Program Structures and Algorithms

Spring 2023(SEC 3)
NAME: Keerthana Satheesh
NUID: 002747795
Assignment: 5

Task:
Your task is to implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. You will consider two different schemes for deciding whether to sort in parallel.

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.

You must prepare a report that shows the results of your experiments and draws a conclusion (or more) about the efficacy of this method of parallelizing sort. Your experiments should involve sorting arrays of sufficient size for the parallel sort to make a difference. You should run with many different array sizes (they must be sufficiently large to make parallel sorting worthwhile, obviously) and different cutoff schemes.

Code:

```
2 usages  xiaohuanlin +1
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, destPos: 0, result.length);
                sort(result, from: 0, to: to - from, pool);
                return result;
            }, pool
    );
}
```

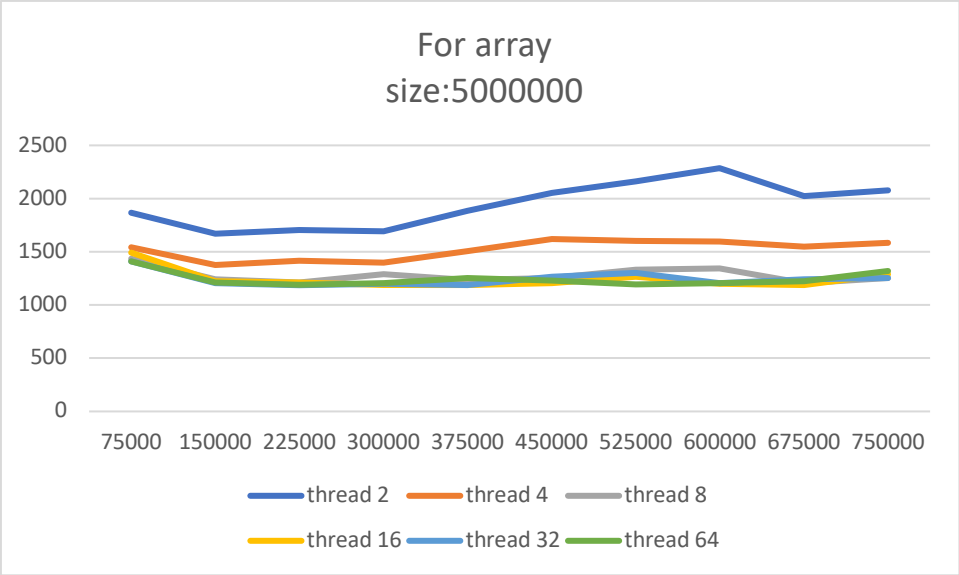
```java
public static void sort(int[] array, int from, int to, ForkJoinPool pool) {
    if (to - from < cutoff) Arrays.sort(array, from, to);
    else {
        // FIXME next few lines should be removed from public repo.
        CompletableFuture<int[]> parsort1 = parsort(array, from,  to: from + (to - from) / 2, pool); // TO IMPLEMENT
        CompletableFuture<int[]> parsort2 = parsort(array,  from: 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;
        });

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

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

    int n = 2500000;
    System.out.println("N: " + n);

    for (int threads = 2; threads <= 64; threads *= 2) {
        ForkJoinPool fork = new ForkJoinPool(threads);
        System.out.println("Degree of parallelism: " + fork.getParallelism());
        System.out.println("Degree of parallelism: " + ForkJoinPool.getCommonPoolParallelism());
        Random random = new Random();
        int[] array = new int[n];
        ArrayList<Long> timeList = new ArrayList<>();
        for (int j = 10; j <= 200; j+=20) {
            ParSort.cutoff = 2000 * (j);
            // 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( bound: 10000000);
                ParSort.sort(array,  from: 0, array.length, fork);
            }
            long endTime = System.currentTimeMillis();
            time = (endTime - startTime);
            timeList.add(time);
```

Evidence:

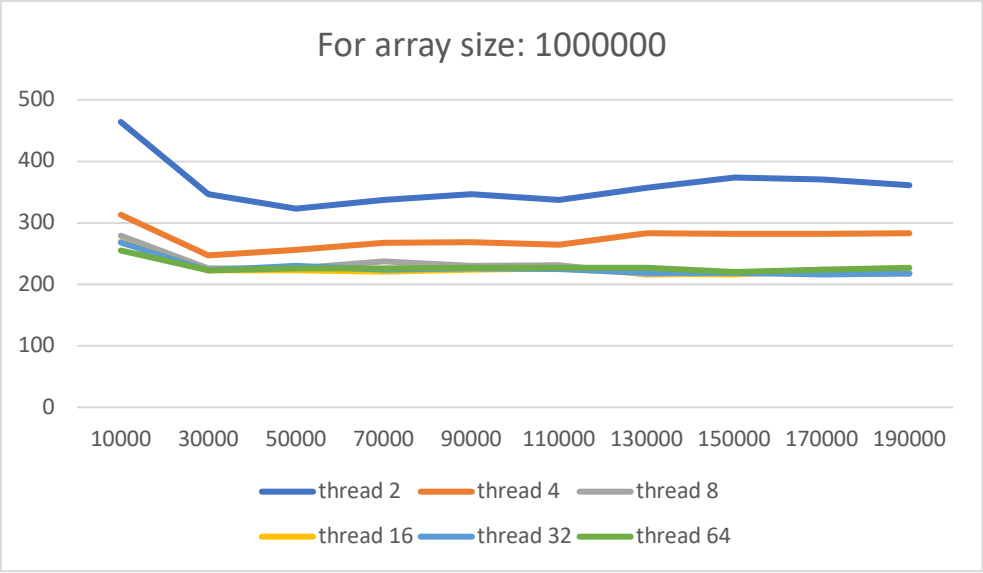
Array Size: 5000000

cutoff	thread 2	thread 4	thread 8	thread 16	thread 32	thread 64
75000	1868	1545	1435	1492	1410	1410
150000	1670	1376	1239	1221	1205	1211
225000	1704	1417	1211	1211	1186	1190
300000	1695	1398	1287	1188	1199	1207
375000	1884	1509	1235	1190	1188	1255
450000	2054	1620	1254	1207	1263	1231
525000	2162	1603	1329	1264	1302	1196
600000	2286	1597	1343	1201	1207	1206
675000	2021	1550	1205	1189	1242	1222
750000	2079	1583	1252	1292	1255	1320



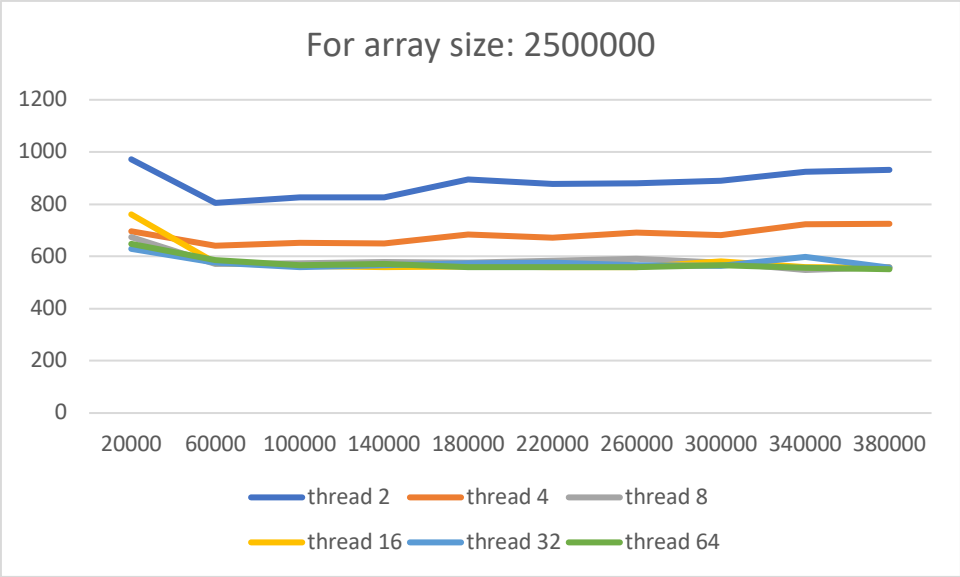
N: 1000000

cutoff	thread 2	thread 4	thread 8	thread 16	thread 32	thread 64
10000	464	313	279	269	268	255
30000	347	247	226	223	223	223
50000	323	256	226	223	230	226
70000	337	267	237	221	224	226
90000	347	268	230	224	226	227
110000	337	264	231	226	225	227
130000	357	283	216	218	219	227
150000	374	282	218	216	219	220
170000	370	282	220	224	216	224
190000	361	283	217	219	217	227



N: 2500000

cutoff	thread 2	thread 4	thread 8	thread 16	thread 32	thread 64
20000	972	696	674	761	629	648
60000	805	641	571	575	575	587
100000	825	653	573	564	559	566
140000	825	649	579	559	567	572
180000	894	685	576	562	574	560
220000	877	672	583	559	577	560
260000	879	692	592	558	566	559
300000	891	682	575	582	565	567
340000	925	723	548	558	599	556
380000	932	725	559	557	557	551



Observations:

In all examples, once the cutoff goes from 1% to 3% there is a notable drop-in execution time. Once we consider a cutoff size greater 3% of array size the overall improvement in performance is minimal.

In the case of thread count, performance improves when we increase the number of threads from 2 to 8. However, there seems to be no difference in performance when increasing the number of threads beyond 8.

From these examples, the ideal cutoff value can be considered to be 3% of the array size considered and the thread count is 8 threads.

```
N: 5000000
Degree of parallelism: 2
cutoff:     75000    Time:  1868    ms
cutoff:     150000   Time:  1670    ms
cutoff:     225000   Time:  1704    ms
cutoff:     300000   Time:  1695    ms
cutoff:     375000   Time:  1884    ms
cutoff:     450000   Time:  2054    ms
cutoff:     525000   Time:  2162    ms
cutoff:     600000   Time:  2286    ms
cutoff:     675000   Time:  2021    ms
cutoff:     750000   Time:  2079    ms
Degree of parallelism: 4
cutoff:     75000    Time:  1545    ms
cutoff:     150000   Time:  1376    ms
cutoff:     225000   Time:  1417    ms
cutoff:     300000   Time:  1398    ms
cutoff:     375000   Time:  1509    ms
cutoff:     450000   Time:  1620    ms
cutoff:     525000   Time:  1603    ms
cutoff:     600000   Time:  1597    ms
cutoff:     675000   Time:  1550    ms
cutoff:     750000   Time:  1583    ms
Degree of parallelism: 8
cutoff:     75000    Time:  1435    ms
cutoff:     150000   Time:  1239    ms
cutoff:     225000   Time:  1211    ms
cutoff:     300000   Time:  1287    ms
cutoff:     375000   Time:  1235    ms
cutoff:     450000   Time:  1254    ms
cutoff:     525000   Time:  1329    ms
cutoff:     600000   Time:  1343    ms
cutoff:     675000   Time:  1205    ms
cutoff:     750000   Time:  1252    ms
```

```
Degree of parallelism: 16
cutoff:     75000    Time:  1492    ms
cutoff:     150000   Time:  1221    ms
cutoff:     225000   Time:  1211    ms
cutoff:     300000   Time:  1188    ms
cutoff:     375000   Time:  1190    ms
cutoff:     450000   Time:  1207    ms
cutoff:     525000   Time:  1264    ms
cutoff:     600000   Time:  1201    ms
cutoff:     675000   Time:  1189    ms
cutoff:     750000   Time:  1292    ms
Degree of parallelism: 32
cutoff:     75000    Time:  1410    ms
cutoff:     150000   Time:  1205    ms
cutoff:     225000   Time:  1186    ms
cutoff:     300000   Time:  1199    ms
cutoff:     375000   Time:  1188    ms
cutoff:     450000   Time:  1263    ms
cutoff:     525000   Time:  1302    ms
cutoff:     600000   Time:  1207    ms
cutoff:     675000   Time:  1242    ms
cutoff:     750000   Time:  1255    ms
Degree of parallelism: 64
cutoff:     75000    Time:  1410    ms
cutoff:     150000   Time:  1211    ms
cutoff:     225000   Time:  1190    ms
cutoff:     300000   Time:  1207    ms
cutoff:     375000   Time:  1255    ms
cutoff:     450000   Time:  1231    ms
cutoff:     525000   Time:  1196    ms
cutoff:     600000   Time:  1206    ms
cutoff:     675000   Time:  1222    ms
cutoff:     750000   Time:  1320    ms
```

```
N: 1000000
Degree of parallelism: 2
cutoff:     10000    Time:  464 ms
cutoff:     30000    Time:  347 ms
cutoff:     50000    Time:  323 ms
cutoff:     70000    Time:  337 ms
cutoff:     90000    Time:  347 ms
cutoff:     110000   Time:  337 ms
cutoff:     130000   Time:  357 ms
cutoff:     150000   Time:  374 ms
cutoff:     170000   Time:  370 ms
cutoff:     190000   Time:  361 ms
Degree of parallelism: 4
cutoff:     10000    Time:  313 ms
cutoff:     30000    Time:  247 ms
cutoff:     50000    Time:  256 ms
cutoff:     70000    Time:  267 ms
cutoff:     90000    Time:  268 ms
cutoff:     110000   Time:  264 ms
cutoff:     130000   Time:  283 ms
cutoff:     150000   Time:  282 ms
cutoff:     170000   Time:  282 ms
cutoff:     190000   Time:  283 ms
Degree of parallelism: 8
cutoff:     10000    Time:  279 ms
cutoff:     30000    Time:  226 ms
cutoff:     50000    Time:  226 ms
cutoff:     70000    Time:  237 ms
cutoff:     90000    Time:  230 ms
cutoff:     110000   Time:  231 ms
cutoff:     130000   Time:  216 ms
cutoff:     150000   Time:  218 ms
cutoff:     170000   Time:  220 ms
cutoff:     190000   Time:  217 ms
```

```
Degree of parallelism: 16
cutoff:     10000    Time:  269 ms
cutoff:     30000    Time:  223 ms
cutoff:     50000    Time:  223 ms
cutoff:     70000    Time:  221 ms
cutoff:     90000    Time:  224 ms
cutoff:     110000   Time:  226 ms
cutoff:     130000   Time:  218 ms
cutoff:     150000   Time:  216 ms
cutoff:     170000   Time:  224 ms
cutoff:     190000   Time:  219 ms
Degree of parallelism: 32
cutoff:     10000    Time:  268 ms
cutoff:     30000    Time:  223 ms
cutoff:     50000    Time:  230 ms
cutoff:     70000    Time:  224 ms
cutoff:     90000    Time:  226 ms
cutoff:     110000   Time:  225 ms
cutoff:     130000   Time:  219 ms
cutoff:     150000   Time:  219 ms
cutoff:     170000   Time:  216 ms
cutoff:     190000   Time:  217 ms
Degree of parallelism: 64
cutoff:     10000    Time:  255 ms
cutoff:     30000    Time:  223 ms
cutoff:     50000    Time:  226 ms
cutoff:     70000    Time:  226 ms
cutoff:     90000    Time:  227 ms
cutoff:     110000   Time:  227 ms
cutoff:     130000   Time:  227 ms
cutoff:     150000   Time:  220 ms
cutoff:     170000   Time:  224 ms
cutoff:     190000   Time:  227 ms
```

```
N: 2500000
Degree of parallelism: 2
cutoff:     20000    Time:  972 ms
cutoff:     60000    Time:  805 ms
cutoff:     100000   Time:  825 ms
cutoff:     140000   Time:  825 ms
cutoff:     180000   Time:  894 ms
cutoff:     220000   Time:  877 ms
cutoff:     260000   Time:  879 ms
cutoff:     300000   Time:  891 ms
cutoff:     340000   Time:  925 ms
cutoff:     380000   Time:  932 ms
Degree of parallelism: 4
cutoff:     20000    Time:  696 ms
cutoff:     60000    Time:  641 ms
cutoff:     100000   Time:  653 ms
cutoff:     140000   Time:  649 ms
cutoff:     180000   Time:  685 ms
cutoff:     220000   Time:  672 ms
cutoff:     260000   Time:  692 ms
cutoff:     300000   Time:  682 ms
cutoff:     340000   Time:  723 ms
cutoff:     380000   Time:  725 ms
Degree of parallelism: 8
cutoff:     20000    Time:  674 ms
cutoff:     60000    Time:  571 ms
cutoff:     100000   Time:  573 ms
cutoff:     140000   Time:  579 ms
cutoff:     180000   Time:  576 ms
cutoff:     220000   Time:  583 ms
cutoff:     260000   Time:  592 ms
cutoff:     300000   Time:  575 ms
cutoff:     340000   Time:  548 ms
cutoff:     380000   Time:  559 ms
```

```
Degree of parallelism: 16
cutoff:     20000    Time:  761 ms
cutoff:     60000    Time:  575 ms
cutoff:     100000   Time:  564 ms
cutoff:     140000   Time:  559 ms
cutoff:     180000   Time:  562 ms
cutoff:     220000   Time:  559 ms
cutoff:     260000   Time:  558 ms
cutoff:     300000   Time:  582 ms
cutoff:     340000   Time:  558 ms
cutoff:     380000   Time:  557 ms
Degree of parallelism: 32
cutoff:     20000    Time:  629 ms
cutoff:     60000    Time:  575 ms
cutoff:     100000   Time:  559 ms
cutoff:     140000   Time:  567 ms
cutoff:     180000   Time:  574 ms
cutoff:     220000   Time:  577 ms
cutoff:     260000   Time:  566 ms
cutoff:     300000   Time:  565 ms
cutoff:     340000   Time:  599 ms
cutoff:     380000   Time:  557 ms
Degree of parallelism: 64
cutoff:     20000    Time:  648 ms
cutoff:     60000    Time:  587 ms
cutoff:     100000   Time:  566 ms
cutoff:     140000   Time:  572 ms
cutoff:     180000   Time:  560 ms
cutoff:     220000   Time:  560 ms
cutoff:     260000   Time:  559 ms
cutoff:     300000   Time:  567 ms
cutoff:     340000   Time:  556 ms
cutoff:     380000   Time:  551 ms
```