New Paths From Splay to Dynamic Optimality

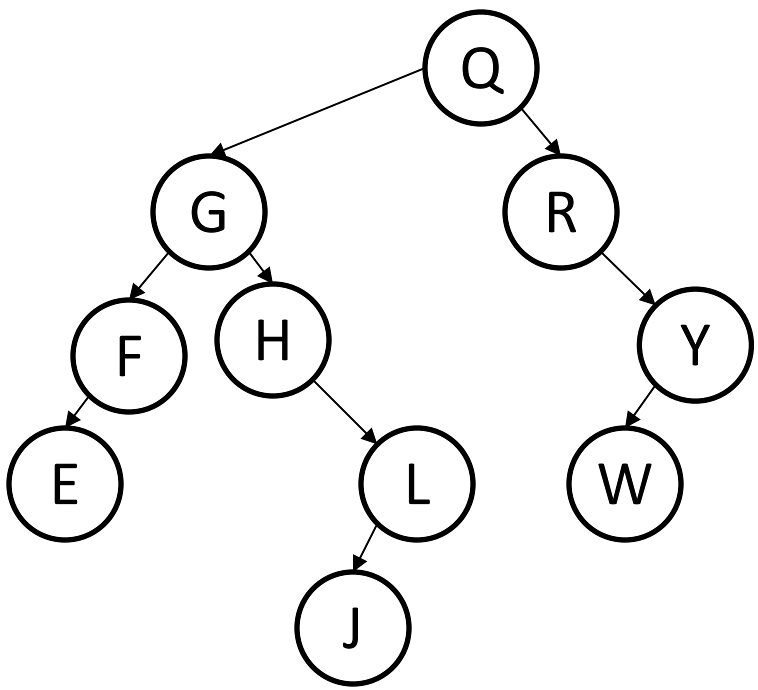
Caleb Levy

FPO: *Ph.D. in Applied and Computational Math*
Princeton University

May 15, 2019

Examiners: Robert Tarjan (*Advisor*)
Bernard Chazelle
Robert Sedgewick
Kurt Mehlhorn (*Reader*)

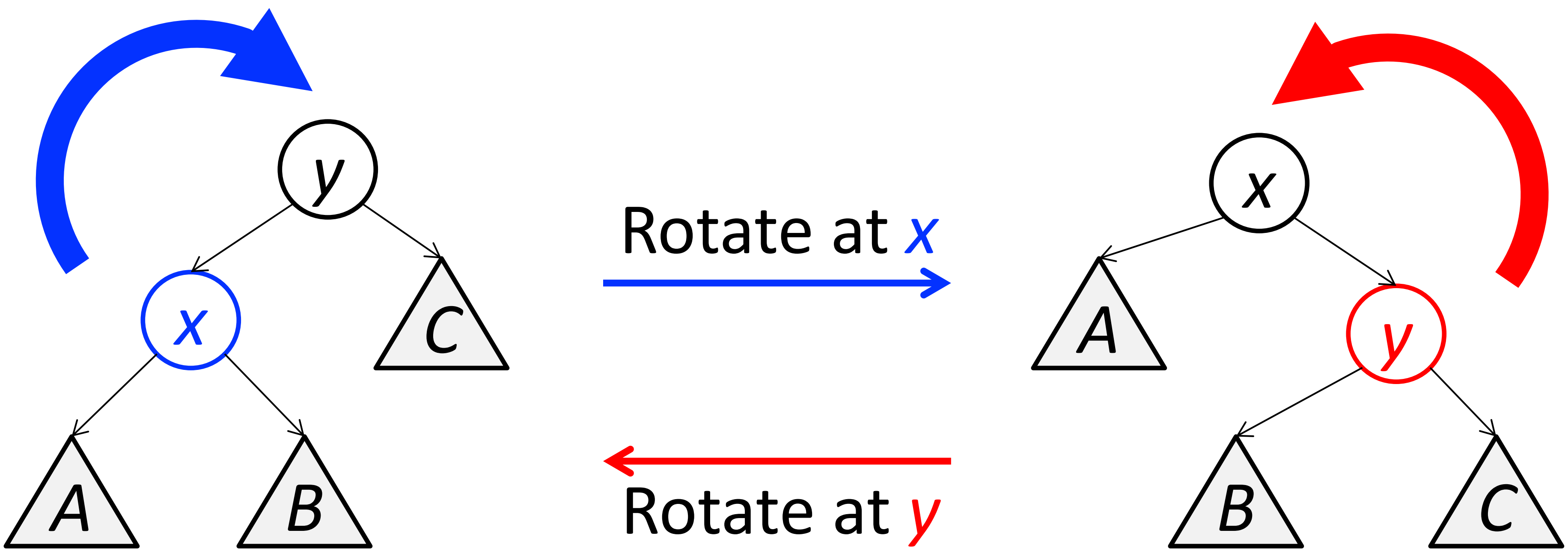
Binary Search Tree



- Root node
- All nodes have left and right children (may be null)
- Symmetric Order

Rotation

A rotation child x with parent y makes y a child of x while preserving symmetric order, and changes $O(1)$ pointers



The Splay Algorithm

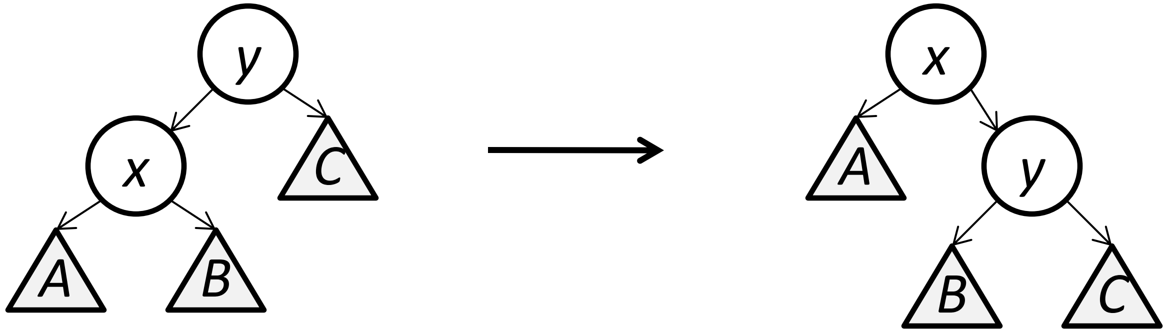
The canonical self-adjusting BST

splay(*x*): Search for *x*, then repeatedly perform a zig, zig-zag, or zig-zig at *x* until it becomes the root

Splaying “spreads out” nodes of the access path along way

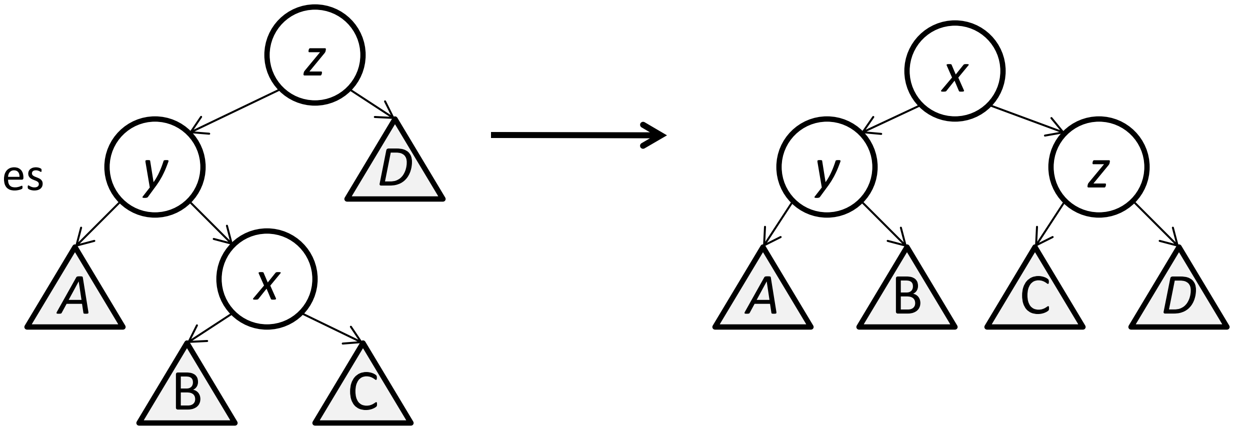
zig

Rotate at *x*



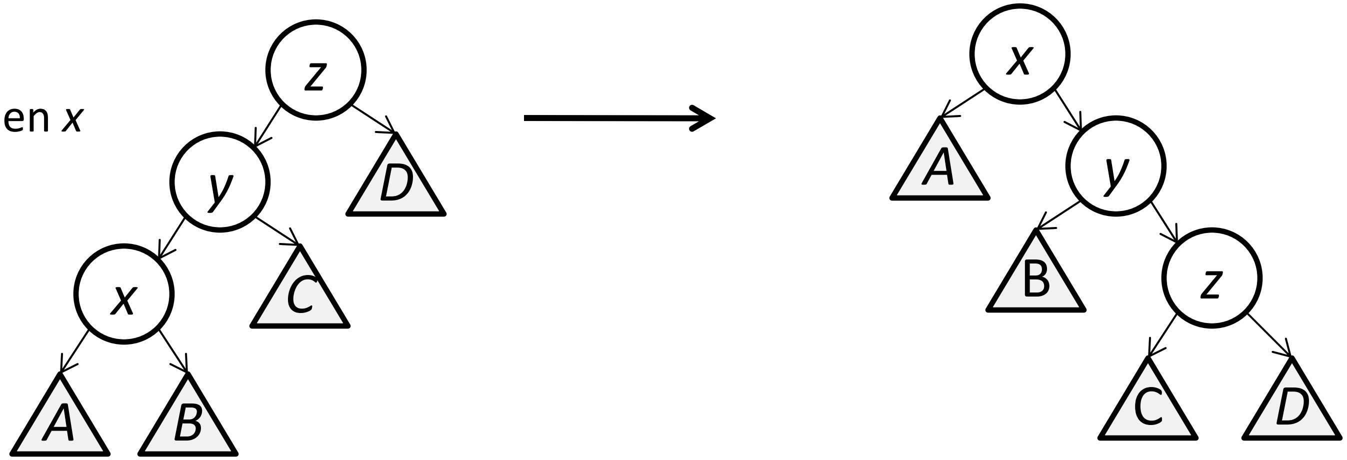
zig-zag

Rotate at *x* two times

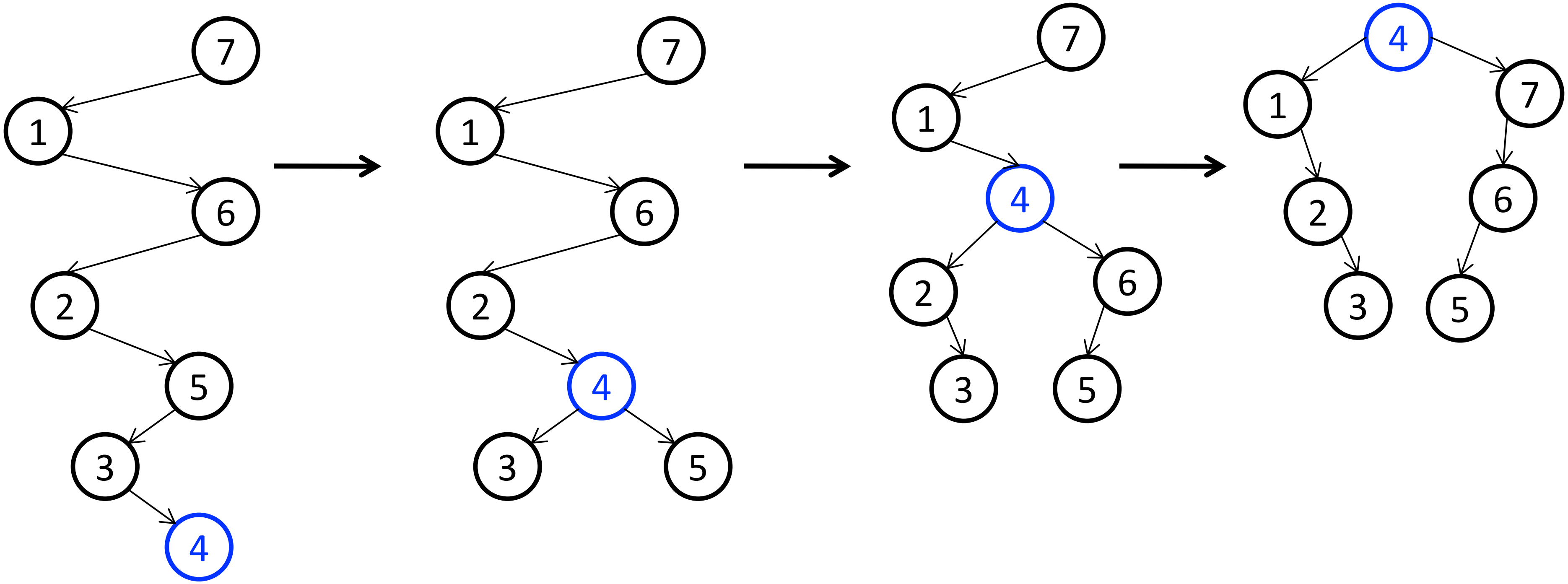


zig-zig

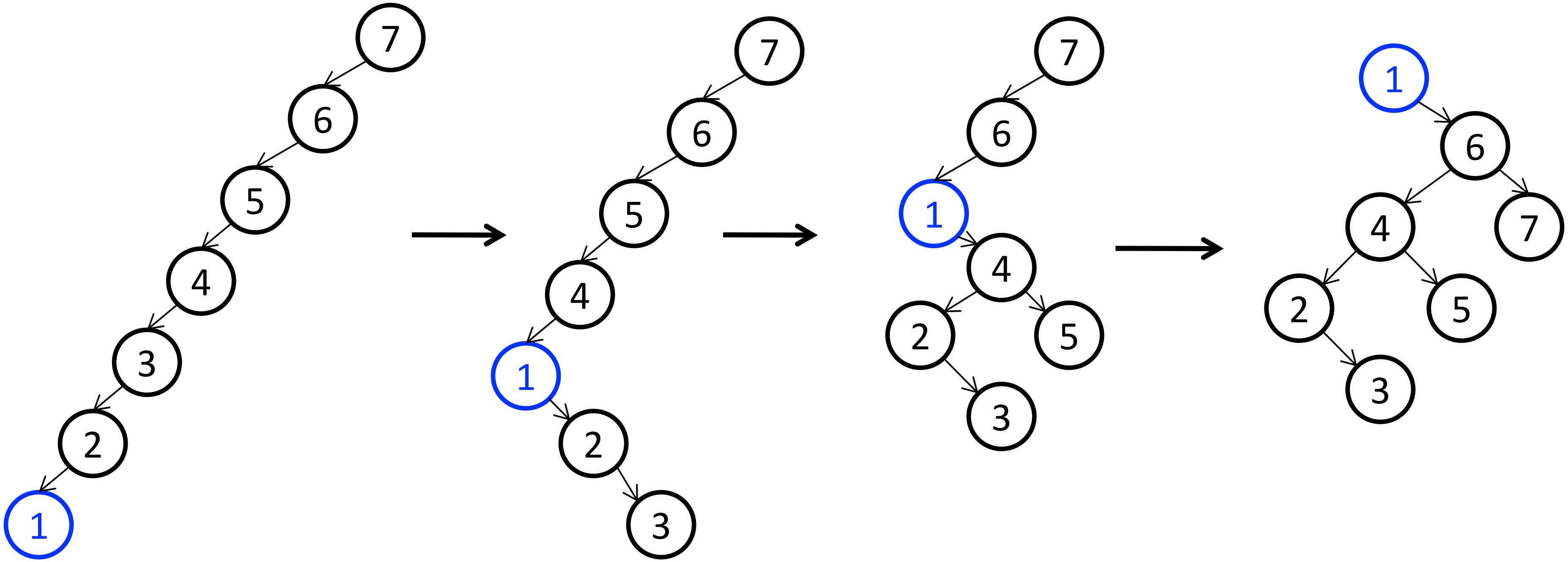
Rotate at *y*, then *x*



Splay: pure zig-zag



Splay: pure zig-zig

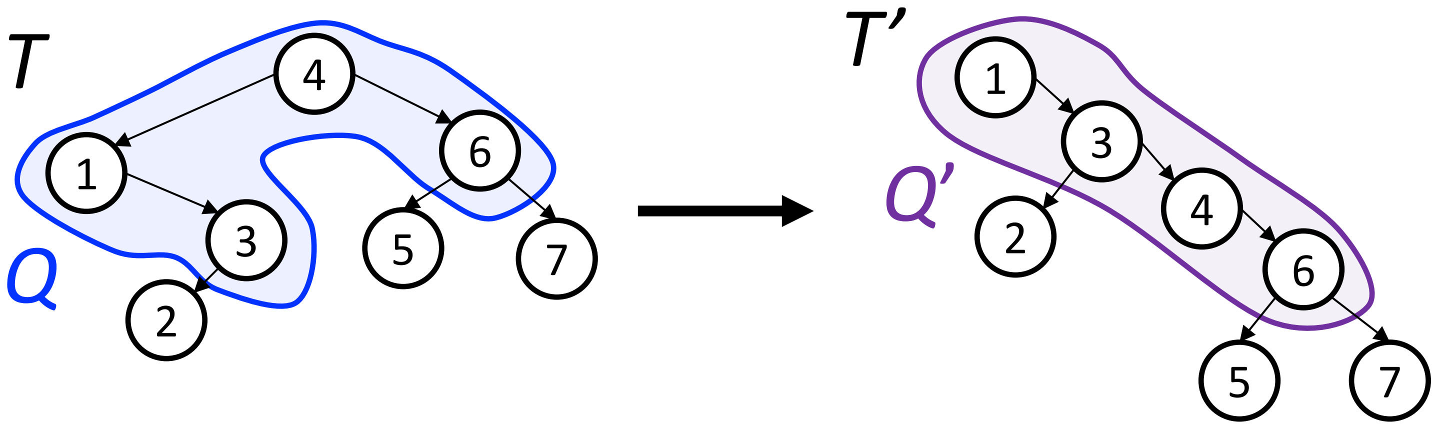


BST ALGORITHMS

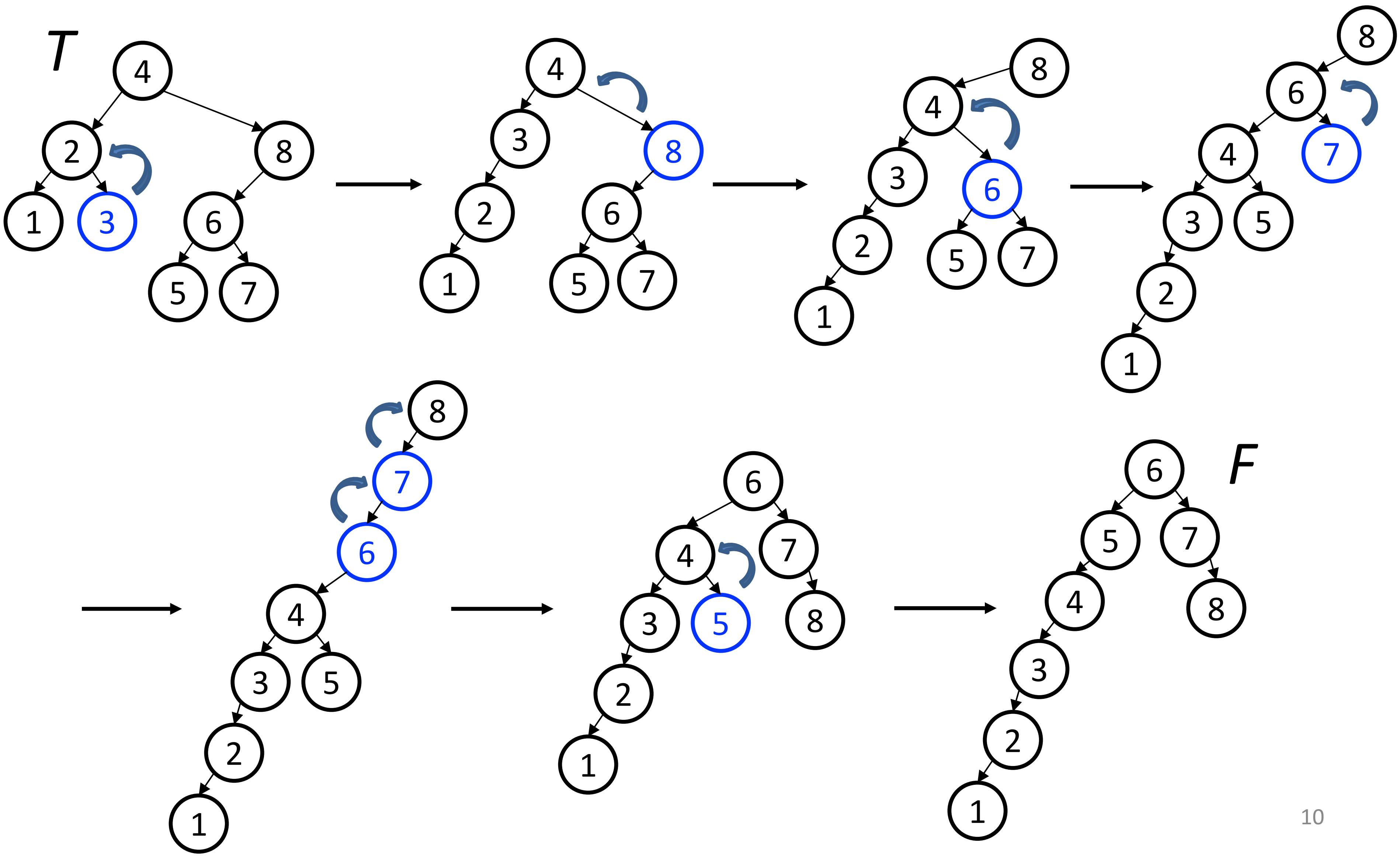
Subtree Transformations

To perform a **subtree transformation** on *T*:

1. Select a connected subtree *Q* with root of *T*
2. Reshape *Q* into a new tree *Q'* with same keys
3. Form *T'* by substituting *Q'* for *Q* in *T*



Transforming with Rotations



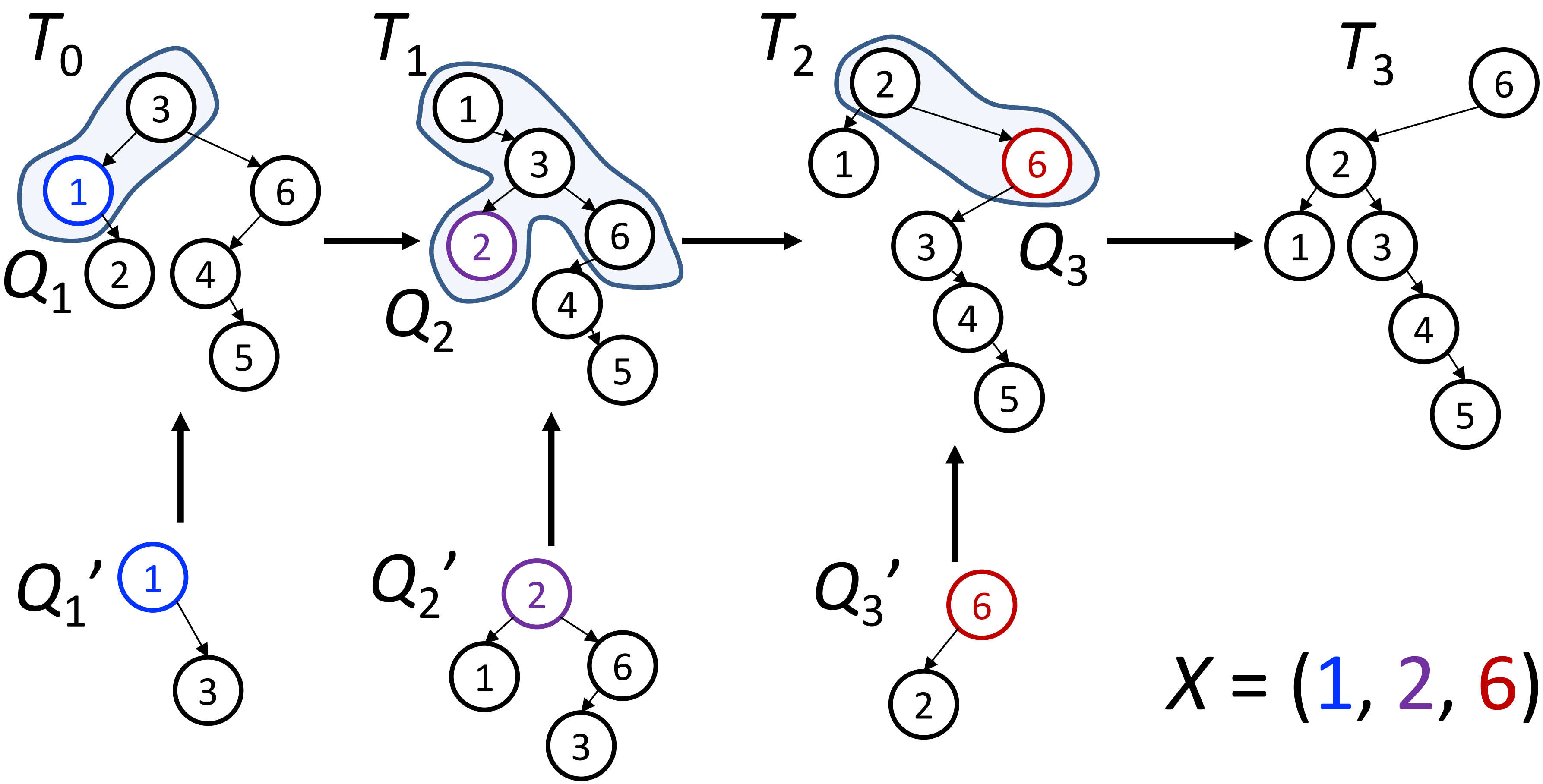
Formal Definition of Executions

An instance is a sequence $X = (x_1, x_2, \ldots, x_m)$ of requested items in specified initial tree $T = T_0$

An execution E is a sequence of subtree transformations $(Q_1 \rightarrow Q_1')$, …, $(Q_m \rightarrow Q_m')$ and after-trees $T_1, \ldots, T_m$ where each transformation brings x_i to the root

The cost of execution E with subtrees $Q_1, \ldots, Q_m$ is the sum of the subtree sizes, $|Q_1| + \ldots + |Q_m|$

Binary Search Tree Executions



$X = (1, 2, 6)$

The Optimal Execution

At least one execution for *X* starting from *T* will have minimum, or optimum, value:

$$\mathrm{OPT}(X,T) = \min_{E \text{ for } X,T} \operatorname{cost}(E)$$

Dynamic Optimality Conjecture

An algorithm A maps instances to executions

An algorithm is constant-competitive, or dynamically optimal, if for some $c > 0$:

$$\mathrm{cost}_A(X, T) \leq c \cdot \mathrm{OPT}(X, T)$$

for all X, T

Conjecture (ST 85): *Splay is dynamically optimal*

State of Knowledge

- Tango trees known to be O(log log *n*) competitive with OPT (DHIPM et. al. 07)
- An *off*-line Greedy algorithm conjectured to be optimal (Lucas 89, Munro 00), later developed into an *on*-line "geometric" version (DHIKPM 09)
- A "meta-algorithm" is known to be optimal if *any* on-line algorithm is (Iacono 13)
- Some lower bounds, none known tight

Strategy: *Why* is this hard?

Difficult to characterize OPT's behavior

- Small changes in present could affect future arbitrarily
- Exact computation of OPT likely NP-Complete (Demaine et. al. 2008)
- No P-time algorithm is known to compute OPT to within a constant factor

Let's eliminate OPT from the picture!

Main Contributions

1) Splay is dynamically optimal $\leftrightarrow$ Splay is *approximately monotone*
2) Splay is optimal $\rightarrow$ Splay has no additive overhead
3) Wilber's "crossing" bound is approximately monotone
4) A proposal for how to adapt the proof of monotonicity from the lower bound to Splay

1A) MONOTONICITY → OPTIMALITY

Approximate Monotonicity

An algorithm A is approximately monotone (or has the "subsequence property") if there is $b > 0$ so that

$$\text{cost}_A(Y, T) \leq b \cdot \text{cost}_A(X, T)$$

for all request sequences X, subsequences Y of X, and initial trees T

E.g. (1, 3, 6) is a subsequence of (1, $\cancel{2}$, 3, $\cancel{5}$, 6)

Simulation Embeddings

A simulation embedding S for algorithm A is a map from executions to request sequences such that, for some $c > 0$

- $\text{cost}_A(S(E), T) \leq c \cdot \text{cost}(E)$, and
- X is a subsequence of $S(E)$

for all request sequences X, all initial trees T, and all executions E for the instance X, T

Monotonicity and Simulations

Theorem: Approximate-monotone algorithm A with a simulation embedding S is constant-competitive

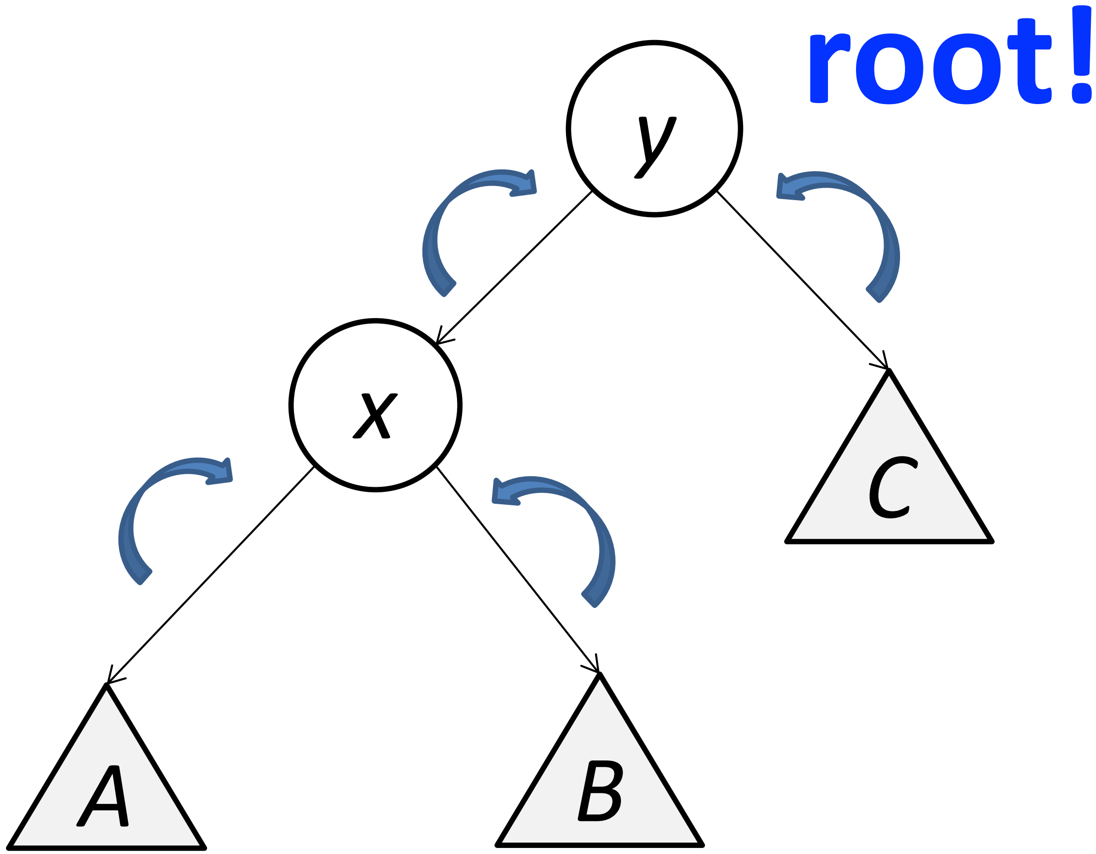
Proof: X is a subsequence of $S(E)$, where E is an *optimal* execution for which $\text{cost}(E) = \text{OPT}(X, T)$, thus $\text{cost}_A(X, T) \le b\cdot\text{cost}_A(S(E), T) \le b\cdot c\cdot\text{OPT}(X, T)$

Plan: Build a simulation embedding for Splay

TREE TRANSFORMS WITH SPLAY

Restricted Rotations

Rotations whose edges must contain the root, or the root's left child.



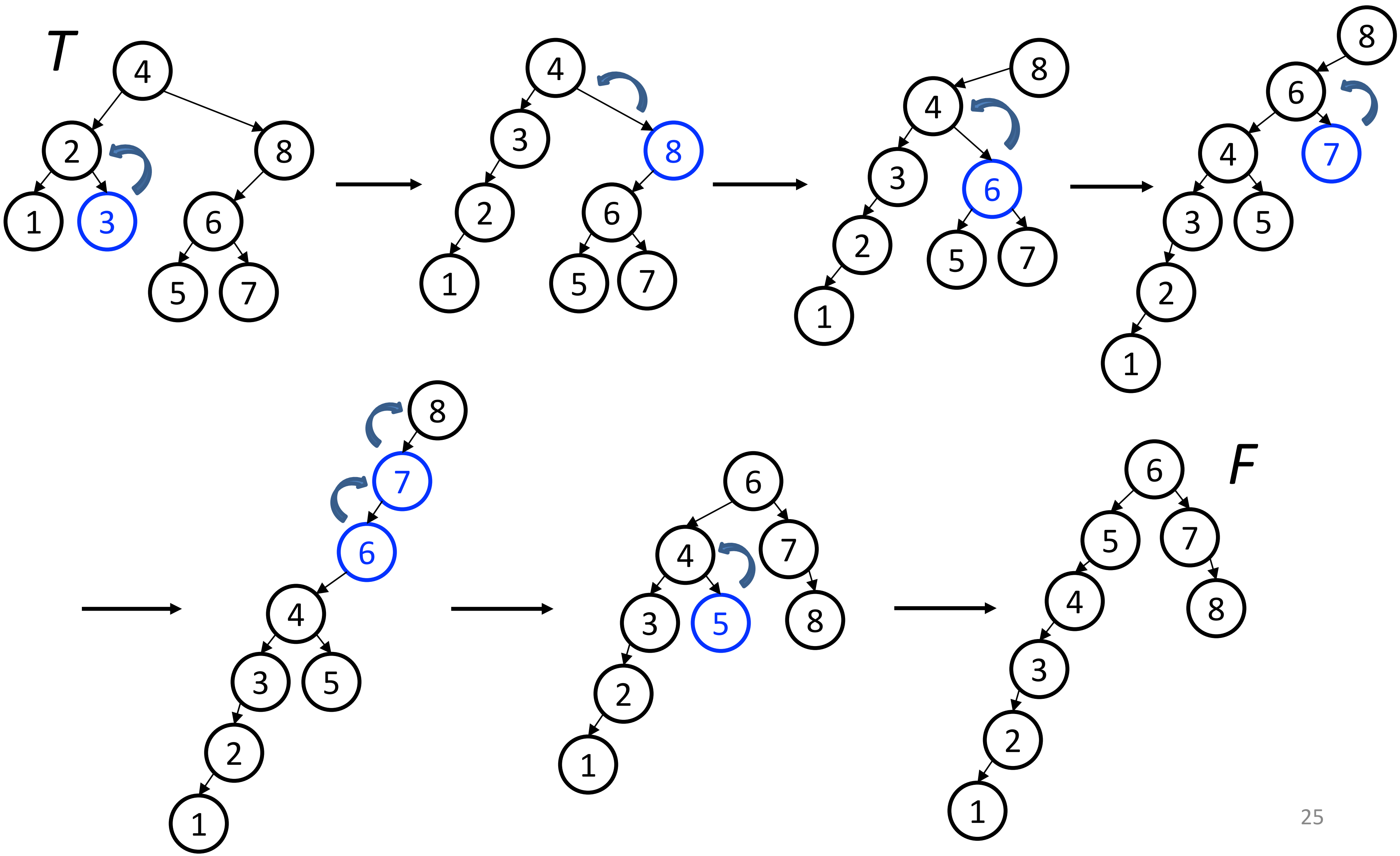
Restricted Rotation Distance

Theorem (Cleary 02, Lucas 04, Levy 16):
Any tree T_1 may be transformed into another tree T_2 on the same set of keys with at most $4n$ restricted rotations.

Proof: "Unwrap" T_1 into a "flat" tree, apply transformation of T_2 into a flat tree in reverse.

("flat" = no left child has a right child and no right child has a left child.)

Unwrapping

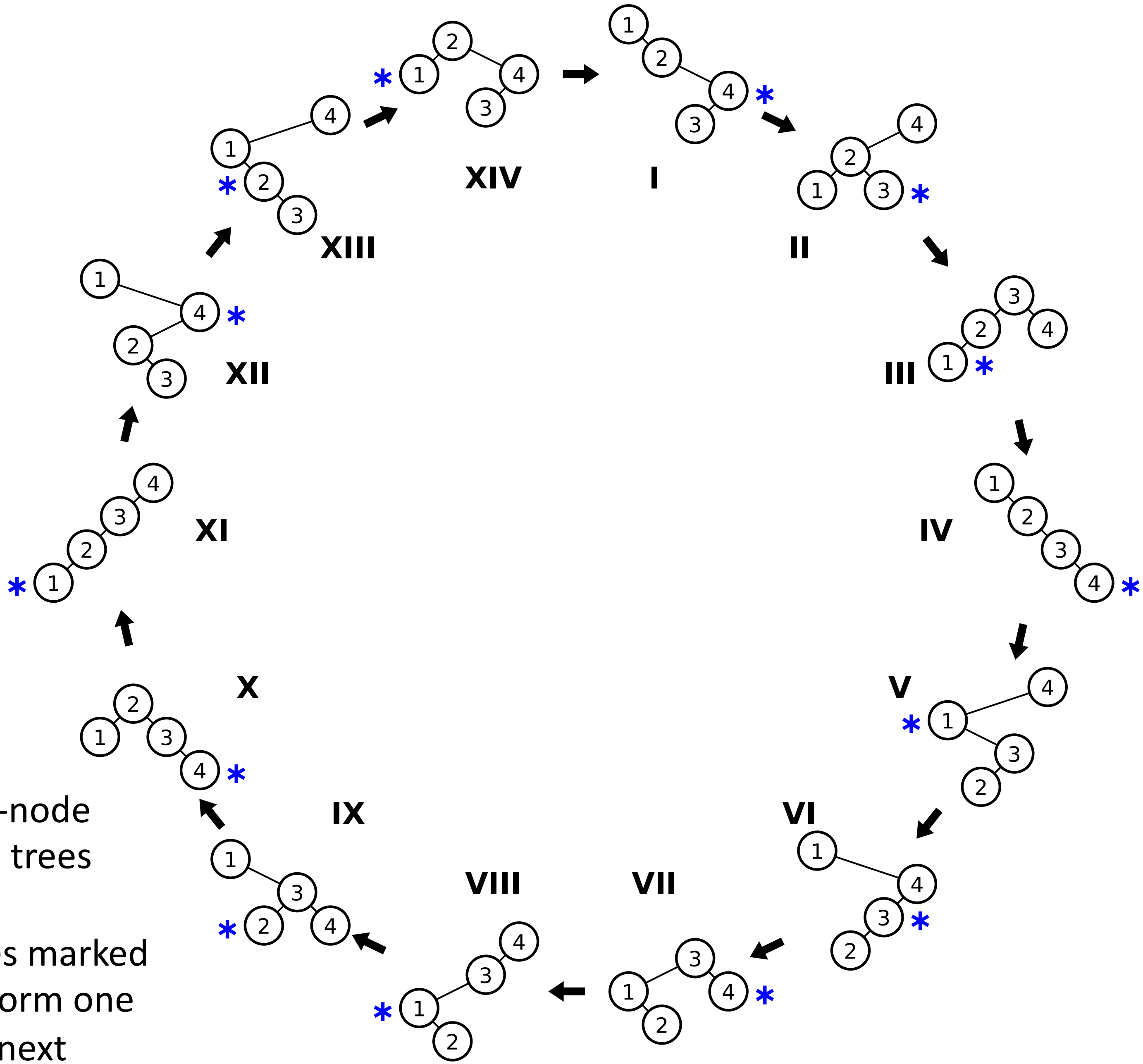


Splay's Transition Graph

Assign a vertex to each binary search tree with 4 nodes, and draw an arc from T_1 to T_2 if we can splay a key in T_1 to obtain T_2

Theorem: This transition graph is strongly connected (in fact its diameter is 5)

We can use this to enact restricted rotations



All possible 4-node binary search trees

Splay at nodes marked by * to transform one tree into the next

Splay Can Transform Subtrees

Theorem: There is a request sequence **T**(Q, Q') inducing Splay to change Q into Q' in linear time

Proof:

- $4n$ restricted rotations to change Q to Q'
- 5 splays to perform each restricted rotation
- Each splay path has length at most 4

Total cost: $4n*5*4 = 80n$ to change Q to Q'

A Simulation Embedding for Splay

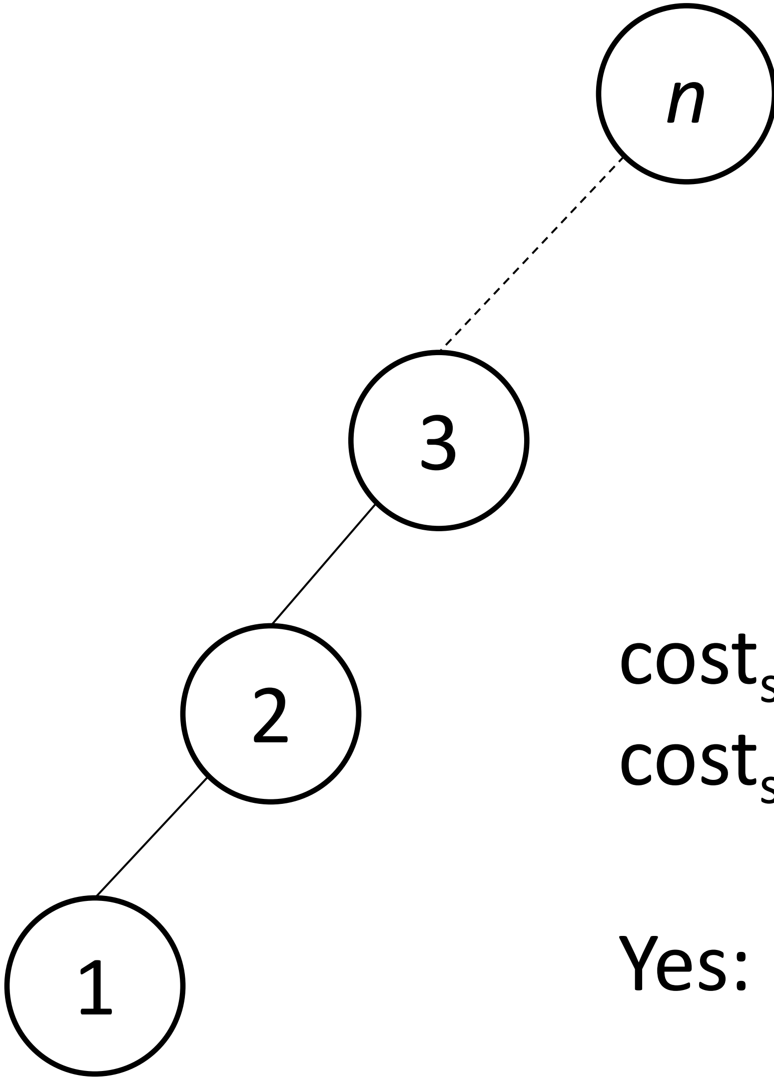
Theorem: $S(E) = \mathbf{T}(Q_1, Q_1') \oplus \ldots \oplus \mathbf{T}(Q_m, Q_m')$ is a simulation embedding for Splay

Proof: Splaying $S(E)$ substitutes in each of the transition trees Q_i at cost at most $80|Q_i|$

- Total cost $80\,(|Q_1| + \ldots + |Q_m|) \leq 80 \cdot \mathrm{OPT}(X, T)$
- X is a subsequence of $S(E)$ since x_i must be the last item in each $\mathbf{T}(Q'_i, Q_i)$

Corollary: Monotonicity → Optimality!

Aside: "Approximate" Monotone?



$X = (3, 1, 2)$
$Y = (1, 2)$

$\text{cost}_{\text{splay}}(X, T) = n + O(1)$
$\text{cost}_{\text{splay}}(Y, T) = 3n/2 + O(1)$

Yes: $a \geq 3/2$

Related Work

- Harmon 06 constructs a simulation embedding for another candidate-optimal algorithm, Greedy, in the geometric view

- Russo 15 analyzes a simulation embedding for Splay via potential functions

1B) OPTIMALITY → MONOTONICITY

OPT is Monotone

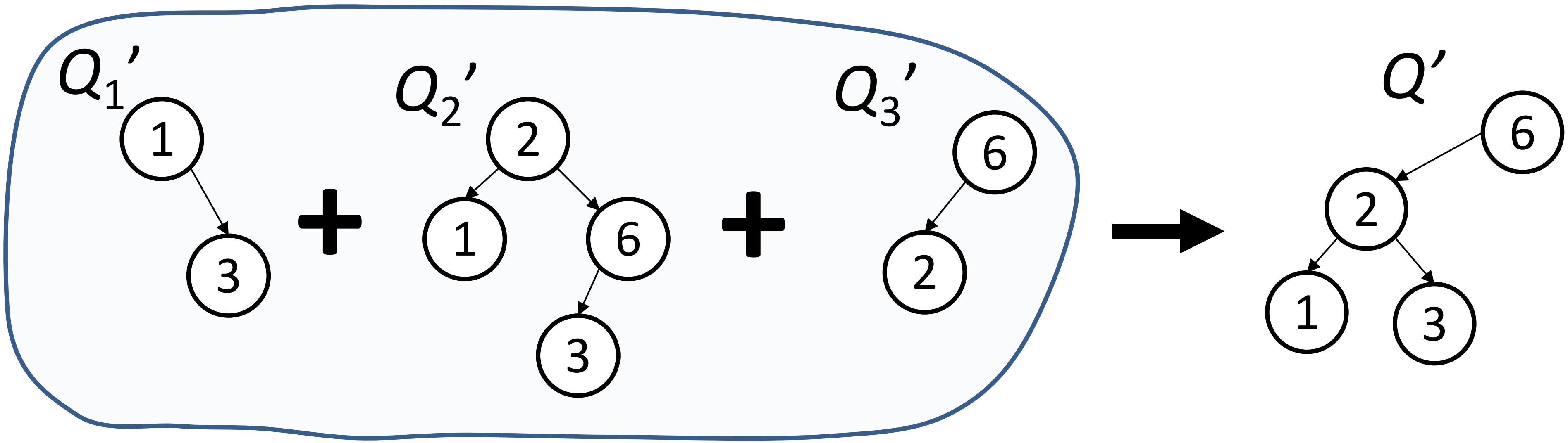
Theorem: If $Y \subseteq X$ then $\mathrm{OPT}(Y, T) \leq \mathrm{OPT}(X, T)$

Proof: By executing Y, T at cost below $\mathrm{OPT}(X, T)$

- Let $Q_1', \ldots, Q_m'$ optimally execute X, T
- Y is formed from $X = (x_1, \ldots, x_m)$ by removing requests at some subset of indices $\{1, \ldots, m\}$
- *Elide* contiguous blocks of removed transition trees $Q_i', \ldots, Q_j'$ into $|Q'|$ to execute Y, T

$|Q'| = |Q_i \cup \ldots \cup Q_j| \leq |Q_i| + \ldots + |Q_j|$

Transition Tree Elision



Necessity of Monotonicity

Theorem: If Splay is dynamically optimal then it is approximately monotone

Proof: For all subsequences Y of X

$$\begin{aligned} \mathrm{cost}_{\mathrm{splay}}(Y, T) &\le \mathrm{OPT}(Y, T) \\ &\le \mathrm{OPT}(X, T) \\ &\le c \cdot \mathrm{cost}_{\mathrm{splay}}(X, T) \end{aligned}$$

OPT is Eliminated!

Splay is dynamically optimal *if and only if* it is approximately monotone

We don't need to understand how OPT behaves to prove dynamic optimality!

2) REMOVING ADDITIVE OVERHEAD

Does Splay Have Overhead?

Perhaps splay has a “startup overhead?”

E.g. *could* $\text{cost}_{\text{splay}}(X, T) \le b \cdot \text{OPT}(X, T)$ $+\, n \cdot \log n$?

$+\, n \cdot \log \log n$?

Splay Has No Overhead!

Perhaps splay has a "startup overhead?"

E.g. *could* $\text{cost}_{\text{splay}}(X, T) \leq b \cdot \text{OPT}(X, T) + n \cdot \log n$?

$+ n \cdot \log \log n$?

NO! Splay has no "intrinsic" additive overhead.

Additive Overhead

We say that A is optimal with additive overhead $g: \mathbb{T} \to \mathbb{N}$ if for some $b > 0$ and all X, T:

$$\text{cost}(X, T) \leq b \cdot \text{OPT}(X, T) + g(T)$$

The overhead is intrinsic if there is a sequence $\{(X_1, T_1), (X_2, T_2), \ldots\}$ of instances such that:

$$\lim_{n\to\infty} \text{cost}_A(X_n, T_n) \,/\, \text{OPT}(X_n, T_n) = \infty$$

Eliminating Overheads (Intuition)

- Start with X, T with "high" overhead $g(T)$
- Augmenting X into new request sequence Z in a way that ensures both the optimal and Splay costs scale *linearly* with the number of repetitions of Z
- With sufficient number of repetitions we can "absorb" $g(T)$

“Amplifying” Splay by Repetition

Lemma 1: If T' is the tree after splaying (X, T) and $Z = X \bigoplus \mathbf{T}(T', T)$ then $k \cdot \mathrm{cost}(X, T) \leq \mathrm{cost}(Z^k, T)$

Proof:

- $\mathrm{cost}(X, T) \leq \mathrm{cost}(Z, T)$ [X is prefix to Z]
- $\mathrm{cost}(Z^k, T) = k \cdot \mathrm{cost}(Z, T)$ [reset]

Upper Bounding OPT

Lemma 2: $\mathrm{OPT}(Z^k, T) \leq 83k \cdot \mathrm{OPT}(X, T)$

Proof: Any execution of Z^k, T provides an upper bound $\mathrm{OPT}(Z^k, T)$

- Concatenate an optimal execution for X, T with Splay's execution of $\mathbf{T}(T', T)$
- Total cost: $k \cdot (\mathrm{OPT}(X, T) + 80|T| + 2|T|)$
- $\mathrm{OPT}(X, T) \geq |T|$

Eliminating Splay's Overhead

Theorem: If $\text{cost}_{\text{splay}}(X, T) \leq b \cdot \text{OPT}(X, T) + g(T)$ then g is *not intrinsic*

Eliminating Splay's Overhead

Theorem: If $\text{cost}_{\text{splay}}(X, T) \leq b \cdot \text{OPT}(X, T) + g(T)$ then g is *not intrinsic*

Proof:

Eliminating Splay's Overhead

Theorem: If $\text{cost}_{\text{splay}}(X, T) \leq b \cdot \text{OPT}(X, T) + g(T)$ then g is *not intrinsic*

Proof:

$k \cdot \text{cost}(X, T) \leq \text{cost}(Z^k, T)$ [Lemma 1]

Eliminating Splay's Overhead

Theorem: If $\text{cost}_{\text{splay}}(X, T) \leq b \cdot \text{OPT}(X, T) + g(T)$ then g is *not intrinsic*

Proof:

$$k \cdot \text{cost}(X, T) \leq \text{cost}(Z^k, T) \qquad \text{[Lemma 1]}$$

$$\leq b \cdot \text{OPT}(Z^k, T) + g(T) \qquad \text{[Overhead]}$$

Eliminating Splay's Overhead

Theorem: If $\text{cost}_{\text{splay}}(X, T) \leq b \cdot \text{OPT}(X, T) + g(T)$ then g is *not intrinsic*

Proof:

$$
\begin{aligned}
k \cdot \text{cost}(X, T) &\leq \text{cost}(Z^k, T) && \text{[Lemma 1]} \\
&\leq b \cdot \text{OPT}(Z^k, T) + g(T) && \text{[Overhead]} \\
&\leq 83b \cdot k \cdot \text{OPT}(X, T) + g(T) && \text{[Lemma 2]}
\end{aligned}
$$

Eliminating Splay's Overhead

Theorem: If $\text{cost}_{\text{splay}}(X, T) \le b\cdot\text{OPT}(X, T) + g(T)$ then g is *not intrinsic*

Proof:

$$
\begin{aligned}
k\cdot\text{cost}(X, T) &\le \text{cost}(Z^k, T) && \text{[Lemma 1]}\\
&\le b\cdot\text{OPT}(Z^k, T) + g(T) && \text{[Overhead]}\\
&\le 83b\cdot k\cdot\text{OPT}(X, T) + g(T) && \text{[Lemma 2]}\\
&\le 84b\cdot k\cdot\text{OPT}(X, T) && [k \ge g(T)/\text{OPT}(X,T)]
\end{aligned}
$$

Eliminating Splay's Overhead

Theorem: If $\mathrm{cost}_{\mathrm{splay}}(X, T) \le b \cdot \mathrm{OPT}(X, T) + g(T)$ then g is *not intrinsic*

Proof:

$$\cancel{k} \cdot \mathrm{cost}(X, T) \le \mathrm{cost}(Z^k, T) \qquad \text{[Lemma 1]}$$

$$\le b \cdot \mathrm{OPT}(Z^k, T) + g(T) \qquad \text{[Overhead]}$$

$$\le 83b \cdot k \cdot \mathrm{OPT}(X, T) + g(T) \qquad \text{[Lemma 2]}$$

$$\le 84b \cdot \cancel{k} \cdot \mathrm{OPT}(X, T) \qquad [k \ge g(T)/\mathrm{OPT}(X,T)]$$

Eliminating Splay's Overhead

Theorem: If $\text{cost}_{\text{splay}}(X, T) \le b \cdot \text{OPT}(X, T) + g(T)$ then g is *not intrinsic*

Proof:

$$\cancel{k} \cdot \text{cost}(X, T) \le \text{cost}(Z^k, T) \qquad \text{[Lemma 1]}$$

$$\le b \cdot \text{OPT}(Z^k, T) + g(T) \qquad \text{[Overhead]}$$

$$\le 83b \cdot k \cdot \text{OPT}(X, T) + g(T) \qquad \text{[Lemma 2]}$$

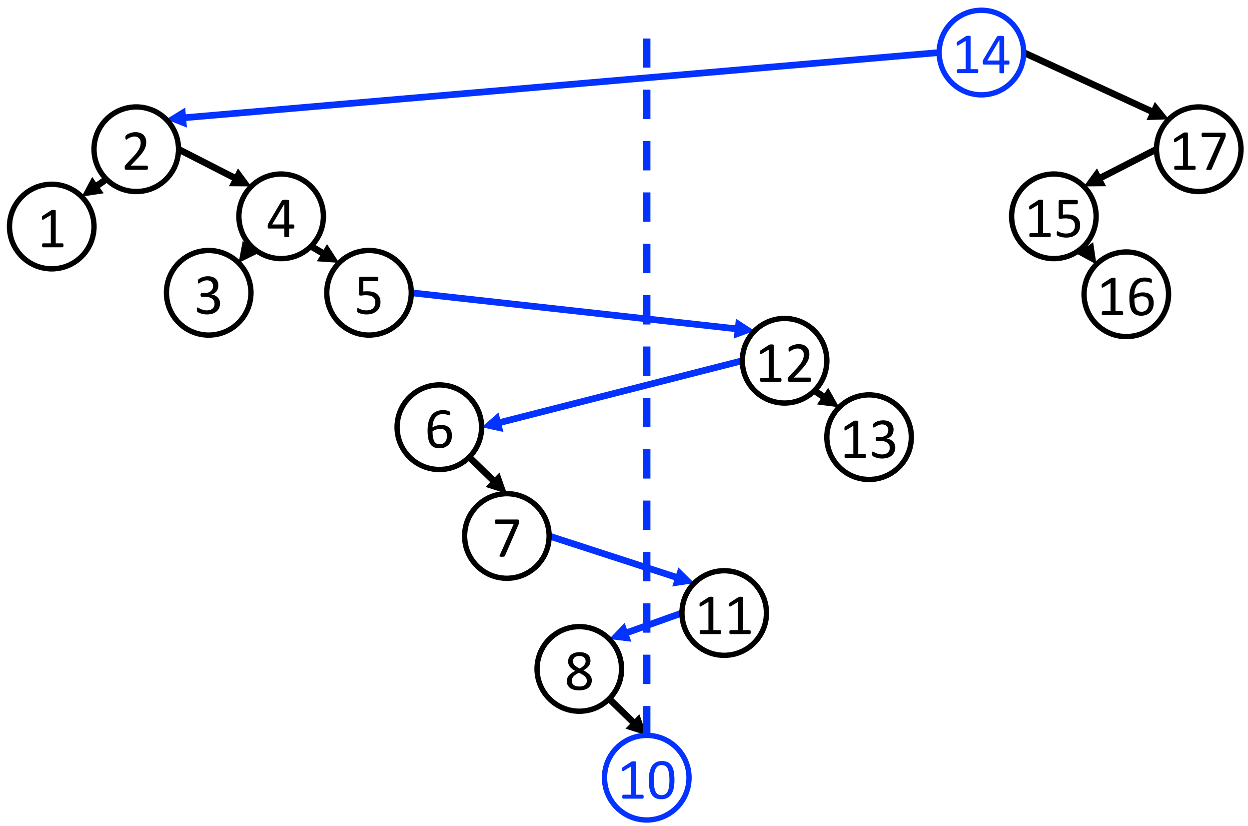
$$\le 84b \cdot \cancel{k} \cdot \text{OPT}(X, T) \qquad [k \ge g(T)/\text{OPT}(X,T)]$$

$\Rightarrow$ h not intrinsic since $\text{cost}(X, T)/\text{OPT}(X, T) \le 84b$

3) THE CROSSING BOUND

Crossings

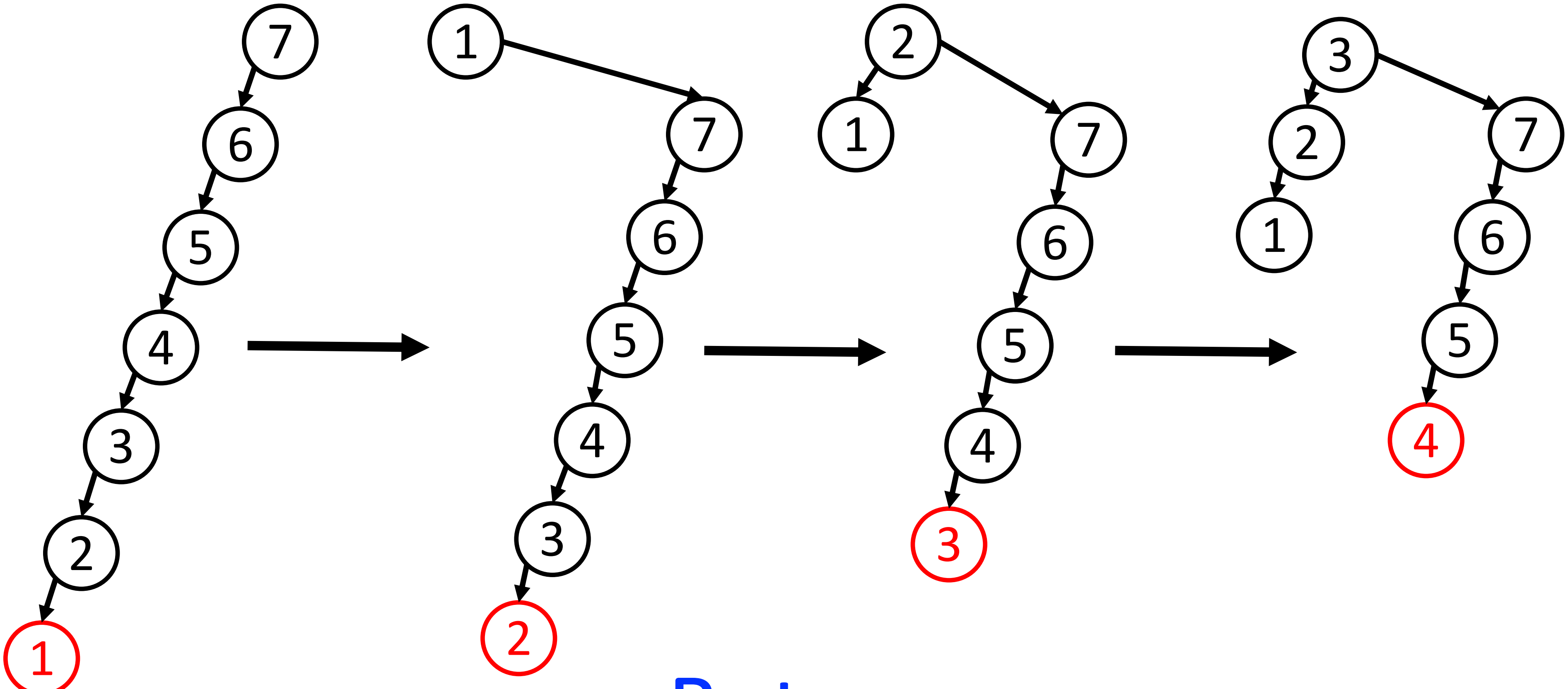
The crossing depth of *x* is number of path endpoints plus number of crossing edges



Move-to-Root

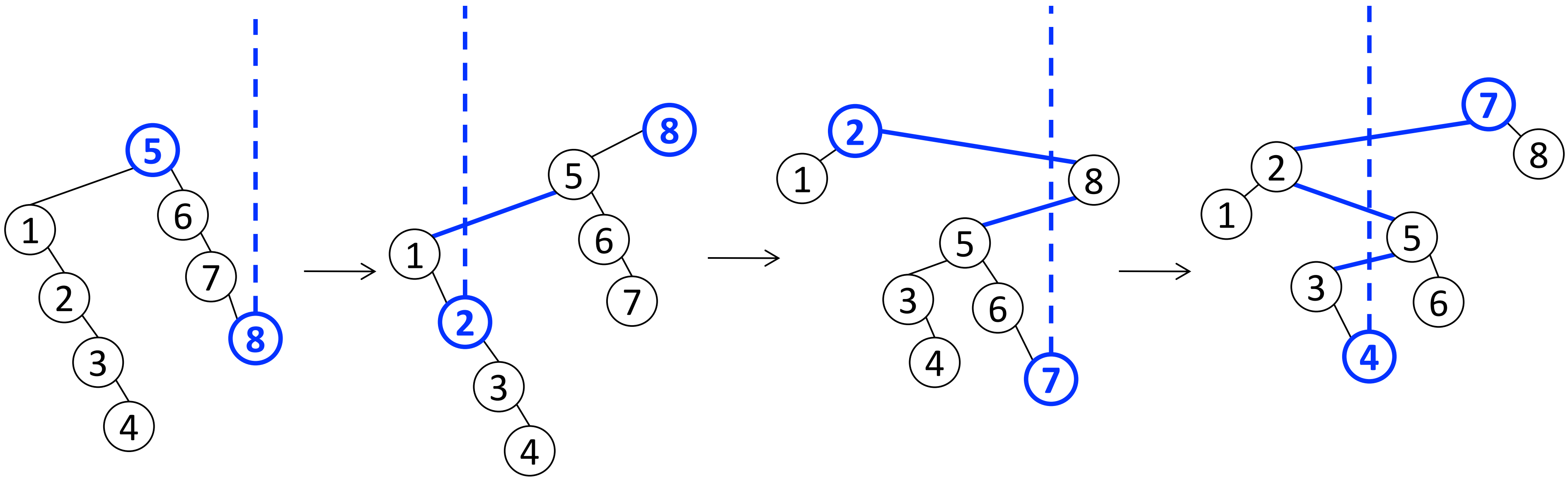
- Simply rotate each requested node to the root
- Maintains a max-heap order with respect to the keys’ most recent access time
- A progenitor for Splay
- Not optimal!

Move-to-Root is *not* Optimal



But...

Wilber's Crossing Lower Bound



$\Lambda(X, T)$ counts total of the crossing depths of Move-to-Root's access paths. Here, $\Lambda = 6$

Theorem [Wilber 89]: $\Lambda(X, T) = O\big(\mathrm{OPT}(X, T)\big)$

Approximate Monotonicity

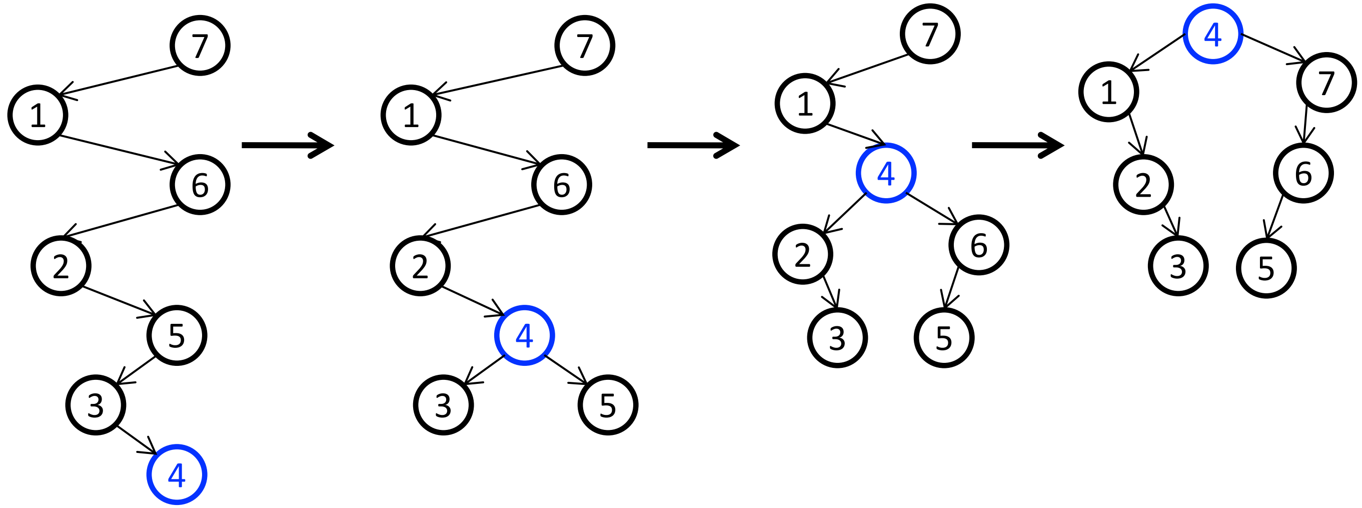
Theorem: $\Lambda(Y, T) \leq 4\Lambda(X, T)$ for all subsequences *Y* of *X*

Proof: By case-analysis of how removing one request affects Move-to-Root's execution of an instance (very complicated)

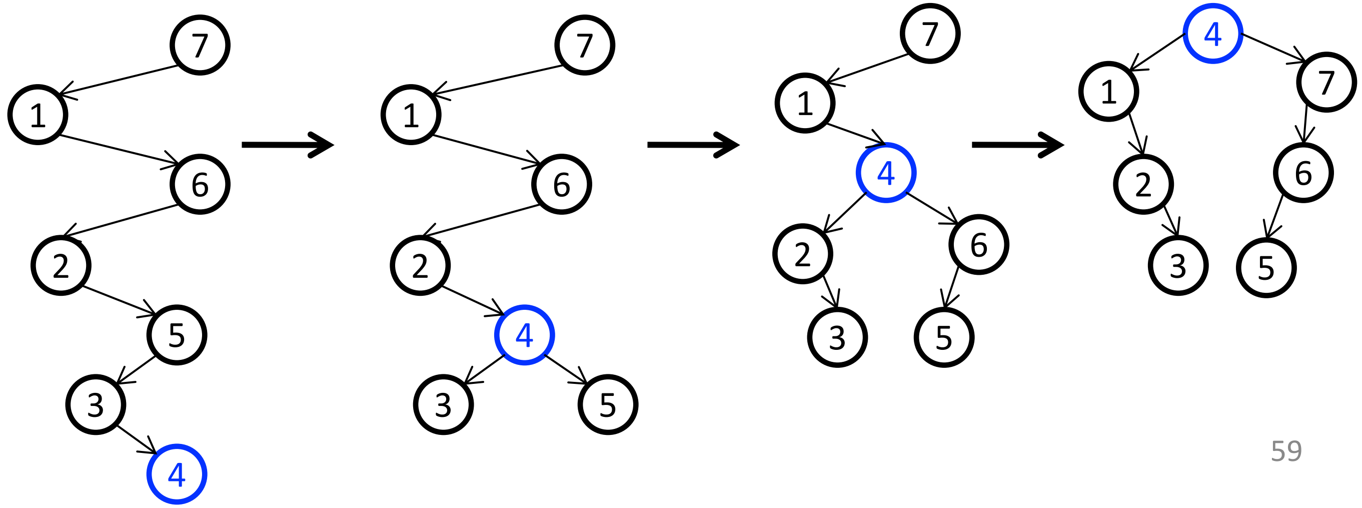
4) THE PATH FORWARD

Move-to-Root vs. Splay: Zig-*Zags*

Move to Root

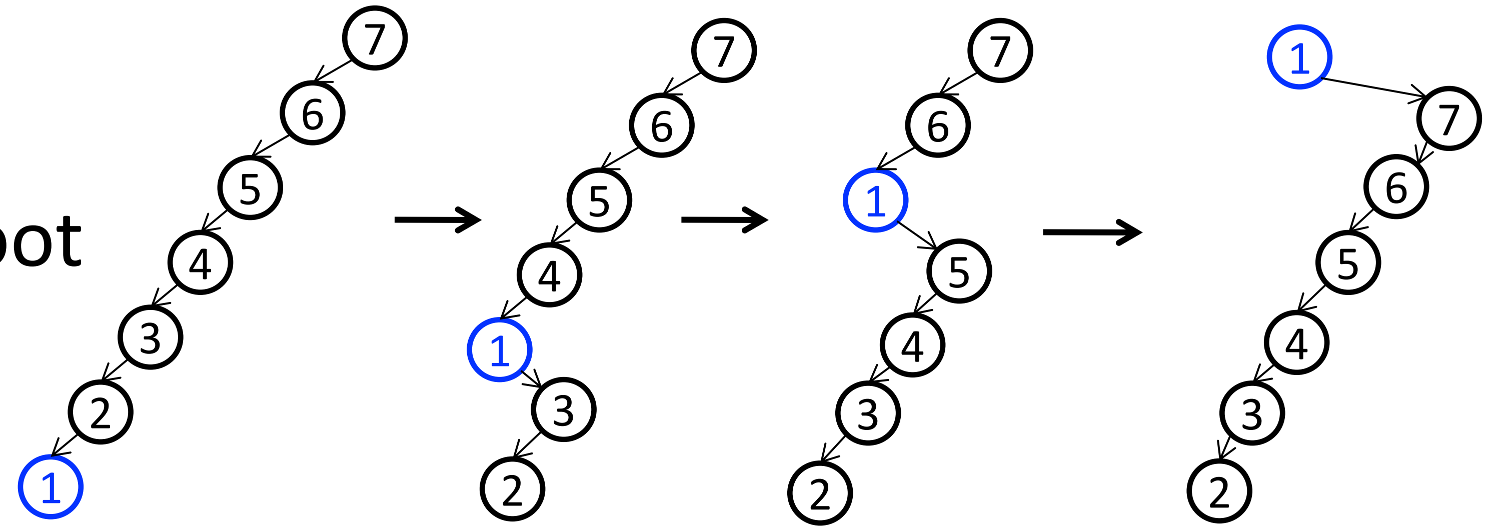


Splay

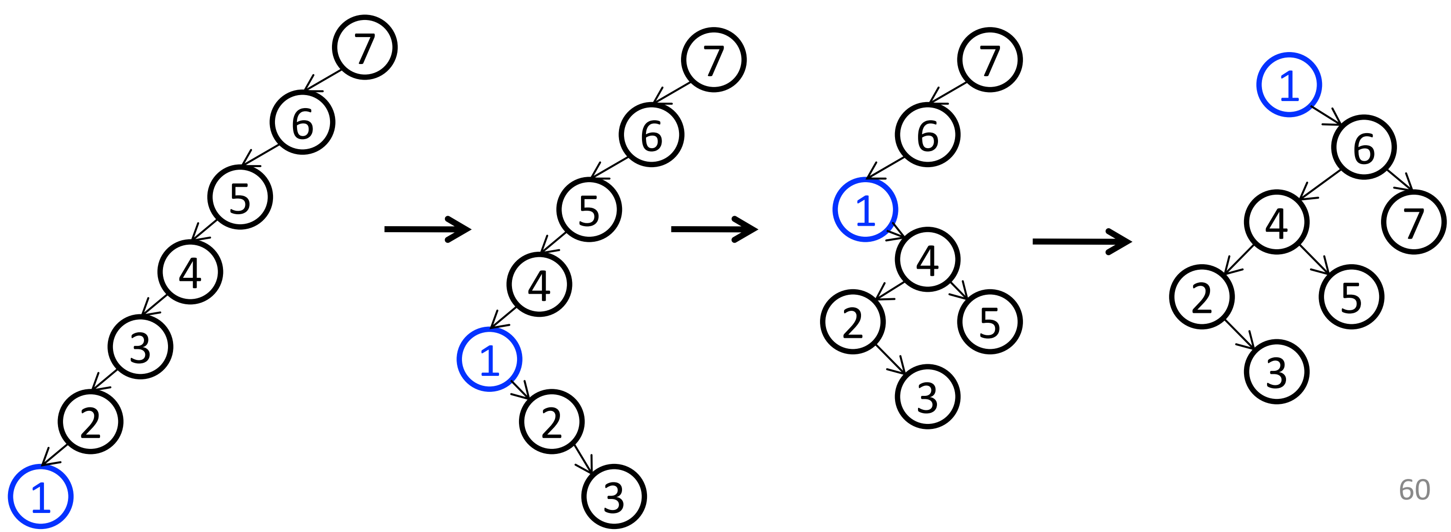


Move-to-Root vs. Splay: Zig-*Zigs*

Move to Root



Splay



Global View of Splay

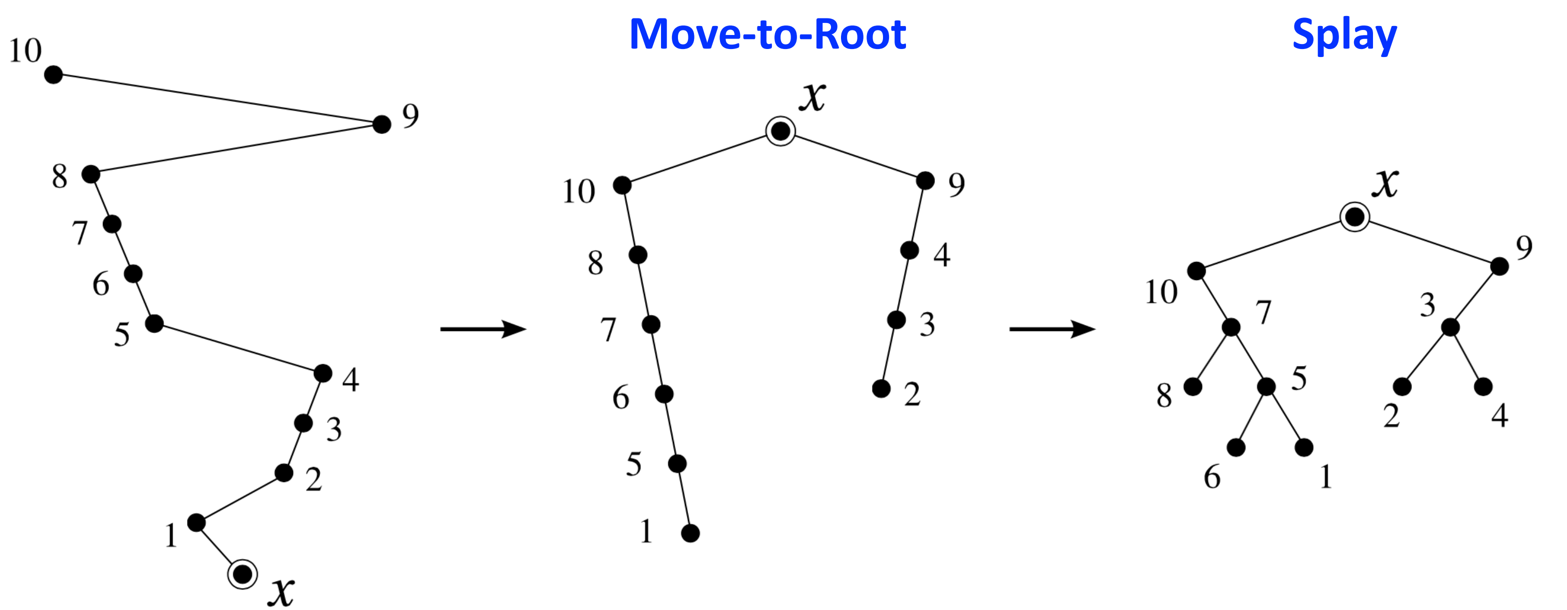


Figure taken from “Binary Search Trees, Rectangles and Patterns,” Lazlo Kozma, 2016.

Observations

- Each *zig-zig* can introduce a violation of max-heap order with respect to most-recent access time in the side arms of the after-tree

- The crossing depth of any key in splay(T, x) and move-to-root(T, x) differs by at most 2

This looks ripe for analysis via *potential functions*…

PROVING DYNAMIC OPTIMALITY

Splay’s Crossing Cost

Conjecture 1: Splay’s crossing cost is approximately monotone

Possible Proof: Add a potential for tracking total number heap-order violations in the Splayed tree to the proof that Wilber’s bound is approximately monotone

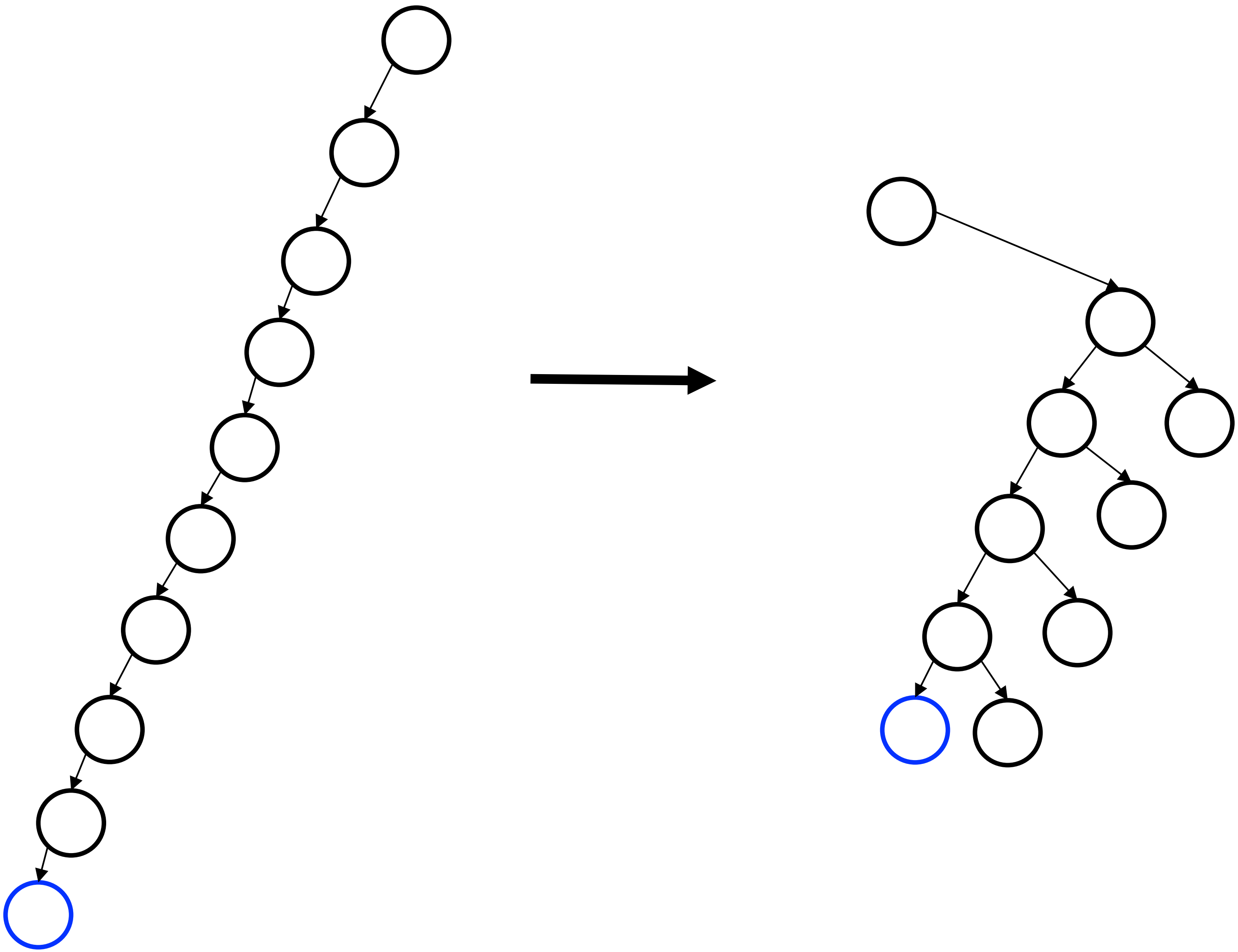
Splay’s Bookkeeping Cost

Conjecture 2: The number of non-crossing nodes on Splay’s access paths is bounded by a fixed multiple of Splay’s crossing cost

Possible Proof: Use Splay’s “depth-halving” properties and the fact that Splay turns bookkeeping nodes into crossings

Need a potential for tracking number of zig-zigs in the tree

Splay's "Depth-Halving" Effect



How to Prove Optimality

- If Conjectures 1 and 2 are true then Splay is dynamically optimal
- Both conjectures are borne out by numerical experiments
- The proof that Λ is approximately monotone provides machinery that we can build on

One More Thing…

A potential function's range:

- Gives the algorithm's "additive overhead"
- Depends on the function's design

We already know Splay's additive overhead!

Thus, we need not explore certain kinds of potential

EPILOGUE

Consequences

Monotonicity also implies:

- Splay performs *m* deque operations starting from *T* with cost $O(m + |T|)$
- Splaying *T* by the preorder of *T'* costs $O(|T|)$

Results generalize to "path-based" algorithms

Additional Results

- “Universal” subtree transformation sequences that are independent of the tree containing them
- Simulation Embedding for Top-Down Splay
- Insertion-splaying postorders takes linear time; Splaying preorders and postorders of weight-balanced search trees takes linear time

Open Questions

1. Does our proposal work?

2. Extend these methods to Greedy, and related algorithms?

3. Further applications of simulation embeddings, approximate monotonicity?

THANKS!