**NATIONAL UNIVERSITY OF MODERN LANGUAGES ISLAMABAD**



DATA STUCTURES

PBL DOCUMENTATION

|  |  |  |  |
| --- | --- | --- | --- |
| ***Submitted To:*** | *Ma’am Maryiam Imtiaz* | | |
| ***Submitted By:*** | *Aniqa Jamil*  *(SP 23868)* | *Hamna Khan*  *(SP 23856)* | *Jahanzaib*  *(SP 23870)* |
|  | *Nimra Nisab*  *(SP 23869)* | *Rania Nisar*  *(SP 23891)* | *Zulqurnain Haider*  *(SP 23859)* |

**GeneLink: DNA Matching Simulator**

**1. Project Overview**

The **GeneLink: DNA Analysis System** is a C++-based console application designed to analyze genetic relationships, detect mutations, and trace ancestral connections in DNA sequences.

At its core, it leverages the **Levenshtein Distance Algorithm** (also known as Edit Distance) to compute the minimum number of single-character edits insertions, deletions, or substitutions required to transform one DNA sequence into another.

This provides:

* A **quantitative similarity score** between DNA strands.
* Tools to **identify familial links** (e.g., parent-child, siblings).
* A method to **trace genetic mutations** (planned).
* An approach for **ancestral gene detection** by comparing input DNA with evolutionary datasets.

**2. Problem Statement**

Analyzing genetic relationships, mutations, and ancestral connections requires precise computational techniques to measure similarities between DNA strands. Traditional sequence alignment methods often fall short in efficiently capturing minimal evolutionary or mutational differences.

The challenge is to design a **DNA Analysis System** in C++ that:

* Accurately **calculates genetic similarity** using the minimum number of edits.
* Determines **biological relationships** between individuals through dataset comparisons.
* **Identifies mutations** by detecting specific base-pair changes.
* **Traces ancestral gene conservation** by identifying preserved genetic regions over generations.

**Key Issues to Address**

✅ Efficient implementation of the Levenshtein algorithm for long DNA sequences.  
✅ Accurate sequence comparison to minimize false positives.  
✅ Scalable design for handling large genomic datasets.  
✅ User-friendly input/output for biologists and researchers.

**Expected Outcome**

A robust, modular C++ system that:

* Accurately computes DNA sequence similarity using dynamic programming.
* Provides clear visual alignments and similarity scores.
* Offers a foundation for mutation detection and evolutionary analysis.

**3. Project Analysis**

GeneLink is structured into **three major modules**:

**Module 1: DNA Similarity Comparison (Manual Selection)**

* **Purpose:** Compare DNA sequences between two or more individuals.
* **Key Components:**
  + Uses map<string, string> to store name–DNA pairs.
  + Applies the Levenshtein distance algorithm to calculate similarity.
  + Provides visual alignment with color-coded matches/mismatches.
  + Allows manual pairwise comparison or single-vs-all best-match detection.

**Module 2: Ancestral Gene Detection (File-Based Matching)**

* **Purpose:** Compare user-input DNA against ancestral records stored in a .txt or .csv file.
* **Key Components:**
  + Reads and parses files; skips malformed lines.
  + Stores results as vector<pair<string, float>> for sorting by similarity score.
  + Highlights top ancestral matches with visual sequence alignments.

**Module 3: Mutation Detection (Planned)**

* **Purpose:** Identify and report detailed mutations between two DNA sequences.
* **Key Components:**
  + Uses traceback on the Levenshtein matrix to detect:
    - Substitutions (e.g., A → G).
    - Insertions (+T).
    - Deletions (–C).
  + Generates mutation reports with location, type, and involved characters.

**Technical Highlights**

* **Data Structures Used:**
  + Map (map<string, string>) → DNA profiles.
  + 2D Vector (vector<vector<int>>) → Dynamic programming matrix.
  + Pair List (pair<string, float>) → Similarity scoring and sorting.
* **String Algorithms:**
  + Levenshtein Distance Algorithm, accounting for insert, delete, substitute operations.
* **User Interface:**
  + Command-line interface (CLI) with ANSI colors, progress bars, and clear visuals.
* **Code Structure:**
  + Modular, function-based design (non-OOP) for clarity and reusability.

**4. Algorithms**

**DNA Similarity Comparison Algorithm**

* **Input:** Two DNA sequences, seq1 and seq2.
* **Output:** Similarity score (%) and visual alignment.

**Steps:**

1. Initialize dynamic programming (DP) table dp[m+1][n+1] (m = length of seq1, n = length of seq2).
2. Fill base cases:
   * dp[i][0] = i (deletions).
   * dp[0][j] = j (insertions).
3. For each i, j:
   * If seq1[i-1] == seq2[j-1]:  
     → dp[i][j] = dp[i-1][j-1].
   * Else:  
     → dp[i][j] = 1 + min(dp[i-1][j], dp[i][j-1], dp[i-1][j-1]).
4. Compute similarity:
   * similarity = 100 - (dp[m][n] / max(m, n)) \* 100.
5. Display colored alignment.

**Ancestral Gene Detection Algorithm**

* **Input:** User DNA (userDNA) and ancestral DNA records from file.
* **Output:** Top ancestral match and sorted similarity list.

**Steps:**

1. Read file, parse Name, DNA pairs into a list.
2. For each ancestral DNA:
   * Compute Levenshtein distance and similarity score with userDNA.
   * Store (Name, Score) pairs.
3. Sort list by similarity score (descending).
4. Display top match with alignment.

**Mutation Detection Algorithm (Planned)**

* **Input:** Two DNA sequences, seq1 and seq2.
* **Output:** Detailed list of mutations.

**Steps:**

1. Build DP matrix with traceback path.
2. Trace from dp[m][n] back to dp[0][0].
3. Identify:
   * Substitutions (diagonal move with cost).
   * Insertions (left move).
   * Deletions (up move).
4. Report mutation type, position, and characters.

**5. Conclusion**

The **GeneLink** offers an educational, functional, and extendable C++ system combining key DSA concepts with real-world genetic analysis.

By integrating the Levenshtein Distance Algorithm into practical modules, it allows:

* Accurate detection of genetic similarity.
* Explorations of evolutionary and ancestral connections.
* A foundation for future development of mutation detection tools.

This project not only enhances understanding of string-based algorithms but also bridges computer science with biological research, offering a promising tool for students, educators, and researchers alike.