

---

# CS481/CS583: Bioinformatics Algorithms

---

Can Alkan

EA509

`calkan@cs.bilkent.edu.tr`

<http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/>

---

# MULTIPLE SEQUENCE ALIGNMENT

---

---

# Multiple Alignment versus Pairwise Alignment

- Up until now we have only tried to align two sequences.
  - What about more than two?
  - 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

# Alignments = Paths in ...

- Align 3 sequences: ATGC, AATC, ATGC

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

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

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

# Alignment Paths

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

x coordinate

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

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

# Alignment Paths

- Align the following 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
--	---	---	---	----	---

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

x coordinate

y coordinate



# Alignment Paths

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

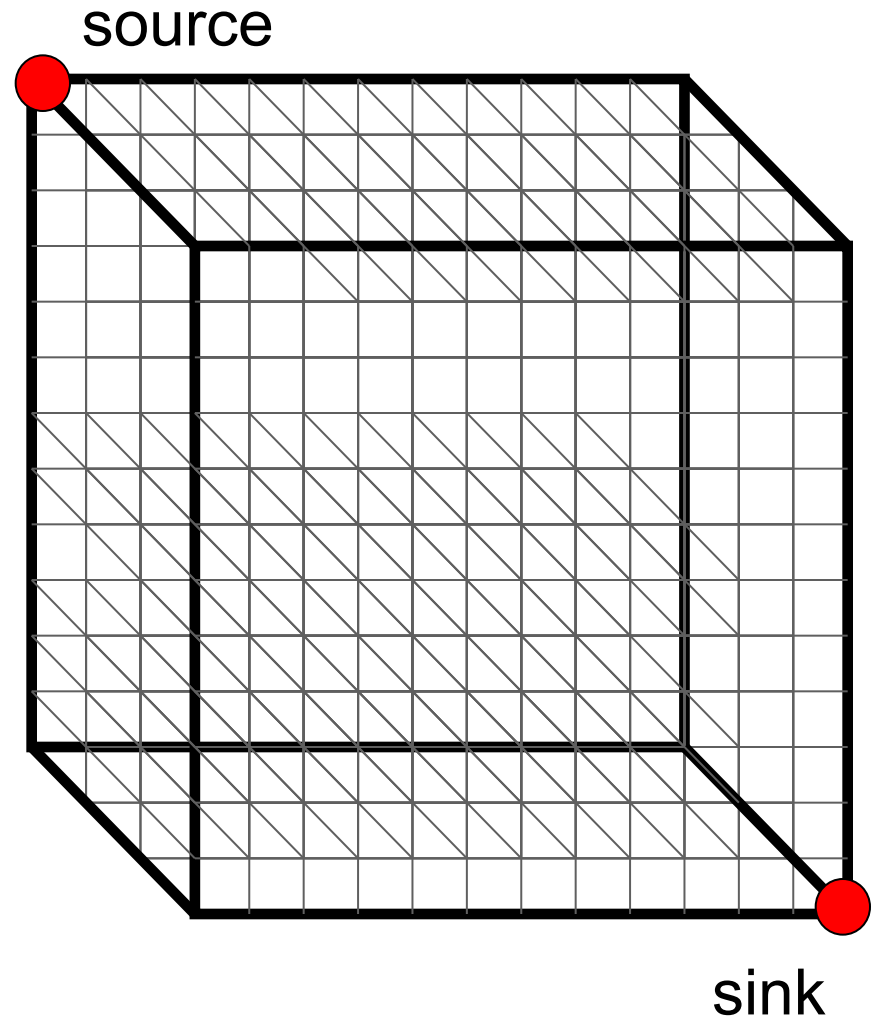
- Resulting path in (x,y,z) space:

$(0,0,0) \rightarrow (1,1,0) \rightarrow (1,2,1) \rightarrow (2,3,2) \rightarrow (3,3,3) \rightarrow (4,4,4)$

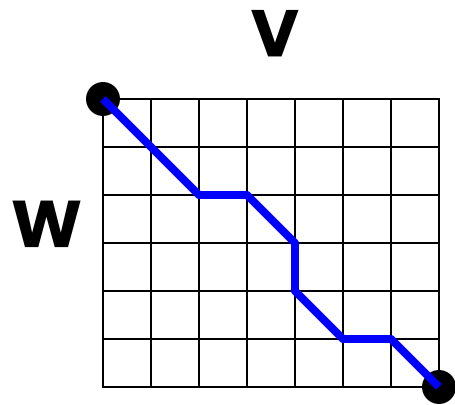


# Aligning Three Sequences

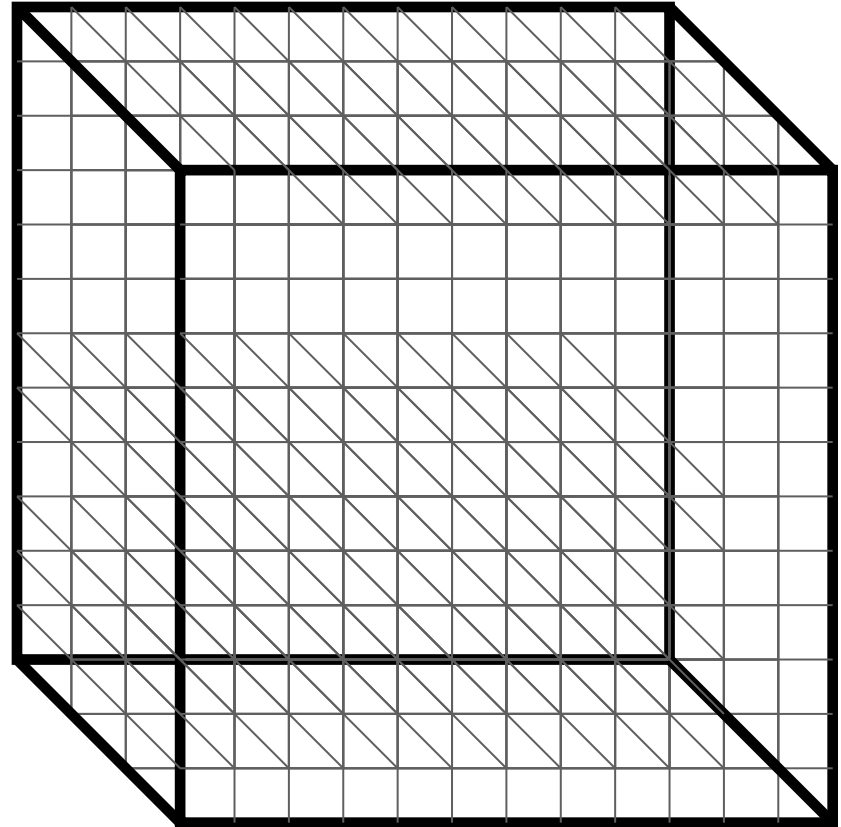
- Same strategy as aligning two sequences
- Use a 3-D “Manhattan Cube”, with each axis representing a sequence to align
- For global alignments, go from source to sink



# 2-D vs 3-D Alignment Grid

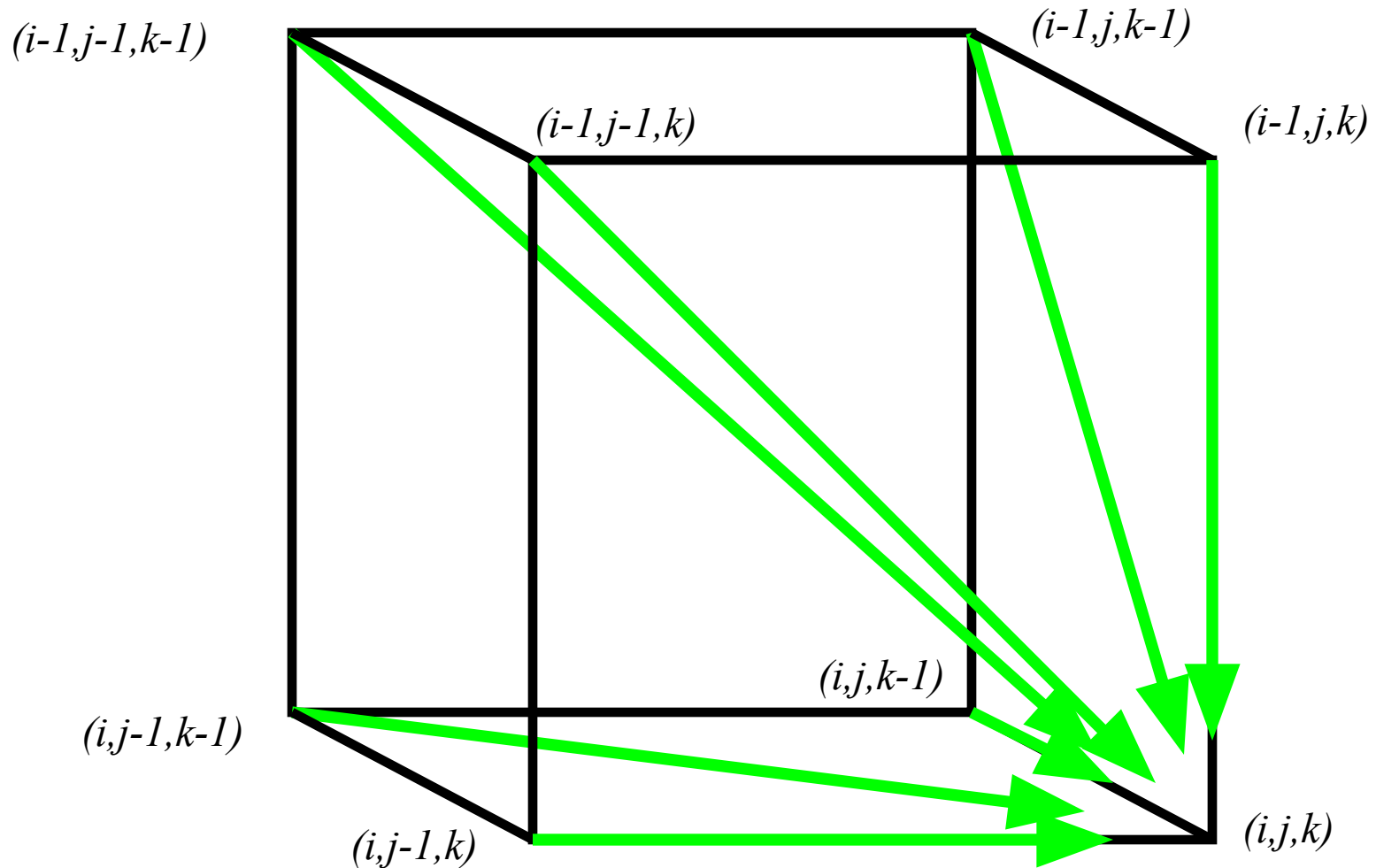


2-D edit graph



3-D edit graph

# Architecture of 3-D Alignment Cell



# Multiple Alignment: Dynamic Programming

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

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

# Multiple Alignment: Running Time

- For 3 sequences of length  $n$ , 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

# 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?

# 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

Pairwise alignments may be inconsistent



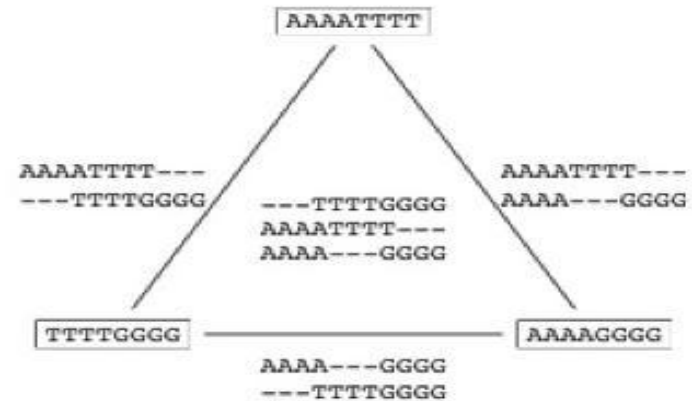
---

# 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
-

# Combining Optimal Pairwise Alignments into Multiple Alignment

Can combine pairwise alignments into multiple alignment



(a) Compatible pairwise alignments

Can **not** combine pairwise alignments into multiple alignment



(b) Incompatible pairwise alignments

# 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		

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

In the past we were aligning a **sequence against a sequence**

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

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

# Aligning alignments

- Given two alignments, can we align them?

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

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

# Aligning alignments

- Given two alignments, can we align them?
- Hint: use alignment of corresponding profiles

**x** GGGCACTGCAT

**y** GGTTACGTC--

**z** GGGAACTGCAG

**w** GGACGTACC--

**v** GGACCT-----

Combined Alignment

# Sequence to profile alignment

Sequence  $S_1$  to Profile C

## ■ Let:

- $p(y,j)$ : frequency that character  $y$  appears in column  $j$  of the profile C ---  $y \in \Sigma \cup \{'-\'}$
- $S(x,j)$ : score for aligning  $x$  with column  $j$ 
  - $\sum_y [\delta(x,y) \times p(y,j)]$  --  $\delta(x,y)$  is the “regular” scoring func.
- $V(i, j)$ : value of optimal alignment of substring  $S[1..i]$  with the first  $j$  columns of profile C

Col	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C	.6	0	0	0	1	0	0	.4	1	0	0	.6	.2	0	0

$p(C, 1) = 0.6, p(C, 2) = 0, p(C, 3) = 0, \dots, p(C, 9) = 1 \dots$

# Sequence to profile alignment

- $V(0, j) = \sum_{k \leq j} S(' - ', k)$
- $V(i, 0) = \sum_{k \leq i} \delta(S_1(k), ' - ')$  where  $S_1(k)$  is the  $k^{\text{th}}$  character of  $S_1$

- $$V(i, j) = \max \begin{cases} V[i - 1, j - 1] + S(S_1(i), j) \\ V[i - 1, j] + \delta(S_1(i), ' - ') \\ V[i, j - 1] + S(' - ', j) \end{cases}$$



# Sequence to profile

*Match = 2*

*Mismatch = -1*

*Gap = -1*

	Profile														
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14
C	-1	0.8	-0.2	-1.2	-2.2	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
A	-2	-0.2	2.8	1.8	0.8	-0.2	-1.2	-1	-2	-3	-4	-5	-6	-7	-8
G	-3	-1.2	1.8	4.8	3.8	2.8	1.8	0.8	-0.2	-1.2	-2.2	-3.2	-4.2	-5.2	-5
G	-4	-2.2	0.8	3.8	4.4	3.4	2.4	1.4	0.4	-0.6	-1.6	-2.6	-3.6	-4	-3.2
T	-5	-3.2	-0.2	2.8	3.4	3.4	5.4	4.4	3.4	2.4	1.4	0.4	-0.6	-1.6	-2.6
A	-6	-4.2	-1.2	1.8	2.4	2.4	4.4	7.4	6.4	5.4	4.4	3.4	2.4	1.4	0.4
C	-7	-5.2	-2.2	0.8	1.4	4.4	3.4	6.4	7.6	8.4	7.4	6.4	5.4	4.4	3.4
C	-8	-6.2	-3.2	-0.2	0.4	3.4	3.4	5.4	6.6	9.6	8.6	8.2	7.2	6.2	5.2
A	-9	-7.2	-4.2	-1.2	-0.6	2.4	2.4	5.4	5.6	8.6	11	10	9	8	7
C	-10	-8.2	-5.2	-2.2	-1.6	1.4	1.4	4.4	5.6	7.6	10	11.8	10.8	9.8	8.8
G	-11	-9.2	-6.2	-3.2	-2.6	0.4	0.4	3.4	4.6	6.6	9	10.8	10.8	11	11.8
G	-12	-10.2	-7.2	-4.2	-3.6	-0.6	-0.6	2.4	3.6	5.6	8	9.8	9.8	11	13

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

# Sequence to profile alignment

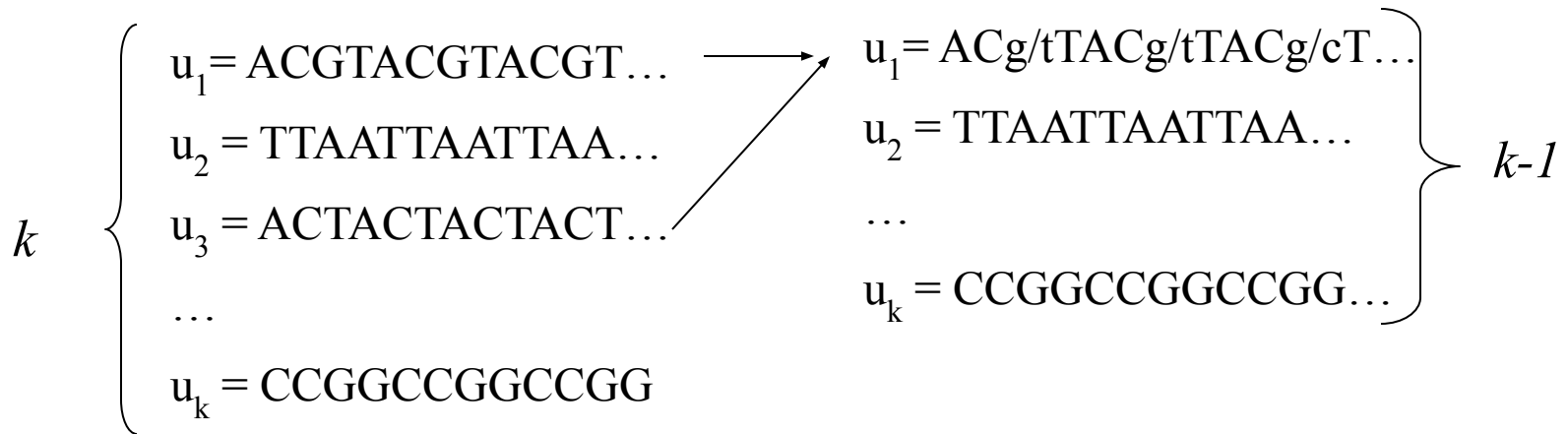
New Profile C:

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

A		1				1			.83				
C	.67			.83		.5	1		.67	.17			
G		1	.33						.17		.50	1	
T	.16				1	.5					.17		
-	.17		.67	.17					.33	.83	.33		

# 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/profiles. **Repeat**
- This is a heuristic greedy method



# Greedy Approach: Example

- Consider these 4 sequences

*s*1    GATTCA

*s*2    GTCTGA

*s*3    GATATT

*s*4    GTCAGC

# Greedy Approach: Example (cont'd)

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

*s2*    **GTC**TGA

*s4*    **GTC**AGC (score = 2)

*s1*    **GATT**CA--

*s4*    **G-T-CA**GC (score = 0)

*s1*    **GAT-T**CA

*s2*    **G-TCT**GA (score = 1)

*s2*    **G-TCT**GA

*s3*    **GATAT**-T (score = -1)

*s1*    **GAT-T**CA

*s3*    **GATAT-T** (score = 1)

*s3*    **GAT-AT**T

*s4*    **G-TCA**GC (score = -1)

# Greedy Approach: Example (cont'd)

$s_2$  and  $s_4$  are closest; combine:

$s_2$	<b>GTCTGA</b>	}	$s_{2,4}$ (profile)	<b>GT</b> C <b>t/a</b> <b>G</b> a/ <b>c</b> <b>A</b>
$s_4$	<b>GTCAGC</b>			

new set of 3 sequences:

$s_1$	GATTCA
$s_3$	GATATT
$s_{2,4}$	<b>GT</b> C <b>t/a</b> <b>G</b> a/ <b>c</b>

---

# 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
-

---

# ClustalW

- Popular multiple alignment tool
  - '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 each other 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)

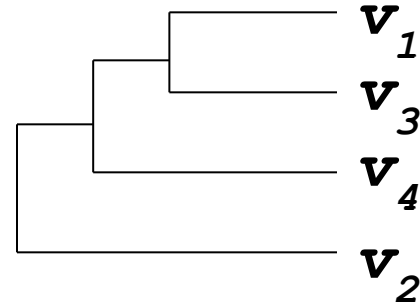
---

## Step 2: Guide Tree

- Create Guide Tree using the similarity matrix
    - ClustalW uses the neighbor-joining method
    - 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:

$v_{1,3}$

= alignment ( $v_1, v_3$ )

$v_{1,3,4}$

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


$v_{1,2,3,4}$

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

# 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-LDLTGGLPEATTPESEEAF TLPLLNDPEPK-PSLEPVKNISNMELKAEPFD
FOS_MOUSE    PEEMSVAS-LDLTGGLPEASTPESEEAF TLPLLNDPEPK-PSLEPVKSISNVELKAEPFD
FOS_CHICK     SEELAAATALDLG-----APSPAAAEFAFALPLMTEAPPAVPPKEPSG--SGLELKAEPFD
FOSB_MOUSE    PGPGPLAEVRDLPG-----STSAKEDGFGWLLPPPPPPP-----LPFQ
FOSB_HUMAN    PGPGPLAEVRDLPG-----SAPAKEDGFSWLLPPPPPPP-----LPFQ
.      .  :      **  .      :..  *:. *      *      .  *      **:
```



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

---

# SCORING ALIGNMENTS

---

---

# 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

# Entropy

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

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

A  
A  
A  
A

AA  
AA  
AT  
ATC



# Entropy: Example

$$\text{entropy} \begin{pmatrix} A \\ A \\ A \\ A \end{pmatrix} = 0 \quad \text{Best case}$$

$$\text{Worst case} \quad \text{entropy} \begin{pmatrix} A \\ T \\ G \\ C \end{pmatrix} = -\sum \frac{1}{4} \log \frac{1}{4} = -4 \left( \frac{1}{4} * -2 \right) = 2$$

# Multiple Alignment: Entropy Score

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

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

# 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 =  $-[1 * \log(1) + 0 * \log 0 + 0 * \log 0 + 0 * \log 0]$   
= 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 =  $0 + 0.811 + 2.0 = +2.811$

# 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

---

Not necessarily optimal

# Sum of Pairs Score(SP-Score)

- Consider pairwise alignment of sequences  $a_i$  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:

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

# Computing SP-Score

Aligning 4 sequences: 6 pairwise alignments

Given  $a_1, a_2, a_3, a_4$ :

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

# SP-Score: Example

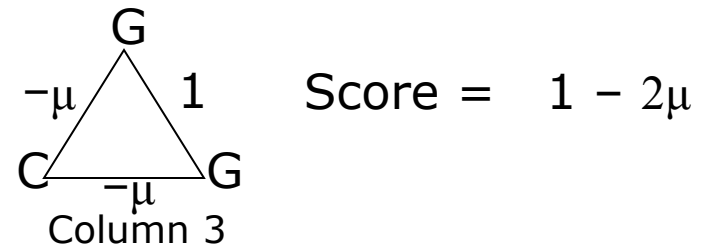
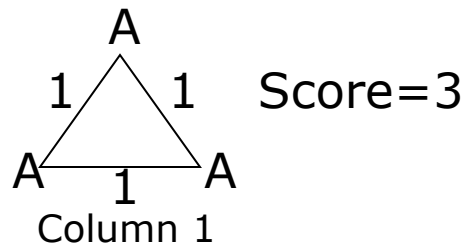
$a_1$  ATG-C-AAT

· A-G-CATAT

$a_k$  ATCCCATTT

To calculate each column:

$$s'(a_1 \dots a_k) = \sum_{i,j} s^*(a_i, a_j) \leftarrow \binom{n}{2} \text{ Pairs of Sequences}$$



---

# Back to guide trees for MSA

- Guide tree construction
  - UPGMA
  - Neighbor Joining
  - ....
- Easy MSA: Center Star



# Star alignments

- Construct multiple alignments using pair-wise alignment relative to a fixed sequence
- Out of a set  $S = \{S_1, S_2, \dots, S_r\}$  of sequences, pick sequence  $S_c$  that maximizes

$$\text{star\_score}(c) = \sum \{\text{sim}(S_c, S_i) : 1 \leq i \leq r, i \neq c\}$$

where  $\text{sim}(S_i, S_j)$  is the optimal score of a pair-wise alignment between  $S_i$  and  $S_j$

# Star alignment Algorithm

1. Compute  $\text{sim}(S_i, S_j)$  for every pair  $(i,j)$
2. Compute  $\text{star\_score}(i)$  for every  $i$
3. Choose the index  $c$  that minimizes  $\text{star\_score}(c)$  and make it the center of the star
4. Produce a multiple alignment  $M$  such that, for every  $i$ , the induced pairwise alignment of  $S_c$  and  $S_i$  is the same as the optimum alignment of  $S_c$  and  $S_i$ .

# Star alignment example

$S_c$  AA--CCTT

$S_c$  A-ACC-TT

$S_1$  AATGCC--

$S_2$  AGACCGT-

$S_c$  A-A--CC-TT

$S_1$  A-ATGCC---

$S_2$  AGA--CCGT-

# Multiple Alignment: History



## **1975 Sankoff**

*Formulated multiple alignment problem and gave dynamic programming solution*

## **1988 Carrillo-Lipman**

*Branch and Bound approach for MSA*

## **1990 Feng-Doolittle**

*Progressive alignment*

## **1994 Thompson-Higgins-Gibson-ClustalW**

*Most popular multiple alignment program*

## **1998 Morgenstern et al.-DIALIGN**

*Segment-based multiple alignment*

## **2000 Notredame-Higgins-Heringa-T-coffee**

*Using the library of pairwise alignments*

## **2004 MUSCLE**

---

# PARTIAL ORDER ALIGNMENT

---

# Basics

Assume 2 sequences:

seq1 : CCGCTTTTCCGC

seq2 : CCGCAAAACCGC

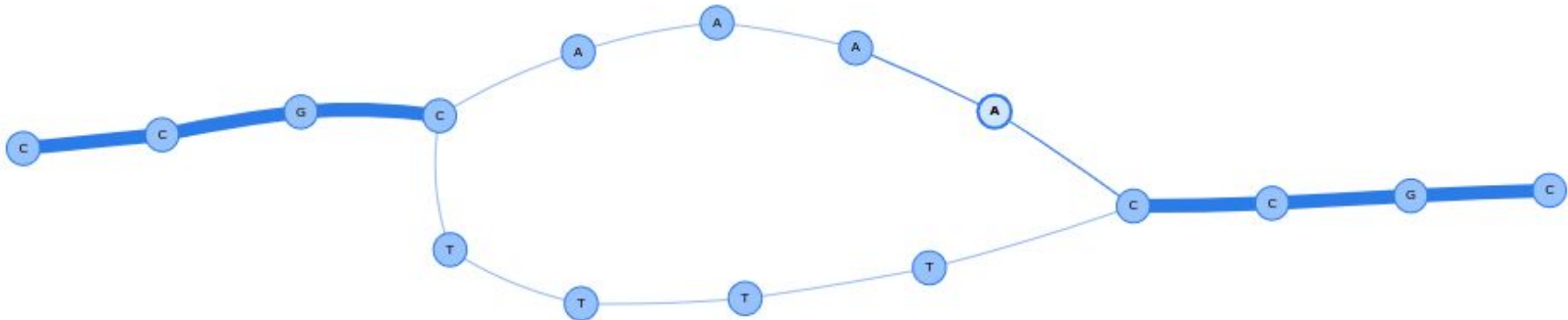
Alignments:

CCGC----TTTTCGCG	CCGCTTTT----CCGC	CCGC-TT-TT--CGCG	CCGC-T-T-T-TCCGC
CCGCAAAA----CGCG	CCGC----AAAACCGC	CCGCA--A--AACCGC	CCGCA-A-A-A-CCGC

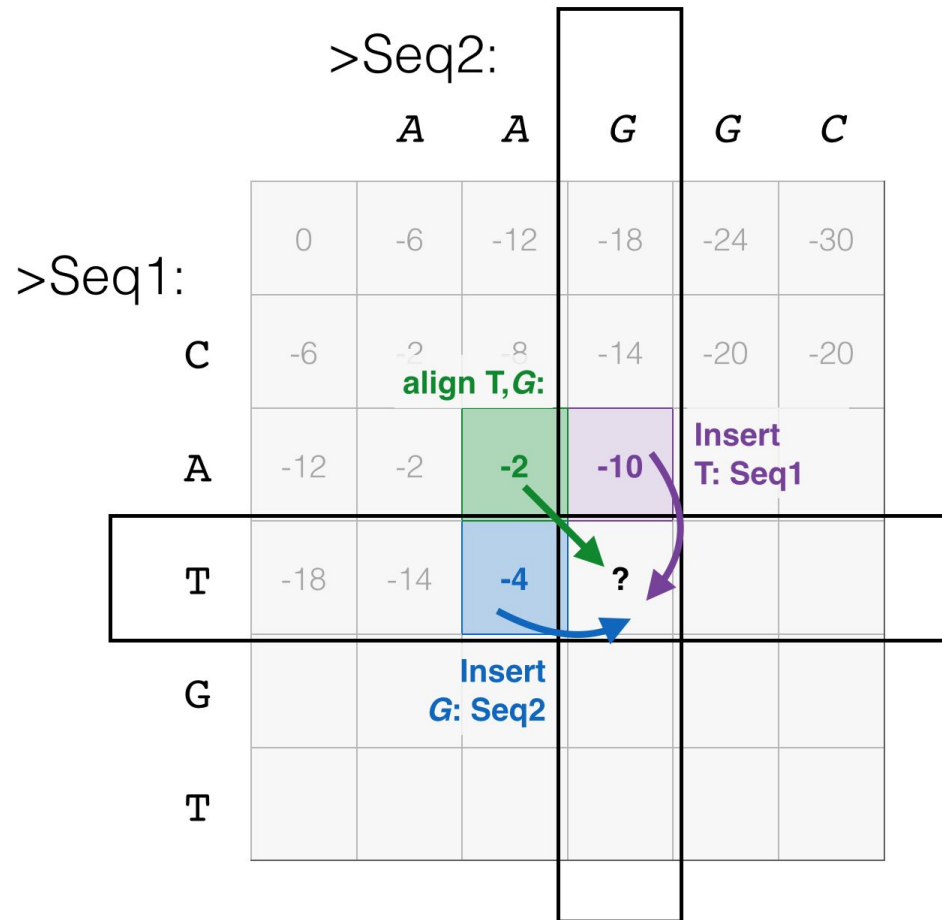
# DAG representation

## Alignments:

CCGC----TTTTTCGCG	CCGCTTTT----CCGC	CCGC-TT-TT--CGCG	CCGC-T-T-T-TCCGC
CCGCAAAA----CGCG	CCGC----AAAACCGC	CCGCA--A--AACCGC	CCGCA-A-A-A-CCGC

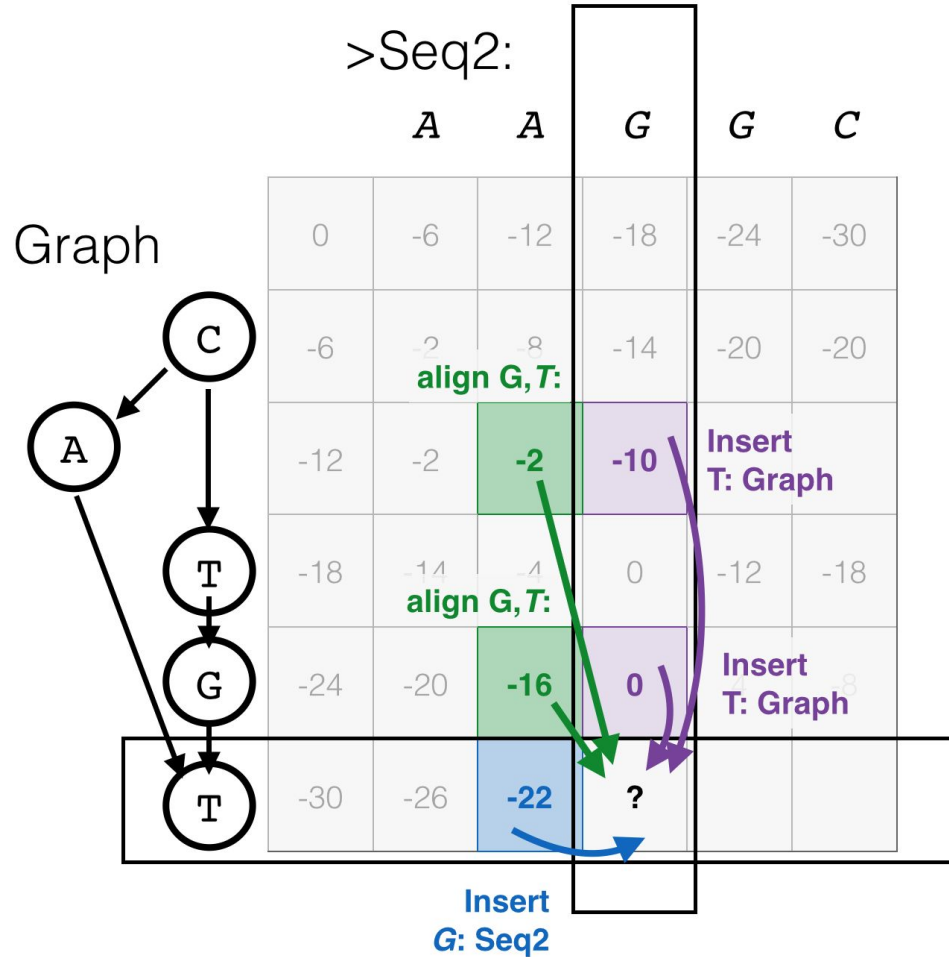


# “Regular” alignment





# String to Graph Alignment



# String to Graph Alignment

- The primary difference for the purposes of dynamic programming is that while a base in a sequence has exactly one predecessor, a base in a graph can have two or more. Thus, the score may have come from one of several previous locations for the same (graph) Insert or Align moves being considered; and thus those scores must be considered too in determining the best previous position. (Note that insertions from the sequence are unchanged).
- The only difference deep inside the dynamic programming loop is that multiple previous scores (and any associated gap-open information) must be considered for insertions or alignments of the graph base. This is implemented by a loop over predecessors for the current base, and all else remains the same.

# String to Graph Alignment

**Problem:** order of dependencies in the graph

**Solution:** topological sort

## Steps:

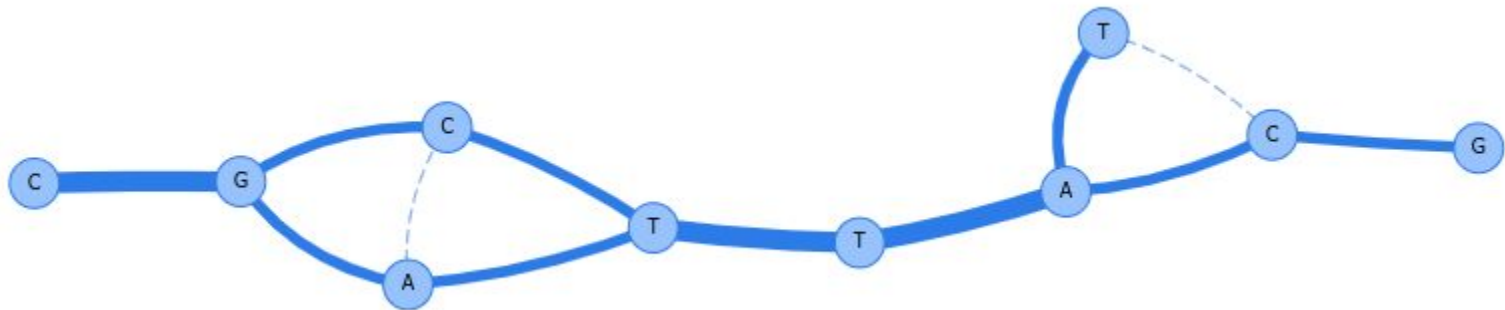
- Perform a topological sort if the graph has been updated
- Do the dynamic programming step as usual, with:
  - ❑ The graph nodes visited in the order of the topological sort, and
  - ❑ Considering all valid predecessors for align/insert moves.

# Pairwise alignment example

CGATTACG

||.||||.

CGCTTAT-



# POA implementations

- Original version:  
<https://sourceforge.net/projects/poamsa/>
- Poapy: <https://github.com/ljdursi/poapy>
- abPOA: <https://github.com/Xinglab/abPOA>
- SIMD-accelerated POA:  
<https://github.com/rvaser/spoa>