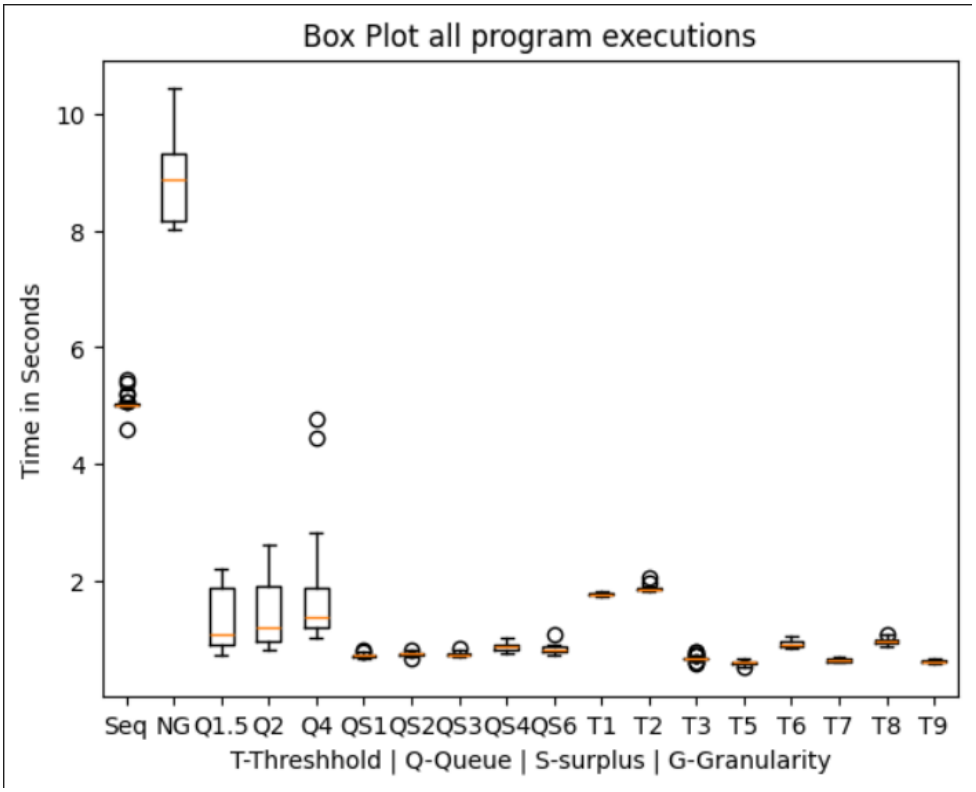
Assignment #2 – Coin Counter

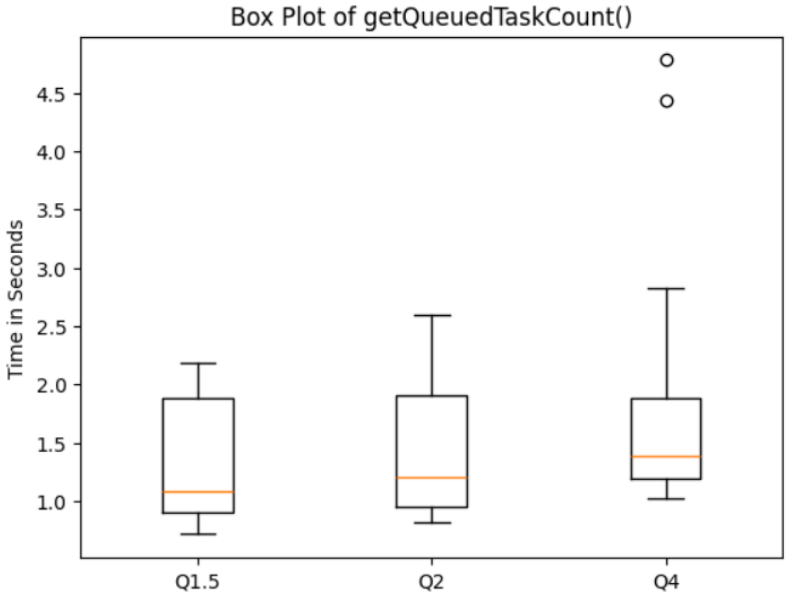
After analysing the sequential version of the code since it uses recursive decomposition to solve the coin problem, I've concluded that the best parallelization strategy to use is ForkJoin since of all the parallelization strategies we've learned this one is more suited for these cases. I also implemented in the code of the project the ForkJoinPool executor service to handle the tasks, being one of the objectives of the project to make the code as fast as possible and improve the performance. ForkJoinPool can execute many tasks in parallel, resulting in improved performance, besides that, it also implements a work-stealing algorithm to allocate tasks to worker threads ensuring that each worker thread always has a task to execute helping to minimize the overhead associated with task scheduling. As such, seemed to me a good programming practice to implement.

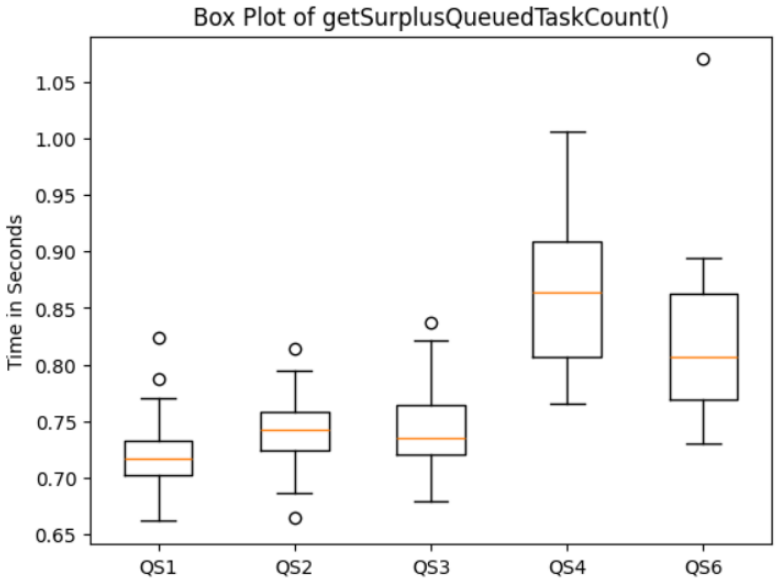
The first run of the parallel program without any granularity took longer to execute than the sequential version over 2 times so I had 3 different granularity methods to improve the performance. I used a threshold to determine de max index, anything over that uses the sequential method, getSurplusQueuedTaskCount() that if the number of tasks waiting is bigger than X it will execute the current instance in sequential and getQueueTaskCount() that if the number of tasks being used is bigger than X times the number of cores of the current instance it will also be done in sequential. To determine the best values for X and Threshold I simply tried different values and measured the performance after running the code, here are the results in a boxplot:

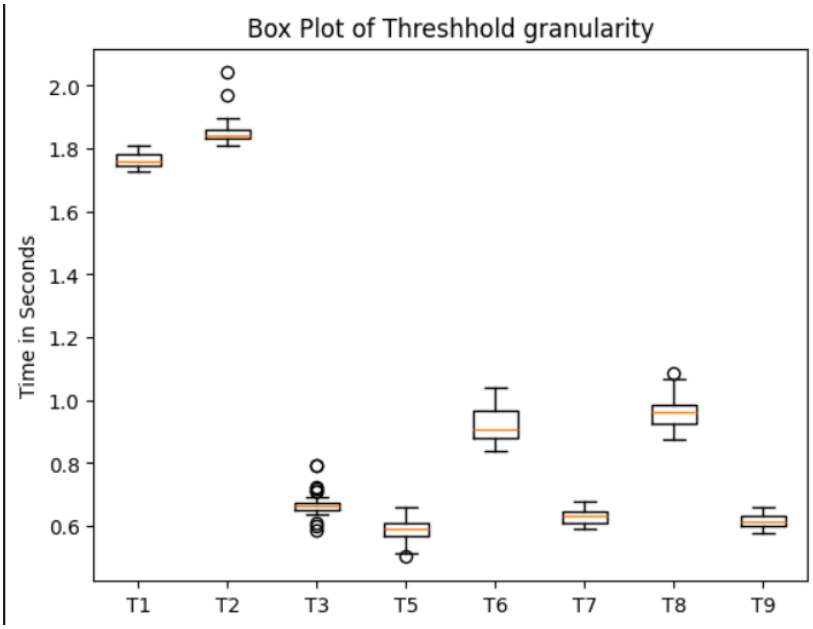
Note: The best performance was if the index was bigger or equal then 5 (Threshold = 5) with an average performance of 8.53 times better than the sequential. Also any values of Threshold bigger than 5 the odd values have better performance than the even values.

Pc specs: AMD Ryzen 7 3700Uwith Radeon Vega Mobile Gfx / 8gb RAM memory.




Box Plot of getQueuedTaskCount()
Time in Seconds
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Q1.5
Q2
Q4


Box Plot of getSurplusQueuedTaskCount()
Time in Seconds
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
QS1
QS2
QS3
QS4
QS6


Box Plot of Threshhold granularity
Time in Seconds
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
T1
T2
T3
T5
T6
T7
T8
T9