Given

For n= 20000000 it takes 225671780

For n= 30000000 it takes 319423388

For n= 40000000 it takes 415830995

For n= 50000000 it takes 515627818

For n= 60000000 it takes 621025103

For n= 70000000 it takes 727586909

For n= 80000000 it takes 826753936

For n= 90000000 it takes 930502061

For n= 100000000 it takes 1071962401

For n= 110000000 it takes 1213937524

For n= 120000000 it takes 1336769236

For n= 130000000 it takes 1344776188

For n= 140000000 it takes 1468317927

For n= 150000000 it takes 1550473020

For n= 160000000 it takes 1894597151

For n= 170000000 it takes 1770158483

For n= 180000000 it takes 1868072873

For n= 190000000 it takes 1969451608

For n= 200000000 it takes 2124556927

For n= 210000000 it takes 2201401877

Parallel

For n= 20000000 it takes 182099858

For n= 30000000 it takes 130754258

For n= 40000000 it takes 166827517

For n= 50000000 it takes 203047789

For n= 60000000 it takes 256954780

For n= 70000000 it takes 316462230

For n= 80000000 it takes 346087260

For n= 90000000 it takes 361796934

For n= 100000000 it takes 426929889

For n= 110000000 it takes 473202795

For n= 120000000 it takes 513993500

For n= 130000000 it takes 521445324

For n= 140000000 it takes 594772130

For n= 150000000 it takes 635998778

For n= 160000000 it takes 673681311

For n= 170000000 it takes 763242313

For n= 180000000 it takes 749772174

For n= 190000000 it takes 811303044

For n= 200000000 it takes 911756563

For n= 210000000 it takes 1007651243

Sequential Stream

For n= 20000000 it takes 305304901

For n= 30000000 it takes 378869158

For n= 40000000 it takes 498918725

For n= 50000000 it takes 627064288

For n= 60000000 it takes 749246261

For n= 70000000 it takes 875715304

For n= 80000000 it takes 997603248

For n= 90000000 it takes 1123977681

For n= 100000000 it takes 1256441388

For n= 110000000 it takes 1386810256

For n= 120000000 it takes 1495357452

For n= 130000000 it takes 1619119481

For n= 140000000 it takes 1990820178

For n= 150000000 it takes 1866446439

For n= 160000000 it takes 2003037679

For n= 170000000 it takes 2128845390

For n= 180000000 it takes 2257409736

For n= 190000000 it takes 2384642140

For n= 200000000 it takes 2522343641

For n= 210000000 it takes 2617514844

What is Time Complexity?

Time Complexity measures the time taken for running an algorithm and it is commonly used to count the number of elementary operations performed by the algorithm to improve the performance. Let’s starts with simple example to understand the meaning of Time Complexity in java.

Project two of COMP182 on Bench Marking assignment. We are given a program and we are given different methods to test the program for its Time Complexity, using three different functions to execute its algorithm. Firstly, the time complexity of the given code in java implementation, secondly in an Parallel Stream of the given equation, and finally testing the equation in a Sequential Stream to test its time complexity. From the given methods to execute on the original code, I have chosen the Trapezium Method that was provided.

For better testing practice of the Java code in Time Complexity we changed the code to take input of substantially large numbers that ran 20 times. Thus at the beginning of the inputs began from 10000000 and each run-in after was increased by 10000000 Consequently this helps us to visualize all three versions of the tests in forms of graphs and therefore it makes it easier for us to understand the outputs of these tests and differences.

Also, I took note that my program is running on 4 processors by using the following code

In Conclusion, from the above graphs that are provided from the numerical data that was gathered from executing different functions, as per observing the graphs I find that the use of Parallel Streams on the given program, the program was able to be executed much faster than the rest. As per the results the highlight was that the parallel streams executed as much as five times faster than the Sequential Stream. Moreover, as per the data provided Sequential Stream was the slowest to execute the program from all, as the Sequential Stream even started off the slowest but however closed in on the speed of the Original Code execution time but up running the program it gradually parted ways from the Original Code execution time and became the slowest to execute the program. Finally the execution time of the Original Code started off almost matching the speeds of the Parallel Stream but how ever upon running it multiple times it got slower and slower than the execution time it started off from. Moreover as per running the program we can conclude that The Sequential Stream was the slowest to execute the program, The Original execution started off fast as fast as the Parallel Stream but however getting slower and slower even closing in on the speeds of the Sequential Stream but however not going slower than the Sequential Stream making it stay right in the center, between Sequential Stream and the Parallel Stream, Finally Parallel Stream was the fastest to execute the program as it started off with similar speeds of the Original execution but however up many runs it increased its speeds of execution greatly than the Original Execution.

Using the compared final result, I noticed that the parallel stream executed the function much quicker than the others. It's final result finished nearly four times quicker than the sequential stream. The

sequential stream started at a quick rate almost matching it's parallel counter-part, but it gradually got much slower as the program pushed through its 20 runs. The original function was slower than both of the stream methods, but it kept a solid pace and ended up surpassing the sequential stream. Looking at the final numbers, it's time complexity appears to be around the dead center of the other two.

To conclude, the parallel stream is quickest, the original function is average, and the sequential stream is the slowest