Remark: Nearly the same result holds for any scoring schema of practical importance; only C and λ must be adapted (Dembo, Karlin and Zeitouni 1994).

Consequences: The smaller the e-value for a local alignment the larger its significance (e-value tells us how likely score b found for our input results from aligning two random sequences).

Question: How to compare alignments computed with different scoring schemes?

Use the normalized scoring B' with

$$B' := \frac{\lambda B - \ln(C)}{\ln(2)}, \text{ i.e. } B = \frac{B' \cdot \ln(2) + \ln(C)}{\lambda}.$$

Then the e-value with respect to B' (called *bit-scores*) becomes

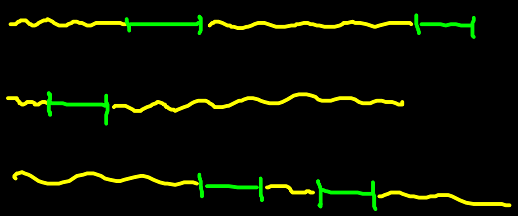
$$E_{B' \geq b} = E_{B \geq \frac{b \cdot \ln(2) + \ln(C)}{\lambda}} \sim m \cdot n \cdot 2^{-b},$$

i.e. asymptotically independent of C and λ.

$\rightsquigarrow$ bit-scores much more appropriate to compare different alignments.

Multiple alignments

Motivation: Alignment of two words only of interest if random events can be ruled out as source of similarities. (Inference of structural or functional relation).
Similarity of two sequences often too small to be detected.
Hope: Common functional parts of multiple sequences get visible by amplifying effects of a multiple alignment.



Definition

Given k words $S_i \in \Sigma^\star$, $1 \le i \le k$, gap symbol $- \notin \Sigma$ and $\Sigma' := \Sigma \cup \{-\}$. Furthermore $h : (\Sigma')^\star \to \Sigma^\star$ the homomorphism induced by $h(a) = a$ for all $a \in \Sigma$ and $h(-) = \varepsilon$.
A multiple alignment of $S_1, \ldots S_k$ is a tuple $(S^{(1)}, \ldots, S^{(k)})$ of words over $(\Sigma')^l$, $l \ge \max\{|S_i| \mid 1 \le i \le k\}$, satisfying:

1. $h(S^{(i)}) = S_i$, $1 \le i \le k$;
2. $(\nexists j \in [1:l])(\forall i \in [1:k] : S^{(i)}_j = -)$.

We call l the length of the multiple alignment.

How to rate such an alignment?

First possibility: *consensus*

Definition

Given a multiple alignment $\mathcal{A} = (S^{(1)}, \ldots, S^{(k)})$ of words $S_1, \ldots,$ $S_k \in \Sigma^\star$ and length l.
A word $\mathcal{C} \in \Sigma^l$ is called consensus of $\mathcal{A}$, if

$$\mathcal{C}_j = \mathrm{argmax}_{a \in \Sigma} |\{S^{(i)}_j = a \mid 1 \le i \le k\}| \text{ for all } 1 \le j \le l$$

holds. The distance $dist(\mathcal{C}, \mathcal{A})$ from $\mathcal{A}$ to $\mathcal{C}$ is defined by

$$dist(\mathcal{C}, \mathcal{A}) := \sum_{1 \le j \le l} |\{S^{(i)}_j \mid 1 \le i \le k \wedge S^{(i)}_j \ne \mathcal{C}_j\}|.$$

Note 1: This definition implicitly uses edit distance. A symbol in column j contributes $\delta(S^{(i)}_j, \mathcal{C}_j) = 1$, if the symbols differ, 0 if they are the same.

Consensus: Consists of symbols $\mathcal{C}_j$, minimising $\sum_{1 \le i \le k} \delta(\mathcal{C}_j, S^{(i)}_j)$.

More general: Allow *arbitrary* δ.

Note 2: This way of finding a consensus is called majority voting.

Caution: Consensus is not necessarily unique!

Lemma

Let $\mathcal{A} = (S^{(1)}, \ldots, S^{(k)})$ a multiple alignment and $\mathcal{C}$ and $\bar{\mathcal{C}}$ two different consensus words for $\mathcal{A}$. Then

$$dist(\mathcal{C}, \mathcal{A}) = dist(\bar{\mathcal{C}}, \mathcal{A}).$$

Proof: Consider column j of an alignment.
Case 1: $\mathcal{C}_j \neq \bar{\mathcal{C}}_j$:
$\Rightarrow$ Symbols $\mathcal{C}_j$ and $\bar{\mathcal{C}}_j$ must occur with the same frequency in the considered column.
$\Rightarrow$ Number of differing symbols is the same for both.
$\Rightarrow$ Same contribution of this column to implicit distance for both consensus strings.

Case 2: $\mathcal{C}_j = \bar{\mathcal{C}}_j$: $\surd$

Example: Given the following alignment of words
$S_1 = AAUGCU$, $S_2 = UCC$ and $S_3 = AUUC$:

$$\begin{array}{lcllllll} S^{(1)} & = & A & A & U & G & C & U \\ S^{(2)} & = & - & - & - & U & C & C \\ S^{(3)} & = & A & - & U & U & C & - \end{array}$$

the resulting consensus is $\mathcal{C} = AAUUCU$ and thus the distance $dist(\mathcal{C}, (S^{(1)}, S^{(2)}, S^{(3)})) = 1 + 2 + 1 + 1 + 0 + 2 = 7$.

Second possibility: Derive score of multiple alignment from scores of all pairwise alignments.

Definition

Let Σ an alphabet, $- \notin \Sigma$ a gap symbol and δ a scoring function for pairwise alignments over Σ with optimization goal min. *We assume δ to be extended by a suitable value for $\delta(-,-)$. The* sum of pairs score *(short* SP-score*) δ_{SP} of a multiple alignment $\mathcal{A} = (S^{(1)}, \ldots, S^{(k)})$ of length l is defined by:*

$$\delta_{SP}(\mathcal{A}) = \sum_{1 \leq j \leq l} \delta_{SP}(S_j^{(1)}, \ldots, S_j^{(k)}),$$

where $\delta_{SP}(S_j^{(1)}, \ldots, S_j^{(k)}) := \sum_{1 \leq i \leq k} \sum_{i+1 \leq r \leq k} \delta(S_j^{(i)}, S_j^{(r)})$.

Example: We consider the multiple alignment from the previous example and assume $\delta(a,a) = 0$ and $\delta(a,b) = 1$, $a \neq b$, for $a, b \in \{-, A, C, G, U\}$. Then

$$\delta_{SP}(S^{(1)}, S^{(2)}, S^{(3)})$$

$$= \sum_{1 \leq j \leq 6} \sum_{1 \leq i \leq 3} \sum_{i+1 \leq r \leq 3} \delta(S_j^{(i)}, S_j^{(r)})$$

$$= \sum_{1 \leq j \leq 6} \left(\delta(S_j^{(1)}, S_j^{(2)}) + \delta(S_j^{(1)}, S_j^{(3)}) + \delta(S_j^{(2)}, S_j^{(3)}) \right)$$

$$= (1+0+1) + (1+1+0) + (1+0+1)$$

$$+(1+1+0) + (0+0+0) + (1+1+1) = 11.$$

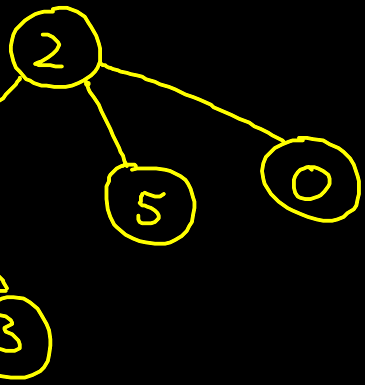
Notes:

1. SP-scoring is a special case of the more general *graph alignment*. There to align k words a graph $G = (V, E)$ with $V = \{1, \dots, k\}$ and a scoring function $\delta : (\Sigma')^2 \to \mathbb{Q}$ are given. A multiple alignment $\mathcal{A} = (S^{(1)}, \dots, S^{(k)})$ is then scored by

$$\sum_{\{r,s\} \in E} \delta(S^{(r)}, S^{(s)}).$$

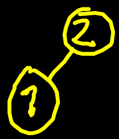
The goal is to minimize this score. The SP-scoring considers the complete graph. Another important special case is *tree alignment* where G is required to be a tree (genealogical tree in the application).

Alignment-Baum T

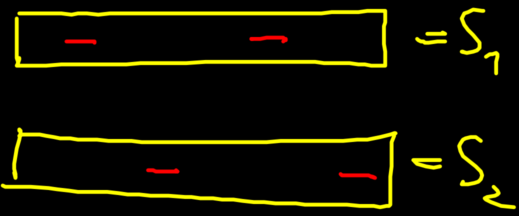


Schritt 0 (Init):

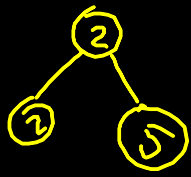
Loop: + Kante

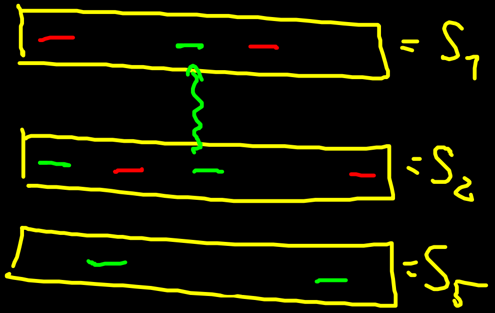


Opt. paarweises $\longrightarrow$ Alignment



+ Kante





2. It is obvious how the notion of similarity (notation *sim*, minimal or maximal δ) can be extended to multiple alignments.

3. If A_i and B_i are words, $1 \leq i \leq k$, and if A_i results from B_i by inserting gaps (into B_i), for each (minimising) scoring δ

$$sim(A_i, \ldots, A_k) \geq sim(B_1, \ldots, B_k)$$

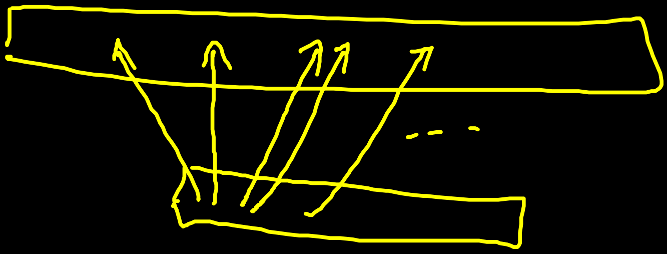
holds. The reason is simple: Each alignment for $A_1, \ldots, A_k$ also is an alignment for $B_1, \ldots, B_k$.

Exact computation

How hard is the problem of multiple alignments?
Consider decision version with SP scoring: Given a scoring function δ and words $S_1, \ldots, S_k$, is there a multiple alignment with SP score at most a given natural number d?

Definition

Let $k, m \in \mathbb{N}$ and $\mathcal{S} = \{S_1, \ldots, S_k\}$ a set of words over $\Sigma = \{0, 1\}$. The Dec-(0,1)-Shortest-Superseq-Problem *is the problem of deciding algorithmically if there is a word $T \in \{0, 1\}^\star$ of length at most m containing all words from $\mathcal{S}$ as subsequence.*



In 1994 Middendorf proved this problem to be NP complete. We leave out the proof but use this result to derive the complexity of the decision version of the multiple alignment with SP scoring.

Lemma

The decision version of the multiple alignment with SP scoring is NP complete.

Proof: See exercises.

Small number of words: For $S(1), \ldots, S(k)$ the words to be aligned, define

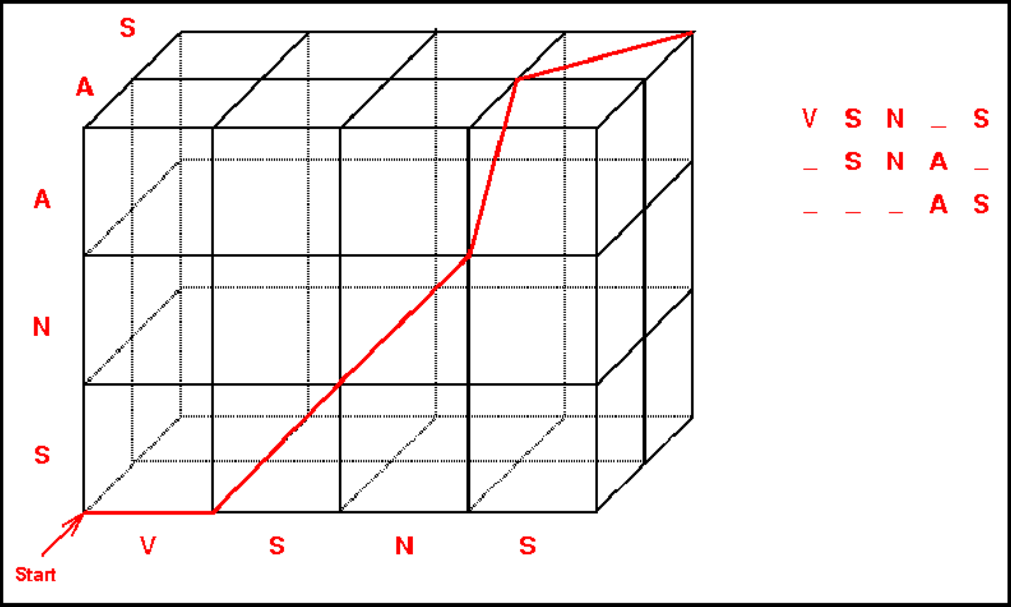
$$M_{i_1,\ldots,i_k} = \text{the minimal score of a multiple alignment of prefixes } S(1)_{1,i_1}, \ldots, S(k)_{1,i_k}.$$

We set $M_{0,\ldots,0} = 0$ and find the recursion

$$M_{i_1,\ldots,i_k} = \min_{b \in \{0,1\}^k, b \neq \vec{0}} M_{i_1-b_1,\ldots,i_k-b_k} + \delta(b_1 \cdot S(1)_{i_1}, \ldots, b_k \cdot S(k)_{i_k}),$$

where $0 \cdot a = -$ and $1 \cdot a = a$ for each character $a \in \Sigma$, i.e. vectors b are an encoding of the current column of the alignment.

Finding the minimum of $2^k - 1$ values corresponds to comparing all feasible placements of gaps in the current column.
Additionally we have to take into account time $\mathcal{O}(D)$ needed to compute the distance of a column. In case of SP scoring this time is quadratic in k.
If $S(i)$ has length n_i, $1 \leq i \leq k$, there are $\prod_{1 \leq i \leq k} n_i$ many entries in the matrix leading to a total running time in $\mathcal{O}(D \cdot 2^k \cdot \prod_{1 \leq i \leq k} n_i)$.



Graph: k-dimensional grid $G = (V, E)$ satisfying

$$V = \{0, \ldots, n_1\} \times \cdots \times \{0, \ldots, n_k\},$$

$$E = \{(p, q) \mid p, q \in V \wedge (\exists b \in \{0,1\}^k \setminus \{\vec{0}\}) : (q = p + b)\}.$$

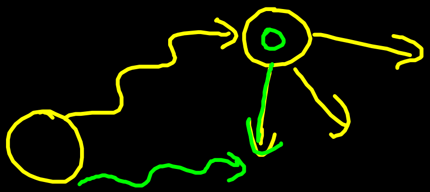
Edge weights $g(p, q)$, corresponding to extending the alignment to a new column. Above algorithm then solves the problem of shortest paths with possibly negative edge weights for this graph.

This problem is generally NP complete but eased here by cycle freeness.

Parenthesis: Dijkstra's algorithm (shortest paths)

Given: cost matrix with entry $c_{i,j} = c$, if edge from node i to node j has label (cost) c. If there is no such edge, $c_{i,j} = \infty$.
WLOG: $V = \{1, 2, \ldots, n\}$, we start with node 1 and set $S := \{1\}$.

Algorithm: Choose $i \in V \setminus S$ such that $1 \rightsquigarrow i$ has minimal cost among all paths $1 \rightsquigarrow j$, $j \in V \setminus S$.
Let $S := S \cup \{i\}$ and for all nodes $j \in V \setminus S$ compare the cheapest known path $1 \rightsquigarrow j$ to $1 \rightsquigarrow i \rightarrow j$ and update if necessary.
Repeat this until $S = V$.

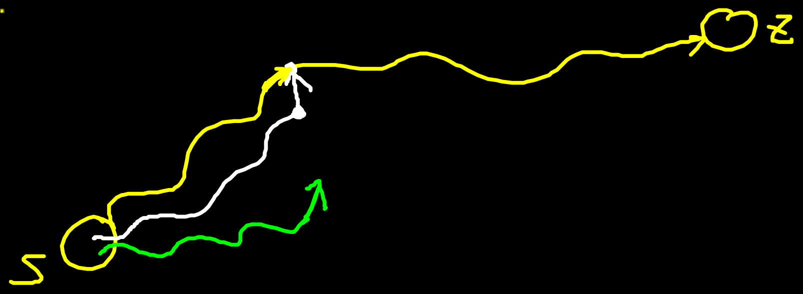


Branch & Bound: Starting node $s = (0, 0, \ldots, 0)$.
General step: Node set $W \subset V$ already explored. Nodes from this set (except *excluded* nodes) are stored in a priority queue. Priority of node v is the current estimation of a shortest path from s to v.

Estimation: Simulation of Dijkstra's algorithm.
In current step let U be stored in PQ.
$\Rightarrow$ Dijkstra's algorithm chooses node $u = (i_1, \ldots, i_k)$ with lowest priority. We set $\nu(i) := S(i)_{1,i_k}$, $1 \leq i \leq k$, and find the priority of u to match the length of a shortest path from s to u and thus $sim(\nu(1), \ldots, \nu(k))$.
After choosing u all direct successors w of u are explored and added to U with priority $sim(\nu(1), \ldots, \nu(k)) + g(u, w)$.
w already in PQ $\Rightarrow$ Update priority.

Branching: Add neighbored nodes.

Bounding: Try to exclude neighbors:
There's a necessity to add neighbors w of u only if an optimal alignment consists of an alignment for $\nu(1), \ldots \nu(k)$ followed by the column specified by $u \rightarrow w$ and an optimal alignment of the rest.
Even if w corresponds to an optimal alignment we only have to add w at the time where u corresponds to a part of the optimal alignment.



Using our coding $w = u + b = (j_1, \ldots, j_k)$ for $b \in \{0,1\}^k \setminus \{\vec{0}\}$.
An optimal ending corresponds to an optimal alignment of words $\mu(i) := S(i)_{j_i+1, n_i}$, $1 \leq i \leq k$. From the definition of SP scoring we get

$$sim(\mu(1), \ldots, \mu(k)) \geq \sum_{1 \leq i < j \leq k} sim(\mu(i), \mu(j)), \qquad (2)$$

as two rows of a multiple alignment also represent a pairwise alignment.
$\Rightarrow$ Lower bound for the rating of an optimal ending.
If we already know an alignment with SP score $\underline{\mathcal{D}}$ and if

$$sim(\nu(1), \ldots, \nu(k)) + g(u, w) + \sum_{1 \leq i < j \leq k} sim(\mu(i), \mu(j)) \underline{> \mathcal{D}}$$

holds, we can exclude w for the time being without knowing the optimal ending.

Computing boundary $\mathcal{D}$: Find good alignment for the similarity of words $S(1), \ldots, S(k)$.
Better bounding method: Abstract from SP scores and use the fact that *we have to* find a shortest path from node $s = (0, \ldots, 0)$ to node $t = (n_1, \ldots, n_k)$ in a graph with non-negative edge weights.
As above let $g(e)$ denote the weight of edge e and assume knowledge of a lower bound $\xi(u)$ for the length of a shortest path from u to t. We choose

$$g^*(u, w) = g(u, w) + \xi(w) - \xi(u)$$

as new edge weights.

Lemma

If $g(u, w) + \xi(w) \geq \xi(u)$ holds, the new edge weights g^* are non-negative and a shortest path for the new weights is also a shortest path for the old weights.

Proof: See exercises.

Notes:

- Constraint is fulfilled with SP scoring and the lower bound from (2).
- Lower bound $\xi(u)$ exact (Obviously this assumption is unrealistic, but the consideration shows that we may expect a fast run time if the bounds are tight.)
  $\Rightarrow$ edges of a shortest path for g^* have weight 0.
  $\Rightarrow$ Priority queue only emits nodes with weight 0 and only nodes on a shortest path are expanded.
- The reweighting and the lower bounds from (2) along with the branch&bound method described have proven in practise.

Approximative Computation

Center-Star Method: Idea: Construct a good multiple alignment from a set of pairwise alignments.

Definition

Let $\mathcal{S} = \{S_1, \dots, S_k\}$ a set of words and $\mathcal{T} \subseteq \mathcal{S}$. Furthermore δ a scoring for alignments satisfying $\delta(-,-) = 0$. Let $\mathcal{A}$ a multiple alignment of $\mathcal{S}$ and $\mathcal{A}' = (S^{(i_1)}, \dots, S^{(i_m)})$ one of $\mathcal{T}$. $\mathcal{A}$ is called compatible to $\mathcal{A}'$, if the restriction of $\mathcal{A}$ to lines $i_1, \dots, i_m$ has the same score as $\mathcal{A}'$.

Example: Let $\mathcal{S} = \{AUG, ACGG, AUCGG\}$, $\mathcal{T}_1 = \{AUG, ACGG\}$, $\mathcal{T}_2 = \{AUCGG, AUG\}$ and let $\delta(a,a) = 0$, $\delta(a,b) = 1$ for $a \neq b$, $a, b \in \{-, A, C, G, U\}$. Then the alignment

$$\begin{array}{ccccc} A & - & - & U & G \\ A & - & C & G & G \\ A & U & C & G & G \end{array}$$

of $\mathcal{S}$ is compatible to the alignment

$$\begin{array}{cccc} A & - & U & G \\ A & C & G & G \end{array}$$

of $\mathcal{T}_1$ but not compatible to the alignment

$$\begin{array}{ccccc} A & U & C & G & G \\ A & U & - & G & - \end{array}$$

of $\mathcal{T}_2$.

Definition

Let $\mathcal{S} = \{S_1, \ldots, S_k\}$ a set of words over Σ. A tree $T = (V, E)$, $V = \{S_1, \ldots, S_k\}$, where each edge $\{S_i, S_j\} \in E$ is labeled with the rating of an optimal pairwise alignments $(S^{(i)}, S^{(j)})$ for S_i and S_j, is called alignment tree for $\mathcal{S}$.

One can show that it is always possible to construct a compatible multiple alignment from an alignment tree.

Theorem

Let $\mathcal{S} = \{S_1, \ldots, S_k\}$ a set of words over Σ, $T = (V, E)$ an alignment tree for $\mathcal{S}$ and δ a scoring function satisfying $\delta(-,-) = 0$. Then a multiple alignment $(S^{(1)}, \ldots, S^{(k)})$ for $\mathcal{S}$ compatible with all optimal alignments corresponding to edges $e = \{S_i, S_j\} \in E$ can be efficiently determined.

Proof: We compute an alignment compatible with T using the following algorithm:
Input: A set of words $\mathcal{S} = \{S_1, \ldots, S_k\}$ over Σ, a scoring $\delta : (\Sigma \cup \{-\})^2 \to \mathbb{Q}$ satisfying $\delta(-,-) = 0$ and an alignment tree T for $\mathcal{S}$.

Step 1: Let $V_1 := \{1\}$ and T_1 the subtree of T consisting only of node 1. Alignment $\mathcal{A}^{(1)}$ then is equal to S_1.

Step 2: For i from 1 to $k-1$ repeat the following steps:
(a) Choose any node $s \notin V_i$, neighbored to a node $r \in V_i$ in T. Let $V_{i+1} := V_i \cup \{s\}$ and add s and edge $\underbrace{\{r, s\}}$ to T_i. $\Rightarrow T_{i+1}$.
(b) Compute optimal alignment $\mathcal{A}$ of S_s and row $\underbrace{\mathcal{A}_r^{(i)}}$ of alignment $\mathcal{A}^{(i)}$ corresponding to S_r. As $\delta(-,-) = 0$ holds, the score of an optimal alignment of $\underbrace{S_s \text{ and } \bigcirc\!\!\!\!\!S_r}$ is equal to the score of an optimal alignment for S_s and $\mathcal{A}_r^{(i)}$.
(c) For each gap added to $\mathcal{A}_r^{(i)}$ by $\mathcal{A}$ add a *gap column* to $\mathcal{A}^{(i)}$. Finally add the row of $\mathcal{A}$ corresponding to S_s to the modified $\mathcal{A}^{(i)}$ creating $\mathcal{A}^{(i+1)}$.

Step 3: Output alignment $\mathcal{A}^{(k)}$.

Observations:

- The algorithm obviously guarantees consistency for the new edge $\{r, s\}$ and adding further edges doesn't change consistency of previously added edges as only gap columns are added.
- As a tree with k nodes always has $k - 1$ edges $\Rightarrow$ the loop in step 2 will process each edge of the alignment tree eventually. $\square$