

# Program Structures & Algorithms

Spring 2022

## Assignment No. 4

Name: Raj Mehta

(NUID): 001094914

- **Task**

- 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.
- 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.
- 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).
- An appropriate combination of these.

- Output screenshot

```
Run: Main
Degree of parallelism: 7
Height: 0
1377
1283
1272
1279
1312
1273
1264
1271
1274
1283
1285
834
821
822
824
823
1326
1288
1279
1274
1281
1271
1274
1275
1271
621
628
688
821
822
1277
1268
1274
1278
1269
688
621
688
824
826
1274
1268
1277
1280
1276
613
624
686
819
830
1279
1281
1274
1278
1265
689
622
687
826
825
1271
Height: 3
```

```
Run: Main
Height: 3
621
628
688
821
822
1277
1268
1274
1278
1269
688
621
688
824
826
1274
1268
1277
1280
1276
613
624
686
819
830
1279
1281
1274
1278
1265
689
622
687
826
825
1271
Height: 6
```

- **Relationship Conclusion**

- We can see from the graphs that at “height = 0”, the performance is the worst. Then, as the height increases, the performance increases considerably. But, from the point of “height = 3”, we see that the performance increase is not very good.

- **Evidence / Graph**

