Jonathan Arndt 8/1/16

Median String Problem

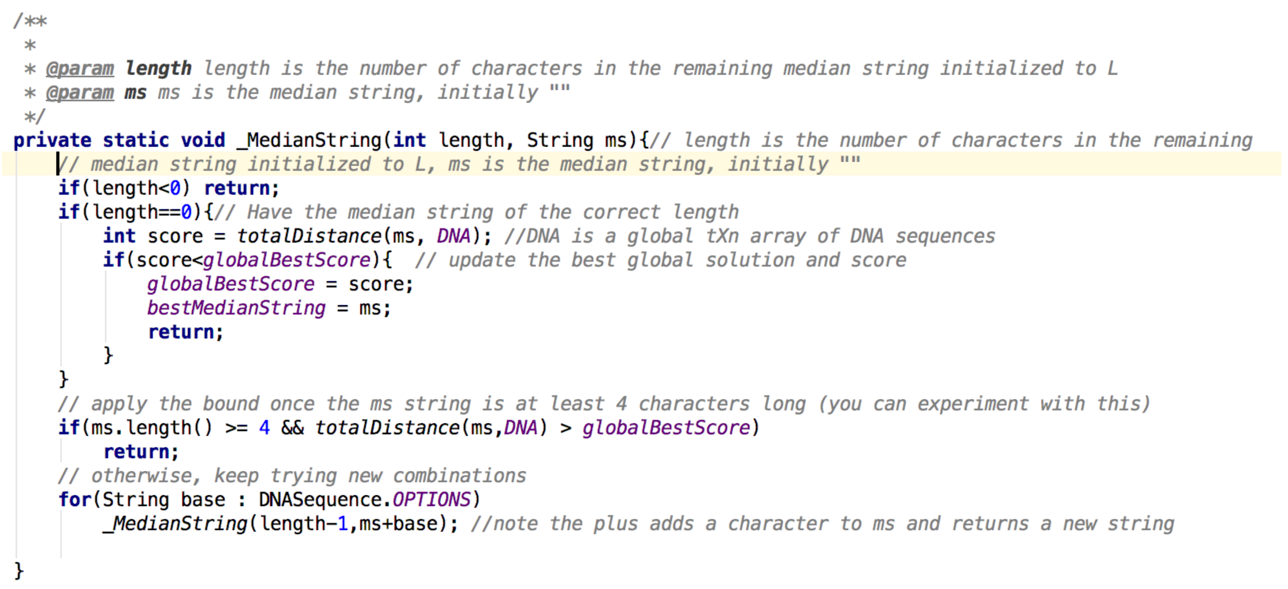
Assignment 3

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

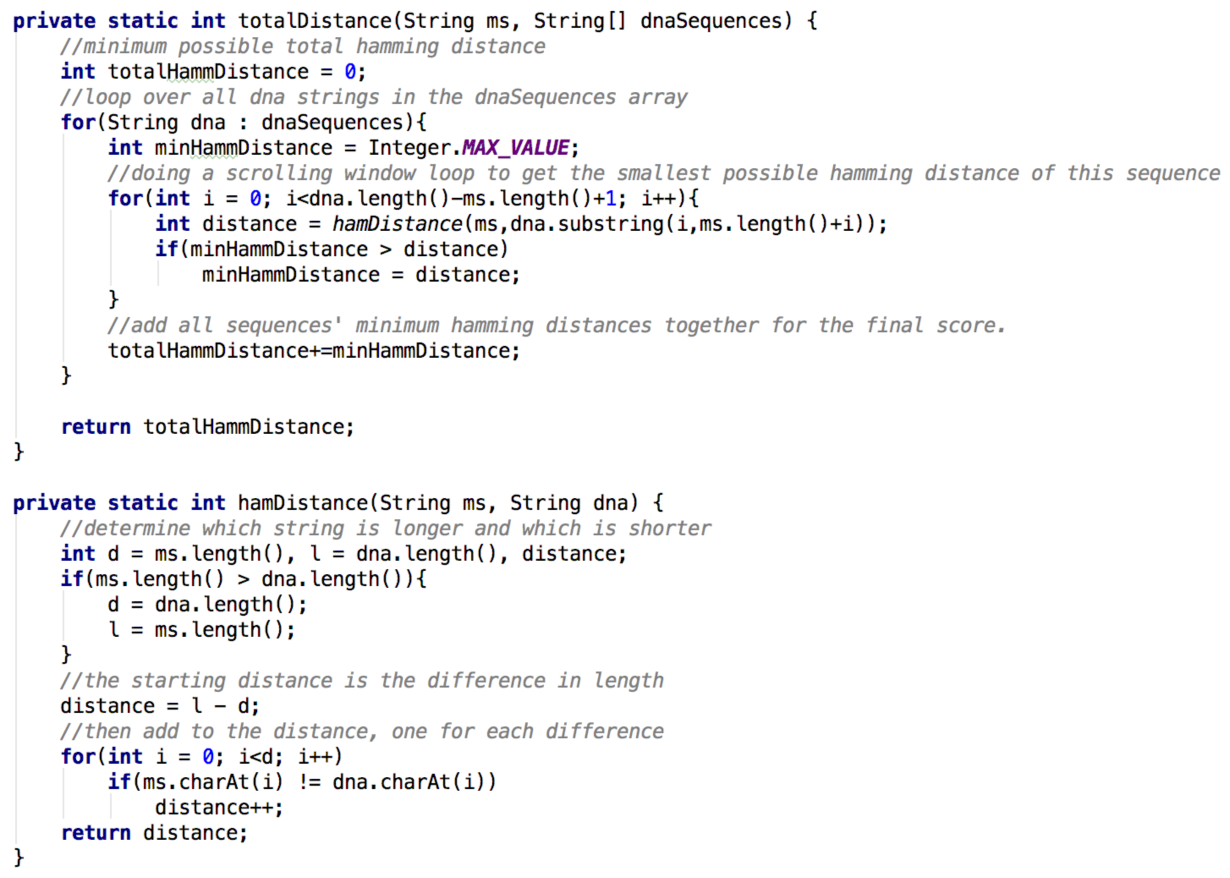
cs-5560/src/main/java/assignment3/MedianString.java



```java
/**
 *
 * @param length length is the number of characters in the remaining median string initialized to L
 * @param ms ms is the median string, initially ""
 */
private static void _MedianString(int length, String ms){// length is the number of characters in the remaining
    // median string initialized to L, ms is the median string, initially ""
    if(length<0) return;
    if(length==0){// Have the median string of the correct length
        int score = totalDistance(ms, DNA); //DNA is a global tXn array of DNA sequences
        if(score<globalBestScore){  // update the best global solution and score
            globalBestScore = score;
            bestMedianString = ms;
            return;
        }
    }
    // apply the bound once the ms string is at least 4 characters long (you can experiment with this)
    if(ms.length() >= 4 && totalDistance(ms,DNA) > globalBestScore)
        return;
    // otherwise, keep trying new combinations
    for(String base : DNASequence.OPTIONS)
        _MedianString(length-1,ms+base); //note the plus adds a character to ms and returns a new string

}
```

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.



```java
private static int totalDistance(String ms, String[] dnaSequences) {
    //minimum possible total hamming distance
    int totalHammDistance = 0;
    //loop over all dna strings in the dnaSequences array
    for(String dna : dnaSequences){
        int minHammDistance = Integer.MAX_VALUE;
        //doing a scrolling window loop to get the smallest possible hamming distance of this sequence
        for(int i = 0; i<dna.length()-ms.length()+1; i++){
            int distance = hamDistance(ms,dna.substring(i,ms.length()+i));
            if(minHammDistance > distance)
                minHammDistance = distance;
        }
        //add all sequences' minimum hamming distances together for the final score.
        totalHammDistance+=minHammDistance;
    }

    return totalHammDistance;
}

private static int hamDistance(String ms, String dna) {
    //determine which string is longer and which is shorter
    int d = ms.length(), l = dna.length(), distance;
    if(ms.length() > dna.length()){
        d = dna.length();
        l = ms.length();
    }
    //the starting distance is the difference in length
    distance = l - d;
    //then add to the distance, one for each difference
    for(int i = 0; i<d; i++)
        if(ms.charAt(i) != dna.charAt(i))
            distance++;
    return distance;
}
```

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.

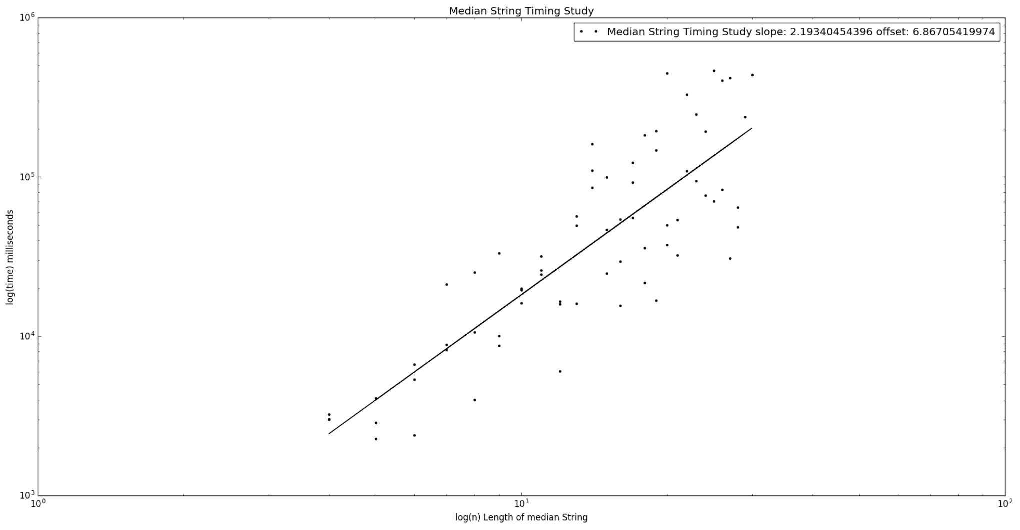
This class also contains a test to try and get a max mutation until failure, but with the parameters I've passed it, no amount of mutations could cause a failure. This I believe is due to factors such as dna length, number of motifs inserted, and length of the motif.

Jonathan Arndt 8/1/16

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.

Jonathan Arndt 8/1/16

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.