Lab 04 Alignment

For Lab 04 | Alignment you will write a program that performs Needleman-Wunsch alignment on two sequences.

Assignment

Follow the instructions in this document and answer the questions in the cell below each question. Submit your answers by uploading a PDF file to gradescope. To generate the pdf, first export the notebook as HTML: >File, >Export to ..., >HTML. Then, open the HTML in a browser and use your browser to print to PDF.

Check to make sure all your cells have been run and the **results** displayed in the PDF file. Gradescope only accepts PDF.

Reminder, provide comments for any code you write to ensure partial credit.

Introduction

The Needleman–Wunsch algorithm was developed to align protein or nucleotide sequences using dynamic programming. By dividing a large problem (i.e. all possible alignments) into a series of smaller problems (i.e. whether it is best to extend an alignment with a gap or mismatch), it uses the solutions to the smaller problems to find an optimal solution to the larger problem. The algorithm assigns a score to every possible extension to an alignment, and the purpose of the algorithm is to find the overall best scoring alignment.

Scoring an alignment

In an alignment, positions are scored according to whether there is a **match**, **mismatch** or **gap** in one of the sequences.

Thus, given an alignment and scoring match = 1, mismatch = -1, and gap = -1, the alignment would have a score of:

```
GCATG-CA
G-ATTACA
+-++--++
```

where the plus and minus below the alignment indicates +1 or -1 scores at those positions.

The total score is the sum of all positions = +5 -3 = 2

Question 1

Write a function that takes a match, mismatch and gap score, as well as two aligned sequences as input, and returns their alignment score.

Apply your function to and print the score for the two sequences (seqa, seqb) as well as for the two sequences in AQY2.fasta.

(5 points)

In [16]:
```
match_score = 2
mismatch_score = -3
gap_score = -5

seqa="CGTATGCTAGC"
seqb="CTTAA-CTAGC"

# Answer
def score_alignment(s1, s2, match, mismatch, gap):
    score = 0

    return (score)

print( score_alignment(seqa, seqb, match_score, mismatch_score, gap_s
core))

# Read in fasta file into seq1 and seq2
filename = 'AQY2.fasta'
```

```
0
```

Dot plots

Before constructing an alignment, lets first implement a dotplot matrix of two sequencies. This will help illustrate how to store the scoring matrix needed for Needleman-Wunsch.

Lets define a function that takes two sequences as input, and outputs a matrix with `1` indicating all the matching positions and `0` indicate non-matching positions.

In [17]:
```
def dotplot(seqa, seqb):
    # First initialize a matrix according to lengths of seqa and seqb
    n = len(seqa)
    m = len(seqb)
    # Make an array (list) of zeros according to length of seqa
    mat = [0] * n
    # For each element in the list, fill in zeros of length seqb
    for i in range(n):
        mat[i] = [0] * m
    # Print space with sequence b on the top
    print("  " + seqb)
    # For each row and each column, enter 1 for match and 0 for misma
tch
    for row in range(n):
        # Print sequence a with space and without new line
        print(seqa[row]+ " ",end="")
        for col in range(m):
            if (seqa[row] == seqb[col]):
                mat[row][col] = 1
            else:
                mat[row][col] = 0
            # Print the elements without a new line
            print(mat[row][col],end="")
        # To get a new line print empty string
        print("")
    return(mat)

m = dotplot("CAGTTTC","CAGTTTT")

# Function to plot the matrix with equal spaces for visualization
def plotmatrix(m):
    for i in range(len(m)):
        for j in range(len(m[0])):
            print('%3s' % m[i][j],end=" ")
        print("")
```

```
  CAGTTTT
C 1000000
A 0100000
G 0010000
T 0001111
T 0001111
T 0001111
C 1000000
```

Scoring matrix

To fill in a scoring matrix using Needleman-Wunsch we can follow the following pseudocode from wikipedia:

Here, d is the gap penalty and S(Ai, Bj) should be a match score if Ai and Bj match, and a mismatch score if Ai and Bj do not match.

Scoring the matrix

Scoring the full matrix is a matter of filling in the matrix using `S(i,j)` as described in lecture.

However, encoding this can be a tricky unless you keep track of i vs j, n vs m, seqa vs seqb. It can help to have a diagram such as this:

Question 2

Write a function that takes a match, mismatch and gap score, as well as two ungapped sequences as input and returns a Needleman-Wunsch scoring matrix with **all** the positions filled in. Use the function to obtain a scoring matrix for the two sequences and print the matrix. Hint: it may be easier to first initialize the matrix with first row and column of gap scores, then fill out the rest of the matrix.

The `max()` function will return the maximum value from a list of values. For example `max(1,7,3)` will return 7.
For help understanding the `max()` function see Python documentation:
https://docs.python.org/3/library/functions.html#max (https://docs.python.org/3/library/functions.html#max)

(5 points)

In [18]:

```python
# Make a score matrix with these two sequences
seqa = "CGTATGCTAGCTTATTTGC"
seqb = "TAACTAGCGATTGCGC"
# And these match, mismatch and gap penalties
match = 1
mismatch = -1
gap = -2

# Answer


def needleman_wunsch_full(seqa, seqb, match, mismatch, gap):
    # length of two sequences
    n = len(seqa)
    m = len(seqb)
    # Initialize scoring matrix with zeros and gap scores for first r
ow and column
    score = [0] * (n+1)
    for i in range(n+1):
        score[i] = [0] * (m+1)

    return score

s = needleman_wunsch_full(seqa, seqb, match, mismatch, gap)

plotmatrix(s)
```

```
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

Question 3

Write a function that takes a match, mismatch and gap score, as well as two ungapped sequences as input and returns two strings of the aligned sequences. Apply your function to the two sequences provided and print the aligned sequences, one sequence per line. Finally, calculate the score of your alignment using the match, mismatch and gap penalty provided.

Not necessary to get the answer, but a suggestion:

- after filling out the scoring matrix, start at the lower right (position score[n,m]) with i and j keeping track of your position
- use a while loop: `while i > 0 and j > 0` and update i and j depending on the best move
- append gaps or nucleotides to two strings that store your alignment
- if you need to reverse a string use: reverse_string = string[::-1]

Note that thare are multiple **best** alignments, but you only need to find one. You can check your results here: http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Needleman-Wunsch (http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Needleman-Wunsch)

(5 points)

In [19]:
```
# Answer

# Make a score matrix with these two sequences
seqa = "CGTATGCTAGCTTATTTGC"
seqb = "TAACTAGCGATTGCGC"
# And these match, mismatch and gap scores
match = 1
mismatch = -1
gap = -2
```

Question 4

Find the score of all pairwise alignments of sequences in the provided file and the match, mismatch and gap scores. The sequences are already aligned to one another. Print the names of the aligned sequences and their alignment score, sorted with the highest scoring alignment first.

(5 points)

In [20]:
```
filename = 'SSA.fasta'
match = 1
mismatch = -1
gap = -1

# Answer
```

Bonus Question

Write a function that takes a match, mismatch, gap open and gep extension score, as well as two aligned sequences as input, and returns their alignment score.

Apply your function to and print the score for the two sequences in AQY2.fasta.

(2 points)

In [21]:
```
match = 2
mismatch = -3
gapopen = -5
gapextension = -1


# Answer
```

In [ ]:
```

```