Multi-splay trees and tango trees

Christian Altamirano

Parker Rule

May 19, 2021

Dynamic optimality

Dynamic optimality

Consider a static universe of keys. Every access sequence *X* has an optimal BST that requires OPT(*X*) unit cost operations.

Dynamic optimality

Consider a static universe of keys. Every access sequence X has an optimal BST that requires OPT(X) unit cost operations.

A BST data structure is *dynamically optimal* if every sequence X requires O(OPT(X)) operations.

Dynamic optimality

Consider a static universe of keys. Every access sequence *X* has an optimal BST that requires OPT(*X*) unit cost operations.

A BST data structure is *dynamically optimal* if every sequence *X* requires *O*(OPT(*X*)) operations.

Recall: it is conjectured that splay trees are dynamically optimal (Sleator and Tarjan 1985). However, this conjecture remains unproven.

O(log log n)-competitive BSTs

O(log log n)-competitive BSTs

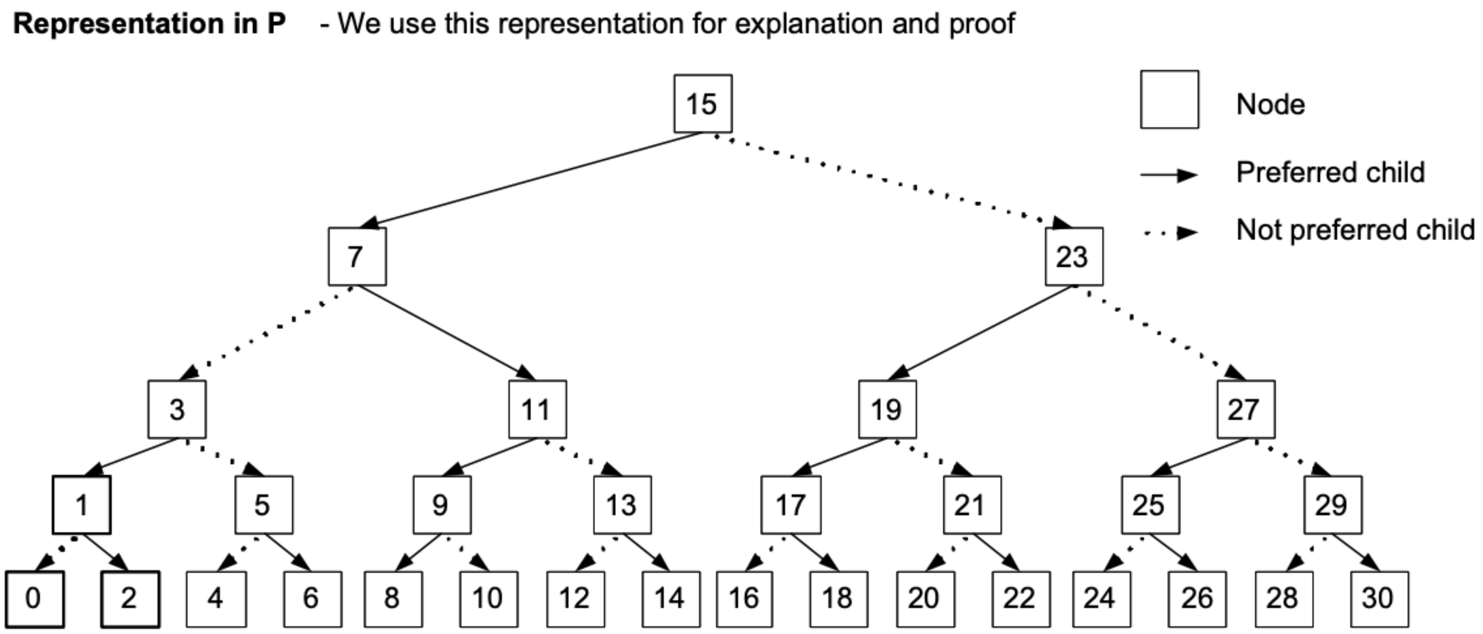
Tango trees (Demaine et al. 2005)**:** preferred paths of length O(log n) are represented as red-black trees in a tree of trees.

O(log log n)-competitive BSTs

Tango trees (Demaine et al. 2005)**:** preferred paths of length O(log n) are represented as red-black trees in a tree of trees.

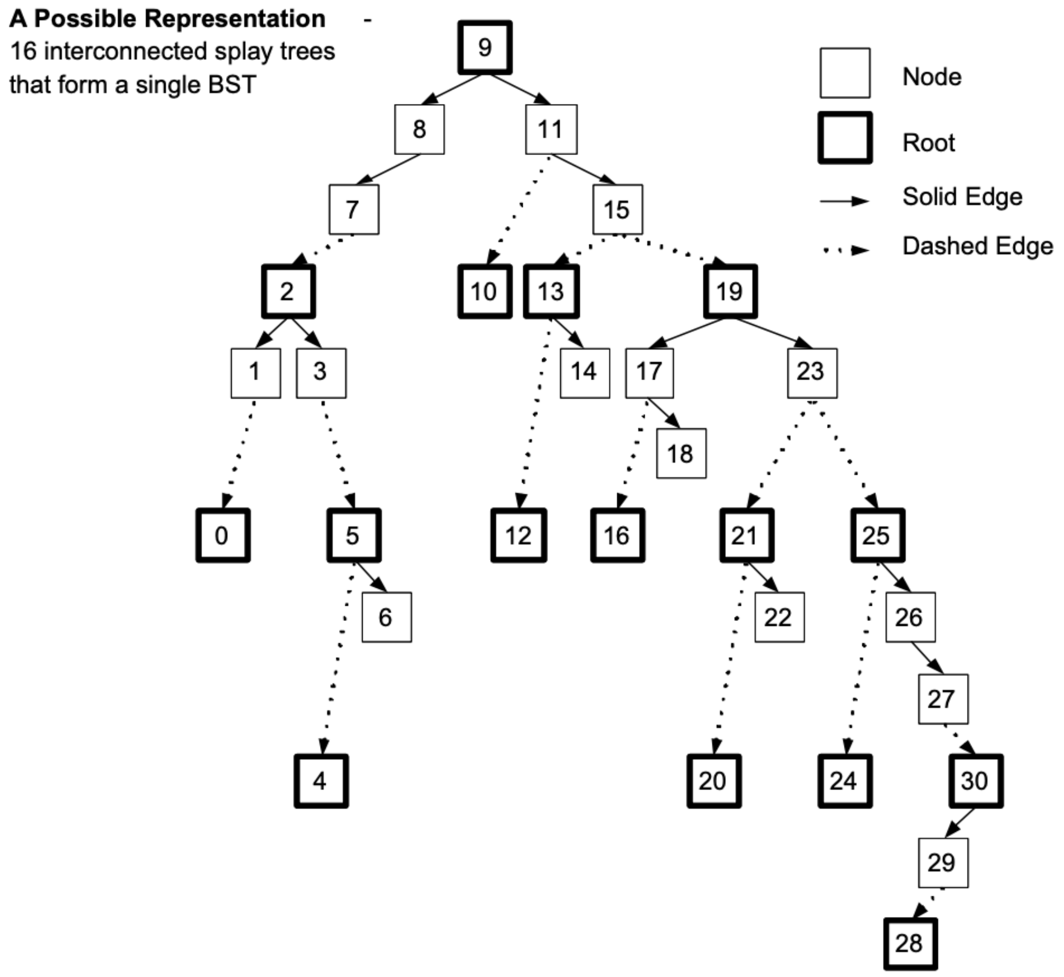
Multi-splay trees (Wang, Derryberry, and Sleator 2006): Somewhat similar to tango trees, but preferred paths are represented as splay trees instead. Better (amortized) worst-case performance.

Multi-splay trees



Source: Sleator and Wang 2004

Multi-splay trees



Source: Sleator and Wang 2004

Multi-splay implementation

Multi-splay implementation

Augmented tree (of splay trees) with Query method implemented

Multi-splay implementation

Augmented tree (of splay trees) with Query method implemented

All modifications to the tree are made via splaying

Multi-splay implementation

Augmented tree (of splay trees) with Query method implemented

All modifications to the tree are made via splaying

Only support static trees, augmentations depend on a perfect binary search tree

Multi-splay Query implementation

Multi-splay Query implementation

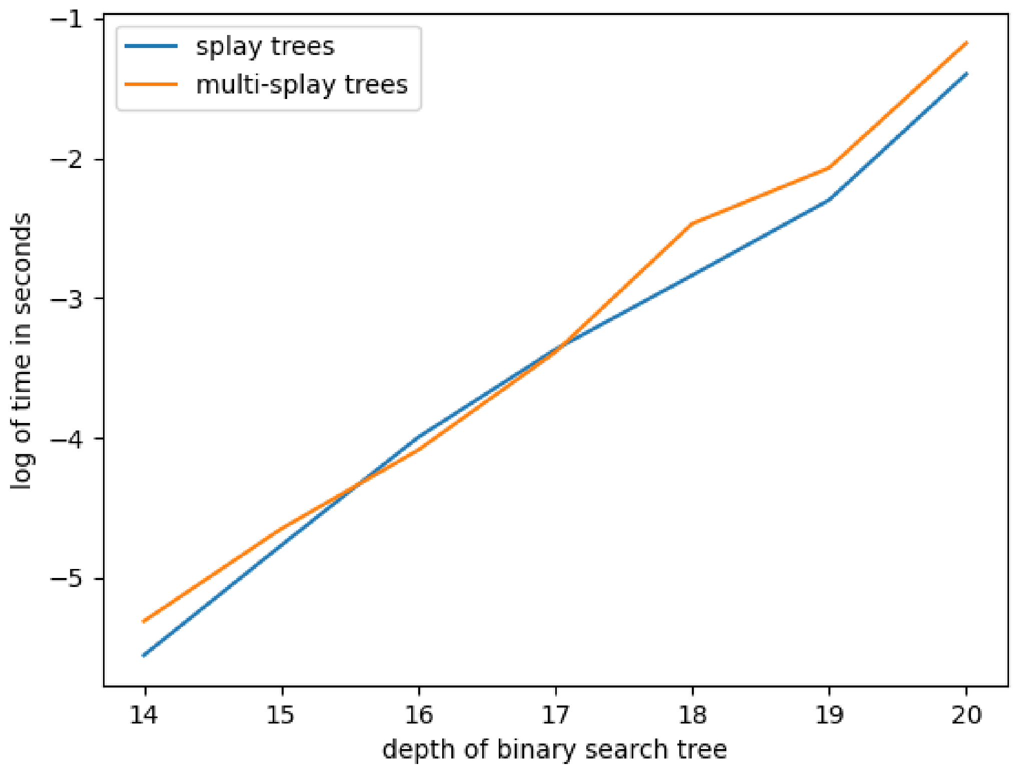
Left2Right(node) and Right2Left(node)
change the preferred child of node

Multi-splay Query implementation

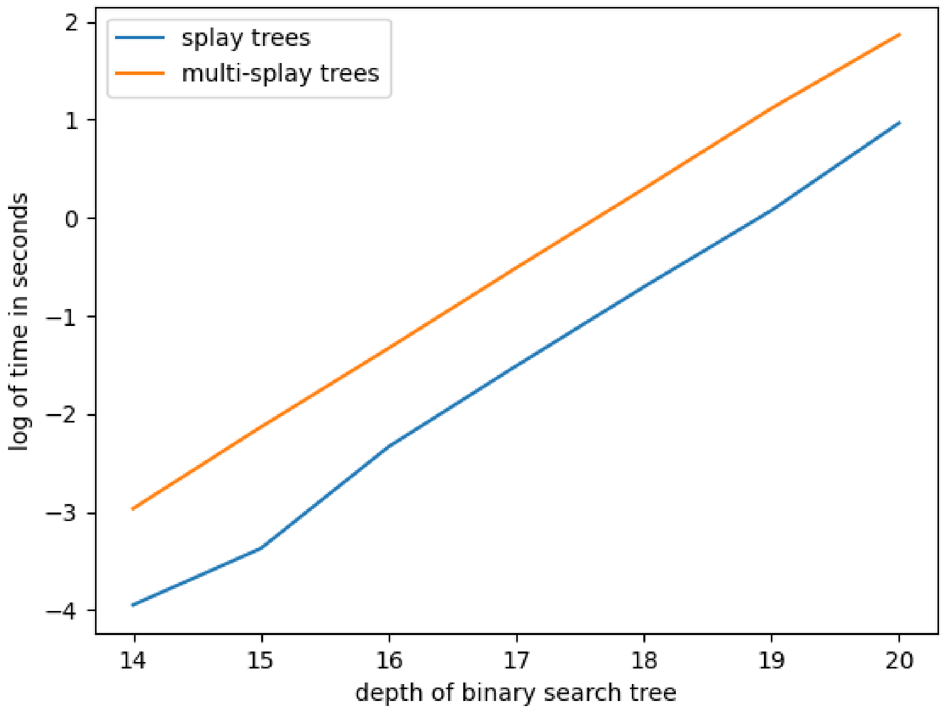
Left2Right(node) and Right2Left(node)
change the preferred child of node

Change the preferred children top-down
and splay such nodes

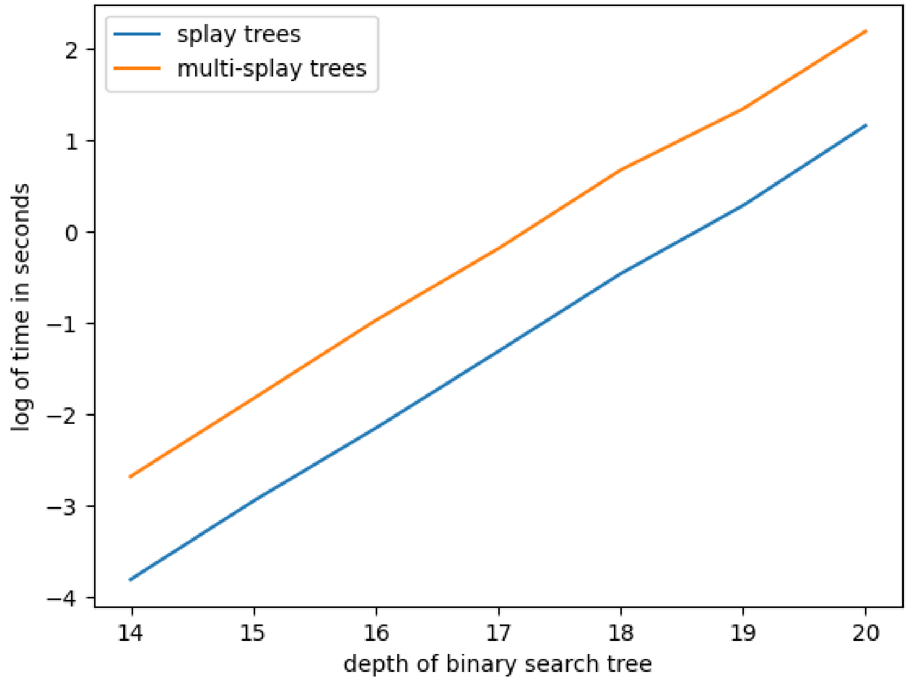
Multi-splay benchmarks: sequential access



Multi-splay benchmarks: random access



Multi-splay benchmarks: bit reversal sequence



Tango trees

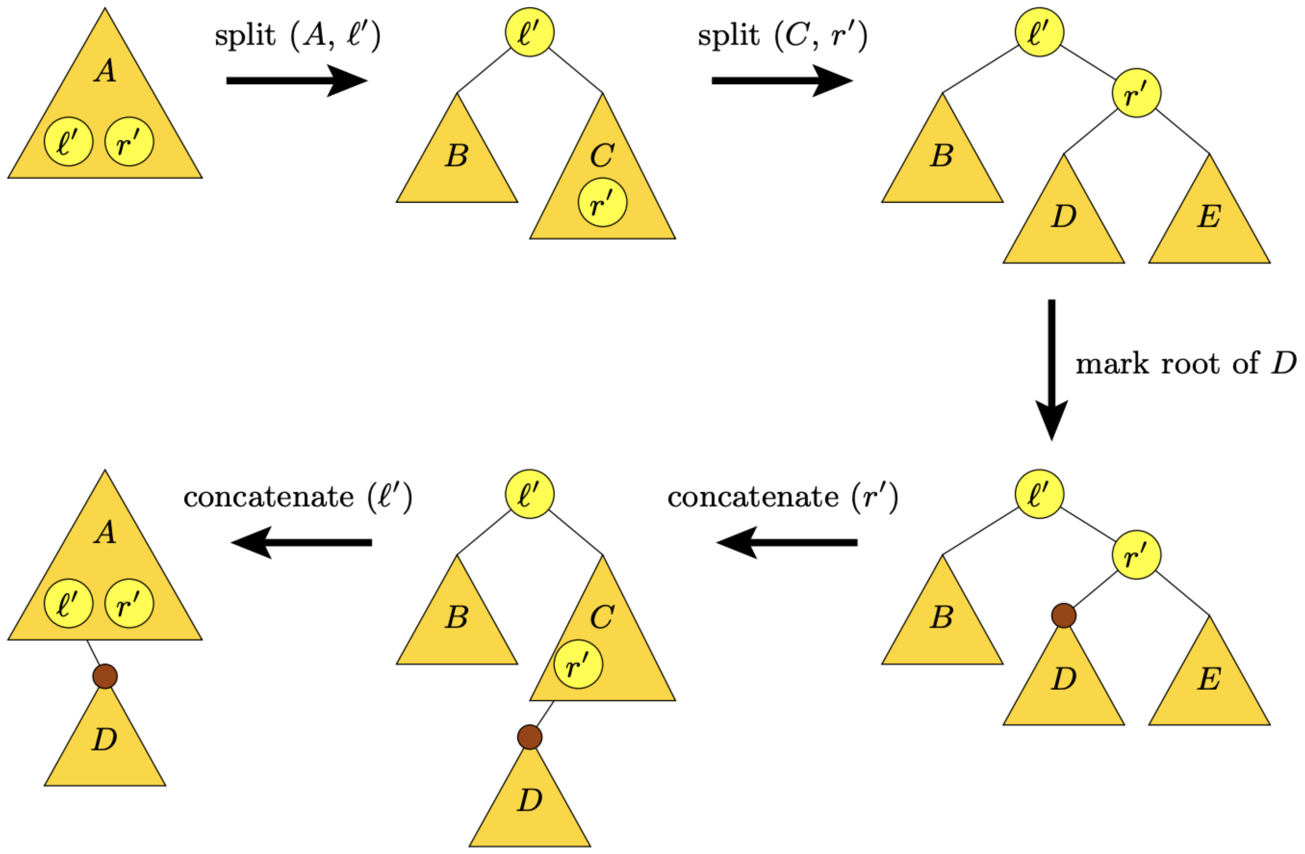


FIG. 3.1. *Implementing cut with split, mark, and concatenate.*

Source: Demaine et al. 2005

Tango implementation progress

Tango implementation progress

Augmented red-black tree (of trees) with split/concatenate implemented

Tango implementation progress

Augmented red-black tree (of trees) with split/concatenate implemented

API: lock() (read-only) and unlock() (write-only)

Tango implementation progress

Augmented red-black tree (of trees) with split/concatenate implemented

API: lock() (read-only) and unlock() (write-only)

Challenge: the implicit tree-of-trees representation makes everything trickier!

Tango implementation

TODO

Tango implementation TODO

(Seemingly) need to implement perfect tree → tango tree conversion

Tango implementation TODO

(Seemingly) need to implement perfect tree → tango tree conversion

After this, finishing cut/join should be relatively simple

Tango implementation TODO

(Seemingly) need to implement perfect tree → tango tree conversion

After this, finishing cut/join should be relatively simple

Empirical work: benchmarks against multi-splay implementation and the like

Thank you!