DNA Profiling: Forensic Analysis

Overview

You will use Binary Search Trees (BSTs) to analyze DNA. You are tasked with identifying who a DNA sequence may belong to using an unknown person's DNA sequence and a provided DNA database. This exercise focuses on the 0.1% of DNA that varies between humans, specifically examining Short Tandem Repeats (STRs).

The primary goal is to efficiently analyze STR data using BST operations. The methods implemented will determine hereditary relationships and flag profiles of interest.

Why Use BSTs?

Binary Search Trees allow efficient insertion, deletion, and search operations (O(log n)) compared to arrays (O(n)). This assignment leverages BSTs to handle DNA profiles efficiently.

Files Provided for Implementation

- Profile class: Creates DNA profiles, storing STR data.
- ForensicAnalysis class: Contains methods to manipulate BSTs for STR analysis.
- Driver class: Used for testing; prompts interactively in the terminal.
- TreeNode class: Represents BST nodes with keys and associated profiles.
- Queue class: Implements enqueue and dequeue operations.
- StdIn/StdOut: Libraries for input and output operations.

Key Methods to Implement in ForensicAnalysis.java

- createSingleProfile: Reads DNA data for a person and creates a profile.
- insertPerson: Inserts a person's profile into the BST.
- markProfileOfInterest: Flags specific profiles based on STR conditions.
- searchByName: Finds and returns a profile by name from the BST.
- listAllProfiles: Lists all profiles in BST order.

Example Input and Output

Example input files include DNA sequences and STR counts. The output includes flagged profiles,
BST traversal results, and identified matches based on STR criteria.