جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الانظمة الطبية الذكية

المرحلة الثانية

Bioinformatics

Subject: Lecture 7 P2
Lecturers: Dr. Maytham N. Meqdad

Here is a Python code to implement sequence alignment using dynamic programming, using the Needleman-Wunsch algorithm (global alignment) and the Smith-Waterman algorithm (local alignment).

Needleman-Wunsch algorithm

```
import numpy as np
def needleman_wunsch(seq1, seq2, match=1, mismatch=-1, gap=-2):
    m, n = len(seq1), len(seq2)
    dp = np.zeros((m+1, n+1))

    # Initialize scoring matrix
    for i in range(m+1):
        dp[i][0] = i * gap
    for j in range(n+1):
        dp[0][j] = j * gap

    # Fill scoring matrix
    for i in range(1, m+1):
        for j in range(1, n+1):
            match_score = dp[i-1][j-1] + (match if seq1[i-1] == seq2[j-1] else mismatch)
            delete = dp[i-1][j] + gap
            insert = dp[i][j-1] + gap
            dp[i][j] = max(match_score, delete, insert)

    # Traceback
    aligned_seq1, aligned_seq2 = "", ""
    i, j = m, n
    while i > 0 or j > 0:
        if i > 0 and j > 0 and dp[i][j] == dp[i-1][j-1] + (match if seq1[i-1] == seq2[j-1] else
mismatch):
            aligned_seq1 = seq1[i-1] + aligned_seq1
            aligned_seq2 = seq2[j-1] + aligned_seq2
            i -= 1
            j -= 1
        elif i > 0 and dp[i][j] == dp[i-1][j] + gap:
            aligned_seq1 = seq1[i-1] + aligned_seq1
            aligned_seq2 = "-" + aligned_seq2
            i -= 1
        else:
            aligned_seq1 = "-" + aligned_seq1
            aligned_seq2 = seq2[j-1] + aligned_seq2
            j -= 1

    return aligned_seq1, aligned_seq2, dp[m][n]

# Example usage
seq1 = "AGCT"
seq2 = "ACGT"
aligned_seq1, aligned_seq2, score = needleman_wunsch(seq1, seq2)
print(f"Aligned Sequences:\n{aligned_seq1}\n{aligned_seq2}\nScore: {score}")
```

Smith-Waterman Algorithm (Local Alignment)

```
import numpy as np
def smith_waterman(seq1, seq2, match=1, mismatch=-1, gap=-2):
    m, n = len(seq1), len(seq2)
    dp = np.zeros((m+1, n+1))
    max_score = 0
    max_pos = (0, 0)

    # Fill scoring matrix
    for i in range(1, m+1):
        for j in range(1, n+1):
            match_score = dp[i-1][j-1] + (match if seq1[i-1] == seq2[j-1] else mismatch)
            delete = dp[i-1][j] + gap
            insert = dp[i][j-1] + gap
            dp[i][j] = max(0, match_score, delete, insert)

            if dp[i][j] > max_score:
                max_score = dp[i][j]
                max_pos = (i, j)

    # Traceback
    aligned_seq1, aligned_seq2 = "", ""
    i, j = max_pos
    while i > 0 and j > 0 and dp[i][j] > 0:
        if dp[i][j] == dp[i-1][j-1] + (match if seq1[i-1] == seq2[j-1] else mismatch):
            aligned_seq1 = seq1[i-1] + aligned_seq1
            aligned_seq2 = seq2[j-1] + aligned_seq2
            i -= 1
            j -= 1
        elif dp[i][j] == dp[i-1][j] + gap:
            aligned_seq1 = seq1[i-1] + aligned_seq1
            aligned_seq2 = "-" + aligned_seq2
            i -= 1
        else:
            aligned_seq1 = "-" + aligned_seq1
            aligned_seq2 = seq2[j-1] + aligned_seq2
            j -= 1

    return aligned_seq1, aligned_seq2, max_score

# Example usage
seq1 = "AGCT"
seq2 = "ACGT"
aligned_seq1, aligned_seq2, score = smith_waterman(seq1, seq2)
print(f"Aligned Sequences:\n{aligned_seq1}\n{aligned_seq2}\nScore: {score}")
```