Cmpe322 Project 2

Furkan Bülbül - 2020400318

Implementation

I implemented the code in a way that the program gets the thread count as the program argument. While implementing first I created the global variables that hold the min, max, mean, etc since the threads share the same global values in the process. I implemented the functions that calculate the values we want. They work independently from each other since we want them to start working at the same time in the threads. I determined which threads run which functions depending on the number of threads. First I implemented the function that runs all calculating functions run on the main thread (Thread count is equal to 0 or 1). After that, I implemented ten threads each thread runs one function. For each other thread count, I determined the calculating function groups so that we could utilize from threads.

Note that I implemented the function that finds the mode in a way that it finds the smallest mode because without sorting the array when the size of the array is large the complexity grows in the order of N^2 - implemented with map. So the thread running the mode function was too slow compared to other ones. So I got the mode after sorting. Also, I printed all my results into **output.txt** file.

Execution Time

I executed the program with 1e8 array size and average execution as follows:

1 thread: 17.502 seconds

5 threads: 8.792 seconds

10 threads: 5.295 seconds As we expected while the thread count increases the execution time decreases. The change is more dramatic when the array size is large because thread creation and joining time do not mean much compare to the operations done. Sometimes increasing the thread count by one does not improve time much and there must be an upper bound but in general, it is efficient to increment the thread count.