

# Tin Sinh học Bioinformatics

## Chương 2. Xếp hàng hai chuỗi

TS. Nguyễn Hồng Quang  
Khoa Kỹ thuật máy tính

Leader of Bioinformatics Group, BK.AI center  
Trường Công nghệ thông tin và Truyền thông  
Trường Đại học Bách Khoa Hà Nội

# Tài liệu tham khảo

Algorithms in Bioinformatics: A  
Practical Introduction  
(nus.edu.sg)

[https://www.comp.nus.edu.sg/~k  
sung/algo\\_in\\_bioinfo/](https://www.comp.nus.edu.sg/~k<br/>sung/algo_in_bioinfo/)

# Nội dung chính

- 2.1. Introduction
- 2.2. Global Alignment Problem
- 2.3. Local Alignment
- 2.4. Semi-Global Alignment
- 2.5. Gap Penalty
- 2.6. Scoring Function
- 2.7. Exercises

# Tại sao chúng ta cần so sánh các trình tự?

- Giả định Sinh học : Cho hai DNA (hoặc RNA, hoặc protein) có độ tương đồng cao => chức năng tương tự hoặc cấu trúc 3D tương tự
- Ứng dụng:
  - Giải trình tự bộ gen
  - Tìm các chuỗi con chung trong hai bộ gen
  - Xác định đột biến gen (SNP)

# String Edit

- Cho hai chuỗi A và B, hãy **chỉnh sửa A thành B** với số thao tác chỉnh sửa tối thiểu:
  - Thay thế một chữ cái bằng một chữ cái khác
  - Chèn một chữ cái
  - Xóa một chữ cái
- Ví dụ.

- A = interestingly  
B = bioinformatics

i\_nterestingly  
b\_i\_o\_informati\_cs\_  
1011011011001111

- Edit distance = 11




**Xác định dãy thao tác  
chỉnh sửa để *Tối ưu*  
*tổng khoảng cách***

# Vấn đề liên kết chuỗi


- Thay vì sử dụng thuật ngữ “chỉnh sửa chuỗi” (string edit), thường sử dụng “căn chỉnh chuỗi” (string alignment).
- Sử dụng hàm tương tự (Similarity function) thay vì hàm chi phí (Cost function).
- Ví dụ hàm tương tự:
  - phù hợp: 2
  - không phù hợp, chèn, xóa: -1

$$\delta(C, G) = -1$$

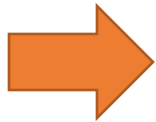


	—	A	C	G	T
—		-1	-1	-1	-1
A	-1	2	-1	-1	-1
C	-1	-1	2	-1	-1
G	-1	-1	-1	2	-1
T	-1	-1	-1	-1	2

# An alignment of two sequences

- An alignment of two sequences:
  - Inserting spaces in arbitrary locations along the sequences so that they end up with the same length and 
  - there are no two spaces at the same position of the two augmented sequences.

-i--nterestingly  
bioinformatics--



find an alignment  $A$  which

$$\text{maximizes } \sum_{(x,y) \in A} \delta(x,y)$$

tổng của hàm tương tự là  
lớn nhất



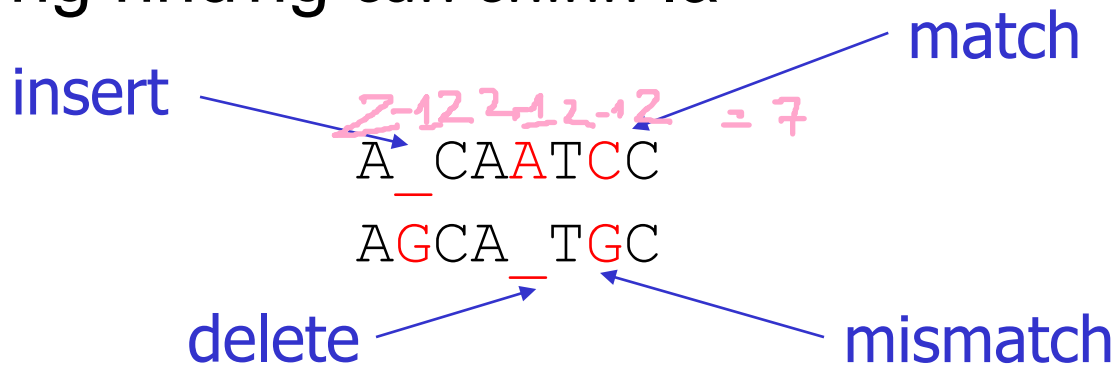
optimal alignment

# String alignment

- Xét hai chuỗi ACAATCC và AGCATGC.

	_	A	C	G	T
_		-1	-1	-1	-1
A	-1	2	-1	-1	-1
C	-1	-1	2	-1	-1
G	-1	-1	-1	2	-1
T	-1	-1	-1	-1	2

Một trong những căn chỉnh là



Trong căn chỉnh trên:

- khoảng trắng ('\_': space) được đưa vào cả hai chuỗi
- Có 5 kết quả phù hợp (match), 1 không khớp (mismatch), 1 chèn (insertion) và 1 xóa (deletion) => 2 indels.



# Bài tập

- Cho hai chuỗi  $S1 = \text{ACAATCC}$  và  $S2 = \text{AGCATGC}$
- Hãy đưa ra ví dụ căn chỉnh chuỗi với độ dài  $L = 7, 8, 9$ . Tính độ tương đồng (matching score) trong từng trường hợp.

	—	A	C	G	T
—		-1	-1	-1	-1
A	-1	2	-1	-1	-1
C	-1	-1	2	-1	-1
G	-1	-1	-1	2	-1
T	-1	-1	-1	-1	2

# String alignment problem

A \_ C A **A** T **C** C  
A **G** C A \_ T **G** C

	_	A	C	G	T
_		-1	-1	-1	-1
A	-1	2	-1	-1	-1
C	-1	-1	2	-1	-1
G	-1	-1	-1	2	-1
T	-1	-1	-1	-1	2


- Các khái niệm cơ bản:
  - Điểm tối đa : **maximum score**
  - Căn chỉnh tối ưu: **optimal alignment**
  - Bài toán căn chỉnh chuỗi (**String alignment problem**): tìm sự liên kết với điểm tương đồng tối đa
  - Bài toán căn chỉnh toàn cục (**Global alignment**) : căn chỉnh toàn bộ các ký tự trong chuỗi

# Thuật toán Needleman-Wunsch

- Xét hai chuỗi  $S[1..n]$  và  $T[1..m]$
- $V(i, j)$  là tổng điểm của liên kết tối ưu giữa  $S[1..i]$  và  $T[1..j]$
- Khởi tạo:
  - $V(0, 0) = 0$
  - $V(0, j) = V(0, j-1) + \delta(\_, T[j])$ 
    - Insert  $j$  times
  - $V(i, 0) = V(i-1, 0) + \delta(S[i], \_)$ 
    - Delete  $i$  times

chuỗi đích

chuỗi nguồn



		chuỗi nguồn							
		—	A	G	C	A	T	G	C
	—	0	-1	-2	-3	-4	-5	-6	-7
A	-1								
C	-2								
A	-3								
A	-4								
T	-5								
C	-6								
C	-7								

# Ví dụ

	—	A	G	C	A	T	G	C
—	0	-1	-2	-3	-4	-5	-6	-7
A	-1							
C	-2							
A	-3							
A	-4							
T	-5							
C	-6							
C	-7							

# Thuật toán Needleman-Wunsch

- Bước lặp : For  $i > 0, j > 0$

$$V(i, j) = \max \begin{cases} V(i-1, j-1) + \delta(S[i], T[j]) & \text{match/mismatch} \\ V(i-1, j) + \delta(S[i], -) & \text{delete} \\ V(i, j-1) + \delta(-, T[j]) & \text{insert} \end{cases}$$

- Trong căn chỉnh, cặp cuối cùng phải khớp / không khớp, xóa hoặc chèn.

xxx...xx	xxx...xx	xxx...x_
yyy...yy	yyy...y_	yyy...yy
match/mismatch	delete	insert

# Ví dụ

	_	A	G	C	A	T	G	C
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1	2	1	0	-1	-2	-3	-4
C	-2	1	1	3	2			
A	-3							

$$V(i, j) = \max \begin{cases} V(i-1, j-1) + \delta(S[i], T[j]) & \text{match/mismatch} \\ V(i-1, j) + \delta(S[i], -) & \text{delete} \\ V(i, j-1) + \delta(-, T[j]) & \text{insert} \end{cases}$$

C	-7							
---	----	--	--	--	--	--	--	--

# Backward: Xác định căn chỉnh tối ưu

A \_CAATCC  
AGCA \_TGC

	_	A	G	C	A	T	G	C
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1	2	1	0	-1	-2	-3	-4
C	-2	1	1	3	2	1	0	-1
A	-3	0	0	2	5	4	3	2
A	-4	-1	-1	1	4	4	3	2
T	-5	-2	-2	0	3	6	5	4
C	-6	-3	-3	0	2	5	5	7
C	-7	-4	-4	-1	1	4	4	7

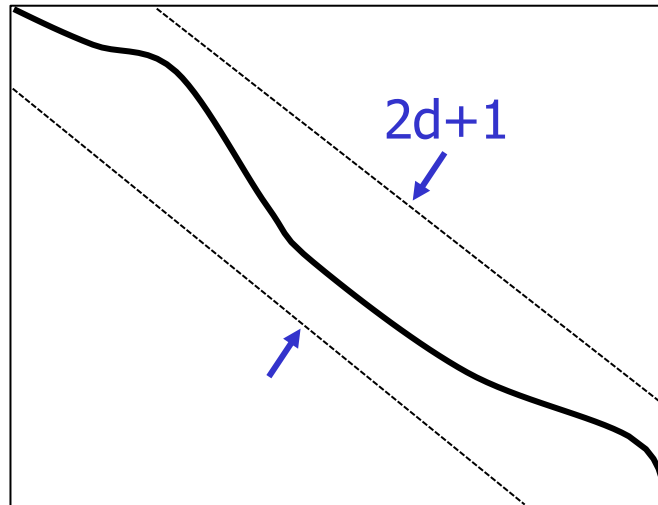
# Phân tích độ phức tạp của thuật toán

- Cần điền vào tất cả các mục trong bảng của ma trận  $n \times m$ .
- Mỗi mục nhập có thể được tính trong thời gian  $O(1)$ .
- Độ phức tạp thời gian tính toán =  $O(nm)$
- Độ phức tạp không gian bộ nhớ =  $O(nm)$



# Giới hạn số thao tác chèn – xóa

- Nếu giới hạn  $d$ : maximum number of insertions or deletions (indel)
- Khi đó căn chỉnh phải nằm trong dải  $2d + 1$ .
- Do đó, không cần điền vào hình tam giác phía dưới và phía trên.
- Độ phức tạp về thời gian:  $O(dn)$ .



# Ví dụ với $d = 3$

A \_ CAATCC

AGCA \_ TGC

	_	A	G	C	A	T	G	C
_	0	-1	-2	-3				
A	-1	2	1	0	-1			
C	-2	1	1	3	2	1		
A	-3	0	0	2	5	4	3	
A		-1	-1	1	4	4	3	2
T			-2	0	3	6	5	4
C				0	2	5	5	7
C					1	4	4	7

# Bài tập

- Cho hai chuỗi

S = TCATAG

T = GATAC

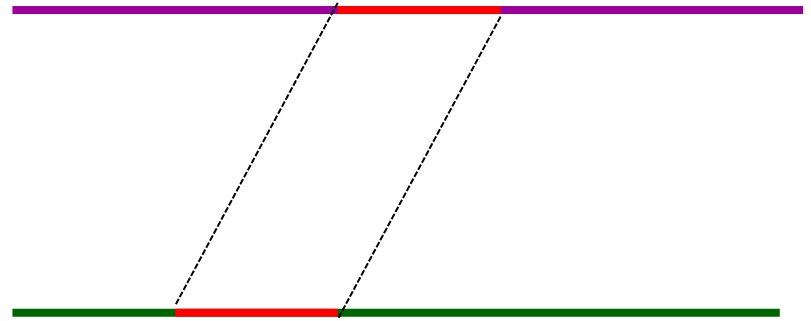
- Biết: match = 1, mismatch = 0, indel = -1

A. Thực hiện Global Sequence Alignment

B. Thực hiện giải thuật trên với giới hạn indels = 2

# Local alignment

## Liên kết cục bộ



- Cho hai đoạn DNA có kích thước lớn, cả hai đều chứa cùng một gene hoặc các gene gần nhau.
- Xác định gene?
- Liên kết cục bộ:
  - Cho hai chuỗi  $S[1..n]$  và  $T[1..m]$ ,
  - Tìm chuỗi con A của S và chuỗi con B của T có liên kết toàn cục với điểm cao nhất

# Brute-force solution

- Thuật toán:

Với mọi chuỗi con  $A = S[i'..i]$  của  $S$ ,

Với mọi chuỗi con  $B = T[j'..j]$  của  $T$ ,

Tính toán điểm liên kết toàn cục của  $A$  và  $B$

Trả lại cặp  $(A, B)$  có điểm cao nhất

- Độ phức tạp thời gian tính toán:

- Có  $n^2/2$  cách chọn  $A$  và  $m^2/2$  cách chọn  $B$ .

- Căn chỉnh tổng thể của  $A$  và  $B$  có thể được tính theo thời gian  $O(nm)$ .

- Tổng cộng, độ phức tạp thời gian =  $O(n^3m^3)$

- Giải pháp tốt hơn: giải thuật Smith-Waterman

# Tiền tố và hậu tố của chuỗi

- $X$  là hậu tố của  $S[1..n]$  nếu  $X = S[k..n]$  với  $1 \leq k \leq n$
- $Y$  là tiền tố của  $S[1..n]$  nếu  $Y = S[1..k]$  đối với  $1 \leq k \leq n$
- Ví dụ.
  - $S[1..7] = \text{ACCGATT}$
  - ACC là tiền tố của  $S$ , GATT là hậu tố của  $S$
  - Chuỗi rỗng mặc định là tiền tố và hậu tố của  $S$

# Nhận xét

- Tất cả các hậu tố của  $S_i = S[1..i]$  : tất cả các chuỗi con trong  $S$  kết thúc bằng  $i$
- Tất cả các chuỗi con của  $S$ : {tất cả các hậu tố của  $S[1..i] \mid i = 1, 2, \dots, n$ }
- Ví dụ với  $S[1..7] = \text{ACCGATT}$

# Dynamic programming for local alignment problem

- Với tất cả các hậu tố  $A$  của  $S_i = S[1..i]$  và tất cả các hậu tố  $B$  của  $T_j = T[1..j]$ , xác định điểm liên kết toàn cục của  $A$  và  $B$
- $V(i, j)$  là điểm tối đa của liên kết toàn cục giữa các cặp xâu con  $A$  (hậu tố của  $S_i$ ) và xâu con  $B$  (hậu tố của  $T_j$ )
- Sau đó, điểm tối ưu cần tìm của sự liên kết cục bộ là:

$$\max_{i, j} V(i, j)$$



# Giải thuật Smith-Waterman

- Khởi tạo:
  - $V(i, 0) = V(0, j) = 0$
- Bước lặp với  $i > 0$  và  $j > 0$ :

$$V(i, j) = \max \begin{cases} 0 & \text{align empty strings} \\ V(i-1, j-1) + \delta(S[i], T[j]) & \text{match/mismatch} \\ V(i-1, j) + \delta(S[i], -) & \text{delete} \\ V(i, j-1) + \delta(-, T[j]) & \text{insert} \end{cases}$$

# Ví dụ

- Khởi tạo:  $V(i, 0) = V(0, j) = 0$

- Score for match = 2
- Score for insert, delete, mismatch = -1

	—	C	T	C	A	T	G	C
—	0	0	0	0	0	0	0	0
A	0							
C	0							
A	0							
A	0							
T	0							
C	0							
G	0							

# Ví dụ

- Score for match = 2
- Score for insert, delete, mismatch = -1

	—	C	T	C	A	T	G	C
—	0	0	0	0	0	0	0	0
A	0	0	0	0	2	1	0	0
C	0	2	1	2	1	1	0	2
A	0	0	1	1	4	3	2	1
A	0	0	0	0	3	3	2	1
T	0	0	2	1	2			

$$V(i, j) = \max \begin{cases} 0 & \text{align empty strings} \\ V(i-1, j-1) + \delta(S[i], T[j]) & \text{match/mismatch} \\ V(i-1, j) + \delta(S[i], -) & \text{delete} \\ V(i, j-1) + \delta(-, T[j]) & \text{insert} \end{cases}$$

# Ví dụ

C A A T C G

C \_ A T \_ G

	—	C	T	C	A	T	G	C
—	0	0	0	0	0	0	0	0
A	0	0	0	0	2	1	0	0
C	0	2	1	2	1	1	0	2
A	0	1	1	1	4	3	2	1
A	0	0	0	0	3	3	2	1
T	0	0	2	1	2	5	4	3
C	0	2	1	4	3	4	4	6
G	0	1	1	3	3	3	6	5

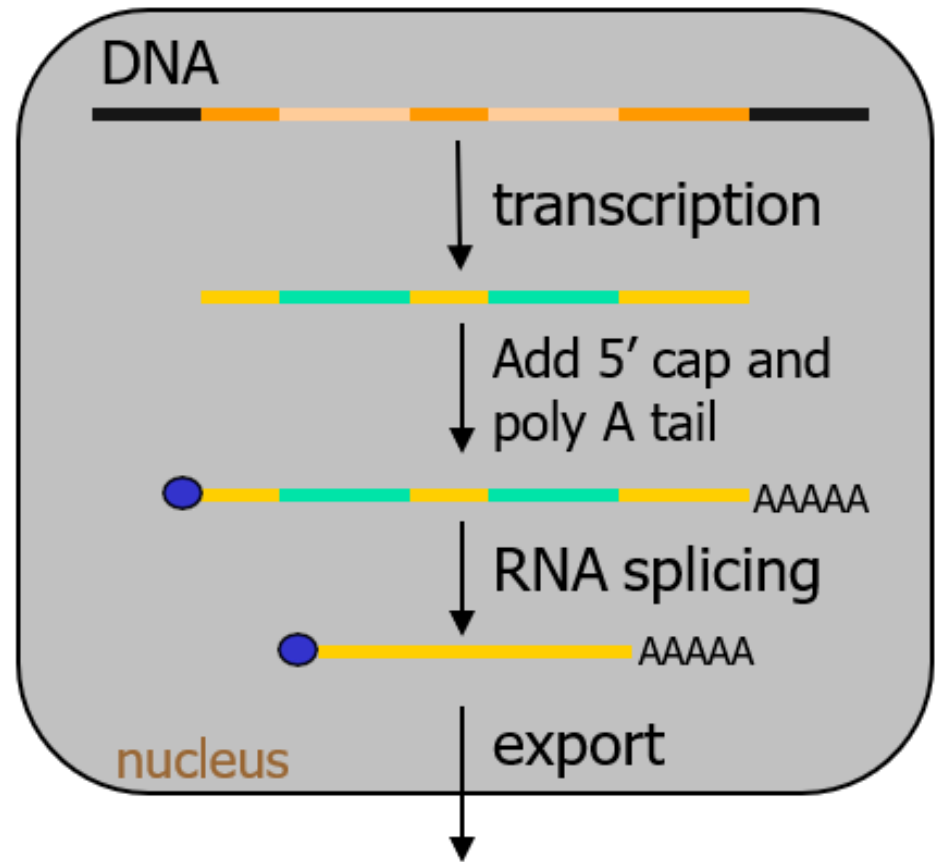
# Độ phức tạp của giải thuật

- Cần điền vào tất cả các mục trong bảng của ma trận  $n \times m$ .
- Mỗi mục có thể được tính trong thời gian  $O(1)$ .
- Cuối cùng, tìm mục có giá trị lớn nhất.
- Độ phức tạp thời gian =  $O(nm)$
- Độ phức tạp không gian =  $O(nm)$

# Semi-global alignment

## Căn chỉnh bán toàn cục

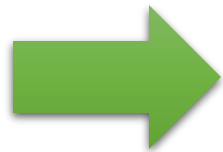
- Căn chỉnh bán toàn cục bỏ qua các ký tự khoảng trắng ở đầu và cuối chuỗi
- Ứng dụng: Xác định vùng exon trong gene



# Ví dụ

$S =$  ATCCGAACATCCAATCGAAGC

$T =$  AGCATGCAAT



ATCCGAACATCCAATCGAAGC

A---G--CATGCAAT-----

- Since the alignment has nine matches (score = 18), one mismatch (score = -1), and eleven deletions (score = -11), the score of the alignment is 6.

# Ví dụ 1: Bỏ qua khoảng trắng đầu và cuối của dãy thứ hai

ATCCGAA-CATCCAATCGAAGC  
-----AGCATGCAAT-----

- Điểm của căn chỉnh: 14
  - 8 matches (điểm = 16),
  - 1 lần xóa (điểm = -1),
  - 1 mismatch (điểm = -1)



# Semi-global alignment

-----ACCTCACGATCCGA  
TCAACGATCACCGCA-----

- Ví dụ 2: bỏ qua khoảng trắng đầu của dãy thứ nhất và dấu cách kết thúc của dãy thứ hai
  - Điểm của sự liên kết trên là 9: 5 matches (điểm = 10), 1 mismatch (điểm = -1)
  - Căn chỉnh này có thể được sử dụng để tìm vùng chung của hai trình tự chồng chéo
- Ứng dụng: trong giải trình tự gen, kết quả bao gồm rất nhiều đoạn ngắn, cần lắp ráp thành hệ gen hoàn chỉnh

# How to compute semi-global alignment?

- Sử dụng lập trình động cho căn chỉnh toàn cục với một số thay đổi nhỏ.
- The idea is that we give a **zero score** instead of a minus score to those spaces in the beginning that are not charged.
- To ignore spaces at the end, we choose the maximum value in the last row or the last column.

Spaces that are not charged	Action
Spaces in the beginning of $S[1..n]$	Initialize first row with zeros
Spaces in the ending of $S[1..n]$	Look for maximum in the last row
Spaces in the beginning of $T[1..m]$	Initialize first column with zeros
Spaces in the ending of $T[1..m]$	Look for maximum in the last column

# Bài tập 1

- Thực hiện căn chỉnh cho ví dụ 1. Bỏ qua khoảng trắng đầu và cuối của dãy thứ hai. Biết score của match = 2, mismatch và indel đều bằng -1.

ATCCGAA-CATCCAATCGAAGC  
-----AGCATGCAAT-----

# Bài tập 2

- Thực hiện căn chỉnh (alignment) cho ví dụ 2: bỏ qua khoảng trắng đầu của dãy thứ nhất và dấu cách kết thúc của dãy thứ hai. Biết score của match = 2, mismatch và indel đều bằng -1.

```
-----ACCTCACGATCCGA  
TCAACGATCACCGCA-----
```

## 2.5. Gaps

- Khoảng trống (gap) trong một liên kết là một chuỗi con tối đa các ký tự space liền kề trong hai trình tự:

This is a gap!

A \_CAACTCGCCTCC  
AGCA\_\_\_\_\_TGC

This is another gap!

# Penalty for gaps

- Giả định hiện tại: penalty (điểm có giá trị  $< 0$ ) cho việc chèn / xóa tỷ lệ thuận với độ dài của một khoảng trống

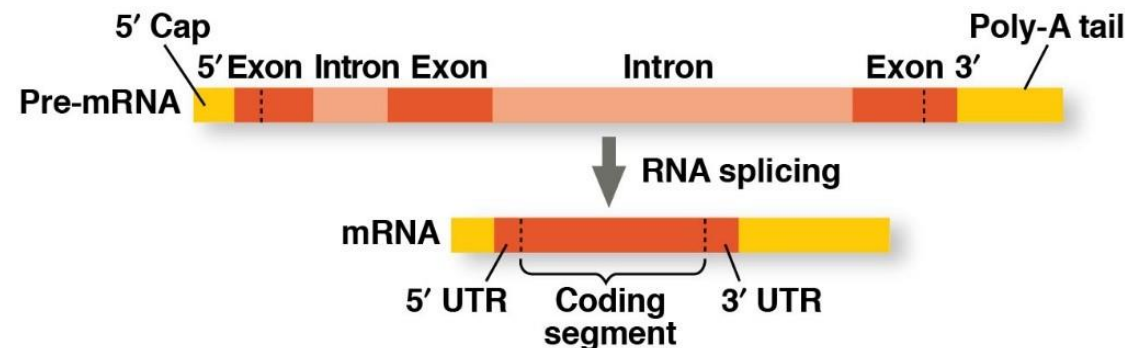
Chèn:  $\delta(\_, T[j])$

Xóa:  $\delta(S[i], \_)$

	_	A	G	C	A	T	G	C
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1							
C	-2							
A	-3							
A	-4							
T	-5							
C	-6							
C	-7							

# Penalty for gaps

- Giả định này không hợp lý trong một số ứng dụng, ví dụ:
  - Ví dụ 1: Sự đột biến có thể gây ra chèn / xóa một chuỗi con lớn hoặc chèn / xóa một cặp nucleotide duy nhất.
  - Ví dụ 2: mRNA bỏ qua phần intron trong gene, khi sắp xếp mRNA với gen của nó. Do vậy không nên tính penalty score của gaps tỷ lệ thuận với độ dài của khoảng trống.



***Cần định nghĩa  
hàm mới để tính  
score cho gap  
hợp lý hơn***

# General gap penalty

- Định nghĩa: hàm  $g(q)$  được biểu thị là penalty của gap có độ dài  $q$
- Giải thuật căn chỉnh toàn cục của  $S[1..n]$  và  $T[1..m]$ 
  - Ký hiệu  $V(i, j)$  là điểm cho sự liên kết toàn cục giữa  $S[1..i]$  và  $T[1..j]$ .



# General gap penalty

- Khởi tạo:

$$V(0, 0) = 0$$

$$V(0, j) = -g(j)$$

$$V(i, 0) = -g(i)$$

	_	A	G	C	A	T	G	C
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1							
C	-2							
A	-3							
A	-4							
T	-5							
C	-6							
C	-7							

- Bước lặp : for  $i > 0$  and  $j > 0$ ,

$$V(i, j) = \max \begin{cases} V(i-1, j-1) + \delta(S[i], T[j]) & \text{match/mismatch} \\ \max_{0 \leq k \leq j-1} \{V(i, k) - g(j-k)\} & \text{Insert } T[k+1..j] \\ \max_{0 \leq k \leq i-1} \{V(k, j) - g(i-k)\} & \text{Delete } S[k+1..i] \end{cases}$$

# Độ phức tạp

- Cần điền vào tất cả các mục trong bảng  $n \times m$ .
- Mỗi mục nhập có thể được tính trong thời gian  $O(n + m)$ .
- Độ phức tạp không gian lưu trữ =  $O(nm)$
- Độ phức tạp thời gian tính toán =  $O(n^2m + nm^2)$
- Cần cải tiến thuật toán để giảm thời gian tính toán

# Affine gap model

- Trong mô hình này, penalty cho một khoảng trống (gap) được chia thành hai phần:
  - Penalty  $h$  khi bắt đầu tạo khoảng trống
  - Penalty  $s$  theo độ dài của khoảng trống
- Xem xét một khoảng cách với  $q$  dấu cách,
  - Hàm penalty:  $g(q) = h + q.s$
  - Độ phức tạp không gian lưu trữ =  $O(nm)$
  - Độ phức tạp thời gian tính toán =  $O(nm)$

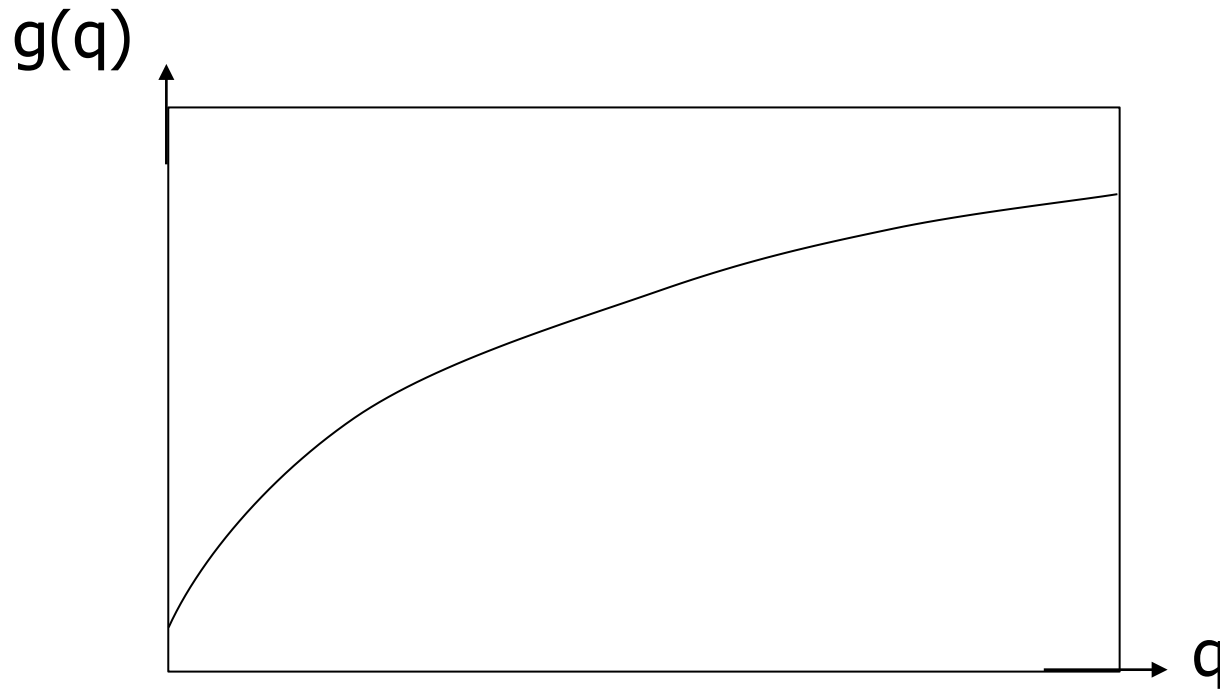
# Is affine gap penalty good?

- Affine gap penalty không đúng với một số cơ chế sinh học thực sự. Ví dụ, affine gap penalty không có lợi cho những khoảng trống dài.
- Đề xuất các other non-affine gap penalty functions.
- Tất cả các hàm này cố gắng đảm bảo:
  - Khi khoảng trống (gap) dài hơn thì penalty phát sinh cho mỗi space giảm.
  - Ví dụ: logarithmic gap penalty
$$g(q) = a \log(q) + b$$

# Convex gap penalty function

- $g(q)$  là a non-negative increasing function sao cho:

$$g(q+1) - g(q) \leq g(q) - g(q-1) \text{ for all } q \geq 1$$



- In total, all entries  $V(i, j)$  can be filled in  $O(nm \log(nm))$  time.

# Bài tập 3

- Cho hai chuỗi  $S1 = \text{ACAT}$  và  $S2 = \text{TGCAT}$
- Hãy thực hiện căn chỉnh 2 chuỗi trên với
  - A. affine penalty gap:  $g(q) = h + q.s$
  - B. logarithmic gap penalty  $g(q) = a \log(q) + b$

	—	A	C	G	T
—		-1	-1	-1	-1
A	-1	2	-1	-1	-1
C	-1	-1	2	-1	-1
G	-1	-1	-1	2	-1
T	-1	-1	-1	-1	2

## 2.6. Scoring function

- Hàm khoảng cách của cả DNA và Protein

$$\delta(C,G) = -1$$

	_	A	C	G	T
_		-1	-1	-1	-1
A	-1	2	-1	-1	-1
C	-1	-1	2	-1	-1
G	-1	-1	-1	2	-1
T	-1	-1	-1	-1	2

# Scoring function for DNA

- It simply gives a positive score for match and a negative score for mismatch.
- The NCBI-BLASTN set the match score and the mismatch score to +2 and -1, respectively.
- The WU-BLASTN and FastA set the match score and the mismatch score to +5 and -4, respectively.

	A	C	G	T
A	5	-4	-4	-4
C	-4	5	-4	-4
G	-4	-4	5	-4
T	-4	-4	-4	5

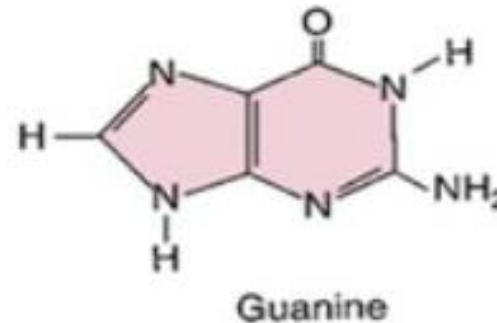
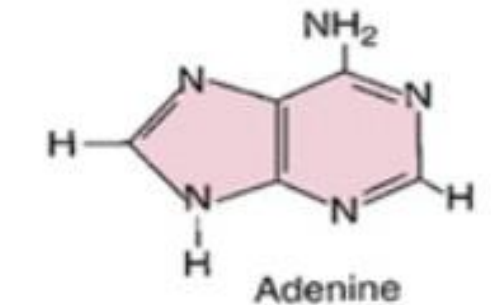


# Scoring function for DNA

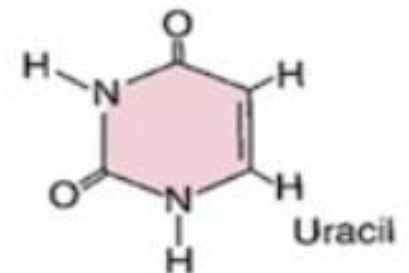
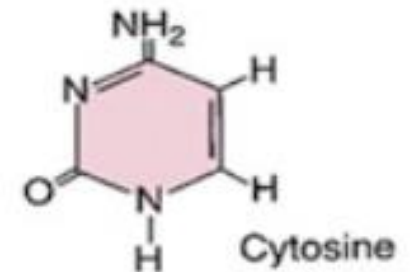
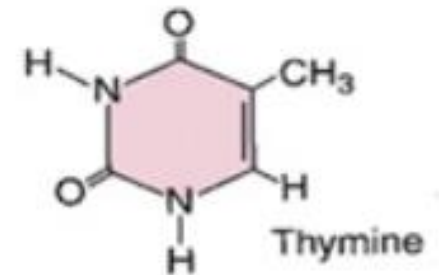
## Transition transversion matrix

- 4 nucleotides được chia thành 2 nhóm : purines (A and G) and pyrimidines (C, U and T).

Purine



Pyrimidine



# Scoring function for DNA

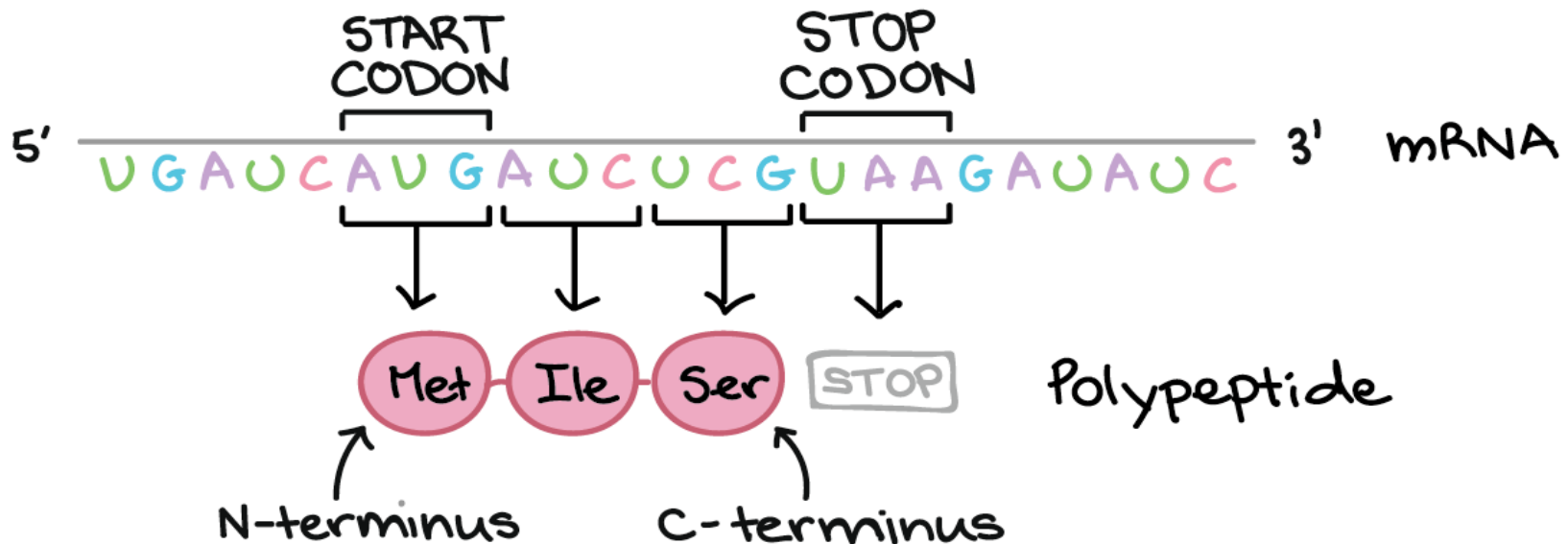
## Transition transversion matrix

- Score sẽ tốt hơn (-1) khi thay thế cùng nhóm: purine by purine hoặc pyrimidine by pyrimidine.
- Score sẽ tồi hơn (-5) khi thay thế khác nhóm : giữa purine và pyrimidine.
- Match score : +1.

	A	C	G	T
A	1	-5	-1	-5
C	-5	1	-5	-1
G	-1	-5	1	-5
T	-5	-1	-5	1

# Scoring function for Protein

- Thông thường, hàm này được tạo ra dựa trên hai tiêu chí:
  - Sự giống nhau về tính chất hóa học / vật lý của các amino acids
  - Các tần số thay thế quan sát được



# Scoring function cho protein sử dụng các đặc tính vật lý / hóa học

- Ý tưởng: *một axit amin có nhiều khả năng bị thay thế bởi một axit amin khác nếu chúng có đặc tính tương tự*
- Ma trận điểm (score matrix) có thể được suy ra dựa trên tính kỵ nước, điện tích, độ âm điện và kích thước của các amino acids
- Ví dụ: cho điểm cao hơn khi thay thế axit amin không phân cực thành một axit amin không phân cực khác

# Tóm tắt các tính chất của axit amin

Amino Acid	1-Letter	3-Letter	Avg. Mass (Da)	volume (Å <sup>3</sup> )	Side chain polarity	Side chain acidity or basicity	Hydropathy index
Alanine	A	Ala	89.09404	67	non-polar	Neutral	1.8
Cysteine	C	Cys	121.15404	86	polar	basic (strongly)	-4.5
Aspartic acid	D	Asp	133.10384	91	polar	Neutral	-3.5
Glutamic acid	E	Glu	147.13074	109	polar	acidic	-3.5
Phenylalanine	F	Phe	165.19184	135	polar	neutral	2.5
Glycine	G	Gly	75.06714	48	polar	acidic	-3.5
Histidine	H	His	155.15634	118	polar	neutral	-3.5
Isoleucine	I	Ile	131.17464	124	non-polar	neutral	-0.4
Lysine	K	Lys	146.18934	135	polar	basic (weakly)	-3.2
Leucine	L	Leu	131.17464	124	non-polar	neutral	4.5
Methionine	M	Met	149.20784	124	non-polar	neutral	3.8
Asparagine	N	Asn	132.11904	96	polar	basic	-3.9
Proline	P	Pro	115.13194	90	non-polar	neutral	1.9
Glutamine	Q	Gln	146.14594	114	non-polar	neutral	2.8
Arginine	R	Arg	174.20274	148	non-polar	neutral	-1.6
Serine	S	Ser	105.09344	73	polar	neutral	-0.8
Threonine	T	Thr	119.12034	93	polar	neutral	-0.7
Valine	V	Val	117.14784	105	non-polar	neutral	-0.9
Tryptophan	W	Trp	204.22844	163	polar	neutral	-1.3
Tyrosine	Y	Tyr	181.19124	141	non-polar	neutral	4.2

# Scoring function for protein based on statistical model

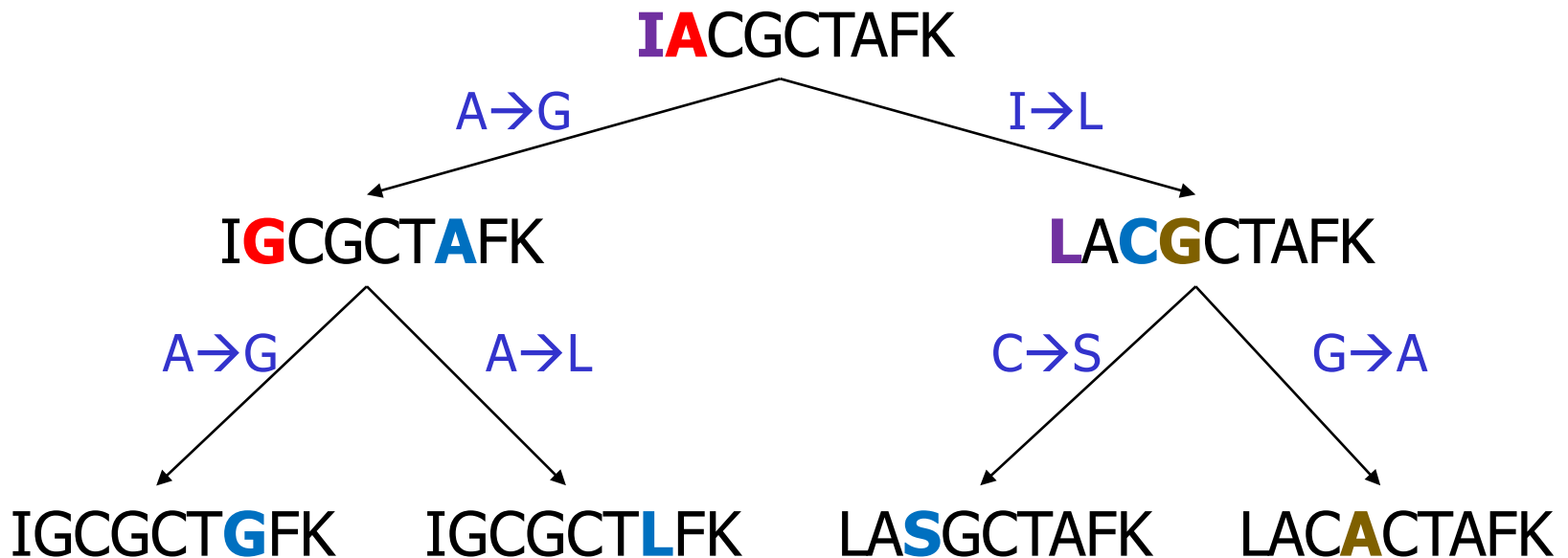
- Point Accepted Mutation (PAM)
- PAM được phát triển bởi Dayhoff (1978).
- Đột biến điểm (Point mutation): thay thế một amino acid này bằng một amino acid khác.
- Đột biến điểm được chấp nhận nếu đột biến không thay đổi chức năng của protein.
- Hai trình tự S1 và S2 được cho là 1 PAM nếu có thể chuyển đổi S1 thành S2 với trung bình 1 đột biến điểm (được chấp nhận) trên 100 amino acids

# PAM matrix by example

- Thu thập các trình tự axit amin có độ tương đồng cao (thường  $> 85\%$ )
- Thực hiện ungapped alignment.
- Ví dụ :
  - IACGCTAFK  
IGCGCTAFK  
LACGCTAFK  
IGCGCTGFK  
IGCGCTLFK  
LASGCTAFK  
LACACTAFK

# PAM matrix by example

- Xây dựng cây phát sinh loài cho các trình tự: để xác định các đột biến

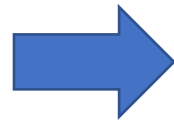




# Tính score matrix: PAM-1 matrix

- Với một cặp amino acids a và b, tính  $O_{a,b}$  và  $E_{a,b}$  với where  $O_{a,b}$  and  $E_{a,b}$  are the observed frequency and the expected frequency.
- Từ đó: 
$$\delta(a,b) = \log \frac{O_{a,b}}{E_{a,b}}$$
- $E_{a,b} = f_a \cdot f_b$  where  $f_a$  is the number of residue **a** divided by the total number of residues

IACGCTAFK  
IGCGCTAFK  
LACGCTAFK  
IGCGCTGFK  
IGCGCTLFK  
LASGCTAFK  
LACACTAFK



$$E_{A,G} = (10/63)(10/63) = 0.0252$$

# Tính score matrix: PAM-1 matrix

- Since PAM-1 assume 1 mutation per 100 residues:

$$O_{a,a} = 99/100.$$

- For  $a \neq b$ :

$$O_{a,b} = F_{a,b} / (100 \sum_{x,y \in \mathcal{A}} F_{x,y})$$

where  $F_{a,b}$  is the frequency of substituting  $a$  by  $b$  or  $b$  by  $a$ .

- Ví dụ:

- $F_{A,G} = 3, F_{A,L} = 1.$

- $O_{A,G} = 3 / (100 * 2 * 6) = 0.0025$

- $\delta(A,G) = \log (0.0025 / 0.0252) = \log (0.09925) = -1.0034$

# Ma trận PAM-2, PAM-N

- $M_{a,b}$  : xác suất a đột biến thành b.
- $M_{a,b} = O_{a,b} / f_a$
- $M^2(a,b)$  : xác suất a đột biến thành b sau 2 lần đột biến.
- $M^2(a,b) = \sum_x M(a,x)M(x,b)$
- Phần tử (a,b) của ma trận PAM-2:  
$$\log(f_a M^2(a,b) / f_a f_b) = \log(M^2(a,b) / f_b)$$

# PAM 250

[illegible]

# BLOSUM (BLOckSUBstitution Matrix)

- Henikoff (1992) đề xuất BLOSUM.
- Không giống như PAM, ma trận BLOSUM được xây dựng trực tiếp từ sự liên kết quan sát được (thay vì ngoại suy)
- BLOSUM p : ma trận được tạo ra bằng cách hợp nhất các chuỗi với độ giống nhau không ít hơn p%.
- BLOSUM 80  $\approx$  PAM 1
- BLOSUM 62  $\approx$  PAM 120
- BLOSUM 45  $\approx$  PAM 250
- BLOSUM 62 là ma trận mặc định cho BLAST 2.0

# BLOSUM 62

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

# Bài tập 1

- Calculate the score for the following alignment.

--TCATAC--TCATGAACT  
GGTAATCCCTC---AA--

- (a) Match= 1, mismatch= 0, indel= -1
- (b) Match= 1, mismatch= -1, initial gap= -2, each space= -1
- (c) Match= 0, mismatch= -1, initial gap= -2, each space= -1

# Bài tập 2

- Cho hai chuỗi

S = TCATACTCATGAACT

T = GGTAATCCCTCAA

- Biết (a) Match= 1, mismatch= 0, indel= -1
- (b) Match= 1, mismatch= -1, initial gap= -2, each space= -1
- (c) Match= 0, mismatch= -1, initial gap= -2, each space= -1
- a. Tính Global Sequence Alignment
- B. Tính Local Sequence Alignment
- C. Tính Semi Sequence Alignment



# Bài tập thực hành

- a. Hãy thực hiện các thao tác so sánh giữa 2 trình tự DNA cụ thể.
- b. Hãy thực hiện các thao tác so sánh giữa 2 trình tự Protein cụ thể.

# Bài tập tìm hiểu

- Tìm hiểu, trình bày nguyên lý và ví dụ ma trận PSSM
- Tài liệu tham khảo:  
<https://www.ncbi.nlm.nih.gov/books/NBK2590/>

# Bài thực hành với BioPython

- **Pairwise sequence alignment**
- <http://biopython.org/DIST/docs/tutorial/Tutorial.html#sec98>