Table of Contents

Homework – 3

Create a class called Queries_HT The purpose of the class will be to contain a dataset of genomic sequences (queries) and all the functions needed to operate on this set. Use the hash table data-structure to store the genomic fragments of a given size. The class must include the size of the hash table (m) as one of its configurable parameters. If you have a duplicate sequence fragment or a duplicate
hash value, use the chaining method to resolve collisions. Use Radix / division scheme for hash function implementation.
At minimum, the class must contain (15pts):

- A constructor
- A destructor
- A function to search the hash table for a given n-mer sequence (returning a presence / absence Boolean value should be sufficient)
- A function to insert a given n-mer sequence into the hash table. (Insert at the start of linked list)
- A function to convert a given sequence to a Radix notation (use double or unsigned int data type to store the radix value) – (Consider base as 5 with characters A, C, G, T, N).

Problem #1 :

A. (20 pts) Assess the impact of the hash table size. Set the hash table to a fixed value (m, see below). Set the size of your hash table (m) to 1 million, 10 million, 30 million, and 60 million elements. Populate the hash table with the sequence fragments from the query dataset.

- For each of your 4 hash table sizes, how many collisions did you observe while populating the hash?

1. 1 million : 124000000

```
Human Genome Reading started
Human Genome Reading completed
Time taken to Read Subject Dataset : 28.135 sec

Query Dataset Reading started
Query Dataset Reading completed
Time taken to populate Hashtable of size 1000000 : 222 sec

Total number of Collisions: 124000000

Searching started
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 0
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 1
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 2
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 3
CCCTAACCCTAACCCTAACCCTAACCCTAACC at 4
CCTAACCCTAACCCTAACCCTAACCCTAACCC at 5
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 6
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 7
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 8
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 9
Searching completed

Total number of Occurances: 300094190

Time taken to Search : 50238.6 sec
```

2. 10 million : 115003428

```
Time taken to Read Subject Dataset : 28.575 sec

Query Dataset Reading started
Query Dataset Reading completed
Time taken to populate Hashtable of size 10000000 : 224.026 sec

Total number of Collisions: 115003428

Searching started
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 0
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 1
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 2
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 3
CCCTAACCCTAACCCTAACCCTAACCCTAACC at 4
CCTAACCCTAACCCTAACCCTAACCCTAACCC at 5
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 6
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 7
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 8
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 9
Searching completed

Total number of Occurances: 300094190

Time taken to Search : 14016.8 sec
```

3. 30 million : 96362393

```
Time taken to Read Subject Dataset : 29.95 sec

Query Dataset Reading started
Query Dataset Reading completed
Time taken to populate Hashtable of size 30000000 : 231.466 sec

Total number of Collisions: 96362393

Searching started
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 0
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 1
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 2
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 3
CCCTAACCCTAACCCTAACCCTAACCCTAACC at 4
CCTAACCCTAACCCTAACCCTAACCCTAACCC at 5
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 6
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 7
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 8
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 9
Searching completed

Total number of Occurances: 300094190

Time taken to Search : 8181.18 sec
```

4. 60 million : 76632638

```
Time taken to Read Subject Dataset : 28.47 sec

Query Dataset Reading started
Query Dataset Reading completed
Time taken to populate Hashtable of size 60000000 : 224.802 sec

Total number of Collisions: 76632638

Searching started
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 0
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 1
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 2
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 3
CCCTAACCCTAACCCTAACCCTAACCCTAACC at 4
CCTAACCCTAACCCTAACCCTAACCCTAACCC at 5
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 6
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 7
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 8
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 9
Searching completed

Total number of Occurances: 300094190

Time taken to Search : 6235.55 sec
```

● For each of your 4 hash table sizes, how long did it take you to populate the hash table? Do the timing results make sense? Explain.

1. 1000000 – 222 sec
2. 10M – 222.026 sec
3. 30M – 231.466 sec
4. 60M – 224.802 sec

Problem #2 :

B. (20 pts) Searching speed: Set the hash table size to 60 million. Populate the hash table with the sequence fragments from the query dataset. Read 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_HT object. Iterate through all 32-character long fragments of the subject dataset, searching for each one in the query dataset.

● How long did it take to search for every possible 32-character long fragment of the subject dataset within the query dataset?

1. Time taken to search for entire genome with Hash table size 60M – 6235.55 Seconds

```
Time taken to Read Subject Dataset : 28.47 sec

Query Dataset Reading started
Query Dataset Reading completed
Time taken to populate Hashtable of size 60000000 : 224.802 sec

Total number of Collisions: 76632638

Searching started
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 0
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 1
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 2
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 3
CCCTAACCCTAACCCTAACCCTAACCCTAACC at 4
CCTAACCCTAACCCTAACCCTAACCCTAACCC at 5
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 6
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 7
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 8
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 9
Searching completed

Total number of Occurances: 300094190

Time taken to Search : 6235.55 sec
```

● How many such fragments did you find?

Total number of Occurances: 300094190

● Print the first 10 fragments of the subject dataset that you found within the Query_HT.

```
Searching started
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 0
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 1
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 2
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 3
CCCTAACCCTAACCCTAACCCTAACCCTAACC at 4
CCTAACCCTAACCCTAACCCTAACCCTAACCC at 5
CTAACCCTAACCCTAACCCTAACCCTAACCCT at 6
TAACCCTAACCCTAACCCTAACCCTAACCCTA at 7
AACCCTAACCCTAACCCTAACCCTAACCCTAA at 8
ACCCTAACCCTAACCCTAACCCTAACCCTAAC at 9
Searching completed
```

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.

Code Execution:

Run make command to generate the executable file. Then execute any if the below commands based on the requirement

1. 1000000 Records
   ./homework3 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human_reads_125_32.fa 1000000
2. 10000000 Records
3. ./homework3 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human_reads_125_32.fa 10000000

4. 30000000 Records

   ./homework3 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human_reads_125_32.fa 30000000

5. 60000000 Records
6. ./homework3 /common/contrib/classroom/inf503/genomes/human.txt /common/contrib/classroom/inf503/human_reads_125_32.fa 60000000

Explanation and Conclusion

- Reading the query file line-by-line and calculating hash values and handling collisions.
- Inserting fragments into the hash table using separate chaining.
- Using the `getRadixHash` method to compute hash values, ensuring uniform distribution.
- terating through the stored genome sequence, calculating hash values for substrings, and searching the hash table.
- Updating the hit counter and printing results if matches are found.
- For Part A, noting that reading time is almost the same for all sizes of hash tables as it needs to calculate the hash for the same number of fragments.
- For Part B, noting that searching speed increases with a 60M hash table size, as a larger hash table decreases collisions and improves the search.

Conclusion:

- Providing fast lookups and handling collisions effectively with the hash table.
- Efficiently and accurately handling large genomic datasets and query fragments.

During your code explanation, which should last no more than 5 minutes, please cover all aspects of your code, including:

i. Explain why hash tables would be efficient for this problem statement. How does it improve the performance compared to homework assignment 1 and 2 in terms of search?
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.

Video Presentation Link

https://nau.zoom.us/rec/share/P0BCS8-DsC4MYwIWkMc8nVFl3J4LiEcg-9u0GdnOztKPRu40-I8NB19beYcz6TYA.cBNDcu39ds-3h5Wb?startTime=1721024096000