Joshan Bajaj
jbajaj1
Parallel Final Project Proposal

Last semester I worked on finding micro inversions within sequence reads for computational genomics. A micro-inversion is a section in the DNA where the expected nucleotides are replaced by the reverse complement of them. The solution I developed for my final project last semester was very slow and was clearly embarrassingly parallelizable. There are two main steps in my process: 1: I preprocess a large sequence into a string and a dictionary, 2) I use the preprocess dictionary to find areas with micro-inversions.

For my parallel final project, I am hoping to parallelize both parts; for the preprocess step, I'm hoping to have multiple processes read from different parts of the sequence at the same step to build my preprocess data structures. For the finding of the inversions, I'm hoping to find a way to execute the search process on multiple reads at the same time. I spoke with Disa, and he suggested I read GIL before deciding on my route of parallelization in Python.

## Edits based on feedback:

I am hoping to see how changing the genome size, read size, and inversion size affects the total run time for my programs when comparing my serial version to the parallel version(s) I will write.

In the serial version, I have recorded data showing that altering those three parameters change the total run time drastically. I am curious how changing those parameters will affect a parallel implementation, especially when compared to the serial version. I suspect that on a smaller genome size, the startup cost for multiple processes or threads will make a parallel implementation perform worse than the serial implementation for preprocessing. Additionally, I also suspect that there will be skew in my findInversions part of the project because some reads take longer than others, so I will need to find a way to change my current method so that reads that will take similar amounts of time to process are given to different threads/processes at the same time to avoid problems with skew. Perhaps finding a way to handle load/balancing would be ideal instead.