**Отчет по лабораторной работе №2**

**Вариант 5: «Нахождение мотива в ДНК»**

**Цель работы:**

Разработка программы для поиска всех вхождений подстроки (мотива) в заданной строке (геноме) с акцентом на эффективность при обработке больших объемов данных. Программа должна точно находить все позиции мотива в геноме, используя комбинацию различных подходов к поиску.

**Задачи:**

1. Реализовать функцию find\_positions(s, t, n), которая принимает три аргумента: геном s, мотив t и порог n.

2. Использовать линейный поиск для строк, длина которых меньше или равна порогу n.

3. Применять стратегию рекурсивного деления для строк, длина которых превышает порог n.

4. Рекурсивно разделить геном на две половины.

5. Найти все вхождения мотива в каждой половине.

6. Обработать перекрывающиеся области, где мотив может пересекать границу между двумя половинами.

7. Объединить результаты из всех частей для получения полного списка позиций.

8. Предоставить возможность пользователю вводить строки s, t и порог n через консоль.

9. Вывести все найденные позиции подстроки t в строке s.

Инструменты:

• Язык программирования: Python 3.

• IDE: VS Code

**Алгоритмы:**

1. **Линейный поиск (базовый случай):**

• Если длина строки s меньше или равна n, используется простой линейный поиск:

\* Перебираем все возможные начальные позиции i в строке s.

\* Для каждой позиции проверяем, совпадает ли подстрока s[i:i + len(t)] с мотивом t.

\* Если совпадает, добавляем индекс i в список positions.

\* Возвращаем список positions.

2. **Стратегия рекурсивного деления (рекурсивный случай):**

• Если длина строки s больше n:

\* Находим середину строки: mid = len(s) // 2.

\* Разделяем строку s на две половины: left\_s = s[:mid] и right\_s = s[mid:].

\* Рекурсивно вызываем find\_positions для каждой половины:

\* left\_positions = find\_positions(left\_s, t, n)

\* right\_positions = find\_positions(right\_s, t, n)

\* Корректируем индексы right\_positions: добавляем mid к каждой позиции, чтобы получить ее правильное положение в исходной строке s.

\* Обрабатываем перекрывающуюся область:

\* Создаем пустой список overlap\_positions.

\* Перебираем позиции i в диапазоне, где мотив может пересекать границу (от max(0, mid - len(t) + 1) до mid).

\* Если подстрока s[i:i + len(t)] совпадает с t, добавляем i в overlap\_positions.

\* Объединяем все три списка: left\_positions, right\_positions и overlap\_positions, чтобы получить окончательный список позиций.

\* Возвращаем объединенный список.

**Реализация кода (Python):**

#метод разделяй и властвую

def find\_positions(s, t, n=1000):

if len(s) <= n: # Если длина строки меньше или равна n, используем простой алгоритм

positions = []

for i in range(len(s) - len(t) + 1):

if s[i:i + len(t)] == t:

positions.append(i)

return positions

else: # Если длина строки больше n, применяем "разделяй и властвуй"

mid = len(s) // 2

left\_s = s[:mid]

right\_s = s[mid:]

left\_positions = find\_positions(left\_s, t, n)

right\_positions = [pos + mid for pos in find\_positions(right\_s, t, n)]

overlap\_positions = []

for i in range(max(0, mid - len(t) + 1), mid):

if s[i:i + len(t)] == t:

overlap\_positions.append(i)

# Объединяем результаты

return left\_positions + right\_positions + overlap\_positions

s = input("what s?")

t = input("what t?")

n = int(input("What n?"))

positions = find\_positions(s, t, n)

print(\*positions)

**Ошибки и их исправления:**

Изначальный код был изменен так, чтобы адаптировать алгоритм поиска подстрок, используя принцип «разделяй и властвуй».

**Выводы:**

• Разработанная программа эффективно находит все позиции заданного мотива в геноме.

• Комбинирование линейного поиска для малых строк со стратегией рекурсивного деления для больших строк позволяет адаптироваться к разным размерам входных данных.

• Параметр n позволяет настраивать порог переключения между двумя подходами, что может быть использовано для оптимизации производительности.

• Данный подход может быть полезен для работы с очень большими геномами, где простой линейный поиск был бы слишком медленным.

• В дальнейшем можно исследовать другие стратегии разделения (например, деление на большее число частей) и динамическую настройку параметра n в зависимости от характеристик генома и мотива.