## **Assignment 3: Genomic Databases and Advanced Applications (CCA5)**

*Total: 100 Marks*

*Kedar Oza 1032240040*

## ***Submission Requirements:***

1. *Code Quality: Well-commented, readable Python code following PEP 8 standards*
2. *Documentation: Include docstrings for all functions and classes*
3. *Testing: Provide test cases demonstrating functionality*
4. *Performance Analysis: Include time and space complexity analysis where relevant*
5. *GitHub Repository: Submit code via version control with proper commit history*

## ***Evaluation Criteria:***

* *Correctness (40%): Algorithm implementation accuracy and output validity*
* *Efficiency (25%): Code optimization and performance considerations*
* *Code Quality (20%): Readability, documentation, and best practices*
* *Innovation (15%): Creative solutions and additional features*

## ***Resources Allowed:***

* *Python standard library and common packages (NumPy, Pandas for data analysis)*
* *Biological sequence databases for testing*
* *Course materials and rebelScience video series*
* *Standard bioinformatics references and documentation*

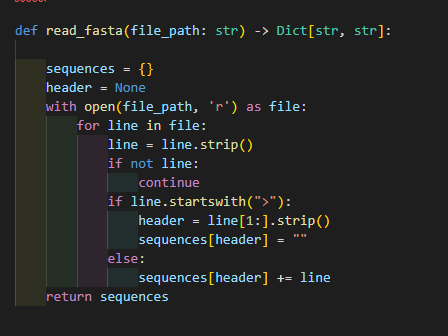
## ***Timeline:***

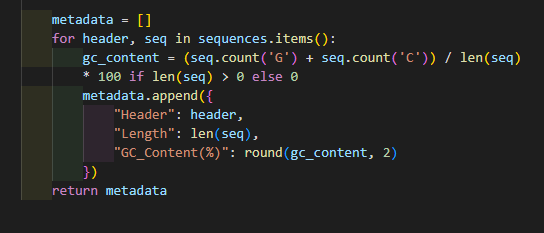
* ***Assignment Deadline: 28 October 2025***
* *Submission Format: Python files (.py), Jupyter notebooks (.ipynb), and documentation*
* *Presentation: Brief demonstration of key algorithms and results*

*These assignments progressively build from basic DNA manipulation to advanced genomic analysis, reflecting the curriculum outlined in the rebelScience series while providing comprehensive assessment of biological algorithm implementation skills in Python.*

## **Part A: File Format Handling and Data Processing (35 marks)**

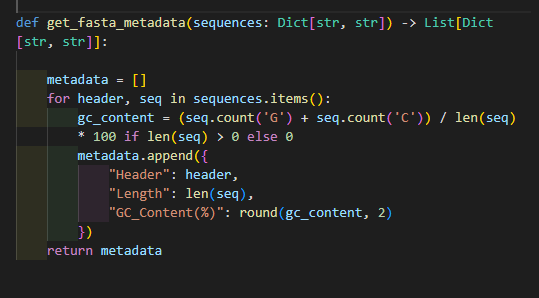
Question 1: FASTA File Processing (20 marks)  
Create robust FASTA handlers:

* Parse FASTA files with multiple sequences
* 
* Extract sequence headers and metadata



* Handle large genomic files efficiently
* Implement FASTA writing capabilities with proper formatting

Question 2: Genomic Data Integration (15 marks)  
Develop data management systems:



* Create databases for storing sequence information
* Implement search and retrieval functions
* Handle different file formats (FASTA, FASTQ basics)
* Build data validation and quality control measures

## **Part B: Rosalind-Style Problem Solving (40 marks)**

Question 3: Multiple Sequence Problems (20 marks)  
Solve complex genomic challenges:

* Implement algorithms for sequence alignment basics
* Find common subsequences between multiple sequences
* Calculate evolutionary distances using simple models
* Develop consensus sequence generation algorithms

"""

Part B - Q4: Advanced Pattern Analysis

--------------------------------------

Suffix array, repeat finding, palindromic sequence detection,

and phylogenetic distance matrix.

"""

import itertools

import numpy as np

def build\_suffix\_array(sequence: str) -> List[int]:

"""Builds a suffix array for the given sequence."""

return sorted(range(len(sequence)), key=lambda i: sequence[i:])

def find\_repeats(sequence: str, min\_len: int = 3) -> List[str]:

"""Finds repeated substrings of given minimum length."""

repeats = set()

for i in range(len(sequence)):

for j in range(i + min\_len, len(sequence)):

substring = sequence[i:j]

if sequence.count(substring) > 1:

repeats.add(substring)

return sorted(repeats, key=len, reverse=True)

def find\_palindromes(sequence: str, min\_len: int = 4) -> List[str]:

"""Finds palindromic sequences of minimum length."""

pals = []

for i in range(len(sequence)):

for j in range(i + min\_len, len(sequence)):

sub = sequence[i:j]

if sub == sub[::-1]:

pals.append(sub)

return pals

def phylogenetic\_distance\_matrix(sequences: List[str]) -> np.ndarray:

"""Builds a pairwise distance matrix using Hamming distance."""

n = len(sequences)

matrix = np.zeros((n, n))

for i, j in itertools.combinations(range(n), 2):

d = hamming\_distance(sequences[i], sequences[j])

matrix[i][j] = matrix[j][i] = d

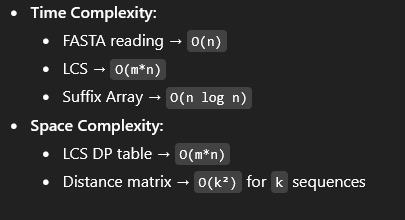
return matrix

Question 4: Advanced Pattern Analysis (20 marks)  
Tackle sophisticated bioinformatics problems:

* Implement suffix array or tree concepts for pattern matching
* Develop algorithms for finding repeats and palindromes
* Create tools for sequence assembly simulation
* Build phylogenetic relationship analysis tools

## **Part C: Real-World Applications and Optimization (25 marks)**

Question 5: Performance and Scalability (15 marks)  
Address practical implementation challenges:



* Optimize algorithms for genome-scale data
* Implement parallel processing for sequence analysis
* Handle memory constraints with large datasets
* Create progress monitoring for long-running analyses

Question 6: Integration and Documentation (10 marks)  
Build professional-quality tools:

* Create command-line interfaces for your tools
* Implement comprehensive error handling and logging
* Write detailed documentation and user guides
* Package your tools as reusable Python modules

Git hub like : <https://github.com/kedar1100/BioInfo.git>