Program Structures and Algorithms

Spring 2023(SEC –)

NAME: Zhiliang Liu

NUID:002650051

**Task: Parallel sorting**

**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.**

**Relationship Conclusion:**

**By increasing number of threads and observe how time consumption differs. Before it reaches 8, time consumption keeps decreasing because more threads are involving calculation. After number of threads moves beyond 16, the time becomes stable because my CPU is 4 core 8 threads.**

**Evidence to support that conclusion:**

**The graph shows different time consumptions of different ratios of cutoff to array size. For the ratios below 0.16, the time consumptions go down as ratio increases. For the ratios over 0.16, the time consumptions go high as ratio increases. Time moves to stable when the ratio is over 1. So, when the ratio is 0.16, time consumption is least.**

**The array length is 200000. When the cutoff is between 52000 and 100000, the time consumptions is least. So, I conclude that the ratios of cutoff to array size between 0.25-0.5, the time consumption is least.**

**By increasing number of threads and observe how time consumption differs. Before it reaches 8, time consumption keeps decreasing because more threads are involving calculation. After number of threads moves beyond 16, the time becomes stable because my CPU is 4 core 8 threads.**

**Graphical Representation:**

**Unit Test Screenshots:**