Gaps

```
ATTACGTACTCCATG
ATTACGT----CATG
```

In an edit script we need 4 edit operations for the gap of length 4.

In maximal score alignments we treat the dash "-" like any other character, hence we charge the s(x,-) costs 4 times.

But

In terms of evolution this gap is probably the result of a single deletion or insertion of length 4.

Biological observations:

Gaps are usually longer then just one character

However, long gaps are less frequent than short gaps

Therefore ...

...gaps should be considered as single units

Gap costs should depend on the length of the gap, they should be monotonously growing, but not as fast as the legth itself.

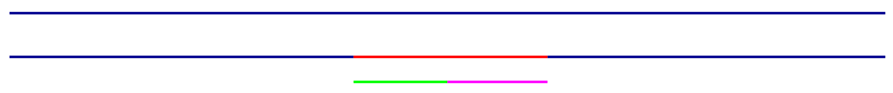
Gap costs should be subadditive:

g(n) gap cost of a gap of length n
n=n1+n2

Subadditivity:

$$g(n) <= g(n1)+g(n2)$$

If not:



Gap is cheaper if it is considered as two successive gaps.

SCORING

Scorematrix for pairs of characters
e.g. VT160

and

Gapcosts g(n)
e.g. g(n)=9+3n

```
MYL--V
M-ACVV
```

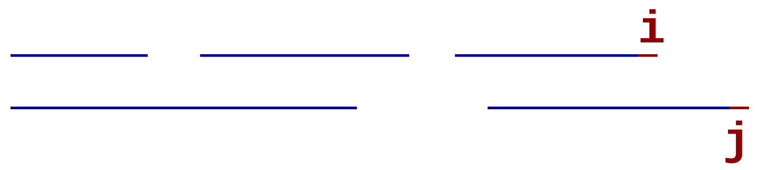
Score= vt(M,M)−g(1)+vt(L,A)−g(2)+vt(V,V)
= 6 −12 −2 −15 +4

= −19

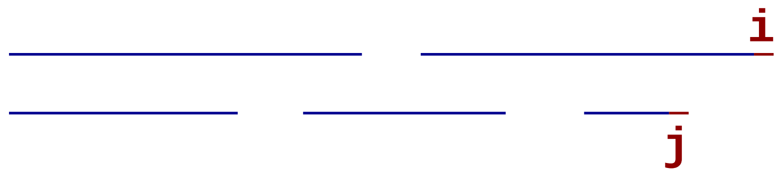
GENERAL GLOBAL ALIGNMENT PROBLEM

Given a score matrix and a subadditive gap cost function, calculate the global maximal score alignment.

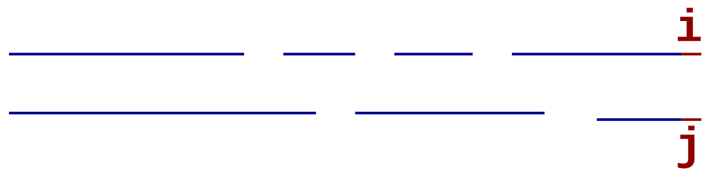
There are three different ways the alignment of S1[1..i] and S2[1..j] can end.



(E)



(F)



(G)

The recurrence relation for maximal score alignments with general gap cost function g(n)

$$S(i,j)=\max\{E(i,j),F(i,j),G(i,j)\}$$

where

$$G(i,j)=S(i-1,j-1)+s(S1(i),S2(j))$$

$$E(i,j)=\max_{k<=j-1}\{V(i,k)-g(j-k)\}$$

$$F(i,j)=\max_{k<=i-1}\{V(k,j)-g(i-k)\}$$

Needleman Wunsch algorithm in a modification by Sankoff.

Initialisation

S(i,0)=-g(i)
S(0,j)=-g(j)

E(i,0)=-g(i)
E(0,i) undefined

F(0,j)=-g(j)
F(j,0) undefined

G(0,0)=0
G(i,0) and G(0,j) undefined

Time Complexity

			W	R	I	T	E	R	S
		0	1	2	3	4	5	6	7
	0	*	*	*	🔵 *	*	*	*	*
V	1	*	*	*	🔵 *	*	*	*	*
I	2	*	*	🟡 *	🔵 *	*	*	*	*
N	3	🔴 *	🔴 *	🔴 *	?				
T	4	*							
N	5	*							
E	6	*							
R	7	*							

🟡	G	$O(nm)$
🔵	F	$O(nm^2)$
🔴	E	$O(mn^2)$

$O(nm^2+mn^2)$

cubic

The number of colored spots depends on the length of the sequences.

Affine gap costs

$g(n)=a+bn$

High costs for opening a gap
but lower costs for extending it

There are two different types of
alignments in the (E) case

```
ATGCTAT-
ACGCAATT        a new gap starts

ATGC----
ACGCAATT        a gap is extended
```

```
ATGCTAT-
ACGCAATT
```

The optimal alignment up to positions i and j−1 is of type (G)

$$E(i,j)=S(i,j-1)-\underset{\text{open}}{\underset{\uparrow}{a}}-\underset{\text{extend}}{\underset{\uparrow}{b}}$$

```
ATGC----
ACGCAATT
```

The optimal alignment up to positions i and j−1 is of type (E)

$$E(i,j)=E(i,j-1)-\underset{\text{extend}}{\underset{\uparrow}{b}}$$

$$E(i,j)=\max\{E(i,j-1),S(i,j-1)-a\}-b$$

Initialisation

```
S(i,0)=E(i,0)=-a-b*i
S(0,j)=F(0,j)=-a-b*j
```

Recurrence relation

```
S(i,j)=max{G(i,j),E(i,j),F(i,j)}

G(i,j)=S(i-1,j-1)+s(S1(i),S2(j))
E(i,j)=max{E(i,j-1),S(i,j-1)-a}-b
F(i,j)=max{F(i-1,j),S(i-1,j)-a}-b
```

Gotoh’s algorithm

TIME COMPLEXITY

General gap costs:

$$E(i,j)= \max_{k<=j-1}\{S(i,k)-g(j-k)\}$$

$$F(i,j)= \max_{k<=i-1}\{S(k,j)-g(i-k)\}$$

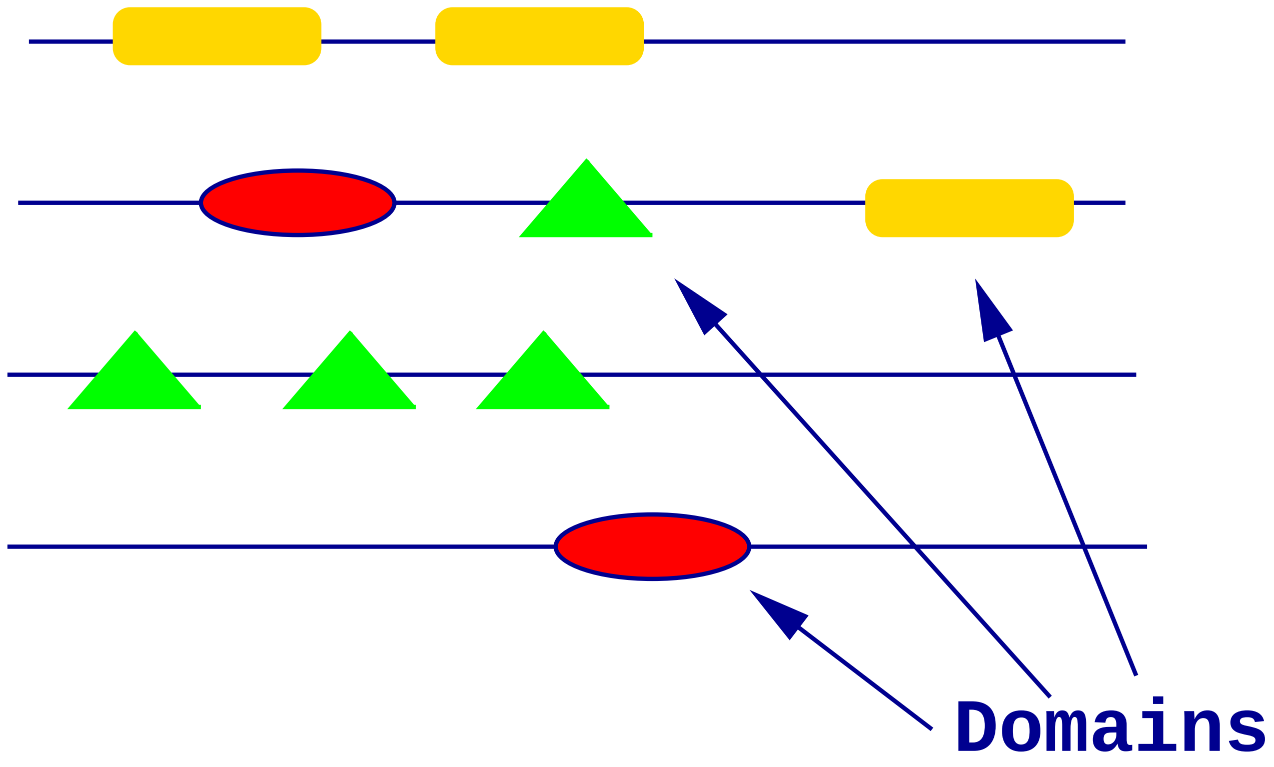
$O(n^3)$

$$E(i,j)=\max\{E(i,j-1),S(i,j-1)-a\}-b$$
$$F(i,j)=\max\{F(i-1,j),S(i-1,j)-a\}-b$$

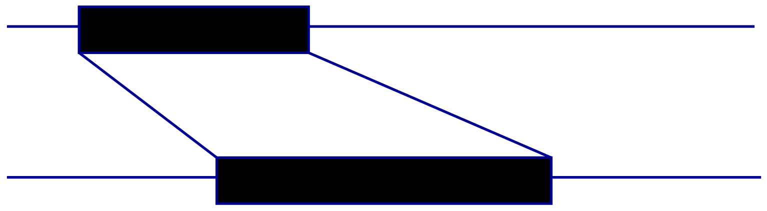
$O(n^2)$

LOCAL CONSERVATION

```
TAGCTAAA CTA ATCGCA
GCGGGACA CTA CTACCT
TCAAAACC CTA ACCAAA
CCCGCTAC CCA TTCAGC
TTCAGCAC CTC CCAGTC
ACTTGCTT CTA ATTTTT
```

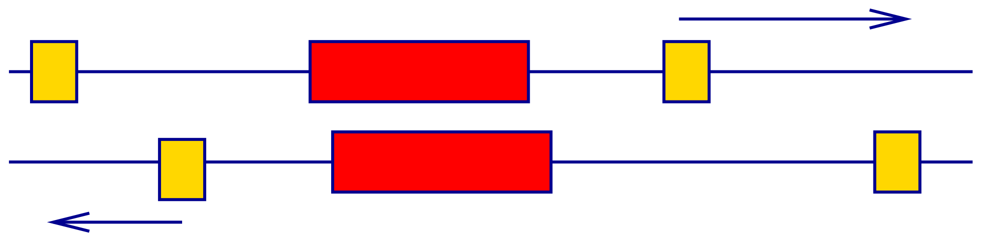

Domains

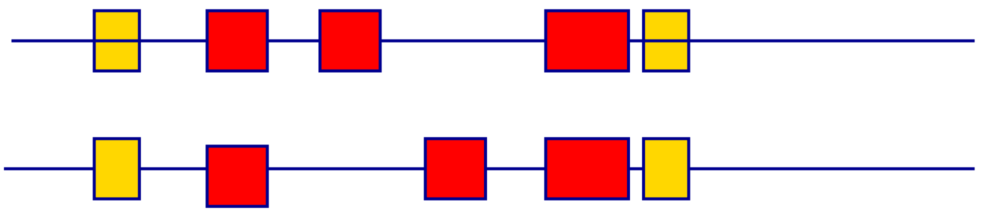
Local Alignment



```
...AT-CTA--TC...
...ATTCCAGATG...
```

Idea: Just detect conserved regions in a global alignment.





The local alignment problem

Given two sequences S1 and S2, find segments a1 and a2 of S1 and S2, whose similarity (global alignment score) is maximal over all pairs of segments from S1 and S2.

We use H(S1,S2) to denote the optimal local alignment score of S1 and S2.

Idea:
We take every pair of segments from S1 and S2, calculate the corresponding optimal global alignment and choose the best one.

Lets say the length of the sequences are n and m.
There are O(n^2m^2) pairs of segments each global segment alignment takes O(nm) time. Hence this algorithm is O(n^3m^3) ... slow

We need to go for a better dynamic programming approach.

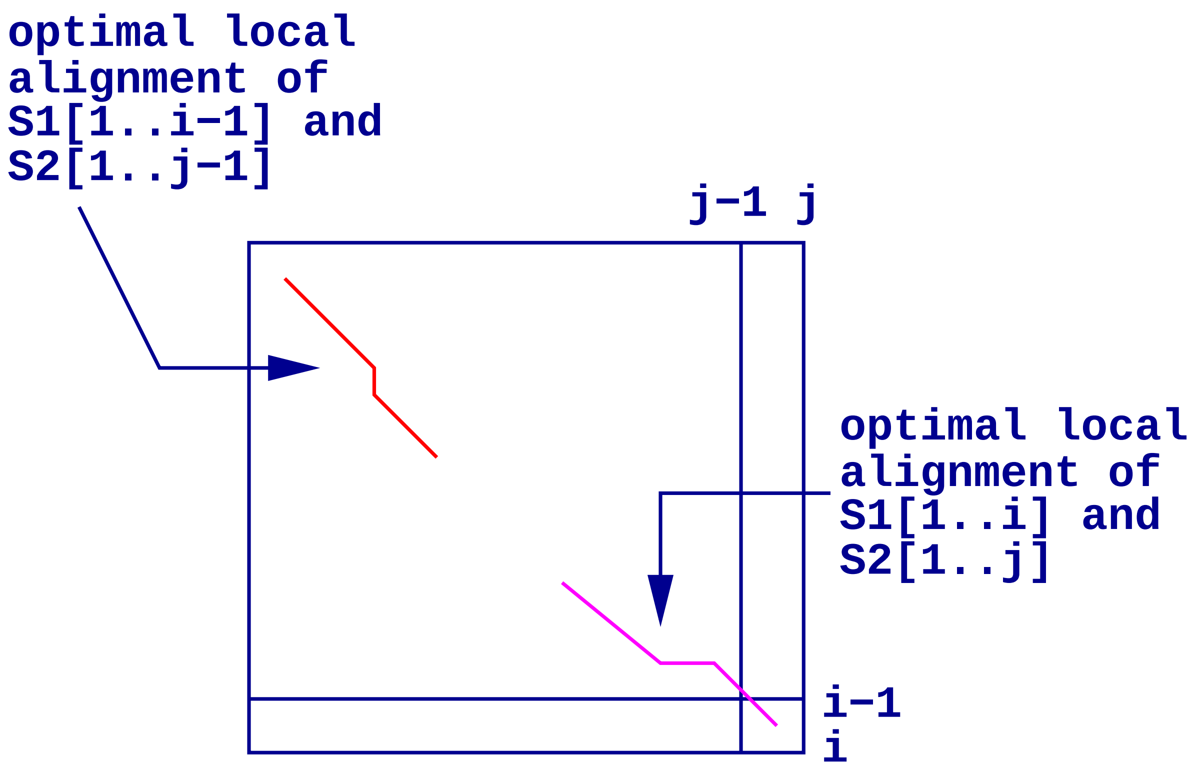
Global Alignment:

D(i,j)=optimal global alignment score of S1[1..i] and S2[1..j].

The optimal alignment is an extension of one of three shorter global alignments.

Local Alignment

H(i,j)=optimal local alignment score of S1[1..i] and S2[1..j] ?

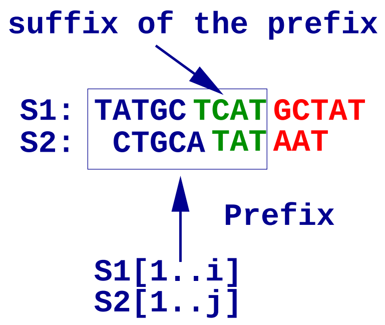

optimal local alignment of S1[1..i−1] and S2[1..j−1]
j−1
j
optimal local alignment of S1[1..i] and S2[1..j]
i−1
i

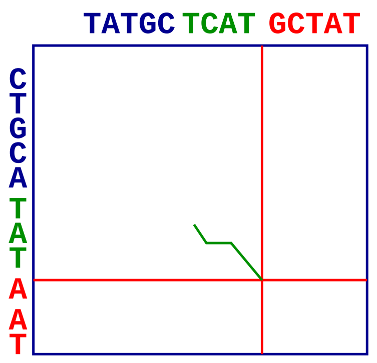
H(i,j)=optimal local alignment score of S1[1..i] and S2[1..j] does not allow a dynamic programming approach.

BUT

What about:
H(i,j)=maximal score of all local alignment, that contain S1(i) and S2(j) as there right most characters.

Optimal suffix alignment score of S1[1..i] and S2[1..j]





THE EMPTY ALIGNMENT

**If none of the alignments that end with matching S1(i) and S2(j) has a non negative score, we say that the optimal local alignment is the empty alignment.
No characters are aligned and the score is zero.**

...assume linear gap costs

4 types of H(i,j)-alignments

(i)
```
...ATCGCT
...TTCCTA
```
H(i−1,j−1)+s(T,A)

(ii)
```
...ATCGCT-
...-TTCCTA
```
H(i,j−1)+g(1)

(iii)
```
...ATCGC-T
...TTCCTA-
```
H(i−1,j)+g(1)

(iv) Empty 0

Note that $\begin{matrix}T\\A\end{matrix}$ is a type (i) extension of the empty alignment

SMITH WATERMAN ALGORITHM

Recurrence relation
for local alignments

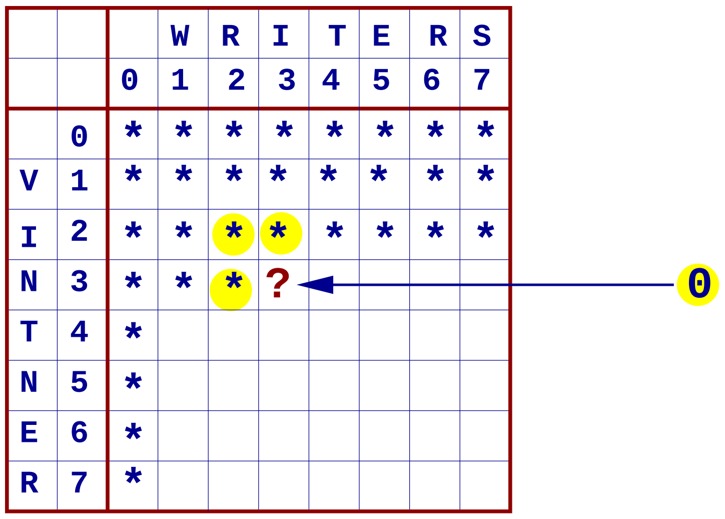
$$H(i,j) = \max \begin{cases} H(i-1,j-1)+s(S1(i),S2(j)) \\ H(i,j-1)+s(-,S2(i)) \\ H(i-1,j)+s(S1(i),-) \\ 0 \end{cases}$$

S(i,j) = optimal global alignment
score of S1[1..i] and S2[1..j].

INITIALIZATION

			W	R	I	T	E	R	S
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
V	1	0							
I	2	0							
N	3	0							
T	4	0							
N	5	0							
E	6	0							
R	7	0							

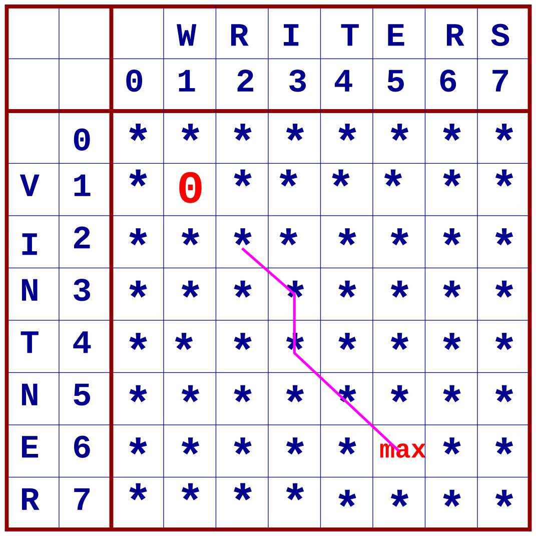
Tabular calculation



			W	R	I	T	E	R	S
		0	1	2	3	4	5	6	7
	0	*	*	*	*	*	*	*	*
V	1	*	*	*	*	*	*	*	*
I	2	*	*	*	*	*	*	*	*
N	3	*	*	*	?				
T	4	*							
N	5	*							
E	6	*							
R	7	*							

The number of colored spots depends on the lengths of the sequences.

Traceback



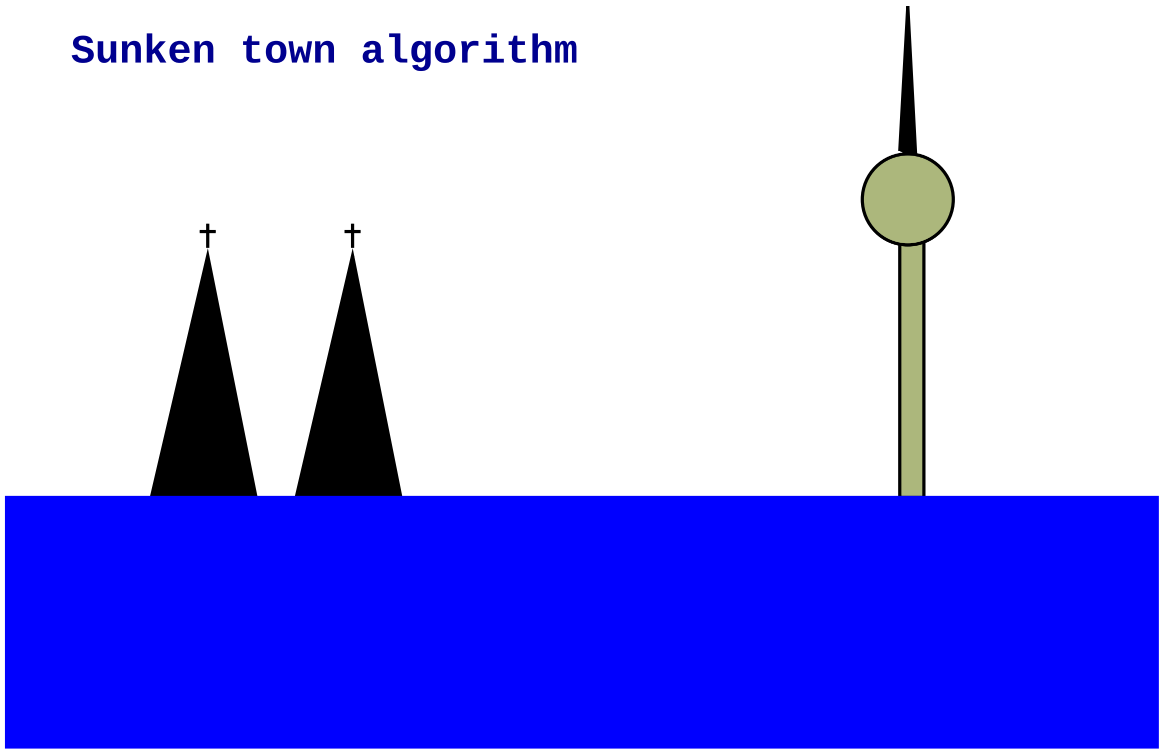
			W	R	I	T	E	R	S
		0	1	2	3	4	5	6	7
	0	*	*	*	*	*	*	*	*
V	1	*	0	*	*	*	*	*	*
I	2	*	*	*	*	*	*	*	*
N	3	*	*	*	*	*	*	*	*
T	4	*	*	*	*	*	*	*	*
N	5	*	*	*	*	*	*	*	*
E	6	*	*	*	*	*	max	*	*
R	7	*	*	*	*	*	*	*	*

The number of colored spots depends on the length of the sequences.

			S	C	A	P	C	A	L
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
E	1	0	0	0	0	0	0	0	0
D	2	0	0	0	0	0	0	0	0
C	3	0	0	12	4	0	12	4	0
P	4	0	1	4	13	10	4	13	5
C	5	0	0	13	5	10	22	14	5
D	6	0	0	5	13	5	14	22	14

```
CAPC          CAPCA
C−PC          C−PCD
```

Sunken town algorithm



```
match    =1
mismatch = −infinity
gap = −infinity
```

Optimal local alignment is the longest common word of the two sequences

A Sequence

S1:a1 a2 a3 a4 a5 a6 a7 a8 a9 a10...

any sequence
a_{i_1} a_{i_2} a_{i_3} a_{i_4} a_{i_5} a_{i_6} a_{i_7} a_{i_8} ...

with $i_1 < i_2 < i_3 < i_4$...
is a subsequence of S1.

Example:

S1 ATTGTTCCTACTGTA

TCCATT

```
S1 ATTGTTCCTACTGTA
S2 TCCGCAACCAATGGTCT
```

TCCATT is a common subsequence of S1 and S2.

mismatch	=	−infinity
match	=	+1
gap	=	0

Optimal local alignment is the longest common subsequence

q(i)=Probability for character ai
in the background model
(relative frequency of amino acid ai)

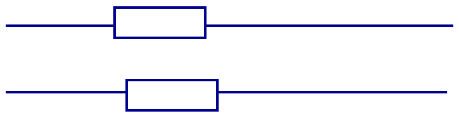
$$E_q[s] = \sum_{i,j} q(i)q(j)s(ai,aj)$$

expected score per match position.

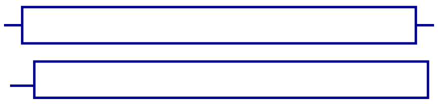
If the local alignment should be a
small region in the alignment
$E_q[s]$ must be negative.

Otherwise one would in average gain
score by randomly matching characters.
The alignment would be almost as long
as the sequences.

$E_q[s]<0$
local

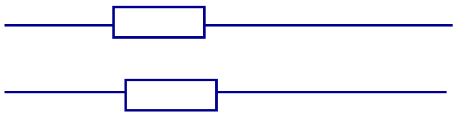


$E_q[s]>0$
quasi global

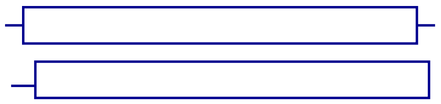


Phase transition $E_q[s]=0$.

expensive gaps
longest common
word -- local



cheap gaps longest
common subsequence
quasi global



Phase transition = ?

Probabilistic scores
are ok for local alignment.

$$s(ai,aj) = \log \frac{M_{ij}}{q_i q_j}$$

$$E_q[s] = \sum q_i q_j \log \frac{M_{ij}}{q_i q_j}$$

$$\le \log \sum q_i q_j \frac{M_{ij}}{q_i q_j}$$

$$= \log(1)$$
$$= 0$$