Program Structures and Algorithms

Assignment 4

Jharna Doda
002117574

Implementation for testing various number of threads

```
public static int threadCount=10;
public static ForkJoinPool myPool=new ForkJoinPool(threadCount);
private static CompletableFuture<int[]> parsort(int[] array, int from, int
to) {
    return CompletableFuture.supplyAsync(
            () -> {
                //System.out.println("here2");
                int[] result = new int[to - from];
                // TO IMPLEMENT
                System.arraycopy(array, from, result, 0, result.length);
                sort(result, 0, to - from);
                return result;
            },myPool
    );
}
```

Modified main method to take input for cutoff and threadcount from user

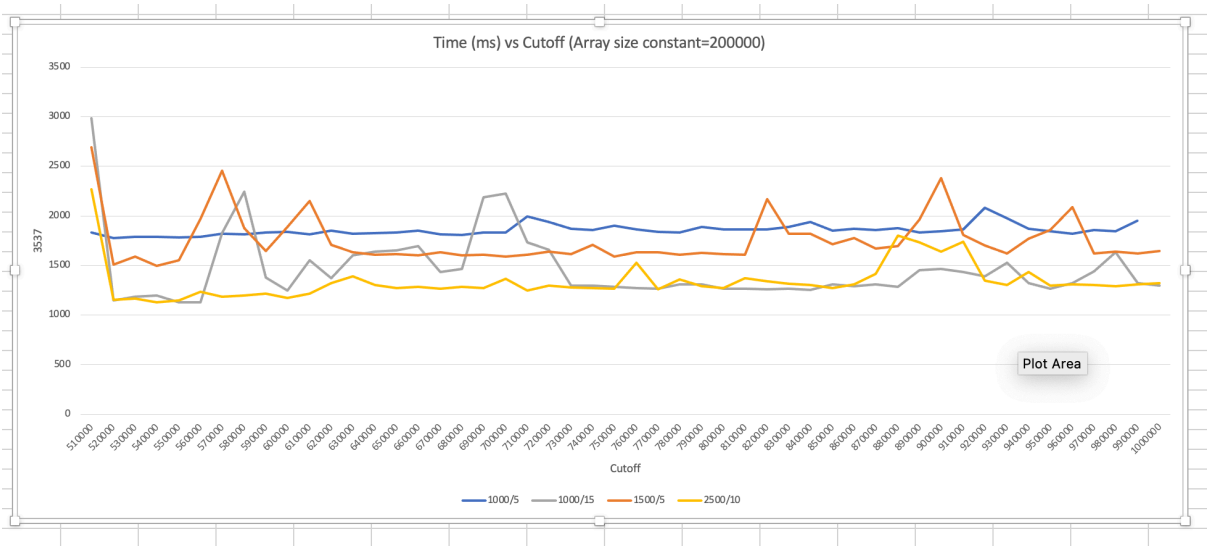
```
public static void main(String[] args) {
    ParSort.threadCount=Integer.parseInt(args[0]);
    ParSort.cutoff=Integer.parseInt(args[1]);
    processArgs(args);
    System.out.println("Degree of parallelism: " +
ForkJoinPool.getCommonPoolParallelism());
    Random random = new Random();
    int[] array = new int[1000000];
    ArrayList<Long> timeList = new ArrayList<>();
    for (int j = 50; j < 100; j++) {
        ParSort.cutoff = 10000 * (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);
        }
        long endTime = System.currentTimeMillis();
        time = (endTime - startTime);
        timeList.add(time);

        System.out.println(ParSort.cutoff+"\t\t"+time);
```

Output:

For Array Size: 200000

Array Size: 200000	Cutoff/thread			
	1000/5	1000/15	1500/5	2500/10
510000	3537	2985	2692	2267
520000	1835	1146	1507	1155
530000	1777	1186	1591	1165
540000	1786	1194	1497	1130
550000	1786	1125	1551	1144
560000	1781	1129	1967	1235
570000	1790	1826	2456	1184
580000	1821	2241	1876	1197
590000	1813	1376	1647	1213
600000	1830	1247	1891	1171
610000	1837	1552	2150	1216
620000	1812	1370	1706	1322
630000	1853	1599	1634	1390
640000	1821	1640	1607	1303
650000	1823	1654	1613	1271
660000	1831	1697	1599	1282
670000	1849	1432	1630	1263
680000	1816	1466	1602	1284
690000	1807	2186	1609	1273
700000	1829	2224	1590	1367
710000	1833	1731	1605	1246
720000	1996	1658	1636	1298
730000	1939	1298	1615	1277
740000	1867	1296	1706	1272
750000	1858	1281	1587	1268
760000	1898	1274	1631	1526
770000	1863	1266	1631	1257
780000	1841	1308	1606	1361
790000	1832	1309	1627	1287
800000	1889	1262	1611	1269
810000	1862	1265	1609	1370
820000	1864	1260	2171	1341



For Array Size: 400000

Array Size: 400000		2500/10	
510000		4295	
520000		2734	
530000		2697	
540000		2750	
550000		2740	
560000		2788	
570000		2804	
580000		2820	
590000		2828	
600000		2793	
610000		2786	
620000		2856	
630000		2806	
640000		2816	
650000		2855	
660000		2809	
670000		2779	
680000		2865	
690000		2821	
700000		2812	
710000		2829	
720000		2847	
730000		2811	
740000		2863	
750000		2819	
760000		2820	
770000		2831	
780000		2801	
790000		2801	
800000		2904	
810000		2809	
820000		2819	
830000		2816	
840000		2824	
850000		3076	
860000		3826	
870000		2861	
880000		2872	

For Array Size: 100000

Array Size: 100000		2500/10	
510000		2573	
520000		608	
530000		621	
540000		605	
550000		580	
560000		577	
570000		586	
580000		577	
590000		576	
600000		570	
610000		569	
620000		583	
630000		631	
640000		570	
650000		567	
660000		575	
670000		665	
680000		632	
690000		638	
700000		884	
710000		861	
720000		762	
730000		634	
740000		612	
750000		614	
760000		775	
770000		621	
780000		605	
790000		700	
800000		592	
810000		596	
820000		598	
830000		611	
840000		621	
850000		604	
860000		606	
870000		600	
880000		607	
890000		606	
900000		604