Program Structures and Algorithms
Spring 2024

NAME: Tarun Angrish
NUID: 002807094
GITHUB LINK: https://github.com/tarunangrish-neu/INFO6205
Task: In this assignment, we were asked to experiment with different values of threads and to ascertain which degree of parallelism in the sorting can help us achieve optimal results. Below is a brief overview of the experiments conducted and how the results were achieved.

Relationship Conclusion:
The assessment of parallel processing effectiveness involves examining execution times across different cutoff settings and degrees of parallelism. The idea is that higher parallelism should theoretically result in quicker execution times. However, the analysis indicates that performance improves up to a certain number of threads, after which it starts to decline. This decline could be attributed to factors like the overhead of creating a large number of threads, synchronization challenges, and various system characteristics such as the number of available threads and the CPU architecture.

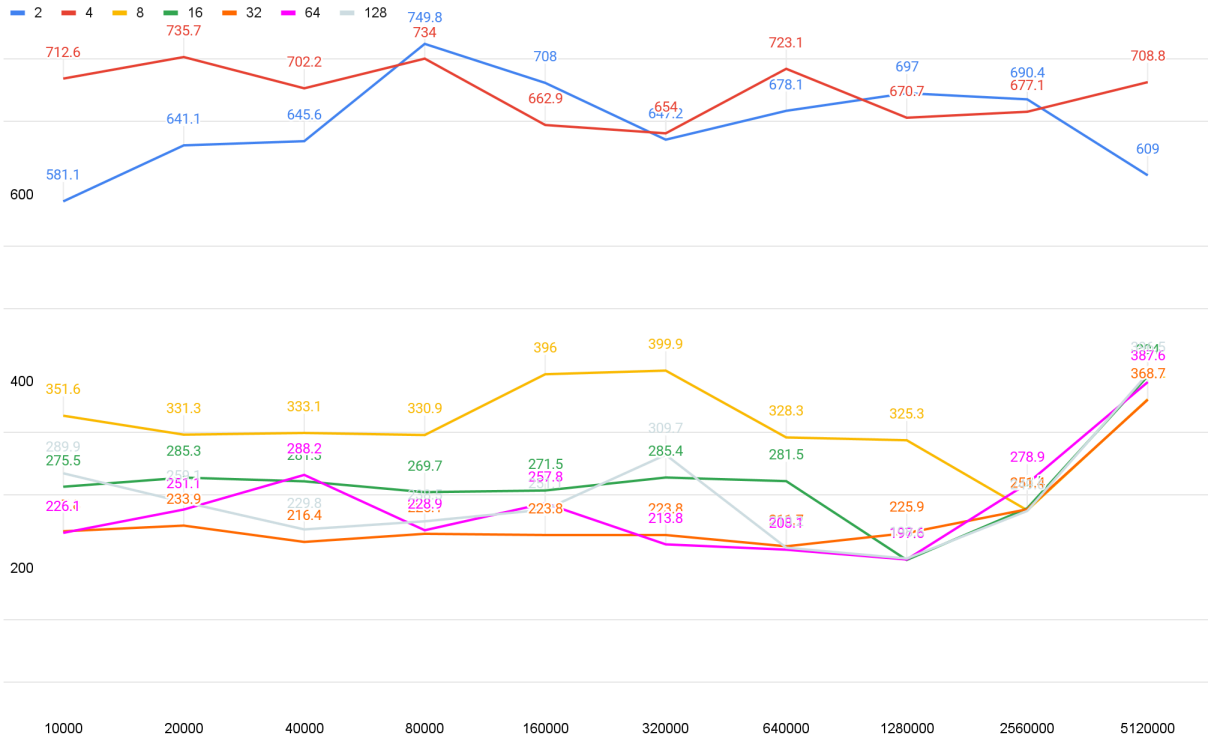
To better understand parallel efficiency, it's valuable to compare execution times with varying thread counts and cutoff values. By conducting multiple experiments and manipulating array sizes, cutoff values, and thread numbers, we observe that increasing parallelism generally leads to faster execution times. However, there are diminishing returns beyond a certain degree of parallelism.

For instance, when considering an array size of 5120000, optimal performance is achieved with a degree of parallelism of 16 or 32 and a cutoff value of 1280000. It's crucial to recognize that the optimal configuration can vary based on specific hardware, the software environment in use, and the characteristics of the input data. This variation underscores the importance of tailoring parallel processing strategies to the unique features of the computing system and data being processed.

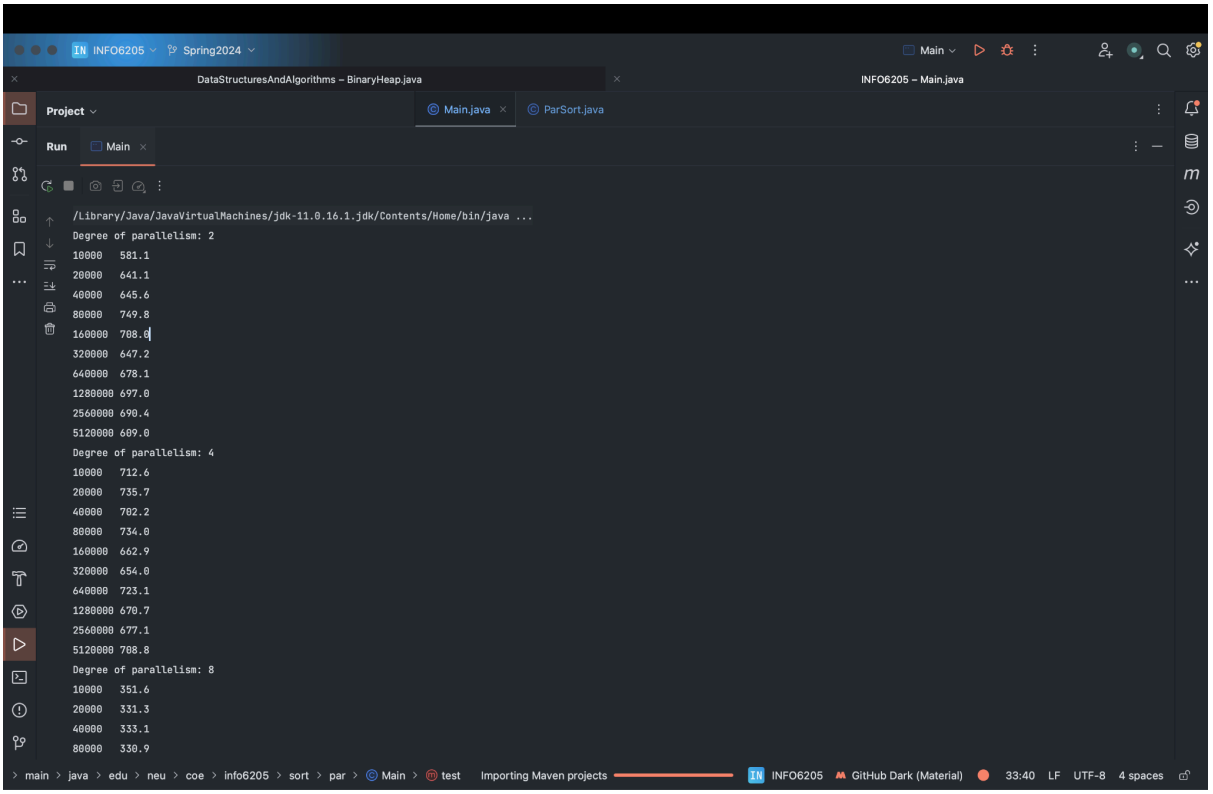
Evidence to support that conclusion:

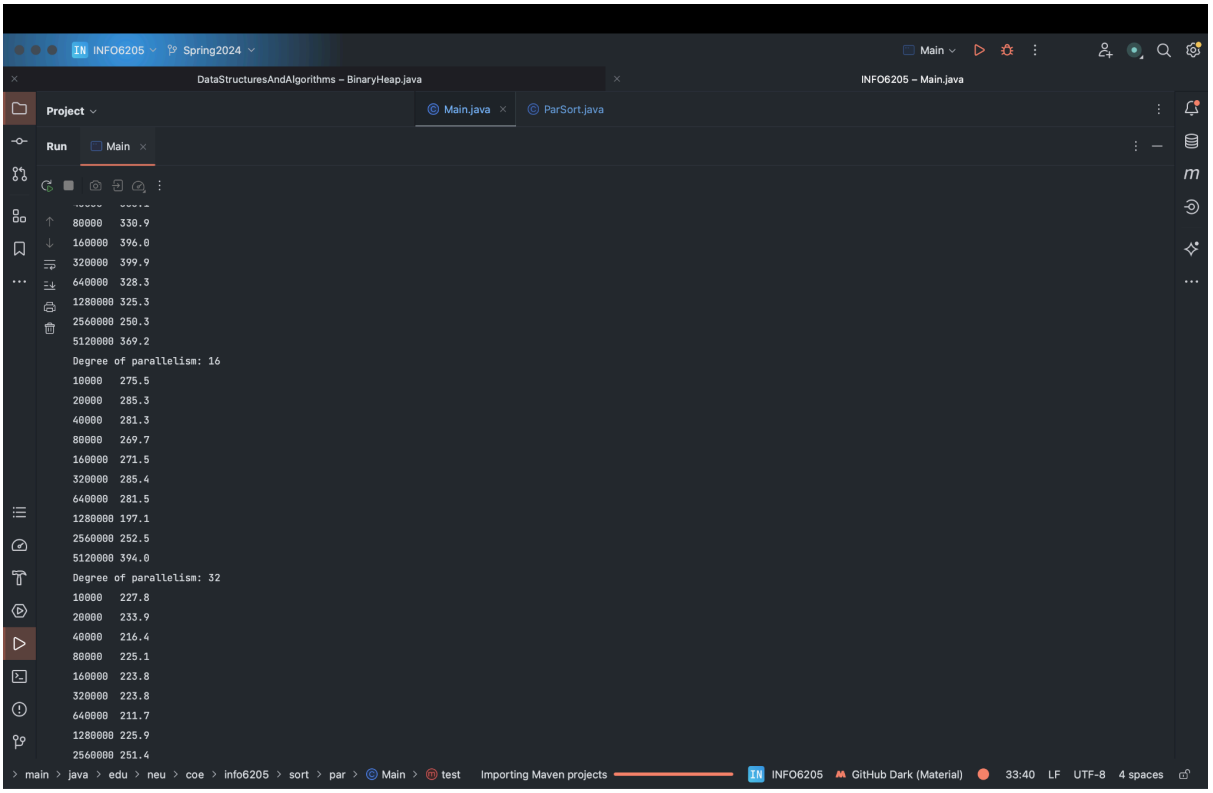
Google Sheet Link

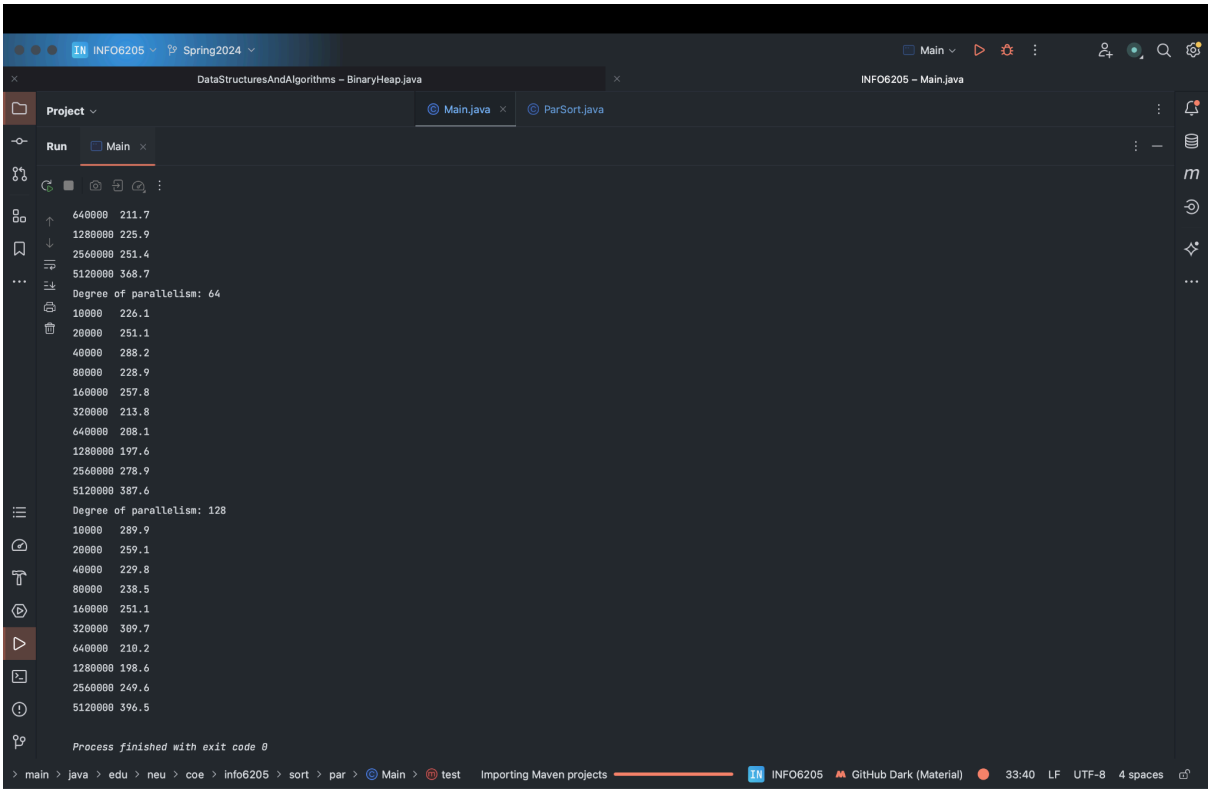
Graphical Representation:



Unit Test Screenshots and Output:




INFO6205
Spring2024
Main
DataStructuresAndAlgorithms – BinaryHeap.java
INFO6205 – Main.java
Project
Main.java
ParSort.java
Run
Main
80000 330.9
160000 396.0
320000 399.9
640000 328.3
1280000 325.3
2560000 250.3
5120000 369.2
Degree of parallelism: 16
10000 275.5
20000 285.3
40000 281.3
80000 269.7
160000 271.5
320000 285.4
640000 281.5
1280000 197.1
2560000 252.5
5120000 394.0
Degree of parallelism: 32
10000 227.8
20000 233.9
40000 216.4
80000 225.1
160000 223.8
320000 223.8
640000 211.7
1280000 225.9
2560000 251.4
main java edu neu coe info6205 sort par Main test
Importing Maven projects
INFO6205
GitHub Dark (Material)
33:40 LF UTF-8 4 spaces


INFO6205
Spring2024
Main
DataStructuresAndAlgorithms – BinaryHeap.java
INFO6205 – Main.java
Project
Main.java
ParSort.java
Run
Main
640000 211.7
1280000 225.9
2560000 251.4
5120000 368.7
Degree of parallelism: 64
10000 226.1
20000 251.1
40000 288.2
80000 228.9
160000 257.8
320000 213.8
640000 208.1
1280000 197.6
2560000 278.9
5120000 387.6
Degree of parallelism: 128
10000 289.9
20000 259.1
40000 229.8
80000 238.5
160000 251.1
320000 309.7
640000 210.2
1280000 198.6
2560000 249.6
5120000 396.5
Process finished with exit code 0
main java edu neu coe info6205 sort par Main test
Importing Maven projects
INFO6205
GitHub Dark (Material)
33:40 LF UTF-8 4 spaces