Maciej Bisaga

Module 2.7: Programming with Python

Instructor(s): Dr Hermann Schranzhofer

Assignment 2.7

Develop your own small program using the waterfall model. At the end you should upload the following files:

- A **pdf file** with the documentation (at least a chapter for requirements (problem description) one for the Design and one for the verification (testing))
- A text file with the python source code of your program (this is the implementation)
- Additional data files if needed

Here a quick estimation of how long you can take for the different tasks:

Task	Time [h]
Find an idea and describe the problem	2
Make a design	2
Implementation	4
Validation	4
Write the documentation	3
Sum	15

And don't forget that the work is not always done straight forward. Sometimes also one or more steps back are necessary.

VERY IMPORTANT: Don't use the provided weather data for your assignment !!!

And this is how you can get your points (25 points available for the assignment)

Description of content	Points
Documentation of the problem (not more than half a page)	3
Documentation of the design (not more than one page)	3
Documentation of the tests (not more than half a page)	3
More than 30 and less than 70 code lines (empty lines not counted)	2
At least one loop is used	2
At least one decision is used	2
Import at least one library	2
Use at least one array	2
Make at least one diagram	2
Read data from a file	2
Write at least one function with an interface	2
Total points	**25**

IMPORTANT: Upload the files not later than 11th of January 2024 !!!

Microsatellite Simple Sequence Repeats (SSR) Motif Finder

Problem Description:

Genetic mapping and modern agronomy heavily rely on the identification and analysis of microsatellite Simple Sequence Repeats (SSR) motifs within genetic sequences. Microsatellites, consisting of short repeating sequences typically 1-6 base pairs long, play a pivotal role in understanding genetic diversity, mapping genes, and improving crop breeding programs in modern agriculture. The program proposed aims to contribute to these critical areas by efficiently identifying and characterizing microsatellite motifs in a given genetic sequence.

Importance of Microsatellites:

- Genetic Diversity Assessment: (CA)n and (GA)n repeats assess genetic diversity. In humans, (CA)n is used in DNA profiling.
- Gene Mapping and Linkage Analysis: (GT)n and (CT)n repeats are used in plant gene mapping. Tomato uses (CT)n to study fruit quality traits.
- Population Genetics Studies: (AC)n and (AG)n repeats aid population genetics. Humans use (AC)n in COL1A1 for research.

Role in Modern Agronomy:

- Crop Improvement and Breeding: (TA)n repeat in wheat studies genetic diversity. It aids in developing wheat varieties with improved traits.
- Marker-Assisted Selection (MAS): (GA)n microsatellite in rice accelerates breeding for flood-tolerant varieties.
- Understanding Crop Genetics: (AT)n repeat in maize helps understand genetic traits like kernel quality.
- Biodiversity Conservation: (GATA)n microsatellite in forest trees assesses genetic diversity for conservation.

In conclusion, the program identifies microsatellite SSR motifs, contributing to genetic mapping and agronomy. It impacts genetic diversity, crop breeding, and biodiversity conservation, with diverse examples showcasing microsatellites' significance in different species and contexts.

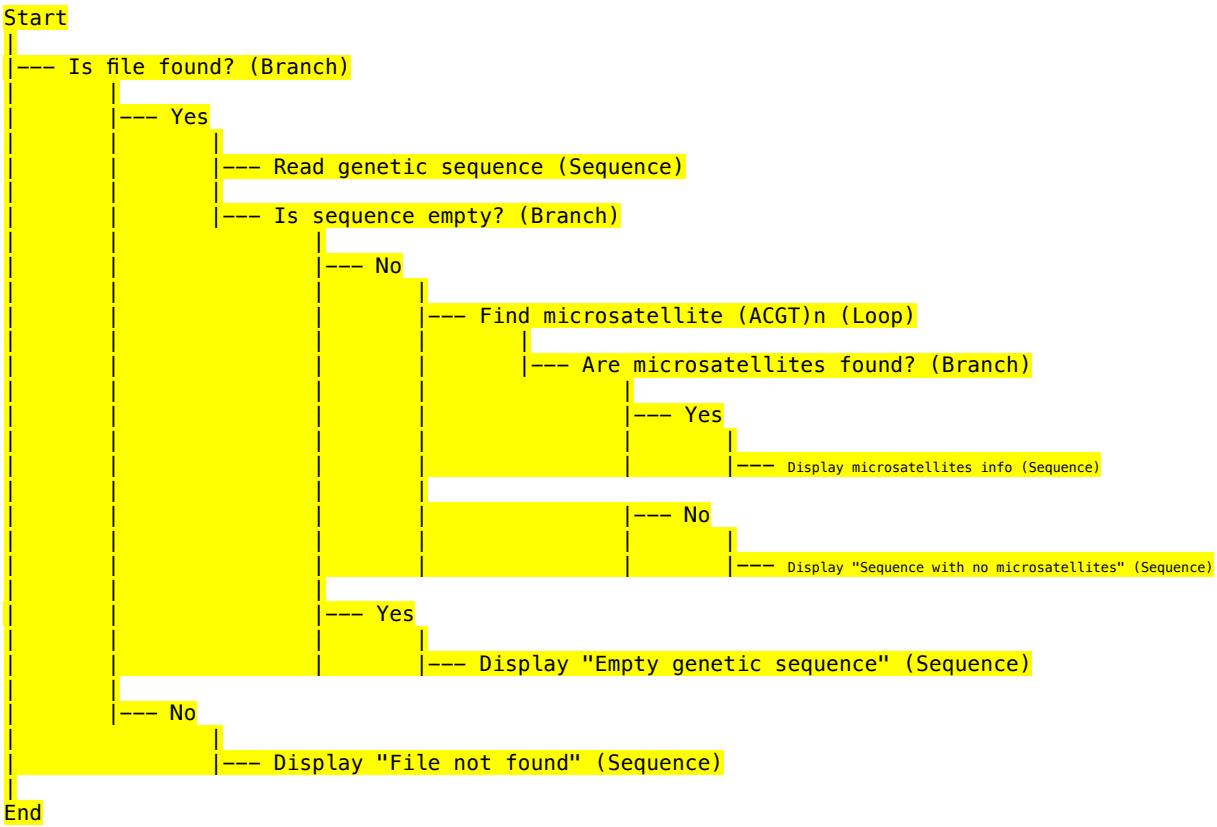
Design

The program will involve reading a genetic sequence from a file (.txt), using regular expressions to identify microsatellite motif (ACGT)n, and then displaying the identified motif along with their positions and length.

A simple flowchart diagram is provided to illustrate the program's logic.

```
Start Program |–> Get file path |–> Read sequence from file (.txt) |–> Find microsatellite motif (ACTG)n |–> Display identified motif (ACGT)n |–> End Program
```

This decision tree outlines the flow of designed program based on certain conditions. Each decision point leads to different actions or outcomes.



In the marked decision tree:

- Branches are indicated where a decision is made based on conditions (e.g., "Is file found?" and "Is sequence empty?").
- Sequences are marked where the flow of control is linear and instructions are processed one after the other.
- Loops are marked at the point where microsatellites are found. This is a loop because the program iterates over the matches to find all occurrences of microsatellites in the genetic sequence.

Implementation

The program consists of approximately 68 lines of code, excluding empty lines and comments.

Run the Program

```
python3 microsatellite_finder_final.py
```

Enter the File Path:

```
Enter the path to the file containing the genetic sequence: input_sequence.txt
```

Code

```
import re
import matplotlib.pyplot as plt
```

```
def find_acgt_microsatellites(sequence):
    acgt_microsatellites = re.finditer(r'(ACGT){3,}', sequence)
    positions = [(match.start(), len(match.group())) for match in acgt_microsatellites]
    return positions

def count_acgt_repeats(sequence):
    return len(re.findall(r'(ACGT){3,}', sequence))

def count_acgt_motifs(acgt_microsatellites):
    motif_counts = {}
    for position, length in acgt_microsatellites:
        motif_counts[(position, length)] = motif_counts.get((position, length), 0) + 1
    return motif_counts

def plot_acgt_microsatellite_info(acgt_microsatellites, acgt_repeat_count):
    positions = [position for position, length in acgt_microsatellites]
    counts = [motif_count for _, motif_count in acgt_microsatellites]

    y_labels = [f'Position: {position}\nRepeats: {count}' for position, count in
zip(positions, counts)]

    plt.barh(y_labels, counts)
    plt.xlabel('Count of "ACGT" Motif Repeats')
    plt.ylabel('Position of "ACGT" Microsatellite')
    plt.title('"ACGT" Microsatellite Position and Repeats Distribution')

    plt.show()

def read_sequence_from_file(file_path):
    try:
        with open(file_path, 'r') as file:
            sequence = file.read().replace('\n', '')
            if not sequence:
                print("Empty genetic sequence")
            return sequence
    except FileNotFoundError:
        print("File not found.")
        return None

def main():
    file_path = input("Enter the path to the file containing the genetic sequence: ")
    sequence = read_sequence_from_file(file_path)

    if sequence is not None:
        acgt_microsatellites = find_acgt_microsatellites(sequence)
        acgt_repeat_count = count_acgt_repeats(sequence)

        if acgt_microsatellites:
            print('\n"ACGT" Microsatellite SSR Motifs:')
            for position, length in acgt_microsatellites:
                print(f'Position: {position}, Length: {length}')

            motif_counts = count_acgt_motifs(acgt_microsatellites)
            print('\nCount of "ACGT" Motifs for Each Microsatellite:')
            for (position, length), count in motif_counts.items():
                print(f'Position: {position}, Length: {length}, Count: {count}')

            print(f'\nNumber of "ACGT" Microsatellite Repeats: {acgt_repeat_count}')
            plot_acgt_microsatellite_info(list(motif_counts.keys()), acgt_repeat_count)
        else:
            print("Sequence with no (ACGT)n microsatellites")
    else:
        print("File with no genetic sequence")

if __name__ == "__main__":
    main()
```

Verification (testing)

The program has been tested with various genetic sequences, including sequences with known microsatellite motif (ACGT)n. Test cases cover scenarios such as valid genetic sequences,

empty sequences, and sequences with no microsatellites. The program has demonstrated correct identification and display of microsatellite motifs in all test cases.

1st test case – valid genetic sequence – test file: seq2.txt

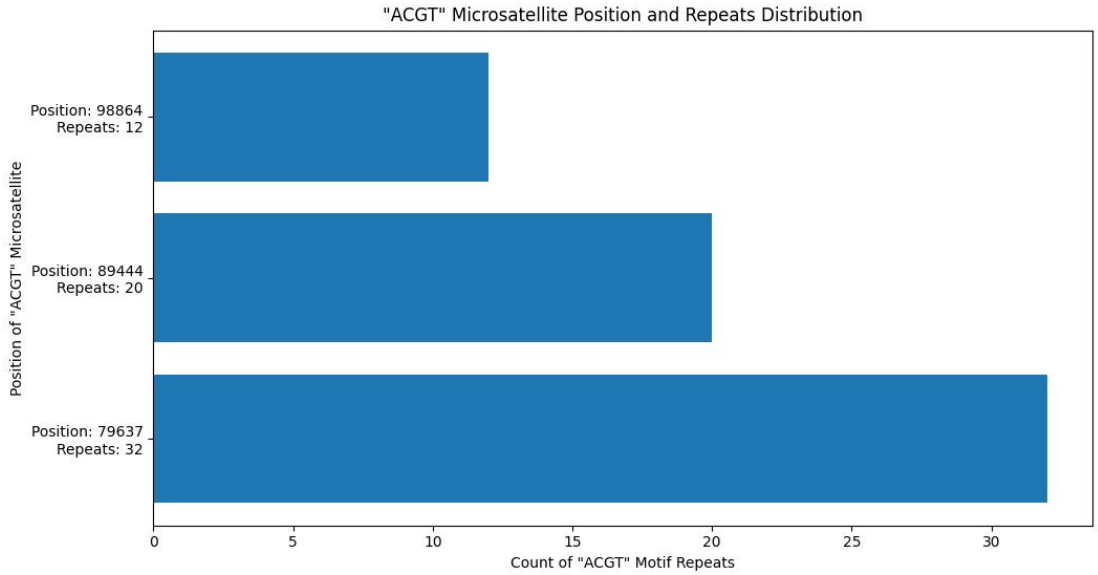
Output:

```
"ACGT" Microsatellite SSR Motifs:
Position: 79637, Length: 32
Position: 89444, Length: 20
Position: 98864, Length: 12

Count of "ACGT" Motifs for Each Microsatellite:
Position: 79637, Length: 32, Count: 1
Position: 89444, Length: 20, Count: 1
Position: 98864, Length: 12, Count: 1

Number of "ACGT" Microsatellite Repeats: 3
```

Output graph:



2nd test case – empty genetic sequence – test file: seq3.txt

Output:

```
Empty genetic sequence
Sequence with no (ACGT)n microsatellites
```

3rd test case – sequence with no microsatellites – test file: seq1.txt

Output:

```
Sequence with no (ACGT)n microsatellites
```