Algorithms for Finding a Most Similar Subforest

Jesper Jansson
(Kyushu University, Japan)

Zeshan Peng
(The University of Hong Kong)

The problem

Input: Ordered labeled forests F and G.

Output: A subforest F' of F which is most similar to G over all possible F'.

F is called "the target forest" and G is "the pattern forest".

The problem

Input: Ordered labeled forests F and G.

Output: A subforest F' of F which is most similar to G over all possible F'.

F is called "the target forest" and G is "the pattern forest".

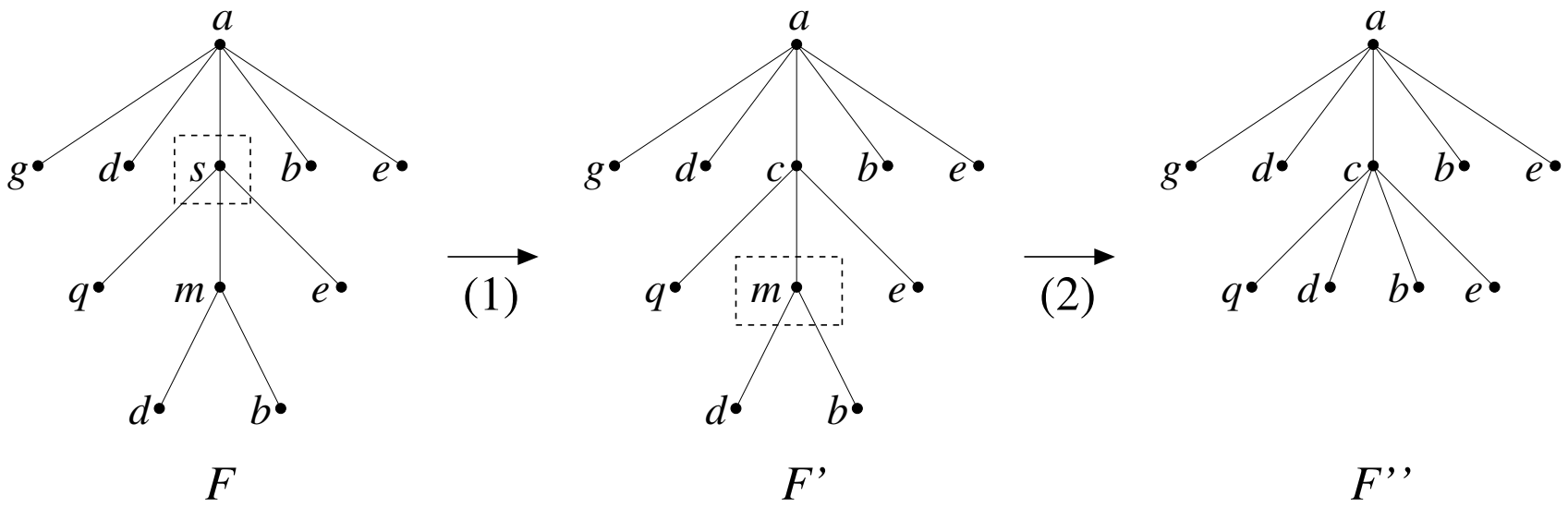
In our paper, we use *forest edit distance* to measure similarity between forests, and consider three types of subforests:

- Simple substructure
- Sibling substructure
- Closed subforest

Forest edit distance

Edit operations on forest F:

- Relabel
- Delete
- Insert



Forest edit distance (cont.)

The *forest edit distance between* F *and* G, denoted by $\delta(F, G)$, equals the minimum number of edit operations needed to transform F into G.

An equivalent formulation: the *minimum cost edit mapping*.

An *edit mapping* [Tai 1979] between F and G is a set M of pairs (i, j) where $i \in F$, $j \in G$ s.t. for any $(i_1, j_1), (i_2, j_2) \in M$:

- $i_1 = i_2$ iff $j_1 = j_2$
- i_1 is an ancestor of i_2 iff j_1 is an ancestor of j_2
- $i_1 < i_2$ iff $j_1 < j_2$

Forest edit distance (cont.)

Each valid M corresponds to one possible way to apply delete and relabel operations on F and G until they become identical.

Let $M_F = \{i \mid (i,j) \in M\}$ and $U_F = F \setminus M_F$, and define M_G and U_G analogously.

Assume the scoring function γ for scoring node pairs is fixed. The *cost* $\delta(M)$ of edit mapping M is defined as:

$$\delta(M) = \sum_{(i,j)\in M} \gamma(f(i), g(j)) + \sum_{i \in U_F} \gamma(f(i), -) + \sum_{j \in U_G} \gamma(-, g(j)).$$

Then, $\delta(F,G) = \min\{\delta(M)\}$ over all possible M.

Subforest definitions

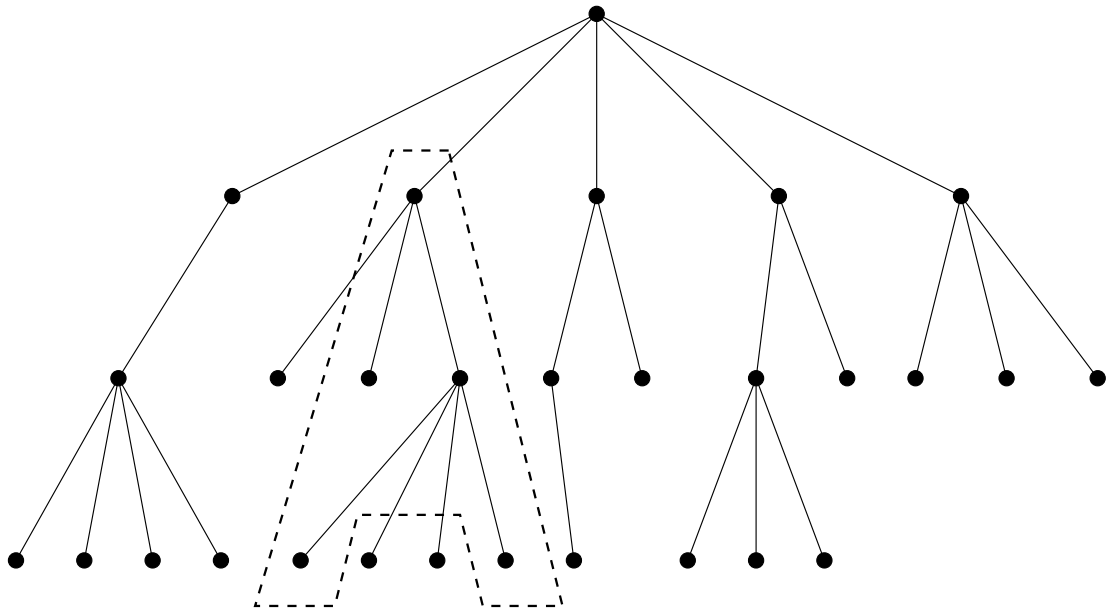
Simple substructure of F:
Any connected subgraph of F.

Sibling substructure of F:
Any set of disjoint simple substructures of F whose roots are siblings (not necessarily consecutive) in F.

Closed subforest of F:
Suppose i_1 and i_2 are siblings in F.
The set of subtrees rooted at $i_1, \ldots, i_2$ forms a *closed subforest of* F, denoted by $F[i_1 \cdot \cdot \, i_2]$.

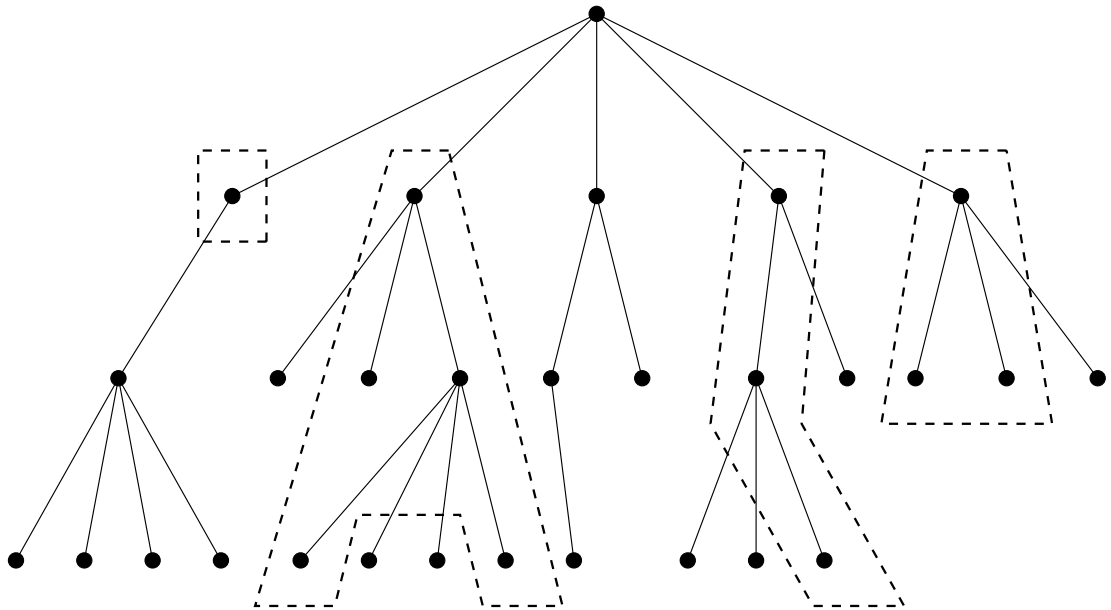
Subforest definitions

Simple substructure of F:
Any connected subgraph of F.



Subforest definitions

Sibling substructure of F:
Any set of disjoint simple substructures of F whose roots are siblings (not necessarily consecutive) in F.

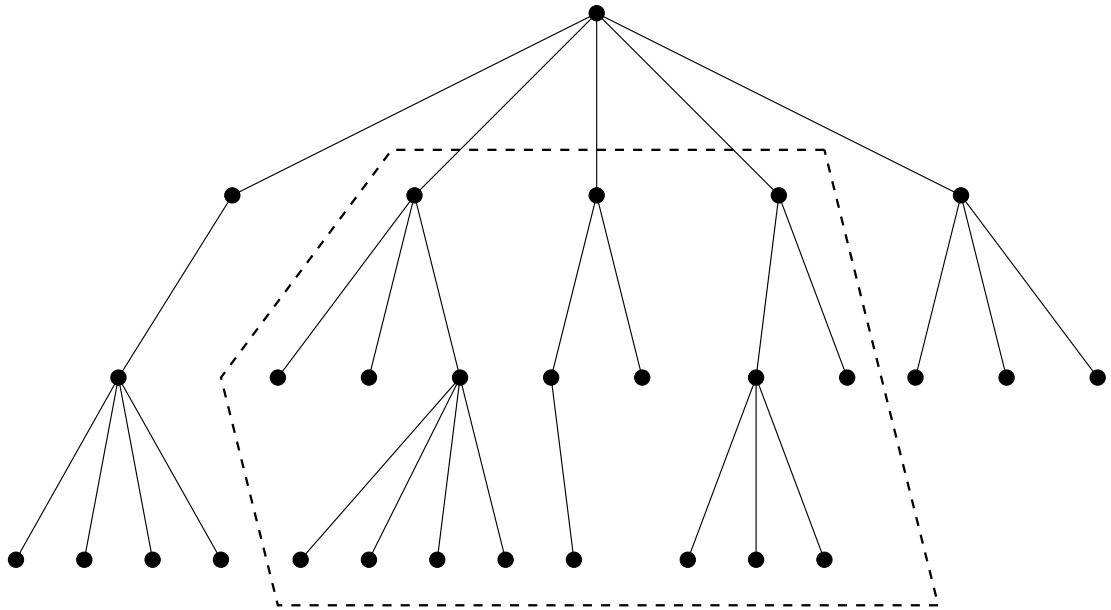


Subforest definitions

Closed subforest of F:

Suppose i_1 and i_2 are siblings in F.

The set of subtrees rooted at $i_1, \ldots, i_2$ forms a *closed subforest of* F, denoted by $F[i_1 \cdot \cdot \, i_2]$.



Our results

Finding a most similar:	Running time
Simple substructure	$O(\|F\|\cdot\|G\|\cdot\min\{\|L(F)\|, dp(F)\}\cdot\min\{\|L(G)\|, dp(G)\})$
Sibling substructure	$O(\|F\|\cdot\|G\|\cdot\min\{\|L(F)\|, dp(F)\}\cdot\min\{\|L(G)\|, dp(G)\})$
Closed subforest	$O(\|F\|\cdot\|G\|\cdot\|L(F)\|\cdot\min\{\|L(G)\|, dp(G)\})$

$|F|$ = The number of nodes in F

$L(F)$ = The set of leaves in F

$dp(F)$ = The depth of F

All algorithms can be implemented to run in $O(|F|\cdot|G|)$ space.

Related results

- Computing the forest edit distance between F and G:
  - Tai (1979)
  - Zhang and Shasha (1989)
  - Klein (1998); Chen (2001); Touzet (2005)

- Computing the forest *alignment* distance between F and G:
  - Jiang, Wang, and Zhang (1995)
  - Jansson and Lingas (2003)

- Computing a most similar closed subforest of F to G using the forest alignment distance:
  - Höchsmann, Töller, Giegerich, and Kurtz (2003)

Finding a most similar closed subforest

Notation:

$p(F)$ = Imaginary parent node of the roots of F

$L(F)$ = The set of leaves in F

$K(F) = \{p(F) \cup i \mid i \in F$ has a proper left sibling$\} =$

$=$ "key nodes of F"

For any subset S of the nodes, $F\|_S$ is the forest obtained by deleting all nodes not in S.

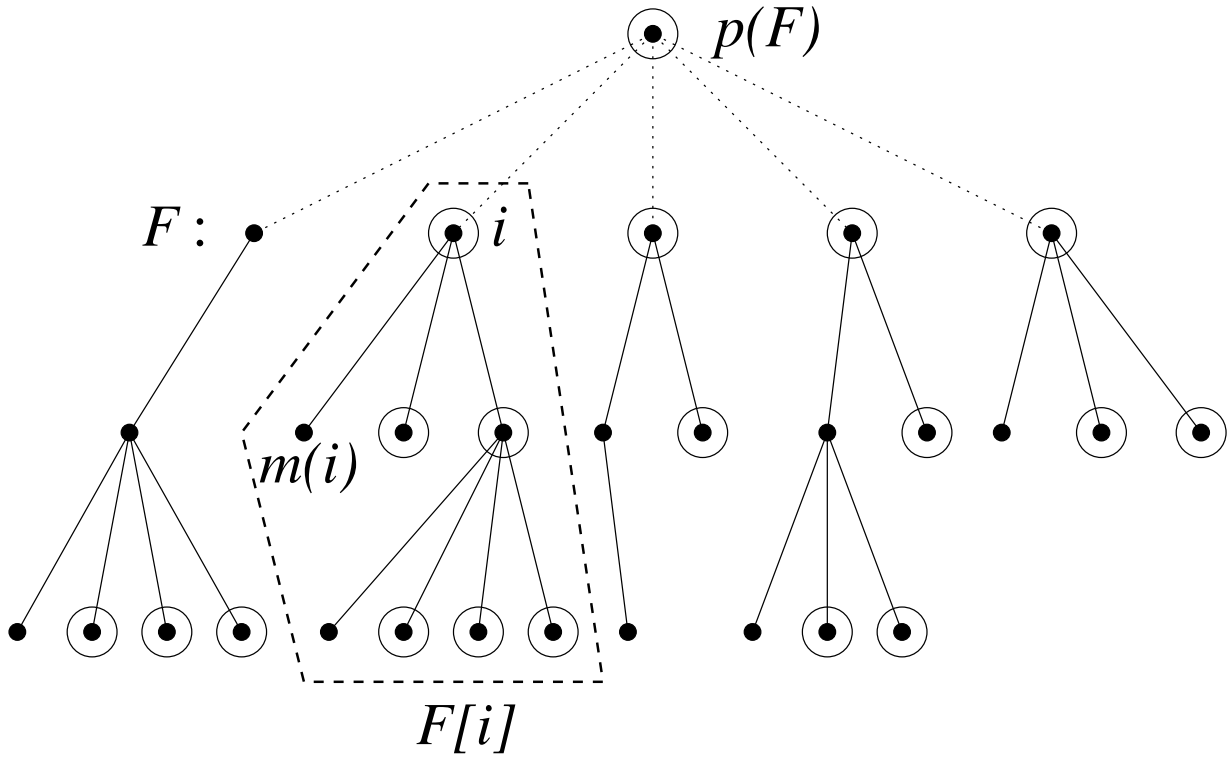
$F[i] =$ the subtree rooted at node i

Number the nodes according to left-to-right postorder traversal.

$m(i) =$ the smallest numbered node in $F[i]$

Preprocessing: Compute $L(F), L(G), K(F), K(G)$ & all $m(i)$.

Example



$K(F)$ is the set of circled nodes.

Main idea: Obtain a recurrence for $\delta(F\|_{l:x}, G\|_{m(j):y})$, where $l \in L(F)$, $j \in K(G)$, $x \in \{l, \ldots, |F|\}$, $y \in \{m(j), \ldots, j)\}$, and apply dynamic programming to compute all these values. Finally, return $\min\{\delta(F\|_{m(i_1):i_2}, G) \mid i_1, i_2$ are siblings in $F\}$. Use some tricks to save time and space.

Main idea: Obtain a recurrence for $\delta(F\|_{l:x}, G\|_{m(j):y})$, where $l \in L(F)$, $j \in K(G)$, $x \in \{l, \ldots, |F|\}$, $y \in \{m(j), \ldots, j)\}$, and apply dynamic programming to compute all these values. Finally, return $\min\{\delta(F\|_{m(i_1):i_2}, G) \mid i_1, i_2$ are siblings in $F\}$. Use some tricks to save time and space.

Correctness:

Suppose $F[a \cdot\cdot\, b]$ is a most similar closed subforest of F to G. $\delta(F\|_{m(a):b}, G)$ is calculated at some point because:

- $m(a) \in L(F)$ and $b \in \{m(a), \ldots, |F|\}$ (i.e., select $l = m(a)$ and $x = b$).
- $G = G\|_{m(p(G)):|G|}$ and $p(G) \in K(G)$ (i.e., select $j = p(G)$ and $y = |G|$).

Main loop:

Input: A target forest F and a pattern forest G.

1: $\delta(\emptyset, \emptyset) := 0$.
2: **for** $l_1 := |L(F)|, \ldots, 1$ **do**
3: **for** $j_1 := 1, \ldots, |K(G)|$ **do**
4: $l := L(F)[l_1]$; $j := K(G)[j_1]$; Call `Compute_Delta`(l, j).
5: **return** $\min\{\delta(F\|_{m(i_1):i_2}, G) \mid i_1, i_2$ are siblings in $F\}$.

Procedure `Compute_Delta`(l, j):

1: **for** $x := l, \ldots, |F|$ **do** $\delta(F\|_{l:x}, \emptyset) := \delta(F\|_{l:x-1}, \emptyset) + \gamma(f(x), -)$.
2: **for** $y := m(j), \ldots, j$ **do** $\delta(\emptyset, G\|_{m(j):y}) := \delta(\emptyset, G\|_{m(j):y-1}) + \gamma(-, g(y))$.
3: **for** $x := l, \ldots, |F|$ **do**
4: **for** $y := m(j), \ldots, j$ **do**
5: Calculate $\delta(F\|_{l:x}, G\|_{m(j):y})$ according to Lemma 7.

Write $l \preceq x$ if $l = x$ or l is a descendant of x (otherwise $l \not\preceq x$).

Since $m(x)$ is precomputed, we can immediately test if $l \preceq x$ simply by checking if $m(x) \leq l \leq x$ is true.

Recurrence: For any $l \in L(F)$, $j \in K(G)$, $x \in \{l, \ldots, |F|\}$, $y \in \{m(j), \ldots, j\}$, $\delta(F\|_{l:x}, G\|_{m(j):y})$ equals the minimum of:

$$\begin{cases} \delta(F\|_{l:x-1}, G\|_{m(j):y}) + \gamma(f(x), -); \\ \delta(F\|_{l:x}, G\|_{m(j):y-1}) + \gamma(-, g(y)); \\ \begin{cases} \delta(\emptyset, G\|_{m(j):m(y)-1}) + \\ \quad \delta(F\|_{l:x-1}, G\|_{m(y):y-1}) + \gamma(f(x), g(y)), & \text{if } l \preceq x. \\ \delta(F\|_{l:m(x)-1}, G\|_{m(j):m(y)-1}) + \\ \quad \delta(F\|_{m(x):x-1}, G\|_{m(y):y-1}) + \gamma(f(x), g(y)), & \text{if } l \not\preceq x. \end{cases} \end{cases}$$

To simplify the implementation:
Rewrite the recurrence to eliminate the dependency on $\delta(\ldots, G\|_{\underline{\mathbf{m(y)}}:y-1})$ and $\delta(F\|_{\underline{\mathbf{m(x)}}:x-1}, \ldots)$.

Lemma 7: For any $l \in L(F)$, $j \in K(G)$, $x \in \{l, \ldots, |F|\}$, $y \in \{m(j), \ldots, j\}$, $\delta(F\|_{l:x}, G\|_{m(j):y})$ equals the minimum of:

$$\begin{cases} \delta(F\|_{l:x-1}, G\|_{m(j):y}) + \gamma(f(x), -); \\ \delta(F\|_{l:x}, G\|_{m(j):y-1}) + \gamma(-, g(y)); \\ \begin{cases} \delta(F\|_{l:x-1}, G\|_{m(j):y-1}) + \gamma(f(x), g(y)), \\ \qquad\qquad \text{if } l \preceq x \text{ and } m(j) = m(y); \\ \delta(\emptyset, G\|_{m(j):m(y)-1}) + \delta(F\|_{l:x}, G[y]), \\ \qquad\qquad \text{if } l \preceq x \text{ and } m(j) \neq m(y); \\ \delta(F\|_{l:m(x)-1}, G\|_{m(j):m(y)-1}) + \delta(F[x], G[y]), \quad \text{if } l \not\preceq x. \end{cases} \end{cases}$$

Space complexity

- While considering each $l \in L(F)$, $j \in K(G)$ in the main loop, store the computed values of $\delta(F\|_{l:x}, G\|_{m(j):y})$ for all $x \in \{l, \ldots, |F|\}$, $y \in \{m(j), \ldots, j\}$ (each call to `Compute_Delta` reuses this space). $\Rightarrow O(|F| \cdot |G|)$ space
- While considering each $l \in L(F)$ in the main loop, store $\delta(F\|_{l:x}, G[y])$ for all $x \in \{l, \ldots, |F|\}$ satisfying $l \preceq x$ and all $y \in G$ (reuse the space for the next $l \in L(F)$).
  $\Rightarrow O(dp(F) \cdot |G|)$ space
- All computed values $\delta(F[x], G[y])$ are stored during the entire execution of the algorithm. $\Rightarrow O(|F| \cdot |G|)$ space

In total, the space complexity is $O(|F| \cdot |G|)$.

Time complexity

All values of δ needed by Step 5 in `Compute_Delta` are available in $O(1)$ time.

Total running time for finding a most similar closed subforest:

$$O(\sum_{l \in L(F)} \sum_{j \in K(G)} |F\|_{l:|F|}| \cdot |G[j]|) =$$

$$\left[\sum_{j \in K(G)} |G[j]| \leq |G| \cdot \min\{|L(G)|, dp(G)\}\right] =$$

$$O(|F| \cdot |G| \cdot |L(F)| \cdot \min\{|L(G)|, dp(G)\})$$

Finding a most similar simple substructure

Allow nodes in F to be *cut* at cost 0.

Finding a most similar simple substructure

Allow nodes in F to be *cut* at cost 0.

$u, v \in F$ are *consistent* if $u \not\preceq v$ and $v \not\preceq u$. A set C of nodes is *consistent* if every pair of nodes in C is consistent.
$\mathcal{C}(F) =$ the set of all consistent sets of nodes in F

For any subforest F' of F and any $C \in \mathcal{C}(F')$, let $F' \ominus C$ be the forest obtained from F' by cutting all nodes in C.
Define $\Psi(F', G) = \min_{C \in \mathcal{C}(F')}\{\delta(F' \ominus C, G)\}$.

Goal of the algorithm: Compute $\min_{i \in F}\{\Psi(F[i], G)\}$.

Finding a most similar simple substructure (cont.)

Strategy: Derive a recurrence for Ψ and use a similar technique as before to compute $\Psi(F\|_{m(i):x}, G\|_{m(j):y})$ for all $i \in K(F)$, $j \in K(G)$, $x \in \{m(i), \ldots, i\}$, $y \in \{m(j), \ldots, j)\}$.
Finally, return $\min_{i \in F}\{\Psi(F[i], G)\}$.

The algorithm can be implemented to run in:

- $O(|F| \cdot |G| \cdot \min\{|L(F)|, dp(F)\} \cdot \min\{|L(G)|, dp(G)\})$ time
- $O(|F| \cdot |G|)$ space

Finding a most similar sibling substructure

Identical to the previous algorithm (finding a most similar simple substructure), except that we look for $\min_{i \in F}\{\Psi(F\|_{m(i):i-1}, G)\}$ instead of $\min_{i \in F}\{\Psi(F\|_{m(i):i}, G)\}$.

$\Rightarrow$ The algorithm's complexity does not change:

- $O(|F| \cdot |G| \cdot \min\{|L(F)|, dp(F)\} \cdot \min\{|L(G)|, dp(G)\})$ time
- $O(|F| \cdot |G|)$ space

Concluding remarks

- We can generalize our algorithms to find a subforest F' of F and a subforest G' of G that are the most similar.

Concluding remarks

- We can generalize our algorithms to find a subforest F' of F and a subforest G' of G that are the most similar.
- Open: Other types of subforests, such as gapped subforests? (*gapped subforest* of F = remove from a closed subforest F' of F a set C of closed subforests such that no two closed subforests in C have the same parent in F')