BCH 571 Bioinformatics for Life Scientists

Lab 9

For this lab, you will implement the Needleman-Wunsch algorithm for sequence alignment. The main recursion is:

$$score_{i,j} = \max \begin{cases} score_{i-1,j-1} + \begin{cases} M \text{ if } v_i = w_j \\ N \text{ if } v_i \neq w_j \end{cases} \\ score_{i-1,j} + I \\ score_{i,j-1} + I \end{cases}$$

Here is the pseudocode (where strings are 1-indexed):

Provide strings v and W and output an alignment as two strings
Also provide I, the insert cost, M the match gain, and N, the non-match cost.
NeedlemanWunsch(v,w,I,M,N)
  Build a 2D array, called score, of size (length(v) + 1)×(length(w) + 1)
  for i=0 to length(v)
    $score_{i,0} \leftarrow I \times i$
  for j=0 to length(w)
    $score_{0,j} \leftarrow I \times j$
  for i=1 to length(v)
    for j=1 to length(w)
      if (v_i=w_j)
        match = M
      else
        match = N
      $score_{i,j} \leftarrow \max(match + score_{i-1,j-1}, score_{i-1,j} + I, score_{i,j-1} + I)$

  Alignment_v ← ""
  Alignment_w ← ""
  i ← length(v)
  j ← length(w)
  while (i > 0 or j > 0)
    if (i > 0 and j > 0)
      if (v_i=w_j)
        match = M
      else
        match = N
      if $score_{i,j} = score_{i-1,j-1} + match$
        Alignment_v ← v_i + Alignment_v
        Alignment_w ← w_j + Alignment_w
        i ← i - 1
        j ← j – 1

else if $score_{i,j} = score_{i-1,j} + l$
Alignment_v ← v_i + Alignment_v
Alignment_w ← "-" + Alignment_w
i ← i – 1
else
Alignment_v ← "-"+ Alignment_v
Alignment_w ← w_j + Alignment_w
j ← j – 1
else if (i > 0)
Alignment_v ← v_i + Alignment_v
Alignment_w ← "-" + Alignment_w
i ← i – 1
else
Alignment_v ← "-"+ Alignment_v
Alignment_w ← w_j + Alignment_w
j← j – 1

return Alignment_v,Alignment_w

One tricky aspect is setting up the 2D score array. In python, this is a list of lists. Here is python code for making the list of lists:

```
#create a list of lists that is indexed for v and then w
#this 0-indexed list of lists has 0 for the nucleotide before the
start
infinity = 10000
score=[]
for nucleotide in v:
    row = [infinity]*(len(w)+1)
    score.append(row)

row = [infinity]*(len(w)+1)
score.append(row)
```

Run the algorithm with the sequences:

v=

ACUUAGCUAAAACGUUUGGUUCAAAACAUUUGCUUGCUGUCUUGGCAUAACAUCAAUAAAGGCAUAAACAU
CGCAAAACAAUGGUUAUAUAUAAAUGGCUAUGAGGAUGGUUUUAGUACGUAGGCGUUGCGGAACUUCGGU
UCAGAUAGAGCAAUGAAUCGUGCAUGCUAGGAAAACUGACCACACGCAGUUGGCAGCCCUAGUAUCUUUCG
AUAGAUUUCCAUACCUCCGCGAUC

w=

ACUUAGCCUAUACACUAUGUUGGAGAGAGACGCUUGCUACCUAGGCAUAAUGUGAAUUAGGUAUAAACAUC

GUGGUUGUAAACUUGAGUGGGUUUUAGUACGGUAUGCGUGAUUACUUCGUAAUCAUGAAUCGUGCAUGC
UAGUGGGGUUUGGCCUCCACUAGUAUCUUUGAAGAUUUUCCUUCCUCAGCGAUC

And with M = 1, N = -0.5, I = -1