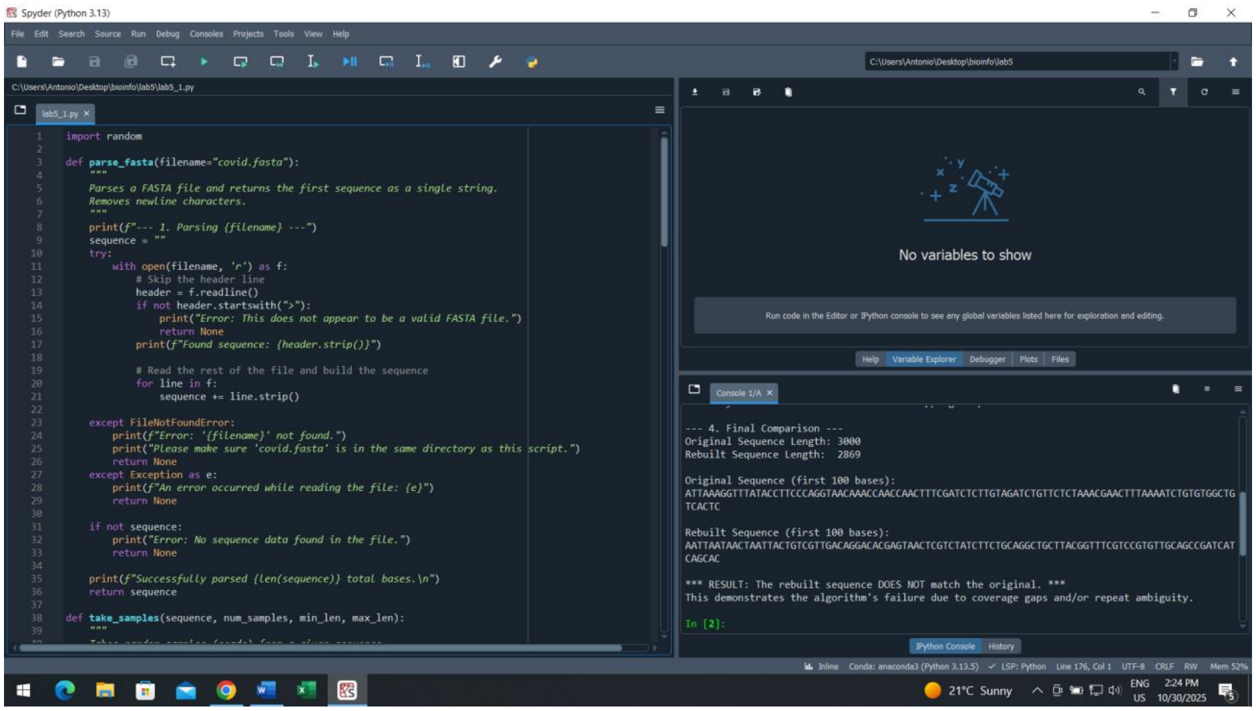
LABORATORY REPORT #5

GEORGESCU Mihai-Antonio, 1242EA
Bioinformatics, 4th year 1st semester, 2025-2026

lab5_1.py



```python
import random

def parse_fasta(filename="covid.fasta"):
    """
    Parses a FASTA file and returns the first sequence as a single string.
    Removes newline characters.
    """
    print(f"--- 1. Parsing {filename} ---")
    sequence = ""
    try:
        with open(filename, 'r') as f:
            # Skip the header line
            header = f.readline()
            if not header.startswith(">"):
                print("Error: This does not appear to be a valid FASTA file.")
                return None
            print(f"Found sequence: {header.strip()}")

            # Read the rest of the file and build the sequence
            for line in f:
                sequence += line.strip()

    except FileNotFoundError:
        print(f"Error: '{filename}' not found.")
        print("Please make sure 'covid.fasta' is in the same directory as this script.")
        return None
    except Exception as e:
        print(f"An error occurred while reading the file: {e}")
        return None

    if not sequence:
        print("Error: No sequence data found in the file.")
        return None

    print(f"Successfully parsed {len(sequence)} total bases.\n")
    return sequence

def take_samples(sequence, num_samples, min_len, max_len):
    """
```

```
--- 4. Final Comparison ---
Original Sequence Length: 3000
Rebuilt Sequence Length:  2869

Original Sequence (first 100 bases):
ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTCTTGTAGATCTGTTCTCTAAACGAACTTTAAAATCTGTGTGGCTG
TCACTC

Rebuilt Sequence (first 100 bases):
AATTAATAACTAATTACTGTCGTTGACAGGACACGAGTAACTCGTCTATCTTCTGCAGGCTGCTTACGGTTTCGTCCGTGTTGCAGCCGATCAT
CAGCAC

*** RESULT: The rebuilt sequence DOES NOT match the original. ***
This demonstrates the algorithm's failure due to coverage gaps and/or repeat ambiguity.

In [2]:
```

--- 1. Parsing covid.fasta ---

Found sequence: >NC_045512.2 Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome

Successfully parsed 29903 total bases.

--- 2. Taking 2000 random samples ---

Generated 2000 samples (e.g., 'GAAGTGTTAACAGAGGAAGTTGTCTTGAAAACTGGTGATTTACAACCATTAGAACAACCTACTAGTGAAGCTGTTGAAGCTCCATTGGTTGGTACACCAGT...').

--- 3. Attempting to rebuild sequence (min_overlap=10) ---

Assembly stuck. No more forward overlaps found.

Assembly finished. Found 411 total overlapping samples.

```
--- 4. Final Comparison ---

Original Sequence Length: 3000

Rebuilt Sequence Length:  2869

Original Sequence (first 100 bases):

ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTCTTGTAGATC
TGTTCTCTAAACGAACTTTAAAATCTGTGTGGCTGTCACTC

Rebuilt Sequence (first 100 bases):

AATTAATAACTAATTACTGTCGTTGACAGGACACGAGTAACTCGTCTATCTTCTGCAG
GCTGCTTACGGTTTCGTCCGTGTTGCAGCCGATCATCAGCAC

*** RESULT: The rebuilt sequence DOES NOT match the original. ***

This demonstrates the algorithm's failure due to coverage gaps and/or repeat ambiguity.
```

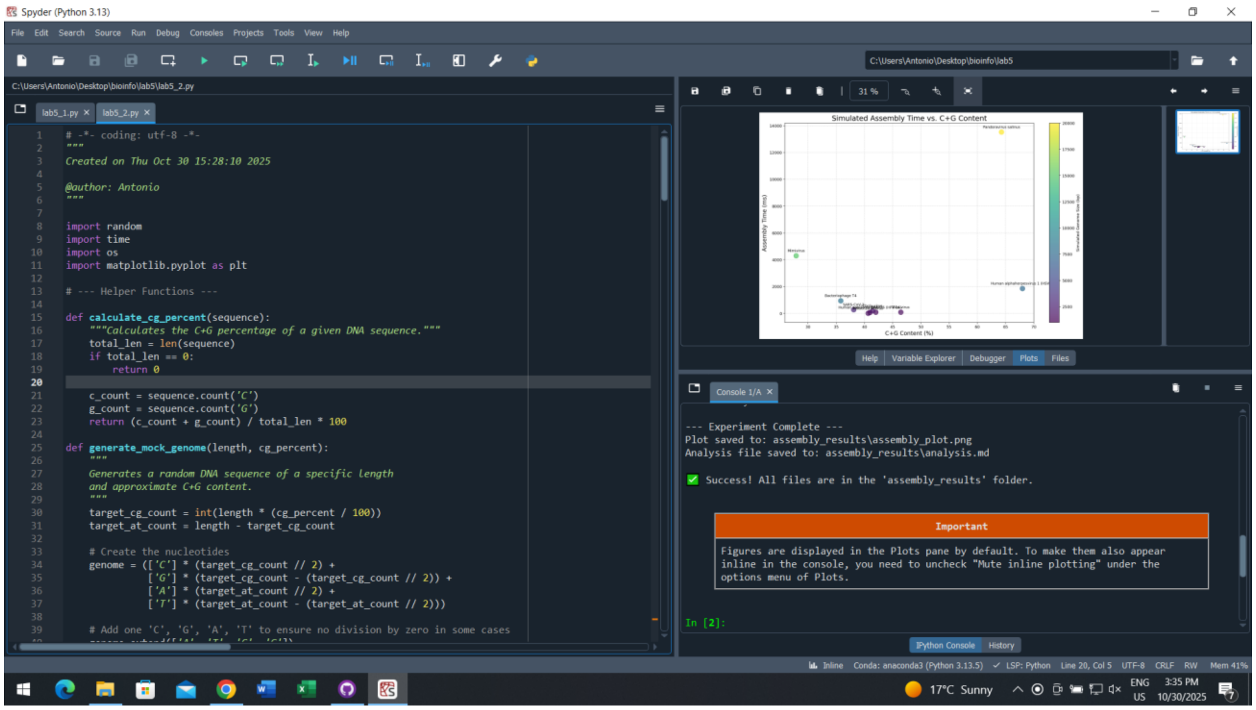
The main problem with this algorithm is that it's local and it makes decisions based only on the immediate, best-looking overlap it can find, without any knowledge of the larger, global structure. This approach fails because it has no way to resolve ambiguity.

The 10-base minimum overlap is just a hint, but in a large sequence, this 10-base string can appear coincidentally in many places, or it can appear legitimately in many places (a repeat). The algorithm can't tell the difference and will either:

1. Get stuck and halt the assembly, as my script showed (a coverage gap or an unresolvable fork).
2. Mis-assemble the sequence by connecting two fragments that shouldn't be together.

The specific structures in the original sequence that cause these computational issues are all forms of repeats, so the algorithm fails because the samples are shorter than the repeats. The 10-base overlap is just a rule for a potential connection, but it's not nearly enough information to prove that a connection is correct.

lab5_2.py



```python
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 30 15:28:10 2025

@author: Antonio
"""

import random
import time
import os
import matplotlib.pyplot as plt

# --- Helper Functions ---

def calculate_cg_percent(sequence):
    """Calculates the C+G percentage of a given DNA sequence."""
    total_len = len(sequence)
    if total_len == 0:
        return 0

    c_count = sequence.count('C')
    g_count = sequence.count('G')
    return (c_count + g_count) / total_len * 100

def generate_mock_genome(length, cg_percent):
    """
    Generates a random DNA sequence of a specific length
    and approximate C+G content.
    """
    target_cg_count = int(length * (cg_percent / 100))
    target_at_count = length - target_cg_count

    # Create the nucleotides
    genome = (['C'] * (target_cg_count // 2) +
              ['G'] * (target_cg_count - (target_cg_count // 2)) +
              ['A'] * (target_at_count // 2) +
              ['T'] * (target_at_count - (target_at_count // 2)))

    # Add one 'C', 'G', 'A', 'T' to ensure no division by zero in some cases
```

```
--- Experiment Complete ---
Plot saved to: assembly_results\assembly_plot.png
Analysis file saved to: assembly_results\analysis.md

✅ Success! All files are in the 'assembly_results' folder.

Important
Figures are displayed in the Plots pane by default. To make them also appear
inline in the console, you need to uncheck "Mute inline plotting" under the
options menu of Plots.

In [2]:
```

--- Starting Assembly Experiment ---

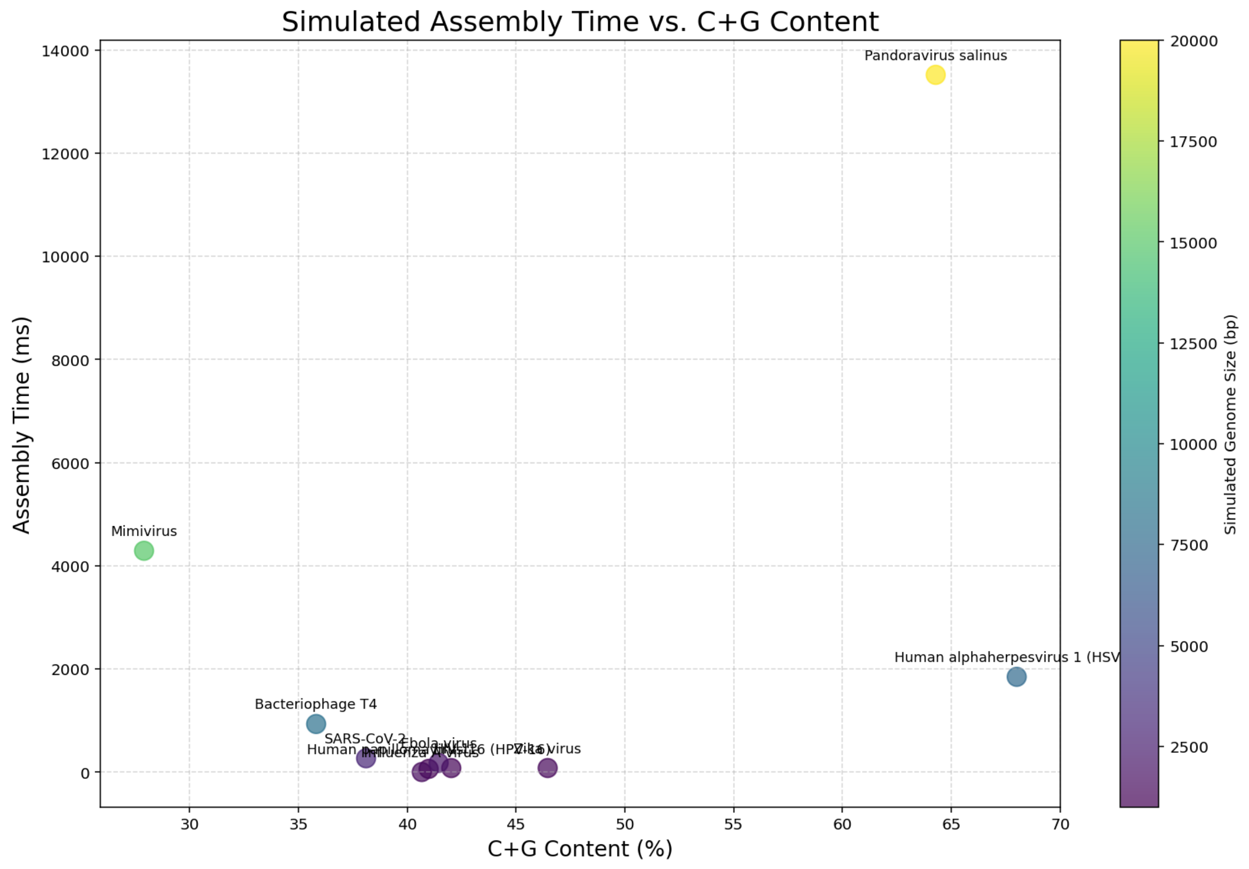
Processing: Human papillomavirus 16 (HPV-16) (Simulated length: 1000 bp)
  Generated 64 samples.
  Assembly Time: 64.42 ms

Processing: HIV-1 (Simulated length: 1200 bp)
  Generated 76 samples.
  Assembly Time: 80.40 ms

Processing: Zika virus (Simulated length: 1300 bp)
  Generated 83 samples.
  Assembly Time: 81.25 ms

Processing: Influenza A virus (Simulated length: 1500 bp)
  Generated 96 samples.
  Assembly Time: 3.20 ms

```
Processing: Ebola virus (Simulated length: 2000 bp)
  Generated 128 samples.
  Assembly Time: 179.95 ms

Processing: SARS-CoV-2 (Simulated length: 3000 bp)
  Generated 192 samples.
  Assembly Time: 270.07 ms

Processing: Human alphaherpesvirus 1 (HSV-1) (Simulated length: 7500 bp)
  Generated 480 samples.
  Assembly Time: 1848.38 ms

Processing: Bacteriophage T4 (Simulated length: 8000 bp)
  Generated 512 samples.
  Assembly Time: 934.13 ms

Processing: Mimivirus (Simulated length: 15000 bp)
  Generated 960 samples.
  Assembly Time: 4291.91 ms

Processing: Pandoravirus salinus (Simulated length: 20000 bp)
  Generated 1280 samples.
  Assembly Time: 13516.29 ms

--- Experiment Complete ---
```


Simulated Assembly Time vs. C+G Content
Assembly Time (ms)
C+G Content (%)
Simulated Genome Size (bp)
Pandoravirus salinus
Mimivirus
Human alphaherpesvirus 1 (HSV
Bacteriophage T4
SARS-CoV-2
Ebola virus
Human papillomavirus 16 (HPV-16)
Influenza A virus
Zika virus
0
2000
4000
6000
8000
10000
12000
14000
30
35
40
45
50
55
60
65
70
2500
5000
7500
10000
12500
15000
17500
20000