In this study there are 10 calculations and time measurement are printed to output files. I uploaded 4 cpp files. (For one thread, five threads, ten threads and optional number of threads). Makefile compiles all cpp files. Their .o files are, nonthread.o, thread-5.o, thread-10.o and thread-free.o. output1.txt for one thread, output2.txt for five threads, output3.txt for 10 threads and output4.txt for optional number of threads.

Functions takes parameter as list and does not change it. There are ten global variables for values which will be printed. Min and max scans the list and find the relevant value. Range, scans list, finds bot min and max value and subtracts min from max. Mode sorts list, scans and finds mode value. Median sorts list too and takes element at the middle if size is odd, otherwise takes average of middle values. Sum scans list and sums all values. Amean function, does same thing with sum does. Also divides to size. Hmean function scans and calculates harmonic mean. Derivation function does same thing with amean does. Then, it scans the list and finds derivation. Lastly, irange function sorts the array and find the result with indexes. If size is odd, sorted list is parted into two without median, otherwise sorted list parted into two without extracting any element.

I have 10 basic functions to find results. In none of them the main list is changed. I did not sort array out of the functions. To calculate median, mode and interquartile range I sorted the copy of array in each of these three functions. When I did not copy the array while sorting, the program gave me segmentation fault. Therefore, I copied them. Besides, I wanted to see benefits of threading, so I wanted functions not to implement different. As the result threading was beneficial on speed of the program. I did not create an extra thread in one thread example. In the others I create extra threads as input number. For example, if user wants to use 10 threads, I created 10 threads, but if user wants to use one thread, I did not create any threads except main thread.

Time is measured with chrono library. The average timing results in 3 tries for each of 10 threads for 300000 sized array: 0.13286 seconds for 1, 0.08643 seconds for 2, 0.08331 seconds for 3, 0.04444 seconds for 4, 0.04703 for 5, 0.04700 for 6, 0.05131 for 7, 0.05058 for 8, 0.04842 for 9, 0.04541 seconds for 10. The complexities of functions are: O(n) for minimum, maximum, range, sum, arithmetic mean, harmonic mean, standard deviation; O(nlogn) for median, mode and interquartile range. Threads are usually beneficial but for instance not in 3 threads compared to 2 threads. This is because, the slowest thread of 3 threads calculate min, max, median and mode. In two-thread example, there is also range in this group. Range is not slow compared to mode and median, so its time is not affected much. If I parted three slow functions in three different groups, it would be faster. In 4-thread example, all slow functions are parted, and program is fastened much. I also realized 10 threads are close to 4 threads on their speeds. This should be caused from the fact creating thread causes is not a quick process. These timing results are measured with wsl in Windows. The program also executed successfully in linux virtual machine but there were no time differences, due to probably limited access to computer.

Thread-free.cpp is optional number of threads file. I implemented many void functions which consist of basic calculation functions for threads. For optional file, if there is no thread number input, the program works with 10 threads. Also, if the number is not integer between 1 and 10, the program creates 10 threads again. In other words, program works with 10 threads unless otherwise stated which makes sense. Makefile compiles all four files. Actually, there is no need other files than optional file, but I wanted to upload the others too. If there is a problem in that, there will be a work in your hand.

YUNUS EMRE ALTUĞ 2019400057