

PROJECT CHARTER

Genome Sequence Storage, Query & Pattern Discovery Tool

Course: Data Structures & Algorithms | **Week:** 9 | **Date:** 02 - December 2025

1. DOMAIN

Bioinformatics / Computational Biology

DNA sequences consist of four nucleotides (A, T, G, C) arranged in specific patterns. Analyzing these sequences is fundamental to genetics research, disease diagnosis, drug development, and evolutionary studies. Current challenges include storing massive genomic datasets efficiently and searching for patterns quickly.

2. CLIENT SCENARIO

Client: Genomics Research Laboratory

Problem Statement: Research teams work with thousands of DNA sequences (100-1000 base pairs each) and need to:

- Search for specific gene patterns across multiple sequences
- Identify frequently occurring motifs (biomarkers for diseases)
- Find similar sequences even with minor mutations (1-2 nucleotide differences)
- Query results must return within seconds, not minutes

Current Pain Points:

- Linear scanning through sequences is too slow ($O(n*m)$ comparisons)
 - No efficient way to discover repeated patterns
 - Manual analysis of approximate matches is error-prone
-

3. SCOPE

In-Scope:

- Store multiple DNA sequences in memory efficiently
- Exact pattern matching with position tracking
- Frequent motif discovery using sliding window approach
- Approximate matching with edit distance ≤ 2
- Query interface for pattern searches

Out-of-Scope:

- Persistent database storage (memory-based only)
 - Real-time streaming of sequences
 - Visualization/GUI (command-line interface only)
 - Sequences longer than 10,000 base pairs
-

4. MAIN FEATURES

Feature 1: Compact Sequence Storage

Store DNA sequences using efficient data structures (vectors/arrays) with minimal memory overhead.

Feature 2: Fast Exact Pattern Matching

Build index using trie/prefix tree to search for exact patterns in $O(m)$ time where m = pattern length.

Feature 3: Frequent Motif Mining

Use rolling hash with sliding windows to discover k-mers (subsequences of length k) appearing \geq threshold times in $O(n)$ time.

Feature 4: Approximate Sequence Matching

Find patterns with small differences using edit distance algorithm (dynamic programming) to handle mutations/sequencing errors.

5. DATA STRUCTURES MAPPING

Feature	Data Structure	Justification
Sequence Storage	Dynamic Array/Vector	O(1) access, flexible sizing
Pattern Indexing	Trie (Prefix Tree)	O(m) search for patterns
Motif Counting	Hash Table	O(1) insert/lookup for frequencies
Sliding Window	Queue/Deque	O(1) add/remove from ends
Edit Distance	2D Array (DP table)	Store subproblem solutions

6. SUCCESS CRITERIA

Performance Targets:

- Store 1000 sequences (avg 100bp) in < 2 seconds
- Pattern search completes in < 100ms
- Motif discovery (k=5) completes in < 1 second

Functional Requirements:

- Correctly identify all pattern occurrences
- Find top-N most frequent motifs accurately
- Approximate matching within specified edit distance

7. SAMPLE TEST DATA

Sequence_1 (50 bp):

ATGCGATCGATCGATCGTAGCTAGCTAGCTAGCTAGCTAG

Sequence_2 (50 bp):

GCTAGCTAGCTAGCTAGCATCGATCGATCGTAGCTAGCTA

Expected Motif (k=5): GCTAGC appears 8 times

Expected Search: Pattern "ATCGA" found at positions [4, 8, 12, 16]

8. PROJECT PLAN SUMMARY

Week	Milestone	Deliverable
9	Charter + Team Formation	This document
10	Architecture Presentation	Design & plan slides
11	Core Implementation	MotifFinder + Index modules
12	Integration & Testing	Full demo with queries
13	Finalization	Complete documentation

9. RISKS & MITIGATION

Risk 1: Trie implementation too complex

- *Mitigation:* Start with simpler hash-based approach, add trie incrementally

Risk 2: Performance targets not met

- *Mitigation:* Use small test datasets (100-500 sequences) for demo

Risk 3: Time constraints

- *Mitigation:* Prioritize core features (exact search + motif finding) over bonus features
-

Team Members: Yusra Haris & Fatima Mumtaz

Repository: <https://github.com/yusra-haris034/DSA-Project.git>

Submission Date: Week 9