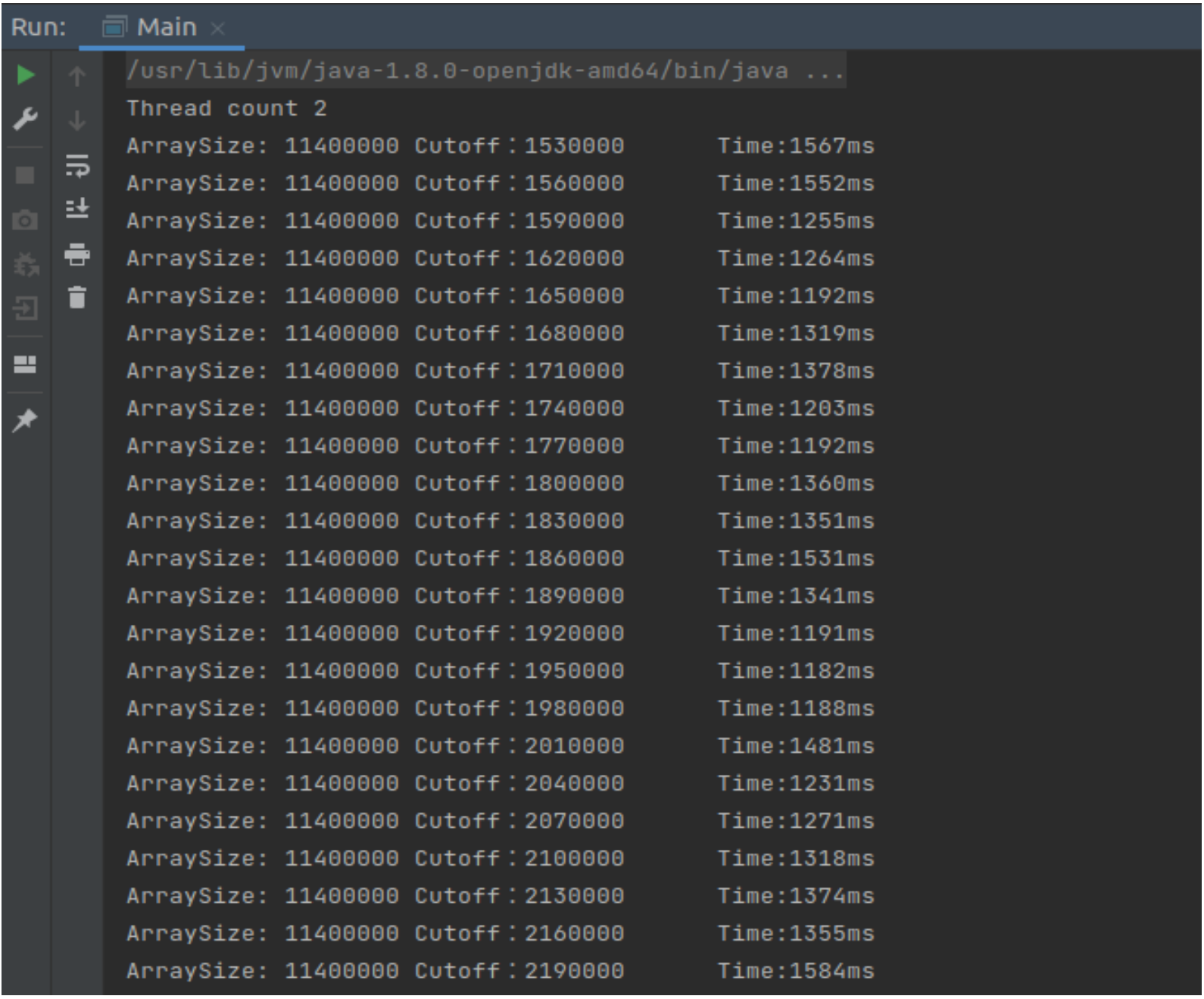
1. Implemented ForkJoinPool object instantiation in Main.java to set the number of threads to be made available.
2. Conducted experiments to determine what size of arrays justifies a parallel sort (large sized arrays work best)
3. Conducted experiments on a large sized array for cut-off values in three regions: near the start of length; near half the length; near the end of length.
4. Conducted above experiments on thread sizes ranging from 2, 4, 8 and 16.
5. Documented readings

Screenshots and Tabulations:

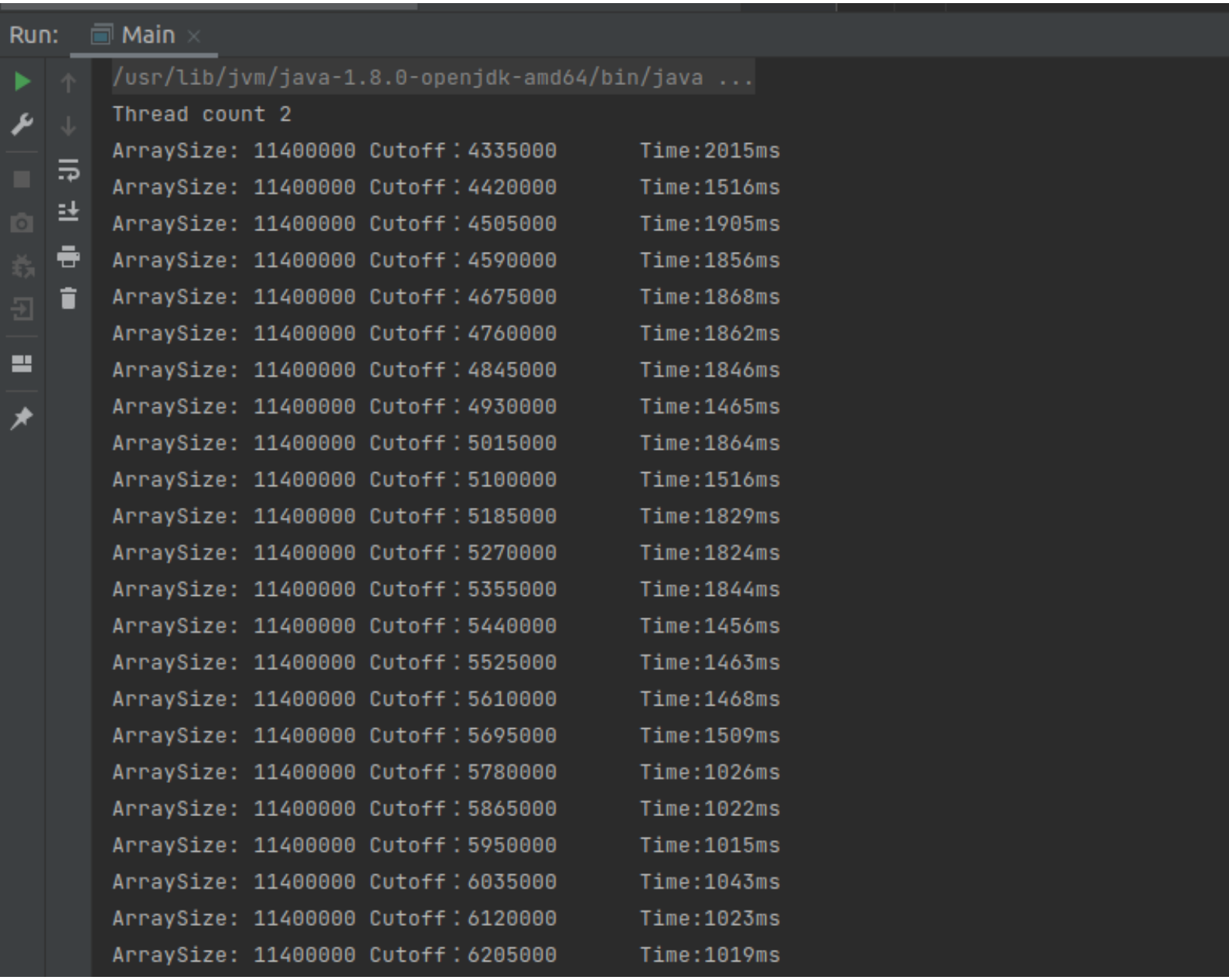
USE CASE: Array Size (N) = 11_400_000

For number of threads = 2 :

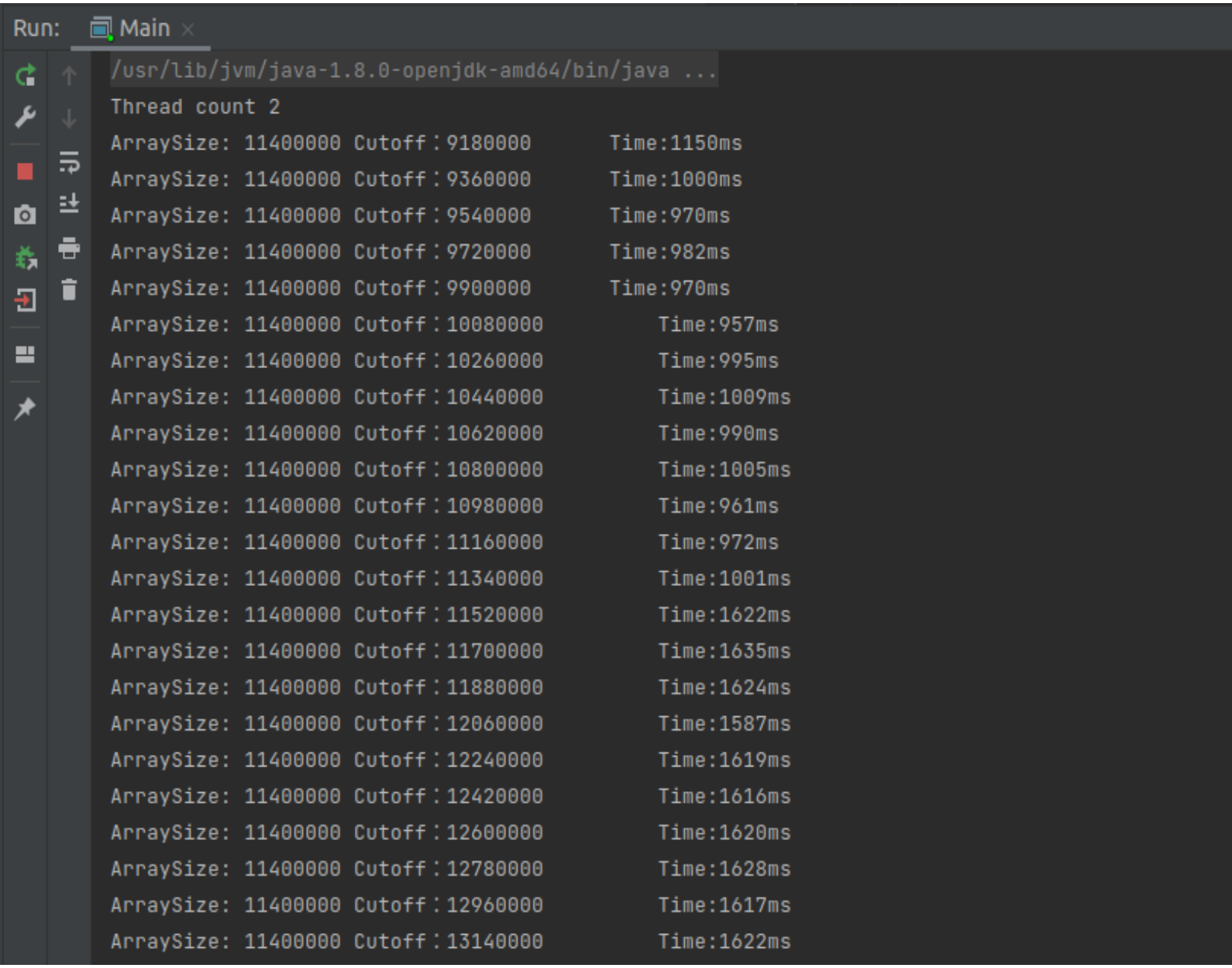
a) Cutoff towards the start of the length



b) Cutoff towards half the length

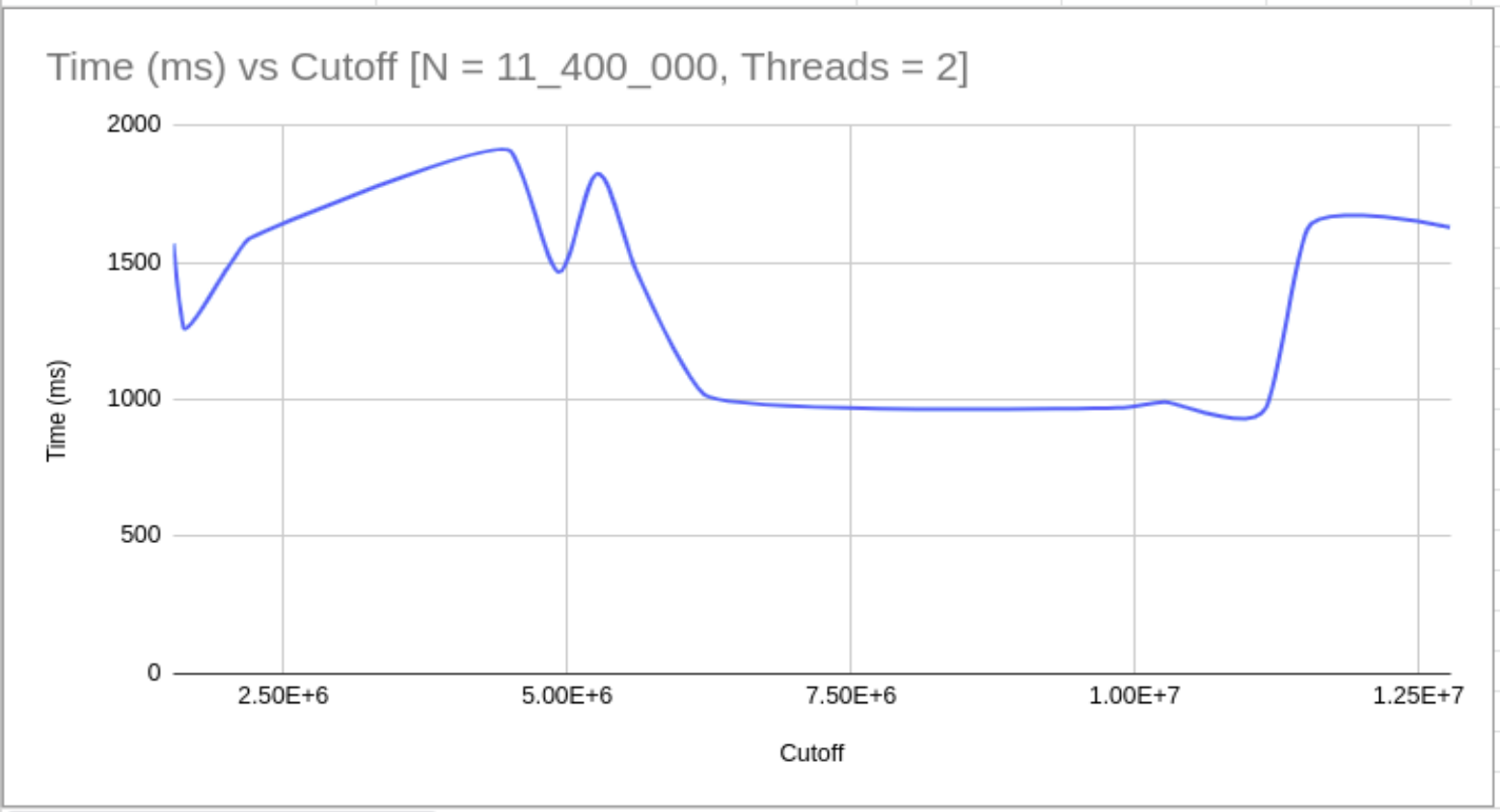


c) Cutoff towards the end of the length



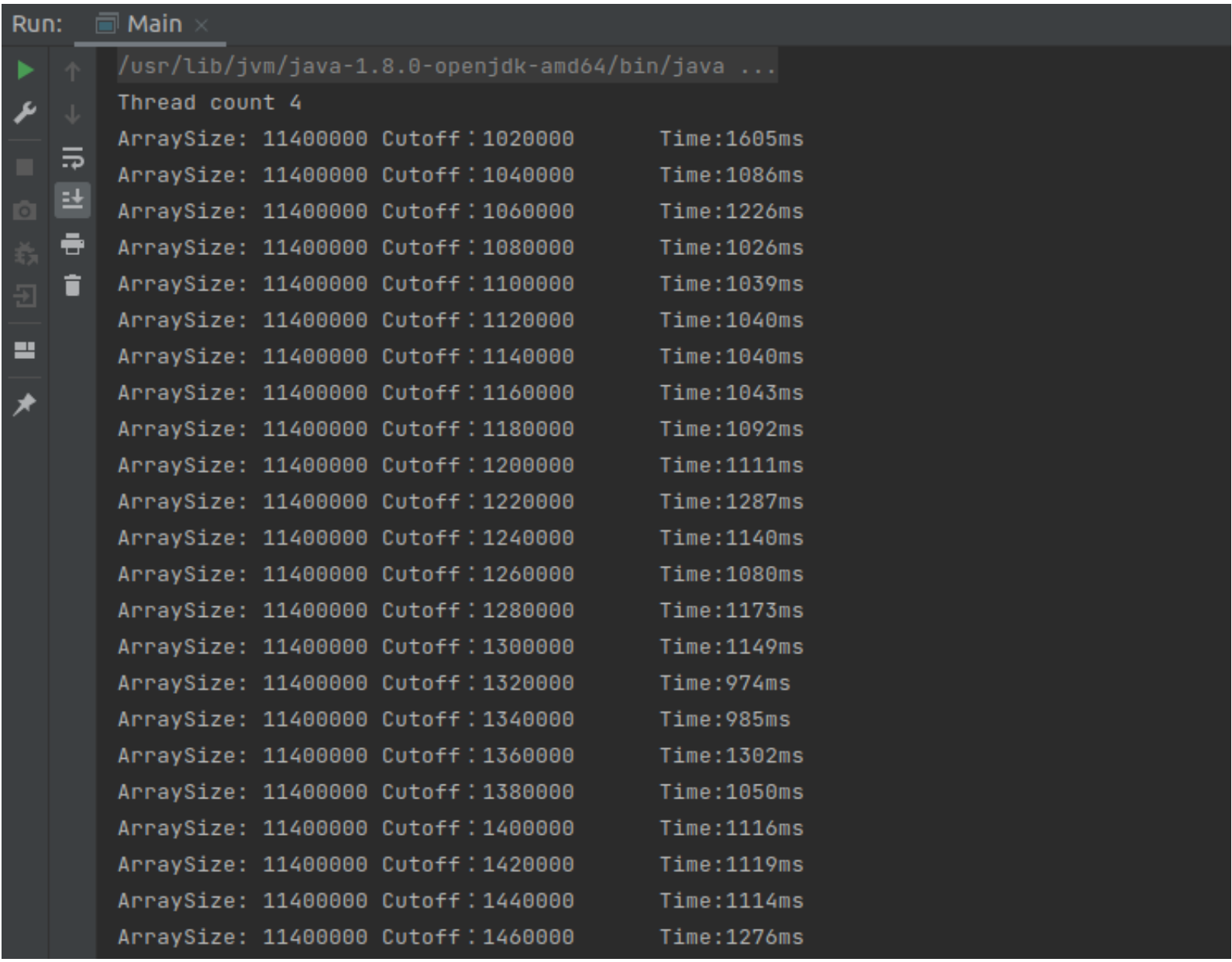
Tabulation:

A	B	C	D	E
Cutoff	Time (ms)			
1530000	1567			
1620000	1264			
2010000	1481			
2190000	1584			
4505000	1905			
4930000	1465			
5270000	1824			
5610000	1468	Halfway mark		
6205000	1019			
9900000	970			
10260000	990			
11160000	972			
11520000	1622			
12780000	1628			

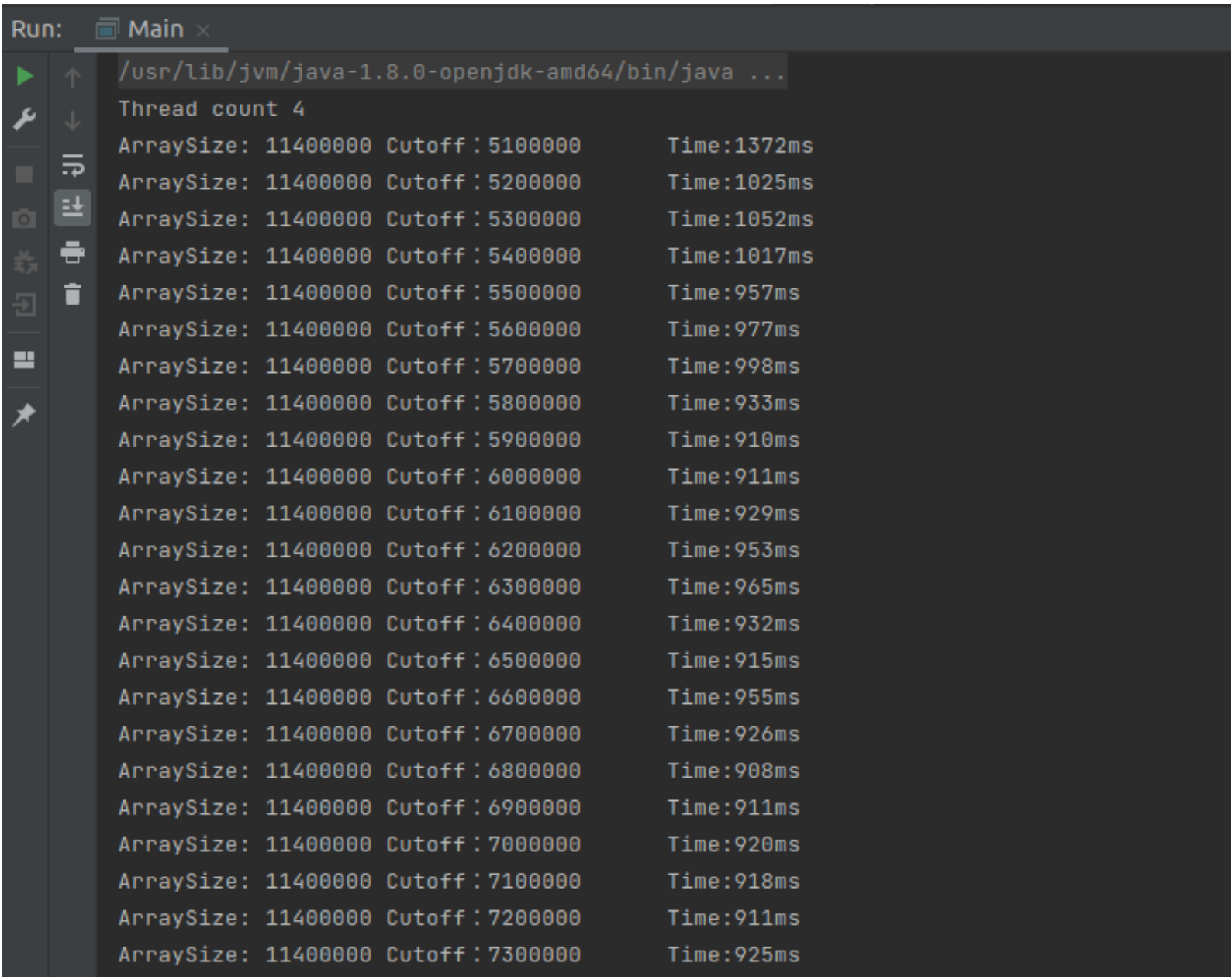


For number of threads = 4 :

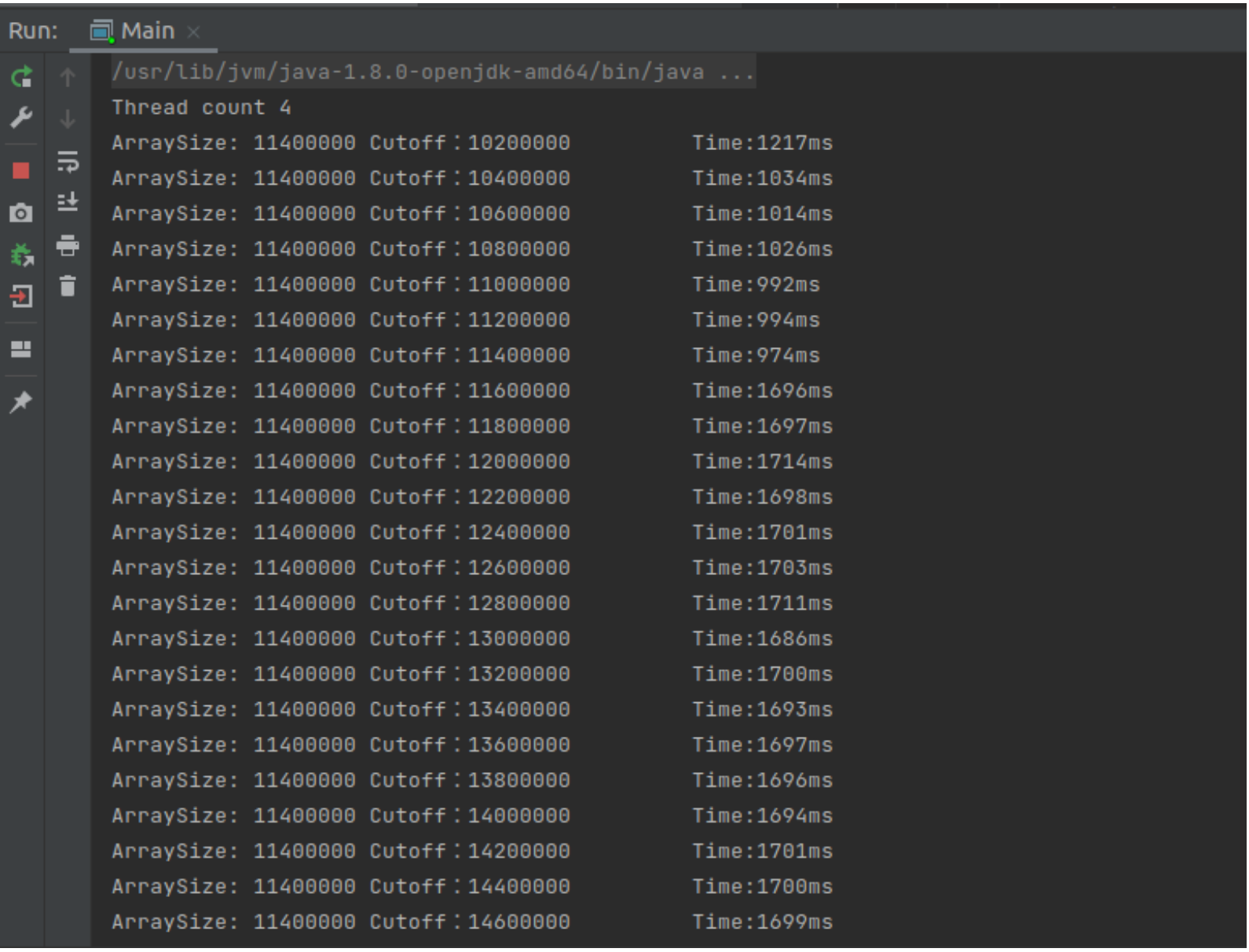
a) Cutoff towards the start of length



b) Cutoff towards half of length

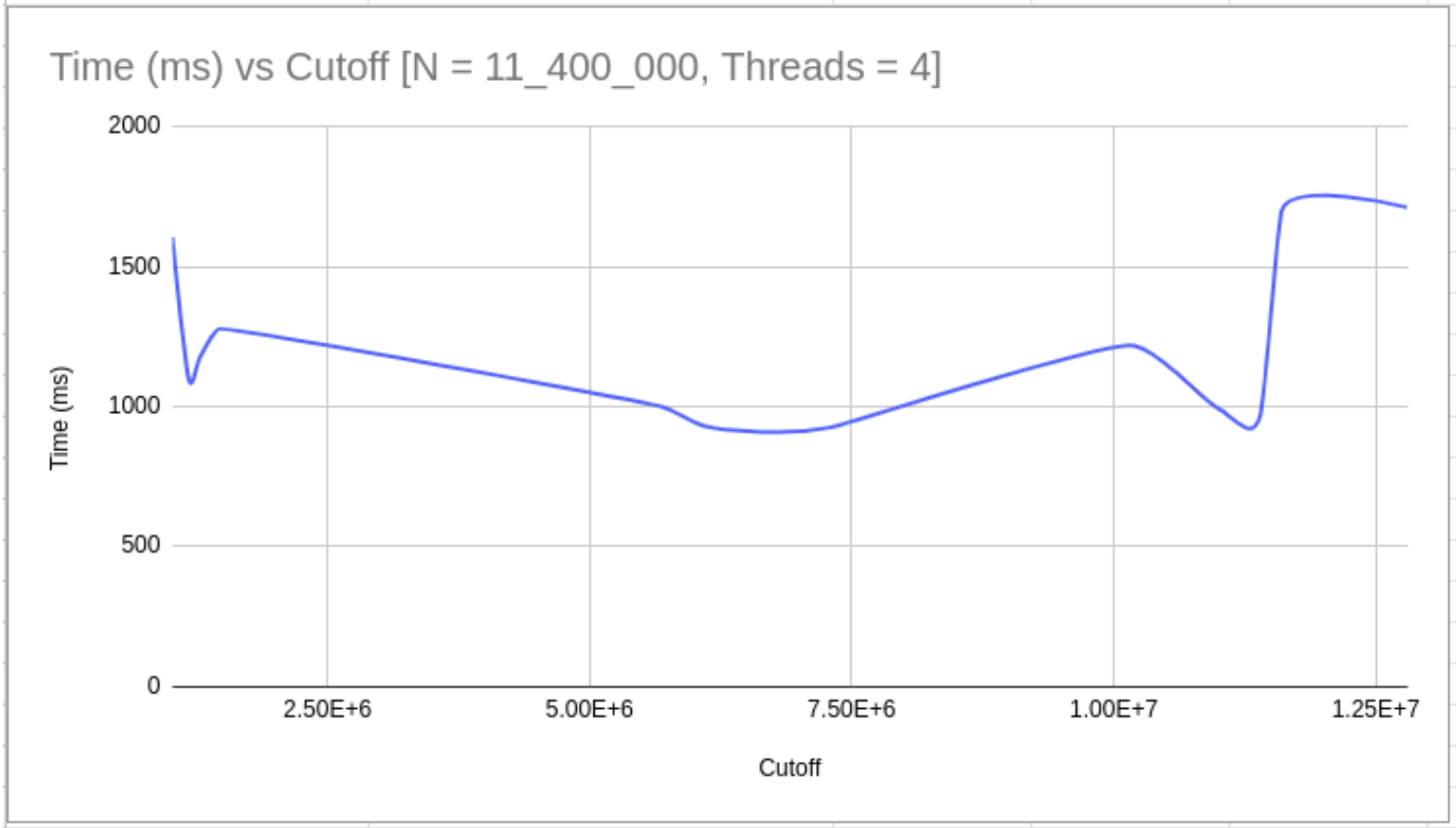


c) Cutoff towards the end of length



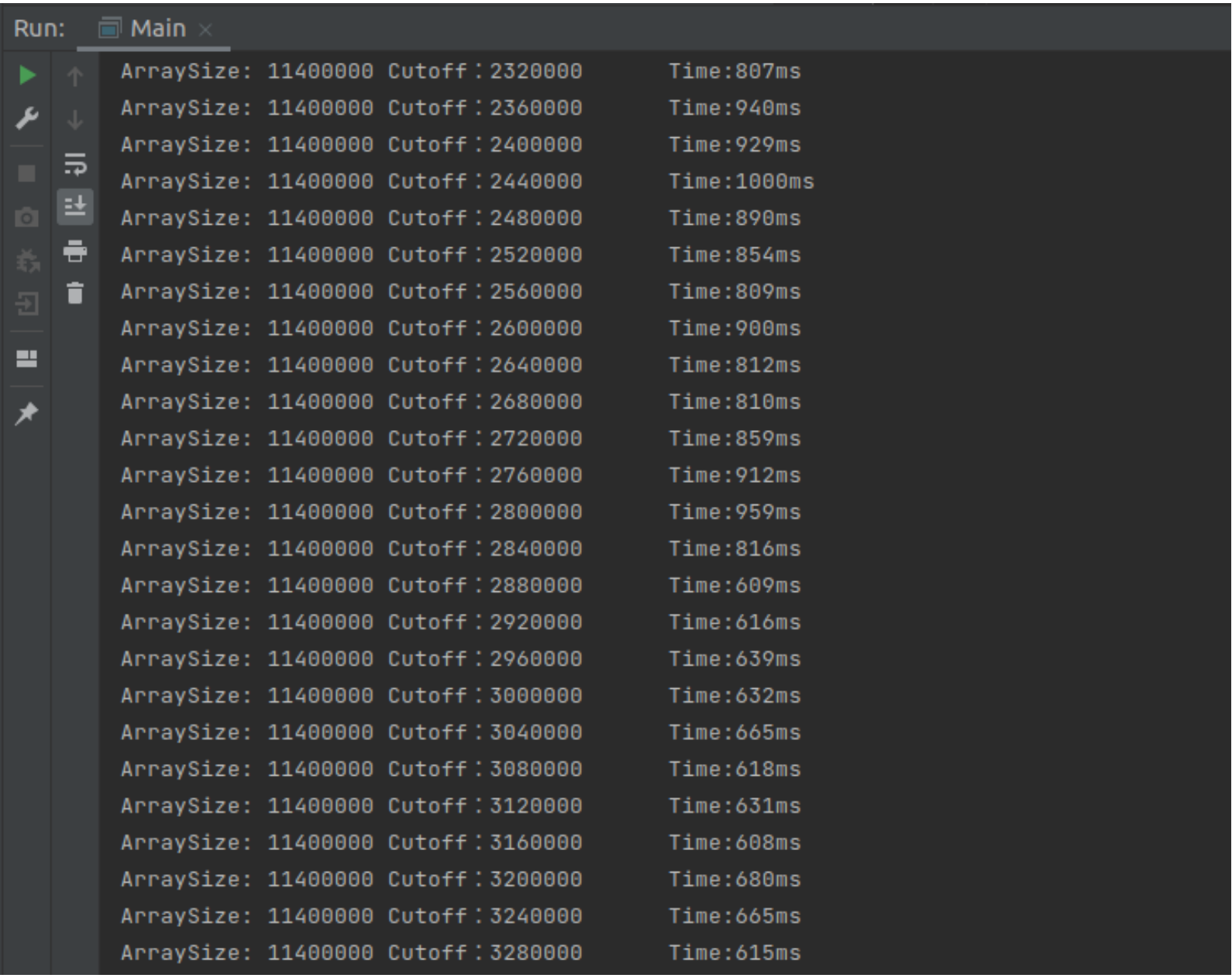
Tabulation:

A	B	C	D	E
Cutoff	Time (ms)			
1020000	1605			
1180000	1092			
1280000	1173			
1460000	1276			
5700000	998	Halfway mark		
6100000	929			
6800000	908			
7300000	925			
10200000	1217			
11000000	992			
11400000	974			
11600000	1696			
12800000	1711			

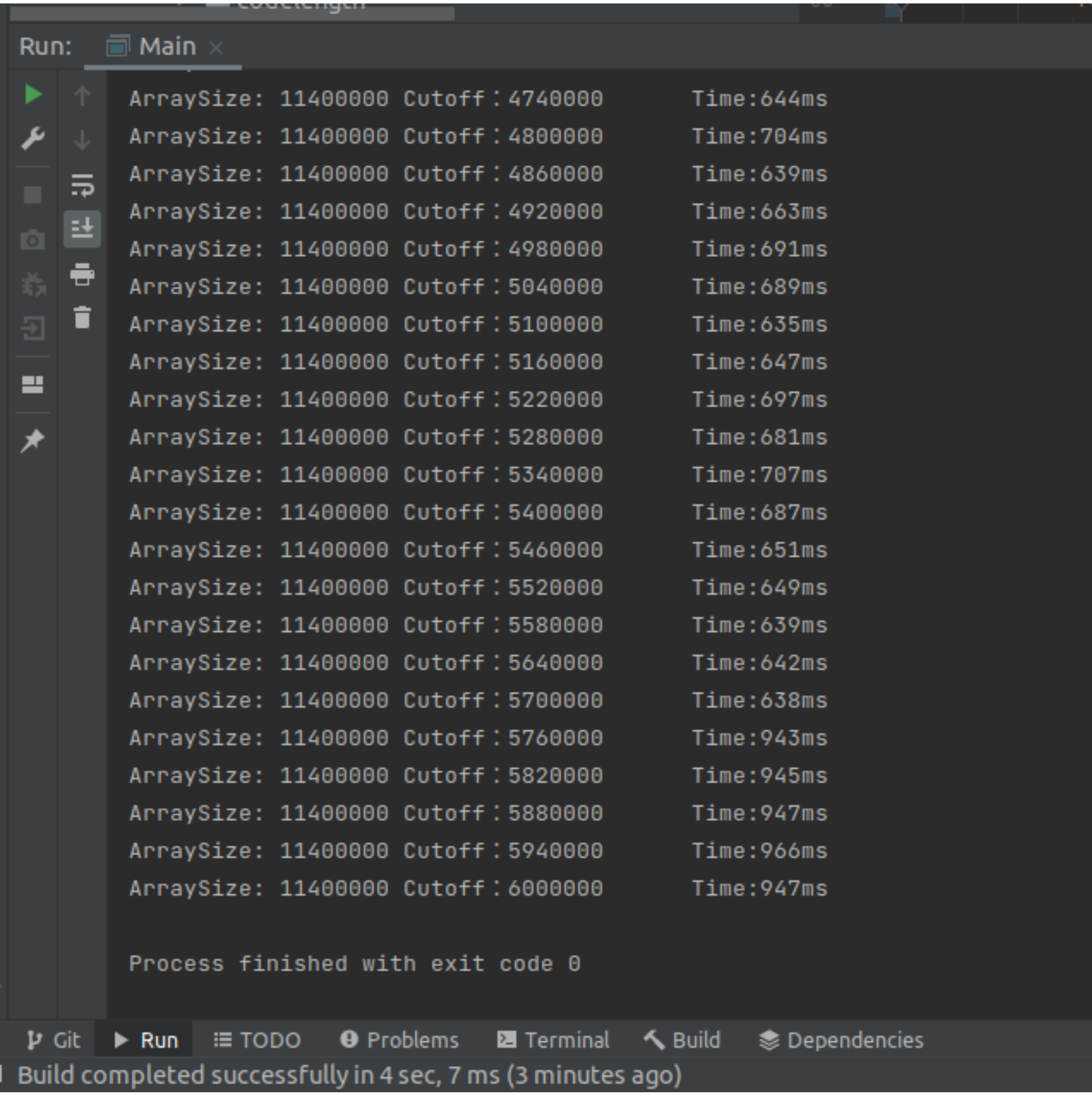


For number of threads = 8

a) Cutoff towards the start of length



b) Cutoff towards half of length

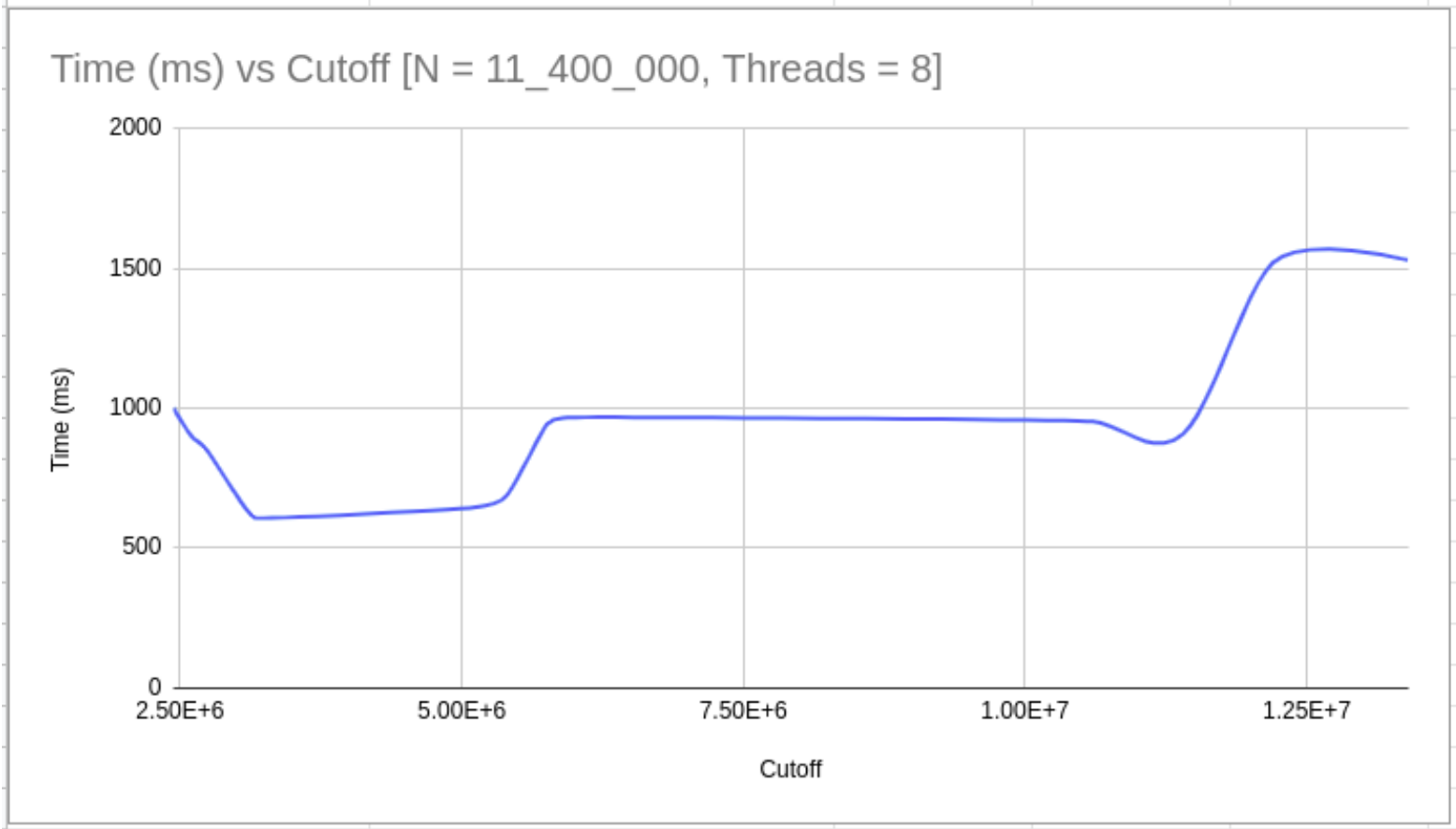


c) Cutoff towards end of length

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Thread count 8
ArraySize: 11400000 Cutoff：10200000          Time:1265ms
ArraySize: 11400000 Cutoff：10400000          Time:948ms
ArraySize: 11400000 Cutoff：10600000          Time:952ms
ArraySize: 11400000 Cutoff：10800000          Time:943ms
ArraySize: 11400000 Cutoff：11000000          Time:915ms
ArraySize: 11400000 Cutoff：11200000          Time:929ms
ArraySize: 11400000 Cutoff：11400000          Time:907ms
ArraySize: 11400000 Cutoff：11600000          Time:1514ms
ArraySize: 11400000 Cutoff：11800000          Time:1525ms
ArraySize: 11400000 Cutoff：12000000          Time:1518ms
ArraySize: 11400000 Cutoff：12200000          Time:1519ms
ArraySize: 11400000 Cutoff：12400000          Time:1512ms
ArraySize: 11400000 Cutoff：12600000          Time:1525ms
ArraySize: 11400000 Cutoff：12800000          Time:1524ms
ArraySize: 11400000 Cutoff：13000000          Time:1509ms
ArraySize: 11400000 Cutoff：13200000          Time:1519ms
ArraySize: 11400000 Cutoff：13400000          Time:1529ms
ArraySize: 11400000 Cutoff：13600000          Time:1521ms
ArraySize: 11400000 Cutoff：13800000          Time:1521ms
ArraySize: 11400000 Cutoff：14000000          Time:1521ms
ArraySize: 11400000 Cutoff：14200000          Time:1518ms
ArraySize: 11400000 Cutoff：14400000          Time:1518ms
ArraySize: 11400000 Cutoff：14600000          Time:1524ms
```

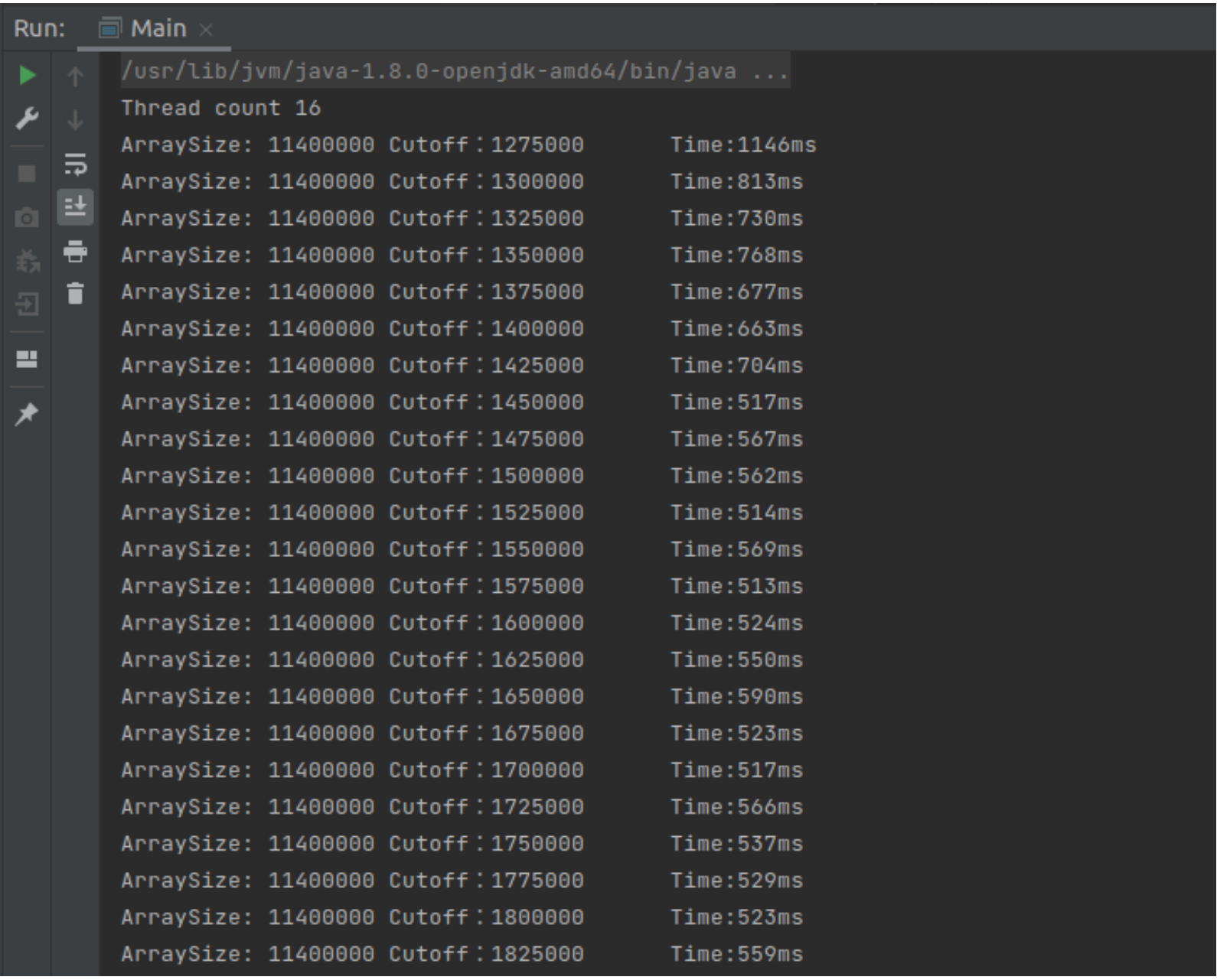
Tabulation:

A	B	C	D	E
Cutoff	Time (ms)			
2440000	1000			
2600000	900			
2720000	859			
3040000	665			
3160000	608			
5160000	647			
5400000	687	Halfway mark		
5760000	943			
5940000	966			
10600000	952			
11400000	907			
12200000	1519			
13400000	1529			

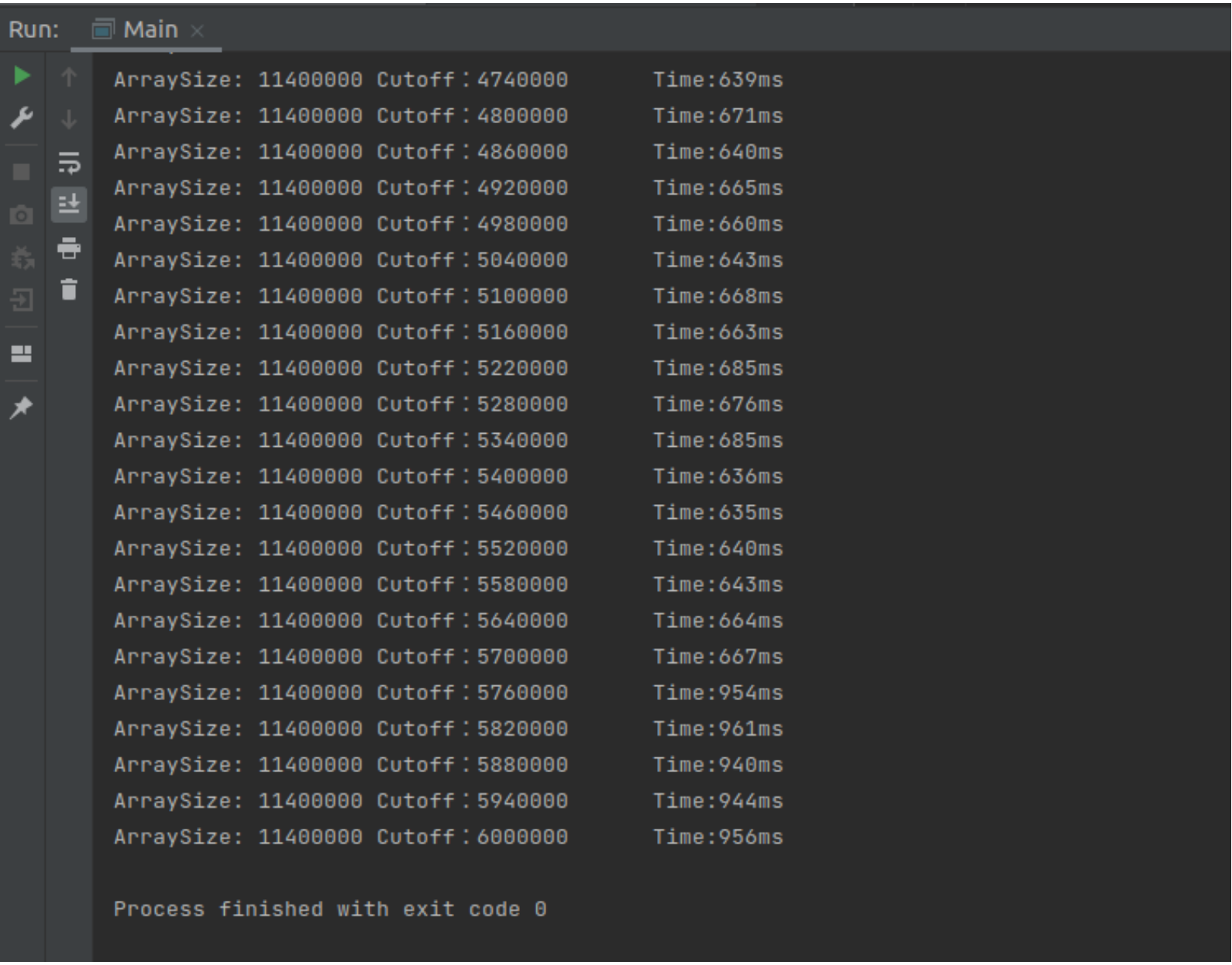


For number of threads = 16

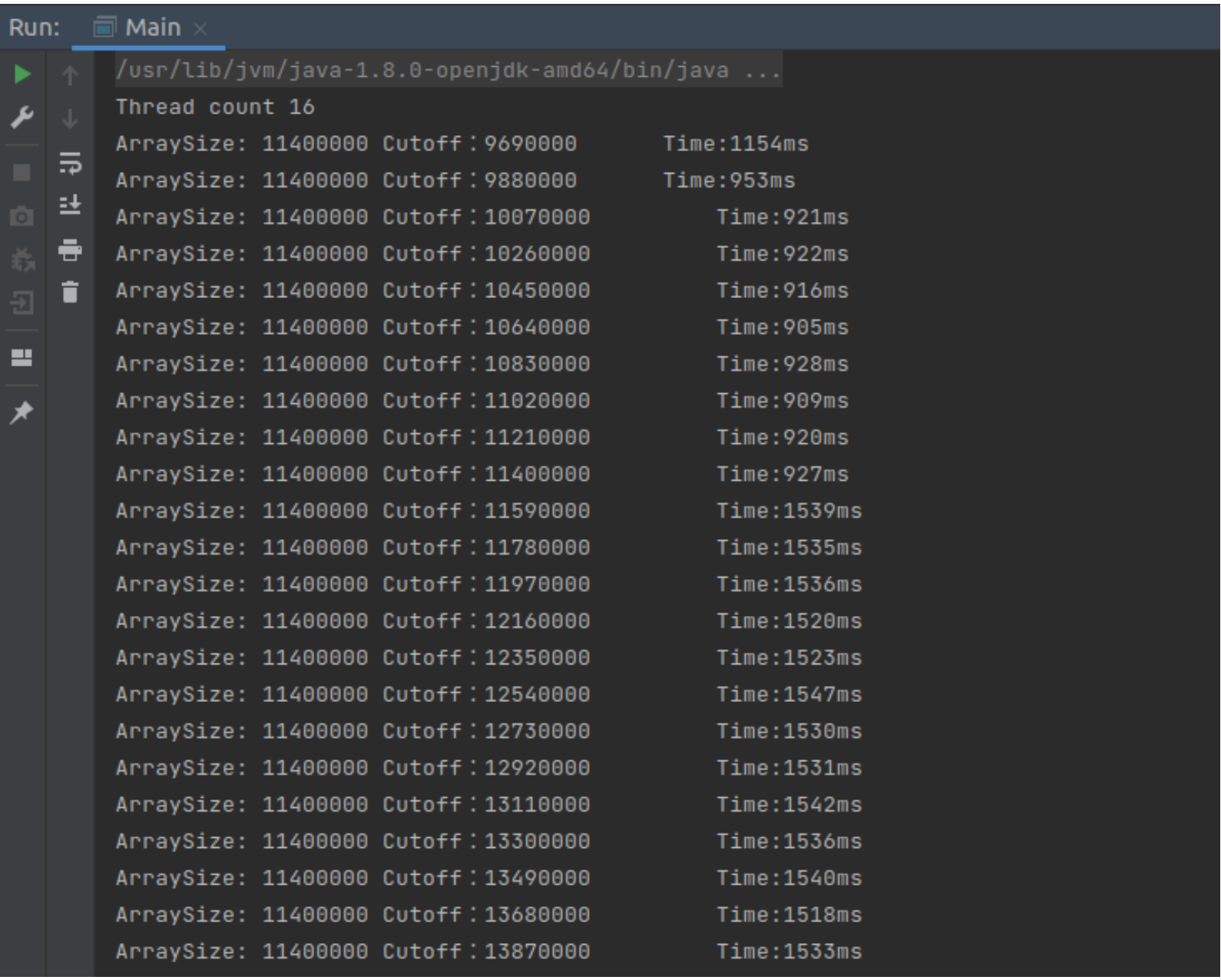
a) Cutoff near start of length



b) Cutoff towards half of length

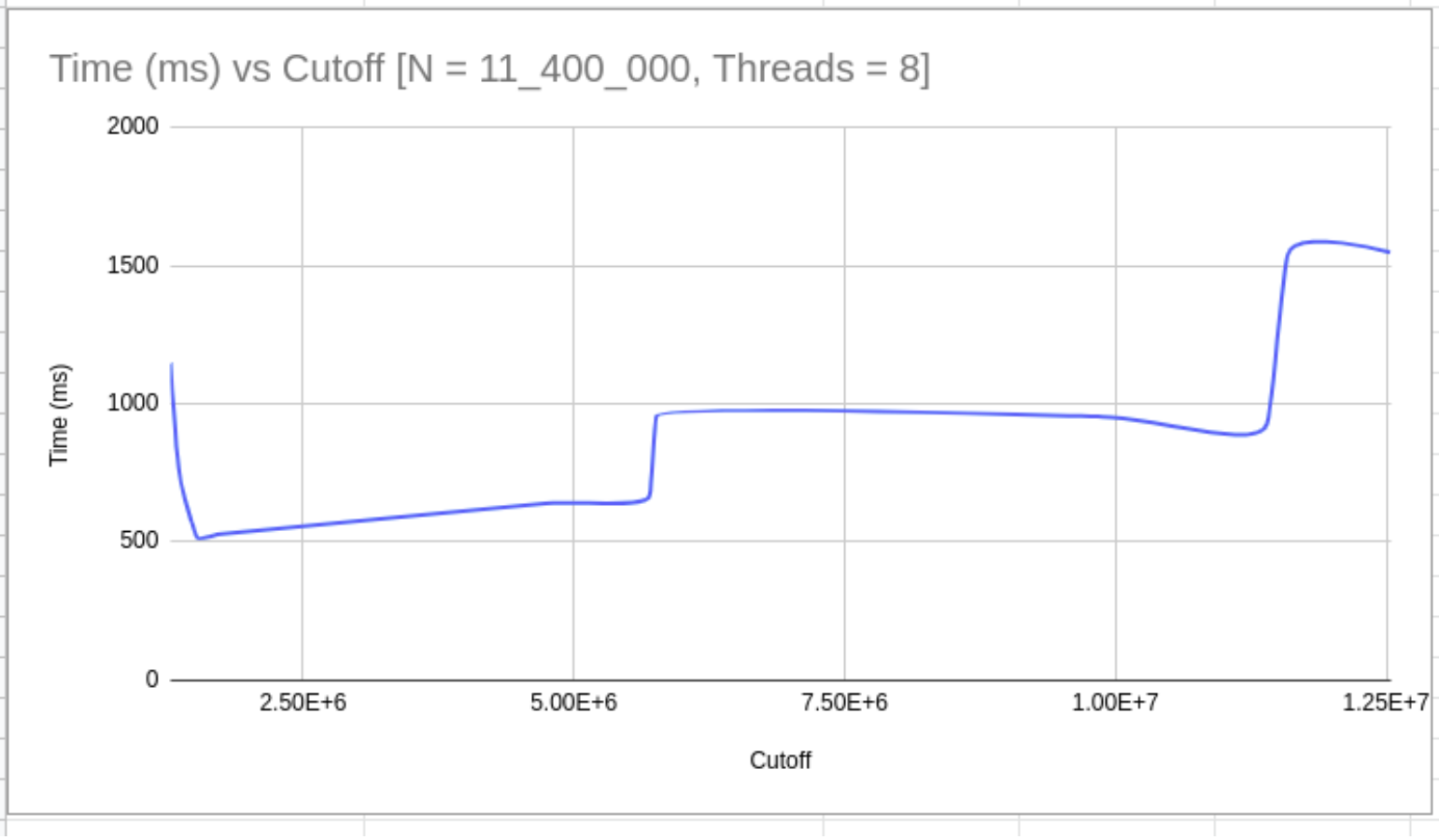


c) Cutoff towards end of length



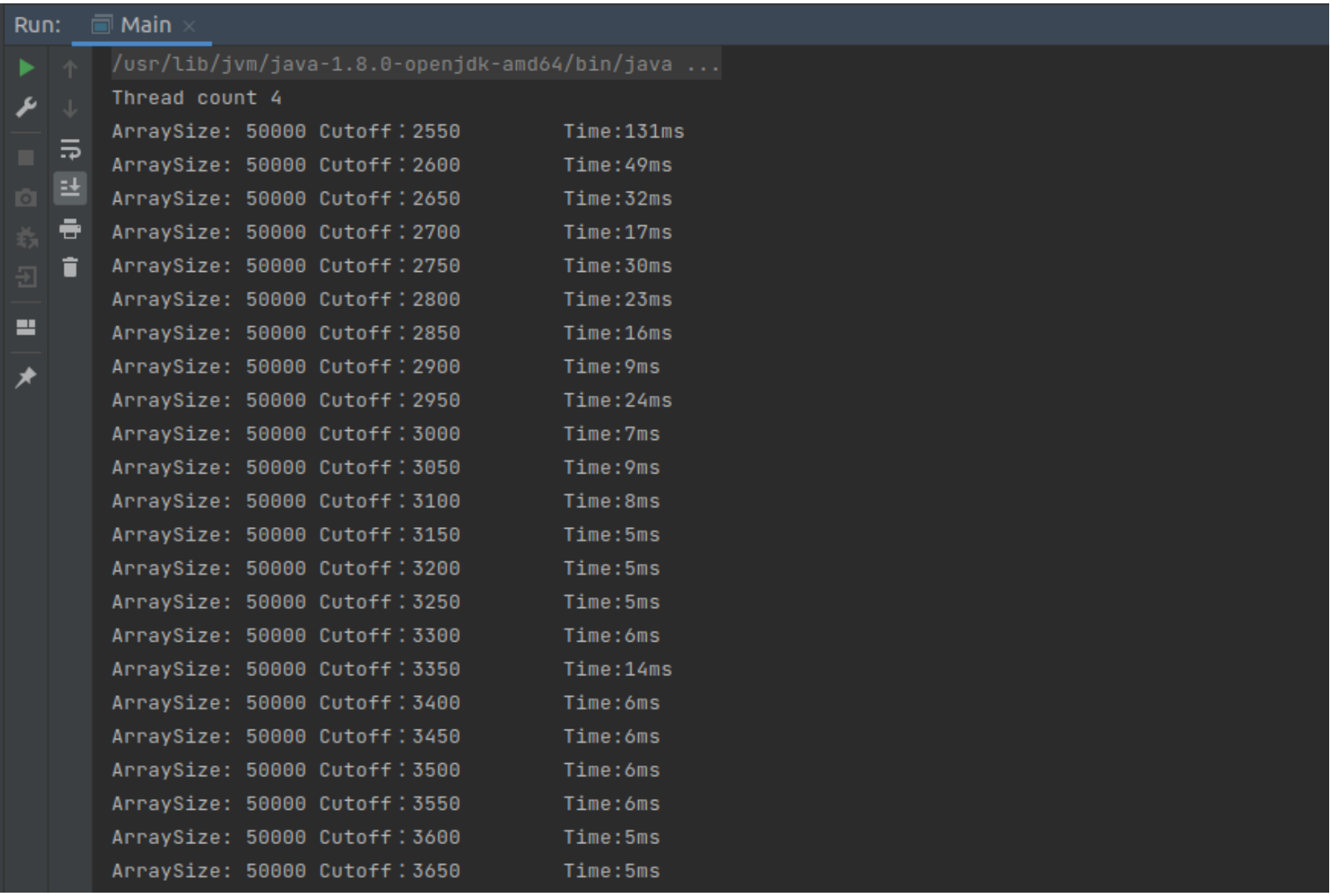
Tabulation:

A	B	C	D	E
Cutoff	Time (ms)			
1275000	1146			
1350000	768			
1475000	567			
1525000	514			
1750000	529			
4740000	639			
5700000	667	Halfway mark		
5760000	954			
9880000	953			
11400000	927			
11590000	1539			
12540000	1547			

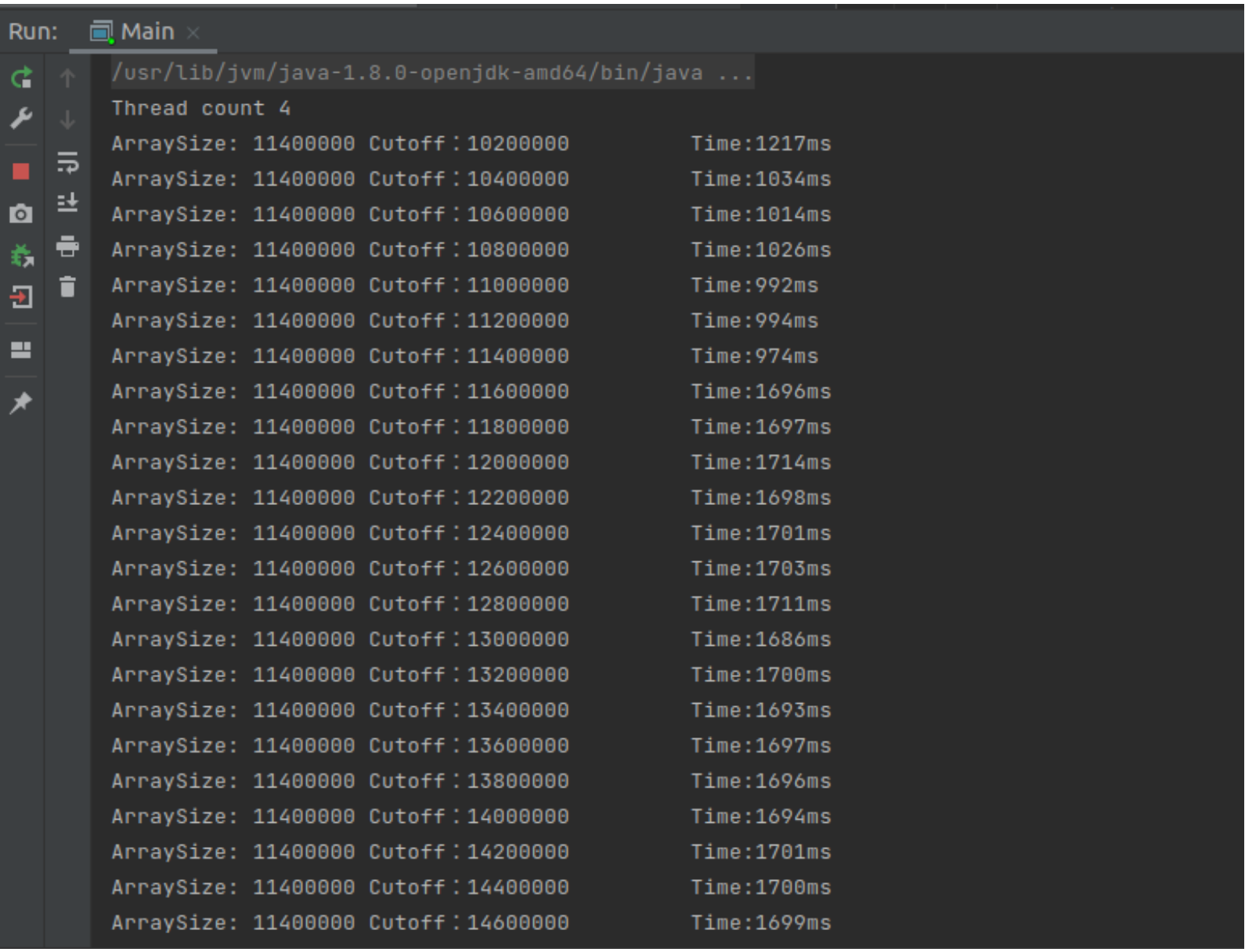


Conclusions:

1) Parallel sort is worthwhile only for larger sized arrays as opposed to smaller sized array which sort quicker using the system sort as seen below:



As seen above, post cutoff (System sort condition), the time taken to sort the array is quicker using the system sort



As seen above, post cutoff value (System sort condition) takes longer than the parallel sort technique for a larger array

2) Using number of threads as 2, 4 and above 16 (8 to 16 being the sweet spot) did not make the sorting more efficient and this could be as a result of threading thresholds and presence of overheads based on system to system but as seen on the screenshots, using number of threads = 2 is not the best optimization that is available.

3) Using thread numbers as either 8 or 16 or any int value in between has shown to have the best results.

4) (Array size used (N) = 11400000, Threads = 8) In the case of threads = 8, it is noticed that the time required to parallel sort is decreased and is at its best when the **cutoff is between ~(N/4) and ~(N/2)** and **after N/2 cutoff is reached, there is a spike in time required to sort and the sorting time starts to increase. Note: Sometimes the lower limit is below ~N/4 too.**

5) (Array size used (N) 11400000, Threads = 16) In the case of threads = 16, the pattern of times for parallel sort is similar to that of times obtained when Thread number = 8. However, the lower times start to appear in the cutoff range very slightly earlier. The **cutoff is between ~(N/4) and ~(N/2)** and **after the N/2 cutoff is reached, there is a spike in time required to sort and the sorting time starts to increase. Note: Sometimes the lower limit is below ~N/4 too.**

6) **The optimal cutoffs vary according to the variation of array sizes** and hence occurs only in context of a given array size but the general trend being observed is that they lie somewhere between **~N/4s to ~N/2** and **post N/2, the time required to parallel sort starts to increase**.

7) **The optimal number of threads also depends on the number of cores available** in the system processor but a general trend is between **8 to 16** beyond which the time taken to perform the sort operations is higher.