Assignment 1: k-mer index with minimizer

CS249

Spring 2025

Objectives

In this assignment, you will implement a database for classifying strings. The database will consist of k-mers — substrings of length k. A k-mer is derived by sliding a window of length k across a string (e.g., a DNA sequence), generating all possible contiguous subsequences of length k.

What you will classify are "sequence reads" — the raw data generated by a genome sequencer. These reads are produced through shotgun sequencing, where the DNA from a sample is fragmented and sequenced in parallel, producing thousands to millions of short DNA sequences (usually 100-150bp in length). In metagenomics, these reads come from multiple organisms simultaneously, creating a complex mixture of DNA from multiple different sources.

The challenge of metagenomics becomes clear when we consider real-world samples. Take a typical stool sample used in gut microbiome analysis: it contains DNA from hundreds of bacterial species comprising the gut microbiota, along with DNA from undigested food organisms, human host cells, and potentially pathogens or parasites. When this sample is sequenced, a sequencing machine produces "reads" from all present organisms indiscriminately. One task is to determine which organisms were present in the original sample and in what proportions, information that proves crucial for understanding gut health, disease states, and the effects of diet or medical treatments. You could use this information, for example, to detect colon cancer or other diseases without surgery or other procedures.

Another example are soil samples, where a single gram of soil might contain DNA from thousands of bacterial species, diverse fungal communities, protists, microeukaryotes, plant root cells, and small soil invertebrates. The sequencing process captures genetic material from this entire ecosystem, and therefore generates a dataset of mixed origin. By classifying these "reads", we can understand soil biodiversity, identify beneficial or harmful organisms, and study ecosystem health — with applications in agriculture and farming, or preventing desertification.

The applications extend beyond environmental and medical research into forensic biology, where rapid identification of biological samples becomes essential for criminal investigations. In all these contexts, the fundamental challenge remains the same: given a sequence read, we must efficiently determine its likely organism of origin using a database of known genomic signatures. The k-mer-based approach provides an efficient method for these classification tasks.

Data

For this project, we will work with five well-characterized bacterial genomes that represent different taxonomic groups and genomic properties (Table 1). These genomes can be downloaded from the NCBI database (`https://www.ncbi.nlm.nih.gov/home/download/`) and will serve as our reference database for sequence classification. The species were chosen to represent various levels of genomic similarity and GC content.

Species	Description	NCBI Accession
E. coli K-12 MG1655	Typical lab strain	GCF_000005845.2
B. subtilis 168	Gram-positive, distinct from *E. coli*	GCF_000009045.1
P. aeruginosa PAO1	Different GC content	GCF_000006765.1
S. aureus NCTC 8325	Clinical relevance, distinct genome	GCF_000013425.1
M. tuberculosis H37Rv	High GC content, very distinct	GCF_000195955.2

Table 1: Reference species to be used

You can find the sequencing reads to use in this project at `https://github.com/bio-ontology-research-group/cs249-2025/tree/main/project1-data`. There are 2 paired-end samples with 10,000 reads each, generated from the genomes above:

- `simulated_reads_no_errors_10k_R*.fastq` — reads that do not contain any sequencing errors
- `simulated_reads_miseq_10k_R*.fastq` — reads generated with an Illumina MiSeq error model

Note: For this entire assignment, the algorithms you develop and apply may be too slow to solve the assignment. You are allowed to *subsample* the query patterns (reads) and k-mers (derived from reads and the reference databases). If you use subsampling, you MUST state this clearly in the report.

Task 1: Metagenome Classification by String Matching

In class, we learned about two fundamentally different approaches for finding substrings: preprocessing the patterns, or preprocessing the text in which we search. In the first category, the Aho-Corasick algorithm enables searching multiple patterns simultaneously by constructing a trie data structure from the patterns. In the second category, data structures like suffix tries, trees, or arrays allow us to find substrings by matching prefixes of suffixes.

For this task, you will implement a sequence classification system using these string matching techniques.

Task 1.1: Multiple Matches (5pts)

Consider that we want to classify an entire collection of reads, where many organisms may share identical sequences in their genomes. Develop and justify a strategy for handling reads that match multiple organisms. Your answer should consider both biological reality and computational efficiency.

Task 1.2: Exact Matching (15pts)

For each read in your dataset:

1. Identify all organisms (from the list of 5 organisms above) in which the read matches exactly
2. Implement an efficient string matching algorithms covered in class (Aho-Corasick or suffix-based approaches)
3. Create a systematic way to handle and report matches to multiple reference genomes
4. Generate a report showing how many reads match each of the five organism

Task 1.3: Approximate Matching (optional — 5 bonus pts)

Since sequencing reads often contain errors, we must consider approximate matching:

1. Extend your implementation to find matches with up to one mismatch
2. Use one of the approximate string matching methods discussed in class
3. Report the number of reads matching each reference genome under this criteria

Task 1.4: Comparison with Bioinformatics tools (5pts)

The problem of read mapping (also called alignment) is fundamental in bioinformatics, leading to the development of highly optimized tools. The Basic Local Alignment Search Tool (BLAST) represents an early milestone in this field, becoming one of the most cited papers of the 1990s (over 115,000 citations). Modern tools like BWA, Bowtie2, or Minimap2 offer performance improvements, but use a somewhat different scoring scheme (not mismatches).

For this task, you should use an existing tool and compare your implementation again it.

1. Use BLAST (`blastn`) or, alternatively, bowtie2 to map your reads allowing up to one mismatch.
2. Compare the performance of your implementation with BLAST/bowtie2:
   - Execution time, memory usage (both during search and in precomputed indices)
3. Analyze and explain any differences in the resulting read counts (Hint: you SHOULD read the original BLAST paper)

Task 2: Metagenomic classification by k-mer index

Instead of scanning entire genomes for every read, we can precompute an *index* that maps k-mers to their positions, or, depending on application, their number of occurrences, in the reference genomes. This allows faster lookup when processing the reads. The k-mer index consists of keys (all unique k-mers from the reference genomes), and values (the set of positions at which each k-mer appears across genomes, or the number of occurrences of the k-mer in the reference genome).

Task 2.1: Build the k-mer Index (20pts)

First, build a k-mer index for the 5 reference genomes. Extract all overlapping k-mers from each genome. While you may experiment with different values of k, for evaluation purposes and the report, you must use $k = 31$. Define a data structure that allows you to track the number of occurrences in each of the reference genomes.
Report requirements:

- Briefly describe your data structure and justify your choices
- How many k-mers are in your index?
- How many k-mers of length $k = 31$ are theoretically possible?
- Explain any discrepancy between these numbers

Task 2.2: Implement Classification (20pts)

Classify reads based on exact k-mer matches. For each read, extract all the k-mers it contains and check for exact matches in the k-mer index. Record the reference genome(s) in which the read matches.

Note that k-mers may match more than one organism, and they may also match multiple times *within* each organism. Implement a way to account for this.
Report requirements:

- Output the number of matching reads (k-mers) for each organism
- Is the ratio identical to the approach based on string matching? Explain any discrepancy (if any)
- Justify how you handle k-mers that have multiple matches in one genome

Task 2.3: Minimizers (15pts)

After implementing a k-mer index, you will notice that storing all k-mers requires significant memory. A more efficient approach is to store only a subset of k-mers. The question is which subset to store. A minimizers scheme is defined by three parameters: the k-mer length (k), the window size (w), and the ordering. A window with size of w corresponds to w consecutive k-mers covering a substring of length $w+k-1$ from which a k-mer is selected as the representative called the minimizer. Minimizers are chosen based on an ordering (i.e., sorting) of the k-mers, such as lexicographic. By choosing the "smallest" k-mer as the minimizer, the selection is not based on the k-mer's position but rather based on the sequence content.

Task: Extend your k-mer index with a basic minimizer scheme. You can find more information about minimizers and how to implement them in `https://genomebiology.biomedcentral.com/articles/10.1186/s13059-024-03414-4`. How much does your memory consumption reduce? Compare the classification accuracy between your full k-mer index and the minimizer-based version.

Task 3: Real-world data and tools

The Kraken2 software (`https://pubmed.ncbi.nlm.nih.gov/31779668/`) uses a k-mer index with minimizers to perform classification of reads in metagenomics. This task will allow you to compare your implementation with a widely-used bioinformatics tool.

Task 3.1: Comparison (10pts)

First, build a Kraken2 database using your reference genomes. Follow the instructions in the Kraken2 manual (`https://software.cqls.oregonstate.edu/updates/docs/kraken2/MANUAL.html#custom-databases`) for creating custom databases. Document the commands you used and configuration choices made during database construction.

Second, use this custom Kraken2 database to classify your read dataset. Record the classification results, including species abundances and any ambiguous or unclassified reads.

Perform a comparison between Kraken2's results and those from your implementations from Tasks 1 and 2. Analyze the differences in:

- Species abundance estimates

- Classification speed and resource usage
- Impact of the minimizer scheme on accuracy

Explain any discrepancies between the different approaches and discuss the advantages and limitations of each method.

Task 3.2: Real-world use case (10pts)

Use Kraken2 to taxonomically characterize two groups of real-world samples that are derived from two distinct environments (human gut and wastewater samples). Due to the differences of physical and chemical properties between these two environments, it is to be expected that the microbial composition of samples derived from them will be distinct.

First, download the 'Standard-8' pre-built Kraken2 database (`https://benlangmead.github.io/aws-indexes/k2`).

Second, download metagenomic samples from the Sequence Read Archive ((`https://www.ncbi.nlm.nih.gov/sra`) using their accessions, listed below:

- SRR11412973
- SRR11412976
- SRR11412979
- SRR11412980
- SRR11412984
- SRR21907296
- SRR21907303
- SRR21907307
- SRR21907332
- SRR21907330

Compare the samples using the pre-built Kraken2 Standard-8 database:

- Taxonomically characterize these 10 metagenomic samples.
- Can you find a data-driven way to separate the samples in two classes? Do these two classes correspond to the environments in which the samples were collected?

Task 3.3: Metagenomic classification algorithm (optional — 300 bonus pts)

Carefully read the Kraken2 paper and the minimizer paper from Tasks 2.3 and 3.2. Implement a metagenomics classification system for large datasets. Evaluate the system following the strategy in `https://pmc.ncbi.nlm.nih.gov/articles/PMC6716367/` and apply it to the AllTheBacteria dataset (`https://allthebacteria.readthedocs.io/`). You must have a precision/recall comparable to (or better than) Kraken 2 (Figure 3 in `https://pmc.ncbi.nlm.nih.gov/articles/PMC6716367/`) while improving time or space (or both) requirements.

You can work on this task as a group. All members of the group will receive the bonus points.