Sequence Analysis 1

B. Sequence alignment

Cheong Xin Chan (CX)

c.chan1@uq.edu.au

Australian Centre for Ecogenomics
School of Chemistry & Molecular Biosciences
The University of Queensland

SCIE2100 | BINF6000 | Bioinformatics 1 – Introduction

Lecture outline

- **Sequence alignment**
  - Modelling insertions and deletions using gap
  - linear *versus* affine gap penalty
- **Dynamic programming in sequence alignment**
  - Needleman-Wunsch algorithm
  - Smith-Waterman algorithm
  - Global versus local alignment
- **Dynamic programming and computational complexity**

Sequence alignment

- The fundamental way of comparing sequences
- Aim: to **quantify similarity** among a set of (two or more) sequences, which informs shared **homology**
- locates **equivalent regions** of two or more sequences to reveal (maximise) the extent of their **similarity**

Applications:

- inference of **phylogenetic** (i.e. evolutionary) relationships (dissimilarity as a measure of evolutionary distance)
- **prediction** of functions, structures and sequence features (e.g. binding sites, splicing signals) in novel sequences based on homology evidence to known sequences

Alignment is a position-by-position hypothesis of homology

Is a substitution matrix sufficient?

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	9																			
S	−1	4																		
T	−1	1	5																	
P	−3	−1	−1	7																
A	0	1	0	−1	4															
G	−3	0	−2	−2	0	6														
N	−3	1	0	−2	−2	0	6													
D	−3	0	−1	−1	−2	−1	1	6												
E	−4	0	−1	−1	−1	−2	0	2	5											
Q	−3	0	−1	−1	−1	−2	0	0	2	5										
H	−3	−1	−2	−2	−2	−2	1	−1	0	0	8									
R	−3	−1	−1	−2	−1	−2	0	−2	0	1	0	5								
K	−3	0	−1	−1	−1	−2	0	−1	1	1	−1	2	5							
M	−1	−1	−1	−2	−1	−3	−2	−3	−2	0	−2	−1	−1	5						
I	−1	−2	−1	−3	−1	−4	−3	−3	−3	−3	−3	−3	−3	1	4					
L	−1	−2	−1	−3	−1	−4	−3	−4	−3	−2	−3	−2	−2	2	2	4				
V	−1	−2	0	−2	0	−3	−3	−3	−2	−2	−3	3	2	1	3	1	4			
F	−2	−2	−2	−4	−2	−3	−3	−3	−3	−3	−1	−3	−3	0	0	0	−1	6		
Y	−2	−2	−2	−3	−2	−3	−2	−3	−2	−1	2	−2	−2	−1	−1	−1	−1	3	7	
W	−2	−3	−2	−4	−3	−2	−4	−4	−3	−2	−2	−3	−3	−1	−3	−2	−3	1	2	11

```
seqA QRVYPSVPFSAGP
seqB LRKYPVLPFSAGP
```

Example

$$\boldsymbol{S}_{AB} = S_{\mathtt{QL}} + S_{\mathtt{RR}} + S_{\mathtt{VK}} + S_{\mathtt{YY}} + S_{\mathtt{PP}} + \ldots$$

Sum of log-odds scores is used to assign a score to an alignment

Are we missing anything?

Insertions and deletions

- evolutionary events that result in the introduction of new bases into (**insertion**), or the removal of existing bases from (**deletion**), the genomic DNA
- **Indels:** mutation that includes both insertions, deletions and the combination thereof
- modelled as **gaps** in sequence alignment

Example

gaps in `Seq1` (**deletion** in `Seq1`)

```
Seq1 CAGTT---CGAT
Seq2 CAGTTAGGCGAT
```

insertion in `Seq2`

Modelling indels using gap penalty

Gaps in an alignment ↑ **Similarity** between two sequences ↓

Gaps are modelled as a **penalty**, i.e. assigned as a negative score

Linear gap penalty for a gap of length n, $g(n)$ is

$$g(n) = -n \times E$$ where E is a cost for a single gap

$$g(n) \propto n$$

Example

```
Seq1 CAGTT---CGAT
Seq2 CAGTTAGGCGAT
```

If $E = 1$, then $g(n) = -3$

Every gap position costs the same

Modelling indels using gap penalty

How many indel event(s) could have occurred in Seq2?

a. An insertion of **A**, an insertion of **G** and an insertion of **G** (3 events)
b. An insertion of **AG**, and an insertion of **G** (2 events)
c. An insertion of **AGG** (1 event), etc.

Example

```
Seq1 CAGTT---CGAT
Seq2 CAGTTAGGCGAT
```

Is it fair to treat every single gap position the same?

Gap open: the cost of opening/starting a gap
Gap extend: the cost of extending a gap by one position

Affine gap penalty for a gap of length n, $g(n)$ is

$$g(n) = -O - (n-1) \times E$$

where O is the gap open cost, E is the gap extend cost

If $O = 3$, $E = 1$, then $g(n) = -5$

Many small gaps cost more than one large gap

Calculating score for an alignment with gap

Alignment score $S = s + g$

s = total substitution score
g = total gap penalty (in negative)

Examples

	Linear gap scheme $g(n) = -n \times E$ $E = 2$	Affine gap scheme $g(n) = -O - (n-1) \times E$ $O = 3, E = 1$
g(3) SVDNA---RHV SISQSAQLSHV 43001 ↑84 −1	$S = 19 + (-6)$ $= 13$	$S = 19 + (-5)$ $= 14$
g(1) *g(2)* SVDN-A--RHV SISQSAQLSHV 4300 4 ↑84 −1	$S = 22 + (-6)$ $= 16$	$S = 22 + (-7)$ $= 15$

Impact of gap penalty on an alignment

Example

Alignment A

```
seqA LNWENPDIMSELLFQNNEIIFKNGDDLRQDMLTLQIIRIMENIWQNQGLDLRMLPYGCLSIGDCVGLIEVVRNSHTIMQIQCKGGLKGAL
seqB --WENPAQNTAHLDQFERIKTLGTGSFGRVMLVKHMETGNHYAMKILDKQKVVKLKQIEHTLNEKRILQAVNFPFLVKLEFSFKDNSNLY

seqA QFNSHTLHQWLKDKNKGEIYDAAIDLFTRSCAGYCVATFILGIGDRHNSNIMVKDDGQLFHIDFGHFLDHKKKFGYKRERVPFVLTQDF
seqB MVMEYVPGGEMFSHLRRIGRFSEPHARFYAAQIVLTFEYLHSLDLIYRDLKPENLLIDQQGYIQVTDFGFAKRVKGRTWXLCGTPEYLAP

seqA LIVISKGAQECTKTREFERFQEMCYKAYLAIRQHANLFINLFSMMLGSGMPELQSFDDIAYIRKTLALDKTEQEALEYFMKQMNDAHHGG
seqB EIILSKGYNKAVDWWALGVLIYEMAAGYPPFFADQPIQIYEKIVSGKVRFPSHFSSDLKDLLRNLLQVDLTKRFGNLKNGVNDIKNHKWF

seqA WTTKMDWIFHTIKQHALN-----------------------------------
seqB ATTDWIAIYQRKVEAPFIPKFKGPGDTSNFDDYEEEEIRVXINEKCGKEFSEF
```

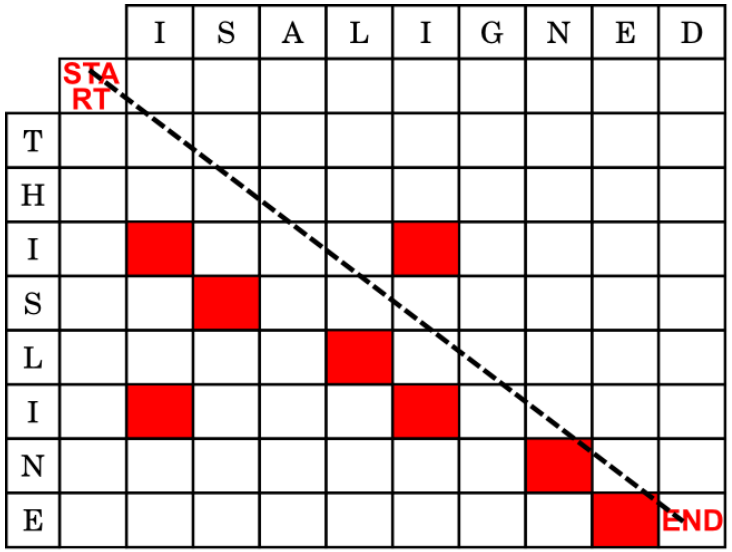
Alignment B

```
seqA LNWENPDIMSELLFQNNEIIFKNGDDLRQDMLTLQIIRIMENIWQNQGLDLRMLPYGCLSIGDCVGLIEVVRNSHTIMQIQCKGGLKGAL
seqB ?-WENPAQNTAHLDQFERIKTLGTGSFGRVMLVKHM--ETGNHYAMKILDKQKV-VKLKQIEHTLNEKRILQAVNFPFLVKLEFSFKDN-

seqA QFNSHTLHQWLKDKNKGEIYDAAIDLFTRSCAGYCVATFILGIGDRHNSNIMVKD-DGQLFHIDFGHFLDHKKKKFGYKRERVPFVL--T
seqB -SNLYMVMEYVPGGEMFSHLRR-IGRFSEPHARFYAAQIVLTFEYLHSLDLIYRDLKPENLLIDQQGYIQVTDFGFAKRVKGRTWXLCGT

seqA QDFL---IVISKGAQECTKTREFERF-QEMC--YKAYLAIRQHANLFINLFSMMLGSGMPELQSFDDIAYIRKTLALDKTEQEALEYFMK
seqB PEYLAPEIILSKGYNKAVDWWALGVLIYEMAAGYPPFFA-DQPIQIYEKIVSGKVRF--PSHFSSDLKDLLRNLLQVDLTKR--FGNLKN

seqA QMNDAHHGGWTTKMDWI-----------------------FHTIKQHAL----N----------
seqB GVNDIKNHKWFATTDWIAIYQRKVEAPFIPKFKGPGDTSNFDDYEEEEIRVXINEKCGKEFSEF
```

Which alignment is a result of having a higher (more costly) gap penalties?

Matrix representation of an alignment

Example



T H I S L I N E

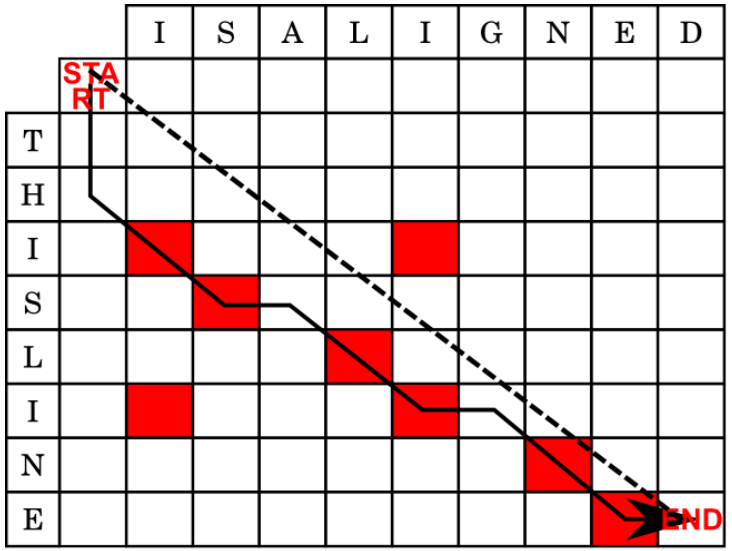
I S A L I G N E D

Gap in first line

Aligned positions

Gap in second line

Dynamic programming aims to find the optimal (best) path



T H I S – L I – N E –

– – I S A L I G N E D

Gap in first line

Aligned positions

Gap in second line

More diagonal moves = more identical positions aligned

Dynamic programming in sequence alignment

- reduces a big, hard, **optimisation problem into smaller problems** whose results can be combined: optimisation of alignment = sum of sub-alignment optimisations

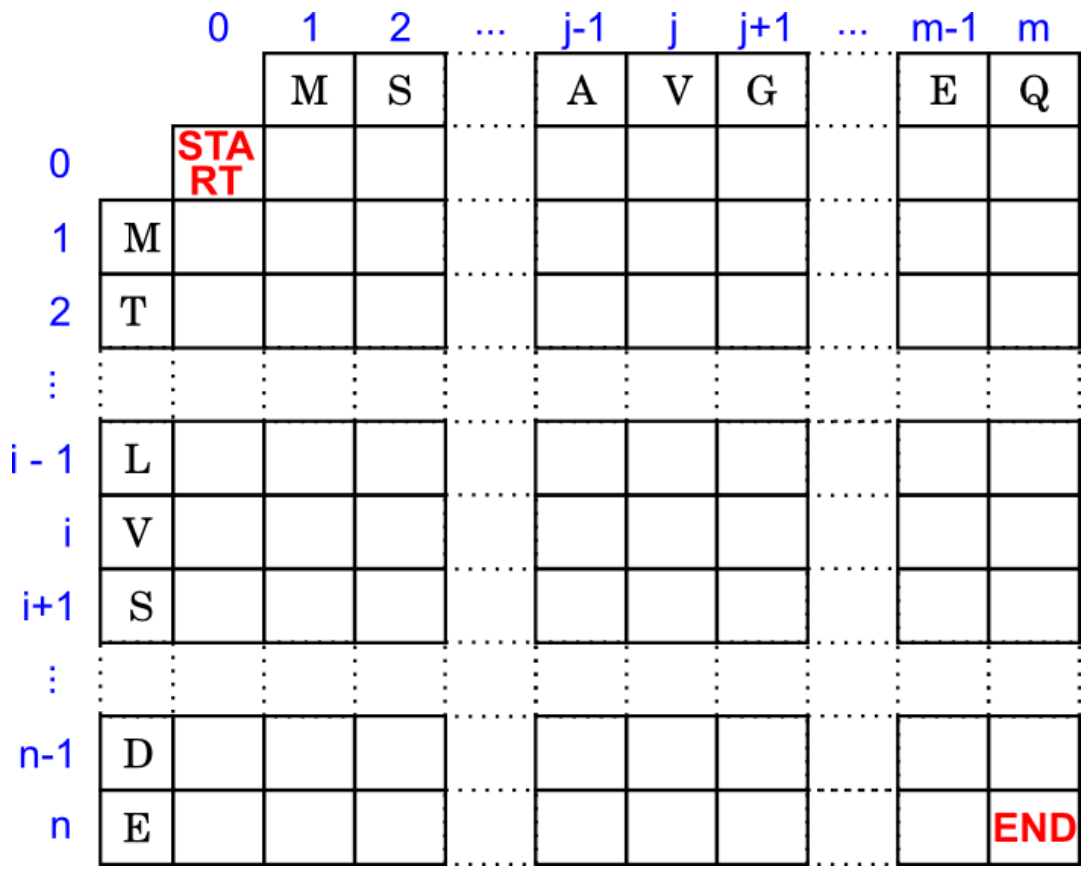
T	H	I	S	–	L	I	–	N	E	–	
–	–	I	S	A	L	I	G	N	E	D	

$$S = S_1 + S_2 + S_3$$

- optimisation problem in time proportional to the product of two sequence lengths, m and n: $O(n \cdot m)$
- transforms a sequence into another using edit operations that **replace**, **insert**, or **remove** an element
- each operation has an associated **cost**; the goal is to find the sequence of edits with the **lowest** total cost

Dynamic programming in sequence alignment

Consider seq1 of length n, seq2 of length m. Let $S_{i,j}$ be the score for the best alignment ending at position i in seq1 and position j in seq2.



Four steps:

1. Initialisation

$S(0,0) = 0$

$S(0, j)$ and $S(i, 0)$ based on gaps

2. Recurrence

With additive costs, we can compute $S_{i,j}$ recursively from $S_{i-1,j-1}$, $S_{i-1,j}$ and $S_{i,j-1}$

3. Termination

4. Trace backward

Needleman-Wunsch algorithm

- First description of a similarity-searching method among biological sequences (Needleman & Wunsh, 1970)
- Aims to produce an alignment by inexact string matching i.e. an alignment that incorporates **matches**, **mismatches** and **gaps** placed as required, in order to give the best possible alignment (the **optimal** alignment).
- **All** positions in a sequence are compared against **all** positions in another (i.e. in a **global** alignment)
- longest line/path is found by tracing back through the matrix (i.e. the similarity procedure)
- adopted in all commonly used alignment and similarity search programs

NW algorithm: initiation

X → Y ↓	gap	T	A	G	C
gap	0	← -3	← -6	← -9	← -12
T	↑ -3				
A	↑ -6				
C	↑ -9				

SeqY: TAC

SeqX: TAGC

Scoring scheme:
- match (**+3**)
- mismatch (**+ 0**)
- gap (**–3**)

- **initialise their back-pointers** (arrows indicate where the accumulative score was obtained)
- scores are accumulative from left to right, from top to bottom

NW algorithm: recurrence

X → Y ↓	gap	T	A	G	C
gap	0	-3	-6	-9	-12
T	-3	#			
A	-6				
C	-9				

SeqX: TAGC

SeqY: TAC

↑: a **gap** relative to Y (**−3**)

←: a **gap** relative to X (**−3**)

↖: comparing the character pair, **T**-to-**T** **match** (**3**)

Scoring scheme:

- match (**+3**)
- mismatch (**+ 0**)
- gap (**−3**)

Sum each score **+** score in the box in the corresponding direction

↑:(−3)+(−3) = −6

←:(−3)+(−3) = −6

↖ :(3)+(0) = 3

maximum score: **3** at direction ↖

NW algorithm: recurrence

X → Y ↓	gap	T	A	G	C
gap	0	← -3	← -6	← -9	← -12
T	↑ -3	↖ 3	@		
A	↑ -6				
C	↑ -9				

SeqX: TAGC

SeqY: TAC

Scoring scheme:
- match (+3)
- mismatch (+ 0)
- gap (−3)

Sum each score + score in the box in the corresponding direction

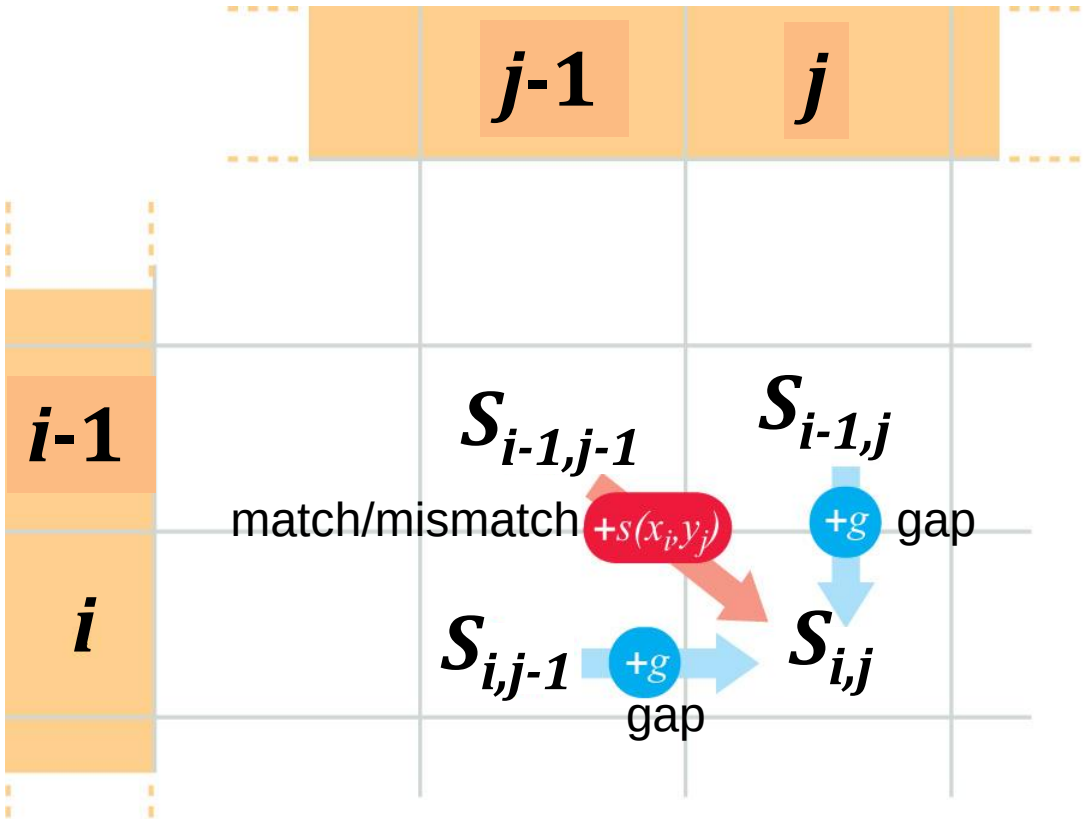
↑:(−3)+(−6) = −9
←:(−3)+(3) = 0
↖ :(0)+(−3) = −3

↑: a **gap** relative to Y (−3)
←: a **gap** relative to X (−3)
↖: comparing the character pair, **A**-to-**T** **mismatch** (0)

maximum score: **0** at direction ←
and so forth, until all boxes are filled

NW algorithm: recurrence

$\boldsymbol{S_{i,j}}$ is the score for the **best** alignment of the initial segments of sequence $\boldsymbol{x}$ and sequence $\boldsymbol{y}$ ending at position $\boldsymbol{i}$ and $\boldsymbol{j}$, respectively



$$\boldsymbol{S_{i,j}} = \max\begin{pmatrix} S_{i-1,j-1} + s(x_i, y_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \end{pmatrix}$$

NW algorithm: termination and tracing back

X → Y ↓	*gap*	**T**	**A**	**G**	**C**
gap	0	← -3	← -6	← -9	← -12
T	↑ -3	↖ 3	← 0	← -3	← -6
A	↑ -6	↑ 0	↖ 6	← 3	← 0
C	↑ -9	↑ -3	↑ 3	↖ 6	↖ 6

`SeqY: TAC`

`SeqX: TAGC`

- Terminates at the **bottom-right** box (the end of both sequences)
- Traces from the **bottom-right** cell backward towards the **top-left** cell

```
seqX TAGC
     :: :
seqY TA-C
```

the **optimal** alignment

total score = **6**

See Fig. 5.11 (p131) on textbook for another example.

Smith-Waterman algorithm

modified from the NW algorithm to find **locally** matched regions between two sequences (i.e. a **local** alignment algorithm)

seqX TAGC

X → Y ↓	gap	T	A	G	C
gap	0	← 0	← 0	← 0	← 0
A	↑ 0	↖ 0	↖ 3	← 0	↖ 0
G	↑ 0	↖ 0	↑ 0	↖ 6	← 3
T	↑ 0	↖ 3	← 0	↑ 3	↖ 5

seqY AGT

the **optimal** alignment
total score **= 6**

```
seqX AG
     ::
seqY AG
```

- **mismatches** are assigned negative scores
- negative score is set to **zero** (i.e. the minimum is non-negative)

$$S_{i,j} = \max \begin{pmatrix} S_{i-1,j-1} + s(x_i, y_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \\ 0 \end{pmatrix}$$

- traces back from the box with the **highest** score, through the alignment until a **zero** is reached

Global versus local alignment

- attempts to align **every residue in every sequence**, i.e including both highly conserved and highly variable regions
- most useful when the sequences are **similar** and of **roughly equal size**
- **less** prone to demonstrating **false homology**

- focuses only on **conserved** regions in the sequences
- useful for **dissimilar sequences** that are expected to contain **conserved** regions (e.g. protein domains) or similar sequence **motifs** (e.g. binding or active sites) within their **larger sequence context**
- **more** prone to demonstrating **false homology**

Global versus local alignment

	Needleman-Wunsch (global)	**Smith-Waterman** (local)
Initiation	$S(0,0)=0$ $S(i,0)=g(i)$ $S(0,j)=g(j)$	$S(0,0)=0$ $S(i,0)=0$ $S(0,j)=0$
Recurrence	$\boldsymbol{S_{i,j}} = \max\begin{pmatrix} S_{i-1,j-1}+s(x_i,y_j) \\ S_{i-1,j}+g \\ S_{i,j-1}+g \end{pmatrix}$	$\boldsymbol{S_{i,j}} = \max\begin{pmatrix} S_{i-1,j-1}+s(x_i,y_j) \\ S_{i-1,j}+g \\ S_{i,j-1}+g \\ 0 \end{pmatrix}$
Trace back from	Bottom-right cell	Highest scoring cell
Trace back until	Top-left cell	Cell with zero score

Dynamic programming and computational complexity

- DP determines optimal alignments by resolving optimal partial alignments; it guarantees the best alignment (solution), largely feasible for **pairwise sequence alignment**
- For ***n* individual sequences**, it requires constructing *n*-dimensional equivalent of the matrix formed in a standard pairwise alignment
- search space increases **exponentially** with ***n*** and strongly dependent on **sequence length**
- feasibility becomes an issue when aligning **three or more** sequences (i.e. **multiple sequence alignment**), prompting for heuristics (Week 4)

Reflection

- *Why would we want to align two DNA sequences (or two protein sequences)?*
- *What are the two common approaches to model insertions and deletions in sequence alignment?*
- *How does dynamic programming help in finding the best pairwise alignment?*
- *What are the key steps in dynamic programming?*
- *What are the key differences between a global alignment and a local alignment?*
- *Is dynamic programming feasible for aligning many, many sequences?*