

Homework #3

Due October 18th, 11:59pm

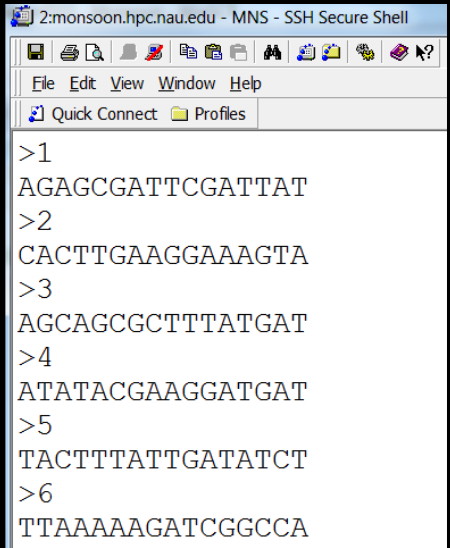
Each homework submission must include:

- An archive (.zip or .gz) file of the source code containing:
 - The makefile used to compile the code on Monsoon **(5pts)**
 - All .cpp and .h files **(5pts)**
 - A readme.txt file outlining all modules (if any) needed for the execution of the code and the exact command lines needed to answer homework's questions **(5pts)**
- A full write-up (.pdf or .doc) file containing answers to homework's questions **(5pts)** – screenshots of code output are ok.

The source code must follow the following guidelines:

- No external libraries that implement data structures discussed in class are allowed, unless specifically stated as part of the problem definition. Standard input/output and utilities libraries (e.g. math.h) are ok.
- All external data sources (e.g. input data) must be passed in as a command line argument (no hardcoded paths within the source code).
- Solutions to sub-problems must be executable separately from each other. For example, via a special flag passed as command line argument **(5pts)**

- For this homework, you will use the High Throughput Sequence reads dataset located on Monsoon:
/common/contrib/classroom/inf503/hw3_dataset.fa
(see insert). Note that the header (line with ">" at the beginning) is in a format that is different from HW#1 and HW#2. For this assignment it will be safe to discard all headers.
- You will also need to use the genome sequence for Bacillus anthracis bacterium (same as HW#2) located at:
/common/contrib/classroom/inf503/test_genome.fasta
 - This genome file contains a header (denoted by '>') followed by ~5.2 million characters of its genomic code (alphabet A, C, G, T)
 - Please be aware that the genome is spread across multiple lines of the file (see insert)

A screenshot of an SSH terminal window titled "2:monsoon.hpc.nau.edu - MNS - SSH Secure Shell". The window has a menu bar with "File", "Edit", "View", "Window", and "Help". Below the menu bar are tabs for "Quick Connect" and "Profiles". The terminal displays six lines of sequence data, each preceded by a header line starting with ">". The sequence data is as follows:

```
>1
AGAGCGATTCGATTAT
>2
CACTTGAAGGAAAGTA
>3
AGCAGCGCTTTATGAT
>4
ATATACGAAGGATGAT
>5
TACTTTATTGATATCT
>6
TTAAAAAGATCGGCCA
```

Problem #1 (of 2): Fun with bit arrays

Create a class called ***FASTAreadset_BitArray***. The purpose of the class will be to contain a FASTA read set (similar to homeworks #1 and #2) and all of the functions needed to operate on this set. Use the **bit array** data-structure to store the genomic sequences of the given read dataset (hint: use array of characters – char[] for your bit array). You will need to read in the genomic sequence fragments (feel free to ignore / discard all headers), convert them to a radix notation number (hint: use a double of unsigned int), and flip the proper bit in the bit array (hint: you will need to use the modulo operator to get the proper byte in the array and then use bit shifting operators to flip the right bit). If the bit is already “ON” (i.e. you are seeing a duplicate fragment), you’ll need to record this ‘collision’. Do not ‘re-flip’ the bit.

At minimum, the class must contain:

- A constructor
- A destructor
- A function to query the bit array
- Private variables to store the total number of collisions and elements stored in bit array (flipped bits)

For the following questions use the High Throughput Sequence reads dataset located on Monsoon: ***/common/contrib/classroom/inf503/hw3_dataset.fa***. Here it is safe to assume that all sequence fragments are exactly 16 characters long.

- A. What is the size of your bit array in bytes and bits?
- B. How many sequence fragments did you query?
- C. How many unique sequences did you observe (number of “ON” bits)?

Remember that you won’t be able to address bit directly and must instead address closest byte and access the bit from there.

Problem #2 (of 2): The hash table

Create a class called ***FASTAreadset_hash***. Use the **hash table** data-structure to store the genomic sequences of the given read dataset (hint: you will need to provide the size of the hash table). If you have a duplicate sequence fragment or a duplicate hash value, use chaining method to resolve collisions. Resizing is optional - you can hard-code the proper hash table size through the constructor. Use Radix / division scheme for hash function implementation.

At minimum, the class must contain:

- A constructor
- A destructor
- A function to query the hash table

For the following questions use the High Throughput Sequence reads dataset located on Monsoon: ***/common/contrib/classroom/inf503/hw3_dataset.fa***. Here it is safe to assume that all sequence fragments are exactly 16 characters long. You will also need the Bacillus anthracis genome sequence located at: ***/common/contrib/classroom/inf503/test_genome.fasta***

- A. Set the size of your hash table to 10,000 elements ($m = 10,000$). How many collisions did you observe while populating the hash (reading the read file)? How long did it take you to read the sequence fragment file?
- B. Set the size of your hash table to 100,000 elements ($m = 100,000$). How many collisions did you observe? How long did it take you to read the sequence fragment file?
- C. Set the size of your hash table to 10,000,000 elements ($m = 10,000,000$). How many collisions did you observe? How long did it take you to read the sequence fragment file? Do the results of A, B, and C make sense to you (explain)?
- D. Set the hash size to 10,000,000. Iterate through all 16-mers found in the genome and use them to query the read set. How many genome 16-mer fragments were found in your read set? How long it take to complete the entire search process (all 16-mers)?