

INF503_Assignment_3_mp2525 - Solutions

Problem #1 (of 2): Fun with direct access arrays

A. Getting started: read in the read data set into your data structure

- What is the size of your hash table?
- How many collisions did you observe?
- How many unique sequences did you observe (number of “ON” Boolean values)?
- What is the load (αT) in your hash table?

Ans:

```
[mp2525@wind ~/large-scale-data-structures/homework3] $ cat /scratch/mp2525/dat_q1a.txt
g++ -c homework.cpp -O3
g++ -c fastaDAT.cpp -O3
g++ -c fastaHT.cpp -O3
g++ -o homework homework.o fastaDAT.o fastaHT.o
The number of arguments passed: 3
The first argument is: /home/mp2525/large-scale-data-structures/homework3/./homework
The second argument is: problem1A
The third argument is: /common/contrib/classroom/inf503/hw3_dataset.fa
////////////////////////////////////
Size of Hash Table: 4 GB
Total sequences: 5999246
Unique sequences: 4414557
Total collisions: 1584689
Load Factor: 0.00102784
////////////////////////////////////
rm -rf *.o homework
```

From the screenshot, we can observe that the size of the hash table is about 4 GB. It is calculated as: $\text{pow}(4, 16) = 4294967296 * \text{sizeof}(\text{bool}) = \mathbf{4294967296 \text{ bytes} = 4 \text{ GB}}$

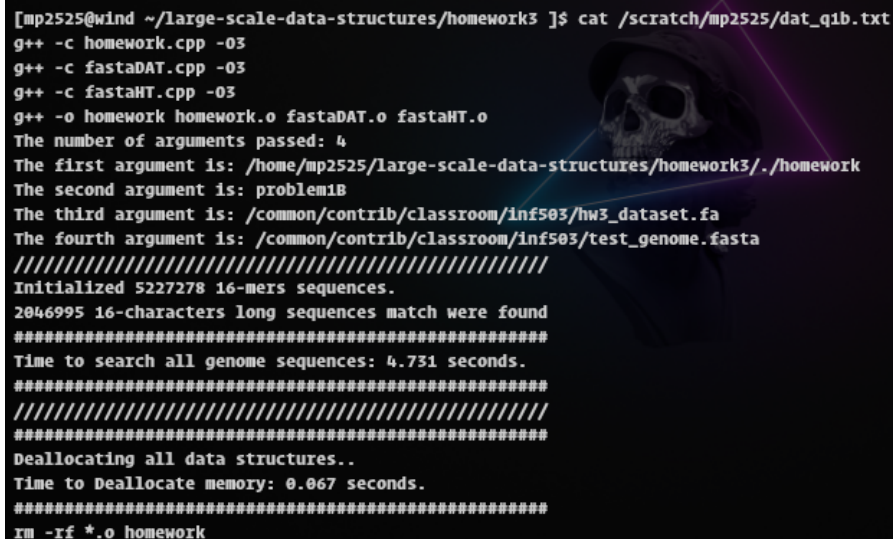
Similarly, We can observe that there are **5999246** total sequences, out of which **4414557** sequences are unique. The total number of detected collisions is **1584689**.

The load factor in the hash table is **0.00102784**. It is because only 4414557 of the sequences are stored in the hash table out of its full capacity (4294967296).

B. Search time in direct access arrays: read in the genome sequence provided above, iterate through all 16-mers found in the genome, and use them to query the read set (similar to what you did in HW#2, problem 2B).

- How many genome 16-mer fragments were found in your read set?
- How long did it take to complete the entire search process (all 16-mers)?

Ans:

A terminal window screenshot showing the execution of a C++ program. The background is dark with a faint skull watermark. The text is white and shows the compilation and execution of a program that searches for 16-mers in a genome dataset.

```
[mp2525@wind ~/large-scale-data-structures/homework3]$ cat /scratch/mp2525/dat_q1b.txt
g++ -c homework.cpp -O3
g++ -c fastaDAT.cpp -O3
g++ -c fastaHT.cpp -O3
g++ -o homework homework.o fastaDAT.o fastaHT.o
The number of arguments passed: 4
The first argument is: /home/mp2525/large-scale-data-structures/homework3/./homework
The second argument is: problem1B
The third argument is: /common/contrib/classroom/inf503/hw3_dataset.fa
The fourth argument is: /common/contrib/classroom/inf503/test_genome.fasta
////////////////////////////////////
Initialized 5227278 16-mers sequences.
2046995 16-characters long sequences match were found
#####
Time to search all genome sequences: 4.731 seconds.
#####
////////////////////////////////////
#####
Deallocating all data structures..
Time to Deallocate memory: 0.067 seconds.
#####
rm -rf *.o homework
```

5227278 genome 16-mers fragments were found in the genome sequence dataset. Out of which, 2046995 matches with the read dataset were found.

It took **4.731 seconds** to complete the entire search process (all 16-mers).

Problem #2 (of 2): The hash table with chaining

A. Assessing the impact of the hash table size. For this, you will need to set the hash table to a fixed value (m , see below) and read in the read set to populate the hash table. Set the size of your hash table (m) to 10 thousand, 100 thousand, 1 million, and 10 million elements.

- For each of your 4 hash table sizes, how many collisions did you observe while populating the hash?
- For each of your 4 hash table sizes, how long did it take you to read the sequence fragment file?
- Do the results make sense? Explain.

Ans:

```
[mp2525@wind ~/large-scale-data-structures/homework3] $ cat /scratch/mp2525/dat_q2a.txt
cat: /scratch/mp2525/dat_q2a.txt: No such file or directory
[mp2525@wind ~/large-scale-data-structures/homework3] $ cat /scratch/mp2525/ht_q2a.txt
g++ -c homework.cpp -O3
g++ -c fastaDAT.cpp -O3
g++ -c fastaHT.cpp -O3
g++ -o homework homework.o fastaDAT.o fastaHT.o
The number of arguments passed: 3
The first argument is: /home/mp2525/large-scale-data-structures/homework3/./homework
The second argument is: problem2A
The third argument is: /common/contrib/classroom/inf503/hw3_dataset.fa
#####
Time to store sequences in 10000 size Hash Table: 6.087 seconds.
#####
Total collisions in 10,000 hash Table: 5989246
#####
Time to store sequences in 100000 size Hash Table: 6.498 seconds.
#####
Total collisions in 100,000 hash Table: 5899246
#####
Time to store sequences in 1000000 size Hash Table: 6.574 seconds.
#####
Total collisions in 1,000,000 hash Table: 5051475
#####
Time to store sequences in 10000000 size Hash Table: 6.591 seconds.
#####
Total collisions in 10,000,000 hash Table: 2618871
#####
Deallocating all data structures..
Time to Deallocate memory: 0.796 seconds.
#####
Deallocating all data structures..
Time to Deallocate memory: 0.751 seconds.
#####
Deallocating all data structures..
Time to Deallocate memory: 0.740 seconds.
#####
Deallocating all data structures..
Time to Deallocate memory: 7.338 seconds.
#####
rm -rf *.o homework
```

Collisions and Read Time for 4 Hash Tables:

Hash Table Size	Number of Collisions	Time to read sequence
10 thousand	5989246	6.087 seconds
100 thousand	5899246	6.498 seconds
1 million	5051475	6.574 seconds
10 million	2618871	6.591 seconds

As the size of the Hash Table is increasing, the number of collisions is decreasing. It is because increasing hash table capacity allows more sequence data to be stored in the table.

Also, The time to read the sequences is increasing with the increasing hash table size, because many elements are stored in the hash table rather than being discarded from the collision.

B. Searching in the chain-linked hash table. Set the hash size to 10,000,000 and populate it using the read set. Read in the genome, iterate through all 16-mers found in the genome, and use them to query the read set (similar to what you did in HW#2, problem 2B).

- How many genome 16-mer fragments were found in your read set?
- How long did it take to complete the entire search process (all 16-mers)?
- How does that compare to the direct access array search times you've implemented as part of problem 1B?

Ans:

```
[mp2525@wind ~/large-scale-data-structures/homework3] $ cat /scratch/mp2525/ht_q2b.txt
g++ -c homework.cpp -O3
g++ -c fastaDAT.cpp -O3
g++ -c fastaHT.cpp -O3
g++ -o homework homework.o fastaDAT.o fastaHT.o
The number of arguments passed: 4
The first argument is: /home/mp2525/large-scale-data-structures/homework3/./homework
The second argument is: problem2B
The third argument is: /common/contrib/classroom/inf503/hw3_dataset.fa
The fourth argument is: /common/contrib/classroom/inf503/test_genome.fasta
////////////////////////////////////
#####
Time to store sequences in 10000000 size Hash Table: 6.614 seconds.
#####
Initialized 5227278 16-mers sequences.
2046995 16-characters long sequences match were found
#####
Time to search all genome sequences: 5.775 seconds.
#####
Deallocating all data structures..
Time to Deallocate memory: 0.942 seconds.
#####
////////////////////////////////////
rm -rf *.o homework
```

2046995 16-mer genome fragments were found in the read set.

It took **5.775 seconds** to complete the entire search process (all 16-mers).

It took longer for Chain Linked Hash Table to find the 16-mers genome fragments compared to the Direct Access Hash Table. It is because, In direct access array, the search time for the element is $O(1)$ whereas the search time in a chain-linked hash table is $O(1) + O(N)$ where N is the length of the chain link. The direct access array provides less search time at the cost of high Hash Table size.