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 $n_0 \ge 0$ such that $f(n) \le c g(n)$ for all $n \ge n_0$

Guideline

• $O(n^a) \subset O(n^b)$ for any positive cor	nstants	a < 1	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
0(1)	Constant
$O(\log n)$	Logarithmic
0(n)	Linear
$O(n^2)$	Quadratic
$O(n^c) = O(\text{poly}(n))$	Polynomial
$O(2^{\operatorname{poly}(n)})$	Exponential

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

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

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

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

- 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) $k = \{\max(m, n), \dots, m + n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

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

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}



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}

	0, Start anywhere	if $i = 0$ and $j = 0$,
$s[i,j] = \max k$	$s[i-1,j] + \delta(v_i,-),$	if $i > 0$,
	$s[i, j-1] + \delta(-, w_j),$	if $j > 0$,
25	$s[i-1, j-1] + \delta(v_i, w_j),$	if $i > 0$ and $j > 0$.
$s^* = \max_{i,j} s$	$\left[i,j ight]$ End anywhere	

tccCAGTTATGTCAGgggacacgagcatgcagagac



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

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

BLOSUM Matrix – a scoring matrix

Ala 4 -15 Arg -2 Asn 0 6 -2 -2 Asp 6 $\Pr(\mathbf{v}, \mathbf{w} | R) = \left[\begin{array}{c} q_{v_i} \cdot \\ \end{array} \right] q_{w_i}$ $\Pr(\mathbf{v}, \mathbf{w} | M) = \left| p_{v_i, w_i} \right|$ Cys 0 -3 -3 -3 9 Gln -1 1 0 0 -3 5 0 $\log \frac{\Pr(\mathbf{v}, \mathbf{w} \mid M)}{\Pr(\mathbf{v}, \mathbf{w} \mid R)} = \sum_{i} s(v_i, w_i) \text{ where } s(a, b) = \log \frac{p_{a,b}}{q_a q_b}$ Glu -10 2 -4 2 5 -3 -2 -2 Gly 0 -2 0 -16 His -2 0 -3 0 0 -2 1 -18 -3 -3 -3 lle -3 -1 -3-2 -2 -3-4 -32 Leu -4 4 2 -3 -2 -3 -2 Lys -10 $^{-1}$ 1 5 Met -3 -10 -2 -3 -1 -1 -2 2 5 Phe -3 -3 -2 -3-3 -3 0 -3 0 0 6 -2 -2 -3Pro $^{-1}$ -3 -3 -1-1-1-4 Ser -11 0 -10 0 0 -1 -2-2 0 -24 Thr 0 -10 -1 -1 -1-2 -2 -1-1 _1 -1 -2 $^{-1}$ 5 Trp -3 -3 -2 -311 -4 -2 -3 -2 -2 -3 $^{-1}$ 1 -3 -2 -4 -3 -2 Tyr -2 -2 -2 -3 -2 -1-2 2 -1-1-2 -13 -3 -2 2 7 Val -3 -3 -2 -2 -3 -3 3 1 -2 -2 -2 -3-10 - 4 Arg Asn Asp Cys Gln Glu Gly His Ile Leu Lys Met Phe Pro Ser Thr Trp Tyr Val Ala

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 $p_{a,b}$

Hirschberg - saving space!



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

- **1.** if j' j > 1
- 2. $i^* \leftarrow \underset{i \leq i'' \leq i'}{\operatorname{arg max wt}(i'')}$

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.
$$i^* \leftarrow \arg \max_{i \le i'' \le i'} \operatorname{wt}(i'')$$

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 + ⅓ + ...) ≤ 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

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

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

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



Algorithm:

1. Precompute $S[\mathbf{v}', \mathbf{w}']$ where $\mathbf{v}', \mathbf{w}' \in \Sigma^t$

t=log(n)

2. Compute block alignment between **v** and **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

- 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) $k = \{\max(m, n), \dots, m + n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

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

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) $k = \{\max(m, n), \dots, m + n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

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

Dynamic Programming - define the recursion

- 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) $k = \{\max(m, n), \dots, m + n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

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

Dynamic Programming - define the base case

- 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) $k = \{\max(m, n), \dots, m + n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

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

Dynamic Programming - define the final solution

- 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) $k = \{\max(m, n), \dots, m + n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$

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

MSA - multiple sequence alignment

Q5E940_BOVIN	MPREDRATWKSNYFLKIIQLLDDYPKCFIYGADNYGSKQMQQIRMSLRGK-AYYLMGKNTMMRKAIRGHLENNPALE	76
RLA0 HUMAN	MPREDRATWKSNYFLKIIQLLDDYPKCFIYGADNYGSKQMQQIRMSLRGK-AYYLMGKNTMMRKAIRGHLENNPALE	76
RLA0 MOUSE	MPREDRATWKSNYFLKIIQLLDDYPKCFIYGADNYGSKQMQQIRMSLRGK-AYYLMGKNTMMRKAIRGHLENNPALE	76
RLAO RAT	MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENNPALE	76
RLA0 CHICK	MPREDRATWKSNYFMKIIQLLDDYPKCFVVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENNPALE	76
RLAO RANSY	MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENNSALE	76
Q7ZUG3 BRARE	MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQTIRLSLRGK-AVVLMGKNTMMRKAIRGHLENNPALE	76
RLA0 ICTPU	mpredratwksnyflkiiqllndypkcfivgadnvgskomqtirlslrgk-aivlmgkntmmrkairghlennpale	76
RLA0 DROME	MYRENKAAWKAQYFIKYYELFDEF <mark>PKCFIYGADNYGS</mark> KQMQNIRTSLRGL-AVYLMGKNTMMRKAIRGHLENNPQLE	76
RLA0 DICDI	MSGAG-SKRKKLFIEKATKLFTTYDKMIVAEADFVGSSOLOKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSKPELD	75
Q54LP0 DICDI	MSGAG-SKRKNYFIEKATKLFTTYDKMIYAEADFYGSSOLOKIRKSIRGI-GAYLMGKKTMIRKYIRDLADSKPELD	75
RLAO PLAF8	MAK LSKOOK KOMY IEK LSSLIQOYSKILIYHYD NYGS NOMASYRKSLRGK-AT ILMGKNTRIRTALKKNLOAYPOIE	76
RLA0 SULAC	MIGLAVTTTKKIAKWKVDEVAELTEKLKTHKTIIIANIEGFPADKLHEIRKKLRGK-ADIKVTKNNLFNIALKNAGYDTK	79
RLA0 SULTO	MRIMAVITOERKIAKWKIEEVKELEOKLREYHTIIIANIEGFPADKLHDIRKKMRGM-AEIKVTKNTLFGIAAKNAGLDVS	80
RLA0 SULSO	MKRLALALKORKVASWKLEEVKELTELIKNSNTILIGNLEGFPADKLHEIRKKLRGK-ATIKVTKNTLFKIAAKNAGIDIE	80
RLA0 AERPE	MSVVSLVGQMYKREKPIPEWKTLMLRELEELFSKHRVVLFADLTGTPTFVVQRVRKKLWKK-YPMMVAKKRIILRAMKAAGLELDDN	86
RLA0 PYRAE	-MMLAIGKRRYVRTROYPARKVKIVSEATELLOKYPYVFLFDLHGLSSRILHEYRYRLRRY-GVIKIIKPTLFKIAFTKVYGGIPAE	85
RLAO METAC	MAEERHHTEHIPOWKKDEIENIKELIQSHKVFGMVGIEGILATKMOKIRRDLKDV-AVLKVSRNTLTERALNQLGETIP	78
RLAO METMA	MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVRIEGILATKIQKIRRDLKDV-AVLKVSRNTLTERALNQLGESIP	78
RLA0 ARCFU	MAAVRGSPPEYKVRAVEEIKRMISSKPVVAIVSFRNVPAGOMOKIRREFRGK-AEIKVVKNTLLERALDALGGDYL	75
RLAO METKA	MAYKAKGOPPSGYEPKVAEWKRREVKELKELMDEYENVGLVDLEGIPAPOLOEIRAKLRERDTIIRMSRNTLMRIALEEKLDERPELE	88
RLA0 METTH	MAHVAEWKKKEVQELHDLIKGYEVVGIANLADIPARQLQKMRQTLRDS-ALIRMSKKTLISLALEKAGRELENVD	74
RLA0 METTL	MITAESEHKIAPWKIEEVNKLKELLKNGQIVALVDMMEVPAROLOEIRDKIR-GTMTLKMSRNTLIERAIKEVAEETGNPEFA	82
RLAO METVA	MIDAKSEHKIAPWKIEEVNALKELLKSANVIALIDMMEVPAVQLQEIRDKIR-DQMTLKMSRNTLIKRAVEEVAEETGNPEFA	82
RLAO METJA	METKYKAHVAPWKIEEVKTLKGLIKSKPVVAIVDMMDVPAPOLOEIRDKIR-DKYKLRMSRNTLIIRALKEAAEELNNPKLA	81
RLA0 PYRAB	MAHVAEWKKKEVEELANLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAQELGKPELE	77
RLA0 PYRHO	MAHVAEWKKKEVEELAKLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAKELGKPELE	77
RLA0 PYRFU	MAHVAEWKKKEVEELANLIKSYPVVALVDVSSMPAYPLSQMRRLIRENNGLLRVSRNTLIELAIKKVAQELGKPELE	77
RLA0 PYRKO	MAHVAEWKKKEVEELANIIKSYPVIALVDVAGVPAYPLSKMRDKLR-GKALLRVSRNTLIELAIKRAAQELGQPELE	76
RLAO HALMA	MSAE SERKTET IPEWKQEEVDAIVEMIESYESYGYVNIAGIPSRQLQDMRRDLHGT-AELRYSRNTLLERALDDVDDGLE	79
RLA0 HALVO	MSESEVRQTEVIPQWKREEVDELVDFIESYESVGVVGVAGIPSRQLQSMRRELHGS-AAVRMSRNTLVNRALDEVNDGFE	79
RLAO HALSA	MSAEEQRTTEEVPEWKRQEVAELVDLLETYDSVGVVNVTGIPSKQLQDMRRGLHGQ-AALRMSRNTLLVRALEEAGDGLD	79
RLAO THEAC	MKEVSQQKKELVNEIT OR IKASRSVAIVOTAGIRT ROIODIRGKNRGK-INLKVIKKTLLFKALENLGDEKLS	72
RLAO THE VO	MRKINPKKKEIVSELAQDITKSKAVAIVDIKGVRTROMODIRAKNRDK-VKIKVVKKTLLFKALDSINDEKLT	72
RLAO PICTO	MTEPAQWKIDFVKNLENEINSRKVAAIVSIKGLRNNEFOKIRNSIRDK-ARIKVSRARLLRLAIENTGKNNIV	72
ruler	$1 \dots 10 \dots 20 \dots 30 \dots 40 \dots 50 \dots 60 \dots 70 \dots 80 \dots 90$	

MSA with k-D scoring matrix using DP



MSA-SP problem: Given strings strings $\mathbf{v}_1, ..., \mathbf{v}_k$ 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 score of the induced pairwise alignment of $(\mathbf{v}_i, \mathbf{v}_j)$ in \mathcal{M}^* 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: $O(n^3)$



$$d[i_{1}, i_{2}, i_{3}] = \min \begin{cases} d[i_{1} - 1, i_{2} - 1, i_{3} - 1] + \delta(\mathbf{v}_{1}[i_{1}], \mathbf{v}_{2}[i_{2}]) + \delta(\mathbf{v}_{1}[i_{1}], \mathbf{v}_{3}[i_{3}]) + \delta(\mathbf{v}_{2}[i_{2}], \mathbf{v}_{3}[i_{3}]) \\ d[i_{1} - 1, i_{2}, i_{3} - 1] + \delta(\mathbf{v}_{1}[i_{1}], \mathbf{v}_{3}[i_{3}]) + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + \delta(\mathbf{v}_{2}[i_{2}], \mathbf{v}_{3}[i_{3}]) + 2\sigma \\ d[i_{1} - 1, i_{2}, i_{3}] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_{1}, i_{2} - 1, i_{3} - 1] + 2\sigma \\ d[i_$$

k-D MSA-SP

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

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



$$d[i_{1}, i_{2}, \dots, i_{k-1}, i_{k}] = \min \begin{cases} s[i_{1} - 1, i_{2} - 1, \dots, i_{k-1} - 1, i_{k} - 1] + \sum_{p=1}^{k} \sum_{q=p+1}^{k} \delta(\mathbf{v_{p}}[i_{p}], \mathbf{v_{q}}[i_{q}]) \\ s[i_{1} - 1, i_{2} - 1, \dots, i_{k-1} - 1, i_{k}] + (k-1)\sigma + \sum_{p=1}^{k-1} \sum_{q=p+1}^{k-1} \delta(\mathbf{v_{p}}[i_{p}], \mathbf{v_{q}}[i_{q}]) \\ \vdots \\ s[i_{1}, i_{2} - 1, \dots, i_{k-1} - 1, i_{k} - 1] + (k-1)\sigma + \sum_{p=2}^{k} \sum_{q=p+1}^{k} \delta(\mathbf{v_{p}}[i_{p}], \mathbf{v_{q}}[i_{q}]) \end{cases}$$
 one gap

MSA-SP using DP using Carrillo Lipman Optimization

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

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



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

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

Greedy Progressive MSA - generate profile



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

$$\begin{split} \tau(x,j) &= \sum_{y \in \Sigma \cup \{-\}} p_{y,j} \ \delta(x,y) \\ s[i,j] &= \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1,j] + \delta(v_i,-), & \text{if } i > 0, & \text{Insert space in profile} \\ s[i,j-1] + \tau(-,j), & \text{if } j > 0, & \text{Insert space in string} \\ s[i-1,j-1] + \tau(v_i,j), & \text{if } i > 0 \text{ and } j > 0. \end{cases} \end{split}$$

- s[i, j] is optimal alignment of v₁, ..., 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

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.

$$k \begin{cases} u_1 = ACGTACGTACGT... & u_1 = ACg/tTACg/tTACg/cT... \\ u_2 = TTAATTAATTAA... & u_2 = TTAATTAATTAA... \\ u_3 = ACTACTACTACT... & ... \\ ... & u_k = CCGGCCGGCCGG \\ & u_k = CCGGCCGGCC$$

- 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*
 - Perform sequence-sequence, sequence-alignment and alignmentalignment 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 ClustalW: guided by a tree constructed via neighbor-joining method

Create Guide Tree using the similarity matrix ("cluster" distances. Details to come...)



ClustalW uses the neighbor-joining method Guide tree roughly reflects evolutionary relationships MUSCLE: tree->alignment->tree->alignment->...-> until converge



Weighted SP-Edit Distance problem - flip the sign.

MSA-SP problem: Given strings $\mathbf{v}_1, ..., \mathbf{v}_k$ and scoring function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathbb{R}$, 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 $\mathbf{v}_1, ..., \mathbf{v}_k$ and cost function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathbb{R}$, 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.



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