Program Structure and Algorithms

Spring 2022

Assignment 4

Himanshu Walia

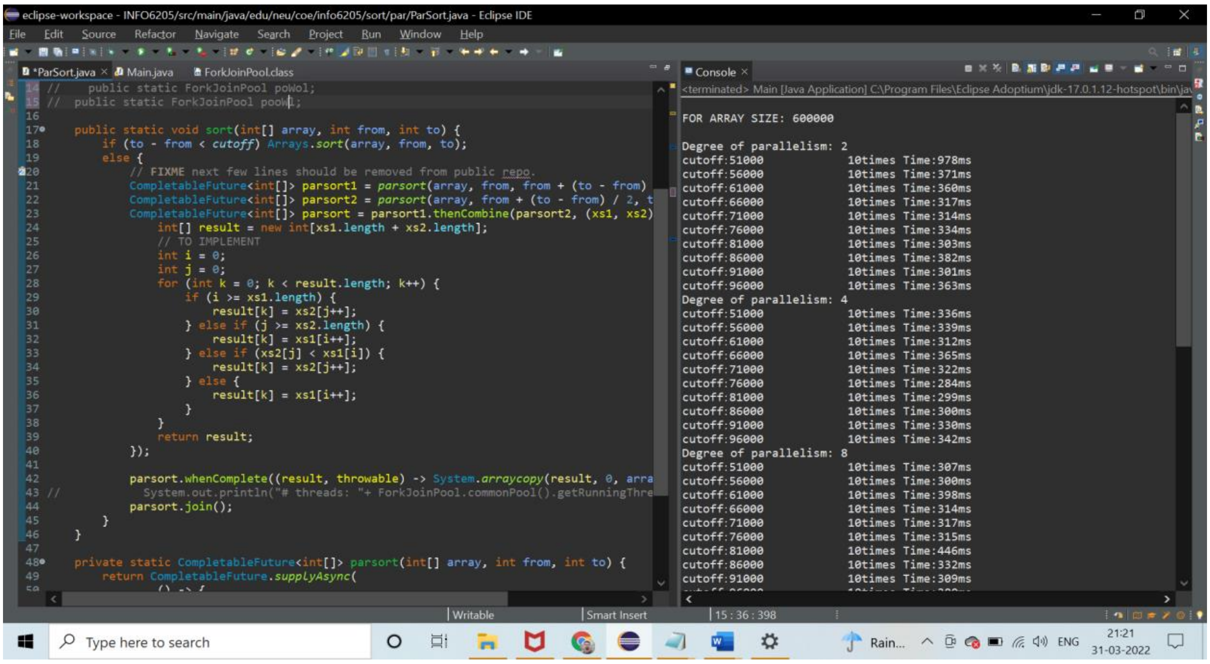
NUID: 002960393

Task: Your 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.

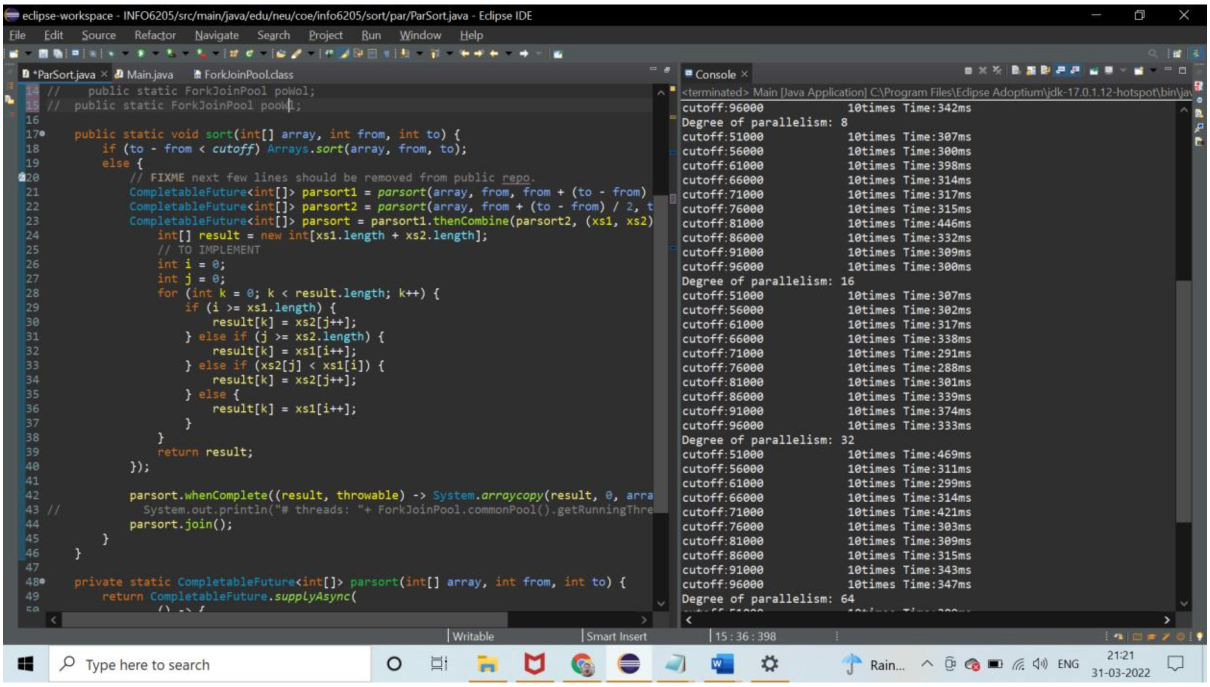
There is a *Main* class and the *ParSort* class in the *sort.par* package of the INFO6205 repository. The *Main* class can be used as is but the *ParSort* class needs to be implemented where you see "TODO..." [it turns out that these TODOs are already implemented].
Unless you have a good reason not to, you should just go along with the Java8-style future implementations provided for you in the class repository.
You must prepare a report that shows the results of your experiments and draws a conclusion (or more) about the efficacy of this method of parallelizing sort. Your experiments should involve sorting arrays of sufficient size for the parallel sort to make a difference. You should run with many different array sizes (they must be sufficiently large to make parallel sorting worthwhile, obviously) and different cutoff schemes.

OUTPUT SCREENSHOT:


eclipse-workspace - INFO6205/src/main/java/edu/neu/coe/info6205/sort/par/ParSort.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*ParSort.java
Main.java
ForkJoinPool.class
Console
Writable
Smart Insert
15 : 36 : 398
Type here to search
Rain...
ENG
21:21
31-03-2022

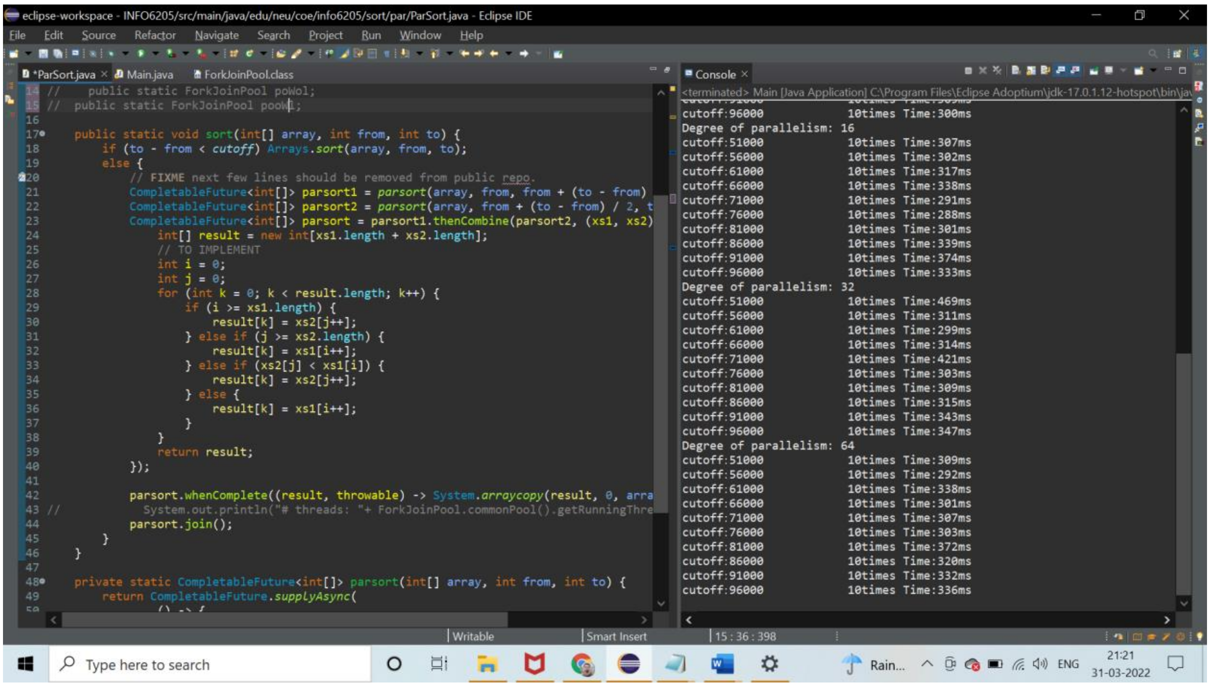
```java
//    public static ForkJoinPool poWol;
//  public static ForkJoinPool pooWl;

    public static void sort(int[] array, int from, int to) {
        if (to - from < cutoff) Arrays.sort(array, from, to);
        else {
            // FIXME next few lines should be removed from public repo.
            CompletableFuture<int[]> parsort1 = parsort(array, from, from + (to - from)
            CompletableFuture<int[]> parsort2 = parsort(array, from + (to - from) / 2, t
            CompletableFuture<int[]> parsort = parsort1.thenCombine(parsort2, (xs1, xs2)
                int[] result = new int[xs1.length + xs2.length];
                // TO IMPLEMENT
                int i = 0;
                int j = 0;
                for (int k = 0; k < result.length; k++) {
                    if (i >= xs1.length) {
                        result[k] = xs2[j++];
                    } else if (j >= xs2.length) {
                        result[k] = xs1[i++];
                    } else if (xs2[j] < xs1[i]) {
                        result[k] = xs2[j++];
                    } else {
                        result[k] = xs1[i++];
                    }
                }
                return result;
            });

            parsort.whenComplete((result, throwable) -> System.arraycopy(result, 0, arra
//            System.out.println("# threads: "+ ForkJoinPool.commonPool().getRunningThre
            parsort.join();
        }
    }

    private static CompletableFuture<int[]> parsort(int[] array, int from, int to) {
        return CompletableFuture.supplyAsync(
                () -> {
```

```
<terminated> Main [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.1.12-hotspot\bin\jav

FOR ARRAY SIZE: 600000

Degree of parallelism: 2
cutoff:51000            10times Time:978ms
cutoff:56000            10times Time:371ms
cutoff:61000            10times Time:360ms
cutoff:66000            10times Time:317ms
cutoff:71000            10times Time:314ms
cutoff:76000            10times Time:334ms
cutoff:81000            10times Time:303ms
cutoff:86000            10times Time:382ms
cutoff:91000            10times Time:301ms
cutoff:96000            10times Time:363ms
Degree of parallelism: 4
cutoff:51000            10times Time:336ms
cutoff:56000            10times Time:339ms
cutoff:61000            10times Time:312ms
cutoff:66000            10times Time:365ms
cutoff:71000            10times Time:322ms
cutoff:76000            10times Time:284ms
cutoff:81000            10times Time:299ms
cutoff:86000            10times Time:300ms
cutoff:91000            10times Time:330ms
cutoff:96000            10times Time:342ms
Degree of parallelism: 8
cutoff:51000            10times Time:307ms
cutoff:56000            10times Time:300ms
cutoff:61000            10times Time:398ms
cutoff:66000            10times Time:314ms
cutoff:71000            10times Time:317ms
cutoff:76000            10times Time:315ms
cutoff:81000            10times Time:446ms
cutoff:86000            10times Time:332ms
cutoff:91000            10times Time:309ms
```


eclipse-workspace - INFO6205/src/main/java/edu/neu/coe/info6205/sort/par/ParSort.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*ParSort.java
Main.java
ForkJoinPool.class
Console
Writable
Smart Insert
15 : 36 : 398
Type here to search
Rain...
ENG
21:21
31-03-2022

```java
//    public static ForkJoinPool poWol;
//  public static ForkJoinPool pooWl;

    public static void sort(int[] array, int from, int to) {
        if (to - from < cutoff) Arrays.sort(array, from, to);
        else {
            // FIXME next few lines should be removed from public repo.
            CompletableFuture<int[]> parsort1 = parsort(array, from, from + (to - from)
            CompletableFuture<int[]> parsort2 = parsort(array, from + (to - from) / 2, t
            CompletableFuture<int[]> parsort = parsort1.thenCombine(parsort2, (xs1, xs2)
                int[] result = new int[xs1.length + xs2.length];
                // TO IMPLEMENT
                int i = 0;
                int j = 0;
                for (int k = 0; k < result.length; k++) {
                    if (i >= xs1.length) {
                        result[k] = xs2[j++];
                    } else if (j >= xs2.length) {
                        result[k] = xs1[i++];
                    } else if (xs2[j] < xs1[i]) {
                        result[k] = xs2[j++];
                    } else {
                        result[k] = xs1[i++];
                    }
                }
                return result;
            });

            parsort.whenComplete((result, throwable) -> System.arraycopy(result, 0, arra
//            System.out.println("# threads: "+ ForkJoinPool.commonPool().getRunningThre
            parsort.join();
        }
    }

    private static CompletableFuture<int[]> parsort(int[] array, int from, int to) {
        return CompletableFuture.supplyAsync(
                () -> {
```

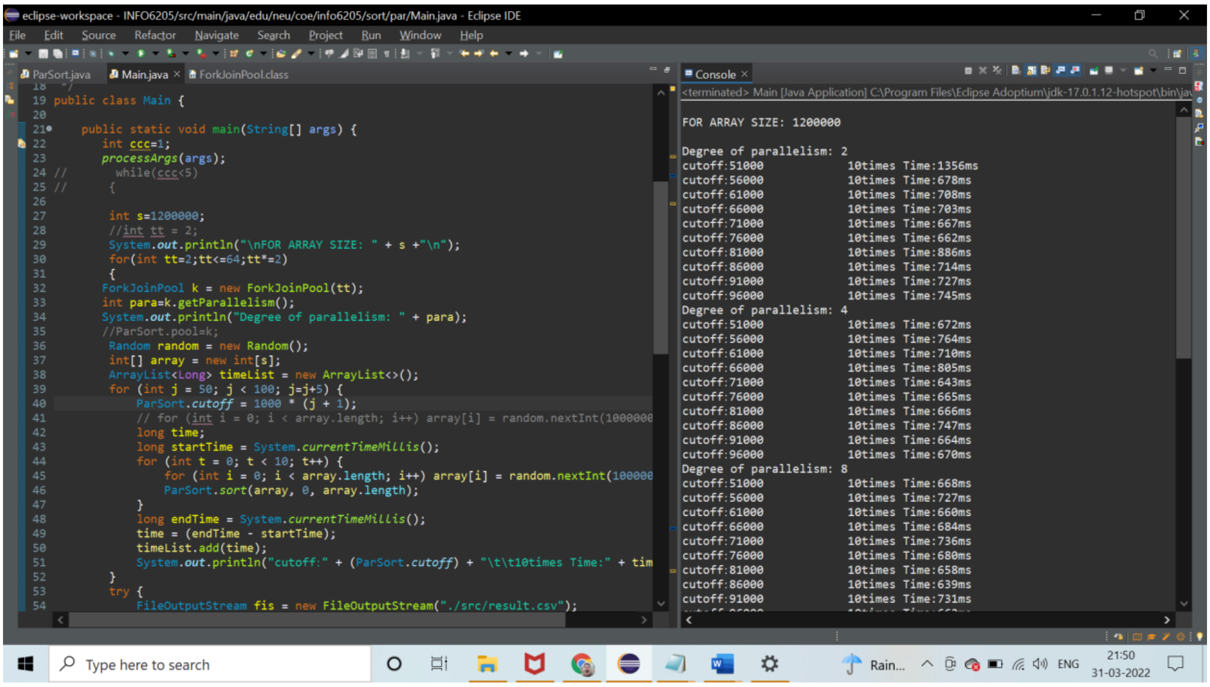
```
<terminated> Main [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.1.12-hotspot\bin\jav
cutoff:96000            10times Time:342ms
Degree of parallelism: 8
cutoff:51000            10times Time:307ms
cutoff:56000            10times Time:300ms
cutoff:61000            10times Time:398ms
cutoff:66000            10times Time:314ms
cutoff:71000            10times Time:317ms
cutoff:76000            10times Time:315ms
cutoff:81000            10times Time:446ms
cutoff:86000            10times Time:332ms
cutoff:91000            10times Time:309ms
cutoff:96000            10times Time:300ms
Degree of parallelism: 16
cutoff:51000            10times Time:307ms
cutoff:56000            10times Time:302ms
cutoff:61000            10times Time:317ms
cutoff:66000            10times Time:338ms
cutoff:71000            10times Time:291ms
cutoff:76000            10times Time:288ms
cutoff:81000            10times Time:301ms
cutoff:86000            10times Time:339ms
cutoff:91000            10times Time:374ms
cutoff:96000            10times Time:333ms
Degree of parallelism: 32
cutoff:51000            10times Time:469ms
cutoff:56000            10times Time:311ms
cutoff:61000            10times Time:299ms
cutoff:66000            10times Time:314ms
cutoff:71000            10times Time:421ms
cutoff:76000            10times Time:303ms
cutoff:81000            10times Time:309ms
cutoff:86000            10times Time:315ms
cutoff:91000            10times Time:343ms
cutoff:96000            10times Time:347ms
Degree of parallelism: 64
```



eclipse-workspace - INFO6205/src/main/java/edu/neu/coe/info6205/sort/par/ParSort.java - Eclipse IDE

```java
//    public static ForkJoinPool poWol;
//  public static ForkJoinPool pooWl;

    public static void sort(int[] array, int from, int to) {
        if (to - from < cutoff) Arrays.sort(array, from, to);
        else {
            // FIXME next few lines should be removed from public repo.
            CompletableFuture<int[]> parsort1 = parsort(array, from, from + (to - from)
            CompletableFuture<int[]> parsort2 = parsort(array, from + (to - from) / 2, t
            CompletableFuture<int[]> parsort = parsort1.thenCombine(parsort2, (xs1, xs2)
                int[] result = new int[xs1.length + xs2.length];
                // TO IMPLEMENT
                int i = 0;
                int j = 0;
                for (int k = 0; k < result.length; k++) {
                    if (i >= xs1.length) {
                        result[k] = xs2[j++];
                    } else if (j >= xs2.length) {
                        result[k] = xs1[i++];
                    } else if (xs2[j] < xs1[i]) {
                        result[k] = xs2[j++];
                    } else {
                        result[k] = xs1[i++];
                    }
                }
                return result;
            });

            parsort.whenComplete((result, throwable) -> System.arraycopy(result, 0, arra
//            System.out.println("# threads: "+ ForkJoinPool.commonPool().getRunningThre
            parsort.join();
        }
    }

    private static CompletableFuture<int[]> parsort(int[] array, int from, int to) {
        return CompletableFuture.supplyAsync(
                () -> {
```

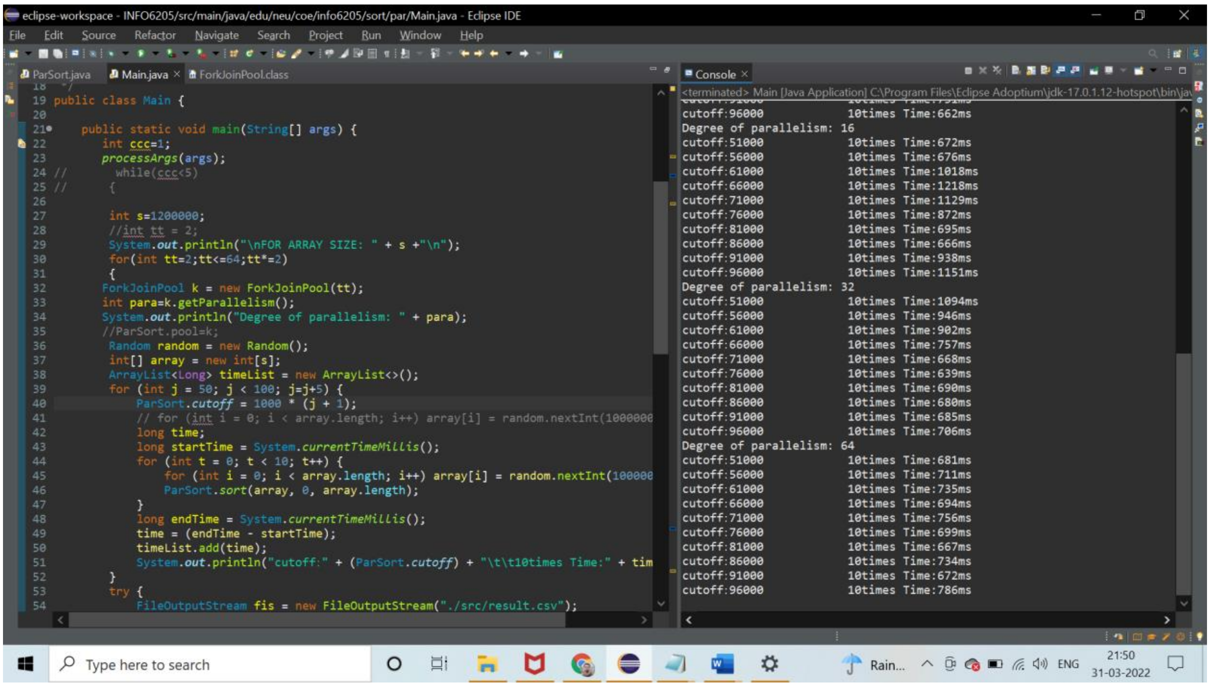
```
<terminated> Main [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.1.12-hotspot\bin\jav
cutoff:96000            10times Time:300ms
Degree of parallelism: 16
cutoff:51000            10times Time:307ms
cutoff:56000            10times Time:302ms
cutoff:61000            10times Time:317ms
cutoff:66000            10times Time:338ms
cutoff:71000            10times Time:291ms
cutoff:76000            10times Time:288ms
cutoff:81000            10times Time:301ms
cutoff:86000            10times Time:339ms
cutoff:91000            10times Time:374ms
cutoff:96000            10times Time:333ms
Degree of parallelism: 32
cutoff:51000            10times Time:469ms
cutoff:56000            10times Time:311ms
cutoff:61000            10times Time:299ms
cutoff:66000            10times Time:314ms
cutoff:71000            10times Time:421ms
cutoff:76000            10times Time:303ms
cutoff:81000            10times Time:309ms
cutoff:86000            10times Time:315ms
cutoff:91000            10times Time:343ms
cutoff:96000            10times Time:347ms
Degree of parallelism: 64
cutoff:51000            10times Time:309ms
cutoff:56000            10times Time:292ms
cutoff:61000            10times Time:338ms
cutoff:66000            10times Time:301ms
cutoff:71000            10times Time:307ms
cutoff:76000            10times Time:303ms
cutoff:81000            10times Time:372ms
cutoff:86000            10times Time:320ms
cutoff:91000            10times Time:332ms
cutoff:96000            10times Time:336ms
```



eclipse-workspace - INFO6205/src/main/java/edu/neu/coe/info6205/sort/par/Main.java - Eclipse IDE

```java
 */
public class Main {

    public static void main(String[] args) {
        int ccc=1;
        processArgs(args);
//          while(ccc<5)
//         {

         int s=1200000;
         //int tt = 2;
         System.out.println("\nFOR ARRAY SIZE: " + s +"\n");
         for(int tt=2;tt<=64;tt*=2)
         {
        ForkJoinPool k = new ForkJoinPool(tt);
        int para=k.getParallelism();
        System.out.println("Degree of parallelism: " + para);
        //ParSort.pool=k;
         Random random = new Random();
         int[] array = new int[s];
         ArrayList<Long> timeList = new ArrayList<>();
         for (int j = 50; j < 100; j=j+5) {
             ParSort.cutoff = 1000 * (j + 1);
             // for (int i = 0; i < array.length; i++) array[i] = random.nextInt(1000000
             long time;
             long startTime = System.currentTimeMillis();
             for (int t = 0; t < 10; t++) {
                 for (int i = 0; i < array.length; i++) array[i] = random.nextInt(100000
                 ParSort.sort(array, 0, array.length);
             }
             long endTime = System.currentTimeMillis();
             time = (endTime - startTime);
             timeList.add(time);
             System.out.println("cutoff:" + (ParSort.cutoff) + "\t\t10times Time:" + tim
         }
         try {
             FileOutputStream fis = new FileOutputStream("./src/result.csv");
```

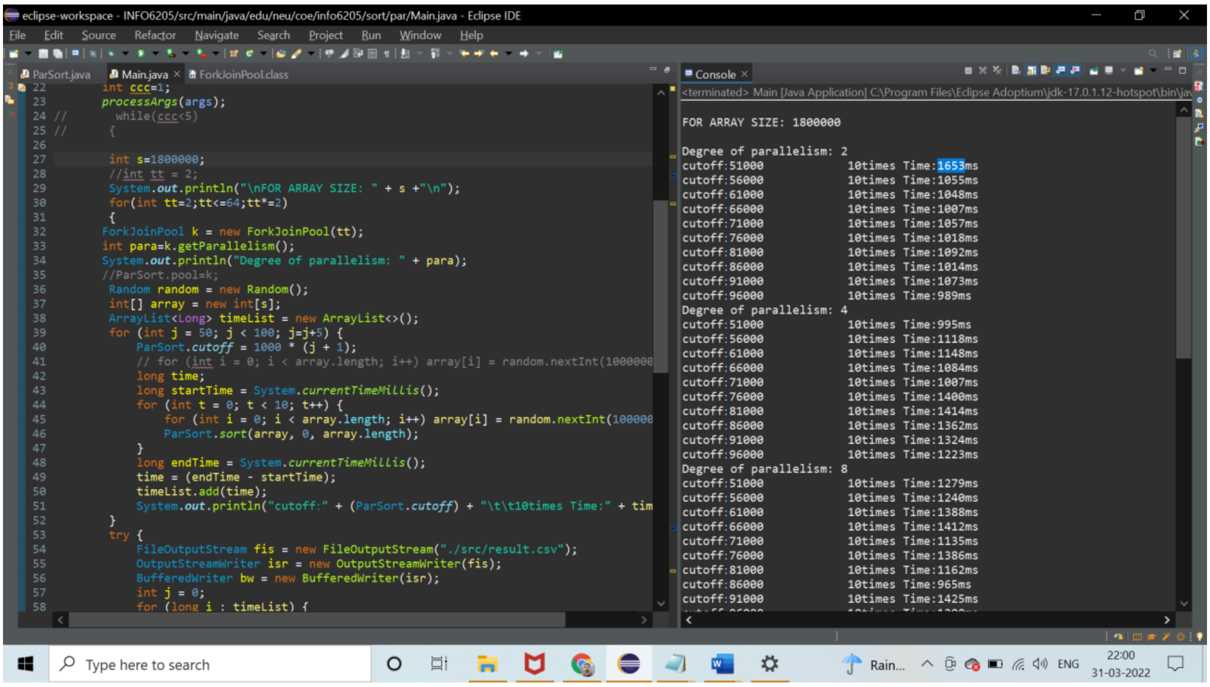
```
<terminated> Main [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.1.12-hotspot\bin\jav

FOR ARRAY SIZE: 1200000

Degree of parallelism: 2
cutoff:51000            10times Time:1356ms
cutoff:56000            10times Time:678ms
cutoff:61000            10times Time:708ms
cutoff:66000            10times Time:703ms
cutoff:71000            10times Time:667ms
cutoff:76000            10times Time:662ms
cutoff:81000            10times Time:886ms
cutoff:86000            10times Time:714ms
cutoff:91000            10times Time:727ms
cutoff:96000            10times Time:745ms
Degree of parallelism: 4
cutoff:51000            10times Time:672ms
cutoff:56000            10times Time:764ms
cutoff:61000            10times Time:710ms
cutoff:66000            10times Time:805ms
cutoff:71000            10times Time:643ms
cutoff:76000            10times Time:665ms
cutoff:81000            10times Time:666ms
cutoff:86000            10times Time:747ms
cutoff:91000            10times Time:664ms
cutoff:96000            10times Time:670ms
Degree of parallelism: 8
cutoff:51000            10times Time:668ms
cutoff:56000            10times Time:727ms
cutoff:61000            10times Time:660ms
cutoff:66000            10times Time:684ms
cutoff:71000            10times Time:736ms
cutoff:76000            10times Time:680ms
cutoff:81000            10times Time:658ms
cutoff:86000            10times Time:639ms
cutoff:91000            10times Time:731ms
```



eclipse-workspace - INFO6205/src/main/java/edu/neu/coe/info6205/sort/par/Main.java - Eclipse IDE

```java
public class Main {

    public static void main(String[] args) {
        int ccc=1;
        processArgs(args);
//        while(ccc<5)
//        {

         int s=1200000;
         //int tt = 2;
         System.out.println("\nFOR ARRAY SIZE: " + s +"\n");
         for(int tt=2;tt<=64;tt*=2)
         {
        ForkJoinPool k = new ForkJoinPool(tt);
        int para=k.getParallelism();
        System.out.println("Degree of parallelism: " + para);
        //ParSort.pool=k;
         Random random = new Random();
         int[] array = new int[s];
         ArrayList<Long> timeList = new ArrayList<>();
         for (int j = 50; j < 100; j=j+5) {
             ParSort.cutoff = 1000 * (j + 1);
             // for (int i = 0; i < array.length; i++) array[i] = random.nextInt(1000000
             long time;
             long startTime = System.currentTimeMillis();
             for (int t = 0; t < 10; t++) {
                 for (int i = 0; i < array.length; i++) array[i] = random.nextInt(100000
                 ParSort.sort(array, 0, array.length);
             }
             long endTime = System.currentTimeMillis();
             time = (endTime - startTime);
             timeList.add(time);
             System.out.println("cutoff:" + (ParSort.cutoff) + "\t\t10times Time:" + tim
         }
         try {
             FileOutputStream fis = new FileOutputStream("./src/result.csv");
```

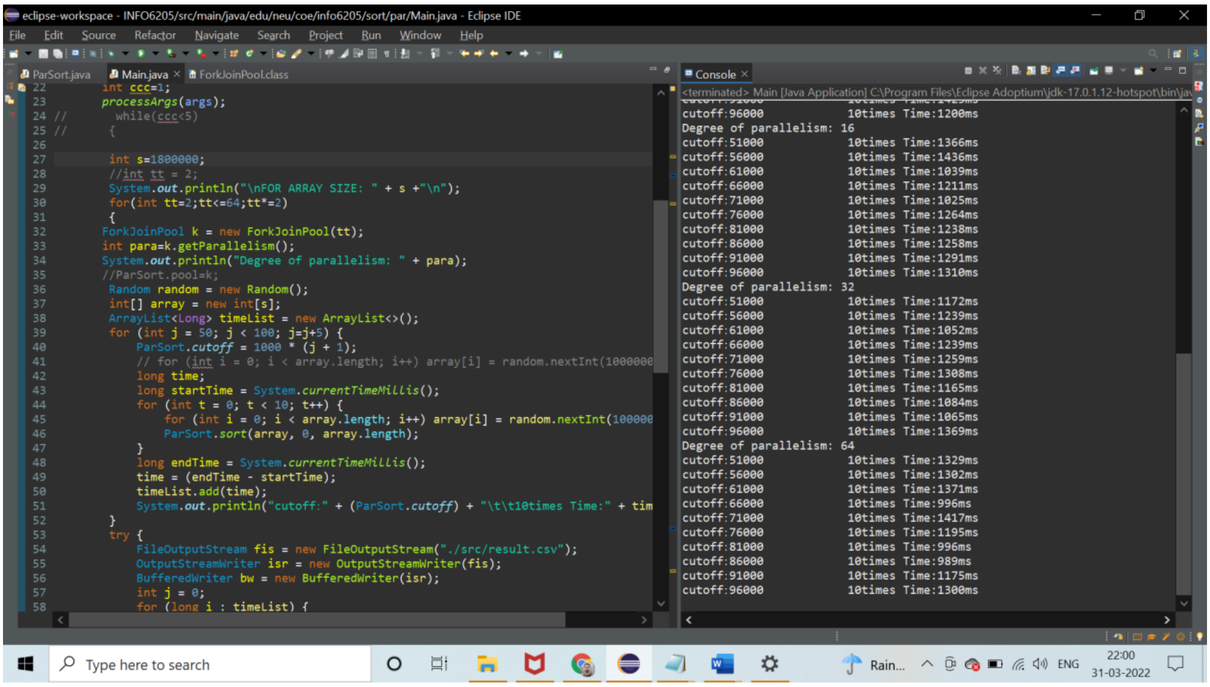
```
<terminated> Main [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.1.12-hotspot\bin\jav
cutoff:96000            10times Time:662ms
Degree of parallelism: 16
cutoff:51000            10times Time:672ms
cutoff:56000            10times Time:676ms
cutoff:61000            10times Time:1018ms
cutoff:66000            10times Time:1218ms
cutoff:71000            10times Time:1129ms
cutoff:76000            10times Time:872ms
cutoff:81000            10times Time:695ms
cutoff:86000            10times Time:666ms
cutoff:91000            10times Time:938ms
cutoff:96000            10times Time:1151ms
Degree of parallelism: 32
cutoff:51000            10times Time:1094ms
cutoff:56000            10times Time:946ms
cutoff:61000            10times Time:902ms
cutoff:66000            10times Time:757ms
cutoff:71000            10times Time:668ms
cutoff:76000            10times Time:639ms
cutoff:81000            10times Time:690ms
cutoff:86000            10times Time:680ms
cutoff:91000            10times Time:685ms
cutoff:96000            10times Time:706ms
Degree of parallelism: 64
cutoff:51000            10times Time:681ms
cutoff:56000            10times Time:711ms
cutoff:61000            10times Time:735ms
cutoff:66000            10times Time:694ms
cutoff:71000            10times Time:756ms
cutoff:76000            10times Time:699ms
cutoff:81000            10times Time:667ms
cutoff:86000            10times Time:734ms
cutoff:91000            10times Time:672ms
cutoff:96000            10times Time:786ms
```



eclipse-workspace - INFO6205/src/main/java/edu/neu/coe/info6205/sort/par/Main.java - Eclipse IDE

```java
        int ccc=1;
        processArgs(args);
//        while(ccc<5)
//        {

         int s=1800000;
         //int tt = 2;
         System.out.println("\nFOR ARRAY SIZE: " + s +"\n");
         for(int tt=2;tt<=64;tt*=2)
         {
        ForkJoinPool k = new ForkJoinPool(tt);
        int para=k.getParallelism();
        System.out.println("Degree of parallelism: " + para);
        //ParSort.pool=k;
         Random random = new Random();
         int[] array = new int[s];
         ArrayList<Long> timeList = new ArrayList<>();
         for (int j = 50; j < 100; j=j+5) {
             ParSort.cutoff = 1000 * (j + 1);
             // for (int i = 0; i < array.length; i++) array[i] = random.nextInt(1000000
             long time;
             long startTime = System.currentTimeMillis();
             for (int t = 0; t < 10; t++) {
                 for (int i = 0; i < array.length; i++) array[i] = random.nextInt(100000
                 ParSort.sort(array, 0, array.length);
             }
             long endTime = System.currentTimeMillis();
             time = (endTime - startTime);
             timeList.add(time);
             System.out.println("cutoff:" + (ParSort.cutoff) + "\t\t10times Time:" + tim
         }
         try {
             FileOutputStream fis = new FileOutputStream("./src/result.csv");
             OutputStreamWriter isr = new OutputStreamWriter(fis);
             BufferedWriter bw = new BufferedWriter(isr);
             int j = 0;
             for (long i : timeList) {
```

```
<terminated> Main [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.1.12-hotspot\bin\jav

FOR ARRAY SIZE: 1800000

Degree of parallelism: 2
cutoff:51000            10times Time:1653ms
cutoff:56000            10times Time:1055ms
cutoff:61000            10times Time:1048ms
cutoff:66000            10times Time:1007ms
cutoff:71000            10times Time:1057ms
cutoff:76000            10times Time:1018ms
cutoff:81000            10times Time:1092ms
cutoff:86000            10times Time:1014ms
cutoff:91000            10times Time:1073ms
cutoff:96000            10times Time:989ms
Degree of parallelism: 4
cutoff:51000            10times Time:995ms
cutoff:56000            10times Time:1118ms
cutoff:61000            10times Time:1148ms
cutoff:66000            10times Time:1084ms
cutoff:71000            10times Time:1007ms
cutoff:76000            10times Time:1400ms
cutoff:81000            10times Time:1414ms
cutoff:86000            10times Time:1362ms
cutoff:91000            10times Time:1324ms
cutoff:96000            10times Time:1223ms
Degree of parallelism: 8
cutoff:51000            10times Time:1279ms
cutoff:56000            10times Time:1240ms
cutoff:61000            10times Time:1388ms
cutoff:66000            10times Time:1412ms
cutoff:71000            10times Time:1135ms
cutoff:76000            10times Time:1386ms
cutoff:81000            10times Time:1162ms
cutoff:86000            10times Time:965ms
cutoff:91000            10times Time:1425ms
```


eclipse-workspace - INFO6205/src/main/java/edu/neu/coe/info6205/sort/par/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
ParSort.java
Main.java
ForkJoinPool.class
Console
Type here to search
Rain...
ENG
22:00
31-03-2022

```
        int ccc=1;
        processArgs(args);
//       while(ccc<5)
//      {

         int s=1800000;
         //int tt = 2;
         System.out.println("\nFOR ARRAY SIZE: " + s +"\n");
         for(int tt=2;tt<=64;tt*=2)
         {
        ForkJoinPool k = new ForkJoinPool(tt);
        int para=k.getParallelism();
        System.out.println("Degree of parallelism: " + para);
        //ParSort.pool=k;
         Random random = new Random();
         int[] array = new int[s];
         ArrayList<Long> timeList = new ArrayList<>();
         for (int j = 50; j < 100; j=j+5) {
             ParSort.cutoff = 1000 * (j + 1);
             // for (int i = 0; i < array.length; i++) array[i] = random.nextInt(1000000
             long time;
             long startTime = System.currentTimeMillis();
             for (int t = 0; t < 10; t++) {
                 for (int i = 0; i < array.length; i++) array[i] = random.nextInt(100000
                 ParSort.sort(array, 0, array.length);
             }
             long endTime = System.currentTimeMillis();
             time = (endTime - startTime);
             timeList.add(time);
             System.out.println("cutoff:" + (ParSort.cutoff) + "\t\t10times Time:" + tim
         }
         try {
             FileOutputStream fis = new FileOutputStream("./src/result.csv");
             OutputStreamWriter isr = new OutputStreamWriter(fis);
             BufferedWriter bw = new BufferedWriter(isr);
             int j = 0;
             for (long i : timeList) {
```

```
<terminated> Main [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.1.12-hotspot\bin\jav
cutoff:96000            10times Time:1200ms
Degree of parallelism: 16
cutoff:51000            10times Time:1366ms
cutoff:56000            10times Time:1436ms
cutoff:61000            10times Time:1039ms
cutoff:66000            10times Time:1211ms
cutoff:71000            10times Time:1025ms
cutoff:76000            10times Time:1264ms
cutoff:81000            10times Time:1238ms
cutoff:86000            10times Time:1258ms
cutoff:91000            10times Time:1291ms
cutoff:96000            10times Time:1310ms
Degree of parallelism: 32
cutoff:51000            10times Time:1172ms
cutoff:56000            10times Time:1239ms
cutoff:61000            10times Time:1052ms
cutoff:66000            10times Time:1239ms
cutoff:71000            10times Time:1259ms
cutoff:76000            10times Time:1308ms
cutoff:81000            10times Time:1165ms
cutoff:86000            10times Time:1084ms
cutoff:91000            10times Time:1065ms
cutoff:96000            10times Time:1369ms
Degree of parallelism: 64
cutoff:51000            10times Time:1329ms
cutoff:56000            10times Time:1302ms
cutoff:61000            10times Time:1371ms
cutoff:66000            10times Time:996ms
cutoff:71000            10times Time:1417ms
cutoff:76000            10times Time:1195ms
cutoff:81000            10times Time:996ms
cutoff:86000            10times Time:989ms
cutoff:91000            10times Time:1175ms
cutoff:96000            10times Time:1300ms
```

Console Output:

```
FOR ARRAY SIZE: 600000

Degree of parallelism: 2
cutoff : 51000                10times Time:978ms
cutoff : 56000                10times Time:371ms
cutoff : 61000                10times Time:360ms
cutoff : 66000                10times Time:317ms
cutoff : 71000                10times Time:314ms
cutoff : 76000                10times Time:334ms
cutoff : 81000                10times Time:303ms
cutoff : 86000                10times Time:382ms
cutoff : 91000                10times Time:301ms
cutoff : 96000                10times Time:363ms
Degree of parallelism: 4
cutoff : 51000                10times Time:336ms
cutoff : 56000                10times Time:339ms
cutoff : 61000                10times Time:312ms
cutoff : 66000                10times Time:365ms
cutoff : 71000                10times Time:322ms
cutoff : 76000                10times Time:284ms
cutoff : 81000                10times Time:299ms
cutoff : 86000                10times Time:300ms
cutoff : 91000                10times Time:330ms
cutoff : 96000                10times Time:342ms
Degree of parallelism: 8
cutoff : 51000                10times Time:307ms
cutoff : 56000                10times Time:300ms
cutoff : 61000                10times Time:398ms
cutoff : 66000                10times Time:314ms
cutoff : 71000                10times Time:317ms
cutoff : 76000                10times Time:315ms
cutoff : 81000                10times Time:446ms
cutoff : 86000                10times Time:332ms
cutoff : 91000                10times Time:309ms
cutoff : 96000                10times Time:300ms
```

```
Degree of parallelism: 16
cutoff :  51000                  10times Time:307ms
cutoff :  56000                  10times Time:302ms
cutoff :  61000                  10times Time:317ms
cutoff :  66000                  10times Time:338ms
cutoff :  71000                  10times Time:291ms
cutoff :  76000                  10times Time:288ms
cutoff :  81000                  10times Time:301ms
cutoff :  86000                  10times Time:339ms
cutoff :  91000                  10times Time:374ms
cutoff :  96000                  10times Time:333ms
Degree of parallelism: 32
cutoff :  51000                  10times Time:469ms
cutoff :  56000                  10times Time:311ms
cutoff :  61000                  10times Time:299ms
cutoff :  66000                  10times Time:314ms
cutoff :  71000                  10times Time:421ms
cutoff :  76000                  10times Time:303ms
cutoff :  81000                  10times Time:309ms
cutoff :  86000                  10times Time:315ms
cutoff :  91000                  10times Time:343ms
cutoff :  96000                  10times Time:347ms
Degree of parallelism: 64
cutoff :  51000                  10times Time:309ms
cutoff :  56000                  10times Time:292ms
cutoff :  61000                  10times Time:338ms
cutoff :  66000                  10times Time:301ms
cutoff :  71000                  10times Time:307ms
cutoff :  76000                  10times Time:303ms
cutoff :  81000                  10times Time:372ms
cutoff :  86000                  10times Time:320ms
cutoff :  91000                  10times Time:332ms
cutoff :  96000                  10times Time:336ms


FOR ARRAY SIZE: 1200000

Degree of parallelism: 2
cutoff :  51000                  10times Time:1356ms
cutoff :  56000                  10times Time:678ms
cutoff :  61000                  10times Time:708ms
cutoff :  66000                  10times Time:703ms
cutoff :  71000                  10times Time:667ms
cutoff :  76000                  10times Time:662ms
cutoff :  81000                  10times Time:886ms
cutoff :  86000                  10times Time:714ms
cutoff :  91000                  10times Time:727ms
cutoff :  96000                  10times Time:745ms
Degree of parallelism: 4
cutoff :  51000                  10times Time:672ms
cutoff :  56000                  10times Time:764ms
cutoff :  61000                  10times Time:710ms
cutoff :  66000                  10times Time:805ms
cutoff :  71000                  10times Time:643ms
cutoff :  76000                  10times Time:665ms
cutoff :  81000                  10times Time:666ms
cutoff :  86000                  10times Time:747ms
cutoff :  91000                  10times Time:664ms
cutoff :  96000                  10times Time:670ms
Degree of parallelism: 8
cutoff :  51000                  10times Time:668ms
cutoff :  56000                  10times Time:727ms
```

```
cutoff： 61000                    10times Time:660ms
cutoff： 66000                    10times Time:684ms
cutoff： 71000                    10times Time:736ms
cutoff： 76000                    10times Time:680ms
cutoff： 81000                    10times Time:658ms
cutoff： 86000                    10times Time:639ms
cutoff： 91000                    10times Time:731ms
cutoff： 96000                    10times Time:662ms
Degree of parallelism: 16
cutoff： 51000                    10times Time:672ms
cutoff： 56000                    10times Time:676ms
cutoff： 61000                    10times Time:1018ms
cutoff： 66000                    10times Time:1218ms
cutoff： 71000                    10times Time:1129ms
cutoff： 76000                    10times Time:872ms
cutoff： 81000                    10times Time:695ms
cutoff： 86000                    10times Time:666ms
cutoff： 91000                    10times Time:938ms
cutoff： 96000                    10times Time:1151ms
Degree of parallelism: 32
cutoff： 51000                    10times Time:1094ms
cutoff： 56000                    10times Time:946ms
cutoff： 61000                    10times Time:902ms
cutoff： 66000                    10times Time:757ms
cutoff： 71000                    10times Time:668ms
cutoff： 76000                    10times Time:639ms
cutoff： 81000                    10times Time:690ms
cutoff： 86000                    10times Time:680ms
cutoff： 91000                    10times Time:685ms
cutoff： 96000                    10times Time:706ms
Degree of parallelism: 64
cutoff： 51000                    10times Time:681ms
cutoff： 56000                    10times Time:711ms
cutoff： 61000                    10times Time:735ms
cutoff： 66000                    10times Time:694ms
cutoff： 71000                    10times Time:756ms
cutoff： 76000                    10times Time:699ms
cutoff： 81000                    10times Time:667ms
cutoff： 86000                    10times Time:734ms
cutoff： 91000                    10times Time:672ms
cutoff： 96000                    10times Time:786ms


FOR ARRAY SIZE: 1800000

Degree of parallelism: 2
cutoff： 51000                    10times Time:1653ms
cutoff： 56000                    10times Time:1055ms
cutoff： 61000                    10times Time:1048ms
cutoff： 66000                    10times Time:1007ms
cutoff： 71000                    10times Time:1057ms
cutoff： 76000                    10times Time:1018ms
cutoff： 81000                    10times Time:1092ms
cutoff： 86000                    10times Time:1014ms
cutoff： 91000                    10times Time:1073ms
cutoff： 96000                    10times Time:989ms
Degree of parallelism: 4
cutoff： 51000                    10times Time:995ms
cutoff： 56000                    10times Time:1118ms
cutoff： 61000                    10times Time:1148ms
cutoff： 66000                    10times Time:1084ms
cutoff： 71000                    10times Time:1007ms
```

```
cutoff：76000                10times Time:1400ms
cutoff：81000                10times Time:1414ms
cutoff：86000                10times Time:1362ms
cutoff：91000                10times Time:1324ms
cutoff：96000                10times Time:1223ms
Degree of parallelism: 8
cutoff：51000                10times Time:1279ms
cutoff：56000                10times Time:1240ms
cutoff：61000                10times Time:1388ms
cutoff：66000                10times Time:1412ms
cutoff：71000                10times Time:1135ms
cutoff：76000                10times Time:1386ms
cutoff：81000                10times Time:1162ms
cutoff：86000                10times Time:965ms
cutoff：91000                10times Time:1425ms
cutoff：96000                10times Time:1200ms
Degree of parallelism: 16
cutoff：51000                10times Time:1366ms
cutoff：56000                10times Time:1436ms
cutoff：61000                10times Time:1039ms
cutoff：66000                10times Time:1211ms
cutoff：71000                10times Time:1025ms
cutoff：76000                10times Time:1264ms
cutoff：81000                10times Time:1238ms
cutoff：86000                10times Time:1258ms
cutoff：91000                10times Time:1291ms
cutoff：96000                10times Time:1310ms
Degree of parallelism: 32
cutoff：51000                10times Time:1172ms
cutoff：56000                10times Time:1239ms
cutoff：61000                10times Time:1052ms
cutoff：66000                10times Time:1239ms
cutoff：71000                10times Time:1259ms
cutoff：76000                10times Time:1308ms
cutoff：81000                10times Time:1165ms
cutoff：86000                10times Time:1084ms
cutoff：91000                10times Time:1065ms
cutoff：96000                10times Time:1369ms
Degree of parallelism: 64
cutoff：51000                10times Time:1329ms
cutoff：56000                10times Time:1302ms
cutoff：61000                10times Time:1371ms
cutoff：66000                10times Time:996ms
cutoff：71000                10times Time:1417ms
cutoff：76000                10times Time:1195ms
cutoff：81000                10times Time:996ms
cutoff：86000                10times Time:989ms
cutoff：91000                10times Time:1175ms
cutoff：96000                10times Time:1300ms
```

Relationship/Conclusion:

It is quite evident from the above outputs that even with the different cutoff values, array sizes there is no rich difference in the time taken after 4 threads, it all comes out to be the same (approximately). That is there is significant increase (especially when the cutoff value is least) in performance when threads are increased from 2 to 4 but not much difference is there when it(threads) is increased subsequently.

Thus, better performance is achieved when cutoff values increases and threads are increased from 2 to 4. But as stated below performance is best in certain cutoff range.

As depicted in the graph, the optimal cutoff range for which the best performance is archived is when it is between 0.20 to 0.40 percentage of the array size.

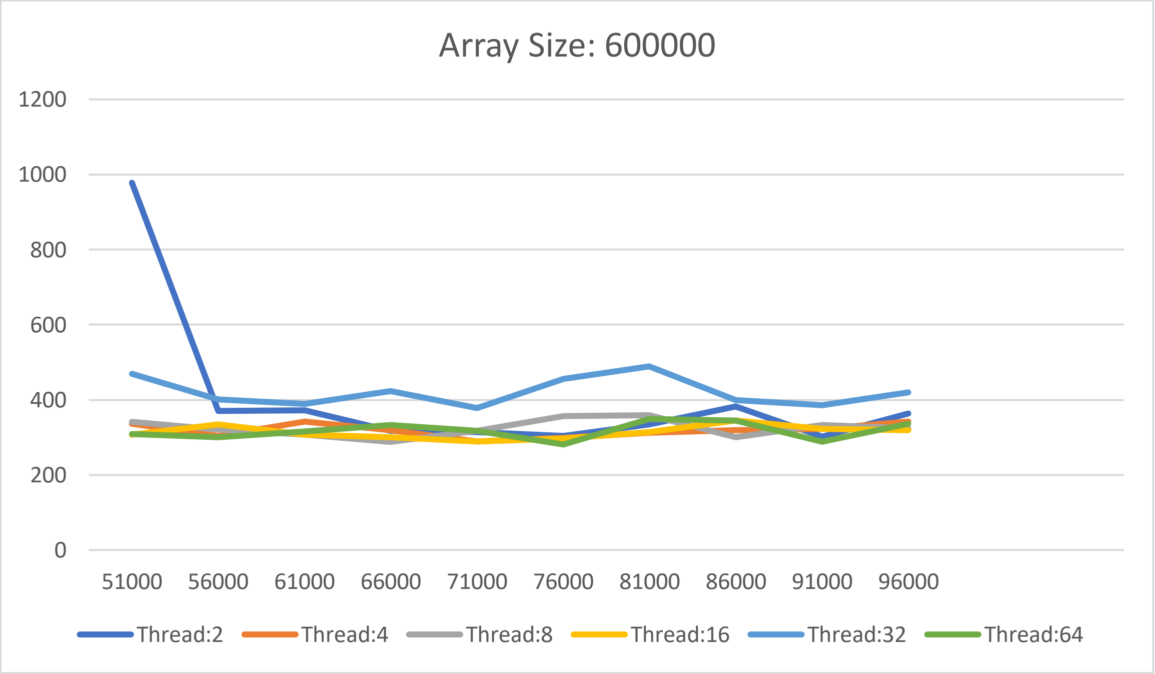
The performance takes a hit when it is between .50 and 1.

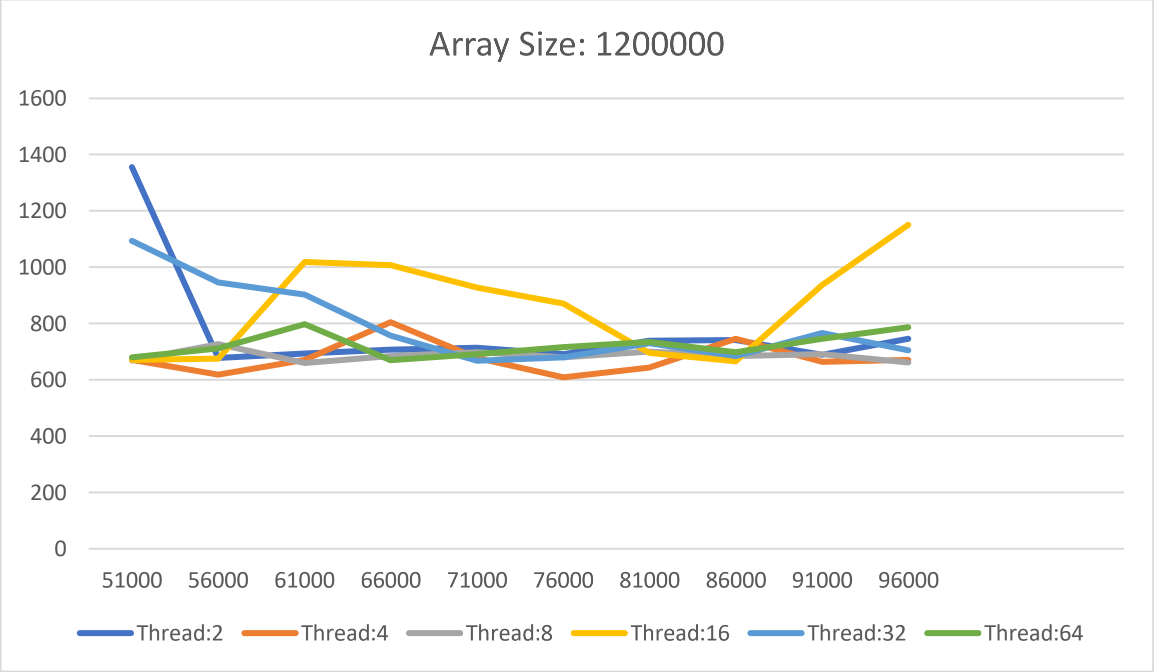
1) Best performance is when threads are 4 (optimal choice)
2) Best performance is when cutoff range is .20 - .40 % of the array size.
3) Thus, to have overall best performance in terms of threads and cutoff range I would say with 4 threads and cutoff of .30% of array size would give us the best performance.

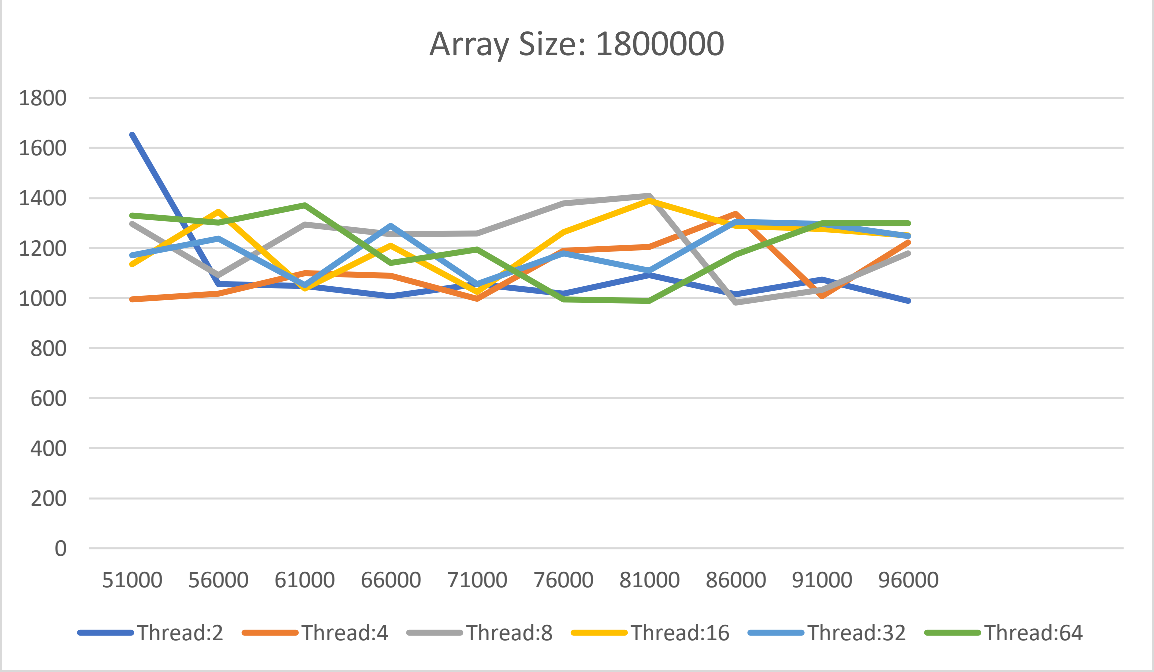
Evidence: Below are the graphs to prove the relationship depicted above.

X axix: Cut off and Y-axis: Time(ms)

Three graphs for three different array sizes.




Array Size: 1200000
1600
1400
1200
1000
800
600
400
200
0
51000
56000
61000
66000
71000
76000
81000
86000
91000
96000
Thread:2
Thread:4
Thread:8
Thread:16
Thread:32
Thread:64


Array Size: 1800000
1800
1600
1400
1200
1000
800
600
400
200
0
51000
56000
61000
66000
71000
76000
81000
86000
91000
96000
Thread:2
Thread:4
Thread:8
Thread:16
Thread:32
Thread:64