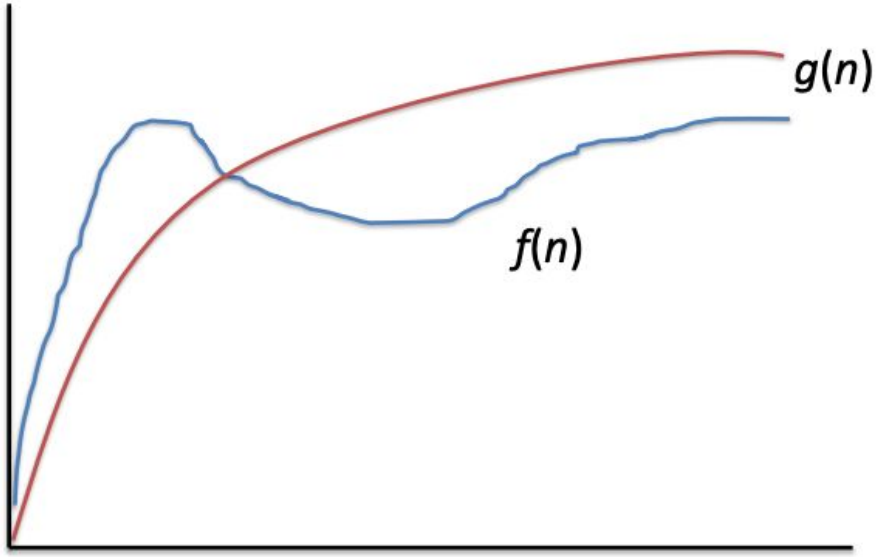
Midterm Review

CS466 Fall 2020

Wesley Wei Qian
10/02/2020

Definition of Running Time

- The **running time** of an algorithm A for problem Π is the maximum number of steps that A will take on any instance of size $n = |X|$
- **Asymptotic running time** ignores constant factors using Big O notation



$f(n)$ is $O(g(n))$ provided there exists $c > 0$ and $n_0 \geq 0$ such that $f(n) \leq c\ g(n)$ for all $n \geq n_0$

Guideline

- $O(n^a) \subset O(n^b)$ for any positive constants $a < b$

- For any constants $a, b > 0$ and $c > 1$,

  $O(a) \subset O(\log n) \subset O(n^b) \subset O(c^n)$

Big Oh	Name
$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(n)$	Linear
$O(n^2)$	Quadratic
$O(n^c) = O(\text{poly}(n))$	Polynomial
$O(2^{\text{poly}(n)})$	Exponential

- We can multiply to learn about other functions. For any constants $a, b > 0$ and $c > 1$,

  $O(an) = O(n) \subset O(n \log n) \subset O(n\, n^b) = O(n^{b+1}) \subset O(nc^n)$

- Base of the logarithm is a constant and can be ignored. For any constants $a, b > 1$,

  $$O(\log_a n) = O(\log_b n / \log_b a) = O(1/(\log_b a) \log_b n) = O(\log_b n)$$

Global Alignment - map entire v and w

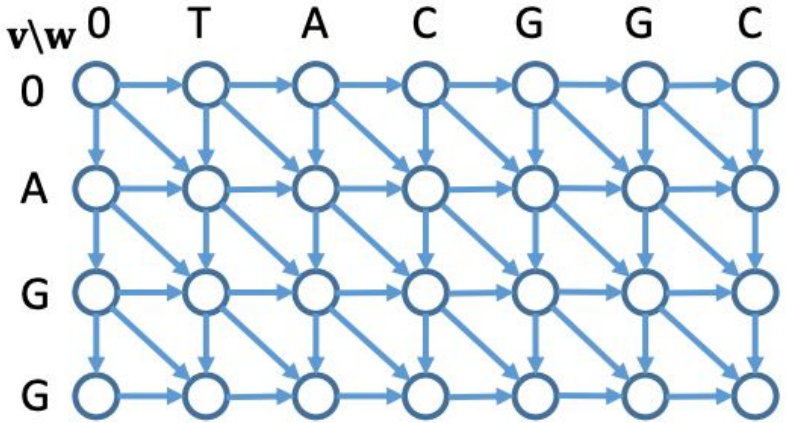
Global Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find alignment with maximum score.

- An alignment is a source-to-sink path in the edit graph
- An alignment $\mathbf{A} = [a_{i,j}]$ is a $2 \times k$ matrix s.t. (i) $k = \{\max(m,n), \ldots, m+n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0, \quad \text{deletion} \\ s[i,j-1] + \delta(-, w_j), & \text{if } j > 0, \quad \text{insertion} \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \quad \text{match/ mismatch} \end{cases}$$

Fitting Alignment - map entire v to part of w

Fitting Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find an alignment of $\mathbf{v}$ and a substring of $\mathbf{w}$ with maximum global alignment score s^* among *all* global alignments of $\mathbf{v}$ and *all* substrings of $\mathbf{w}$



$$s[i,j] = \max \begin{cases} 0, \text{ Start anywhere on first row} & \text{if } i = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0, \\ s[i,j-1] + \delta(-, w_j), & \text{if } i > 0 \text{ and } j > 0, \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$

$$s^* = \max\{s[m,0], \ldots, s[m,n]\} \quad \text{End anywhere on last row}$$

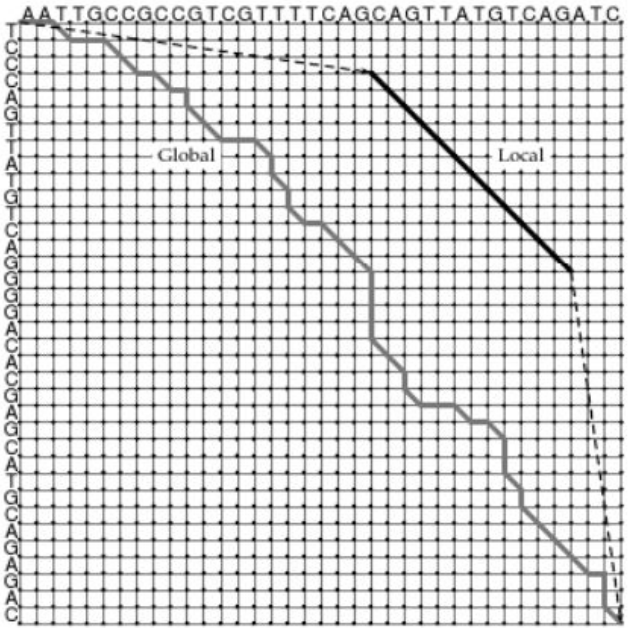
Local Alignment - map part of v to part of w

Local Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find a substring of $\mathbf{v}$ and a substring of $\mathbf{w}$ whose alignment has maximum global alignment score s^* among *all* global alignments of *all* substrings of $\mathbf{v}$ and $\mathbf{w}$

$$s[i,j] = \max \begin{cases} 0, & \text{Start anywhere} \quad \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0, \\ s[i,j-1] + \delta(-, w_j), & \text{if } j > 0, \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$

$$s^* = \max_{i,j} s[i,j] \quad \text{End anywhere}$$

```
                                    tccCAGTTATGTCAGgggacacgagcatgcagagac
                                    ||||||||||||
                aattgccgccgtcgttttcagCAGTTATGTCAGatc
```



Gapped Alignment - not a new problem but a new scoring fn.

$$s^{\rightarrow}[i,j] = \max \begin{cases} s^{\rightarrow}[i,j-1] - \sigma, & \text{if } j > 1, \\ s^{\searrow}[i,j-1] - (\sigma + \rho), & \text{if } j > 0, \end{cases}$$

$$s^{\searrow}[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s^{\rightarrow}[i,j], & \text{if } j > 0, \\ s^{\downarrow}[i,j], & \text{if } i > 0, \\ s^{\searrow}[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0, \end{cases}$$

$$s^{\downarrow}[i,j] = \max \begin{cases} s^{\downarrow}[i-1,j] - \sigma, & \text{if } i > 1, \\ s^{\searrow}[i-1,j] - (\sigma + \rho), & \text{if } i > 0. \end{cases}$$

BLOSUM Matrix - a scoring matrix

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
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

We need two models:

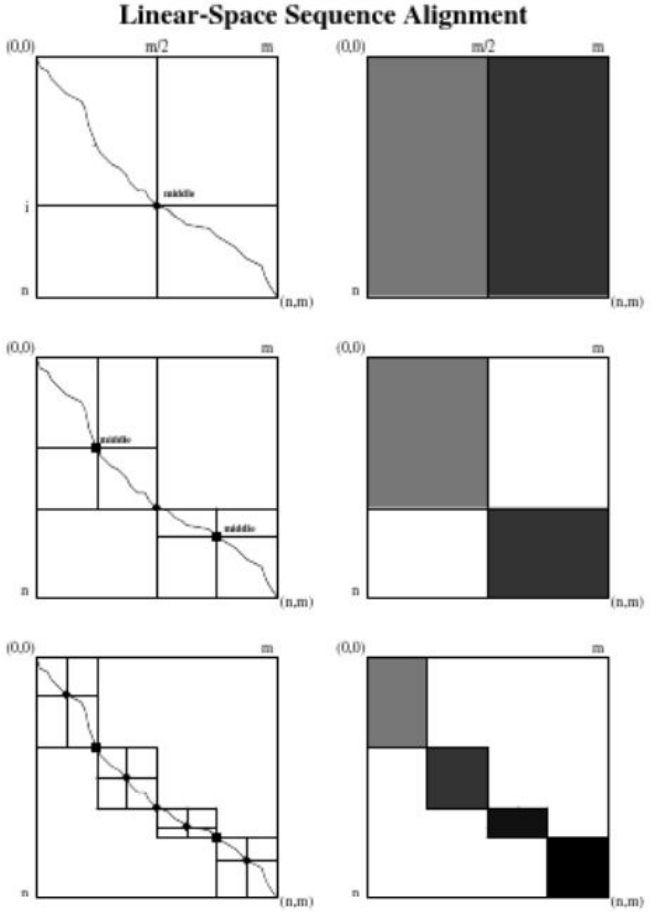
- Random model R: each letter $a \in \Sigma$ occurs independently with probability q_a
- Match model M: aligned pair $(a, b) \in \Sigma \times \Sigma$ occur with joint probability $p_{a,b}$

$$\Pr(\mathbf{v}, \mathbf{w}|\, R) = \prod_i q_{v_i} \cdot \prod_i q_{w_i}$$

$$\Pr(\mathbf{v}, \mathbf{w}|\, M) = \prod_i p_{v_i, w_i}$$

$$\log \frac{\Pr(\mathbf{v}, \mathbf{w}|\, M)}{\Pr(\mathbf{v}, \mathbf{w}|\, R)} = \sum_i s(v_i, w_i) \text{ where } s(a, b) = \log \frac{p_{a,b}}{q_a q_b}$$

Hirschberg - saving space!



Hirschberg(i, j, i', j')

1. **if** $j' - j > 1$
2. $i^* \leftarrow \arg\max_{i \le i'' \le i'} \mathrm{wt}(i'')$
3. Report $(i^*, j + \frac{j'-j}{2})$
4. Hirschberg$(i, j, i^*, j + \frac{j'-j}{2})$
5. Hirschberg$(i^*, j + \frac{j'-j}{2}, i', j')$

Time:

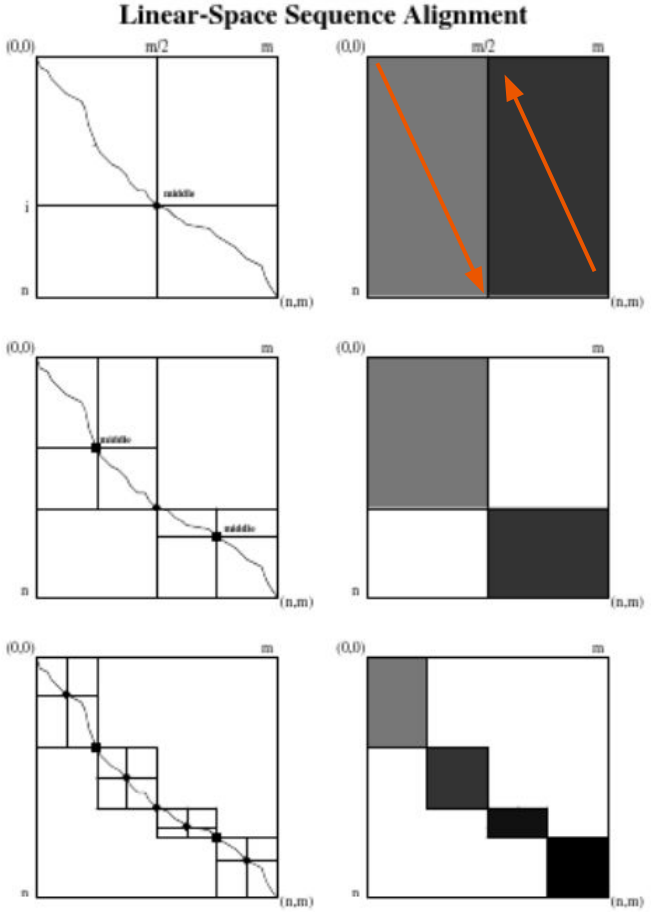
area + area/2 + area/4 + …

= area (1 + ½ + ¼ + ⅛ + …)

≤ 2 × area = O(mn)

Space: O(m)

Hirschberg - saving space by sacrificing a little bit of time!



Hirschberg(i, j, i', j')

1. **if** $j' - j > 1$
2. $i^* \leftarrow \arg\max_{i \le i'' \le i'} \mathrm{wt}(i'')$
3. Report $(i^*, j + \frac{j'-j}{2})$
4. Hirschberg$(i, j, i^*, j + \frac{j'-j}{2})$
5. Hirschberg$(i^*, j + \frac{j'-j}{2}, i', j')$

Time:

area + area/2 + area/4 + …

= area (1 + ½ + ¼ + ⅛ + …)

≤ 2 × area = O(mn)

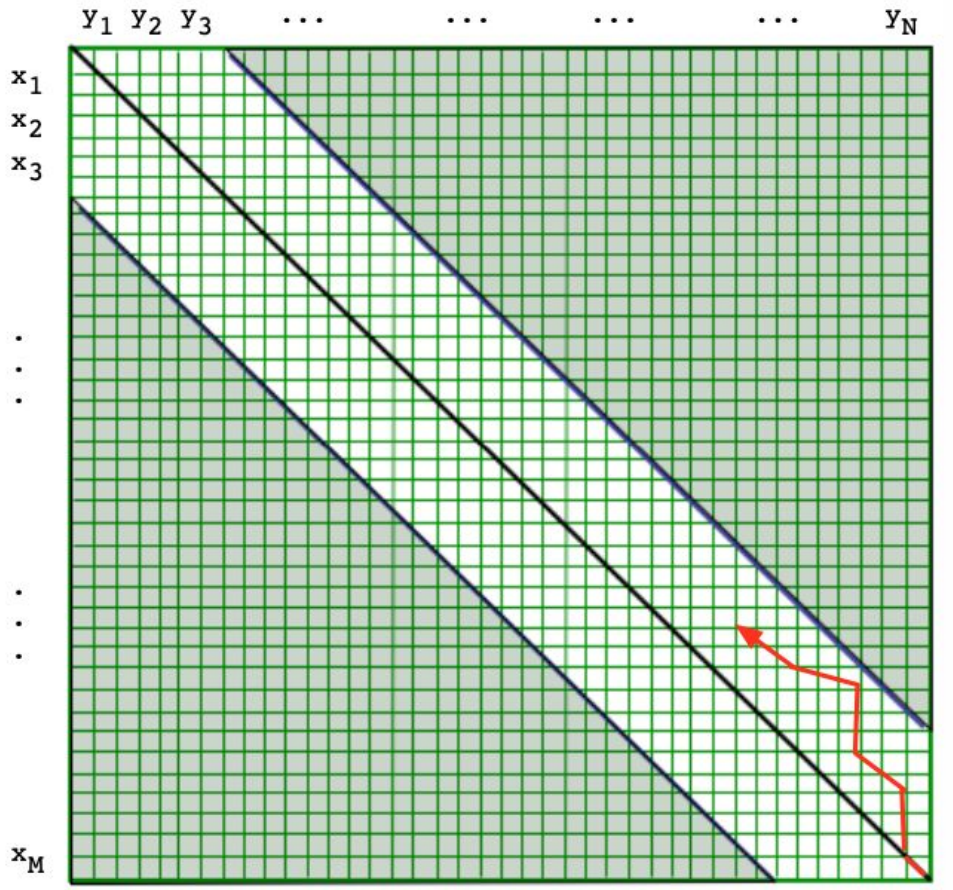
Space: O(m)

Banded Alignment - saving time with approximation!

Constraint path to band of width k around diagonal

Running time: $O(nk)$

Gives a good approximation of highly identical sequences



Banded Alignment - saving time with approximation!

$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0 \text{ and } j - i < k, \\ s[i,j-1] + \delta(-, w_j), & \text{if } j > 0 \text{ and } i - j < k, \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$

Banded Alignment - saving time with approximation!

$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0 \text{ and } j - i < k, \\ s[i,j-1] + \delta(-, w_j), & \text{if } j > 0 \text{ and } i - j < k, \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$

Block Alignment/Four Russian Tech. - also approximation for speed!

$0 \le i, j \le n/t$ and $\beta(i,j)$ is max score alignment between block i of $\mathbf{v}$ and block j of $\mathbf{w}$

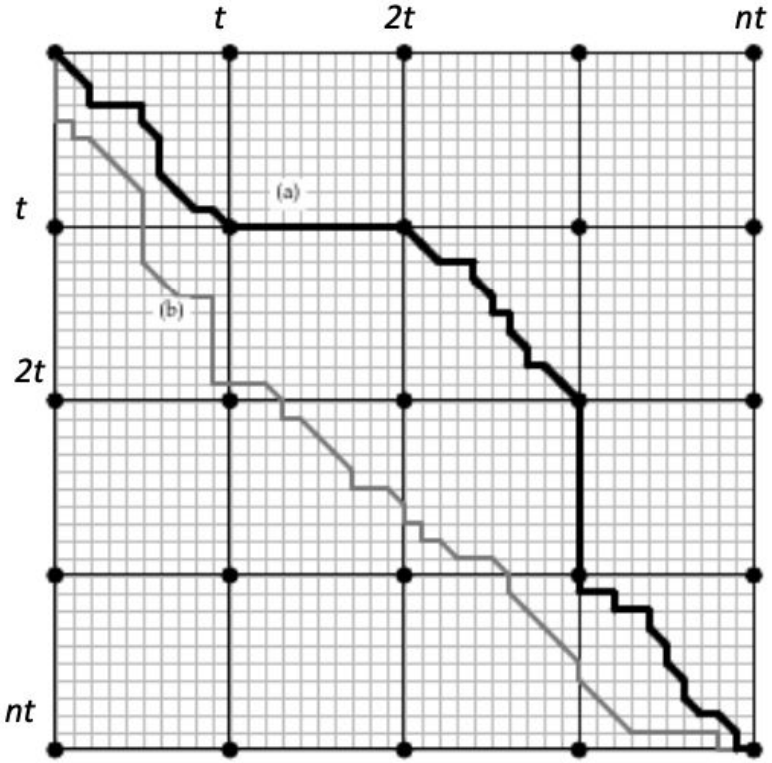
$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] - \sigma, & \text{if } i > 0, \\ s[i,j-1] - \sigma, & \text{if } j > 0, \\ s[i-1,j-1] + \beta(i,j), & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$

Question:
How much time to compute all $\beta(i,j)$?

Computing $\beta(i,j)$ takes $O(t^2)$ time

There are $n/t \times n/t$ values $\beta(i,j)$

Total: $O\left(\frac{n}{t} \times \frac{n}{t} \times t^2\right) = O(n^2)$ time



Block Alignment/Four Russian Tech. - also approximation for speed!

t=log(n)

Algorithm:

1. Precompute $S[\mathbf{v}', \mathbf{w}']$ where $\mathbf{v}', \mathbf{w}' \in \Sigma^t$
2. Compute block alignment between $\mathbf{v}$ and $\mathbf{w}$ using S

$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] - \sigma, & \text{if } i > 0, \\ s[i,j-1] - \sigma, & \text{if } j > 0, \\ s[i-1,j-1] + S[v(i), w(j)], & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$

Dynamic Programming - translate the problem to an objective

Global Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find alignment with maximum score.

- An alignment is a source-to-sink path in the edit graph
- An alignment $\mathbf{A} = [a_{i,j}]$ is a $2 \times k$ matrix s.t. (i) $k = \{\max(m,n), \ldots, m+n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0, \quad \text{deletion} \\ s[i,j-1] + \delta(-, w_j), & \text{if } j > 0, \quad \text{insertion} \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \quad \text{match/mismatch} \end{cases}$$

Dynamic Programming - define a variable that allows decomposition

Global Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find alignment with maximum score.

- An alignment is a source-to-sink path in the edit graph
- An alignment $\mathbf{A} = [a_{i,j}]$ is a $2 \times k$ matrix s.t. (i) $k = \{\max(m,n), \ldots, m+n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0, \quad \text{deletion} \\ s[i,j-1] + \delta(-, w_j), & \text{if } j > 0, \quad \text{insertion} \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \quad \text{match/mismatch} \end{cases}$$

Dynamic Programming - define the recursion

Global Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find alignment with maximum score.

- An alignment is a source-to-sink path in the edit graph
- An alignment $\mathbf{A} = [a_{i,j}]$ is a $2 \times k$ matrix s.t. (i) $k = \{\max(m,n), \ldots, m+n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0, \quad \text{deletion} \\ s[i,j-1] + \delta(-, w_j), & \text{if } j > 0, \quad \text{insertion} \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \quad \text{match/ mismatch} \end{cases}$$

Dynamic Programming - define the base case

Global Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find alignment with maximum score.

- An alignment is a source-to-sink path in the edit graph
- An alignment $\mathbf{A} = [a_{i,j}]$ is a $2 \times k$ matrix s.t. (i) $k = \{\max(m,n), \ldots, m+n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0, \quad \text{deletion} \\ s[i,j-1] + \delta(-, w_j), & \text{if } j > 0, \quad \text{insertion} \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \quad \text{match/mismatch} \end{cases}$$

Dynamic Programming - define the final solution

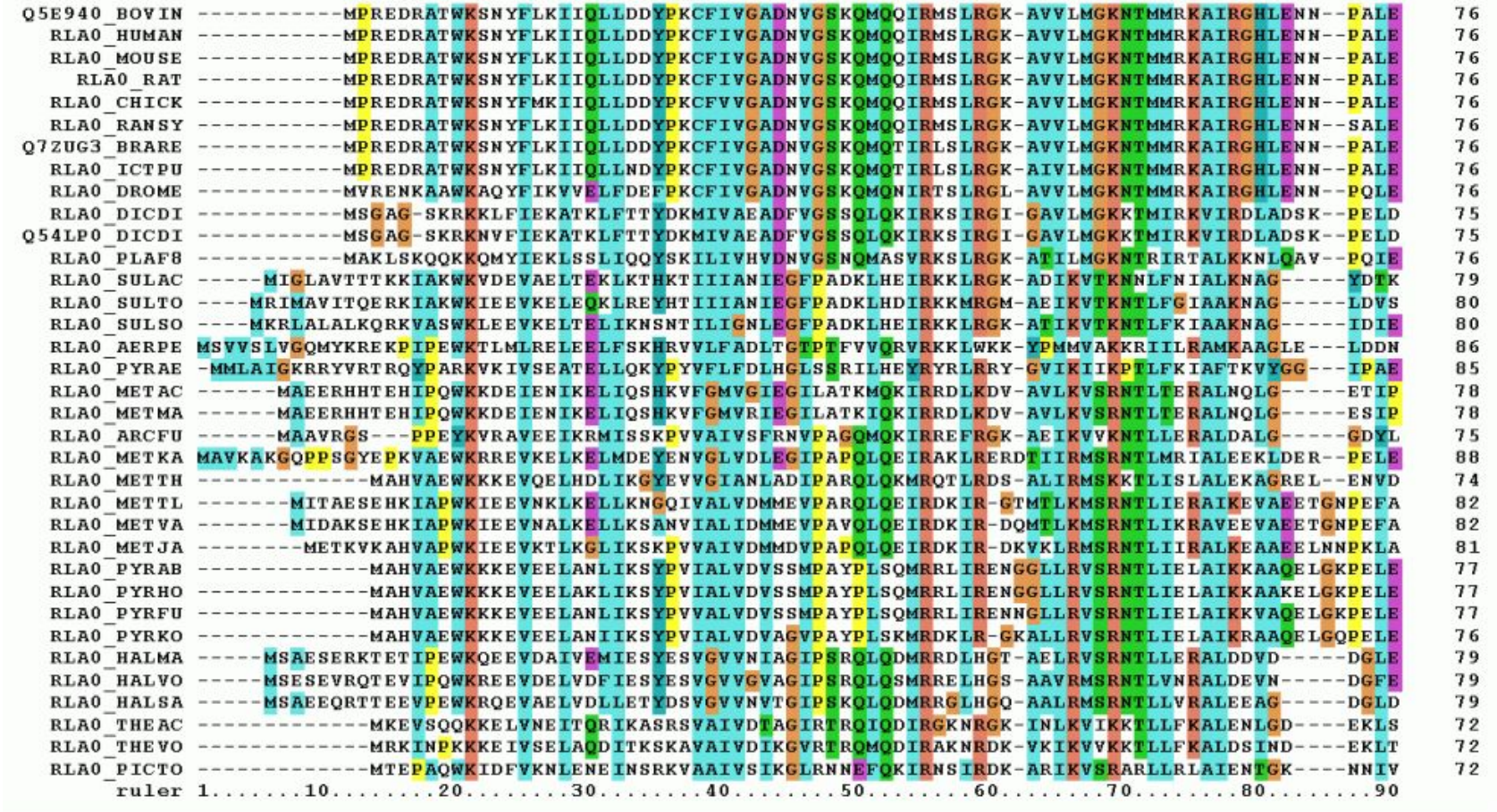
Global Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find alignment with maximum score.

- An alignment is a source-to-sink path in the edit graph
- An alignment $\mathbf{A} = [a_{i,j}]$ is a $2 \times k$ matrix s.t. (i) $k = \{\max(m,n), \ldots, m+n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

s[m,n]

$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0, \quad \text{deletion} \\ s[i,j-1] + \delta(-, w_j), & \text{if } j > 0, \quad \text{insertion} \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \quad \text{match/mismatch} \end{cases}$$

MSA - multiple sequence alignment

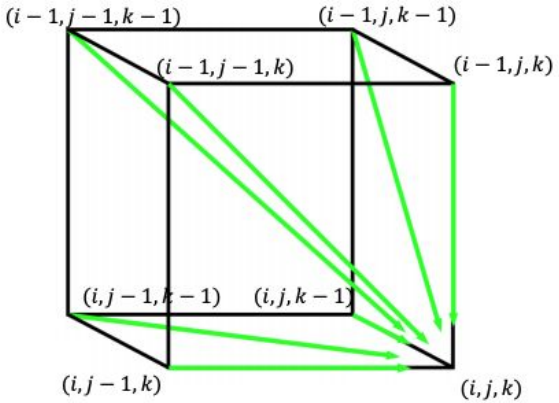


MSA with k-D scoring matrix using DP

3-D Sequence Alignment

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

Given three sequences each of length n, running time: $O(n^3)$



$$s[i,j,k] = \max \begin{cases} s[i-1,j-1,k-1] + \delta(v_i, w_j, u_k), & \text{no gaps} \\ s[i-1,j-1,k] + \delta(v_i, w_j, -), & \\ s[i-1,j,k-1] + \delta(v_i, -, u_k), & \text{one gap} \\ s[i,j-1,k-1] + \delta(-, w_j, u_k), & \\ s[i-1,j,k] + \delta(v_i, -, -), & \\ s[i,j-1,k] + \delta(-, w_j, -), & \text{two gaps} \\ s[i,j,k-1] + \delta(-, -, u_k), & \end{cases}$$

MSA with SP-score (MSA-SP)

MSA-SP problem: Given strings strings $\mathbf{v}_1, \ldots, \mathbf{v}_k$ find multiple sequence alignment $\mathcal{M}^*$ with **minimum** value of

$$\text{SP-score}(\mathcal{M}^*) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} S(\mathbf{v}_i, \mathbf{v}_j)$$

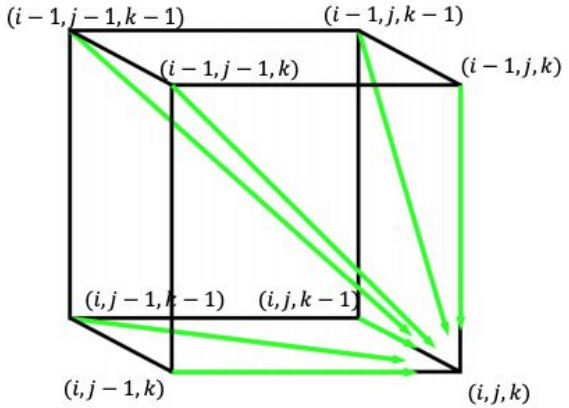
where $S(\mathbf{v}_i, \mathbf{v}_j)$ is the score of the induced pairwise alignment of $(\mathbf{v}_i, \mathbf{v}_j)$ in $\mathcal{M}^*$

MSA-SP using DP

3-D MSA-SP

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

Given three sequences each of length n, running time: $O(n^3)$



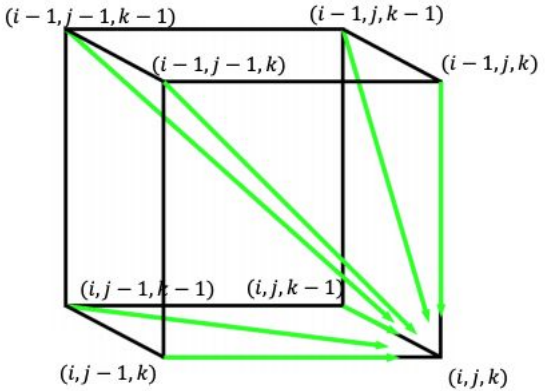
$$d[i_1, i_2, i_3] = \min \begin{cases} d[i_1-1, i_2-1, i_3-1] + \delta(\mathbf{v_1}[i_1], \mathbf{v_2}[i_2]) + \delta(\mathbf{v_1}[i_1], \mathbf{v_3}[i_3]) + \delta(\mathbf{v_2}[i_2], \mathbf{v_3}[i_3]) & \text{no gaps} \\ d[i_1-1, i_2-1, i_3] + \delta(\mathbf{v_1}[i_1], \mathbf{v_2}[i_2]) + 2\sigma \\ d[i_1-1, i_2, i_3-1] + \delta(\mathbf{v_1}[i_1], \mathbf{v_3}[i_3]) + 2\sigma & \text{one gap} \\ d[i_1, i_2-1, i_3-1] + \delta(\mathbf{v_2}[i_2], \mathbf{v_3}[i_3]) + 2\sigma \\ d[i_1-1, i_2, i_3] + 2\sigma \\ d[i_1, i_2-1, i_3] + 2\sigma & \text{two gaps} \\ d[i_1, i_2, i_3-1] + 2\sigma \end{cases}$$

MSA-SP using DP

k-D MSA-SP

Computing SP-score in each case: $O(k^2)$ time

Given k sequences each of length n,
running time: $O(k^2 2^k n^k)$

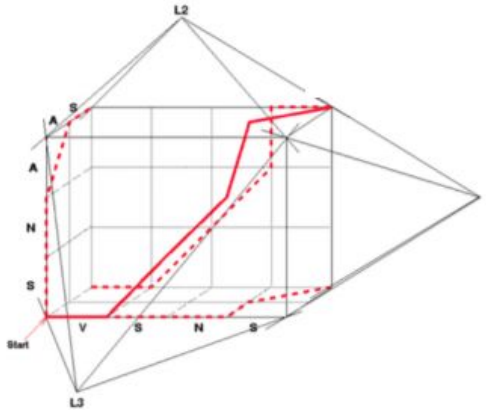


$$d[i_1, i_2, \ldots, i_{k-1}, i_k] = \min \begin{cases} s[i_1-1, i_2-1, \ldots, i_{k-1}-1, i_k-1] + \sum_{p=1}^{k} \sum_{q=p+1}^{k} \delta(\mathbf{v_p}[i_p], \mathbf{v_q}[i_q]) & \text{no gaps} \\ s[i_1-1, i_2-1, \ldots, i_{k-1}-1, i_k] + (k-1)\sigma + \sum_{p=1}^{k-1} \sum_{q=p+1}^{k-1} \delta(\mathbf{v_p}[i_p], \mathbf{v_q}[i_q]) & \\ \quad \vdots & \text{one gap} \\ s[i_1, i_2-1, \ldots, i_{k-1}-1, i_k-1] + (k-1)\sigma + \sum_{p=2}^{k} \sum_{q=p+1}^{k} \delta(\mathbf{v_p}[i_p], \mathbf{v_q}[i_q]) & \\ \quad \vdots & \\ s[i_1-1, i_2, \ldots, i_{k-1}, i_k] + (k-1)\sigma & \\ \quad \vdots & k-1 \text{ gaps} \\ s[i_1, i_2, \ldots, i_{k-1}, i_k-1] + (k-1)\sigma & \end{cases}$$

MSA-SP using DP using Carrillo Lipman Optimization

$$D^+(i,j,k) = d^+_{1,2}(i,j) + d^+_{1,3}(i,k) + d^+_{2,3}(j,k) \geq D^+_{1,2}(i,j) + D^+_{1,3}(i,k) + D^+_{2,3}(j,k)$$

$$D(i,j,k) + D^+(i,j,k) \geq D(i,j,k) + D^+_{1,2}(i,j) + D^+_{1,3}(i,k) + D^+_{2,3}(j,k)$$



Question: What if we have an alignment with cost z?

If $z < D(i,j,k) + D^+_{1,2}(i,j) + D^+_{1,3}(i,k) + D^+_{2,3}(j,k)$
then (i,j,k) not on optimal path => **Prune**!

Greedy Progressive MSA - generate profile

```
  -  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
```

A **profile** $P = [p_{i,j}]$ is a $(|\Sigma| + 1) \times l$ matrix,
where $p_{i,j}$ is the frequency of i-th letter in j-th position of alignment

Greedy Progressive MSA - align sequence to profile

Aligning String to Profile

$$\tau(x,j) = \sum_{y \in \Sigma \cup \{-\}} p_{y,j} \cdot \delta(x,y)$$

$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i,-), & \text{if } i > 0, \\ s[i,j-1] + \tau(-,j), & \text{if } j > 0, \\ s[i-1,j-1] + \tau(v_i,j), & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$

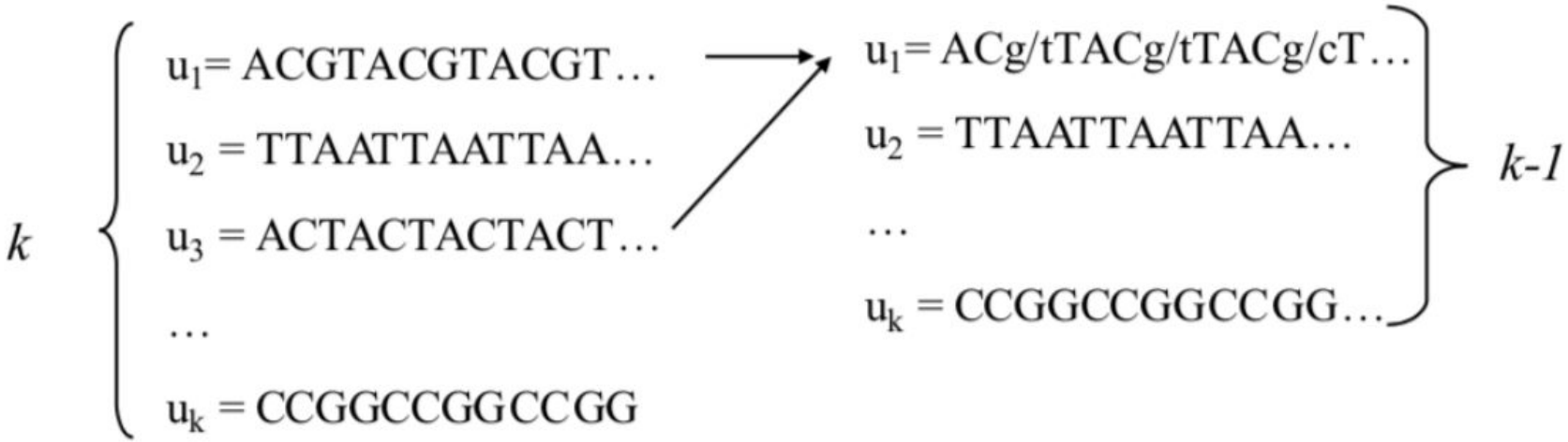
Insert space in profile

Insert space in string

- $s[i,j]$ is optimal alignment of $v_1, \ldots, v_i$ and first j columns of P
- $\delta(x,y)$ is score for aligning characters x and y
- $\tau(x,j)$ is score for aligning character x and column j of P

Greedy Progressive MSA - the greedy algorithm

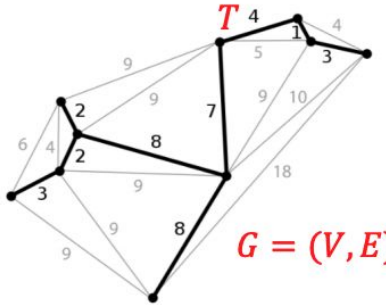
Choose most similar pair among ***k*** input strings, combine into a profile. This reduces the original problem to alignment of ***k-1*** sequences to a profile. Repeat.



Progressive MSA - order matters!

- **Feng and Doolittle: guided by a minimum spanning tree from a complete graph of pairwise sequence alignment**

1. Compute pairwise sequence alignments of n sequences
2. Generate complete graph $G = (V, E)$ with edge weights w : $E \to \mathbb{R}$
3. Compute a (rooted) minimum spanning tree T of G
4. Perform sequence-sequence, sequence-alignment and alignment-alignment alignment to construct MSA according to guide tree T (from most similar to least similar)



Minimum spanning tree is a tree T spanning all vertices of G with minimum total weight

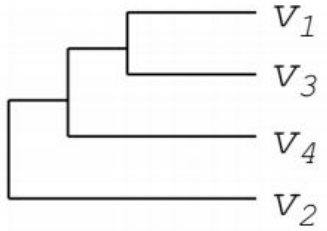
Progressive MSA - order matters!

- **ClustalW: guided by a tree constructed via neighbor-joining method**

Create Guide Tree using the similarity matrix

("cluster" distances. Details to come...)

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

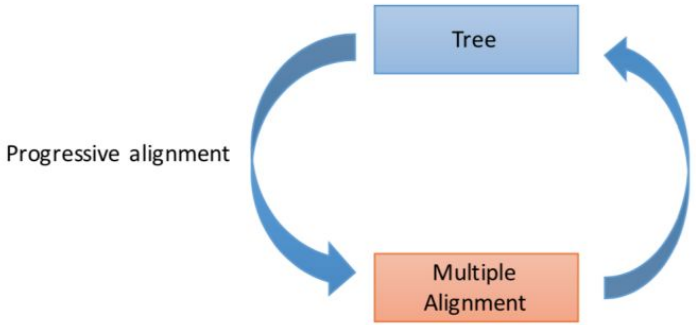


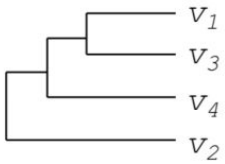
ClustalW uses the neighbor-joining method

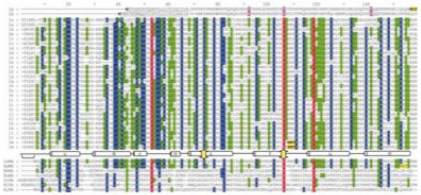
Guide tree roughly reflects evolutionary relationships

Progressive MSA - order matters!

- MUSCLE: tree->alignment->tree->alignment->...-> until converge







Circularity!

Ideally, want to derive alignment and tree simultaneously → Hard

Weighted SP-Edit Distance problem - flip the sign.

MSA-SP problem: Given strings $\mathbf{v}_1, \ldots, \mathbf{v}_k$ and scoring function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \to \mathbb{R}$, find multiple sequence alignment $\mathcal{M}^*$ with **maximum** value of SP-score$(\mathcal{M}^*) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} S(\mathbf{v}_i, \mathbf{v}_j)$ where $S(\mathbf{v}_i, \mathbf{v}_j)$ is the score of the induced pairwise alignment of $(\mathbf{v}_i, \mathbf{v}_j)$ in $\mathcal{M}^*$ using δ

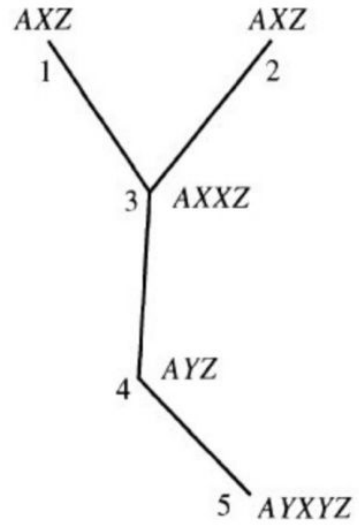
Weighted SP-Edit Distance problem: Given strings $\mathbf{v}_1, \ldots, \mathbf{v}_k$ and cost function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \to \mathbb{R}$, find multiple sequence alignment $\mathcal{M}^*$ with **minimum** value of SP-score$(\mathcal{M}^*) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} S(\mathbf{v}_i, \mathbf{v}_j)$ where $S(\mathbf{v}_i, \mathbf{v}_j)$ is the cost of the induced pairwise alignment of $(\mathbf{v}_i, \mathbf{v}_j)$ in $\mathcal{M}^*$ using δ

Weighted SP-Edit Distance problem - tree alignment.

The following theorem states that it is easy to compute an alignment that is consistent with a given tree T.

Theorem 1 (Gusfield [1]). *Let T be a tree whose k nodes are each labeled by a distinct string from $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$. We can compute an alignment $A(T)$ of $\mathbf{v}_1, \ldots, \mathbf{v}_k$ that is consistent with T in $O(k^2n^2)$ time.*

a)



b)

3	A	X	X	_	Z
1	A	X	_	_	Z
2	A	_	X	_	Z
4	A	Y	_	_	Z
5	A	Y	X	X	Z

Weighted SP-Edit Distance problem - center star alignment.

Recall that $D(\mathbf{v}_i, \mathbf{v}_j)$ is the optimal (weighted) edit distance between $\mathbf{v}_i$ and $\mathbf{v}_j$. We have the following definition.

Definition 3. *Given strings* $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \Sigma^*$, *the* center string $\mathbf{v}_c$ *(where* $c \in [k]$*) is the input string that minimizes* $\sum_{i=1}^{k} D(\mathbf{v}_c, \mathbf{v}_i)$. *The* center star *is a star tree of* k *nodes with the center node labeled by* $\mathbf{v}_c$ *and each of the remaining* $k-1$ *nodes labeled by a distinct string from* $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \setminus \{\mathbf{v}_c\}$.

Theorem 1 (Gusfield [1])**.** *Let* T *be a tree whose* k *nodes are each labeled by a distinct string from* $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$. *We can compute an alignment* $A(T)$ *of* $\mathbf{v}_1, \ldots, \mathbf{v}_k$ *that is consistent with* T *in* $O(k^2n^2)$ *time.*

Theorem 2. $d(A_c)/d(A^*) \leq 2(k-1)/k < 2.$