Assignment 0: Warmup

CS249

Spring 2025

Objectives

In this assignment, you will implement a basic pattern matching algorithm to identify Alu sequences in different human genome assemblies. This will give you hands-on experience with:

- Working with large-scale genomic data
- Implementing fundamental string matching algorithms
- Performance analysis and optimization

Requirements

Data Acquisition

1. Download the following human genome assemblies from NCBI Datasets:
   - GRCh38 (hg38)
   - T2T CHM13v2.0

   Access URL: `https://www.ncbi.nlm.nih.gov/datasets/genome/?taxon=9606`

2. Retrieve the AluY consensus sequence from `https://www.dfam.org/family/DF000000002/download` or directly use this sequence (in FASTA format):

```
>DF000000002.4 AluY
GGCCGGGCGCGGTGGCTCACGCCTGTAATCCCAGCACTTTGGGAGGCCGAGGCGGGCGGA
TCACGAGGTCAGGAGATCGAGACCATCCTGGCTAACACGGTGAAACCCCGTCTCTACTAA
AAATACAAAAAATTAGCCGGGCGTGGTGGCGGGCGCCTGTAGTCCCAGCTACTCGGGAGG
CTGAGGCAGGAGAATGGCGTGAACCCGGGAGGCGGAGCTTGCAGTGAGCCGAGATCGCGC
CACTGCACTCCAGCCTGGGCGACAGAGCGAGACTCCGTCTCAAAAAAAAAAAAAAAAAAA
AAAAAAAAAA
```

Implementation Requirements

1. (40 points, 10 bonus points) First, your program MUST[1] output the total number of exact matches of AluY in the genome. Your program MAY output the location of each exact AluY match (chromosome, start, end). You MUST implement exact pattern matching using only character-to-character comparisons (i.e.: you MUST NOT use built-in string matching functions).

2. (40 points, 10 bonus points) Second, your program MUST output the total number of matches of AluY with up to one mismatch in the genome. Consider these type of mismatches: replacing a character, inserting a character, deleting a character. Your program MAY output the location of each match (chromosome, start, end). You MUST implement pattern matching with mismatch using only character-to-character comparisons (i.e.: you MUST NOT use built-in string matching functions).

3. (10 points) You MUST apply your program to both the GRCh38 and CHM13 human assembly, and report the output.

4. (10 points, 10 bonus points) For both tasks, you MUST output the total runtime and you SHOULD output peak memory usage. Both runtime and memory usage SHOULD be determined outside your program, e.g., using the `time` command for time, and a memory profiler like `heaptrack` or `valgrind` for memory.

5. (20 bonus points) Your program SHOULD

    - (10 bonus points) read compressed (bgzipped) FASTA files due to file sizes,
    - (10 bonus points) be able to handle chromosomes/sequences larger than available RAM.

Submission Guidelines

The following is required for your submission:

1. Source code with documentation

2. Requirements file listing dependencies, or make file, or a Dockerfile

3. Report in PDF or Markdown format (max 4 pages) containing:

    - Brief implementation description (1 paragraph)
    - Results on AluY counts in two genome assemblies
    - Performance analysis results, and comparison across genome assemblies

[1] See `https://datatracker.ietf.org/doc/html/rfc2119`.

The source code and report should be submitted as gzipped file. Alternatively, you can submit the link to a public Github repository.