Fast Algorithms for Computing Tree LCS

Shay Mozes[1] Dekel Tsur[2]
Oren Weimann[3] Michal Ziv-Ukelson[2]

[1]Brown University

[2]Ben-Gurion University

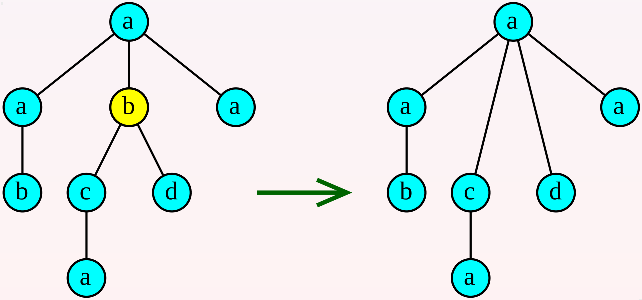
[3]Massachusetts Institute of Technology

Definition of tree LCS

Definition

The LCS of two trees is the size of largest forest that can be obtained from both trees by deleting nodes.

Deletion of a node v in a forest F means that the children of v become children of the parent of v (if it exists).

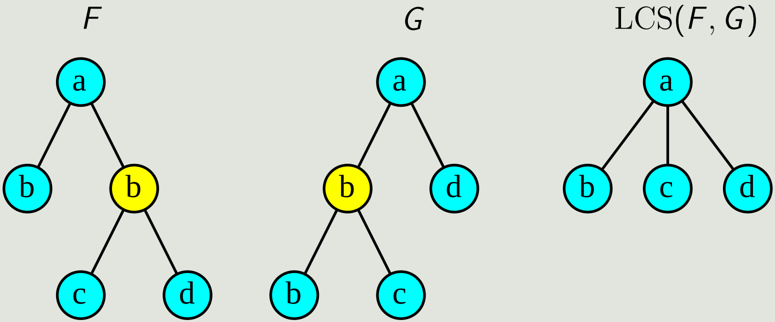


Definition of tree LCS

Definition

The LCS of two trees is the size of largest forest that can be obtained from both trees by deleting nodes.

Example



Known & new results

Let F and G be the input trees, with sizes $n \geq m$.
Known algorithms for tree edit distance/tree LCS:

- $O(nm \cdot \mathrm{leaves}(F)^2 \cdot \mathrm{leaves}(G)^2) = O(n^6)$ (Tai 79)
- $O(nm \cdot \mathrm{cdepth}(F) \cdot \mathrm{cdepth}(G)) = O(n^4)$ (Zhang & Shasha 89)
- $O(m^2 n \log n) = O(n^3 \log n)$ (Klein 98)
- $O(nm^2(1 + \log \frac{n}{m})) = O(n^3)$ (Demaine et al. 07)

Known & new results

Let F and G be the input trees, with sizes $n \geq m$.
Known algorithms for tree edit distance/tree LCS:

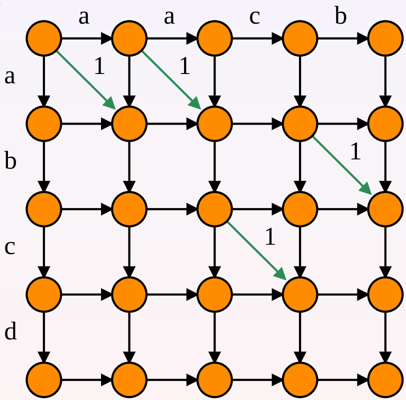
- $O(nm \cdot \text{leaves}(F)^2 \cdot \text{leaves}(G)^2) = O(n^6)$ (Tai 79)
- $O(nm \cdot \text{cdepth}(F) \cdot \text{cdepth}(G)) = O(n^4)$ (Zhang & Shasha 89)
- $O(m^2 n \log n) = O(n^3 \log n)$ (Klein 98)
- $O(nm^2(1 + \log \frac{n}{m})) = O(n^3)$ (Demaine et al. 07)

New algorithms for tree LCS:

- $O(r \cdot \text{height}(F) \cdot \text{height}(G) \cdot \log\log m)$
- $O(Lr \log r \cdot \log\log m)$

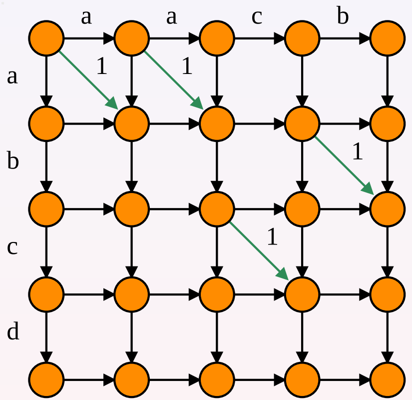
$r =$ number of match pairs (pairs of vertices $v \in F$, $w \in G$ with equal labels)
$L = \text{LCS}(F, G)$

Alignment graph for string LCS



- Let S and T be strings of lengths n and m.
- The alignment graph has vertices (i, j) for every $0 \leq i \leq n$ and $0 \leq j \leq m$. Vertex (i, j) corresponds to the pair of prefixes $S[1..i]$, $T[1..j]$.

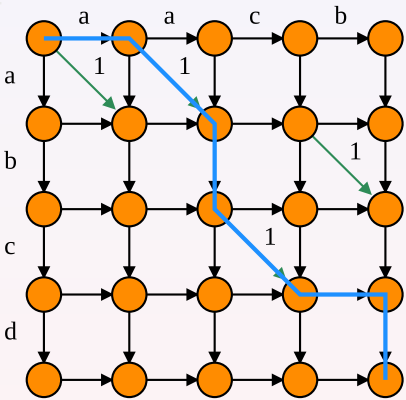
Alignment graph for string LCS



The edges are

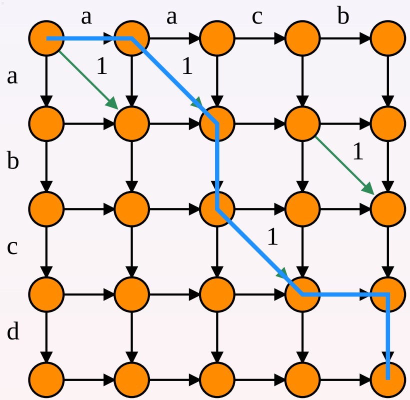
- Vertical and hotizontal edges of weight 0 corresponding to deletion of character from S or T.
- Diagonal edges for every match pair of weight 1.

Alignment graph for string LCS



- A path of weight k from $(0,0)$ to (n,m) corresponds to a common substring of size k.
- A common substring of size k corresponds to a path of weight k from $(0,0)$ to (n,m).

Alignment graph for string LCS

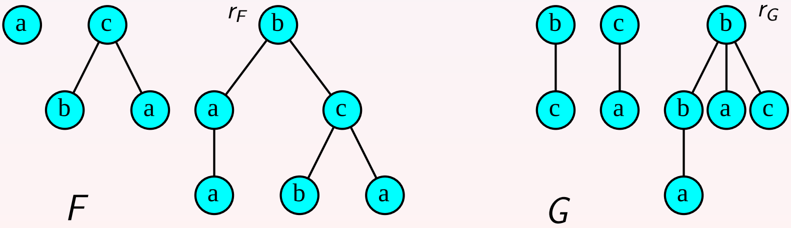


- The number of nonzero weight edges is r.
- The maximum weight path can be found in time $O(r \log \log m)$ (Hunt-Szymanski 77).

The algorithm of Zhang & Shasha

R_F = rightmost tree in F. r_G = root of R_G.

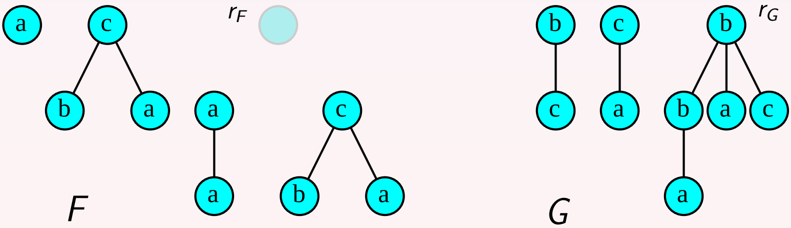
$$\mathrm{LCS}(F,G) = \max \begin{cases} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G), \\ \mathrm{LCS}(R_F - r_F, R_G - r_G) \\ \quad + \mathrm{LCS}(F - R_F, G - R_G) + 1 \\ \text{if } \mathrm{label}(r_F) = \mathrm{label}(r_G) \end{cases}$$



The algorithm of Zhang & Shasha

R_F = rightmost tree in F. r_G = root of R_G.

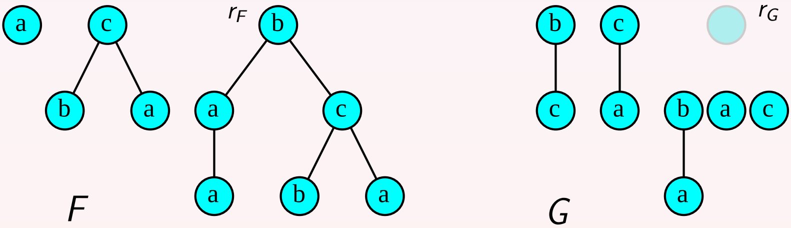
$$\mathrm{LCS}(F,G) = \max \left\{ \begin{array}{l} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G), \\ \mathrm{LCS}(R_F - r_F, R_G - r_G) \\ \quad + \mathrm{LCS}(F - R_F, G - R_G) + 1 \\ \quad \text{if } \mathrm{label}(r_F) = \mathrm{label}(r_G) \end{array} \right\}$$



The algorithm of Zhang & Shasha

R_F = rightmost tree in F. r_G = root of R_G.

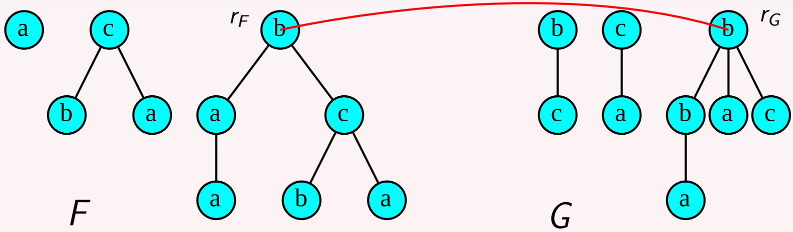
$$\mathrm{LCS}(F,G) = \max \begin{cases} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G), \\ \mathrm{LCS}(R_F - r_F, R_G - r_G) \\ \quad + \mathrm{LCS}(F - R_F, G - R_G) + 1 \\ \text{if } \mathrm{label}(r_F) = \mathrm{label}(r_G) \end{cases}$$



The algorithm of Zhang & Shasha

R_F = rightmost tree in F. r_G = root of R_G.

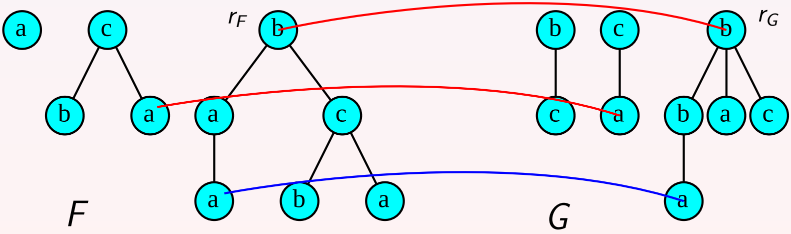
$$\mathrm{LCS}(F,G) = \max \begin{cases} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G), \\ \mathrm{LCS}(R_F - r_F, R_G - r_G) \\ \quad + \mathrm{LCS}(F - R_F, G - R_G) + 1 \\ \quad \text{if } \mathrm{label}(r_F) = \mathrm{label}(r_G) \end{cases}$$



The algorithm of Zhang & Shasha

R_F = rightmost tree in F. r_G = root of R_G.

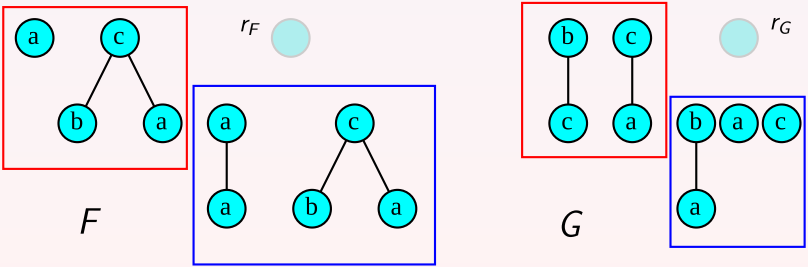
$$\mathrm{LCS}(F,G) = \max \begin{Bmatrix} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G), \\ \mathrm{LCS}(R_F - r_F, R_G - r_G) \\ \quad + \mathrm{LCS}(F - R_F, G - R_G) + 1 \\ \quad \text{if } \mathrm{label}(r_F) = \mathrm{label}(r_G) \end{Bmatrix}$$



The algorithm of Zhang & Shasha

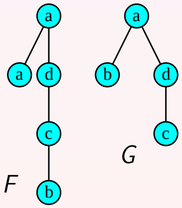
R_F = rightmost tree in F. r_G = root of R_G.

$$\mathrm{LCS}(F,G) = \max \begin{cases} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G), \\ \mathrm{LCS}(R_F - r_F, R_G - r_G) \\ \quad + \mathrm{LCS}(F - R_F, G - R_G) + 1 \\ \quad \text{if } \mathrm{label}(r_F) = \mathrm{label}(r_G) \end{cases}$$



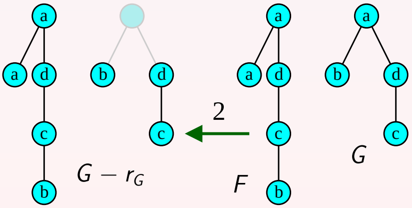
Relevant subproblems

$$\mathrm{LCS}(F, G) = \min \left\{ \begin{array}{l} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G) \\ \ldots \end{array} \right\} \begin{array}{l} 1 \\ 2 \\ \ \end{array}$$



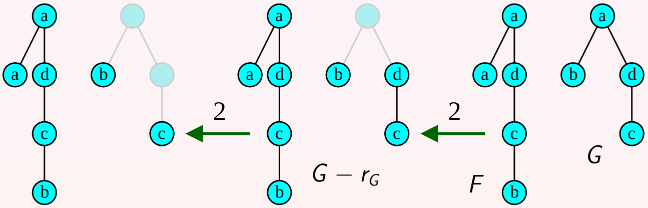
Relevant subproblems

$$\mathrm{LCS}(F, G) = \min \left\{ \begin{array}{l} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G) \\ \ldots \end{array} \right\} \begin{array}{l} 1 \\ 2 \\ \end{array}$$



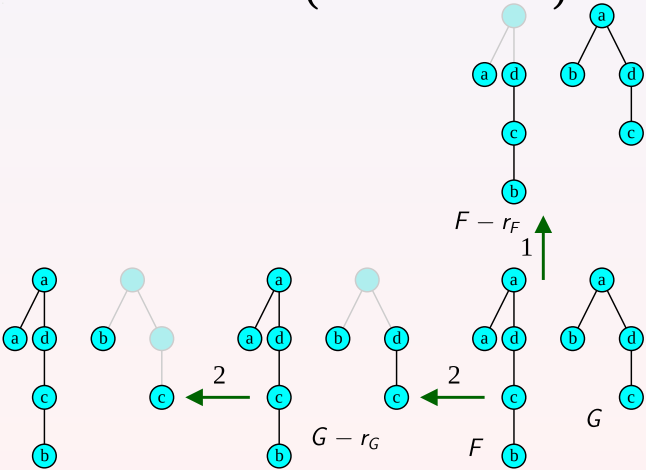
Relevant subproblems

$$\mathrm{LCS}(F, G) = \min \left\{ \begin{array}{l} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G) \\ \dots \end{array} \right\} \begin{array}{l} 1 \\ 2 \\ \end{array}$$



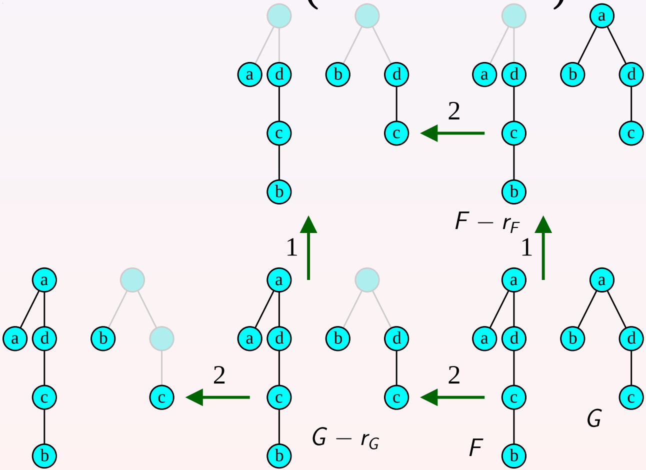
Relevant subproblems

$$\mathrm{LCS}(F, G) = \min \left\{ \begin{array}{l} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G) \\ \ldots \end{array} \right\} \begin{array}{l} 1 \\ 2 \\ \end{array}$$



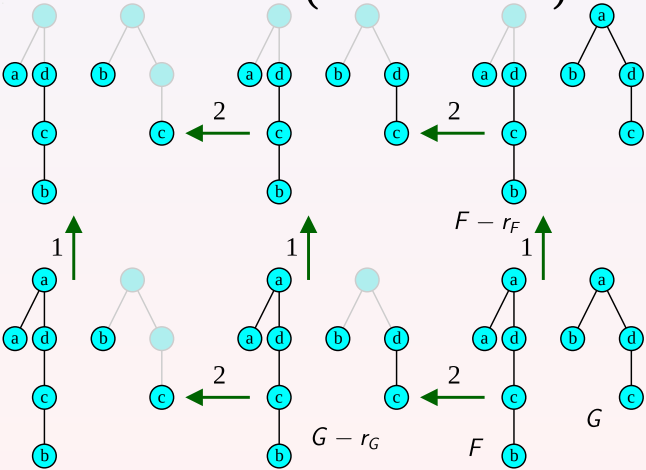
Relevant subproblems

$$\mathrm{LCS}(F, G) = \min \left\{ \begin{array}{l} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G) \\ \ldots \end{array} \right\} \begin{array}{l} 1 \\ 2 \\ \end{array}$$



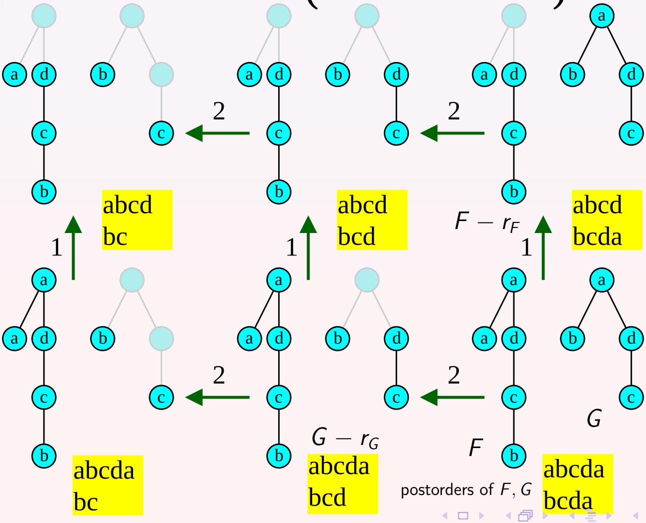
Relevant subproblems

$$\mathrm{LCS}(F, G) = \min \left\{ \begin{array}{l} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G) \\ \dots \end{array} \right\} \begin{array}{l} 1 \\ 2 \\ \\ \end{array}$$

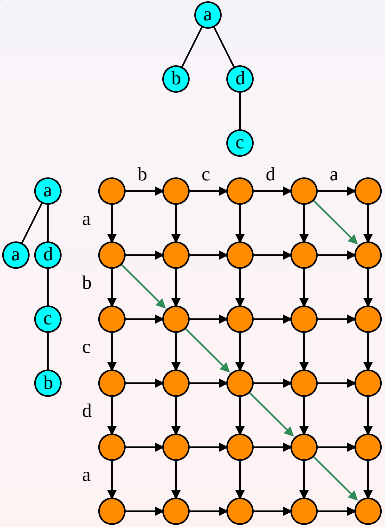


Relevant subproblems

$$\mathrm{LCS}(F, G) = \min \begin{cases} \mathrm{LCS}(F - r_F, G), & 1 \\ \mathrm{LCS}(F, G - r_G) & 2 \\ \dots \end{cases}$$



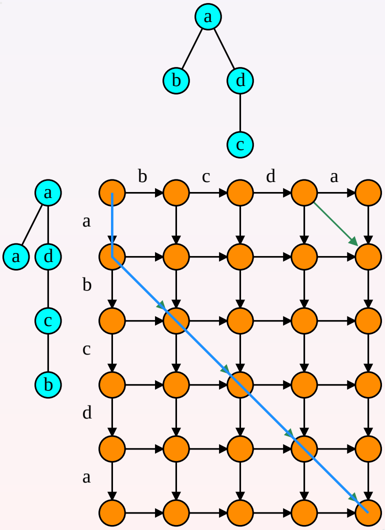
Alignment graph (Backofen et al., Touzet)



Consider the alignment graph for the two postorder strings of F and G.

A path of weight k does not correspond to a common subforest of size k.

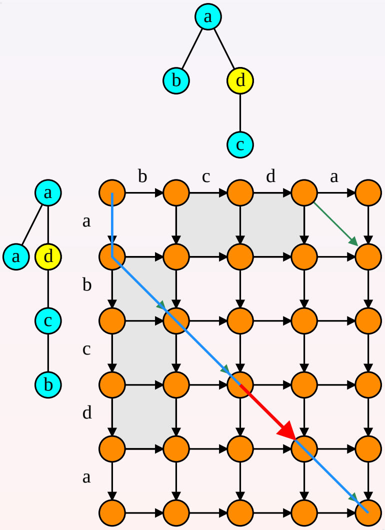
Alignment graph (Backofen et al., Touzet)



Consider the alignment graph for the two postorder strings of F and G.

A path of weight k does not correspond to a common subforest of size k.

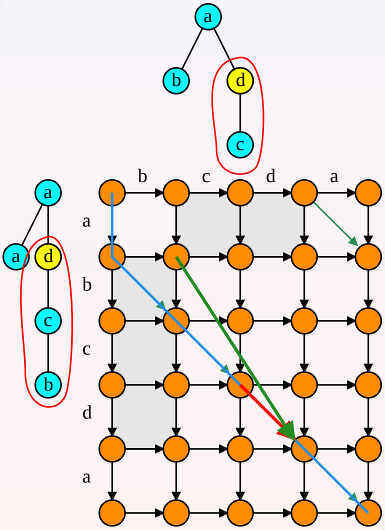
Alignment graph (Backofen et al., Touzet)



Consider the alignment graph for the two postorder strings of F and G.

A path of weight k does not correspond to a common subforest of size k.

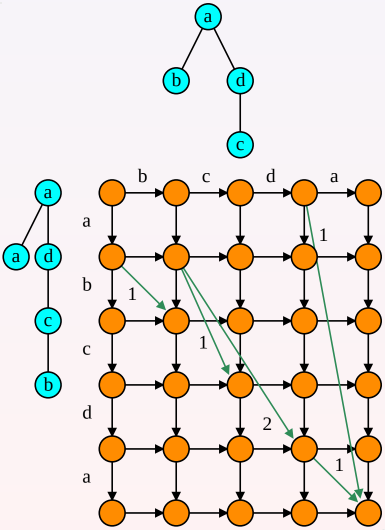
Alignment graph (Backofen et al., Touzet)



Consider the alignment graph for the two postorder strings of F and G.

A path of weight k does not correspond to a common subforest of size k.

Alignment graph (Backofen et al., Touzet)



For each match pair v, w (with $v \in F$, $w \in G$) add an edge

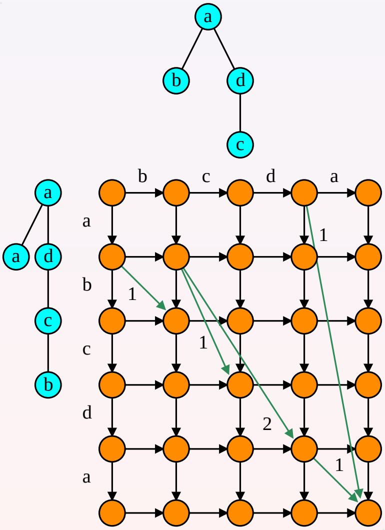
$$(i - |F_v|, j - |G_w|) \rightarrow (i, j)$$

of weight $\mathrm{LCS}(F_v, G_w)$, where i and j are the ranks of v and w in the postorders of F and G.

For the match pair of the roots we just add an edge $(n-1, m-1) \rightarrow (n, m)$ of weight 1.

The LCS values are computed recursively.

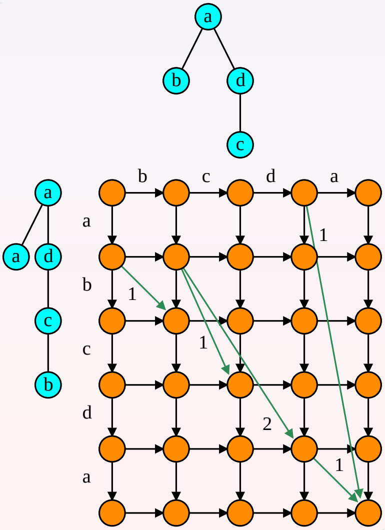
Alignment graph (Backofen et al., Touzet)



Every path of weight k from $(0,0)$ to (n,m) corresponds to a common subforest of size k.

Every largest common subforest corresponds to a path of weight $\mathrm{LCS}(F,G)$.

Alignment graph (Backofen et al., Touzet)



There are exactly r edges with nonzero weight.

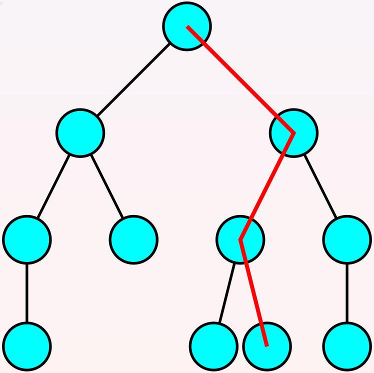
Computing the maximum weight path in the graph takes $O(r \log \log m)$ time.

The time including the recursive calls is $O(r \cdot \text{height}(F) \cdot \text{height}(G) \cdot \log \log m)$

Heavy path

Definition

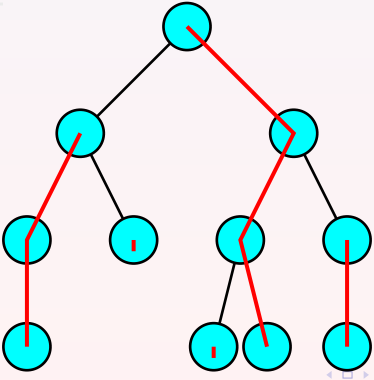
A heavy path of a tree is a path that starts at the root and goes from each vertex v to the child of v whose subtree contains more vertices.



Heavy path decomposition

Definition

A heavy path decomposition of a tree F is a collection of paths obtained by taking a heavy path of F, removing its vertices, and then building a heavy path decomposition for every remaining tree.



The algorithm of Klein

R_S = rightmost tree in S. r_S = root of R_S.

$$\mathrm{LCS}(F,G) = \max \begin{Bmatrix} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G), \\ \mathrm{LCS}(R_F - r_F, R_G - r_G) \\ \quad + \mathrm{LCS}(F - R_F, G - R_G) + 1 \\ \quad \text{if } \mathrm{label}(r_F) = \mathrm{label}(r_G) \end{Bmatrix}$$

The algorithm of Klein

R_S = rightmost tree in S. r_S = root of R_S.

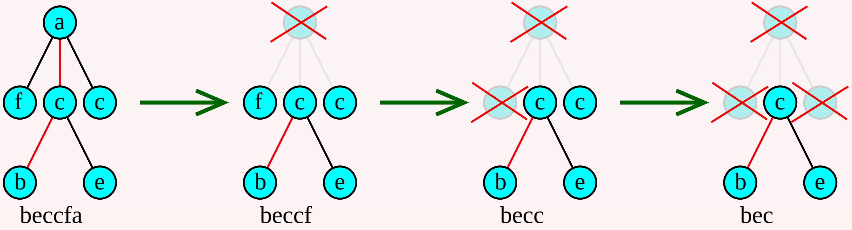
$$\mathrm{LCS}(F,G) = \max \begin{cases} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G), \\ \mathrm{LCS}(R_F - r_F, R_G - r_G) \\ \quad + \mathrm{LCS}(F - R_F, G - R_G) + 1 \\ \quad \text{if } \mathrm{label}(r_F) = \mathrm{label}(r_G) \end{cases}$$

L_S = leftmost tree in S. l_S = root of L_S.

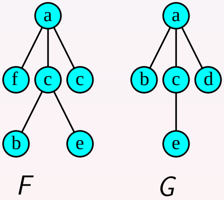
$$\mathrm{LCS}(F,G) = \max \begin{cases} \mathrm{LCS}(F - l_F, G), \\ \mathrm{LCS}(F, G - l_G), \\ \mathrm{LCS}(L_F - l_F, L_G - l_G) \\ \quad + \mathrm{LCS}(F - L_F, G - L_G) + 1 \\ \quad \text{if } \mathrm{label}(l_F) = \mathrm{label}(l_G) \end{cases}$$

The algorithm of Klein

- Let F and G be the input trees, with $|F| \geq |G|$.
- Build a heavy path decomposition of F.
- Recursively compute $\mathrm{LCS}(F, G)$ using left & right rules.
- To compute $\mathrm{LCS}(F', G')$ for some F' and G', let p be the highest path in decomposition that contain vertices of F'. Use the left rule if l_F is not on p. O/w use left rule.

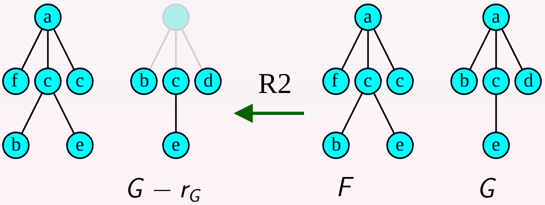


Relevant subproblems



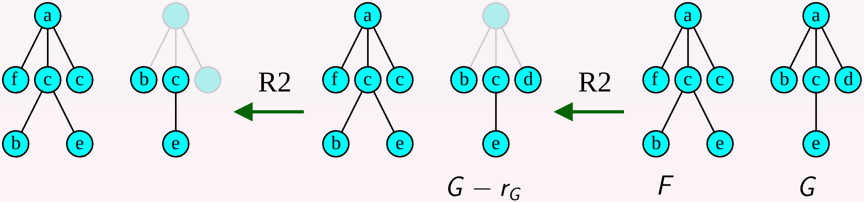
$$\mathrm{LCS}(F, G) = \min \begin{Bmatrix} \mathrm{LCS}(F - l_F, G), \\ \mathrm{LCS}(F, G - l_G) \\ \dots \end{Bmatrix} \begin{matrix} L1 \\ L2 \\ \end{matrix} \quad \mathrm{LCS}(F, G) = \min \begin{Bmatrix} \mathrm{LCS}(F - r_F, G), \\ \mathrm{LCS}(F, G - r_G) \\ \dots \end{Bmatrix} \begin{matrix} R1 \\ R2 \\ \end{matrix}$$

Relevant subproblems



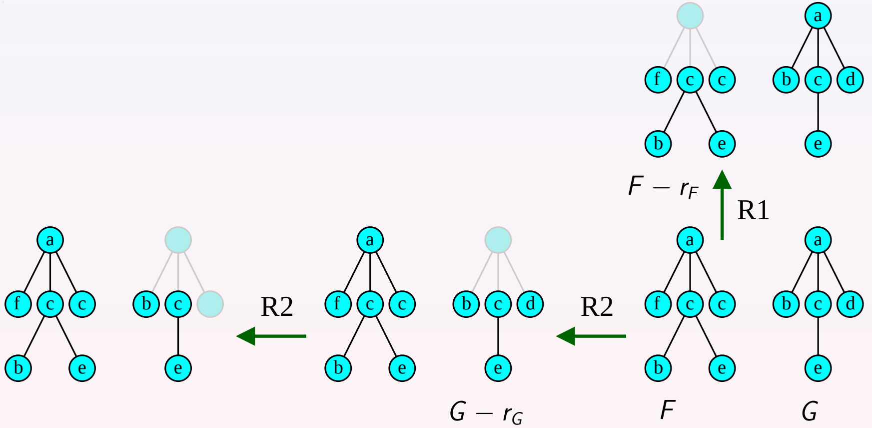
$$\mathrm{LCS}(F,G) = \min \begin{cases} \mathrm{LCS}(F - l_F, G), & L1 \\ \mathrm{LCS}(F, G - l_G) & L2 \\ \ldots \end{cases} \quad \mathrm{LCS}(F,G) = \min \begin{cases} \mathrm{LCS}(F - r_F, G), & R1 \\ \mathrm{LCS}(F, G - r_G) & R2 \\ \ldots \end{cases}$$

Relevant subproblems



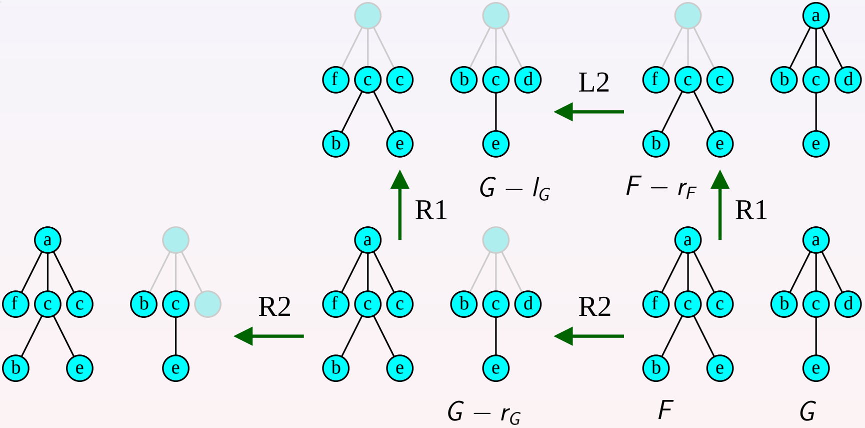
$$\mathrm{LCS}(F,G) = \min \begin{cases} \mathrm{LCS}(F - l_F, G), & L1 \\ \mathrm{LCS}(F, G - l_G) & L2 \\ \dots \end{cases} \qquad \mathrm{LCS}(F,G) = \min \begin{cases} \mathrm{LCS}(F - r_F, G), & R1 \\ \mathrm{LCS}(F, G - r_G) & R2 \\ \dots \end{cases}$$

Relevant subproblems



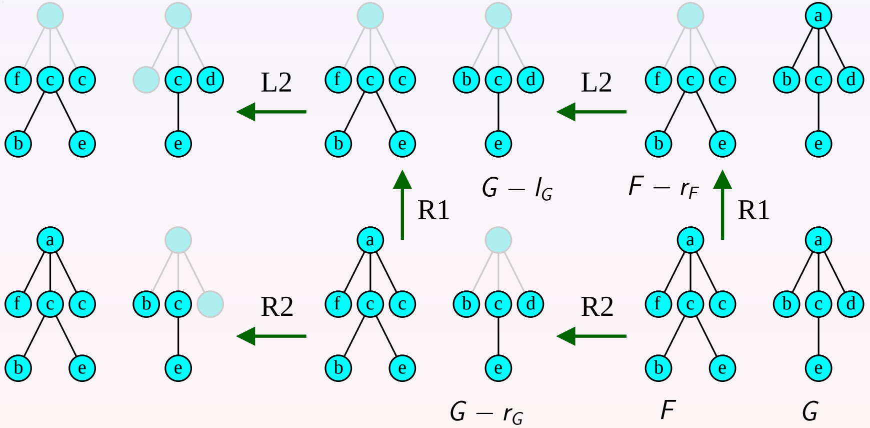
$$\mathrm{LCS}(F,G) = \min \begin{cases} \mathrm{LCS}(F - l_F, G), & L1 \\ \mathrm{LCS}(F, G - l_G) & L2 \\ \ldots \end{cases} \qquad \mathrm{LCS}(F,G) = \min \begin{cases} \mathrm{LCS}(F - r_F, G), & R1 \\ \mathrm{LCS}(F, G - r_G) & R2 \\ \ldots \end{cases}$$

Relevant subproblems



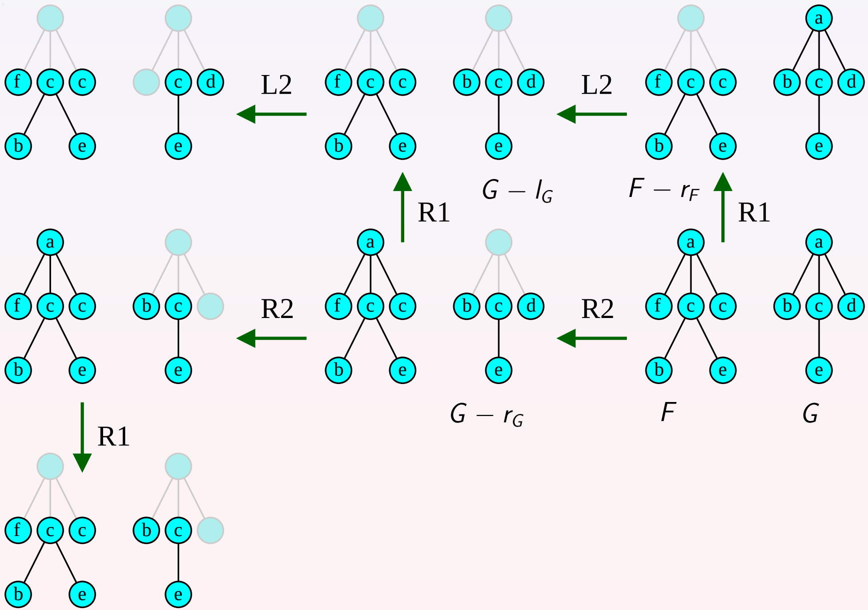
$$\mathrm{LCS}(F, G) = \min \begin{cases} \mathrm{LCS}(F - l_F, G), & L1 \\ \mathrm{LCS}(F, G - l_G) & L2 \\ \dots \end{cases} \quad \mathrm{LCS}(F, G) = \min \begin{cases} \mathrm{LCS}(F - r_F, G), & R1 \\ \mathrm{LCS}(F, G - r_G) & R2 \\ \dots \end{cases}$$

Relevant subproblems

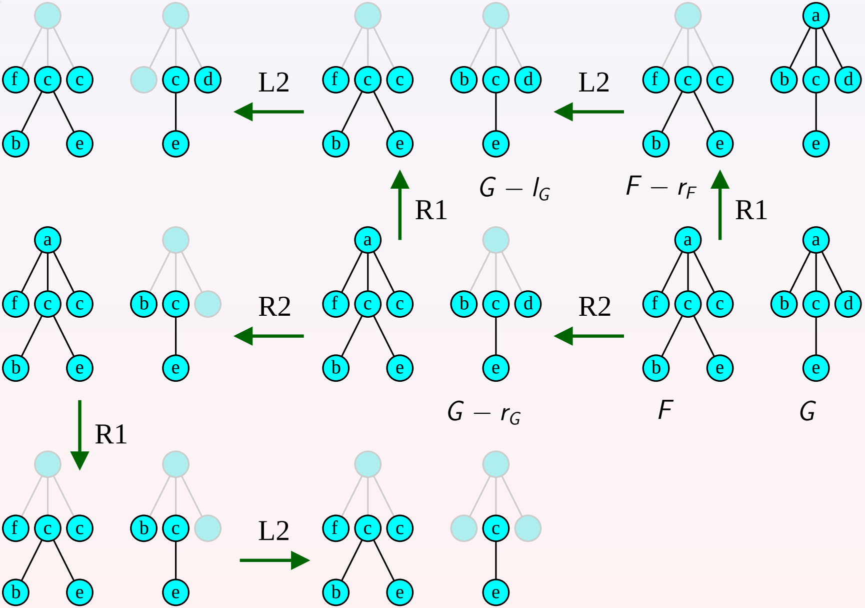


$$\mathrm{LCS}(F,G) = \min \begin{cases} \mathrm{LCS}(F - l_F, G), & L1 \\ \mathrm{LCS}(F, G - l_G) & L2 \\ \dots \end{cases} \qquad \mathrm{LCS}(F,G) = \min \begin{cases} \mathrm{LCS}(F - r_F, G), & R1 \\ \mathrm{LCS}(F, G - r_G) & R2 \\ \dots \end{cases}$$

Relevant subproblems

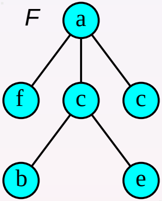


Relevant subproblems

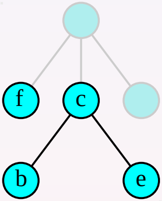


Relevant subproblems

- F represented by a string S_F according to deletion order.

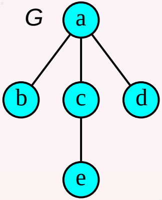


$S_F = \text{beccfa}$

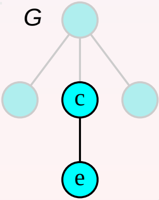


$S_F[1..4] = \text{becc}$

- G represented by its Euler string S_G.



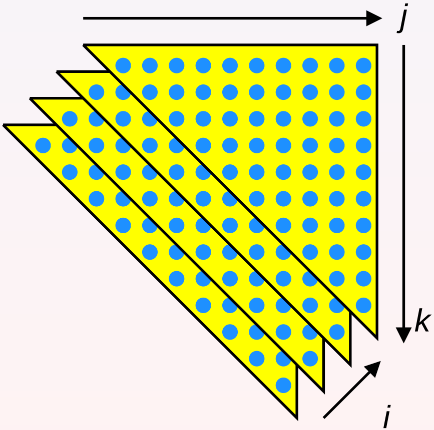
$S_G = \text{abbceecdda}$



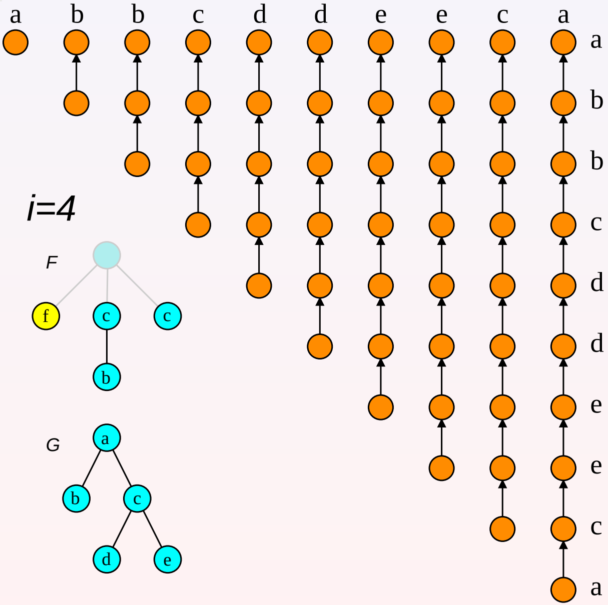
$S_G[3..7] = \text{bceec}$

Alignment graph

The alignment graph has vertex (i, j, k) for every $0 \leq i \leq n$ and $1 \leq j \leq k \leq 2m$.
Vertex (i, j, k) corresponds to the pair $S_F[1..i], S_G[j..k]$



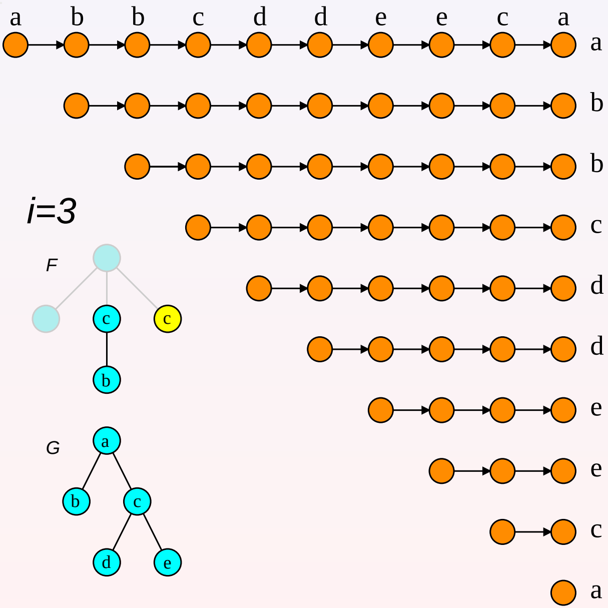
Alignment graph



Edges corresponding to deleting leftmost root of $S_G[j..k]$.

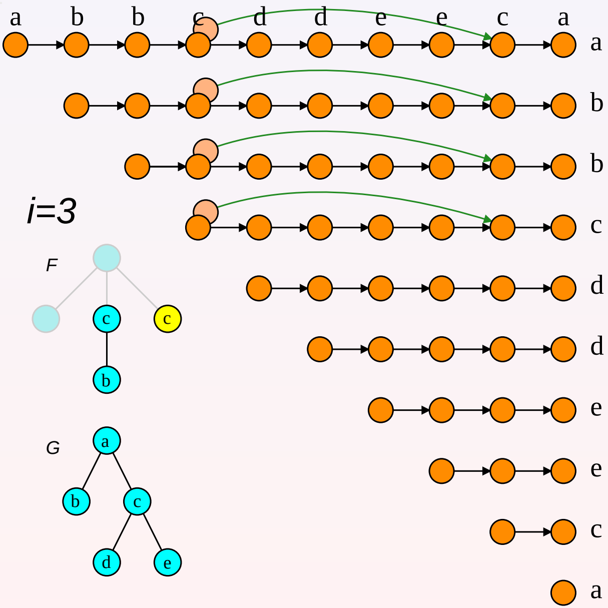
Weights: 0

Alignment graph



Edges corresponding to deleting rightmost root of $S_G[j..k]$.
Weights: 0

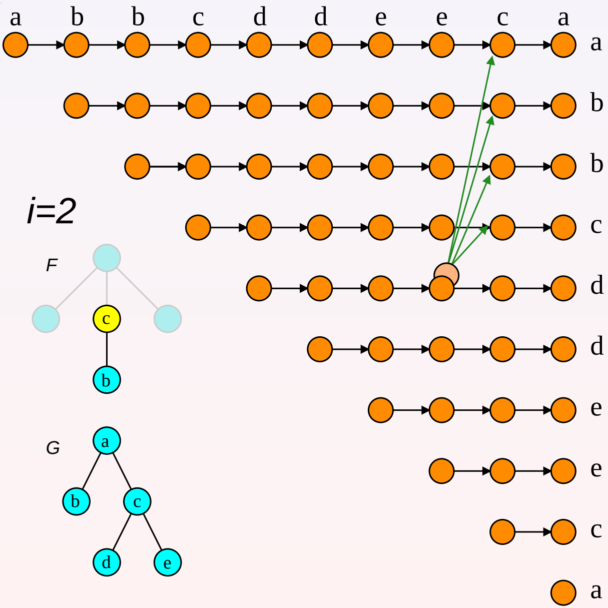
Alignment graph



Edges corresponding to matching rightmost roots of $S_F[1..i]$ and $S_G[j..k]$.

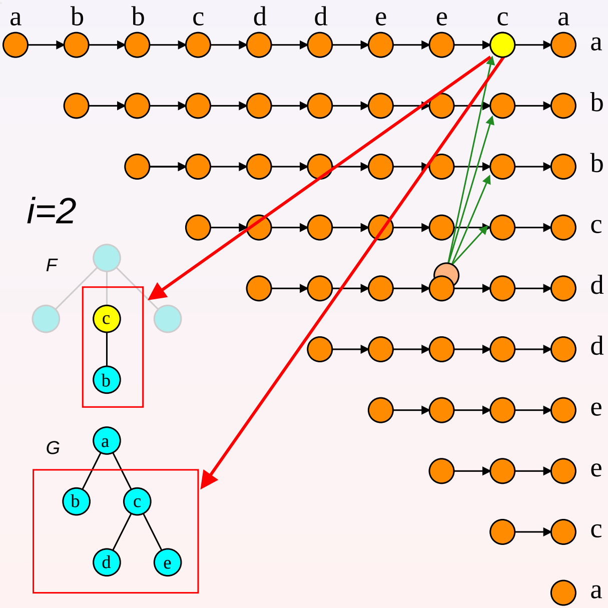
Weights: ≥ 1 (LCS)

Alignment graph



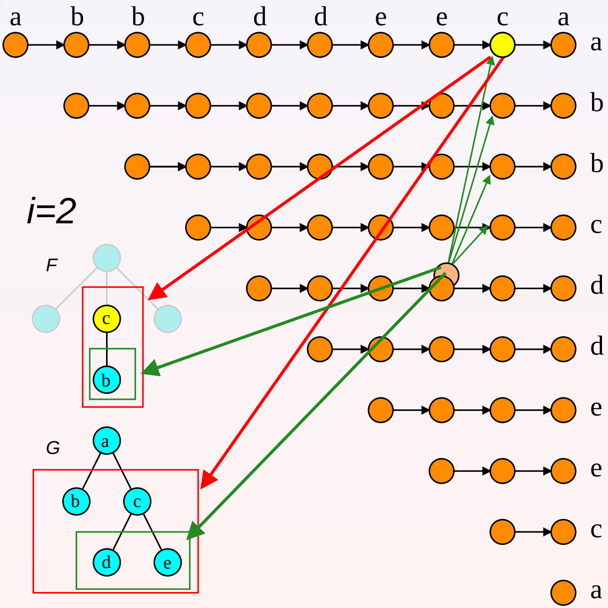
If $S_F[1..i]$ is a tree, the edges are different.
Weights: 1

Alignment graph



If $S_F[1..i]$ is a tree, the edges are different.
Weights: 1

Alignment graph



If $S_F[1..i]$ is a tree, the edges are different.
Weights: 1

Summary

Algorithms for tree LCS:

- $O(r \cdot \mathrm{height}(F) \cdot \mathrm{height}(G) \cdot \log\log m)$
- $O(Lr \log r \cdot \log\log m)$

Algorithm for homeomorphic tree LCS

- $O(r \lg\lg m \cdot (\mathrm{height}(F) + \mathrm{height}(G)))$.