Lab Worksheet 13 Solutions

For this assignment, your goal is to implement the Needleman-Wunsch algorithm in Python. You can read more about the Needleman-Wunsch algorithm on Wikipedia (https://en.wikipedia.org/wiki/Needleman–Wunsch_algorithm). The Wikipedia page contains psuedo-code which you might find helpful.

Part 1

Write a function that takes two sequences as input, and returns a matrix of scores as we saw in Class 25. You **do not** have to do the back-tracing, just fill out the matrix.

To get you started, a matrix can be represented in Python as a list of lists. Let's say we want to make a matrix that looks like this:

1	3	5	7
2	3	4	5
5	2	20	3

In [1]:
```
# Here's how to make the matrix above from a list of lists
my_matrix = []
# Fill out the 0th row
my_matrix.append([1, 3, 5, 7])
# Fill out the 1st row
my_matrix.append([2, 3, 4, 5])
# Fill out the 2nd row
my_matrix.append([5, 2, 20, 3])

# Here is a helper function to print out matrices
def print_matrix(mat):
    # Loop over all rows
    for i in range(0, len(mat)):
        print("[", end = "")
        # Loop over each column in row i
        for j in range(0, len(mat[i])):
            # Print out the value in row i, column j
            print(mat[i][j], end = "")
            # Only add a tab if we're not in the last column
            if j != len(mat[i]) - 1:
                print("\t", end = "")
        print("]\n")

print_matrix(my_matrix)

# To retrieve the value from the 2nd row, in the 0th column, is relatively simp
le:
print("The value in the 2nd row and the 0th column is:", my_matrix[2][0])

# The format is always my_matrix[row][column].
```

```
[1	3	5	7]

[2	3	4	5]

[5	2	20	3]

The value in the 2nd row and the 0th column is: 5
```

Part 1 Hints

Break the problem down into as many small steps as possible. Here are a few hints:

- Before you calculate any scores, make an empty matrix of the appropriate size using the `zeros()` function defined below.
- Fill out the 0th row and 0th column before you calculate any other scores.
- The `max()` function will return the maximum value from a list of values. For example `max(1,7,3)` will return 7.
- Make liberal use of the `range()` function.
- Use the `print_matrix()` function to print out your matrix as frequently as possible. Always make sure that your code is doing what you think it's doing!
- Remember, in Python, we start counting from 0.

In [2]:
```
# Use these values to calculate scores
gap_penalty = -1
match_award = 1
mismatch_penalty = -1

# Make a score matrix with these two sequences
seq1 = "ATTACA"
seq2 = "ATGCT"

# A function for making a matrix of zeroes
def zeros(rows, cols):
    # Define an empty list
    retval = []
    # Set up the rows of the matrix
    for x in range(rows):
        # For each row, add an empty list
        retval.append([])
        # Set up the columns in each row
        for y in range(cols):
            # Add a zero to each column in each row
            retval[-1].append(0)
    # Return the matrix of zeros
    return retval

# A function for determining the score between any two bases in alignment
def match_score(alpha, beta):
    if alpha == beta:
        return match_award
    elif alpha == '-' or beta == '-':
        return gap_penalty
    else:
        return mismatch_penalty

# The function that actually fills out a matrix of scores
def needleman_wunsch(seq1, seq2):

    # length of two sequences
    n = len(seq1)
    m = len(seq2)

    # Generate matrix of zeros to store scores
    score = zeros(m+1, n+1)

    # Calculate score table

    # Your code goes here

    # Fill out first column
    for i in range(0, m + 1):
        score[i][0] = gap_penalty * i

    # Fill out first row
    for j in range(0, n + 1):
        score[0][j] = gap_penalty * j

    # Fill out all other values in the score matrix
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            # Calculate the score by checking the top, left, and diagonal cells
            match = score[i - 1][j - 1] + match_score(seq1[j-1], seq2[i-1])
            delete = score[i - 1][j] + gap_penalty
            insert = score[i][j - 1] + gap_penalty
            # Record the maximum score from the three possible scores calculate
```

```
[0      -1      -2      -3      -4      -5      -6]
[-1     1       0       -1      -2      -3      -4]
[-2     0       2       1       0       -1      -2]
[-3     -1      1       1       0       -1      -2]
[-4     -2      0       0       0       1       0]
[-5     -3      -1      1       0       0       0]
```

Part 2: If that was easy...

Modify your code from Part 1 to back-trace through the score matrix and print out the final alignment. **HINT:** For the back-tracing, you'll want to use a `while` loop (or several of them).

In [3]:
```
def needleman_wunsch(seq1, seq2):

    # Store length of two sequences
    n = len(seq1)
    m = len(seq2)

    # Generate matrix of zeros to store scores
    score = zeros(m+1, n+1)

    # Calculate score table

    # Fill out first column
    for i in range(0, m + 1):
        score[i][0] = gap_penalty * i

    # Fill out first row
    for j in range(0, n + 1):
        score[0][j] = gap_penalty * j

    # Fill out all other values in the score matrix
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            # Calculate the score by checking the top, left, and diagonal cells
            match = score[i - 1][j - 1] + match_score(seq1[j-1], seq2[i-1])
            delete = score[i - 1][j] + gap_penalty
            insert = score[i][j - 1] + gap_penalty
            # Record the maximum score from the three possible scores calculate
d above
            score[i][j] = max(match, delete, insert)

    # Traceback and compute the alignment

    # Create variables to store alignment
    align1 = ""
    align2 = ""

    # Start from the bottom right cell in matrix
    i = m
    j = n

    # We'll use i and j to keep track of where we are in the matrix, just like
above
    while i > 0 and j > 0: # end touching the top or the left edge
        score_current = score[i][j]
        score_diagonal = score[i-1][j-1]
        score_up = score[i][j-1]
        score_left = score[i-1][j]

        # Check to figure out which cell the current score was calculated from,
        # then update i and j to correspond to that cell.
        if score_current == score_diagonal + match_score(seq1[j-1], seq2[i-1]):
            align1 += seq1[j-1]
            align2 += seq2[i-1]
            i -= 1
            j -= 1
        elif score_current == score_up + gap_penalty:
            align1 += seq1[j-1]
            align2 += '-'
            j -= 1
        elif score_current == score_left + gap_penalty:
            align1 += '-'
            align2 += seq2[i-1]
            i -= 1
```

```
ATTACA
A-TGCT
```

In [ ]: