Project 3: Xiao Wei's Problem II

Zheran Fang

1 Dec 2014

The main purpose of this project is to get you familiar with string matching algorithms. This is an individual assignment; you may not share code with other students. Java is the acceptable programming language.

Introduction

Xiao Wei majors in biology. Recently he is working on a project which requires searching a database of DNA sequences. Due to the huge size of the database, searching for DNA sequences manually is mountains of work. Therefore he turns to your for help, *again*.

Since the DNA sequences are basically strings, we can treat the DNA sequence matching problem as a string matching problem. The simplest algorithm for implementing this string matching problem is to loop over combinations of the database sequences and query sequences and compare the strings. This naïve algorithm can be terribly slow as the size of the input sequences increases. There are algorithms for string matching that are much faster than the naïve algorithm, for example, the Knuth–Morris–Pratt algorithm, the Boyer–Moore algorithm and the Rabin–Karp algorithm. Moreover, you can leverage other techniques, such as multithreading, hashing and MapReduce, to optimize the performance.

Specification

Your Job

Write a program to search a database file of DNA sequences, represented as strings of characters, to find matches of other DNA subsequences. Both the

database sequences and the query sequences are made up of only four characters: A, C, G and T. The output must report the location of an exact match within any given input DNA sequence for each input search query sequence. If the query sequence matches within multiple database sequences within the database file, each result must be reported; and if the query sequence matches multiple locations within the same database sequence, the earliest position that matches exactly must be reported. The file names for this problem (database file, query file, output results) will be given on the command line as arguments. The usage of the command line program should be:

```
java -jar dna_matching.jar db_file query_file output_file
```

File Format

The input database files and query files will have the same format. Each sequence will be prefixed by a line starting with a greater than character (`>`) followed by a description of the sequence. The sequence will then begin on the next line. The end of the file will be signified by the descriptor (`>EOF`).

For each query sequence contained within the input query file, the output file should print the description of the query sequence and the description of any database sequences that contain a match as well as the position within the database sequence of that exact match. If the query sequence is not found within any database sequences, a message (`NOT FOUND`) to that effect should be printed after the query sequence description.

Thus the program takes as input two files: the database file and the query file. We make some assumptions about the input:

- Each sequence is less than 1,000,000 bytes.
- The content of the entire database file fits in main memory, but the content of the entire query file may be too big to fit in main memory.
- There are less than 2^{32} strings in a database file or a query file.

Here is an database file example:

```
>DB description string 1
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTC
TGATAGCAGCTTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGG
TCACTAAATACTTTAACCAATATAGGCATAGCGCACAGACAGATAAAAATTACA
>DB description string 2
AACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAGCCCGCACCTGACAGTGCGGG
CTTTTTTTTTCGACCAAAGGTAACGAGGTAACAACCATGCGAGTGTTGAAGTTCGGCGGT
ACATCAGTGGCAAATGCAGAACGTTTTCTGCGTGTTGCCGATATTCTGGAAAGCAATGC
>EOF
```

Here is an query file example:

```
>Query description string 1
CATTCTGACTGCAA
>Query description string 2
AAAAAAG
>Query description string 3
GTAA
>Query description string 4
AGAGAGAGAGAGAGAGAGAGAGAGAGAG
>EOF
```

The output would then be

```
Query description string 1
    [DB description string 1] at offset 7

Query description string 2
    [DB description string 1] at offset 47
    [DB description string 2] at offset 33

Query description string 3
    [DB description string 1] at offset 94
    [DB description string 2] at offset 79

Query description string 4
    NOT FOUND
```

Test Environment

- Operating system: Windows 7 (32-bit)

- CPU: Intel quad-core Core™ i5-2400 @3.10GHz
- Memory: 3.24 GB
- Java version: 1.8.0_25
- Maximum heap size of Java virtual machine: 2048 MB

Grading

- Correctness: 60%
- Time performance: 20% (Prerequisite: pass all correctness tests)
- Project development document: 15%
- User manual: 5%
- GUI not required

Submission

Create a zip file named *YourStudentID*.zip that contains the following items:

- Source code
- Executable jar
- User manual
- Development document, in which you can describe your project design in detail, the problems you have encountered, as well as your solutions or ideas. In addition, you can analyze the time/memory complexity of your program and introduce the methods you have leveraged to optimize the performance of your program.

And upload your zip file to the FTP server.

After submission , we will set up a face-to-face interview one by one, so get yourself ready for it.

Deadline

20 Dec 2014 23:59 GMT+08:00

Credit

This project is designed based on the “String Matching” problem in the *Intel 2009 Threading Challenge* and Yuzhen Xie's adapted version.