Median String Problem

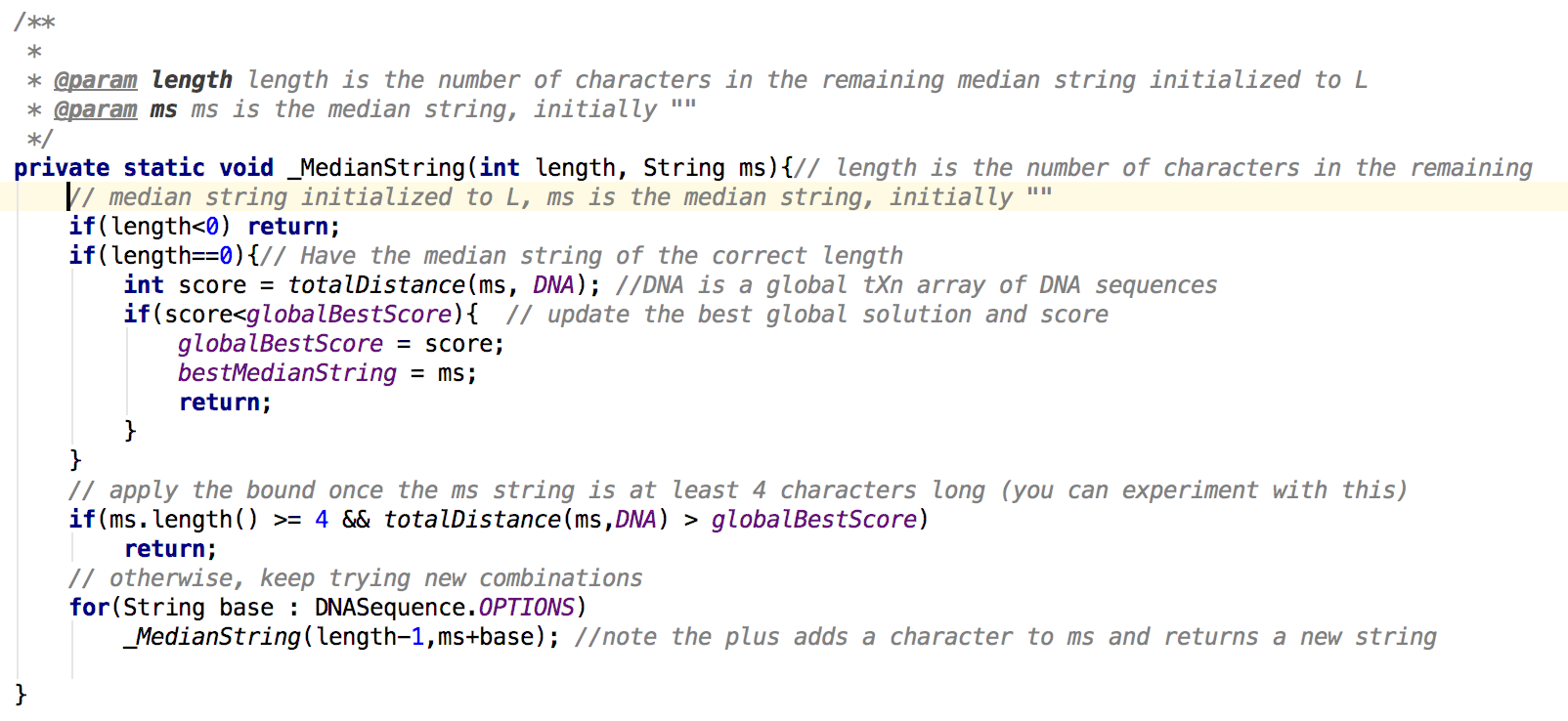
Assignment 3

Finding genetic diseases and traits in DNA is very important for a number of reasons. This helps us determine the proper treatments and let’s up develop GMOs, as well as many other advantages. DNA often times will encode motifs or patterns before genes, which is what helps us find those genes in the first place, hence an algorithm is needed to filter and find these motifs and subsequent genes.

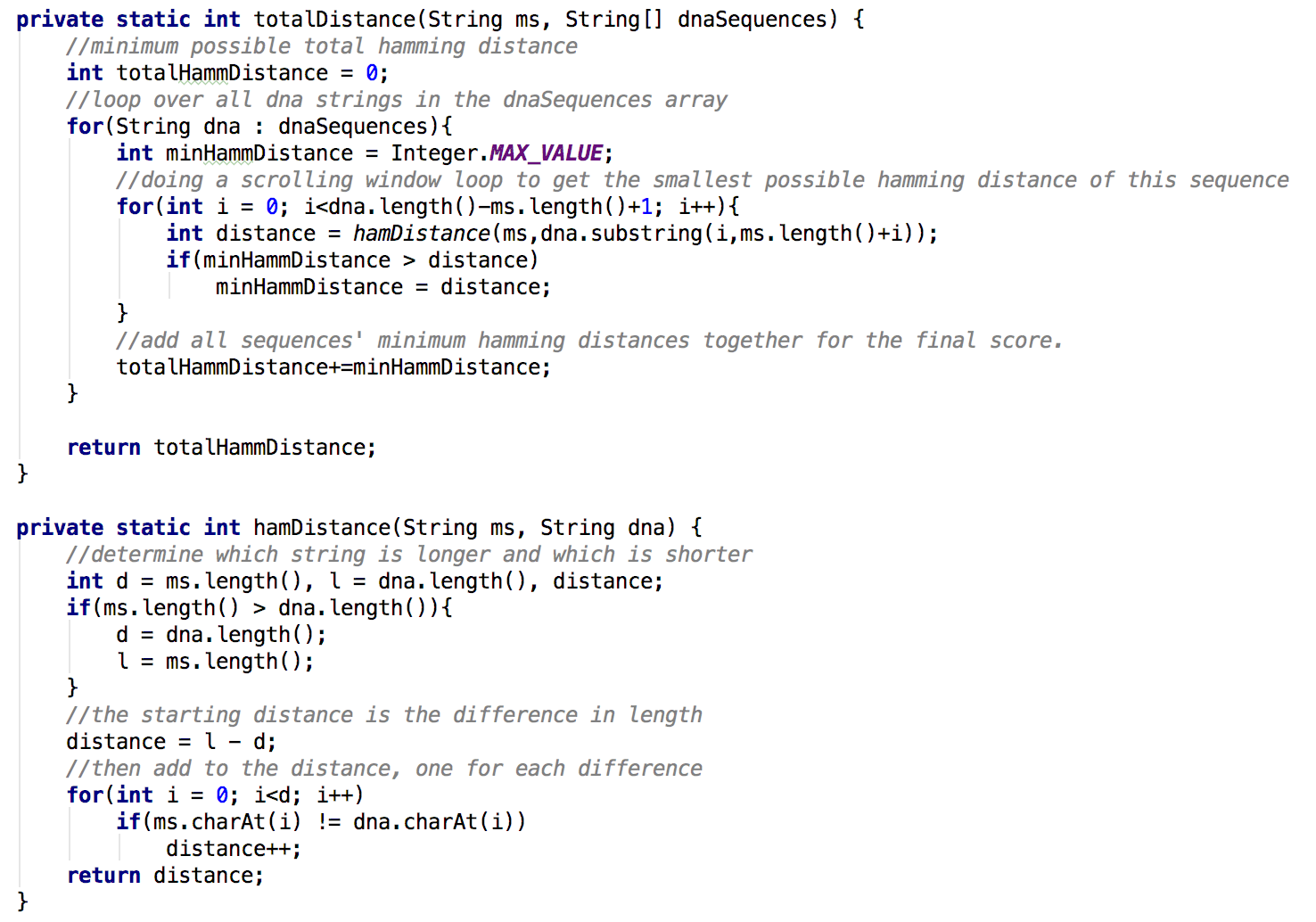
The problem at face value seems simple: Find a substring with the most occurrences. However, due to genetic mutations, this problem becomes much more difficult, changing from a simple n time algorithm to a polynomial algorithm since every combination of a pattern much be checked and compared.

# THE ALGORITHM

cs-5560/src/main/java/assignment3/MedianString.java



The following code snippet is the Java code used to solve this problem. Simply specify a desired motif length and then character by character, add to the motif and get a total hamming distance for each combination.



The above image illustrates the calculation required to find the hamming distance for each median string vs the dna sequences.

# THE TEST CASES

cs-5560/src/main/test/assignment3/MedianStringTest.java

This class contains the TestNG test cases required to verify that this algorithm works as expected. It generates random dna sequences and inserts random motifs of certain sizes and then gives them mutations as desired.

This class also contains a test to try and get a max mutation until failure, but with the parameters I’ve passed it, no amount of mutations could cause a failure. This I believe is due to factors such as dna length, number of motifs inserted, and length of the motif.

# TIMING STUDIES

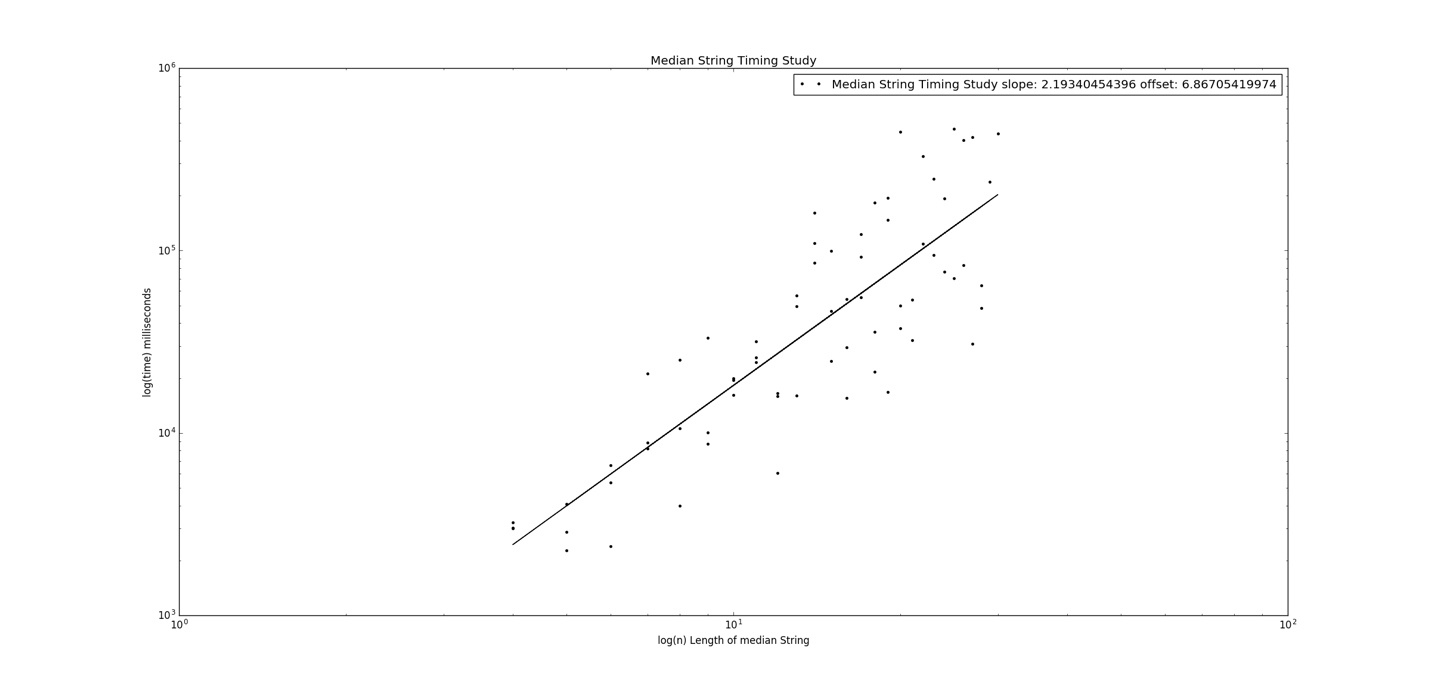
cs-5560/src/main/java/assignment3/MedianStringAnalysis.java

All results from this code is located in:

cs-5560/src/main/resources/assignment3/timing\_study.txt

Using python to generate the below graph, the file is located in:

cs-5560/src/main/python/assignment3/plotter.py



The above figure is the loglog graph of length of motif vs time in milliseconds to find that motif with all other factors held constant as: mutations = 0, dnaLength = 500, numMotifInserts = 5, numSequences = dnaLength.

This is the output of the results.

Median String Timing Study: [ 2.19340454 6.8670542 ]

Where the first number is the offset and the second number is the slope of the fitted line. This means that the algorithm from motif length 4 to 30 appears to about n^6.86, which is indeed polynomial.

# PROBLEM SOLUTION

cs-5560/src/main/java/assignment3/MedianStringAnalysis.java

All results from this code is located in:

cs-5560/src/main/resources/assignment3/promotorRegionsAnalysis.txt

In this file exists the solutions to all the dna sequences given and asked to find motifs of length 12.