PROJECT REPORT: DNA MOTIF FINDING ALGORITHM (SUBS)

Date: February 26, 2026

Subject: Computational Biology

Environment: Python 3.10+, VS Code

1. Introduction

The objective of this project is to implement an efficient solution for the **"Finding a Motif in DNA" (SUBS)** problem from the Rosalind platform. The algorithm identifies all 1-based starting positions of a given sub-sequence (motif) within a primary DNA sequence, accounting for overlapping occurrences.

2. Methodology

The application is designed using a **modular programming approach** in Python. By separating the core logic from the user interface, the code achieves high readability and maintainability.

2.1. Algorithm Logic

The core function utilizes a sliding window technique. It iterates through the DNA string and extracts a slice of length k (where k is the length of the motif) to compare it with the target pattern.

$$\text{Range of iteration} = L_{\text{sequence}} - L_{\text{motif}} + 1$$

2.2. Features

- **Modular Functions:** Dedicated functions for manual input, sample testing, and UI display.
- **Input Standardization:** Automatic stripping of whitespaces and conversion to uppercase to prevent input errors.
- **1-Based Indexing:** Adjusted output format to match bioinformatics standards.

3. Implementation (Python Code)

Python
```
def find_motif_indices(sequence, motif):
    """
    Core Logic: Finds all 1-based starting positions of a motif in a
DNA sequence.
    """
    indices = []
    for i in range(len(sequence) - len(motif) + 1):
        if sequence[i : i + len(motif)] == motif:
            indices.append(i + 1)
    return ' '.join(map(str, indices))

def run_sample():
    """Handles the predefined sample dataset."""
    sample_seq = "GATATATGCATATACTT"
    sample_motif = "ATAT"
    result = find_motif_indices(sample_seq, sample_motif)
    print(f"Sample Result: {result}")

def run_manual_input():
    """Handles user-provided input from the terminal."""
    seq = input("Enter DNA sequence (s): ").strip().upper()
    motif = input("Enter motif to find (t): ").strip().upper()
    if seq and motif:
        print(f"Result: {find_motif_indices(seq, motif)}")

def main():
    """Main program loop and menu management."""
    while True:
        print("\n1. Run Sample | 2. Manual Input | 3. Exit")
        choice = input("Select: ").strip()
        if choice == "1": run_sample()
        elif choice == "2": run_manual_input()
        elif choice == "3": break
```

4. Testing and Results

The algorithm was tested using the standard Rosalind sample dataset.

Input Sequence (s)	**Motif (t)**	**Expected Output**	**Status**
`GATATATGCATATACTT`	`ATAT`	`2 4 10`	**Passed**
`ACGTACGT`	`AC`	`1 5`	**Passed**

5. Conclusion

The implementation successfully handles overlapping motifs and provides a clean command-line interface for the user. Future iterations could include support for large-scale FASTA file processing and complexity analysis for extremely long genomic sequences.