CSCI-1200 Data Structures — Fall 2015
Homework 7 — MiniBlast Maps

BLAST (**B**asic **L**ocal **A**lignment **S**earch **T**ool) is an algorithm for fast, efficient searching of databases of proteins, DNA, and RNA sequences. A BLAST search allows a researcher to compare a query sequence against a library of sequences. For example, after sequencing a gene from a newly discovered bacteria, BLAST can be used to determine if the gene is similar to any genes in known bacteria. You can read more about BLAST at https://en.wikipedia.org/wiki/BLAST or try it out at http://blast.ncbi.nlm.nih.gov/Blast.cgi.

In this homework assignment, we create an extremely simplified version of BLAST. Our version works in a manner somewhat similar to BLAST, but the BLAST algorithm is sophisticated and BLAST software is highly optimized. BLAST is used to regularly search Genbank, a repository of sequence data containing over 1 trillion bases (letters) of assembled sequence data. Online searches are typically returned in less than a minute. We won't try anything quite so ambitious. Our version of BLAST, miniblast, will search small genome files with query strings in a DNA alphabet (A,C,G,T).

The genome files that we will search will consist of a number of lines, each line containing letters from the DNA alphabet. Here are the first few lines of the *genome_small.txt* file:

```
$ head genome_small.txt
TAATGACCTAAATAATCTAAACAAAAGGAAGAGAGATAGTCCGGATTACC
TGGGACATGGAAAACCCTCCTTTCTCTCATCAGCTTCCCACCCCACCTCT
GCCCAGCGCTAATCATGATTTAATAGCCTTCCTTAATCACTTACTCTGTT
TGCTGCTTCATCTAAAAACTTAAGATGCTCTGGGTTAGATCACAGTCTAA
CTCATCACAAATGGATAGAACGACCTGGTAGTTTTCCAGATTTCCATTGT
CCAAACTAATCAGCCAACACACTCAATGATGCACATTATTTTCCACGTAT
ATGGCCTTAGAGATGGGACTAAAAGTCCCGACTGTACTGAGGATGTTTGA
CAGGTTTTGCCATTCTAACTGCTAGTGCTGTGTAATGTGGCTAGGAAGAA
GCAAGGAACAGAGAGATACAGATATGATTTCCTGGGACCAGCTATAGGAG
AGATTCTTCAATTACATCATCTTTGCTCATCCCAAACACCTTGACAAGTA
```

The genome data above is a small region from human chromosome 18.

The queries will be also strings from the DNA alphabet. A typical query could look like:

```
CTCATCACAAATGGATAGAACGACCTGGTA
```

This query can be located at the start of the 5th line.

The strategy that we will employ is to index the genome file with a series of *k-mers*. A *k-mer* is a sequence of k letters from the DNA alphabet, where k is an integer.

We index the genome by building an **std::map** with the *k-mers* as the key and the *k-mer* positions as values. We build the index by iterating through the genome sequence with a series of overlapping windows of length k. That is, the first *k-mer* is the genome sequence from 0 to k-1; the second is the sequence from 1 to k, etc. When indexing, we have to allow for the fact that the *k-mer* may appear multiple times in the genome.

When searching a biological sequence database, it's often the case that we do not find an exact match to a query string. Miniblast will process queries of varying lengths and allow for mismatches between the genome and query string. To search the genome, miniblast uses the first k letters of the query string as a seed. It is important that searching the database for the initial seed be efficient. We require that the database can be searched in *O(log L)* time, where L is the number of keys in the database. If the seed can be found in

the index, the program should try to extend the match by adding letters from the indexed genomic position until the full query is matched or the allowed number of mismatches is exceeded. For simplicity we require that the seed be an exact match. The mismatches may occur anywhere after the seed in match string.

Please carefully read the entire assignment before beginning your implementation.

Input/Output & Basic Functionality

The program should read a series of commands from STDIN and write responses to STDOUT. Sample input and output files have been provided. You can redirect the input and output to your proram using the instructions in the section **Redirecting Input & Output** found at http://www.cs.rpi.edu/academics/courses/fall15/csci1200/other_information.php

Your program should accept the following commands:

- *genome filename* - Read a genome sequence from *filename*. The genome file consists of lines DNA characters.
- *kmer k* - k is an integer. The command will index the genome sequence with k-*mers* of length k.
- *query m query_string* - Search the genome for *query_string* allowing for m mismatches.
- *quit* - Exit the program.

Here is some sample input, showing the commands:

```
genome genome_small.txt
kmer 10
query 2 TATTACTGCCATTTTGCAGATAAGAAATCAGAAGCTC
query 2 TTGACCTTTGGTTAACCCCTCCCTTGAAGGTGAAGCTTGTAAA
query 2 AAACACACTGTTTCTAATTCAGGAGGTCTGAGAAGGGA
query 2 TCTTGTACTTATTCTCCAATTCAGTCACAGGCCTTGTGGGCTACCCTTCA
query 5 TTTTTTTTTTTTTTTCTTTTTT
quit
```

For output, the program will report the query, and if matches are found, the genome position (positions start at 0) of the match, the number of mismatches between the genome and the query string, and the genome sequence matching the query. If a match can't be found, the program will report "No Match".

The corresponding output to the input above should be:

```
Query: TATTACTGCCATTTTGCAGATAAGAAATCAGAAGCTC
504 0 TATTACTGCCATTTTGCAGATAAGAAATCAGAAGCTC
Query: TTGACCTTTGGTTAACCCCTCCCTTGAAGGTGAAGCTTGTAAA
5002 2 TTGACCTTTGGTTAACCAATCCCTTGAAGGTGAAGCTTGTAAA
Query: AAACACACTGTTTCTAATTCAGGAGGTCTGAGAAGGGA
4372 0 AAACACACTGTTTCTAATTCAGGAGGTCTGAGAAGGGA
Query: TCTTGTACTTATTCTCCAATTCAGTCACAGGCCTTGTGGGCTACCCTTCA
No Match
Query: TTTTTTTTTTTTTTTCTTTTTT
4428 0 TTTTTTTTTTTTTTTCTTTTTT
4429 3 TTTTTTTTTTTTTTTCTTTTTTG
4430 4 TTTTTTTTTTTTTTCTTTTTTGA
4431 5 TTTTTTTTTTTTTCTTTTTTGAG
```

You are not explicitly required to create any new classes when completing this assignment, but please do so if it will improve your program design. We expect you to use `const` and pass by reference/alias as appropriate throughout your assignment.

Order Notation

In your `README.txt` file, report the time and space order of your implementation for building the index for a genome of length **L**. Does the *k-mer* size, **k**, and the average number of locations, **p**, where the key is found affect your answer? What is the order time notation for matching a query of length **q** in a genome of length **L** when the key size is **k** and the key is found at **p** different genomic positions.

Extra Credit

Add a new command to implement the database using one of the other data structures that we have covered so far in the course: vectors, lists, arrays etc. Compare the performance your alternative method to the homework method by making a table of run times for each of the genomes and query sets provided with the homework and compare the times to build the index and the times to process the queries. Document any new commands you have added in your README file.

Submission

Use good coding style and detailed comments when you design and implement your program. Please use the provided template `README.txt` file for any notes you want the grader to read, including work for extra credit. You must do this assignment on your own, as described in the "Collaboration Policy & Academic Integrity". If you did discuss the problem or error messages, etc. with anyone, please list their names in your `README.txt` file.