**1. Цель работы**

Определить процентное содержание гуанина и цитозина (GC-состав) в последовательностях ДНК, хранящихся в FASTA-файлах, и найти последовательность с наибольшим GC-составом среди всех анализируемых файлов.

**2. Задачи**

Для достижения цели необходимо решить следующие задачи:

1. Прочитать все файлы с расширением .fasta в текущей директории.
2. Распарсить содержимое каждого файла в формате FASTA с использованием библиотеки Biopython.
3. Для каждой последовательности ДНК:
   * вычислить GC-долю (долю гуанина и цитозина),
   * сохранить значение в словарь, где ключ— это идентификатор последовательности, а значение — GC-доля.
4. Найти максимальное значение GC-доли среди всех последовательностей.
5. Вывести на экран идентификатор(ы) последовательности(й) с наибольшей GC-долей и само значение.

**3. Ход работы (подробно)**

**Шаг 1: Импорт необходимых библиотек**

**glob** — стандартный модуль Python для поиска файлов по шаблону. Используется для нахождения всех файлов с расширением .fasta в текущей директории. **Bio.SeqIO** — модуль библиотеки Biopython для чтения/записи последовательностей в биоинформатических форматах, таких как FASTA.

**gc\_fraction** (переименован в GC) функция, которая рассчитывает долю GC в последовательности (где G — гуанин, C — цитозин). Возвращает значение от 0 до 1.

Шаг 2: Определение функции gc\_ratio(filename)

Вход: имя FASTA-файла.

Процесс:

1. Файл открывается для чтения.
2. С помощью SeqIO.parse() читаются все последовательности из файла (может быть одна или несколько).
3. Для каждой последовательности:
   * Сохраняется сама ДНК-цепочка: record.seq.
   * Рассчитывается GC-доля: GC(dna).
   * Результат округляется до 6 знаков после запятой.
   * Результат сохраняется в словарь: ключ — record.id, значение — GC-доля.

Выход: словарь вида {идентификатор: GC-доля}.

Шаг 3: Получение списка файлов

С помощью glob.glob() создаётся список всех файлов с расширением .fasta в текущем каталоге. Это позволяет автоматически обрабатывать любое количество файлов.

Шаг 4: Обработка всех файлов и сбор результатов

Инициализируется общий словарь all\_sequences\_ratio, в который будут собраны все данные.

Для каждого FASTA-файла вызывается функция gc\_ratio().

Результаты--(словарь seq) добавляются в общий словарь с помощью метода .update().

Шаг 5: Нахождение максимального GC-содержания

Извлекаются все значения (GC-доли) из словаря.

С помощью max() находится наибольшее значение среди них.

Шаг 6: Поиск и вывод последовательности с максимальным GC-содержанием

Перебираются все пары идентификатор — GC-доля.

Если GC-доля равна найденному максимуму, то:

* + Выводится идентификатор последовательности (key).
  + Выводится значение GC-доли (value).

Это нужно на случай, если несколько последовательностей из разных файлов имеют одинаково высокий GC-процент.

**Встреченные проблемы**

1. Много последовательностей в одном файле
2. **Неправильное использование gc\_fraction:**  
   Функция gc\_fraction() возвращает значение от 0 до 1 (долю, а не процент), и это нужно учитывать при интерпретации результатов.
3. **glob не находит файлы:**  
   Если скрипт запущен из другой директории, glob.glob("\*.fasta") может не находить нужные файлы. Нужно быть уверенным, что рабочая директория совпадает с местоположением файлов.
4. Нет файлов с нужным расширением:
5. Код не содержит обработку ошибок, поэтому при:
   1. отсутствии файлов,
   2. повреждённом FASTA-файле,
   3. невозможности открыть файл  
      скрипт завершится с исключением.
6. Погрешности при сравнении float

**4. Выводы**

* Программа успешно решает задачу поиска последовательности с наибольшим содержанием гуанина и цитозина среди всех файлов формата FASTA в текущем каталоге.
* Использование библиотеки Biopython позволяет удобно обрабатывать биоинформатические форматы данных.
* Применение модуля glob делает решение универсальным: код автоматически обрабатывает все подходящие файлы без необходимости их явно указывать.
* Скрипт легко расширяется: можно добавить, например, вывод всех GC-долей, визуализацию или сортировку.