# Assignment -5 | PARALLEL SORTING

**PROBLEM**: To implement a parallel sorting algorithm such that each array partition is sorted in parallel. You will consider two different schemes for deciding whether to sort in parallel.

**CODE SCREENSHOTS**:

Main.java ParSort.java

```java
public class Main {

    public static void main(String[] args) {
        processArgs(args);
        int thread = 2, arraySize = 900000;
        while (thread < 65) {
            ForkJoinPool pool = new ForkJoinPool(thread);
            System.out.println("Degree of parallelism: " + pool.getParallelism());
            Random random = new Random();
            int[] array = new int[arraySize];
            ArrayList<Long> timeList = new ArrayList<>();
            for (int j = 0; j < 10; j++) {
                ParSort.cutoff = 9000 * (j + 1);
                // for (int i = 0; i < array.length; i++) array[i] = random.nextInt(10000000);
                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(10000000);
                    ParSort.sort(array, 0, array.length,pool);//array, 0, array.length, pool);
                }
                long endTime = System.currentTimeMillis();
                time = (endTime - startTime);
                timeList.add(time);

                System.out.println("cutoff: " + (ParSort.cutoff) + "\t\t10 times Time:" + time + "ms");
            }
            try {
                FileOutputStream fis = new FileOutputStream("./src/result.csv" + arraySize + "thread" + thread + ".csv");
                OutputStreamWriter isr = new OutputStreamWriter(fis);
                BufferedWriter bw = new BufferedWriter(isr);
```

Main.java × ParSort.java ×

```java
                int j = 0;
                for (long i : timeList) {
                    String content = (double) ParSort.cutoff * (j + 1) / arraySize + "," + (double) i / 10 + "\n";
                    j++;
                    bw.write(content);
                    bw.flush();
                }
                bw.close();

            } catch (IOException e) {
                e.printStackTrace();
            }
            thread *= 2;
        }
    }


    1 usage
    private static void processArgs(String[] args) {
        String[] xs = args;
        while (xs.length > 0)
            if (xs[0].startsWith("-")) xs = processArg(xs);
    }

    1 usage
    private static String[] processArg(String[] xs) {
        String[] result = new String[0];
        System.arraycopy(xs, srcPos: 2, result, destPos: 0, length: xs.length - 2);
        processCommand(xs[0], xs[1]);
        return result;
```

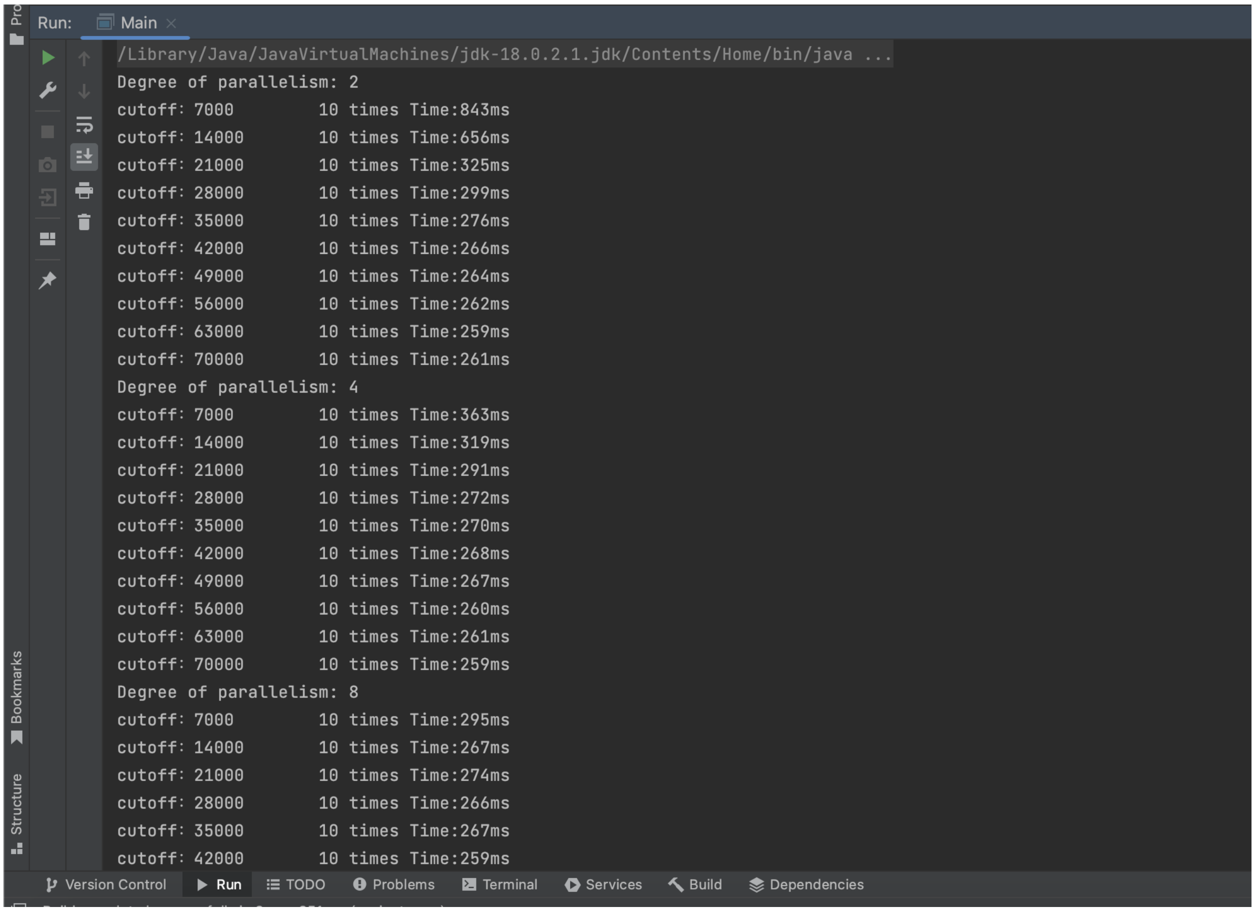
```java

    1 usage
    private static String[] processArg(String[] xs) {
        String[] result = new String[0];
        System.arraycopy(xs, srcPos: 2, result, destPos: 0, length: xs.length - 2);
        processCommand(xs[0], xs[1]);
        return result;
    }

    1 usage
    private static void processCommand(String x, String y) {
        if (x.equalsIgnoreCase( anotherString: "N")) setConfig(x, Integer.parseInt(y));
        else
            // TODO sort this out
            if (x.equalsIgnoreCase( anotherString: "P")) //noinspection ResultOfMethodCallIgnored
                ForkJoinPool.getCommonPoolParallelism();
    }

    1 usage
    private static void setConfig(String x, int i) {
        configuration.put(x, i);
    }

    1 usage
    @SuppressWarnings("MismatchedQueryAndUpdateOfCollection")
    private static final Map<String, Integer> configuration = new HashMap<>();


}
```

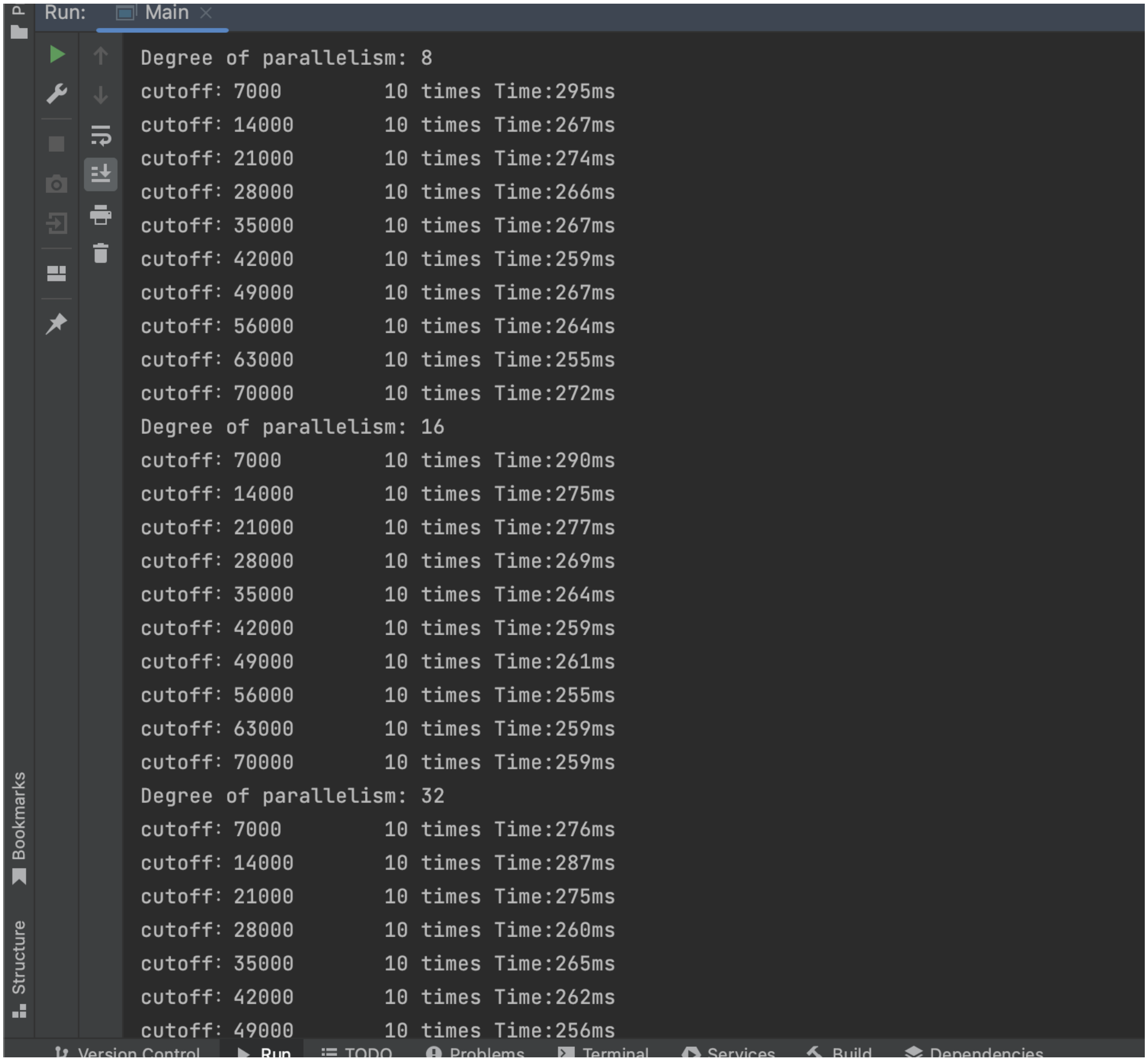
```java
class ParSort {

    public static int cutoff = 10000;

    public static void sort(int[] array, int from, int to, ForkJoinPool pool) {
        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) / 2, pool); // TO IMPLEMENT
            CompletableFuture<int[]> parsort2 = parsort(array, from + (to - from) / 2, to, pool); // TO IMPLEMENT
            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;
            });
```

```java

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

    2 usages
    private static CompletableFuture<int[]> parsort(int[] array, int from, int to, ForkJoinPool pool) {
        return CompletableFuture.supplyAsync(
                () -> {
                    int[] result = new int[to - from];
                    // TO IMPLEMENT
                    System.arraycopy(array, from, result, 0, result.length);
                    sort(result, 0, to - from,pool);
                    return result;
                }
        );
    }
}
```

**OUTPUT**:

1. **Cutoff 7000, Array Size: 700000**





```
Degree of parallelism: 8
cutoff: 7000        10 times Time:295ms
cutoff: 14000       10 times Time:267ms
cutoff: 21000       10 times Time:274ms
cutoff: 28000       10 times Time:266ms
cutoff: 35000       10 times Time:267ms
cutoff: 42000       10 times Time:259ms
cutoff: 49000       10 times Time:267ms
cutoff: 56000       10 times Time:264ms
cutoff: 63000       10 times Time:255ms
cutoff: 70000       10 times Time:272ms
Degree of parallelism: 16
cutoff: 7000        10 times Time:290ms
cutoff: 14000       10 times Time:275ms
cutoff: 21000       10 times Time:277ms
cutoff: 28000       10 times Time:269ms
cutoff: 35000       10 times Time:264ms
cutoff: 42000       10 times Time:259ms
cutoff: 49000       10 times Time:261ms
cutoff: 56000       10 times Time:255ms
cutoff: 63000       10 times Time:259ms
cutoff: 70000       10 times Time:259ms
Degree of parallelism: 32
cutoff: 7000        10 times Time:276ms
cutoff: 14000       10 times Time:287ms
cutoff: 21000       10 times Time:275ms
cutoff: 28000       10 times Time:260ms
cutoff: 35000       10 times Time:265ms
cutoff: 42000       10 times Time:262ms
cutoff: 49000       10 times Time:256ms
```

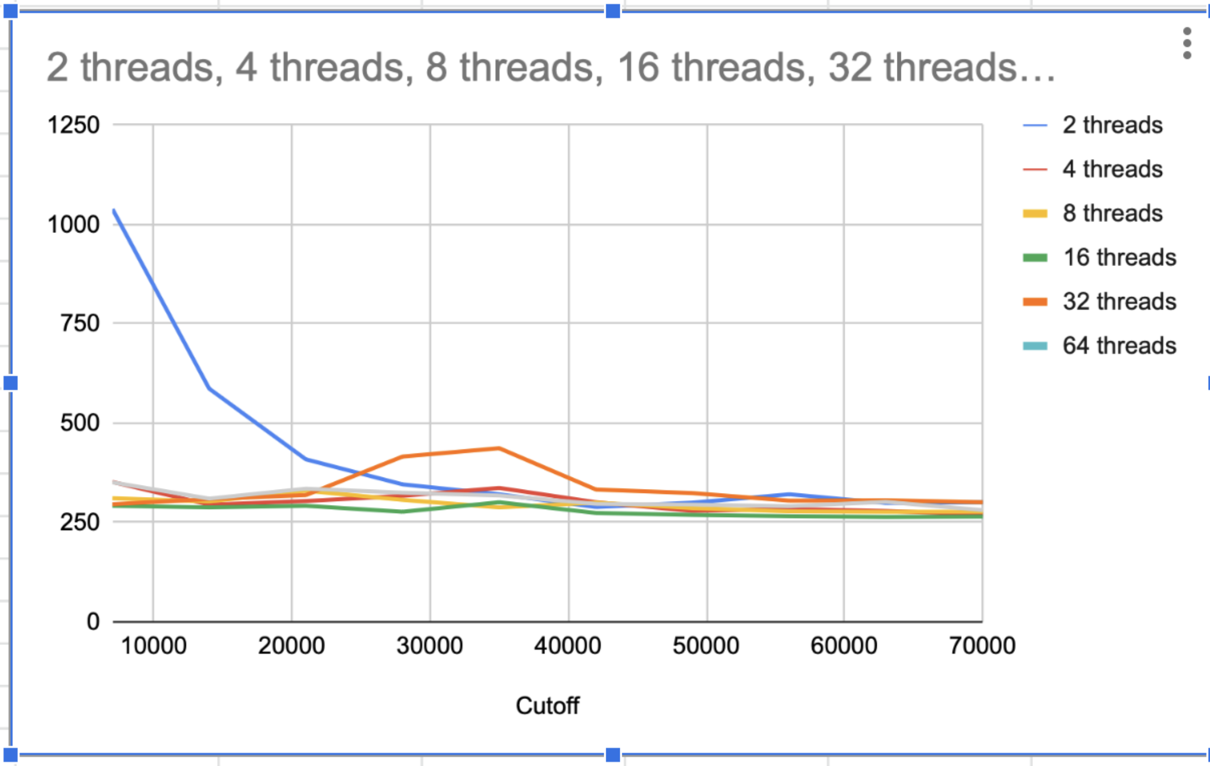
```
cutoff: 70000        10 times Time:259ms
Degree of parallelism: 32
cutoff: 7000         10 times Time:276ms
cutoff: 14000        10 times Time:287ms
cutoff: 21000        10 times Time:275ms
cutoff: 28000        10 times Time:260ms
cutoff: 35000        10 times Time:265ms
cutoff: 42000        10 times Time:262ms
cutoff: 49000        10 times Time:256ms
cutoff: 56000        10 times Time:260ms
cutoff: 63000        10 times Time:258ms
cutoff: 70000        10 times Time:259ms
Degree of parallelism: 64
cutoff: 7000         10 times Time:278ms
cutoff: 14000        10 times Time:264ms
cutoff: 21000        10 times Time:284ms
cutoff: 28000        10 times Time:266ms
cutoff: 35000        10 times Time:262ms
cutoff: 42000        10 times Time:273ms
cutoff: 49000        10 times Time:255ms
cutoff: 56000        10 times Time:258ms
cutoff: 63000        10 times Time:262ms
cutoff: 70000        10 times Time:261ms

Process finished with exit code 0
```

Bookmarks Structure

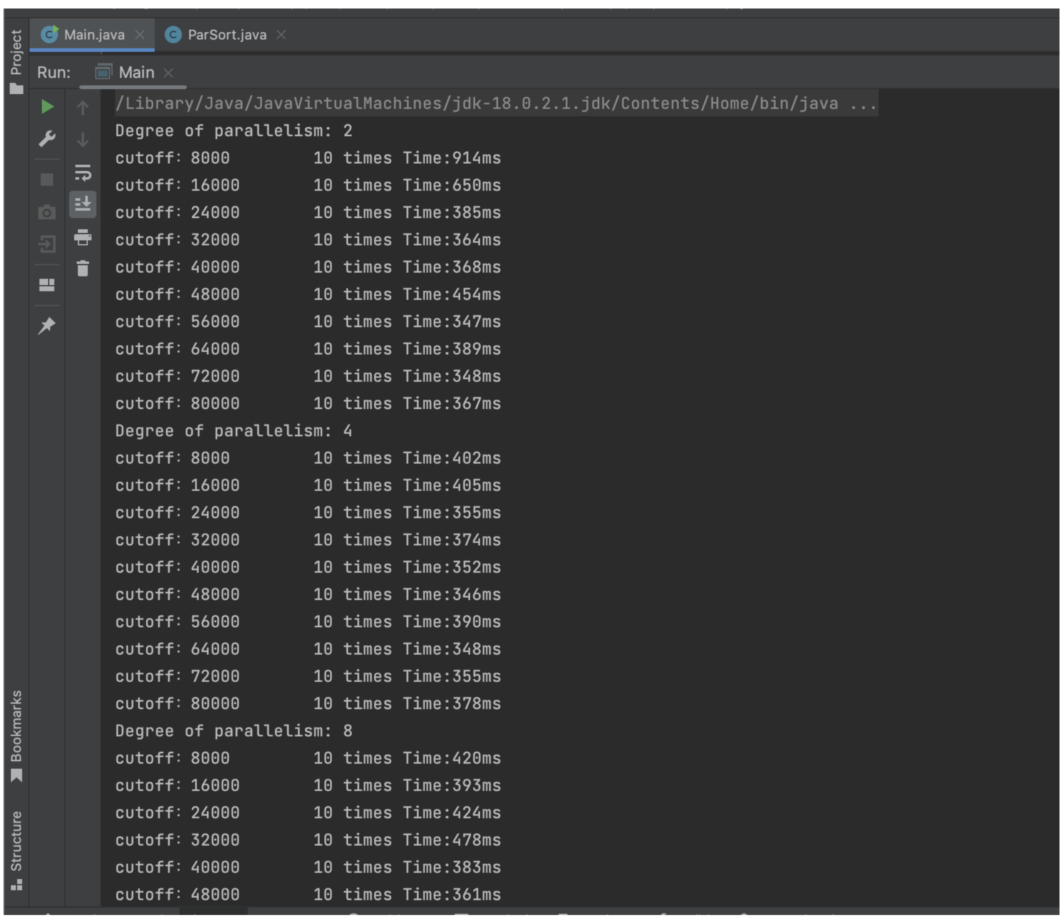
Version Control Run TODO Problems Terminal Services Build Depende

Build completed successfully in 3 sec, 351 ms (a minute ago)

| Cutoff | 2 threads | 4 threads | 8 threads | 16 threads | 32 threads | 64 threads |
|---|---|---|---|---|---|---|
| 7000 | 1039 | 352 | 311 | 292 | 296 | 351 |
| 14000 | 587 | 295 | 303 | 288 | 309 | 310 |
| 21000 | 409 | 304 | 330 | 292 | 319 | 335 |
| 28000 | 346 | 317 | 307 | 277 | 416 | 325 |
| 35000 | 321 | 337 | 288 | 301 | 437 | 318 |
| 42000 | 289 | 301 | 300 | 274 | 333 | 297 |
| 49000 | 300 | 279 | 286 | 269 | 324 | 295 |
| 56000 | 321 | 284 | 278 | 266 | 305 | 291 |
| 63000 | 299 | 279 | 277 | 264 | 306 | 302 |
| 70000 | 301 | 270 | 277 | 265 | 301 | 281 |




2. **Cutoff 8000, Array Size: 800000**




```
/Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...
Degree of parallelism: 2
cutoff: 8000        10 times Time:914ms
cutoff: 16000       10 times Time:650ms
cutoff: 24000       10 times Time:385ms
cutoff: 32000       10 times Time:364ms
cutoff: 40000       10 times Time:368ms
cutoff: 48000       10 times Time:454ms
cutoff: 56000       10 times Time:347ms
cutoff: 64000       10 times Time:389ms
cutoff: 72000       10 times Time:348ms
cutoff: 80000       10 times Time:367ms
Degree of parallelism: 4
cutoff: 8000        10 times Time:402ms
cutoff: 16000       10 times Time:405ms
cutoff: 24000       10 times Time:355ms
cutoff: 32000       10 times Time:374ms
cutoff: 40000       10 times Time:352ms
cutoff: 48000       10 times Time:346ms
cutoff: 56000       10 times Time:390ms
cutoff: 64000       10 times Time:348ms
cutoff: 72000       10 times Time:355ms
cutoff: 80000       10 times Time:378ms
Degree of parallelism: 8
cutoff: 8000        10 times Time:420ms
cutoff: 16000       10 times Time:393ms
cutoff: 24000       10 times Time:424ms
cutoff: 32000       10 times Time:478ms
cutoff: 40000       10 times Time:383ms
cutoff: 48000       10 times Time:361ms
```

```
cutoff: 48000        10 times Time:361ms
cutoff: 56000        10 times Time:341ms
cutoff: 64000        10 times Time:368ms
cutoff: 72000        10 times Time:350ms
cutoff: 80000        10 times Time:337ms
Degree of parallelism: 16
cutoff: 8000         10 times Time:370ms
cutoff: 16000        10 times Time:376ms
cutoff: 24000        10 times Time:376ms
cutoff: 32000        10 times Time:357ms
cutoff: 40000        10 times Time:360ms
cutoff: 48000        10 times Time:358ms
cutoff: 56000        10 times Time:353ms
cutoff: 64000        10 times Time:356ms
cutoff: 72000        10 times Time:349ms
cutoff: 80000        10 times Time:359ms
Degree of parallelism: 32
cutoff: 8000         10 times Time:385ms
cutoff: 16000        10 times Time:379ms
cutoff: 24000        10 times Time:369ms
cutoff: 32000        10 times Time:370ms
cutoff: 40000        10 times Time:357ms
cutoff: 48000        10 times Time:381ms
cutoff: 56000        10 times Time:344ms
cutoff: 64000        10 times Time:370ms
cutoff: 72000        10 times Time:351ms
cutoff: 80000        10 times Time:341ms
Degree of parallelism: 64
cutoff: 8000         10 times Time:394ms
cutoff: 16000        10 times Time:395ms
```

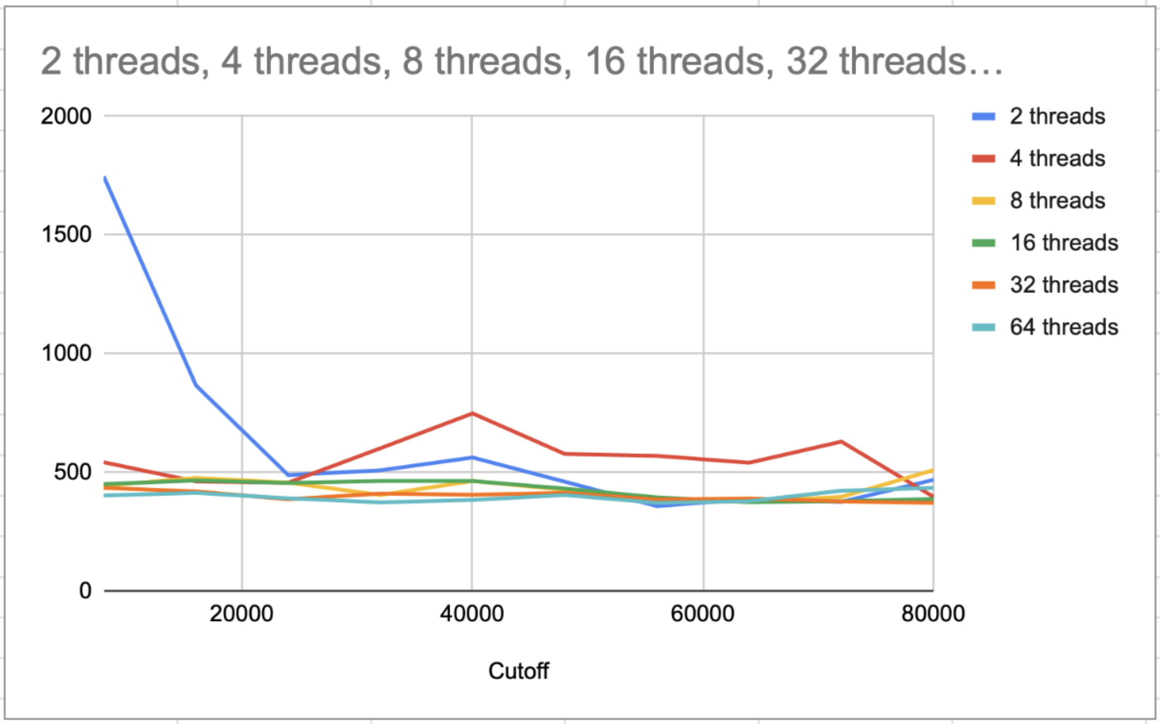
Version Control | Run | TODO | Problems | Terminal | Services | Build | Dependencies

```
cutoff: 80000        10 times Time:341ms
Degree of parallelism: 64
cutoff: 8000         10 times Time:394ms
cutoff: 16000        10 times Time:395ms
cutoff: 24000        10 times Time:382ms
cutoff: 32000        10 times Time:383ms
cutoff: 40000        10 times Time:376ms
cutoff: 48000        10 times Time:374ms
cutoff: 56000        10 times Time:358ms
cutoff: 64000        10 times Time:357ms
cutoff: 72000        10 times Time:353ms
cutoff: 80000        10 times Time:347ms

Process finished with exit code 0
```

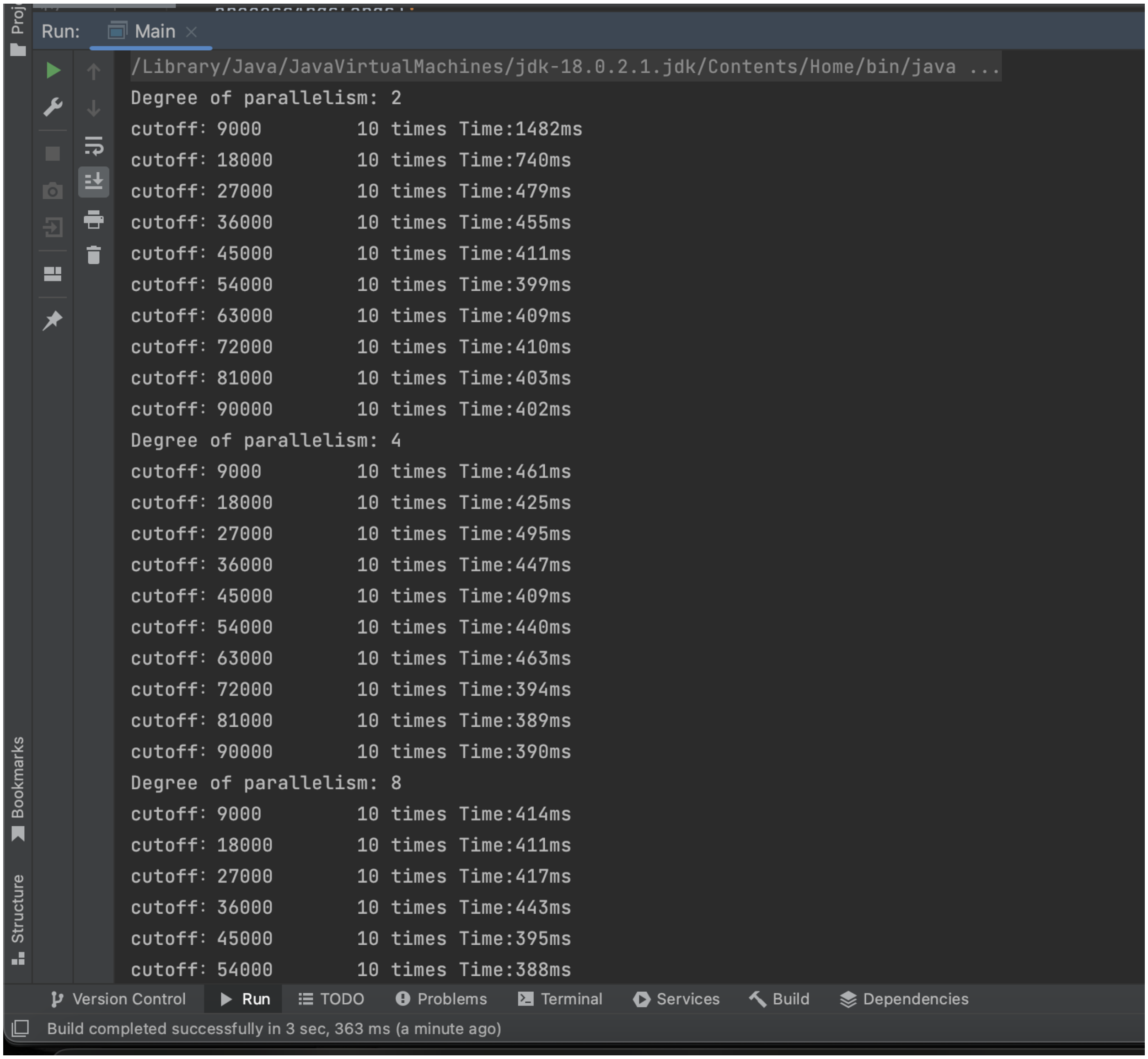
Version Control | Run | TODO | Problems | Terminal | Services | Build | Dependencies

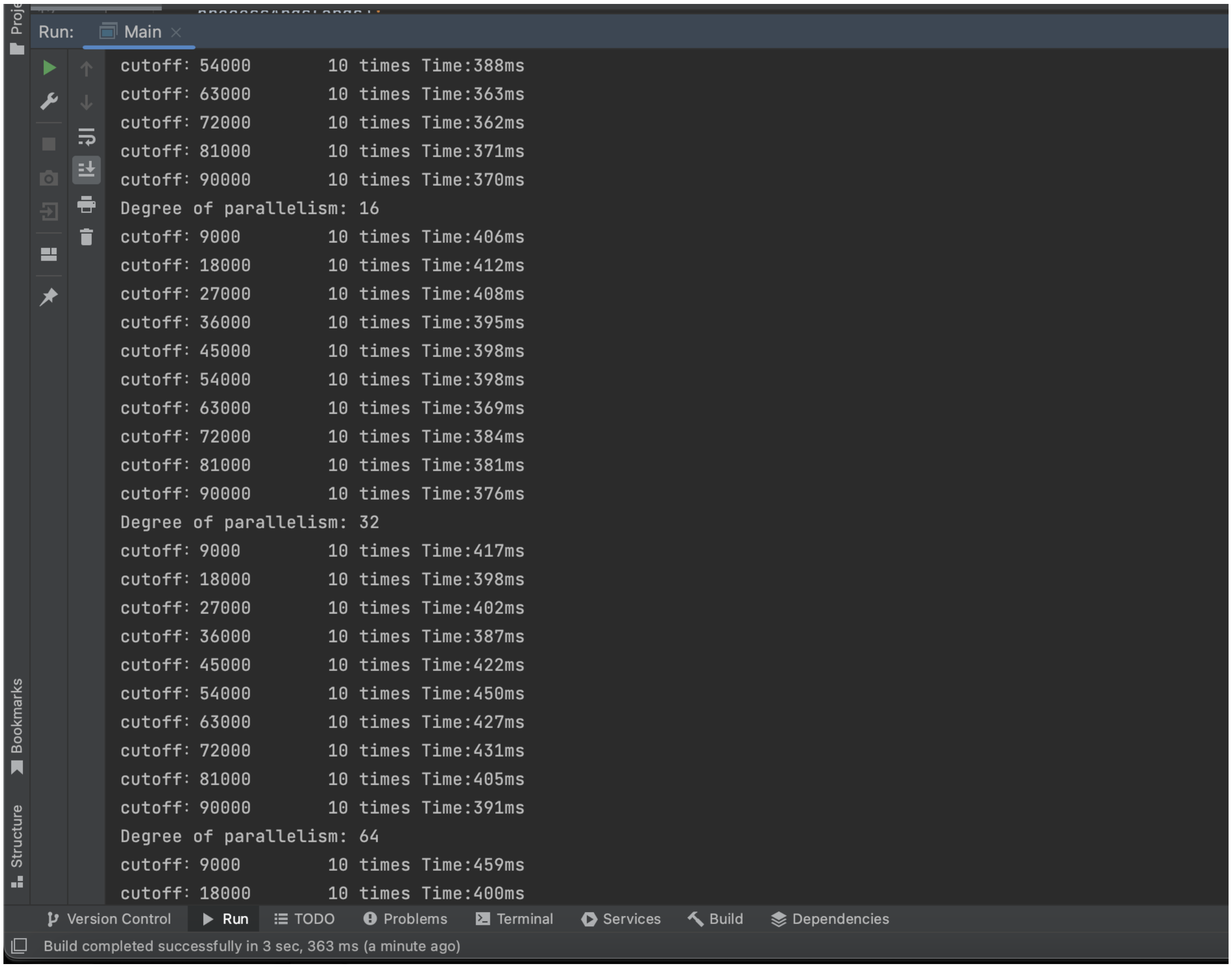
Build completed successfully in 3 sec, 477 ms (2 minutes ago)

| Cutoff | 2 threads | 4 threads | 8 threads | 16 threads | 32 threads | 64 threads |
|---|---|---|---|---|---|---|
| 8000 | 1747 | 542 | 443 | 450 | 434 | 402 |
| 16000 | 866 | 461 | 476 | 465 | 419 | 413 |
| 24000 | 488 | 456 | 456 | 455 | 386 | 390 |
| 32000 | 508 | 601 | 404 | 463 | 410 | 373 |
| 40000 | 562 | 748 | 463 | 463 | 405 | 383 |
| 48000 | 460 | 577 | 426 | 431 | 413 | 404 |
| 56000 | 357 | 569 | 378 | 394 | 385 | 371 |
| 64000 | 387 | 540 | 376 | 374 | 389 | 378 |
| 72000 | 375 | 629 | 396 | 378 | 377 | 422 |
| 80000 | 468 | 396 | 509 | 387 | 371 | 434 |



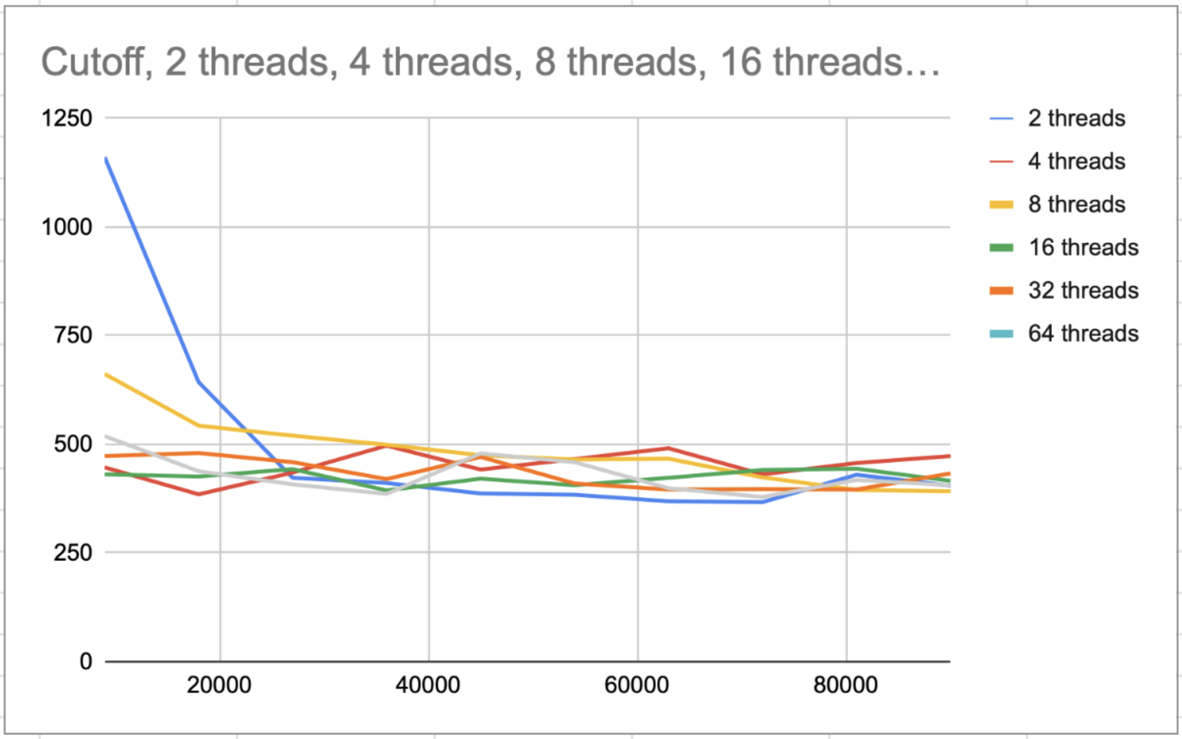
3. **Cutoff 9000, Array Size: 900000**




Run:
Main
cutoff: 54000 10 times Time:388ms
cutoff: 63000 10 times Time:363ms
cutoff: 72000 10 times Time:362ms
cutoff: 81000 10 times Time:371ms
cutoff: 90000 10 times Time:370ms
Degree of parallelism: 16
cutoff: 9000 10 times Time:406ms
cutoff: 18000 10 times Time:412ms
cutoff: 27000 10 times Time:408ms
cutoff: 36000 10 times Time:395ms
cutoff: 45000 10 times Time:398ms
cutoff: 54000 10 times Time:398ms
cutoff: 63000 10 times Time:369ms
cutoff: 72000 10 times Time:384ms
cutoff: 81000 10 times Time:381ms
cutoff: 90000 10 times Time:376ms
Degree of parallelism: 32
cutoff: 9000 10 times Time:417ms
cutoff: 18000 10 times Time:398ms
cutoff: 27000 10 times Time:402ms
cutoff: 36000 10 times Time:387ms
cutoff: 45000 10 times Time:422ms
cutoff: 54000 10 times Time:450ms
cutoff: 63000 10 times Time:427ms
cutoff: 72000 10 times Time:431ms
cutoff: 81000 10 times Time:405ms
cutoff: 90000 10 times Time:391ms
Degree of parallelism: 64
cutoff: 9000 10 times Time:459ms
cutoff: 18000 10 times Time:400ms
Bookmarks
Structure
Version Control
Run
TODO
Problems
Terminal
Services
Build
Dependencies
Build completed successfully in 3 sec, 363 ms (a minute ago)

```
cutoff: 90000       10 times Time:376ms
Degree of parallelism: 32
cutoff: 9000        10 times Time:417ms
cutoff: 18000       10 times Time:398ms
cutoff: 27000       10 times Time:402ms
cutoff: 36000       10 times Time:387ms
cutoff: 45000       10 times Time:422ms
cutoff: 54000       10 times Time:450ms
cutoff: 63000       10 times Time:427ms
cutoff: 72000       10 times Time:431ms
cutoff: 81000       10 times Time:405ms
cutoff: 90000       10 times Time:391ms
Degree of parallelism: 64
cutoff: 9000        10 times Time:459ms
cutoff: 18000       10 times Time:400ms
cutoff: 27000       10 times Time:368ms
cutoff: 36000       10 times Time:372ms
cutoff: 45000       10 times Time:382ms
cutoff: 54000       10 times Time:382ms
cutoff: 63000       10 times Time:370ms
cutoff: 72000       10 times Time:366ms
cutoff: 81000       10 times Time:377ms
cutoff: 90000       10 times Time:366ms

Process finished with exit code 0
```

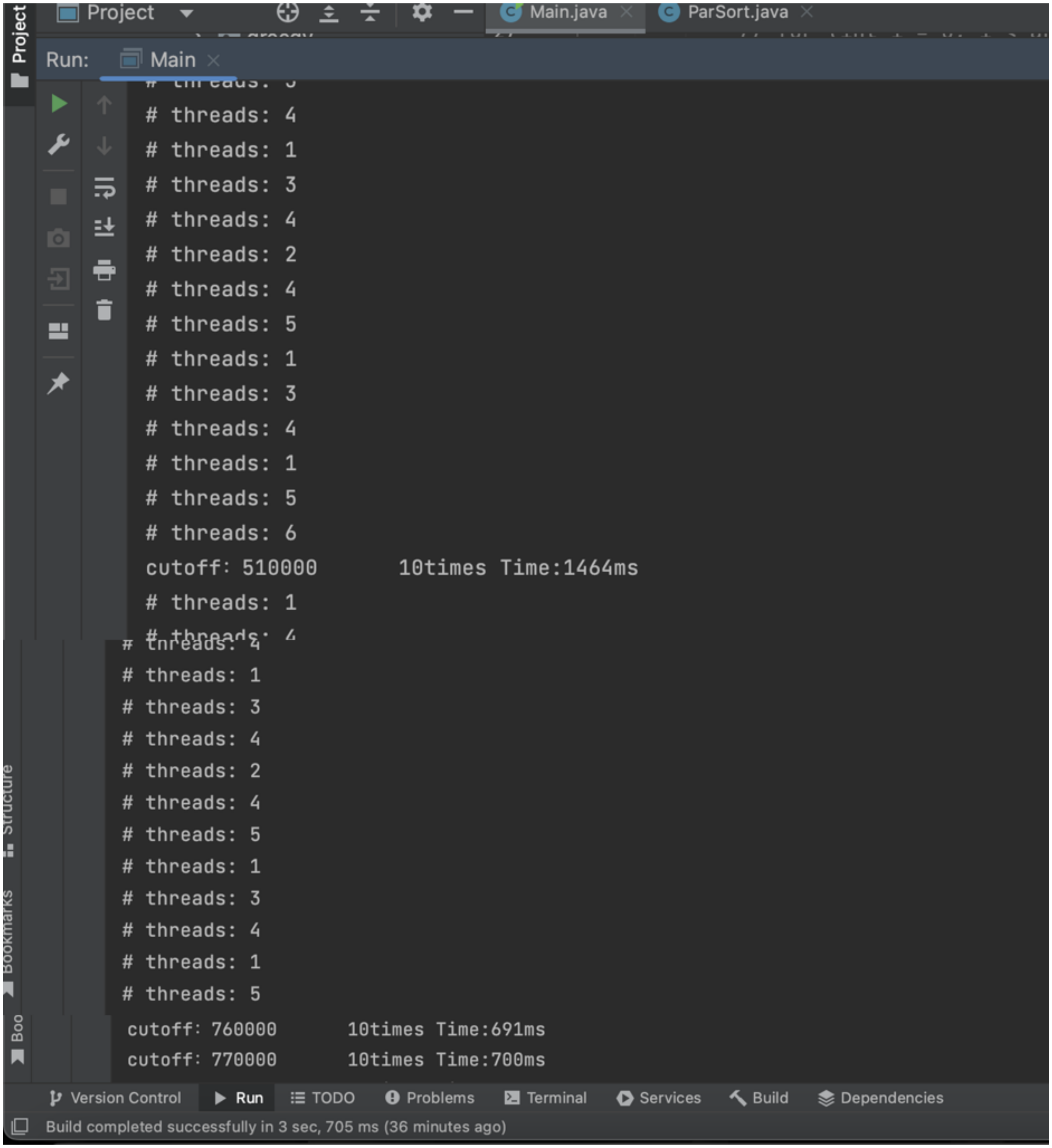
| Cutoff | 2 threads | 4 threads | 8 threads | 16 threads | 32 threads | 64 threads |
|---|---|---|---|---|---|---|
| 9000 | 1161 | 447 | 662 | 431 | 473 | 519 |
| 18000 | 643 | 385 | 543 | 426 | 480 | 438 |
| 27000 | 423 | 435 | 520 | 443 | 459 | 408 |
| 36000 | 411 | 497 | 499 | 394 | 420 | 386 |
| 45000 | 387 | 442 | 475 | 421 | 471 | 479 |
| 54000 | 384 | 466 | 465 | 406 | 410 | 459 |
| 63000 | 369 | 491 | 467 | 423 | 396 | 399 |
| 72000 | 367 | 431 | 424 | 441 | 397 | 379 |
| 81000 | 430 | 457 | 395 | 444 | 396 | 418 |
| 90000 | 405 | 473 | 392 | 416 | 433 | 406 |



**OUTPUT WITH THREADS (SCREENSHOTS):**

Run: Main

```
Degree of parallelism: 7
# threads: 6
# threads: 6
# threads: 6
# threads: 1
# threads: 4
# threads: 6
# threads: 1
# threads: 3
# threads: 4
# threads: 1
# threads: 2
# threads: 2
# threads: 1
# threads: 3
# threads: 4
# threads: 1
# threads: 3
# threads: 4
# threads: 1
# threads: 3
# threads: 4
# threads: 2
# threads: 4
# threads: 5
# threads: 1
# threads: 3
# threads: 4
# threads: 1
# threads: 5
```


Project
Main.java
ParSort.java
Run:
Main
# threads: 4
# threads: 1
# threads: 3
# threads: 4
# threads: 2
# threads: 4
# threads: 5
# threads: 1
# threads: 3
# threads: 4
# threads: 1
# threads: 5
# threads: 6
cutoff: 510000 10times Time:1464ms
# threads: 1
# threads: 4
# threads: 1
# threads: 3
# threads: 4
# threads: 2
# threads: 4
# threads: 5
# threads: 1
# threads: 3
# threads: 4
# threads: 1
# threads: 5
cutoff: 760000 10times Time:691ms
cutoff: 770000 10times Time:700ms
Version Control
Run
TODO
Problems
Terminal
Services
Build
Dependencies
Build completed successfully in 3 sec, 705 ms (36 minutes ago)
Structure
Bookmarks

**CONCLUSION**:

The parallel sort algorithm divides the dataset recursively into smaller partitions until the partition size falls below the specified cutoff value. At this point, each partition is sorted in parallel using its own thread or processing element. The sorted partitions are then d back together in parallel, using a merge operation.
Overall, the efficiency of the parallel sort algorithm will depend on various factors, such as the specific details of the implementation. However, it generally offers a significant speedup over sequential sort for large data sets on systems with many processing elements. The choice of the cutoff value will impact the algorithm's performance, as higher values may result in better performance for distributed-memory systems with higher communication overhead.