Self-Adjusting Binary Search Trees: Recent Results

Talks based on papers in WADS 2015, ESA 2015, FOCS 2015.

Parinya Chalermsook
Mayank Goswami
László Kozma¹

Kurt Mehlhorn
Thatchaphol Saranurak²

¹Department of Computer Science, Saarland University
²KTH Royal Institute of Technology, Stockholm.
Self-Adjusting Binary Search Trees

Example: Search sequence 1, 2, 3, 4, ..., n can be served in linear time $O(n)$.

- with knowing access sequence: offline optimum (OPT).
- without knowing it: self-adjusting BST, online algorithm.
  - either with arbitrary initial tree or with chosen initial tree.
Self-Adjusting Binary Search Trees

Example: Search sequence $1, 2, 3, 4, \ldots, n$ can be served in linear time $O(n)$.

- with knowing access sequence: offline optimum (OPT).
- without knowing it: self-adjusting BST, online algorithm.
  - either with arbitrary initial tree or with chosen initial tree.
After an access, replace the search path by an arbitrary tree (the after-tree) on the same set of nodes rooted at the accessed element.

- Reattach the dangling subtrees (uniquely defined).
- Cost = length of search path.

Question: Which re-arrangements lead to an efficient online algorithm?
The Self-Adjusting BST-Model

- After an access, replace the search path by an arbitrary tree (the after-tree) on the same set of nodes rooted at the accessed element.
- Reattach the dangling subtrees (uniquely defined).
- \textbf{Cost} = length of search path.

Question: Which re-arrangements lead to an efficient online algorithm?
The Self-Adjusting BST-Model

- After an access, replace the search path by an arbitrary tree (the after-tree) on the same set of nodes rooted at the accessed element.
- Reattach the dangling subtrees (uniquely defined).
- Cost = length of search path.

Question: Which re-arrangements lead to an efficient online algorithm?
Splay Trees (Sleator and Tarjan, 1983)

After each access: Re-arrange search path bottom-up, until accessed element is root.

![Diagram of splay trees](image)
Splay Trees (Sleator and Tarjan, 1983)

III. 6.1.2. Splay-Bäume 249

Abb. 95. 4 Schüttle-Operationen.

[K. Mehlhorn: Data structures and algorithms (German ed. 1986)]
The Dynamic Optimality Conjecture (Sleator/Tarjan ’85)

**Splay trees are** $O(1)$-**competitive**, i.e., for every access sequence $X$, the cost of serving $X$ by splay trees is at most a constant factor larger than serving $X$ optimally.

A path towards proving or disproving the conjecture:

- Understand better which variants of splay trees might also work.
- Show special cases of the dynamic optimality conjecture.
- Exhibit additional easy sequences, i.e., sequences which OPT serves in time $o(n \log n)$. 

Splay trees have many nice properties, e.g.,

- Logarithmic access cost
- Working set property
- Sequential access

---

Static optimality
Static finger property
Dynamic finger property

We give sufficient (and necessary) conditions for logarithmic access cost, static optimality, working set and static finger property.

Previous work: Sleator, Tarjan, Subramanian, Georgakopoulos, McClurkin prove logarithmic access cost, static optimality, working set and static finger property for splay-trees and variants thereof.

All these results are corollaries of our main theorem (in the ESA paper). Also prove new results about depth-halving.
Main Result

Characteristic quantities of the search path and the after-tree.

- length of the search path: $|P| = 12$
- number of side changes: $z = 4$
- number of leaves: $\ell = 5$
- max left-depth of left subtree (max right-depth of right subtree): $d = 3$

Theorem: If accessed element goes to root, $d = O(1)$, and $\ell = \Omega(|P| - z)$, then the BST has logarithmic access, is statically optimal, and has the working set and static finger properties.

We also have a partial converse (more later).
Main Result

Characteristic quantities of the search path and the after-tree.

- length of the search path: $|P| = 12$
- number of side changes: $z = 4$
- number of leaves: $\ell = 5$
- max left-depth of left subtree (max right-depth of right subtree): $d = 3$

Theorem: If accessed element goes to root, $d = O(1)$, and $\ell = \Omega(|P| - z)$, then the BST has logarithmic access, is statically optimal, and has the working set and static finger properties.

We also have a partial converse (more later).
Main Result

Characteristic quantities of the search path and the after-tree.

- length of the search path: $|P| = 12$
- number of side changes: $z = 4$
- number of leaves: $\ell = 5$
- max left-depth of left subtree (max right-depth of right subtree): $d = 3$

Theorem: If accessed element goes to root, $d = O(1)$, and $\ell = \Omega(|P| - z)$, then the BST has logarithmic access, is statically optimal, and has the working set and static finger properties.

We also have a partial converse (more later).
Main Result

Characteristic quantities of the search path and the after-tree.

- length of the search path: $|P| = 12$
- number of side changes: $z = 4$
- number of leaves: $\ell = 5$
- max left-depth of left subtree (max right-depth of right subtree): $d = 3$

Theorem: If accessed element goes to root, $d = O(1)$, and $\ell = \Omega(|P| - z)$, then the BST has logarithmic access, is statically optimal, and has the working set and static finger properties.

We also have a partial converse (more later).
Main Result

Characteristic quantities of the search path and the after-tree.

- length of the search path: $|P| = 12$
- number of side changes: $z = 4$
- number of leaves: $\ell = 5$
- max left-depth of left subtree (max right-depth of right subtree): $d = 3$

Theorem: If accessed element goes to root, $d = O(1)$, and $\ell = \Omega(|P| - z)$, then the BST has logarithmic access, is statically optimal, and has the working set and static finger properties.

We also have a partial converse (more later).
Main Result

Characteristic quantities of the search path and the after-tree.

- length of the search path: $|P| = 12$
- number of side changes: $z = 4$
- number of leaves: $\ell = 5$
- max left-depth of left subtree (max right-depth of right subtree): $d = 3$

Theorem: If accessed element goes to root, $d = O(1)$, and $\ell = \Omega(|P| - z)$, then the BST has logarithmic access, is statically optimal, and has the working set and static finger properties.

We also have a partial converse (more later).
Main Result

Characteristic quantities of the search path and the after-tree.

- length of the search path: $|P| = 12$
- number of side changes: $z = 4$
- number of leaves: $\ell = 5$
- max left-depth of left subtree (max right-depth of right subtree): $d = 3$

Theorem: If accessed element goes to root, $d = O(1)$, and $\ell = \Omega(|P| - z)$, then the BST has logarithmic access, is statically optimal, and has the working set and static finger properties.

We also have a partial converse (more later).
- Split the search path at $s$ and swap adjacent odd-even pairs.
- **This is a global view on splay trees.**
- accessed element becomes root
- max right-(left) depth is \( d = 2 \)
- \( z + \ell \geq |P|/2 - 1 \)

Proof: There are \( |P|/2 - 1 \) odd-even pairs. Each side change can move the elements of one pair to different sides. Thus

\[
\text{# of leaves} \geq |P|/2 - 1 - \text{# of side changes}
\]
Application I: Splay Trees

- accessed element becomes root
- max right-(left) depth is $d = 2$
- $z + \ell \geq |P|/2 - 1$

Proof: There are $|P|/2 - 1$ odd-even pairs. Each side change can move the elements of one pair to different sides. Thus

$$\text{# of leaves} \geq |P|/2 - 1 - \text{# of side changes}$$
- accessed element becomes root
- max right-(left) depth is $d = 2$
- $z + \ell \geq |P|/2 - 1$

Proof: There are $|P|/2 - 1$ odd-even pairs. Each side change can move the elements of one pair to different sides. Thus

\[
\text{# of leaves} \geq |P|/2 - 1 - \text{# of side changes}
\]
accessed element becomes root
max right-(left) depth is \( d = 2 \)
\( z + \ell \geq |P|/2 - 1 \)

Proof: There are \(|P|/2 - 1\) odd-even pairs. Each side change can move the elements of one pair to different sides. Thus

\[
\text{# of leaves} \geq |P|/2 - 1 - \text{# of side changes}
\]
Depth Halving

In splay every node on the search path roughly halves its depth.

Sleator: is this property sufficient?

We don’t know, but strict depth-halving is sufficient: the accessed element becomes the root and every node $x$ on the search path loses at least $(1/2 + \epsilon)d(x) - O(1)$ ancestors and gains at most $O(1)$ new descendants.
In splay every node on the search path roughly halves its depth.

Sleator: is this property sufficient?

We don’t know, but strict depth-halving is sufficient: the accessed element becomes the root and every node \( x \) on the search path loses at least \((1/2 + \epsilon)d(x) - O(1)\) ancestors and gains at most \(O(1)\) new descendants.
Partial Convereses

If the after-tree may have non-constant left-depth or right-depth, then the good properties (logarithmic access, static optimality, \ldots) cannot be shown with the sum-of-logs potential function.

If the number of leaves of the after-tree is allowed to be $o(|P|)$ – number of side changes), then the traversal conjecture does not hold.
Partial Convereses

If the after-tree may have non-constant left-depth or right-depth, then the good properties (logarithmic access, static optimality, . . . ) cannot be shown with the sum-of-logs potential function.

If the number of leaves of the after-tree is allowed to be $o(|P| - \text{number of side changes})$, then the traversal conjecture does not hold.
The Dynamic Optimality Conjecture (Sleator/Tarjan ’85)

Splay trees are $O(1)$-competitive, i.e., for every access sequence $X$, the cost of serving $X$ by splay trees is at most a constant factor larger than serving $X$ optimally.

A path towards proving or disproving the conjecture:

- Show special cases of the dynamic optimality conjecture.
- Exhibit additional easy sequences, i.e., sequences which OPT serves in time $o(n \log n)$.

Traversal conjecture: Let $T$ be a tree on $[n]$ and let $X$ be the preorder traversal of $T$. Process $X$ starting with a tree $T’$. OPT has cost $O(n)$.

Only shown for $T’ = T$ and for $X = 1, 2, \ldots, n$. 
Pattern Avoiding Accesses (FOCS 2015)

An access sequence $X$ avoids a pattern $P$ if there is no subsequence of $X$ that is order-isomorphic to $P$.

- $X = 1, 2, \ldots, n$ avoids 2, 1.
- Preorder traversal of a tree avoids 2, 3, 1.
- Upper bounds for GREEDY (Def. on next slides)
  - GREEDY serves any sequence that avoids a permutation pattern of size $k$ with cost $O(n^{2^\alpha(n)}O(k^2))$.
  - GREEDY with chosen initial tree serves any such sequence with cost $O(n^{2O(k^2)})$.
  - Traversal conjecture: $k = 3$.
- New easy sequences.
  - OPT serves any sequence that avoids all permutations patters of size $k + 1$ with cost $O(n \log k)$. 
An access sequence $X$ avoids a pattern $P$ if there is no subsequence of $X$ that is order-isomorphic to $P$.

- $X = 1, 2, \ldots, n$ avoids 2, 1.
- Preorder traversal of a tree avoids 2, 3, 1.
- Upper bounds for GREEDY (Def. on next slides)
  - GREEDY serves any sequence that avoids a permutation pattern of size $k$ with cost $O(n^{2^\alpha(n)}O(k^2))$.
  - GREEDY with chosen initial tree serves any such sequence with cost $O(n^{2^{O(k)}})$.
  - Traversal conjecture: $k = 3$.
- New easy sequences.
  - OPT serves any sequence that avoids all permutations patterns of size $k + 1$ with cost $O(n \log k)$. 
- $M$ is a \{0, 1\}-matrix (a point set).
- $\square_{pq} =$ closed rectangle with corners $p$ and $q$.
- $M$ is **satisfied** if for any two points $p, q \in M$ with distinct $x$ and $y$ coordinates there is another point $r \in M \cap \square_{pq} \setminus \{p, q\}$.
- An access sequence $X \in [n]^m$ gives rise to a matrix $X$, where there is a point $(x, t) \in X$ iff the element $x$ is accessed at time $t$.
- A tree $T$ gives rise to a matrix $T$ where in each column $x$, there is a stack of points $(x, 1), \ldots, (x, d - d(x) + 1) \in T$. 
Geometric BSTs (Demaine, Harmon, Iacono, Kane, Patrascu (SODA '09))

Cost = 8

**Geometric BST** \( \mathcal{A} \) on input \( [X]_T \) outputs a satisfied matrix \( [\mathcal{A}_T(X)]_T \), where \( \mathcal{A}_T(X) \supseteq X \).

Cost = number of points (ones) in \( \mathcal{A}_T(X) \).

Chosen initial tree: On input \( X \) outputs \( \mathcal{A}(X) \).

Offline versus online.

Theorem (DHIKP): (online) geometric BSTs are essentially equivalent to (online) BSTs.
Geometric BSTs (Demaine, Harmon, Iacono, Kane, Patrascu (SODA ’09))

Geometric BST $\mathcal{A}$ on input $[X^T]$ outputs a satisfied matrix $[\mathcal{A}_T(X)]$, where $\mathcal{A}_T(X) \supseteq X$.

Cost = number of points (ones) in $\mathcal{A}_T(X)$.

Chosen initial tree: On input $X$ outputs $\mathcal{A}(X)$.

Offline versus online.

Theorem(DHIKP): (online) geometric BSTs are essentially equivalent to (online) BSTs.
Geometric BSTs (Demaine, Harmon, Iacono, Kane, Patrascu (SODA ’09))

Geometric BST $\mathcal{A}$ on input $\begin{bmatrix} X_T \end{bmatrix}$ outputs a satisfied matrix $\begin{bmatrix} \mathcal{A}_T(X) \end{bmatrix}$, where $\mathcal{A}_T(X) \supseteq X$.

Cost = number of points (ones) in $\mathcal{A}_T(X)$.

Chosen initial tree: On input $X$ outputs $\mathcal{A}(X)$.

Offline versus online.

Theorem (DHIKP): (online) geometric BSTs are essentially equivalent to (online) BSTs.
**Geometric BSTs (Demaine, Harmon, Iacono, Kane, Patrascu (SODA ’09))**

Geometric BST $\mathcal{A}$ on input $[X_T]$ outputs a satisfied matrix $[\mathcal{A}_T(X)]$, where $\mathcal{A}_T(X) \supseteq X$.

**Cost** = number of points (ones) in $\mathcal{A}_T(X)$.

Chosen initial tree: On input $X$ outputs $\mathcal{A}(X)$.

Offline versus online.

**Theorem (DHIKP):** (online) geometric BSTs are essentially equivalent to (online) BSTs.
Geometric BSTs (Demaine, Harmon, Iacono, Kane, Patrascu (SODA ’09))

Geometric BST $\mathcal{A}$ on input $[X]$ outputs a satisfied matrix $[\mathcal{A}_T(X)]$, where $\mathcal{A}_T(X) \supseteq X$.

Cost = number of points (ones) in $\mathcal{A}_T(X)$.

Chosen initial tree: On input $X$ outputs $\mathcal{A}(X)$.

Offline versus online.

Theorem(DHIKP): (online) geometric BSTs are essentially equivalent to (online) BSTs.
Greedy (Lucas, Munro, DHIKP)

- An access to $x$ at time $t$ touches point $(y, t)$ iff $\square(x, t), (y, t')$ is empty (not satisfied), where $(y, t')$ is the highest point in column $y$.

- Notice that these points have to be touched. However, Greedy is not optimal.

- Conjecture (Lucas, Munro, DHIKP): Greedy is $O(1)$-competitive.

- Theorem: Greedy almost satisfies traversal conjecture (cost $n \cdot 2^{\alpha(n)^{O(1)}}$).

Greedy with chosen initial tree satisfies traversal conjecture.
An access to $x$ at time $t$ touches point $(y, t)$
 iff $\square(x, t), (y, t')$ is empty (not satisfied),
 where $(y, t')$ is the highest point in column $y$.

Notice that these points have to be touched. However, Greedy is not optimal.

Conjecture (Lucas, Munro, DHIKP):
Greedy is $O(1)$-competitive.

Theorem: Greedy almost satisfies traversal conjecture (cost $n \cdot 2^{\alpha(n)O(1)}$).

Greedy with chosen initial tree satisfies traversal conjecture.
Greedy (Lucas, Munro, DHIKP)

- An access to $x$ at time $t$ touches point $(y, t)$ iff $\square(x, t), (y, t')$ is empty (not satisfied), where $(y, t')$ is the highest point in column $y$.

- Notice that these points have to be touched. However, Greedy is not optimal.

- Conjecture (Lucas, Munro, DHIKP): Greedy is $O(1)$-competitive.

- Theorem: Greedy almost satisfies traversal conjecture (cost $n \cdot 2^{\alpha(n)^{O(1)}}$).

Greedy with chosen initial tree satisfies traversal conjecture.
Let $M$ and $P$ be $n \times n$ and $k \times k$ matrices such that $M$ avoids $P$.

- (Marcus, Tardos, Fox): If $P$ is a permutation matrix, the number of ones in $M$ is at most $n2^{O(k)}$.
- (Klasar, Keszegh): If $P$ is light (only one 1 per column), the number of ones in $M$ is at most $n2^{\alpha(n)O(k^2)}$.

S. Pettie pioneered the use of forbidden matrix theory for the study of data structures.
Thm: If the access sequence $X$ avoids a pattern $P$, then for any initial tree $T$, $\text{GREEDY}_T(X)$ avoids $P \otimes \text{Cap}$, where $\text{Cap} = (\cdot \cdot \cdot)$.

For $P = \left(\begin{array}{ccc} \cdot \\ \cdot \\ \cdot \end{array}\right)$, $P \otimes \text{Cap} = \left(\begin{array}{ccc} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{array}\right)$.

Proof: Assume that at time $t$ columns $a$ and $b$ are touched. If all accesses after time $t$ are to columns $\leq a$ or $\geq b$, then columns $a+1$ to $b-1$ will not be touched after time $t$.

Thus if $\text{GREEDY}_T(X)$ contains a point in $[a+1, \ldots, b-1] \times [t+1, \ldots, t+c]$, $X$ must contain a point in this set.
Thm: If the access sequence $X$ avoids a pattern $P$, then for any initial tree $T$, $\text{GREEDY}_T(X)$ avoids $P \otimes \text{Cap}$, where $\text{Cap} = (\cdot \cdot \cdot)$.

For $P = (\cdot \cdot \cdot)$, $P \otimes \text{Cap} = \begin{pmatrix} \cdot & \cdot & \cdot \end{pmatrix}$.

Thm: For preorder access matrix $P$, $P \otimes \text{Cap}$ is light and hence $\text{GREEDY}$ with arbitrary initial tree serves preorder sequences with cost $n \cdot 2^{\alpha(n)O(1)}$. 
Thm: If access sequence $X$ avoids a pattern $P$, then $\text{GREEDY}(X)$ avoids $P \otimes P'$, where $P'$ is a particular permutation matrix (of the same size as $P$).

This proof is more involved, but not really difficult.

For $P = \begin{pmatrix} \cdot & \cdot \\ \cdot & \cdot \end{pmatrix}$, $P \otimes P'$ is $9 \times 9$.

Thm: Greedy with chosen initial tree satisfies traversal conjecture.
A 4-decomposable permutation.

- $k$-decomposable = avoid all non-decomposable permutations of size $k + 1$ or more.
- OPT serves $k$-decomposable permutations with cost $O(n \log k)$.

We introduce an offline variant of GREEDY and analyse its behavior on $k$-decomposable permutations.
Decomposable Permutations

- $k$-decomposable = avoid all non-decomposable permutations of size $k + 1$ or more.
- **OPT** serves $k$-decomposable permutations with cost $O(n \log k)$.
- **GREEDY** serves $k$-decomposable sequences with cost $O(n^2 \alpha(n)^{O(k^2)})$.
- **GREEDY** with chosen initial tree serves with cost $O(n^2^{O(k^2)})$.
\texttt{GREEDY} has dynamic finger property, i.e., total search cost is $O(\sum_i \log |x_i - x_{i-1}|)$.

Cole has previously proven dynamic finger property for splay trees (80 page paper, complex proof).

Proof for \texttt{GREEDY} is 10 pages and easy to check.
Summary

- A wide class of BSTs with logarithmic access cost, static optimality, working set and static finger property.
- **GREEDY** does well on inputs that avoid patterns. In particular,
  - Traversal conjecture almost holds for GREEDY.
  - Traversal conjecture holds for GREEDY with chosen initial tree.
- $k$-decomposable sequences are easy: $O(n \log k)$.
- A new challenge for self-adjusting BSTs.
- Next step: arbitrary initial tree, accesses are deque-like, e.g., 1,2,3,n,n-1,4,5,n-2,n-3,n-4,6,....
Summary

- A wide class of BSTs with logarithmic access cost, static optimality, working set and static finger property.
- \textsc{Greedy} does well on inputs that avoid patterns. In particular,
  - Traversal conjecture almost holds for \textsc{Greedy}.
  - Traversal conjecture holds for \textsc{Greedy} with chosen initial tree.
- $k$-decomposable sequences are easy: $O(n \log k)$.
- A new challenge for self-adjusting BSTs.

- Next step: arbitrary initial tree, accesses are deque-like, e.g., $1,2,3,n,n-1,4,5,n-2,n-3,n-4,6,\ldots$
Summary

- A wide class of BSTs with logarithmic access cost, static optimality, working set and static finger property.
- **GReedy** does well on inputs that avoid patterns. In particular,
  - Traversal conjecture almost holds for GReedy.
  - Traversal conjecture holds for GReedy with chosen initial tree.

- $k$-decomposable sequences are easy: $O(n \log k)$.

- A new challenge for self-adjusting BSTs.

- Next step: arbitrary initial tree, accesses are deque-like, e.g., 1,2,3,n,n-1,4,5,n-2,n-3,n-4,6,....
A wide class of BSTs with logarithmic access cost, static optimality, working set and static finger property.

**GREEDY** does well on inputs that avoid patterns. In particular,
- Traversal conjecture almost holds for **GREEDY**.
- Traversal conjecture holds for **GREEDY** with chosen initial tree.

$k$-decomposable sequences are easy: $O(n \log k)$.

A new challenge for self-adjusting BSTs.

Next step: arbitrary initial tree, accesses are deque-like, e.g., 1,2,3,n,n-1,4,5,n-2,n-3,n-4,6,....