PROGRESS

- Choose 3 main sorting algorithms to run the main analysis.
- Implement code artefacts for each sorting algorithm.
- Implement code artefacts for data generators(random, all ones, descending and ascending).
- Run the code for each algorithm trying datasets containing from 1 to 10^8 integers.
- Register results in a document and start identifying trends between results.

BARRIERS

- Memory error when running 10^9 case on random numbers.
- Time of compilation for really big sets numbers especially with all ones, ascending and descending.
- Anomaly for insertion sort – average case(should be O(n^2)) is significantly faster than quicksort which complexity is O(nlogn) .Trying to prove the range of performance for quicksort with small datasets – timer may not be precise.

PLANNED NEXT STEPS

- Find a remote supercomputer to study how the algorithms behave for very large datasets.
- Perform a detailed analysis on the current results and illustrate their trends with graphics.
- Introduce hybrid algorithm between quicksort and insertion sort in the study and check how it compares to quicksort.
- Find the window of max efficiency for quicksort algorithm.
- Introduce a modified version of quicksort that sacrifices space over time by changing from recursive to iterative and pushing subarrays to a stack.