Pairwise Sequence Alignment

BMI/CS 576
www.biostat.wisc.edu/bmi576/
Mark Craven
craven@biostat.wisc.edu
Fall 2011

Pairwise alignment: task definition

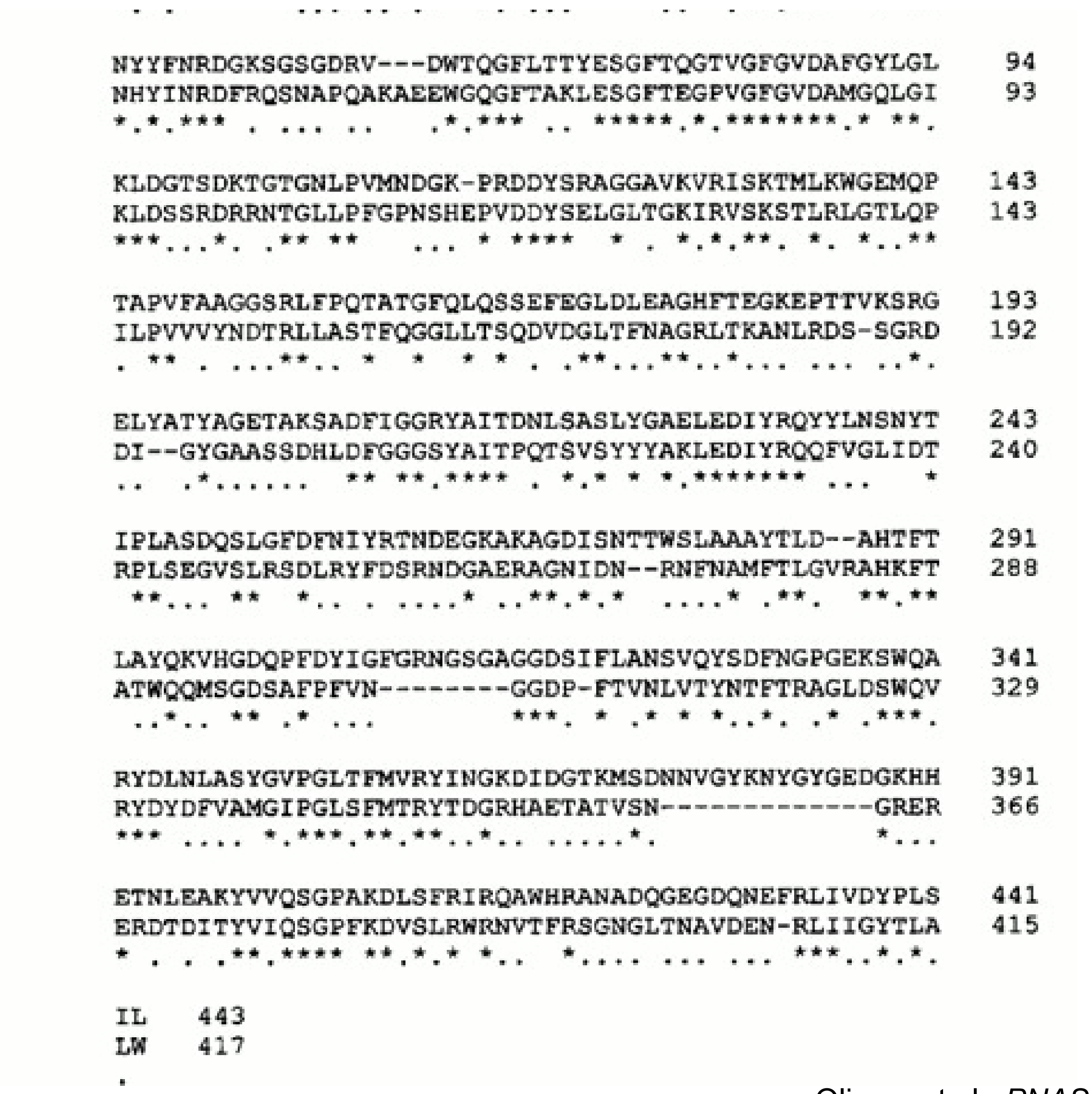
Given

- –a pair of sequences (DNA or protein)
- –a method for scoring a candidate alignment

Do

- –determine the correspondences between substrings in the sequences such that the similarity score is maximized

Protein alignment example



```
 . .       ... .. .    . ...     ..  .  .

NYYFNRDGKSGSGDRV---DWTQGFLTTYESGFTQGTVGFGVDAFGYLGL     94
NHYINRDFRQSNAPQAKAEEWGQGFTAKLESGFTEGPVGFGVDAMGQLGI     93
*.*.*** . ... ..   .*.*** .. *****.*.*******.* **.

KLDGTSDKTGTGNLPVMNDGK-PRDDYSRAGGAVKVRISKTMLKWGEMQP    143
KLDSSRDRRNTGLLPFGPNSHEPVDDYSELGLTGKIRVSKSTLRLGTLQP    143
***...*. .** **   ... * ****  * . *.*.**. *. *..**

TAPVFAAGGSRLFPQTATGFQLQSSEFEGLDLEAGHFTEGKEPTTVKSRG    193
ILPVVVYNDTRLLASTFQGGLLTSQDVDGLTFNAGRLTKANLRDS-SGRD    192
. ** . ...**.. *  *  * * . .**...**..*... ... ..*.

ELYATYAGETAKSADFIGGRYAITDNLSASLYGAELEDIYRQYYLNSNYT    243
DI--GYGAASSDHLDFGGGSYAITPQTSVSYYYAKLEDIYRQQFVGLIDT    240
..  .*......  ** **.**** . *.* * *.******* ...   *

IPLASDQSLGFDFNIYRTNDEGKAKAGDISNTTWSLAAAYTLD--AHTFT    291
RPLSEGVSLRSDLRYFDSRNDGAERAGNIDN--RNFNAMFTLGVRAHKFT    288
 **... **  *.. . ....* ..**.*.*  ....* .**.  **.**

LAYQKVHGDQPFDYIGFGRNGSGAGGDSIFLANSVQYSDFNGPGEKSWQA    341
ATWQQMSGDSAFPFVN--------GGDP-FTVNLVTYNTFTRAGLDSWQV    329
 ..*.. ** .* ...        ***. * .* * *..*. .* .***.

RYDLNLASYGVPGLTFMVRYINGKDIDGTKMSDNNVGYKNYGYGEDGKHH    391
RYDYDFVAMGIPGLSFMTRYTDGRHAETATVSN-------------GRER    366
*** .... *.***.**.**..*.. .....*.             *...

ETNLEAKYVVQSGPAKDLSFRIRQAWHRANADQGEGDQNEFRLIVDYPLS    441
ERDTDITYVIQSGPFKDVSLRWRNVTFRSGNGLTNAVDEN-RLIIGYTLA    415
* . . .**.**** **.*.* *..  *.... ... ... ***..*.*.

IL    443
LW    417
.
```

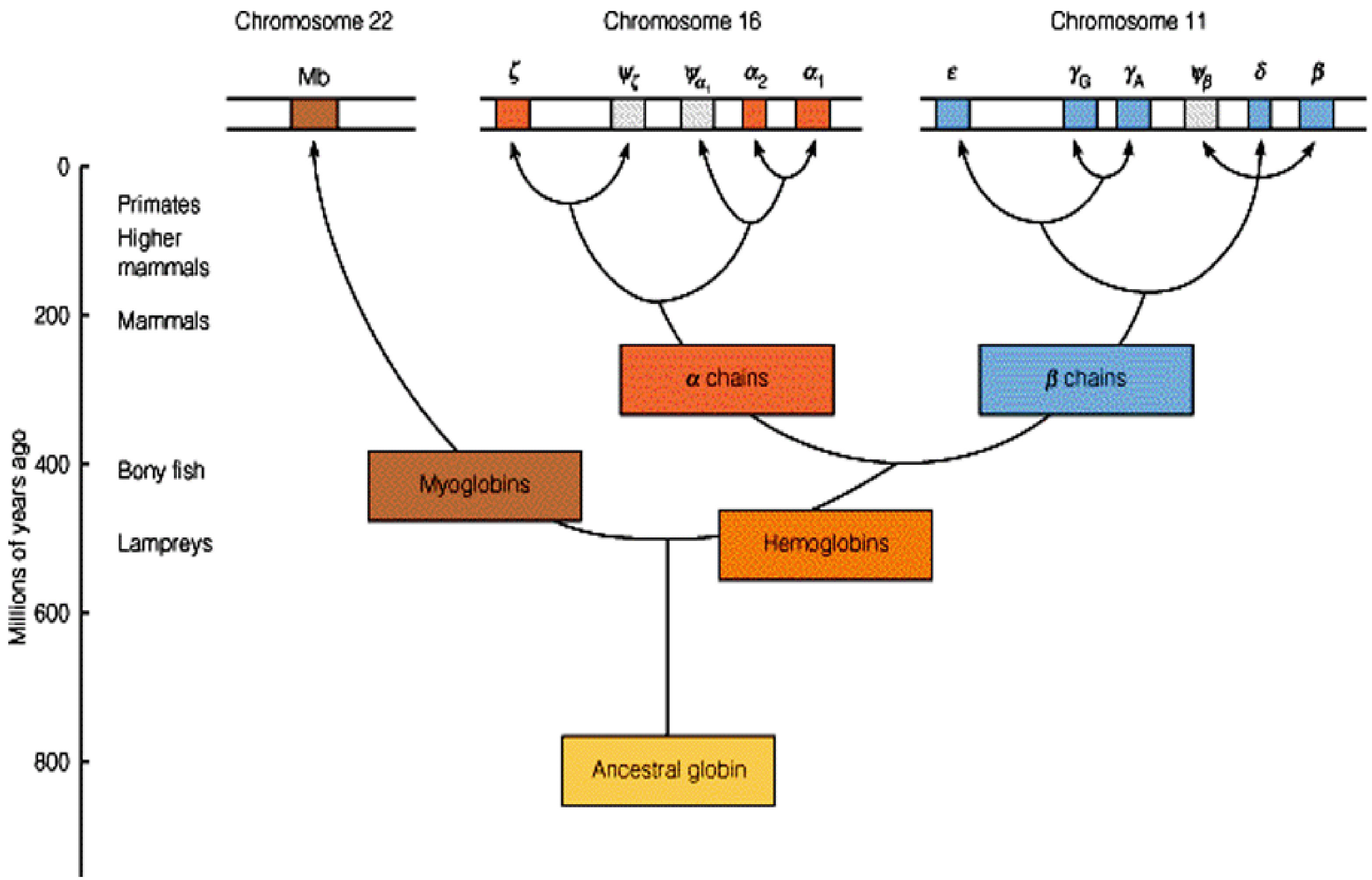
Alignment of the PhaK protein from Pseudomonas putida and OprD protein from Pseudomonas aeruginos

Olivera et al., *PNAS* 95:6419-6424, 1998

The role of homology in alignment

- *homology*: similarity due to descent from a common ancestor
- often we can infer homology from similarity
- thus we can sometimes infer structure/function from sequence similarity

Homology example: evolution of the globins

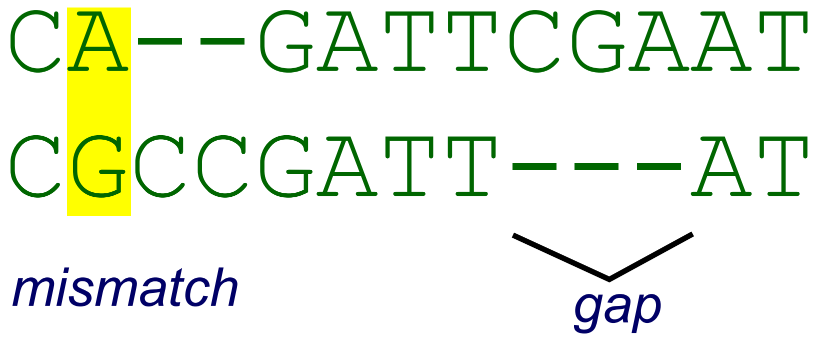


Homology

- homologous sequences can be divided into two groups
  - *orthologous sequences*: sequences that differ because they are found in different species (e.g. human α-globin and mouse α-globin)
  - *paralogous sequences*: sequences that differ because of a gene duplication event (e.g. human α-globin and human β-globin, various versions of both)

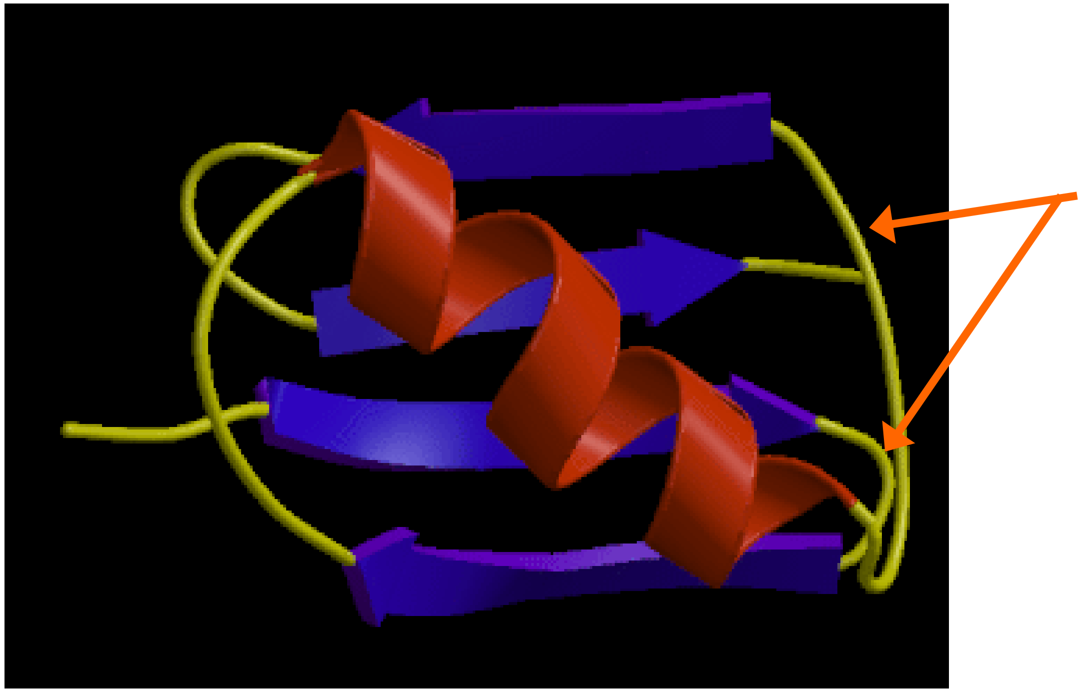
Mismatches and gaps

- substitutions in *homologous* sequences result in mismatches in an alignment
- insertions/deletions in *homologous* sequences result in gaps in an alignment



Insertions/deletions and protein structure

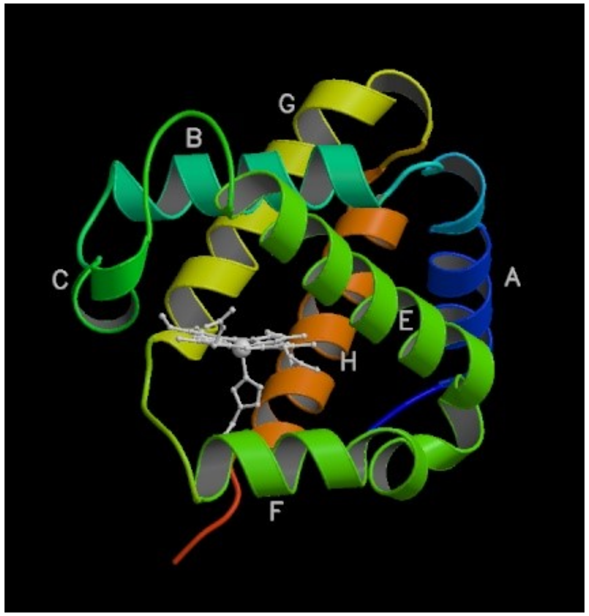
- Why is it that two "similar" sequences may have large insertions/deletions?
  - some insertions and deletions may not significantly affect the structure of a protein

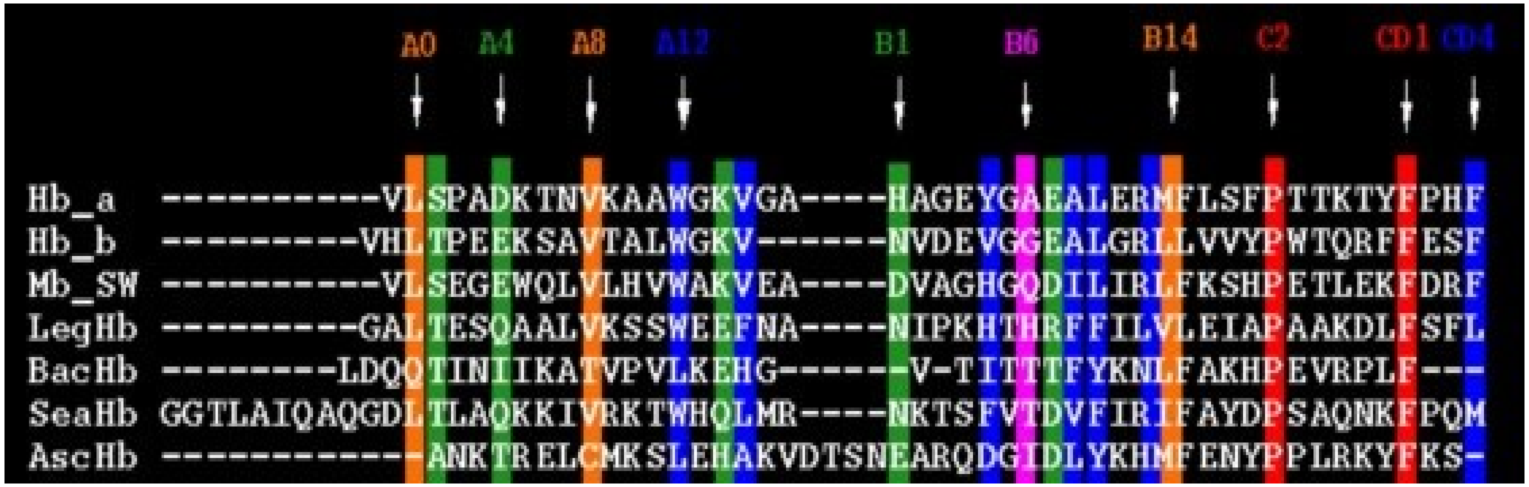


loop structures: insertions/deletions here not so significant

Example alignment: globins

- Right: prototypical structure of globins
- Below: partial alignment for 8 globins





```
                    A0  A4   A8   A12        B1     B6     B14   C2     CD1 CD4
Hb_a   ----------VLSPADKTNVKAAWGKVGA----HAGEYGAEALERMFLSFPTTKTYFPHF
Hb_b   ---------VHLTPEEKSAVTALWGKV------NVDEVGGEALGRLLVVYPWTQRFFESF
Mb_SW  ----------VLSEGEWQLVLHVWAKVEA----DVAGHGQDILIRLFKSHPETLEKFDRF
LegHb  ---------GALTESQAALVKSSWEEFNA----NIPKHTHRFFILVLEIAPAAKDLFSFL
BacHb  --------LDQQTINIIKATVPVLKEHG------V-TITTTFYKNLFAKHPEVRPLF---
SeaHb  GGTLAIQAQGDLTLAQKKIVRKTWHQLMR----NKTSFVTDVFIRIFAYDPSAQNKFPQM
AscHb  ------------ANKTRELCMKSLEHAKVDTSNEARQDGIDLYKHMFENYPPLRKYFKS-
```

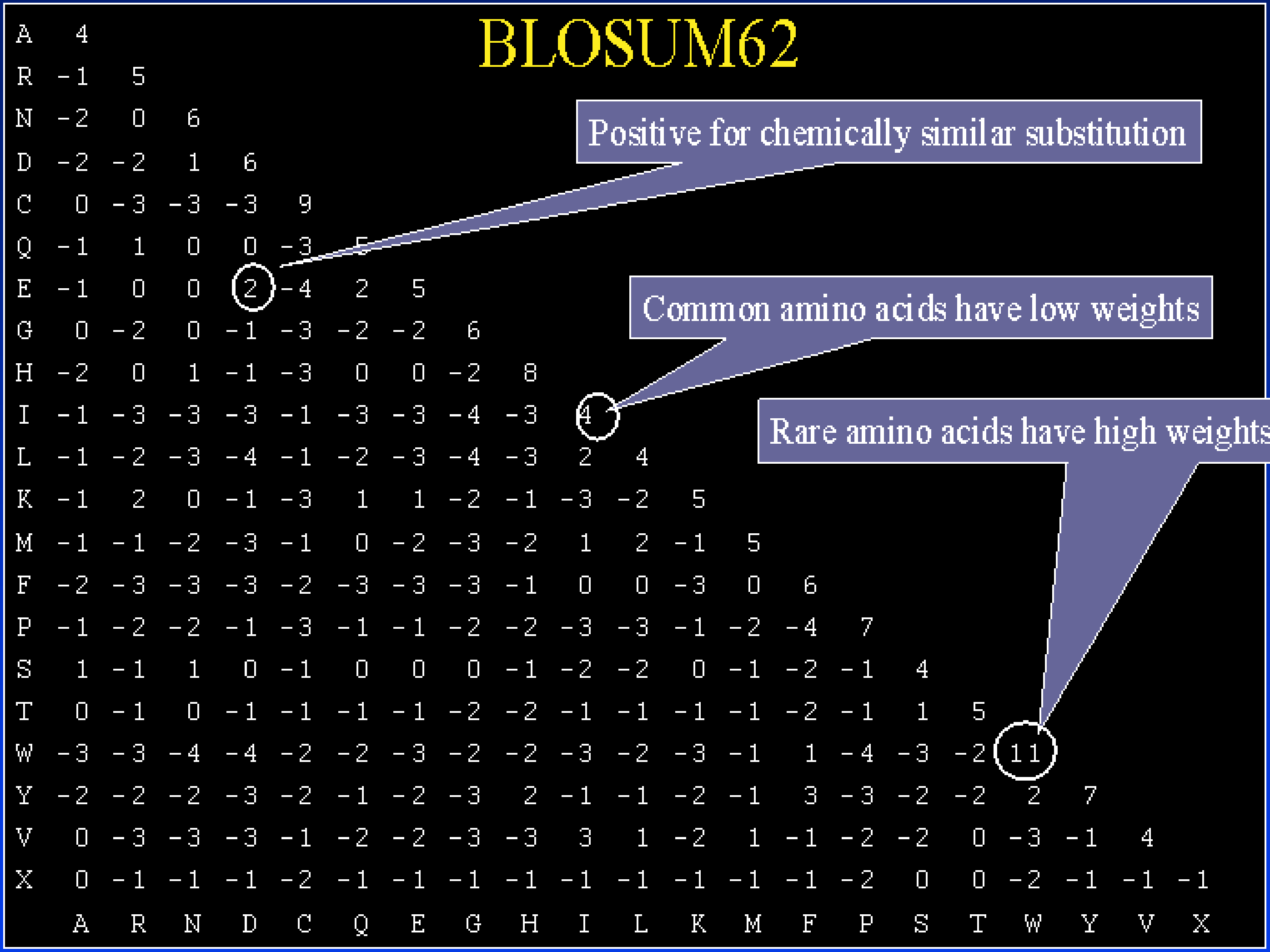
Types of alignment

- *global*: find best match of both sequences in their entirety
- *local*: find best subsequence match
- *semi-global*: find best match without penalizing gaps on the ends of the alignment

Scoring an alignment: what is needed?

- substitution matrix
  - *s(a,b)* indicates score of aligning character *a* with character *b*
- gap penalty function
  - *w(g)* indicates cost of a gap of length *g*

Blosum 62 substitution matrix



BLOSUM62

Positive for chemically similar substitution

Common amino acids have low weights

Rare amino acids have high weights

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

Linear gap penalty function

- different gap penalty functions require somewhat different dynamic programming algorithms
- the simplest case is when a linear gap function is used

$$w(g) = -g \times d$$

- where d is a constant
- we'll start by considering this case

Scoring an alignment

•the score of an alignment is the sum of the scores for pairs of aligned characters plus the scores for gaps

•example: given the following alignment

```
□VAHV---D--DMPNALSALSDLHAHKL
□AIQLQVTGVVVTDATLKNLGSVHVSKG
```

□we would score it by
s(**V**,**A**) + s(**A**,**I**) + s(**H**,**Q**) + s(**V**,**L**) – 3*d* + s(**D**,**G**) – 2d …

The space of global alignments

•some possible global alignments for `ELV` and `VIS`

```
ELV        -ELV        --ELV        ELV-
VIS        VIS-        VIS--        -VIS


E-LV        ELV--       EL-V
VIS-        --VIS       -VIS
```

•Can we find the highest scoring alignment by enumerating all possible alignments and picking the best?

Number of possible alignments

- given sequences of length *m* and *n*
- assume we don't count as distinct $\begin{matrix}\texttt{C-}\\\texttt{-G}\end{matrix}$ and $\begin{matrix}\texttt{-C}\\\texttt{G-}\end{matrix}$
- we can have as few as 0 and as many as min{*m*, *n*} aligned pairs
- therefore the number of possible alignments is given by

$$\sum_{k=0}^{\min\{m,n\}} \binom{n}{k}\binom{m}{k} = \binom{n+m}{n}$$

k: the number of exact matches in an alignment

Number of possible alignments

•there are

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \approx \frac{2^{2n}}{\sqrt{\pi n}}$$

possible global alignments for 2 sequences of length *n*

e.g. two sequences of length 100 have $\approx 10^{59}$ possible alignments

but we can use *dynamic programming* to find an optimal alignment efficiently

Pairwise alignment via dynamic programming

- first algorithm by Needleman & Wunsch, *Journal of Molecular Biology*, 1970

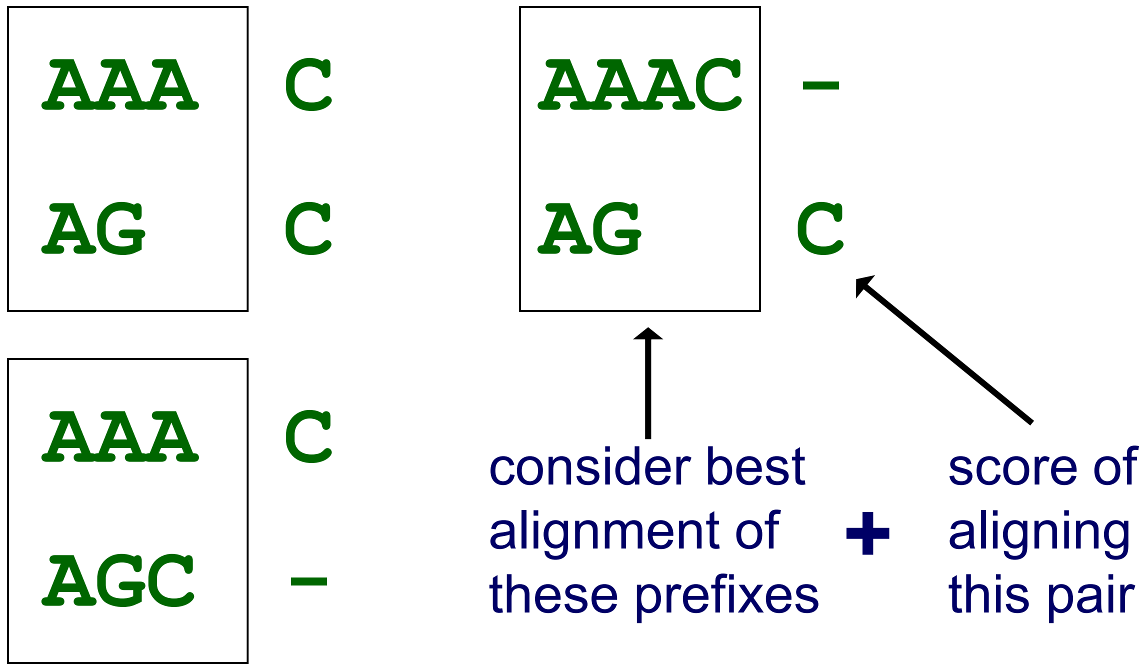
- *dynamic programming*: solve an instance of a problem by taking advantage of computed solutions for smaller subparts of the problem

- determine best alignment of two sequences by determining best alignment of all prefixes of the sequences

Dynamic programming idea

- consider last step in computing alignment of **AAAC** with **AGC**

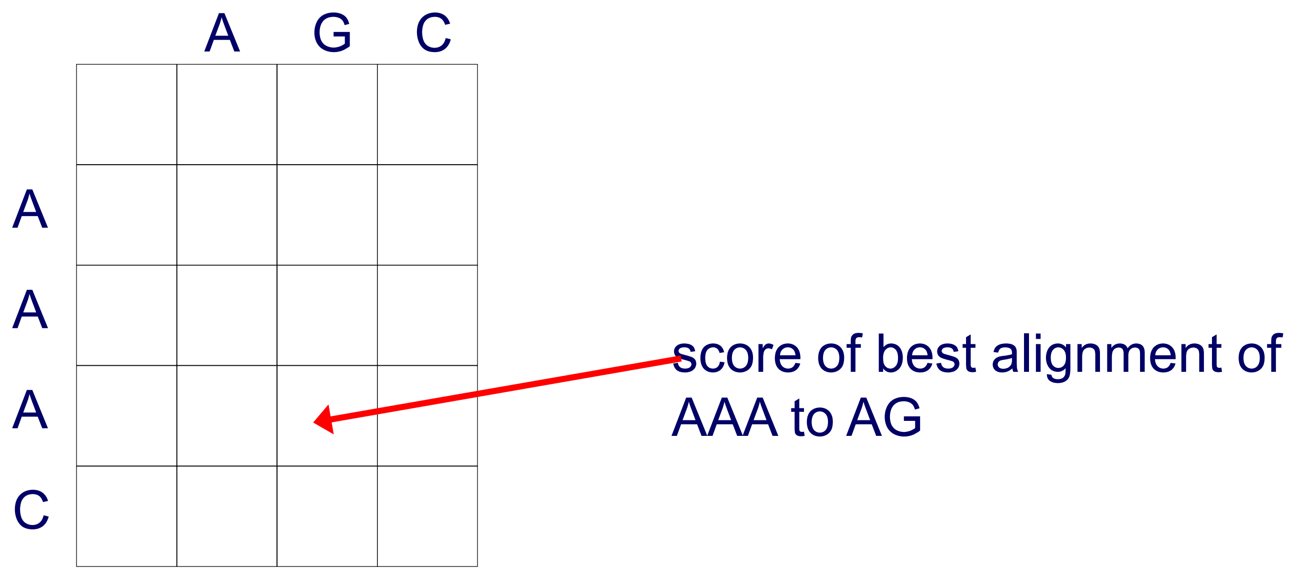
- three possible options; in each we'll choose a different pairing for end of alignment, and add this to best alignment of previous characters



AAA C
AG C
AAAC –
AG C
AAA C
AGC –
consider best alignment of these prefixes
+
score of aligning this pair

Dynamic programming idea

- given an *n*-character sequence *x*, and an *m*-character sequence *y*
- construct an $(n+1) \times (m+1)$ matrix *F*
- $F(i, j)$ = score of the best alignment of $x[1\ldots i]$ with $y[1\ldots j]$



DP algorithm for global alignment with linear gap penalty

- one way to specify the DP is in terms of its recurrence relation:

$$F(i,j)=\max\begin{cases} F(i-1,j-1)+s(x_i,y_j) \\ F(i-1,j)-d \\ F(i,j-1)-d \end{cases}$$

Initializing matrix: global alignment with linear gap penalty

$$\begin{tikzcd}
 & & A & G & C \\
 & 0 & -d \arrow[l] & -2d \arrow[l] & -3d \arrow[l] \\
A & -d \arrow[u] & & & \\
A & -2d \arrow[u] & & & \\
A & -3d \arrow[u] & & & \\
C & -4d \arrow[u] & & &
\end{tikzcd}$$

DP algorithm sketch: global alignment

- initialize first row and column of matrix
- fill in rest of matrix from top to bottom, left to right
- for each $F(i, j)$, save pointer(s) to cell(s) that resulted in best score
- $F(m, n)$ holds the optimal alignment score; trace pointers back from $F(m, n)$ to $F(0, 0)$ to recover alignment

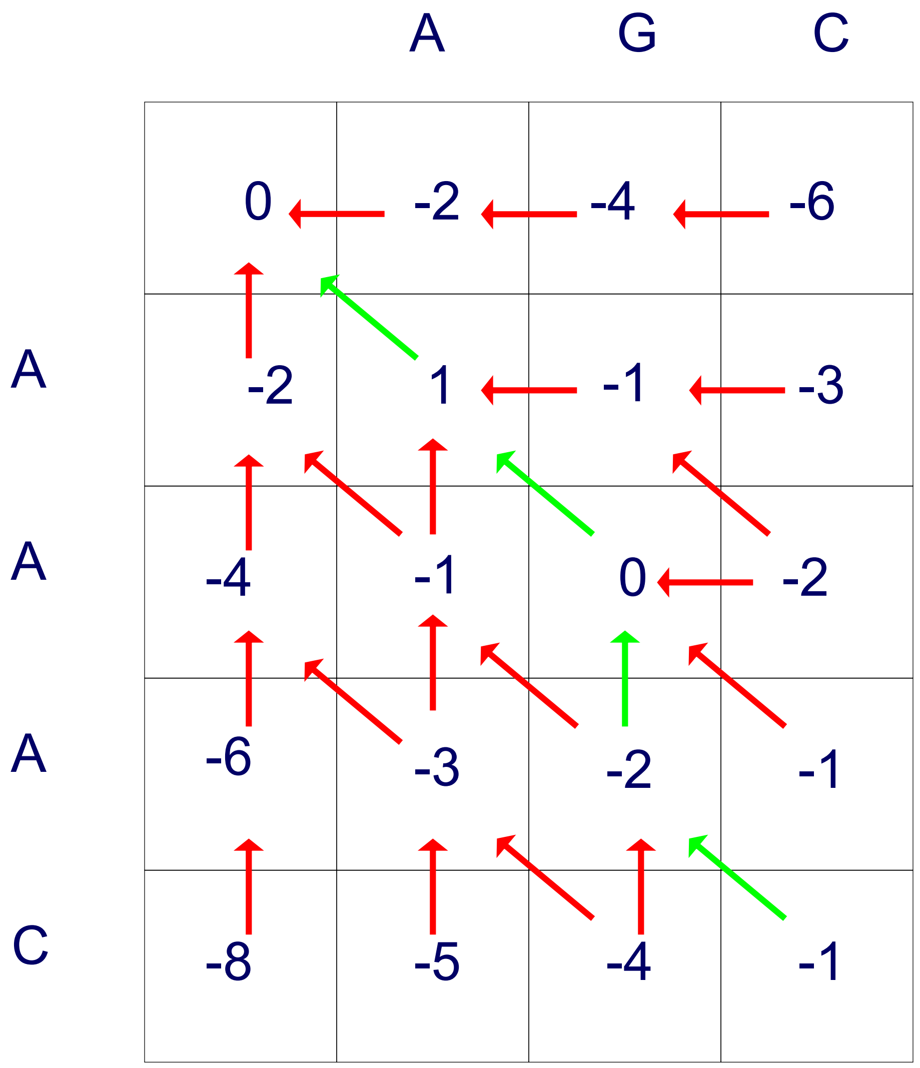
Global alignment example

- suppose we choose the following scoring scheme:

$$s(x_i, y_i) = \begin{cases} +1 & \text{when } x_i = y_i \\ -1 & \text{when } x_i \neq y_i \end{cases}$$

d (penalty for aligning with a gap) = 2

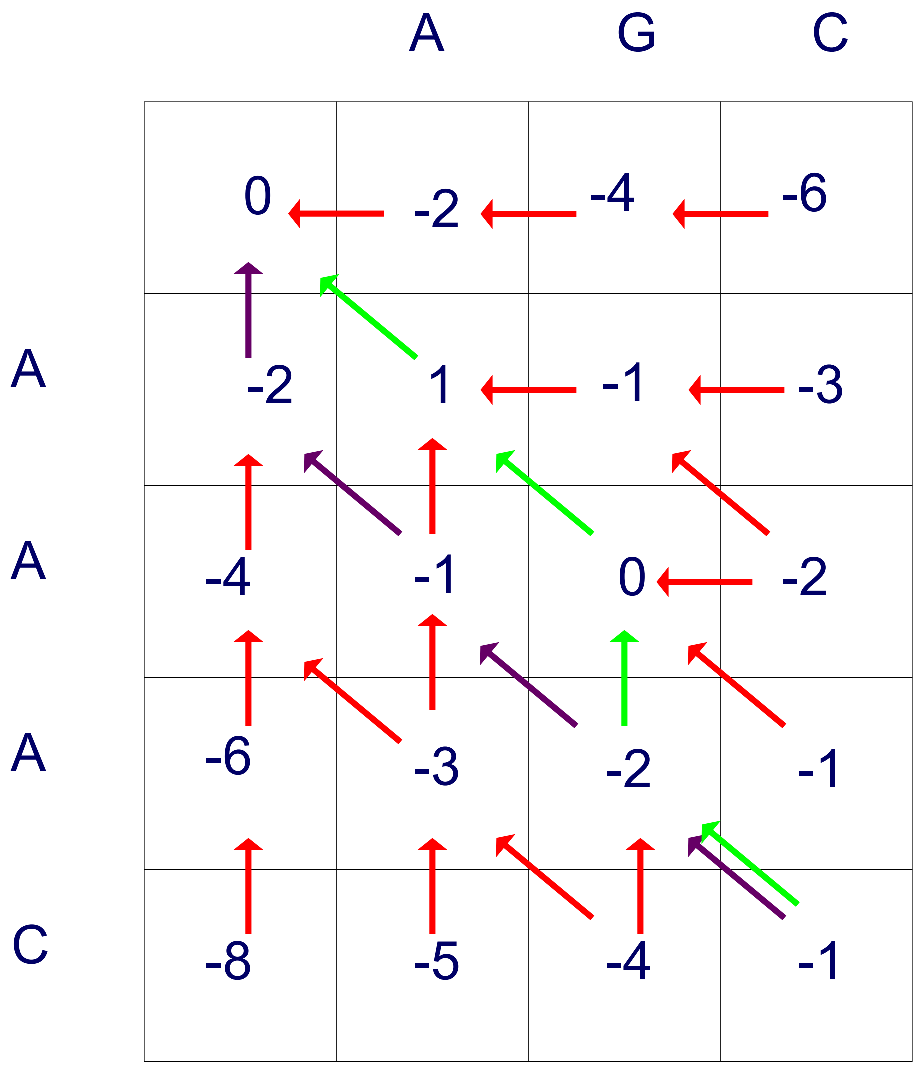
Global alignment example



<u>one optimal alignment</u>

x:	A	A	A	C
y:	A	G	-	C

Highroad & lowroad alignments



<u>highroad alignment</u>

x:	A	A	A	C
y:	A	G	-	C

<u>lowroad alignment</u>

x:	A	A	A	C
y:	-	A	G	C

Computational complexity

- initialization: $O(m)$, $O(n)$ where sequence lengths are m, n
- filling in rest of matrix: $O(mn)$
- traceback: $O(m + n)$
- hence, if sequences have nearly same length, the computational complexity is

$$O(n^2)$$

Related problems solved by DP

- **Local** alignment
  - the best match between <u>subsequences</u> of *x* and *y*
  - so far we have discussed *global alignment*, where we are looking for best match between sequences from one end to the other
- More realistic **gap functions**
  - a gap of length *k* is more probable than *k* gaps of length 1
  - a gap may be due to a single mutational event that inserted/deleted a stretch of characters
  - separated gaps are probably due to distinct mutational events

Local alignment

- Motivation
  - a common *motif* (conserved pattern) or *domain* (independently folded unit) but differ elsewhere
  - more sensitive when comparing highly diverged sequences
- Original formulation
  - Smith & Waterman, *Journal of Mol. Biology*, 1981
- Implementation
  - the recurrence relation is slightly different from global alignment
    - maximize also with 0
    - begins and ends anywhere

Local alignment motivation

- useful for comparing protein sequences that share a common *motif* (conserved pattern) or *domain* (independently folded unit) but differ elsewhere

- useful for comparing DNA sequences that share a similar *motif* but differ elsewhere

- useful for comparing protein sequences against *genomic DNA sequences* (long stretches of uncharacterized sequence)

- more sensitive when comparing highly diverged sequences

Example local alignment

- aligning “Mark Craven” against the sequence for dTDP-4-dehydrorhamnose reductase from the bacterium *opitutus terrae*

```
                             MARKCRAVEN
…LSGAYHLAASGHTSWHGFASAIIDLMPLDARKCRAVEAIT…
```

Local alignment DP algorithm

- original formulation: Smith & Waterman, *Journal of Molecular Biology*, 1981

- interpretation of array values is somewhat different:
- $F(i, j)$ = score of the best alignment between
  - <u>a suffix of</u> $x[1\ldots i]$

    and
  - <u>a suffix of</u> $y[1\ldots j]$

Local alignment DP algorithm

- the recurrence relation is slightly different than for global algorithm

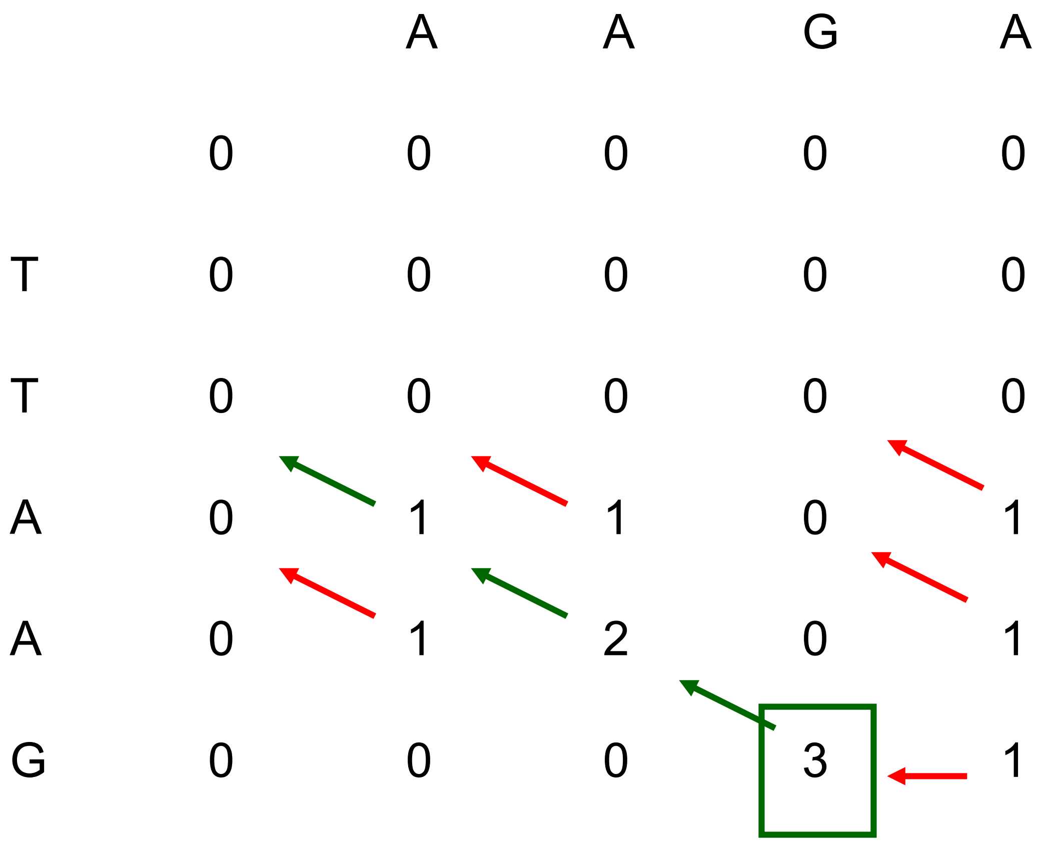
$$F(i,j) = \max\begin{cases} F(i-1,j-1)+s(x_i,y_j) \\ F(i-1,j)-d \\ F(i,j-1)-d \\ 0 \end{cases}$$

Local alignment DP algorithm

- initialization: first row and first column initialized with 0’s

- traceback:
    - find maximum value of *F(i, j)*; can be anywhere in matrix
    - stop when we get to a cell with value 0

Local alignment example



		A	A	G	A
	0	0	0	0	0
T	0	0	0	0	0
T	0	0	0	0	0
A	0	1	1	0	1
A	0	1	2	0	1
G	0	0	0	$\boxed{3}$	1

x: A A G

y: A A G

Gap penalty functions

- linear: $$w(g) = -g \times d$$

- affine: $$w(g) = \begin{cases} -d - (g-1)e, & g \geq 1 \\ 0, & g = 0 \end{cases}$$

- convex: as gap length increases, magnitude of penalty for each additional character decreases

e.g.

$$w(g) = -d - \log(g) \times e$$

More on gap penalty functions

- a gap of length *k* is more probable than *k* gaps of length 1
  - a gap may be due to a single mutational event that inserted/deleted a stretch of characters
  - separated gaps are probably due to distinct mutational events

- a linear gap penalty function treats these cases the same

- it is more common to use gap penalty functions involving two terms
  - a penalty *d* associated with <u>opening</u> a gap
  - a smaller penalty *e* for <u>extending</u> the gap

Dynamic programming for the affine gap penalty case

- to do in [redacted] time, need 3 matrices instead of 1

[redacted] best score given that *x*[*i*] is aligned to *y*[*j*]

[redacted] best score given that *x*[*i*] is aligned to a gap

[redacted] best score given that *y*[*j*] is aligned to a gap

Global alignment DP for the affine gap penalty case

$$M(i,j) = \max \begin{cases} M(i-1,j-1) + s(x_i,y_j) \\ I_x(i-1,j-1) + s(x_i,y_j) \\ I_y(i-1,j-1) + s(x_i,y_j) \end{cases}$$

[redacted]

[redacted]

Note: This set of recurrence equations does not lead to optimality in all situations. Can you update it to be always optimal?

Global alignment DP for the affine gap penalty case

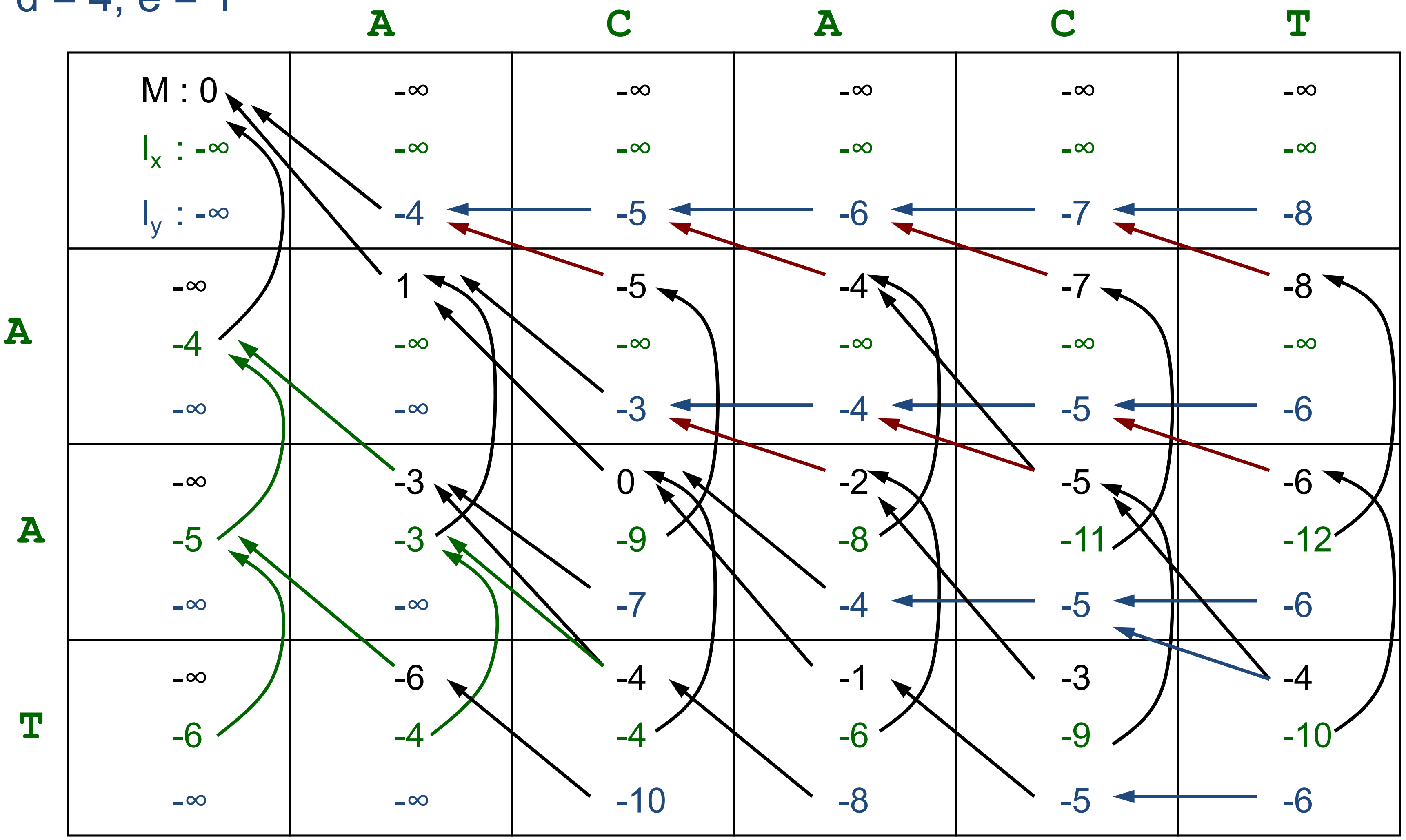
- initialization

$$M(0,0) = 0$$
$$I_x(i, 0) = -d - (i-1)e \quad \text{for } i > 0$$
$$I_y(0, j) = -d - (j-1)e \quad \text{for } j > 0$$
other cells in top row and leftmost column $= -\infty$

- traceback
  - start at largest of [redacted]
  - stop at [redacted]
  - note that pointers may traverse all three matrices

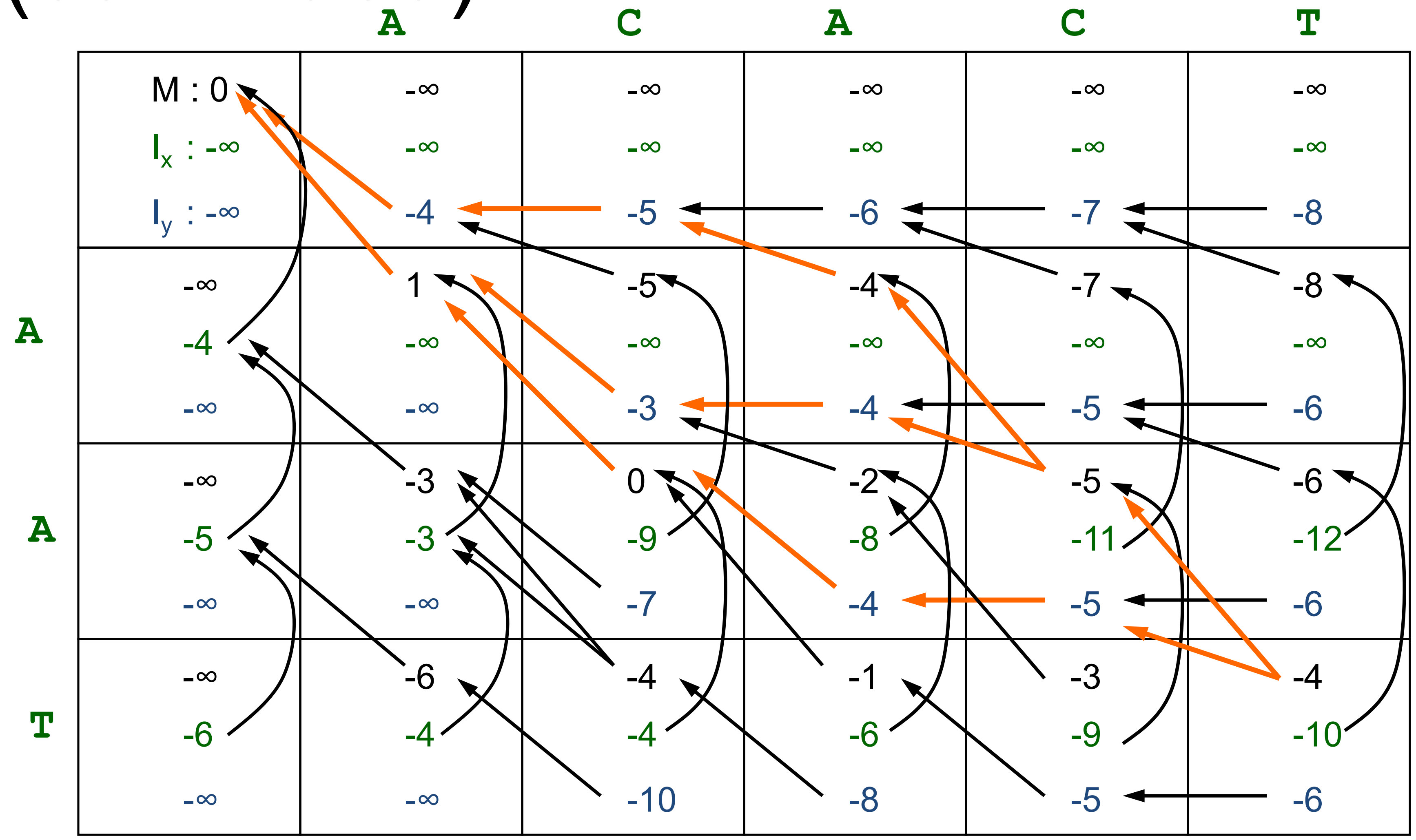
Global alignment example
(affine gap penalty)

d = 4, e = 1



			A	C	A	C	T
	M	0	-∞	-∞	-∞	-∞	-∞
	I_x	-∞	-∞	-∞	-∞	-∞	-∞
	I_y	-∞	-4	-5	-6	-7	-8
A		-∞	1	-5	-4	-7	-8
		-4	-∞	-∞	-∞	-∞	-∞
		-∞	-∞	-3	-4	-5	-6
A		-∞	-3	0	-2	-5	-6
		-5	-3	-9	-8	-11	-12
		-∞	-∞	-7	-4	-5	-6
T		-∞	-6	-4	-1	-3	-4
		-6	-4	-4	-6	-9	-10
		-∞	-∞	-10	-8	-5	-6

Global alignment example (continued)



		A	C	A	C	T
	M : 0; I_x : -∞; I_y : -∞	-∞; -∞; -4	-∞; -∞; -5	-∞; -∞; -6	-∞; -∞; -7	-∞; -∞; -8
A	-∞; -4; -∞	1; -∞; -∞	-5; -∞; -3	-4; -∞; -4	-7; -∞; -5	-8; -∞; -6
A	-∞; -5; -∞	-3; -3; -∞	0; -9; -7	-2; -8; -4	-5; -11; -5	-6; -12; -6
T	-∞; -6; -∞	-6; -4; -∞	-4; -4; -10	-1; -6; -8	-3; -9; -5	-4; -10; -6

three optimal alignments:

```
ACACT    ACACT    ACACT
AA--T    A--AT    --AAT
```

Why three matrices are needed

- consider aligning the sequences **WFP** and **FW** using d = 5, e = 1 and the following values from the BLOSUM-62 substitution matrix:

s(**F**, **W**) = 1	s(**W**, **W**) = 11
s(**F**, **F**) = 6	s(**W**, **P**) = -4
s(**F**, **P**) = -4	

- the matrix shows the highest-scoring partial alignment for each pair of prefixes

		W	**F**	**P**
	0	-5	-6	-7
F	-5	1	1	-4
W	-6	6	2	0

-WFP
FW--

optimal alignment

WF
FW

best alignment of these prefixes; to get optimal alignment, need to also remember

-WF
FW-

Pairwise alignment summary

- the number of possible alignments is exponential in the length of sequences being aligned
- dynamic programming can find optimal-scoring alignments in polynomial time
- the specifics of the DP depend on
    - local vs. global alignment
    - gap penalty function
- affine penalty functions are most commonly used