Homework #2

100 Points Possible

3/5/2024

Attempt 1

In Progress

NEXT UP: Submit Assignment

Add Comment

Unlimited Attempts Allowed

2/20/2024 to 5/3/2024

Details

Homework #2

Due March 5th, 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 full write-up (.pdf of .doc) file containing answers to homework's questions **(5pts)**, including the exact command line needed to execute every subproblem of the homework

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 **(5pts)**.
- 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 *query dataset* located on Monsoon: **/common/contrib/classroom/inf503/human_reads.fa**.

For this homework, you will also need to use the *subject dataset* (human genome assembly that you used in HW#1). Recall that it is located at: **/common/contrib/classroom/inf503/genomes/human.txt**

- This file contains multiple scaffolds that comprise the human genome
- The genome is in FASTA format (see insert)

- The headers are unique and always begin with the “>” character. These can be discarded for this homework. Each line of genome file is exactly 80 characters long (plus carriage return character)
- The genomic sequences consist of the following alphabet {A, C, G, T, N}

Problem #1 (of 1)

Create a class called ***Queries_AR***. The purpose of the class will be to contain a dataset of genomic sequences (queries) and all of the functions needed to operate on this set. Use the **2D array** data-structure to store the genomic sequences of the dataset. For this assignment, you can completely disregard the headers of the sequence fragments. At minimum, the class must contain **(15pts):**

- A default constructor (that zeroes everything out)
- At least one custom constructor (e.g. one taking a file path or ifstream as input)
- A function to read the query dataset file
- A search function designed to find a sequence fragment within class’s data
- A function to sort the fragments of the Queries_AR object
- A destructor

A. (30 pts) Read in the entire *query dataset* and store it in an instance of the Queries_AR class. Read in the entire *subject dataset* into a single, concatenated character array (same way you did it in HW#1). Implement a search function which would search for 32 character fragments of the subject sequence within the Queries_AR object. The search function should return the location (index) of the match OR a negative value if a ‘hit’ was not found. Iterate through 32-character long fragments of the *subject dataset*, searching for each one in the *query dataset*.

- How long did it take you to search for the first 10K, 100K, and 1M 32-character long fragments of the *subject dataset* within the *query dataset*?
- How long would it take to search for every possible 32-character long fragment of the *subject dataset* within the *query dataset*? Please note that depending on the efficiency of your algorithm, this step may take a long time. If the total time is greater than 24 CPU hours, provide an estimate rather than exact number.
- Print the 20 query fragments found within the *subject dataset* that have the largest indices (i.e. found later in the subject) for the first 10K, 100K, and 1M 32 character long fragmetns.

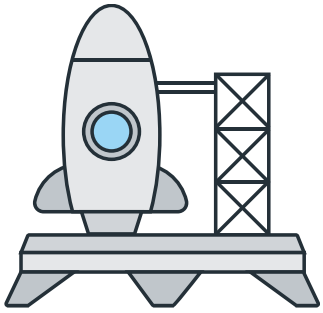
B. (30 pts) Read in the entire *query dataset* and store it in an instance of the Queries_AR class. Sort all character fragments in alphabetic (lexographic) order. Any sorting algorithm will do. Read in the

entire *subject dataset* into a single, concatenated character array (same way you did it in HW#1). Implement a search function which would search for 32 character fragments of the subject sequence within the Queries_AR object. The search function should return the location (index) of the match OR a negative value if a 'hit' was not found. Iterate through 32-character long fragments of the *subject dataset*, searching for each one in the *query dataset*.

- How long did it take you to search for the first 10K, 100K, and 1M 32-character long fragments of the *subject dataset* within the *query dataset*?
- How long would it take to search for every possible 32-character long fragment of the *subject dataset* within the *query dataset*? Please note that depending on the efficiency of your algorithm, this step may take a long time. If the total time estimate is greater than 24 CPU hours, provide estimate rather than exact number.
- Print the 20 query fragments found within the *subject dataset* that have the largest indices (i.e. found later in the subject) for the first 10K, 100K, and 1M 32 character long fragmetns.

Choose a submission type

Upload | Canvas Studio | More



Choose a file to upload

or

Webcam Photo

Canvas Files

Submit Assignment