

# 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.

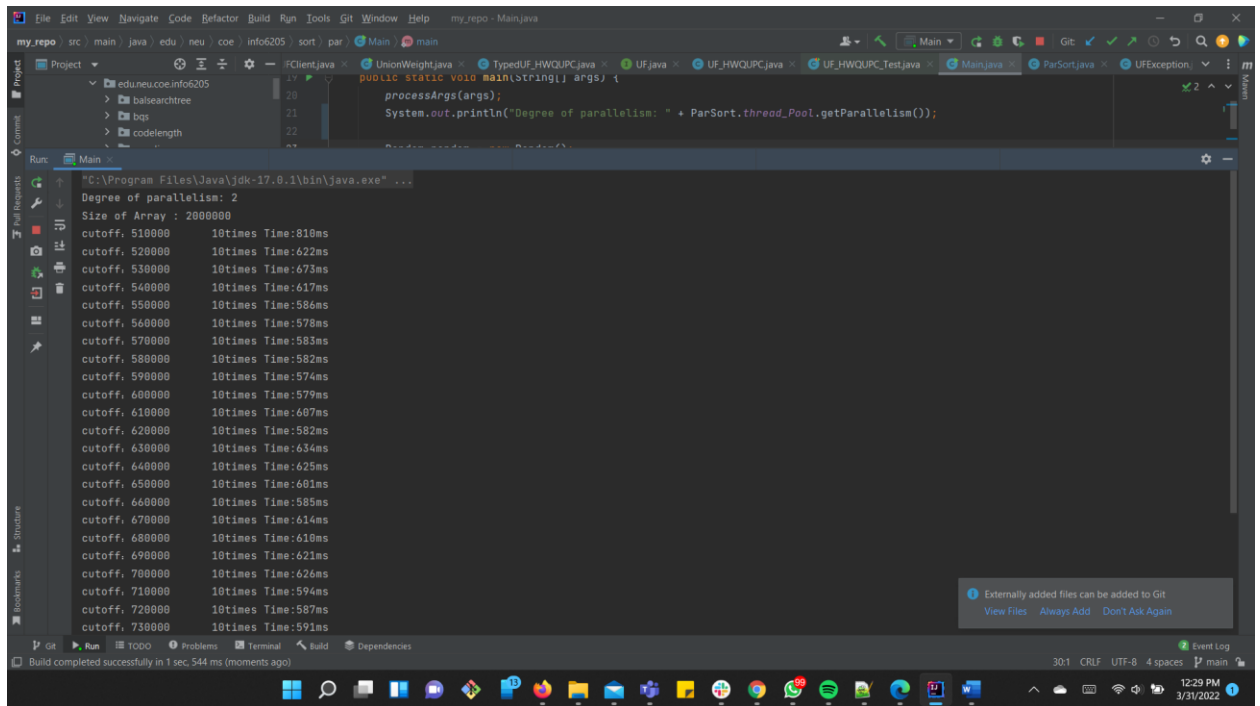
1. A cutoff (defaults to, say, 1000) which you will update according to the first argument in the command line when running. It's your job to experiment and come up with a good value for this cutoff. If there are fewer elements to sort than the cutoff, then you should use the system sort instead.
2. 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



```
public static void main(String[] args) {
    processArgs(args);
    System.out.println("Degree of parallelism: " + ParSort.thread_Pool.getParallelism());
}
```

Run: Main

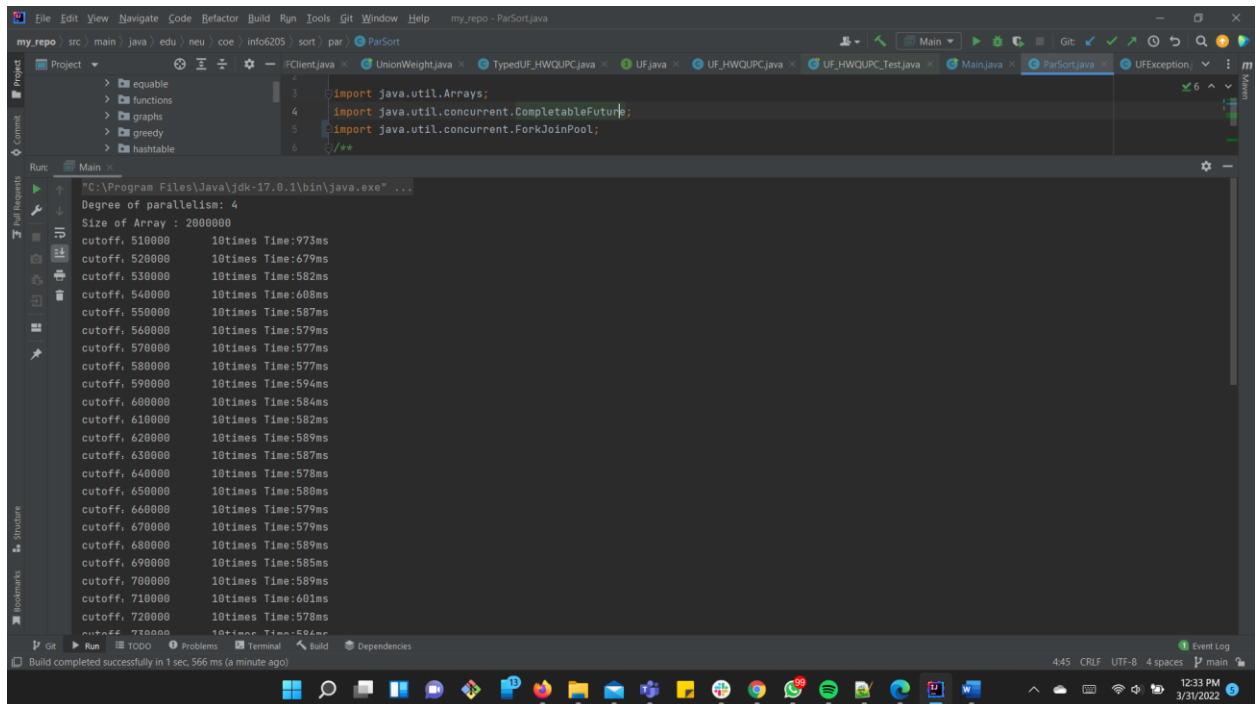
Build completed successfully in 1 sec: 544 ms (moments ago)

30:1 CRLF UTF-8 4 spaces P main

12:29 PM 3/31/2022

## 2. Thread Count: 4

Array size: 2000000



```
import java.util.Arrays;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ForkJoinPool;
/**
```

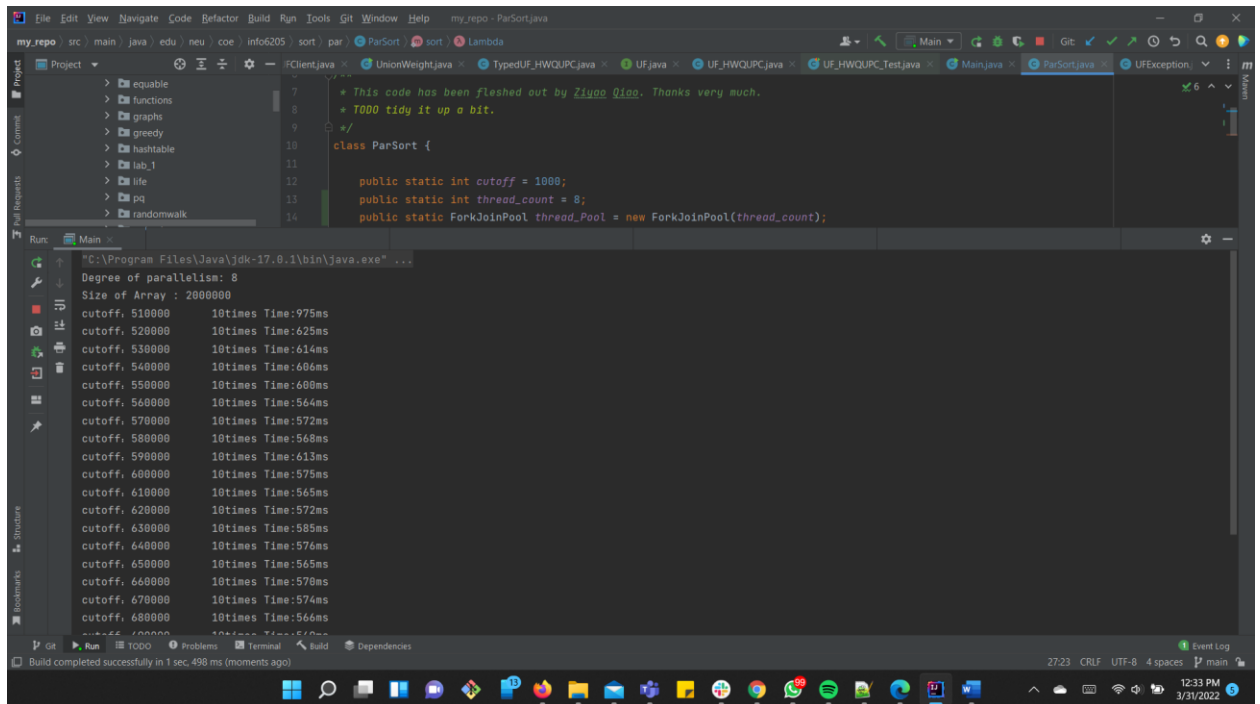
Run: Main

Build completed successfully in 1 sec: 566 ms (a minute ago)

4:45 CRLF UTF-8 4 spaces P main

12:33 PM 3/31/2022

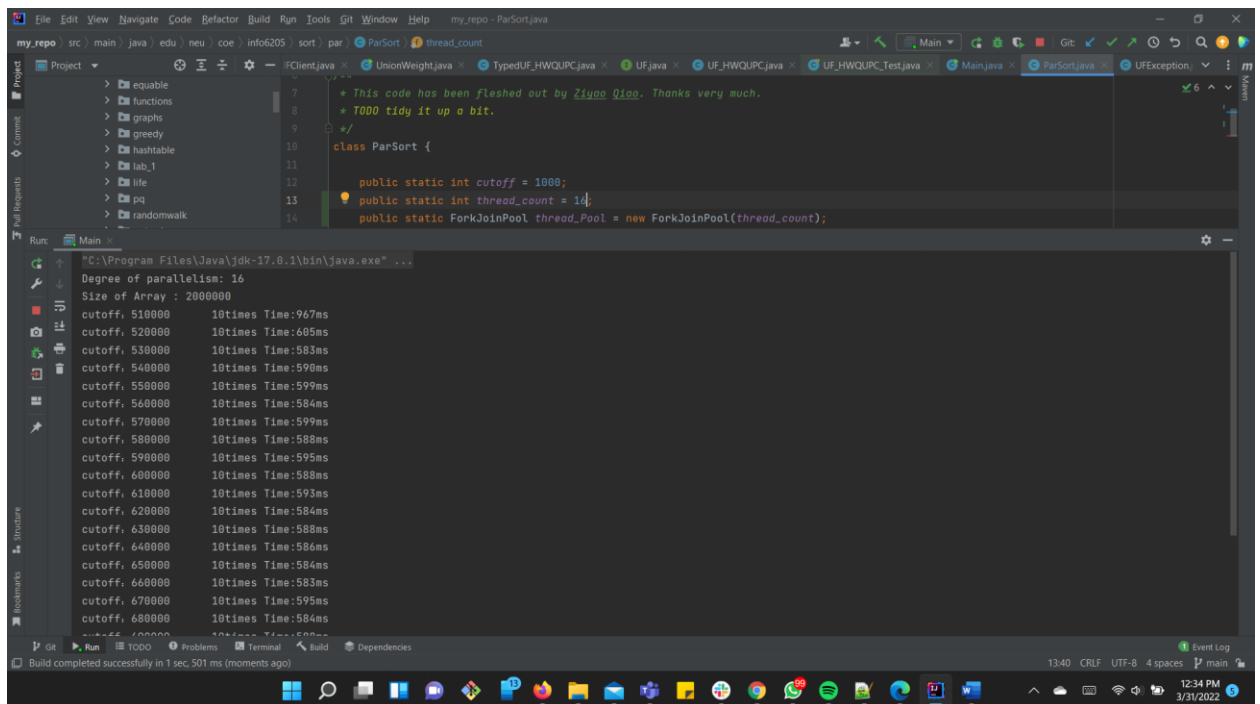
### 3. Thread Count: 8 Array size: 2000000



```
my_repo - ParSort.java
7 // This code has been fleshed out by Ziyao Qiao. Thanks very much.
8 // TODO tidy it up a bit.
9
10 class ParSort {
11
12     public static int cutoff = 1000;
13     public static int thread_count = 8;
14     public static ForkJoinPool thread_Pool = new ForkJoinPool(thread_count);
15 }

Run: Main
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
Degree of parallelism: 8
Size of Array : 2000000
cutoff, 510000    10times Time:975ms
cutoff, 520000    10times Time:625ms
cutoff, 530000    10times Time:614ms
cutoff, 540000    10times Time:606ms
cutoff, 550000    10times Time:600ms
cutoff, 560000    10times Time:564ms
cutoff, 570000    10times Time:572ms
cutoff, 580000    10times Time:568ms
cutoff, 590000    10times Time:613ms
cutoff, 600000    10times Time:575ms
cutoff, 610000    10times Time:565ms
cutoff, 620000    10times Time:572ms
cutoff, 630000    10times Time:585ms
cutoff, 640000    10times Time:576ms
cutoff, 650000    10times Time:565ms
cutoff, 660000    10times Time:570ms
cutoff, 670000    10times Time:574ms
cutoff, 680000    10times Time:566ms
cutoff, 690000    10times Time:566ms
Build completed successfully in 1 sec. 498 ms (moments ago)
```

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



```
my_repo - ParSort.java
7 // This code has been fleshed out by Ziyao Qiao. Thanks very much.
8 // TODO tidy it up a bit.
9
10 class ParSort {
11
12     public static int cutoff = 1000;
13     public static int thread_count = 16;
14     public static ForkJoinPool thread_Pool = new ForkJoinPool(thread_count);
15 }

Run: Main
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
Degree of parallelism: 16
Size of Array : 2000000
cutoff, 510000    10times Time:967ms
cutoff, 520000    10times Time:605ms
cutoff, 530000    10times Time:583ms
cutoff, 540000    10times Time:598ms
cutoff, 550000    10times Time:599ms
cutoff, 560000    10times Time:584ms
cutoff, 570000    10times Time:599ms
cutoff, 580000    10times Time:588ms
cutoff, 590000    10times Time:595ms
cutoff, 600000    10times Time:588ms
cutoff, 610000    10times Time:593ms
cutoff, 620000    10times Time:584ms
cutoff, 630000    10times Time:588ms
cutoff, 640000    10times Time:586ms
cutoff, 650000    10times Time:584ms
cutoff, 660000    10times Time:583ms
cutoff, 670000    10times Time:595ms
cutoff, 680000    10times Time:584ms
cutoff, 690000    10times Time:584ms
Build completed successfully in 1 sec. 501 ms (moments ago)
```

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

```
my_repo - ParSort.java
src \ main \ java \ edu \ neu \ coe \ info6205 \ sort \ par \ ParSort \ thread_count
Project
  > equible
  > functions
  > graphs
  > greedy
  > hashtable
  > lab_1
  > life
  > pq
  > randomwalk
  > ...
7 // This code has been fleshed out by Ziyao Qiao. Thanks very much.
8 // TODO tidy it up a bit.
9 //
10 class ParSort {
11
12     public static int cutoff = 1000;
13     public static int thread_count = 32;
14     public static ForkJoinPool thread_Pool = new ForkJoinPool(thread_count);
15
16 }
Run: Main
C:\Program Files\Java\jdk-17.0.1\bin\java.exe ...
Degree of parallelism: 32
Size of Array : 2000000
cutoff, 510000 10times Time:936ms
cutoff, 520000 10times Time:597ms
cutoff, 530000 10times Time:645ms
cutoff, 540000 10times Time:636ms
cutoff, 550000 10times Time:614ms
cutoff, 560000 10times Time:618ms
cutoff, 570000 10times Time:608ms
cutoff, 580000 10times Time:611ms
cutoff, 590000 10times Time:591ms
cutoff, 600000 10times Time:595ms
cutoff, 610000 10times Time:601ms
cutoff, 620000 10times Time:603ms
cutoff, 630000 10times Time:591ms
cutoff, 640000 10times Time:598ms
cutoff, 650000 10times Time:586ms
cutoff, 660000 10times Time:605ms
cutoff, 670000 10times Time:700ms
cutoff, 680000 10times Time:634ms
cutoff, 690000 10times Time:607ms
Build completed successfully in 1 sec.555 ms (moments ago)
13:40 CRLF UTF-8 4 spaces P main 12:35 PM 3/31/2022
```

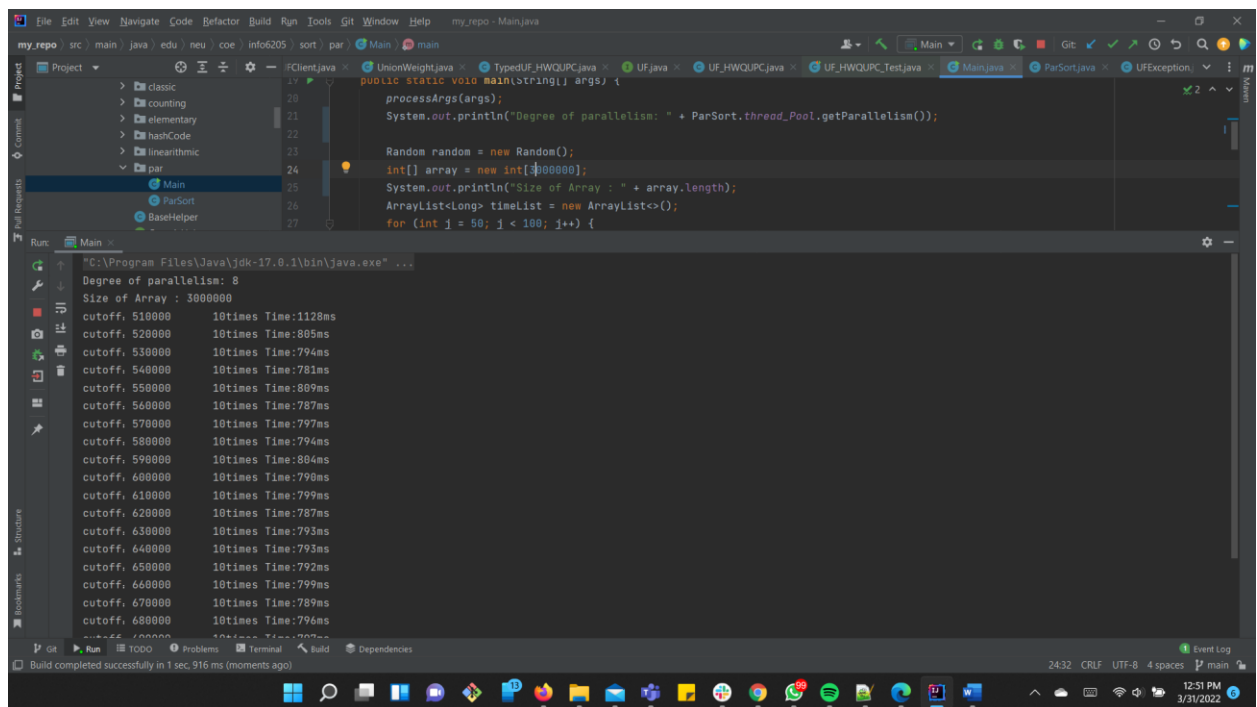
## 6. Thread Count: 64 Array size: 2000000

```
my_repo - ParSort.java
src \ main \ java \ edu \ neu \ coe \ info6205 \ sort \ par \ ParSort \ thread_count
Project
  > equible
  > functions
  > graphs
  > greedy
  > hashtable
  > lab_1
  > life
  > pq
  > randomwalk
  > ...
7 // This code has been fleshed out by Ziyao Qiao. Thanks very much.
8 // TODO tidy it up a bit.
9 //
10 class ParSort {
11
12     public static int cutoff = 1000;
13     public static int thread_count = 64;
14     public static ForkJoinPool thread_Pool = new ForkJoinPool(thread_count);
15
16 }
Run: Main
C:\Program Files\Java\jdk-17.0.1\bin\java.exe ...
Degree of parallelism: 64
Size of Array : 2000000
cutoff, 510000 10times Time:885ms
cutoff, 520000 10times Time:616ms
cutoff, 530000 10times Time:648ms
cutoff, 540000 10times Time:668ms
cutoff, 550000 10times Time:683ms
cutoff, 560000 10times Time:611ms
cutoff, 570000 10times Time:610ms
cutoff, 580000 10times Time:605ms
cutoff, 590000 10times Time:610ms
cutoff, 600000 10times Time:617ms
cutoff, 610000 10times Time:636ms
cutoff, 620000 10times Time:612ms
cutoff, 630000 10times Time:623ms
cutoff, 640000 10times Time:608ms
cutoff, 650000 10times Time:617ms
cutoff, 660000 10times Time:605ms
cutoff, 670000 10times Time:619ms
cutoff, 680000 10times Time:619ms
cutoff, 690000 10times Time:607ms
Build completed successfully in 1 sec.528 ms (a minute ago)
13:40 CRLF UTF-8 4 spaces P main 12:36 PM 3/31/2022
```

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



The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project structure with folders like 'classic', 'counting', 'elementary', 'hashCode', 'linearithmic', and 'par'. The 'Main' class is selected under the 'par' folder.
- Code Editor:** Displays the code for 'Main.java'. The code includes a 'main' method that prints the degree of parallelism, initializes a random array of size 3000000, and prints its size. It also contains a loop for testing different cutoff values.
- Run Console:** Shows the output of the program. It starts with 'Degree of parallelism: 8' and 'Size of Array : 3000000'. Below this, there is a list of cutoff values and their corresponding execution times, all showing a time of 10 times.
- Bottom Bar:** Indicates that the build completed successfully in 1 sec, 916 ms (moments ago).

```
public static void main(String[] args) {
    processArgs(args);
    System.out.println("Degree of parallelism: " + ParSort.thread_Pool.getParallelism());

    Random random = new Random();
    int[] array = new int[3000000];
    System.out.println("Size of Array : " + array.length);
    ArrayList<Long> timeList = new ArrayList<>();
    for (int j = 50; j < 100; j++) {
```

Run Console Output:

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
Degree of parallelism: 8
Size of Array : 3000000
cutoff, 510000    10times Time:1128ms
cutoff, 520000    10times Time:805ms
cutoff, 530000    10times Time:794ms
cutoff, 540000    10times Time:781ms
cutoff, 550000    10times Time:809ms
cutoff, 560000    10times Time:787ms
cutoff, 570000    10times Time:797ms
cutoff, 580000    10times Time:794ms
cutoff, 590000    10times Time:884ms
cutoff, 600000    10times Time:798ms
cutoff, 610000    10times Time:799ms
cutoff, 620000    10times Time:787ms
cutoff, 630000    10times Time:793ms
cutoff, 640000    10times Time:793ms
cutoff, 650000    10times Time:792ms
cutoff, 660000    10times Time:799ms
cutoff, 670000    10times Time:789ms
cutoff, 680000    10times Time:796ms
cutoff, 690000    10times Time:797ms
```

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

The screenshot shows an IDE window with a Java project named 'my\_repo'. The 'Main.java' file is open, displaying the following code:

```
public static void main(String[] args) {  
    processArgs(args);  
    System.out.println("Degree of parallelism: " + ParSort.thread_Pool.getParallelism());  
  
    Random random = new Random();  
    int[] array = new int[4000000];  
    System.out.println("Size of Array : " + array.length);  
    ArrayList<Long> timeList = new ArrayList<>();  
    for (int j = 50; j < 100; j++) {
```

The 'Run' console shows the following output:

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...  
Degree of parallelism: 8  
Size of Array : 4000000  
cutoff, 510000    10times Time:1473ms  
cutoff, 520000    10times Time:1044ms  
cutoff, 530000    10times Time:1027ms  
cutoff, 540000    10times Time:1045ms  
cutoff, 550000    10times Time:1021ms  
cutoff, 560000    10times Time:1041ms  
cutoff, 570000    10times Time:1027ms  
cutoff, 580000    10times Time:1033ms  
cutoff, 590000    10times Time:1026ms  
cutoff, 600000    10times Time:1040ms  
cutoff, 610000    10times Time:1035ms  
cutoff, 620000    10times Time:1019ms  
cutoff, 630000    10times Time:1056ms  
cutoff, 640000    10times Time:1047ms  
cutoff, 650000    10times Time:1040ms  
cutoff, 660000    10times Time:1048ms  
cutoff, 670000    10times Time:1028ms  
cutoff, 680000    10times Time:1039ms
```

### 3. Thread Count: 8 Array size: 5000000

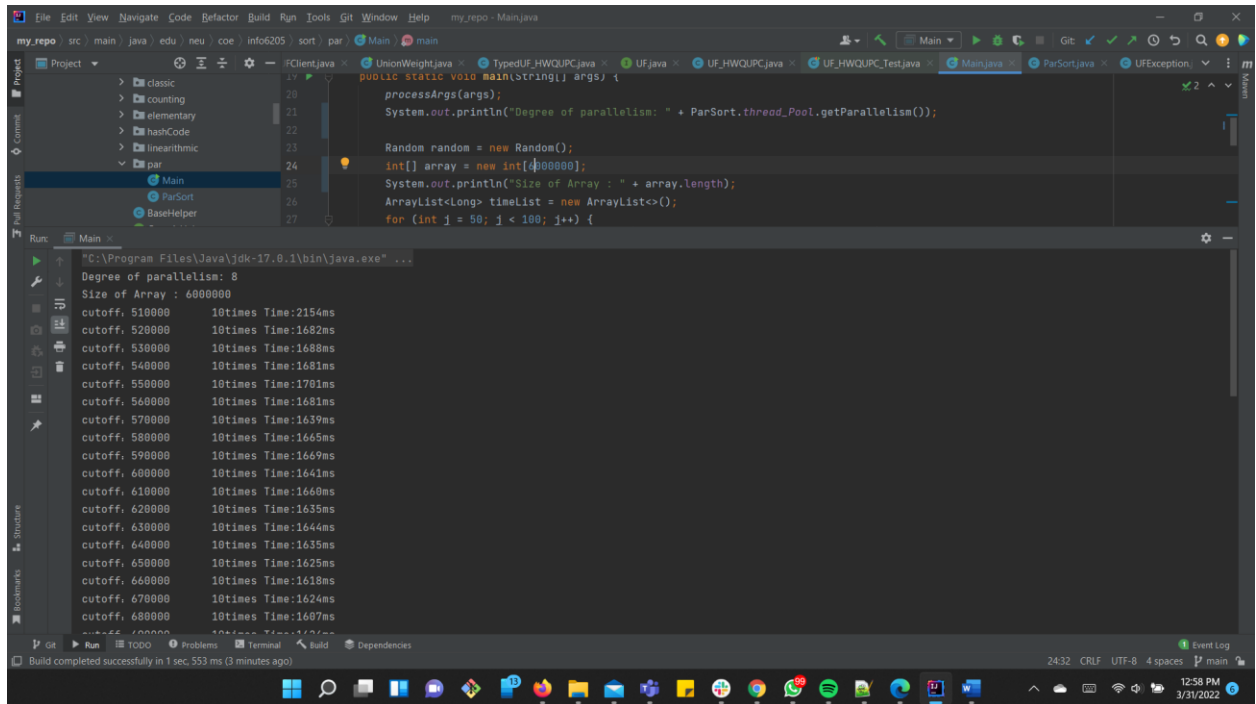
The screenshot shows the same IDE window with the 'Main.java' file open, displaying the following code:

```
public static void main(String[] args) {  
    processArgs(args);  
    System.out.println("Degree of parallelism: " + ParSort.thread_Pool.getParallelism());  
  
    Random random = new Random();  
    int[] array = new int[5000000];  
    System.out.println("Size of Array : " + array.length);  
    ArrayList<Long> timeList = new ArrayList<>();  
    for (int j = 50; j < 100; j++) {
```

The 'Run' console shows the following output:

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...  
Degree of parallelism: 8  
Size of Array : 5000000  
cutoff, 510000    10times Time:2053ms  
cutoff, 520000    10times Time:1479ms  
cutoff, 530000    10times Time:1433ms  
cutoff, 540000    10times Time:1483ms  
cutoff, 550000    10times Time:1446ms  
cutoff, 560000    10times Time:1421ms  
cutoff, 570000    10times Time:1359ms  
cutoff, 580000    10times Time:1322ms  
cutoff, 590000    10times Time:1316ms  
cutoff, 600000    10times Time:1342ms  
cutoff, 610000    10times Time:1334ms  
cutoff, 620000    10times Time:1342ms  
cutoff, 630000    10times Time:1331ms  
cutoff, 640000    10times Time:1282ms  
cutoff, 650000    10times Time:1341ms  
cutoff, 660000    10times Time:1349ms  
cutoff, 670000    10times Time:1374ms  
cutoff, 680000    10times Time:1503ms
```

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



The screenshot shows an IDE with a project named 'my\_repo'. The 'Main.java' file is open, showing the following code:

```
public static void main(String[] args) {  
    processArgs(args);  
    System.out.println("Degree of parallelism: " + ParSort.thread_Pool.getParallelism());  
  
    Random random = new Random();  
    int[] array = new int[6000000];  
    System.out.println("Size of Array : " + array.length);  
    ArrayList<Long> timeList = new ArrayList<>();  
    for (int j = 50; j < 100; j++) {
```

The 'Run' console shows the output of the program:

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...  
Degree of parallelism: 8  
Size of Array : 6000000  
cutoff, 510000    10times Time:2154ms  
cutoff, 520000    10times Time:1682ms  
cutoff, 530000    10times Time:1688ms  
cutoff, 540000    10times Time:1681ms  
cutoff, 550000    10times Time:1701ms  
cutoff, 560000    10times Time:1681ms  
cutoff, 570000    10times Time:1639ms  
cutoff, 580000    10times Time:1665ms  
cutoff, 590000    10times Time:1669ms  
cutoff, 600000    10times Time:1641ms  
cutoff, 610000    10times Time:1660ms  
cutoff, 620000    10times Time:1635ms  
cutoff, 630000    10times Time:1644ms  
cutoff, 640000    10times Time:1635ms  
cutoff, 650000    10times Time:1625ms  
cutoff, 660000    10times Time:1618ms  
cutoff, 670000    10times Time:1624ms  
cutoff, 680000    10times Time:1607ms
```

The status bar at the bottom indicates that the build completed successfully in 1 sec, 553 ms (3 minutes ago).

## 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