Lecture 10:
Local Alignments

Study Chapter 6.8-6.10

Outline

- Edit Distances
- Longest Common Subsequence
- Global Sequence Alignment
- Scoring Matrices
- Local Sequence Alignment
- Alignment with Affine Gap Penalties
- Multiple Alignment problem

Local vs. Global Alignment

- The Global Alignment Problem tries to find the best path between vertices *(0,0)* and (*n*,*m*) in the edit graph.

- The Local Alignment Problem tries to find the best path among paths between **arbitrary vertices** (*i*,*j*) and (*i′*, *j′*) in the edit graph.

- In the edit graph with negatively-scored edges, Local Alignment may score higher than Global Alignment

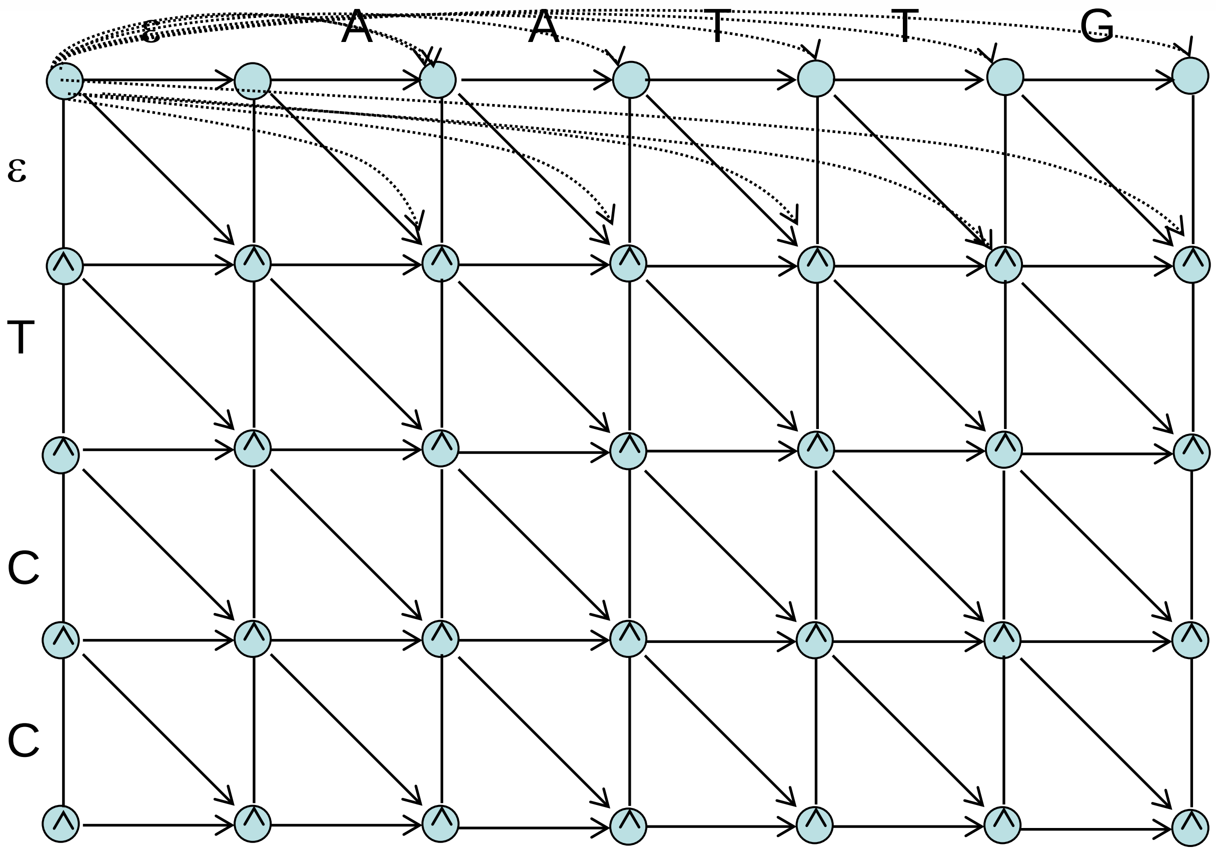
The Local Alignment Recurrence

- The largest value of $s_{i,j}$ over the whole edit graph is the score of the best local alignment.
- Smith-Waterman local alignment
- The recurrence:

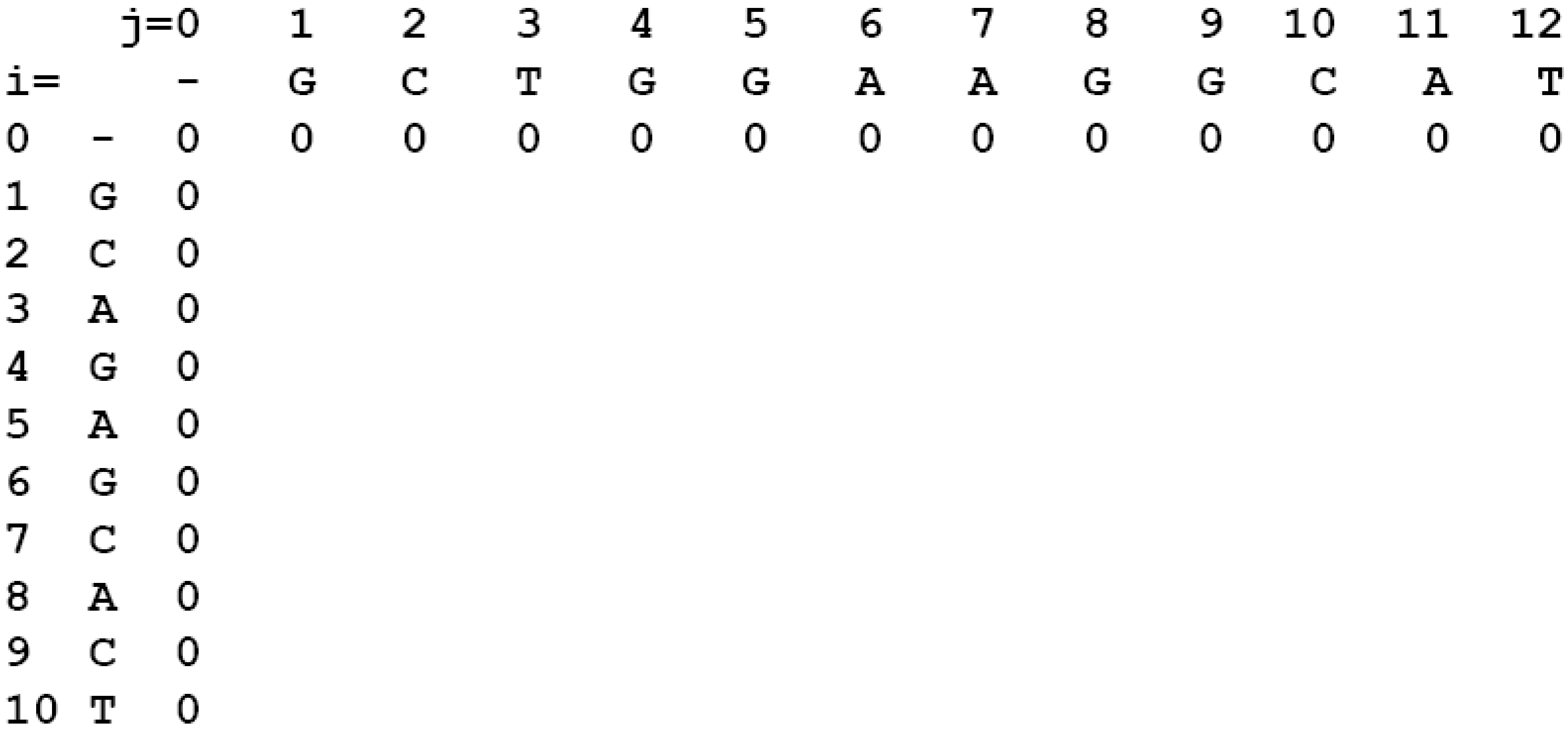
$$s_{i,j} = max \begin{cases} 0 \\ s_{i-1,j-1} + \delta(v_i, w_j) \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \end{cases}$$

Power of ZERO: there is only this change from the original recurrence of a Global Alignment - since there is only one “free ride” edge entering into every vertex

Smith-Waterman Local Alignment



An Example



		j=0	1	2	3	4	5	6	7	8	9	10	11	12
i=		-	G	C	T	G	G	A	A	G	G	C	A	T
0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
1	G	0												
2	C	0												
3	A	0												
4	G	0												
5	A	0												
6	G	0												
7	C	0												
8	A	0												
9	C	0												
10	T	0												

Match = 5, Mismatch = -4, Indel = -7

Local Alignment

		j=0	1	2	3	4	5	6	7	8	9	10	11	12
i=		-	G	C	T	G	G	A	A	G	G	C	A	T
0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
1	G	0	$S_{1,1}$											
2	C	0												
3	A	0												
4	G	0												
5	A	0												
6	G	0												
7	C	0												
8	A	0												
9	C	0												
10	T	0												

Match = 5, Mismatch = -4, Indel = -7

Local Alignment

		j=0	1	2	3	4	5	6	7	8	9	10	11	12
i=		-	G	C	T	G	G	A	A	G	G	C	A	T
0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
1	G	0	5	$S_{1,2}$										
2	C	0												
3	A	0												
4	G	0												
5	A	0												
6	G	0												
7	C	0												
8	A	0												
9	C	0												
10	T	0												

Match = 5, Mismatch = -4, Indel = -7

Local Alignment

		j=0	1	2	3	4	5	6	7	8	9	10	11	12
i=		-	G	C	T	G	G	A	A	G	G	C	A	T
0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
1	G	0	5	0										
2	C	0	0	$S_{2,2}$										
3	A	0												
4	G	0												
5	A	0												
6	G	0												
7	C	0												
8	A	0												
9	C	0												
10	T	0												

Match = 5, Mismatch = -4, Indel = -7

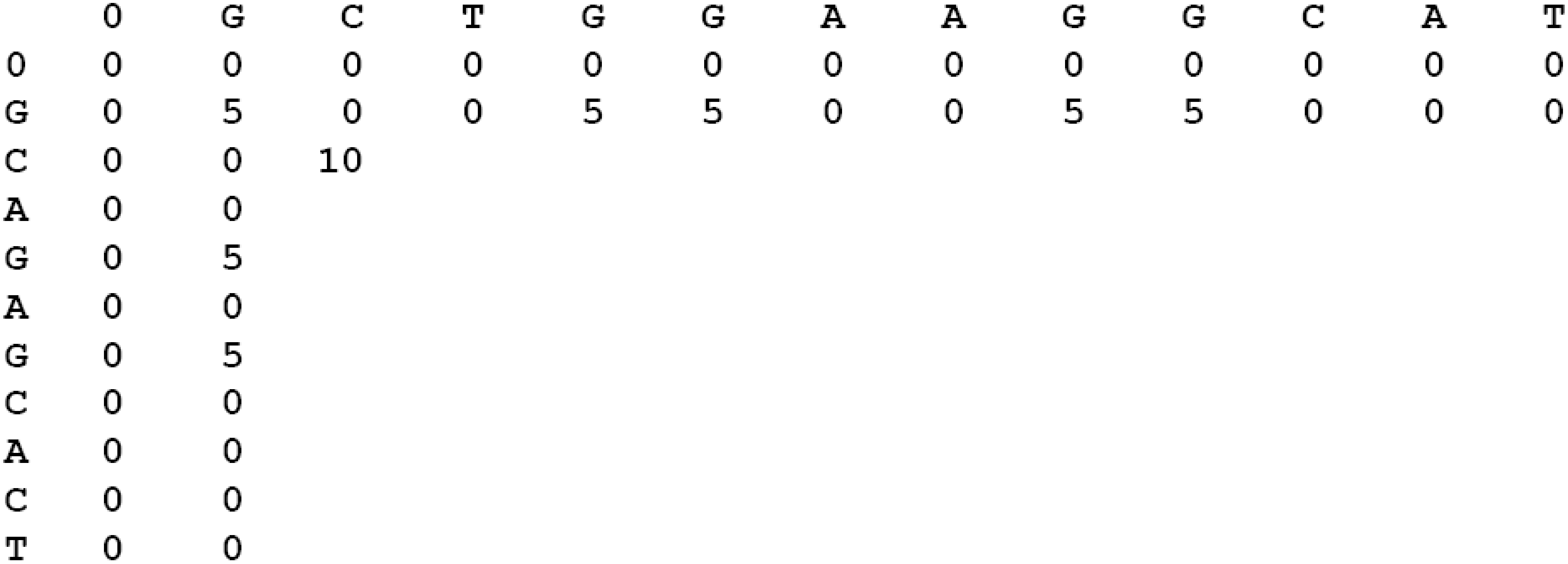
Local Alignment

		j=0	1	2	3	4	5	6	7	8	9	10	11	12
i=		-	G	C	T	G	G	A	A	G	G	C	A	T
0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
1	G	0	5	0										
2	C	0	0	$s_{2,2}$										
3	A	0												
4	G	0												
5	A	0												
6	G	0												
7	C	0												
8	A	0												
9	C	0												
10	T	0												

$$S_{2,2} = \max \left\{ \begin{array}{l} S_{1,1} + s_{C,C} = 5 + 5 = 10 \\ S_{2,1} + w \ = 0 - 7 = -7 \\ S_{1,2} + w \ = 0 - 7 = -7 \\ \qquad\qquad 0 \end{array} \right\} = 10$$

Match = 5, Mismatch = -4, Indel = -7

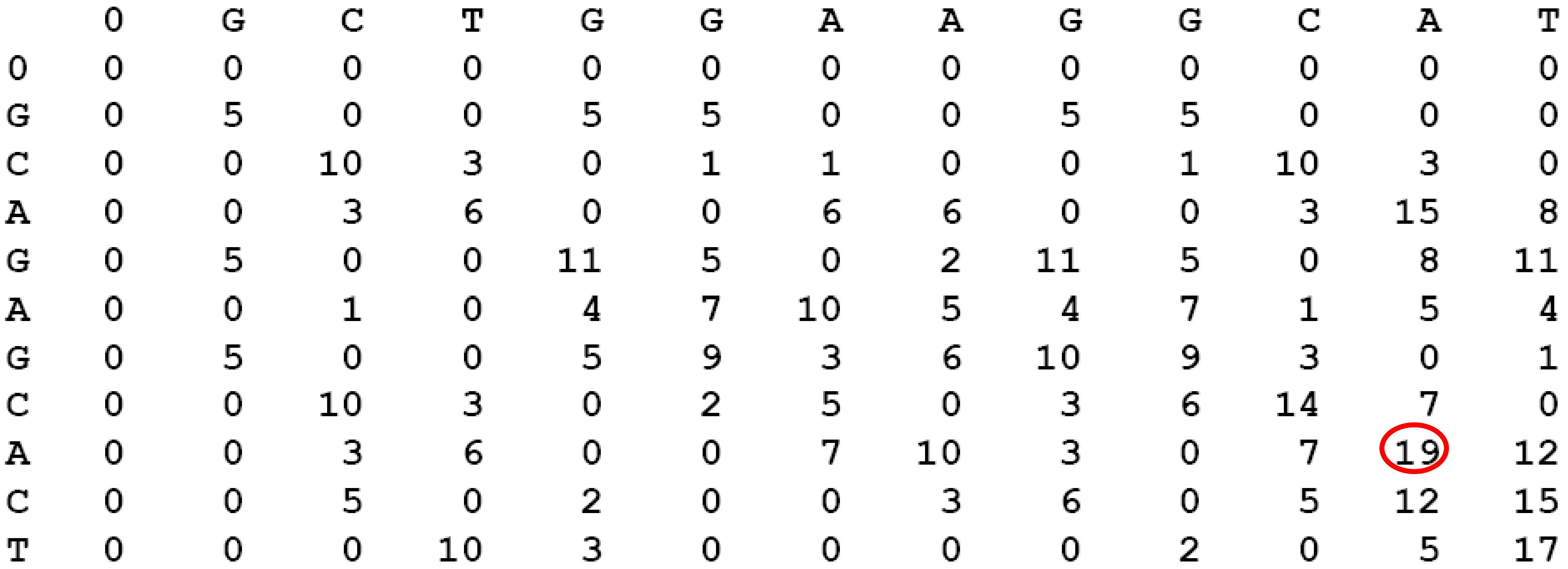
Local Alignment



	0	G	C	T	G	G	A	A	G	G	C	A	T
0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5	0	0	5	5	0	0	5	5	0	0	0
C	0	0	10										
A	0	0											
G	0	5											
A	0	0											
G	0	5											
C	0	0											
A	0	0											
C	0	0											
T	0	0											

Match = 5, Mismatch = -4, Indel = -7

Local Alignment



	0	G	C	T	G	G	A	A	G	G	C	A	T
0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5	0	0	5	5	0	0	5	5	0	0	0
C	0	0	10	3	0	1	1	0	0	1	10	3	0
A	0	0	3	6	0	0	6	6	0	0	3	15	8
G	0	5	0	0	11	5	0	2	11	5	0	8	11
A	0	0	1	0	4	7	10	5	4	7	1	5	4
G	0	5	0	0	5	9	3	6	10	9	3	0	1
C	0	0	10	3	0	2	5	0	3	6	14	7	0
A	0	0	3	6	0	0	7	10	3	0	7	19	12
C	0	0	5	0	2	0	0	3	6	0	5	12	15
T	0	0	0	10	3	0	0	0	0	2	0	5	17

Match = 5, Mismatch = -4, Indel = -7

Local Alignment

V W →

↓	0	G	C	T	G	G	A	A	G	G	C	A	T
0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5	0	0	5	5	0	0	5	5	0	0	0
C	0	0	10	3	0	1	1	0	0	1	10	3	0
A	0	0	3	6	0	0	6	6	0	0	3	15	8
G	0	5	0	0	11	5	0	2	11	5	0	8	11
A	0	0	1	0	4	7	10	5	4	7	1	5	4
G	0	5	0	0	5	9	3	6	10	9	3	0	1
C	0	0	10	3	0	2	5	0	3	6	14	7	0
A	0	0	3	6	0	0	7	10	3	0	7	19	12
C	0	0	5	0	2	0	0	3	6	0	5	12	15
T	0	0	0	10	3	0	0	0	0	2	0	5	17

Match = 5, Mismatch = -4, Indel = -7

Local Alignment

```
v =                 G   C   A   G   A   G   C   A   C   T
                    |   ↕   |   |       |   |   |
w = C   C   T   G   G   A   A   G   -   G   C   A
```

6 matches: 6 × 5 = 30

1 mismatch: -4

1 indel: -7

Total: 19

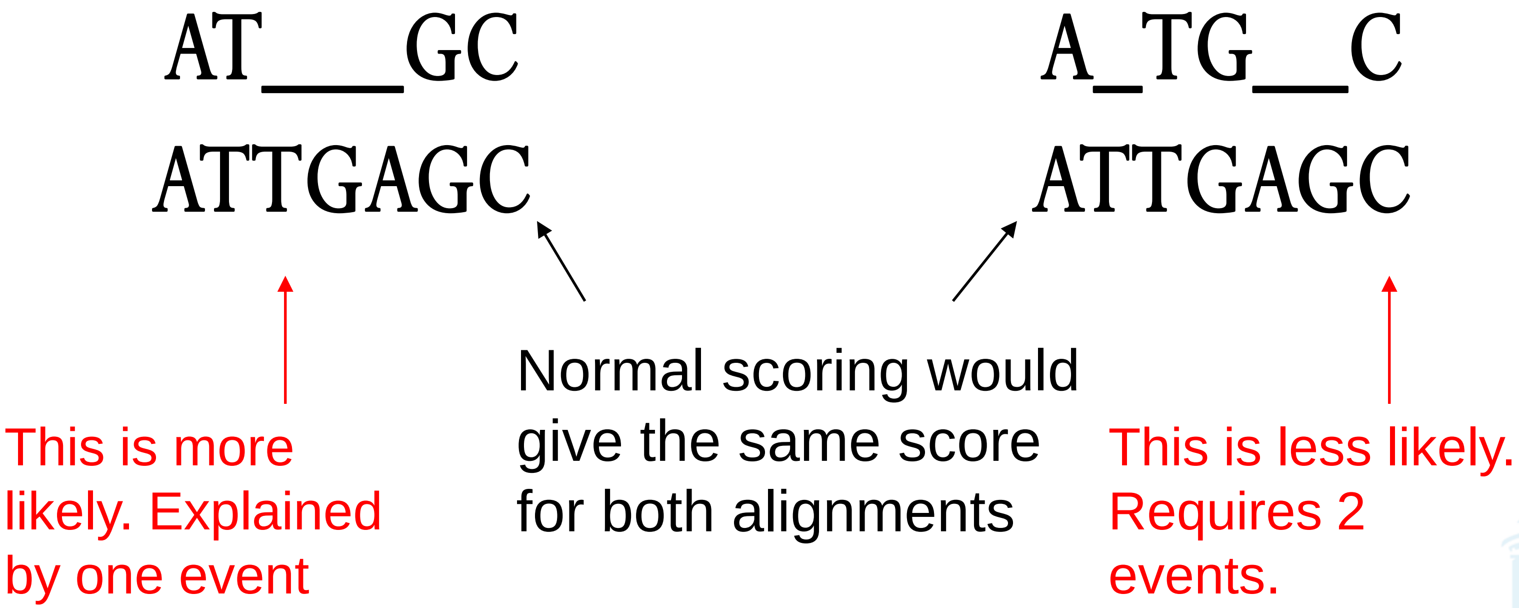
Scoring Indels: Naive Approach

- A fixed penalty σ is given to every indel:
  - $-\sigma$ for 1 indel,
  - -2σ for 2 consecutive indels
  - -3σ for 3 consecutive indels, etc.

Can be too severe penalty for a series of 100 consecutive indels

Affine Gap Penalties

- In nature, a series of *k* indels often come as a single event rather than a series of *k* single nucleotide events:



AT___GC
ATTGAGC

A_TG__C
ATTGAGC

This is more likely. Explained by one event

Normal scoring would give the same score for both alignments

This is less likely. Requires 2 events.

Accounting for Gaps

- *Gaps*- contiguous sequence of indels in one of the rows

- Modify the scoring for a gap of length *x* to be:

$$-(\rho + \sigma x)$$

where $\rho+\sigma > 0$ is the penalty for introducing a gap:

gap opening penalty

andσ is the cost of extending it further $(\rho+\sigma >> \sigma)$:

gap extension penalty

because you do not want to add too much of a penalty for further extending the gap, once it is opened.

Affine Gap Penalties

- Gap penalties:

  $-\rho-\sigma$ when there is 1 indel

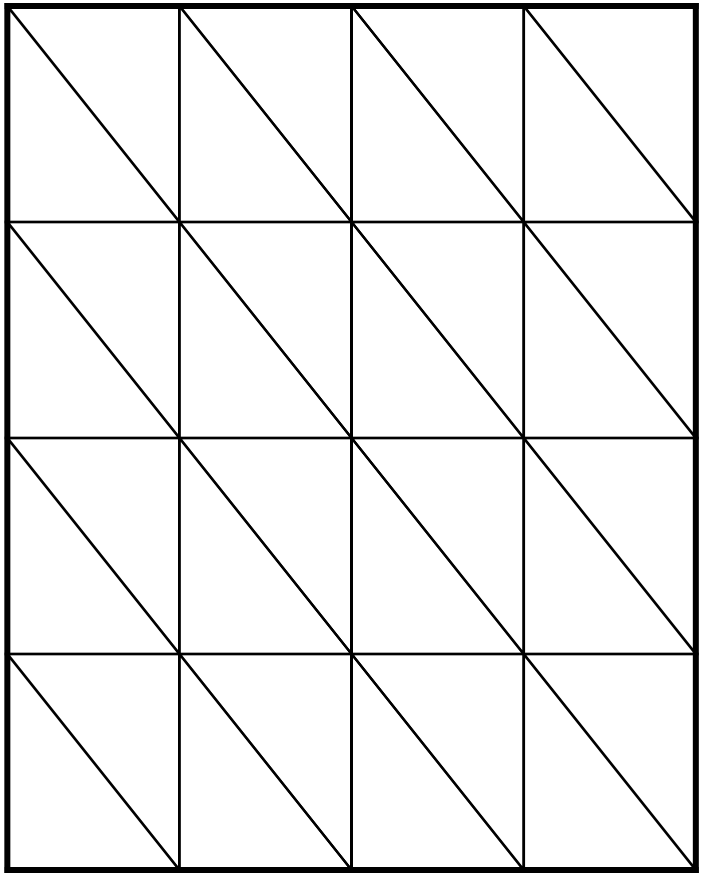
  $-\rho-2\sigma$ when there are 2 indels

  $-\rho-3\sigma$ when there are 3 indels, etc.

  $-\rho- x\cdot\sigma$ (-gap opening - x gap extensions)

- Somehow reduced penalties (as compared to naïve scoring) are given to runs of horizontal and vertical edges

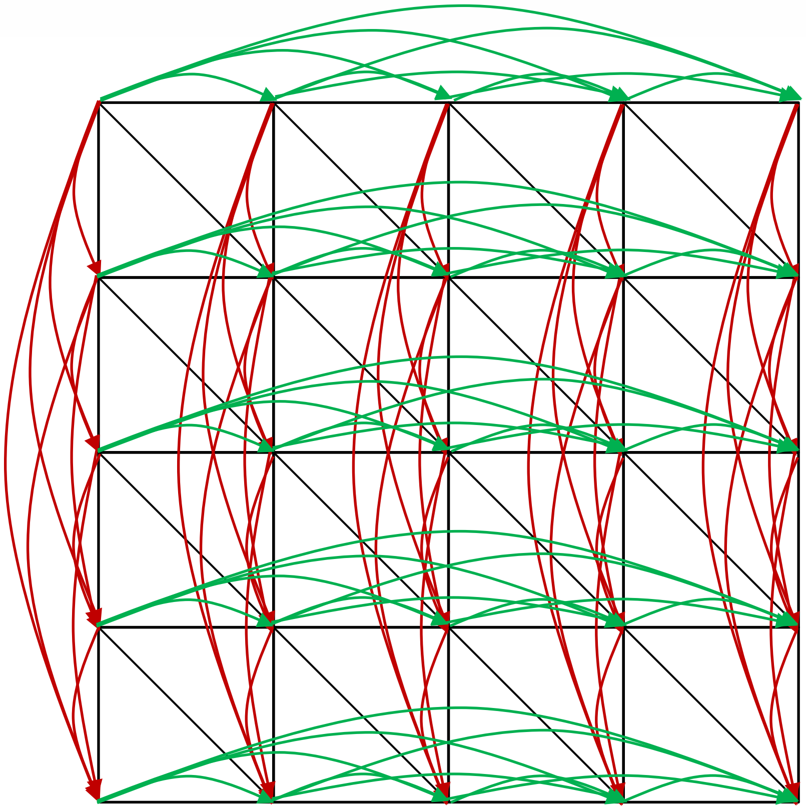
Affine Gap Penalties and Edit Graph



To reflect affine gap penalties we have to add "long" horizontal and vertical edges to the edit graph. Each such edge of length *x* should have weight

$$-\rho - x * \sigma$$

Adding "Affine Penalty" Edges to the Edit Graph



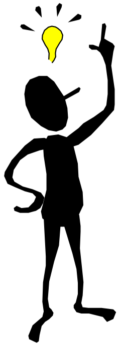
There are many such edges!

Adding them to the graph increases the running time of the alignment algorithm by a factor of ***n*** (where ***n*** is the number of vertices)

So the complexity increases from O(n^2) to O(n^3)

Affine Gap Penalty Recurrences

Keep track of these intermediate values in two new tables



$$\downarrow s_{i,j} = \max \begin{cases} \downarrow s_{i-1,j} - \sigma \\ s_{i-1,j} - (\rho+\sigma) \end{cases}$$

Continue Gap in *w* (deletion)

Start Gap in *w* (deletion): from middle

$$\overrightarrow{s}_{i,j} = \max \begin{cases} \overrightarrow{s}_{i,j-1} - \sigma \\ s_{i,j-1} - (\rho+\sigma) \end{cases}$$

Continue Gap in *v* (insertion)

Start Gap in *v* (insertion):from middle

$$s_{i,j} = \max \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) \\ \downarrow s_{i,j} \\ \overrightarrow{s}_{i,j} \end{cases}$$

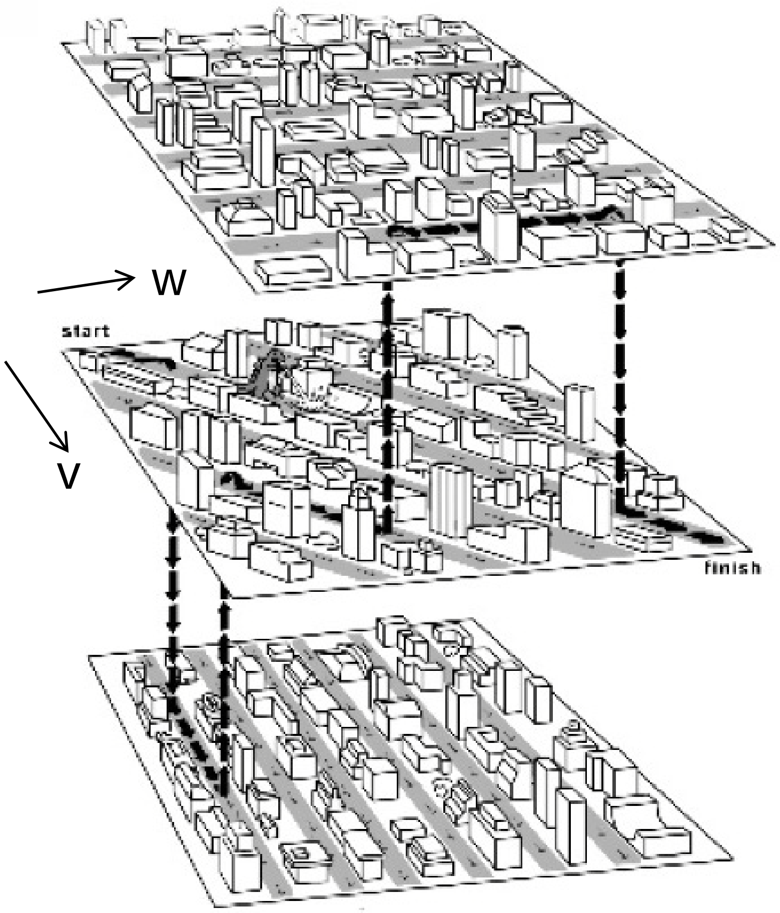
Match or Mismatch

End deletion: from top

End insertion: from left

Complexity O(nm)

The 3-leveled Manhattan Grid



Gaps in v

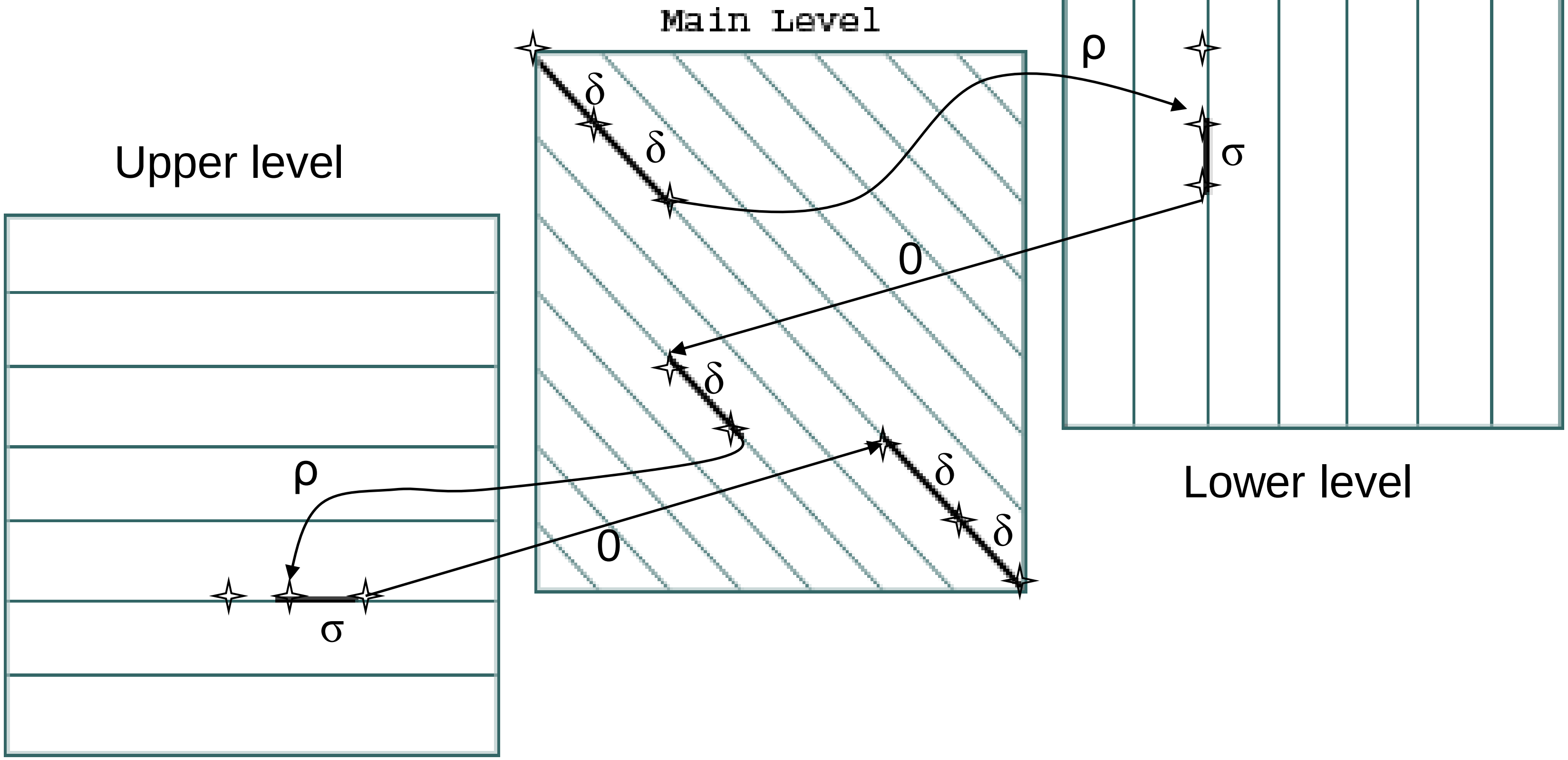
Matches/Mismatches

Gaps in w

Affine Gap Penalties and 3 Layer Manhattan Grid

- The three recurrences for the scoring algorithm creates a 3-layered graph.
- The top level creates/extends gaps in the sequence *v*.
- The bottom level creates/extends gaps in sequence *w*.
- The middle level extends matches and mismatches.

Manhattan in 3 Layers



Switching between 3 Layers

- Levels:
  - The **main level** is for diagonal edges
  - The **lower level** is for vertical edges
  - The **upper level** is for horizontal edges
- A jumping penalty is assigned to moving from the main level to either the upper level or the lower level ($-\rho - \sigma$)
- There is a gap extension penalty for each continuation on a level other than the main level ($-\sigma$)

Multiple Alignment versus Pairwise Alignment

- Up until now we have only tried to align two sequences.
- What about more than two? And what for?
- A faint similarity between two sequences becomes significant if present in many
- Multiple alignments can reveal subtle similarities that pairwise alignments do not reveal

Generalizing the Notion of Pairwise Alignment

- Alignment of 2 sequences is represented as a 2-row matrix
- In a similar way, we represent alignment of 3 sequences as a 3-row matrix

```
A T _ G C G _
A _ C G T _ A
A T C A C _ A
```

- Score: more conserved columns, better alignment

Alignment Paths

- Align 3 sequences: ATGC, AATC,ATGC

0	1	1	2	3	4
	A	--	T	G	C
0	1	2	3	3	4
	A	A	T	--	C
0	0	1	2	3	4
	--	A	T	G	C

x coordinate

y coordinate

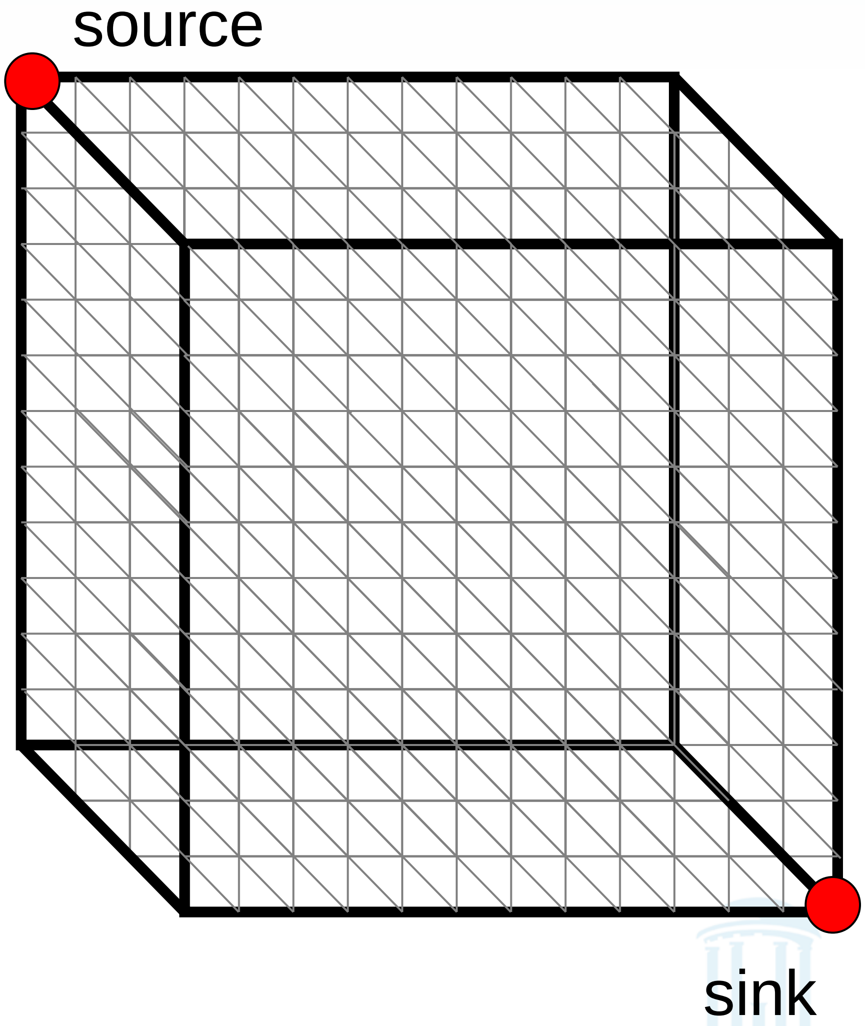
z coordinate

- Resulting path in *(x,y,z)* space:

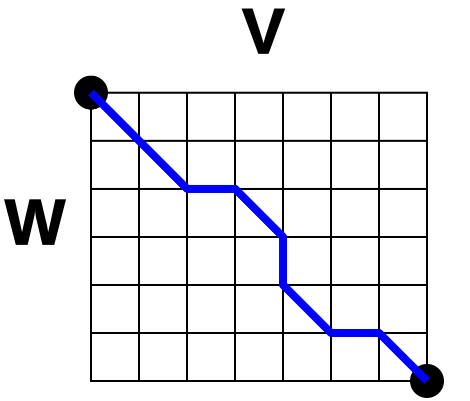
(0,0,0)→(1,1,0)→(1,2,1) →(2,3,2) →(3,3,3) →(4,4,4)

Aligning Three Sequences

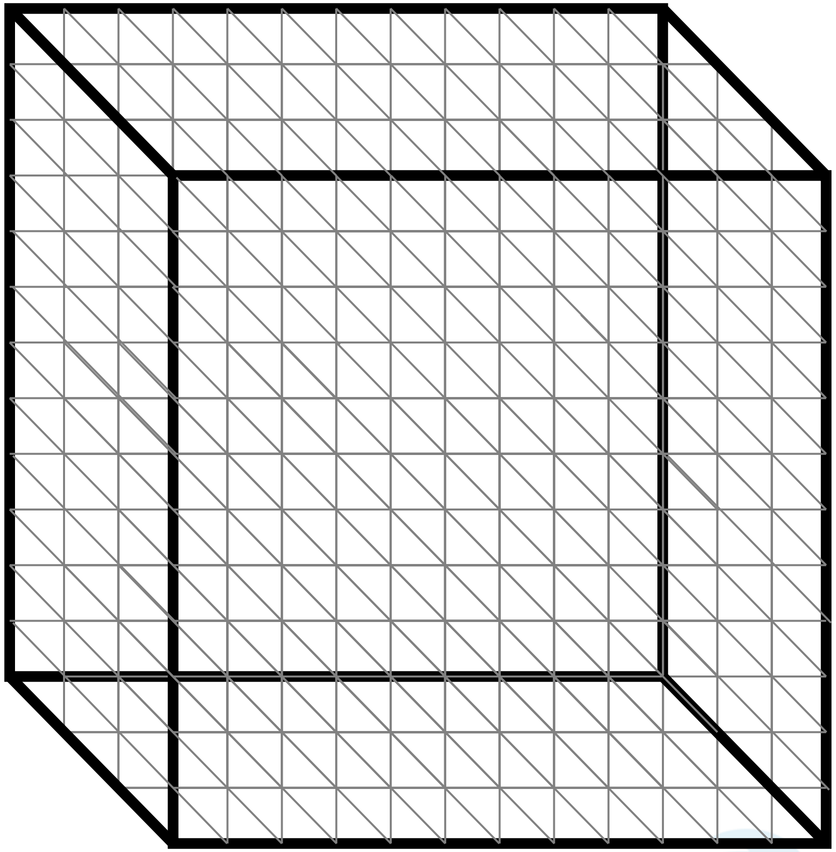
- Same strategy as aligning two sequences
- Use a 3-D "Manhattan Cube", with each axis representing a sequence to align
- For global alignments, go from source to sink



2-D vs 3-D Alignment Grid

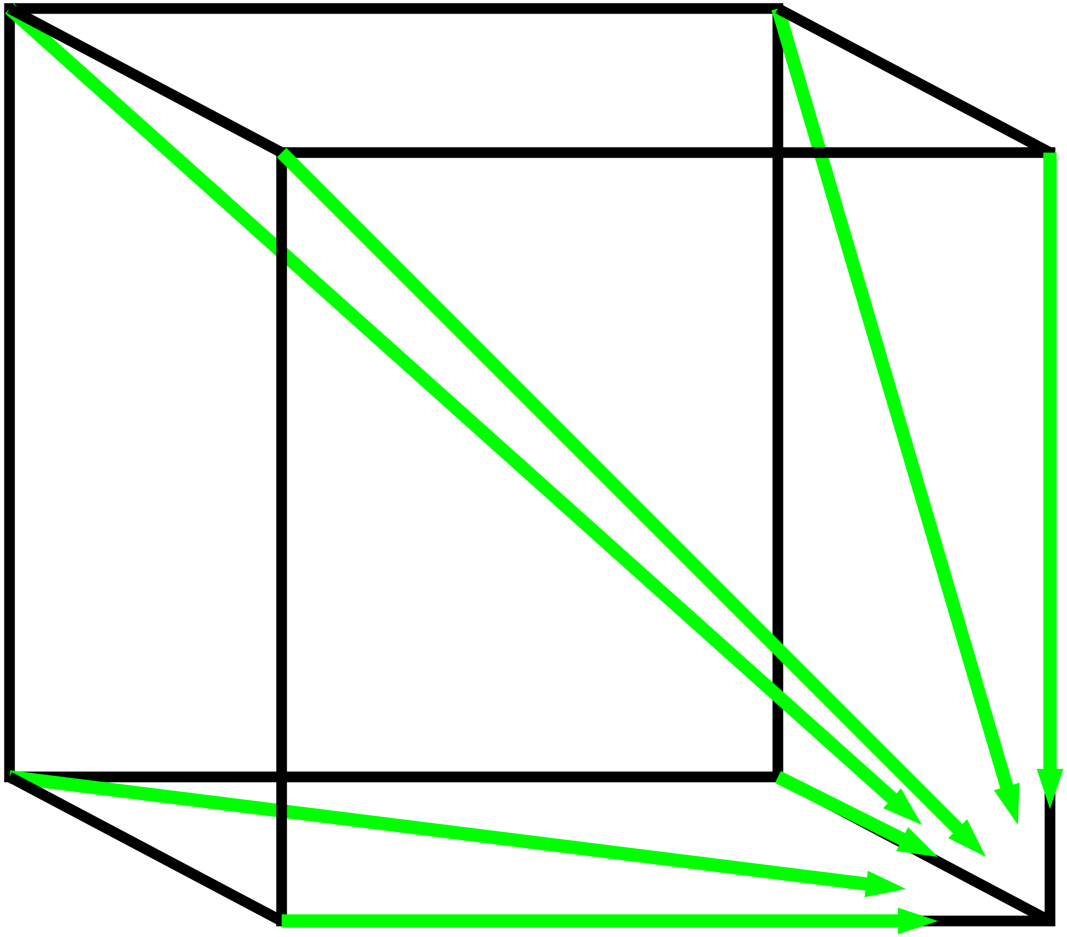


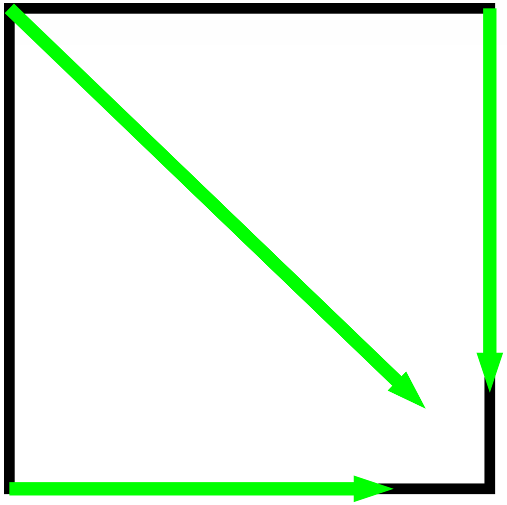
2-D edit graph



3-D edit graph

2-D cell versus 2-D Alignment Cell





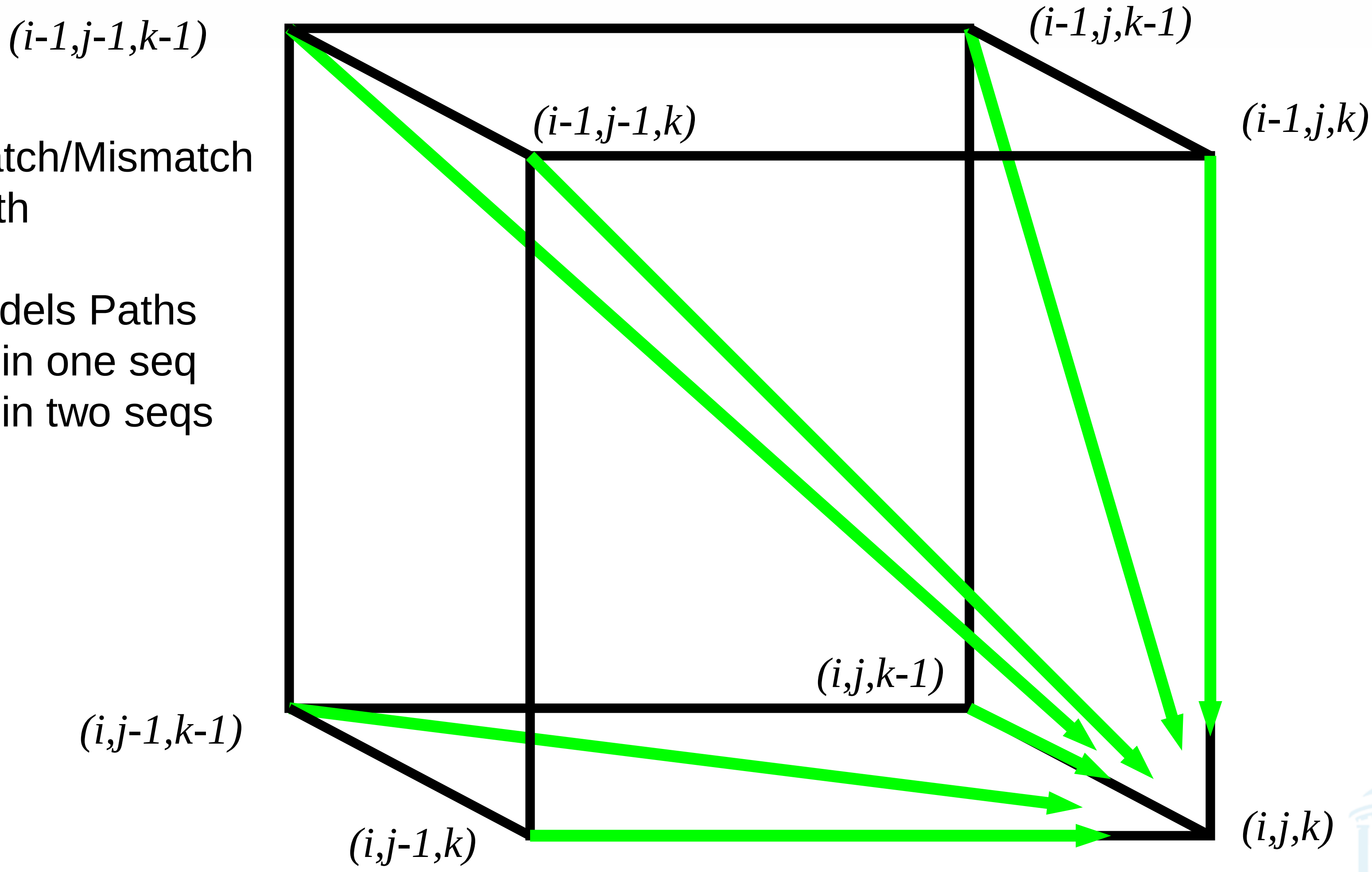
In **2-D**, 3 edges lead to each interior vertex

In **3-D**, 7 edges lead to each interior vertex

Architecture of 3-D Alignment Cell

1- Match/Mismatch Path

6 – indels Paths
3 in one seq
3 in two seqs



Multiple Alignment: Dynamic Programming

- $$s_{i,j,k} = \max \left\{ \begin{array}{ll} \left. s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) \right\} & \text{cube diagonal: no indels} \\ \left. \begin{array}{l} s_{i-1,j-1,k} + \delta(v_i, w_j, \_) \\ s_{i-1,j,k-1} + \delta(v_i, \_, u_k) \\ s_{i,j-1,k-1} + \delta(\_, w_j, u_k) \end{array} \right\} & \text{face diagonal: one indel} \\ \left. \begin{array}{l} s_{i-1,j,k} + \delta(v_i, \_, \_) \\ s_{i,j-1,k} + \delta(\_, w_j, \_) \\ s_{i,j,k-1} + \delta(\_, \_, u_k) \end{array} \right\} & \text{cube edge: two indels} \end{array} \right.$$

- $\delta(x, y, z)$ is an entry in the 3-D scoring matrix

Multiple Alignment: Running Time

- For 3 sequences of length $\boldsymbol{n}$, the run time is $7\boldsymbol{n}^3$; $O(\boldsymbol{n}^3)$

- For $\boldsymbol{k}$ sequences, build a $\boldsymbol{k}$-dimensional Manhattan, with run time $(2^k-1)(\boldsymbol{n}^k) = O(2^k\boldsymbol{n}^k)$

- Conclusion: dynamic programming approach for alignment between two sequences is easily extended to $\boldsymbol{k}$ sequences but it is impractical due to exponential running time

Multiple Alignment Induces Pairwise Alignments

Every multiple alignment induces pairwise alignments

```
x:   AC-GCGG-C
y:   AC-GC-GAG
z:   GCCGC-GAG
```

Induces:

```
x: ACGCGG-C;  x: AC-GCGG-C;  y: AC-GCGAG
y: ACGC-GAC;  z: GCCGC-GAG;  z: GCCGCGAG
```

Reverse Problem: Constructing Multiple Alignment from Pairwise Alignments

Given 3 arbitrary pairwise alignments:

```
x: ACGCTGG-C;   x: AC-GCTGG-C;   y: AC-GC-GAG
y: ACGC--GAC;   z: GCCGCA-GAG;   z: GCCGCAGAG
```

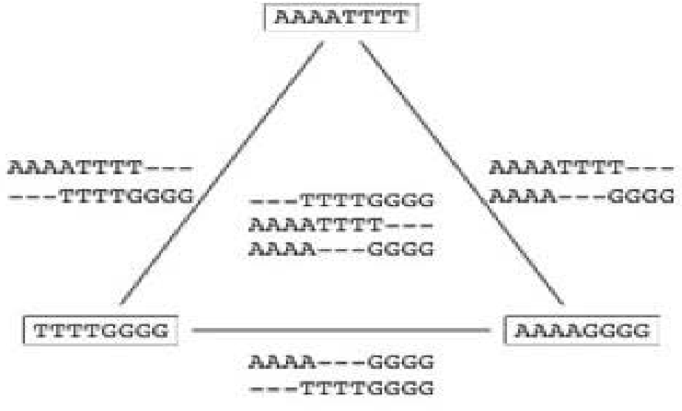
Can we construct a multiple alignment that induces them?

NOT ALWAYS

Why? Because pairwise alignments may be arbitrarily inconsistent

Combining Optimal Pairwise Alignments into Multiple Alignment

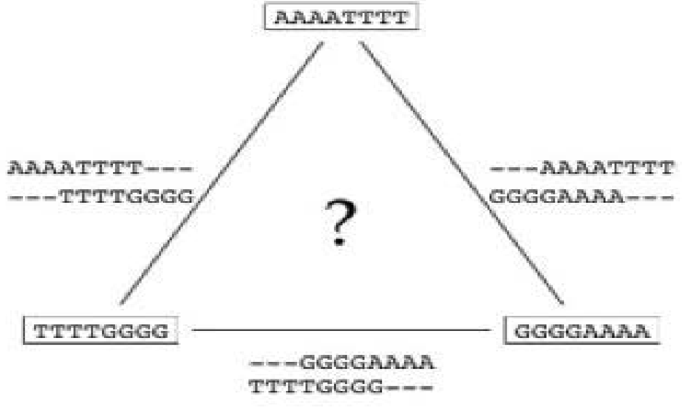
Can combine pairwise alignments into multiple alignment



```
AAAATTTT

AAAATTTT---
---TTTTGGGG

AAAATTTT---
AAAA---GGGG

---TTTTGGGG
AAAATTTT---
AAAA---GGGG

TTTTGGGG          AAAAGGGG

AAAA---GGGG
---TTTTGGGG
```

(a) Compatible pairwise alignments

Can *not* combine pairwise alignments into multiple alignment



```
AAAATTTT

AAAATTTT---
---TTTTGGGG

---AAAATTTT
GGGGAAAA---

?

TTTTGGGG          GGGGAAAA

---GGGGAAAA
TTTTGGGG---
```

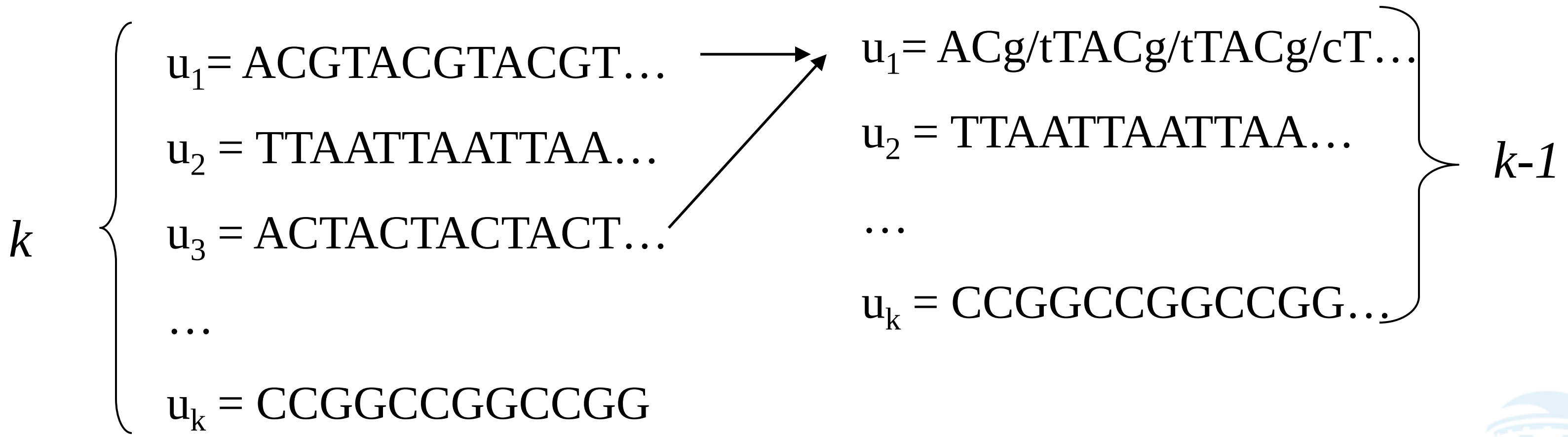
(b) Incompatible pairwise alignments

Inferring Multiple Alignment from Pairwise Alignments

- From an optimal multiple alignment, we can infer pairwise alignments between all pairs of sequences, but they are not necessarily optimal
- It is difficult to infer a "good" multiple alignment from optimal pairwise alignments between all sequences
- Are we stuck, or is there some other trick?

Multiple Alignment: Greedy Approach

- Choose most similar pair of strings and combine into a profile, thereby reducing alignment of *k* sequences to an alignment of *k-1* sequences. **Repeat**
- This is a heuristic greedy method



k

u_1= ACGTACGTACGT…

u_2 = TTAATTAATTAA…

u_3 = ACTACTACTACT…

…

u_k = CCGGCCGGCCGG

u_1= ACg/tTACg/tTACg/cT…

u_2 = TTAATTAATTAA…

…

u_k = CCGGCCGGCCGG…

k-*1*

Greedy Approach: Example

- Consider these 4 sequences

s1	GATTCA
s2	GTCTGA
s3	GATATT
s4	GTCAGC

w/Scoring Matrix:
Match = 1
Mismatch = -1
Indel = -1

Greedy Approach: Example

- There are $\binom{4}{2}$ = 6 possible alignments

```
s2  GTCTGA
s4  GTCAGC (score = 2)
```

```
s1  GAT-TCA
s2  G-TCTGA (score = 1)
```

```
s1  GAT-TCA
s3  GATAT-T (score  = 1)
```

```
s1  GATTCA--
s4  G-T-CAGC (score = 0)
```

```
s2  G-TCTGA
s3  GATAT-T (score = -1)
```

```
s3  GAT-ATT
s4  G-TCAGC (score = -1)
```

Greedy Approach: Example

s_2 *and* s_4 are closest; combine:

s2 GTCTGA

s4 GTCAGC

$s_{2,4}$ (profile) GTCt/aGa/c

new set of 3 sequences:

s_1	GATTCA
s_3	GATATT
$s_{2,4}$	GTCt/aGa/c

Repeat

Progressive Alignment

- *Progressive alignment* is a variation of greedy algorithm with a somewhat more intelligent strategy for choosing the order of alignments.
- Progressive alignment works well for close sequences, but deteriorates for distant sequences
  - Gaps in consensus string are permanent
  - Use profiles to compare sequences
- CLUSTAL

ClustalW

- Popular multiple alignment tool today
- ‘W’ stands for ‘weighted’ (different parts of alignment are weighted differently).
- Three-step process

  1.) Construct pairwise alignments

  2.) Build Guide Tree

  3.) Progressive Alignment guided by the tree

Step 1: Pairwise Alignment

- Aligns each sequence against others giving a similarity matrix
- Similarity = exact matches / sequence length (percent identity)

	v_1	v_2	v_3	v_4
v_1	-			
v_2	.17	-		
v_3	.87	.28	-	
v_4	.59	.33	.62	-

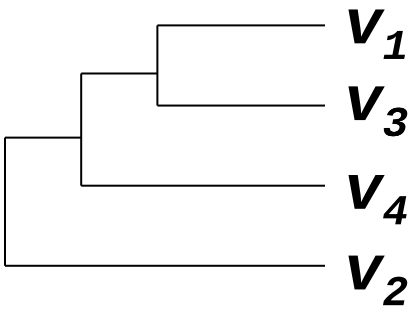
(.17 means 17 % identical)

Step 2: Guide Tree

- Create Guide Tree using the similarity matrix
  - ClustalW uses the neighbor-joining method (we will discuss this later in the course, in the section on clustering)
  - Guide tree roughly reflects evolutionary relations

Step 2: Guide Tree (cont'd)

	v_1	v_2	v_3	v_4
v_1	-			
v_2	.17	-		
v_3	.87	.28	-	
v_4	.59	.33	.62	-



Calculate:

$v_{1,3} = \text{alignment}\ (v_1,\ v_3)$

$v_{1,3,4} = \text{alignment}((v_{1,3}),v_4)$

$v_{1,2,3,4} = \text{alignment}((v_{1,3,4}),v_2)$

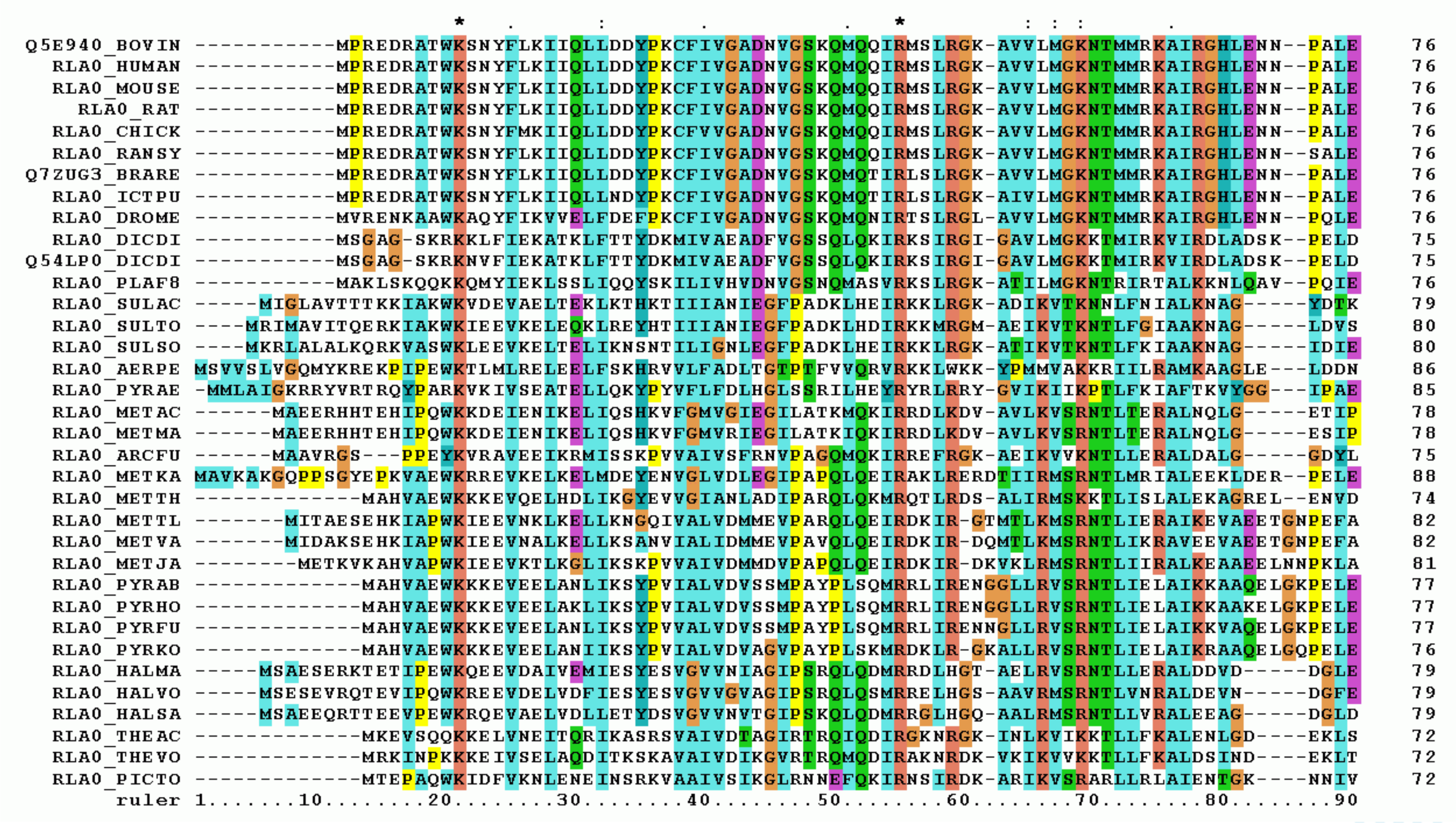
Step 3: Progressive Alignment

- Start by aligning the two most similar sequences
- Following the guide tree, add in the next sequences, aligning to the existing alignment
- Insert gaps as necessary

```
FOS_RAT         PEEMSVTS-LDLTGGLPEATTPESEEAFTLPLLNDPEPK-PSLEPVKNISNMELKAEPFD
FOS_MOUSE       PEEMSVAS-LDLTGGLPEASTPESEEAFTLPLLNDPEPK-PSLEPVKSISNVELKAEPFD
FOS_CHICK       SEELAAATALDLG----APSPAAAEEAFALPLMTEAPPAVPPKEPSG--SGLELKAEPFD
FOSB_MOUSE      PGPGPLAEVRDLPG-----STSAKEDGFGWLLPPPPPPP-----------------LPFQ
FOSB_HUMAN      PGPGPLAEVRDLPG-----SAPAKEDGFSWLLPPPPPPP-----------------LPFQ
                .   . :   ** .     :..  *:.*   *   . *                   **:
```

Dots and stars show how well-conserved a column is.

ClustalW Alignment



```
                                 *   .      :       .          .   *         : : :      .
Q5E940_BOVIN -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_HUMAN -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_MOUSE -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
    RLA0_RAT -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_CHICK -----------MPREDRATWKSNYFMKIIQLLDDYPKCFVVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_RANSY -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--SALE    76
Q7ZUG3_BRARE -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQTIRLSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_ICTPU -----------MPREDRATWKSNYFLKIIQLLNDYPKCFIVGADNVGSKQMQTIRLSLRGK-AIVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_DROME -----------MVRENKAAWKAQYFIKVVELFDEFPKCFIVGADNVGSKQMQNIRTSLRGL-AVVLMGKNTMMRKAIRGHLENN--PQLE    76
  RLA0_DICDI -----------MSGAG-SKRKKLFIEKATKLFTTYDKMIVAEADFVGSSQLQKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSK--PELD    75
Q54LP0_DICDI -----------MSGAG-SKRKNVFIEKATKLFTTYDKMIVAEADFVGSSQLQKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSK--PELD    75
  RLA0_PLAF8 -----------MAKLSKQQKKQMYIEKLSSLIQQYSKILIVHVDNVGSNQMASVRKSLRGK-ATILMGKNTRIRTALKKNLQAV--PQIE    76
  RLA0_SULAC -----MIGLAVTTTKKIAKWKVDEVAELTEKLKTHKTIIIANIEGFPADKLHEIRKKLRGK-ADIKVTKNNLFNIALKNAG-----YDTK    79
  RLA0_SULTO ----MRIMAVITQERKIAKWKIEEVKELEQKLREYHTIIIANIEGFPADKLHDIRKKMRGM-AEIKVTKNTLFGIAAKNAG-----LDVS    80
  RLA0_SULSO ----MKRLALALKQRKVASWKLEEVKELTELIKNSNTILIGNLEGFPADKLHEIRKKLRGK-ATIKVTKNTLFKIAAKNAG-----IDIE    80
  RLA0_AERPE MSVVSLVGQMYKREKPIPEWKTLMLRELEELFSKHRVVLFADLTGTPTFVVQRVRKKLWKK-YPMMVAKKRIILRAMKAAGLE---LDDN    86
  RLA0_PYRAE -MMLAIGKRRYVRTRQYPARKVKIVSEATELLQKYPYVFLFDLHGLSSRILHEYRYRLRRY-GVIKIIKPTLFKIAFTKVYGG---IPAE    85
  RLA0_METAC ------MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVGIEGILATKMQKIRRDLKDV-AVLKVSRNTLTERALNQLG-----ETIP    78
  RLA0_METMA ------MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVRIEGILATKIQKIRRDLKDV-AVLKVSRNTLTERALNQLG-----ESIP    78
  RLA0_ARCFU ------MAAVRGS---PPEYKVRAVEEIKRMISSKPVVAIVSFRNVPAGQMQKIRREFRGK-AEIKVVKNTLLERALDALG-----GDYL    75
  RLA0_METKA MAVKAKGQPPSGYEPKVAEWKRREVKELKELMDEYENVGLVDLEGIPAPQLQEIRAKLRERDTIIRMSRNTLMRIALEEKLDER--PELE    88
  RLA0_METTH -------------MAHVAEWKKKEVQELHDLIKGYEVVGIANLADIPARQLQKMRQTLRDS-ALIRMSKKTLISLALEKAGREL--ENVD    74
  RLA0_METTL -------MITAESEHKIAPWKIEEVNKLKELLKNGQIVALVDMMEVPARQLQEIRDKIR-GTMTLKMSRNTLIERAIKEVAEETGNPEFA    82
  RLA0_METVA -------MIDAKSEHKIAPWKIEEVNALKELLKSANVIALIDMMEVPAVQLQEIRDKIR-DQMTLKMSRNTLIKRAVEEVAEETGNPEFA    82
  RLA0_METJA --------METKVKAHVAPWKIEEVKTLKGLIKSKPVVAIVDMMDVPAPQLQEIRDKIR-DKVKLRMSRNTLIIRALKEAAEELNNPKLA    81
  RLA0_PYRAB -------------MAHVAEWKKKEVEELANLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAQELGKPELE    77
  RLA0_PYRHO -------------MAHVAEWKKKEVEELAKLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAKELGKPELE    77
  RLA0_PYRFU -------------MAHVAEWKKKEVEELANLIKSYPVVALVDVSSMPAYPLSQMRRLIRENNGLLRVSRNTLIELAIKKVAQELGKPELE    77
  RLA0_PYRKO -------------MAHVAEWKKKEVEELANIIKSYPVIALVDVAGVPAYPLSKMRDKLR-GKALLRVSRNTLIELAIKRAAQELGQPELE    76
  RLA0_HALMA -----MSAESERKTETIPEWKQEEVDAIVEMIESYESVGVVNIAGIPSRQLQDMRRDLHGT-AELRVSRNTLLERALDDVD-----DGLE    79
  RLA0_HALVO -----MSESEVRQTEVIPQWKREEVDELVDFIESYESVGVVGVAGIPSRQLQSMRRELHGS-AAVRMSRNTLVNRALDEVN-----DGFE    79
  RLA0_HALSA -----MSAEEQRTTEEVPEWKRQEVAELVDLLETYDSVGVVNVTGIPSKQLQDMRRGLHGQ-AALRMSRNTLLVRALEEAG-----DGLD    79
  RLA0_THEAC -------------MKEVSQQKKELVNEITQRIKASRSVAIVDTAGIRTRQIQDIRGKNRGK-INLKVIKKTLLFKALENLGD----EKLS    72
  RLA0_THEVO -------------MRKINPKKKEIVSELAQDITKSKAVAIVDIKGVRTRQMQDIRAKNRDK-VKIKVVKKTLLFKALDSIND----EKLT    72
  RLA0_PICTO -------------MTEPAQWKIDFVKNLENEINSRKVAAIVSIKGLRNNEFQKIRNSIRDK-ARIKVSRARLLRLAIENTGK----NNIV    72
       ruler 1.......10........20........30........40........50........60........70........80........90
```

Multiple Alignments: Scoring

- Number of matches (multiple longest common subsequence score)
- Entropy score
- Sum of pairs (SP-Score)

Multiple LCS Score

- A column is a "match" if all the letters in the column are the same

AAA
AAA
AAT
ATC

- Only good for very similar sequences

Profile Representation of Multiple Alignment

	-	A	G	G	C	T	A	T	C	A	C	C	T	G
	T	A	G	–	C	T	A	C	C	A	-	-	-	G
	C	A	G	–	C	T	A	C	C	A	-	-	-	G
	C	A	G	–	C	T	A	T	C	A	C	–	G	G
	C	A	G	–	C	T	A	T	C	G	C	–	G	G
A		1					1			.8				
C	.6				1			.4	1		.6	.2		
G			1	.2						.2			.4	1
T	.2					1		.6					.2	
-	.2			.8							.4	.8	.4	

Entropy

- Define frequencies for the occurrence of each letter in each column of multiple alignment
  - $p_A = 1, p_T=p_G=p_C=0$ (1st column)
  - $p_A = 0.75, p_T = 0.25, p_G=p_C=0$ (2nd column)
  - $p_A = 0.50, p_T = 0.25, p_C=0.25\ p_G=0$ (3rd column)
- Compute entropy of each column

$$-\sum_{X=A,T,G,C} p_X \log p_X$$

AAA
AAA
AAT
ATC

Entropy: Example

$$entropy\begin{pmatrix} A \\ A \\ A \\ A \end{pmatrix} = 0$$

Best case

Worst case

$$entropy\begin{pmatrix} A \\ T \\ G \\ C \end{pmatrix} = -\sum \frac{1}{4}\log\frac{1}{4} = -4(\frac{1}{4} * -2) = 2$$

Multiple Alignment: Entropy Score

Entropy for a multiple alignment is the sum of entropies of its columns:

$$\Sigma_{\text{over all columns}} \Sigma_{X=A,T,G,C}\, p_X \log p_X$$

Entropy of an Alignment: Example

column entropy:

$$-(\, p_A \log p_A + p_C \log p_C + p_G \log p_G + p_T \log p_T)$$

A	A	A
A	C	C
A	C	G
A	C	T

- Column 1 = -[1*log(1) + 0*log0 + 0*log0 +0*log0]
  = 0
- Column 2 = $-[({}^{1}/_{4})*\log({}^{1}/_{4}) + ({}^{3}/_{4})*\log({}^{3}/_{4}) + 0*\log 0 + 0*\log 0]$
  $= -[\, ({}^{1}/_{4})*(-2) + ({}^{3}/_{4})*(-.415)\,] = 0.811$
- Column 3 = $-[({}^{1}/_{4})*\log({}^{1}/_{4})+({}^{1}/_{4})*\log({}^{1}/_{4})+({}^{1}/_{4})*\log({}^{1}/_{4}) +({}^{1}/_{4})*\log({}^{1}/_{4})]$
  $= 4* -[({}^{1}/_{4})*(-2)] = +2.0$
- Alignment Entropy = 0 + 0.811 + 2.0 = 2.811

Sum of Pairs Score(SP-Score)

- Consider pairwise alignment of sequences

$$a_i \text{ and } a_j$$

imposed by a multiple alignment of k sequences

- Denote the score of this suboptimal (not necessarily optimal) pairwise alignment as

$$s^*(a_i, a_j)$$

- Sum up the pairwise scores for a multiple alignment:

$$s(a_1,\ldots,a_k) = \Sigma_{i,j}\, s^*(a_i, a_j)$$

Computing SP-Score

Aligning 4 sequences: 6 pairwise alignments

Given a_1, a_2, a_3, a_4:

$$\begin{aligned} s(a_1 \ldots a_4) = \Sigma s^*(a_i, a_j) &= s^*(a_1, a_2) + s^*(a_1, a_3) \\ &+ s^*(a_1, a_4) + s^*(a_2, a_3) \\ &+ s^*(a_2, a_4) + s^*(a_3, a_4) \end{aligned}$$

SP-Score: Example

a_1 ATG–C–AAT

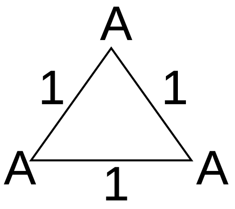
. A–G–CATAT

a_k ATCCCATTT

To calculate each column:

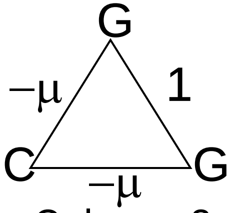
$$s\,(a_1 \ldots a_k) = \sum_{i,j} s^*(\,a_i, a_j) \longleftarrow \binom{k}{2}$$

Pairs of Sequences



Column 1

Score=3



Column 3

Score = $1 - 2\mu$

Profile Representation of Multiple Alignment

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

A		1					1			.8				
C	.6				1			.4	1		.6	.2		
G			1	.2						.2			.4	1
T	.2					1		.6					.2	
-	.2			.8							.4	.8	.4	

Thus far we have aligned a **sequence against a sequence**

Can we align a **sequence against a profile?**

Can we align a **profile against a profile?**

Next Time

- Gene Prediction