Sequence
Alignment Algorithms

Morten Nielsen
Department of Health Technology,
DTU

Why learn about alignment algorithms?

- All publicly available alignment programs do the same thing
  - Sequence alignment using amino acids substitution matrices and affine gap penalties
- This is fast but not optimal
  - Protein alignment is done much more accurate using sequence profiles and position specific gap penalties (price for gaps depends on the structure)
- Must implement your own alignment algorithm to do this

Outline

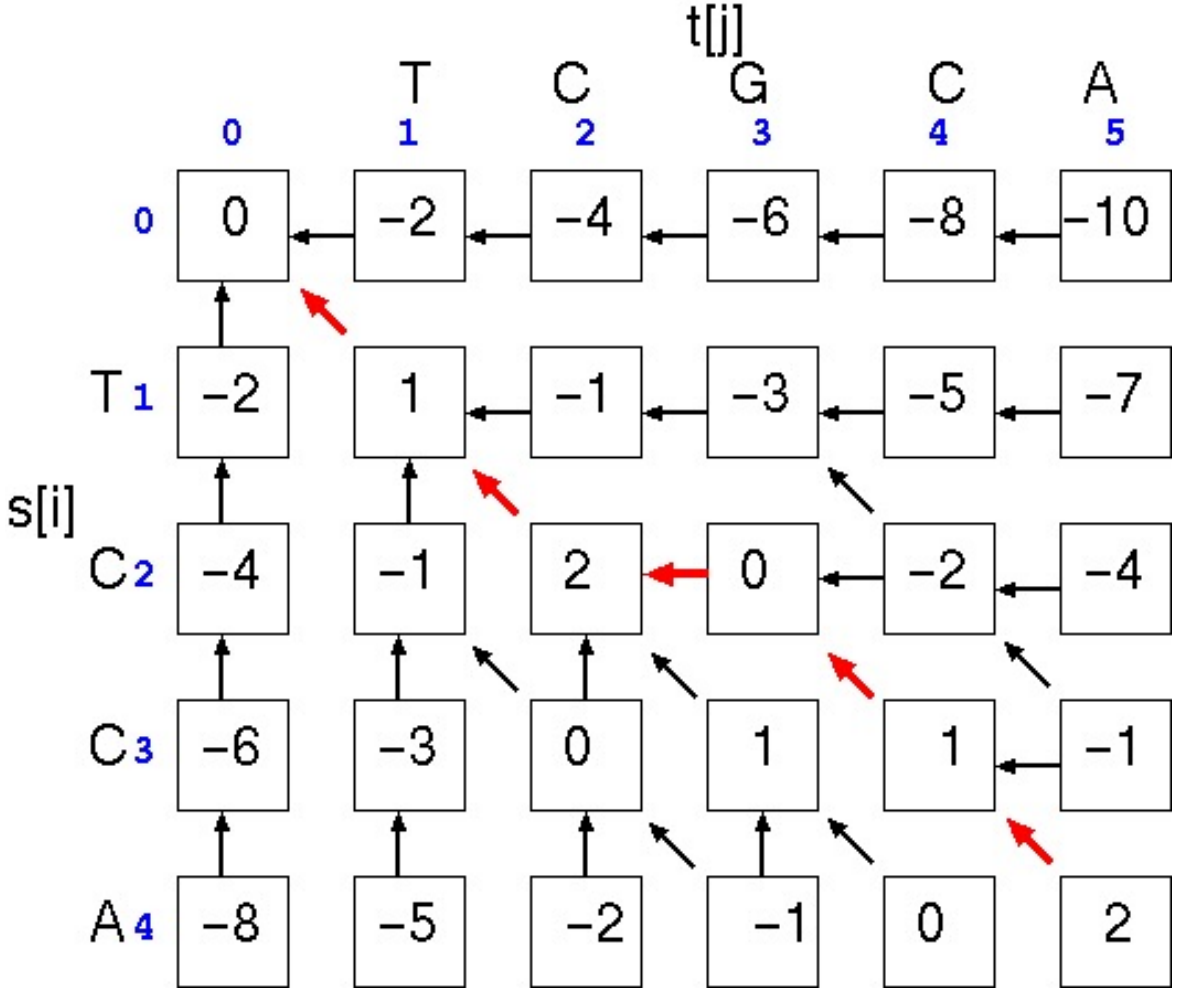
- What you have been told is not entirely true :-)
  - Alignment algorithms are more complex
- The true sequence alignment algorithm story
  - The slow algorithm (O3)
  - The fast algorithm (O2)

Sequence alignment
The old Story

Pairwise alignment: the solution

”Dynamic programming”

(the Needleman-Wunsch algorithm)

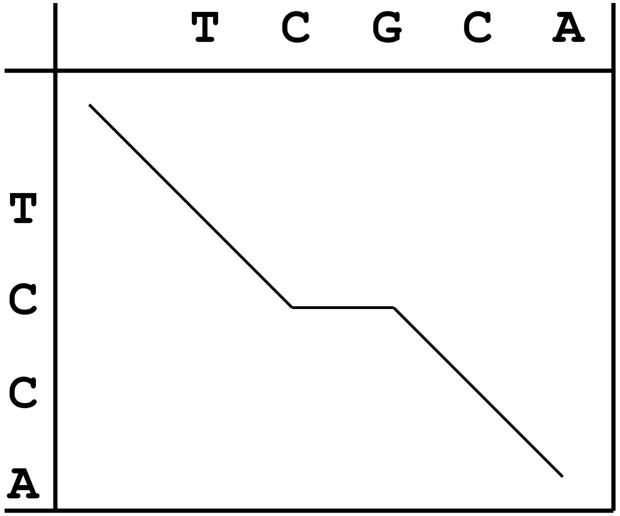


	0	T 1	C 2	G 3	C 4	A 5
0	0	−2	−4	−6	−8	−10
T 1	−2	1	−1	−3	−5	−7
C 2	−4	−1	2	0	−2	−4
C 3	−6	−3	0	1	1	−1
A 4	−8	−5	−2	−1	0	2

Match score = 1
Mismatch score -1
Gap penalty = -2

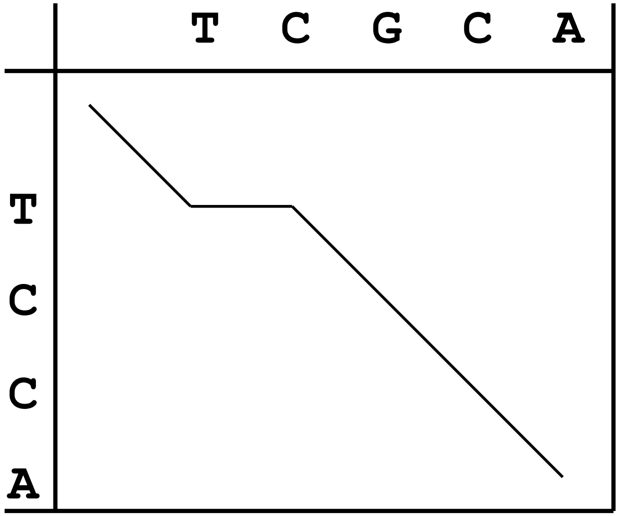
Alignment depicted as path in matrix

Match score = 1
Mismatch score = -1
Gap penalty = -2



```
TCGCA
TC-CA
```

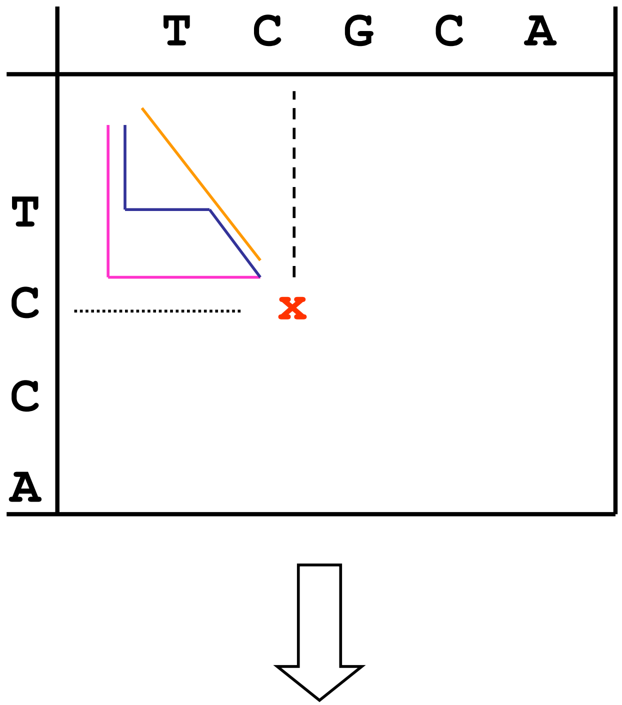
Score=2



```
TCGCA
T-CCA
```

Score=0

Alignment depicted as path in matrix



Meaning of point in matrix: all residues up to this point have been aligned (but there are many different possible paths).

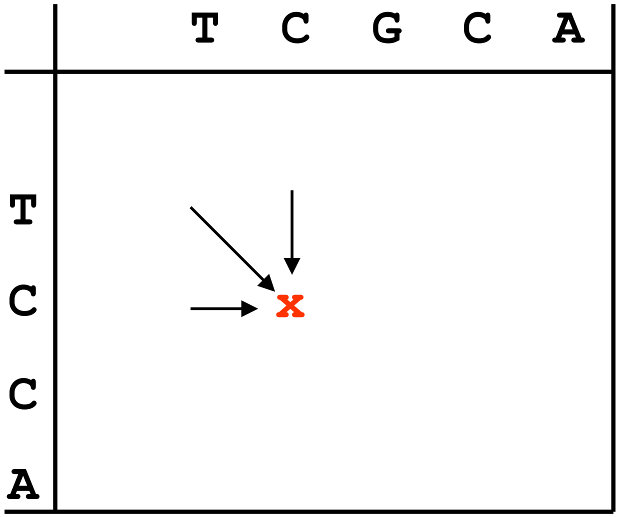
Position labeled "**x**": **TC** aligned with **TC**

```
--TC
TC--
```

```
-TC
T-C
```

```
TC
TC
```

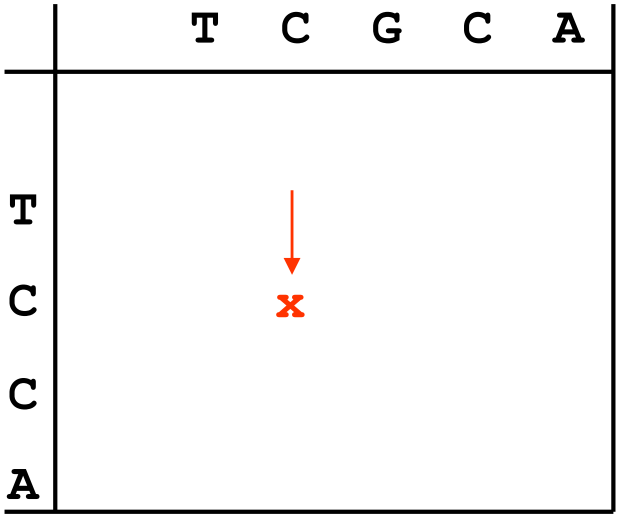
Dynamic programming: computation of scores



Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

Dynamic programming: computation of scores

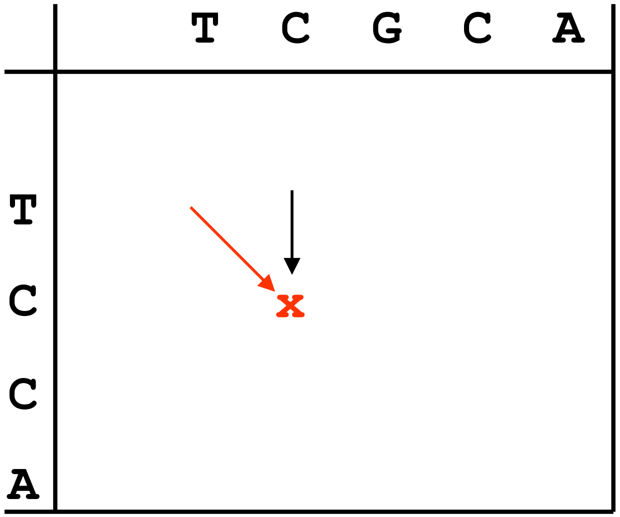


Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

$$\text{score}(x,y) = \max \begin{cases} \text{score}(x,y-1) - \text{gap-penalty} \\ \\ \end{cases}$$

Dynamic programming: computation of scores

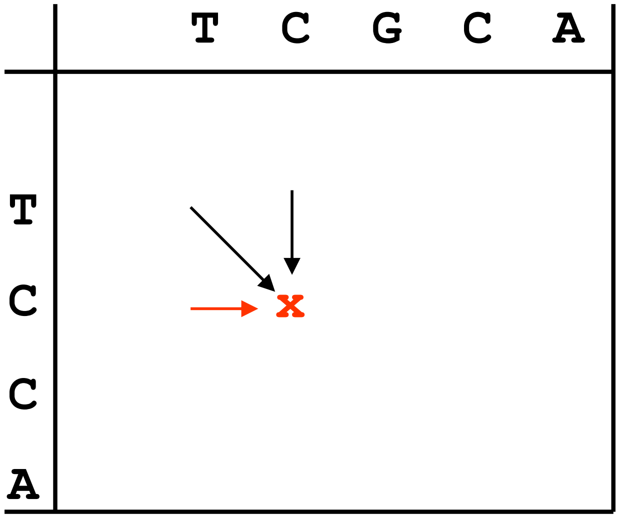


Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

$$\text{score}(x,y) = \max \begin{cases} \text{score}(x,y\text{-}1) - \text{gap-penalty} \\ \text{score}(x\text{-}1,y\text{-}1) + \text{substitution-score}(x,y) \\ \end{cases}$$

Dynamic programming: computation of scores

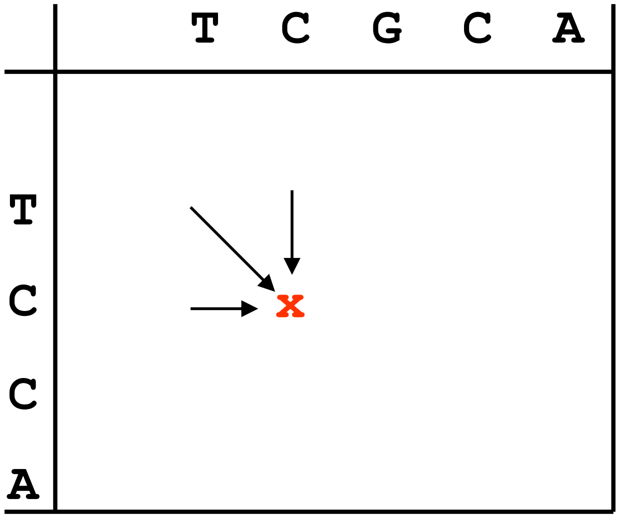


Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

$$\text{score}(x,y) = \max \begin{cases} \text{score}(x,y-1) - \text{gap-penalty} \\ \text{score}(x-1,y-1) + \text{substitution-score}(x,y) \\ \text{score}(x-1,y) - \text{gap-penalty} \end{cases}$$

Dynamic programming: computation of scores



Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

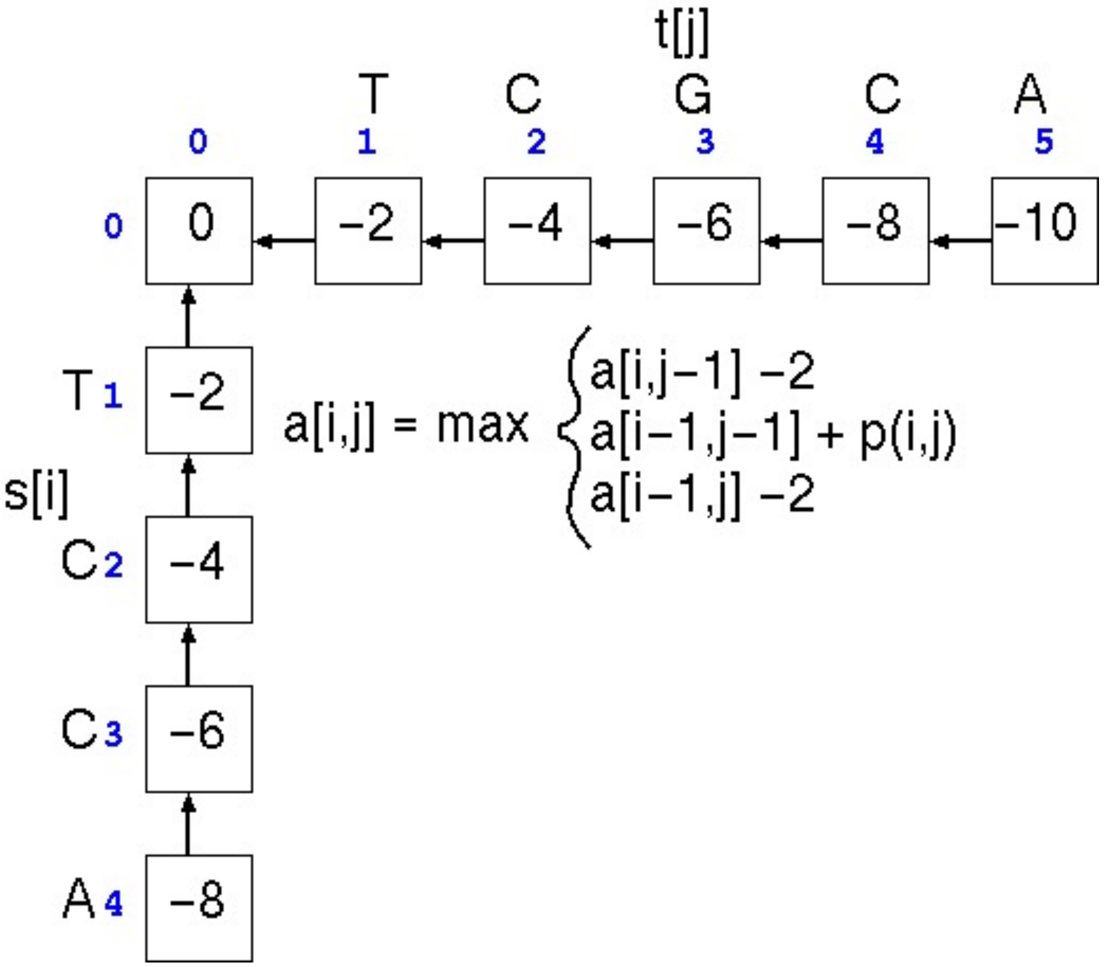
=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

Each new score is found by choosing the maximum of three possibilities. For each square in matrix: keep track of where best score came from.

Fill in scores one row at a time, starting in upper left corner of matrix, ending in lower right corner.

$$\text{score}(x,y) = \max \begin{cases} \text{score}(x,y-1) - \text{gap-penalty} \\ \text{score}(x-1,y-1) + \text{substitution-score}(x,y) \\ \text{score}(x-1,y) - \text{gap-penalty} \end{cases}$$

Dynamic programming: example



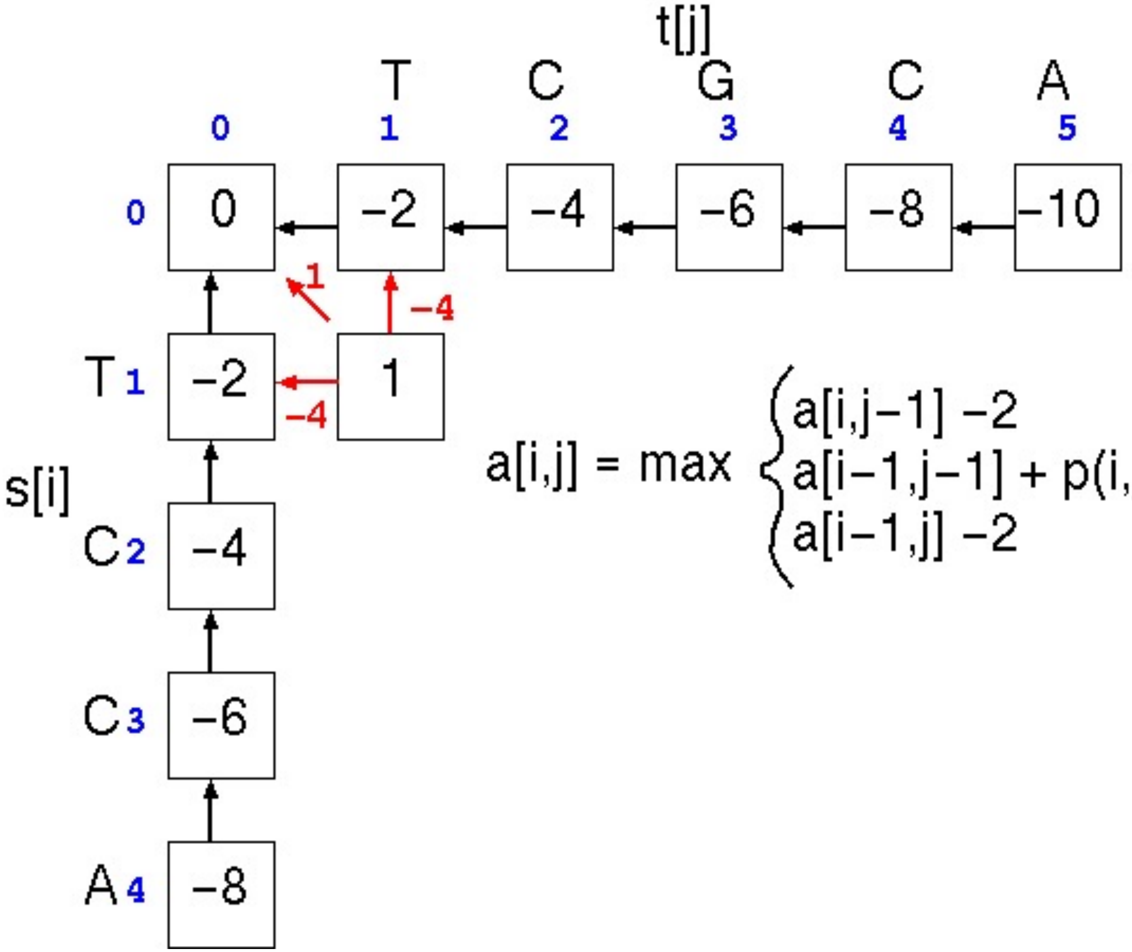
			T	C	G	C	A
		0	1	2	3	4	5
	0	0	-2	-4	-6	-8	-10
T	1	-2					
C	2	-4					
C	3	-6					
A	4	-8					

$$a[i,j] = \max \begin{cases} a[i,j-1]\ -2 \\ a[i-1,j-1] + p(i,j) \\ a[i-1,j]\ -2 \end{cases}$$

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

Gaps: -2

Dynamic programming: example

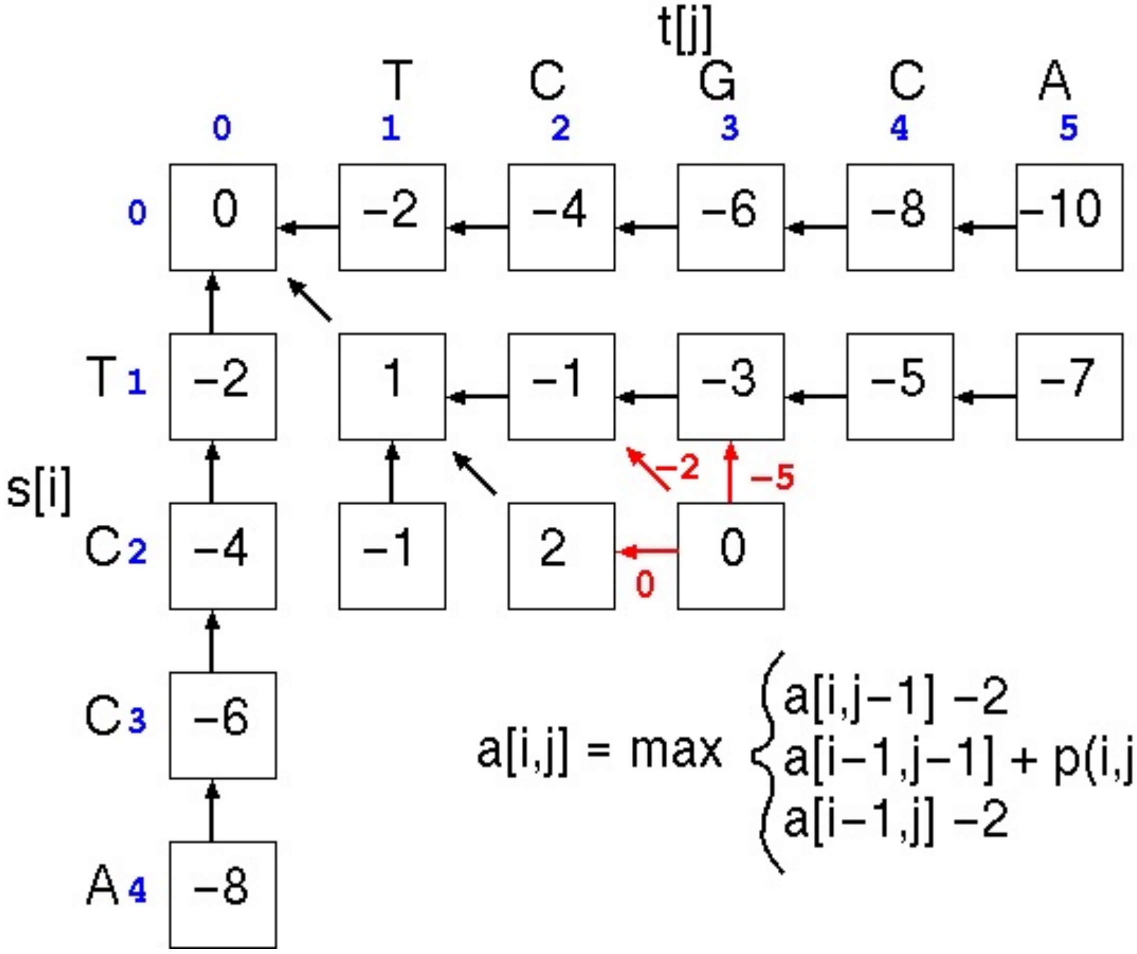


$$a[i,j] = \max \begin{cases} a[i,j-1]\ -2 \\ a[i-1,j-1] + p(i,j) \\ a[i-1,j]\ -2 \end{cases}$$

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

Gaps: -2

Dynamic programming: example

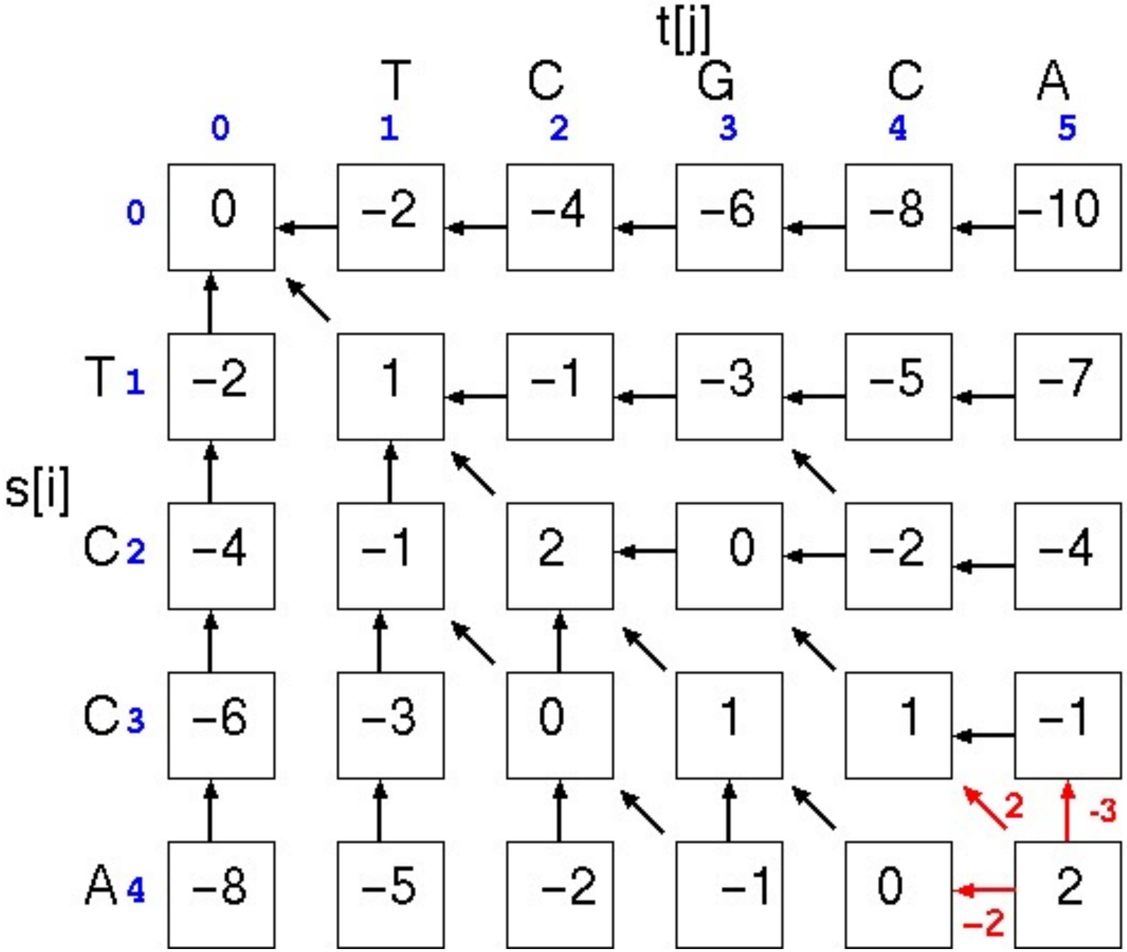


$$a[i,j] = \max \begin{cases} a[i,j-1] - 2 \\ a[i-1,j-1] + p(i,j) \\ a[i-1,j] - 2 \end{cases}$$

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

Gaps: -2

Dynamic programming: example

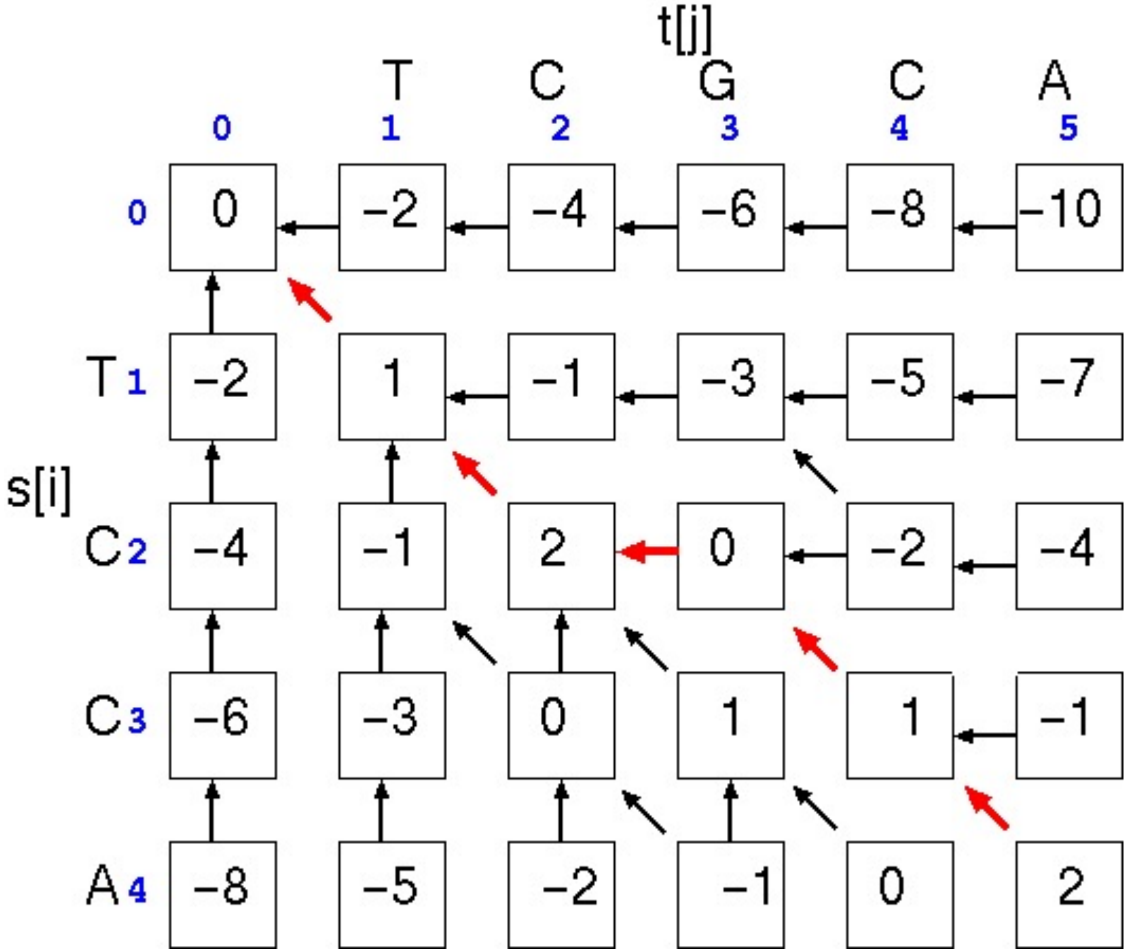


		T	C	G	C	A
	0	1	2	3	4	5
0	0	−2	−4	−6	−8	−10
T 1	−2	1	−1	−3	−5	−7
C 2	−4	−1	2	0	−2	−4
C 3	−6	−3	0	1	1	−1
A 4	−8	−5	−2	−1	0	2

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

Gaps: -2

Dynamic programming: example

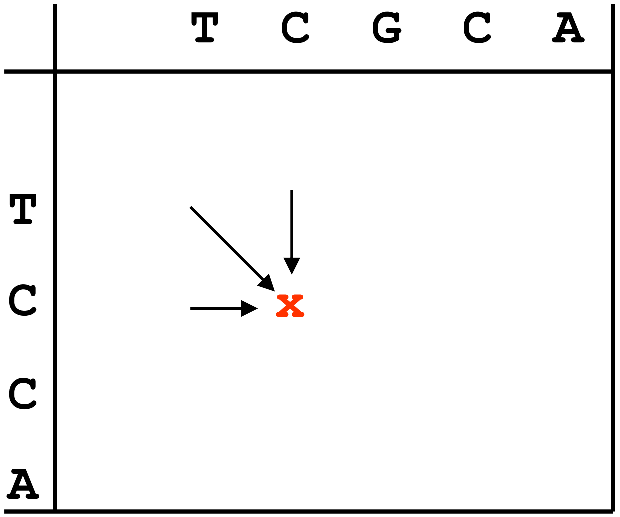


	0	T 1	C 2	G 3	C 4	A 5
0	0	−2	−4	−6	−8	−10
T 1	−2	1	−1	−3	−5	−7
C 2	−4	−1	2	0	−2	−4
C 3	−6	−3	0	1	1	−1
A 4	−8	−5	−2	−1	0	2

```
T C G C A
: :   : :
T C - C A
1+1-2+1+1 = 2
```

But in reality is a little more complex

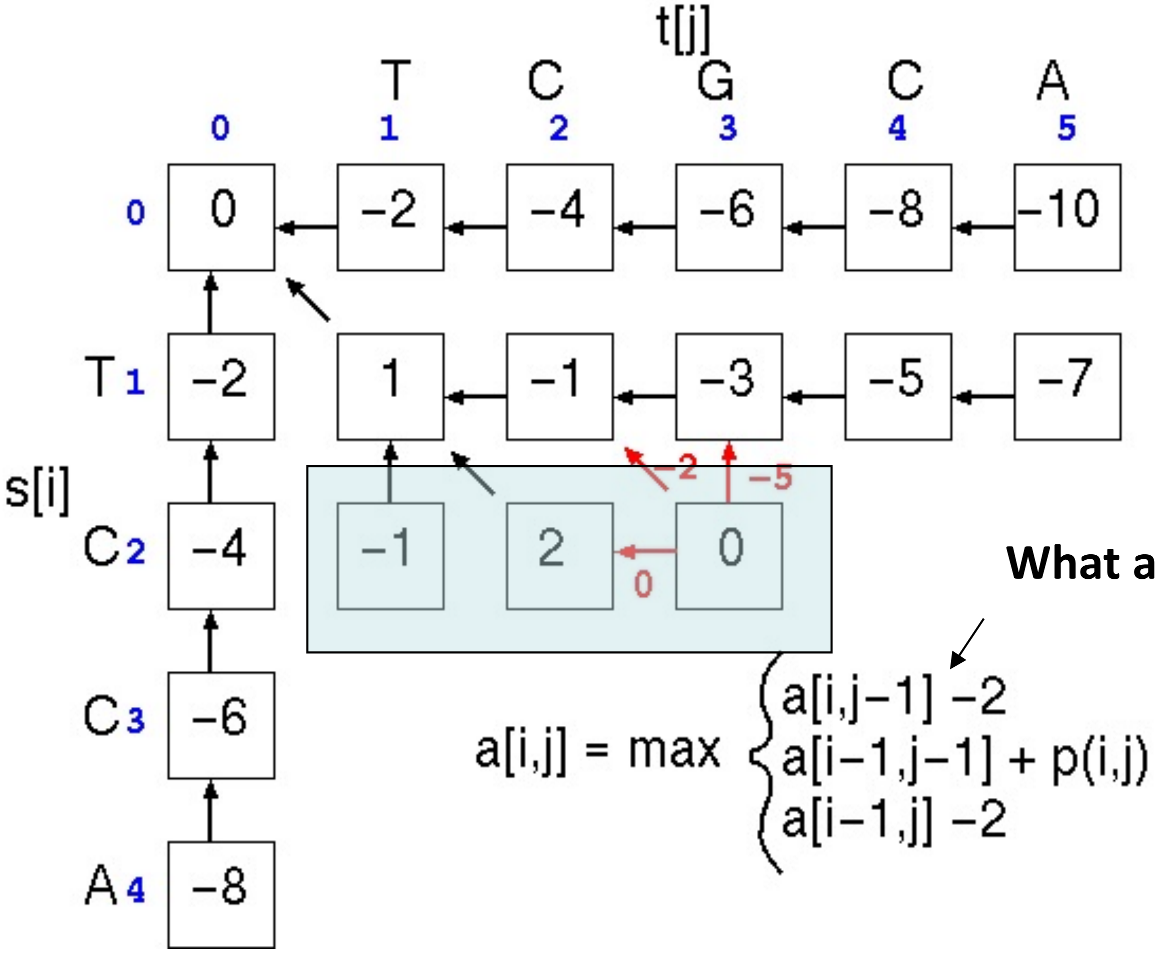
Dynamic programming: computation of scores



Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

Dynamic programming: example



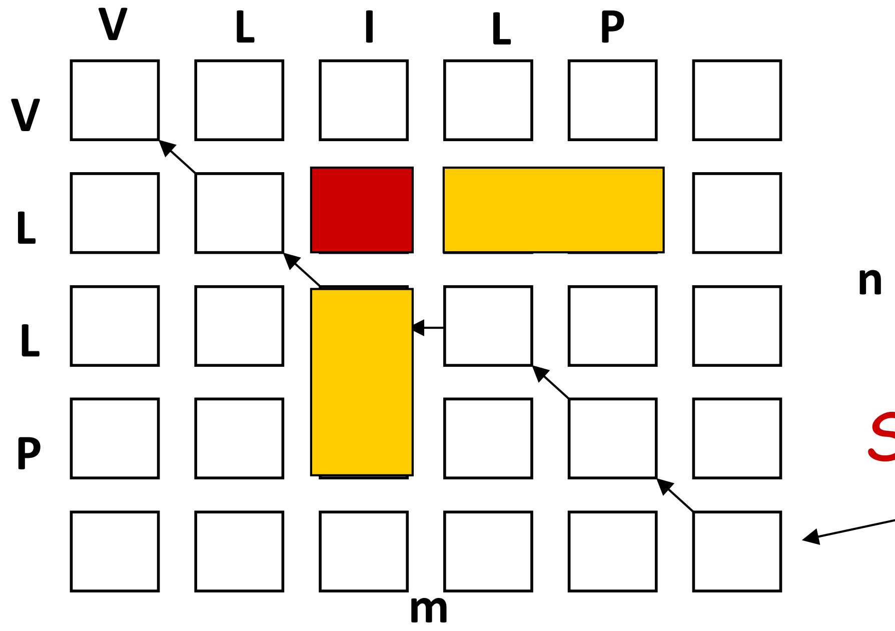
		0	T 1	C 2	G 3	C 4	A 5
	0	0	−2	−4	−6	−8	−10
T	1	−2	1	−1	−3	−5	−7
C	2	−4	−1	2	0		
C	3	−6					
A	4	−8					

$$a[i,j] = \max \begin{cases} a[i,j-1] -2 \\ a[i-1,j-1] + p(i,j) \\ a[i-1,j] -2 \end{cases}$$

What about j-2 and a gap extension?

And now the true algorithm

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ \underset{1 \le k \le N-m}{Max}\left[D_{m+k,n} + w_k\right] \\ \underset{1 \le k \le N-n}{Max}\left[D_{m,n+k} + w_k\right] \\ 0 \text{ (for local alignment)} \end{cases}$$



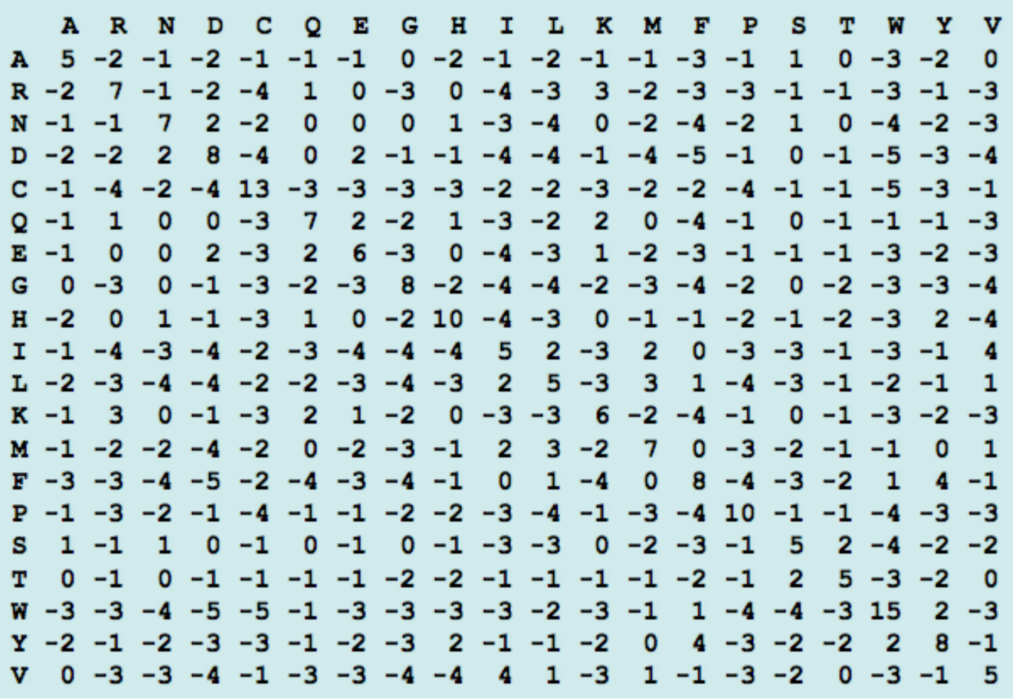
Start from here

How does it work (score matrix d)?

d matrix

	V	L	I	L	P
V	5	1	4	1	-3
L	1	5	2	5	-4
L	1	5	2	5	-4
P	-3	-4	-3	-4	10

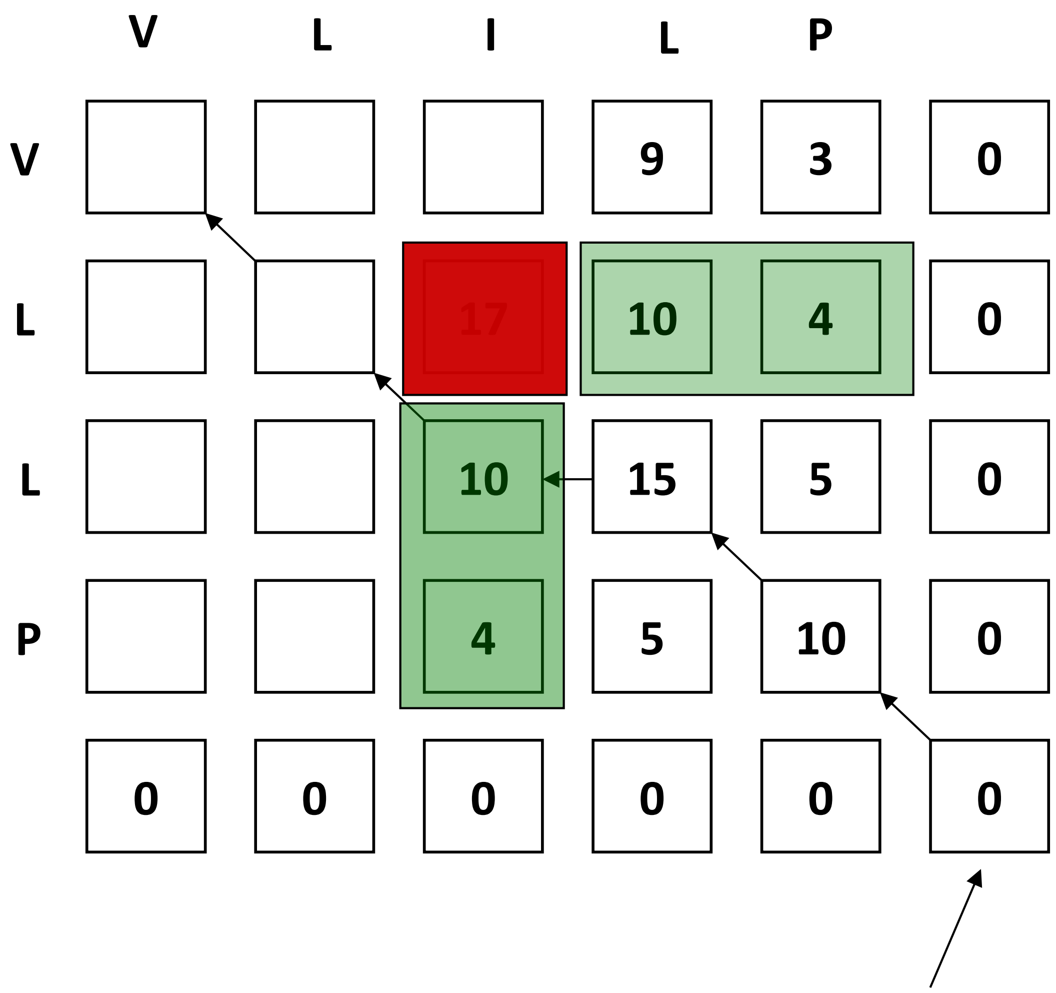
Blosum 50 matrix



	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5

How does it work? (The slow way O3)

D matrix



	V	L	I	L	P	
V				9	3	0
L			17	10	4	0
L			10	15	5	0
P			4	5	10	0
	0	0	0	0	0	0

Start from here!

d matrix

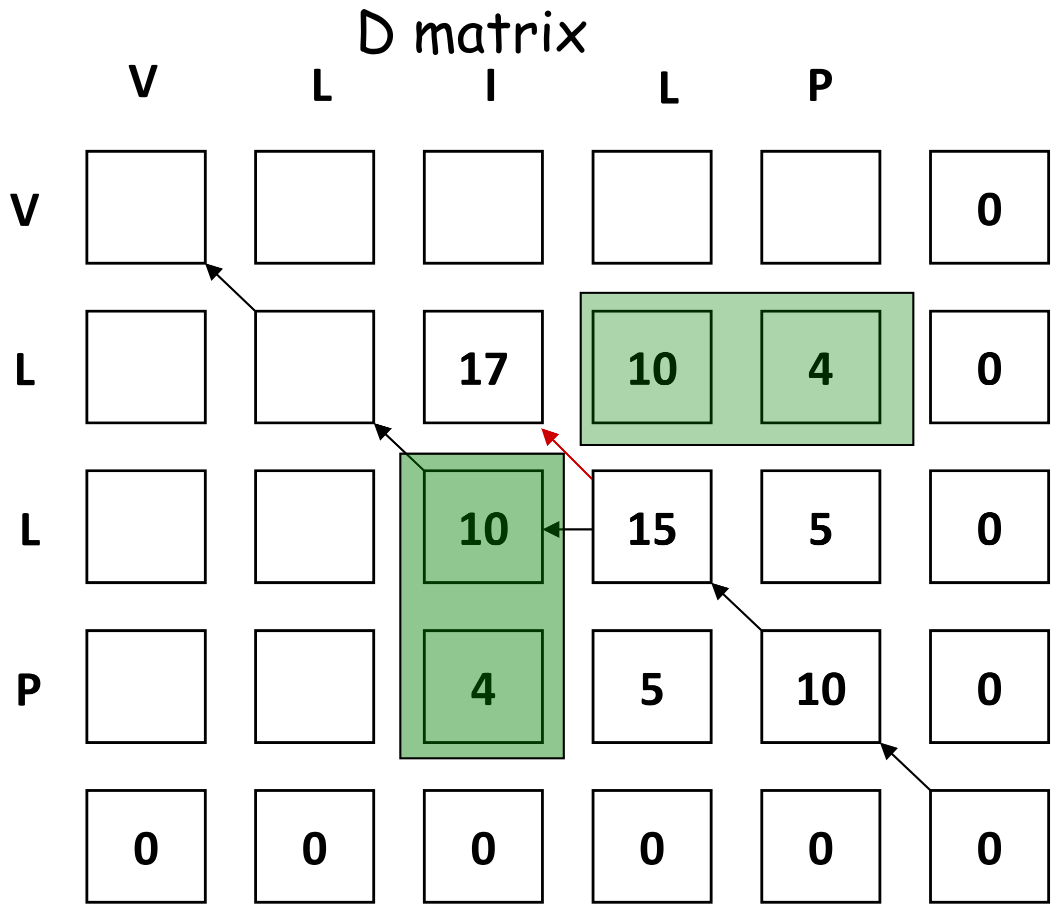
	V	L	I	L	P
V	5	1	4	1	-3
L	1	5	2	5	-4
L	1	5	2	5	-4
P	-3	-4	-3	-4	10

Gap-open $W_1 = -5$
Gap-extension $U = -1$

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ \underset{1\le k\le N-m}{Max}\left[D_{m+k,n} + w_k\right] \\ \underset{1\le k\le N-n}{Max}\left[D_{m,n+k} + w_k\right] \\ 0 \text{ (for local alignment)} \end{cases}$$

How does it work? (The slow way O3)

D matrix



	V	L	I	L	P	
V						0
L			17	10	4	0
L			10	15	5	0
P			4	5	10	0
	0	0	0	0	0	0

d matrix

	V	L	I	L	P
V	5	1	4	1	-3
L	1	5	2	5	-4
L	1	5	2	5	-4
P	-3	-4	-3	-4	10

Gap-open $W_1 = -5$
Gap-extension $U = -1$

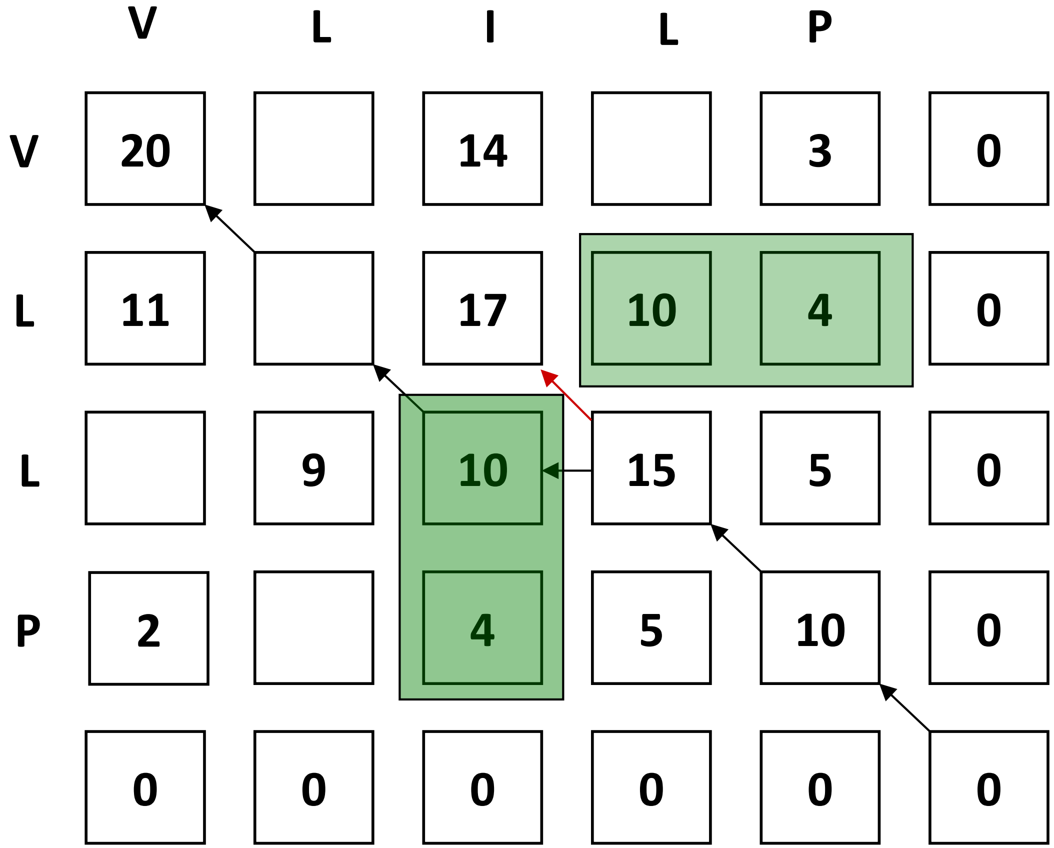
$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ \underset{1\le k\le N-m}{Max}\left[D_{m+k,n} + w_k\right] \\ \underset{1\le k\le N-n}{Max}\left[D_{m,n+k} + w_k\right] \\ 0 \text{ (for local alignment)} \end{cases}$$

- Check all positions in (green) row and column to check score for gap extension.
- CPU intensive (O3)

And now you!

How does it work? Fill out the D matrix

D matrix



	V	L	I	L	P	
V	20		14		3	0
L	11		17	10	4	0
L		9	10	15	5	0
P	2		4	5	10	0
	0	0	0	0	0	0

- Check all positions in (green) row and column to check score for gap extension.
- CPU intensive (O3)

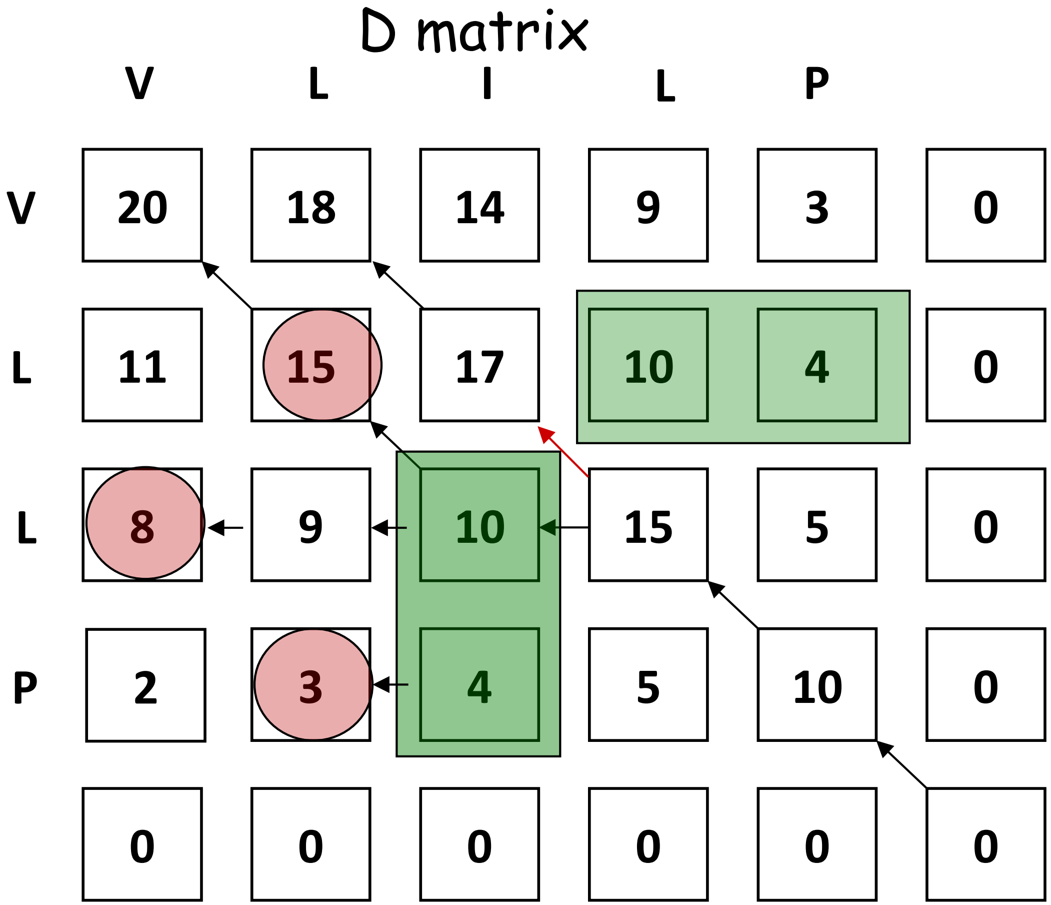
d matrix

	V	L	I	L	P
V	5	1	4	1	-3
L	1	5	2	5	-4
L	1	5	2	5	-4
P	-3	-4	-3	-4	10

Gap-open $W_1 = -5$
Gap-extension $U = -1$

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ \underset{1 \le k \le N-m}{Max}\left[D_{m+k,n} + w_k\right] \\ \underset{1 \le k \le N-n}{Max}\left[D_{m,n+k} + w_k\right] \\ 0 \text{ (for local alignment)} \end{cases}$$

How does it work (The slow way O3)



D matrix

	V	L	I	L	P	
V	20	18	14	9	3	0
L	11	15	17	10	4	0
L	8	9	10	15	5	0
P	2	3	4	5	10	0
	0	0	0	0	0	0

d matrix

	V	L	I	L	P
V	5	1	4	1	-3
L	1	5	2	5	-4
L	1	5	2	5	-4
P	-3	-4	-3	-4	10

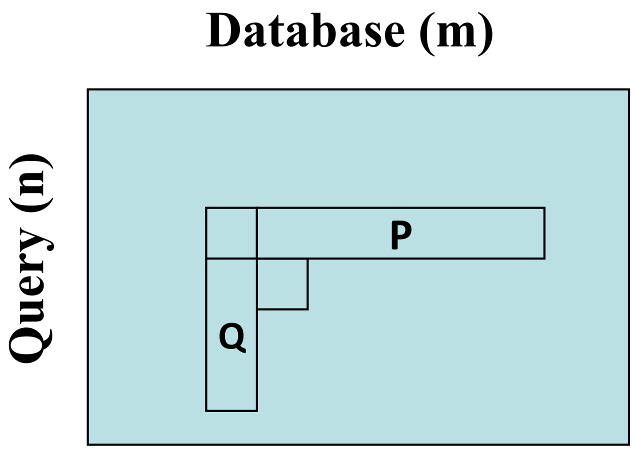
Gap-open $W_1 = -5$
Gap-extension $U = -1$

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ \underset{1\le k\le N-m}{Max}\left[D_{m+k,n} + w_k\right] \\ \underset{1\le k\le N-n}{Max}\left[D_{m,n+k} + w_k\right] \\ 0 \text{ (for local alignment)} \end{cases}$$

- Check all positions in (green) row and column to check score for gap extension.
- CPU intensive (O3)

And now the fast algorithm (O2)

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ P_{m,n}, \text{insertion in database} \\ Q_{m,n}, \text{insertion in query} \\ 0 \end{cases}$$



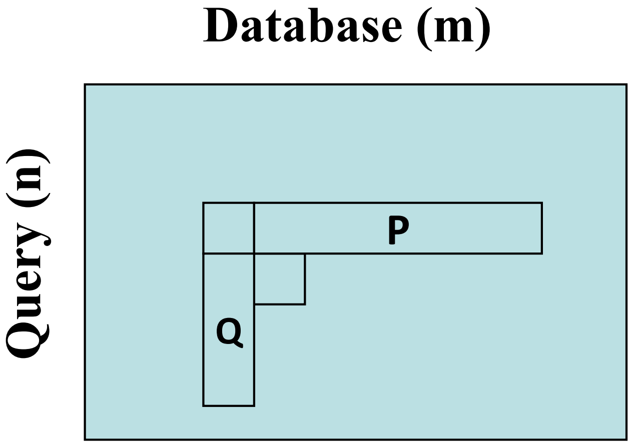
Affine gap penalties

$$\boxed{w_k = w_1 + u\cdot(k-1)}$$

Open a gap Extending a gap

And now the fast algorithm (O2)

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ P_{m,n}, \text{insertion in database} \\ Q_{m,n}, \text{insertion in query} \\ 0 \end{cases}$$



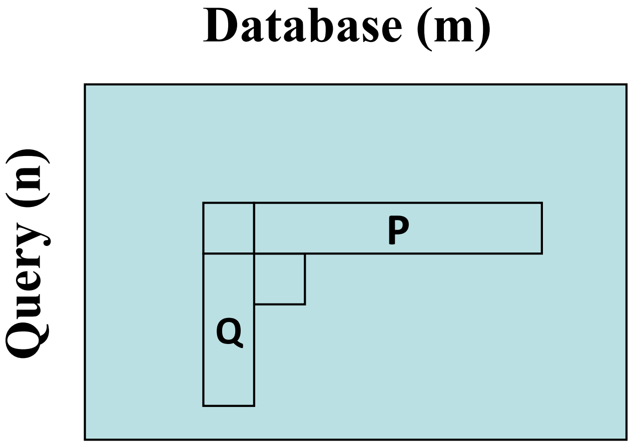
$$\begin{aligned} P_{m,n} &= \underset{1\le k\le N-m}{Max}\left[D_{m+k,n} + w_k\right] \\ &= Max\left[D_{m+1,n} + w_1, \underset{1\le k\le N-m-1}{Max}(D_{m+1+k,n} + w_{k+1})\right] \\ &= Max\left[D_{m+1,n} + w_1, \underset{1\le k\le N-m-1}{Max}(D_{m+1+k,n} + w_k) + u\right], \text{k} + 1 > 1 \\ &= Max\left[D_{m+1,n} + w_1, P_{m+1,n} + u\right] \end{aligned}$$

Open a gap

Extending a gap

And now the true algorithm (cont.)

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ P_{m,n}, \text{insertion in database} \\ Q_{m,n}, \text{insertion in query} \\ 0 \end{cases}$$



$$\boxed{\begin{aligned} P_{m,n} &= Max\left[D_{m+1,n} + w_1, P_{m+1,n} + u\right] \\ Q_{m,n} &= Max\left[D_{m,n+1} + w_1, Q_{m,n+1} + u\right] \end{aligned}}$$

How does it work (D,Q, and P-matrices)

D

m

n

	V	L	I	L	P	
V						0
L						0
L					5	0
P	2	3	4	5	10	0
	0	0	0	0	0	0

$$\boxed{P_{m,n} = Max\left[D_{m+1,n} + w_1, P_{m+1,n} + u\right]}$$

$$\boxed{P_{m,n} = Max\left[5-5,-1-1\right]=0}$$

P

	V	L	I	L	P	
V						0
L						0
L				🟥	-1	0
P				5	-1	0
	0	0	0	0	0	0

W_1 = -5
U = -1

How does it work (D,Q,P, and E-matrices)

D

m

n

	V	L	I	L	P	
V						0
L						0
L					5	0
P	2	3	4	5	10	0
	0	0	0	0	0	0

P

	V	L	I	L	P	
V						0
L						0
L				0	-1	0
P				5	-1	0
	0	0	0	0	0	0

P works horizontally

$$\boxed{P_{m,n} = Max\left[D_{m+1,n} + w_1, P_{m+1,n} + u\right]}$$

$$\boxed{P_{m,n} = Max\left[5-5, -1-1\right] = 0}$$

W_1 = -5
U = -1

How does it work (D,Q, and P-matrices)

D

m

n

	V	L	I	L	P	
V						0
L						0
L					5	0
P	2	3	4	5	10	0
	0	0	0	0	0	0

Q

	V	L	I	L	P	
V						
L						
L				🟥	5	0
P	-1	-1	-1	-1	-1	0
	0	0	0	0	0	0

$$\boxed{Q_{m,n} = Max\left[D_{m,n+1} + w_1, Q_{m,n+1} + u\right]}$$

$$\boxed{Q_{m,n} = Max\left[5-5, -1-1\right] = 0}$$

$W_1 = -5$
$U = -1$

How does it work (D,Q,P, and E-matrices)

D

m

n

	V	L	I	L	P	
V						0
L						0
L					5	0
P	2	3	4	5	10	0
	0	0	0	0	0	0

Q

	V	L	I	L	P	
V						
L						
L				0	5	0
P	-1	-1	-1	-1	-1	0
	0	0	0	0	0	0

$$\boxed{Q_{m,n} = Max\left[D_{m,n+1} + w_1, Q_{m,n+1} + u\right]}$$

$$\boxed{Q_{m,n} = Max\left[5-5, -1-1\right] = 0}$$

W_1 = -5
U = -1

How does it work (D, Q, and P-matrices)

D

m

n

	V	L	I	L	P	
V						0
L						0
L				🟥	5	0
P	2	3	4	5	10	0
	0	0	0	0	0	0

Q

	V	L	I	L	P	
V						
L						
L				0	5	0
P	-1	-1	-1	-1	-1	0
	0	0	0	0	0	0

P

	V	L	I	L	P	
V						0
L						0
L				0	-1	0
P				5	-1	0
	0	0	0	0	0	0

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n),\text{match} \\ P_{m,n},\text{insertion in database} \\ Q_{m,n},\text{insertion in query} \end{cases}$$

$$D_{m,n} = Max\begin{cases} 10+5 \\ 0 \\ 0 \end{cases} = 15$$

How does it work (D, Q, and P-matrices)

Q works vertically

D

m

n

	V	L	I	L	P	
V						0
L						0
L				15	5	0
P	2	3	4	5	10	0
	0	0	0	0	0	0

Q

	V	L	I	L	P	
V						
L						
L				0	5	0
P	-1	-1	-1	-1	-1	0
	0	0	0	0	0	0

P

	V	L	I	L	P	
V						0
L						0
L				0	-1	0
P				5	-1	0
	0	0	0	0	0	0

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ P_{m,n}, \text{insertion in database} \\ Q_{m,n}, \text{insertion in query} \end{cases}$$

$$D_{m,n} = Max\begin{cases} 10+5 \\ 0 \\ 0 \end{cases} = 15$$

How does it work. Eij-matrix.
(Keeping track on the path)

	V	L	I	L	P	
V						0
L						0
L					2	0
P				4	1	0
	0	0	0	0	0	0

eij = 1 match
eij = 2 insertion open query
(move vertical)
eij = 3 insertion-extension query
(move vertical)
eij = 4 insertion-opening
database
(move horizontal)
eij = 5 insertion-extension
database
(move horizontal)

The fast algorithm (cont.)

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ D_{m+1,n} + w_1, \text{ insertion (opening) in database} \\ P_{m+1,n} + u, \text{insertion (extenton) in database} \\ D_{m,n+1} + w_1, \text{insertion (opening) in query} \\ Q_{m,n+1} + u, \text{insertion (extension) in query} \\ 0 \end{cases}$$

E

1

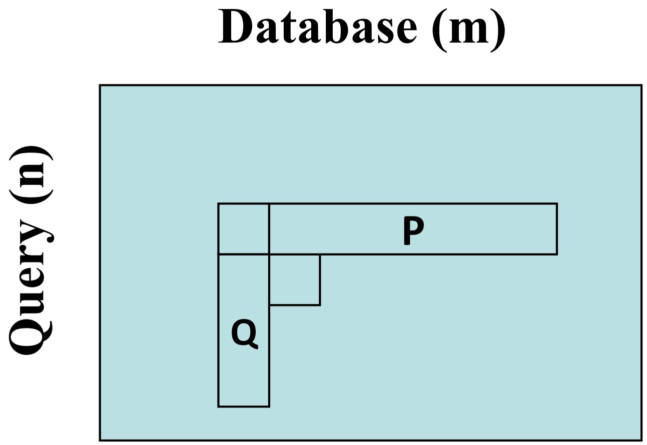
4

5

2

3

eij = 1 match
eij = 2 gap-opening database
eij = 3 gap-extension database
eij = 4 gap-opening query
eij = 5 gap-extension query



How does it work (D,Q,P, and E-matrices)

D

m

n

	V	L	I	L	P	
V						0
L						0
L				15	5	0
P	2	3	4	5	10	0
	0	0	0	0	0	0

Q

	V	L	I	L	P	
V						
L						
L				0	5	0
P	-1	-1	-1	-1	-1	0
	0	0	0	0	0	0

P

	V	L	I	L	P	
V						0
L						0
L				0	-1	0
P				5	-1	0
	0	0	0	0	0	0

E

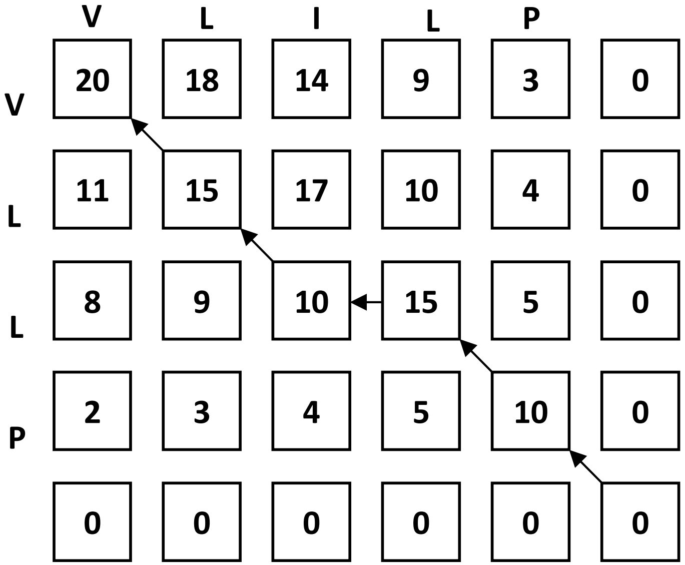
	V	L	I	L	P	
V						0
L						0
L				1	2	0
P				4	1	0
	0	0	0	0	0	0

How does it work (D,Q,P, and E-matrices)

D

m

n



	V	L	I	L	P	
V	20	18	14	9	3	0
L	11	15	17	10	4	0
L	8	9	10	15	5	0
P	2	3	4	5	10	0
	0	0	0	0	0	0

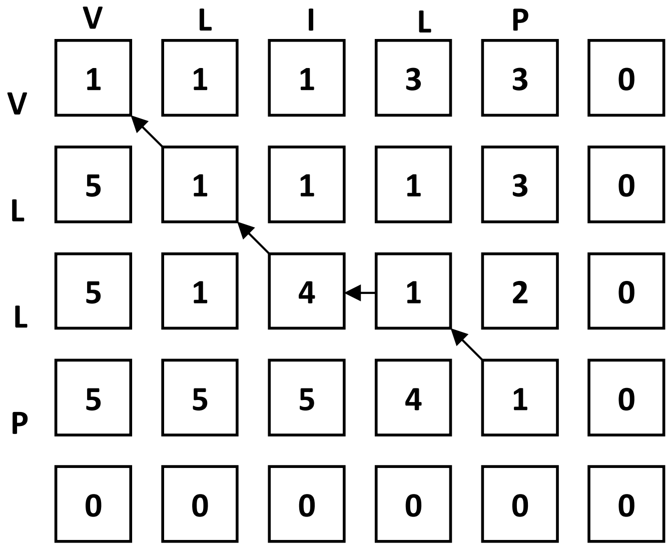
Q

	V	L	I	L	P	
V	6	10	12	9	3	0
L	3	4	5	10	4	0
L	-2	-2	-1	0	5	0
P	-1	-1	-1	-1	-1	0
	0	0	0	0	0	0

P

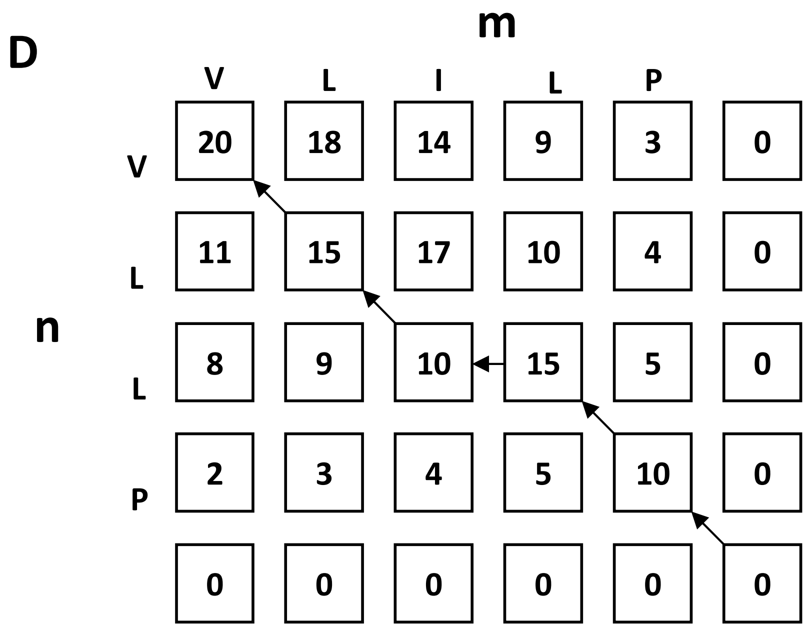
	V	L	I	L	P	
V	13	9	4	-2	-1	0
L	11	12	5	-1	-1	0
L	8	9	10	0	-1	0
P	2	3	4	5	-1	0
	0	0	0	0	0	0

E

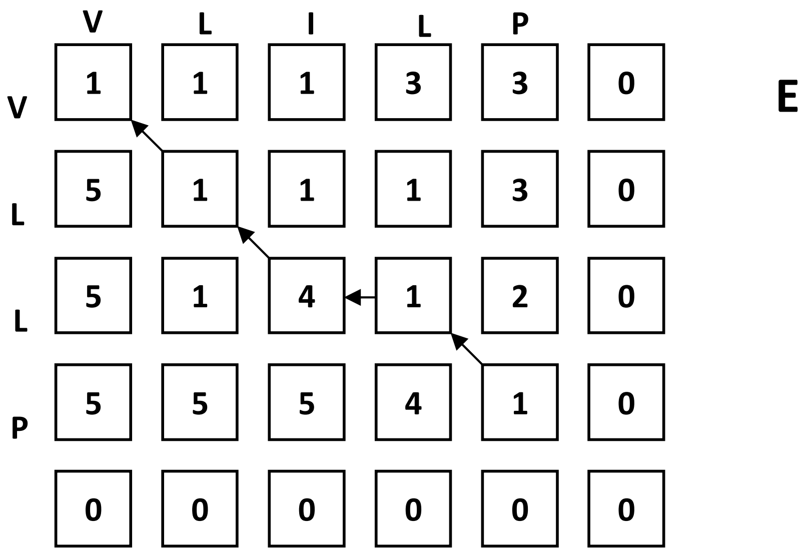


	V	L	I	L	P	
V	1	1	1	3	3	0
L	5	1	1	1	3	0
L	5	1	4	1	2	0
P	5	5	5	4	1	0
	0	0	0	0	0	0

And the alignment



	V	L	I	L	P	
V	20	18	14	9	3	0
L	11	15	17	10	4	0
L	8	9	10	15	5	0
P	2	3	4	5	10	0
	0	0	0	0	0	0

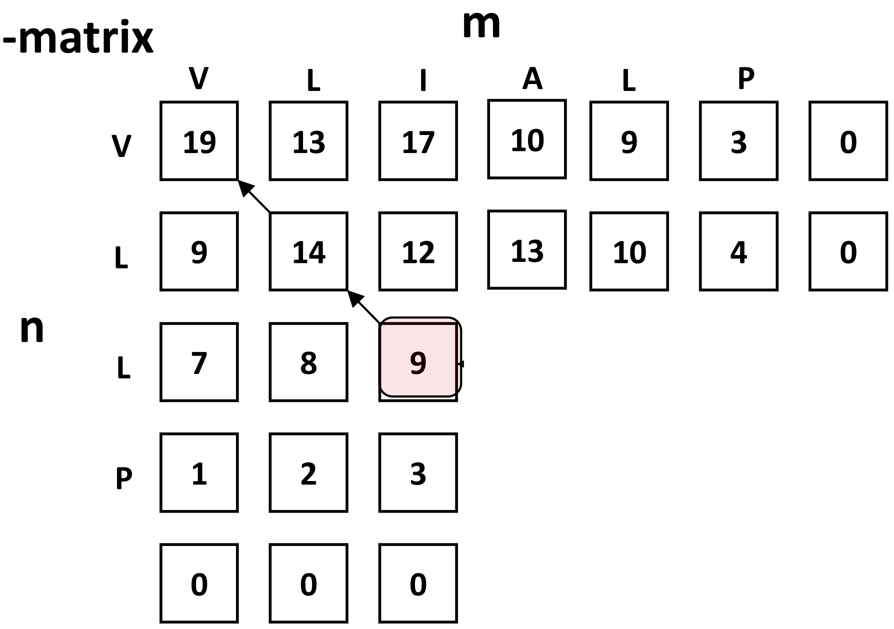


	V	L	I	L	P	
V	1	1	1	3	3	0
L	5	1	1	1	3	0
L	5	1	4	1	2	0
P	5	5	5	4	1	0
	0	0	0	0	0	0

```
VLILP
VL-LP
```

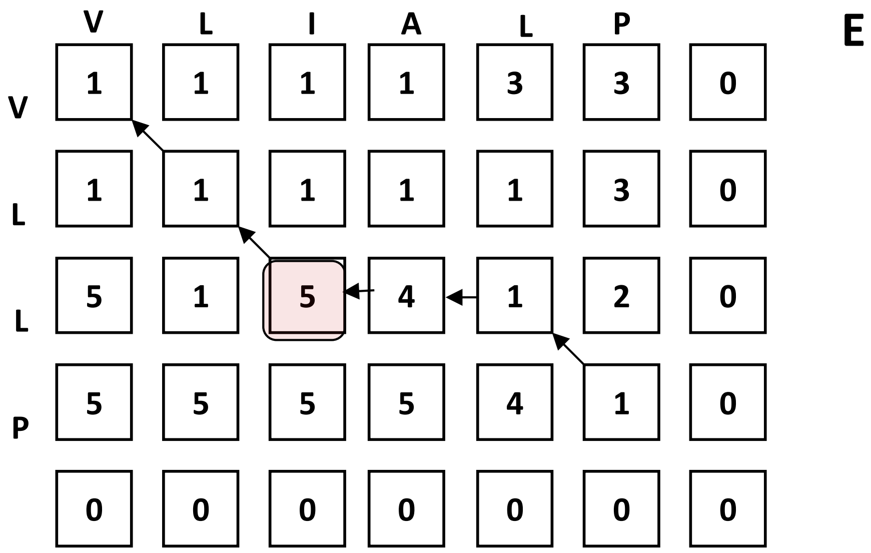
And the alignment. Gap extensions

D-matrix



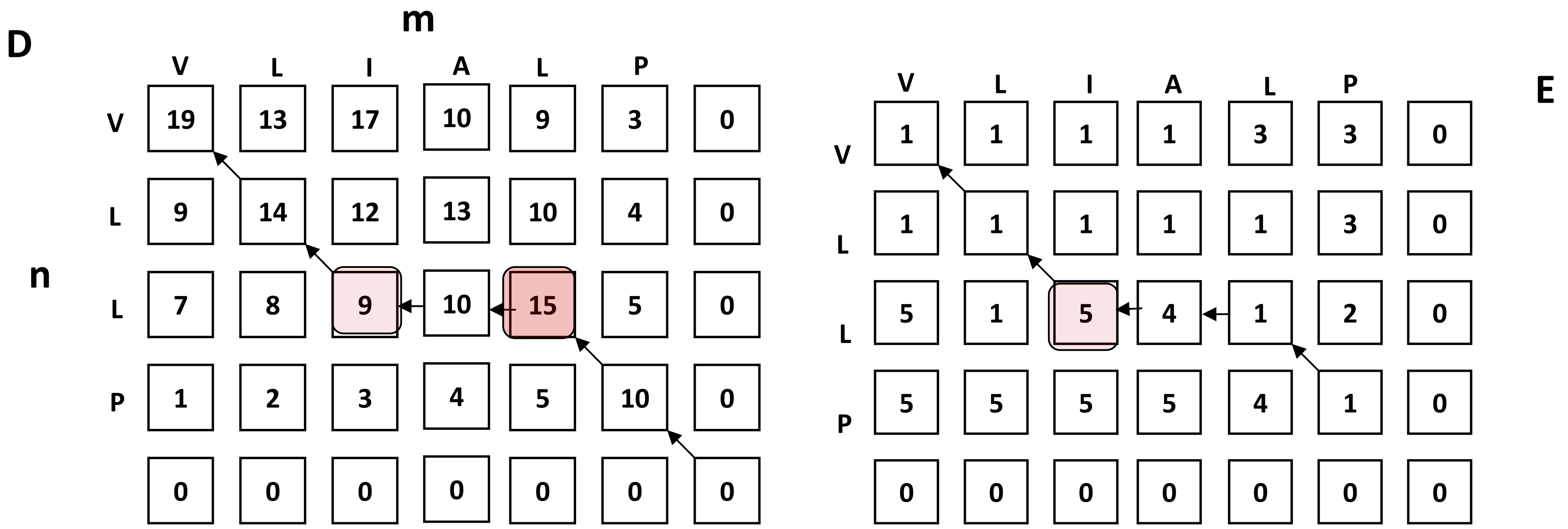
	V	L	I	A	L	P	
V	19	13	17	10	9	3	0
L	9	14	12	13	10	4	0
L	7	8	9				
P	1	2	3				
	0	0	0				

E-matrix



	V	L	I	A	L	P	
V	1	1	1	1	3	3	0
L	1	1	1	1	1	3	0
L	5	1	5	4	1	2	0
P	5	5	5	5	4	1	0
	0	0	0	0	0	0	0

And the alignment

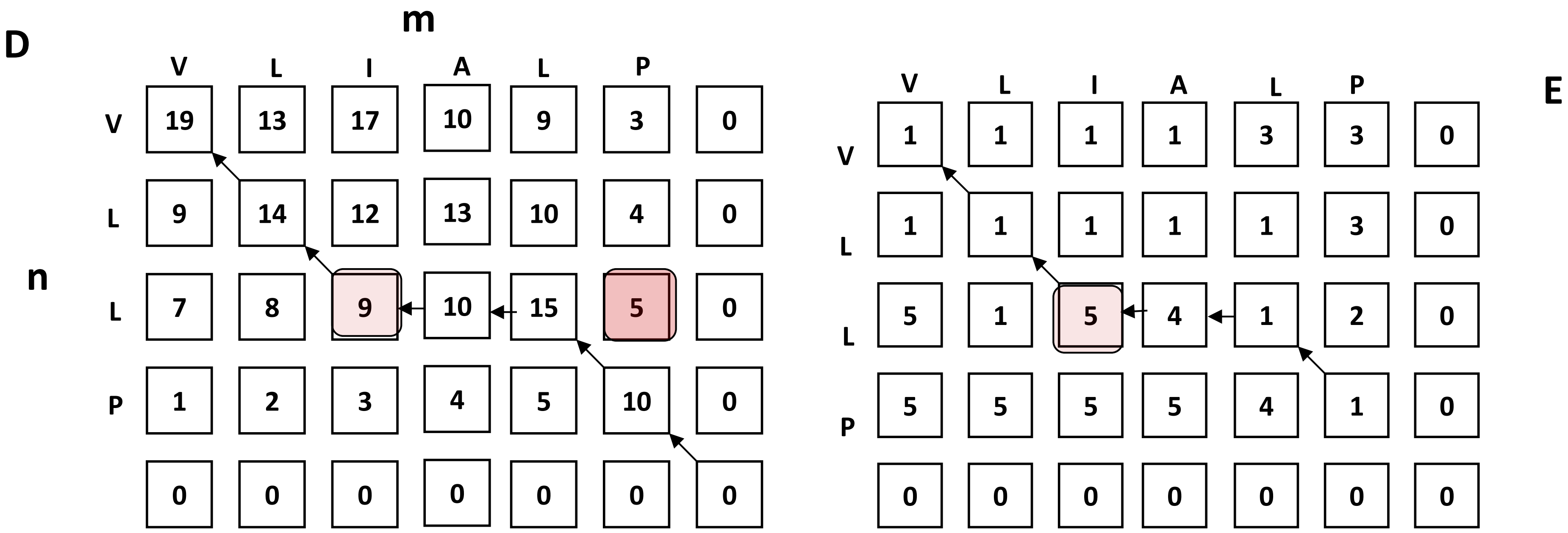


D	V	L	I	A	L	P	
V	19	13	17	10	9	3	0
L	9	14	12	13	10	4	0
L	7	8	9	10	15	5	0
P	1	2	3	4	5	10	0
	0	0	0	0	0	0	0

E	V	L	I	A	L	P	
V	1	1	1	1	3	3	0
L	1	1	1	1	1	3	0
L	5	1	5	4	1	2	0
P	5	5	5	5	4	1	0
	0	0	0	0	0	0	0

15 -5 -1 = 9?

And the alignment



D

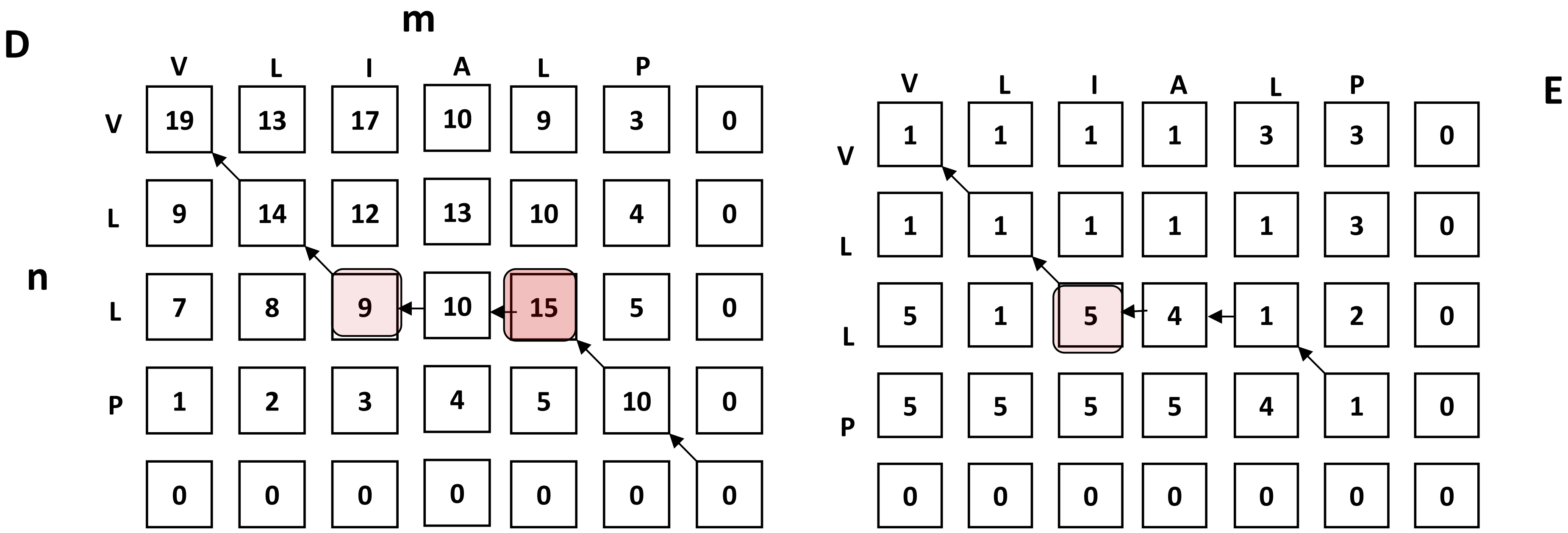
	V	L	I	A	L	P	
V	19	13	17	10	9	3	0
L	9	14	12	13	10	4	0
L	7	8	9	10	15	5	0
P	1	2	3	4	5	10	0
	0	0	0	0	0	0	0

E

	V	L	I	A	L	P	
V	1	1	1	1	3	3	0
L	1	1	1	1	1	3	0
L	5	1	5	4	1	2	0
P	5	5	5	5	4	1	0
	0	0	0	0	0	0	0

5 -5 -1 = 9?

And the alignment



D	V	L	I	A	L	P	
V	19	13	17	10	9	3	0
L	9	14	12	13	10	4	0
L	7	8	9	10	15	5	0
P	1	2	3	4	5	10	0
	0	0	0	0	0	0	0

E	V	L	I	A	L	P	
V	1	1	1	1	3	3	0
L	1	1	1	1	1	3	0
L	5	1	5	4	1	2	0
P	5	5	5	5	4	1	0
	0	0	0	0	0	0	0

```
VLIALP
VL--LP
```

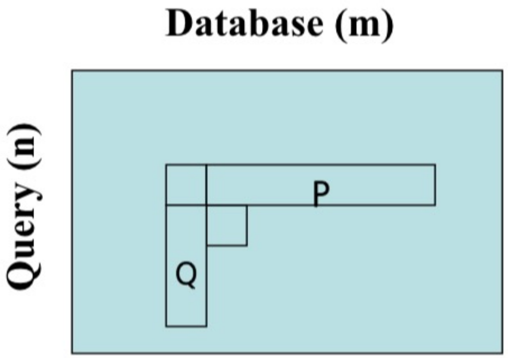
15 -5 -1 = 9? ***

5 -5 -1 = 9?

And now you!

Doing it yourself

$$D_{m,n} = Max\begin{cases} D_{m+1,n+1} + d(m,n), \text{match} \\ P_{m,n}, \text{gap in query} \\ Q_{m,n}, \text{gap in database} \\ 0 \text{ (for local alignment)} \end{cases}$$

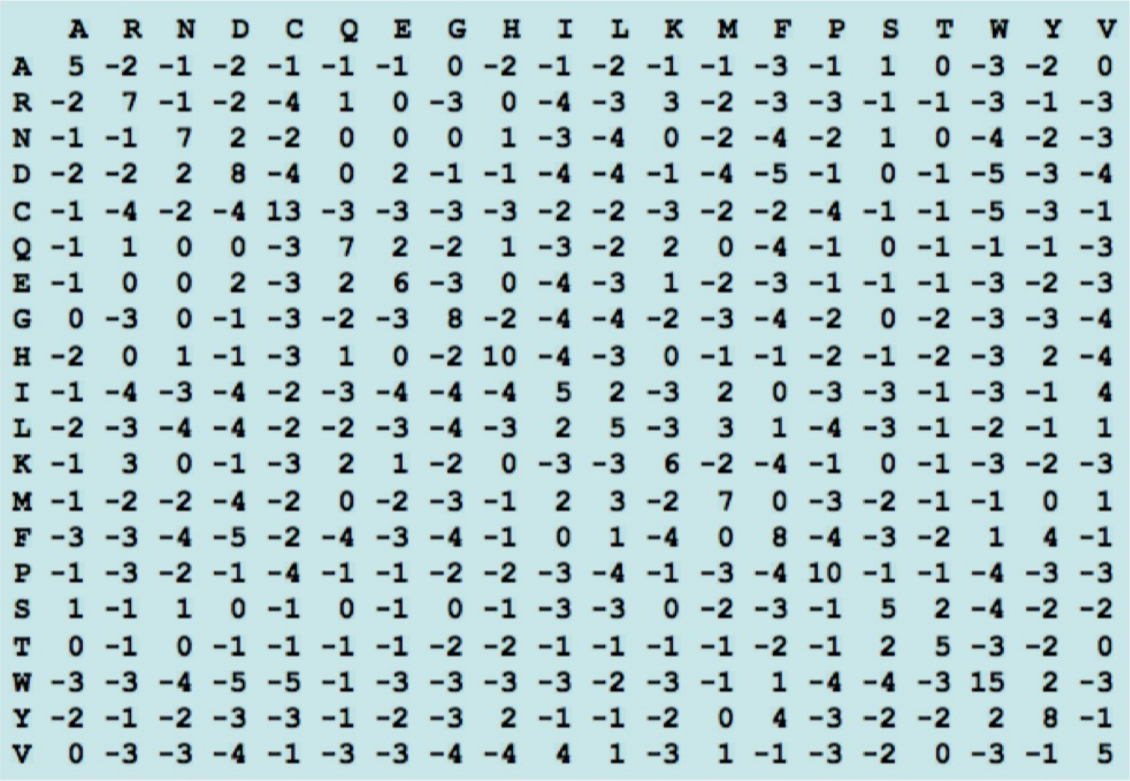


$$P_{m,n} = Max\left[D_{m+1,n} + w_1, P_{m+1,n} + u\right]$$
$$Q_{m,n} = Max\left[D_{m,n+1} + w_1, Q_{m,n+1} + u\right]$$

eij = 1 match
eij = 2 gap-opening database
eij = 3 gap-extension database
eij = 4 gap-opening query
eij = 5 gap-extension query

D is highest
Q is highest, gap open
Q is highest, gap extension
P is highest, gap open
P is highest, gap extension

Blosum50 scoring matrix



	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5

$W_1 = -2$
$U = -1$

How does it work (score matrix d)

	V	L	L	P	V	L	L	P	
V	5	1	1	-3	5	1	1	-3	
L	1	5	5	-4	1	5	5	-4	
P			-4		-3		-4	10	
V	5	1	1	-3	5	1	1	-3	n
L	1	5	5	-4	1	5	5	-4	
I	4		2	-3		2	2	-3	
L	1	5	5	-4	1	5		-4	
P	-3	-4	-4	10	-3	-4	-4	10	
				m					

D

	V	L	L	P	V	L	L	P	
V	41	39	36	30	20	16	13	7	0
L	35	36	38	31	20	15	15	8	0
P	29	30		33	21	15	10	10	0
V	24	21	20	21	23	16		5	0
L	18	19	20	16	18	18	12	6	0
I	17	17		15	17	17	13	7	0
L	12	13	15	11	12	13		8	0
P	6	7	8	10	6	7	8	10	0
	0	0	0	0	0	0	0	0	0

n

m

Q

	V	L	L	P	V	L	L	P	
V	33	34		30	19	13	13	7	0
L	27	28	29	31	20	14	9	8	0
P	22	19	18	19	21		10	4	0
V	16	17	18	14	16	16	11	5	0
L	15		12	13	15	15	12	6	0
I	10	11	13	9	10		13	7	0
L	4	5	6	8	4	5	6	8	0
P	-1	-1	-1	-1	-1	-1	-1	-1	0
	0	0	0	0	0	0	0	0	0

n

m

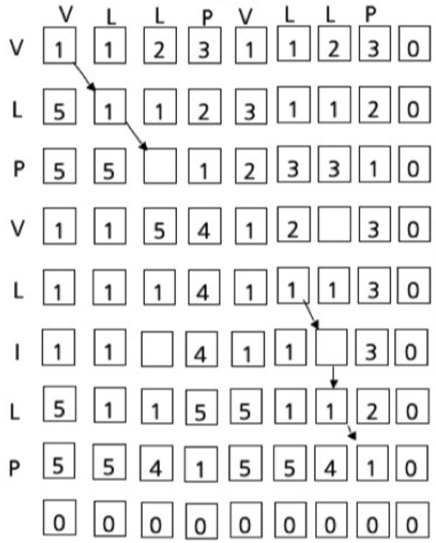
P

	V	L	L	P	V	L	L	P	
V	37	34	28	18	14	11	5	-1	0
L		36	29	18	13	13	6	-1	0
P	29	30	31	19	13		8	-1	0
V	19	19	20	21	14	9	3	-1	0
L	17		15	16	16	10	4	-1	0
I	15	13	14	15	15	11	5	-1	0
L	12	13	10	11	12	13	6	-1	0
P	6	7		5	6	7	8	-1	0
	0	0	0	0	0	0	0	0	0

n

m

E



	V	L	L	P	V	L	L	P	
V	1	1	2	3	1	1	2	3	0
L	5	1	1	2	3	1	1	2	0
P	5	5		1	2	3	3	1	0
V	1	1	5	4	1	2		3	0
L	1	1	1	4	1	1	1	3	0
I	1	1		4	1	1		3	0
L	5	1	1	5	5	1	1	2	0
P	5	5	4	1	5	5	4	1	0
	0	0	0	0	0	0	0	0	0

$W_1 = -2$
$U = -1$

eij = 1 match
eij = 2 gap-opening database
eij = 3 gap-extension database
eij = 4 gap-opening query
eij = 5 gap-extension query

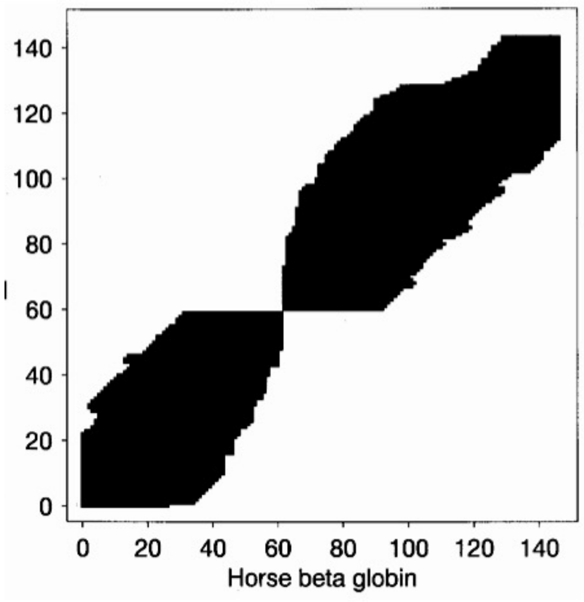
The alignment

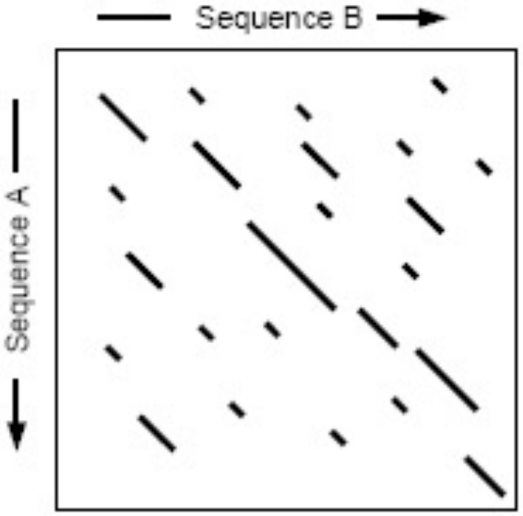
```
ALN score:
QAL
DAL
```

Query sequence alignment
Database sequence alignment

Summary

- Alignment is more complicated than what you have been told.
- Simple algorithmic tricks allow for alignment in O2 time
- More heuristics to improve speed
  - Limit gap length
  - Look for high scoring regions





Find runs of identities