A Fast and Space-Economical Algorithm for the Tree Inclusion Problem

Yangjun Chen
University of Winnipeg, Canada

Yibin Chen
University of Winnipeg, Canada

INTRODUCTION

Let T be a rooted tree. We say that T is *ordered* and *labeled* if each node is assigned a symbol from an alphabet Σ and a left-to-right order among siblings in T is specified. Let v be a node different of the root in T with parent node u. Denote by $delete(T, v)$ the tree obtained by removing the node v from T, by which the children of v become part of the children of u as illustrated in Figure 1.

Given two ordered labeled trees P and T, called the pattern and the target, respectively. We may ask: Can we obtain pattern P by deleting some nodes from target T? That is, is there a sequence $v_1,...,v_k$ of nodes such that for

$T_0 = T$ and

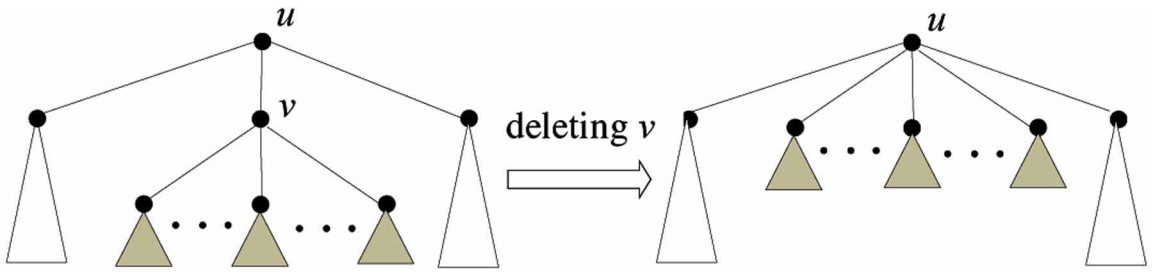
$T_{i+1} = delete(T_i, v_{i+1})$ for $i = 0,..., k - 1$,

we have $T_k = P$? If this is the case, we say, P is included in T (Kilpeläinen and Mannila, 1995). Such a problem is called the *tree inclusion problem*. It has many applications in the computer engineering as described below.

BACKGROUND

The first interesting application of the tree inclusion is used as an important query primitive for XML data (Mannila and Räiha, 1990), where a structured document database is considered as a collection of parse trees that represent the structure of the stored texts and the tree inclusion is used as a means of retrieving information from them. As an example, consider the tree shown in Figure 2, representing an XML document for the book *Arts of Programming* authored by (Knuth, 1969). One might want to find this book in an XML database by forming a pattern tree as shown in Figure 3 as a query, which can be obtained by deleting some nodes from the tree shown in Figure 2. Thus, a tree inclusion checking needs to be conducted to evaluate this query.

Figure 1. Illustration of node deletion



DOI: 10.4018/978-1-5225-2255-3.ch391

Figure 2. A XML document (target) tree

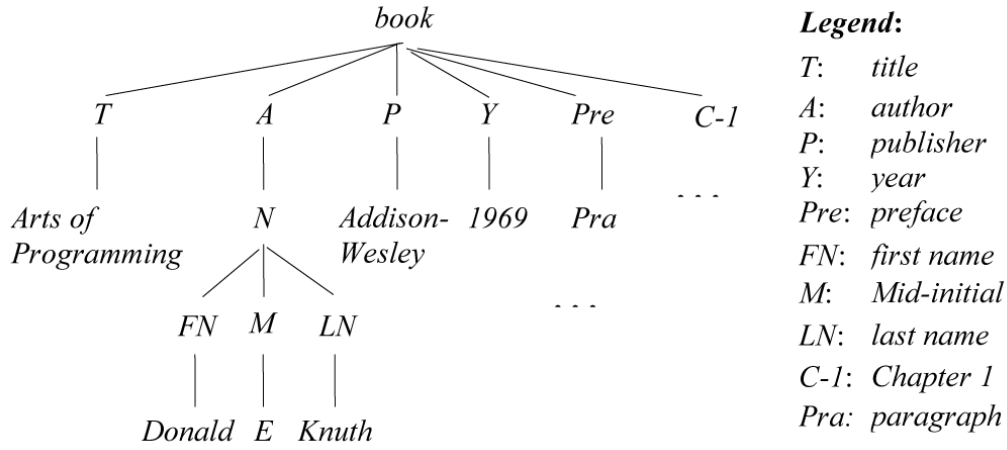
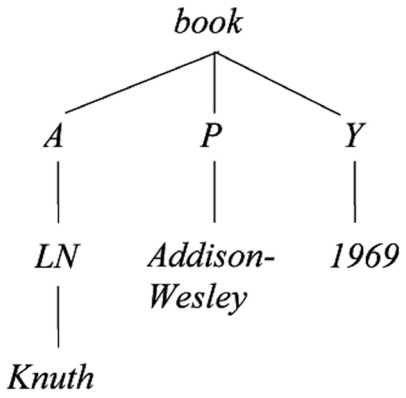


Figure 3. A pattern tree



Another application of this problem is to query the grammatical structures of *English* sentences, which can also be represented as an ordered tree since a sentence can always be divided into several ordered components such as noun phases, verb phrases, and adverbs; and a noun phrase itself normally contains an article and a noun while a verb phrase may contain a verb, a noun phrase, an adverb, and so on. To check whether a concrete sentence is grammatically correct, we will represent it as a pattern tree and make a tree inclusion checking against some target grammatical tree structures.

A third application of the ordered tree inclusion is the video content-based retrieval. According to (Rui and Huang, 1999), a video can be successfully decomposed into a hierarchical tree structure, in which each node represents a scene, a group, a shot, a frame, a feature, and so on. Especially, such a tree is an ordered one since the temporal order is very important for video. Some other areas, in which the ordered tree inclusion finds its applications, are the scene analysis, the computational biology, such as *RNA* structure matching (Lyngs, Zuker and Pedersen, 1999), and the data mining, such as tree mining discussed in (Zaki, 2002), just to name a few.

Up to now, the best algorithm for this problem requires $O(|T| + |P|)$ space and $\Theta(|T| \cdot |\text{leaves}(P)|)$ time (Alonso and Schott, 1993; Chen, 1998; Chen and Chen, 2006; Bille and Gørtz, 2011), where leaves(P) stands for the set of the leaves of P.

In this chapter, we propose an efficient algorithm for this problem. Its time and space complexities are bounded by $O(|T| \cdot min\{h_P, |\text{leaves}(P)|\})$, and $O(|T| + |P|)$, respectively, where h_P is the height of P, defined to be the number of edges on the longest downward path from the root to a leaf node.

BASIC DEFINITIONS

In this section, we mainly define the notations that will be used throughout the paper. Let T be a