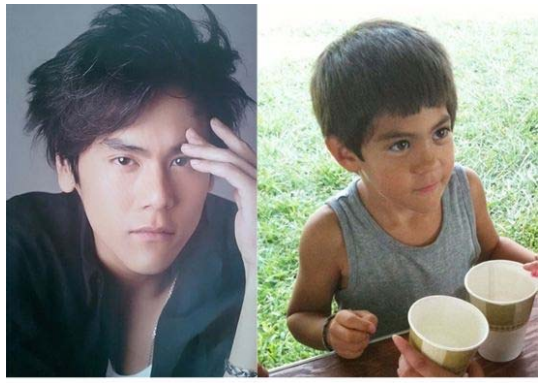
序列相似性

- 序列相似性的重要性

相似的序列往往起源于一个共同的祖先序列。它们很可能有相似的空间结构和生物学功能，因此对于一个已知序列但未知结构和功能的蛋白质，如果与它序列相似的某些蛋白质的结构和功能已知，则可以推测这个未知结构和功能的蛋白质的结构和功能。

结构相似？功能相似？


第二节

比对算法概要

Section 2 Alignment Algorithms

人民卫生出版社
PEOPLE'S MEDICAL PUBLISHING HOUSE

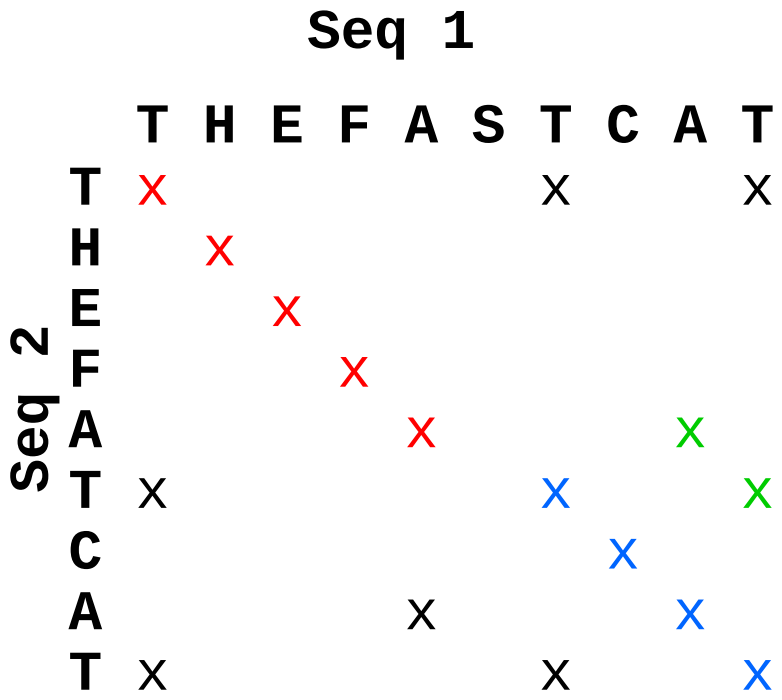
打点法的用途

1. THEFA
2. TCAT
3. AT

连续的对角线及对角线的平行线代表两条序列中相同的区域

Seq1:
THEFASTCAT

Seq2:
THEFATCAT



二、双序列全局比对

序列两两比较：序列比对法

比较两个长度不同的序列的方法：打点法、序列比对法

序列比对（alignment），也叫对位排列、联配、对齐等。运用特定的算法找出两个或多个序列之间产生最大相似度得分的空格插入和序列排列方案。

序列s和t的比对：把s和t这两个字符串上下排列起来，在某些位置插入空格（空位，gap），然后依次比较它们在每一个位置上字符的匹配情况，从而找出使这两条序列产生最大相似度得分的排列方式和空格插入方式。

序列s：LQRHKRTHTGEKPYE-CNQCGKAFAQ-
序列t：LQRHKRTHTGEKPYMNVINMVKPLHNS

多序列比对

```
SRNICYDAFVSYSERD---
-GENIYDAFVIYSSQD---
SQTF-YDAYISYDIKDASV
PDCC-YDAFIVYDIKDPAV
EDALPYDAFVVFDKIQSAV
TEQFEYAAYIIHAYKD---
PDMYKYDAYLCFSSKD---
    : *:: .   :
```

双序列比对

```
LQRHKRTHTGEKPYE-CNQCGKAFAQ-
LQRHKRTHTGEKPYMNVINMVKPLHNS
**************     :  *.: :
```

全局比对

```
LQRHKRTHTGEKPYE-CNQCGKAFAQ
    KRTHT
    *****
```

局部比对

双序列全局比对及算法

Needleman-Wunsch算法，1970年，Saul Needleman和Christian Wunsch两人首先将动态规划算法应用于两条序列的全局比对，这个算法后称为Needleman-Wunsch算法。

对于：

序列p：ACGTC
序列q：AATC

m=length(p)
n=length(q)

gap = -5

	A	G	C	T	-
A	10	-1	-3	-4	
G	-1	7	-5	-3	-5
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$
\begin{aligned}
&s(0,0) = 0\\
&s(0,j) = gap * j,\ 1<=j<=m\\
&s(i,0) = gap * i,\ 1<=i<=n\\
&s(i,j) = \max\begin{cases} s(i-1,j-1) + w(i,j)\\ s(i-1,j) + gap\\ s(i,j-1) + gap \end{cases}
\end{aligned}
$$

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

		0	1	2	3	4	5
			A	C	G	T	C
0							
1	A						
2	A						
3	T						
序列q 4	C						

序列p

得分矩阵

双序列全局比对及算法

Needleman-Wunsch算法，1970年，Saul Needleman和Christian Wunsch两人首先将动态规划算法应用于两条序列的全局比对，这个算法后称为Needleman-Wunsch算法。

对于：

序列p：**ACGTC**
序列q：**AATC**

```
m=length(p)
n=length(q)

gap = -5
```

	A	G	C	T	-
A	10	-1	-3	-4	-5
G	-1	7	-5	-3	
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

$$
\begin{aligned}
&s(0,0) = 0\\
&s(0,j) = gap * j,\ 1<=j<=m\\
&s(i,0) = gap * i,\ 1<=i<=n\\
&s(i,j) = \max\begin{cases} s(i-1,j-1) + w(i,j)\\ s(i-1,j) + gap\\ s(i,j-1) + gap \end{cases}
\end{aligned}
$$

		0	1 A	2 C	3 G	4 T	5 C
		0					
1	A						
2	A						
3	T						
序列q 4	C						

序列p

得分矩阵

双序列全局比对及算法

Needleman-Wunsch算法，1970年，Saul Needleman和Christian Wunsch两人首先将动态规划算法应用于两条序列的全局比对，这个算法后称为Needleman-Wunsch算法。

对于：

序列p：**ACGTC**
序列q：**AATC**

```
m=length(p)
n=length(q)

gap = -5
```

s(i,j)是按照替换记分矩阵得到的前缀**q[1…i]**与**p[1…j]**最大相似性的得分。

w(i,j)是字符**q[i]**和**p[j]**按照替换记分矩阵计算的得分

	A	G	C	T	-
A	10	-1	-3	-4	-5
G	-1	7	-5	-3	
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$
\begin{aligned}
&s(0,0) = 0\\
&s(0,j) = gap * j,\ 1<=j<=m\\
&s(i,0) = gap * i,\ 1<=i<=n\\
&s(i,j) = \max\begin{cases} s(i-1,j-1) + w(i,j)\\ s(i-1,j) + gap\\ s(i,j-1) + gap \end{cases}
\end{aligned}
$$

		0	1	2	3	4	5
			A	C	G	T	C
0		0	-5	-10	-15	-20	-25
1	A						
2	A						
3	T						
序列 q	4 C						

序列 p

得分矩阵

双序列全局比对及算法

Needleman-Wunsch算法
人首先将动态规划算法
Needleman-Wunsch算法

$$s(1,1) = \max \begin{cases} s(0,0) + w(1,1) = 0 + 10 = 10 \\ s(0,1) + gap = -5 + -5 = -10 \\ s(1,0) + gap = -5 + -5 = -10 \end{cases}$$

对于：

序列p：ACGTC
序列q：AATC

m=length(p)
n=length(q)

gap = -5

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

	A	G	C	T	-
A	10	-1	-3	-4	-5
G	-1	7	-5	-3	
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$s(0,0) = 0$$
$$s(0,j) = gap * j,\ 1<=j<=m$$
$$s(i,0) = gap * i,\ 1<=i<=n$$
$$s(i,j) = \max \begin{cases} s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}$$

		0	1 A	2 C	3 G	4 T	5 C
		0	-5	-10	-15	-20	-25
1	A	-5	10				
2	A	-10					
3	T	-15					
4	C	-20					

序列 p　　序列 q

得分矩阵

双序列全局比对及算法

Needleman-Wunsch算法
人首先将动态规划算法
Needleman-Wunsch算法

$$s(1,2) = \max\begin{cases} s(0,1) + w(1,2) = -5 + -3 = -8 \\ s(0,2) + gap = -10 + -5 = -15 \\ s(1,1) + gap = 10 + -5 = 5 \end{cases}$$

对于：

序列p：ACGTC
序列q：AATC

m=length(p)
n=length(q)

gap = -5

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

	A	G	C	T	-
A	10	-1	-3	-4	
G	-1	7	-5	-3	
C	-3	-5	9	0	-5
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$s(0,0) = 0$$
$$s(0,j) = gap * j,\ 1<=j<=m$$
$$s(i,0) = gap * i,\ 1<=i<=n$$
$$s(i,j) = \max\begin{cases} s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}$$

		0	1 A	2 C	3 G	4 T	5 C
		0	-5	-10	-15	-20	-25
1	A	-5	10	5			
2	A	-10					
3	T	-15					
序列q 4	C	-20					

序列p

得分矩阵

双序列全局比对及算法

Needleman-Wunsch算法，1970年，Saul Needleman和Christian Wunsch两人首先将动态规划算法应用于两条序列的全局比对，这个算法后称为Needleman-Wunsch算法。

对于：

序列p：ACGTC
序列q：AATC

m=length(p)
n=length(q)

gap = -5

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

	A	G	C	T	-
A	10	-1	-3	-4	-5
G	-1	7	-5	-3	
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$
\begin{aligned}
&s(0,0) = 0\\
&s(0,j) = gap * j,\ 1<=j<=m\\
&s(i,0) = gap * i,\ 1<=i<=n\\
&s(i,j) = \max\begin{cases} s(i-1,j-1) + w(i,j)\\ s(i-1,j) + gap\\ s(i,j-1) + gap \end{cases}
\end{aligned}
$$

		0	1	2	3	4	5
			A	C	G	T	C
0		0	-5	-10	-15	-20	-25
1	A	-5	10	5	0	-5	-10
2	A	-10	?				
3	T	-15					
4	C	-20					

序列 p　　序列 q

得分矩阵

双序列全局比对及算法

Needleman-Wunsch算法，1970年，Saul Needleman和Christian Wunsch两人首先将动态规划算法应用于两条序列的全局比对，这个算法后称为Needleman-Wunsch算法。

对于：

序列p：**ACGTC**
序列q：**AATC**

```
m=length(p)
n=length(q)
```

```
gap = -5
```

s(i,j)是按照替换记分矩阵得到的前缀**q[1…i]**与**p[1…j]**最大相似性的得分。

w(i,j)是字符**q[i]**和**p[j]**按照替换记分矩阵计算的得分

	A	G	C	T	-
A	10	-1	-3	-4	-5
G	-1	7	-5	-3	
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$
\begin{aligned}
&s(0,0) = 0\\
&s(0,j) = gap * j,\ 1<=j<=m\\
&s(i,0) = gap * i,\ 1<=i<=n\\
&s(i,j) = \max\begin{cases} s(i-1,j-1) + w(i,j)\\ s(i-1,j) + gap\\ s(i,j-1) + gap \end{cases}
\end{aligned}
$$

		0	1 A	2 C	3 G	4 T	5 C
0		0	-5	-10	-15	-20	-25
1	A	-5	10	5	0	-5	-10
2	A	-10	5				
3	T	-15					
序列 q 4	C	-20					

序列 p

得分矩阵

双序列全局比对及算法

Needleman-Wunsch算法，1970年，Saul Needleman和Christian Wunsch两人首先将动态规划算法应用于两条序列的全局比对，这个算法后称为Needleman-Wunsch算法。

对于：

序列p：**ACGTC**
序列q：**AATC**

```
m=length(p)
n=length(q)
```

```
gap = -5
```

s(i,j)是按照替换记分矩阵得到的前缀**q[1…i]**与**p[1…j]**最大相似性的得分。

w(i,j)是字符**q[i]**和**p[j]**按照替换记分矩阵计算的得分

	A	G	C	T	-
A	10	-1	-3	-4	-5
G	-1	7	-5	-3	
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$s(0,0) = 0$$

$$s(0,j) = gap * j,\ 1<=j<=m$$

$$s(i,0) = gap * i,\ 1<=i<=n$$

$$s(i,j) = \max \begin{cases} s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}$$

		0	1	2	3	4	5
			A	C	G	T	C
0		0	-5	-10	-15	-20	-25
1	A	-5	10	5	0	-5	-10
2	A	-10	5	7	4	-1	-6
3	T	-15	0	5	4	12	7
序列 q 4	C	-20	-5	9	4	7	21

序列 p

得分矩阵

人民卫生出版社 PEOPLE'S MEDICAL PUBLISHING HOUSE

双序列全局比对及算法

Needleman-Wunsch算法，1970年，Saul Needleman和Christian Wunsch两人首先将动态规划算法应用于两条序列的全局比对，这个算法后称为Needleman-Wunsch算法。

对于：

序列p：**ACGTC**
序列q：**AATC**

```
m=length(p)
n=length(q)

gap = -5
```

s(i,j)是按照替换记分矩阵得到的前缀**q[1…i]**与**p[1…j]**最大相似性的得分。

w(i,j)是字符**q[i]**和**p[j]**按照替换记分矩阵计算的得分

	A	G	C	T	-
A	10	-1	-3	-4	-5
G	-1	7	-5	-3	
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$s(0,0) = 0$$

$$s(0,j) = gap * j,\ 1<=j<=m$$

$$s(i,0) = gap * i,\ 1<=i<=n$$

$$s(i,j) = \max \begin{cases} s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}$$

		0	1 A	2 C	3 G	4 T	5 C
0		0	-5	-10	-15	-20	-25
1	A	-5	10	5	0	-5	-10
2	A	-10	5	7	4	-1	-6
3	T	-15	0	5	4	12	7
序列q 4	C	-20	-5	9	4	7	21

序列 p

得分矩阵

双序列全局比对及算法

Needleman-Wunsch算法，1970年，Saul Needleman和Christian Wunsch两人首先将动态规划算法应用于两条序列的全局比对，这个算法后称为Needleman-Wunsch算法。

对于：

序列p：ACGTC
序列q：AATC

m=length(p)
n=length(q)

gap = -5

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

	A	G	C	T	-
A	10	-1	-3	-4	-5
G	-1	7	-5	-3	
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$
\begin{aligned}
&s(0,0) = 0\\
&s(0,j) = gap * j,\ 1<=j<=m\\
&s(i,0) = gap * i,\ 1<=i<=n\\
&s(i,j) = \max\begin{cases} s(i-1,j-1) + w(i,j)\\ s(i-1,j) + gap\\ s(i,j-1) + gap \end{cases}
\end{aligned}
$$

		0	1 A	2 C	3 G	4 T	5 C
0		0	-5	-10	-15	-20	-25
1	A	-5	10	5	0	-5	-10
2	A	-10	5	7	4	-1	-6
3	T	-15	0	5	4	12	7
序列q 4	C	-20	-5	9	4	7	21

序列 p

得分矩阵

双序列全局比对及算法

Needleman-Wunsch算法，1970年，Saul Needleman和Christian Wunsch两人首先将动态规划算法应用于两条序列的全局比对，这个算法后称为Needleman-Wunsch算法。

对于：

序列p：**ACGTC**
序列q：**AATC**

↖：字符对字符

←：字符对空位
箭头指着的序列为空位

↑：字符对空位
箭头指着的序列为空位

		0	1	2	3	4	5	序列 p
			A	C	G	T	C	
0		0	-5	-10	-15	-20	-25	
1	A	-5	10	5	0	-5	-10	
2	A	-10	5	7	4	-1	-6	
3	T	-15	0	5	4	12	7	
序列 q	4 C	-20	-5	9	4	7	21	

得分矩阵

序列p:	A	C	G	T	C
序列q:	A	-	A	T	C

全局序列比对结果

Changing the Scoring

- **Longest Common Subsequence (LCS) problem**
  - the simplest form of sequence alignment
  - allows only insertions and deletions (no mismatches).
- **In the LCS Problem, we scored 1 for matches and 0 for indels**
- **Consider penalizing indels and mismatches with negative scores**
- **Simplest *scoring schema*:**
  - ***+1* : match premium**
  - **$-\mu$: mismatch penalty**
  - **$-\sigma$: indel penalty**

三、双序列局部比对

- 处理子序列与完整序列（或短序列与长序列）比对的一般过程是：设短序列***a***和长序列***b***，它们的长度分别为L_a和L_b，比对是在***b***序列中寻找L_a长度的***a***序列的过程。

$$S(i,j)=\max\begin{cases}0 & \\ S(i-1,j-1)+w(a_i,b_j) & \text{匹配或错配}\\ S(i-1,j)+w(a_i,-) & \text{插入}\\ S(i,j-1)+w(-,b_j) & \text{缺失}\end{cases}$$

三、双序列局部比对

双序列局部比对及算法

全局比对（global alignment）：用于比较两个长度近似的序列
局部比对（local alignment）：用于比较一长一短两条序列

全局比对
序列a: ASTDTPYMNVIPPCDEEFV
序列c: -----PYINVF--------
比对得分: -46

全局比对
序列b: ATPY-ELFFV
序列c: --PYINVF--
比对得分: 8

局部比对
序列a: PYMNVI
序列c: PYINVF
比对得分: 24

对于:

序列p: ACGTC　　m=length(p)
序列q: CG　　n=length(q)
　　gap = -5

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

$$
\begin{aligned}
&s(0,0) = 0\\
&s(0,j) = 0,\ 1<=j<=m\\
&s(i,0) = 0,\ 1<=i<=n\\
&s(i,j) = \max\begin{cases} 0 \\ s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}
\end{aligned}
$$

双序列局部比对及算法

全局比对（global alignment）：用于比较两个长度近似的序列
局部比对（local alignment）：用于比较一长一短两条序列
1981年 Temple Smith 和 Michael Waterman 对局部比对进行了研究，产生了Smith-Waterman算法。

对于：

序列p: ACGTC
序列q: CG

m=length(p)
n=length(q)
gap = -5

	A	G	C	T	-
A	10	-1	-3	-4	
G	-1	7	-5	-3	-5
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$
\begin{aligned}
&s(0,0) = 0\\
&s(0,j) = 0,\ 1<=j<=m\\
&s(i,0) = 0,\ 1<=i<=n\\
&s(i,j) = \max\begin{cases} 0 \\ s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}
\end{aligned}
$$

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

		0	1	2	3	4	5	序列 p
			A	C	G	T	C	
0		0	0	0	0	0	0	
1	C	0	0					
序列 q 2	G	0						

得分矩阵

双序列局部比对及算法

全局比对（global alignment）：用于比较两个长度近似的序列
局部比对（local alignment）：用于比较一长一短两条序列
1981年 Temple Smith 和 Michael Waterman 对局部比对进行了研究，产生了Smith-Waterman算法。

对于：

序列p: ACGTC
序列q: CG

m=length(p)
n=length(q)
gap = -5

	A	G	C	T	-
A	10	-1	-3	-4	
G	-1	7	-5	-3	-5
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$s(0,0) = 0$$
$$s(0,j) = 0,\ 1<=j<=m$$
$$s(i,0) = 0,\ 1<=i<=n$$
$$s(i,j) = \max \begin{cases} 0 \\ s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}$$

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

		0	1	2	3	4	5	序列 p
			A	C	G	T	C	
0		0	0	0	0	0	0	
1	C	0	0	9				
序列 q 2	G	0						

得分矩阵

双序列局部比对及算法

全局比对（global alignment）：用于比较两个长度近似的序列
局部比对（local alignment）：用于比较一长一短两条序列
1981年 Temple Smith 和 Michael Waterman 对局部比对进行了研究，产生了Smith-Waterman算法。

对于：

序列p: ACGTC
序列q: CG

m=length(p)
n=length(q)
gap = -5

	A	G	C	T	-
A	10	-1	-3	-4	
G	-1	7	-5	-3	-5
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$s(0,0) = 0$$
$$s(0,j) = 0,\ 1<=j<=m$$
$$s(i,0) = 0,\ 1<=i<=n$$
$$s(i,j) = \max \begin{cases} 0 \\ s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}$$

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

		0	1	2	3	4	5	序列 p
			A	C	G	T	C	
0		0	0	0	0	0	0	
1	C	0	0	9	4	?		
序列 q 2	G	0						

得分矩阵

双序列局部比对及算法

全局比对（global alignment）：用于比较两个长度近似的序列
局部比对（local alignment）：用于比较一长一短两条序列
1981年 Temple Smith 和 Michael Waterman 对局部比对进行了研究，产生了Smith-Waterman算法。

对于：

序列p: ACGTC
序列q: CG

m=length(p)
n=length(q)
gap = -5

	A	G	C	T	-
A	10	-1	-3	-4	
G	-1	7	-5	-3	-5
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$s(0,0) = 0$$
$$s(0,j) = 0,\ 1<=j<=m$$
$$s(i,0) = 0,\ 1<=i<=n$$
$$s(i,j) = \max \begin{cases} 0 \\ s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}$$

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

		0	1	2	3	4	5	序列 p
			A	C	G	T	C	
0		0	0	0	0	0	0	
1	C	0	0	9	4	0		
序列 q	2 G	0						

得分矩阵

双序列局部比对及算法

全局比对（global alignment）：用于比较两个长度近似的序列
局部比对（local alignment）：用于比较一长一短两条序列
1981年 Temple Smith 和 Michael Waterman 对局部比对进行了研究，产生了Smith-Waterman算法。

对于：

序列p: ACGTC
序列q: CG

m=length(p)
n=length(q)
gap = -5

	A	G	C	T	-
A	10	-1	-3	-4	
G	-1	7	-5	-3	-5
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$s(0,0) = 0$$
$$s(0,j) = 0,\ 1<=j<=m$$
$$s(i,0) = 0,\ 1<=i<=n$$
$$s(i,j) = \max \begin{cases} 0 \\ s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}$$

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

		0	1	2	3	4	5	序列 p
			A	C	G	T	C	
0		0	0	0	0	0	0	
1	C	0	0	9	4	0	9	
序列 q	2 G	0	0	4	16	13	8	

得分矩阵

双序列局部比对及算法

全局比对（global alignment）：用于比较两个长度近似的序列
局部比对（local alignment）：用于比较一长一短两条序列
1981年 Temple Smith 和 Michael Waterman 对局部比对进行了研究，产生了Smith-Waterman算法。

对于：

序列p: ACGTC
序列q: CG

m=length(p)
n=length(q)
gap = -5

	A	G	C	T	-
A	10	-1	-3	-4	
G	-1	7	-5	-3	-5
C	-3	-5	9	0	
T	-4	-3	0	8	
-	-5				

替换记分矩阵

$$s(0,0) = 0$$
$$s(0,j) = 0,\ 1<=j<=m$$
$$s(i,0) = 0,\ 1<=i<=n$$
$$s(i,j) = \max \begin{cases} 0 \\ s(i-1,j-1) + w(i,j) \\ s(i-1,j) + gap \\ s(i,j-1) + gap \end{cases}$$

s(i,j)是按照替换记分矩阵得到的前缀q[1…i]与p[1…j]最大相似性的得分。

w(i,j)是字符q[i]和p[j]按照替换记分矩阵计算的得分

		0	1	2	3	4	5	序列 p
			A	C	G	T	C	
0		0	0	0	0	0	0	
1	C	0	0	9	4	0	9	
序列 q	2 G	0	0	4	16	13	8	

得分矩阵

双序列局部比对及算法

全局比对（global alignment）：用于比较两个长度近似的序列
局部比对（local alignment）：用于比较一长一短两条序列
1981年 Temple Smith 和 Michael Waterman 对局部比对进行了研究，产生了Smith-Waterman算法。

对于：

序列p：**ACGTC**
序列q：**CG**

↖：字符对字符

←：字符对空位
箭头指着的序列为空位

↑：字符对空位
箭头指着的序列为空位

		0	1	2	3	4	5	序列 p
			A	C	G	T	C	
0		0	0	0	0	0	0	
1	C	0	0	9	4	0	9	
序列 q	2 G	0	0	4	16	13	8	

得分矩阵

序列p: C G
序列q: C G
局部序列比对结果：**16**

序列p: A C G T C
序列q: - C G - -
全局序列比对结果：**1**

一致度和相似度的正确算法

如果两个序列长度相同：

一致度（identity）= （一致字符的个数 / 全局比对长度）×100%

相似度（similarity）= （一致及相似的字符的个数 / 全局比对长度）×100%

序列1：CVHK-LA identity = (4/7)*100% = 57%
序列2：C-HKTIA similarity = ((4+1)/7)*100% = 71%

如果两个序列长度不相同：

一致度（identity）=（一致字符的个数 / 全局比对长度）×100%

相似度（similarity）=（一致及相似的字符的个数 / 全局比对长度）×100%

序列1：CVHKAT identity = (4/6)*100% = 67%
序列2：CIHK-T similarity = ((4+1)/6)*100% = 83%

无论两个序列长度是否相同，都要先做双序列全局比对，然后根据比对结果及比对长度计算它们的一致度和相似度。

MOOC内容

5月7日学习

- 第三章 序列比较 3.7 3.8