Information Processing Letters 113 (2013) 345–349

Contents lists available at SciVerse ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Building Cartesian trees from free trees with k leaves

Brian C. Dean *, Raghuveer Mohan

Clemson University, School of Computing, Division of Computer Science, Box 340974, Clemson, SC 29630, United States

ARTICLE INFO

Article history:
Received 28 August 2010
Received in revised form 25 February 2013
Accepted 28 February 2013
Available online 5 March 2013
Communicated by B. Doerr

Keywords:
Cartesian tree
Adaptive algorithms
Range queries
Data structures

ABSTRACT

One can build a Cartesian tree from an n-element sequence in $O(n)$ time, and from an n-node free tree in $O(n \log n)$ time (with a matching worst-case lower bound in the comparison model of computation). We connect these results together by describing a Cartesian tree construction algorithm based on a "bitonicity transform" running in $O(n \log k)$ time on a free tree with k leaves, noting that a path is the special case of a tree with just 2 leaves. We also provide a matching worst-case lower bound in the comparison model.

1. Introduction

One can define a *Cartesian* tree from either an n-element sequence or an edge-weighted n-node free tree. As shown in Fig. 1(a), we define the Cartesian tree arising from a sequence $A_1 \ldots A_n$ by placing its minimum element A_i at the root[1]; its left and right subtrees are recursively defined to be Cartesian trees of the subsequences $A_1 \ldots A_{i-1}$ and $A_{i+1} \ldots A_n$. The Cartesian tree in this case is a heap-ordered binary tree whose in-order traversal yields the original sequence $A_1 \ldots A_n$. We define the Cartesian tree of a free tree T similarly, as shown in Fig. 1(b). The root node of the Cartesian tree corresponds to the edge e of minimum weight in T, and its two children are Cartesian trees of the subtrees into which T splits upon removal of e. Internal nodes in the Cartesian tree correspond to edges in T, while leaves in the Cartesian tree correspond to nodes in T. Cartesian trees have a variety of algorithmic applications, mostly due to their use in relating *range minimum queries* (RMQs) with *lowest common ancestor* (LCA) queries (see, e.g., [4,11]). In both a sequence and a free tree, the answer to an RMQ along a subsequence or subpath corresponds to the answer of an LCA query in the Cartesian tree, as indicated in Fig. 1.

In this work, we address the problem of building a Cartesian tree. One can easily build a Cartesian tree in $O(n)$ time from an n-element sequence and in $O(n \log n)$ time from an n-node tree. Moreover, there is a matching $\Omega(n \log n)$ worst-case lower bound on the worst-case construction time of a Cartesian tree from a free tree in the comparison model, since the Cartesian tree of a star-shaped tree is a depth-n sorted path (see Fig. 2(a)), so the process of Cartesian tree construction can be used to sort. We connect the dots between these two cases, giving an $O(n \log k)$ algorithm for Cartesian tree construction from an n-node free tree with k leaves (and we provide a matching lower bound in the comparison model). Such an algorithm could be termed an "adaptive" algorithm with respect to k, in the same manner as adaptive sorting algorithms (see, e.g., [7]) gracefully scale in running time between $O(n)$ and $O(n \log n)$ depending on some auxiliary parameter beyond just the problem size n that characterizes the intrinsic hardness of an instance (e.g., number of inversions).

* Corresponding author.
E-mail address: bcdean@clemson.edu (B.C. Dean).

[1] For simplicity, let us assume throughout this paper that all numbers in our input are distinct. It is easy to extend the concepts and results in our discussion to the general case with duplicates present.

0020-0190/$ – see front matter
http://dx.doi.org/10.1016/j.ipl.2013.02.014

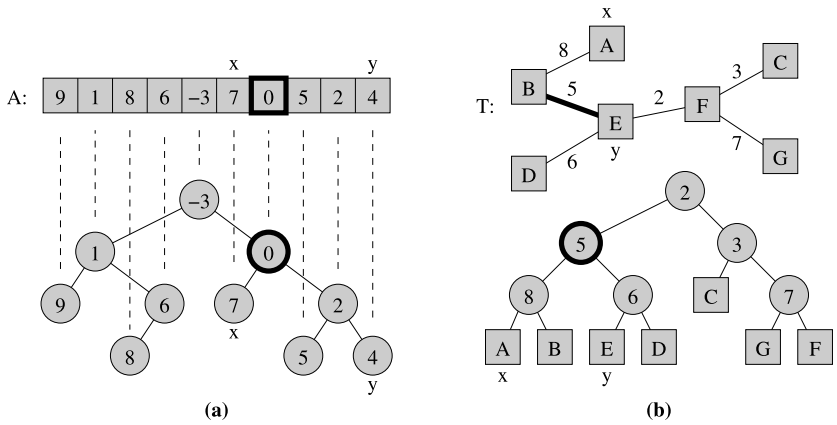

Fig. 1. Examples of (a) the Cartesian tree of a sequence $A_1 \ldots A_n$ and (b) the Cartesian tree of a free tree T. In both cases, we have highlighted the correspondence between a range minimum query along the path from x to y and the lowest common ancestor of x and y in the Cartesian tree (both shown in bold).

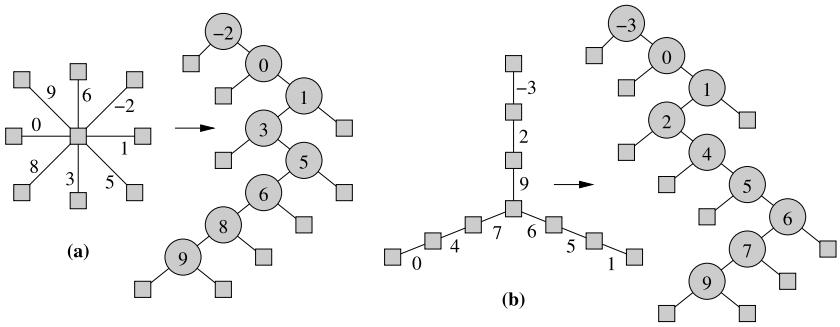

Fig. 2. Sorting via Cartesian tree construction from (a) a star, and (b) a spider with k sorted legs of length n/k. Note that the designation between left and right children is not particularly relevant when building a Cartesian tree from a free tree, so there are many path-shaped Cartesian tree shapes that could be valid above.

2. Background

The Cartesian tree of a sequence was initially introduced by Vuillemin [12]. It is a close relative of the *treap* [1], another hybrid between a binary heap and a binary search tree. Gabow et al. [9] first showed how to build a Cartesian tree from a sequence in $O(n)$ time using a simple inductive approach: starting with a Cartesian tree representing the sequence $A_1 \ldots A_{i-1}$, we obtain the Cartesian tree representing $A_1 \ldots A_i$ by inserting A_i at the bottom of the right spine and rotating it upward until we have restored the heap property. This approach spends only 2 units of work per element, 1 when it is inserted and another 1 later on when it is potentially rotated off the right spine permanently. Bender and Farach-Colton [2] give a particularly clear description of the use of Cartesian trees in relating RMQ problems in sequences with LCA problems. In particular, they give a simple approach for solving either problem with $O(n)$ preprocessing time and $O(1)$ query time (a result first achieved by Harel and Tarjan [10]). Reductions between LCA and RMQ problems are also described in the early literature in [3] and [9].

Cartesian trees of free trees were introduced by Chazelle [5] and then subsequently rediscovered by Demaine et al. [6], who note that the $\Omega(n \log n)$ comparison-based lower bound on their worst-case construction time applies even to trees of bounded degree, and also describe how to build a Cartesian tree from a free tree in only $O(n)$ time after first sorting its edge weights as a preprocessing step. It is useful to note that the Cartesian tree of a free tree T reflects precisely the hierarchical structure of the merging operations performed by Kruskal's minimum (or rather maximum, in this case) spanning tree algorithm, when executed on T. For this reason, the authors suggest that the term "Kruskal tree" might also be well-suited for describing such a Cartesian tree.

3. Lower bounds

Theorem 1. *In the comparison model, there is an $\Omega(n \log k)$ worst-case lower bound for building a Cartesian tree from an n-node free tree with k leaves.*

Proof. The Cartesian tree resulting from a "spider" with k incident sorted paths of length n/k is one long sorted path (Fig. 2(b)). Hence, any algorithm for Cartesian tree construction from an n-node free tree with k leaves can be used to solve the more fundamental problem of merging k sorted lists on n total elements. Moreover, this k-way merging problem has an $\Omega(n \log k)$ worst-case lower