CSE8803/CX4803 Machine Learning in Computational Biology

Lecture 3: Sequence Alignment II

Xiuwei Zhang

School of Computational Science and Engineering

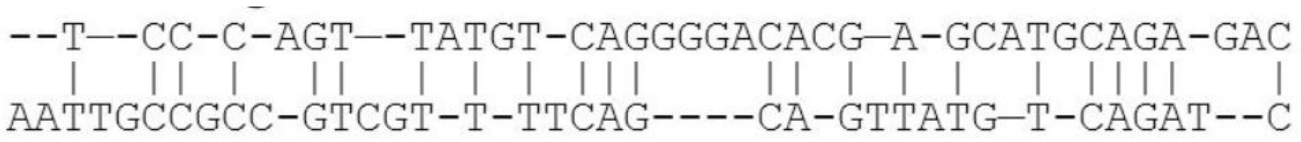
Based on slides from Carl Kingsford

Paper presentation teams

- 77 students in total, 33 presentation slots
- 22 groups of 2 students; 11 groups of 3 students
- Submit your team information to Canvas->Quizzes by 1/28 Friday (no grace period)
- We may adjust the teams (randomly)
- Teams can also slightly change after midterm withdraw

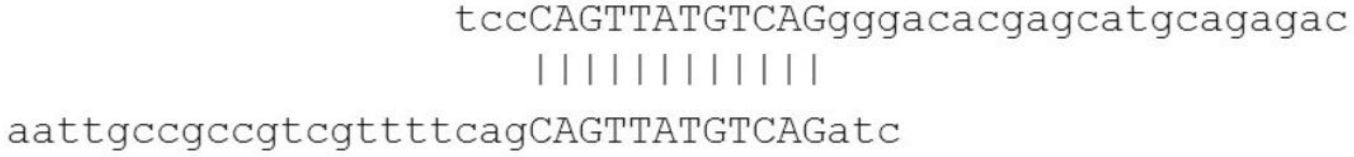
Global alignment vs local alignment

- Global alignment



```
--T—-CC-C-AGT—-TATGT-CAGGGACACG—A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || | | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG—T-CAGAT--C
```

- Local alignment



```
                  tccCAGTTATGTCAGggacacgagcatgcagagac
                     ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

Today’s outline

Local alignment

More details on scoring mismatches and gaps

Multiple sequence alignment

Local alignment

- **Local** alignment is much more common than **global** alignment
    - Example: aligning two protein sequences that have a common domain but are otherwise different
    - Mapping short reads to the genome
- Compared to **global** alignment, the **local** alignment problem appears to be significantly more complex
- **Naïve approach**:
    - Given that we know how to compute the global alignment between two sequences in O(*mn*) time
    - We can take all possible combinations of substrings of ***x*** and substrings of ***y***
    - How many all possible combinations of substrings?

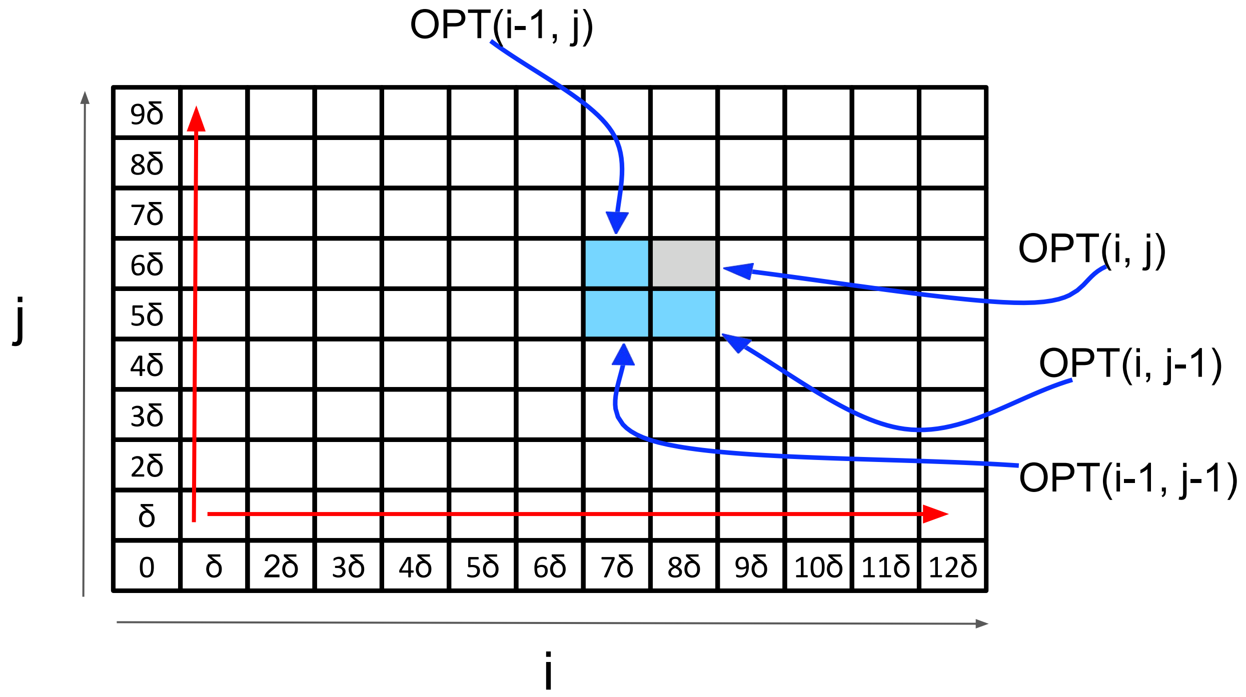
The running time will be $O(m^3n^3)$

Recall: Global Alignment Matrix

OPT(*i*,*j*) contains the score for the best alignment between:

the first *i* characters of string *x* [prefix *i* of *x*]

the first *j* character of string *y* [prefix *j* of *y*]



Maximization vs. Minimization

Global alignment:

$$OPT(i,j) = \min \begin{cases} \text{cost}(a_i, b_j) + OPT(i-1, j-1) \\ \text{gap} + OPT(i-1, j) \\ \text{gap} + OPT(i, j-1) \end{cases}$$

$$OPT(i,0) = i \times \text{gap} \text{ and } OPT(0,j) = j \times \text{gap}$$

Sequence Similarity: replace *min* with a *max* and *negate* the parameters.

gap penalty → gap benefit (probably negative)
cost → score
Minimization → maximization

Local Alignment

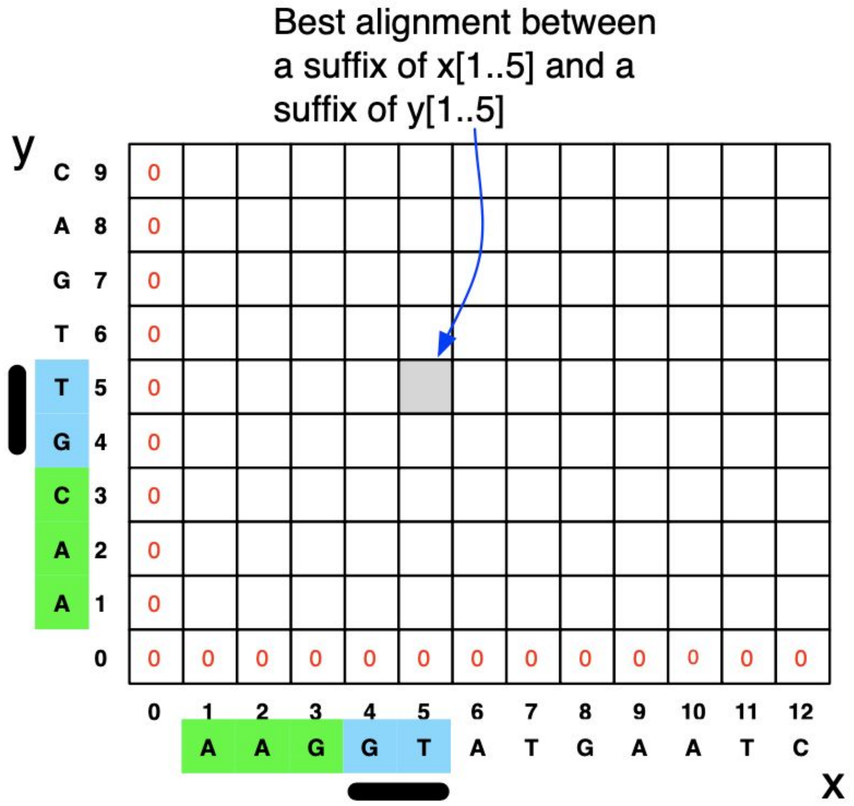
New meaning of entry of matrix entry:

A[i, j] = best score between:

some suffix of *x*[1…*i*]

and

some suffix of *y*[1…*j*]



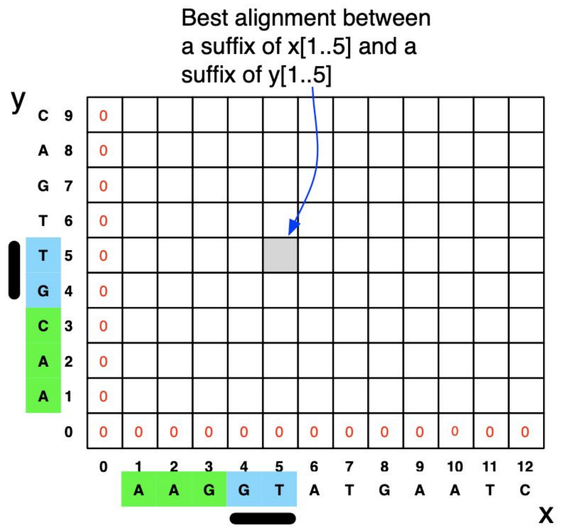
How do we fill in the local alignment matrix?

$$A[i, j] = \max \begin{cases} A[i, j-1] + \text{gap} & (1) \\ A[i-1, j] + \text{gap} & (2) \\ A[i-1, j-1] + \text{match}(i, j) & (3) \\ 0 & (4) \end{cases}$$

(1), (2), and (3): same cases as before: gap in x, gap in y, match x and y

New case: 0
allows you to say the best alignment between a suffix of *x* and a suffix of *y* is the empty alignment.

Lets us "start over"



Local Alignment

- Initialize first row and first column to be 0.
- The score of the best local alignment is the largest value in the entire array.
- To find the actual local alignment:
    - start at an entry with the maximum score
    - traceback as usual
    - stop when we reach an entry with a score of 0

Local Alignment Example #1

X = AG**CGT**AG

Y = CT**CGT**C

Score(match) = 10

Score(mismatch)= -5

Score(gap) = -7

	*	A	G	C	G	T	A	G
*	0	0	0	0	0	0	0	0
C	0	0	0	10	3	0	0	0
T	0	0	0	3	5	13	6	0
C	0	0	0	10	3	6	8	1
G	0	0	10	3	20	13	6	18
T	0	0	3	5	13	30	23	16
C	0	0	0	13	6	23	25	18

Note: this table written top-to-bottom instead of bottom-to-top

Local Alignment Example #2

X = best**oft**imes

Y = s**oft**en

Score(match) = 10

Score(mismatch) = -5

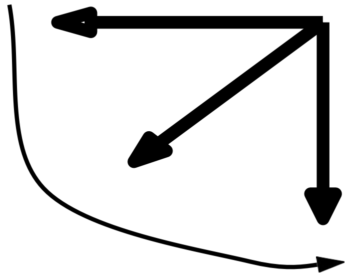
Score(gap) = -7

	*	b	e	s	t	o	f	t	i	m	e	s
*	0	0	0	0	0	0	0	0	0	0	0	0
s	0	0	0	10	3	0	0	0	0	0	0	10
o	0	0	0	3	5	13	6	0	0	0	0	3
f	0	0	0	0	0	6	23	16	9	2	0	0
t	0	0	0	0	10	3	16	33	26	19	12	5
e	0	0	10	3	3	5	9	26	28	21	29	22
n	0	0	3	5	0	0	2	19	21	23	22	24

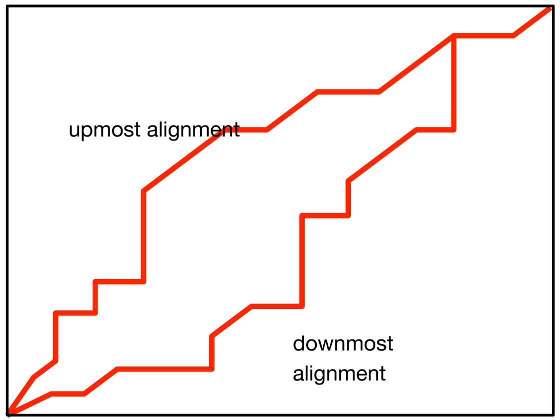
Note: this table written top-to-bottom instead of bottom-to-top

Multiple optimal aligntments

When there are ties in the max{ }, we have a choice about which arrow to follow.



This gives us multiple optimal alignments which have the same cost/score.

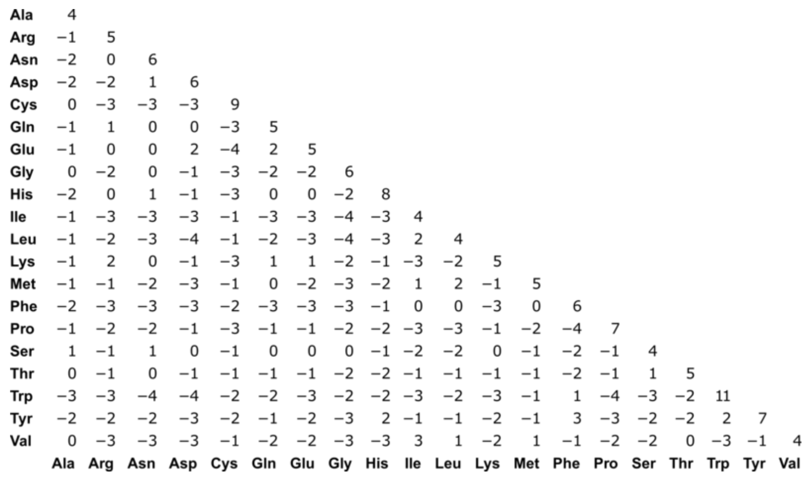


Local / Global Recap

- Alignment cost sometimes called the "edit distance" between two strings.
- Algorithm for global alignment is sometimes called "Needleman-Wunsch"
- Algorithm for local alignment is sometimes called "Smith-Waterman"
- Same basic algorithm, however.

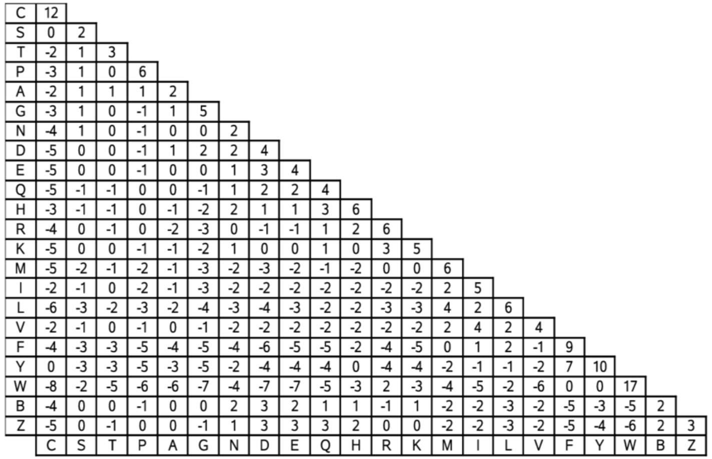
Scoring mismatches

BLOSUM (BLOcks SUbstitution Matrix) matrix



Ala	4																			
Arg	−1	5																		
Asn	−2	0	6																	
Asp	−2	−2	1	6																
Cys	0	−3	−3	−3	9															
Gln	−1	1	0	0	−3	5														
Glu	−1	0	0	2	−4	2	5													
Gly	0	−2	0	−1	−3	−2	−2	6												
His	−2	0	1	−1	−3	0	0	−2	8											
Ile	−1	−3	−3	−3	−1	−3	−3	−4	−3	4										
Leu	−1	−2	−3	−4	−1	−2	−3	−4	−3	2	4									
Lys	−1	2	0	−1	−3	1	1	−2	−1	−3	−2	5								
Met	−1	−1	−2	−3	−1	0	−2	−3	−2	1	2	−1	5							
Phe	−2	−3	−3	−3	−2	−3	−3	−3	−1	0	0	−3	0	6						
Pro	−1	−2	−2	−1	−3	−1	−1	−2	−2	−3	−3	−1	−2	−4	7					
Ser	1	−1	1	0	−1	0	0	0	−1	−2	−2	0	−1	−2	−1	4				
Thr	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	1	5			
Trp	−3	−3	−4	−4	−2	−2	−3	−2	−2	−3	−2	−3	−1	1	−4	−3	−2	11		
Tyr	−2	−2	−2	−3	−2	−1	−2	−3	2	−1	−1	−2	−1	3	−3	−2	−2	2	7	
Val	0	−3	−3	−3	−1	−2	−2	−3	−3	3	1	−2	1	−1	−2	−2	0	−3	−1	4
	Ala	**Arg**	**Asn**	**Asp**	**Cys**	**Gln**	**Glu**	**Gly**	**His**	**Ile**	**Leu**	**Lys**	**Met**	**Phe**	**Pro**	**Ser**	**Thr**	**Trp**	**Tyr**	**Val**

PAM (Point accepted mutation) matrix



C	12																					
S	0	2																				
T	-2	1	3																			
P	-3	1	0	6																		
A	-2	1	1	1	2																	
G	-3	1	0	-1	1	5																
N	-4	1	0	-1	0	0	2															
D	-5	0	0	-1	1	2	2	4														
E	-5	0	0	-1	0	0	1	3	4													
Q	-5	-1	-1	0	0	-1	1	2	2	4												
H	-3	-1	-1	0	-1	-2	2	1	1	3	6											
R	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6										
K	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5									
M	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6								
I	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5							
L	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6						
V	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4					
F	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9				
Y	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10			
W	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17		
B	-4	0	0	-1	0	0	2	3	2	1	1	-1	1	-2	-2	-3	-2	-5	-3	-5	2	
Z	-5	0	-1	0	0	-1	1	3	3	3	2	0	0	-2	-2	-3	-2	-5	-4	-6	2	3
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	B	Z

For DNA sequences, A-G and C-T mismatches have less penalty than other mismatches

Varying gap cost models

- Linear gap penalty: $w(k)=k*\delta$
- Affine gap penalty
    - Big initial cost for starting (or ending) a gap
    - Incremental penalty for each additional gap
    - $w(k)=h+k*\delta$ if $k>=1$; $w(k)=0$ if $k=0$.
- General gap penalty
    - Any cost function
    - May not be computable with the same DP model

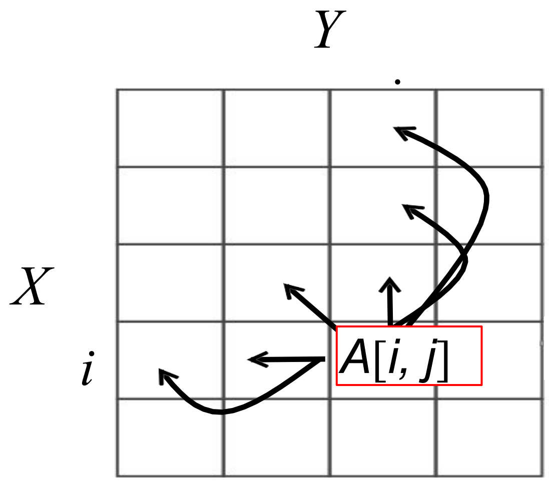
Dynamic programming for affine gaps?

Possible cases:

A[i-1,j-1]

A[i-1, j], A[i-2, j], …

A[i, j-1], A[i, j-2], …



- We can use the same approach we used for the linear gap, but…
- Running time increases from O(*mn*) to O(*mn(m+n)*)
- For *m*=*n*, the increase is from $O(n^2)$ to $O(n^3)$

DP for affine gap

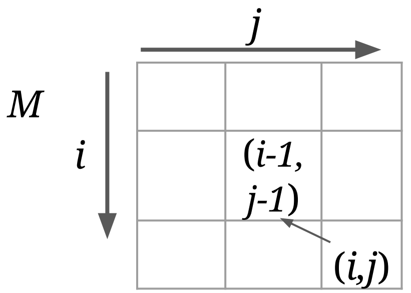
- We can reduce the time to $O(mn)$, but we need 3 matrices instead of 1

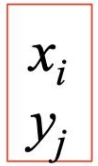
$M(i,j)$ = score of the best alignment of $x[1\ldots i]$ with $y[1\ldots j]$ given that $\underline{x[i] \text{ is aligned to } y[j]}$ $\boxed{\begin{matrix} x_i \\ y_j \end{matrix}}$

$I_x(i,j)$ = score of the best alignment of $x[1\ldots i]$ with $y[1\ldots j]$ given that $\underline{x[i] \text{ is aligned to a gap}}$ $\boxed{\begin{matrix} x_i \\ - \end{matrix}}$

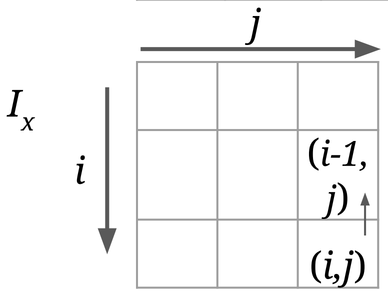
$I_y(i,j)$ = score of the best alignment of $x[1\ldots i]$ with $y[1\ldots j]$ given that $\underline{y[j] \text{ is aligned to a gap}}$ $\boxed{\begin{matrix} - \\ y_j \end{matrix}}$

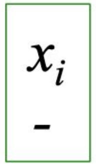
DP for affine gap



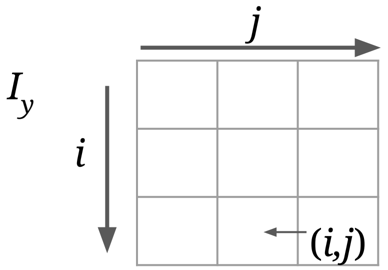


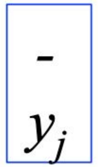
$$M(i,j) = \max \begin{cases} M(i-1,j-1) + s(x_i,y_j) \\ I_x(i-1,j-1) + s(x_i,y_j) \\ I_y(i-1,j-1) + s(x_i,y_j) \end{cases}$$





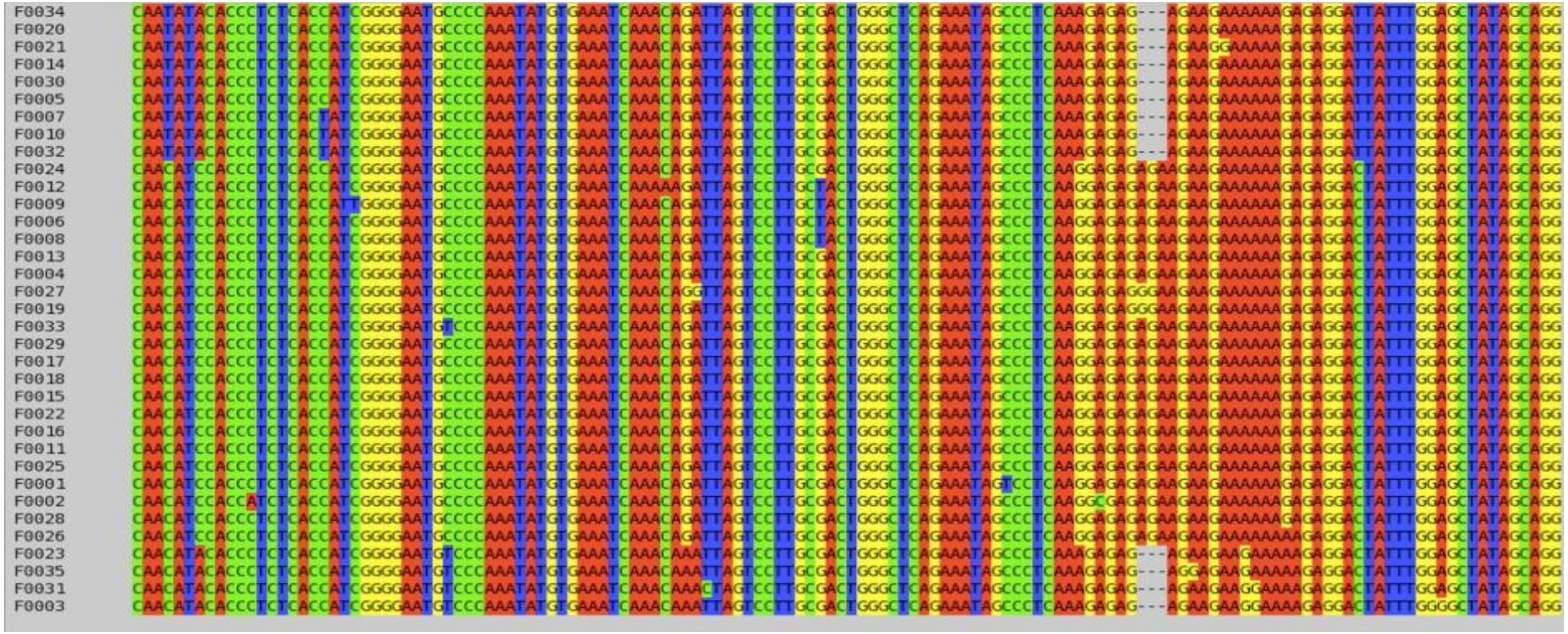
$$I_x(i,j) = \max \begin{cases} M(i-1,j) + h + g \\ I_x(i-1,j) + g \\ I_y(i-1,j) + h + g \end{cases}$$





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

Multiple Sequence Alignment (MSA)



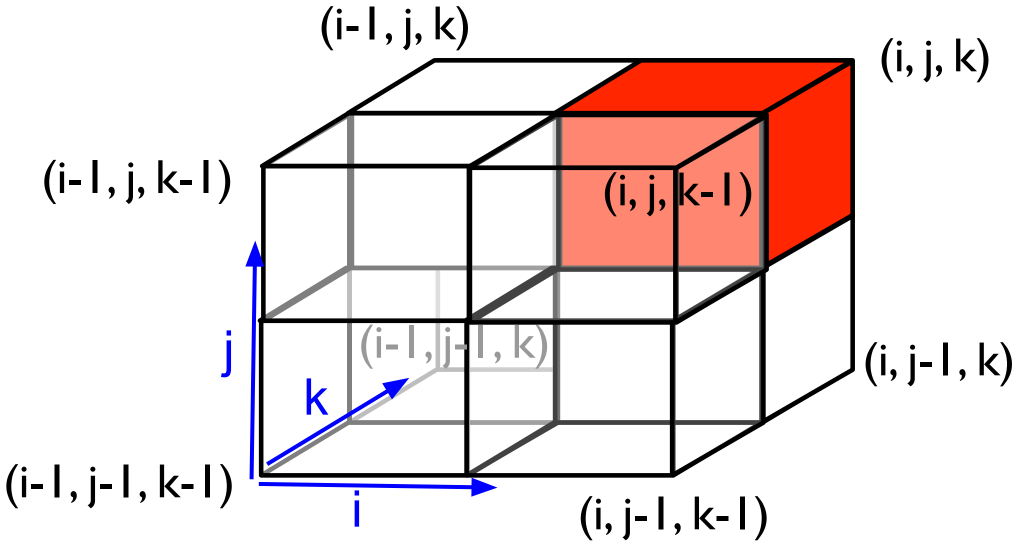
```
F0034   CAATATACACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAAAAAGAGAGGATTATTTGGAGCTATAGCAGG
F0020   CAATATACACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAAAAAGAGAGGATTATTTGGAGCTATAGCAGG
F0021   CAATATACACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGGAAAAAGAGAGGATTATTTGGAGCTATAGCAGG
F0014   CAATATACACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAAAAAGAGAGGATTATTTGGAGCTATAGCAGG
F0030   CAATATACACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAAAAAGAGAGGATTATTTGGAGCTATAGCAGG
F0005   CAATATACACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAAAAAGAGAGGATTATTTGGAGCTATAGCAGG
F0007   CAATATACACCCTCTCACTATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAAAAAGAGAGGATTATTTGGAGCTATAGCAGG
F0010   CAATATACACCCTCTCACTATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAAAAAGAGAGGATTATTTGGAGCTATAGCAGG
F0032   CAATATACACCCTCTCACTATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAAAAAGAGAGGATTATTTGGAGCTATAGCAGG
F0024   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0012   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAAAAGATTAGTCCTTGCTACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0009   CAACATCCACCCTCTCACCATTGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCTACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0006   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCTACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0008   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCTACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0013   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0004   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0027   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGGTTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGGGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0019   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGGGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0033   CAACATCCACCCTCTCACCATCGGGGAATGTCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0029   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0017   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0018   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0015   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0022   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0016   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0011   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0025   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0001   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGTCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0002   CAACATCCACCATCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGCGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0028   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGGACTATTTGGAGCTATAGCAGG
F0026   CAACATCCACCCTCTCACCATCGGGGAATGCCCCAAATATGTGAAATCAAACAGATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAAAGAGGACTATTTGGAGCTATAGCAGG
F0023   CAACATACACCCTCTCACCATCGGGGAATGTCCCAAATATGTGAAATCAAACAAATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAGAAAAAGAGGACTATTTGGAGCTATAGCAGG
F0035   CAACATACACCCTCTCACCATCGGGGAATGTCCCAAATATGTGAAATCAAACAAATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGGAGAAGAAAAAGAGGACTATTTGGAGCTATAGCAGG
F0031   CAACATACACCCTCTCACCATCGGGGAATGTCCCAAATATGTGAAATCAAACAAACTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAGGAAAAGAGGACTATTTGGAGCTATAGCAGG
F0003   CAACATACACCCTCTCACCATCGGGGAATGTCCCAAATATGTGAAATCAAACAAATTAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAGGAAAAGAGGACTATTTGGGGCTATAGCAGG
```

Multiple sequence alignment: find more subtle patterns & find common patterns between all sequence.

From 2 sequences to multiple - Dynamic Programming?

Suppose you had just 3 sequences.

Apply the same DP idea as sequence alignment for 2 sequences, but now with a 3-dimensional matrix

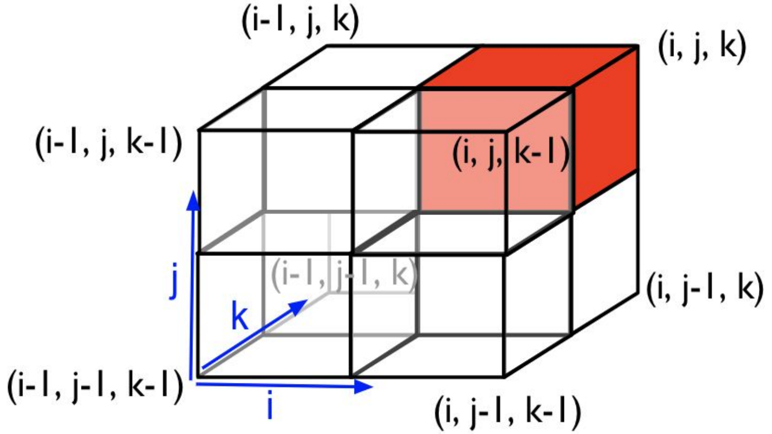


DP Recurrence for 3 sequences

$$A[i, j, k] = \min \begin{cases} \text{cost}(x_i, y_j, z_k) + A[i-1, j-1, k-1] \\ \text{cost}(x_i, -, -) + A[i-1, j, k] \\ \text{cost}(x_i, y_j, -) + A[i-1, j-1, k] \\ \text{cost}(-, y_j, z_k) + A[i, j-1, k-1] \\ \text{cost}(-, y_j, -) + A[i, j-1, k] \\ \text{cost}(x_i, -, z_k) + A[i-1, j, k-1] \\ \text{cost}(-, -, z_k) + A[i, j, k-1] \end{cases}$$

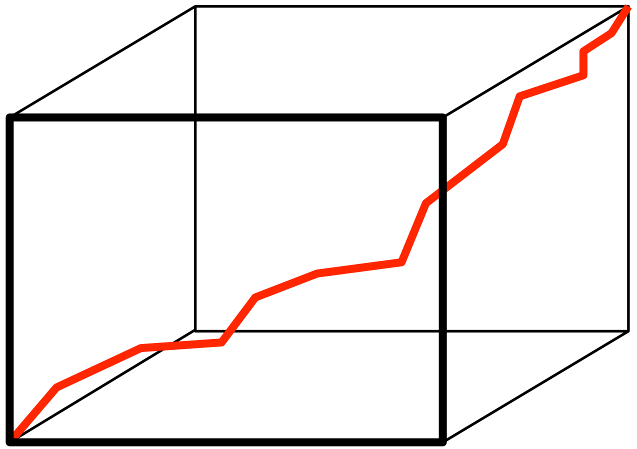
Every possible pattern for the gaps.

2^3-1 cases.



Running time

- n^3 subproblems, each takes 2^3 time $\Rightarrow$ O(n^3) time.
- For p sequences: n^p subproblems, each takes 2^p time for the min $\Rightarrow$ O($n^p 2^p$)
- Even O(n^3) is often too slow for the length of sequences encountered in practice.
- One solution: approximation algorithm.



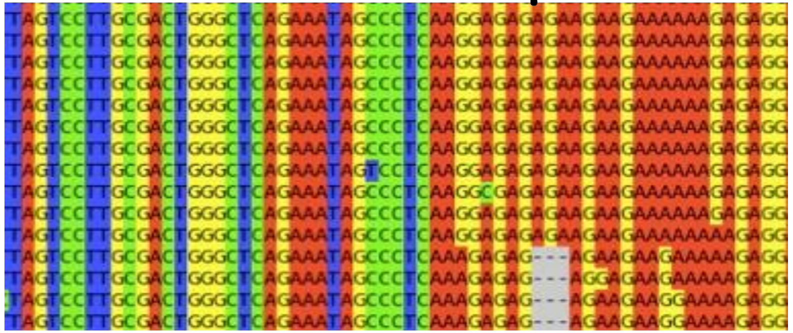
Generalizing Alignment to > 2 Sequences

Input: Sequences $S_1, S_2, \ldots, S_p$

Let $cost(x_1,x_2,\ldots x_p)$ be a user-supplied function that computes the quality of a column: an alignment between characters $x_1, x_2, \ldots x_p$.

- **Goal**: find alignment M to **minimize** $\sum$ cost of the columns:

$cost(x_1,x_2,\ldots x_p)$ = cost( )



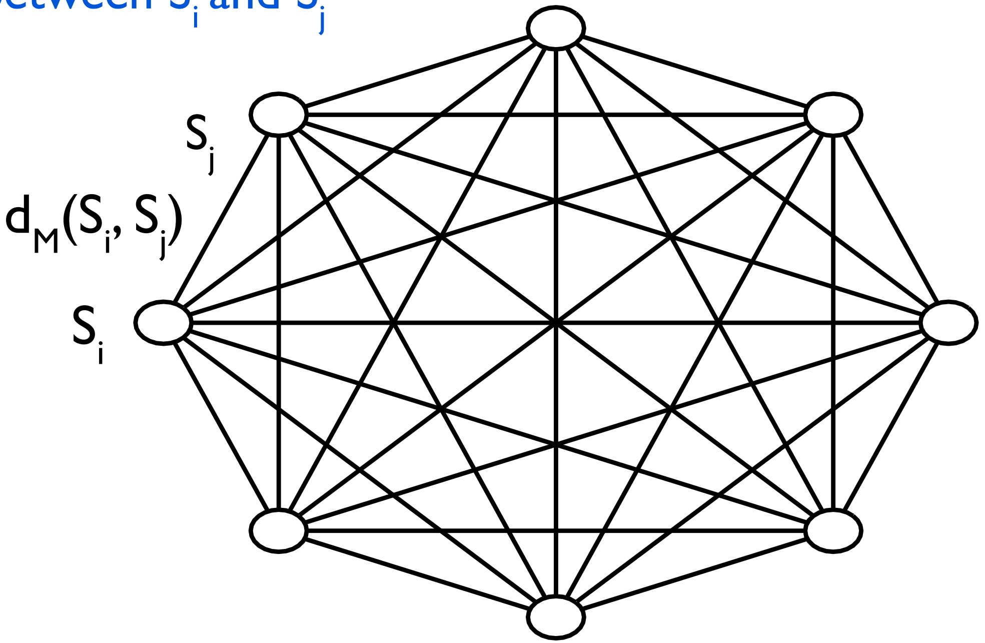
How do we define cost function for multiple sequence alignment?

A particular cost() function, the SP-Score (sum-of-pairs), is commonly used and allows us to design an approximation algorithm for the MSA problem.

$d_M(S_i, S_j)$ = the cost of the alignment between S_i and S_j **as implied by MSA** M.

SP-Score(M) = $\sum_{i<j} d_M(S_i, S_j)$

= sum of all the scores of the pairwise alignments implied by M.



Multiple Sequence Alignment

- A multiple sequence alignment (MSA) implies a pairwise alignment between every pair of sequences.
- This implied/induced alignment need not be optimal, however:

match = -1, a mismatch = 1, gap = 2 Sequences: AT, A, T, AT, AT

Optimal MSA:

```
  AT
  A-
  -T
  AT
  AT
+2  +2  =  4
```

Optimal Alignment between A and T:

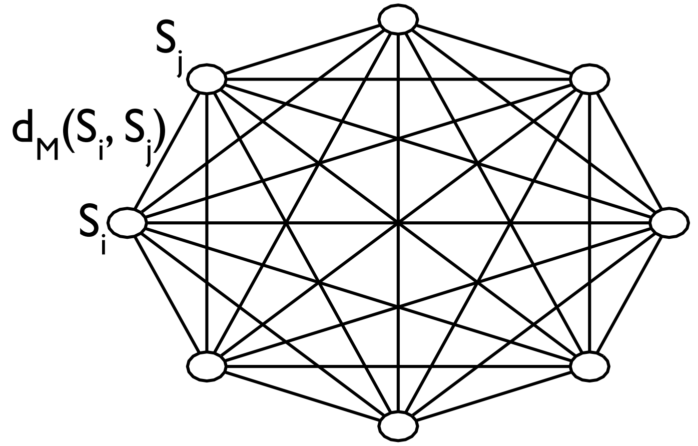
```
A
T
+1
```

Calculating the SP-score for column 1:

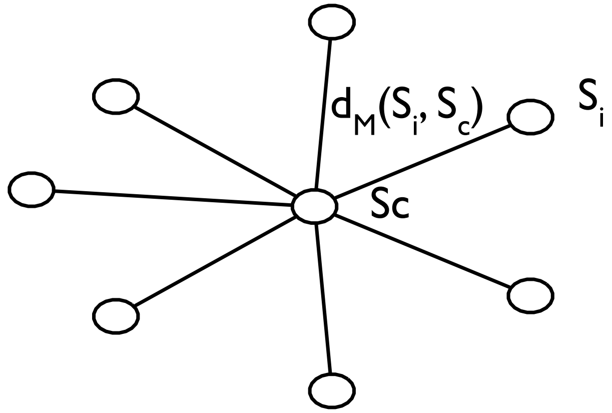
(A,A), (A,-), (A,A), (A,A), (A, -), (A,A), (A,A) (-,A), (-,A), (A,A)

-1 + 2 -1 -1 +2 -1 -1 +2 +2 -1 = +2

Star Alignment Approximation



SP-Score



Star-Score = $\sum_i d_M(S_i, S_c)$

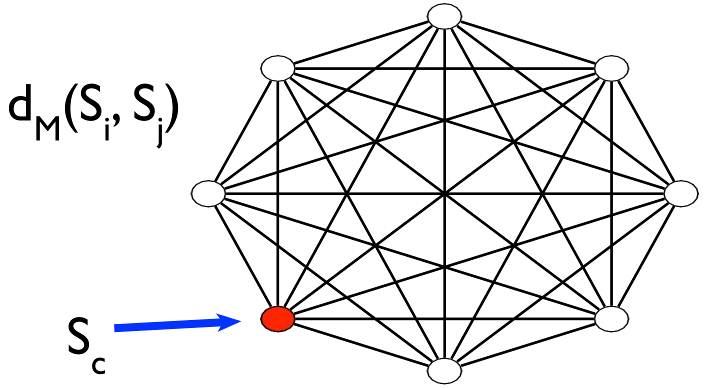
Star Alignment Algorithm

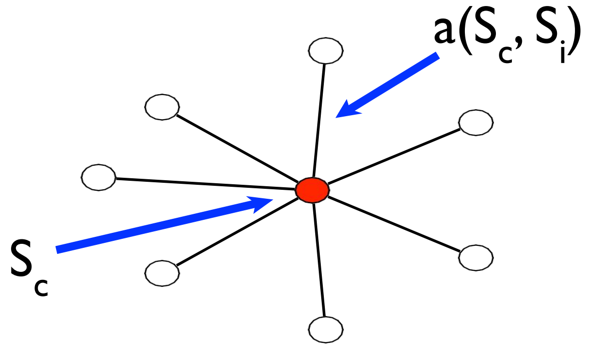
Input: sequences $S_1, S_2, ..., S_p$

- Build all $O(p^2)$ pairwise alignments.
- Let S_c = the sequence in $S_1, S_2, ..., S_p$ that is closest to the others. That is, choose S_c to minimize:

$$\sum_{i \neq c} a(S_c, S_i)$$

- *Progressively align* all other sequences to S_c.





Progressive Alignment

Idea: Build a multiple sequence alignment up from pairwise alignments.

Start with an alignment between S_c and some other sequence:

```
SC YFPHFDLSHGSAQVKAHGKKVGDALTLAVGHLDDLPGAL
S1 YFPHFDLSHG-AQVKG--KKVADALTNAVAHVDDMPNAL
```

Add 3rd sequence, say S2, and use the SC - S2 alignment as a guide, adding spaces into the MSA as needed.

SC - S2 alignment:

```
SC YFPHF-DLS-----HGSAQVKAHGKKVGDALTLAVGHL----DDLPGAL
S2 FFPKFKGLTTADQLKKSADVRWHAERII----NAVNDAVASMDDTEKMS
```

New {SC, S1, S2} alignment (*carry all gaps from pairwise alignments*):

```
SC             YFPHF-DLS-----HGSAQVKAHGKKVGDALTLAVGHL----DDLPGAL
S1             YFPHF-DLS-----HG-AQVKG--KKVADALTNAVAHV----DDMPNAL
S2 FFPKFKGLTTADQLKKSADVRWHAERII----NAVNDAVASMDDTEKMS
```

Continue with S3, S4, ...

Performance of “star” progressive alignment

Assume the cost function satisfies the triangle inequality:

$$\text{cost}(x,y) \le \text{cost}(x, z) + \text{cost}(z,y)$$

STAR = cost of result of star algorithm under SP-score

OPT = cost of optimal multiple sequence alignment (under SP-score)

Theorem. If cost satisfies the triangle inequality, then STAR $\le 2\times$OPT.

Example: if optimal alignment has cost 10, the star alignment will have cost ≤ 20.

2-approximation of STAR: Proof (1)

Theorem. If cost satisfies the triangle inequality, then STAR ≤ 2OPT.

$$\frac{\text{STAR}}{\text{OPT}} \leq 2$$

For some *B* we will prove the 2 statements:

$$\boxed{\begin{array}{l}\text{STAR} \leq 2B \\ \text{OPT} \geq B\end{array}}$$

This will imply:

$$\Longrightarrow \frac{\text{STAR}}{\text{OPT}} \leq \frac{2B}{B} = 2$$

2-approximation of STAR: Proof (2)

Theorem. If cost satisfies the triangle inequality, then STAR ≤ 2OPT.

$$2 \cdot \mathrm{STAR} = \sum_{ij} d_{\mathrm{STAR}}(S_i, S_j)$$ defn of SP-score

by triangle inequality
$$\leq \sum_{ij} (d_{\mathrm{STAR}}(S_i, S_c) + d_{\mathrm{STAR}}(S_c, S_j))$$

because STAR alignment is optimal for pairs involving Sc
$$= \sum_{ij} (\mathsf{a}(S_i, S_c) + \mathsf{a}(S_c, S_j))$$

distribute ∑
$$= \sum_{ij} \mathsf{a}(S_i, S_c) + \sum_{ij} \mathsf{a}(S_c, S_j)$$

$$\leq 2p \sum_{i} \mathsf{a}(S_i, S_c)$$

sums are the same and each term appears ≤ p (# of sequences) times.

2-approximation of STAR: Proof (3)

Theorem. If cost satisfies the triangle inequality, then STAR ≤ 2OPT.

$$2 \cdot \mathrm{OPT} = \sum_{ij} d_{\mathrm{OPT}}(S_i, S_j)$$

defn of SP-score

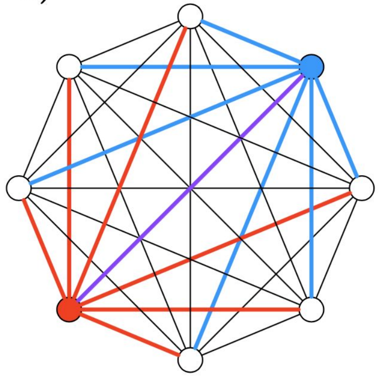
optimal pairwise alignment is ≤ pairwise alignment induced by any MSA

$$\geq \sum_{ij} \mathbf{a}(S_i, S_j)$$

$$\geq p \sum_{i} \mathbf{a}(S_i, S_c)$$

sum of cost of all pairwise alignments is = the sum of *p* different stars.

We chose S_c because it was the lowest-cost star.



2-approximation of STAR: End of Proof

For some B we will prove the 2 statements:

$$\boxed{\begin{array}{l}\text{STAR} \le 2B \\ \text{OPT} \ge B\end{array}}$$

This will imply:

$$\implies \frac{\text{STAR}}{\text{OPT}} \le \frac{2B}{B} = 2$$

$$\boxed{\begin{array}{rcl} 2 \cdot STAR & \le & 2p \sum_i \mathsf{a}(S_i, S_c) \\ 2 \cdot OPT & \ge & p \sum_i \mathsf{a}(S_i, S_c) \end{array}}$$

$$\implies \frac{\text{STAR}}{\text{OPT}} \le \frac{2p \sum_i \mathsf{a}(S_i, S_c)}{p \sum_i \mathsf{a}(S_i, S_c)} = 2$$

Consensus Sequence

For every column j, choose $c \in A_j$ that minimizes $\sum_i \mathrm{cost}(c, S_i[j])$

(typically this means the most common letter)

```
S1                 YFPHF-DLS-----HGSAQVKAHGKKVG-----DALTLAVAHLDDLPGAL
S2                 YFPHF-DLS-----HG-AQVKG—GKKVA-----DALTNAVAHVDDMPNAL
S3                 FFPKFKGLTTADQLKKSADVRWHAERII-----NAVNDAVASMDDTEKMS
S4                 LFSFLKGTSEVP--QNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATL
CO YFPHFKDLS-----HGSAQVKAHGKKVG-----DALTLAVAHVDDTPGAL
```

- Consensus is a summarization of the whole alignment.
- Consensus sequence is sometimes used as an estimate for the ancestral sequence.
- Sometimes the MSA problem is formulated as: find MSA M that minimizes: $\sum_i d_M(CO, S_i)$

Other progressive alignment strategy

First align the most similar sequences

How do we represent an alignment such that we can align a sequence to an alignment, or align two alignments?

Profiles

- Another way to summarize an MSA:

```
S1     ACG-TT-GA
S2     ATC-GTCGA
S3     ACGCGA-CC
S4     ACGCGT-TA
```

Column in the alignment

Character

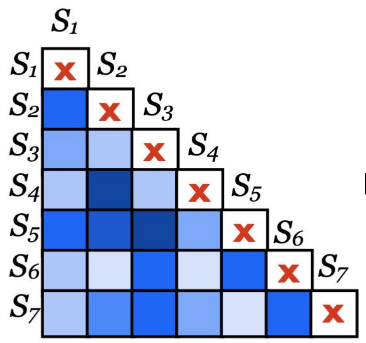
	1	2	3	4	5	6	7	8	9
A	1	0	0	0	0	0.25	0	0	0.75
C	0	0.75	0.25	0.5	0	0	0.25	0.25	0.25
G	0	0	0.75	0	0.75	0	0	0.5	0
T	0	0.25	0	0	0.25	0.75	0	0.25	0
-	0	0	0	0.5	0	0	0.75	0	0

Call this profile matrix R

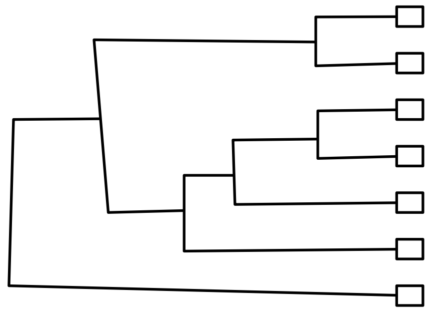
Fraction of time given column had the given character

CLUSTLW

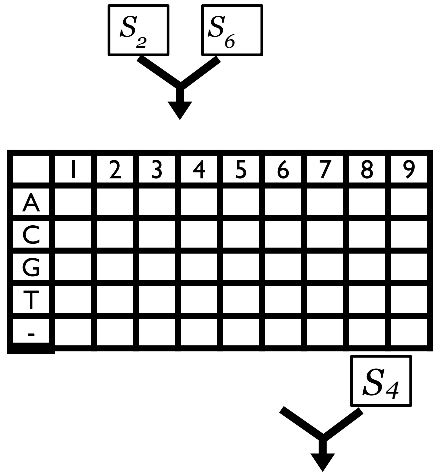
- CLUSTLW is a widely used, “classical” heuristic multiple aligner.
- Not the fastest, not the most accurate, but pretty good.
- Large # of heuristic tricks included in the software, but basic idea is straightforward:



(1): Build pairwise distance matrix



(2): Build guide tree



(3): Align sequences / **sets of sequences** from the most similar to least similar

Profile-based Alignment

gap in profile
introduced to better
fit sequence

R =

	1	2	3	4	5	6	7	8	9
A	1	0	0	0	0	0.25	0	0	0.75
C	0	0.75	0.25	0.5	0	0	0.25	0.25	0.25
G	0	0	0.75	0	0.75	0	0	0.5	0
T	0	0.25	0	0	0.25	0.75	0	0.25	0
-	0	0	0	0.5	0	0	0.75	0	0

A C G - A G A C G A

Score of matching character x with column j of the profile:

$$P(x, j) = \sum_{c \in \Sigma} \text{sim}(x, c) \times R[c, j]$$

sim(x,c) = how similar character x is to character c.

$$A[i,j] = \max \begin{cases} A[i-1, j-1] + P(x_i, j) & \text{align } x_i \text{ to column } j \\ A[i-1, j] + \text{gap} & \text{introduce gap into profile} \\ A[i, j-1] + P(\text{“\_“}, j) & \text{introduce gap into } x \end{cases}$$

MSA Recap

- Multiple sequence alignments (MSAs) are a fundamental tool. They help reveal subtle patterns, compute consistent distances between sequences, etc.
- Quality of MSAs often measured using the SP-score: sum of the scores of the pairwise alignments implied by the MSA.
- Same DP idea as pairwise alignment leads to exponentially slow algorithm for MSA for general *p*.
- 2-approximation obtainable via star alignments.
- MSAs often used to create profiles summarizing a family of sequences.

Further reading

- Durbin, R., Eddy, S. R., Krogh, A. & Mitchison, G. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. (Cambridge University Press, 1998), Chapter 6.
- Feng, D.-F. & Doolittle, R. F. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *J. Mol. Evol.* **25**, 351–360 (1987)
- Higgins, D. G., Thompson, J. D. & Gibson, T. J. [22] Using CLUSTAL for multiple sequence alignments. in *Methods in Enzymology* vol. 266 383–402 (Academic Press, 1996).
- Thompson, J. D., Linard, B., Lecompte, O. & Poch, O. A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. *PLoS One* **6**, e18093 (2011)
- Notredame, C. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics* **3**, 131–144 (2002)