Michael Schultz
Boston Lighthouse Innovations Assessment
09/29/2020

Initially, I knew most of the challenge of this puzzle would come from refamiliarizing myself with Python. It has been a couple years since I used Python but have been using C++ primarily, which has a good number of similarities. Sorting the input data immediately stood out to me as an efficient way to pre-process the data. Knowing that once a read's start position was higher than a given loci position allowed read values to be eliminated since they could not possibly overlap the loci position. I decided to use a variable to keep track of the first match that was found through each iteration of the loop since there is the possibility that a value that matched a previous loci may match the following loci. This process of setting boundaries between which reads may apply to loci coverage greatly reduces the amount of total reads that need to be checked against a given loci position. Rather than requiring each loci position to be run against every read, only reads with start values less than or equal to the loci position are checked, and even some of those are discarded beforehand because of the position tracker.

There may be a way to avoid values within the adjusted range of each loop that were processed previously but did not overlap the previous loci position and so can not overlap the following loci either, but I believe that such implementation would likely lead to minimal benefit and make the code much more complicated. For the sake of preserving readability I decided to forego this possible optimization.

I had also considered maintaining a copy of the original loci list before sorting so that output could simply be appended to the existing file, rather than overwriting the loci file, so as to minimize file write time. I decided to not move forward with this idea since the list would remain unsorted. In the future this would allow the file to remain sorted for further use (assuming that would be helpful) or nearly sorted in case of editing of the file. While I am unaware of the underlying implementation of the list.sort() method, I assumed that having the input be already sorted or nearly sorted would save time sorting during processReads() and decided this may be a helpful optimization given the time complexity of sorting. If this is helpful, it may also be worth considering similarly rewriting the reads file since that file is much bigger, though I am unsure if the same reads would be reused.