**Assignment 5**

Aparna Anil Shelar(NU ID 001009233)

1. Task
2. Tested for single thread performance of 300 different cuttoff values time taken
3. Tested for multiple thread performance of 300 different cuttoff values time taken
4. Created table and graph for my observation.
5. Also Wrote conclusion as per my observations
6. Code
7. for( int k = 1; k <= 8; k = k\*4) {
8. ParSort.threadCount = k;
9. ParSort.threadFJP = new ForkJoinPool(ParSort.threadCount);
10. String seriesName = "" + k + " Threads";
11. // XYSeries timeSeries = new XYSeries(seriesName);
12. double min = 99999;
13. int minCutoff = 0;
14. double avg = 0;
15. for (int j = 0; j < 300; j++) {
17. ParSort.cutoff = 10000 \* (j + 1);
18. // for (int i = 0; i < array.length; i++) array[i] = random.nextInt(10000000);
19. long time;
20. long startTime = System.currentTimeMillis();
21. for (int t = 0; t < 20; t++) {
22. for (int i = 0; i < array.length; i++)
23. array[i] = random.nextInt(10000000);
24. ParSort.sort(array, 0, array.length);
25. public static int threadCount = 1;
26. static ForkJoinPool threadFJP = new ForkJoinPool(threadCount);
27. Output
28. Graph and Table

Table

Description automatically generated

Table

Description automatically generated

Chart

Description automatically generated with medium confidence

Graphical user interface, chart

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

Graphical user interface

Description automatically generated with medium confidence

**5.Conclusion**

1. For single thread, performance is not good compares to multi thread.

2. For multi thread, as shown in above graph when multiple thread comes into the picture time taken to sort array decreases. 16 thread giving best performance.

3. For multi thread, smaller the cutoffs mean more segments to process in parallel hence improving time.

3. Also, when array size and cutoff value decreases time also decreases.