國立交通大學

資訊科學與工程研究所

碩 士 論 文

一個新的樹編輯距離之演算法

An Algorithm for A New Tree Editing Distance Problem

研 究 生：許伯鈞

指導教授：吳毅成　教授

中 華 民 國 九 十 五 年 一 月

一個新的樹編輯距離之演算法

An Algorithm for A New Tree Editing Distance Problem

研 究 生：許伯鈞　　Student：Po-Chun Hsu

指導教授：吳毅成　　Advisor：I-Chen Wu

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

January 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年一月

一個新的樹編輯距離之演算法

研究生：許伯鈞　　指導教授：吳毅成

國立交通大學 資訊工程學系

摘要

在 1989 年，Zhang 和 Shasha 提出了傳統型樹編輯距離的演算法，其時間複雜度為 $O(|S||T|\min(L_S,D_S)\min(L_T,D_T))$，其中 $|S|$ 和 $|T|$ 分別為第一、第二顆樹的節點個數，L_S 和 D_S 為第一顆樹的葉節點個數和第一顆樹的深度，L_T 和 D_T 為第二顆樹的葉節點個數和第二顆樹的深度。在 1995 年 Zhang 提出了限制型樹編輯距離的演算法，其時間複雜度為 $O(|S|\times|T|)$。本篇論文提出了一個混合型編輯距離的問題。這樣的一個編輯距離的規範可應用於 XML 文件的比對、HTML/XHTML 文件的比對、圖樣的辨識等應用。我們也提出了一個時間為 $O(d_S^{max}\times d_T^{max}\times|S|\times|T|)$ 的演算法來解決此問題，其中 d_S^{max} 為樹 S 中所有不符合限制型節點之路徑的最大值，d_T^{max} 為樹 T 中所有不符合限制型節點之路徑的最大值。

An Algorithm for A New Tree Editing Distance Problem

Student: Po-Chun Hsu Advisor: I-Chen Wu

Department of Computer Science and Information Engineering

National Chiao Tung University

ABSTRACT

In 1989, Zhang and Shasha proposed an algorithm for general tree editing problem in time $O(|S||T|\min(L_S,D_S)\min(L_T,D_T))$, where $|S|$ and $|T|$ are the node number of tree S and tree T, L_S is the number of tree S's leaves, D_S is the depth of tree S, L_T is the number of tree T's leaves, and D_T is the depth of tree T. In 1995, Zhang proposed an algorithm for constrained editing problem in time $O(|S|\times|T|)$. This paper considers the problem of computing the hybrid editing distance between two hybrid ordered label trees. This problem can be applied to XML change detection, HTML/XHTML change detection, pattern recognition and other applications. We present an algorithm for solving the problem in time $O(d_S^{max} \times d_T^{max} \times |S| \times |T|)$, where d_S^{max} is the maximum depth of all paths with nodes excluding these with constrained requirements of tree S and d_T^{max} is the maximum depth of all paths with nodes excluding these with constrained requirements of tree T.

誌謝

在這段作研究的過程中，非常感謝神在我的週圍安排了許多的貴人來相助。

感謝我的父母，讓我能衣食無缺的，沒有後顧之憂的作研究，我要說，「媽，您的每封信、每個簡訊，我都收到了…」，「爸，我都有正常的吃水果跟運動…」

感謝我的指導老師，吳毅成博士，總是好像對我的畢業比我還要著急，在完成在我看來是「不可能的任務」的研究，您的指導總是適時而且及時。

感謝陳隆彬學長，在台中的四個月麻煩您照顧了。

感謝蘇瑞元學長，總是很有耐心，詳細地解決我的大小不同的問題。

感謝林秉宏學長，在獨力作戰的同時，至少還有可以一起討論的同伴。

感謝汪益賢、徐健智學長，在研究之餘，解決我一些無厘頭的問題。

感謝專班的同學益洲、一芳，雖然我們都不知道這段過程什麼時候會結束，但是很高興，到最後我們都沒有放棄，並且我們也到達成功那一端了。

感謝學妹韻筑、宜融剛進實驗室就接受學長們的荼毒，辛苦你們了。

感謝兩年多來一起同住的學弟們，雖然你們不是很清楚我作的是什麼研究，但是能跟你們同住，著實讓我難忘，在弟兄之家的日子，謝謝有你們的陪伴。

感謝 6-4 區的弟兄姊妹們，這半年來，我能夠在這裡接受大家的照顧和牧養，真的很幸福，也謝謝你們是如此迫切的為我代禱。

感謝材料系的吳耀銓教授，這兩年來的關心，我銘記在心，很喜樂從頭到尾每一次的交通，都叫我身心靈得著滿足。

最後，要感謝每一個過程一直陪著我的洪盈潔姊妹，謝謝妳能接受最高的我，也能包容最低的我，最後這半年，我們一起的禱告與交通，那主來的時候，都算的數的。

感謝神，真的是憑著信，走到這一步了，所有的歡笑、淚水，能在你們的見證下經過，真的很感謝。

特別是最後半年，真是更多經歷人的「有限」與神的「無限 」。

真的，要感謝的人實在太多了，我能見證，神的愛的確藉著你們洋溢出來了。

願這一切的榮耀都歸給我所信所愛的神。

目錄

圖表目錄

第一章 緒論

隨著網路的蓬勃發展數以萬計的網站林立，所產生的網頁更是不可勝數。近年來企業間彼此訊息的交換及 web service 的盛行，也使得 XML 大量的被應用。再加上資訊生物研究的興起，上述所提，無論是網頁、是 XML、資訊生物研究裡 RNA 的第二階皆可視為有序的標籤含根樹。也因為資料量的增加十分快速，也使得樹編輯距離的需要也漸漸增多。

在傳統的問題中，最早在 1976 年由 WANGER 所提出的字串編輯距離問題。後來在[Se177]中由字串編輯距離問題提出有限度的樹編輯距離問題(只能在葉節點進行刪除、插入)。之後由[Tai79]正式提出樹編輯距離問題，但由於演算法複雜而且所花時間甚巨，達到 $O(|S||T|D_S^2D_T^2)$，其中 $|S|$ 和 $|T|$ 分別為兩顆樹的節點個數，D_S 和 D_T 分別為兩顆樹的深度。十年後[ZS89]提出快速且簡單的方法，其時間複雜度為 $O(|S||T|\min(L_S,D_S)\min(L_T,D_T))$，其中 L_S 和 L_T 分別兩顆樹的葉節點個數。考量到樹本身結構限制的問題，[Zha95]和[Ric97]分別提出時間複雜度為 $O(|S|\times|T|)$、$O(|S||T|I_SI_T)$，其中 I_S 和 I_T 分別為為兩顆樹的分支度，限制的型的樹編輯距離問題(後來的研究已証明著二者為等價)。本論文提出一個整合[ZS89]及[Zha95]的新演算法，時間複雜度為 $O(d_S^{max}\times d_T^{max}\times|S|\times|T|)$，其中 d_S^{max} 為樹 S 中所有不符合限制型節點之路徑的最大值，d_T^{max} 為樹 T 中所有不符合限制型節點之路徑的最大值，更符合實際的編輯動作。

1.1 基本定義

在本論文中，我們使用下列定義。

- 樹(Tree)
  - 為一種資料結構，節點(node)與邊(edge)的集合，其中節點與節點間可以有邊相連，但不可形成循環。

- 含根樹(Rooted Tree)
  - 為一樹，其中有一節點為整顆樹的根節點(root)。
- 有序樹(Ordered Tree)
  - 為一樹，其中每一個節點的子節點有從左到右的順序之分。
- 有序標籤含根樹(Ordered Labeled Rooted Tree)
  - 其中有序及含根的定義如上所述，此外每一個節點都有一個標籤(label)。在本論文中，在不造成混淆的情況下，我們指的樹皆屬此類。

1.2 論文大綱

本篇論文共有五章。在第二章中我們將說明關於樹編輯距離問題兩篇重要的論文，包括他們的演算法及結果。第三章會說明我們的動機與目標。第四章則是我們所提出的混合型編輯距離問題及演算法。第六章為結論及未來展望。最後是參考文獻及附錄。

第二章 背景說明

本章將說明整個樹編輯距離問題及介紹 1989 年由 Zhang 和 Shasha 所提出的演算法和 1995 年由 Zhang 所提出的演算法。

2.1 樹狀結構之編輯距離問題

(Tree Editing Distance Problem)

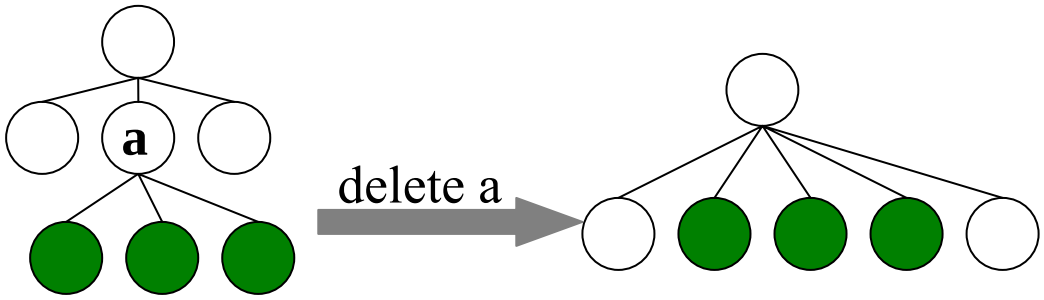
此問題首見於[Tai79]一文，但後述引用均以[ZS89]版本為主。此問題也稱作 Tree-to-Tree Correction Problem。

給定兩顆有序標籤含根樹 S 與 T，並一個對 S、T 的操作集合 OP，欲求在有限個操作之下，將 S 轉換成 T 的最小成本(即最短距離)。傳統上，OP 包含三個操作，刪除一節點(Delete)、插入一節點(Insert)、更換兩節點的標籤(Relabel)。

2.2 基本操作

有三個包括，刪除一節點(Delete)、插入一節點(Insert)、更換兩節點的標籤(Relabel)。下面將以圖示來說明這三個操作的意義。

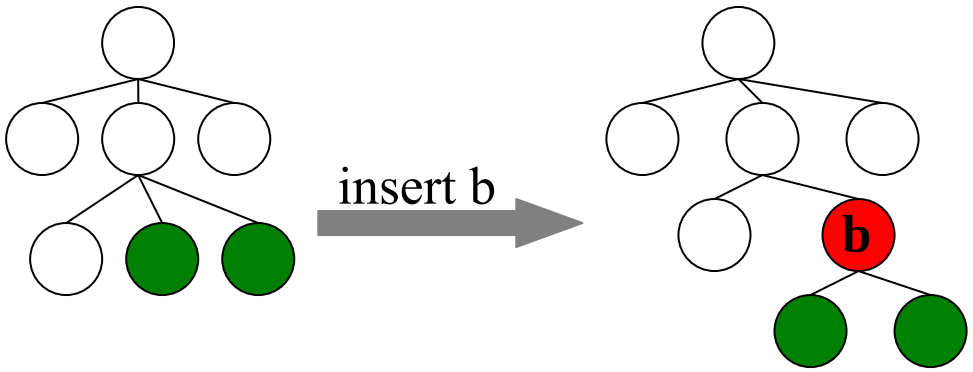
2.2.1 刪除一節點(Delete)



圖表 2.1 刪除一節點

以 a→Λ，表示刪去 a 節點。Λ表示空節點(empty node)。

2.2.2 插入一節點(Insert)



圖表 2.2 插入一節點

以Λ→b，表示插入 b 節點。

2.2.3 更換兩節點的標籤(Relabel)

圖表 2.3 更換一節點標籤爲另一節點標籤

以 a→b，表示將節點 a 的標籤換成節點 b 的標籤。

上述所提的三個操作中，每一個操作都有一成本，分別以成本函式 Cost(a→Λ)、Cost(Λ→b)、Cost(a→b)，代表刪除 a 節點的成本、插入 b 節點的成本、更換 b 節點標籤為 a 節點標籤的成本。成本為一非負的實數。成本函式基本上為使用者定義。下文中將以 γ 表示節點操作的成本函式。

因此，若我們欲將樹 S 轉換成樹 T，需要經過 $e_1e_2e_3e_4\cdots e_n$ 個操作，每

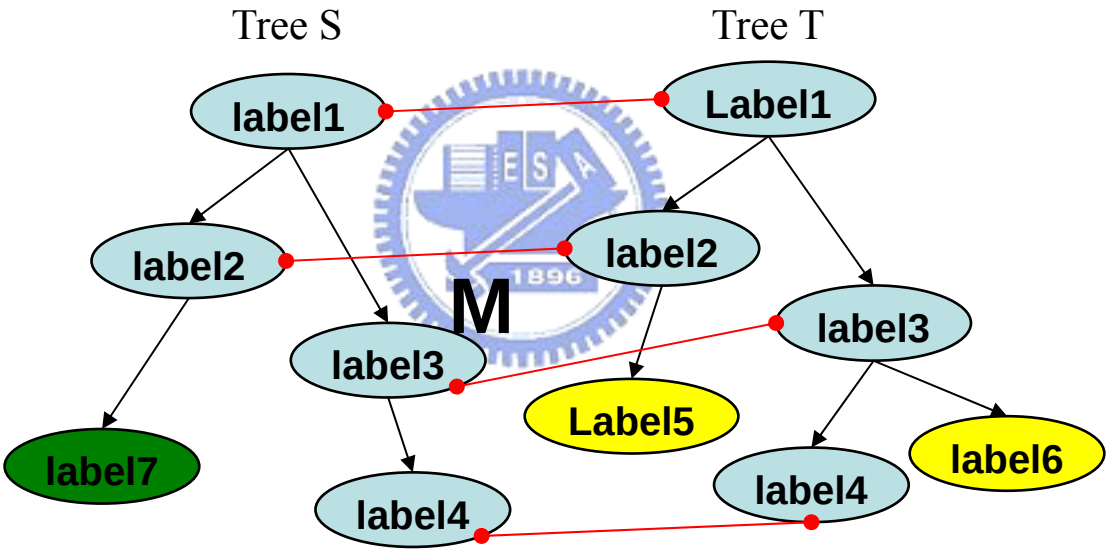
個 e_i (i=1, 2, 3, …)為一次操作。取 E= $e_1e_2e_3e_4\cdots e_n$ 為一連續的操作序列。所以，將 S 經由 E 轉換成 T 總共需要下述成本。

$$\sum \text{Cost(E)} = \sum_{1}^{n} \text{Cost}(e_i) = \text{Cost}(e_1) + \text{Cost}(e_2) + \ldots + \text{Cost}(e_n)$$

由此式便可求將 S 經由 E 轉換成 T 所需要的成本。

2.3 樹的編輯配對(Tree Editing Mapping)

為了求得在 S 跟 T 之間如何轉換，S 經過了多少操作之後轉換成 T，我們在 S 跟 T 之間建立了一個配對的關係。以下圖為例：



圖表 2.4 Tree S 與 Tree T 間的編輯配對示意圖

我們在兩顆樹 S 跟 T 之間建立一個配對，我們稱之為 S 跟 T 的編輯配對，以(M,S,T)表示。

在上述的編輯配對中 Tree S 裡 label7 的節點不在(M,S,T)中，表示被刪除的節點;Tree T 裡 label5 的節點跟 label6 的節點不在(M,S,T)中，表示被插入的節點。因此，我們可以根據所得的編輯配對(M,S,T)來計算，將 S 轉換成 T 所需要的成本為何。令 I 為第一類的節點集合。J 為第三類的節點集合。第二類的點在(M,S,T)中。

(M,S,T)的成本便可用下述式子來表示。

$$\text{Cost}((M,S,T)) = \sum_{i \in I} \gamma(s(i) \to \Lambda) + \sum_{s(i),t(j) \in (M,S,T)} \gamma(s(i) \to t(j)) + \sum_{j \in J} \gamma(\Lambda \to t(j))$$

2.3.1 傳統型編輯配對(General Editing Mapping)

為了要取得上述的操作序列 E，我們定義了一個 S、T 間節點的配對(Mapping)以(M,S,T)表示，其中 S 有|S|個節點，T 有|T|個節點，S 與 T 使用某個已知的排序方法對每個節點進行編號，s[i]表示 S 中的第 i 節點，t[j]表示 T 中的第 j 個節點。

(M,S,T)的定義如下：

$$(M,S,T) = \{(s[i],t[j]) \mid s[i] \in S, t[j] \in T\} = \{(s[i],t[j]) \mid (s[i],t[j]) \in S \times T\}$$

一個合法的傳統型編輯配對(M_G,S,T)，應符合下列規範：

- 一對一配對
  - 配對(M_G,S,T)中任何兩個配對組合$(i_1,j_1),(i_2,j_2)$，若 $i_1=j_1 \Leftrightarrow i_2=j_2$。
- 保留兄弟節點的順序
  - 配對(M_G,S,T)中任何兩個配對組合$(i_1,j_1),(i_2,j_2)$，若i_1節點在j_1的右邊$\Leftrightarrow$ i_2節點在j_2的右邊。
- 保留祖先節點與後代節點的順序
  - 配對(M_G,S,T)中任何兩個配對組合$(i_1,j_1),(i_2,j_2)$，若i_1節點為i_2節點的祖先$\Leftrightarrow$ j_1節點為j_2節點的祖先。

上面三個點中，第一點是確保配對(M_G,S,T)中不會有一對多的情形。第二點是確保配對(M_G,S,T)中會保留同一層節點由左到右的順序。第三點是確保配對(M_G,S,T)中祖先跟後代的關係。以過使用傳統型配對的演算法主要有四篇，表列如下：

論文	時間複雜度
[Tai79]	$O(\|S\|\|T\|D_S^2D_T^2)$
[ZS89]	$O(\|S\|\|T\|\min(L_S,D_S)\min(L_T,D_T))$
[Kle98]	$O(\|S\|^2\|T\|\log\|T\|)$

[Che01]	$O(\|S\|\|T\|+\min(L_S^2\|T\|+L_S^{2.5}L_T, L_T^2\|S\|+L_T^{2.5}L_S))$

圖表 2.5 以過使用傳統型配對的演算法列表

其中 D_S、D_T 分別為樹 S、樹 T 的深度。L_S、L_T 分別為樹 S、樹 T 的葉節點個數。

2.3.2 限制型編輯配對(Constraind Editing Mapping)

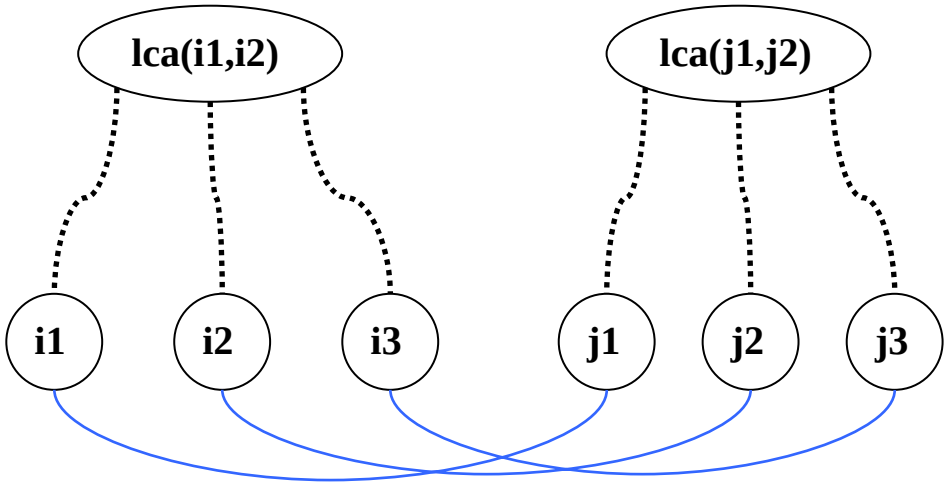
限制型編輯配對與傳統型編輯配對惟一不同的地方在於 S、T 之間配對的限制條件不同。我們定義 (M_C,S,T) 為 S 到 T 的限制型編輯配對。

一個合法的限制型編輯配對 (M_C,S,T)，應符合下列規範：

- 一對一配對
  - 配對 (M_C,S,T) 中任何兩個配對組合 $(i_1,j_1),(i_2,j_2)$，若 $i_1=j_1 \Leftrightarrow i_2=j_2$。
- 保留兄弟節點的順序
  - 配對 (M_C,S,T) 中任何兩個配對組合 $(i_1,j_1),(i_2,j_2)$，若 i_1 節點在 j_1 的右邊 $\Leftrightarrow$ i_2 節點在 j_2 的右邊。
- 保留祖先節點與後代節點的順序
  - 配對 (M_C,S,T) 中任何兩個配對組合 $(i_1,j_1),(i_2,j_2)$，若 i_1 節點為 i_2 節點的祖先 $\Leftrightarrow$ j_1 節點為 j_2 節點的祖先。
- 保留最近的共同祖先的(Least Common Ancestor)關係
  - 配對 (M_C,S,T) 中任何三個配對組合 $(i_1,j_1),(i_2,j_2),(i_3,j_3)$，若 $s[lca(i_1,i_2)]$ 節點為 $s[i_3]$ 節點的祖先 $\Leftrightarrow$ $t[lca(j_1,j_2)]$ 節點為 $t[j_3]$ 節點的祖先。

其中 lca 為一函式，傳入一顆樹的兩個節點，傳回這兩個節點的最近的祖先。上述規範中前三個點，與傳統編輯配對的要求一樣。這也說出，若有一個編輯配對 M_X 為一限制型編輯配對，則也必為傳統型編輯配對。我們將利用下面幾個圖來探究限制型編輯配對的涵義：

根據規範的第四點，已知三組配對 $(s[i_1],t[j_1]),(s[i_2],t[j_2]),(s[i_3],t[j_3])$，和 $lca(s[i_1],s[i_2])$ 及 $lca(t[j_1],t[j_2])$，共六個點，如下圖所示：

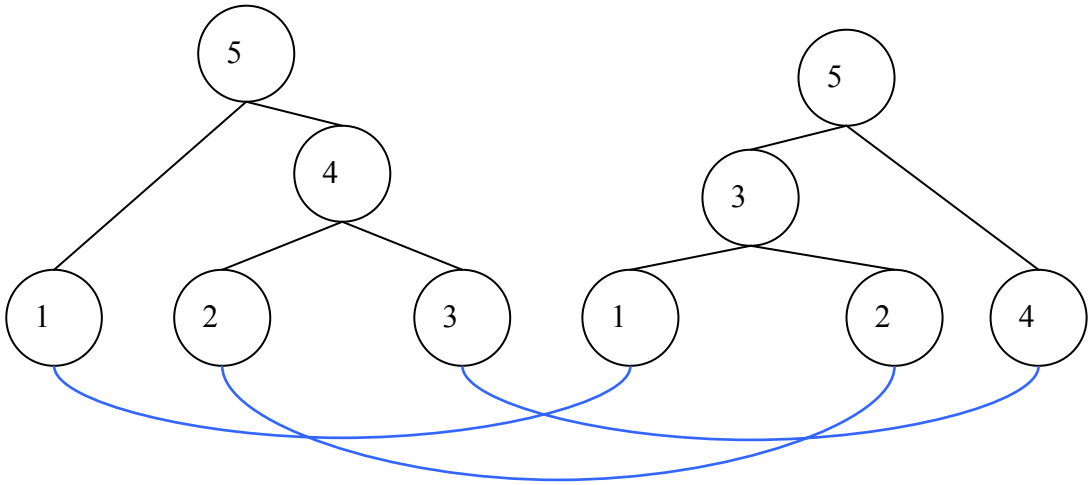


圖表 2.6 限制型編輯配對示意圖

用虛線連接表示中間可能有零個以上的節點。很顯然此圖符合前三點規範，故此圖為一傳統型編輯配對。並且它也符合第四點規範，故為一限制型編輯配對。因此，若一配對為限制型編輯配對，也必為一傳統型編輯配對。

第四點規範的意義主要就是說，它能保證配對一定在同一顆樹(或說子樹)下產生。簡單的說，若 S、T 中，已知有兩組配對 $(s[i_1],t[j_1])$、$(s[i_2],t[j_2])$，則 S 中有節點 $s[i_1]$、$s[i_2]$ 並有 $lca(i_1,i_2)$ 為該子樹的根節點，該子樹以 $S[lca(i_1,i_2)]$ 表示；T 中有節點 $t[j_1]$、$t[j_2]$ 並有 $lca(t[j_1],t[j_2])$ 為該子樹的根節點，該子樹以 $T[lca(j_1,j_2)]$ 表示。則 $S[lca(i_1,i_2)]$、$T[lca(j_1,j_2)]$ 兩子樹間若還存在第三組節點配對 $(s[i_3],t[j_3])$ 的話，則 $s(i_3) \in S[lca(i_1,i_2)]$，$t(j_3) \in T[lca(j_1,j_2)]$。

以下圖為例，若已知有一配對 $M=\{(s[1],t[1]),(s[2],t[2]),(s[3],t[3])\}$，$s[lca(1,2)]=s[5]$ 為 $s[3]$ 的祖先。$t[lca(1,2)]=t[3]$ 不為 $t[4]$ 的祖先。由限制型配對規範得知，不符合第四點規範，故此配對 M 不為限制型編輯配對。然而，此配對滿足配對規範的前三點，故此配對 M 為傳統型編輯配對，但非限制型編輯配對。



圖表 2.7 傳統型編輯配對，非限制型編輯配對

以過使用限制型編輯配對的演算法主要有二篇，表列如下：

論文	時間複雜度
[Zha95]	$O(\|S\|\|T\|)$
[Ric97]	$O(\|S\|\|T\|I_S I_T)$

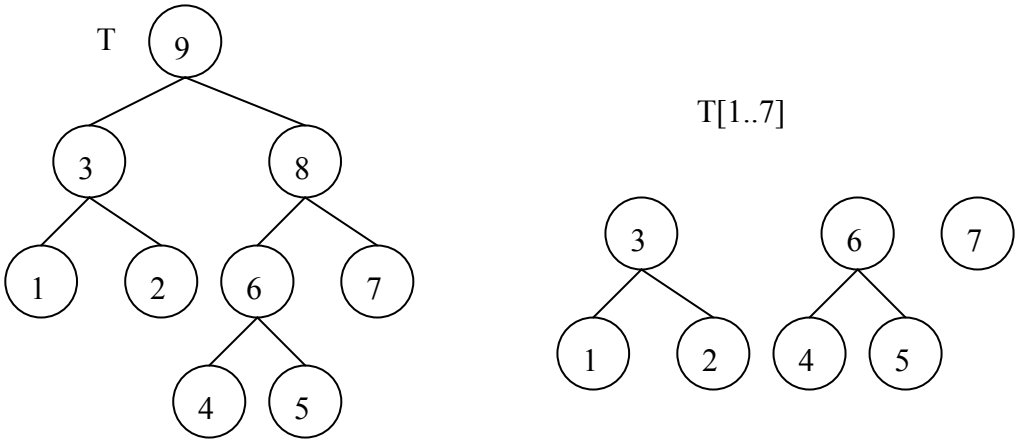
圖表 2.8 以過使用限制型編輯配對的演算法列表

其中 I_S、I_T 分別為 S 與 T 的分支度。

2.4 傳統型樹編輯距距離的演算法

在傳統型成本模型下，最具代表性，也是近年來這類研究引用最多的論文就是 Kaizhong Zhang 和 Denis Shasha 在 1989 年所提的[ZS89]。

首先，我們用後序法來追蹤一顆樹，以後序追蹤的順序來標示樹中的每一個節點。以左下圖為例，為一顆樹 T 用後序追蹤標號後的情形。如下圖所示。

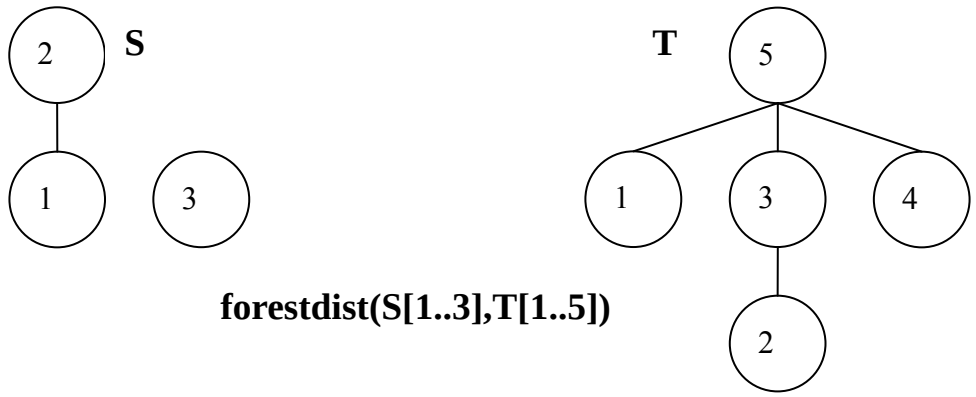


圖表 2.9 使用後序追蹤對樹標號

在上圖右方為 1 到 7 號的節點所形成的 forest，記為 T[1..7]。每一個節點 i 都有一個 lleaf(最左邊的葉節點)，例如，節點 9 的 lleaf 為節點 1，記為 lleaf(9)。節點 8 的 lleaf 為節點 4，記為 lleaf(8)。

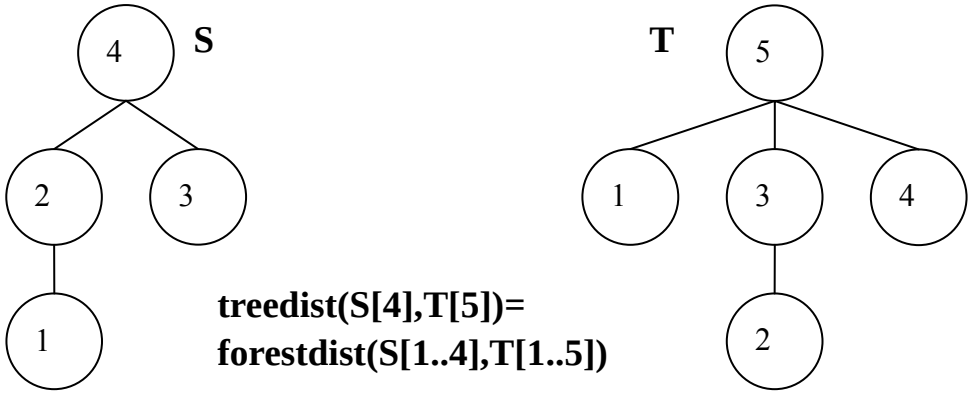
因此，若我們寫 T[lleaf(9)..9]表示 1 到 9 號的節點所形成的 forest，同時也是 Tree[9]所表示的樹。依此類推，Tree[8]也可寫成 T[lleaf(8)..8]或 T[4..8]。另外，樹 S 中的節點 i 可記為 s[i]，樹 T 中的節點 j 可記為 t[j]。節點 i 的父節點記為 p(i)。此外，我們定義 $p(i)^0=i, p(i)^1=p(i), p(i)^2=p(p(i)^1)$ …依此類推。定義 $anc(i)=\{p^k(i)|0 \le k \le depth(i)\}$。

在本演算法中它先求兩個有序 forest 之間的編輯距離進而求兩顆樹之間的編輯距離。我們用 forestdist 函式來表示兩個有序 forest 間的編輯距離，如下圖。



圖表 2.10 forestdist(S[1..3],T[1..5])

用 treedist 函式來表示兩顆樹的編輯距離，如下圖。



圖表 2.11 treedist(S[4],T[5])

本篇論文中，在不造成混淆的狀況下，treedist、forestdist 的第一個參數都表示左邊那個顆樹的節點，第二個參數都表示右邊那個顆樹的節點，因此上述圖中 treedist(S[4],T[5])=forestdist(1..4,1..5)，也可直接表示成 treedist(4,5)=forestdist(S[1..4],T[1..5])。

接下來我們來看 forestdist 的遞迴關系式。首先，forestdist 可以有下列初始化關系

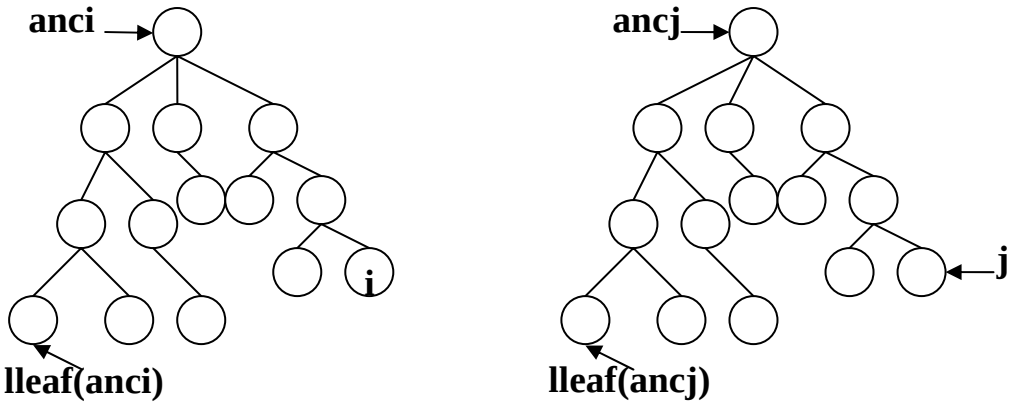
forestdist(θ,θ)=0

forestdist(lleaf(anci)..i,θ)=forestdist(lleaf(anci)..i-1,θ)+$\gamma(i \to \Lambda)$

forestdist(θ,lleaf(ancj)..j)=forestdist(θ,lleaf(ancj)..j-1)+$\gamma(\Lambda \to j)$

其中 anci ∈ anc(i)，ancj ∈ anc(j)。

在 forestdist(lleaf(anci)..i,lleaf(ancj)..j) 中，我們以 s[i]和 t[j]可簡單的分為三種狀況，下列式子中，anci(ancj)表示節點 i(節點 j)的祖先。示意圖如下。



圖表 2.12

狀況一：節點 i 被刪除

forestdist(lleaf(anci)..i-1,lleaf(ancj)..j)+$\gamma(s[i] \to \Lambda)$。

狀況二：插入節點 j

forestdist(lleaf(anci)..i,lleaf(ancj)..j-1)+$\gamma(\Lambda \to t[j])$。

狀況三：節點 i 的標籤換為節點 j 的標籤

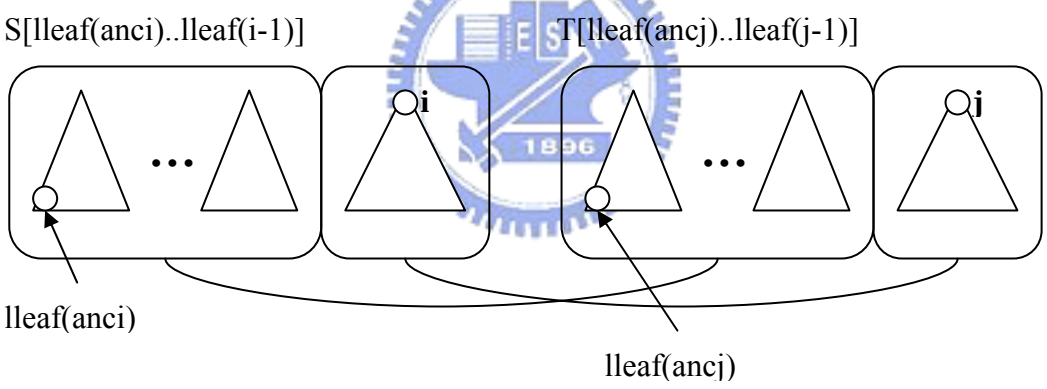
forestdist(lleaf(anci)..i-1,lleaf(ancj)..j-1)+$\gamma(s[i] \to t[j])$。

由上述三種狀況，forestdist 的遞迴關系式如下：

forestdist(lleaf(anci)..i,lleaf(ancj)..j)=

$$\min\begin{Bmatrix} \text{forestdist(lleaf(anci)..i-1,lleaf(ancj)..j)}+\gamma(s[i] \to \Lambda) \\ \text{forestdist(lleaf(anci)..i,lleaf(ancj)..j-1)}+\gamma(\Lambda \to t[j]) \\ \text{forestdist(lleaf(anci)..i-1,lleaf(ancj)..j-1)}+\gamma(s[i] \to t[j]) \end{Bmatrix}$$

另外，在狀況三時需另外考慮一種狀況，就是當 lleaf(i)≠lleaf(anci)或 lleaf(j)≠lleaf(ancj)時，以下圖說明。



圖表 2.13

在此情況下，s[i]和 t[j]本身即為一顆子樹的根節點，因此，若 lleaf(i)為 i'，lleaf(j)為 j'，狀況三又可寫成下列式子：

forestdist(lleaf(anci)..i'-1,lleaf(ancj)..j'-1)+forestdist(i'..i,j'..j)+$\gamma(s[i] \to t[j])$

$\Rightarrow$ forestdist(lleaf(anci)..i'-1,lleaf(ancj)..j'-1)+treedist(i,j)

因此根據上述推導，forestdist 可寫成下列遞迴關係式。

令 i∈anc(i)，j∈anc(j)，則：

狀況一：若 lleaf(i)=lleaf(s) 且 lleaf(j)=lleaf(t)

forestdist(S[lleaf(i)..s],T[lleaf(j)..t])=

$$\min\begin{Bmatrix} \text{forestdist(lleaf(i)..s-1,lleaf(j)..t)}+\gamma(S[s]\rightarrow\Lambda) \\ \text{forestdist(lleaf(i)..s,lleaf(j)..t-1)}+\gamma(\Lambda\rightarrow T[t]) \\ \text{forestdist(lleaf(i)..s-1,lleaf(j)..t-1)}+\gamma(S[s]\rightarrow T[t]) \end{Bmatrix}$$

狀況二：若 lleaf(i) ≠ lleaf(s) 或 lleaf(j) ≠ lleaf(t)

forestdist(S[lleaf(i)..s],T[lleaf(j)..t])=

$$\min\begin{Bmatrix} \text{forestdist(lleaf(i)..s-1,lleaf(j)..t)}+\gamma(S[s]\rightarrow\Lambda), \\ \text{forestdist(lleaf(i)..s,lleaf(j)..t-1)}+\gamma(\Lambda\rightarrow T[t]), \\ \text{forestdist(lleaf(i)..lleaf(s)-1,lleaf(j)..lleaf(t)-1)}+ \\ \text{treedist(s,t)} \end{Bmatrix}$$

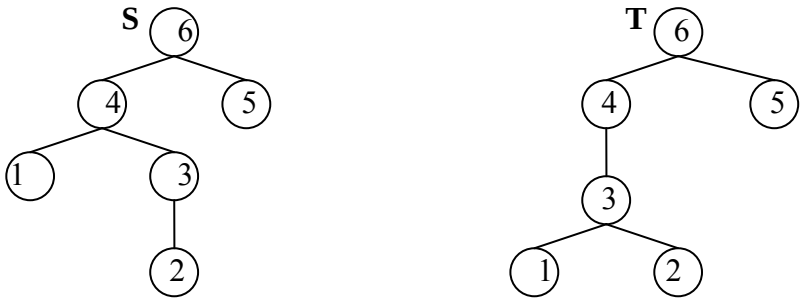
由上述遞迴關系式，我們可得到三項推論。

第一、 我們可使用動態規畫(Dynamic Programming)的方式求解。

第二、 在計算 treedist(i, j)之前必需先求得 treedist(s, t)的值。同時，這也說出我們可以使用下而上(bottom-up)的演算方式求解。

第三、 由上述遞迴關系式中，當 s 在 lleaf(i)到 i 的路徑上，且 t 在 lleaf(j)到 j 的路徑上時，無需計算 treedist(s, t)。因為 treedist(i, j)在計算的時候就會順便求出 treedist(s, t)了。

因此，我們定義樹 S 中這些關鍵的節點為 LR_keyroots。定義如下：
LR_keyroots(S) = {k| exist no k'>k such that lleaf(k)=lleaf(k')}



圖表 2.14 LR_keyroots

在樹 S 中 LR_keyroots(S)={3, 5, 6}；在樹 T 中 LR_keyroots(T)={2, 5, 6}。

主要的演算法便可寫成下列形式：

輸入：樹 S 及樹 T

輸出：treedist(i, j)，其中$1 \le i \le |S|, 1 \le j \le |T|$

```
1  Preprocessing(compute lleaf(),LR_keyroots1,LR_keyroots2)
   LR_keyroots1即 LR_keyroots(S)，LR_keyroots2即LR_keyroots(T)
   main loop
2 for i'=1 to |LR_keyroots(S)|
3     for j'=1 to |LR_keyroots(T)|
4         i= LR_keyroots1[i']
5         j=LR_keyroots2[j']
6         Compute treedist(i,j)
```

在實作上，在 S[i]跟 T[j]中($1 \le i \le |S|, 1 \le j \le |T|$)，我們可以用一個二維陣列 D 來記錄 treedist(i, j)的值。以上圖為例可得下表。

		T[6]						
		0	1	2	3	4	5	6
S[6]	0	0						
	1							
	2							
	3							
	4							
	5							
	6							

其中 D[4,3]表示 S[4]到 T[3]的編輯距離。 D[3,3]表示 S[3]到 T[3]的編輯距離。故 D[6,6]為 S[6]到 T[6]的編輯距離，即為所求。

圖表 2.15

在計算每一個 treedist(i, j)時，每一格的計算需要利用暫時的二維陣列 FD 來計算。例如，當我們要求 treedist(4, 6)也就是填入 D[4, 6]值時。以下圖為例。

		T[6]=T[1..6]						
		0	1..1	1..2	1..3	1..4	1..5	1..6
S[4]= S[1..4]	0	0						
	1..1							
	1..2							
	1..3							
	1..4							

FD[4,6]表示 forestdist(1..4,1..6)，就是 treedist(4,6)，也就是 D[4,6]的值。因為 S[3]的 lleaf 為 S[2]，故 forestdist(1..3,1..4)= forestdist(1..1,0)+treedist(3,4)

圖表 2.16

因此，對每一個 D[i, j]，函式 treedist(i, j)，表列如下。

1 initial step
  main loop
2 for s=lleaf(i) to i
3   for t = lleaf(j) to j
4   if lleaf(s)=lleaf(i) and lleaf(t)=lleaf(j)
5   forestdist(S[lleaf(i)..s],T[lleaf(j)..t])=

$$\min\begin{Bmatrix}\text{forestdist}(\text{lleaf}(i)..s\text{-}1,\text{lleaf}(j)..t)+\gamma(S[s]\to\Lambda)\\ \text{forestdist}(\text{lleaf}(i)..s,\text{lleaf}(j)..t\text{-}1)+\gamma(\Lambda\to T[t])\\ \text{forestdist}(\text{lleaf}(i)..s\text{-}1,\text{lleaf}(j)..t\text{-}1)+\gamma(S[s]\to T[t])\end{Bmatrix}$$

     treedist(s,t) = forestdist(S[lleaf(i)..s],T[lleaf(j)...t])
    else
6   forestdist(S[lleaf(i)..s],T[lleaf(j)..t])=

$$\min\begin{Bmatrix}\text{forestdist}(\text{lleaf}(i)..s\text{-}1,\text{lleaf}(j)..t)+\gamma(S[s]\to\Lambda),\\ \text{forestdist}(\text{lleaf}(i)..s,\text{lleaf}(j)..t\text{-}1)+\gamma(\Lambda\to T[t]),\\ \text{forestdist}(\text{lleaf}(i)..\text{lleaf}(s)\text{-}1,\text{lleaf}(j)..\text{lleaf}(t)\text{-}1)+\\ \text{treedist}(s,t)\end{Bmatrix}$$

圖表 2.17 treedist(i,j)

空間複雜度分析

整個演算法在 treedist 時使用一個固定的二維陣列，在 forestdist 時使用一個暫時的二維陣列。因此，空間複雜度為 O(|S||T|)。

時間複雜度分析

首先，我們知道，每一個 LR_keytoos 都有一個 lleaf，故對一顆樹 S 來說 $|\text{LR\_keyroots}(S)| \le |L_S|$。

在上述推導過程中，我們得知，不是每一個子樹對子樹的編輯距離都需要計算，因此，一個節點在參與演算法計算的程度由節點的深度決定。我們對樹 X 中的節點 i 可定義節點的截斷深度(collapse depth)如下：

$$LR\_colldephth(i)=|anc(i)\cap LR\_keyroots(S)|$$

則樹 X 的 collapse depth 可用下式表示。

$$LR\_colldephth(X)=\max\{LR\_colldephth(i)\}$$

因此我們可得下式：

$$LR\_colldephth(i)\le \min(L_X,D_X)\Rightarrow LR\_colldephth(X)\le \min(L_X,D_X)$$

且下式成立，

$$\sum_{i=1}^{i=|LR\_keyroots|} treesize(X[i])=\sum_{j=1}^{j=|X|}|LR\_colldepth(j)|。$$

整個演算法的時間複雜度可寫成下式：

$$\sum_{i=1}^{i=|LR\_keyroots(S)|}\sum_{j=1}^{j=|LR\_keyroots(T)|} treesize(S[i])\times treesize(T[j])$$

$$=\sum_{i=1}^{i=|LR\_keyroots(S)|} treesize(S[i])\times\sum_{j=1}^{j=|LR\_keyroots(T)|} treesize(T[j])$$

$$\Rightarrow\sum_{i=1}^{|S|}|LR\_colldepth(i)|\times\sum_{j=1}^{|T|}|LR\_colldepth(j)|$$

$$\le O(|S||T|\min(L_S,D_S)\min(L_T,D_T))$$

此演算法的時間複雜度為，$O(|S||T|\min(L_S,D_S)\min(L_T,D_T))$ 其中 L_S 為樹 S 的葉節點個數，L_T 為樹 T 的葉節點個數。

2.5 限制型樹編輯距離的演算法

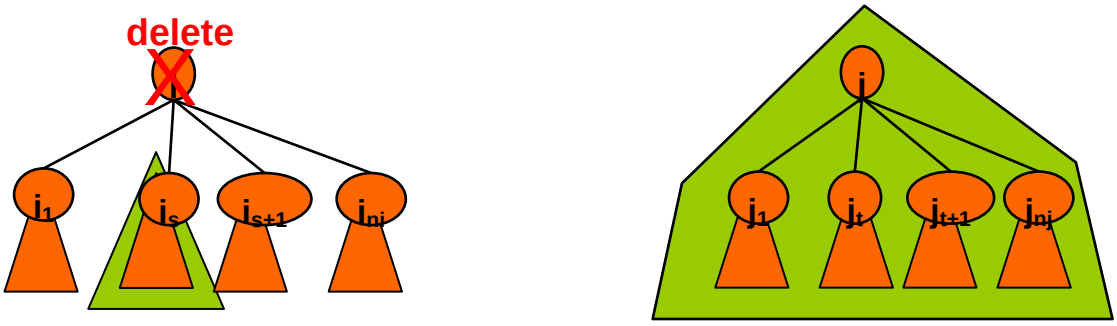
在此演算法下，我們用一二維陣列 D 來存放每一個子樹在限制型成本下 diff 的結果。若 D(S[i],T[j]) 表示兩顆以節點 i 為根節點和以節點 j 為根節點的子樹在限制型成本下的編輯距離。以 s[i]、t[j]來看的話，其遞迴關係如下：

狀況一：節點 i 被刪除。

$$D(S[i],\theta)+\min_{1\le s\le n_i}\{D(S[i_s],T[j])-D(S[i_s],\theta)\}$$

其中 n_i 為節點 i 下面的子節點總數。i_s 為 S[i] 的其中一個子節點。其

中$1 \le s \le n_i$，其示意圖如下：



圖表 2.18 刪除節點 i

分析如下：

在此狀況下，由於節點 i 被刪除，因此節點 i 下面的每一個子節點都可能成為新的 lca，所以，以節點 j 為根節點的子樹與節點 i 下面任一以子節點為根節點的子樹求其編輯距離，其餘的節點刪掉。在此狀況下，D(S[i],T[j]) 可寫成下式。

$$D(S[i],T[j])$$
$$=D(S[i_s],T[j])+D(S[i_1],\theta)+...+D(S[i_{s-1}],)+D(S[i_{s+1}],\theta)+...+D(S[i_{n_i}],\theta)$$
$$+\gamma(s[i]\rightarrow\lambda)$$
$$=D(S[i_s],T[j])+D(S[i],\theta)-D(S[i_s],\theta)$$

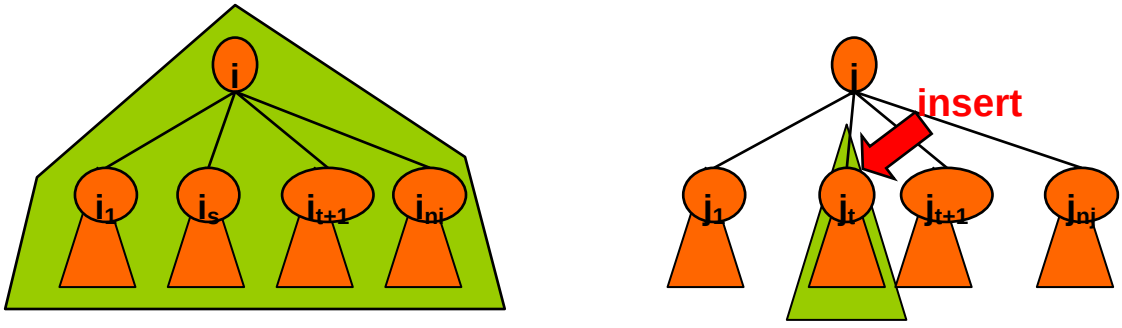
對 s 的部分取最小值得一般化的式子如下：

$$D(S[i],\theta)+\min_{1\le s\le n_i}\{D(S[i_s],T[j])-D(S[i_s],\theta)\}$$

狀況二：插入節點 j。

$$D(\theta,T[j])+\min_{1\le t\le n_j}\{D(S[i],T[j_t])-D(\theta,T[j_t])\}$$

其中 n_j 為節點 j 下面的子節點總數。j_t 為 t[j] 的其中一個子節點。其中 $1 \le t \le n_j$，其示意圖如下：



圖表 2.19 插入節點 j

分析如下：

在此狀況下，D(S[i],T[j])可寫成下式。

$$D(S[i],T[j])$$
$$=D(S[i],T[j_t])+D(\theta,T[j_1])+...+D(\theta,T[j_{t-1}])+D(\theta,T[j_{t+1}])+...+D(\theta,T[j_{n_j}])$$
$$+\gamma(\lambda \rightarrow t[j])$$
$$=D(S[i],T[j_t])+D(\theta,T[j])-D(\theta,T[j_t])$$

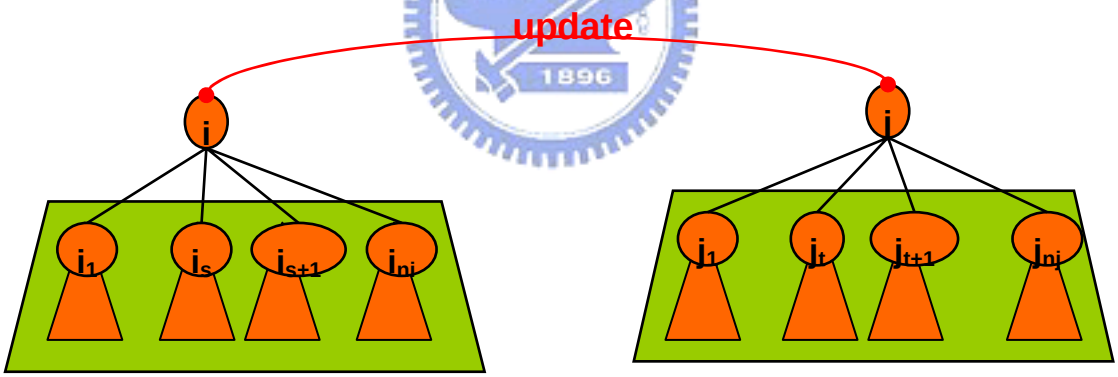
對 t 的部分取最小值得一般化的式子如下：

$$D(\theta,T[j])+\min_{1\le t\le n_j}\{D(S[i],T[j_t])-D(\theta,T[j_t])\}$$

狀況三：節點 i 的標籤換為節點 j 的標籤

$$D(F_1[i],F_2[j])+\gamma(s[i]\rightarrow t[j])$$

$F_1[i]$ 表示除了根節點 i 之外的節點。$F_2[j]$ 表示除了根節點 j 之外的節點。$D(F_1[i],F_2[j])$ 表示 $F_1[i]$ 跟 $F_2[j]$ 之間的編輯距離。其示意圖如下：



圖表 2.20 將節點 i 的標籤更換爲節點 j 的標籤

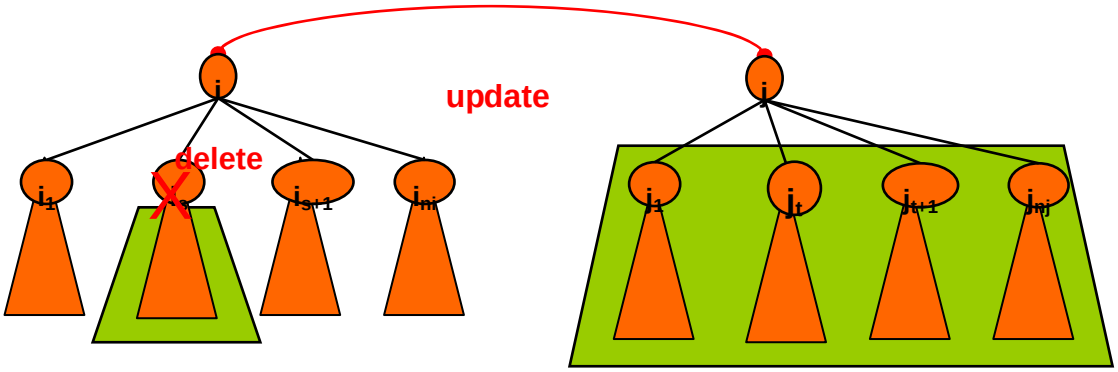
狀況三分析如下：

在計算 $D(F_1[i],F_2[j])$ 時，我們必需知道 $F_1[i]$ 跟 $F_2[j]$ 之間的編輯配對為何。我們定義 RM(i, j)為一在 $F_1[i]$ 跟 $F_2[j]$ 之間的嚴格配對(Restricted Mapping)。其定義如下。

- RM(i, j)為一在 $F_1[i]$ 跟 $F_2[j]$ 之間的配對。
- 若 $(p,q)\in M_H$ 且 $p\in S[i_s]$，$q\in T[j_t]$，則對任何的 $(p_1,q_1)\in M_H$，$p_1\in S[i_s]$，$q_1\in T[j_t]$。

因此，在狀況三的情況下我們欲求 $D(F_1[i],F_2[j])$ 也就是 $F_1[i]$ 跟 $F_2[j]$ 之間的編輯配對為何，也可分作三種狀況。

狀況 A：節點 i_s 被刪掉，示意圖如下：



圖表 2.21 刪除節點 i_s

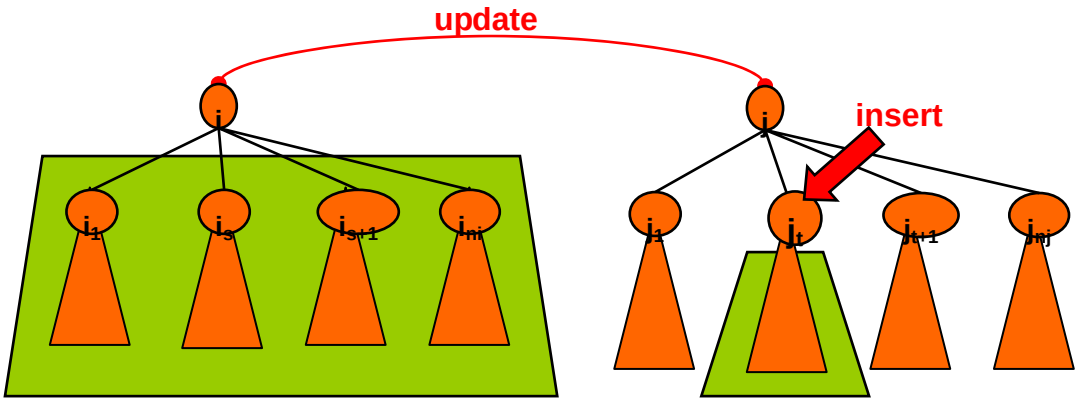
在此狀況下，$D(F_1[i],F_2[j])$ 可寫成下式。

$$D(F_1[i],F_2[j])$$
$$=D(F_1[i_s],F_2[j])+D(F_1[i_1],\theta)+...+D(F_1[i_{s-1}],\theta)+D(F_1[i_{s+1}],\theta)+...+D(F_1[i_{n_i}],\theta)$$
$$+\gamma(s[i_s]\rightarrow\lambda)$$
$$=D(F_1[i_s],F_2[j])+D(F_1[i],\theta)-D(F_1[i_s],\theta)$$

對 s 的部分取最小值得一般化的式子如下：

$$D(F_1[i],\theta)+\min_{1\le s\le n_i}\{D(F_1[i_s],F_2[j])-D(F_1[i_s],\theta)\}$$

狀況 B：插入節點 j_t，示意圖如下：



圖表 2.22 插入節點 j_t

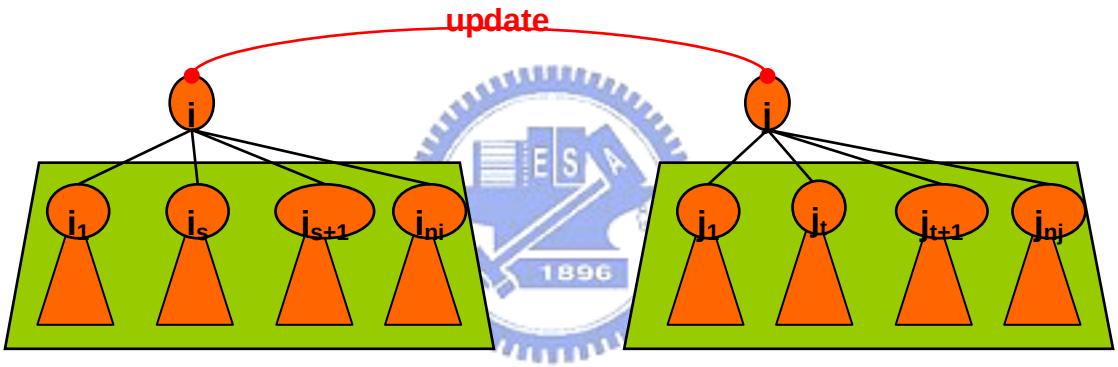
在此狀況下，$D(F_1[i],F_2[j])$ 可寫成下式。

$$D(F_1[i],F_2[j])$$
$$=D(F_1[i],F_2[j_t])+D(\theta,F_2[j_1])+...+D(\theta,F_2[j_{t-1}])+D(\theta,F_2[j_{t+1}])+...+D(\theta,F_2[j_{n_j}])$$
$$+\gamma(\lambda\rightarrow t[j_t])$$
$$=D(F_1[i],F_2[j_t])+D(\theta,F_2[j])-D(\theta,F_2[j_t])$$

對 t 的部分取最小值得一般化的式子如下：

$$D(\theta,F_2[j])+\min_{1\le t\le n_j}\{D(F_1[i],F_2[j_t])-D(\theta,F_2[j_t])\}$$

狀況 C：子樹間彼此的配對，示意圖如下：



圖表 2.23 子樹間彼此的配對

在此部分的編輯配對裡，是一顆子樹對一顆子樹的配對，我們可以利用字串比對裡的遞迴關係作一點變化如下：

$$E(s,t)=\min\begin{Bmatrix}E(s-1,t)+D(S[s],\theta)\\E(s,t-1)+D(\theta,S[t])\\E(s-1,t-1)+D(S[s],T[t])\end{Bmatrix}$$

其中 E(s, t)代表以 i_1 為根節點的子樹到以 s 為根節點的子樹跟以 j_1 為根節點的子樹到以 t 為根節點的子樹間的配對。此部分所需的時間為 $O(n_i\times n_j)$。

因此，根據上述三種狀況。$D(S[i],T[j])$ 的遞迴關係式如下：

$$D(S[i],T[j])=\min\begin{Bmatrix} D(\theta,T[j])+\min\limits_{1\le t\le n_j}\{D(S[i],T[j_t])-D(\theta,T[j_t])\} \\ D(S[i],\theta)+\min\limits_{1\le s\le n_i}\{D(S[i_s],T[j])-D(S[i_s],\theta)\} \\ D(F_1[i],F_2[j])+\gamma(s[i]\rightarrow t[j]) \end{Bmatrix}$$

$D(F_1[i],F_2[j])$ 的遞迴關係式如下：

$$D(F_1[i],F_2[j])=\min\begin{Bmatrix} D(\theta,F_2[j])+\min\limits_{1\le t\le n_j}\{D(F_1[i],F_2[j_t])-D(\theta,F_2[j_t])\} \\ D(F_1[i],\theta)+\min\limits_{1\le s\le n_i}\{D(F_1[i_s],F_2[j])-D(F_1[i_s],\theta)\} \\ E[s,t] \end{Bmatrix}$$

我們使用下列初始化條件：

$$D(\theta,\theta)=0$$

$$D(F_1[i],\theta)=\sum_{k=1}^{n_i} D(S[i_k],\theta);$$

$$D(S[i],\theta)=D(F_1[i],\theta)+\gamma(s[i]\rightarrow\Lambda)$$

$$D(\theta,F_2[j])=\sum_{k=1}^{n_j} D(\theta,S[i_k]);$$

$$D(\theta,T[j])=D(\theta,F_2[i])+\gamma(\Lambda\rightarrow t[j])$$

在上面初始化中，θ 表示空樹，Λ 表示空節點，完整的演算法如下：

1 for i = 1 to $|T_1|$

2 for j = 1 to $|T_2|$

3 $E(0,0) = 0$

4 for s=1 to n_i

5 $E(s,0) = E(s-1,0)+D(s[i_s] \rightarrow \lambda)$

6 for t=1 to n_j

7 $E(0,t)=E(0,t-1)+D(\lambda \rightarrow t[j_t])$

8 for s=1 to n_i

9 for t=1 to n_j

10
$$E(s,t)=\min\begin{Bmatrix} E(s,t-1)+D(\theta,T[j_t]) \\ E(s-1,t)+D(S[i_s],\theta) \\ E(s-1,t-1)+D(S[i_s],T[j_t]) \end{Bmatrix}$$

11
$$D(F_1[i],F_2[j])=\min\begin{Bmatrix} D(\theta,F_2[j])+\min\limits_{1\le t\le n_j}\{D(F_1[i],F_2[j_t])-D(\theta,F_2[j_t])\} \\ D(F_1[i],\theta)+\min\limits_{1\le s\le n_i}\{D(F_1[i_s],F_2[j])-D(F_1[i_s],\theta)\} \\ E(n_i,n_j) \end{Bmatrix}$$

12
$$D(S[i],T[j])=\min\begin{Bmatrix} D(\theta,T[j])+\min\limits_{1\le t\le n_j}\{D(S[i],T[j_t])-D(\theta,T[j_t])\} \\ D(S[i],\theta)+\min\limits_{1\le s\le n_i}\{D(S[i_s],T[j])-D(S[i_s],\theta)\} \\ D(F_1[i],F_2[j])+\gamma(s[i] \rightarrow t[j]) \end{Bmatrix}$$

空間複雜度分析

由於我們只需要儲存二個 2 維陣列 $D(S[i],T[j])$ 跟 $D(F_1[i],F_2[j])$。因此，空間複雜度為 $O(|S|\times|T|)$。

時間複雜度分析

主要決定時間複雜度在第 8、9 行的時間複雜度為 $O(n_i \times n_j)$。全部的演算法時間複雜度如下：

$$\sum_{i=1}^{|S|}\sum_{j=1}^{|T|}O(n_i \times n_j) \le O(\sum_{i=1}^{|S|}\sum_{j=1}^{|T|}(n_i \times n_j)) = \sum_{i=1}^{|S|}n_i \sum_{j=1}^{|T|}n_j = |S|\times|T|$$

因此，此演算法時間複雜度為O(|S|×|T|)。

第三章 動機與目標

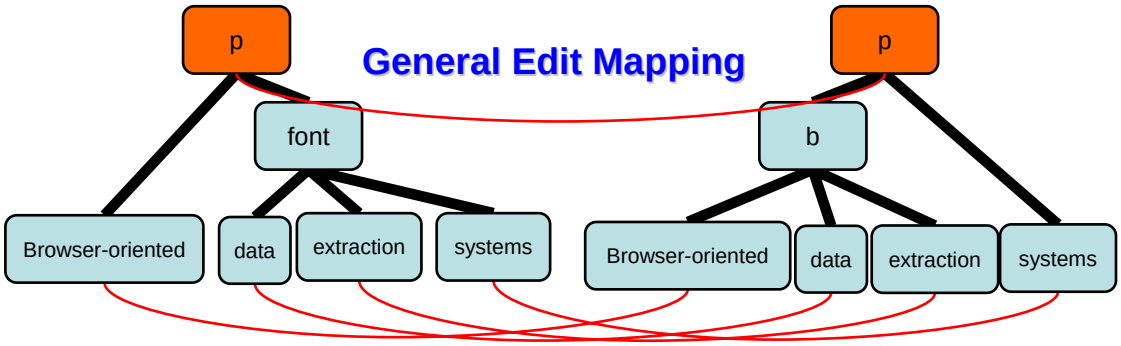
在 HTML 中，根據 W3C 的規範，body 節點下面的標籤主要可以分為兩類。一類為 Inline-level 的標籤；另一類為 Block-level 的標籤。詳細請參閱附錄。

Inline-level 的標籤主要是用在文字資料的修飾，如：b(粗體)、em(強調)、font(變更字型或顏色)、span(進行文字風格的設定)…。使用這些標籤並不會產生畫面上的斷行。此類標籤多半屬於傳統型編輯配對，我們稱這些節點為 G-node。以比較淺色的節點表示。

Block-level 的標籤主要是用於畫面上的排版、版面配置，如：table(表格)、ol(有序列)、form(表單)、h1(標題)、p(分段)…。使用這些標籤在畫面上會產生與前後標籤的分隔行。此類標籤多半屬於限制型編輯配對，我們稱這些節點為 C-node。以比較深色的節點表示。

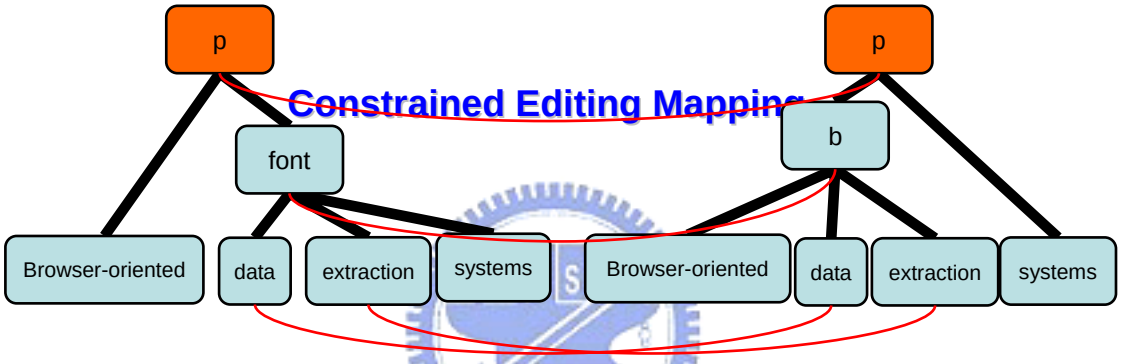
在我們的觀察中，我們發現文字型的資料適合使用傳統型編輯配對。舉例如下：

比如說我們今天在我們的網頁上有一行字為”Browser-oriented data extraction system”。在製作網頁的時候很可能我一開始要強調的是”data extraction system”，所以我把這三個字使用另一種字型表示。但是下次修改網頁時，我可能要強調”Browser-oriented data extraction”這種技術，所以我把這幾個字加粗體。
下圖表示我們使用傳統型編輯配對。可以發現在傳統型編輯配對下，每一個文字節點都能配對成功，且原先的 font 標籤被刪除了，後來的樹則插進了一個 b 標籤。



圖表 3.1 文字資料使用傳統型編輯配對

但若我們以限制型編輯配對的話，如下圖。



圖表 3.2 文字型資料使用限制型編輯配對

可以發現，在限制型編輯配對下，因為要維持一顆子樹對應於一顆子樹上的結構性考量，所以所得到的結果是在原本的樹中，文字”Browser-oriented”跟文字”systems”被刪除了，而在新的樹中，則是插入了文字”Browser-oriented”跟文字”systems”。這樣的結果雖然沒有錯，但是對於文字型的資料使用傳統型編輯配對所找出的結果是更接近於實際 HTML 改寫的操作。

相較於文字型資料，網頁的另一部分則是頁面中版面的配置(layout)。也就是多半由 table、div、p、…等標籤所表示的部分。在 HTML 版面改版時，比如其中一個表格由原本的地方移動到頁面的別處；其中一個段落刪除；或是多了一個表單…等版面上的操作。幾乎都是一塊一塊的操作或者說是一個單元(cell)一個單元的操作，因此在 Block-level 的標籤比文字型資料更具有結構上的考量，因此用限制型編輯配對所找出的結果是更接近於實際 HTML 改版的操作。

3.1 我們的目標

在我們的觀察裡，我們已經可以看出來，在一份樹狀結構的文件中，不一定所有的節點都適用於單一的一種編輯配對。因此，我們要提出一個新的混合型編輯配對以符合現實生活中所遭遇到的狀況。也就是說，在一份文件中，可能有部分的節點是適用於傳統型編輯配對；而另外有一部分的節點適用於傳統型的編輯配對。至於哪些節點屬於傳統型編輯配對，哪些節點又屬於限制型編輯配對，這需要視文件的類型及使用者的需求而定，在本論文中，我們主要以 HTML 文件為例來說明。

本論文所要解決的問題乃是在已知有兩份樹狀節構文件 S 與 T 中(二者皆為有序標籤含根樹)，已知有某些標籤屬於傳統型編輯配對，另有標籤屬於限制型編輯配對的情況下，欲求由 S 轉換成 T 的的最小成本。

在本篇論文中，使用傳統型編輯配對的節點稱為 G-node，圖形上以淺色節點表示；使用限制型編輯配對的節點稱為 C-node，圖形上以深色節點表示。

3.2 混合型編輯配對(Hybrid Editing Mapping)

混合型成本模型與上述兩種編輯配對具有對樹相同的操作，包括刪除一節點(Delete)、插入一節點(Insert)、更換兩節點的標籤(Relabel)，細節如上所述。

我們定義 (M_H,S,T) 為 S 到 T 的混合型編輯配對。

一個合法的混合型編輯配對 (M_H,S,T)，應符合下列規範：

- 一對一配對
  - 配對 (M_H,S,T) 中任何兩個配對組合 $(i_1,j_1),(i_2,j_2)$，若 $i_1=j_1 \Leftrightarrow i_2=j_2$。
- 保留兄弟節點的順序

- 配對 (M_H,S,T) 中任何兩個配對組合 $(i_1,j_1),(i_2,j_2)$，若 i_1 節點在 j_1 的右邊 $\Leftrightarrow$ i_2 節點在 j_2 的右邊。

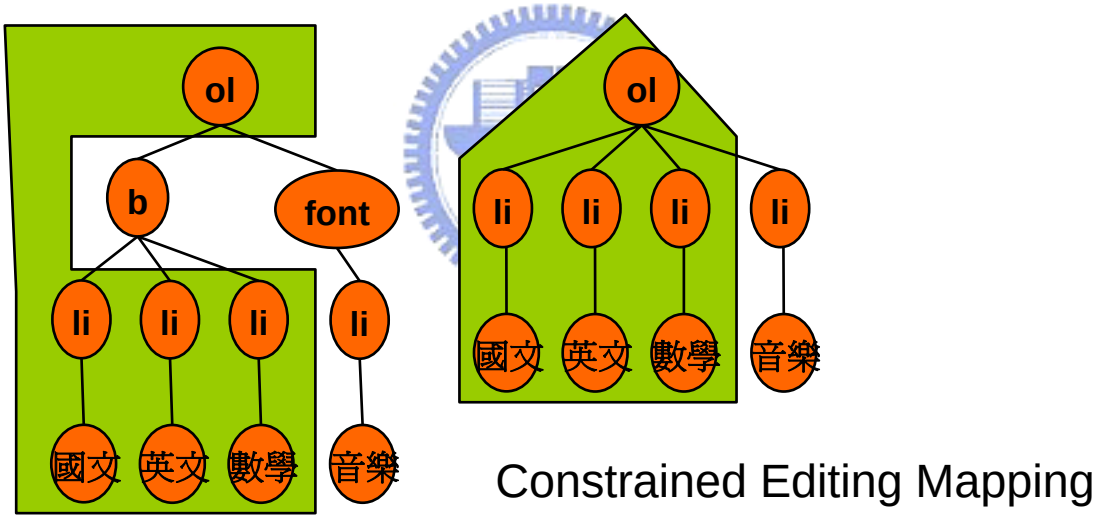
• 保留祖先節點與後代節點的順序

- 配對 (M_H,S,T) 中任何兩個配對組合 $(i_1,j_1),(i_2,j_2)$，若 i_1 節點為 i_2 節點的祖先 $\Leftrightarrow$ j_1 節點為 j_2 節點的祖先。

• 保留最近的限制型共同祖先的(Constrained Least Common Ancestor)關係

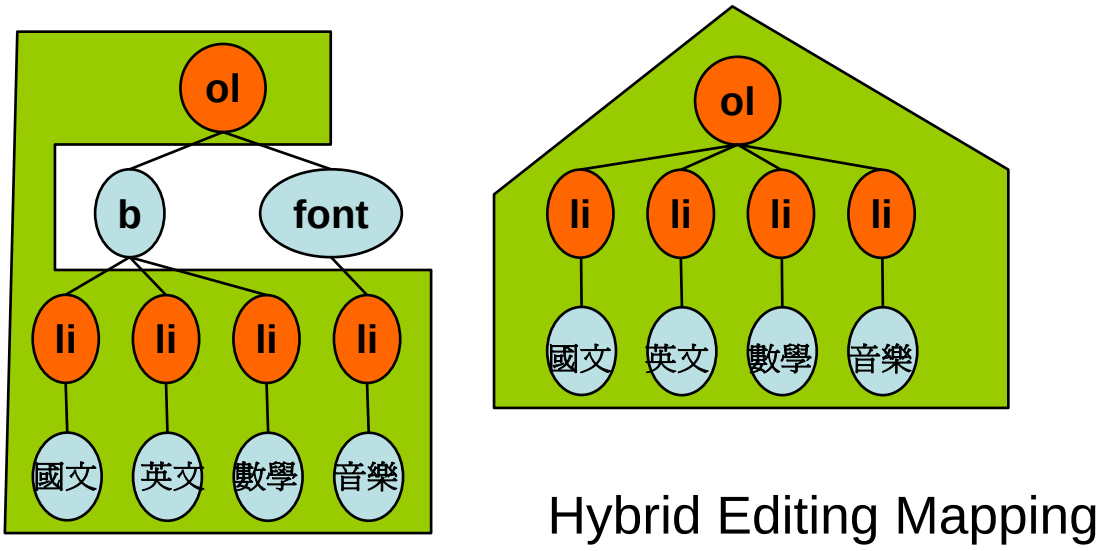
- 配對 (M_H,S,T) 中任何三個配對組合 $(i_1,j_1),(i_2,j_2),(i_3,j_3)$，若 $s[clca(i_1,i_2)]$ 節點為 $s[i_3]$ 節點的祖先 $\Leftrightarrow$ $t[clca(j_1,j_2)]$ 節點為 $t[j_3]$ 節點的祖先。

其中 clca 為一函式，表示傳入一顆樹的兩個節點，傳回這兩個節點的最近為 C-node 的祖先節點。以下圖為例可以看出與限制型編輯配對的不同。



圖表 3.3 使用限制型編輯配對

這是一個有四個項目的有序列。其中前三個項目被標上粗體、第四個項目被指定字型。經過網頁編輯過後，粗體跟字型標籤被移除了。在限制型配對下，前三個項目(標籤 li)的 lca 為 b 標籤，因此第四個項目(font 下的 li)沒有辦法配對成功，它屬於另一顆子樹。但若我們以混合型編輯配對來作的話，可得到下面的結果。



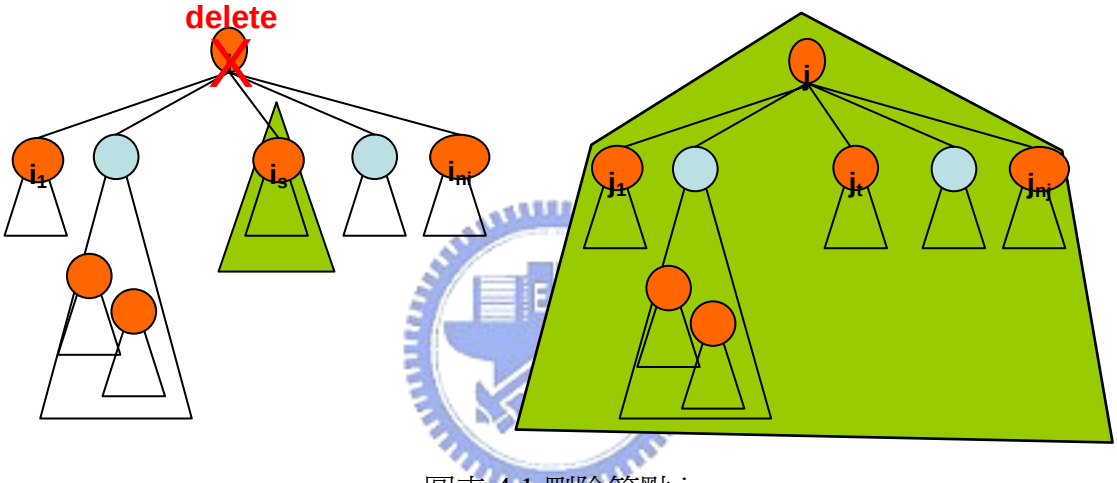
圖表 3.4 使用混合型編輯配對

四個項目(li)都配對成功，因為它們都在同一個 clca(標籤 ol)下，也就是在同一個 C-node 的子樹下。

第四章 混合型樹編輯距離的演算法

在我們的演算法中，可簡單分為兩個部分。一部分是 Tree-to-Tree 的狀況；另一部分是 Forest-to-Forest 的狀況。首先我們先看 Tree-to-Tree 的狀況，我們假設根節節為 C-node，也就是說根節點為最上層的 clca。以兩個 C-node 節點，節點 i 和節點 j 來看，總共有三個狀況。

狀況一：刪除節點 i，節點 i 為 C-node，見下圖。



圖表 4.1 刪除節點 i

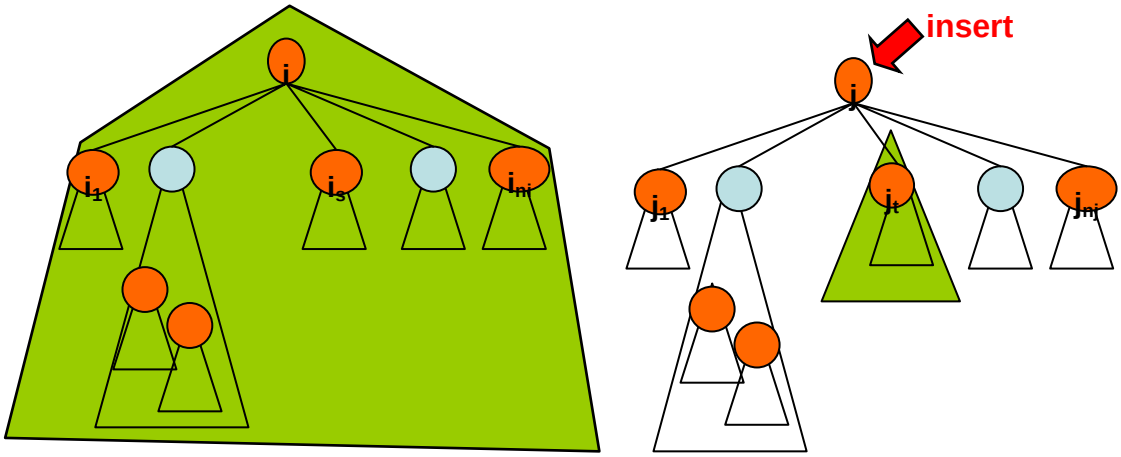
由於節點 i 被刪除，由節點 i 往葉節點的路徑(path)上第一個碰到的 C-node 都有可能成為下面節點的 clca。因此，在此情況下，$D_H(S[i],T[j])$ 可寫成下式。其中 n_i 包括由節點 i 往葉節點的路徑(path)上第一個碰到的 C-node 的個數。

$$D_H(S[i],T[j])$$
$$=D_H(S[i_s],T[j])+D_H(S[i_1],\theta)+\ldots+D_H(S[i_{s-1}],\theta)+D_H(S[i_{s+1}],\theta)+\ldots+D_H(S[i_{n_i}],\theta)$$
$$+\gamma(s[i]\rightarrow\Lambda)$$
$$=D_H(S[i_s],T[j])+D_H(S[i],\theta)-D_H(S[i_s],\theta)$$

對 s 的部分取最小值得一般化的式子如下：

$$D_H(S[i],\theta)+\min_{1\le s\le n_i}\{D_H(S[i_s],T[j])-D_H(S[i_s],\theta)\}$$

狀況二：插入節點 j，j 為 C-node，見下圖。



圖表 4.2 插入節點 i

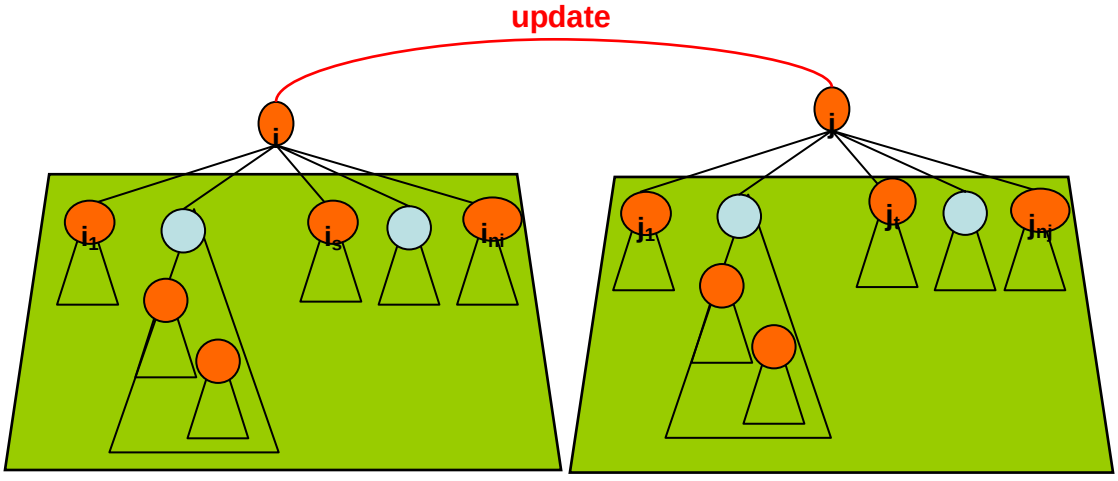
由於插入節點 i，由節點 i 往葉節點的路徑(path)上第一個碰到的 C-node 都有可能成為下面節點的 clca。因此，在此情況下，$D_H(S[i],T[j])$ 可寫成下式。其中 n_j 包括由節點 j 往葉節點的路徑(path)上第一個碰到的 C-node 的個數。

$$D_H(S[i],T[j])$$
$$=D_H(S[i],T[j_t])+D_H(\theta,T[j_1])+\ldots+D_H(\theta,T[j_{t-1}])+D_H(\theta,T[j_{t+1}])+\ldots+D_H(\theta,T[j_{n_j}])$$
$$+\gamma(\Lambda \rightarrow t[j])$$
$$=D_H(S[i],T[j_t])+D_H(\theta,T[j])-D_H(\theta,T[j_t])$$

對 t 的部分取最小值得一般化的式子如下：

$$D_H(\theta,T[j])+\min_{1\le t\le n_j}\{D_H(S[i],T[j_t])-D_H(\theta,T[j_t])\}$$

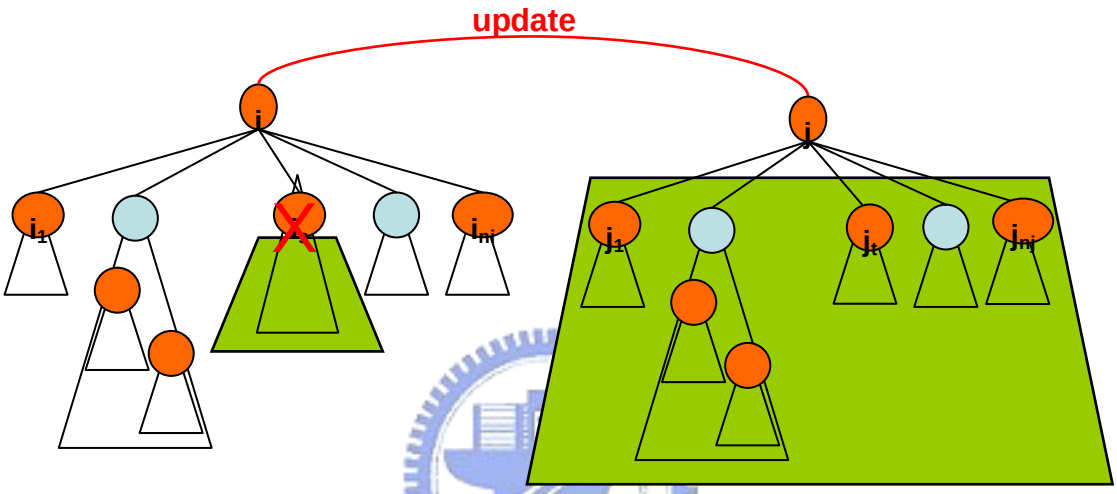
狀況三：節點 i 的標籤換為節點 j 的標籤，節點 i 節點皆為 C-node。



圖表 4.3 節點 i 的標籤換爲節點 j 的標籤

令 $F_1[i]$ 表示除了根節點 i 之外的節點、$F_2[j]$ 表示除了根節點 j 之外的節點。$D_H(F_1[i],F_2[j])$ 表示 $F_1[i]$ 跟 $F_2[j]$ 之間的編輯距離。因此，在狀況三的情況下我們欲求 $D_H(F_1[i],F_2[j])$ 也就是 $F_1[i]$ 跟 $F_2[j]$ 之間的最小編輯配對為何，可以有下列的遞迴關係。

狀況一：刪除節點 i_s，示意圖如下。



圖表 4.4 節點 i_s 被刪除

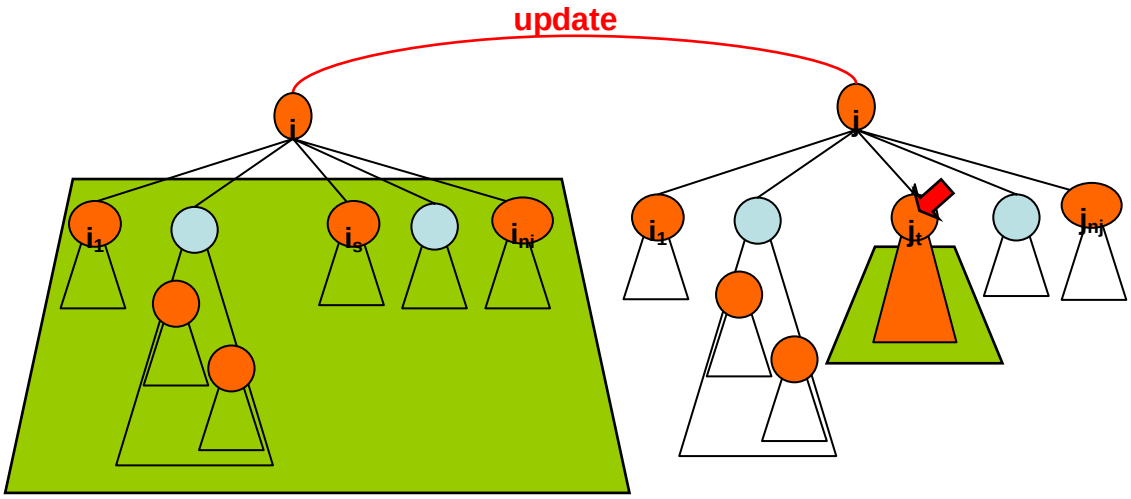
在此情況下，$D_H(F_1[i],F_2[j])$ 可寫成下式。其中 n_j 包括由節點 j 往葉節點的路徑(path)上第一個碰到的 C-node 的個數。

$$D_H(F_1[i],F_2[j])$$
$$=D_H(F_1[i_s],F_2[j])+D_H(F_1[i_1],\theta)+...+D_H(F_1[i_{s-1}],\theta)+D_H(F_1[i_{s+1}],\theta)+...+D_H(F_1[i_{n_i}],\theta)$$
$$+\gamma(s[i]\rightarrow\Lambda)$$
$$=D_H(F_1[i_s],F_2[j])+D_H(F_1[i],\theta)-D_H(F_1[i_s],\theta)$$

對 s 的部分取最小值得一般化的式子如下：

$$D_H(F_1[i],\theta)+\min_{1\le s\le n_i}\{D_H(F_1[i_s],F_2[j])-D_H(F_1[i_s],\theta)\}$$

狀況二：插入節點 j_t，示意圖如下。



圖表 4.5 插入節點 j_t

在此情況下，$D_H(F_1[i],F_2[j])$ 可寫成下式。

$$D_H(F_1[i],F_2[j])$$
$$=D_H(F_1[i],F_2[j_t])+D_H(\theta,F_2[j_1])+\ldots+D_H(\theta,F_2[j_{t-1}])+D_H(\theta,F_2[j_{t+1}])+\ldots+D_H(\theta,F_2[j_{n_j}])$$
$$+\gamma(\Lambda\rightarrow t[j])$$
$$=D_H(F_1[i],F_2[j_t])+D_H(\theta,F_2[j])-D_H(\theta,F_2[j_t])$$

對 t 的部分取最小值得一般化的式子如下：

$$D_H(\theta,F_2[j])+\min_{1\le t\le n_j}\{D_H(F_1[i],F_2[j_t])-D_H(\theta,F_2[j_t])\}$$

上式中 n_j 包括由節點 j 往葉節點的路徑(path)上第一個碰到的 C-node 的個數。

狀況三：在此情況下，我們定義一個新的編輯配對為 M_X。

M_X 的定義如下：

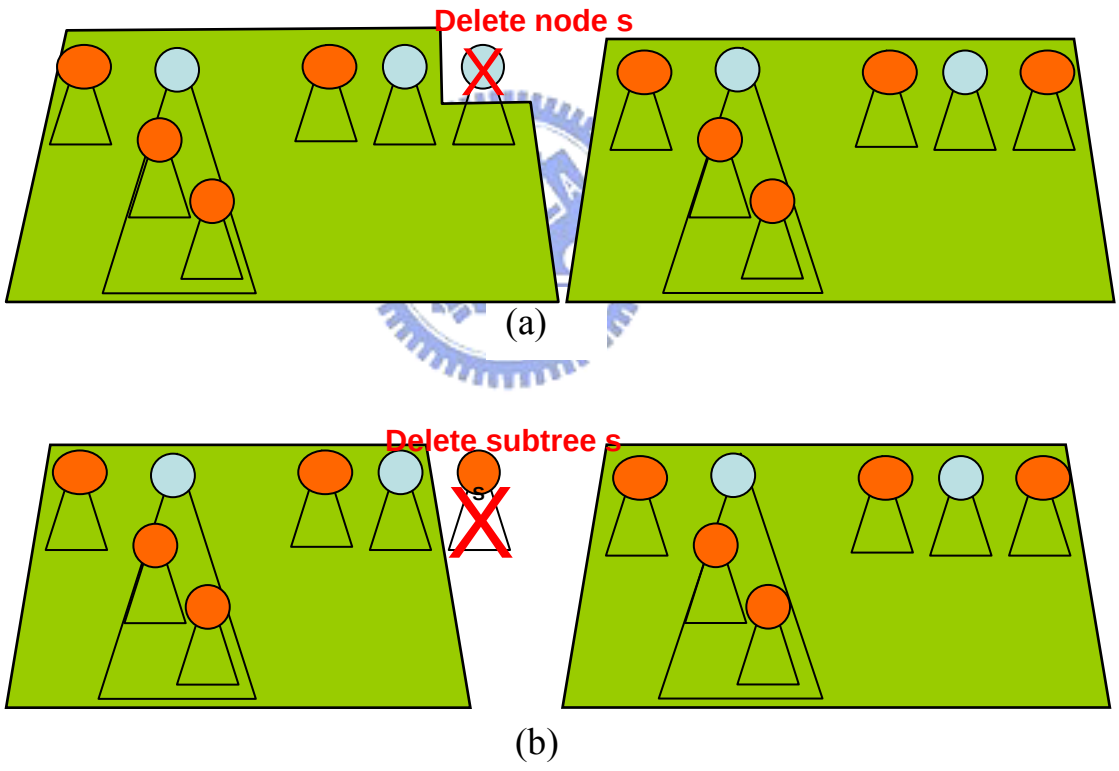
- 為 $F_1[i]$ 跟 $F_2[j]$ 的編輯配對。
- 若 $(p,q)\in M_X$ 且 $p\in S[i_s]$，$q\in T[j_t]$，其中 i_s 跟 j_t 皆為 C-node，則對任何的 $(p_1,q_1)\in M_X$，$p_1\in S[i_s]$，$q_1\in T[j_t]$。
- 若 $(x_1,y_1),(x_2,y_2)\in M_X$ 且 x_1,y_1,x_2,y_2 為 G-node，則下列三式成立：
  - $x_1=x_2\Leftrightarrow y_1=y_2$。

- x_1 節點在節點 y_1 的右邊 ⇔ x_2 節點在節點 y_2 的右邊。
- x_1 節點為 x_2 的祖先 ⇔ y_1 節點為 y_2 的祖先。

定義在樹 S 中此部分的點集合為 h_i；在樹 T 中此部分的點集合為 h_j。h_i 為節點 i 到葉節點的路徑上遇到第一個 C-node 前的所有點(有包括第一個 C-node)。d_i 為 h_i 的深度。h_j 為節點 j 到葉節點的路徑上遇到第一個 C-node 前的所有點(有包括第一個 C-node)。d_i 為 h_i 的深度。

我們定義 $FD_H(i'..s,j'..t)$ 為 h_i 到 h_i 的最小編輯距離。其中 $i' \le s \le i-1$，$j' \le t \le j-1$。i’ 的定義為從節點 i 到最左邊葉節點的路徑上碰到的第一個 C-node，若此節點不存在，則 i’ 為 $lleaf_H(i)$。

第一類：刪除一個節點或是刪去一顆樹，見下圖。



圖表 4.6 (a)刪除一個節點 (b)刪去一顆樹

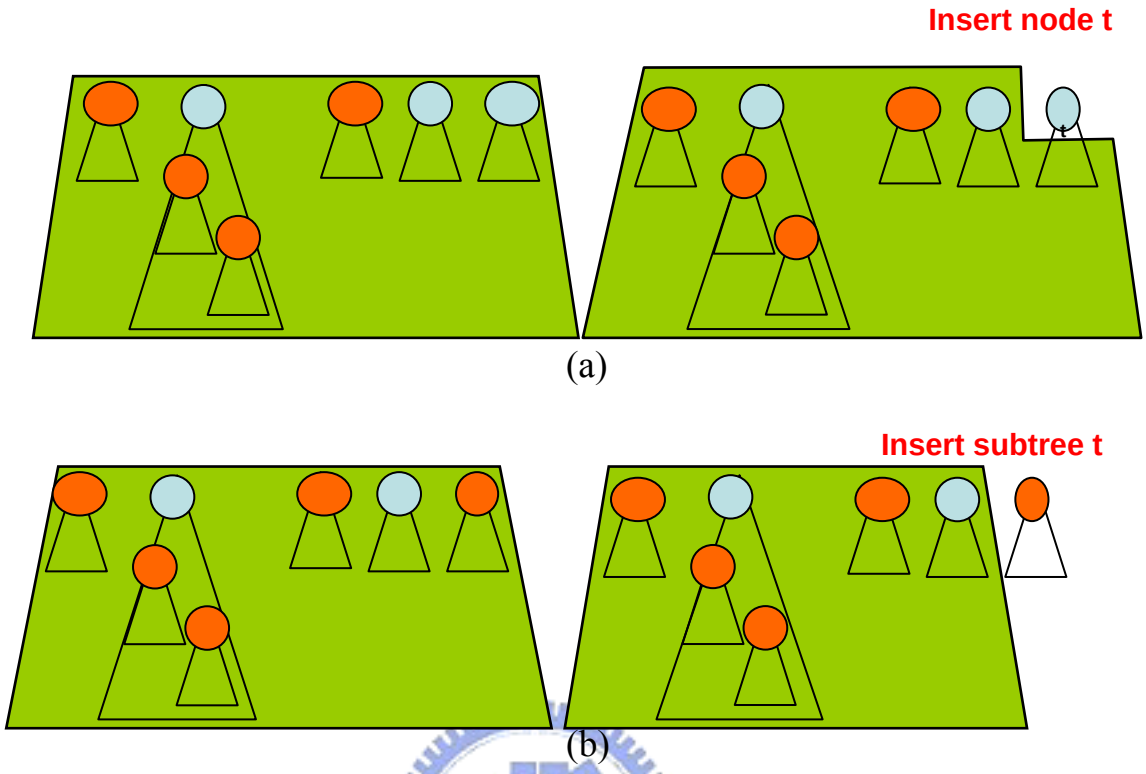
若 s 為 G-node 則表示刪除一個節點，可寫成下式。

$$FD_H(i'..s-1,j'..t)+\gamma(s \to \Lambda)$$

若 s 為 C-node 則表示刪去一顆以 s 為根節點的子樹，可寫成下式。

$$FD_H(i'..s-1,j'..t)+D_H(S[s],\theta)$$

第二類：插入一個節點或插入一顆樹，見下圖。



圖表 4.7 (a)插入一個節點 (b)插入一顆樹

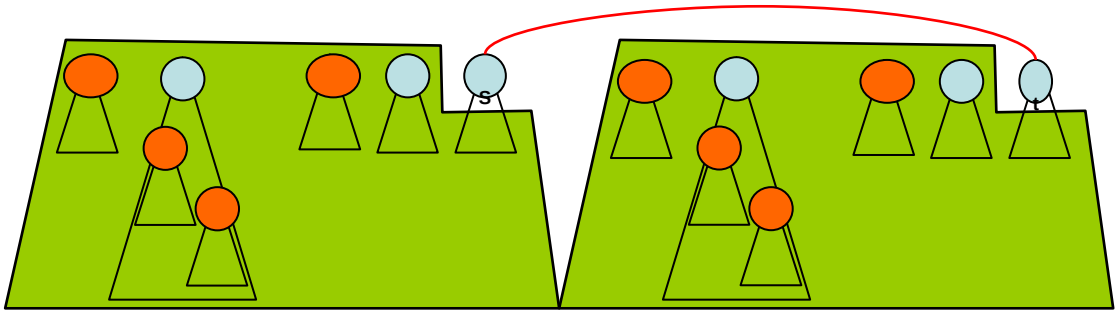
若 t 為 G-node 則表示插入一個節點。

$$FD_H(i'..s,j'..t\text{-}1)+\gamma(\Lambda \rightarrow t)$$

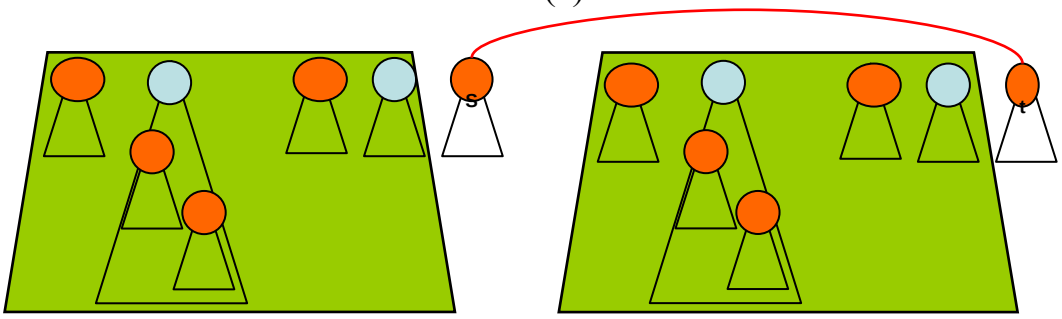
若 t 為 C-node 則表示插入一顆樹。

$$FD_H(i'..s,j'..t\text{-}1)+D_H(\theta,T[t])$$

第三類：將一個節點的標籤更換成另一個節點的標籤或是求一顆樹跟另一顆樹的最小編輯配對，見下圖。



(a)



(b)

圖表 4.8 (a)將一個節點的標籤更換成另一個節點的標籤 (b)求一顆樹跟另一顆樹的最小編輯配對

若 s 為 G-node 且 t 為 G-node 則表示將節點 s 的標籤更換成節點 t 的標籤，可寫成下式。

$$FD_H(i'..s-1,j'..t-1)+\gamma(s \to t)$$

若 s 為 C-node 且 t 為 C-node 則表示要求 $D_H(S[s],T[t])$，可寫成下式。

$$FD_H(i'..s-1,j'..t-1)+D_H(S[s],T[t])$$

由上述推導，可得 $FD_H(i'..s,j'..t)$ 的遞迴關係如下：若 s 為 C-node，t 為 C-node，則

$$FD_H(i'..s,j'..t)=\min\begin{Bmatrix} FD_H(i'..s-1,j'..j_K)+D_H(S[s],\theta) \\ FD_H(i'..s,j'..j_K-1)+D_H(\theta,T[t]) \\ FD_H(i'..s-1,j'..j_K-1)+D_H(S[s],T[t]) \end{Bmatrix}。$$

若 s 為 C-node，t 為 G-node，則

$$FD_H(i'..s,j'..t)=\min\begin{Bmatrix} FD_H(i'..s\text{-}1,j'..t)+D_H(T_l[s],\theta) \\ FD_H(i'..s,j'..t\text{-}1)+\gamma(\Lambda\rightarrow t) \end{Bmatrix}。$$

若 s 為 G-node，t 為 C-node，則

$$FD_H(i'..s,j'..t)=\min\begin{Bmatrix} FD_H(i'..s\text{-}1,j'..t)+\gamma(s\rightarrow\Lambda) \\ FD_H(i'..s,j'..t\text{-}1)+D_H(\theta,T[t]) \end{Bmatrix}。$$

若 s 為 G-node，t 為 G-node，則

$$FD_H(i'..s,j'..t)=\min\begin{Bmatrix} FD_H(i'..s\text{-}1,j'..t)+\gamma(s\rightarrow\Lambda) \\ FD_H(i'..s,j'..t\text{-}1)+\gamma(\Lambda\rightarrow t) \\ FD_H(i'..s\text{-}1,j'..t\text{-}1)+\gamma(s\rightarrow t) \end{Bmatrix}。$$

另外，在此情況下若 $lleaf_H(s)\neq lleaf(ancs)$ 或 $lleaf_H(t)\neq lleaf(anct)$ 時，可寫成下式：

$$FD_H(i'..s,j'..t)=\min\begin{Bmatrix} FD_H(i'..s\text{-}1,j'..t)+\gamma(s\rightarrow\Lambda) \\ FD_H(i'..s,j'..t\text{-}1)+\gamma(\Lambda\rightarrow t) \\ FD_H(i'..s\text{-}1,j'..t\text{-}1)+treedist_H(s,t) \end{Bmatrix}$$

因為在此情況下節點 s 或節點 t 為某個子樹的根節點。

此部分的時間複雜度由[ZS89]可得知為 $O(h_i\times h_j\times d_i\times d_j)$。
其中 d_i 為 h_i 的深度，d_j 為 h_j 的深度。
我們使用下列初始化條件：

$$D_H(\theta,\theta)=0$$

$$D_H(F_l[i],\theta)=\sum_{k=1}^{n_i} D_H(S[i_k],\theta);$$

$$D_H(S[i],\theta)=D_H(F_l[i],\theta)+\gamma(s[i]\rightarrow\Lambda)$$

$$D_H(\theta,F_2[j])=\sum_{k=1}^{n_j} D_H(\theta,S[i_k]);$$

$$D_H(\theta,T[j])=D_H(\theta,F_2[i])+\gamma(\Lambda\rightarrow t[j])$$

在上式中，θ 為空樹，Λ 為空節點，完整演算法如下：

1 for i = 1 to $|T_1|$

2 for j = 1 to $|T_2|$

3 if i is C-node and j is C-node

4 rst = $FD_H(i'..i\text{-}1,j'..j\text{-}1)$

5 $$D_H(F_1[i],F_2[j])=\min\begin{cases} D_H(\theta,F_2[j])+\min\limits_{1\le t\le n_j}\{D_H(F_1[i],F_2[j_t])-D_H(\theta,F_2[j_t])\} \\ D_H(F_1[i],\theta)+\min\limits_{1\le s\le n_i}\{D_H(F_1[i_s],F_2[j])-D_H(F_1[i_s],\theta)\} \\ rst \end{cases}$$

6 $$D_H(T_1[i],T_2[j])=\min\begin{cases} D_H(\theta,T_2[j])+\min\limits_{1\le t\le n_j}\{D_H(T_1[i],T_2[j_t])-D_H(\theta,T_2[j_t])\} \\ D_H(T_1[i],\theta)+\min\limits_{1\le s\le n_i}\{D_H(T_1[i_s],T_2[j])-D_H(T_1[i_s],\theta)\} \\ D_H(F_1[i],F_2[j])+\gamma(t_1[i]\to t_2[j]) \end{cases}$$

$FD_H(i'..i\text{-}1,j'..j\text{-}1)$ 的初始化條件如下：

$FD_H[\theta,\theta] = 0;$

foreach node a in h_i

  if a is C-node

    $FD_H[a,\theta] = FD_H[a\text{-}1,\theta]+D_H[a,\theta];$

  else

    $FD_H[a,\theta] = FD_H[a\text{-}1,\theta]+\gamma(a\to\Lambda)$

foreach node b in h_j

  if b is C-node

    $FD_H[\theta,b] = FD_H[\theta,b\text{-}1]+D_H[0,b];$

  else

    $FD_H[\theta,b] = FD_H[\theta,b\text{-}1]+\gamma(\Lambda\to b)$

計算 $FD_H(i'..i\text{-}1,j'..j\text{-}1)$ 的主程式如下：

1 Preprocessing(compute $lleaf_H()$,$LR\_keyroots1_H$,$LR\_keyroots2_H$)
main loop
2 foreach node s in $LR\_keyroots1_H$)
3 foreach node t in $LR\_keyroots2_H$)
4 htreedist(s,t)

函式 htreedist(s,t) 即利用 $FD_H(i'..s,j'..t)$ 的遞迴關係求解 rst。

空間複雜度

由於我們只需儲存 $D_H(F_1[i],F_2[j])$ 、 $D_H(T_1[i],T_2[j])$ 和 $FD_H(i'..s,j'..t)$ 的空間，故為 $O(|S|\times|T|)$。

時間複雜度

令 $|S|=C_1+G_1$，其中 C_1 為 S 中 C-node 的個數，G_1 為 T 中 G-node 的個數。令 $|T|=C_2+G_2$，其中 C_2 為 S 中 C-node 的個數，G_2 為 T 中 G-node 的個數。時間複雜度主要由第三行的 $FD_H(i'..i\text{-}1,j'..j\text{-}1)$ 決定。

$$\sum_{i=1}^{C_1}\sum_{j=1}^{C_2} O(h_i\times h_j\times d_i\times d_j) \le O(\sum_{i=1}^{C_1} h_i\times d_i\times \sum_{j=1}^{C_2} h_j\times d_j)$$

$$\le O(\max(d_i)\times\max(d_j)\times\sum_{i=1}^{C_1} h_i\times\sum_{j=1}^{C_2} h_j)$$

$$\le O(\max(d_i)\times\max(d_j)\times|S|\times|T|)$$

$$=O(d_S^{max}\times d_T^{max}\times|S|\times|T|)$$

其中 d_i 為 h_i 的深度，d_j 為 h_j 的深度，d_S^{max} 為 $\max(d_i)$，d_T^{max} 為 $\max(d_j)$。

第五章 結論與未來展望

過去的研究中同一顆樹中只有一種編輯配對，本篇論文提出了一個新的樹編輯配對的演算法，時間複雜度為$O(d_S^{max} \times d_T^{max} \times |S| \times |T|)$，以補足過去研究上的不足。使用我們所提出的演算法，使用者可以自己分析所處理的樹，自行定義哪些節點為限制型編輯配對的節點，哪些節點為傳統型編輯配對的節點，使最後找出來的結果更符合樹編輯的過程。

因此，使用我們所提的演算法具有下列特性：

第一、 適用於 XML、HTML/XHTML 文件的比對。

第二、 在兩顆樹 diff 的過程中具有較大的彈性，可依使用者的需求進行 diff。

甲、 若使用者需要進行傳統型編輯配對的演算法，只需在進行演算法之前把每一個樹的節點標為 G-node 即可。

乙、 若使用者需要進行限制型編輯配對的演算法，只需在進行演算法之前把每一個樹的節點標為 C-node 即可。

未來我們希望此演算法的最佳化證明及提出一分文件的節點該如何標示為 G-node 或是 C-node 的導引。再者，在某些標籤(如 HTML 中的 HEAD 標籤)下可能或有無序的節點，這也是我們未來探討的範圍。

參考文獻

[Che01]

Weimin Chen. New algorithm for ordered tree-to-tree correction problem. *Journal of Algorithms*, 40:135.158, 2001.

[Kle98]

P.N. Klein. Computing the edit-distance between unrooted ordered trees. *In Proceedings of the 6th annual European Symposium on Algorithms (ESA)* 1998., pages 91.102. Springer-Verlag, 1998.

[LST01]

Chin Lung Lu, Zheng-Yao Su, and Chuan Yi Tang. A new measure of edit distance between labeled trees. *In Proceedings of the 7th Annual International Conference on Computing and Combinatorics (COCOON), volume 2108 of Lecture Notes in Computer Science (LNCS)*. Springer, 2001.

[Sel77]

Stanley M. Selkow. The tree-to-tree editing problem. *Information Processing Letters*, 6(6):184.186, 1977.

[Tai79]

Kuo-Chung Tai. The tree-to-tree correction problem. *Journal of the Association for Computing Machinery (JACM)*, 26:422.433, 1979.

[ZS89]

K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18(6):1245-1262, 1989.

[Zha89]

Kaizhong Zhang. The Editing Distance Between Trees: Algorithms and Applications. PhD thesis, Courant Institute, Departement of Computer Science, 1989.

[Zha95]

Kaizhong Zhang. Algorithms for the constrained editing problem between ordered labeled trees and related problems. *Pattern Recognition*, 28:463.474, 1995.

附錄

根據 HTML 4.0 的規範，Block-level Element、Inline-level Element 表示如下。

Block-level Elements

ADDRESS - Address
BLOCKQUOTE - Block quotation
CENTER - Centered block
DIR - Directory list
DIV - Generic block-level container
DL - Definition list
FIELDSET - Form control group
FORM - Interactive form
H1 - Level-one heading
H2 - Level-two heading
H3 - Level-three heading
H4 - Level-four heading
H5 - Level-five heading
H6 - Level-six heading
HR - Horizontal rule
ISINDEX - Input prompt
MENU - Menu list
NOFRAMES - Frames alternate content
NOSCRIPT - Alternate script content
OL - Ordered list
P - Paragraph
PRE - Preformatted text
TABLE - Table
UL - Unordered list

註：下列節點在 HTML 的架構中也都含在 Block-level Element 中，也視為 Block-level Element。

DD - Definition description
DT - Definition term

FRAMESET - Frameset
LI - List item
TBODY - Table body
TD - Table data cell
TFOOT - Table foot
TH - Table header cell
THEAD - Table head
TR - Table row

Inline-level Elements

A - Anchor
ABBR - Abbreviation
ACRONYM - Acronym
B - Bold text
BASEFONT - Base font change
BDO - BiDi override
BIG - Large text
BR - Line break
CITE - Citation
CODE - Computer code
DFN - Defined term
EM - Emphasis
FONT - Font change
I - Italic text
IMG - Inline image
INPUT - Form input
KBD - Text to be input
LABEL - Form field label
Q - Short quotation
S - Strike-through text
SAMP - Sample output
SELECT - Option selector
SMALL - Small text
SPAN - Generic inline container
STRIKE - Strike-through text
STRONG - Strong emphasis
SUB - Subscript
SUP - Superscript

TEXTAREA - Multi-line text input
TT - Teletype text
U - Underlined text
VAR – Variable

註：下列節點中，可視為 Block-level Element 或是 Inline-level Element。視為 Inline-level Element 時，內層不得含有 Block-level Element。

APPLET - Java Applet
BUTTON - Button
DEL - Deleted text
IFRAME - Inline frame
INS - Inserted text
MAP - Image map
OBJECT - Object
SCRIPT - Client-side script