On the Bonsai Structure of Green-Brown Trees

Ethan Cecchetti
Cornell University
ethan@cs.cornell.edu

Abstract

There are a wide variety of tree-based data structures with widely varying properties and performance characteristics. Structures such as AVL trees and red-black trees provide strong guarantees about the structure and performance, but are often difficult to understand and provide very little leeway to maintainers. Conversely, Bonsai trees provide maintainers with a nearly complete ability to control the structure and layout of the trees, but are correspondingly difficult to maintain and require enormous resource allocations.

To alleviate many of these problems, we introduce *green-black trees*, a new data structure that combines the simplicity and lightweight properties of common binary trees with the flexibility and control of Bonsai trees. Like in the Bonsai case, maintainers can easily determine not just the level of balance, but the exact shape of the tree. It is similarly easy to identify the exact location of leaf data. Unlike Bonsai trees, however, green-black trees achieve these results through a very simple set of invariants that are nearly trivial to maintain programmatically.

1 Introduction

In both computing and elsewhere trees are nearly ubiquitous. They are used to store data in a variety of cases and are evident in nearly every setting. Many classic binary tree structures are, however, geared specifically towards storing data that is efficiently comparable. For example, AVL trees and red-black trees both provide strong assurances on the maximum runtime of various operations, but do so by restricting the shape and layout of the tree itself. While this may be appropriate for many applications, sometimes it is necessary to provide the maintainer with more precise control over the shape of the tree itself.

Other trees, such as Bonsai trees, provide extremely fine-grained control over their shape and appearance to managers. With appropriate setup and a watchful hand, nearly anything is possible. Unfortunately these trees are extremely complex, difficult to maintain, and resource-intensive.

We attempt to bridge this gap with the introduction of the *green-brown tree*. These trees are lightweight data structures that are easy to understand and manage, yet provide many of the same guarantees as Bonsai trees.

2 Green-Brown Trees

Green-brown trees are a lightweight data structure that provide a large amount of flexibility and configurability. They are use minimal resources and still allow for extremely precise control over the balance and shape of the tree and location of any data stored within.

In order to achieve these goals, we borrow a core idea from red-black trees: we color each node in the tree one of two values—in this case green or brown. We then maintain the following three invariants on the tree nodes based on color:

1. All leaves are green.
2. All nodes with a green child are brown.
3. All color changes are local.

The first two invariants are straightforward. For the third, we mean that whenever a node is moved, only that node and its direct parent and children can change color. Since we primarily insert new nodes as leaves, this means that the new leaf must be green and we must always color its parent brown upon insertion.

2.1 Bonsai Properties

We note that Bonsai trees are carefully maintained to always have green (external) leaves, while keeping all internal structure brown. The three invariants mentioned above create a simple mechanism for maintaining exactly this property in green-brown trees.

Moreover, Bonsai trees allow for nearly any relative placement of nodes in the tree and thus any level of balance (or imbalance). Similarly, green-brown trees allow

the data manager to structure the tree in any way he or she desires. It is possible to create a perfectly balanced tree, one that is so imbalanced that it is hard to immediately identify as a tree, or anything in between.

3 Related Work

There are numerous varieties of trees and many are quite common in different areas.

Redwoods provide enormous storage capacity and are highly reliable. The leaves, however, start very deep into the tree thus wasting usable space closer to the root and requiring a large amount of time to access anything therein.

Pine trees look superficially similar to redwoods, but are actually quite different. They are naturally balanced, giving the maintainer less precise control over the tree layout. Moreover, unlike green-brown trees (and Bonsai trees), they have internal green nodes with often make things difficult to locate and understand.

Oak trees, much like pines, are extremely prevalent and naturally quite balanced. They do not, however, produce very efficient access as the tree fans out massively very close to the root, thereby losing many of the convenient properties of binary trees.

Finally, maple trees present some interesting properties. They are not inherently balanced or wasteful in the ways that redwoods or pines are, but they have extremely unstable leaves. The leaves will periodically change color arbitrarily, which will it looks impressive, is actually very difficult to track and significantly hurts reliability. Worse, maple's sometimes drop leaves with very little warning, which can cause catastrophic data loss. While maples are efficient, this problem requires expensive safeguards and can make them very difficult to employ in practice. Also they are sometimes sticky. The stuff tastes good, but that is not relevant to our use-case.

4 Conclusion Haiku

A green-and-brown tree
can be any shape or size
without work. Bonsai!

Acknowledgements

The author would like to acknowledge the conference chair for encouraging this work. He would also like to thank numerous people in the Cornell Computer Science department for their ideas and input during the process.