Program Structures and Algorithms
Spring 2024

NAME: Rohit Varma Mudundi
NUID: 002688431
GITHUB LINK: https://github.com/rohit26300/Rohit-Varma.git

TO DO:

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.

Conclusion:

- When we increase the cutoff/size ratio, sorting times generally become shorter, making sorting more effective and faster.
- Sort times are usually reduced as the cutoff/size ratio goes up, but the rate of improvement slows down as the ratio approaches 1. Sometimes, very high ratios can even lead to longer sorting times. This implies that there's an ideal ratio beyond which further increases don't significantly improve sorting.
- Different array sizes react differently to changes in the cutoff/size ratio. Larger arrays often benefit more from higher ratios compared to smaller arrays.
- Increasing the cutoff/size ratio typically increases memory usage because more elements are sorted in memory before being written to disk. However, we can adjust this trade-off between memory consumption and sorting time based on what the system needs and the available resources.

Evidence to Support the Conclusion:

Array of Size 2000000

```java
public class Main {

    xiaohuanlin *
    public static void main(String[] args) {
        processArgs(args);
        System.out.println("Degree of parallelism: " + ForkJoinPool.getCommonPoolParallelism());
        Random random = new Random();
        int[] array = new int[2000000];
        ArrayList<Long> timeList = new ArrayList<>();
        for (int j = 50; j < 100; j++) {
            ParSort.cutoff = 10000 * (j + 1);
```

Degree of parallelism: 7

cutoff : 510000	10times Time:1291ms
cutoff : 520000	10times Time:1342ms
cutoff : 530000	10times Time:1255ms
cutoff : 540000	10times Time:1419ms
cutoff : 550000	10times Time:1451ms
cutoff : 560000	10times Time:1425ms
cutoff : 570000	10times Time:1245ms
cutoff : 580000	10times Time:1379ms
cutoff : 590000	10times Time:1440ms
cutoff : 600000	10times Time:1422ms
cutoff : 610000	10times Time:1434ms
cutoff : 620000	10times Time:1382ms
cutoff : 630000	10times Time:1372ms
cutoff : 640000	10times Time:1441ms
cutoff : 650000	10times Time:1310ms
cutoff : 660000	10times Time:1400ms
cutoff : 670000	10times Time:1437ms
cutoff : 680000	10times Time:1410ms
cutoff : 690000	10times Time:1457ms
cutoff : 700000	10times Time:1442ms
cutoff : 710000	10times Time:1464ms
cutoff : 720000	10times Time:1373ms
cutoff : 730000	10times Time:1415ms
cutoff : 740000	10times Time:1271ms
cutoff : 750000	10times Time:1275ms
cutoff : 760000	10times Time:1337ms
cutoff : 770000	10times Time:1375ms
cutoff : 780000	10times Time:1312ms
cutoff : 790000	10times Time:1388ms
cutoff : 800000	10times Time:1387ms
cutoff : 810000	10times Time:1404ms
cutoff : 820000	10times Time:1436ms

cutoff : 830000 10times Time:1314ms
cutoff : 840000 10times Time:1318ms
cutoff : 850000 10times Time:1295ms
cutoff : 860000 10times Time:1395ms
cutoff : 870000 10times Time:1405ms
cutoff : 880000 10times Time:1389ms
cutoff : 890000 10times Time:1462ms
cutoff : 900000 10times Time:1477ms
cutoff : 910000 10times Time:1324ms
cutoff : 920000 10times Time:1235ms
cutoff : 930000 10times Time:1317ms
cutoff : 940000 10times Time:1295ms
cutoff : 950000 10times Time:1382ms
cutoff : 960000 10times Time:1428ms
cutoff : 970000 10times Time:1433ms
cutoff : 980000 10times Time:1443ms
cutoff : 990000 10times Time:1363ms
cutoff : 1000000 10times Time:1395ms

Process finished with exit code 0

Array of Size 3000000

Degree of parallelism: 7
cutoff : 510000 10times Time:2072ms
cutoff : 520000 10times Time:2115ms
cutoff : 530000 10times Time:2194ms
cutoff : 540000 10times Time:2322ms
cutoff : 550000 10times Time:2345ms
cutoff : 560000 10times Time:2247ms
cutoff : 570000 10times Time:2087ms
cutoff : 580000 10times Time:2273ms
cutoff : 590000 10times Time:2260ms
cutoff : 600000 10times Time:2328ms
cutoff : 610000 10times Time:2438ms
cutoff : 620000 10times Time:2346ms
cutoff : 630000 10times Time:2061ms
cutoff : 640000 10times Time:2244ms
cutoff : 650000 10times Time:2463ms
cutoff : 660000 10times Time:2439ms
cutoff : 670000 10times Time:2222ms

cutoff : 680000 10times Time:2417ms
cutoff : 690000 10times Time:2303ms
cutoff : 700000 10times Time:1755ms
cutoff : 710000 10times Time:1766ms
cutoff : 720000 10times Time:2274ms
cutoff : 730000 10times Time:2440ms
cutoff : 740000 10times Time:2454ms
cutoff : 750000 10times Time:2411ms
cutoff : 760000 10times Time:2391ms
cutoff : 770000 10times Time:2377ms
cutoff : 780000 10times Time:2418ms
cutoff : 790000 10times Time:2457ms
cutoff : 800000 10times Time:2317ms
cutoff : 810000 10times Time:2426ms
cutoff : 820000 10times Time:2414ms
cutoff : 830000 10times Time:2367ms
cutoff : 840000 10times Time:2251ms
cutoff : 850000 10times Time:2207ms
cutoff : 860000 10times Time:2355ms
cutoff : 870000 10times Time:2477ms
cutoff : 880000 10times Time:2279ms
cutoff : 890000 10times Time:2042ms
cutoff : 900000 10times Time:2311ms
cutoff : 910000 10times Time:2423ms
cutoff : 920000 10times Time:2398ms
cutoff : 930000 10times Time:2398ms
cutoff : 940000 10times Time:2233ms
cutoff : 950000 10times Time:2282ms
cutoff : 960000 10times Time:2362ms
cutoff : 970000 10times Time:2467ms
cutoff : 980000 10times Time:2309ms
cutoff : 990000 10times Time:2268ms
cutoff : 1000000 10times Time:1970ms

Process finished with exit code 0