Advanced Sequence Alignment

```
CLUSTAL O(1.2.1) multiple sequence alignment


Cat         MAPWTRLLPLLALLSLWIPAPTRAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAED 60
Pig         MALWTRLLPLLALLALWAPAPAQAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAEN 60
Human       MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAED 60
Dog         MALWMRLLPLLALLALWAPAPTRAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREVED 60
            ** * *********:** * *: ******************************:***.*:

Cat         LQGKDAELGEAPGAGGLQPSALEAPLQKRGIVEQCCASVCSLYQLEHYCN   110
Pig         PQAGAVELGG--GLGGLQALALEGPPQKRGIVEQCCTSICSLYQLENYCN   108
Human       LQ------------GSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN   98
Dog         LQVRDVELAGAPGEGGLQPLALEGALQKRGIVEQCCTSICSLYQLENYCN   110
             *            *.**  ***.  **********:*:*******.***
```

- Midterm on Wednesday
  - Covers up to and including Lecture 11
  - Online; it can be downloaded at the start of class; same Jupyter Nookbook format as Problem Sets
  - Open Computer, Open Notes
  - You can add extra cells for scratch work, but only the indicated answer cells will be graded
  - Mix of short answer, multiple choice, and writing code fragments
  - Hard deadline for submission! Bank versions as the time limit approaches.

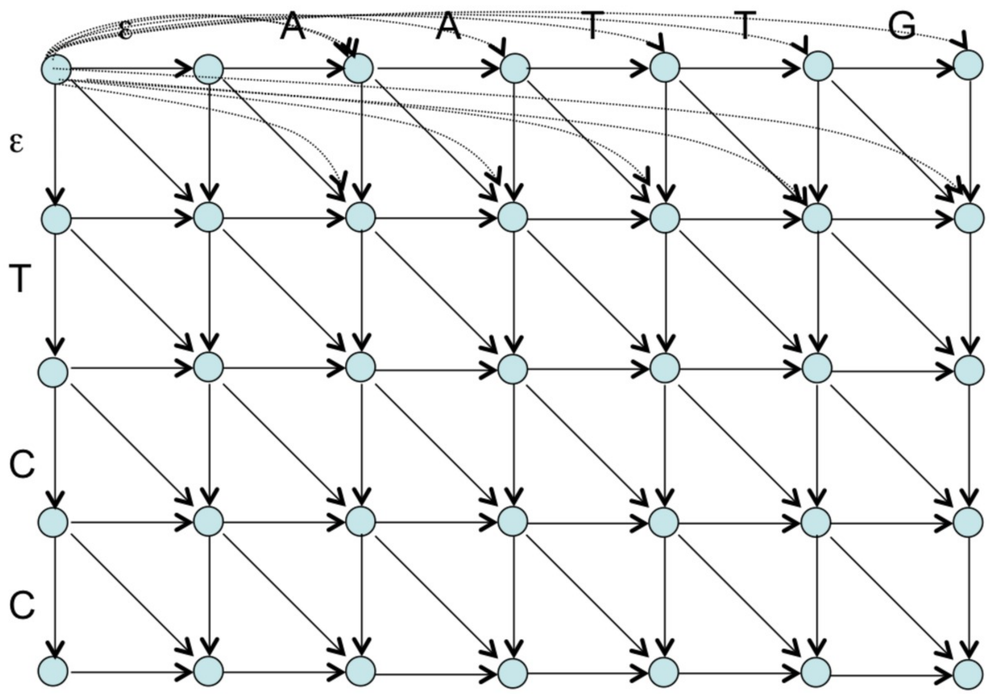
Recall Local Alignment

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

Notice there is only this small change from the original recurrence of a Global Alignment

- The *zero* is our *free ride* that allows the node to restart with a score of 0 at any point
  - What does this imply?
- After solving for the entire score matrix, we then search for $s_{i,j}$ with the highest score, this is (i_2, j_2)
- We follow our back tracking matrix until we reach a *score* of 0, whose coordinate becomes (i_1, j_1)

Smith-Waterman Local Alignment



Key idea: Adding *"free-rides"* from the source to any intersection

A Local Alignment 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

A Local Alignment Example - continued

		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												

$$S_{1,1} = \max \begin{Bmatrix} S_{0,0} + s_{G,G} = 0 + 5 = 5 \\ S_{1,0} + w = 0 - 7 = -7 \\ S_{0,1} + w = 0 - 7 = -7 \\ 0 \end{Bmatrix} = 5$$

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

A Local Alignment Example - continued

		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												

$$S_{1,2} = \max \begin{Bmatrix} S_{0,1} + s_{G,C} = 0 - 4 = -4 \\ S_{1,2} + w = 5 - 7 = -2 \\ S_{0,2} + w = 0 - 7 = -7 \\ 0 \end{Bmatrix} = 0$$

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

A Local Alignment Example - continued

		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												

$$S_{2,2} = \max \begin{Bmatrix} S_{1,1} + s_{C,C} = 5 + 5 = 10 \\ S_{2,1} + w = 0 - 7 = -7 \\ S_{1,2} + w = 0 - 7 = -7 \\ 0 \end{Bmatrix} = 10$$

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

A Local Alignment Example - continued

	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

A Local Alignment Example - continued

	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

Once the score table is completed, find the largest score attained, then bracktrack from there to find the alignment.

A Local Alignment Example - continued

```
G C T G G A A G - G C A T
        |   | |   | | |
        G C A G A G C A C T
```

6 matches: 6 × 5 = 30
1 mismatch: -4
1 indel: -7
Total: 19

Scoring Indels: Naive Approach

```
ATCTTCAGCCATAAAAGATGAAGTT        Reference
ATCTTCAGCCAAAGATGAAGTT           3 base deletion relative to the reference
ATCTTCAGCCATAAAAGATGAAGTT        version 1
ATCTTCAGCCATAAAAGATGAAGTT        version 2
ATCTTCAGCCATAAAAGATGAAGTT        version 3
ATCTTCAGCCATAAAAGATGAAGTT        version 4
ATCTTCAGCCATAAAAGATGAAGTT        version 5
ATCTTCAGCCATATGTGAAAGATGAAGTT    4 base insertion
```

- A fixed penalty σ is given to every indel:
  - -σ 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
  - large insertions or deletions might result from a single event

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        A_TG__C
ATTGAGC        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:

ρ = gap opening penalty

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

σ = gap extension penalty

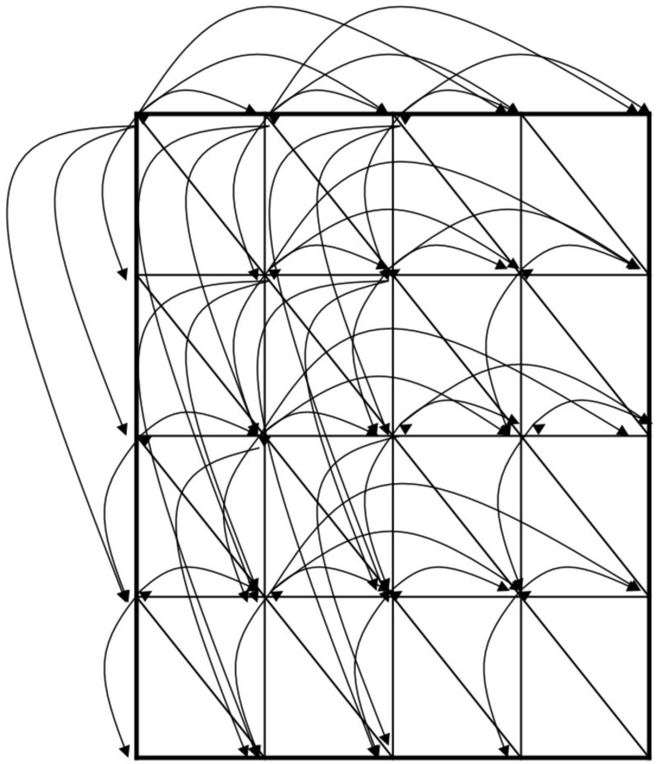
because you do not want to add too much of a penalty for further extending the gap, once it is opened.

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

Adding Affine Gap Penalties to our 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 $-\rho - x \cdot \sigma$
- 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 $O(n^3)$



Adding Two More Tables

- Affine Gap penalties can be expressed in terms of 3 recurrences

Keep track of these intermediate values in two new tables

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

Continue Gap in *w* (deletion)

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

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

Continue Gap in *v* (insertion)

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

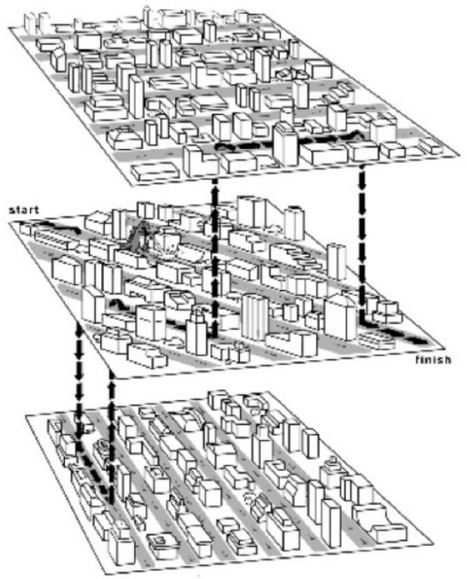
$$s_{i,j} = \max \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) \\ t_{i,j} \\ u_{i,j} \end{cases}$$

Match or Mismatch

End deletion: from top

End insertion: from left

A 3-level Manhattan Grid



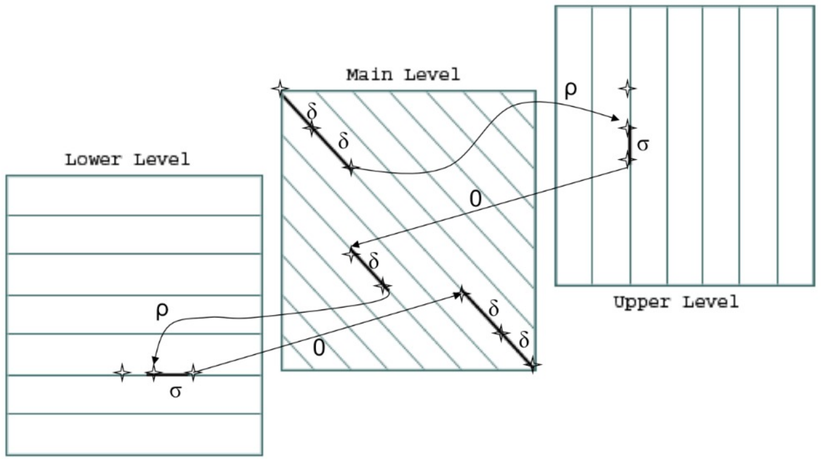
Gaps in w (t-table)

Matches/Mismatches (s-table)

Gaps in v (u-table)

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

Switching between 3 Layers



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

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 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

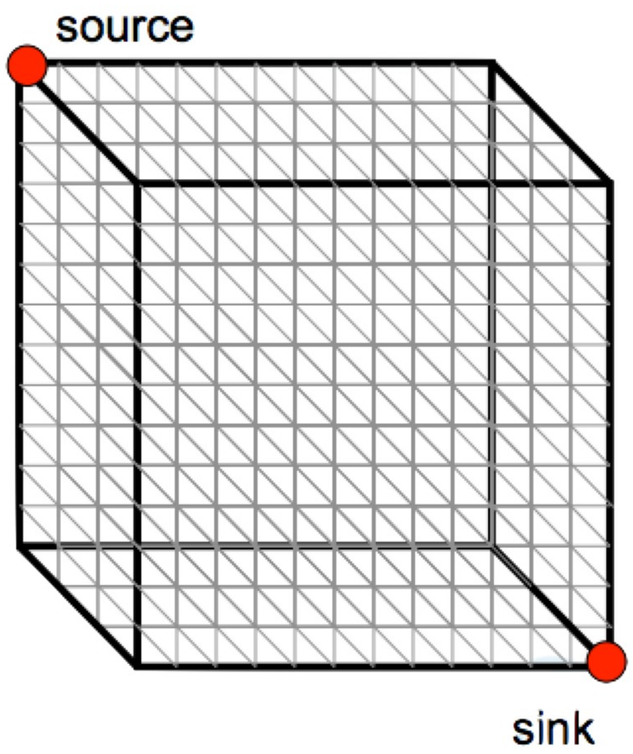
Three-D Alignment Paths

- An alignment of 3 sequences: ATGC, AATC, ATGC

0	1	1	2	3	4	*x* coordinate
	A	--	T	G	C	
0	1	2	3	3	4	*y* coordinate
	A	A	T	--	C	
0	0	1	2	3	4	*z* coordinate
	--	A	T	G	C	

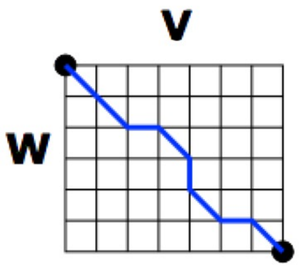
- Resulting path in (x,y,z) space:
  (0,0,0) → (1,1,0) → (1,2,1) → (2,3,2) → (3,3,3) → (4,4,4)
- Is there a better one?

Aligning Three Sequences

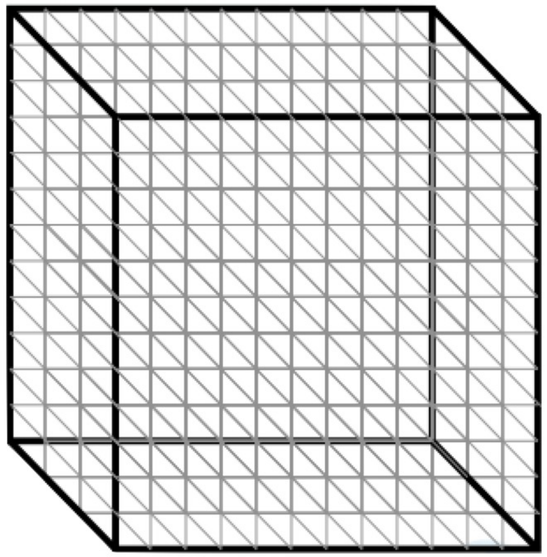


- Same strategy as aligning two sequences
- Use a 3-D “Manhattan Cube”, with each axis representing a sequence to align
- For global alignments, go from source to sink

2-sequence vs 3-sequence Alignment

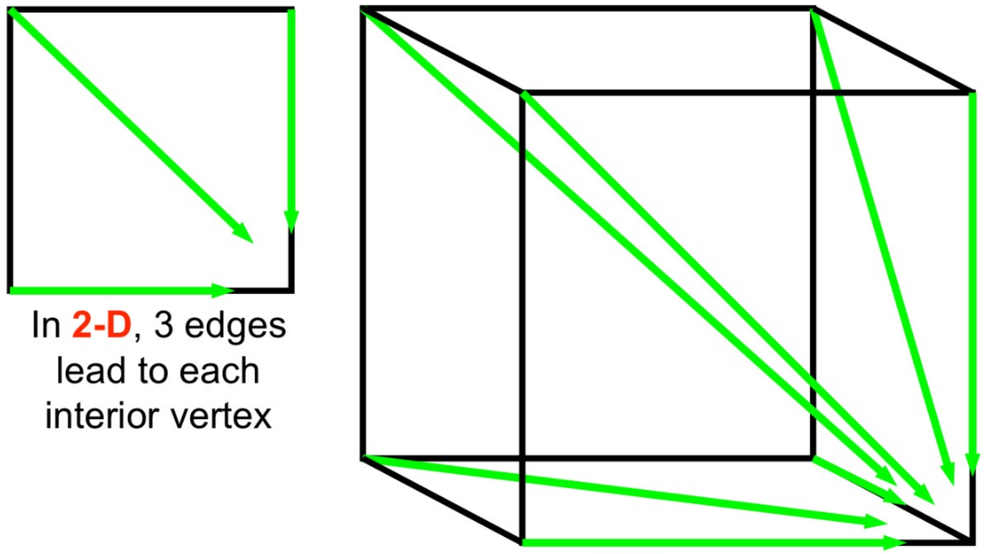


2-D edit graph



3-D edit graph

A 2-D cell versus a 3-D Alignment Cell



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

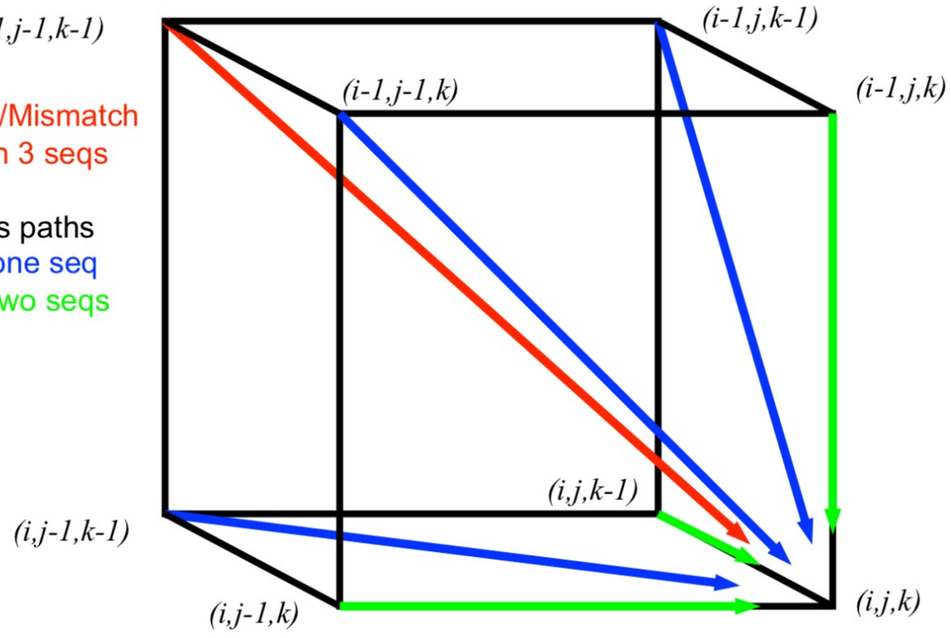
In **3-D**, 7 edges lead to each interior vertex

- 2-D *[(i-1,j-1), (i-1,j), (i,j-1)] → (i,j)*
- 3-D *[(i-1,j-1,k-1), (i-1,j,k), (i,j-1,k), (i,j,k-1), (i,j-1,k-1), (i-1,j,k-1), (i-1,j-1,k),] → (i,j,k)*

Structure of a 3-D Alignment Cell

1- Match/Mismatch
Path in 3 seqs

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



Multiple Alignment: Recursion Relation

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

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

Multiple Alignment: Running Time

- For 3 sequences of length n, the run time is $7n^3$; $O(n^3)$
- For k sequences, build a k-dimensional Manhattan, with run time $(2^k - 1)(n^k)$; $O(2^k n^k)$
- Conclusion: dynamic programming approach for alignment between two sequences is easily extended to 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
```

Inverse Problem

Do Pairwise Alignments imply a Multiple Alignment?

- 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?

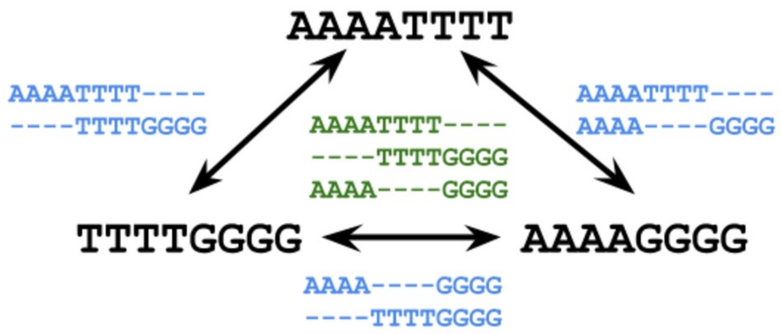
NOT ALWAYS

- Why? Because pairwise alignments may be arbitrarily inconsistent

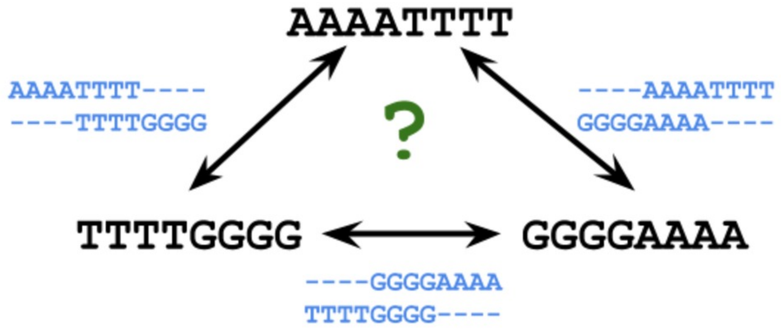
Combining Optimal Pairwise Alignments

- In some cases we can combine pairwie alignments into a single multiple alignment
- But, in others we cannot because one alignment makes a choice that is inconsistent with the overall best choice

```
AAAATTTT--------          ----AAAATTTT----
----TTTTGGGG----    -OR-  --------TTTTGGGG
--------GGGGAAAA          GGGGAAAA--------
```



- Is there another way?



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 using a Profile Scores

- We used profile scores earlier when we discussed Motif finding

	-	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	0	5	0	0	0	0	5	0	0	4	0	0	0	0
C	3	0	0	0	5	0	0	2	5	0	3	1	0	0
G	0	0	5	1	0	0	0	0	0	1	0	0	2	5
T	1	0	0	0	0	5	0	3	0	0	0	0	1	0
-	1	0	0	4	0	0	0	0	0	0	2	4	2	0

- Thus far we have aligned sequences against other sequences
- Can we align a sequence against a profile?
- Can we align a profile against a profile?

Aligning Alignments

A more general version of the multi-alignment problem:

- Given two alignments, can we align them?

```
x: GGGCACTGCAT
y: GGTTACGTC--    Alignment 1
z: GGGAACTGCAG

w: GGACGTACC--    Alignment 2
v: GGACCT-----
```

- Idea: don't use the sequences, but align their profiles

```
x: GGGCAC=TGCAT
y: GGTTAC=GTC--
z: GGGAAC=TGCAG     Combined Alignment
   ||  || | |
w: GG==ACGTACC--
v: GG==ACCT-----
```

Profile-Based Multiple Alignment: A Greedy Approach

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

$$k \left\{ \begin{array}{l} u_1 = \text{ACGTACGTACGT}\ldots \\ u_2 = \text{TTAATTAATTAA}\ldots \\ u_3 = \text{ACTACTACTACT}\ldots \\ \ldots \\ u_k = \text{CCGGCCGGCCGG} \end{array} \right. \longrightarrow \left. \begin{array}{l} u_1 = \text{ACg/tTACg/tTACg/cT}\ldots \\ u_2 = \text{TTAATTAATTAA}\ldots \\ \ldots \\ u_k = \text{CCGGCCGGCCGG}\ldots \end{array} \right\} k\text{-}1$$

Example

- Consider these 4 sequences

```
s1:     GATTCA
s2:     GTCTGA
s3:     GATATT
s4:     GTCAGC
```

- with the scoring matrix: {Match = 1, Mismatch = -1, Indel = -1}

Example (continued)

- There are $\binom{4}{2} = 6$ possible pairwise 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)
```

- The best pairwise score, 2, is between s_2 and s_4

Example (continued)

- Combine s_2 and s_4:

```
s2:  G T C T G A
     | | |   |        →      s2,4:  G T C t/a G a/c
s4:  G T C A G C
```

- Giving a set of three sequences:

```
s1  :    G  A  T  T  C  A
s3  :    G  A  T  A  T  T
s2,4:    G  T  C t/a G a/c
```

- **Repeat** for $\binom{3}{2} = 3$ possible pairwise alignments

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

s1  :  GAT-TCA
s2,4:  G-TCtGa (score  = 2 - 2 + 2 - 2 + 1 - 1 + 1 = 1)

s3  :  GATAT-T
s2,4:  G-TCtGa (score  = 2 - 2 + 2 - 2 + 1 - 1 - 1 = -1)
```

Progressive Alignment

- Progressive alignment is a variation of a greedy profile alignment 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
  - Once a gap appears in a consensus string it is permanent
  - Uses profiles to compare sequences
- CLUSTAL OMEGA

Clustal Omega

- A popular multiple alignment tool commonly used 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

Clustal Omega's First Step

Pairwise alignment

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

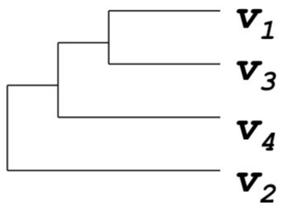
	v_1	v_2	v_3	v_4
v_1	-			
v_2	.17	-		
v_3	.87	.28	-	
v_4	.59	.33	.62	-

(.17 means 17 % identical)

ClustalW's Second Step

- 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

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



Calculate:

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

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

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

ClustalW's Third Step

- 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.

Next Time

- Other approaches to sequence alignment
- Midterm next Wednesday
- Covers material up to Lecture 13