Yanrong Huang (NUID 001563514)

**Program Structures & Algorithms**

**FALL 2021**

**Assignment No. 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.

* **Output**

To find the best points of cutoff, I set the degree of parallelism as 64, then change array size and cutoff multiple times to test the running time. The results are shown below:

The first col means cutoff/array size, the second col means the cutoff points and the third col means run 10 times (I didn’t get the average time because it is same trend).

表格, Excel

描述已自动生成

For the ideal number of separate threads should stick to powers of 2 since it is decided by the recursion depth. I test the running time when the array size is 2000000.

表格

描述已自动生成

* **Relationship Conclusion**

1. The “cutoff-time chart” obtained from the results is shown below:

图表, 折线图

描述已自动生成

From this chart we can see that each group(with different array size) is intended to decrease at first, reach the lowest point then increase a bit and have a minimum cutoff value around 0.2 of the array size and it doesn’t have more point lower that it. Therefore, it can be concluded that the lowest time point is about where cutoff is about 20% of array size.

1. I increase the number of threads to sort the array size = 2000000 with power of 2 from 1 to 256

图表, 折线图

描述已自动生成

From thread 1 to thread 256

In this graph we can see that diff thread have small different after thread = 4. Thus, the conclusion is when thread increases from 1-8, running time will decrease, after 8, time will not change too much.

图表, 散点图

描述已自动生成

Compare the time it consumed on sorting where array size is 2000000 and cutoff is 400000 which is 20% of the array size. We can see that the lowest time happened when it comes to 8 threads.

Thus, the optimum cutoff is 20% of array size and the optimum number of threads is 8. And it is worth to choose parallel sorting because it save time.