



Introduction to Bioinformatics

BS (CS – 460)

Lecture Set 03

Dr. Hafeez Ur Rehman

Assessment of Sequence Alignment

Methods of Sequence Alignment

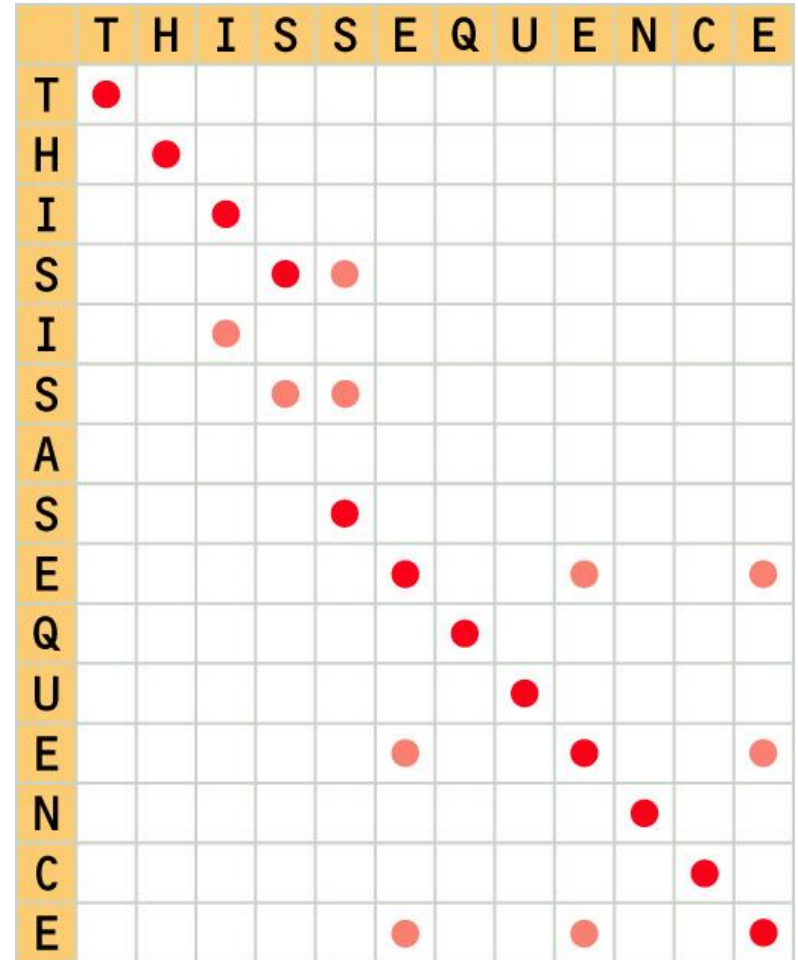
- Dot Plots
- Dynamic Programming
- Heuristics Methods

The Dot plot

- A “better” way of doing alignment is to represent each sequence as a table or matrix, where one sequence represents the rows and the other the columns. The Dot plot Matrix is a **visual** way of seeing the alignment between two sequences:
 - The first sequence (query sequence) represents the rows and the other sequence (subject sequence) represents the columns.
 - All elements (row/column) are checked for a match and if there is one, then the cell is marked.
 - This will show all areas of both sequences where matches occur.

Dot plot

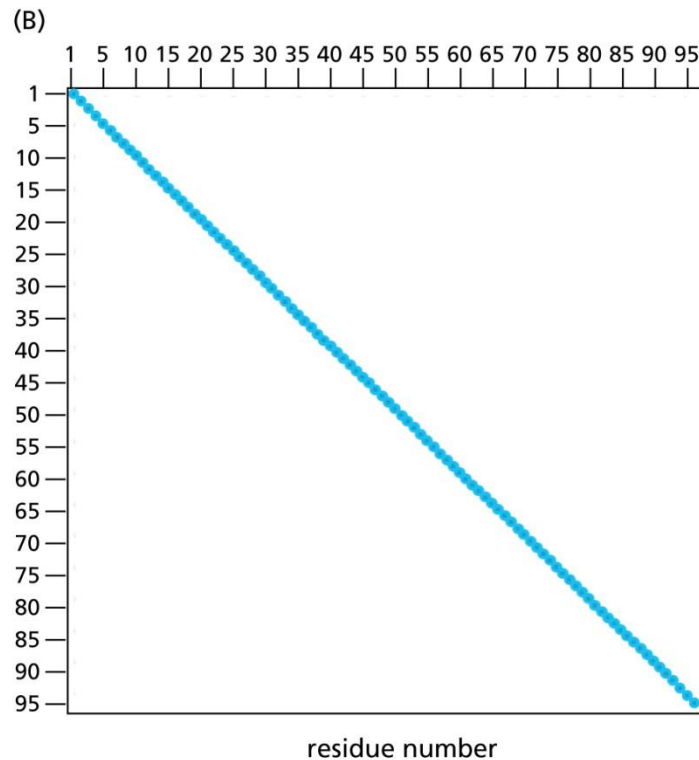
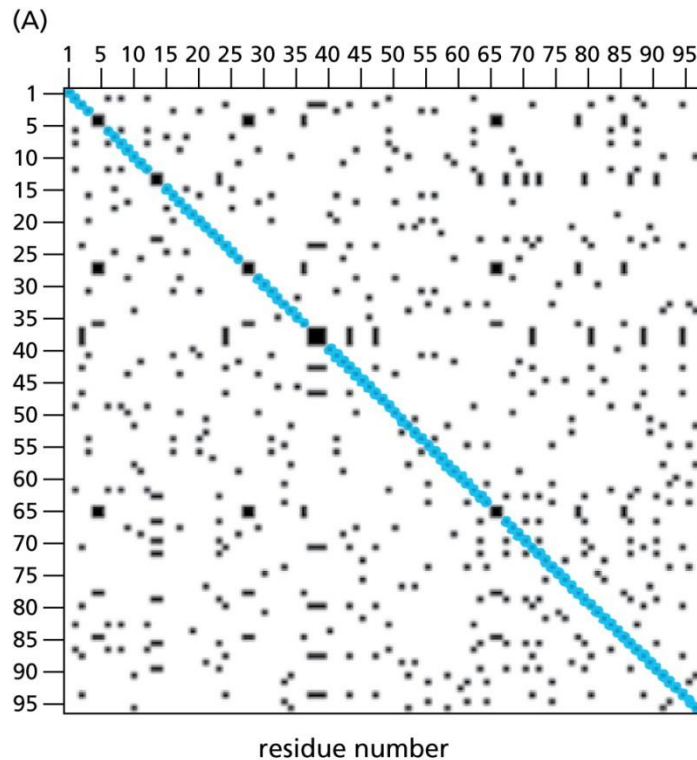
- Consider the following:
 - Diagonal lines represent an alignments (match)
 - Horizontal lines between aligned sequences indicate gaps are required (where the gaps indicate a deletion/insertion)
- Longest sequence of alignments are:
 - *“THIS” ; and “SEQUENCE”;*
 - *“IS” would be considered as gaps*
- *The pink dots: they can represent noise (spurious alignments)*



Dot plot Matrix: purpose

- This allows us to **visualise** areas of “**local alignment**” as opposed to global alignment.
- One of the main purpose to find domains / motifs that match . This could be useful for many reasons; e.g. promoter factor binding site, finding exons....
- For visualisation of pair-wise alignment you have one query on the x-axis and the other on the y-axis.

Dot Plot noise



This shows the effect of noise (blue line has been inserted to highlight alignment of interest. The figure on the left represents **SH2** sequence plotted against itself. The one on the right has been filter; in this case an alignment must be at least 10 residues long with a score of 3.

Applying Dot Plot Filter

- Evaluating similarity between 2 sequences
- **Window size** – number of nucleotides compare each time (usually odd number)*
- **Stringency** – the minimum number of nucleotides in the window must be “match”, so that a dot can be placed
- **Mismatch Limit** – the maximum number of nucleotides in the window can be “*not* match”, so that a dot can still be placed
- **Mismatch Limit = Window size - Stringency**

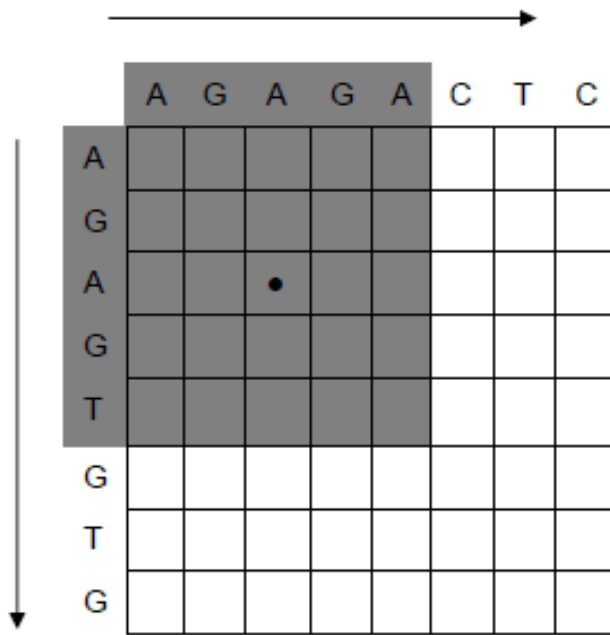
* Because we put dot in the center of the window

Example

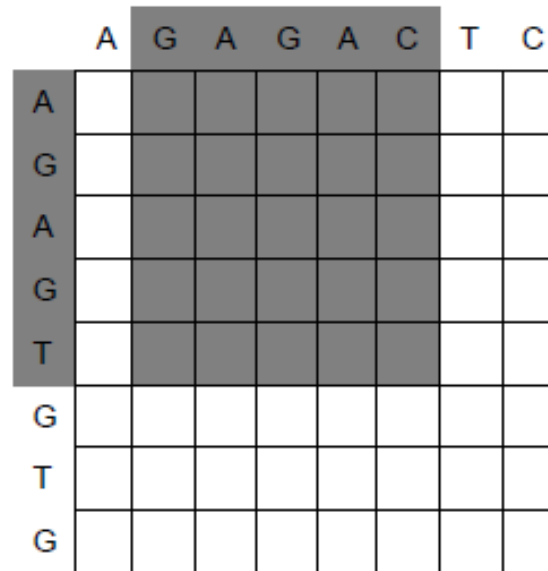
- Example 1: Compare the following sequences, with window size = 5, stringency = 3

AGAGACTC

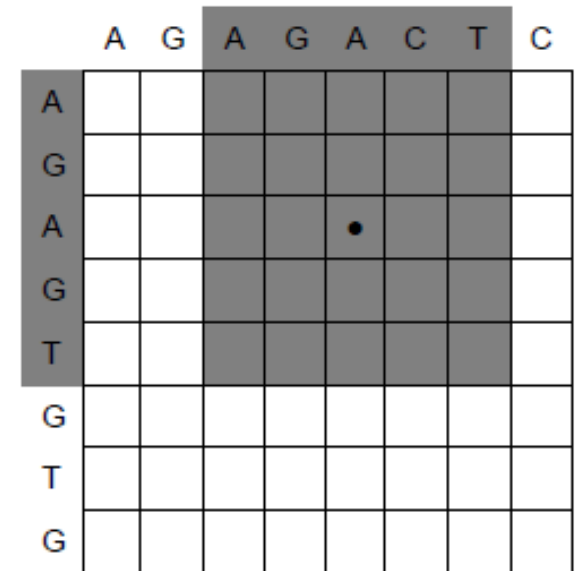
AGAGTGTG



4 matches



0 match



4 matches

Example (contd...)

	A	G	A	G	A	C	T	C
A								
G								
A								
G								
T								
G								
T								
G								

0 match

	A	G	A	G	A	C	T	C
A								
G								
A								
G								
T								
G								
T								
G								

0 match

	A	G	A	G	A	C	T	C
A								
G								
A								
G								
T								
G								
T								
G								

3 matches

Final answer →

	A	G	A	G	A	C	T	C
A								
G								
A			•		•			
G				•		•		
T			•					
G								
T								
G								

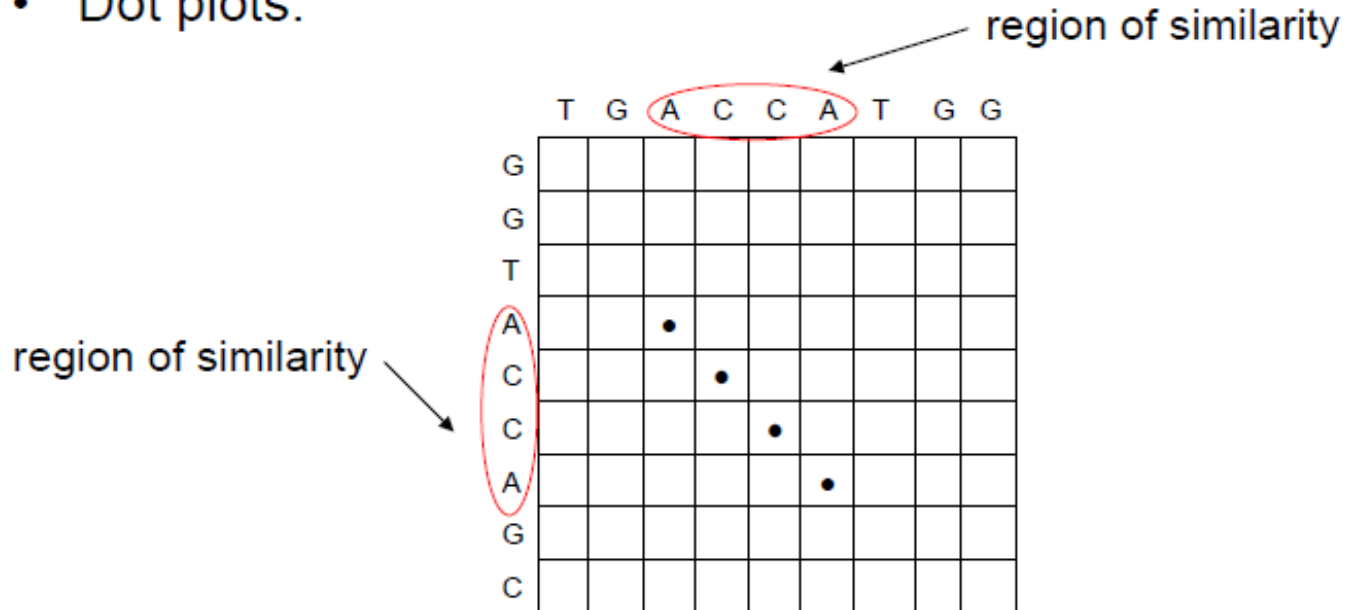
Example 02

- Example 2: Compare the following sequences and find the regions of similarity between two sequences. (window size = 5, stringency = 3)

TGACCATGG

GGTACCAGC

- Dot plots:

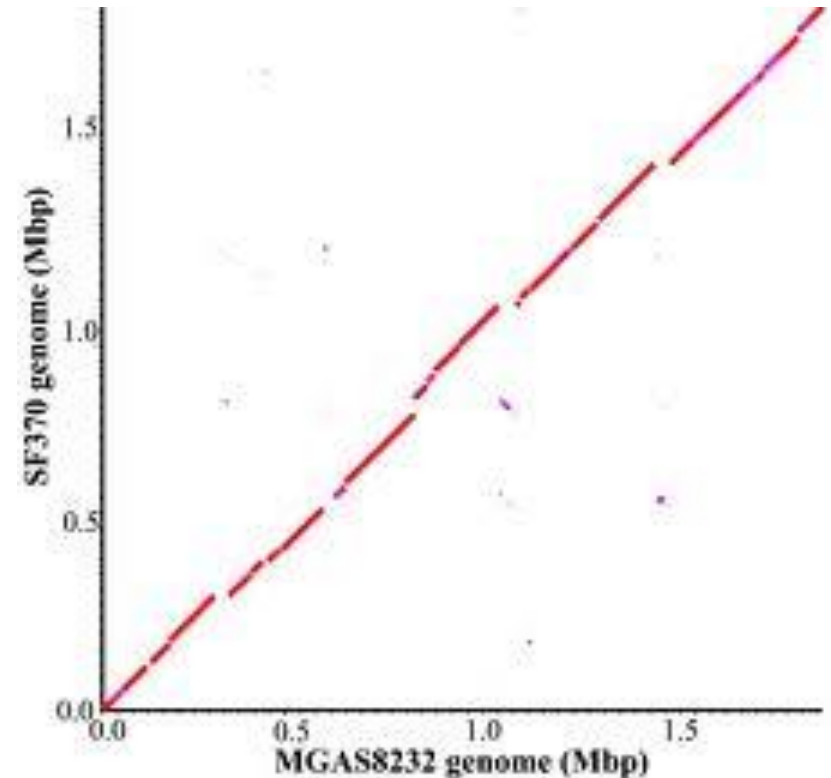


Which Window Size to Choose?

- Long window
 - Clean dot plots
 - Little sensitivity
- Short window
 - Noisy dot plots
 - Very sensitive
- The size of the window should be in the range of the elements you are looking for
 - Conserved domains: 50 amino acids
 - Transmembrane segments: 20 amino acids
- Shorten the window to compare distantly related sequences

Dot plot Matrix: imperfect match

- Some alignments **require gaps** to **increase the matching score**; the gaps are used to represent inclusion/deletion mutations
- The diagram shows that most of the 2 sequences are aligned.
- Gaps indicate areas of non-alignment i.e., gaps or substitutions.



Tandem repeat dot plot

- To determine if there is tandem repeats the sequence is compared with itself (see table 1 in next slide)
- The more diagonals the more repeats
- The **diagonals at the bottom** left compare the start with the finish
- The fact the main diagonal means the both sequences are the same .
- The lines are symmetrical around the main diagonal (next slide).

Dot plot of tandem repeats

	A	B	R	A	C	A	D	A	B	R	A	C	A	D	A	B	R	A
A	A							A							A			
B		B							B							B		
R			R							R							R	
A				A							A							A
C					C							C						
A						A							A					
D							D							D				
A	A							A							A			
B		B							B							B		
R			R							R							R	
A				A							A							A
C					C							C						
A						A							A					
D							D							D				
A	A							A							A			
B		B							B							B		
R			R							R							R	
A				A							A							A

Table 1: Dot plot of tandem repeats, adapted from Iesek 2008

The above dotplot illustrates the presence of **Tandem repeats** (there are 2TRs in the above dot plot: the main diagonal is not one)

A tandem repeat is a sequence that is repeated: consider the lower left hand diagonal, the sequences that cause this are: ABRA and ABRA Tandem repeats can be short tandem repeats (STR) 2 to 5 bp or tandem repeats (10 to 60 bp); the number of repeating segments can vary from 2 upwards and they can be used in areas such as genetic testing and intron detection. A comprehensive review is given by Gemayel et al 2010.

Tandem repeat as a sequence

Tandem repeat 1

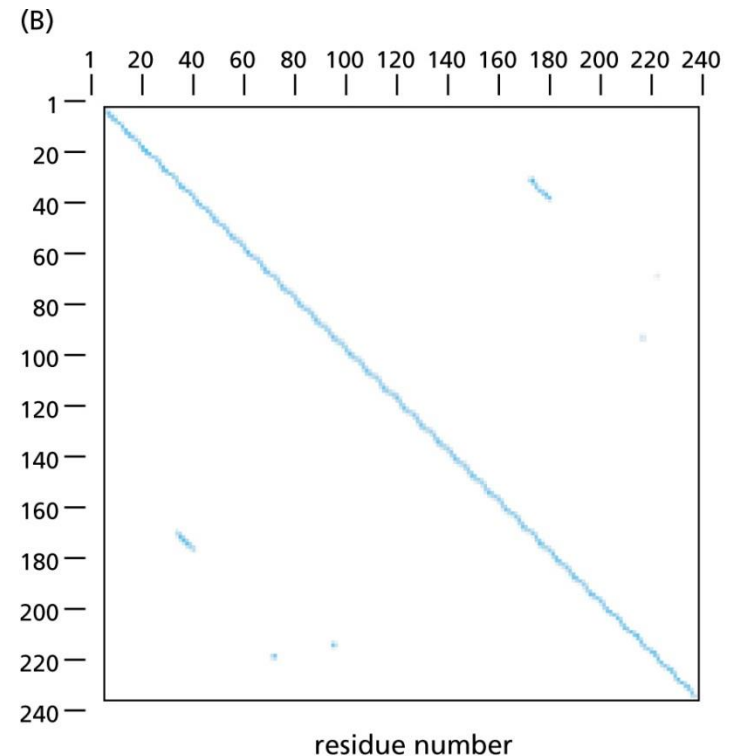
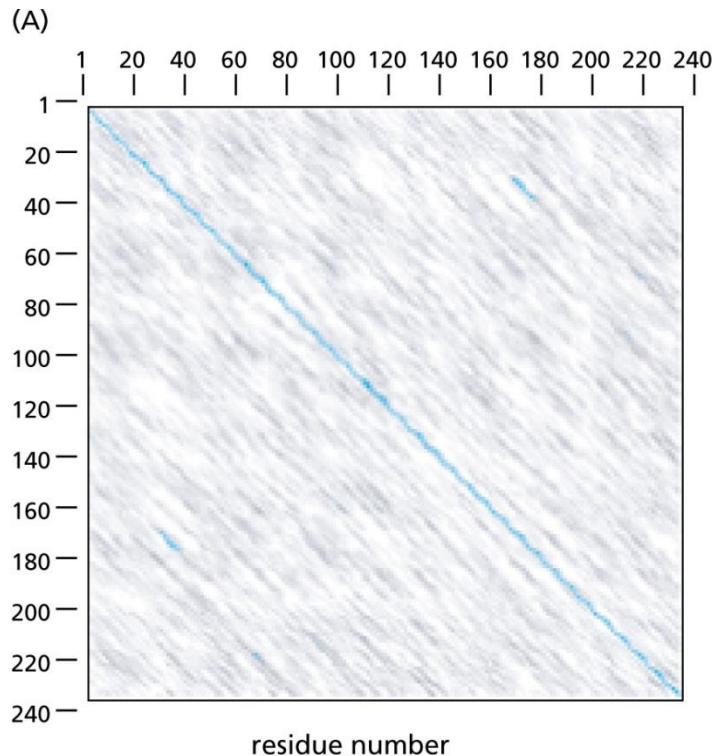
A	B	R	A	C	A	D	A	B	R	A	C	A	D	A	B	R	A
A	B	R	A	C	A	D	A	B	R	A	C	A	D	A	B	R	A

Tandem repeat 2

A	B	R	A	C	A	D	A	B	R	A	C	A	D	A	B	R	A
A	B	R	A	C	A	D	A	B	R	A	C	A	D	A	B	R	A

Tandem repeats (Example)

- BRCA2 gene has a number of BRC repeats (39 residues long). The diagram shows two plots: one with noise (unfiltered) and the other showing two repeating sequences.



Dynamic Programming

Pair-wise sequence alignments

Idea: Display one sequence above another with spaces inserted in both to reveal similarity

A:	C	A	T	-	T	C	A	-	C
B:	C	-	T	C	G	C	A	G	C

Two types of alignment

S = CTGTCGCTGCACG
T = TGCCGTG

Global alignment

CTGTCG-CTGCACG
-TGC-CG-TG----

Local alignment

CTGTCGCTGCACG--
-----TGC-CGTG

Global alignment: Scoring

CTGTCG-CTGCACG

-TGC-CG-TG-----

Reward for matches: α

Mismatch penalty: β

Space penalty (indel): γ

$$\text{score}(A) = \alpha w - \beta x - \gamma y$$

$w = \text{\#matches}$ $x = \text{\#mismatches}$ $y = \text{\#spaces}$

Global alignment: Scoring

Reward for matches: 10
Mismatch penalty: 2
Space penalty: 5

C	T	G	T	C	G	—	C	T	G	C
—	T	G	C	—	C	G	—	T	G	—

| —5 | 10 | 10 | —2 | —5 | —2 | —5 | —5 | 10 | 10 | —5 |

Total = 11

Optimum Alignment

- The score of an alignment is a measure of its quality
- **Optimum alignment problem:** Given a pair of sequences X and Y , find an alignment (global or local) with maximum score
- The **similarity** between X and Y , denoted $sim(X, Y)$, is the maximum score of an alignment of X and Y

Alignment algorithms

- Global: Needleman-Wunsch
- Local: Smith-Waterman
- Both use ***dynamic programming***
- Variations:
 - Gap penalty functions (Affine Penalty)
 - Scoring matrices

1. Global Alignment: Needleman-Wunsch

$S_{1..i}$ = Prefix of length i of S

$T_{1..j}$ = Prefix of length j of T

$C(i, j)$ = Score of optimum alignment of $S_{1..i}$ and $T_{1..j}$

$$w(a, b) = \begin{cases} +\alpha & \text{if } a = b & \text{Match Reward} \\ -\beta & \text{if } a \neq b & \text{Mismatch Penalty} \end{cases}$$

Theorem. $C(i,j)$ satisfies the following relationships:

Initial conditions:

$$C(i,0) = -i \cdot \gamma$$

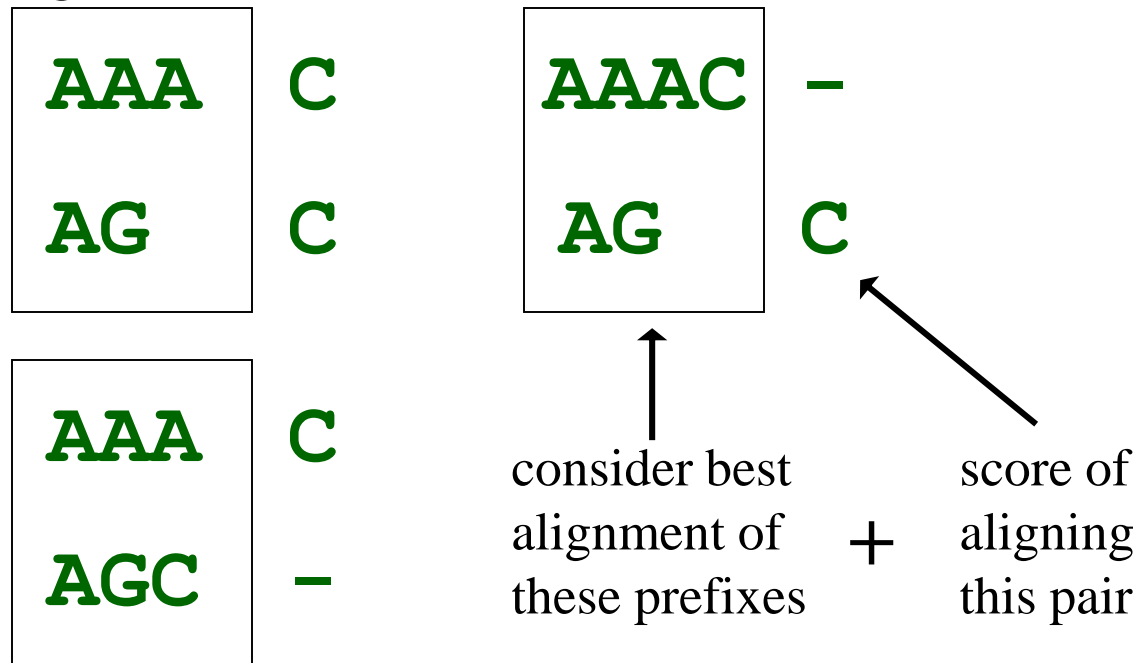
$$C(0,j) = -j \cdot \gamma$$

Recurrence relation: For $1 \leq i \leq n, 1 \leq j \leq m$:

$$C(i,j) = \max \begin{cases} C(i-1,j-1) + w(S_i, T_j) \\ C(i-1,j) - \gamma \\ C(i,j-1) - \gamma \end{cases}$$

Dynamic Programming Idea

- consider last step in computing alignment of **AAAC** with **AGC**
- three possible options; in each we'll choose a different pairing for end of alignment, and add this to the best alignment of previous characters



Justification

$$\begin{array}{|c|} \hline S_1 \ S_2 \ \dots \ S_{i-1} \ S_i \\ \hline T_1 \ T_2 \ \dots \ T_{j-1} \ T_j \\ \hline \end{array}$$

$$\underbrace{\hspace{10em}}_{C(i-1,j-1)} + \underbrace{\hspace{1em}}_{w(S_i, T_j)}$$

$$\begin{array}{|c|} \hline S_1 \ S_2 \ \dots \ S_{i-1} \ S_i \\ \hline T_1 \ T_2 \ \dots \ T_j \ - \\ \hline \end{array}$$

$$\underbrace{\hspace{10em}}_{C(i-1,j)} + \underbrace{\hspace{1em}}_{-\gamma}$$

$$\begin{array}{|c|} \hline S_1 \ S_2 \ \dots \ S_i \ - \\ \hline T_1 \ T_2 \ \dots \ T_{j-1} \ T_j \\ \hline \end{array}$$

$$\underbrace{\hspace{10em}}_{C(i,j-1)} + \underbrace{\hspace{1em}}_{-\gamma}$$

Example

Case 1: Line up S_i with T_j

					$i-1$		i		
S:	C	A	T	T	C		A	C	
T:	C	-	T	T	C		A	G	
					$j-1$		j		

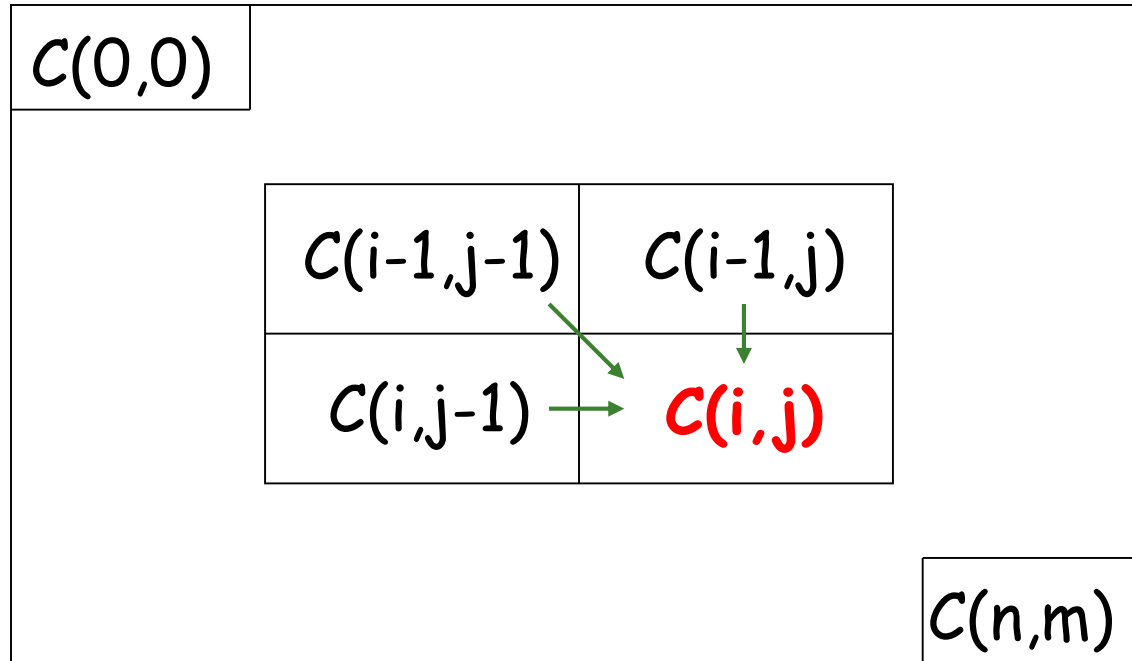
Case 2: Line up S_i with space

					$i-1$		i		
S:	C	A	T	T	C	A		-	C
T:	C	-	T	T	C	A		G	-
						j			

Case 3: Line up T_j with space

						i			
S:	C	A	T	T	C	A		C	-
T:	C	-	T	T	C	A		-	G
					$j-1$			j	

Computation Procedure



$$C(i, j) = \max \{ C(i-1, j-1) + w(S_i, T_j), C(i-1, j) - \gamma, C(i, j-1) - \gamma \}$$

	λ	C	T	C	G	C	A	G	C
λ	0	-5	-10	-15	-20	-25	-30	-35	-40
C	-5	10	5						
A	-10								
T	-15								
T	-20								
C	-25								
A	-30								
C	-35								

+10 for match, -2 for mismatch, -5 for space

	λ	C	T	C	G	C	A	G	C
λ	0	-5	-10	-15	-20	-25	-30	-35	-40
C	-5	10	5	0	-5	-10	-15	-20	-25
A	-10	5	8	3	-2	-7	0	-5	-10
T	-15	0	15	10	5	0	-5	-2	-7
T	-20	-5	10	13	8	3	-2	-7	-4
C	-25	-10	5	20	15	18	13	8	3
A	-30	-15	0	15	18	13	28	23	18
C	-35	-20	-5	10	13	28	23	26	33

Traceback can yield both optimum alignments

Traceback

- Horizontal arrow means gap in Y axis sequence.
- Vertical arrow means gap in X axis sequence.
- Diagonal arrow means insert corresponding characters.

Example Exercise:

- Align the following protein sequences globally using Needleman algorithm:

V L S P A V

S V L S A V

2. End-gap free alignment

- Gaps at the start or end of alignment are not penalized

$S = c a c t g t a c$
 $T = g a c a c t t g$

Match: +2

Mismatch and space: -1

Best global

c a c - - t - g t a c
g a c a c t t g - - -

Score = 1

Best end-gap free

- - c a c - t g t a c
g a c a c t t g - - -

Score = 9

Motivation: Shotgun assembly

- **Shotgun assembly** experiment produces large set of partially **overlapping subsequences** from many copies of one **unknown DNA sequence**.
- Problem: Use the overlapping sections to “paste” the subsequences together.
- Overlapping pairs will have low global alignment score, but high end-space free score because of overlap.

Motivation: Shotgun assembly

Strand	Sequence
Original	AGCATGCTGCAGTCATGCTTAGGCTA
First shotgun sequence	AGCATGCTGCAGTCATGCT-----1
	-----TAGGCTA
Second shotgun sequence	AGCATG-----
	-----CTGCAGTCATGCTTAGGCTA2
Reconstruction	AGCATGCTGCAGTCATGCTTAGGCTA

Algorithm

- Same as global alignment, except:
 - Initialize with zeros (free gaps at start)
 - Locate max in the last row/column (free gaps at end)

	λ	C	T	C	G	C	A	G	C
λ	0	0	0	0	0	0	0	0	0
C	0	10	5	10	5	10	5	0	10
A	0	5	8	5	8	5	20	15	10
T	0	0	15	10	5	6	15	18	13
T	0	-2	10	13	8	3	10	13	16
C	0	10	5	20	15	18	13	8	23
A	0	5	8	15	18	13	28	23	18
G	0	0	3	10	25	20	23	38	33

+10 for match, -2 for mismatch, -5 for gap

3. Local Alignment: Motivation

- **Ignoring stretches of non-coding DNA:**
 - Non-coding regions are more likely to be subjected to mutations than coding regions.
 - Local alignment between two sequences is likely to be between two exons.
- **Locating protein domains:**
 - Proteins of different kind and of different species often exhibit local similarities
 - Local similarities may indicate “functional subunits”.

Local alignment: Example

$S =$ g g t c t g a g
 $T =$ a a a c g a

Match: +2

Mismatch and space: -1

Best local alignment:

g g t c t g a g
a a a c - g a -

Score = 5

3. Local Alignment: Smith-Waterman Algorithm

$C[i, j]$ = Score of optimally aligning a suffix of s with a suffix of t .

$$C[i, j] = \max \begin{cases} C[i-1, j-1] + \text{score}(s[i], t[j]) \\ C[i-1, j] - \gamma \\ C[i, j-1] - \gamma \\ 0 \end{cases}$$

Initialize top row and leftmost column to zero.

	λ	C	T	C	G	C	A	G	C
λ	0	0	0	0	0	0	0	0	0
C	0	1	0	1	0	1	0	0	1
A	0	0	0	0	0	0	2	0	0
T	0	0	1	0	0	0	0	1	0
T	0	0	1	0	0	0	0	0	0
C	0	1	0	2	0	1	0	0	1
A	0	0	0	0	1	0	2	0	0
C	0	1	0	1	0	2	0	1	1

+1 for a match, -1 for a mismatch, -5 for a space

Backtracking starts at the highest scoring matrix cell and proceeds backwards until a cell with score zero is encountered, yielding the highest scoring local alignment.

Finally Computational Complexity

- initialization: $O(m)$, $O(n)$ where sequence lengths are m , n
- filling in rest of matrix: $O(mn)$
- traceback: $O(m + n)$
- hence, if sequences have nearly same length, the computational complexity is

$$O(n^2)$$

Thank you for your attention!

Questions?

Quiz # 03- Dated: 18-10-2017

1. Use dynamic programming to find local alignment of the following sequence pair:

G C G C T T C

T A G C T C G

In your answer I expect the filled matrix, tracing arrows and final alignment(s).

Match = +10

Mismatch= -5

Indel= -1