Program Structures & Algorithms

Spring 2022

Assignment No. 4

Parallel Sorting

Name: Aishwarya Surve

(NUID): 002123768

TASK

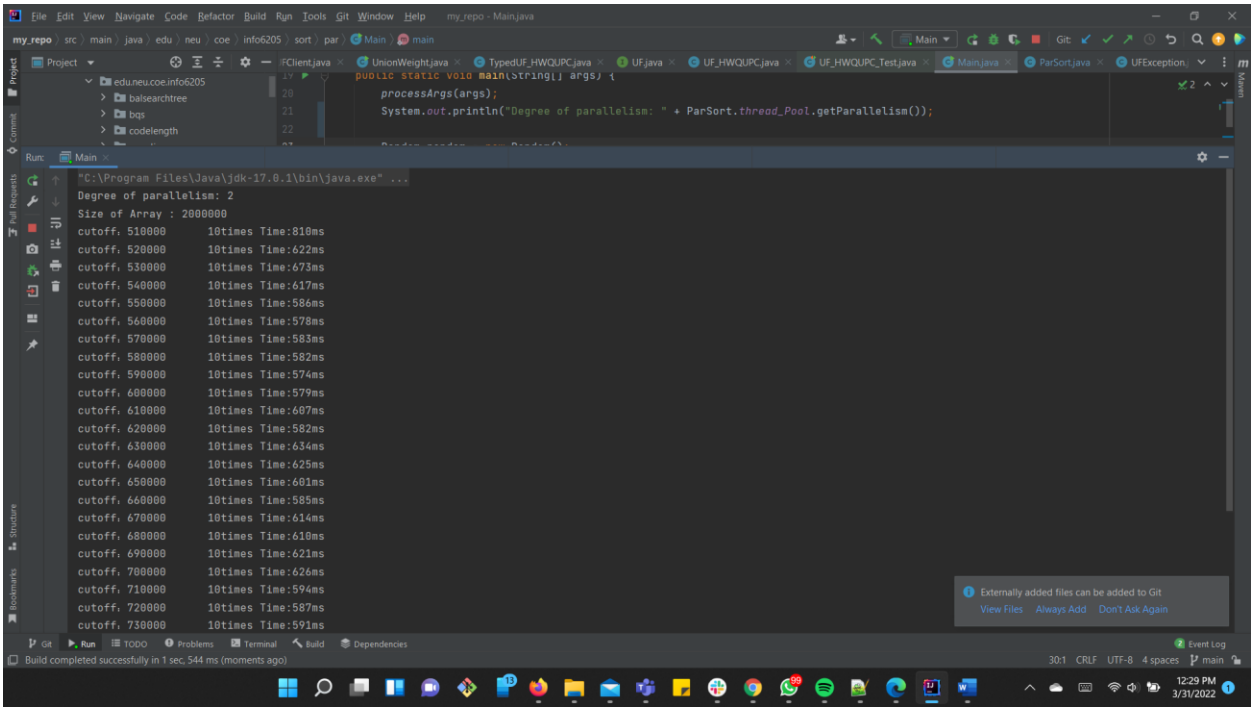
The task is to implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. You will consider two different schemes for deciding whether to sort in parallel.

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.

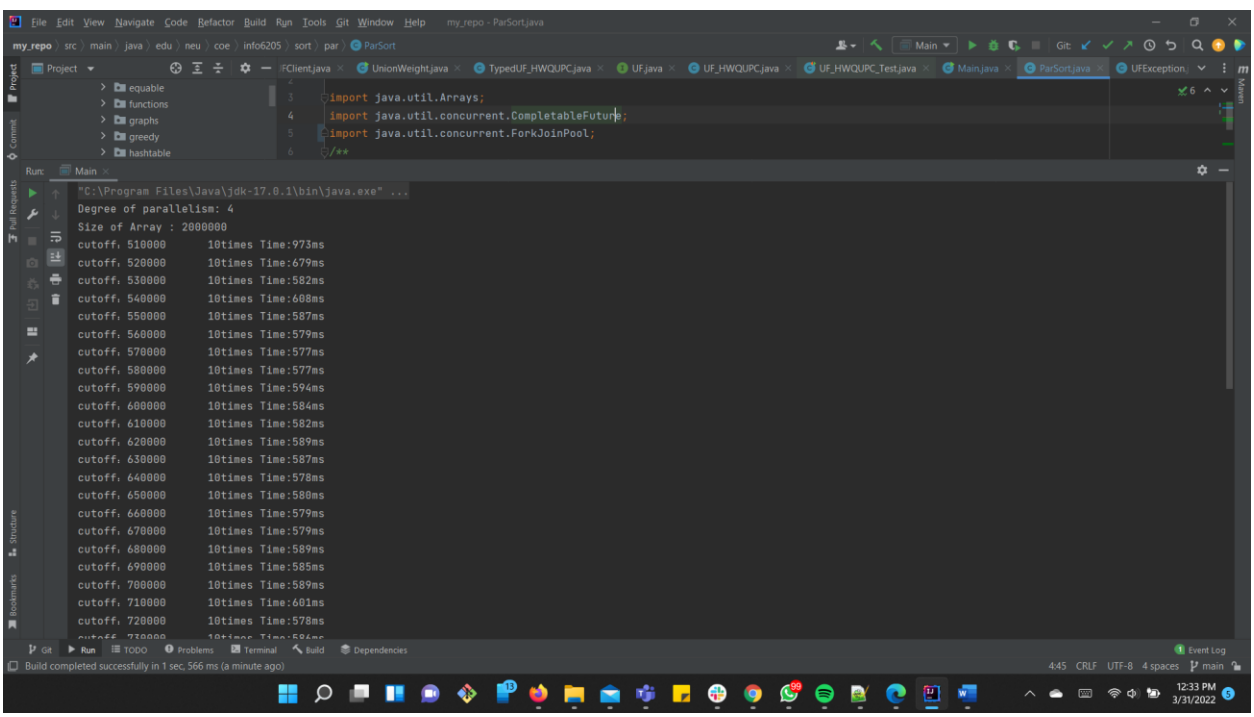
OUTPUT

I started by writing thread count (power of 2) and fixed array size code in main.java.

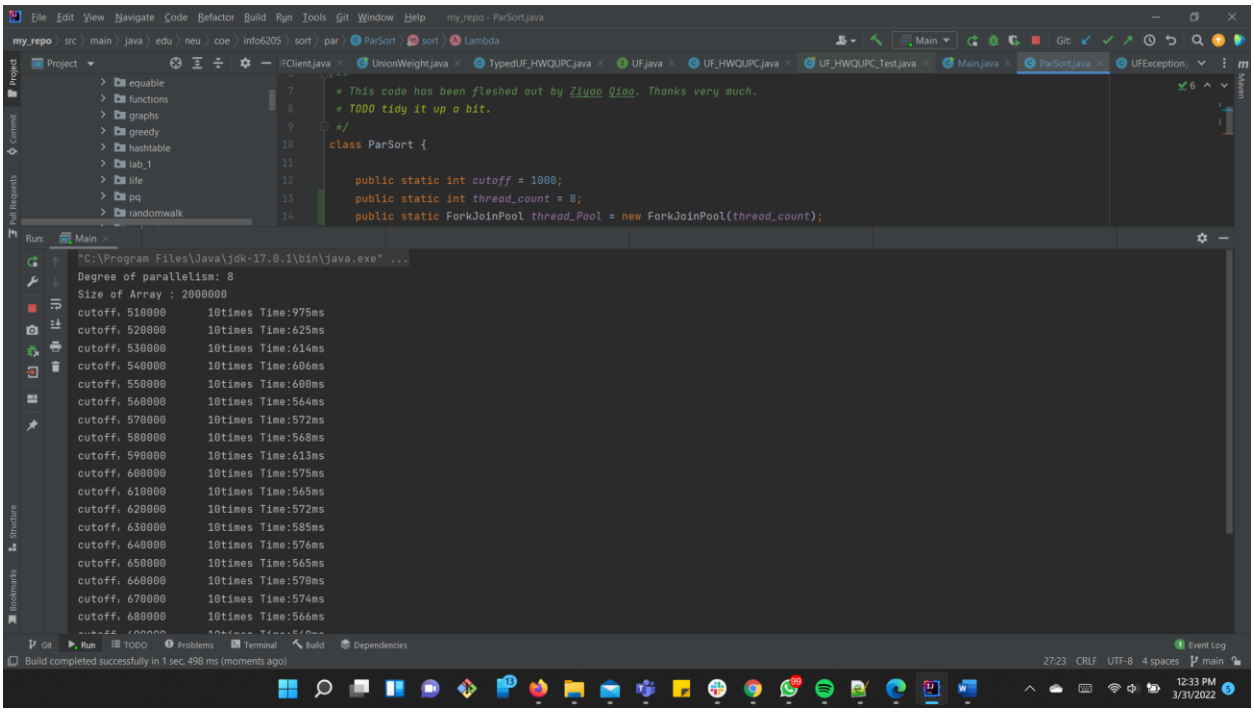
1. **Thread Count: 2**
   Array size: 2000000



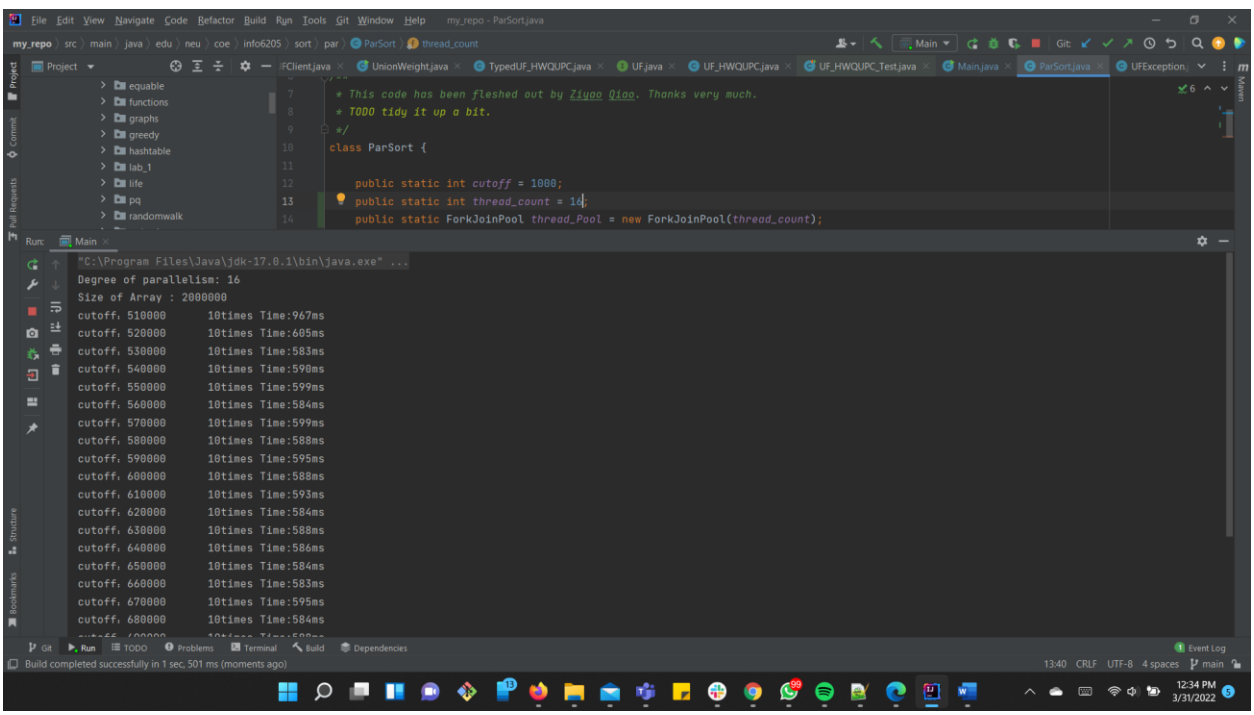
2. **Thread Count: 4**
   Array size: 2000000



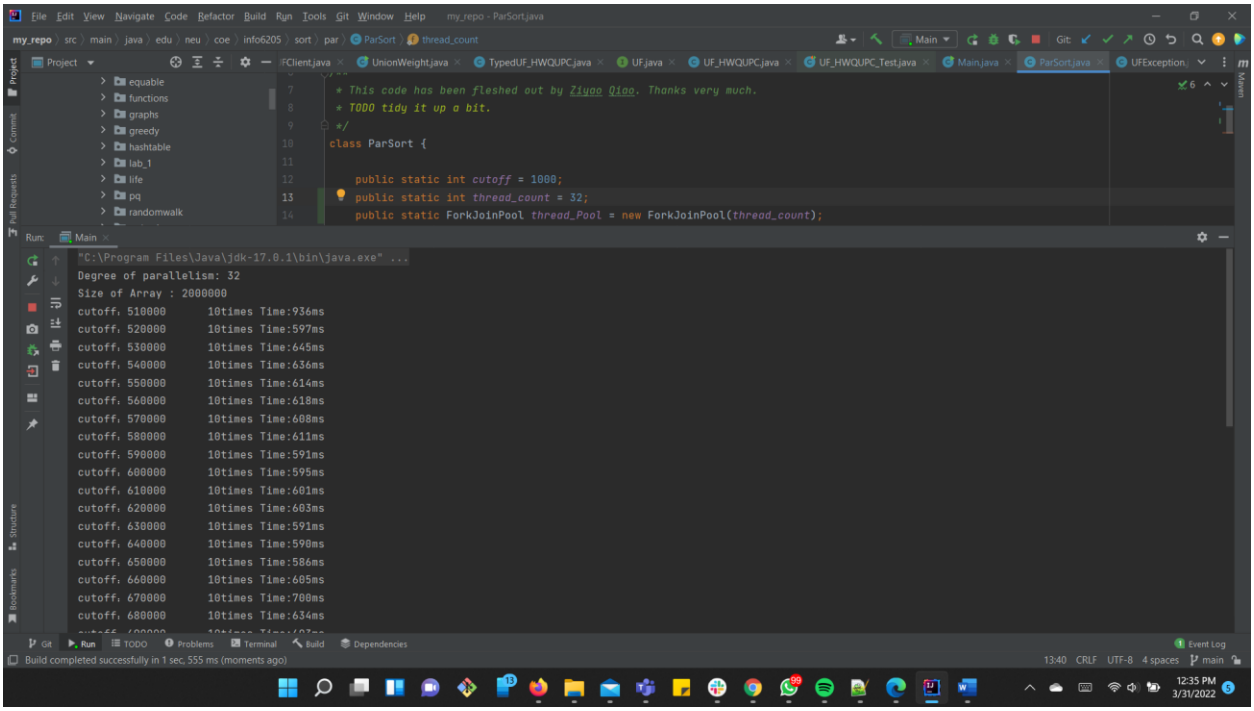
3. **Thread Count: 8**
   Array size: 2000000



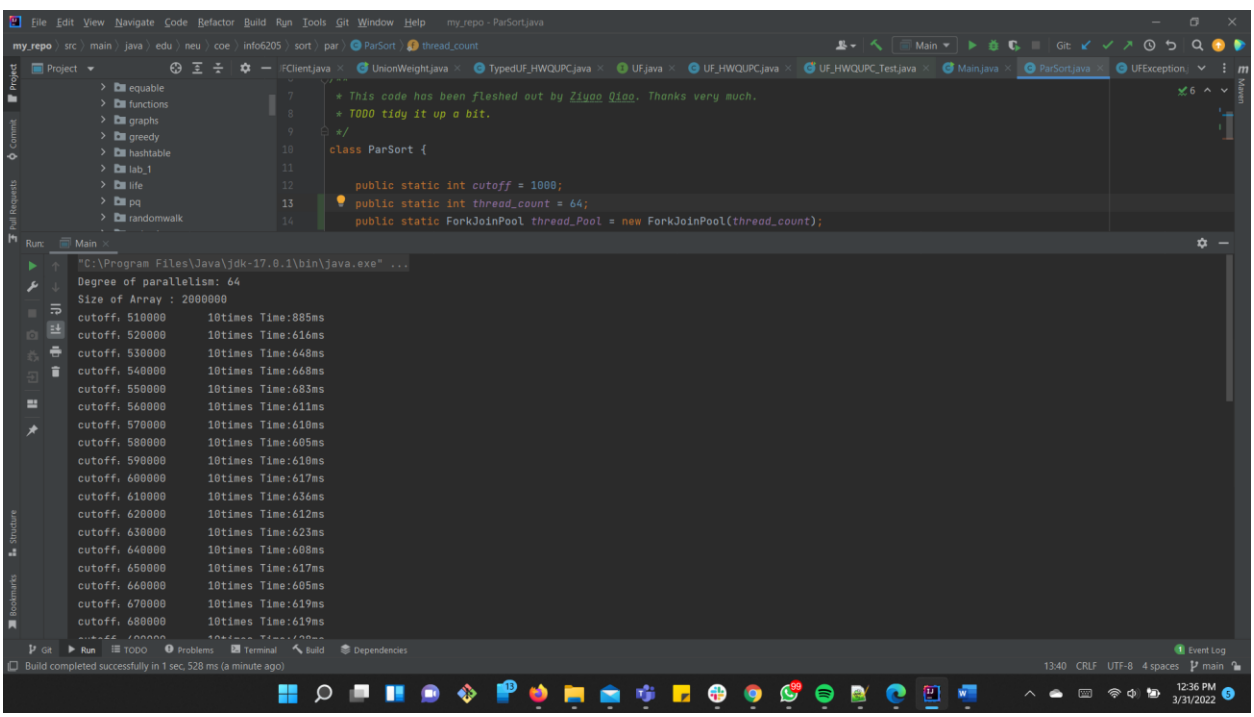
4. **Thread Count: 16**
   Array size: 2000000



5. **Thread Count: 32**
   Array size: 2000000



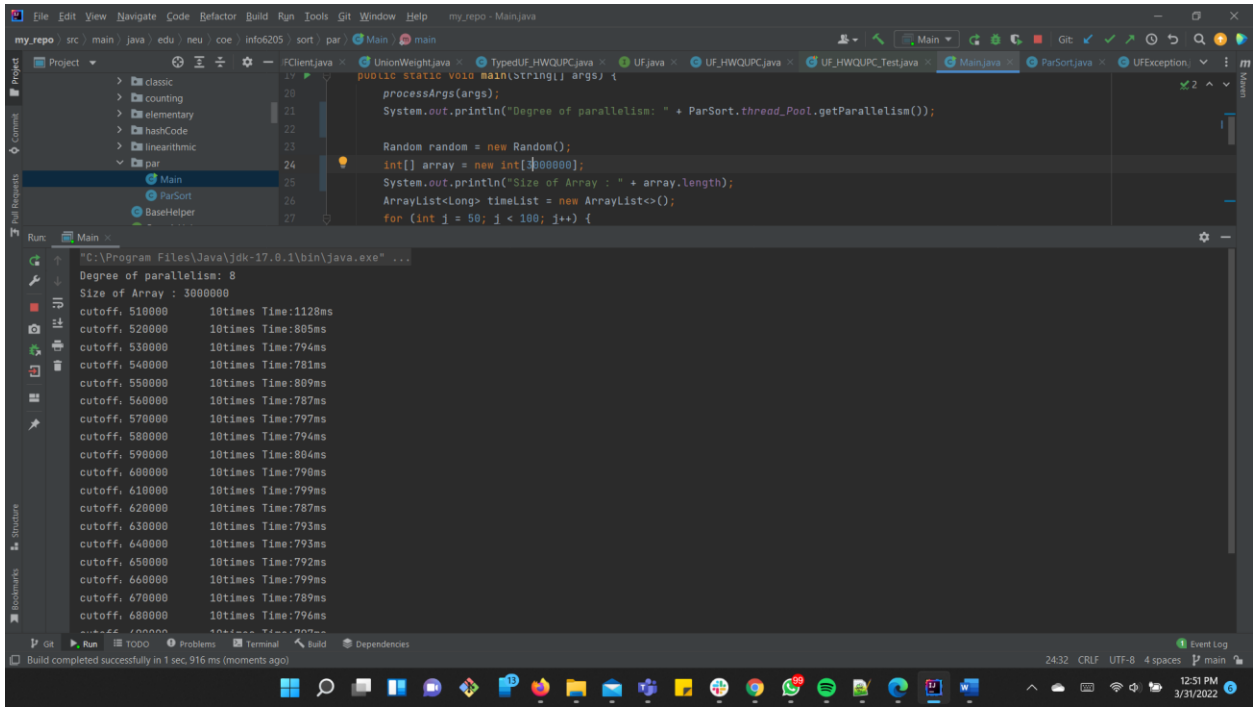
6. **Thread Count: 64**
   Array size: 2000000



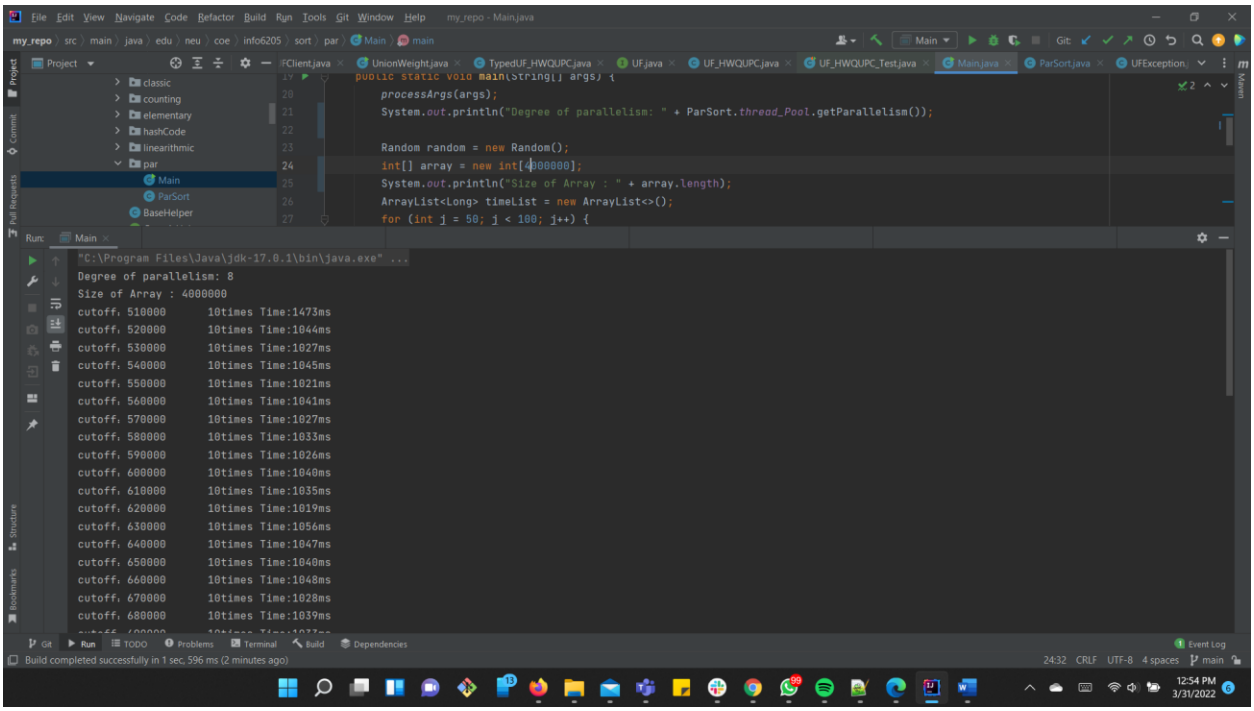
The above experiment was conducted using a cutoff range of 510000–990000, an array size of 2000000, and a thread count of 2-64. From the above tests, it is noticed that the optimal cut-off value is between 580000 - 650000. As a result, the thread count is 620000, and it is most efficient when the thread count is 8.

Now, let's validate this by modifying main.java with respect to array size and running an experiment with a thread count of 8 and a range of 510000-920000 for varied sizes of the array.

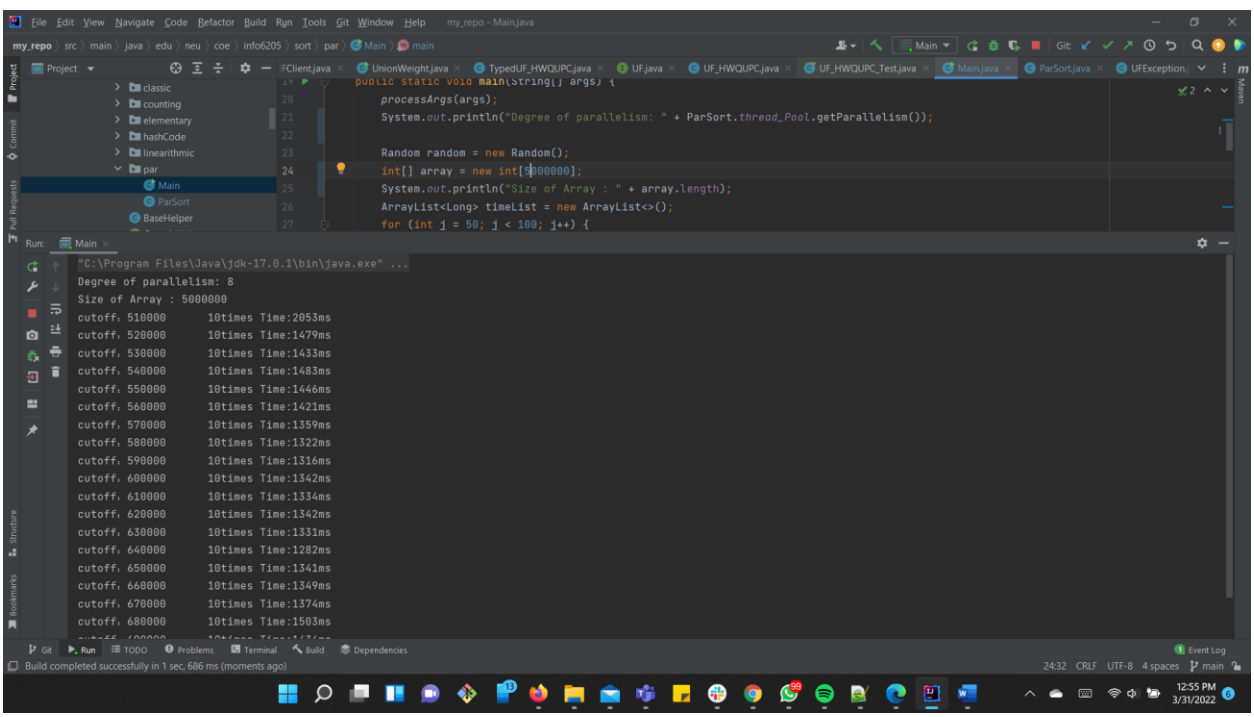
1. **Thread Count: 8**
   Array size: 3000000



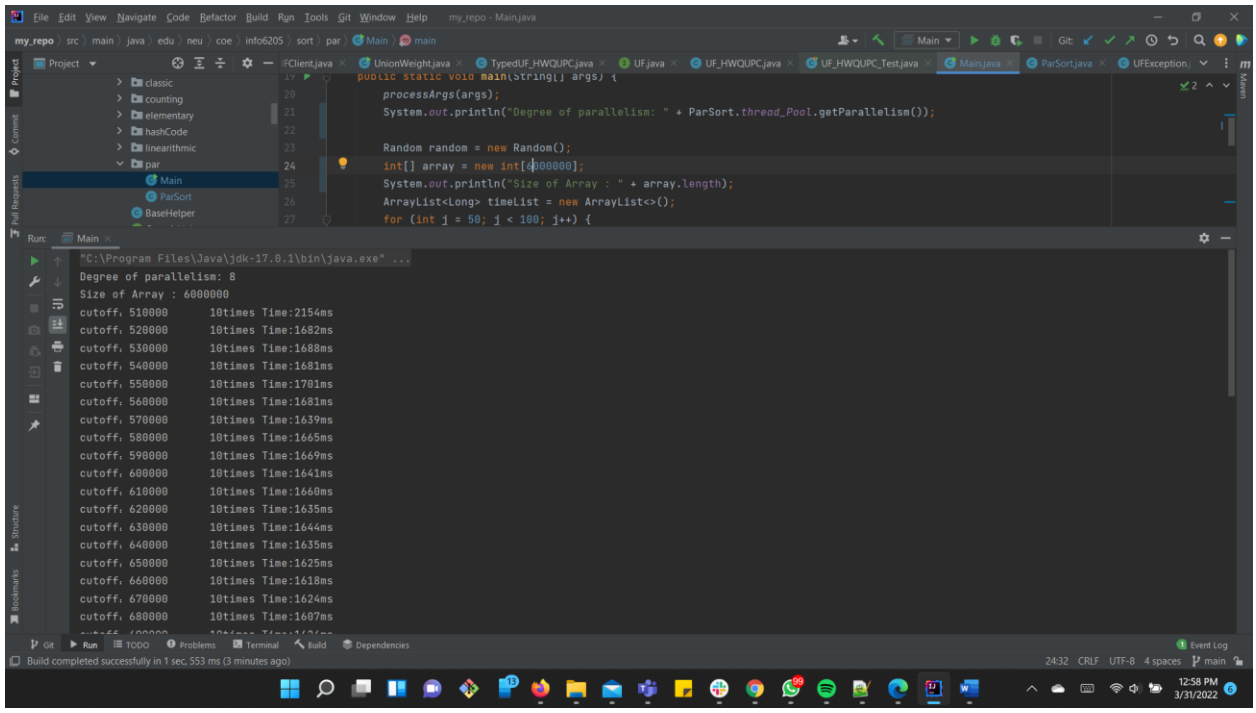
2. **Thread Count: 8**
   Array size: 4000000



3. Thread Count: 8
Array size: 5000000



4. **Thread Count: 8**
   Array size: 6000000



Conclusion:

According to the results of the above experiments, even though we raised the Array size from 2000000 to 6000000 by a multiplier of 1000000, it took longer, but when the cutoff value is 620000 and the thread count is 8, the method works most efficiently.

Cut-off time: 620000

Thread count: 8