Erarica Mehra
NUID: 002113683

Assignment 4:
Parallel Sorting

Tasks

Implemented **Parallel Sorting** such that large arrays can be sorted based on the following schemes:

- Determine an optimal value for **cutoff**. If elements in the array are less than cutoff, then switch to system sort.
- Modify the cutoff values while sorting the array partitions asynchronously.
- Determining **recursion deception** (that is the number of threads). These threads sort the array parallelly.
- Deciding an ideal number of threads to efficiently sort the array. Partitioning must be stopped if recursion depth reaches log n.
- Combination of these schemes: Here I am taking an initial thread count(2) and an initial array size value(1000000) from the command line and doubling these on each iteration, while monitoring the cutoff values and performance of parallel sorting of a large array.

Relationship/Conclusion:

- Parallel Sorting is dependent on balancing the load among the different processors of the system. Efficiency of the parallel sort is highly dependent on the cutoff values. The cutoff value determines when to switch to system sort.
- When the cutoff values are as high as the size of the array and the number of threads are high (that is 32), we can switch to system sort.
- When the cutoff is low and the number of threads are high, the time taken to parallel sort the array increases. Hence the performance of the parallel sort decreases.
- Therefore we can conclude that when the cutoff values are low, the number of threads should be low for better performance.Therefore, it can be rightly said that cutoff values are inversely proportional to the number of threads. Based on the given trials and observations we can decide that the number of threads in this

case can be 4. With 4 threads, the system is efficiently using the processors of the system and the overhead to switch between threads/tasks is minimum.

Observations/ Evidence :

All of the given data is captured in respective CSVs and uploaded with the project in the results folder.

1. **Screenshot of the console output:**
   I have taken a different array sizes with different degrees of parallelization that has produced the following output.

```
Enter inital thread count :
2
Enter inital array size :
1000000
Degree of parallelism: 2
Size of the array: 1000000
cutoff: 510000          10 times Time :832ms
cutoff: 520000          10 times Time :448ms
cutoff: 530000          10 times Time :441ms
cutoff: 540000          10 times Time :457ms
cutoff: 550000          10 times Time :464ms
cutoff: 560000          10 times Time :451ms
cutoff: 570000          10 times Time :439ms
cutoff: 580000          10 times Time :446ms
cutoff: 590000          10 times Time :453ms
cutoff: 600000          10 times Time :455ms
cutoff: 610000          10 times Time :458ms
cutoff: 620000          10 times Time :455ms
cutoff: 630000          10 times Time :440ms
cutoff: 640000          10 times Time :446ms
cutoff: 650000          10 times Time :446ms
cutoff: 660000          10 times Time :444ms
cutoff: 670000          10 times Time :466ms
cutoff: 680000          10 times Time :445ms
cutoff: 690000          10 times Time :498ms
cutoff: 700000          10 times Time :451ms
cutoff: 710000          10 times Time :447ms
cutoff: 720000          10 times Time :462ms
cutoff: 730000          10 times Time :458ms
cutoff: 740000          10 times Time :477ms
cutoff: 750000          10 times Time :439ms
cutoff: 760000          10 times Time :442ms
cutoff: 770000          10 times Time :448ms
cutoff: 780000          10 times Time :439ms
cutoff: 790000          10 times Time :439ms
cutoff: 800000          10 times Time :439ms
cutoff: 810000          10 times Time :440ms
cutoff: 820000          10 times Time :455ms
cutoff: 830000          10 times Time :466ms
cutoff: 840000          10 times Time :444ms
```

```
Degree of parallelism: 4
Size of the array: 2000000
cutoff: 510000          10 times Time :1051ms
cutoff: 520000          10 times Time :997ms
cutoff: 530000          10 times Time :1003ms
cutoff: 540000          10 times Time :983ms
cutoff: 550000          10 times Time :1053ms
cutoff: 560000          10 times Time :989ms
cutoff: 570000          10 times Time :974ms
cutoff: 580000          10 times Time :967ms
cutoff: 590000          10 times Time :983ms
cutoff: 600000          10 times Time :977ms
cutoff: 610000          10 times Time :987ms
cutoff: 620000          10 times Time :986ms
cutoff: 630000          10 times Time :970ms
cutoff: 640000          10 times Time :960ms
cutoff: 650000          10 times Time :957ms
cutoff: 660000          10 times Time :969ms
cutoff: 670000          10 times Time :958ms
cutoff: 680000          10 times Time :952ms
cutoff: 690000          10 times Time :961ms
cutoff: 700000          10 times Time :956ms
cutoff: 710000          10 times Time :973ms
cutoff: 720000          10 times Time :992ms
cutoff: 730000          10 times Time :965ms
cutoff: 740000          10 times Time :981ms
cutoff: 750000          10 times Time :957ms
cutoff: 760000          10 times Time :989ms
cutoff: 770000          10 times Time :1001ms
cutoff: 780000          10 times Time :1114ms
cutoff: 790000          10 times Time :1156ms
cutoff: 800000          10 times Time :1043ms
cutoff: 810000          10 times Time :997ms
cutoff: 820000          10 times Time :1073ms
cutoff: 830000          10 times Time :1008ms
cutoff: 840000          10 times Time :956ms
cutoff: 850000          10 times Time :951ms
cutoff: 860000          10 times Time :945ms
cutoff: 870000          10 times Time :970ms
cutoff: 880000          10 times Time :952ms
```

```
Degree of parallelism: 8
Size of the array: 4000000
cutoff: 510000          10 times Time :2165ms
cutoff: 520000          10 times Time :2048ms
cutoff: 530000          10 times Time :2138ms
cutoff: 540000          10 times Time :1822ms
cutoff: 550000          10 times Time :1747ms
cutoff: 560000          10 times Time :1967ms
cutoff: 570000          10 times Time :1880ms
cutoff: 580000          10 times Time :1722ms
cutoff: 590000          10 times Time :1759ms
cutoff: 600000          10 times Time :1835ms
cutoff: 610000          10 times Time :1806ms
cutoff: 620000          10 times Time :1654ms
cutoff: 630000          10 times Time :1801ms
cutoff: 640000          10 times Time :1738ms
cutoff: 650000          10 times Time :1653ms
cutoff: 660000          10 times Time :1736ms
cutoff: 670000          10 times Time :1744ms
cutoff: 680000          10 times Time :1707ms
cutoff: 690000          10 times Time :1772ms
cutoff: 700000          10 times Time :1784ms
cutoff: 710000          10 times Time :1790ms
cutoff: 720000          10 times Time :1785ms
cutoff: 730000          10 times Time :1813ms
cutoff: 740000          10 times Time :1853ms
cutoff: 750000          10 times Time :1753ms
cutoff: 760000          10 times Time :1728ms
cutoff: 770000          10 times Time :1764ms
cutoff: 780000          10 times Time :1754ms
cutoff: 790000          10 times Time :1787ms
cutoff: 800000          10 times Time :1725ms
cutoff: 810000          10 times Time :1764ms
cutoff: 820000          10 times Time :1732ms
cutoff: 830000          10 times Time :1718ms
cutoff: 840000          10 times Time :1742ms
cutoff: 850000          10 times Time :1929ms
cutoff: 860000          10 times Time :1747ms
cutoff: 870000          10 times Time :1771ms
cutoff: 880000          10 times Time :1778ms
```

```
Degree of parallelism: 16
Size of the array: 8000000
cutoff: 510000          10 times Time :3242ms
cutoff: 520000          10 times Time :3194ms
cutoff: 530000          10 times Time :3967ms
cutoff: 540000          10 times Time :4101ms
cutoff: 550000          10 times Time :3665ms
cutoff: 560000          10 times Time :3397ms
cutoff: 570000          10 times Time :3105ms
cutoff: 580000          10 times Time :3061ms
cutoff: 590000          10 times Time :3962ms
cutoff: 600000          10 times Time :3353ms
cutoff: 610000          10 times Time :3239ms
cutoff: 620000          10 times Time :4256ms
cutoff: 630000          10 times Time :3318ms
cutoff: 640000          10 times Time :3183ms
cutoff: 650000          10 times Time :3121ms
cutoff: 660000          10 times Time :3210ms
cutoff: 670000          10 times Time :3384ms
cutoff: 680000          10 times Time :3603ms
cutoff: 690000          10 times Time :3812ms
cutoff: 700000          10 times Time :3290ms
cutoff: 710000          10 times Time :3278ms
cutoff: 720000          10 times Time :3180ms
cutoff: 730000          10 times Time :3324ms
cutoff: 740000          10 times Time :3299ms
cutoff: 750000          10 times Time :3189ms
cutoff: 760000          10 times Time :3207ms
cutoff: 770000          10 times Time :3329ms
cutoff: 780000          10 times Time :3451ms
cutoff: 790000          10 times Time :3211ms
cutoff: 800000          10 times Time :3275ms
cutoff: 810000          10 times Time :3576ms
cutoff: 820000          10 times Time :3138ms
cutoff: 830000          10 times Time :3256ms
cutoff: 840000          10 times Time :3157ms
cutoff: 850000          10 times Time :3175ms
cutoff: 860000          10 times Time :3262ms
cutoff: 870000          10 times Time :3206ms
```

```
Degree of parallelism: 32
Size of the array: 16000000
cutoff: 510000          10 times Time :8378ms
cutoff: 520000          10 times Time :10597ms
cutoff: 530000          10 times Time :9460ms
cutoff: 540000          10 times Time :9215ms
cutoff: 550000          10 times Time :9186ms
cutoff: 560000          10 times Time :8942ms
cutoff: 570000          10 times Time :7792ms
cutoff: 580000          10 times Time :7267ms
cutoff: 590000          10 times Time :8033ms
cutoff: 600000          10 times Time :9385ms
cutoff: 610000          10 times Time :9681ms
cutoff: 620000          10 times Time :7650ms
cutoff: 630000          10 times Time :9481ms
cutoff: 640000          10 times Time :8507ms
cutoff: 650000          10 times Time :8509ms
cutoff: 660000          10 times Time :10009ms
cutoff: 670000          10 times Time :9419ms
cutoff: 680000          10 times Time :9340ms
cutoff: 690000          10 times Time :10793ms
cutoff: 700000          10 times Time :8907ms
cutoff: 710000          10 times Time :13059ms
cutoff: 720000          10 times Time :11173ms
cutoff: 730000          10 times Time :10775ms
cutoff: 740000          10 times Time :16045ms
cutoff: 750000          10 times Time :12111ms
cutoff: 760000          10 times Time :12978ms
cutoff: 770000          10 times Time :8502ms
cutoff: 780000          10 times Time :9338ms
cutoff: 790000          10 times Time :6297ms
cutoff: 800000          10 times Time :6969ms
cutoff: 810000          10 times Time :12023ms
cutoff: 820000          10 times Time :10202ms
cutoff: 830000          10 times Time :10107ms
cutoff: 840000          10 times Time :9766ms
cutoff: 850000          10 times Time :10977ms
cutoff: 860000          10 times Time :10797ms
cutoff: 870000          10 times Time :9145ms
cutoff: 880000          10 times Time :9672ms
```

2. Spreadsheet Data:

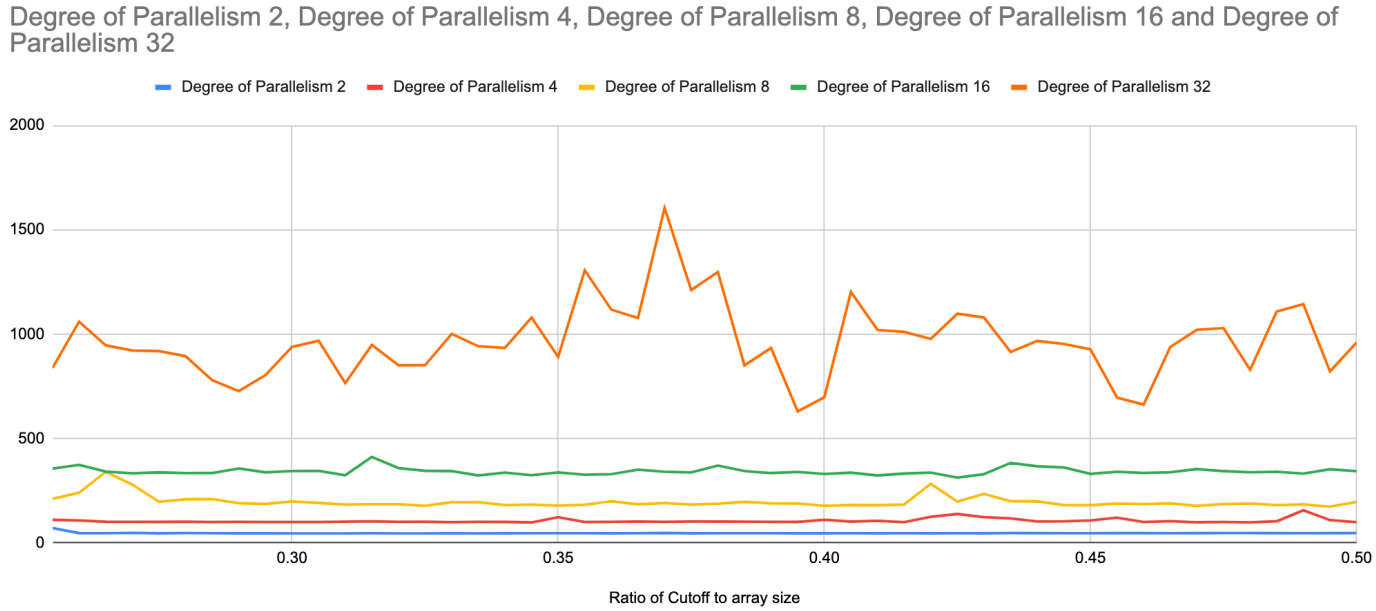
Here I have used spreadsheet data to monitor ratio of cutoff to array size and the degree of parallelism.

Ratio of Cutoff to array size	Degree of Parallelism 2	Degree of Parallelism 4	Degree of Parallelism 8	Degree of Parallelism 16	Degree of Parallelism 32
0.255	**70.2**	**109.8**	**210.5**	**355.3**	**837.8**
0.26	46.2	106.8	239.9	373.2	1059.7
0.265	45.6	100.8	339.6	341.5	946
0.27	47.2	100.6	279	333.1	921.5
0.275	45.3	99.9	196.3	337.5	918.6
0.28	46.7	100.7	208.4	333.9	894.2
0.285	45.5	99.6	209.3	334.1	779.2
0.29	45.3	100.2	189.9	355.9	726.7
0.295	45.3	99.5	186	337.5	803.3
0.3	44.8	99.8	198	343.6	938.5
0.305	45.2	99.3	191.1	344.3	968.1
0.31	45.1	101	183.1	323.8	765
0.315	45.9	102.6	184.7	411.4	948.1
0.32	45.1	100.4	184.6	357.8	850.7
0.325	45.2	100.7	177.3	345	850.9
0.33	45.4	99	194.3	343.4	1000.9
0.335	45.2	100.4	194.4	322.7	941.9
0.34	45.3	100	180.6	336.3	934
0.345	45.5	97.4	183	324.1	1079.3
0.35	45.7	122.7	178.2	336.8	890.7
0.355	45.5	99.5	182.4	326.5	1305.9
0.36	45.4	100.4	198.6	328.9	1117.3
0.365	45.5	101.9	184.4	350.5	1077.5
0.37	46.9	100.5	190.4	340.4	1604.5
0.375	45.3	101.9	183.3	336.9	1211.1
0.38	45.5	101.6	187.4	370.3	1297.8
0.385	45.6	101.1	195.9	343.7	850.2
0.39	45.7	100.2	188.8	334.1	933.8
0.395	45.4	100	187.9	339.3	629.7
0.4	45.4	110.1	177.5	329.9	696.9
0.405	45.7	101.7	181	335.8	1202.3
0.41	45.4	105.1	180.3	322.7	1020.2

0.415	45.6	98.6	183.4	331.7	1010.7
0.42	45.4	124.5	282.5	336.3	976.6
0.425	45.7	138	197.3	312.2	1097.7
0.43	45.4	122.8	234.1	329.1	1079.7
0.435	46.6	117.1	199.3	382.7	914.5
0.44	46.2	101.8	198.5	366.5	967.2
0.445	46	102.6	180.5	361.3	953
0.45	45.8	106.9	180.3	330.2	927.1
0.455	46.1	120.4	188.3	340.6	695.8
0.46	46.2	99.4	185.9	334.8	662.3
0.465	46	103.7	188.7	337.7	938.5
0.47	46.1	98.2	177	353.3	1020.6
0.475	46.6	99.4	185.5	343.2	1028.8
0.48	46.6	97.8	188.3	338	828.7
0.485	46.1	103.3	180.9	340.4	1108.4
0.49	45.9	155.8	184.1	331.6	1143.8
0.495	46.1	108.5	173.4	352.4	821.3
0.5	46.5	98.4	195.2	342.9	959.8

3. Graphical Representation:

The above data can be plotted on a graph :

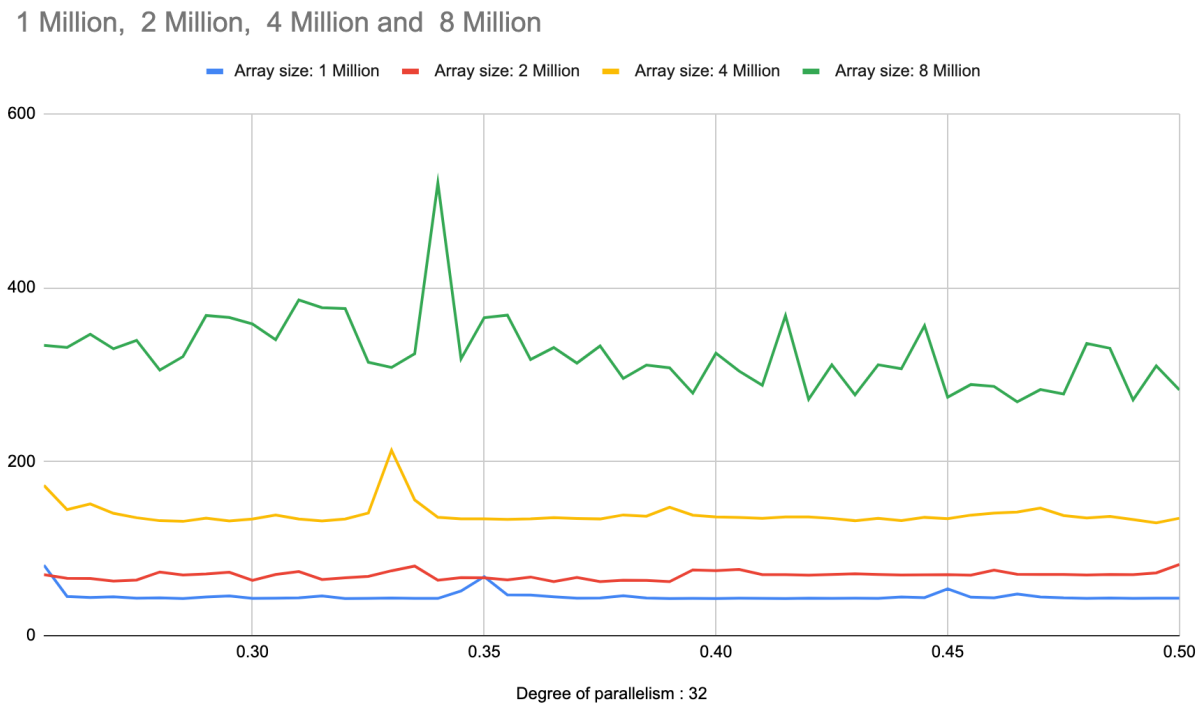


4. **Evidence to study the behavior of the algorithm when thread count is 32 and size of the array increases in each iteration.**

   For this I have created a new Test32DegreeParallelism.

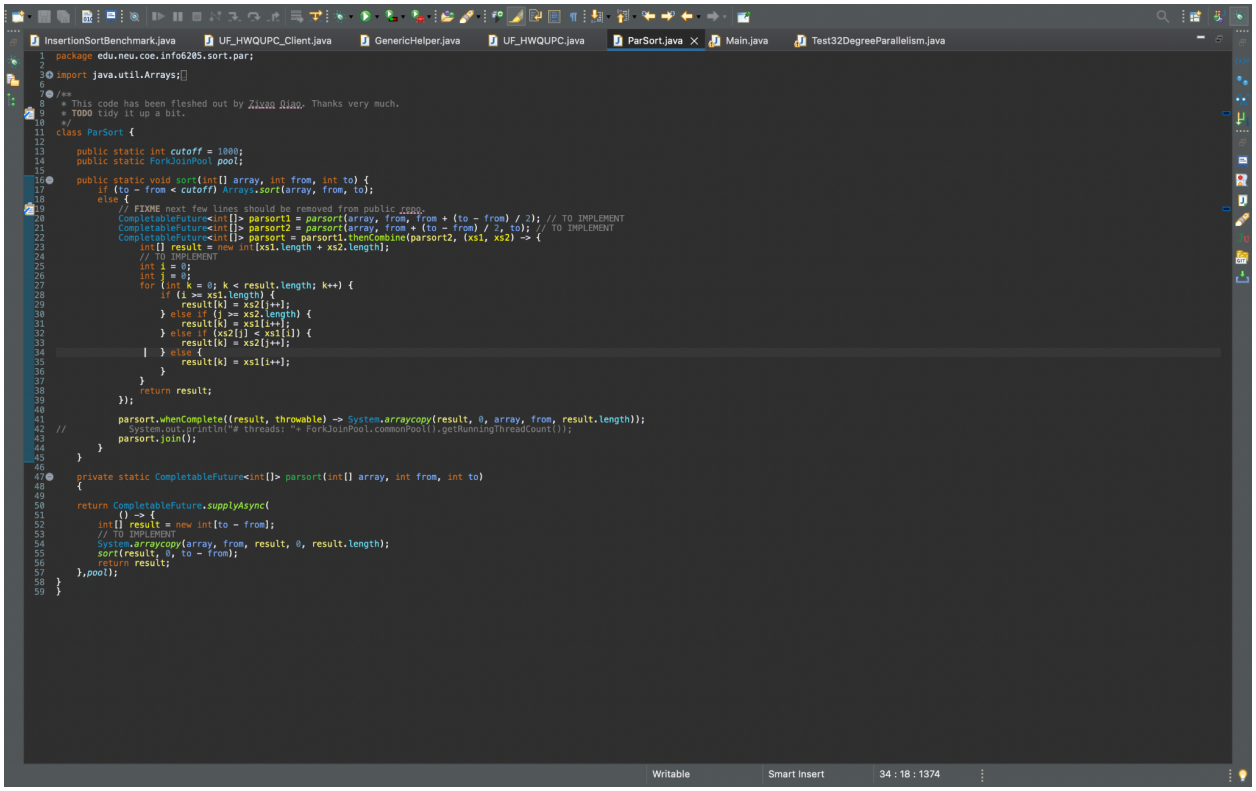
Ratio of Cutoff to array size	Array size: 1 Million	Array size: 2 Million	Array size: 4 Million	Array size: 8 Million
0.255	**81**	**70**	**172.7**	**333.8**
0.26	45	65.9	144.8	331.4
0.265	43.8	65.6	151.3	346.5
0.27	44.6	62.7	140.6	329.8
0.275	42.9	63.8	135.5	339.5
0.28	43.4	73.1	132.1	305.3
0.285	42.6	69.7	131.3	320.8
0.29	44.3	70.8	134.9	368.1
0.295	45.6	72.8	131.8	365.8
0.3	42.7	63.5	134	358.4
0.305	43	70.2	138.5	340.3
0.31	43.3	73.6	134	386
0.315	45.5	64.4	131.8	377.1

0.32	42.5	66.5	134	376
0.325	42.7	68.1	140.9	314.3
0.33	43.1	74.5	213.1	308.3
0.335	42.7	79.9	155.9	324.1
0.34	42.7	63.6	136	520.5
0.345	51.2	66.7	134.2	318.2
0.35	67.7	66.5	134.2	365.6
0.355	46.8	64.1	133.5	368.5
0.36	46.5	67.3	134.2	317.4
0.365	44.5	62	135.6	331.2
0.37	42.9	66.7	134.6	313.3
0.375	43.2	62	134.1	333.1
0.38	45.7	63.6	138.7	295.7
0.385	43.2	63.5	137.3	311
0.39	42.6	62	147.6	307.9
0.395	42.8	75.4	138.4	278.8
0.4	42.6	74.6	136.5	324.7
0.405	42.9	76.1	135.9	304.1
0.41	42.8	70	134.9	287.8
0.415	42.6	70	136.4	367.9
0.42	43	69.5	136.4	271.7
0.425	42.8	70.3	134.7	311.4
0.43	43	71.1	132.1	276.8
0.435	42.8	70.2	134.8	311.4
0.44	44.4	69.6	132.2	306.8
0.445	43.6	69.8	135.9	356.3
0.45	53.7	70	134.5	274.1
0.455	44.1	69.5	138.5	288.7
0.46	43.3	75.3	140.8	286.5
0.465	47.7	70.4	142.1	268.8
0.47	44.4	70.2	146.6	282.9
0.475	43.3	70.3	138	277.8
0.48	42.8	69.7	135.3	335.9
0.485	43.2	70.3	137	330.4
0.49	42.7	70.1	133.4	271
0.495	43	72.1	129.6	310.2
0.5	42.9	81.7	134.9	282.4



Code Snippet:

The following are the changes made to ParSort class



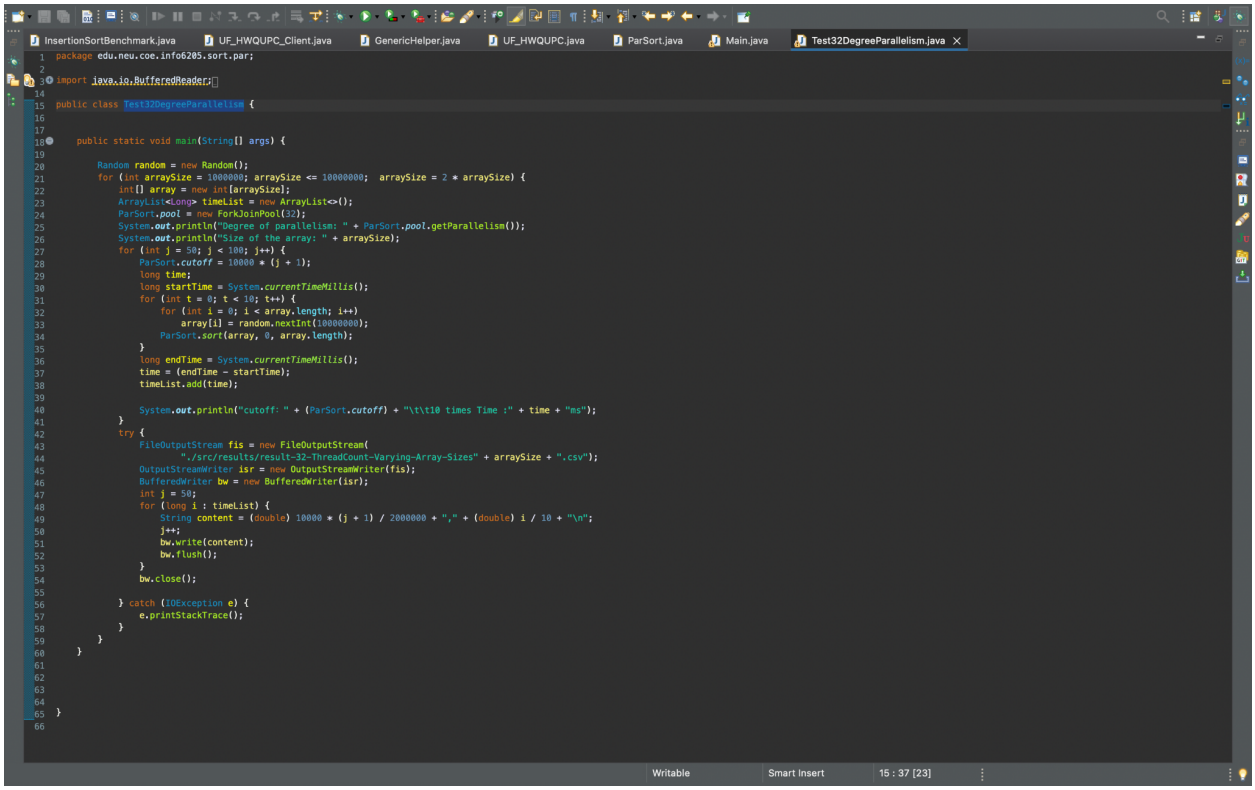
The following are the changes made to Main class

```java
public static void main(String[] args) {
    processArgs(args);
    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
    int initailThreadCount = 0;
    int initialArraySize = 0; // 1000000
    try {
        System.out.println("Enter inital thread count :");
        String thread = reader.readLine();
        initailThreadCount = Integer.parseInt(thread);
        System.out.println("Enter inital array size : ");
        String size = reader.readLine();
        initialArraySize = Integer.parseInt(size);
    } catch (IOException e) {
        e.printStackTrace();
    }
    Random random = new Random();
    int arraySize = initialArraySize;
    for (int threadCount = initailThreadCount; threadCount <= 32; threadCount = 2 * threadCount, arraySize = 2
            * arraySize) {
        int[] array = new int[arraySize];
        ArrayList<Long> timeList = new ArrayList<>();
        ParSort.pool = new ForkJoinPool(threadCount);
        System.out.println("Degree of parallelism: " + ParSort.pool.getParallelism());
        System.out.println("Size of the array: " + arraySize);
        for (int j = 50; j < 100; j++) {
            ParSort.cutoff = 10000 * (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(10000000);
                ParSort.sort(array, 0, array.length);
            }
            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/results/result-" + threadCount + "-threads-arraySize-" + arraySize + ".csv");
            OutputStreamWriter isr = new OutputStreamWriter(fis);
            BufferedWriter bw = new BufferedWriter(isr);
            int j = 50;
            for (long i : timeList) {
                String content = (double) 10000 * (j + 1) / 2000000 + "," + (double) i / 10 + "\n";
                j++;
                bw.write(content);
                bw.flush();
            }
            bw.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

I also created a new test class for my own experimentation:




```java
package edu.neu.coe.info6205.sort.par;

import java.io.BufferedReader;

public class Test32DegreeParallelism {


    public static void main(String[] args) {

        Random random = new Random();
        for (int arraySize = 1000000; arraySize <= 10000000;  arraySize = 2 * arraySize) {
            int[] array = new int[arraySize];
            ArrayList<Long> timeList = new ArrayList<>();
            ParSort.pool = new ForkJoinPool(32);
            System.out.println("Degree of parallelism: " + ParSort.pool.getParallelism());
            System.out.println("Size of the array: " + arraySize);
            for (int j = 50; j < 100; j++) {
                ParSort.cutoff = 10000 * (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(10000000);
                    ParSort.sort(array, 0, array.length);
                }
                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/results/result-32-ThreadCount-Varying-Array-Sizes" + arraySize + ".csv");
                OutputStreamWriter isr = new OutputStreamWriter(fis);
                BufferedWriter bw = new BufferedWriter(isr);
                int j = 50;
                for (long i : timeList) {
                    String content = (double) 10000 * (j + 1) / 2000000 + "," + (double) i / 10 + "\n";
                    j++;
                    bw.write(content);
                    bw.flush();
                }
                bw.close();

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }



}

```