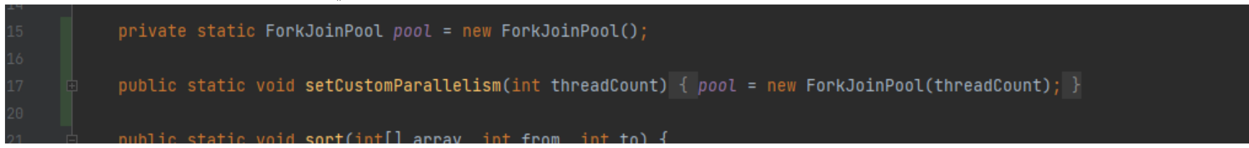
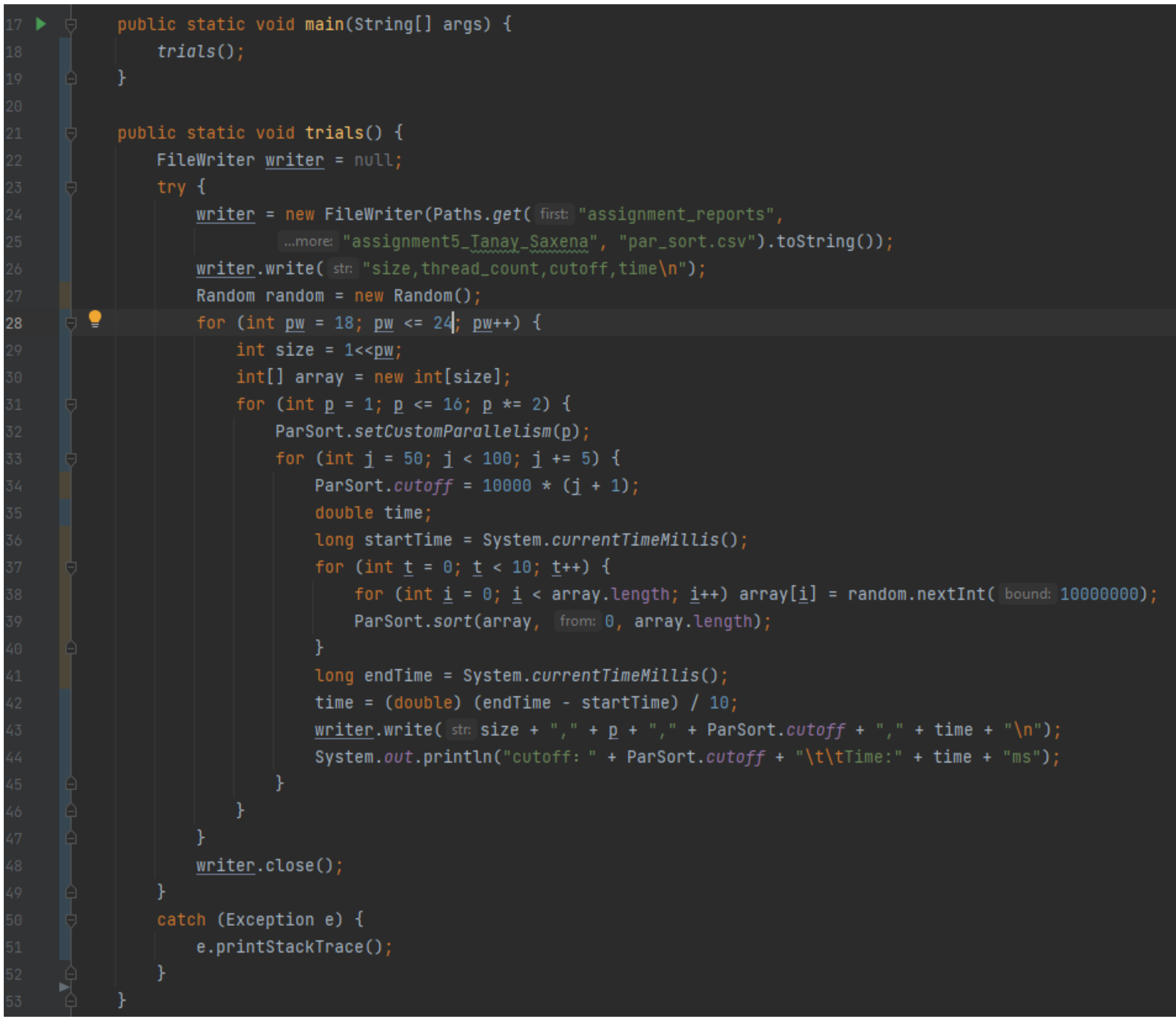
Tanay Saxena (001586302)
Program Structures & Algorithms
Fall 2021
Assignment No. 5

- Task (List down the tasks performed in the Assignment)
    - Added code to enable control of parallelism -
      Class UF_HWQUPC.java
      setCustomParallelism() -



```java
    private static ForkJoinPool pool = new ForkJoinPool();

    public static void setCustomParallelism(int threadCount) { pool = new ForkJoinPool(threadCount); }

    public static void sort(int[] array, int from, int to) {
```

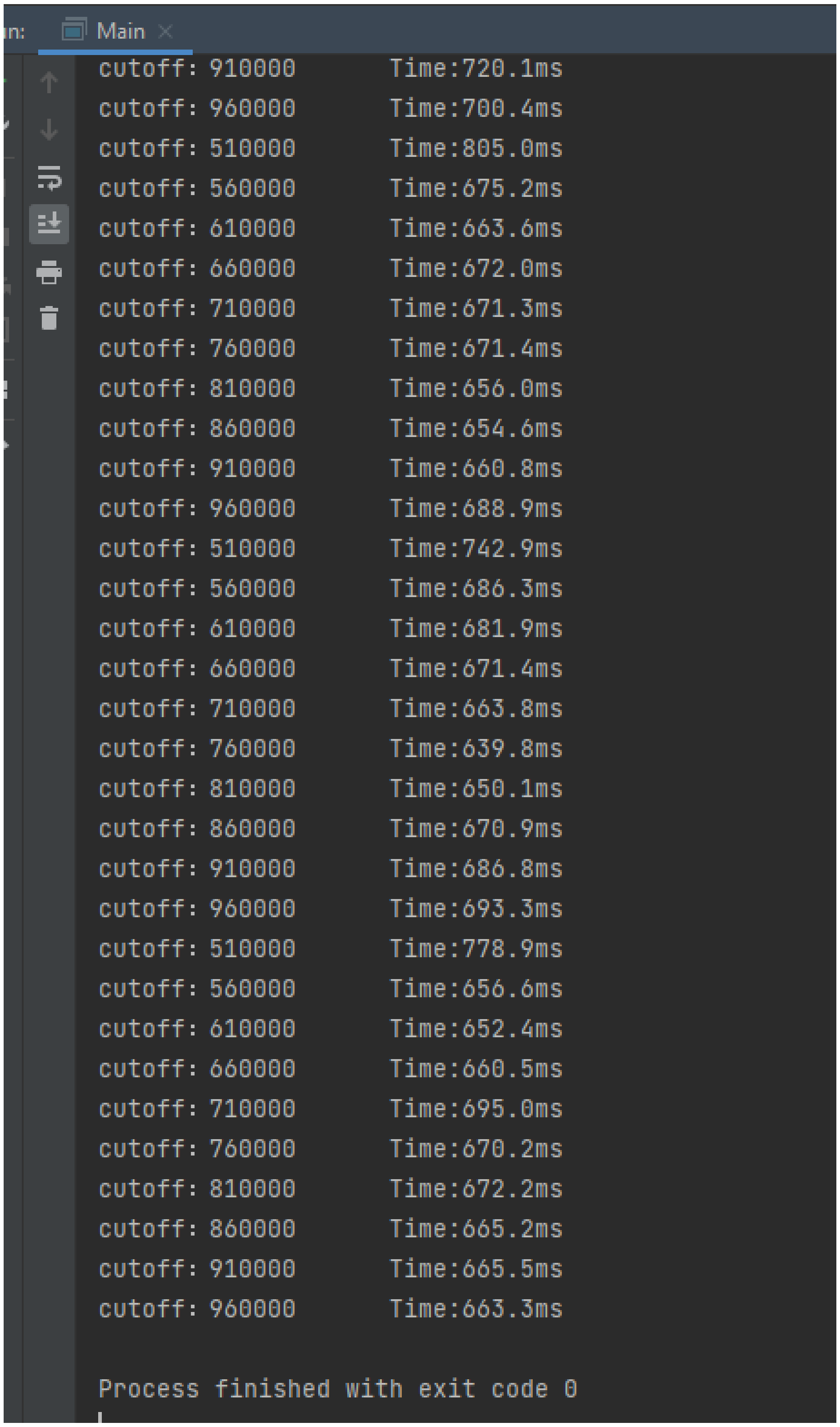
○ Created a method “trials()” to perform the experiments and write results to a CSV file, and called the method it in main method-



```java
    public static void main(String[] args) {
        trials();
    }

    public static void trials() {
        FileWriter writer = null;
        try {
            writer = new FileWriter(Paths.get("assignment_reports",
                    "assignment5_Tanay_Saxena", "par_sort.csv").toString());
            writer.write("size,thread_count,cutoff,time\n");
            Random random = new Random();
            for (int pw = 18; pw <= 24; pw++) {
                int size = 1<<pw;
                int[] array = new int[size];
                for (int p = 1; p <= 16; p *= 2) {
                    ParSort.setCustomParallelism(p);
                    for (int j = 50; j < 100; j += 5) {
                        ParSort.cutoff = 10000 * (j + 1);
                        double 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);
                        }
                        long endTime = System.currentTimeMillis();
                        time = (double) (endTime - startTime) / 10;
                        writer.write(size + "," + p + "," + ParSort.cutoff + "," + time + "\n");
                        System.out.println("cutoff: " + ParSort.cutoff + "\t\tTime:" + time + "ms");
                    }
                }
            }
            writer.close();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
```

○ Generated the output by executing the main method -

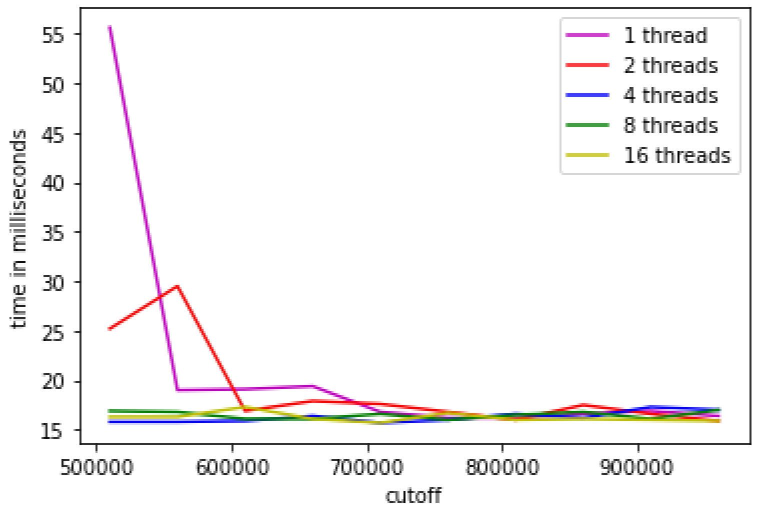
…..



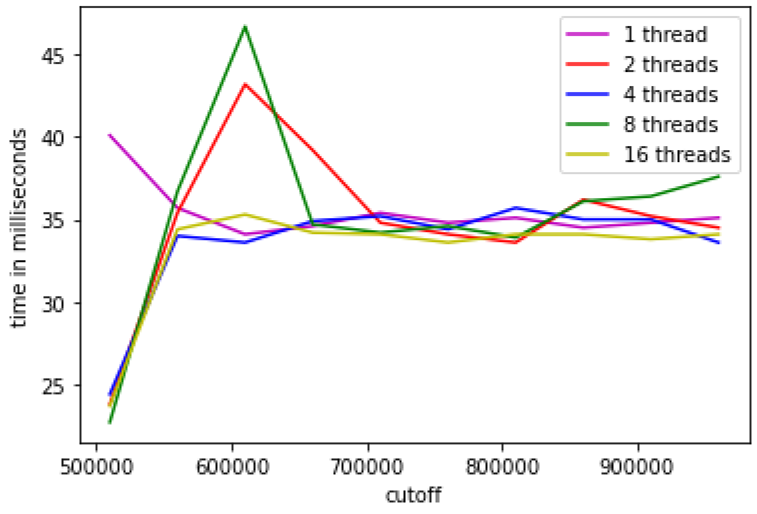
```
cutoff: 910000      Time:720.1ms
cutoff: 960000      Time:700.4ms
cutoff: 510000      Time:805.0ms
cutoff: 560000      Time:675.2ms
cutoff: 610000      Time:663.6ms
cutoff: 660000      Time:672.0ms
cutoff: 710000      Time:671.3ms
cutoff: 760000      Time:671.4ms
cutoff: 810000      Time:656.0ms
cutoff: 860000      Time:654.6ms
cutoff: 910000      Time:660.8ms
cutoff: 960000      Time:688.9ms
cutoff: 510000      Time:742.9ms
cutoff: 560000      Time:686.3ms
cutoff: 610000      Time:681.9ms
cutoff: 660000      Time:671.4ms
cutoff: 710000      Time:663.8ms
cutoff: 760000      Time:639.8ms
cutoff: 810000      Time:650.1ms
cutoff: 860000      Time:670.9ms
cutoff: 910000      Time:686.8ms
cutoff: 960000      Time:693.3ms
cutoff: 510000      Time:778.9ms
cutoff: 560000      Time:656.6ms
cutoff: 610000      Time:652.4ms
cutoff: 660000      Time:660.5ms
cutoff: 710000      Time:695.0ms
cutoff: 760000      Time:670.2ms
cutoff: 810000      Time:672.2ms
cutoff: 860000      Time:665.2ms
cutoff: 910000      Time:665.5ms
cutoff: 960000      Time:663.3ms

Process finished with exit code 0
```

- Created a jupyter notebook and performed analysis on the output (csv format). Each graph below is for a specific array size as mentioned, the thread count is mentioned in the legend of each graph, time in ms and cutoff value chosen for the experiments have been represented by the y-axis and x-axis respectively.

```
Array Size: 262144
```

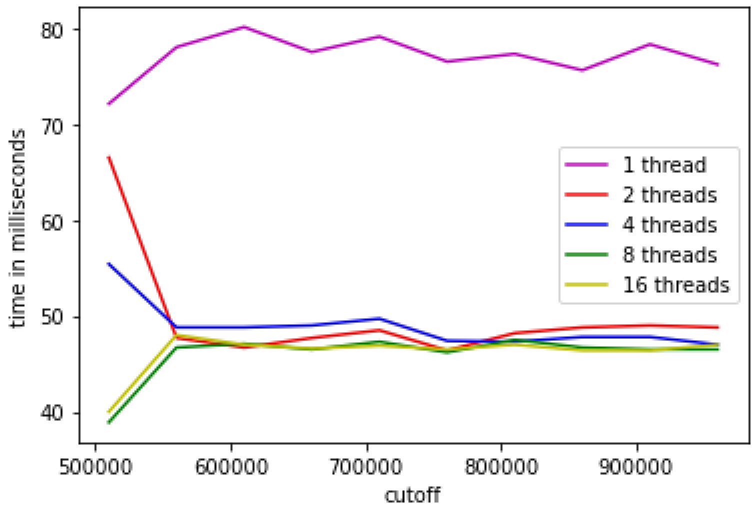


```
Array Size: 524288
```

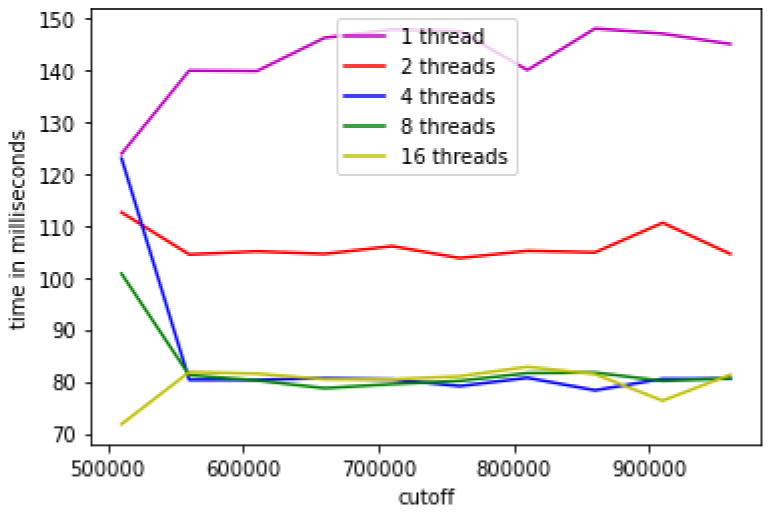


○

```
Array Size: 1048576
```

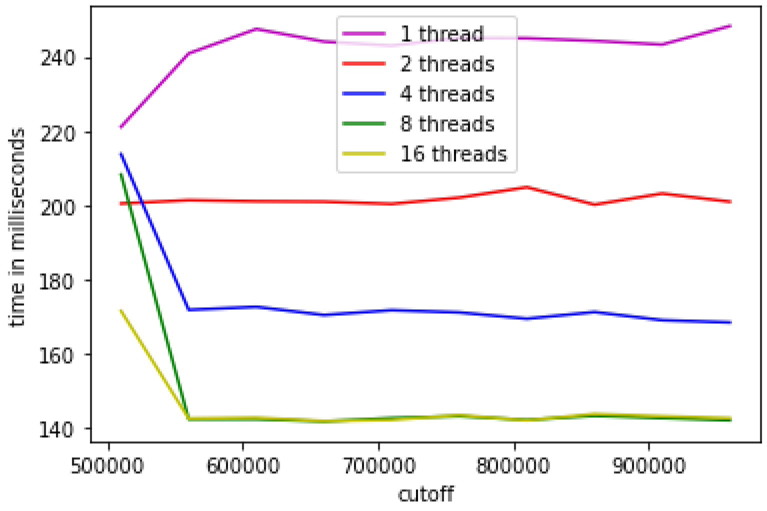


```
Array Size: 2097152
```

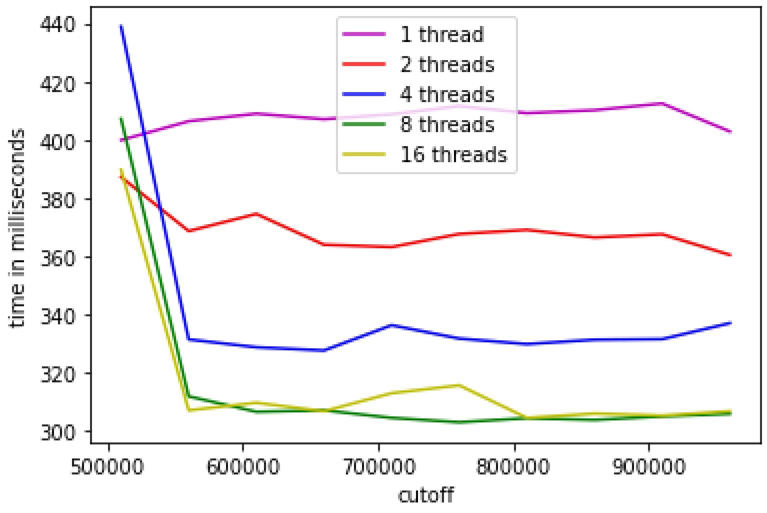


○

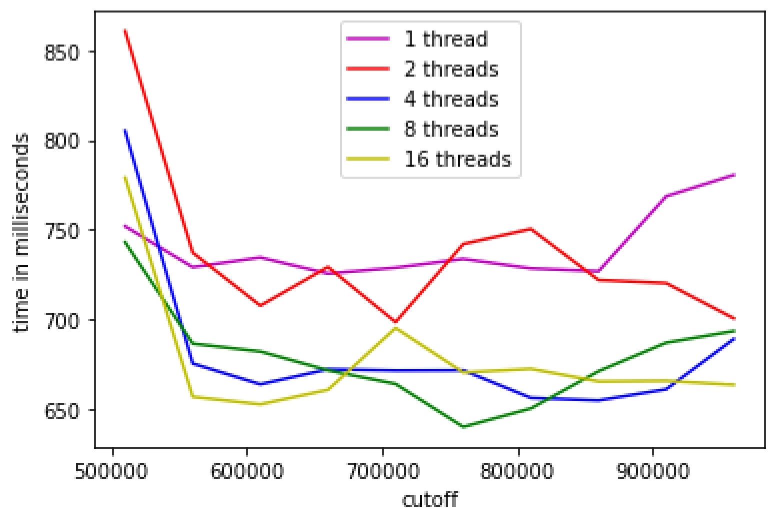
```
Array Size: 4194304
```



```
Array Size: 8388608
```



```
Array Size: 16777216
```

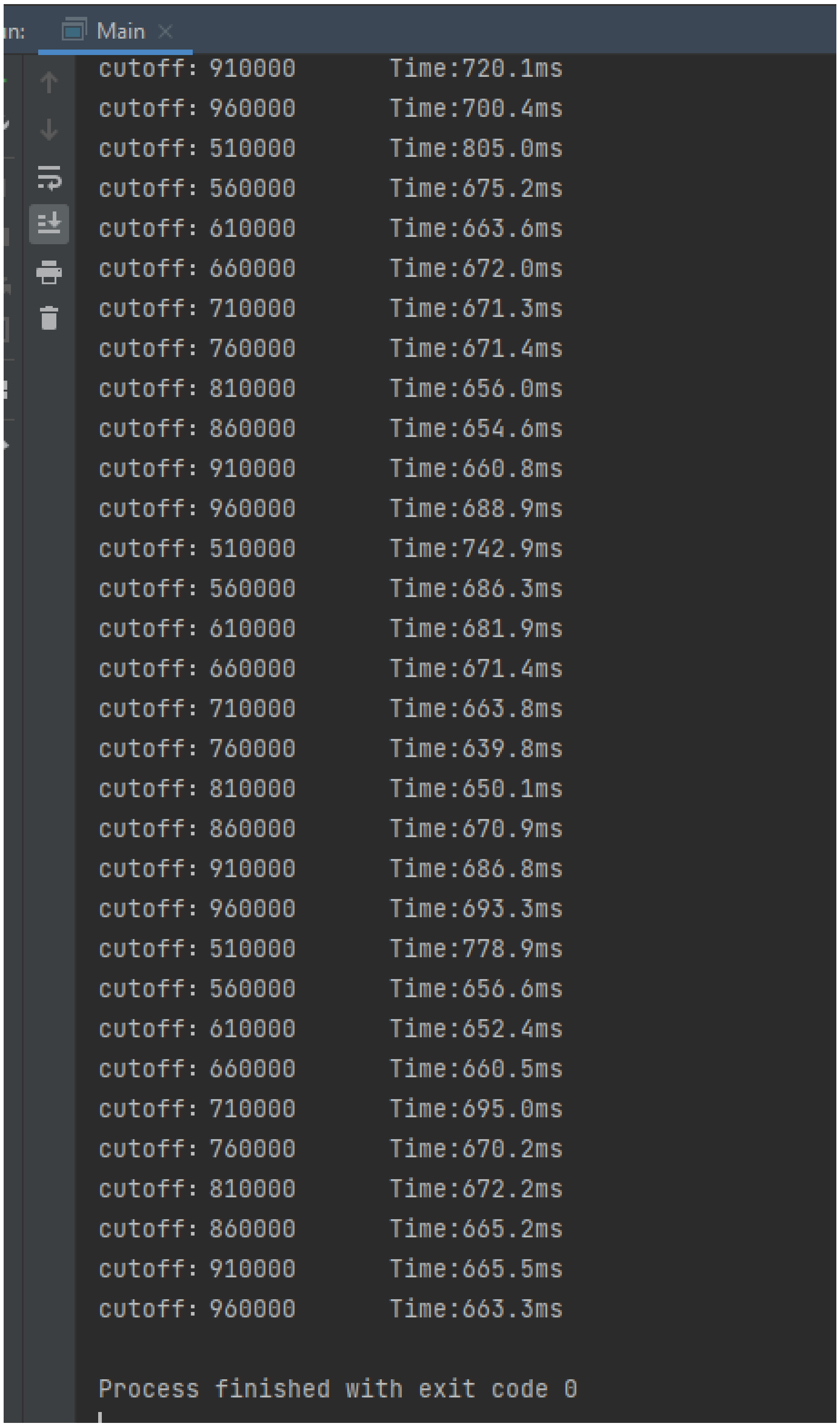


- ○

- Conclusion:

  The CPU used for this experiment has **4 cores running at 4.1 GHz, 8 system threads via intel's hyperthreading** (Intel core i5 8300h).
  - As we can see for smaller array sizes, and cutoff values> 600K, all the different thread count configurations work similarly.
  - Additionally, for all the input sizes, the single-threaded configuration performs worse (represented by magenta color) as compared to other configs. , for obvious reasons, it works close to the regular merge sort algorithm.
  - Also, note that as we increase the number of threads (8, 16) for the experiment, the performance gain diminishes, this could be caused by only 8 system threads available to the processor.
  - Hence, in this case, the optimal number should be 8 user threads, one for each system thread.
  - Also, for higher sizes of the input array, all the cutoff > 600K work almost the same across thread configurations
  - Hence, the optimal algorithm should use **threads equal to system thread count and cutoff should be in the range 600K-900K**.

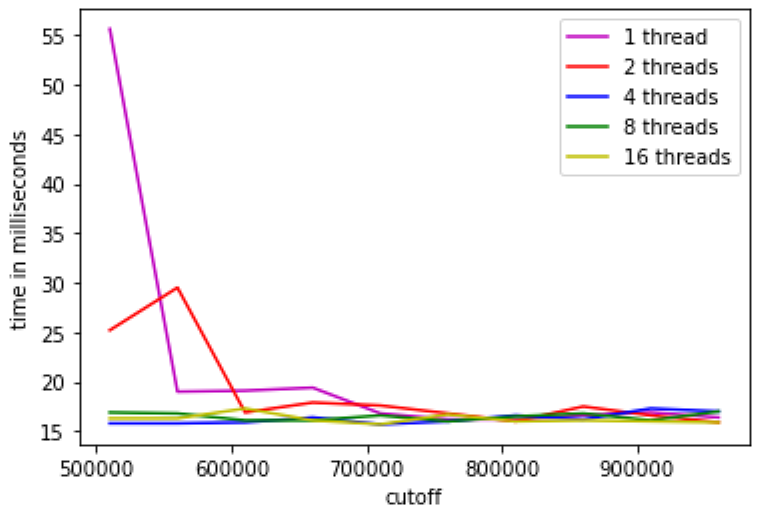
- Evidence to support the conclusion:
  1. Output (Snapshot of Code output in the terminal)



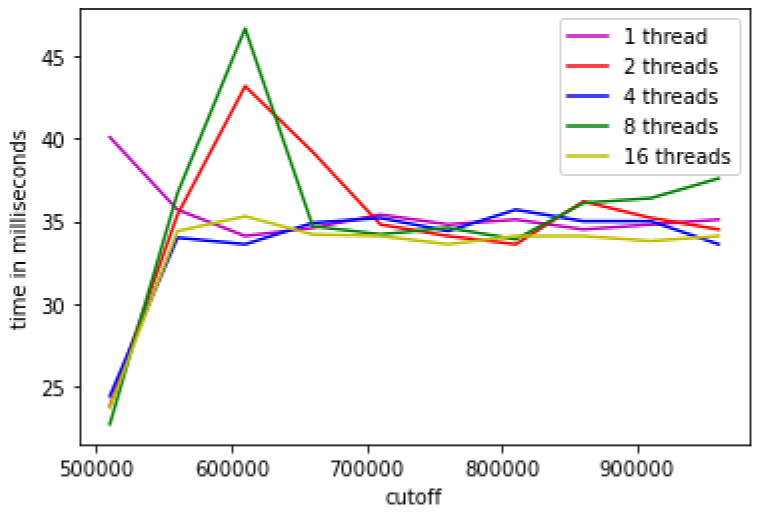
```
cutoff: 910000       Time:720.1ms
cutoff: 960000       Time:700.4ms
cutoff: 510000       Time:805.0ms
cutoff: 560000       Time:675.2ms
cutoff: 610000       Time:663.6ms
cutoff: 660000       Time:672.0ms
cutoff: 710000       Time:671.3ms
cutoff: 760000       Time:671.4ms
cutoff: 810000       Time:656.0ms
cutoff: 860000       Time:654.6ms
cutoff: 910000       Time:660.8ms
cutoff: 960000       Time:688.9ms
cutoff: 510000       Time:742.9ms
cutoff: 560000       Time:686.3ms
cutoff: 610000       Time:681.9ms
cutoff: 660000       Time:671.4ms
cutoff: 710000       Time:663.8ms
cutoff: 760000       Time:639.8ms
cutoff: 810000       Time:650.1ms
cutoff: 860000       Time:670.9ms
cutoff: 910000       Time:686.8ms
cutoff: 960000       Time:693.3ms
cutoff: 510000       Time:778.9ms
cutoff: 560000       Time:656.6ms
cutoff: 610000       Time:652.4ms
cutoff: 660000       Time:660.5ms
cutoff: 710000       Time:695.0ms
cutoff: 760000       Time:670.2ms
cutoff: 810000       Time:672.2ms
cutoff: 860000       Time:665.2ms
cutoff: 910000       Time:665.5ms
cutoff: 960000       Time:663.3ms

Process finished with exit code 0
```

2. Graphical Representation(Observations)

```
Array Size: 262144
```

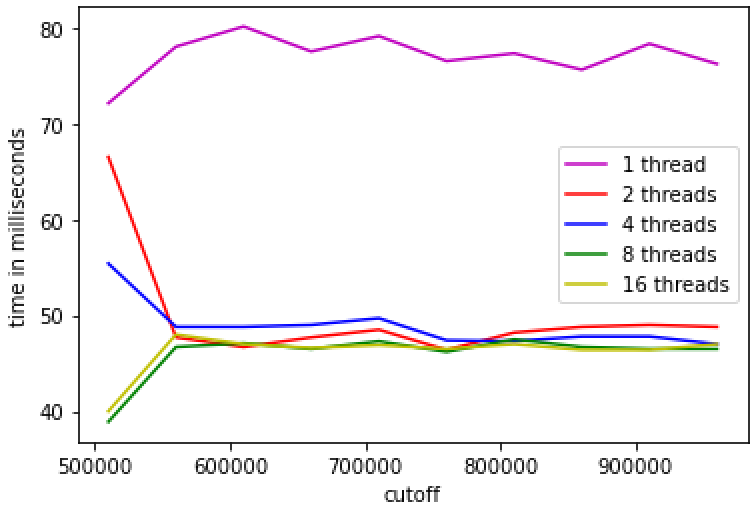


```
Array Size: 524288
```

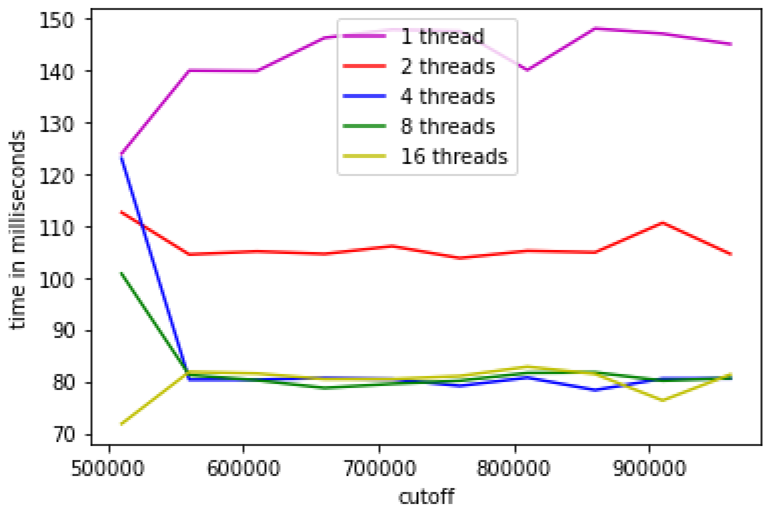


○

```
Array Size: 1048576
```

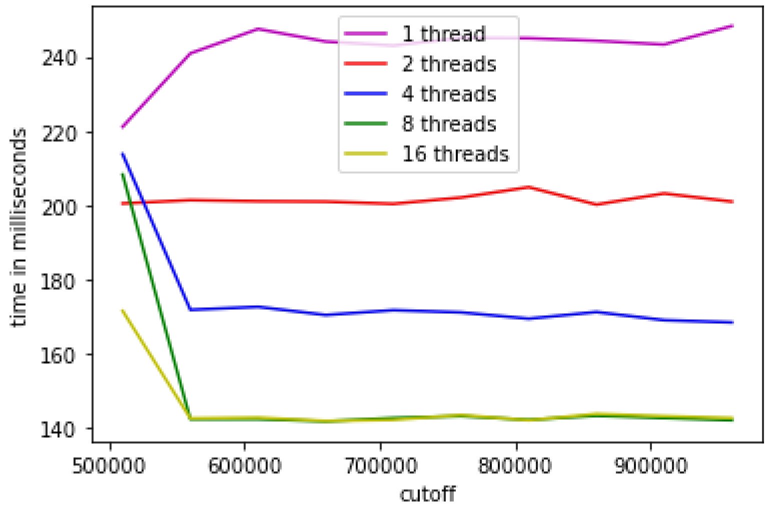


```
Array Size: 2097152
```

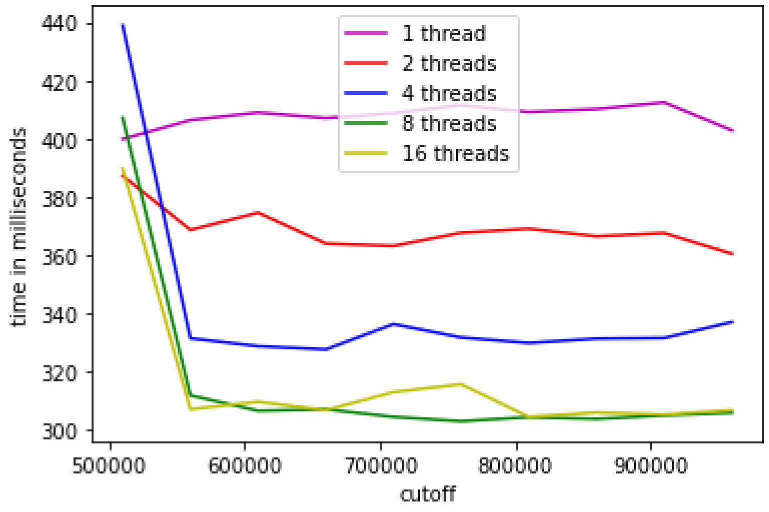


○

```
Array Size: 4194304
```

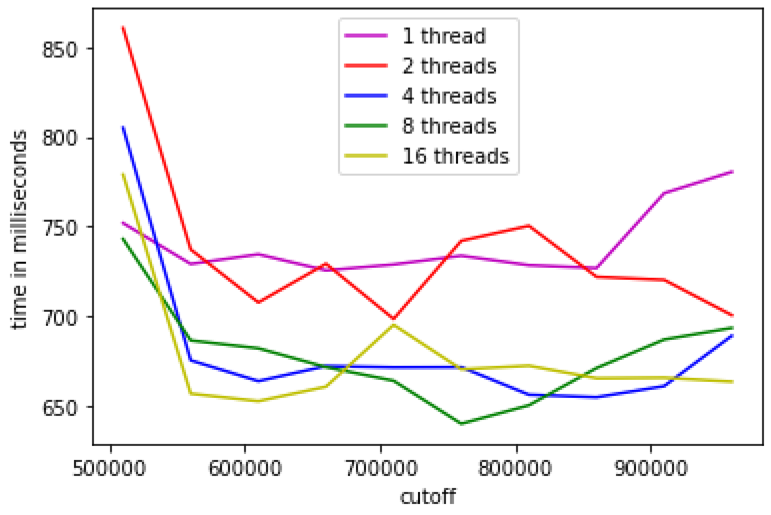


```
Array Size: 8388608
```



○

```
Array Size: 16777216
```



○