CS 6013: Advance Data Structures and Algorithms

Assignment 3

Name: Gagandeep Goyal, CS19MTECH01003

This document is generated by LaTeX

Contents

1 Global Sequence Allignment Q-1

1.1 Algorithmic Approch

1.1.1 Needleman-Wunsch algorithm

a) The Needleman-Wunsch algorithm is an example of dynamic programming used to build best alignment using optimal alignments of smaller subsequences.

b) This algorithm takes into consideration all pairs of residue from both the sequences i.e Target sequence and query sequence.

c) Above consideration leads us with a 2-D Matrix representation.

d) According to the algorithm we require two matrices namely SCORING matrix and TRACE-BACK matrix.

e) The Algorithm Has main three steps of course :

- Initializtion of Scoring Matrix
- Calculation of scores and filling traceback matrix
- Deducing the alignment from the traceback matrix

Recurrence Relation To fill Scoring Matrix

$$D(i,j) = max \left\{ \begin{array}{l} D(i-1,j-1) + s(x_i, y_i) \\ D(i-1,j), +g \\ D(i,j-1) + g \end{array} \right\} \tag{1.1}$$

where $s(x_i, y_i)$ is match/mis-match cost and g is gap penalty

f) Correspondingly the trace Matrix is maintained to remember the position i.e (diagonal, beneath or behind) from which every particular index element has derived.

g) Final Step comes is Deducing the alignment from the traceback Matrix.

h) It always begins with the bottom right cell of the scoring matrix.

i) Their are 3 various moves -

- Diagonal - Letters from both sequences are aligned.
- Behind - A Gap is introduced in Target sequence.
- Up - A Gap is introduced in Query sequence.

1.2 Implemented Code Description

a) Get the target and query string.

b) Declared vectors for Scoring and traceback matrix.

c) Fills the Scoring and traceback matrix according to the algorithm described above.

d) The next while loop traverse the traceback matrix.

e) Prints the results accordingly.

1.3 Time Complexity -

$O(mn)$ where **m** is Target seq and **n** is Query seq

1.4 Space Complexity -

$O(mn)$ where **m** is Target seq and **n** is Query seq

2 Global Sequence Allignment Q-2

2.1 Algorithmic Approch

2.1.1 Needleman-Wunsch algorithm

The Algorithm is same as defined above in Question 1.

2.2 Implemented Code Description

a) Implementation is the same as Question 1.

b) The basic difference is that a Matrix for match/mismatch penalty is provided in the question.

c) So instead of adding constant values for match/mismatch penalty in the scoring matrix, now the code makes call to match function which return the particular penalty corresponding to it.

2.3 Time Complexity -

$O(mn)$ where **m** is Target seq and **n** is Query seq

2.4 Space Complexity -

$O(mn)$ where **m** is Target seq and **n** is Query seq

3 Global Sequence Allignment Q-3

3.1 Algorithmic Approch

3.1.1 Affince-Gap Penalty

a)The Algorithm states that instead of giving fixed gap penalty, penalize the first gap differently, possibly more than the rest, then each subsequent gap will be penalized linerarly

b) The point a) clearifies that now we need to keep a check over either opening gap penalty or subsequent gap penalty.

c) For gap penalty, we penalize it by e, otherwise, we penalize as it by d.

d) To detect opening of gap, we need to distinguish between three kinds of alignments.

- Alignments that end with no gaps
- Alignments that end with a gap aligned with Target sequence.
- Alignments that end with a gap aligned with Query sequence.

f) So there is a requirement of 3 types of block in alignment.

- A: block containing 1 character in Target seq aligned to 1 character in Query seq.
- B: block containing max number of contiguous gaps aligned with Target seq.
- C: block containing max number of contiguous gaps aligned with Query seq.

g) Now the dynamic programming table maintains three matrices of 3 kinds of alignments.
Recurrence Relation To fill Scoring Matrix

$$A(i,j) = min \left\{ \begin{array}{l} A(i-1,j-1) + s(x_i, y_i) \\ B(i-1,j-1) + s(x_i, y_i) \\ C(i-1,j-1) + s(x_i, y_i) \end{array} \right\} \tag{3.1}$$

$$B(i,j) = min \left\{ \begin{array}{l} A(i-1,j) + e \\ B(i-1,j) + d \\ C(i-1,j) + e \end{array} \right\} \tag{3.2}$$

$$C(i,j) = min \left\{ \begin{array}{l} A(i,j-1) + e \\ B(i,j-1) + e \\ C(i,j-1) + d \end{array} \right\} \tag{3.3}$$

h) Correspondingly the 3 trace Matrix for each A,B,C matrix is maintained to remember from which position particular index element has derived.

i) Final Step comes is Deducing the alignment from these traceback Matrix.

j) It always begins with the minimum out of three bottom right cell of the 3 scoring matrix A,B,C.

3.2 Implemented Code Description

a) Get the target and query string.

b) Declared 3 long long vectors for each Scoring matrix and traceback matrix.

c) Fills the Scoring matrix and traceback matrix according to the algorithm described above.

d) The next while loop traverse the traceback matrix.

e) The element can even derive from other traceback matrix as well. So we need to check whether the accessed element came from which traceback matrix (in my code I keep a check on it using "which_array" variable), and move into that particular direction respectively.

f) The next 2 while loops eat away any of the elements left in the target or Query sequence by adding gap for the other one.

e) Prints the results accordingly.

3.3 Time Complexity -

$O(mn)$ where **m** is Target seq and **n** is Query seq

3.4 Space Complexity -

$O(mn)$ where **m** is Target seq and **n** is Query seq