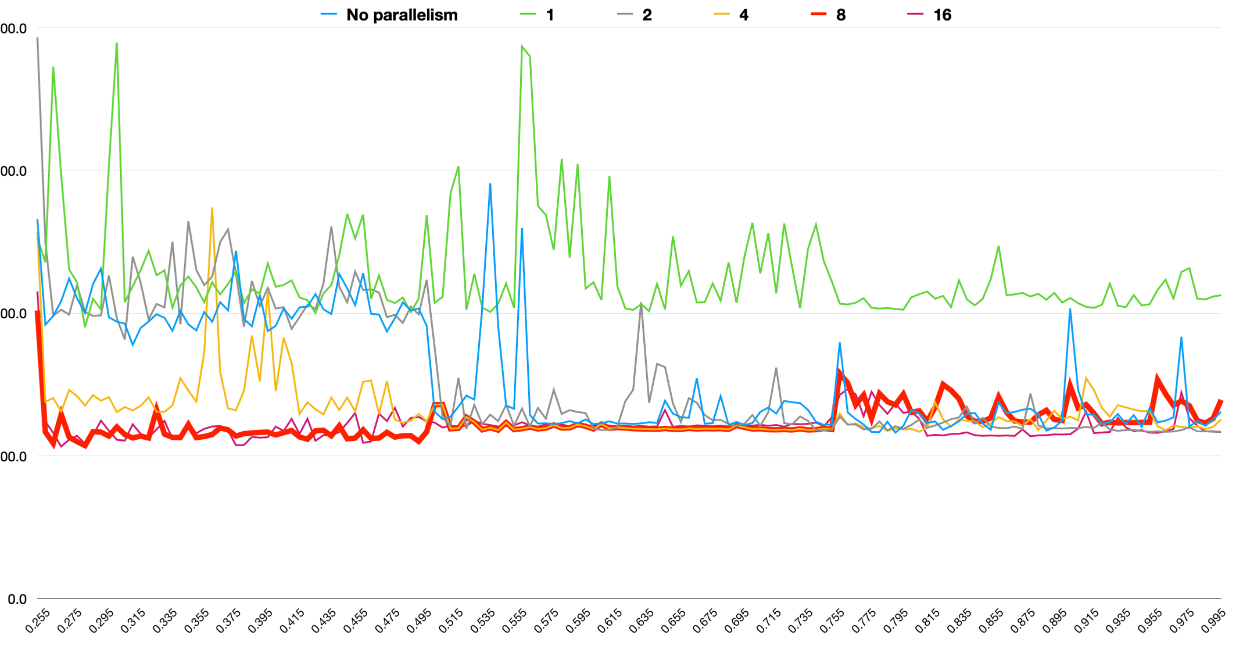
Nien-chi Hung (002980433)

Program Structures & Algorithms
Fall 2021
Assignment NO. 5

- Tasks:
  - In the main() function of the class Main, each cutoff was tested for 10 times, each test was ran with a fixed-size array with different values
  - In the parsort() function of the class ParSort, an instance of ForkJoinPool was passed as the second parameter of CompletableFuture.*supplyAsync()* to determine the thread count
  - If the partition size is smaller than cutoff, use Arrays.sort(), otherwise, parsort() was called recursively for each partition
  - The experiments were ran with
    - Arrays of size 2,000,000
    - Cutoff from 51,000 to 2,000,000
    - Thread count: 1, 2, 4, 8, 16 (stick to the power of 2)

- Conclusion:
  - When the thread count is set to 8, the sort is fastest. If the thread count is set to higher than 8 (16, 32…), the performance was almost the same, it won't get any better (probably has something to with the capacity of CPU I think(?)
  - The run time sightly went up when the cutoff was set to 1,000,000. So I would suggest that if the array size is smaller than 1,000,000, we should switch to the system sort
  - When the thread count is set to 1, it performed really bad. Probably because we strictly restrict the thread to be only 1 and didn't allow the OS to optimize resource

- Evidence to Support the Conclusion:


No parallelism