Program Structures & Algorithms Spring 2022

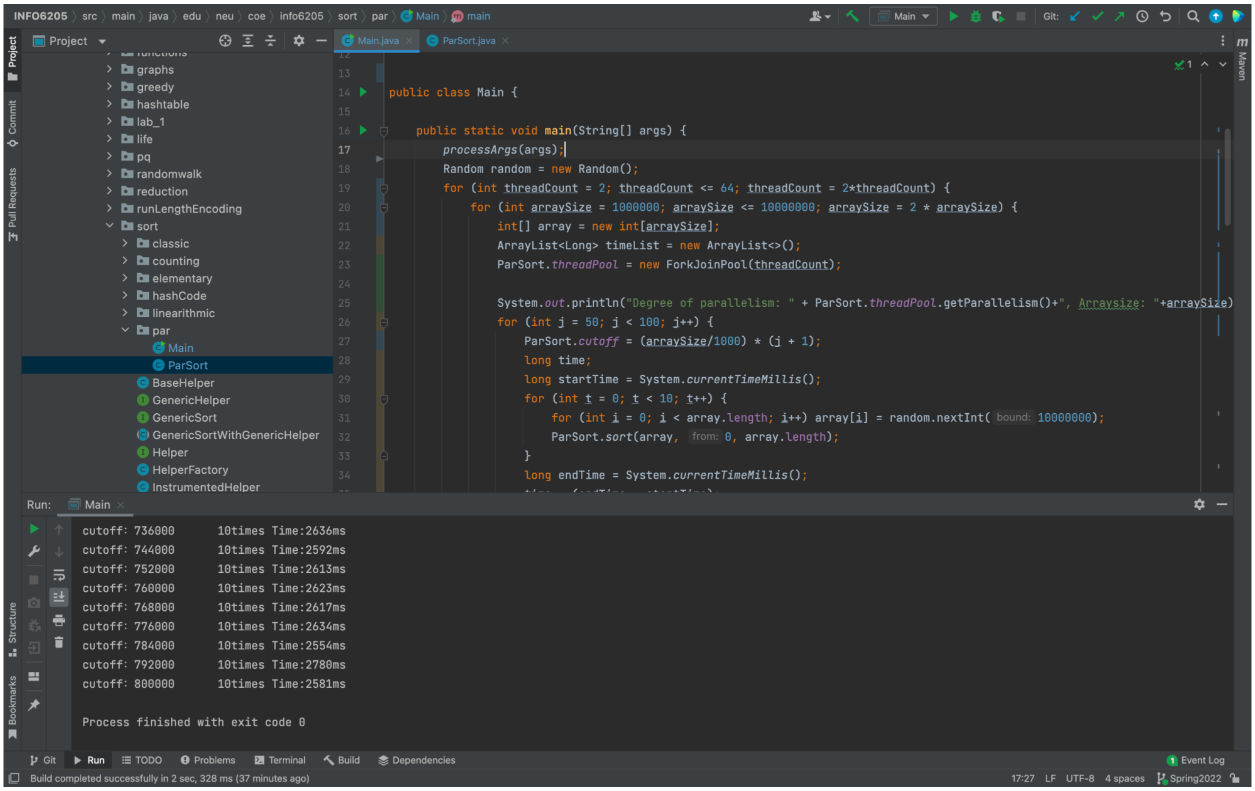
Assignment 4

Name: Sai Harish Reddy Gunda
(NUID): 002981776

Task:

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.

Screenshot with Output:



```java
public class Main {

    public static void main(String[] args) {
        processArgs(args);
        Random random = new Random();
        for (int threadCount = 2; threadCount <= 64; threadCount = 2*threadCount) {
            for (int arraySize = 1000000; arraySize <= 10000000; arraySize = 2 * arraySize) {
                int[] array = new int[arraySize];
                ArrayList<Long> timeList = new ArrayList<>();
                ParSort.threadPool = new ForkJoinPool(threadCount);

                System.out.println("Degree of parallelism: " + ParSort.threadPool.getParallelism()+", Arraysize: "+arraySize)
                for (int j = 50; j < 100; j++) {
                    ParSort.cutoff = (arraySize/1000) * (j + 1);
                    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( bound: 10000000);
                        ParSort.sort(array,  from: 0, array.length);
                    }
                    long endTime = System.currentTimeMillis();
```

```
cutoff: 736000      10times Time:2636ms
cutoff: 744000      10times Time:2592ms
cutoff: 752000      10times Time:2613ms
cutoff: 760000      10times Time:2623ms
cutoff: 768000      10times Time:2617ms
cutoff: 776000      10times Time:2634ms
cutoff: 784000      10times Time:2554ms
cutoff: 792000      10times Time:2780ms
cutoff: 800000      10times Time:2581ms

Process finished with exit code 0
```

Build completed successfully in 2 sec, 328 ms (37 minutes ago)

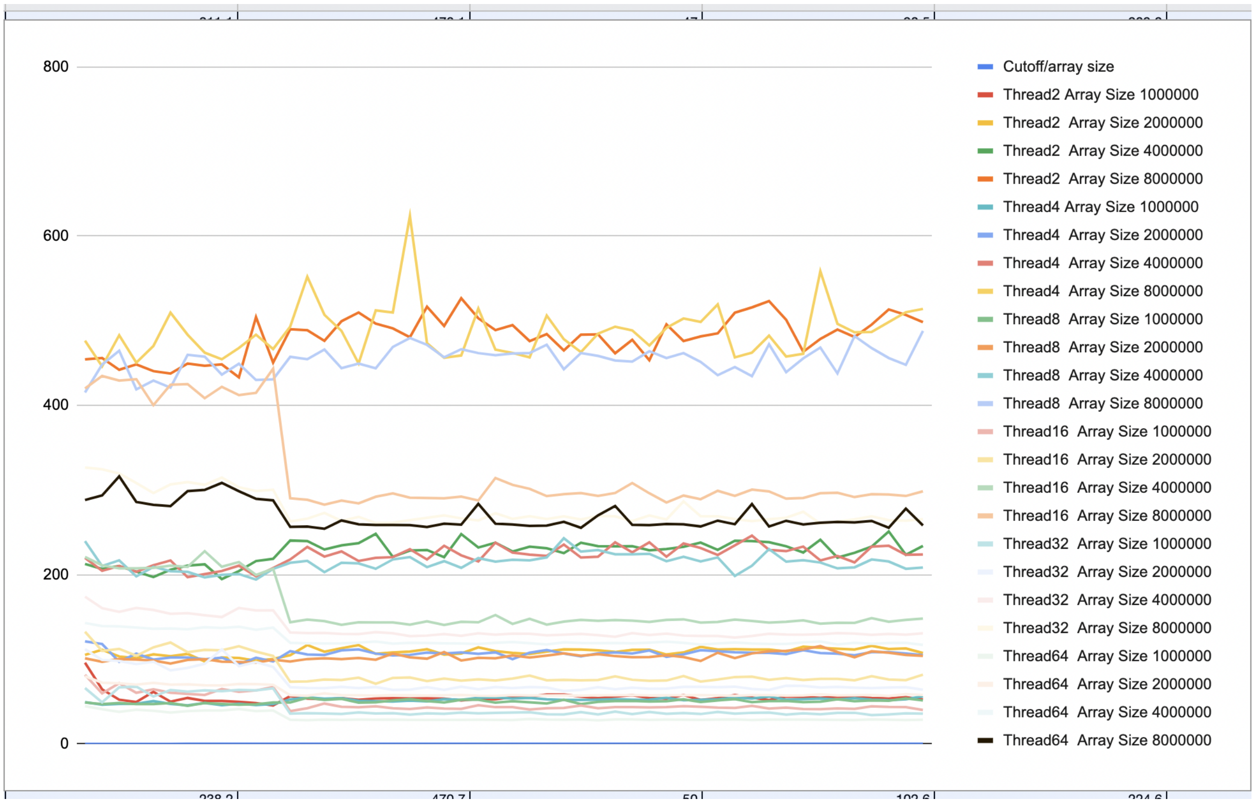
Observation/Conclusion:

Plotted the graph of how Time varies with cutoff/arraysize values for every different combination of thread sizes and array sizes.

Array sizes varied from 1,000,000 to 8,000,000
Thread count values varied from 2 to 64

- For each array size, the best performing thread count was 64 - which is obvious
- Cutoff values didn't affect the performance except when the thread count was 64. The performance increased drastically when the cutoff value was 0.06. I,e for smaller arrays and few threads cutoff didn't effect the performance.

Evidence: (Cutoff/arraysize) vs TIme for each thread count from 2 to 64



output files in the github