INF-503

Table of Contents

[Homework – 2 2](#_Toc170940765)

[Problem #1 : 2](#_Toc170940766)

[Code Execution: 8](#_Toc170940767)

[**Part A: Searching Fragments Using Linear Search** 9](#_Toc170940768)

[**Part B: Sorting and Searching with Binary Search** 9](#_Toc170940769)

[Video Presentation Link 10](#_Toc170940770)

# Homework – 2

## Problem #1 :

(20 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 5k, 10K, 100K, and 1M 32-character long  
fragments of the subject dataset within the query dataset?

1. For first 5k records

A screenshot of a computer

Description automatically generated

1. For first 10k records

A screenshot of a computer

Description automatically generated

1. For first 100k

A screenshot of a computer

Description automatically generated

1. For 1M records

For 5000 records 2977.915 sec / 5000 records = 0.5955 sec/rec

For 10000 records 10765.00 / 10000 records = 1.07 sec / rec

For 100000 records 42388.635 sec / 100000 records = 0.42 sec / rec

Avg time per record = average of all three = 0.69 sec/rec

For 1000000 records = 0.69 \* 1000000 = 690000seconds

● 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 an exact number.

Time to run for total genome (3057186663)

3057186663 \* 0.69 = 210,94,58,797.47 seconds

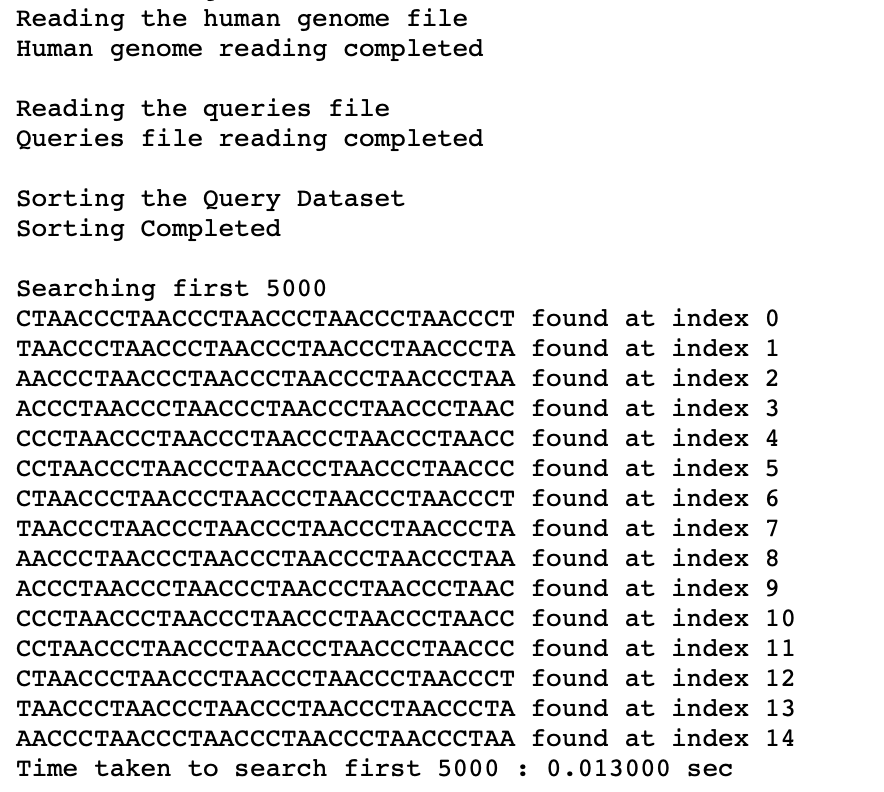
● Print the first 15 fragments of the subject dataset along with it’s indices that you found  
within the Query AR object (if any)

A screenshot of a computer

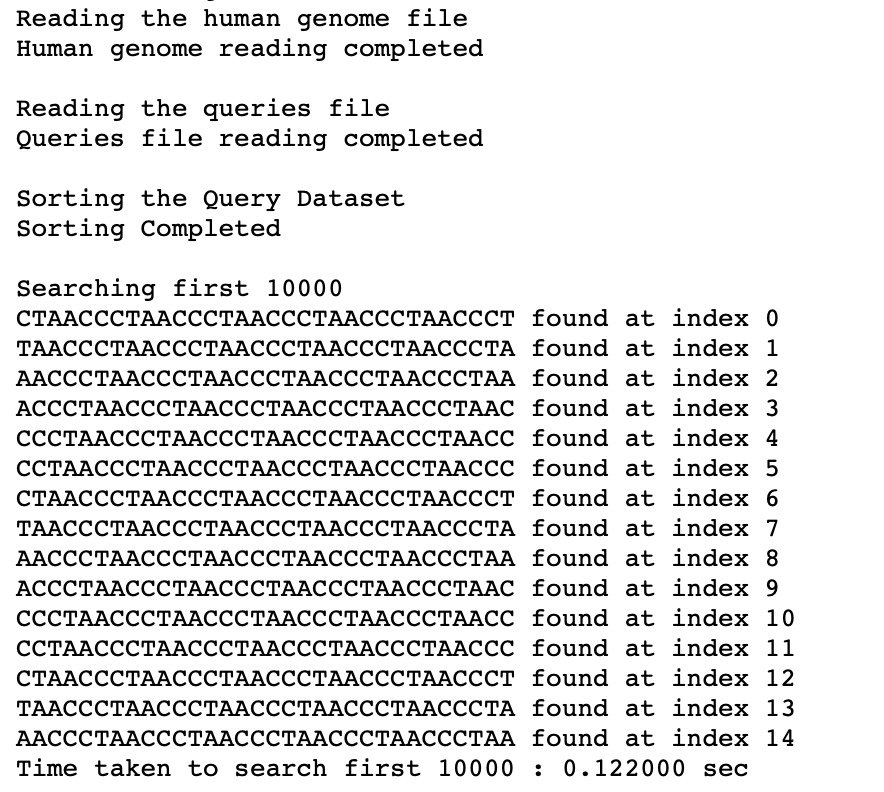
Description automatically generated

B. 20 pts) Read in the entire query dataset and store it in an instance of the Queries\_AR class. Sort all  
character fragments in alphabetic (lexicographic) 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 you implement should be optimal in time  
compared to the search function implemented in Part A and 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 5k, 10K, 100K, and 1M 32-character long  
fragments of the subject dataset within the query dataset?

1. For first 5k records



1. For first 10k records

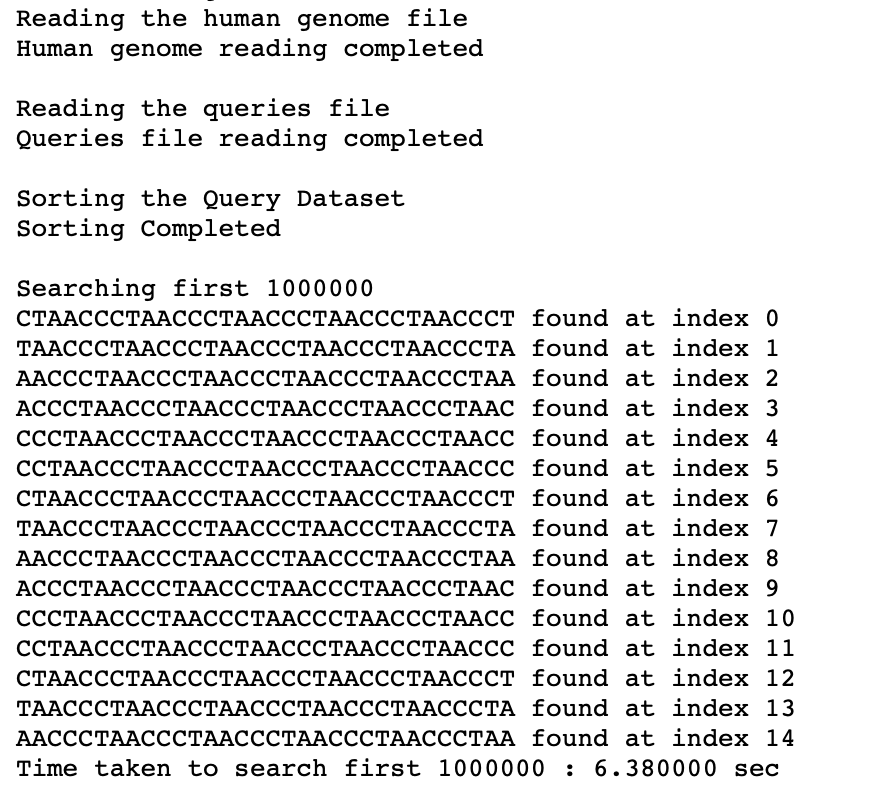


1. For first 100k records

A screenshot of a computer

Description automatically generated

1. For first 1M records



● 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 an estimate rather than an exact number.

1. To run entire Genome

A screenshot of a computer

Description automatically generated  
● Print the first 15 fragments of the subject dataset along with it’s indices that you found  
within the Query AR object (if any)

A screenshot of a computer

Description automatically generated

c. (20 pts) Explain the following in a video recording of duration for at most 8 minutes.  
● Using your write-up (.pdf format) that you have submitted, please provide a detailed  
explanation of your approach to solving the problem. You should cover the following  
points within a maximum time limit of 3 minutes:  
i. Provide a detailed explanation of the answers submitted in the write-up document  
for both Part A and Part B. Elaborate on why the results obtained are logical, and  
present your conclusions based on those results.  
● During your code explanation, which should last no more than 5 minutes, please cover  
all aspects of your code, including:  
i. Explain the logic used in implementing the search algorithm in Part A for  
searching the first 5k, 10K, 100K, and 1M 32-character long fragments of the  
subject dataset within the query dataset. Clearly state the motive of this  
function and detail the steps taken in its implementation.  
ii. Describe the specific bugs and issues you encountered while solving this  
assignment. These bugs could be from any part of your code for this homework.  
Provide detailed explanations of these challenges, avoiding trivial errors such as  
"missing a semicolon in the code."  
iii. Highlight at least one specific optimization you made to improve the code's  
efficiency or readability.

## Code Execution:

Firstly, run the make command to ready the executable file

srun make

To run code for all the lengths given in the assignment **without sort**, use the below command(s)

5000 - unsorted

srun ./homework2 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human\_reads\_125\_32.fa 5000 unsorted

10000 - unsorted

srun ./homework2 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human\_reads\_125\_32.fa 10000 unsorted

100000 - unsorted

srun ./homework2 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human\_reads\_125\_32.fa 100000 unsorted

1000000 - unsorted

srun ./homework2 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human\_reads\_125\_32.fa 1000000 unsorted

Entire Genome - unsorted

srun ./homework2 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human\_reads\_125\_32.fa 0 unsorted

To run code for all the lengths given in the assignment **with sort**, use the below command(s)

5000 - sorted

srun ./homework2 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human\_reads\_125\_32.fa 5000 sorted

10000 - sorted

srun ./homework2 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human\_reads\_125\_32.fa 10000 sorted

100000 - sorted

srun ./homework2 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human\_reads\_125\_32.fa 100000 sorted

1000000 - sorted

srun ./homework2 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human\_reads\_125\_32.fa 1000000 sorted

Entire Genome - sorted

srun ./homework2 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human\_reads\_125\_32.fa 0 sorted

**Part A: Searching Fragments Using Linear Search**

1. Loaded genome data into Queries\_AR from files using functions (ReadFile() and ReadQueriesFile()).
2. Implemented a basic linear search to find 32-character fragments within the genomic data (HumanGenome).
3. Iterated through each fragment in the dataset and compared it with all fragments in QueriesArray until a match was found or all possibilities completed and returing -1.
4. Calculated the time taken for different fragment counts (5k, 10k, 100k, 1M) using a timer (chrono library) to assess efficiency.
5. Displayed the first 15 matching fragments and indices

**Part B: Sorting and Searching with Binary Search**

1. Used Quick Sort to alphabetically sort all fragments in QueriesArray.
2. Utilized binary search, a more efficient method post-sorting, to find 32-character fragments within QueriesArray.
3. Started searching from the middle of the sorted list, dividing possibilities with each comparison until finding a match or not.
4. Measured search time for various fragment counts (5k, 10k, 100k, 1M) using binary search.
5. Validated results by printing the first 15 matching fragments and their positions in the sorted dataset.

## Video Presentation Link

<https://nau.zoom.us/rec/play/kKG-5MQ9lV-8DYW2yqWbv__fcm1sOe4Yh4K09iiTMujN22U6tnYCdIIq53tYN9Z7RNMP1DM1qe6g-S93.KFlYx3_O5PKCiuO3?autoplay=true&startTime=1720070672000>