論文番号: C2-3

構造の異なる文書データを同じ構造に変換するアルゴリズムについて

鈴木 伸崇　佐藤 洋一郎　早瀬 道芳
岡山県立大学
総社市窪木 111
{suzuki, sato, hayase}@cse.oka-pu.ac.jp

あらまし　本稿では，半構造データ，特に，構造が類似しているが一定ではない文書データの集合について考える．データ構造が一定でない場合，データ格納や検索などの効率が低下する．そのため，データ構造は一定であることが望ましい．本稿では，構造の異なる複数の文書データを同じ構造に変換するアルゴリズムについて考える．このアルゴリズムでは，文書データを順序木とみなし，頂点の追加，削除，ラベル（要素名）の変更という三種の操作を用いて文書データの構造を変換する．本稿では，まず，二つの文書データを最小の変換コストで同じ構造に変換する多項式時間アルゴリズムを示す．ここで，変換コストとは，変換の際に要する操作のコストの和である．次に，任意の数の文書データを最小の変換コストで同じ構造に変換するアルゴリズムを示す．最後に，任意の数の文書データを最小の変換コストで同じ構造に変換する問題が NP 困難であることを示す．

キーワード: 半構造データ，データ変換，アルゴリズム，NP 困難性

On Algorithms for Transforming Semistructured Documents into the Same Structure

Nobutaka Suzuki　Yoichirou Sato　Michiyoshi Hayase
Okayama Prefectural University
Soja-shi, 719-1197 Japan
{suzuki, sato, hayase}@cse.oka-pu.ac.jp

Abstract　In this paper, we consider semistructured data consisting of documents whose structures are somewhat similar but not same. Leaving such documents unarranged brings several difficulties such as inefficient data retrieval and wasteful data storage, thereby it is desirable to transform the documents into ones with the same structure. In this paper, we consider algorithms that make such transformations, where each document is modeled by a rooted ordered tree and a transformation is achieved by a sequence of operations such as (i) node addition, (ii) node deletion, and (iii) renaming the labels of nodes. In this paper, we first show a polynomial-time algorithm that optimally transforms given two ordered trees into ones with the same structure. Second, we show an algorithm that optimally transforms given an arbitrary number of ordered trees into ones with the same structure. Finally, we show that the problem of transforming an arbitrary number of ordered trees into ones with the same structure is NP-hard.

keywords: semistructured data, data transformation, algorithm, NP-hardness

1. はじめに

本稿では，半構造データ，特に，構造がある程度類似しているが一定ではない文書データの集合について考える．データ構造が一定でない場合，データ格納や検索などの効率が低下する．このため，データ構造は一定であることが望ましい．本稿では，構造の異なる複数の文書データを同じ構造に変換するアルゴリズムについて考える．

文書データは，要素を頂点，要素名を頂点のラベルとする根付き順序木 (以下，単に木) としてモデル化される．その例を図 1 に示す．構造の変換は，頂点の追加，削除，ラベルの変更という 3 種の操作によって行う (各操作にはコストが付与される)．本稿では，以下の結果を示す．まず，2 個の文書データを最小のコストで同じ構造に変換する多項式時間アルゴリズムを示す．次に，任意の個数の文書データを最小のコストで同じ構造に変換するアルゴリズムを示す．最後に，任意の個数の文書データを最小のコストで同じ構造に変換する問題が NP 困難であることを示す．

木 t_1, t_2 を同じ構造に変換する手順の概要は以下の通りである．

1. t_1, t_2 の各頂点に対して，後順による番号を付加する．
2. t_1 に対する森 f_1 と t_2 に対する森 f_2 を設ける．f_1 と f_2 の初期値は共に $\emptyset$ である．
3. f_1 には t_1 の頂点を，f_2 には t_2 の頂点を一つづつ追加していく．これを次の条件で行う．
   - 追加の順序は，1. で与えられた番号順とする．
   - f_1 と f_2 は常に同型であるようにする．同型性を維持するため，3. の頂点の追加の際，必要に応じて頂点の追加，削除，ラベル変更の操作を施す．

例 1: 図 2 における木 t_1, t_2 を考える．ここで，t_1 と t_2 の各頂点の番号はその頂点に対する後順の順序を表す．以下に t_1 と t_2 を同じ構造に変換する過程の一例を示す．以下，木 t における i 番目の頂点を $t[i]$ と表す．また，t_1 (t_2) に対する森で j 番目 (図 2 における番号 j の列) に得られたものを f_1^j (f_2^j) と表す ($1 \leq j \leq 6$)．このような同型の森からなる組 (f_1^j, f_2^j) を導出対と呼ぶ．

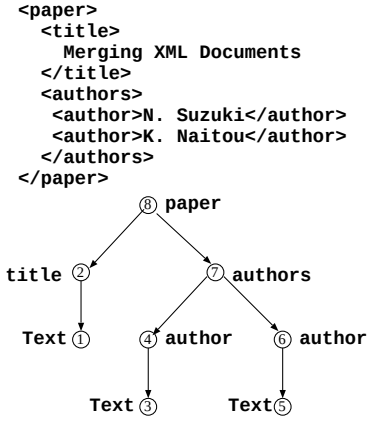

図 1: XML データとその木構造

1. 初期状態は，$f_1^1 = f_2^1 = \emptyset$ である．
2. まず，f_2^1 に $t_2[1]$ を追加し f_2^2 を得る．次に，t_1 は $t_2[1]$ に相当するラベル b をもつ頂点を含まない．そこで，同型性を維持するためラベル b をもつ頂点を新たに作成し，f_1^1 に追加する．これが f_1^2 となる．
3. f_1^2 に $t_1[1]$ を追加し f_1^3 を得る．また，f_2^2 に $t_2[2]$ を追加し f_2^3 を得る．$t_1[1]$ と $t_2[2]$ は同じラベルをもつので，同型性を維持したまま両者を f_1^2 と f_2^2 に追加できる．
4. まず，f_1^3 に $t_1[2]$ を追加し f_1^4 を得る．次に，t_2 は $t_1[2]$ に相当するラベル c をもつ頂点を含まない．そこで，同型性を維持するためラベル c をもつ頂点を新たに作成し，f_2^3 に追加する．これが f_2^4 となる．
5. t_2 はラベル d をもつ頂点 $t_2[3]$ を含むが，t_1 はそのような頂点を含まない．この段階では，同型性を維持するため $t_2[3]$ を森に追加せずに次のステップへ移る ($t_2[3]$ の削除)．この操作では頂点は追加されないので，$f_1^4 = f_1^5$ かつ $f_2^4 = f_2^5$ である．
6. f_1^5 に $t_1[3]$ を追加し f_1^6 を得る．また，f_2^5 に $t_2[4]$ を追加し f_2^6 を得る．$t_1[3]$ と $t_2[4]$ は同じラベルをもつので，同型性を維持したまま両者を f_1^5 と f_2^5 に追加できる．$t_1[3]$ と $t_2[4]$ はそれぞれ t_1 と t_2 における最終の頂点なので，ここで変換処理は終了する． □

これまで，XML データを変換するための操作言語はいくつか提案されている [6]．他方，データ変換を最小コストで行うアルゴリズムは考察されていない．文献 [2, 4] では，与えられた順序木を包含する最小木を求めるアルゴリズムが提案されているが，頂点の削除やラベルの変更は考慮されていない．文献 [1, 5] では，2 個の木 t_1, t_2 に対して，t_1 を t_2 に変換するアルゴリズムが提案されている．この変換は t_1 から t_2 への一方的な写像であるのに対し，本稿のアルゴリズムは t_1 と t_2 の共通木を求めている．また，これら文献では任意の個数の木に対する変換は考察されていない．

2. 諸定義

t を根付き順序木 (以下，単に木) とする．t の頂点数を $|t|$ と表す．t における各頂点を後順による番号で識別する．

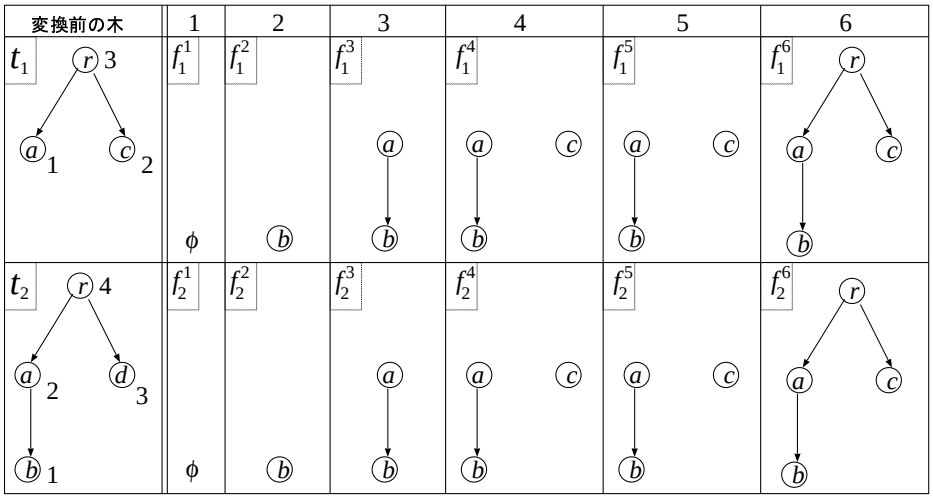

図 2: データ変換の過程

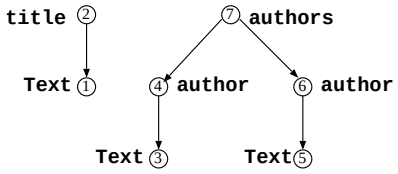

図 3: 森 $t[1,7]$

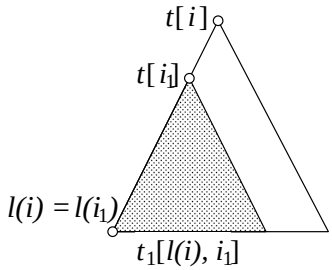

図 4: $t_1[l(i), i_1]$

例えば，図 1 の各頂点に記された数字は，その頂点の番号を表す．t における i 番目の頂点を $t[i]$ と表す．$t[i]$ を根とする t の部分木において，最左葉の番号を $l(i)$ と表す．例えば，図 1 の木において $l(7) = 3$ かつ $l(5) = 5$ である．頂点 $t[i], t[i+1], \cdots, t[j]$ からなる森を $t[i, j]$ と表す．例えば，図 3 は森 $t[1, 7]$ を表し，ここで t は図 1 の木である．もし $i > j$ ならば，$t[i, j] = \emptyset$ と定義する．各頂点 n はラベル $lb(n)$ をもつ．頂点 n の親を $par(n)$，右隣の弟を $rs(n)$ とそれぞれ表す．$par(n) = n'$ であることと n' から n への有向辺が存在することは同値である．森 f における，頂点 n の祖先からなる集合を $anc(f, n)$ と表す．

以下，5 種の操作 ϵ，add, del, ren, $union$ を定義する．操作系列 op が操作系列 op' に操作 e を連接したものであることを $op = op' + e$ と表す．以下，任意の頂点 n に対して $par(n)$，$rs(n)$，及び，$mark(n)$ の初期値は $null$ であり，$t_1[i] \in anc(t_1, t_1[i_1])$ かつ $t_2[j] \in anc(t_2, t_2[j_1])$ と仮定する ($mark(n)$ の意味は後述する)．

Epsilon (ϵ)： 任意の葉 $t_1[l(i)], t_2[l(j)]$ に対して，$(\epsilon(t_1)[l(i), l(i)-1], \epsilon(t_2)[l(j), l(j)-1])$ は導出対である．ϵ は空の操作を表し，$\epsilon(t_1)[l(i), l(i)-1] = t_1[l(i), l(i)-1] = \emptyset$ かつ $\epsilon(t_2)[l(j), l(j)-1] = t_2[l(j), l(j)-1] = \emptyset$ である．

$add(t_2, t_1[i_1])$：もし $(op'(t_1)[l(i), i_1 - 1], op'(t_2)[l(j), j_1])$ が導出対，$op = op' + add(t_2, t_1[i_1])$，かつ，以下の条件が成り立つならば，$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は導出対である．

条件: (i) $l(i) = l(i_1)$ かつ $l(j) = l(j_1)$，または，(ii) $j_1 = l(j) - 1$.

条件 (i) は $t_1[l(i), i_1]$ と $t_2[l(j), j_1]$ がそれぞれ $t_1[i_1]$ と $t_2[j_1]$ を根とする木であることを表し (図 4)，条件 (ii) は $t_2[l(j), j_1] = \emptyset$ であることを表す．$ap(t_1[i_1]) = |op'(t_1)[l(i), i_1 - 1]| + 1$ と定義する．ここで，$ap(t_1[i_1])$ は $t_1[i_1]$ の出現順序を表し，兄弟間の順序を決定するのに用いる (後述)．なお，例 1 のステップ 4 は $add(t_2, t_1[2])$ に相当する．$op(t_1)[l(i), i_1]$ を以下のように構成する．

(1a) $t_1[i]$ を親とすべき各頂点 n に対して，$par(n) = t_1[i]$ とする．具体的には，まず，$par(n) = null$ かつ $t_1[i_1] \in anc(t_1, n)$ を満たす各頂点 $n \in op'(t_1)[l(i), i_1 - 1]$ に対して，$par(n) = t_1[i_1]$ とする．次に，$par(n) = null$ かつ $t_1[i_1] \in anc(t_1, mark(n))$ を満たす各頂点 $n \in op'(t_1)[l(i), i_1 - 1]$ に対して，$par(n) = t_1[i_1]$ とする．

(1b) $t_1[i]$ の子の順序を定める．具体的には，(1a) で $par(n) = t_1[i_1]$ とされたすべての頂点 n からなる系列で，$ap(n_1) < ap(n_2) < \cdots < ap(n_k)$ を満たすものを $n_1, n_2, \cdots, n_k$ とする．各 $2 \leq i \leq k$ に対して $rs(n_{i-1}) = n_i$ とする．

$op(t_2)[l(j), j_1]$ を以下のように構成する．

(2a) 新たな頂点 n_{new} を $op'(t_2)[l(j), j_1]$ に追加する．n_{new} と $t_1[i]$ は対応するという．$lb(n_{new}) = lb(t_1[i_1])$ かつ $ap(n_{new}) = |op'(t_2)[l(j), j_1]| + 1$ とする．更に，もし $t_2[l(j), j_1] \neq \emptyset$ ならば，$mark(n_{new}) = t_2[j_1]$ とする．$mark(n_{new})$ の値が $t_2[j_1]$ であることは，$t_2[j_1]$ の子孫で $op(t_2)[l(j), j_1]$ に属するものはすべて n_{new} の子孫であることを表す.

(2b) n_{new} を親とすべき各頂点 n に対して，$par(n) = n_{new}$ とする．具体的には，(1a) において $par(n) = t_1[i_1]$ とされた各頂点 $n \in op(t_1)[l(i), i_1]$ に対して n に対応する頂点 $n' \in op'(t_2)[l(j), j_1]$ を求め，$par(n') = n_{new}$ とする．

(2c) n_{new} の子の順序を定める．具体的には，(2b) で $par(n') = n_{new}$ とされたすべての頂点 n' からなる系列で，$ap(n'_1) < ap(n'_2) < \cdots < ap(n'_k)$ を満たすものを $n'_1, n'_2, \cdots, n'_k$ とする．各 $2 \leq i \leq k$ に対して $rs(n'_{i-1}) = n'_i$ とする．

$\underline{add(t_1, t_2[j_1])}$: もし $(op'(t_1)[l(i), i_1], op'(t_2)[l(j), j_1 - 1])$ が導出対，$op = op' + add(t_1, t_2[j_1])$，かつ，以下の条件が成り立つならば，$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は導出対である．

条件: (i) $l(i) = l(i_1)$ かつ $l(j) = l(j_1)$，または，(ii) $i_1 = l(i) - 1$.

$op(t_1)[l(i), i_1]$ と $op(t_2)[l(j), j_1]$ の構成は $add(t_2, t_1[i_1])$ と同様である．例えば，例 1 のステップ 1 は $add(t_1, t_2[1])$ に相当する．

$\underline{del(t_2, t_2[j_1])}$: もし $(op'(t_1)[l(i), i_1], op'(t_2)[l(j), j_1 - 1])$ が導出対かつ $op = op' + del(t_2, t_2[j_1])$ ならば，$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は導出対である．$op(t_1)[l(i), i_1]$ 及び $op(t_2)[l(j), j_1]$ は，それぞれ $op'(t_1)[l(i), i_1]$ 及び $op'(t_2)[l(j), j_1 - 1]$ と等しいと定義する．すなわち，$del(t_2, t_2[j_1])$ は t_2 から $t_2[j_1]$ を削除することを表す．例えば，例 1 のステップ 5 は $del(t_2, t_2[3])$ に相当する．

$\underline{del(t_1, t_1[i_1])}$: もし $(op'(t_1)[l(i), i_1 - 1], op'(t_2)[l(j), j_1])$ が導出対かつ $op = op' + del(t_1, t_1[i_1])$ ならば，$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は導出対である．$op(t_1)[l(i), i_1]$ 及び $op(t_2)[l(j), j_1]$ は，それぞれ $op'(t_1)[l(i), i_1 - 1]$ 及び $op'(t_2)[l(j), j_1]$ と等しいと定義する．すなわち，$del(t_1, t_1[i_1])$ は t_1 から $t_1[i_1]$ を削除することを表す.

$\underline{ren(t_1[i_1], t_2[j_1])}$: もし $l(i) = l(i_1)$，$l(j) = l(j_1)$，$(op'(t_1)[l(i), i_1 - 1], op'(t_2)[l(j), j_1 - 1])$ が導出対，かつ，$op = op' + ren(t_1[i_1], t_2[j_1])$ ならば，$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は導出対である．$t_1[i_1]$ と $t_2[j_1]$ は対応するという．$ren(t_1[i_1], t_2[j_1])$ は $t_1[i_1]$ と $t_2[j_1]$ を「等しい」とみなす操作である．例えば，例 1 のステップ 3 とステップ 6 は，それぞれ $ren(t_1[1], t_2[2])$ と $ren(t_1[3], t_2[4])$ に相当する．$op(t_1)[l(i), i_1]$ と $op(t_2)[l(j), j_1]$ を以下のように構成する (1 は $t_1[i_1]$ と $t_2[j_1]$ のラベルの設定，2–4 は $t_1[i_1]$ と $t_2[j_1]$ の子の同定，5 はそれら子の順序の設定である).

1. 任意のラベル l を定め，$lb(t_1[i_1]) = lb(t_2[j_1]) = l$ とする．
2. $par(n_1) = null$ かつ $t_1[i_1] \in anc(t_1, n_1)$ を満たす各頂点 $n_1 \in op'(t_1)[l(i), i_1 - 1]$ に対して，以下の操作を行う．
   (a) $par(n_1) = t_1[i_1]$ とする．
   (b) n_1 に対応する頂点 $n'_1 \in op'(t_2)[l(j), j_1 - 1]$ を求める．$par(n'_1) = t_2[j_1]$ とする．
3. $par(n_2) = null$ かつ $t_2[j_1] \in anc(t_2, n_2)$ を満たす各頂点 $n_2 \in op'(t_2)[l(j), j_1 - 1]$ に対して，以下の操作を行う．
   (a) $par(n_2) = t_2[j_1]$ とする．
   (b) n_2 に対応する頂点 $n'_2 \in op'(t_1)[l(i), i_1 - 1]$ を求める．$par(n'_2) = t_1[i_1]$ とする．
4. $ap(t_1[i_1]) = |op'(t_1)[l(i), i_1 - 1]| + 1$ かつ $ap(t_2[j_1]) = |op'(t_2)[l(j), j_1 - 1]| + 1$ と定義する．
5. $t_1[i_1]$ と $t_2[j_1]$ の子の順序は，add と同様に定める．

$\underline{union}$: もし $(op_1(t_1)[l(i), l(i_1) - 1], op_1(t_2)[l(j), l(j_1) - 1])$ と $(op_2(t_1)[l(i_1), i_1], op_2(t_2)[l(j_1), j_1])$ が共に導出対かつ $op = op_1 + op_2$ ならば，$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は導出対である．$op(t_1)[l(i), i_1]$ を $op_1(t_1)[l(i), l(i_1) - 1]$ と $op_2(t_1)[l(i_1), i_1]$ の和と定義する．同様に，$op(t_2)[l(j), j_1]$ を $op_1(t_2)[l(j), l(j_1) - 1]$ と $op_2(t_2)[l(j_1), j_1]$ の和と定義する (図 5)．各頂点 $n_1 \in op_2(t_1)[l(i_1), i_1]$ に対して，$ap(n_1)$ に $|op_1(t_1)[l(i), l(i_1) - 1]|$ を加算する．同様に，各頂点 $n_2 \in op_2(t_2)[l(j_1), j_1]$ に対して，$ap(n_2)$ に $|op_1(t_2)[l(j), l(j_1) - 1]|$ を加算する．

$(op(t_1)[l(i_1), i'_1], op(t_2)[l(i_2), i'_2])$ を導出対とする．もし任意の異なる頂点 $n, n' \in t_l[l(i_l), i'_l] \cap op(t_l)[l(i_l), i'_l]$ $(l \in [1, 2])$ に対して「$n \in anc(t_l[l(i_l), i'_l], n')$ かつその

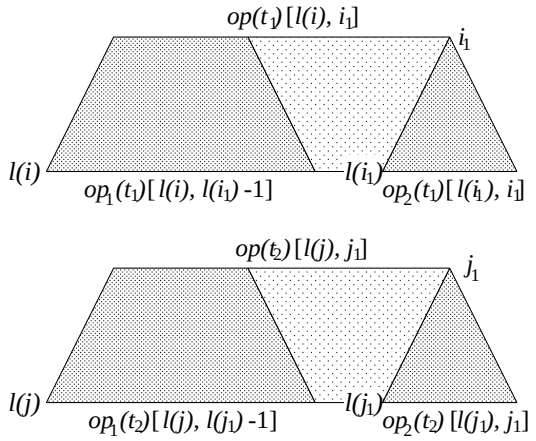

図 5: $(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$

ときのみ $n \in anc(op(t_l)[l(i_l), i'_l], n')$ である」ならば，$op(t_l)[l(i_l), i'_l]$ は祖先順序保存であるという．森 f と頂点 $n_1, n_2 \in f$ に対して，もしある $m_1 \in anc(f, n_1)$ と $m_2 \in anc(f, n_2)$ が存在して m_1 が m_2 の兄 (弟) であるならば，n_1 は n_2 の左 (右) に位置するという．もし任意の異なる頂点 $n, n' \in t_l[l(i_l), i'_l] \cap op(t_l)[l(i_l), i'_l]$ に対して「もし $op(t_l)[l(i_l), i'_l]$ において n が n' の左 (右) に位置するならば，$t_l[l(i_l), i'_l]$ において n が n' の左 (右) に位置する」ならば，$op(t_l)[l(i_l), i'_l]$ は兄弟順序保存であるという．もし次の 2 条件が成り立つならば，導出対 $(op(t_1)[l(i_1), i'_1], op(t_2)[l(i_2), i'_2])$ は妥当であるという．

- $op(t_1)[l(i_1), i'_1]$ と $op(t_2)[l(i_2), i'_2]$ は同型である．
- $op(t_1)[l(i_1), i'_1]$ と $op(t_2)[l(i_2), i'_2]$ は祖先順序保存かつ兄弟順序保存である．

操作 e のコストを $\gamma(e)$ と表す．e が add，del，ren である場合は $\gamma(e) \geq 0$，ϵ または $union$ である場合は $\gamma(e) = 0$ と仮定する．$op = e_1 e_2 \cdots e_n$ を操作系列とする．op のコスト $\gamma(op)$ を $\gamma(op) = \sum_{i=1}^{n} \gamma(e_i)$ と定義する．$t_1[l(i), i_1]$ と $t_2[l(j), j_1]$ を同じ構造に変換するための最小コスト $c_m(t_1[l(i), i_1], t_2[l(j), j_1])$ を次のように定義する．

$$c_m(t_1[l(i), i_1], t_2[l(j), j_1]) = \min\{\gamma(op) \mid (op(t_1)[l(i), i_1], op(t_2)[l(j), j_1]) \text{ は妥当 }\}$$

もし $(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ が妥当かつ $\gamma(op) = c_m(t_1[l(i), i_1], t_2[l(j), j_1])$ ならば，op は $(t_1[l(i), i_1], t_2[l(j), j_1])$ に対して最適であるという．

3. 2 個の木を変換するための最小コスト

本節では，与えられた 2 個の木 t_1, t_2 に対して $c_m(t_1, t_2)$ を求める多項式時間アルゴリズムを示す．

t を木とする．$keyroots(t)$ を次の条件を満たすすべての i からなる集合とする．

- $t[i]$ は t の根である，または，
- $t[i]$ は t の部分木の根かつ $t[i]$ の兄が 1 つ以上存在する．

例えば，図 1 の木 t において $keyroots(t) = \{6, 7, 8\}$ である．

以下に $c_m(t_1, t_2)$ を求める多項式時間アルゴリズムを示す (アルゴリズム 1)．5 行目と 7 行目で用いられる Cost は，動的計画法を用いて $c_m(t_1[l(i), i], t_2[l(j), j])$ を計算する (アルゴリズム 2)．Cost において，配列 c_g は「グローバル変数」である．Cost のある呼び出しで得られた c_g の値 (9 行目) は呼び出し後も保存され，後に別の Cost の呼び出しで参照される (11 行目)．

アルゴリズム 1: $c_l(t_1, t_2)$ の計算

```
Input: 木 t1, t2.
Output: 変換コスト c_m(t1, t2).
 begin
1. Compute keyroots(t1) and keyroots(t2).
2. for i' ← 1 to |keyroots(t1)| do
3.   for j' ← 1 to |keyroots(t2)| do begin
4.     Let i be the i'th item in keyroots(t1), and
       let j be the j'th item in keyroots(t2).
5.     Cost(t1[l(i), i], t2[l(j), j]);
6.   end
7.  return Cost(t1[1, |t1|], t2[1, |t2|]);
 end
```

アルゴリズム 2: Cost

```
Input: 木 t1[l(i), i], t2[l(j), j].
Output: 変換コスト c_l(t1[l(i), i], t2[l(j), j]).
 begin
1. c_l(t1[l(i), l(i) − 1], t2[l(j), l(j) − 1]) ← 0;
2. for i1 ← l(i) to i do
3.   c_l(t1[l(i), i1], t2[l(j), l(j) − 1])
       ← c_l(t1[l(i), i1 − 1], t2[l(j), l(j) − 1])
         + min{γ(add(t2, t1[i1])), γ(del(t1, t1[i1]))};
4. for j1 ← l(j) to j do
5.   c_l(t1[l(i), l(i) − 1], t2[l(j), j1])
       ← c_l(t1[l(i), l(i) − 1], t2[l(j), j1 − 1])
         + min{γ(add(t1, t2[j1])), γ(del(t2, t2[j1]))};
6. for i1 ← l(i) to i do
7.   for j1 ← l(j) to j do
8.     if l(i1) = l(i) and l(j1) = l(j) then begin
9.       c_l(t1[l(i), i1], t2[l(j), j1]) ← min{
           c_l(t1[l(i), i1 − 1], t2[l(j), j1]) + γ(add(t2, t1[i1])),
           c_l(t1[l(i), i1], t2[l(j), j1 − 1]) + γ(add(t1, t2[j1])),
           c_l(t1[l(i), i1], t2[l(j), j1 − 1]) + γ(del(t2, t2[j1])),
           c_l(t1[l(i), i1 − 1], t2[l(j), j1]) + γ(del(t1, t1[i1])),
           c_l(t1[l(i), i1 − 1], t2[l(j), j1 − 1])
            + γ(ren(t1[i1], t2[j1]))};
           c_g(t1[l(i), i1], t2[l(j), j1]) ← c_l(t1[l(i), i1], t2[l(j), j1]);
10.    else
11.      c_l(t1[l(i), i1], t2[l(j), j1]) ← min{
           c_l(t1[l(i), i1], t2[l(j), j1 − 1]) + γ(del(t2, t2[j1])),
           c_l(t1[l(i), i1 − 1], t2[l(j), j1]) + γ(del(t1, t1[i1])),
           c_l(t1[l(i), l(i1) − 1], t2[l(j), l(j1) − 1])
```

$+ c_g(t_1[l(i_1), i_1], t_2[l(j_1), j_1])\}$;
12**return** $c_l(t_1[l(i), l(i)], t_2[l(j), l(j)])$;
end

アルゴリズム 1 の実行時間が多項式であることは容易に示せる．また，上記アルゴリズムを用いて，最適な操作系列を求めることができる (紙数の都合で省略)．以下，アルゴリズム 1 の正当性を次の順序で示す．

1. 任意の導出対は妥当である (補題 1 と補題 2)．
2. 導出対に関する最小コストを与える (補題 3 と補題 4)．

まず，以下の補題を示す．

補題 1: 任意の導出対 $(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は次の条件を満たす．

(A) $op(t_1)[l(i), i_1]$ と $op(t_2)[l(j), j_1]$ は同型である．

(B) 任意の頂点 $n_1 \in op(t_1)[l(i), i_1]$ と対応する頂点 $n_2 \in op(t_2)[l(j), j_1]$ に対して，$ap(n_1) = ap(n_2)$ である．

略証: $|op|$ に関する帰納法で示せる (詳細は紙数の都合で省略)． □

補題 2: 任意の導出対 $(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ に対して，$op(t_1)[l(i), i_1]$ と $op(t_2)[l(j), j_1]$ は祖先・兄弟順序保存である．

証明: 任意の導出対 $(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ に対して，$|op|$ に関する帰納法で $op(t_1)[l(i), i_1]$ と $op(t_2)[l(j), j_1]$ が祖先/兄弟順序保存であることを示す．

初期段: $|op| = 0$ である．定義から，どの導出対もある i' と j' に対して $(\epsilon(t_1)[i', i' - 1], \epsilon(t_2)[j', j' - 1])$ と表せる．$\epsilon(t_1)[i', i' - 1]$ と $\epsilon(t_2)[j', j' - 1]$ は共に $\emptyset$ であり，よって祖先・兄弟順序保存である．

帰納段: 帰納法の仮定として，もし $|op| < m$ ならば，任意の導出対 $(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ に対して $op(t_1)[l(i), i_1]$ と $op(t_2)[l(j), j_1]$ は祖先/兄弟順序保存である．$|op| = m$ の場合を考える．$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ が導出対であると仮定する．op の最後の操作が (1) add, (2) del, (3) ren, (4) $union$ の場合に分けて考える．

(1) の場合: (i) $l(i) = l(i_1)$ かつ $l(j) = l(j_1)$, 及び，(ii) $t_2[l(i), j_1] = \emptyset$ の場合に分けて考える．以下，(i) の場合を示す．このとき，(a) $op = op' + add(t_2, t_1[i_1])$ または (b) $op = op' + add(t_1, t_2[j_1])$ である．以下，(a) の場合を考える（(b) の場合も同様）．$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は導出対かつ $op = op' + add(t_2, t_1[i_1])$ なので，$(op'(t_1)[l(i), i_1 - 1], op'(t_2)[l(j), j_1])$ は導出対である．したがって，帰納法の仮定から $op'(t_1)[l(i), i_1 - 1]$ と $op'(t_2)[l(j), j_1]$ は祖先・兄弟順序保存である．まず，$op(t_1)[l(i), i_1]$ が祖先順序保存であることを示す．$op'(t_1)[l(i), i_1 - 1]$ は祖先順序保存なので，任意の頂点 $n \in op(t_1)[l(i), i_1] \cap t_1[l(i), i_1]$ に対して以下の 2 つが同値であることを示せばよい．

- $t_1[i_1] \in anc(t_1[l(i), i_1], n)$ である．
- $t_1[i_1] \in anc(op(t_1)[l(i), i_1], n)$ である．

($\Leftarrow$) $t_1[i_1] \in anc(op(t_1)[l(i), i_1], n)$ と仮定する．このとき，$par(n') = t_1[i_1]$ かつ $n' \in anc(op(t_1)[l(i), i_1], n)$ を満たす頂点 $n' \in op(t_1)[l(i), i_1]$ が存在する．(i) $n' \in t_1$ と (ii) $n' \notin t_1$ の場合に分けて考える． まず，(i) の場合を考える．$n' \in anc(op(t_1)[l(i), i_1], n)$ より $n' \in anc(op'(t_1)[l(i), i_1 - 1], n)$ である．$n' \in t_1$, $n' \in anc(op'(t_1)[l(i), i_1 - 1], n)$, かつ，$op'(t_1)[l(i), i_1 - 1]$ は祖先順序保存なので，$n' \in anc(t_1[l(i), i_1 - 1], n)$ である．add の (1a) において $par(n')$ に $t_1[i_1]$ が代入されているので $t_1[i_1] \in anc(t_1, n')$ であり，よって $t_1[i_1] \in anc(t_1[l(i), i_1], n')$ である． $n' \in anc(t_1[l(i), i_1 - 1], n)$ かつ $t_1[i_1] \in anc(t_1[l(i), i_1], n')$ なので，$t_1[i_1] \in anc(t_1[l(i), i_1], n)$ である．次に，(ii) の場合を考える．$n' \notin t_1$ かつ add の (1a) より $par(n')$ に $t_1[i_1]$ が代入されるので，ある i' に対して $mark(n') = t_1[i']$ かつ $t_1[i_1] \in anc(t_1, t_1[i'])$ である．したがって，$t_1[i_1] \in anc(t_1[l(i), i_1], t_1[i'])$ が成り立つ．$mark(n') = t_1[i']$ なので，add の定義から n' のすべての子孫 $t_1[i'']$ $\in op'(t_1)[l(i), i_1 - 1]$ に対して，$l(i') \leq i'' \leq i'$ である．よって，$n' \in anc(op(t_1)[l(i), i_1], n)$ より $t_1[i'] \in anc(t_1[l(i), i_1 - 1], n)$ である．$t_1[i'] \in anc(t_1[l(i), i_1 - 1], n)$ かつ $t_1[i_1] \in anc(t_1[l(i), i_1], t_1[i'])$ なので，$t_1[i_1] \in anc(t_1[l(i), i_1], n)$ である．

($\Rightarrow$) $t_1[i_1] \in anc(t_1[l(i), i_1], n)$ と仮定する．n' を次の条件を満たす頂点とする．

- $T = op'(t_1)[l(i), i_1 - 1] \cap anc(t_1[l(i), i_1], n)$.
- n' は T における最上の頂点，すなわち，任意の頂点 $n \in T$ に対して n' は n の祖先である．

$op'(t_1)[l(i), i_1 - 1]$ は祖先順序保存，$n' \in op'(t_1)[l(i), i_1 - 1]$，かつ，$n' \in anc(t_1[l(i), i_1 - 1], n)$ なので，$n' \in anc(op'(t_1)[l(i), i_1 - 1], n)$ である．まず，$op'(t_1)[l(i), i_1 - 1]$ において $par(n') = null$ である場合を考える．$n' \in anc(t_1[l(i), i_1], n)$ かつ $t_1[i_1] \in anc(t_1[l(i), i_1], n)$ なので，$t_1[i_1] \in anc(t_1[l(i), i_1], n')$ である．したがって，add の (1a) において $par(n')$ に $t_1[i_1]$ が代入される．$par(n') = t_1[i_1]$ かつ $n' \in anc(op'(t_1)[l(i), i_1 - 1], n)$ なので，$t_1[i_1] \in anc(op(t_1)[l(i), i_1], n)$ である．次に，

$par(n') \neq null$ と仮定する．ある $n_2 \in op'(t_1)[l(i), i_1 - 1]$ に対して $par(n') = n_2$ である．$n_3 \in op'(t_1)[l(i), i_1 - 1]$ を，$n_3 \in anc(op'(t_1)[l(i), i_1 - 1], n_2)$ かつ $par(n_3) = null$ を満たす頂点とする．ここで，n' は T において最上なので，$n_2, n_3 \notin t_1$ である．$n_3 \notin t_1$ より，add の (2a) よりある $l(i) \leq i' \leq i_1$ に対して $mark(n_3) = t_1[i']$ である．$mark(n_3) = t_1[i']$ かつ $l(i) \leq i' \leq i_1$ なので，add の定義から $t_1[i_1] \in anc(t_1, t_1[i'])$ である．$t_1[i_1] \in anc(t_1, t_1[i'])$ かつ $par(n_3) = null$ なので，ステップ (1a) において $par(n_3)$ に $t_1[i_1]$ が代入される．$n_3 \in anc(op'(t_1)[l(i), i_1 - 1], n_2)$，$n_2 \in anc(op'(t_1)[l(i), i_1 - 1], n)$，かつ，$par(n_3) = t_1[i_1]$ なので，$t_1[i_1] \in anc(op(t_1)[l(i), i_1], n)$ である．

$t_1[i_1] \notin t_2$ なので，$op(t_2)[l(j), j_1]$ は明らかに祖先順序保存である．次に，$op(t_1)[l(i), i_1]$ が兄弟順序保存であることを示す（$op(t_2)[l(j), j_1]$ も同様）．$m_1, m_2 \in t_1[l(i), i_1] \cap op(t_1)[l(i), i_1]$ を互いに異なる頂点とする．更に，$n_1 \in op(t_1)[l(i), i_1]$ を $n_1 \in anc(op(t_1)[l(i), i_1], m_1)$ かつ $par(n_1) = t_1[i_1]$ を満たす頂点とする．同様に，$n_2 \in op(t_1)[l(i), i_1]$ を $n_2 \in anc(op(t_1)[l(i), i_1], m_2)$ かつ $par(n_2) = t_1[i_1]$ を満たす頂点とする（図 6 を参照）．もし $n_1 = n_2$ ならば，$op'(t_1)[l(i), i_1 - 1]$ は兄弟順序保存なので，$t_1[l(i), i_1]$ において m_1 が m_2 の左に位置することと $op(t_1)[l(i), i_1]$ において m_1 が m_2 の左に位置することは同値である．ここで，$n_1 \neq n_2$ と仮定する．以下，$t_1[l(i), i_1]$ において m_1 が m_2 の左に位置することと $op(t_1)[l(i), i_1]$ において m_1 が m_2 の左に位置することが同値であることを示す．

($\Leftarrow$) $op(t_1)[l(i), i_1]$ において m_1 が m_2 の左に位置すると仮定する．このとき，定義から n_1 は n_2 の兄である．したがって，$ap(n_1) < ap(n_2)$ である．$union$ の定義から，n_2 の任意の子孫 $n_2' \in op(t_1)[l(i), i_1]$ と n_1 の任意の子孫 $n_1' \in op(t_1)[l(i), i_1]$ に対して，$ap(n_1') < ap(n_2')$ が成り立つ．よって，$ap(m_1) < ap(m_2)$ である．i' と j' を，$t_1[i'] = m_1$ かつ $t_2[j'] = m_2$ なる整数とする．このとき，$i' < j'$ が成り立つ．$op'(t_1)[l(i), i_1 - 1]$ は祖先順序保存なので，$m_1 \notin anc(t_1[l(i), i_1], m_2)$ かつ $m_2 \notin anc(t_1[l(i), i_1], m_1)$ である．$i' < j'$，$m_1 \notin anc(t_1[l(i), i_1], m_2)$，かつ，$m_2 \notin anc(t_1[l(i), i_1], m_1)$ なので，定義から $t_1[l(i), i_1]$ において m_1 は m_2 の左に位置する．

($\Rightarrow$) $t_1[l(i), i_1]$ において m_1 が m_2 の左に位置すると仮定する．このとき，$t_1[i'] = m_1$ かつ $t_2[j'] = m_2$ なる i', j' に対して $i' < j'$ であり，よって $ap(m_1) < ap(m_2)$ である．$union$ の定義から，m_1 の任意の祖先 $n \in op'(t_1)[l(i), i_1 - 1]$ と m_2 の任意の祖先 $n' \in op'(t_1)[l(i), i_1 - 1]$ に対して，$ap(n) < ap(n')$ である．したがって，$ap(n_1) < ap(n_2)$ である．$ap(n_1) < ap(n_2)$ なので，定義から $op(t_1)[l(i), i]$

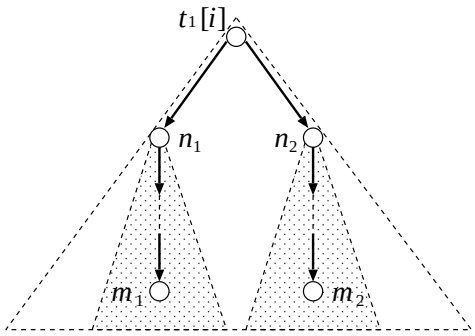

図 6: $op(t_1)[l(i), i_1]$ における頂点 n_1, n_2, m_1, m_2

において m_1 は m_2 の左に位置する．

(2) の場合: (a) $op = op' + del(t_2, t_2[j_1])$ と (b) $op = op' + del(t_1, t_1[i_1])$ の場合に分けて考える．(a) の場合について示す（(b) の場合も同様）．$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は導出対かつ $op = op' + del(t_2, t_2[j_1])$ なので，$(op'(t_1)[l(i), i_1], op'(t_2)[l(j), j_1 - 1])$ は導出対である．帰納法の仮定から，$op'(t_1)[l(i), i_1]$ と $op'(t_2)[l(j), j_1 - 1]$ は祖先・兄弟順序保存である．更に，定義から $op'(t_1)[l(i), i_1]$ と $op'(t_2)[l(j), j_1 - 1]$ はそれぞれ $op(t_1)[l(i), i_1]$ と $op(t_2)[l(j), j_1]$ と等しい．したがって，$op(t_1)[l(i), i_1]$ と $op(t_2)[l(j), j_1]$ は祖先・兄弟順序保存である．

(3) の場合: $l(i) = l(i_1)$，$l(j) = l(j_1)$，かつ，$op = op' + ren(t_1[i_1], t_2[j_1])$ である．$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は導出対かつ $op = op' + ren(t_1[i_1], t_2[j_1])$ なので，$(op'(t_1)[l(i), i_1 - 1], op'(t_2)[l(j), j_1 - 1])$ は導出対である．したがって，帰納法の仮定から $op'(t_1)[l(i), i_1 - 1]$ と $op'(t_2)[l(j), j_1 - 1]$ は祖先・兄弟順序保存である．以下，$op(t_1)[l(i), i_1]$ が祖先順序保存であることを示す．まず，次の条件が成り立つことを示す．

- $par(n) = null$，かつ，$t_1[i_1] \in anc(t_1, mark(n))$ を満たす任意の $n \in op'(t_1)[l(i), i_1 - 1]$ に対して，$par(n)$ に $t_1[i_1]$ が代入される．

$n \in op'(t_1)[l(i), i_1 - 1]$ を $par(n) = null$ かつ $t_1[i_1] \in anc(t_1, mark(n))$ を満たす頂点とする．$(op'(t_1)[l(i), i_1 - 1], op'(t_2)[l(j), j_1 - 1])$ は導出対なので，補題 1 より $op'(t_1)[l(i), i_1 - 1]$ と $op'(t_2)[l(j), j_1 - 1]$ は同型である．$n' \in op'(t_2)[l(j), j_1 - 1]$ を n に対応する頂点とする．このとき，$par(n) = null$ かつ $op'(t_1)[l(i), i_1 - 1]$ と $op'(t_2)[l(j), j_1 - 1]$ は同型なので，$par(n') = null$ である．更に，$mark(n) \neq null$ と $n' \in op'(t_2)[l(j), j_1 - 1]$ より $n' \in t_2[l(j), j_1 - 1]$ である．$n' \in t_2[l(j), j_1 - 1]$ かつ $l(j) = l(j_1)$ より $t_2[j_1] \in anc(t_2[l(j), j_1], n')$ である．$par(n') = null$ かつ $t_2[j_1] \in anc(t_2[l(j), j_1], n')$ なので，

ステップ (3a) から $par(n')$ に $t_2[j_1]$ が代入される．したがって，ステップ (3b) より $par(n)$ に $t_1[i_1]$ が代入される．よって，上記条件が成り立つ．このとき，(1) の場合と同様にして $op(t_1)[l(i), i_1]$ が祖先順序保存であることが示せる．最後に，兄弟順序保存に関する証明は (1) の場合と同様である．

(4) の場合: $op = op_1 + op_2$ である．$|op_1| < |op|$ かつ $|op_2| < |op|$ なので，帰納法の仮定から $op_1(t_1)[l(i), l(i_1) - 1]$ と $op_1(t_2)[l(j), l(j_1) - 1]$ は祖先・兄弟順序保存である．同様に，$op_2(t_1)[l(i_1), i_1]$ と $op_2(t_2)[l(j_1), j_1]$ は祖先・兄弟順序保存である．ここで，op_2 の構成における各操作は，$(op_1(t_1)[l(i), l(i_1) - 1], op_1(t_2)[l(j), l(j_1) - 1])$ の構造に影響を与えない．よって，$(op_2(t_1)[l(i_1), i_1], op_2(t_2)[l(j_1), j_1])$ の構成の前後において，$(op_1(t_1)[l(i), l(i_1) - 1], op_1(t_2)[l(j), l(j_1) - 1])$ の構造は変化しない．したがって，$op(t_1)[l(i), i_1]$ と $op(t_2)[l(j), j_1]$ は祖先・兄弟順序保存である． □

次に，導出対に関する最小コストについて，2 つの補題を示す．補題 3 の 1. は Cost の 1 行目，2. は 3 行目，3. は 5 行目に対応する．

補題 3: $t_1[i] \in anc(t_1, t_1[i_1])$ と $t_2[j] \in anc(t_2, t_2[j_1])$ を頂点とする．次の条件が成り立つ．

1. $c_m(t_1[l(i), l(i) - 1], t_2[l(j), l(j) - 1]) = 0$.
2. もし $i_1 \geq l(i)$ かつ $j_1 < l(j)$ ならば，

$$\begin{aligned} &c_m(t_1[l(i), i_1], t_2[l(j), j_1]) \\ &= c_m(t_1[l(i), i_1 - 1], t_2[l(j), j_1]) \\ &\quad + \min\{\gamma(add(t_2, t_1[i_1])), \gamma(del(t_1, t_1[i_1]))\}. \end{aligned}$$

3. もし $j_1 \geq l(j)$ かつ $i_1 < l(i)$ ならば，

$$\begin{aligned} &c_m(t_1[l(i), i_1], t_2[l(j), j_1]) \\ &= c_m(t_1[l(i), i_1], t_2[l(j_1), j_1 - 1]) \\ &\quad + \min\{\gamma(add(t_1, t_2[j_1])), \gamma(del(t_2, t_2[j_1]))\}. \end{aligned}$$

証明: $t_1[l(i), l(i) - 1] = t_2[l(j), l(j) - 1] = \emptyset$ なので，定義から条件 (1) が成り立つ．条件 (2) が成り立つことを示す (条件 (3) も同様)．$i_1 \geq l(i)$ かつ $j_1 < l(j)$ と仮定する．このとき，$t_1[l(i), i_1] \neq \emptyset$ かつ $t_2[l(j), j_1] = \emptyset$ である．op を $(t_1[l(i), i_1], t_2[l(j), j_1])$ に対する最適操作系列とする．$op(t_2)[l(j), j_1]$ が $t_1[i_1]$ に対応する頂点 n を含む場合とそうでない場合に分けて考える．まず，前者の場合を考える． n は $t_1[i_1]$ に対応しかつ $t_2[l(j), j_1] = \emptyset$ なので，ある操作系列 op' に対して $op = op' + add(t_2, t_1[i_1])$ である．$(op(t_1)[l(i), i_1], op(t_2)[l(j), j_1])$ は導出対かつ $op = op' + add(t_2, t_1[i_1])$ なので，定義から $(op'(t_1)[l(i), i_1 - 1], op'(t_2)[l(j), j_1])$ は導出対である．更に，この導出対は補題 1 と補題 2 より妥当である．また，op は最適なので，op' も $(t_1[l(i), i_1 - 1], t_2[l(j), j_1])$ に関して最適，すなわち，$\gamma(op') = c_m(t_1[l(i), i_1 - 1], t_2[l(j), j_1])$ である．したがって，定義から以下が成り立つ．

$$\begin{aligned} &c_m(t_1[l(i), i_1], t_2[l(j), j_1]) \qquad (3.1) \\ &= c_m(t_1[l(i), i_1 - 1], t_2[l(j), j_1]) + \gamma(add(t_2, t_1[i_1])). \end{aligned}$$

次に，後者の場合を考える．$op(t_2)[l(j), j_1]$ は $t_1[i_1]$ に対応する頂点を含まないので，ある操作系列 op' に対して $op = op' + del(t_1, t_1[i_1])$ である．以下，前者の場合と同様にして次式が示せる．

$$\begin{aligned} &c_m(t_1[l(i), i_1], t_2[l(j), j_1]) \qquad (3.2) \\ &= c_m(t_1[l(i), i_1 - 1], t_2[l(j), j_1]) + \gamma(del(t_1, t_1[i_1])). \end{aligned}$$

(3.1) と (3.2) より条件 (2) が成り立つ． □

最後に，以下の補題が成り立つ (証明は補題 3 と同様)．この補題の前半の条件は Cost の 9 行目，後半の条件が 11 行目に対応する．

補題 4: $t_1[i] \in anc(t_1, t_1[i_1])$ と $t_2[j] \in anc(t_2, t_2[j_1])$ を $i_1 \geq l(i)$ と $j_1 \geq l(j)$ を満たす頂点とする．もし $l(i) = l(i_1)$ かつ $l(j) = l(j_1)$ ならば，次式が成り立つ．

$$\begin{aligned} &c_m(t_1[l(i), i_1], t_2[l(j), j_1]) = \min\{ \\ &\quad c_m(t_1[l(i), i_1 - 1], t_2[l(j), j_1]) + \gamma(add(t_2, t_1[i_1])), \\ &\quad c_m(t_1[l(i), i_1], t_2[l(j), j_1 - 1]) + \gamma(add(t_1, t_2[j_1])), \\ &\quad c_m(t_1[l(i), i_1], t_2[l(j), j_1 - 1]) + \gamma(del(t_2, t_2[j_1])), \\ &\quad c_m(t_1[l(i), i_1 - 1], t_2[l(j), j_1]) + \gamma(del(t_1, t_1[i_1])), \\ &\quad c_m(t_1[l(i), i_1 - 1], t_2[l(j), j_1 - 1]) \\ &\quad + \gamma(ren(t_1[i_1], t_2[j_1]))\}, \end{aligned}$$

そうでない場合，次式が成り立つ．

$$\begin{aligned} &c_m(t_1[l(i), i_1], t_2[l(j), j_1]) = \min\{ \\ &\quad c_m(t_1[l(i), i_1], t_2[l(j), j_1 - 1]) + \gamma(del(t_2, t_2[j_1])), \\ &\quad c_m(t_1[l(i), i_1 - 1], t_2[l(j), j_1]) + \gamma(del(t_1, t_1[i_1])), \\ &\quad c_m(t_1[l(i), l(i_1) - 1], t_2[l(j), l(j_1) - 1]) \\ &\quad + c_m(t_1[l(i_1), i_1], t_2[l(j_1), j_1])\}. \end{aligned}$$

□

以上の 4 補題から，次の定理が成り立つ．

定理 1: アルゴリズム 1 は正当である．すなわち，任意の木 t_1, t_2 に対して，アルゴリズム 1 の出力と $c_m(t_1, t_2)$ は一致する． □

4. 任意の個数の木を変換するための最小コスト

前節のアルゴリズムを拡張することにより，n 個の木 ($n \geq 2$) を同じ構造に変換するための最小コストを求めるアルゴリズムを構成できる (紙数の都合で，定義の詳細は省略する)．n 個の森 $t_1[l(i_1), j_1], \cdots, t_n[l(i_n), j_n]$ を同じ構造に変換する最小コストを $c_m(t_1[l(i_1), j_1], \cdots, t_n[l(i_n), j_n])$ と表す．n 個の木 $t_1, \cdots, t_n$ に対して $c_m(t_1, \cdots, t_n)$ を求めるアルゴリズムを示す (アルゴリズム 3)．2 行目は $i_1, \cdots, i_n$ に関する n 個の (ネストした) for 文を略記したものである．4 行目と 6 行目の Cost2 は $c_m(t_1[l(k_1), k_1], \cdots, t_n[l(k_n), k_n])$ を計算する (アルゴリズム 4)．Cost2 において，7 行目は $j_1, \cdots, j_n$ に関する n 個の for 文を略記したものである．9 行目と 15 行目は del ($t_k[j_k]$ の削除)，16 行目は $union$ のコストを求めている．10 行目において，$R_k = e_1 + \cdots + e_n$ に関して次の条件が成り立つことを $ar(R_k)$ と表す．

条件: $l \neq k$ なる任意の $1 \leq l \leq n$ に対して，(i) $e_l = ren(t_l[j_l], t_k[j_k])$ かつ $j'_l = j_l - 1 \geq l(i_l) - 1$，または，(ii) $e_l = add(t_l, t_k[j_k])$ かつ $j'_l = j_l$，である．

10 行目は「$t_k[j_k]$ を t_k に残すという前提で，各 $1 \leq l \leq n$ ($l \neq k$) に対して $add(t_l, t_k[j_k])$ または $ren(t_l[j_l], t_k[j_k])$ のいずれかを行い，$t_1[j_1], \cdots, t_n[j_n]$ の同型性を維持する」という操作のコストを求めている．

アルゴリズム 3: $c_m(t_1, \cdots, t_n)$ の計算

Input: 木 $t_1, \cdots, t_n$.
Output: 変換コスト $c_m(t_1, \cdots, t_n)$.
begin
1. Compute $keyroots(t_1), \cdots, keyroots(t_n)$.
2. **for** $(i_1, \cdots, i_n) \leftarrow (1, 1, \cdots, 1)$ **to** $(|keyroots(t_1)|, \cdots, |keyroots(t_n)|)$ **do**
3. Let k_j be the i_jth item in $keyroots(t_j)$ $(1 \leq j \leq n)$.
4. Cost2$(t_1[l(k_1), k_1], \cdots, t_n[l(k_n), k_n])$;
5. **end**
6. **return** Cost2$(t_1[1, |t_1|], \cdots, t_n[1, |t_n|])$;
end

アルゴリズム 4: Cost2

Input: 木 $t_1[l(i_1), i_1], \cdots, t_n[l(i_n), i_n]$.
Output: 変換コスト $c_l(t_1[l(i_1), i_1], \cdots, t_n[l(i_n), i_n])$.
begin
1. $c_l(t_1[l(i_1), l(i_1) - 1], \cdots, t_n[l(i_n), l(i_n) - 1]) \leftarrow 0$;
2. **for** $k \leftarrow 1$ **to** n **do**
3. $j_l \leftarrow l(i_l) - 1$ for $1 \leq l \leq n$;
4. **for** $j_k \leftarrow l(i_k)$ **to** i_k **do**
5. $c_l(t_1[l(i_1), j_1], \cdots, t_k[l(i_k), j_k], \cdots, t_n[l(i_n), j_n])$
$\leftarrow c_l(t_1[l(i_1), j_1], \cdots, t_k[l(i_k), j_k - 1], \cdots, t_n[l(i_n), j_n])$
6. $+ \min\{\sum_{1 \leq l \leq n, l \neq k} add(t_l, t_k[j_k]), del(t_k, t_k[j_k])\}$;
7. **for** $(j_1, \cdots, j_n) \leftarrow (l(i_1) - 1, \cdots, l(i_n) - 1)$ **to** $(i_1, \cdots, i_n)$ **do**
8. **if** $l(j_1) = l(i_1) \wedge \cdots \wedge l(j_n) = l(i_n)$ **then**
9. $D \leftarrow \{c_l(t_1[l(i_1), j_1], \cdots, t_k[l(i_k), j_k - 1], \cdots, t_n[l(i_n), j_n])$
$+ \gamma(del(t_k, t_k[j_k])) \mid 1 \leq k \leq n\}$;
10. $A \leftarrow \{c_l(t_1[l(i_1), j'_1], \cdots, t_k[l(i_k), j_k - 1], \cdots, t_n[l(i_n), j'_n])$
$+ \gamma(R_k) \mid ar(R_k), 1 \leq k \leq n\}$;
11. $c_l(t_1[l(i_1), j_1], \cdots, t_n[l(i_n), j_n]) \leftarrow \min(D \cup A)$;
12. $c_g(t_1[l(i_1), j_1], \cdots, t_n[l(i_n), j_n])$
$\leftarrow c_l(t_1[l(i_1), j_1], \cdots, t_n[l(i_n), j_n])$;
13. **end**
14. **else**
15. $D \leftarrow \{c_l(t_1[l(i_1), j_1], \cdots, t_k[l(i_k), j_k - 1], \cdots, t_n[l(i_n), j_n])$
$+ \gamma(del(t_k, t_k[j_k])) \mid 1 \leq k \leq n\}$;
16. $U \leftarrow c_l(t_1[l(i_1), l(j_1) - 1], \cdots, t_n[l(i_n), l(j_n) - 1])$
$+ c_g(t_1[l(j_1), j_1], \cdots, t_n[l(j_n), j_n])$;
17. $c_l(t_1[l(i_1), j_1], \cdots, t_n[l(i_n), j_n]) \leftarrow \min(D \cup \{U\})$;
18. **end**
19. **return** $c_l(t_1[l(i_1), i_1], \cdots, t_n[l(i_n), i_n])$;
end

Cost2 の 7 行目から，アルゴリズム 3 の実行時間は n に関して指数的に増大する．この時間を多項式に低減することは，以下の定理から困難である．

定理 2: n 個の順序木に対する最適な操作系列を求める問題は NP 困難である．

略証: SCS 最適化問題 [3] からの帰着により示せる． □

5. むすび

本稿では，構造の異なる文書データを同じ構造に変換するアルゴリズムについて考察した．今後の課題として，次のような事項が挙げられる．

- 任意の個数の文書データを同じ構造に変換する多項式時間近似アルゴリズムを開発する．
- 本稿のアルゴリズムや上記近似アルゴリズムに関する評価実験を行う．

参考文献

[1] S. Chawathe, A. Rajaraman, H. Gracia-Molina, and J. Widom, "Change Detection in Hierarchically Structured Information," Proc. ACM SIGMOD, pp.493–504, 1996.

[2] A. Gupta and N. Nishimura, "Finding Largest Subtrees and Smallest Supertrees," Algorithmica, Vol.21, pp.183–210, 1998.

[3] D. Maier, "The Complexity of Some Problems on Subsequences and Supersequences," J. ACM, Vol.25, No.2, pp.322–336, 1978.

[4] A. Yamaguchi, K. Nakano, and S. Miyano, "An Approximation Algorithm for the Minimum Common Supertree Problem," Nordic Journal of Computing, Vol.4, pp.303–316, 1997.

[5] K. Zhang and D. Shasha, "Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems," SIAM J. Comput., Vol.18, No.6, pp.1245–1262, 1989.

[6] 田島敬史, "XML のための検索／操作言語," bit, Vol.33, No.4, pp.87–93, 共立出版, 2001.