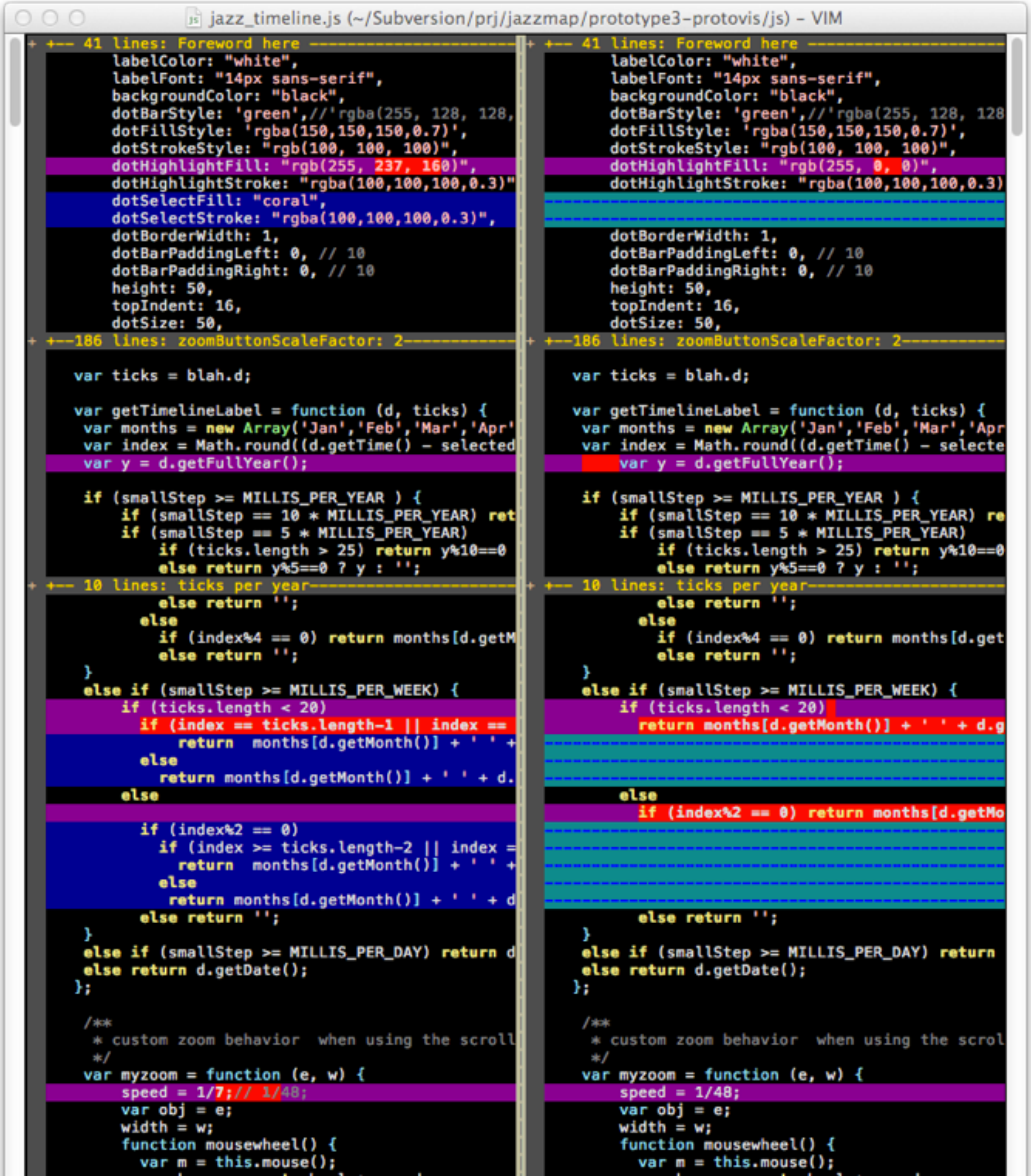
CSE 549: Computational Biology

Multiple Sequence Alignment

Stony Brook University

slides *not* marked with * by Carl Kingsford



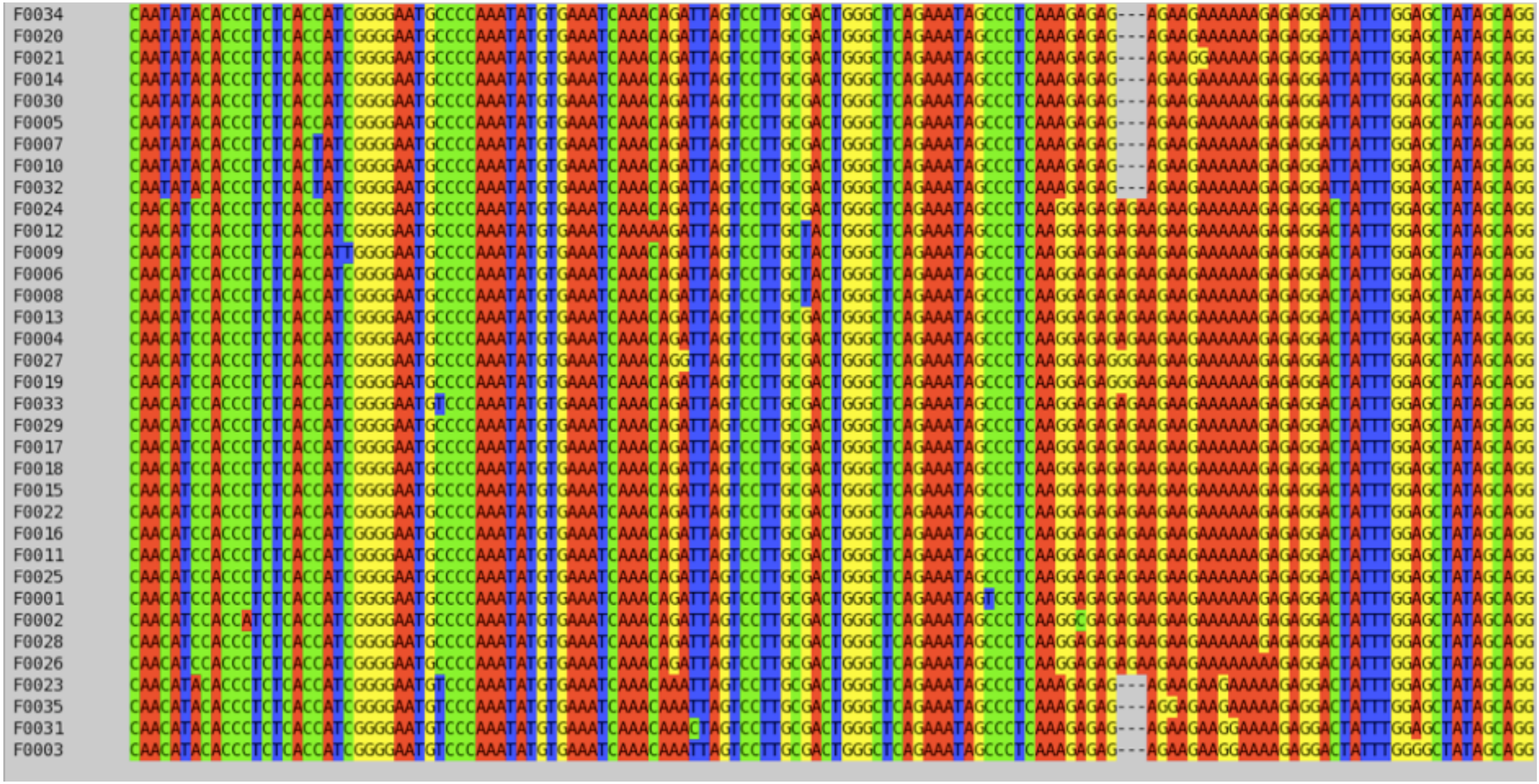
jazz_timeline.js (~/Subversion/prj/jazzmap/prototype3-protovis/js) - VIM

```
+ +-- 41 lines: Foreword here ----------------------------
        labelColor: "white",
        labelFont: "14px sans-serif",
        backgroundColor: "black",
        dotBarStyle: 'green',//'rgba(255, 128, 128,
        dotFillStyle: 'rgba(150,150,150,0.7)',
        dotStrokeStyle: "rgb(100, 100, 100)",
        dotHighlightFill: "rgb(255, 237, 160)",
        dotHighlightStroke: "rgba(100,100,100,0.3)"
        dotSelectFill: "coral",
        dotSelectStroke: "rgba(100,100,100,0.3)",
        dotBorderWidth: 1,
        dotBarPaddingLeft: 0, // 10
        dotBarPaddingRight: 0, // 10
        height: 50,
        topIndent: 16,
        dotSize: 50,
+ +--186 lines: zoomButtonScaleFactor: 2-------------

   var ticks = blah.d;

   var getTimelineLabel = function (d, ticks) {
    var months = new Array('Jan','Feb','Mar','Apr'
    var index = Math.round((d.getTime() - selected
    var y = d.getFullYear();

    if (smallStep >= MILLIS_PER_YEAR ) {
        if (smallStep == 10 * MILLIS_PER_YEAR) ret
        if (smallStep == 5 * MILLIS_PER_YEAR)
            if (ticks.length > 25) return y%10==0
            else return y%5==0 ? y : '';
+ +-- 10 lines: ticks per year---------------------------
            else return '';
          else
            if (index%4 == 0) return months[d.getM
            else return '';
    }
    else if (smallStep >= MILLIS_PER_WEEK) {
        if (ticks.length < 20)
          if (index == ticks.length-1 || index ==
              return  months[d.getMonth()] + ' ' +
          else
            return months[d.getMonth()] + ' ' + d.
        else

          if (index%2 == 0)
            if (index >= ticks.length-2 || index =
              return  months[d.getMonth()] + ' ' +
            else
             return months[d.getMonth()] + ' ' + d
          else return '';
    }
    else if (smallStep >= MILLIS_PER_DAY) return d
    else return d.getDate();
   };

    /**
     * custom zoom behavior  when using the scroll
     */
    var myzoom = function (e, w) {
        speed = 1/7;// 1/48;
        var obj = e;
        width = w;
        function mousewheel() {
          var m = this.mouse();
```

```
+ +-- 41 lines: Foreword here ----------------------------
        labelColor: "white",
        labelFont: "14px sans-serif",
        backgroundColor: "black",
        dotBarStyle: 'green',//'rgba(255, 128, 128
        dotFillStyle: 'rgba(150,150,150,0.7)',
        dotStrokeStyle: "rgb(100, 100, 100)",
        dotHighlightFill: "rgb(255, 0, 0)",
        dotHighlightStroke: "rgba(100,100,100,0.3)
---------------------------------------------------
---------------------------------------------------
        dotBorderWidth: 1,
        dotBarPaddingLeft: 0, // 10
        dotBarPaddingRight: 0, // 10
        height: 50,
        topIndent: 16,
        dotSize: 50,
+ +--186 lines: zoomButtonScaleFactor: 2-------------

   var ticks = blah.d;

   var getTimelineLabel = function (d, ticks) {
    var months = new Array('Jan','Feb','Mar','Apr
    var index = Math.round((d.getTime() - selecte
        var y = d.getFullYear();

    if (smallStep >= MILLIS_PER_YEAR ) {
        if (smallStep == 10 * MILLIS_PER_YEAR) re
        if (smallStep == 5 * MILLIS_PER_YEAR)
            if (ticks.length > 25) return y%10==0
            else return y%5==0 ? y : '';
+ +-- 10 lines: ticks per year---------------------------
            else return '';
          else
            if (index%4 == 0) return months[d.get
            else return '';
    }
    else if (smallStep >= MILLIS_PER_WEEK) {
        if (ticks.length < 20)
          return months[d.getMonth()] + ' ' + d.g
---------------------------------------------------
---------------------------------------------------
---------------------------------------------------
        else
          if (index%2 == 0) return months[d.getMo
---------------------------------------------------
---------------------------------------------------
---------------------------------------------------
---------------------------------------------------
---------------------------------------------------
          else return '';
    }
    else if (smallStep >= MILLIS_PER_DAY) return
    else return d.getDate();
   };

    /**
     * custom zoom behavior  when using the scrol
     */
    var myzoom = function (e, w) {
        speed = 1/48;
        var obj = e;
        width = w;
        function mousewheel() {
          var m = this.mouse();
```

3rd source file here

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.

Patterns that seem **insignificant** in a pair may become **significant** if shared by many sequences.

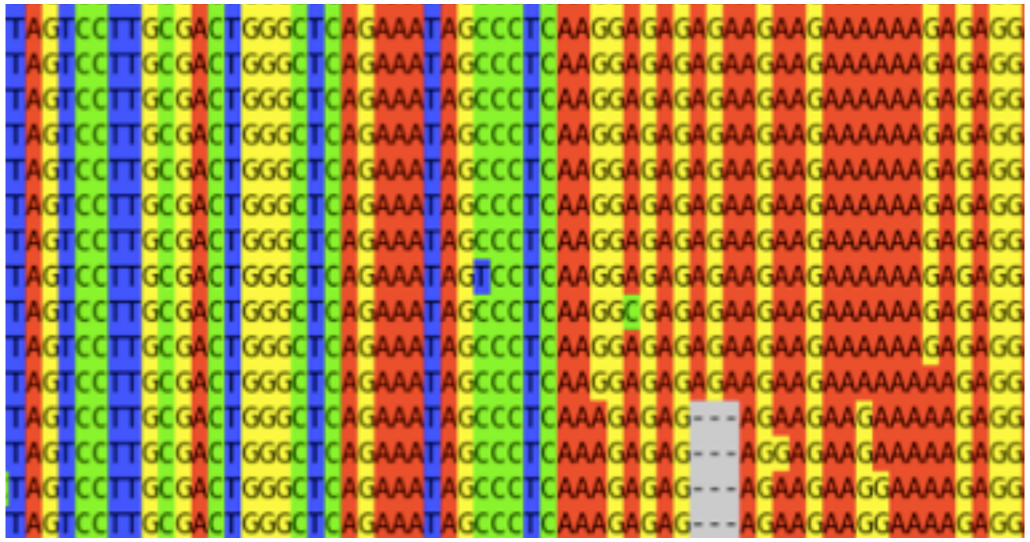
Generalizing Alignment to > 2 Strings

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

Let $\mathit{cost}(x_1,x_2,\ldots x_p)$ 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** ∑ cost of the columns:

$$\underbrace{\mathit{cost}(x_1,x_2,\ldots x_p) = \mathrm{cost}(\quad \mathrm{A\ A\ A\ A\ A\ A\ A\ A\ A\ A\ -\ -\ -\ -} \quad)}$$

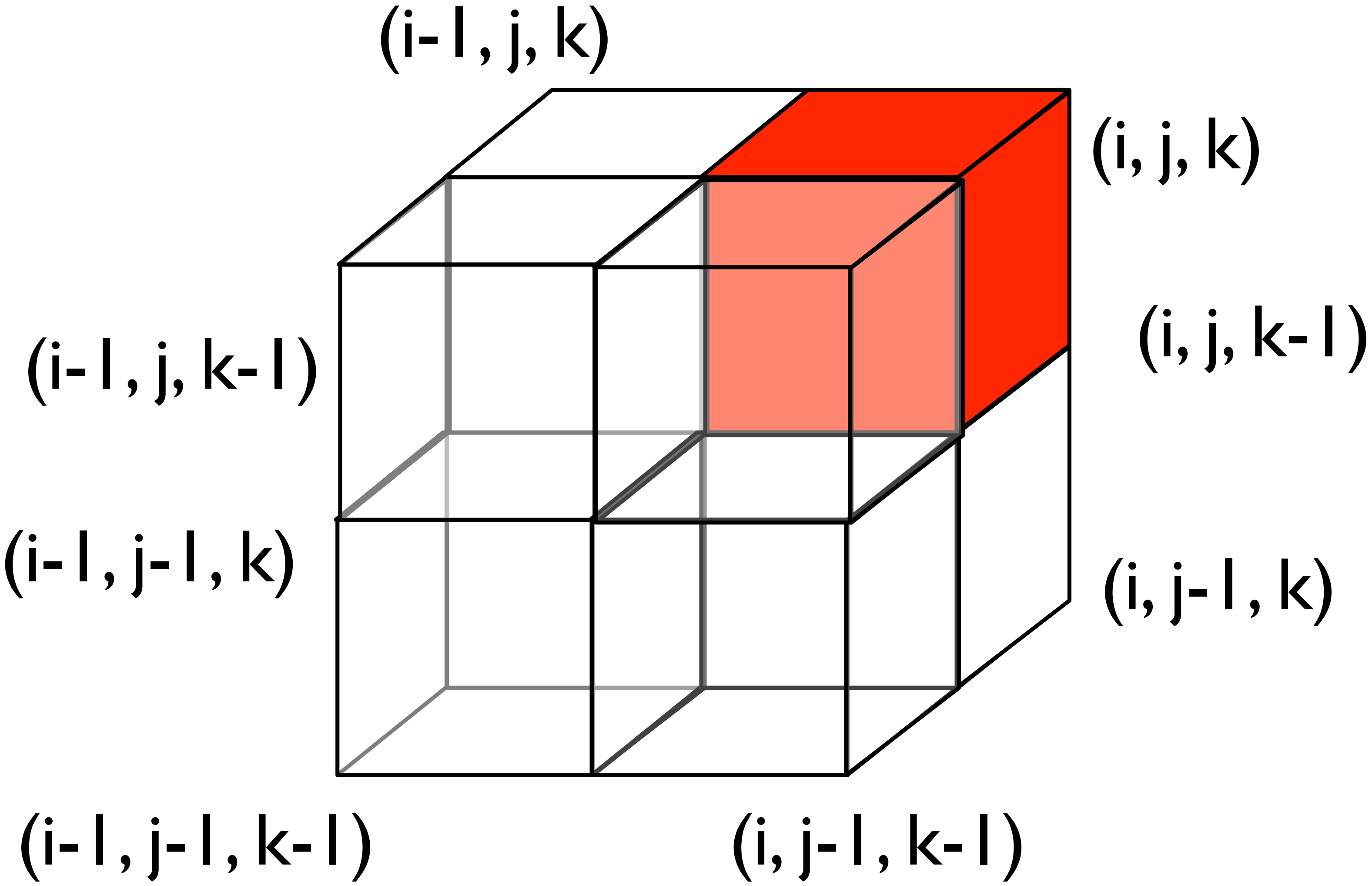


```
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGTCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGCGAGAGAAGAAGAAAAAAGAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAGAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAGGAGAGAGAAGAAGAAAAAAAAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAGAAAAAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGGAGAAGAAAAAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAGGAAAAGAGG
TAGTCCTTGCGACTGGGCTCAGAAATAGCCCTCAAAGAGAG---AGAAGAAGGAAAAGAGG
```

Slow Dynamic Programming

Suppose you had just 3 sequences.

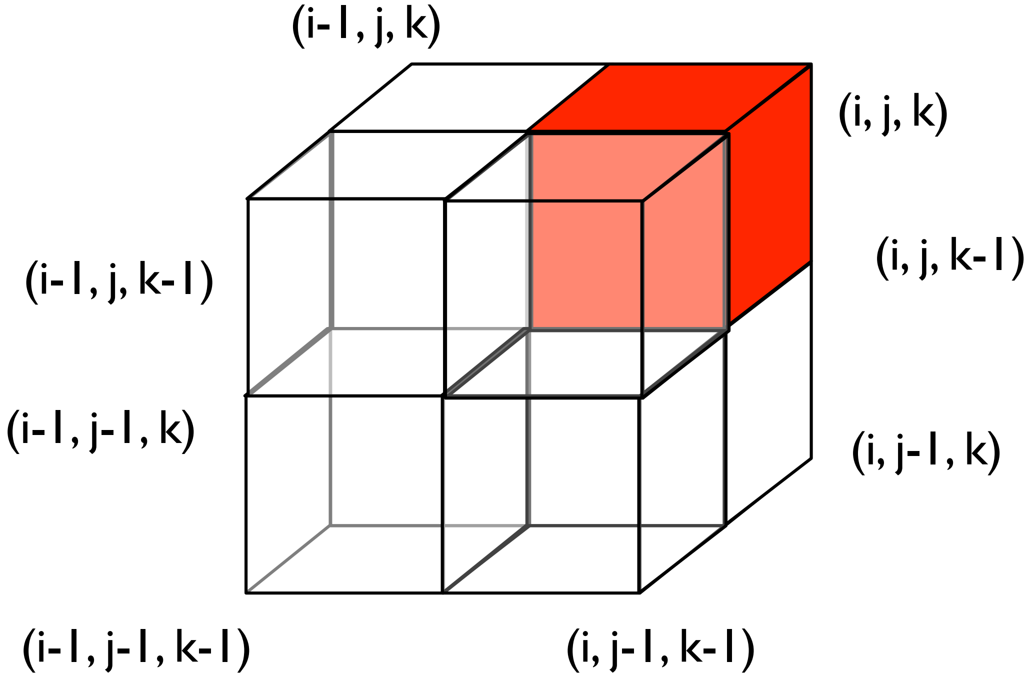
Apply the same DP idea as sequence alignment for 2 sequences, but now with a 3-dimensional matrix



DP Recurrence for 3 sequences

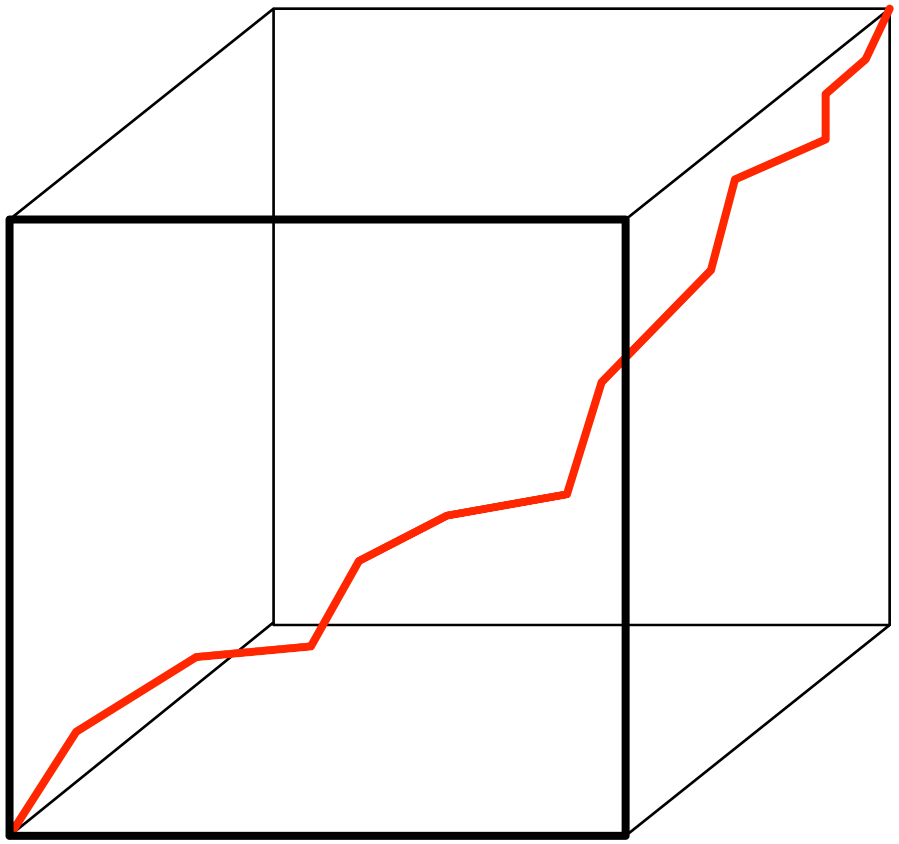
$$A[i,j,k] = \min \begin{cases} \text{cost}(x_i, y_j, z_k) + A[i-1, j-1, k-1] \\ \text{cost}(x_i, -, -) + A[i-1, j, k] \\ \text{cost}(x_i, y_j, -) + A[i-1, j-1, k] \\ \text{cost}(-, y_j, z_k) + A[i, j-1, k-1] \\ \text{cost}(-, y_j, -) + A[i, j-1, k] \\ \text{cost}(x_i, -, z_k) + A[i-1, j, k-1] \\ \text{cost}(-,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

Every possible pattern for the gaps.



Running time

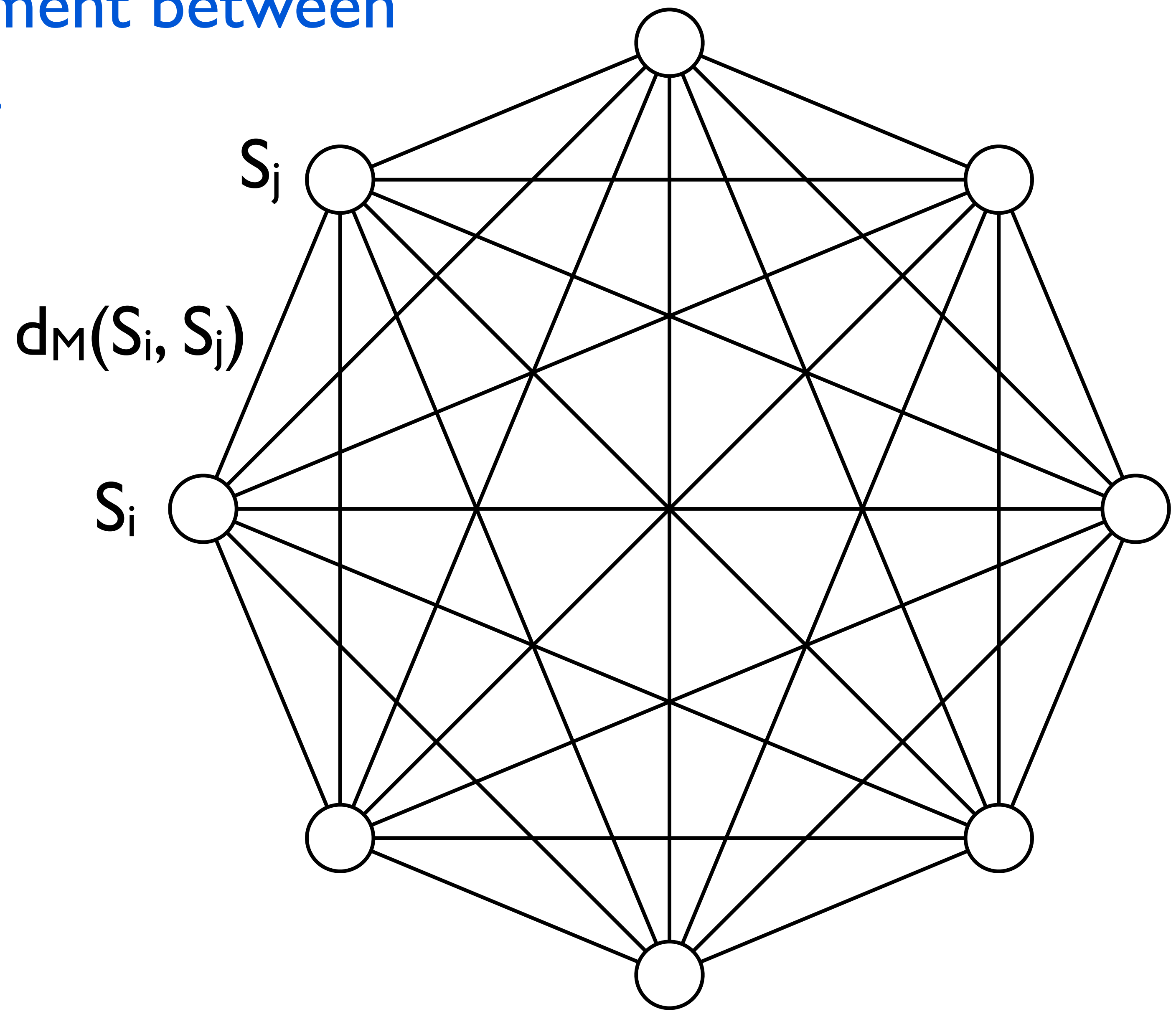
- n^3 subproblems, each takes 2^3 time ⇒ $O(n^3)$ time.
- For p sequences: n^p subproblems, each takes 2^p time for the min and p^2 to compute cost(., ., …) ⇒ $O(p^2 n^p 2^p)$
- Even $O(n^3)$ is often too slow for the length of sequences encountered in practice.
- One solution: approximation algorithm.



SP-Cost

A particular cost() function, the SP-Cost, 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-Cost(M) = $\sum_{i<j} d_M(S_i, S_j)$

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

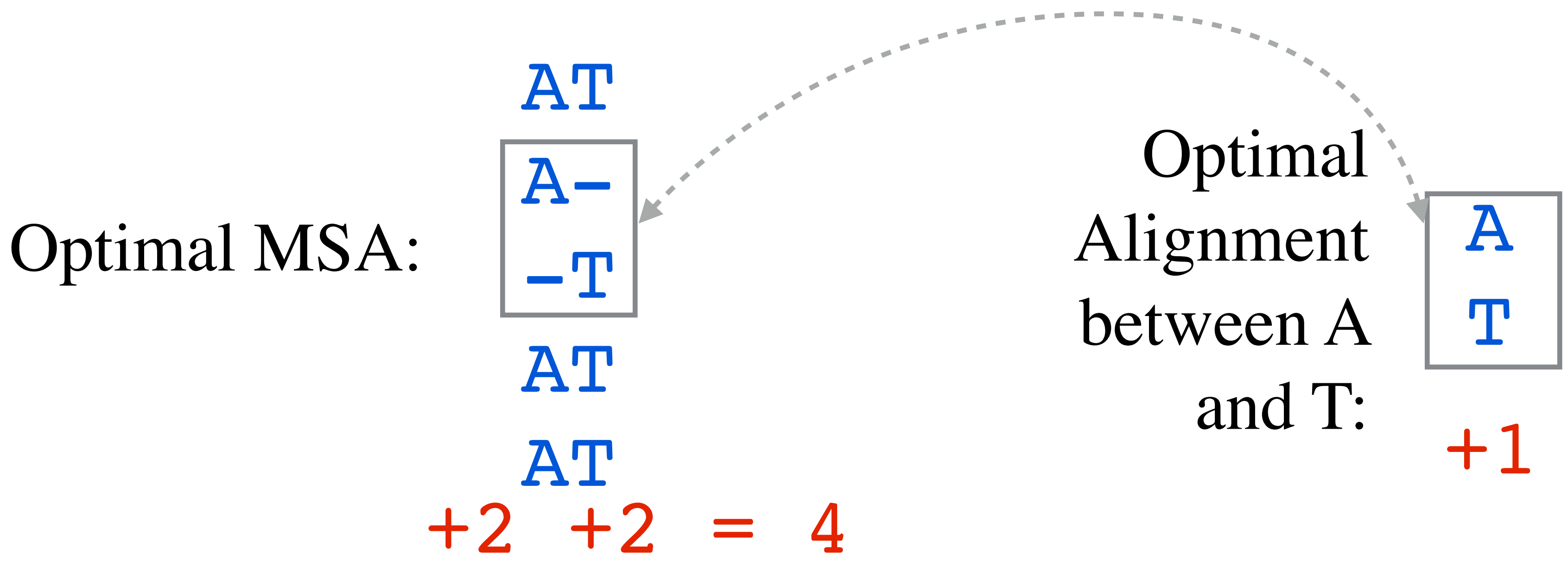
MSA

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

match = -1, a mismatch = 1, gap = 2

Sequences: AT, A, T, AT, AT

Note: Here we consider the **minimization** problem



Optimal MSA:

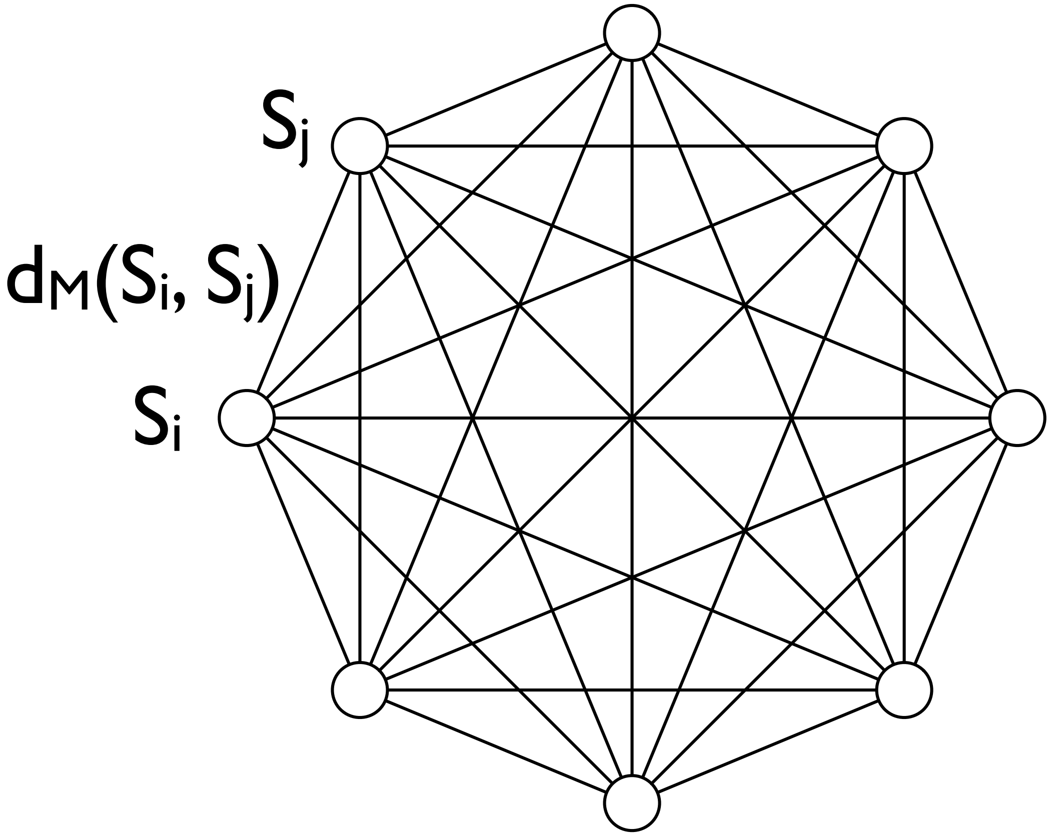
```
AT
A-
-T
AT
AT
+2 +2 = 4
```

Optimal Alignment between A and T:

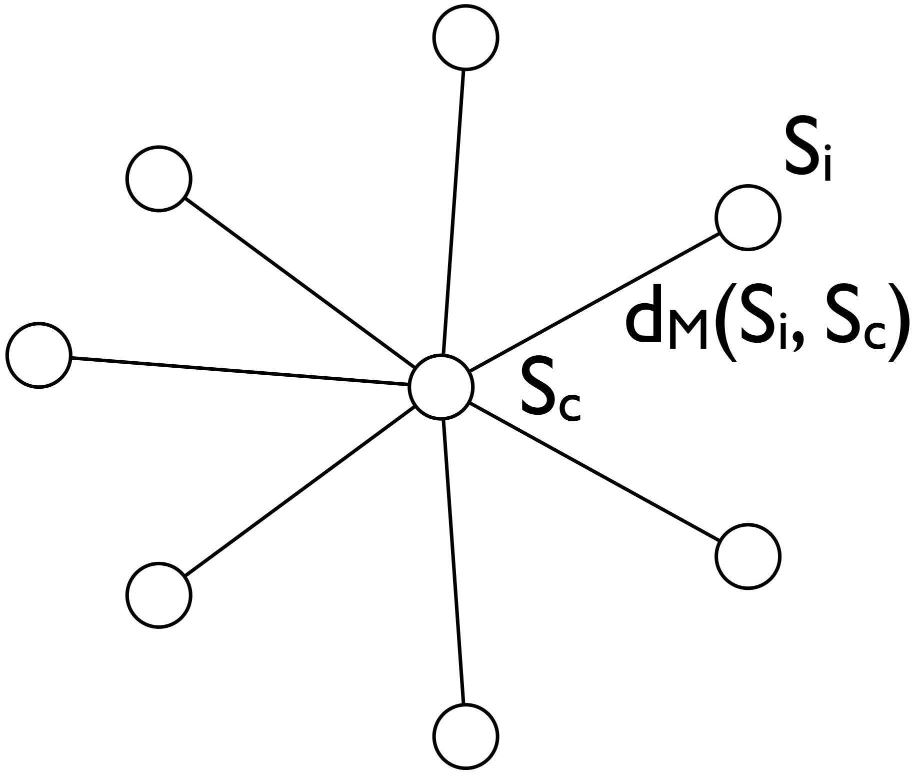
```
A
T
+1
```

(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-Cost



Star-Cost =

$\sum_i d_M(S_i, S_c)$

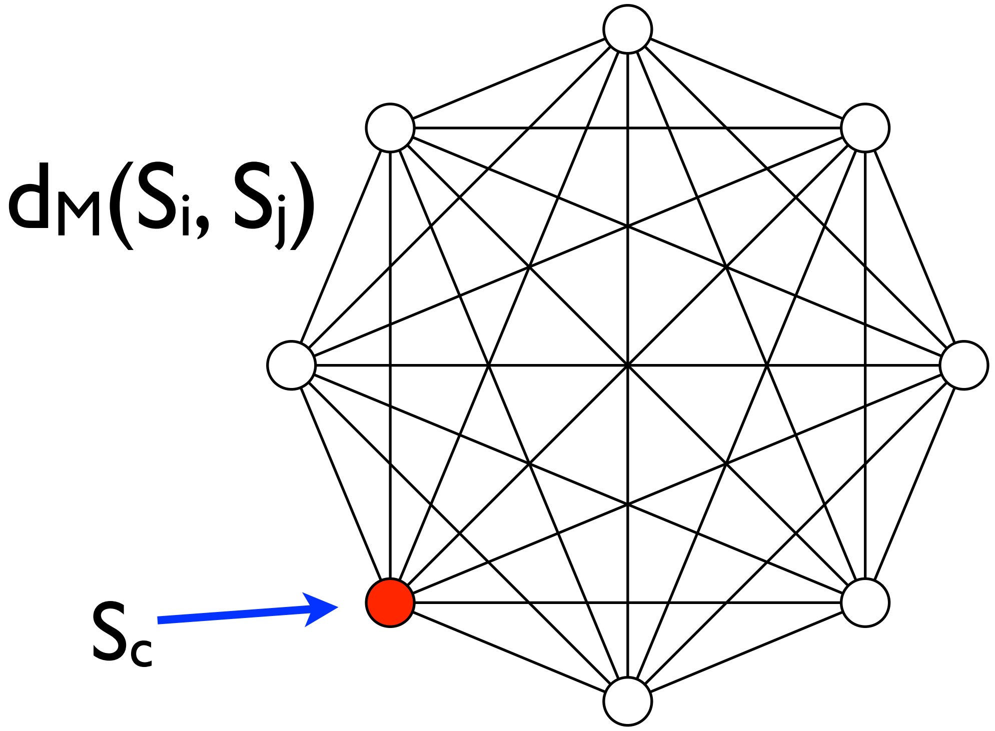
STAR Alignment Algorithm

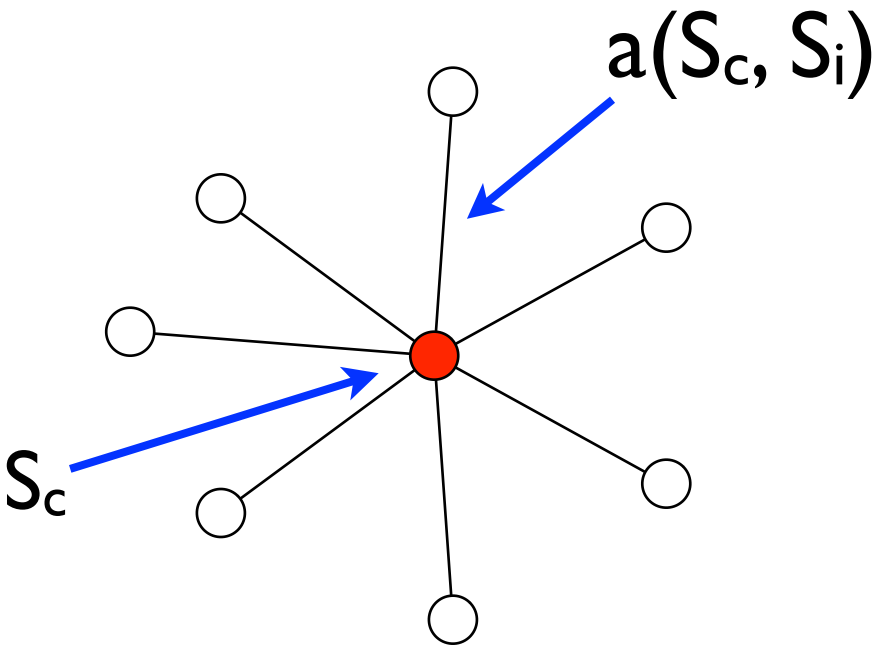
Input: sequences $S_1, S_2, ..., S_p$

- 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

- Build a multiple sequence alignment up from pairwise alignments.

Start with an alignment between S_c and some other sequence:

```
SC YFPHF-DLS-----HGSAQVKAHGKKVGDALTLAVGHL----DDLPGAL
S1 FFPKFKGLTTADQLKKSADVRWHAERII----NAVNDAVASMDDTEKMS
```

Add 3rd sequence, say S2, and use the SC ↔ S1 alignment as a guide, adding spaces into the MSA as needed.

SC ↔ S2 alignment:

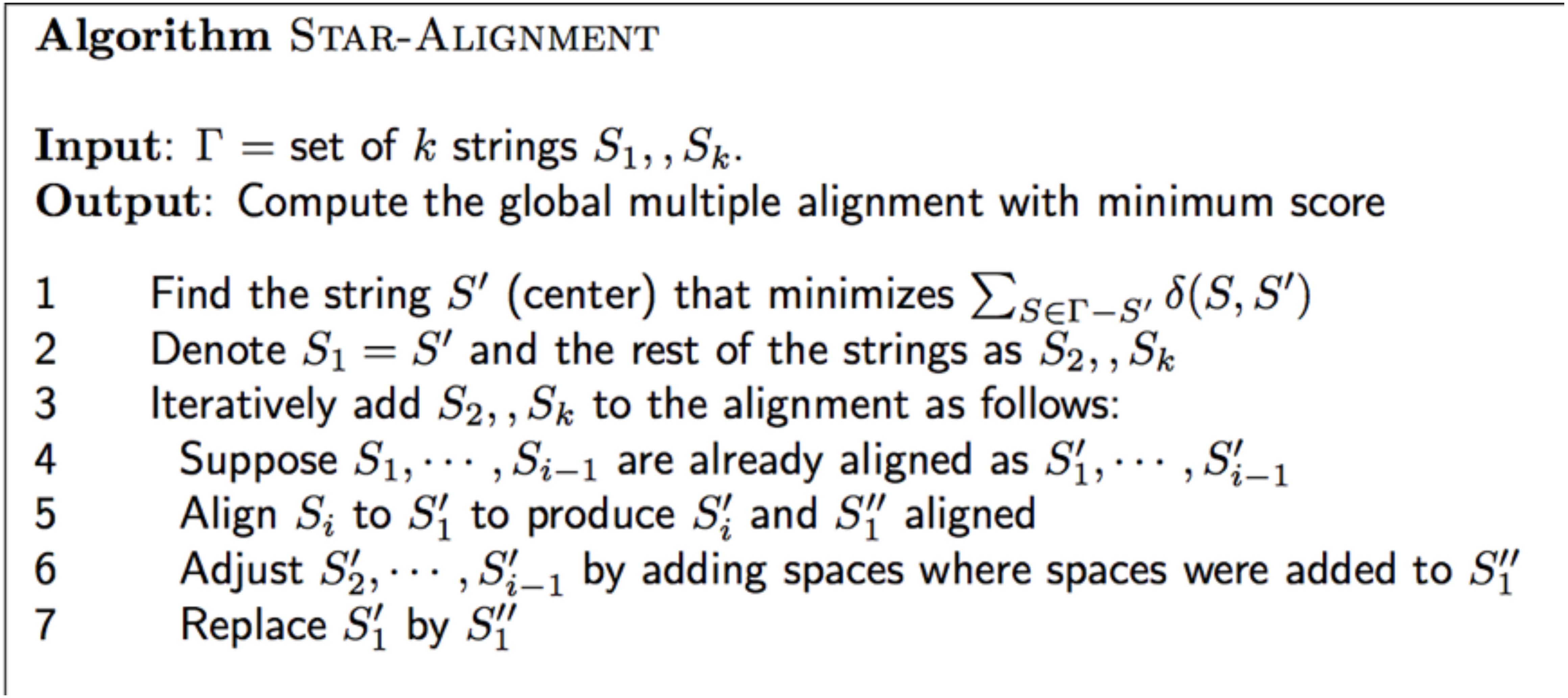
```
SC YFPHFDLSHGSAQVKAHGKKVGDALTLAVGHLDDLPGAL
S2 YFPHFDLSHG-AQVKG--KKVADALTNAVAHVDDMPNAL
```

New {SC, S1, S2} alignment (red gaps added in S2):

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

Continue with S3, S4, ...

The STAR algorithm



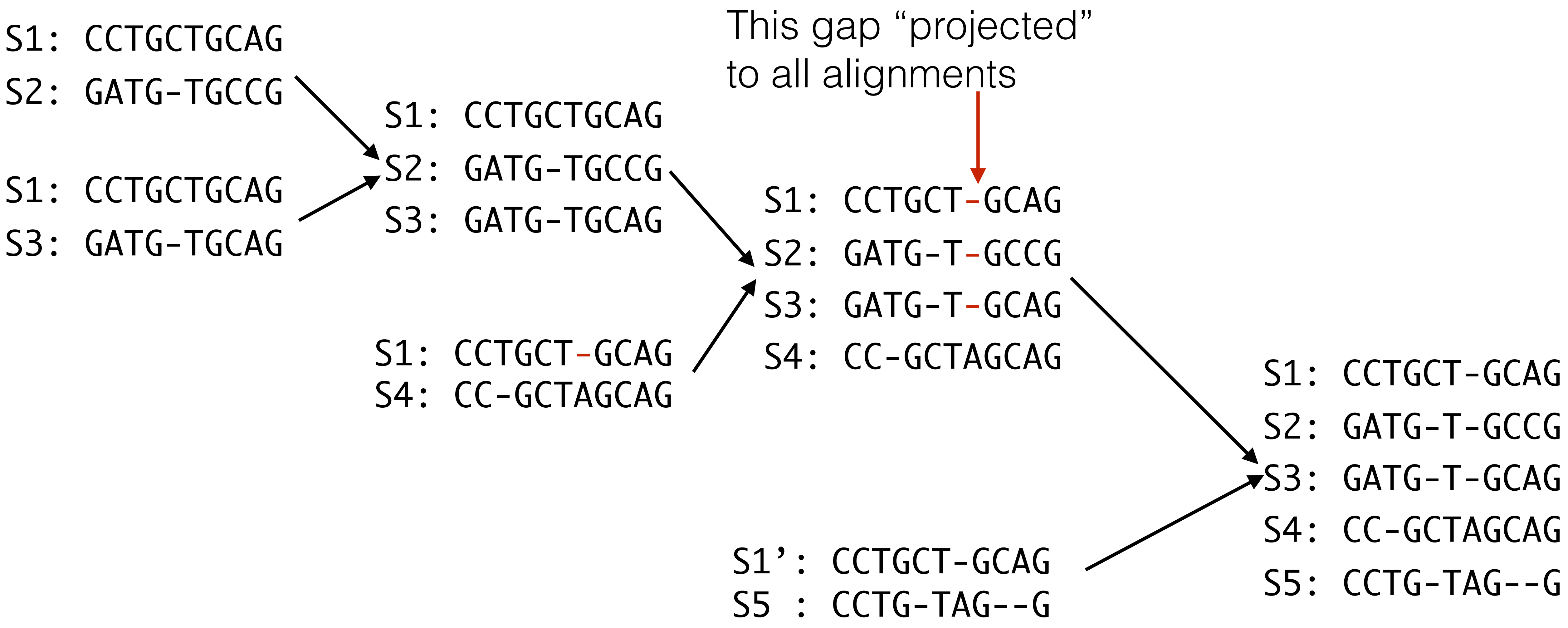
Algorithm Star-Alignment

Input: Γ = set of k strings $S_1, , S_k$.
Output: Compute the global multiple alignment with minimum score

```
Find the string $S'$ (center) that minimizes $\sum_{S \in \Gamma - S'} \delta(S, S')$
Denote $S_1 = S'$ and the rest of the strings as $S_2, , S_k$
Iteratively add $S_2, , S_k$ to the alignment as follows:
  Suppose $S_1, \cdots, S_{i-1}$ are already aligned as $S'_1, \cdots, S'_{i-1}$
  Align $S_i$ to $S'_1$ to produce $S'_i$ and $S''_1$ aligned
  Adjust $S'_2, \cdots, S'_{i-1}$ by adding spaces where spaces were added to $S''_1$
  Replace $S'_1$ by $S''_1$
```

* pseudocode from: http://compbio.cs.uic.edu/~tanya/teaching/CompBio/notes/06-MultSeqAlign-02-05.pdf

Merging pairwise alignments



```
S1: CCTGCTGCAG
S2: GATG-TGCCG
```

```
S1: CCTGCTGCAG
S3: GATG-TGCAG
```

```
S1: CCTGCTGCAG
S2: GATG-TGCCG
S3: GATG-TGCAG
```

```
S1: CCTGCT-GCAG
S4: CC-GCTAGCAG
```

This gap “projected” to all alignments

```
S1: CCTGCT-GCAG
S2: GATG-T-GCCG
S3: GATG-T-GCAG
S4: CC-GCTAGCAG
```

```
S1’: CCTGCT-GCAG
S5 : CCTG-TAG--G
```

```
S1: CCTGCT-GCAG
S2: GATG-T-GCCG
S3: GATG-T-GCAG
S4: CC-GCTAGCAG
S5: CCTG-TAG--G
```

The “once a gap, always a gap” rule is applied — the MSA will tend to get *longer* with more sequences

* example from http://www.comp.nus.edu.sg/~ksung/algo_in_bioinfo/slides/Ch6_MSA.pdf

Performance

Assume the cost function satisfies the triangle inequality:

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

Example: $\underbrace{\text{cost}(A, C)}_{\substack{\text{cost of a}\\ \text{mutation from}\\ A\to C}} \leq \underbrace{\text{cost}(A, T) + \text{cost}(T,C)}_{\substack{\text{cost of a mutation}\\ \text{from } A\to T \text{ and}\\ \text{then from } T\to C}}$

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

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

Theorem. If cost satisfies the triangle inequality, then STAR $\leq$ 2×OPT.

Example: if optimal alignment has cost 10, the star alignment will have cost $\leq$ 20.

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:

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

Proof (2)

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

d_M = distance under STAR

$$
\begin{aligned}
&= \sum_{1\le i<j\le k} d_{\mathcal{M}}(i,j) \\
&= \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k} d_{\mathcal{M}}(i,j) \\
&\le \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\left[D(S_c,S_i)+D(S_c,S_j)\right] \\
&= \frac{k}{2}\sum_{i=1}^{k} D(S_c,S_i)+\frac{k}{2}\sum_{j=1}^{k} D(S_c,S_j) \\
&= k\sum_{j} D(S_c,S_j)
\end{aligned}
$$

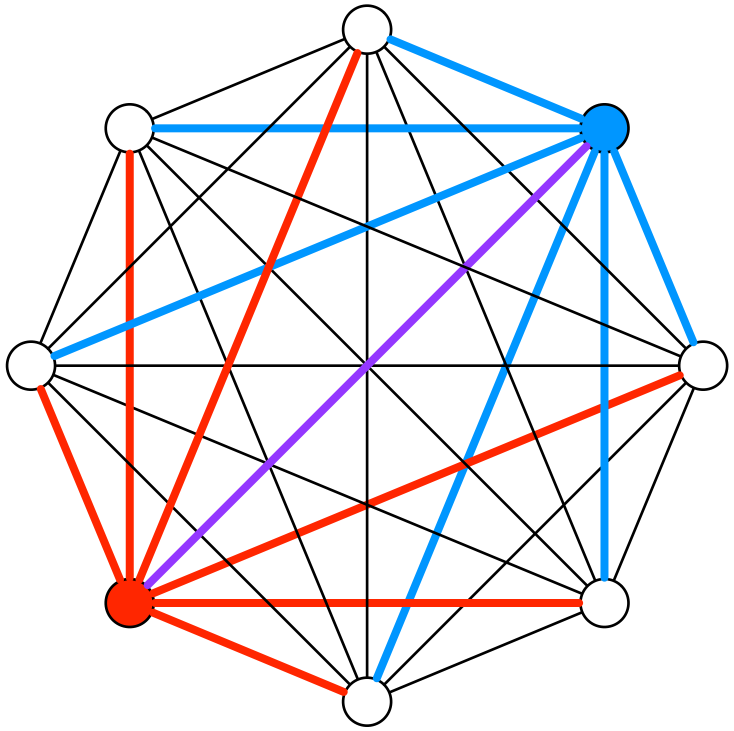
Proof (3)

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

d_{M^*} = distance under OPT

$$
\begin{aligned}
&= \sum_{1 \le i < j \le k} d_{\mathcal{M}^*}(i,j) \\
&\ge \sum_{1 \le i < j \le k} D(S_i, S_j) \\
&= \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} D(S_i, S_j) \\
&\ge \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} D(S_c, S_j) \\
&= \frac{k}{2} \sum_{j=1}^{k} D(S_c, S_j)
\end{aligned}
$$

optimal pairwise alignment is ≤ pairwise alignment induced by any MSA



End of Proof

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

$$\boxed{\begin{aligned}\text{STAR} &\leq 2B \\ \text{OPT} &\geq B\end{aligned}}$$

This will imply:

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

$$\boxed{\begin{aligned} 2 \cdot STAR &\leq 2k \sum_i \mathsf{D}(S_i, S_c) \\ 2 \cdot OPT &\geq k \sum_i \mathsf{D}(S_i, S_c) \end{aligned}}$$

$$\implies \frac{\text{STAR}}{\text{OPT}} \leq \frac{2k \sum_i \mathsf{D}(S_i, S_c)}{k \sum_i \mathsf{D}(S_i, S_c)} = 2$$

Consensus Sequence

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

(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: $\sum_i d_M(CS, S_i)$

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

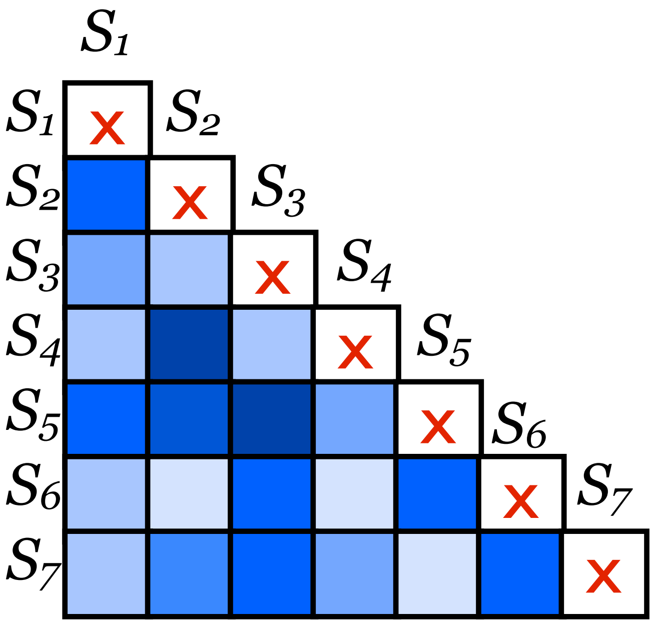
	1	2	3	4	5	6	7	8	9
A	1	0	0	0	0	0.25	0	0	0.75
C	0	0.75	0.25	0.5	0	0	0.25	0.25	0.25
G	0	0	0.75	0	0.75	0	0	0.5	0
T	0	0.25	0	0	0.25	0.75	0	0.25	0
-	0	0	0	0.5	0	0	0.75	0	0

Call this profile matrix R

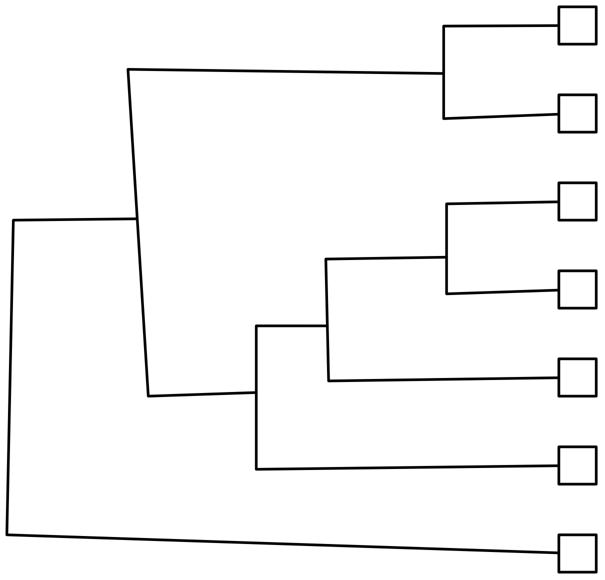
Fraction of time given column had the given character

CLUSTLW

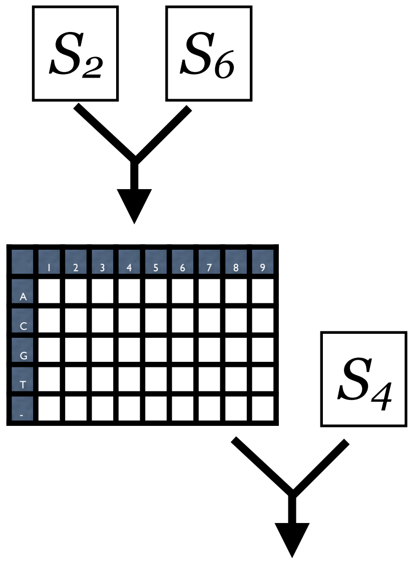
- CLUSTLW is a widely used, "classical" heuristic multiple aligner.
- Not the fastest, not the most accurate, but pretty good.
- Large # of heuristic tricks included in the software, but basic idea is straightforward:



Step (1): Build pairwise distance matrix



Step (2): Build guide tree



Step (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 C - 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 (e.g. score(x,c)).

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

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.