Introduction to Quantitative Biology(BIO213-IQB)

Assignment 1

Efforts By: Pritish Wadhwa (2019440)

In [1]:
```
#importing the required libraries
import numpy as np
import math
```

In [2]:
```
#the given DNA Sequences
dnaSeq1 = "ATCAGAGTA"
dnaSeq2 = "TTCAGTA"
```

In [3]:
```
#calculating the number of rows and columns for the DP table
rows = len(dnaSeq1) + 1
cols = len(dnaSeq2) + 1
```

In [4]:
```
#initilaizing the original match, gap and mismatch scores
matchPenalty = 2
mismatchPenalty = -1
gapPenalty = -1
```

In [5]:
```
#initializing the DP matrix with 0's
dpMatrix = np.zeros((rows, cols), dtype = np.int)
```

In [6]:
```
#creating a visited matrix for backtracking
visited = np.zeros((rows, cols), dtype=np.int)
```

In [7]:
```
#function to check whether a cell is valid as the last cell or not(while generat
def isValid(i, j):
    if i == 0:
        if dpMatrix[i][j-1] + gapPenalty != dpMatrix[i][j]:
            return True
        else:
            return False
    elif j == 0:
        if dpMatrix[i-1][j] + gapPenalty != dpMatrix[i][j]:
            return True
        else:
            return False
    else:
        if (dpMatrix[i-1][j] + gapPenalty != dpMatrix[i][j]) and (dpMatrix[i][j-
            return True
```

```
        else:
            return False
```

In [8]:

```
#function to calculate scores of the 2 sequences depending upon the scoring sche
def calculateScore(seq1, seq2):
    score = 0
    for i in range(len(seq1)):
        if seq1[i] == ' ':
            continue
        elif seq1[i] == seq2[i]:
            score += matchPenalty
        elif seq1[i] != seq2[i]:
            if seq1[i] == '-' or seq2[i] == '-':
                score += gapPenalty
            else:
                score += mismatchPenalty
    return score
```

In [9]:

```
#the function to print the sequences along with their matching sequences
def printSequences(seq1, seq2):
    seq1 = seq1.replace("", " ")
    seq2 = seq2.replace("", " ")
    seq1 = seq1[1: -1]
    seq2 = seq2[1: -1]
    score = calculateScore(seq1, seq2)
    print(seq1)
    for i in range(len(seq1)):
        if seq1[i] == ' ' or seq1[i] != seq2[i] or seq1[i] == seq2[i] == '-':
            print(' ', end = '')
        elif seq1[i] == seq2[i]:
            print('|', end = '')
    print()
    print(seq2)
    print("The alignment score for the above sequences with given penalties is:
    print()
```

In [10]:

```
#function to print the dp matrix
def printMatrix():
    column0 = '0' + dnaSeq2
    row0 = '0' + dnaSeq1
    for i in range(rows + 1):
        for j in range(cols + 1):
            if i == 0 and j == 0:
                print('X', end = '\t')
            elif i == 0:
                print(column0[j-1], end = '\t')
            elif j == 0:
                print(row0[i-1], end = '\t')
            else:
                print(dpMatrix[i-1][j-1], end = "\t")
        print()
```

In [11]:

```
#function to generate the global alignments of the given 2 sequences
def globalAlignment(rows, cols, seq1, seq2):
    for i in range(1, rows):
```

```
        for j in range(1, cols):
            if seq1[i-1] == seq2[j-1]:
                dpMatrix[i][j] = dpMatrix[i-1][j-1] + matchPenalty
            else:
                dpMatrix[i][j] = max(dpMatrix[i-1][j-1] + mismatchPenalty, dpMat
```

In [12]:

```
#function to generate the local alignments of the given 2 sequences
#It returns a list of positions where the dp matrix has the max value
def localAlignment(rows, cols, seq1, seq2):
    maxVal = -math.inf
    l = []
    for i in range(0, rows):
        for j in range(0, cols):
            if i == 0 or j == 0:
                dpMatrix[i][j] = 0
            elif seq1[i-1] == seq2[j-1]:
                dpMatrix[i][j] = dpMatrix[i-1][j-1] + matchPenalty
            else:
                dpMatrix[i][j] = max(0, dpMatrix[i-1][j-1] + mismatchPenalty, dp
            if dpMatrix[i][j] >= maxVal:
                maxVal = dpMatrix[i][j]
    for i in range(0, rows):
        for j in range(0, cols):
            if dpMatrix[i][j] == maxVal:
                l.append((i, j))
    return l
```

In [13]:

```
#function to generate sequences in both global and local alignment scenarios
def generateSequences(i, j, seq1, seq2, ans1, ans2):
    if (i == 0 and j == 0) or (dpMatrix[i][j] == 0 and isValid(i, j)):
        ans1 = ans1[::-1]
        ans2 = ans2[::-1]
        printSequences(ans1, ans2)
        return
    elif dpMatrix[i][j] == 0:
        ans1 = ans1[::-1]
        ans2 = ans2[::-1]
        printSequences(ans1, ans2)
        ans1 = ans1[::-1]
        ans2 = ans2[::-1]
    elif i < 0 or j < 0 or visited[i][j] == 1:
        return
    visited[i][j] = 1
    flag = 0
    if seq1[i-1] == seq2[j-1]:
        flag = 1
        generateSequences(i-1, j-1, seq1, seq2, ans1 + seq1[i-1], ans2 + seq2[j-
    if dpMatrix[i][j] == dpMatrix[i-1][j-1] + mismatchPenalty:
        flag = 1
        generateSequences(i-1, j-1, seq1, seq2, ans1 + seq1[i-1], ans2 + seq2[j-
    if dpMatrix[i][j] == dpMatrix[i-1][j] + gapPenalty:
        flag = 1
        generateSequences(i-1, j, seq1, seq2, ans1 + seq1[i-1], ans2 + '-')
    if dpMatrix[i][j] == dpMatrix[i][j-1] + gapPenalty:
        flag = 1
        generateSequences(i, j-1, seq1, seq2, ans1 + '-', ans2 + seq2[j-1])
    if flag == 0:
```

```
        generateSequences(i-1, j-1, seq1, seq2, ans1 + '-', ans2 + '-')
    visited[i][j] = 0
```

Question 1)

In [14]:
```
#initializing the 0th row and 0th column in DP matrix for global Alignment
for i in range(rows):
    dpMatrix[i][0] = -1 * i
for j in range(cols):
    dpMatrix[0][j] = -1 * j
```

In [15]:
```
globalAlignment(rows, cols, dnaSeq1, dnaSeq2)
```

1-a)

The generated dynamic programming matrix:

In [16]:
```
printMatrix()
```

```
X       0       T       T       C       A       G       T       A
0       0       -1      -2      -3      -4      -5      -6      -7
A       -1      -1      -2      -3      -1      -2      -3      -4
T       -2      1       1       0       -1      -2      0       -1
C       -3      0       0       3       2       1       0       -1
A       -4      -1      -1      2       5       4       3       2
G       -5      -2      -2      1       4       7       6       5
A       -6      -3      -3      0       3       6       6       8
G       -7      -4      -4      -1      2       5       5       7
T       -8      -5      -2      -2      1       4       7       6
A       -9      -6      -3      -3      0       3       6       9
```

1-b)

Yes, There are more than 1 optimal alingments for the given pair of DNA Sequences. This occurs because, the highest possible alignment score for the given pair of sequences is possible in more than 1 way

1-c)

The global allignments with their scores are:

In [17]:
```
generateSequences(rows - 1, cols - 1, dnaSeq1, dnaSeq2, "", "")
```

```
A T C A G A G T A
  | |     | | | |
T T C - - A G T A
The alignment score for the above sequences with given penalties is: 9

A T C A G A G T A
  | | |     | | |
T T C A - - G T A
The alignment score for the above sequences with given penalties is: 9
```

```
A T C A G A G T A
  | | | |     | |
T T C A G - - T A
The alignment score for the above sequences with given penalties is: 9
```

Question 2)

In [18]:
```
l = localAlignment(rows, cols, dnaSeq1, dnaSeq2)
```

2-a)

The generated dynamic programming matrix:

In [19]:
```
printMatrix()
```

X	0	T	T	C	A	G	T	A
0	0	0	0	0	0	0	0	0
A	0	0	0	0	2	1	0	2
T	0	2	2	1	1	1	3	2
C	0	1	1	4	3	2	2	2
A	0	0	0	3	6	5	4	4
G	0	0	0	2	5	8	7	6
A	0	0	0	1	4	7	7	9
G	0	0	0	0	3	6	6	8
T	0	2	2	1	2	5	8	7
A	0	1	1	1	3	4	7	10

2-b)

The local allignments with their scores are:

In [20]:
```
for i in l:
    generateSequences(i[0], i[1], dnaSeq1, dnaSeq2, "", "")
```

```
T C A G A G T A
| |     | | | |
T C - - A G T A
The alignment score for the above sequences with given penalties is: 10

T C A G A G T A
| | |     | | |
T C A - - G T A
The alignment score for the above sequences with given penalties is: 10

T C A G A G T A
| | | |     | |
T C A G - - T A
The alignment score for the above sequences with given penalties is: 10
```

Question 3)

The changes required in the program to perform local alignment rather then global alignment are:

- For the global alignment, the entries of the 0^{th} row and column were initialised with 0, -1, -2 and so on, whereas in the local aligment, the entries of the 0^{th} row and column were initialised by 0.
- For the other entries, in the dp matrix of the global alignmnet, -ve values are possible but that is not the case for the dp matrix of local alignment. There the entries can only be a whole number.
- The above points are valid because while in the global alignment, while filling the table, only the maximum value of the adjoining cells with the added penalties are considered, whereas in the local alignment, while filling the table, the maximum of the adoining cells with added penalties, and 0 is considered.
- While reconstructing the sequence, in the global alignment, the maximum value is always at the last cell of the matrix, where as the maximum value in the local alignment can be anywhere in the matrix, not necessarily at the very end.
- In global alignment, while only one cell can accomodate the maximum value, whereas in local alignment, the maximum value can present in multiple cells.
- In the global alignment, the base case is reached when you are at the $(0,0)^{th}$ cell, whereas in local alignment, the base case is reached when, you can't move anywhere from the current cell, and the current cell has the value 0 in it.
- Another case that can be considered for local alignment is that if a value of 0 is reached while generating the sequence, at that instant, the generated sequence is also optimally aligned, but the function can not return from there as it is not the ultimate base case.

Question 4)

In [21]:
```
#updated scores for the last question
matchPenalty = 2
mismatchPenalty = -1
gapPenalty = -2
```

Global Alignment

In [22]:
```
#initializing dp matrix for global alignment
for i in range(rows):
    dpMatrix[i][0] = -1 * i
for j in range(cols):
    dpMatrix[0][j] = -1 * j
globalAlignment(rows, cols, dnaSeq1, dnaSeq2)
```

Generated Dynamic Programming Matrix:

In [23]:
```
printMatrix()
```

```
X       0       T       T       C       A       G       T       A
0       0       -1      -2      -3      -4      -5      -6      -7
A       -1      -1      -2      -3      -1      -3      -5      -4
T       -2      1       1       -1      -3      -2      -1      -3
C       -3      -1      0       3       1       -1      -3      -2
A       -4      -3      -2      1       5       3       1       -1
G       -5      -5      -4      -1      3       7       5       3
A       -6      -6      -6      -3      1       5       6       7
G       -7      -7      -7      -5      -1      3       4       5
T       -8      -5      -5      -7      -3      1       5       3
A       -9      -7      -6      -6      -5      -1      3       7
```

Optimally Aligned Sequences:

In [24]:
```
generateSequences(rows - 1, cols - 1, dnaSeq1, dnaSeq2, "", "")
```

```
A T C A G A G T A
  | |     | | | |
T T C - - A G T A
The alignment score for the above sequences with given penalties is: 7

A T C A G A G T A
  | | |     | | |
T T C A - - G T A
The alignment score for the above sequences with given penalties is: 7

A T C A G A G T A
  | | | |     | |
T T C A G - - T A
The alignment score for the above sequences with given penalties is: 7
```

- After updatind the scoring scheme, even though the generated sequence didn't change, the corresponding score did change.
- This was because, as in global alignments, there has to the complete involvement of both the subsequences.
- Since the difference in the 2 strings is of 2 characters, there has to be atleast 2 gaps to be inserted.
- Now since the gap penalty is made more negative, thus, the DP Matrix changes.
- Since the difference in the new values was not that much significant for the given sequences when being considered for global alignment, we get the same old alignments but a different score for each

Local Alignment

In [25]:
```
l = localAlignment(rows, cols, dnaSeq1, dnaSeq2)
```

Generated Dynamic Programming Matrix:

In [26]:
```
printMatrix()
```

```
X       0       T       T       C       A       G       T       A
0       0       0       0       0       0       0       0       0
A       0       0       0       0       2       0       0       2
T       0       2       2       0       0       1       2       0
```

```
C       0       0       1       4       2       0       0       1
A       0       0       0       2       6       4       2       2
G       0       0       0       0       4       8       6       4
A       0       0       0       0       2       6       7       8
G       0       0       0       0       0       4       5       6
T       0       2       2       0       0       2       6       4
A       0       0       1       1       2       0       4       8
```

Optimally Aligned Sequences:

In [27]:
```
for i in l:
    generateSequences(i[0], i[1], dnaSeq1, dnaSeq2, "", "")
```

```
T C A G
| | | |
T C A G
The alignment score for the above sequences with given penalties is: 8

T C A G - A
| | | |   |
T C A G T A
The alignment score for the above sequences with given penalties is: 8

A G T A
| | | |
A G T A
The alignment score for the above sequences with given penalties is: 8

T C A G A G T A
| |     | | | |
T C - - A G T A
The alignment score for the above sequences with given penalties is: 8

T C A G A G T A
| | |     | | |
T C A - - G T A
The alignment score for the above sequences with given penalties is: 8

T C A G A G T A
| | | |     | |
T C A G - - T A
The alignment score for the above sequences with given penalties is: 8
```

- After updatind the scoring scheme, the generated sequences changed and the corresponding score did change.
- This was because, as in local alignment, unlike global one, there is no need for complete involvement of both the subsequences.
- Now since the gap penalty is made more negative, thus, the DP Matrix changes.
- Since the difference in the new values was significant for the given sequences when being considered for local alignment, we get three more sequences, along with the ones we got with the previos scoring scheme.