上海建桥学院

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 软件测试技术 |
| 学 号： | 2311806 |
| 姓 名： | 宋卓立 |
| 专 业： | 软件工程 |
| 班 级： | 软工B21-4 |
| 指导教师： | 龙永辉 |

**课内实验目录及成绩**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 实验名称 | 页码 | 成绩 |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |

信息技术学院

2024年 4月 14日

**上海建桥学院实验报告**

课程名称： 软件测试技术 实验类型： python练习

实验项目名称：

实验地点： 实验日期： 2024 年 4 月 14 日

1. 实验目的
2. 实验过程及结果

编写两个函数，一个函数来生成斐波那契数列，另一个函数来生成素数序列通过调用这两个函数生成分子和分母，组合成分数数列，并计算其总和。最后，输出结果。

# 函数1: 生成斐波那契数列

def fibonacci(n):

    fib\_sequence = [0, 1]  # 前两项

    for i in range(2, n):

        fib\_sequence.append(fib\_sequence[-1] + fib\_sequence[-2])

    return fib\_sequence

# 函数2: 判断是否为素数

def is\_prime(num):

    if num <= 1:

        return False

    elif num <= 3:

        return True

    elif num % 2 == 0 or num % 3 == 0:

        return False

    i = 5

    while i \* i <= num:

        if num % i == 0 or num % (i + 2) == 0:

            return False

        i += 6

    return True

# 主函数: 生成分数数列并计算前20项之和

def sum\_fraction\_series(n):

    fib\_sequence = fibonacci(n)  # 生成斐波那契数列

    primes = []

    i = 2

    while len(primes) < n:

        if is\_prime(i):

            primes.append(i)

        i += 1

    fraction\_series = []

    for i in range(n):

        numerator = fib\_sequence[i]  # 分子为斐波那契数列中的值

        denominator = primes[i]  # 分母为素数序列中的值

        fraction\_series.append(numerator / denominator)

    sum\_series = sum(fraction\_series)  # 计算分数数列的和

    return sum\_series

# 计算前20项之和并输出结果

result = sum\_fraction\_series(20)

print("前20项分数数列之和为:", result)



通过迭代输入的DNA序列，计算每个可能长度的子序列的GC-Ratio，并记录下最高的GC-Ratio以及对应的子序列长度。

calculate\_gc\_ratio 函数计算了一个给定子序列的GC-Ratio，而 find\_highest\_gc\_ratio 函数找到了最高的GC-Ratio和对应的子序列长度。然后对示例DNA序列调用这两个函数，并输出结果。

def calculate\_gc\_ratio(sequence):

    gc\_count = sequence.count('G') + sequence.count('C')  # 统计GC出现次数

    total\_bases = len(sequence)  # 总碱基数

    gc\_ratio = gc\_count / total\_bases

    return gc\_ratio

def find\_highest\_gc\_ratio(sequence):

    max\_gc\_ratio = 0

    max\_gc\_ratio\_length = 0

    sequence\_length = len(sequence)

    # 迭代不同长度的子序列

    for length in range(1, sequence\_length + 1):

        for start\_index in range(sequence\_length - length + 1):

            sub\_sequence = sequence[start\_index:start\_index + length]

            gc\_ratio = calculate\_gc\_ratio(sub\_sequence)

            if gc\_ratio > max\_gc\_ratio:

                max\_gc\_ratio = gc\_ratio

                max\_gc\_ratio\_length = length

    return max\_gc\_ratio, max\_gc\_ratio\_length

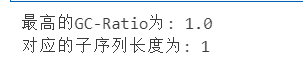
# 示例DNA序列

dna\_sequence = "ATGCGCTAAGCTAGCTAGCTGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCG"

max\_gc\_ratio, max\_gc\_ratio\_length = find\_highest\_gc\_ratio(dna\_sequence)

print("最高的GC-Ratio为:", max\_gc\_ratio)

print("对应的子序列长度为:", max\_gc\_ratio\_length)



计算两个字符串的最大公共子串长度可以使用动态规划算法。我们可以创建一个二维数组，其中 dp[i][j] 表示以字符串1的第 i 个字符和字符串2的第 j 个字符结尾的公共子串的长度。如果两个字符相同，那么 dp[i][j] 可以通过 dp[i-1][j-1] + 1 来计算；否则，dp[i][j] 应该为0。定义了一个 longest\_common\_substring\_length 函数来计算两个字符串的最大公共子串的长度。在函数中，我们先将两个字符串转换为小写，然后初始化一个二维数组 dp 用于动态规划计算。接着我们遍历两个字符串，如果当前字符相同，则更新 dp[i][j] 为前一个对角线的值加1，同时更新最大公共子串的长度。最后返回最大公共子串的长度。

def longest\_common\_substring\_length(s1, s2):

    # 将两个字符串转换为小写，以便字符不区分大小写

    s1 = s1.lower()

    s2 = s2.lower()

    # 初始化动态规划数组

    dp = [[0] \* (len(s2) + 1) for \_ in range(len(s1) + 1)]

    max\_length = 0  # 记录最大公共子串的长度

    # 动态规划计算公共子串长度

    for i in range(1, len(s1) + 1):

        for j in range(1, len(s2) + 1):

            if s1[i - 1] == s2[j - 1]:

                dp[i][j] = dp[i - 1][j - 1] + 1

                max\_length = max(max\_length, dp[i][j])

    return max\_length

# 示例字符串

string1 = "HelloWorld"

string2 = "WelcomedToTheWorld"

result = longest\_common\_substring\_length(string1, string2)

print("最大公共子串的长度为:", result)

