**Homework #4**

**Problem #1 (of 2)**

**PART-A**

**(10 pts) Searching speed.** Store all fragments of the query dataset in your Queries\_NW class. Randomly pick 1K, 10K, 100K n-mers of the subject dataset to conduct fuzzy searching within the query dataset using NW algorithm.

**For each of your searches (1K, 10K, 100K), how long did the search take?**

**For 1K**

Comparing 1000 random generated genome fragments of size 32 with 100Million query fragments to conduct fuzzy searching using NW algorithm would take more than 24 hours. So I would like to estimate the time for it by comparing

1000 genome fragments - 1K query fragments

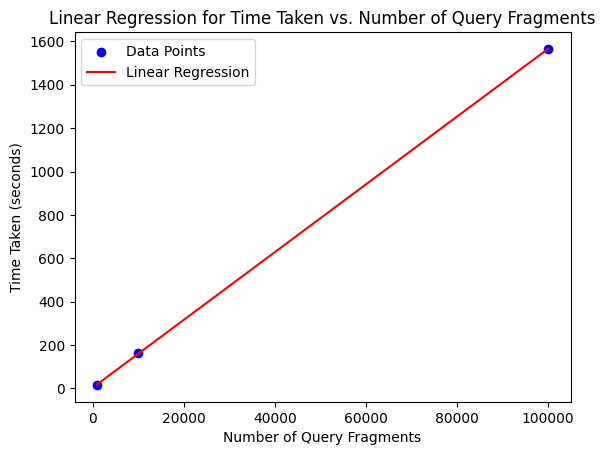
1000 genome fragments - 10k query fragments

1000 genome fragments - 100k query fragments.

Time taken for 1000 genome fragments - 1K query fragments is 16.2819 seconds.

Time taken for 1000 genome fragments - 10k query fragments is 162.127 seconds.

Time taken for 1000 genome fragments - 100k query fragments is 1565.07 seconds.



**Predicted time taken for 100 million query fragments: 1562162.5300360362 seconds**

**For 10K**

Comparing 10k random generated genome fragments of size 32 with 100Million query fragments to conduct fuzzy searching using NW algorithm would take more than 24 hours. So I would like to estimate the time for it by comparing

10k genome fragments - 1K query fragments.

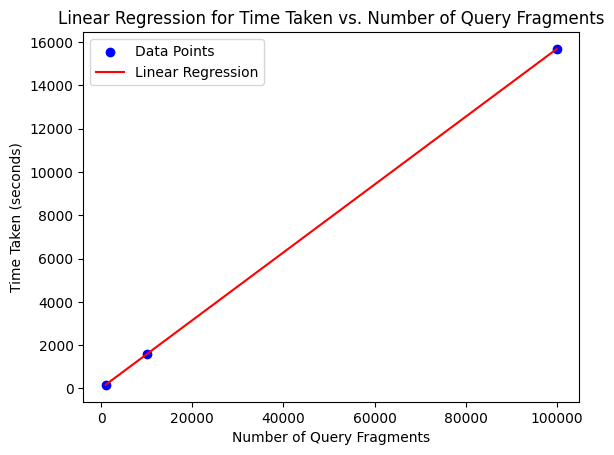
10k genome fragments - 10k query fragments.

10k genome fragments - 100k query fragments.

Time taken for 10K genome fragments - 1K query fragments is 162.294 seconds.

Time taken for 10K genome fragments - 10k query fragments is 1600.36 seconds.

Time taken for 10K genome fragments - 100k query fragments is 15677.9 seconds.

****

**Predicted time taken for 100 million query fragments: 15659935.865405407 seconds**

**For 100K**

Comparing 100k random generated genome fragments of size 32 with 100Million query fragments to conduct fuzzy searching using NW algorithm would take more than 24 hours. So I would like to estimate the time for it by comparing

1000 genome fragments - 1K query fragments.

1000 genome fragments - 10k query fragments.

Time taken for 100K genome fragments - 1K query fragments is 1622.85 seconds.

Time taken for 100K genome fragments - 10k query fragments is 16113.9 seconds.

****

**Predicted time taken for 100 million query fragments: 161011679.39999995 seconds.**

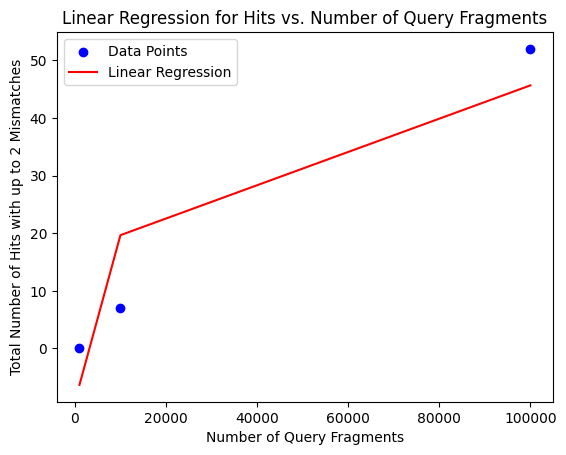
**For each of your searches (1K, 10K, 100K), how many ‘hits’ with up to 2 mismatches did you find?**

**For 1K**

The total number of hits with up to 2 mismatches for randomly picked 1k fragments – 1K query\_frags is 0.

The total number of hits with up to 2 mismatches for randomly picked 1k fragments – 10K query\_frags-7.

The total number of hits with up to 2 mismatches for randomly picked 1k fragments – 100K query\_frags-52.



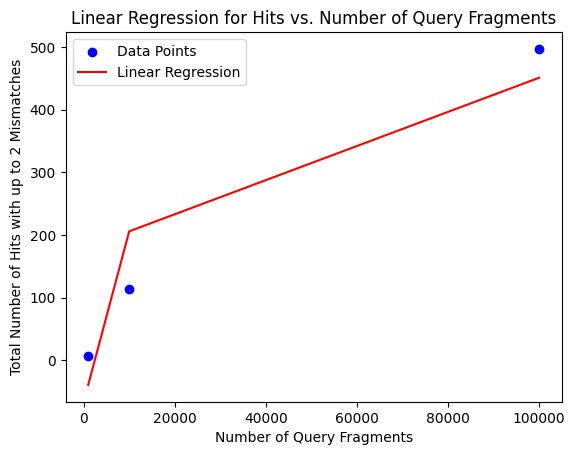
**Predicted total number of hits with up to 2 mismatches for 100 million query fragments: 123.6666666666666**

**For 10K**

The total number of hits with up to 2 mismatches for randomly picked 10k fragments – 1K query\_frags is 7.

The total number of hits with up to 2 mismatches for randomly picked 10k fragments – 10K query\_frags-114.

The total number of hits with up to 2 mismatches for randomly picked 10k fragments – 100K query\_frags- 497.

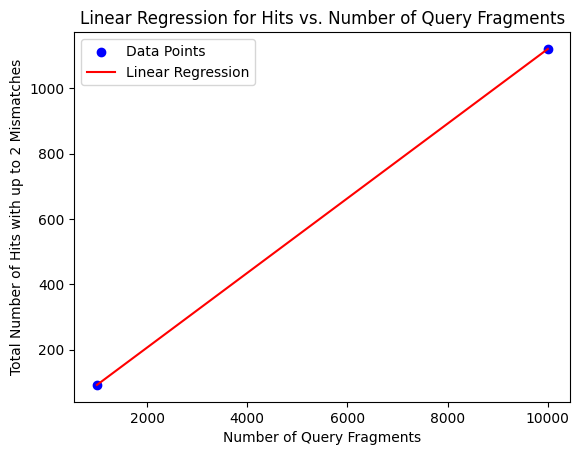


**Predicted total number of hits with up to 2 mismatches for 100 million query fragments: 1185.9999999999993**

**For 100K:**

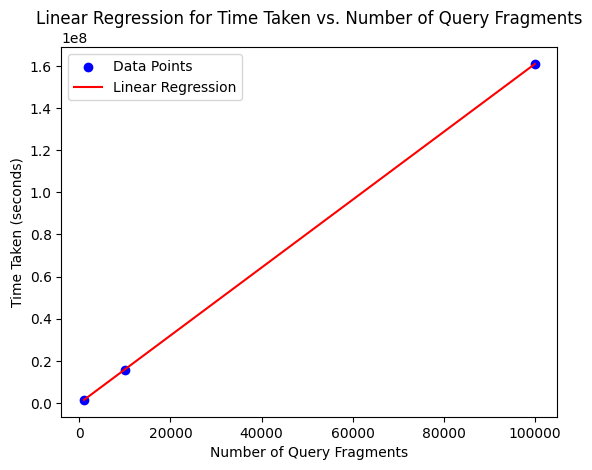
The total number of hits with up to 2 mismatches for randomly picked 100k fragments – 1K query\_frags is 92.

The total number of hits with up to 2 mismatches for randomly picked 100k fragments – 10K query\_frags is 1120.

****

**Predicted total number of hits with up to 2 mismatches for 100 million query fragments: 5232.00000000002**

**How long would the search take for the entire subject dataset?**

****

**Predicted time taken for three\_billion fragments: 4837176771137.477 seconds**

**PART-B**

Store all fragments of the query dataset in your Queries\_NW class. Generate completely random 1K, 10K, 100K n-mers to conduct fuzzy searching within the query dataset using NW algorithm.

**For each of your searches (1K, 10K, 100K), how long did the search take?**

**For 1K**

Comparing 1000 completely random generated genome fragments of size 32 with 100Million query fragments to conduct fuzzy searching using NW algorithm would take more than 24 hours. So I would like to estimate the time for it by comparing

1000 completely genome fragments - 1K query fragments

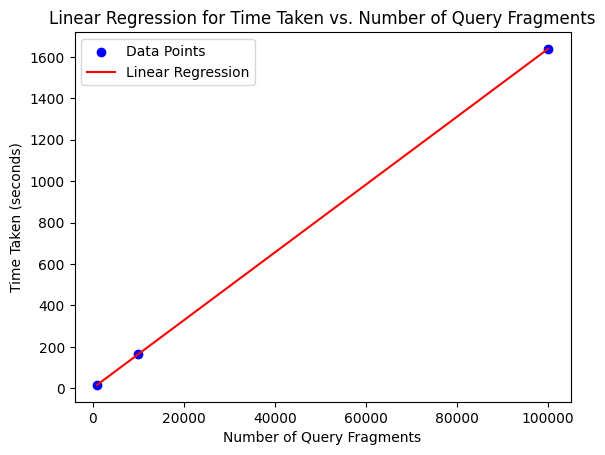
1000 completely genome fragments - 10k query fragments

1000 completely genome fragments - 100k query fragments.

Time taken for 1000 completely genome fragments - 1K query fragments is 16.4278 seconds.

Time taken for 1000 completely genome fragments - 10k query fragments is 164.21 seconds.

Time taken for 1000 completely genome fragments - 100k query fragments is 1637.98 seconds.



**Predicted time taken for 100Million fragments: 1637765.8182522522 seconds**

**For 10K**

Comparing 10K completely random generated genome fragments of size 32 with 100Million query fragments to conduct fuzzy searching using NW algorithm would take more than 24 hours. So I would like to estimate the time for it by comparing

10K completely genome fragments - 1K query fragments

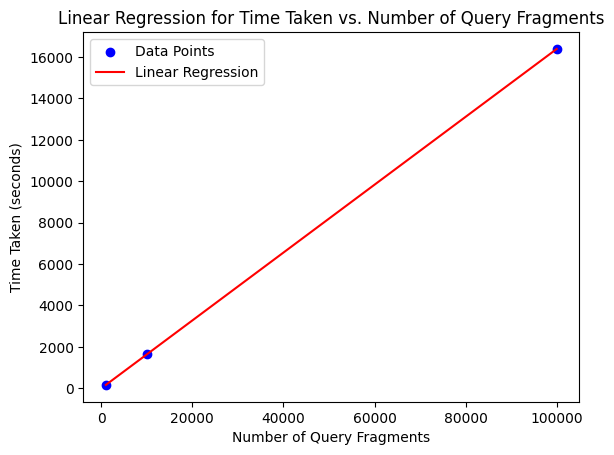
10K completely genome fragments - 10k query fragments

10K completely genome fragments - 100k query fragments.

Time taken for 10K completely genome fragments - 1K query fragments is 164.369 seconds.

Time taken for 10K completely genome fragments - 10k query fragments is 1637.22 seconds.

Time taken for 10K completely genome fragments - 100k query fragments is 16389.9 seconds.



**Predicted time taken for 100Million fragments: 16390414.391126128 seconds**

**For 100K**

Comparing 100K completely random generated genome fragments of size 32 with 100Million query fragments to conduct fuzzy searching using NW algorithm would take more than 24 hours. So I would like to estimate the time for it by comparing

100K completely genome fragments - 1K query fragments

100K completely genome fragments - 10k query fragments

Time taken for 100K completely genome fragments - 1K query fragments is 1639.32 seconds.

Time taken for 100K completely genome fragments - 10k query fragments is 16381.3 seconds.



**Predicted time taken for 100Million fragments: 163799779.09999996 seconds**

**For each of your searches (1K, 10K, 100K), how many ‘hits’ with up to 2 mismatches did you find?**

For 1K – 0 hits.

For 10K – 0 hits.

For 100K – 0 hits.

**How does the search time compare to the results of Problem 1A? Does this make sense – explain why or why not?**

Predicted time taken for 100 million query fragments for randomly generated index: 161011679 seconds.

Predicted time taken for 100 Million fragments for completely randomly generated 32 –length fragment: 163799779. seconds.

The time taken for completely randomly generated fragments is almost 10 times more than the time taken for problem1A. And it does make sense because there are no hits with completely randomly generated fragment. So program performs Needleman wunsch algorithm for all the query fragments for each genome fragment. Because, in problem1A if we get a hit, then we stop comparing to query fragments. Whereas for problem1B, there is no way we get a hit. So, completely randomly generated fragments is an perfect example for the worst case time complexity of Needleman-Wunsch algorithm which is O(G)O(N)O(n^2)O(2\*n).

**Problem #2 (of 2)**

**A (10 pts) Searching speed.** Store all fragments of the query dataset in your Queries\_BL class. Randomly pick 1K, 10K, 100K n-mers of the subject dataset to conduct fuzzy searching within the query dataset using BLAST algorithm.

1. For each of your searches (1K, 10K, 100K), how many ‘hits’ with up to 2 mismatches did you find?

1K : 587 Hits

10K: 6545 Hits

100K: 98767 Hits

1. For each of your searches (1K, 10K, 100K), how long did the search take?

1K: 914.89 seconds.

10K: 1256.2 seconds.

100K: 2587.4 seconds.

1. How long would the search take for the entire subject dataset? How does this search time compare to the results of problem 1A? Does this make sense – explain why or why not.

**For entire genome: 16,812.976 seconds.**

Doing with BLAST algorithm, it is way faster. Because, it looks for exact matches of short words, instead of for similar sequences that match due to gaps in either the query or genome. So that’s why there is a huge difference between problem1A compared to problem2A.

In the homework, I have stored all the 16fragments of 100Million queries in a hashtable. Now for each 32sized genome, it’s 16-mers are searched in the hashtable which is a constant operation though. So the overall timecomplexity would be O(G)\*O(2\*n)O(n^2).

O(G) – for entire genome.

O(2\*n) – for seed extension to both left and right.

O(n^2) – for performing smith waterman on genome and query fragments.

For problem1A – The time complexity is O(G)O(N)O(n^2)O(2\*n). For problem2A, we are able to vomit the O(N) using BLAST algorithm.

**B (10 pts) Searching speed**: Store all fragments of the query dataset in your Queries\_BL class. Generate completely random 1K, 10K, 100K n-mers to conduct fuzzy searching within the query dataset using BLAST algorithm.

1. For each of your searches (1K, 10K, 100K), how many ‘hits’ with up to 2 mismatches did you find? How does this compare to the results of 1B? Does this make sense – explain why or why not.

1K – 0 Hits.

10K – 0 Hits.

100K – 0 Hits.

1. For each of your searches (1K, 10K, 100K), how long did the search take? How does that compare with the benchmarks from problem 1, part B.

1K : 1139.08 seconds.

10K: 1206.2 seconds.

100K: 1877.4 seconds.

As said, problem1B is perfect example for worst case time complexity. But, using BLAST we could vomit that O(N). So it is way faster than it again. But It is even faster than problem2A. Because in problem2A, once we get a hit we perform for smith-waterman which is O(n^2). But for completely random, anyway there is not going to be a hit for 99.99%. So this problem2B even vomits O(n^2).

So the time complexity for problem2B would be O(G)\*16. Where is 16 is no of 16-mers in each 32sized genome fragment.

**Steps of execution:**

* 1. Created total of 3 files main.cpp, header\_definitions.cpp, header.h.
  2. Header.h contains all the header files that are used in the program.
  3. The main.cpp contains the main function and all the function calls required to get desired output
  4. Header\_definitions.cpp contains all the function definitions which are declared in the header file
  5. Created a make file to run the code
  6. Uploaded all the above files to a directory on monsoon
  7. The command for execution of problem 1 – part A is

srun ./source /common/contrib/classroom/inf503/human\_reads\_version2.fa /common/contrib/classroom/inf503/genomes/human.txt partA problem1

* 1. The command for execution of problem 1 – part B is

srun ./source /common/contrib/classroom/inf503/human\_reads\_version2.fa /common/contrib/classroom/inf503/genomes/human.txt partB problem1

* 1. The command for execution of problem 2 - partA is

srun ./source /common/contrib/classroom/inf503/human\_reads\_version2.fa /common/contrib/classroom/inf503/genomes/human.txt partA problem2

* The command for execution of problem 2 - partB is

srun ./source /common/contrib/classroom/inf503/human\_reads\_version2.fa

/common/contrib/classroom/inf503/genomes/human.txt partB problem2