



生物信息学

第六章 双序列比对 (1)

为什么要序列比对



- 基于同源序列鉴定的功能预测
- 基本假设：
序列的保守性 $\longleftrightarrow$ 功能的保守性
- 注意：
 - ✿ 蛋白质一般在三级结构的层面上执行功能
 - ✿ 蛋白质序列的保守性决定于其编码DNA的保守性

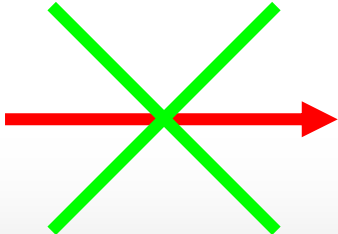
序列同源性模型中的进化假设



- 所有的生物都起源于同一个祖先
- 序列不是随机产生，而是在进化上，不断发生着演变
- 基本假设：

序列保守性  结构保守性

- 注意：反之可以不为真

结构保守性  序列保守性

同源序列：定义



- ❑ Ortholog（直系同源序列）：两个基因通过物种形成的事件而产生，或源于不同物种的最近의共同祖先的两个基因，或者两个物种中的同一基因，一般具有相同的功能
- ❑ Paralog（旁系同源序列）：两个基因在同一物种中，通过至少一次基因复制的事件而产生
- ❑ Xenolog（异同源序列）：由某一个水平基因转移事件而得到的同源序列

直系同源序列：物种形成



Human Plk1



mouse Plk1



fly POLO



Yeast Cdc5

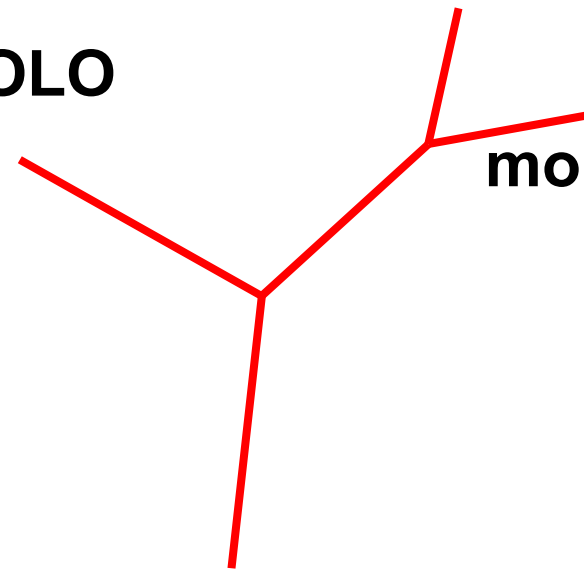
或者

Human Plk1

fly POLO

mouse Plk1

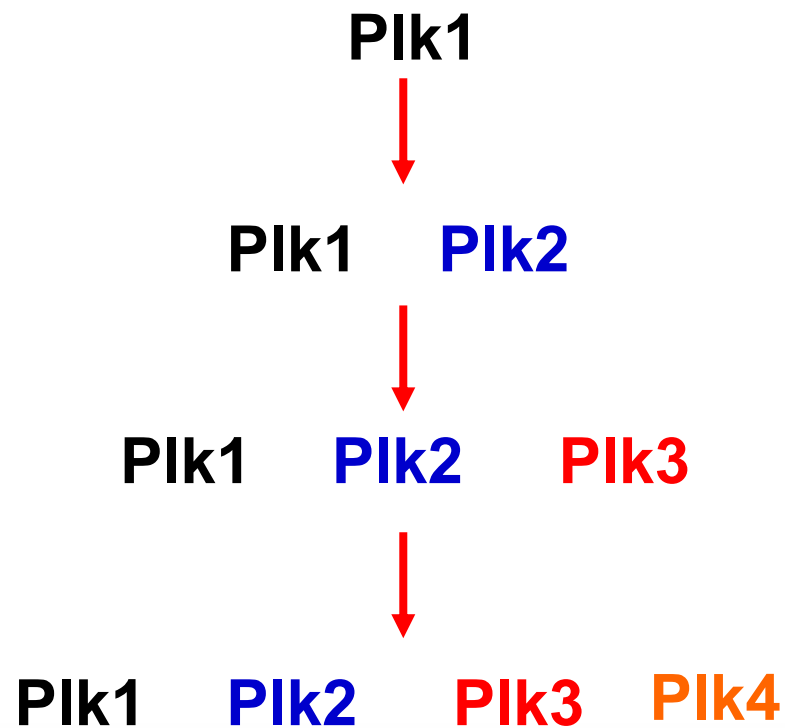
Yeast Cdc5



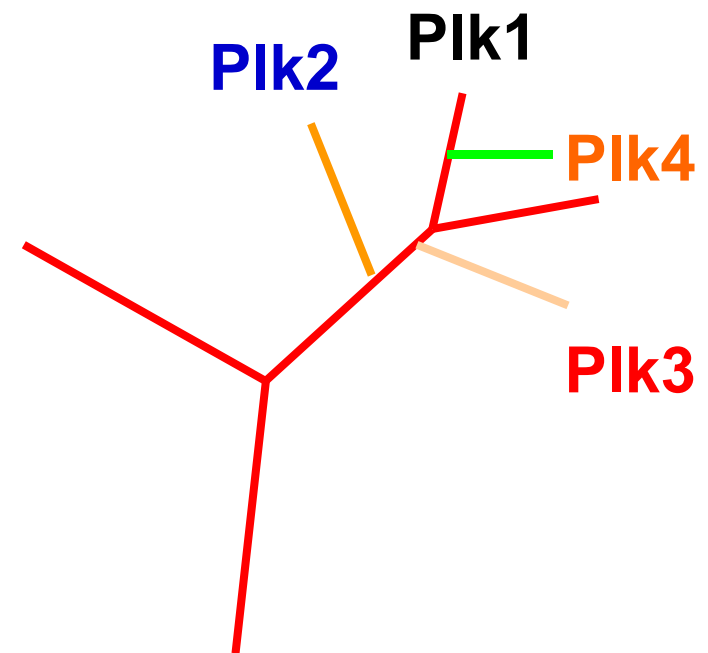
旁系同源序列：基因复制



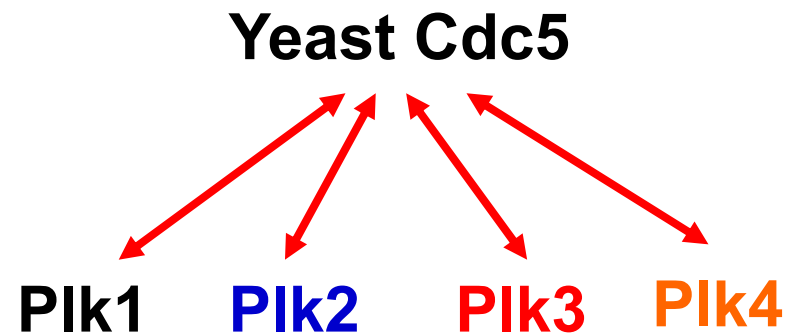
Human



或者



一个复杂的问题



□ 直系同源序列 vs. 旁系同源序列？

双序列比对的算法



- ❑ Dot Matrix, 点阵法

- ❑ 动态规划算法:

 - ✿ Global: Needleman-Wunsch

 - ✿ Local: Smith-Waterman

- ❑ Word or k -tuple算法: FASTA, BLAST

点阵法



- ❑ 1970年, Gibbs & McIntyre
- ❑ 寻找两条序列间所有可能的比对
- ❑ 发现蛋白质或者DNA序列上正向或者反向的重复
- ❑ 发现RNA上可能存在的互补区域
- ❑ 工具:
 - ✿ <https://dotlet.vital-it.ch/>
 - ✿ <https://myhits.isb-sib.ch/cgi-bin/dotlet>
 - ✿ <http://www.bioinformatics.nl/cgi-bin/emboss/dotmatcher>

点阵法



- ❑ 1970年, Gibbs & McIntyre
- ❑ 寻找两条序列间所有可能的比对
- ❑ 发现蛋白质或者DNA序列上正向或者反向的重复
- ❑ 发现RNA上可能存在的互补区域
- ❑ 工具:
 - ✿ <https://dotlet.vital-it.ch/>
 - ✿ <https://myhits.isb-sib.ch/cgi-bin/dotlet>
 - ✿ <http://www.bioinformatics.nl/cgi-bin/emboss/dotmatcher>

Dotlet JS



← → ↻ <https://dotlet.vital-it.ch/> 📄 ☆ 👤 ⋮

Dotlet JS *beta*



SEQUENCE 1

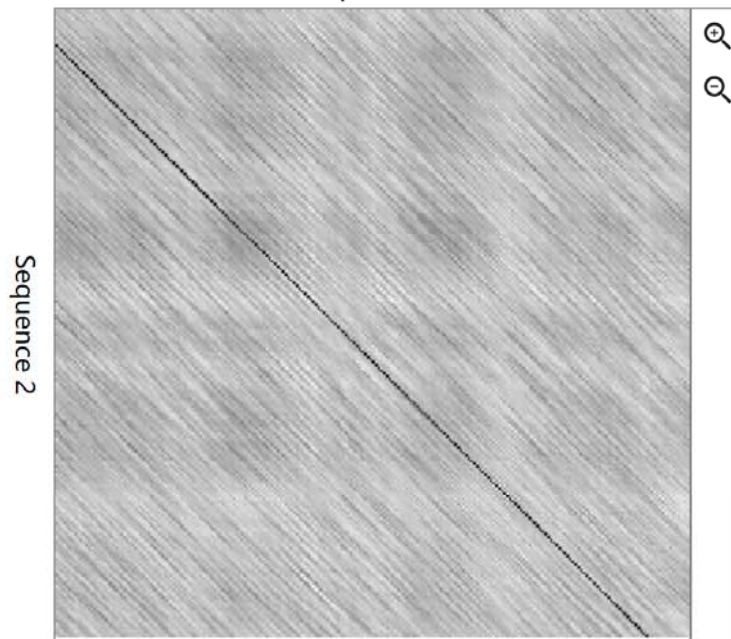
SEQUENCE 2

Window size
15

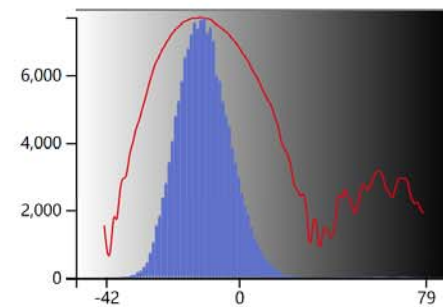
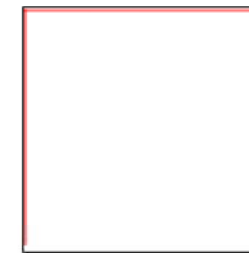
Scoring matrix
BLOSUM 62



Sequence 1



[246 x 244] # Score at (1:M, 1:R) : -8



自身的比对



	A	K	G	F	K	C	A	D	E
A	1	0	0	0	0	0	1	0	0
K		1	0	0	1	0	0	0	0
G			1	0	0	0	0	0	0
F				1	0	0	0	0	0
K					1	0	0	0	0
C						1	0	0	0
A							1	0	0
D								1	0
E									1

重复序列



	A	K	G	F	D	K	G	F	E
A	1	0	0	0	0	0	0	0	0
K		1	0	0	0	1	0	0	0
G			1	0	0	0	1	0	0
F				1	0	0	0	1	0
D					1	0	0	0	0
K		1				1	0	0	0
G			1				1	0	0
F				1				1	0
E									1

反向重复/回文



	A	U	G	C	A	C	G	U	C
A	1	0	0	0	1	0	0	0	0
U		1	0	0	0	0	0	1	0
G			1	0	0	0	1	0	0
C				1	0	1	0	0	0
A					1	0	0	0	0
C						1	0	0	1
G							1	0	0
U								1	0
C									1

不同序列的比对



	P	K	D	F	C	K	A	L	V
P	1	0	0	0	0	0	0	0	0
K		1	0	0	0	1	0	0	0
F			0	1	0	0	0	0	0
T					0	0	0	0	0
K		1				1	0	0	0
A							1	0	0
I								0	0
V								0	1

PKDFCKALV

PK-FTKAIV

动态规划算法



- 打分模型、替代矩阵以及空位罚分
- 比对算法：递归及动态规划算法
- 全局优化比对： **Needleman-Wunsch**
 - ✿ BLAST (Global Alignment),
<https://blast.ncbi.nlm.nih.gov/Blast.cgi>
- 局部优化比对： **Smith-Waterman**
 - ✿ EMBOSS Water,
https://www.ebi.ac.uk/Tools/psa/emboss_water/

好的 vs. 差的比对



□ 两条序列的相似性 -> 相似/相同的生物学功能

Good

SUMO-1	5	EAKPSTEDLGDKKEGEYIKLVIGQDSSEIHFKVKMTTHLKKLKESYCQRQGVPMNSLRF	64
		E KP G K E ++I LKV GQD S + FK+K T L KL ++YC+RQG+ M +RF	
SUMO-3	3	EEKPKE---GVKTENDHINLKVAGQDGSVVQFKIKRHTPLSKLMKAYCERQGLSMRQIRF	59
SUMO-1	65	LFEGQRIADNHTPKELGMEEEDVIEWYQEQTGG	97
		F+GQ I + TP +L ME+ED I+V+Q+QTGG	
SUMO-3	60	RFDGQPINETDTPAQLEMEDEDTIDVFQQQTGG	92

Bad

SUMO-1	1	MSDQEAKPSTEDLGDKKEGEYIKLVIGQDSSEIHFK--VKMTTHLKKLKE	49
		+S + S D+G K Y+KL +G+ + FK K+T +L LKE	
PCTK3	152	LSRMSRRASLSDIGFGKLETYVKLDKLGEPTYATVFKGRSKLTENLVALKE	202

无空位罚分的双序列比对



$$O(n)=mn=n^2$$



计算效率/计算复杂性



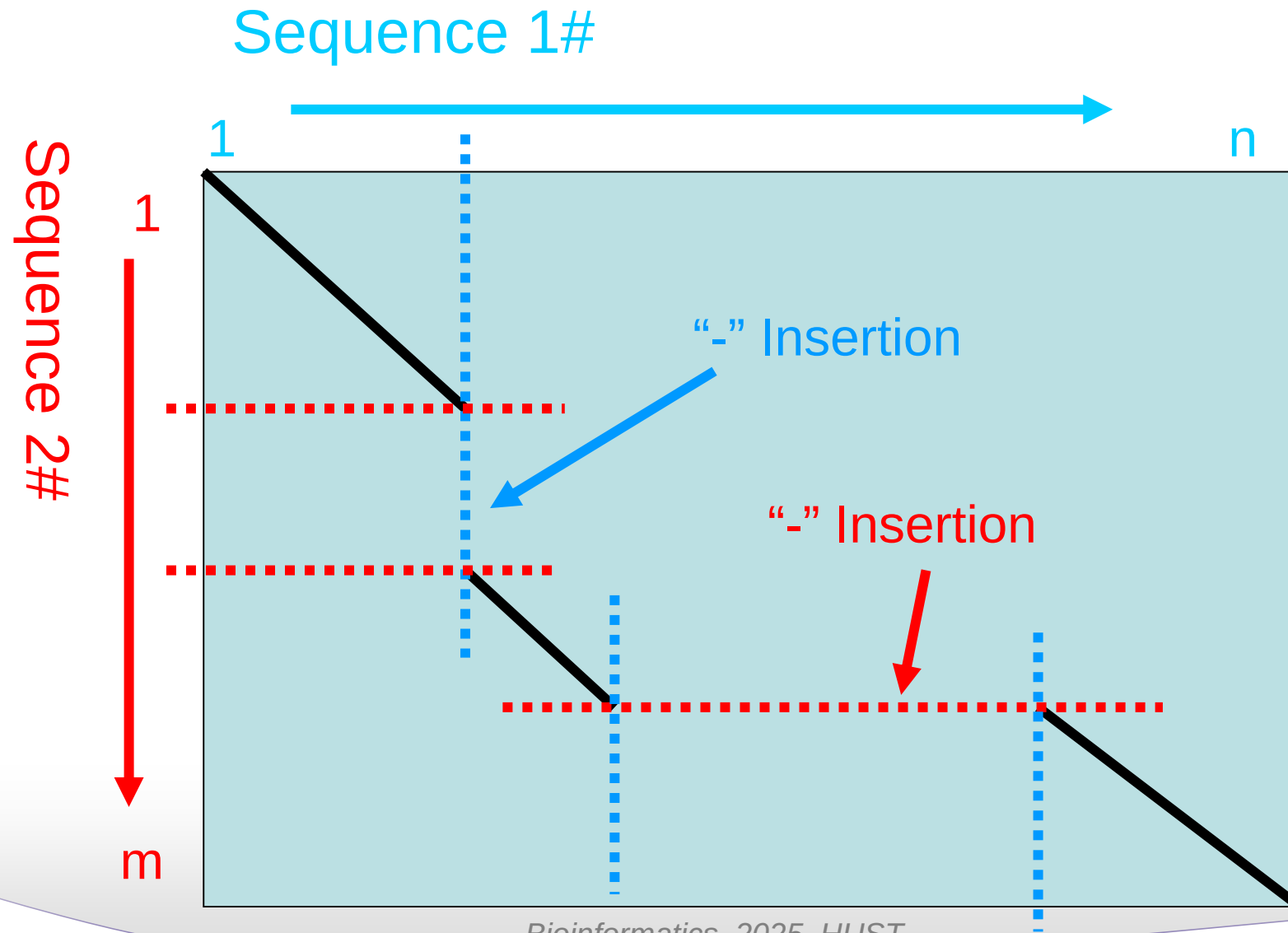
- 用CPU的计算时间和内存占用量来衡量
- $O(\) = \dots$, 时间复杂度
- 对于需要解决的问题，其单位数量 n 运算的时间是一定的 $f(n)$
- 如果需要解决的问题的大小与单位数量 n 的平方成正比，则 $O(n) = n^2$
- 对于算法来说： $O(\log n) > O(n) > O(n^2)$

NP问题



- 一般的, $O(n^k)$, 当 $k \leq 3$ 时, 为多项式时间, 较为容易处理
- 当 $O(n^k)$ = 指数级时间, 则难以处理
- **NP难题**: 无法找到能够在多项式时间复杂度内解决的问题
- 近似算法/优化算法, 求近似解

有空位罚分的双序列比对



有空位罚分的双序列比对



□ $n=1$, 3种比对

□ $n=2$, 13种比对

□ ...

□ 归纳法

□
$$a(n, n) \geq \frac{(n+2)(2n)!}{2(n!)^2}$$

□ $n=1$, $a(n, n) = 3$

□ $n=2$, $a(n, n) = 12 < 13$

A	A-	-A
B	-B	B-

AB	-AB	-AB	AB-
CD	CD-	C-D	-CD

A-B	A-B	AB--
CD-	-CD	--CD

A-B-	AB-	--AB
-C-D	C-D	CD--

-A-B	-AB-	A--B
C-D-	C--D	-CD-

有空位罚分的双序列比对



□ 两条序列比对，允许空位，时间复杂度为：

$$\square \frac{(n+2)(2n)!}{2(n!)^2} \geq \frac{\sqrt{2\pi}(n+2)e^{-2\pi}(2n)^{2n+\frac{1}{2}}}{2e^{-2n+2}n^{2n+1}} = \frac{\sqrt{\pi}(n+2)2^{2n}}{e^2\sqrt{n}}$$

□ NP-hard问题！

□ 其中，斯特林公式：

$$x! = \sqrt{2\pi x} x^{x+\frac{1}{2}} e^{-x}$$

打分模型



□ 替代矩阵

- ✿ 字符相同: **identity**

- ✿ 字符替代: **similarity**, 相似性, 氨基酸/碱基之间的替代和突变

□ 插入和缺失

□ 空位罚分

替代矩阵的模型



- 考虑长度为 n 的序列 x 和长度为 m 的序列 y
- 令 x_i 为 x 序列中的第 i 位； y_j 为 y 序列中的第 j 位
- 对于不相关或者随机的模型 R ，假设 x_i 出现的频率为 q_{xi} ， y_j 出现的频率为 q_{yj} ，则两条序列匹配的概率为：

$$P(x, y | R) = \prod_i q_{xi} \prod_j q_{yj}$$

替代矩阵的模型 (2)



- 对于另择假设/匹配模型M，两个字符匹配的概率为连接概率 p_{ab} ，因此：

$$P(x, y | M) = \prod_i P_{xiyi}$$

- 两个似然性值之间的比值称为几率值(odds ratio):

$$\frac{P(x, y | M)}{P(x, y | R)} = \frac{\prod_i p_{xiyi}}{\prod_i q_{xi} \prod_i q_{yi}} = \prod_i \frac{p_{xiyi}}{q_{xi} q_{yi}}$$

替代矩阵的模型 (3)



- 连乘->连加；取对数，求对数几率值(log-odds ratio):

$$S = \sum_i s(x_i, y_i)$$

- 并且

$$s(a, b) = \log\left(\frac{P_{ab}}{q_a q_b}\right)$$

变连乘为连加

空位罚分



- 线性罚分：d, 每次罚分的分数；g, 空位数

$$r(g) = -gd$$

- 修正的罚分：d, 第一次罚分的分数；g, 空位数；e, 修正后的参数

$$r(g) = -d - (g - 1)e$$

递归和动态规划算法 (2)



- ❑ 有空位的双序列比对，时间复杂度为： $O(2^{2n})$ ，指数增加，无法求最优解
- ❑ 动态规划算法：比较所有可能的字符对，考虑匹配、错配以及空位罚分，并且将比对次数控制在多项式时间内
- ❑ 替代矩阵：BLOSUM62，
 - ✿ 空位罚分：11
 - ✿ 延伸的空位罚分：1 (BLAST工具)

例：双序列比对



- 序列1: **V D S - C Y**
- 序列2: **V E S L C Y**
- 替代矩阵中的分数: **4 2 4 -11 9 7**
- 两序列比对的总分:
- $\text{Score} = \Sigma(\text{AA pair scores}) - \text{gap penalty} = 15$

BLOSUM62替代矩阵



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
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W

动态规划算法：全局比对



	Gap	V	D	S	C	Y
Gap	0	1gap	2gap	...		
V	1gap					
E	2gap					
S	...					
L						
C						
Y						

本例：线性罚分

$$r(g) = -gd$$



全局比对 (2)

	Gap	V	D	S	C	Y
Gap	0	-11	-22	-33	-44	-55
V	-11	S_{ij}				
E	-22					
S	-33					
L	-44					
C	-55					
Y	-66					

要求解 S_{ij} 的分数，我们必须先知道 $S_{i-1,j-1}$ ， $S_{i-1,j}$ ，以及 $S_{i,j-1}$ 的分数，这种方法叫做递归算法；采用这种方法，可以把大的问题分割成小的问题逐一解决，即动态规划算法；需要存储如何得到 S_{ij} 分数的过程。

全局比对 (3)



	Gap	V	D	S	C	Y
Gap	0	-11	-22	-33	-44	-55
V	-11	S_{ij}				
E	-22					
S	-33					
L	-44					
C	-55					
Y	-66					

Needleman-Wunsch算法;

时间复杂度 $O(n^2)$;

$$S_{ij} = \max \begin{cases} S_{i-1,j-1} + \sigma(x_i, y_j) \\ S_{i-1,j} - d \text{ (从左到右)} \\ S_{i,j-1} - d \text{ (从上到下)} \end{cases}$$

全局比对 (4)



	Gap	V	D	S	C	Y
Gap	0	-11	-22	-33	-44	-55
V	-11	S_{ij}				
E	-22					
S	-33					
L	-44					
C	-55					
Y	-66					

Needleman-Wunsch算法;

时间复杂度 $O(n^2)$;

$$S_{ij} = \max \begin{cases} S_{i-1,j-1} + \sigma(x_i, y_j) \\ S_{i-1,j} - d \text{ (从左到右)} \\ S_{i,j-1} - d \text{ (从上到下)} \end{cases}$$

全局比对 (6)



	Gap	V	D	S	C	Y
Gap	0	-11	-22	-33	-44	-55
V	-11	4	S_{ij}			
E	-22					
S	-33					
L	-44					
C	-55					
Y	-66					

Needleman-Wunsch算法;

时间复杂度 $O(n^2)$;

$$S_{ij} = \max \begin{cases} S_{i-1,j-1} + \sigma(x_i, y_j) \\ S_{i-1,j} - d \text{ (从左到右)} \\ S_{i,j-1} - d \text{ (从上到下)} \end{cases}$$

全局比对 (7)



	Gap	V	D	S	C	Y
Gap	0	-11	-22	-33	-44	-55
V	-11	4	-7			
E	-22					
S	-33					
L	-44					
C	-55					
Y	-66					

全局比对 (8)



	Gap	V	D	S	C	Y
Gap	0	-11	-22	-33	-44	-55
V	-11	4	-7	-18	-29	-40
E	-22	-7	6	-5	-16	-27
S	-33	-18	-5	10	-1	-12
L	-44	-29	-16	-1	9	-3
C	-55	-40	-27	-12	8	7
Y	-66	-51	-38	-23	-3	15

回溯：比对结果



	Gap	V	D	S	C	Y
Gap	0	11	-22	-33	-44	-55
V	-11	4	-7	-18	-29	-40
E	-22	-7	6	-5	-16	-27
S	-33	-18	-5	10	-1	-12
L	-44	-29	-16	-1	9	-3
C	-55	-40	-27	-12	8	7
Y	-66	-51	-38	-23	-3	15

Diagram illustrating sequence alignment results with a dynamic programming table. The table shows scores for alignments between sequences (Gap, V, E, S, L, C, Y) and a reference sequence (Gap, V, D, S, C, Y). The optimal alignment path is highlighted with red arrows, starting from (0,0) and ending at (Y, Y). Blue dashed lines indicate other possible paths. A green arrow points to the cell (S, S) with a score of 10.

比对结果:

V D S - C Y
V E S L C Y



	Gap	V	D	S	C	Y
Gap	0	11	-22	-33	-44	-55
V	-11	4	-7	-18	-29	-40
E	-22	-7	6	-5	-16	-27
S	-33	-18	-5	10	-1	-12
L	-44	-29	-16	-1	9	-3
C	-55	-40	-27	-12	8	7
Y	-66	-51	-38	-23	-3	15

局部优化比对



□ 下例：局部优化打分

□ 两条序列如下：

L	D	S	-	C	H
G	E	S	L	C	K

□ 目标：使用局部优化算法寻找比对的结果

局部优化比对 (2)



- Smith-Waterman算法
- 时间复杂度 $O(n^2)$
- $S_{ij} = \max \begin{cases} 0 \\ S_{i-1,j-1} + \sigma(x_i, y_j) \\ S_{i-1,j} - A \text{ (从左到右)} \\ S_{i,j-1} - A \text{ (从上到下)} \end{cases}$
- 本例中: gap: 12, 线性罚分模型

局部优化比对 (3)



	Gap	L	D	S	C	H
Gap	0	0	0	0	0	0
G	0	S_{ij}				
E	0					
S	0					
L	0					
C	0					
K	0					

Smith-Waterman算法;

$$S_{ij} = \max \begin{cases} S_{i-1,j-1} + \sigma(x_i, y_j) \\ S_{i-1,j} - d \text{ (从左到右)} \\ S_{i,j-1} - d \text{ (从上到下)} \\ 0 \end{cases}$$

局部优化比对 (5)



	Gap	L	D	S	C	H
Gap	0	0	0	0	0	0
G	0	0	0			
E	0					
S	0					
L	0					
C	0					
K	0					

Diagram illustrating a local alignment step in a dynamic programming table. Red arrows indicate the path of the alignment:

- A red arrow points from the cell (Gap, L) to the cell (G, D) with a label -1 .
- A red arrow points from the cell (G, L) to the cell (G, D) with a label -12 .
- A red arrow points from the cell (G, D) to the cell (G, D) with a label -12 .

局部优化比对 (6)



	Gap	L	D	S	C	H
Gap	0	0	0	0	0	0
G	0	0	0	0	0	0
E	0	0	2	0	0	0
S	0	0	2	6	0	0
L	0	4	0	0	5	0
C	0	0	1	0	9	2
K	0	0	0	0	0	8

Diagram illustrating a local optimization step in sequence alignment. Red arrows indicate the path of the alignment from the cell (S, S) to (L, L) and then to (C, C). The values -2 and -12 are shown next to the arrows, indicating the change in the alignment score at each step.

局部优化比对 (7)



	Gap	L	D	S	C	H
Gap	0	0	0	0	0	0
G	0	0	0	0	0	0
E	0	0	2	0	0	0
S	0	0	2	6	0	0
L	0	4	0	0	5	0
C	0	0	1	0	9	2
K	0	0	0	0	0	8

比对结果:

L D S - C H
G E S L C K



	Gap	L	D	S	C	H
Gap	0	0	0	0	0	0
G	0	0	0	0	0	0
E	0	0	2	0	0	0
S	0	0	2	6	0	0
L	0	4	0	0	5	0
C	0	0	1	0	9	2
K	0	0	0	0	0	8

打分有何不同？



L	D	S	-	C	H
G	E	S	L	C	K

- Smith-waterman算法打分：9分
- 直接打分： $-4+2+4-12+9-1=-2$
- 为何不同？

Needleman-Wunsch算法



❑ <https://blast.ncbi.nlm.nih.gov/Blast.cgi>

Specialized searches

SmartBLAST

Find proteins highly similar to your query

Primer-BLAST

Design primers specific to your PCR template

Global Align

Compare two sequences across their entire span (Needleman-Wunsch)

CD-search

Find conserved domains in your sequence

IgBLAST

Search immunoglobulins and T cell receptor sequences

VecScreen

Search sequences for vector contamination

CDART

Find sequences with similar conserved domain architecture

Targeted Loci

Search markers for phylogenetic analysis

Multiple Alignment

Align sequences using domain and protein constraints

MOLE-BLAST

Establish taxonomy for uncultured or environmental sequences

Global Alignment



U.S. National Library of Medicine

NCBI National Center for Biotechnology Information

[Sign in to NCBI](#)

BLAST® >> Global Alignment

[Home](#)

[Recent Results](#)

[Saved Strategies](#)

Needleman-Wunsch Global Align Nucleotide Sequences

Nucleotide

[Protein](#)

Enter Query Sequence

Needleman-Wunsch alignment of two nucleotide sequences ⓘ

[Reset page](#) [Bookmark](#)

Enter accession number, gi, or FASTA sequence ⓘ

[Clear](#)

Query subrange ⓘ

From

To

Or, upload file

未选择任何文件 ⓘ

Job Title

Enter a descriptive title for your BLAST search ⓘ

**BLAST results will be displayed
in a new format by default**

You can always switch back to the
Traditional Results page.



Enter Subject Sequence

Enter accession number, gi, or FASTA sequence ⓘ

[Clear](#)

Subject subrange ⓘ

From

To

Or, upload file

未选择任何文件 ⓘ

☐ Show results in a new window

[+ Algorithm parameters](#)

Smith-Waterman算法



❏ https://www.ebi.ac.uk/Tools/psa/emboss_water/

EMBOSS Water

[Input form](#)[Web services](#)[Help & Documentation](#)[Bioinformatics Tools FAQ](#)[Feedback](#)[Share](#)

Tools > Pairwise Sequence Alignment > EMBOSS Water

Pairwise Sequence Alignment

EMBOSS Water uses the Smith-Waterman algorithm (modified for speed enhancements) to calculate the local alignment of two sequences.

STEP 1 - Enter your protein sequences

Enter a pair of

PROTEIN

sequences. Enter or paste your first **protein** sequence in any supported format:

Or, upload a file: 未选择任何文件

[Use a example sequence](#) | [Clear sequence](#) | [See more example inputs](#)

AND

Enter or paste your second **protein** sequence in any supported format: