1. I checked all the classes with the test harness and couple of tests that I made.

I have a few bugs, which I think they all come from GenomeMatcherImpl class.

First of all FindRelatedGenomes() doesn’t work.

Also, find GenomeWithThisDNA has a few bugs when I tried it with the test harness.

1. When I loaded all the files and was looking for exact matches with ACGAATCACGTGCGAGA Which should return two matches; however, my code would find another match:

Here is what my code would return:

3 matches of ACGAATCACGTGCGAGA found:

1.length 17 position 568 in NZ\_AOJK01000080.1 Halorubrum californiensis DSM 19288 contig\_80, whole genome shotgun sequence

2.length 17 position 661 in NZ\_AOJK01000080.1 Halorubrum californiensis DSM 19288 contig\_80, whole genome shotgun sequence

3.length 12 position 1977 in NZ\_FOCX01000065.1 Halorientalis persicus strain IBRC-M 10043, whole shotgun sequence

I am pretty sure that the Trie class works properly since I tested it with multiple test cases

1. Also, I think when I look for exactMatches==false, sometimes doesn’t work properly. For instance, if I pass in a sequence with two mismatches, it should return the fragment of the sequence before finding the second mismatch based on my understanding of the assignment. Mine would not do that, it says no matches found.
2. Most of the methods follow the Big-O except findGenomeWithThisDNA(), my method has a O(Q\*S\*T), where Q is the time it takes to go through all elements that Trie::find() return, S is number of Genomes in the library, and T is how long does it take to extract a fragment

findRelatedMatches()-> O(T\*Q\*X\*N) where T is number of objects in the library, Q is the length in DNA bases of the query sequence (e.g., 3 million bases), and X is the function in the big-O of findGenomesWithThisDNA() method, and N in number of structures in vector<GenomeMatch> result sequ

33 3,3.

Trie:: find()

First compare the first character of string key with the first one in the node, if they don’t match, return despite being a true match or exact match

If the first letter matches, call the helper function which return a vector<ValueType> tempVector. The helper function would get the point to the current node, key string, Boolean for exactMatch, and vector<ValueType> tempVector .

In helper function, if the current node is NULL return temp

If key has become empty due to finding the matches, push the values in that node into tempVector.

If none of the above met, for each child node of Trie

If the exactMatch is true

Check the current child with the first letter of the key, if matched,

Call the helper again by passing the matching child pointer And moving one character in key, pass true for exactMatch

Otherwise check the next child node

If the exactMatch is false

If the first char of the key matched with the current child pointer

Call helper by passing that child node, moving one char in key, pass false for the exact match

If the first char didn’t match with the current child, found a mismatch

Call helper by passing the current child, moving one char forward with key, and true for exact match, so no more mismatch is allowed

Return the tempVector

GenomeMatcherImpl::findGenomeWithThisDNA()

Note: My Trie::find() would return a vector of int which is position in which the match was found in the TireNode

This function would return false if:

1. fragment's length is less than minimumLength, or

2. minimumLength is less than the minSearchLength value passed to the

GnomeMatcher's constructor, or

3. There are no matches between fragment and any segment of any genome in the

GenomeMatcher object's library

Otherswise call find by passing the first minumumSearchLength char of the fragment, this how it got insert in Trie, so find would return the position of the subfragment if found

Then for each positon found, call the helper function which pass the fragment, the the current position found, minmumLength, exactMatch, and vector struct of DNAMatch. Stor the result of this call in that vector of struct.

In the helper check genome in the library,

For each genome extract sub fragments starting with the position that has been passed and variant lengths depending of the minimumLenght pass and length of the fragment. If match found put that in the longest found.

Check for bigger fragments if found a match, store in the longest.

For the current genome in the library, store the name, position found, and length of the longest found in the struct and push it into the vector of the DNAMatch Struct