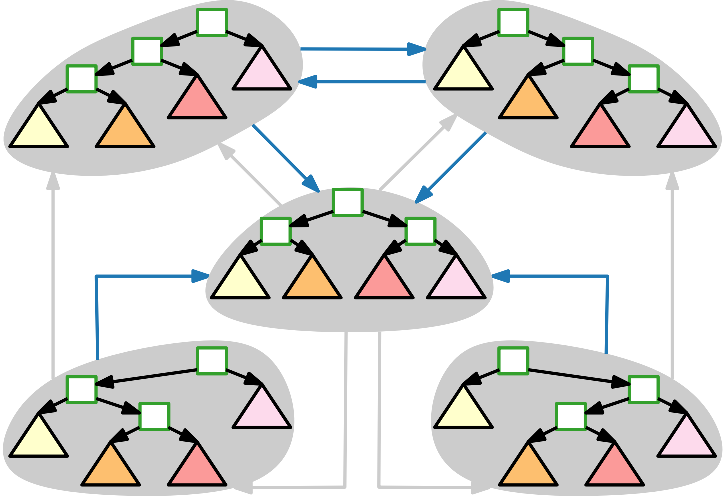
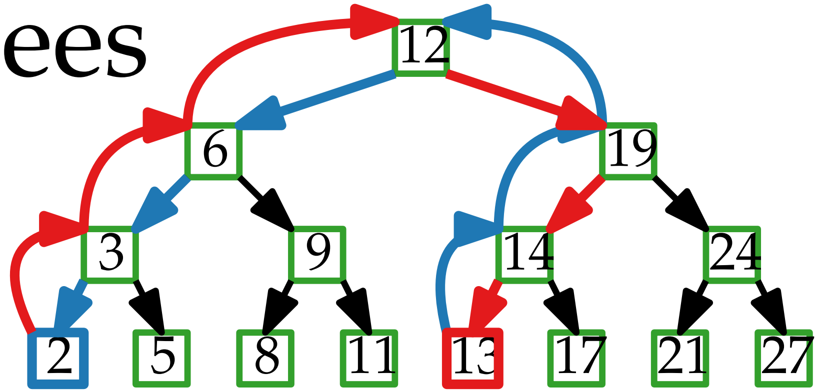
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Advanced Algorithms

Optimal Binary Search Trees

Splay Trees





Philipp Kindermann · WS20

Part I:
How Good is a Binary Search Tree?

How Good is a Binary Search Tree?

optimal

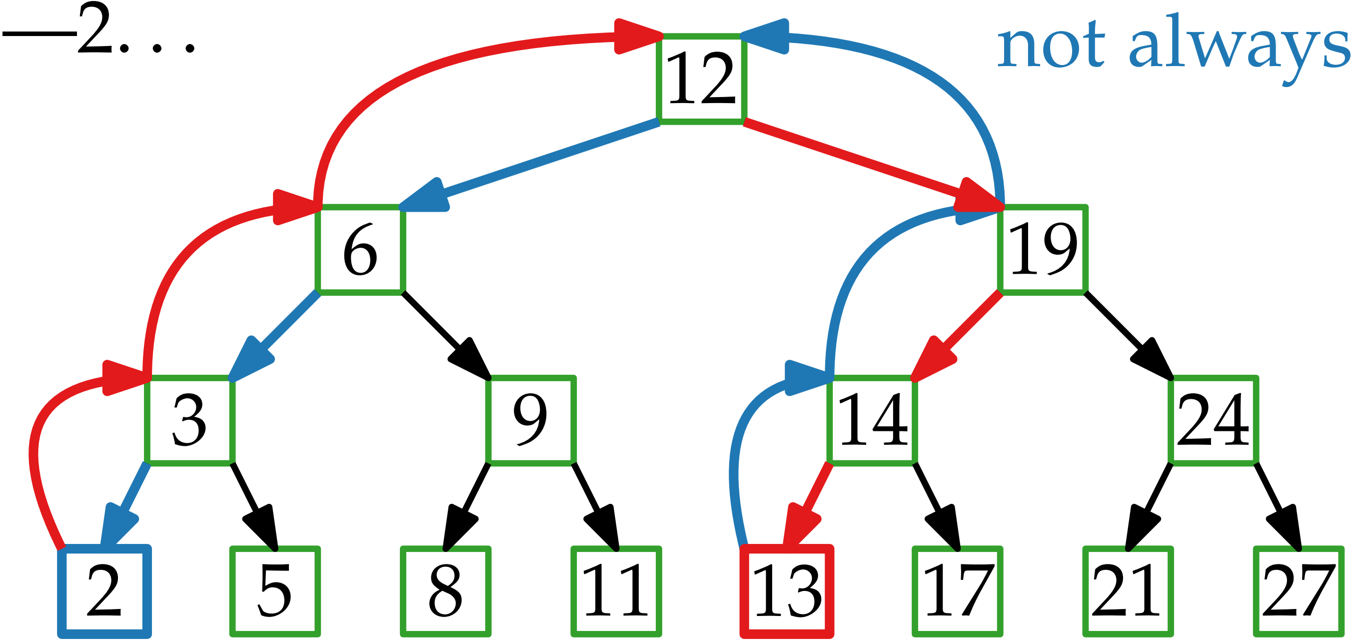
Binary search tree: w.c. query time $\Theta(n)$

Balanced binary search tree: w.c. query time $\Theta(\log n)$
(e.g. Red-Black-Tree)

What if we *know* the query before? w.c. query time 1

Sequence of queries? $O(\log n)$ per query
e.g. 2—13—5
or 2—13—2—13—2. . .

optimal?
not always!

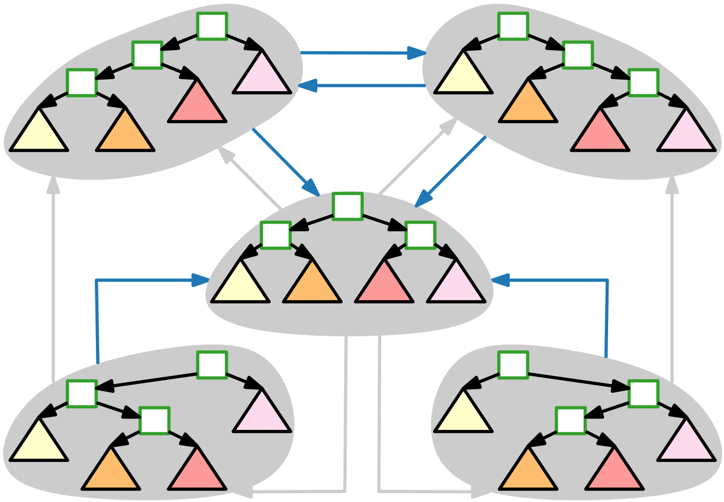


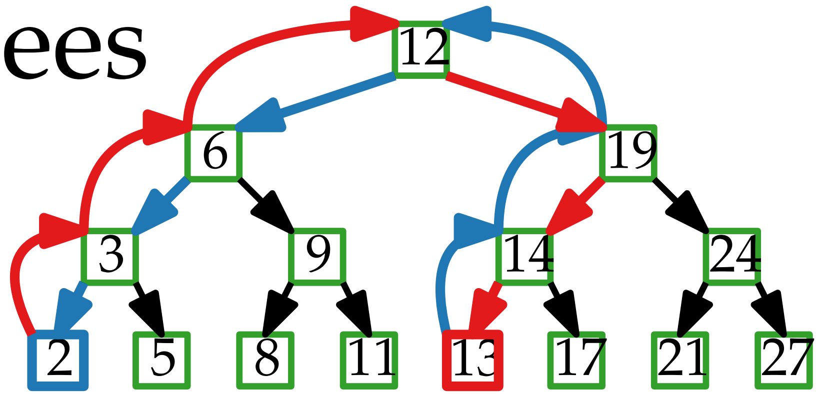
The performance of a BST depends on the model!

Advanced Algorithms

Optimal Binary Search Trees

Splay Trees





Philipp Kindermann · WS20

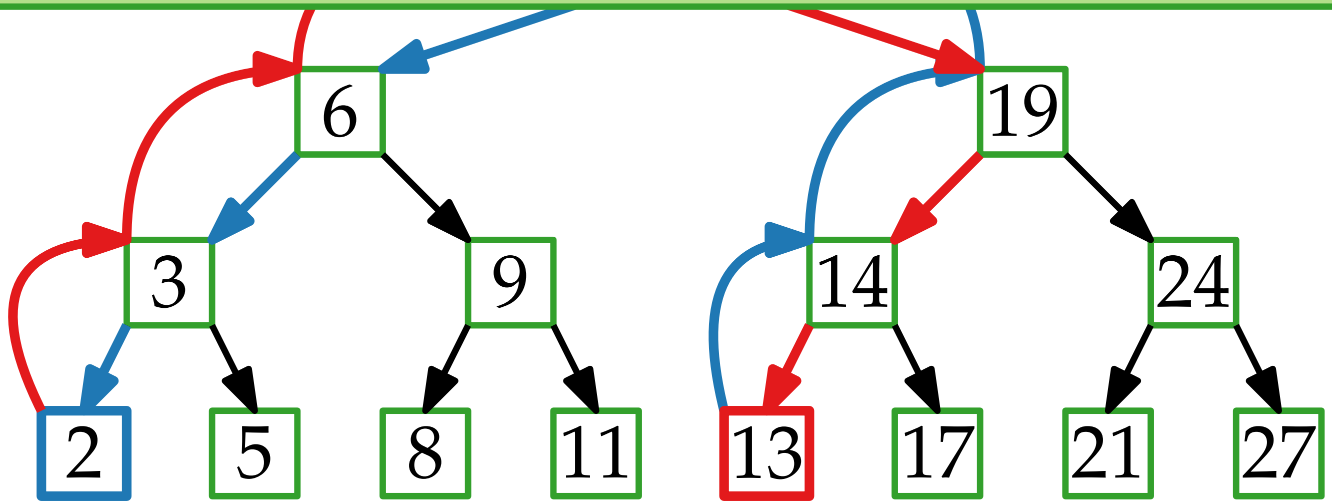
Part II:
Models of Optimality

Model 1: Malicious Queries

Given a BST, what is the worst sequence of queries?

Lemma. The worst-case malicious query cost in any BST with n nodes is at least $\Omega(\log n)$ per query.

Definition. A BST is **balanced** if the cost of *any* sequence of m queries is $O(m \log n + n \log n)$.
$\Rightarrow$ the (amortized) cost of each query is $O(\log n)$ (for at least n queries)

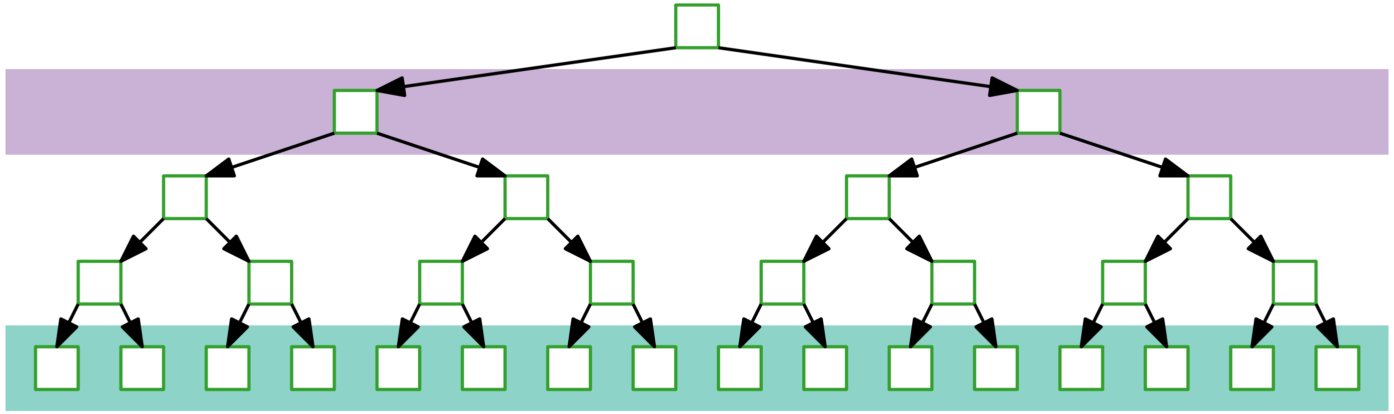


Model 2: Known Probability Distribution

Access Probabilities:

2	3	5	6	8	9	11
2%	20%	30%	8%	20%	15%	5%

Idea: Place nodes with higher probability higher in the tree.



prob. $\leq 1/2$

OPT: prob. $p \Rightarrow$ level $\log(1/p)$

prob. $\leq 1/2^{\ell-1}$

Lemma. The expected query cost in any BST is at least $\Omega(1+H)$ per query with $H = \sum_{i=1}^{n} -p_i \log p_i$.

Definition. A BST has the **entropy property** if it reaches this bound.

$$p_i = 1/n \Rightarrow H = \sum_{i=1}^{n} 1/n \cdot \log(n) = \log n$$

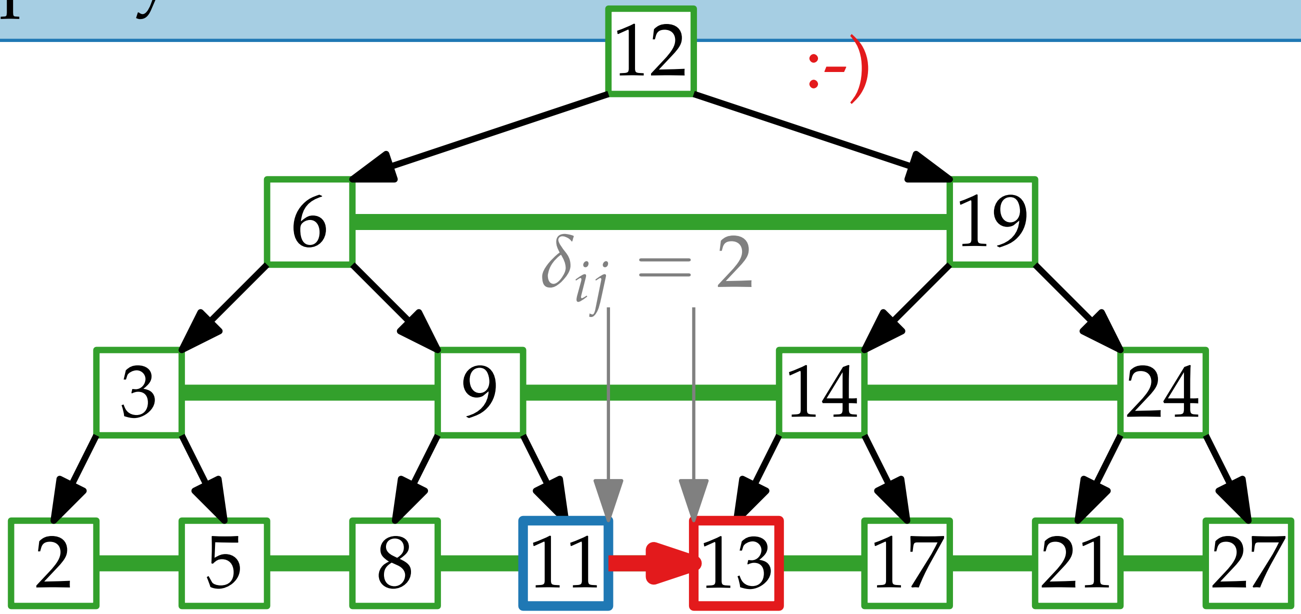
$$p_1 = 1, p_i = 0 \Rightarrow H = -\log 1 = 0$$

Model 3: Spacial Locality

If a key is queried, then keys with nearby values are more likely to be queried.
Suppose we queried key x_i and want to query key x_j next.
Let $\delta_{ij} = |\,\mathrm{rank}(x_j) - \mathrm{rank}(x_i)|$.

Definition. A BST has the **dynamic finger property** if the (amortized) cost of queries are $O(\log \delta_{ij})$.

Lemma. A level-linked Red-Black-Tree has the dynamic finger property.



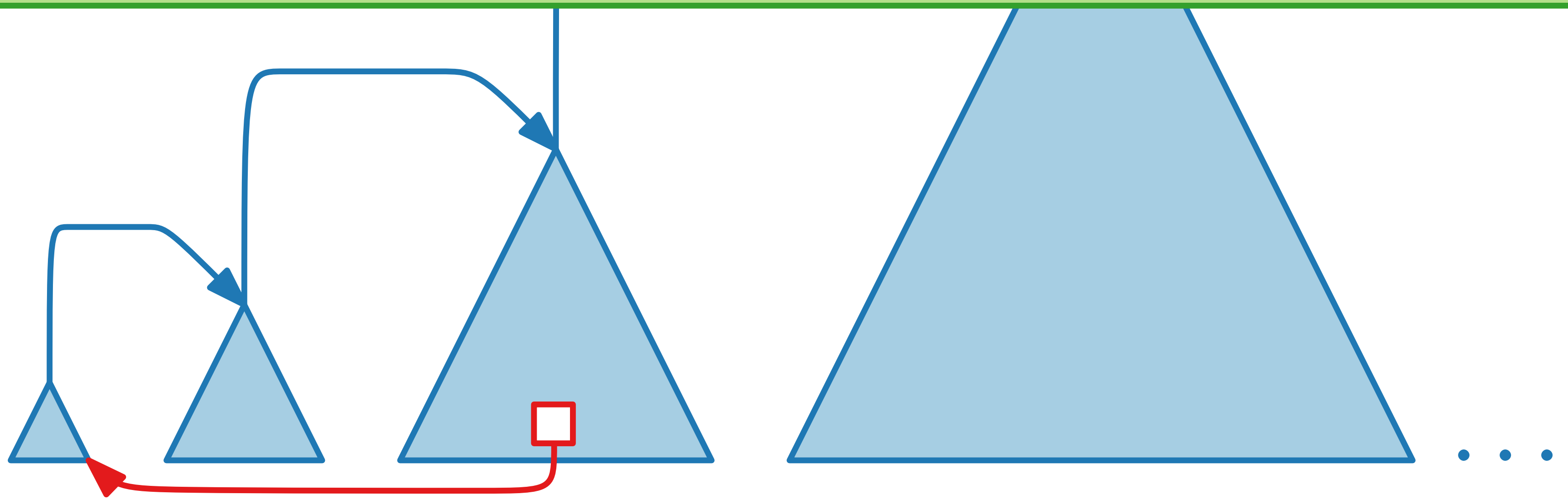
Model 4: Temporal Locality

If a key is queried, then it's likely to be queried again soon.

A static tree will have a hard time...
What if we can move elements?

Idea: Use a sequence of trees
Move queried key to first tree, then kick out oldest key.

Definition. A BST has the **working set property** if the (amortized) cost of a query for key x is $O(\log t)$, where t is the number of keys queried more recently than x.



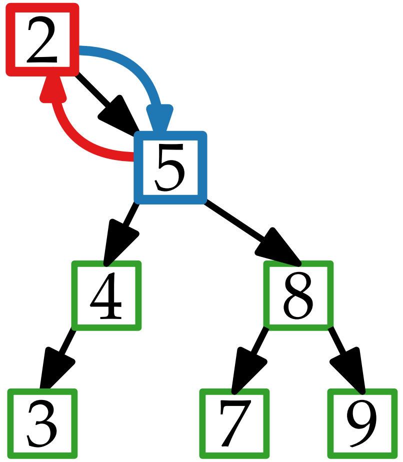
Model 5: Static Optimality

Given a sequence S of queries.
Let T_S^* be the *optimal* static tree with the shortest query time OPT_S for S.

e.g. $S = 2, 5, 2, 5, 2, \dots, 5$

T^*:

OPT: $|S|$



Definition. A BST is **statically optimal** if queries take (amort.) $O(\mathrm{OPT}_S)$ time for every S.

All These Properties...

Balanced: Queries take (amort.) $O(\log n)$ time

Entropy: Queries take expected $O(1 + H)$ time

Dynamic Finger: Queries take $O(\log \delta_i)$ time (δ_i: rank diff.)

Working Set: Queries take $O(\log t)$ time (t: recency)

Static Optimality: Queries take (amort.) $O(\mathrm{OPT}_S)$ time.

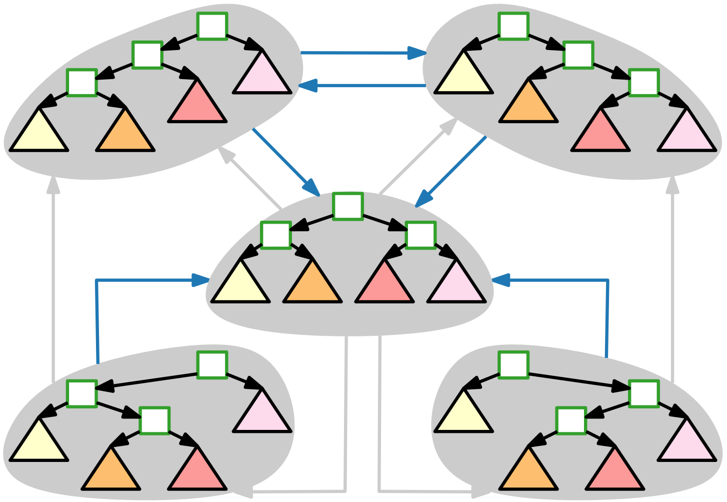
... is there one BST to rule them all?

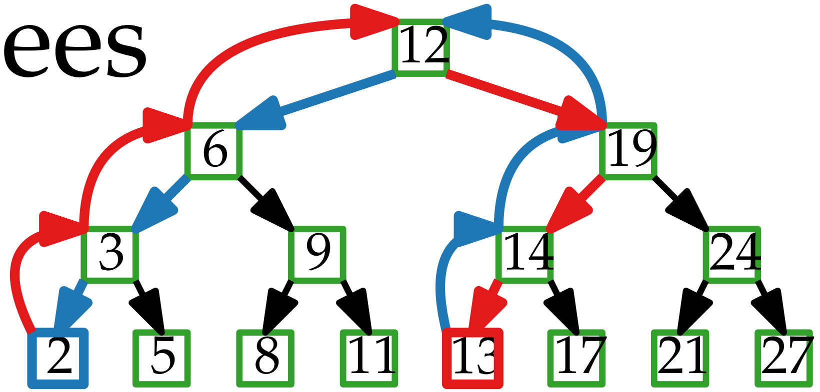
Yes!

Advanced Algorithms

Optimal Binary Search Trees

Splay Trees

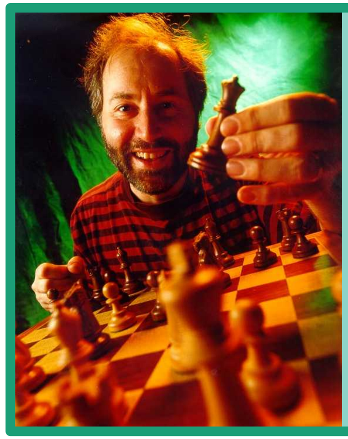




Philipp Kindermann · WS20

Part III:
Splay Trees

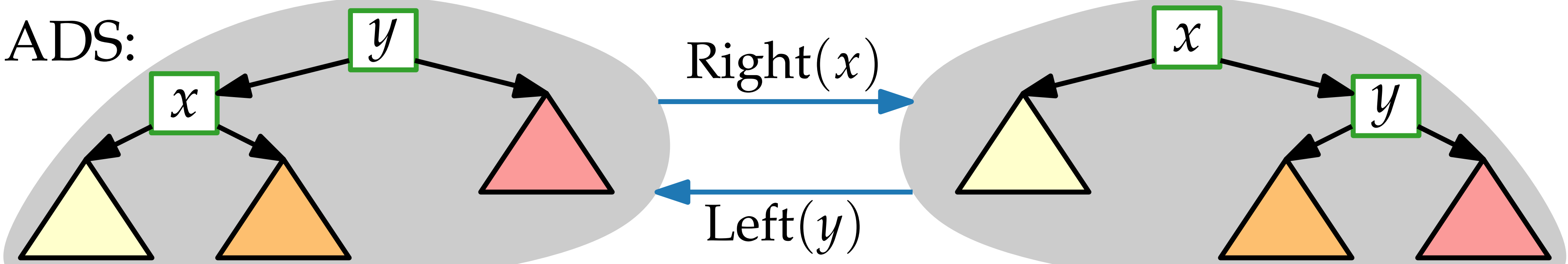
Splay Trees



Daniel D. Sleator
Robert E. Tarjan
J. ACM 1985

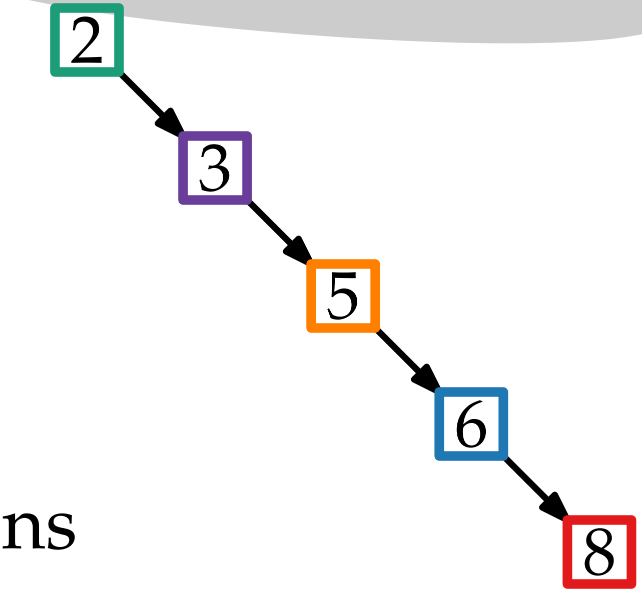
Idea: Whenever we query a key, rotate it to the root.

ADS:



Splay(x): Rotate x to the root

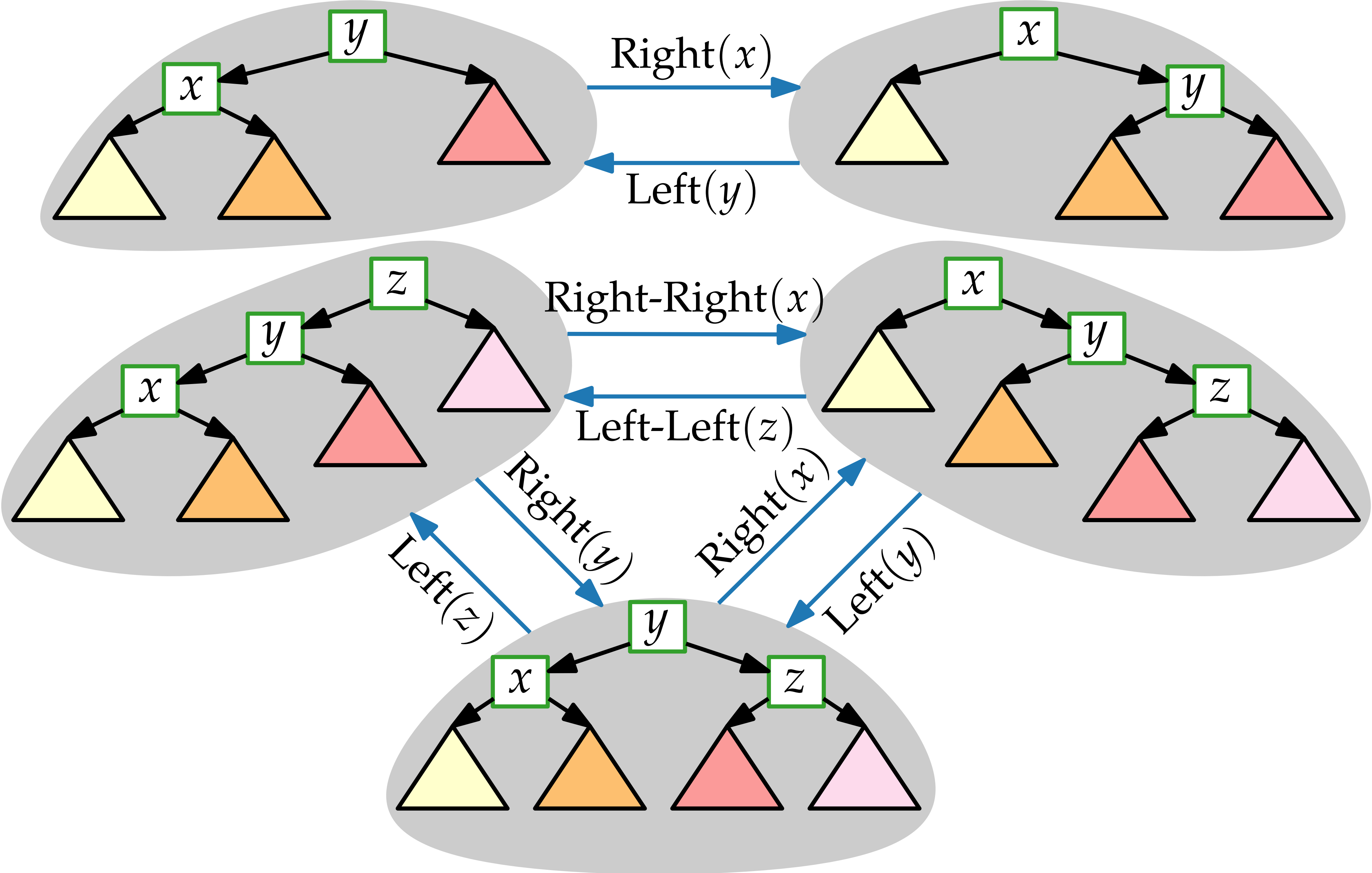
Query(x): Splay(x), then return root



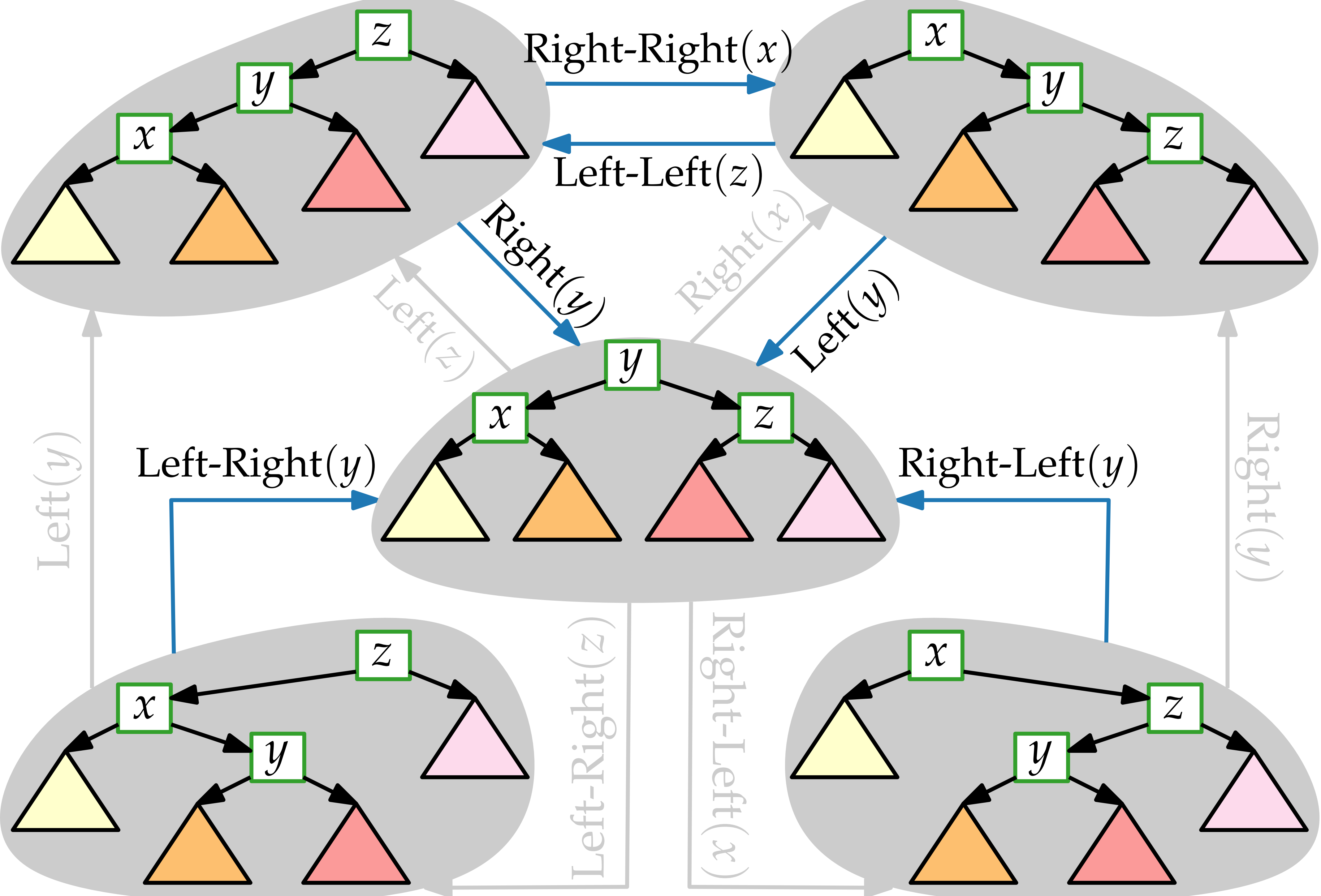
Query(8) Query(6) Query(5)
Query(3)
Query(2)

We're back at the start...
and we did $\Theta(n^2)$ rotations

Rotations II



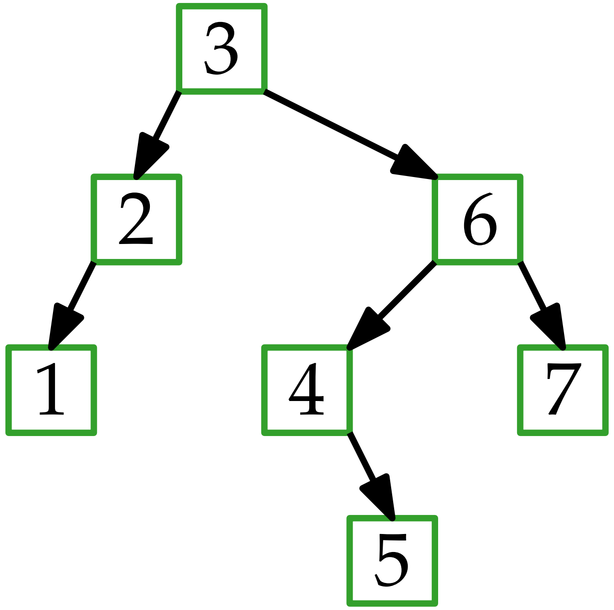
Rotations II



Splay

```
**Algorithm:** Splay($x$)
if $x \neq root$ then
    $y$ = parent of $x$
    if $y = root$ then
        if $x < y$ then Right($x$)
        if $y < x$ then Left($x$)
    else
        $z$ = parent of $y$
        if $x < y < z$ then Right-Right($x$)
        if $z < y < x$ then Left-Left($x$)
        if $y < x < z$ then Left-Right($x$)
        if $z < x < y$ then Right-Left($x$)
    Splay($x$)
```

Splay(3):



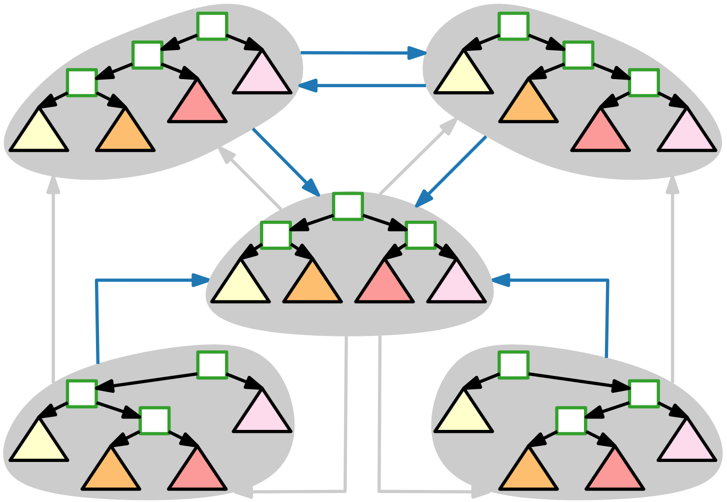
Call Splay(x):

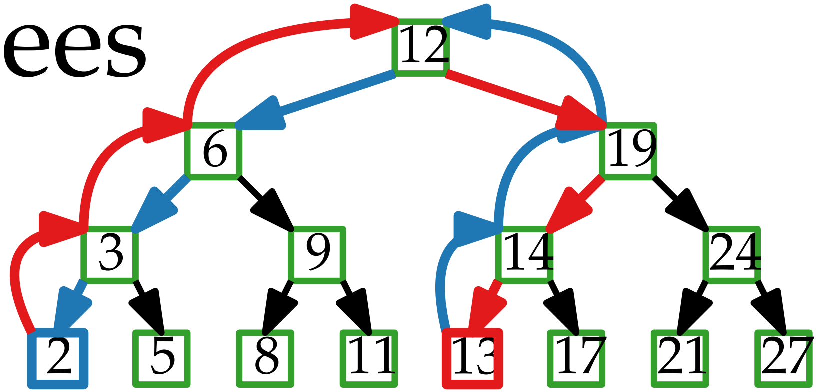
- after Search(x)
- after Insert(x)
- before Delete(x)

Advanced Algorithms

Optimal Binary Search Trees

Splay Trees





Philipp Kindermann · WS20

Part IV:
Potential

Why is Splay Fast?

$w(x)$: weight of x (here 1), $W = \sum w(x)$ (here n)

$s(x)$: sum of all $w(x)$ in subtree of x_i

mark edges:

$\longrightarrow$ $s(\text{child}) \leq s(\text{parent})/2$

$\longrightarrow$ $s(\text{child}) > s(\text{parent})/2$

Cost to query x_i: $O(\log W + \#\text{red})$

Idea: blue edges halve the weight

$\Rightarrow \#\text{blue} \in O(\log W)$

How can we amortize red edges?

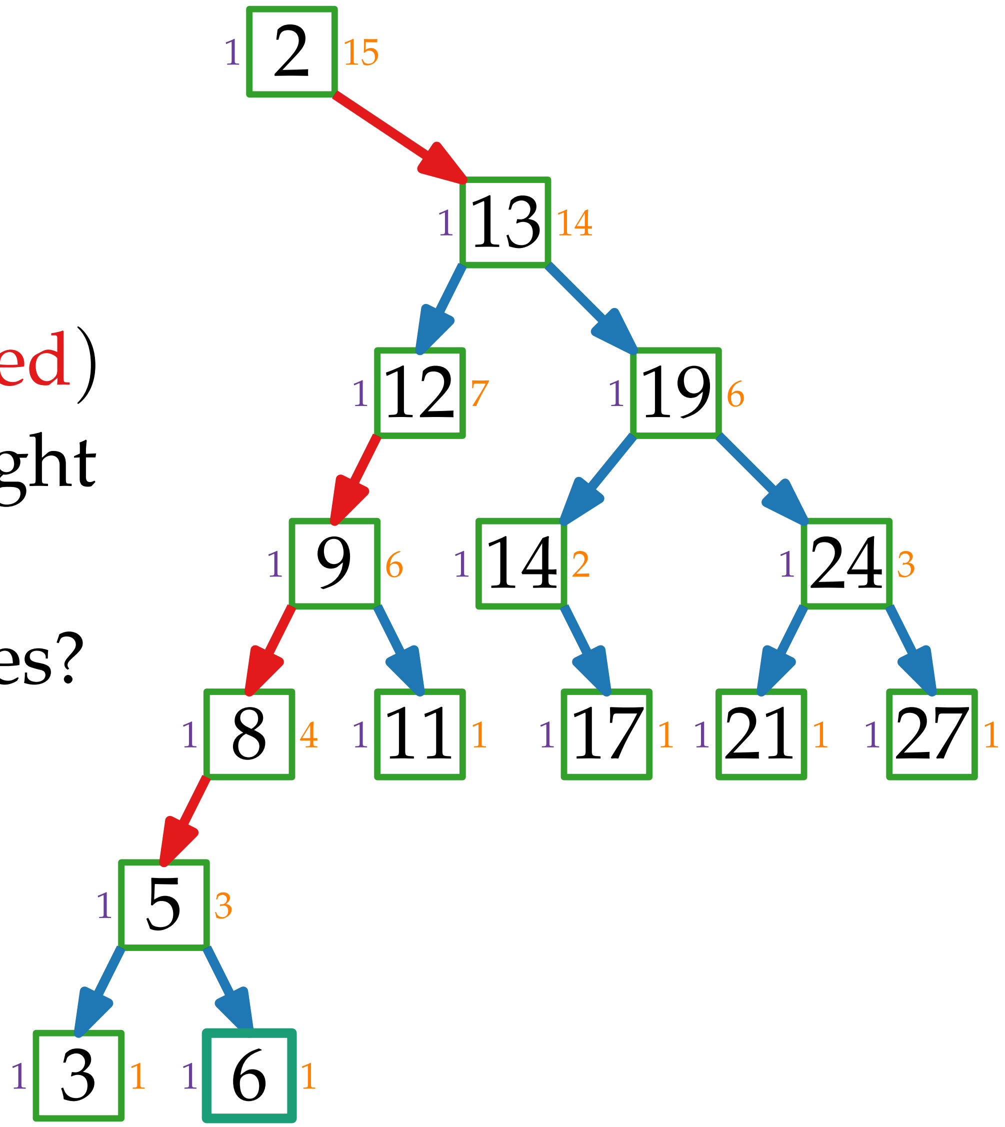
Use sum-of-logs potential

$\Phi = \sum \log s(x)$

Amortized cost:

real cost + $\Phi_+ - \Phi$

(potential after splay)



What is Potential?

Φ represent work that has been “paid for” but not yet performed.

amortized cost per step: real cost $+ \Phi_+ - \Phi$

total cost $= \Phi_0 - \Phi_{\text{end}} + \sum$ amortized cost

Example (ADS): Stack with multipop

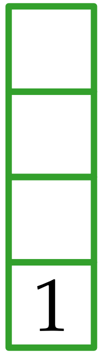
Φ :=size of the stack

push: $1 + \Phi_+ - \Phi \; = 2$

$\text{pop}(k)$: $k + \Phi_+ - \Phi \; = 0$

$$\begin{aligned} \text{total cost} &= \Phi_0 - \Phi_{\text{end}} + \text{amortized cost} \\ &\leq \Phi_0 - \Phi_{\text{end}} + 2n \\ &\leq 2n \; = O(n) \end{aligned}$$

$\text{pop}(2)$



$\Phi = 1$

Why is Splay Fast?

$w(x)$: weight of x (here 1), $W = \sum w(x)$ (here n)

$s(x)$: sum of all $w(x)$ in subtree of x_i

mark edges:

→ $s(\text{child}) \leq s(\text{parent})/2$

→ $s(\text{child}) > s(\text{parent})/2$

Cost to query x_i: $O(\log W + \#\text{red})$

Idea: blue edges halve the weight

$\Rightarrow \#\text{blue} \in O(\log W)$

How can we amortize red edges?

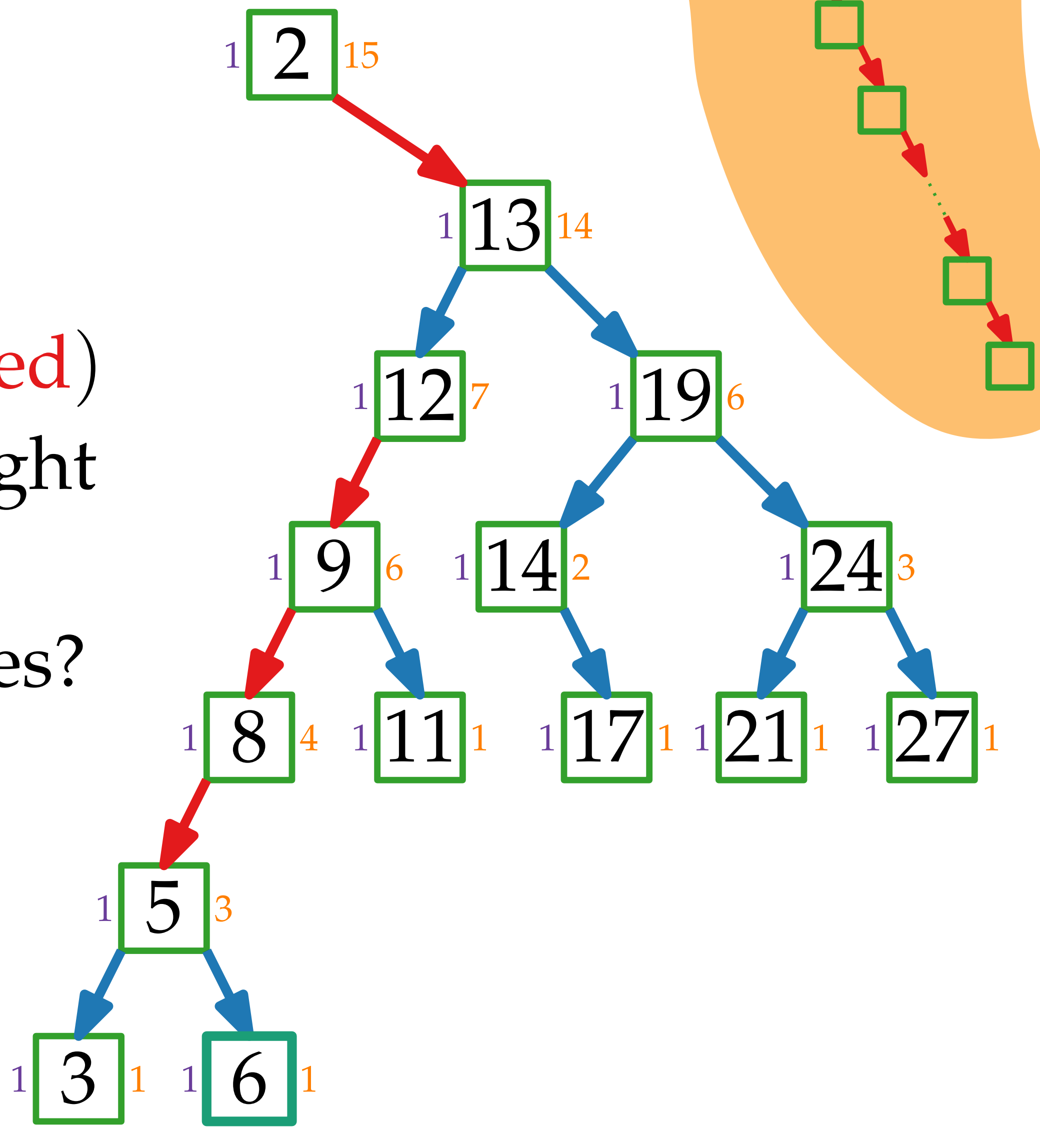
Use sum-of-logs potential

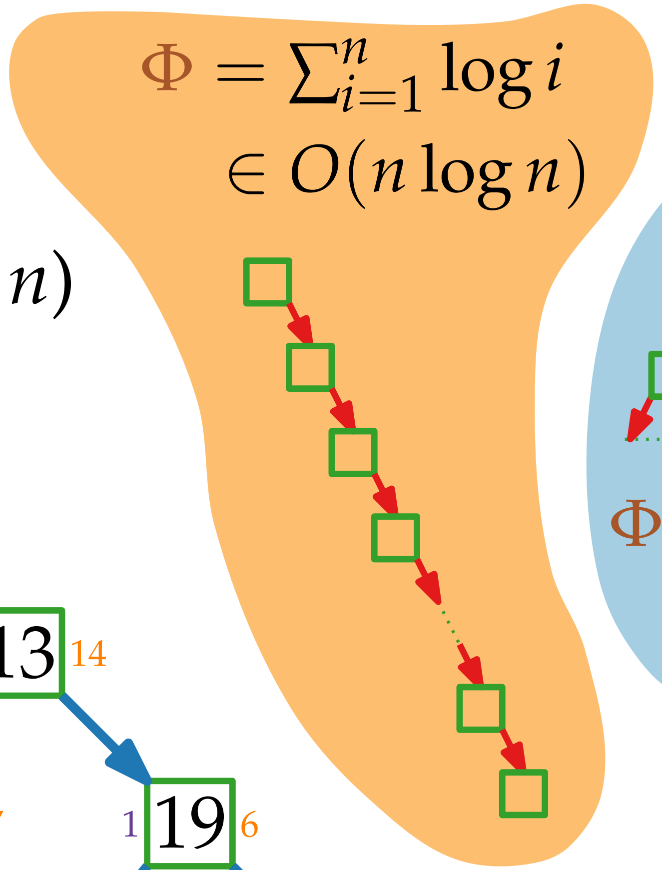
$\Phi = \sum \log s(x)$

Amortized cost:

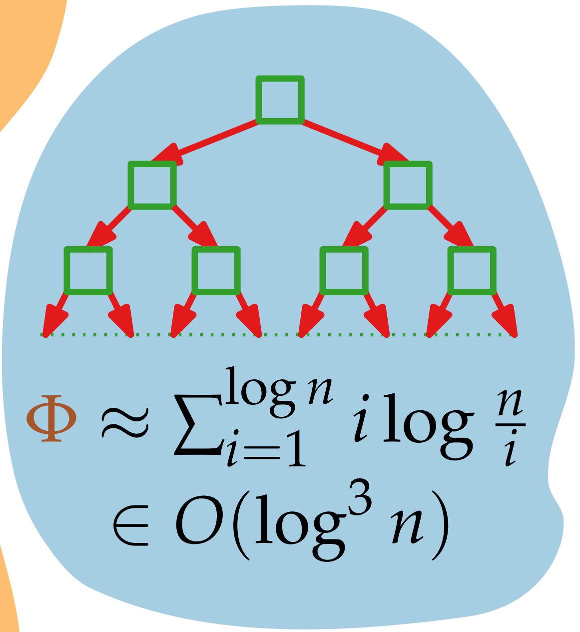
real cost + $\Phi_+ - \Phi$

(potential after splay)





$\Phi = \sum_{i=1}^{n} \log i$
$\in O(n \log n)$

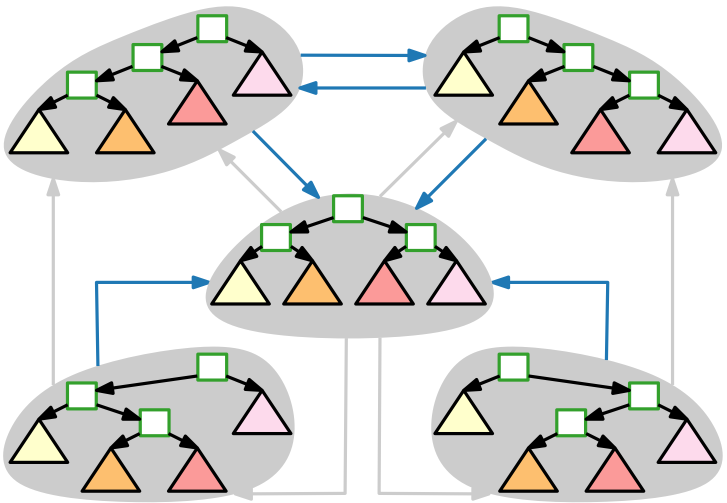


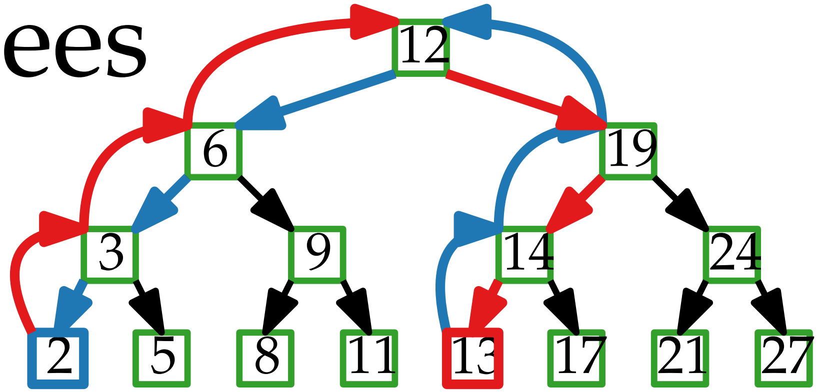
$\Phi \approx \sum_{i=1}^{\log n} i \log \frac{n}{i}$
$\in O(\log^3 n)$

Advanced Algorithms

Optimal Binary Search Trees

Splay Trees





Philipp Kindermann · WS20

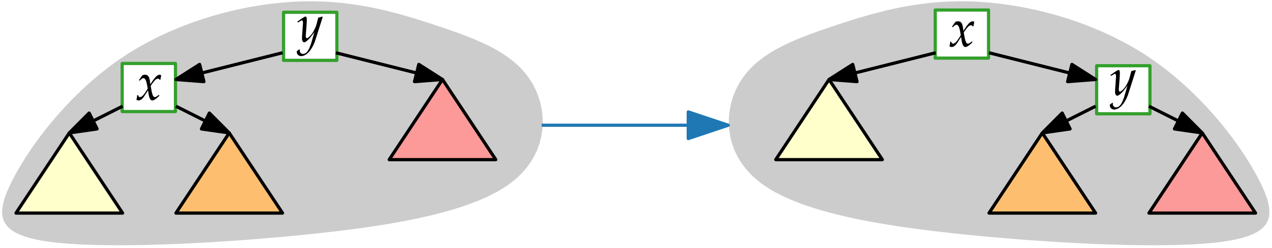
Part V:
Access Lemma and Running Time of Splay

Potential after Rotation

Consider any rotation; $s(x)$ before rotation, $s_+(x)$ afterwards

Lemma. After a single rotation, the potential increases by $\leq 3\,(\log s_+(x) - \log s(x))$.

Proof. Right(x)



Observe: Only $s(x)$ and $s(y)$ change.

$$\begin{aligned} \text{pot. change} \quad &= \log s_+(x) + \log s_+(y) \\ &\quad - \log s(x) - \log s(y) \\ (s_+(y) \leq s(y)) \quad &\leq \log s_+(x) - \log s(x) \\ (s_+(x) > s(x)) \quad &\leq 3\,(\log s_+(x) - \log s(x)) \quad \checkmark \end{aligned}$$

Left(x) analogue ✓

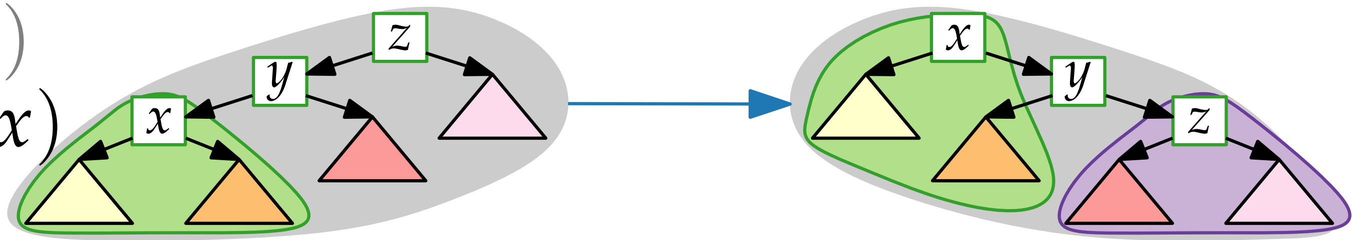
Potential after Rotation

Consider any rotation; $s(x)$ before rotation, $s_+(x)$ afterwards

Lemma. After a double rotation, the potential increases by $\leq 3\,(\log s_+(x) - \log s(x)) - 2$.

Proof. / Left-Left(x)

Case 1. Right-Right(x)



$$\begin{aligned}
&\text{pot. change} && = \log s_+(x) + \log s_+(y) + \log s_+(z) \\
& && \quad - \log s(x) - \log s(y) - \log s(z) \\
&(s_+(x) = s(z)) && = \log s_+(y) + \log s_+(z) - \log s(x) - \log s(y) \\
&(s(x) \leq s(y)) && \leq \log s_+(y) + \log s_+(z) - 2\log s(x) \\
&(s_+(y) \leq s_+(x)) && \leq \log s_+(x) + \log s_+(z) - 2\log s(x) \\
& && \leq 3\log s_+(x) - 3\log s(x) - 2 \quad \checkmark
\end{aligned}$$

$$s(x) + s_+(z) \leq s_+(x) \Rightarrow \log s(x) + \log s_+(z) = \log s(x)s_+(z)$$

$$\underset{\text{(AM-GM)}}{\leq} \log\,(s_+(x)/2)^2 \leq 2\log s_+(x) - 2$$

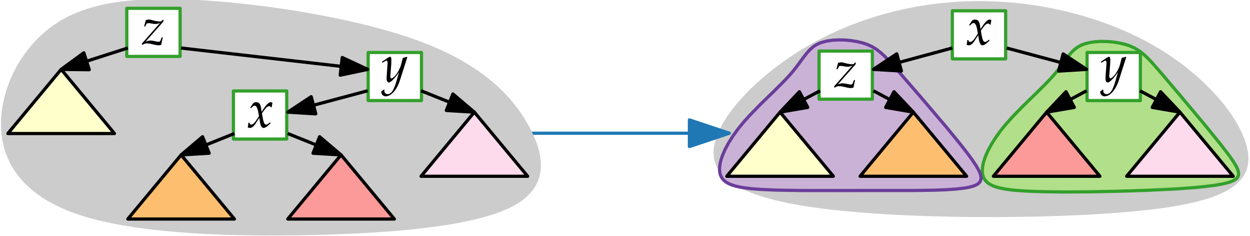
Potential after Rotation

Consider any rotation; $s(x)$ before rotation, $s_+(x)$ afterwards

Lemma. After a double rotation, the potential increases by $\leq 3\,(\log s_+(x) - \log s(x)) - 2$.

Proof. / Left-Right(x)

Case 2. Right-Left(x)



$$\begin{aligned}
\text{pot. change} \quad &= \log s_+(x) + \log s_+(y) + \log s_+(z) \\
&\quad - \log s(x) - \log s(y) - \log s(z) \\
(s_+(x) = s(z)) \quad &= \log s_+(y) + \log s_+(z) - \log s(x) - \log s(y) \\
(s(x) \leq s(y)) \quad &\leq \log s_+(y) + \log s_+(z) - 2\log s(x) \\
&\leq 2\log s_+(x) - 2\log s(x) - 2 \\
(s_+(x) > s(x)) \quad &\leq 3\log s_+(x) - 3\log s(x) - 2
\end{aligned}$$

✓

$s_+(y) + s_+(z) \leq s_+(x) \quad \Rightarrow \log s_+(y) + \log s_+(z)$
$\leq 2\log s_+(x) - 2$

Access Lemma

Lemma. After a single rotation, the potential increases by $\leq 3\,(\log s_+(x) - \log s(x))$.
After a double rotation, the potential increases by $\leq 3\,(\log s_+(x) - \log s(x)) - 2$.

Lemma. The (amortized) cost of Splay(x) is $\leq 1 + 3\log(W/w(x))$

Proof. W.l.o.g. k double rotations and 1 single rotation.
Let $s_i(x)$ be $s(x)$ after i single/double rotations.
Potential increases by at most

$$\sum_{i=1}^{k} (3\,(\log s_i(x) - \log s_{i-1}(x)) - 2)$$
$$+3\,(\log s_{k+1}(x) - \log s_k(x))$$

root!

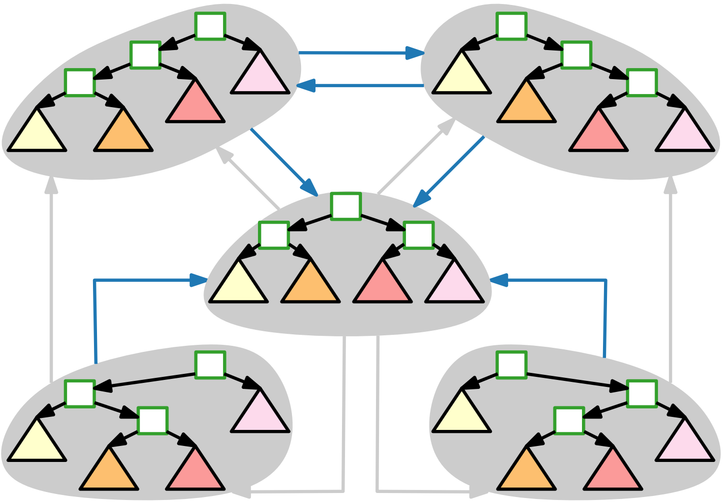
$$= 3\,(\log s_{k+1}(x) - \log s(x)) - 2k$$
$$= 3\,(\log W - \log s(x)) - 2k$$
$$(s(x) \geq w(x)) \quad \leq 3\,(\log W - \log w(x)) - 2k \;= 3\log(W/w(x)) - 2k$$

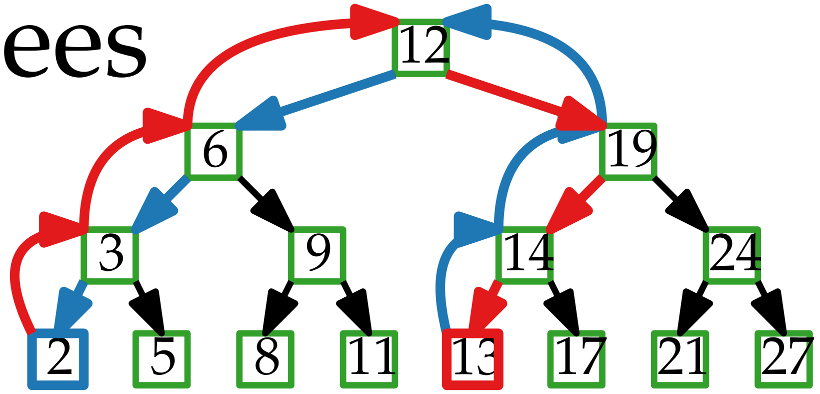
$2k + 1$ rotations $\Rightarrow$ (amort.) cost $\leq 1 + 3\log(W/w(x))$ $\square$

Advanced Algorithms

Optimal Binary Search Trees

Splay Trees





Philipp Kindermann · WS20

Part VI:
Properties of Splay Trees

All These Properties...

Balanced: Queries take (amort.) $O(\log n)$ time

Entropy: Queries take expected $O(1 + H)$ time

Dynamic Finger: Queries take $O(\log \delta_i)$ time (δ_i: rank diff.)

Working Set: Queries take $O(\log t)$ time (t: recency)

Static Optimality: Queries take (amort.) $O(\mathrm{OPT}_S)$ time.

... is there one BST to rule them all?

Yes!

Querying a Sequence

Let S be a sequence of queries.

What is the *real* cost of querying S?

Let Φ_k be the potential after query k.

$\Rightarrow$ total cost $\Phi_0 - \Phi_{|S|} + \sum_{x \in S} \text{Splay}(x)$

How can we bound $\Phi_0 - \Phi_{|S|}$?

Reminder: $\Phi = \sum \log s(x)$

$s(x) \geq w(x) \qquad \Rightarrow \Phi_{|S|} \geq \sum_{x \in T} \log w(x)$

$s(\text{root}) = \log W \qquad \Rightarrow \Phi_0 \leq \sum_{x \in T} \log W$

$\Rightarrow \Phi_0 - \Phi_{|S|} \leq \sum_{x \in T} (\log W - \log w(x)) \leq \sum_{x \in T} O\left(\text{Splay}(x)\right)$

$\Rightarrow$ as long as every key is queried at least once, it doesn't change the asymptotic running time.

Balance

Lemma. The (amortized) cost of Splay(x) is $\leq 1 + 3\log(W/w(x))$

Definition. A BST is **balanced** if the (amortized) cost of *any* query is $O(\log n)$ (for at least n queries in total).

Theorem. Splay Trees are balanced.

Proof. Choose $w(x) = 1$ for each x $\Rightarrow W = n$

Splay(x) costs at least as much as finding x

$$\begin{aligned}\Rightarrow \text{total time} &= \Phi_0 - \Phi_{|S|} + \textstyle\sum_{x \in S} \text{Splay}(x) \\ &\leq \textstyle\sum_{x \in T} (\log W - \log w(x)) + \sum_{x \in S} \text{Splay}(x) \\ &\leq n \log n + \textstyle\sum_{x \in S} (1 + 3\log(W/w(x))) \\ &\leq n \log n + |S| + 3|S| \log n \ \in O(|S| \log n)\end{aligned}$$

$\Rightarrow$ Queries take (amort.) $O(\log n)$ time. $\square$

Entropy

Lemma. The (amortized) cost of Splay(x) is $\leq 1 + 3\log(W/w(x))$

Definition. A BST has the **entropy property** if queries take expected $O(1 - \sum_{i=1}^{n} p_i \log p_i)$ time.

Theorem. Splay Trees have the entropy property.

Proof. Choose $w(x_i) = p_i \quad \Rightarrow W = 1$

Time to query x_i:

$$\leq 1 + 3\log(W/w(x_i))$$
$$= 1 + 3\log(1/p_i)$$
$$= 1 - 3\log p_i$$

$\Rightarrow$ expected query time:

$$O(\textstyle\sum_{i=1}^{n} p_i(1 - 3\log p_i))$$
$$= O(1 - \textstyle\sum_{i=1}^{n} p_i \log p_i) \qquad \square$$

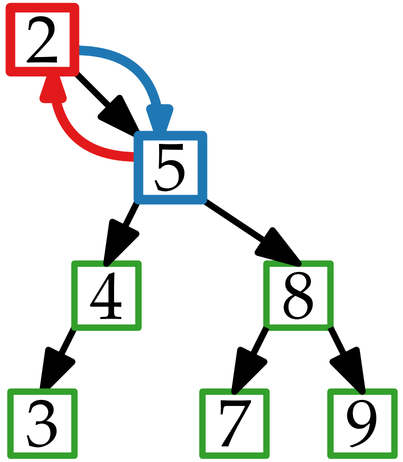
Static Optimality

Given a sequence S of queries.
Let T^*_S be the *optimal* static tree with the shortest query time OPT_S for S.

e.g. $S = 2, 5, 2, 5, 2, \ldots, 5$

T^*:

OPT: $|S|$



Definition. A BST is **statically optimal** if queries take (amort.) $O(\mathrm{OPT}_S)$ time for every S.

Theorem. Splay Trees are statically optimal.

Proof. Let f_i be the number of items on path to x_i in T^*.

Let $w_i := 3^{-f_i}$. $\Rightarrow W \leq 1$

$$\Rightarrow \mathrm{Splay}(x_i) = 1 + 3\log(W/w(x))$$
$$\leq 1 + 3\log 3^{f_i} \quad = O(f_i)$$

□

Dynamic Optimality

Given a sequence S of queries.
Let D^*_S be the optimal *dynamic* tree with the shortest query time OPT^*_S for S.
(That is, modifications are allowed, e.g. rotations)

Definition. A BST is **dynamically optimal** if queries take (amort.) $O(\mathrm{OPT}^*_S)$ time for every S.

Splay Trees: Queries take $O(\mathrm{OPT}^*_S \cdot \log n)$ time.
Tango Trees: Queries take $O(\mathrm{OPT}^*_S \cdot \log \log n)$ time.
[Demaine, Harmon, Iacono, Pătrașcu '04]

Open Problem. Does a dynamically optimal BST exist?

This is one of the biggest open problems in algorithms.

Conjecture. Splay Trees are dynamically optimal.