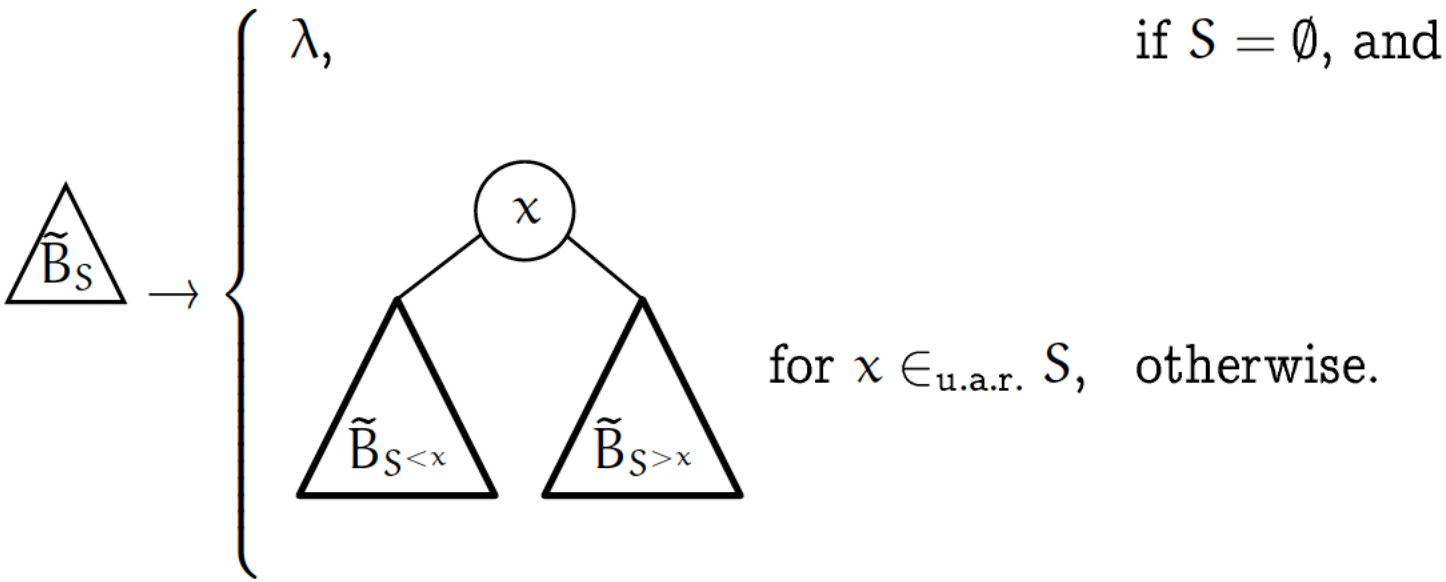
Algorithms, Probability, and Computing

Angelika Steger
Institut für Theoretische Informatik

steger@inf.ethz.ch

Random Search Trees



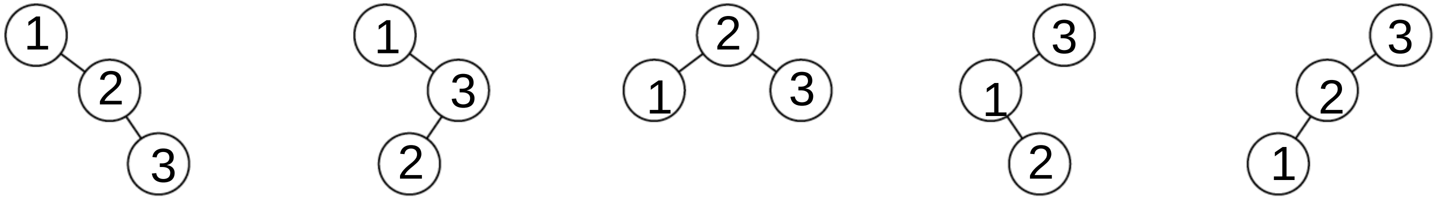
Lemma 2.1. $S \subseteq \mathbf{R}$, *finite. Given a tree in* $\mathcal{B}_S$, *we let* $w(v)$, v *a node, denote the number of nodes in the subtree rooted at* v.

The probability of the tree according to the above distribution is $\prod_v \frac{1}{w(v)}$, *where the product is over all nodes* v *of the tree.*

Depth of Smallest Key

D_n := depth of smallest key; $d_n := E[D_n]$

n=3:



$\frac{1}{6} \cdot 0$ $\frac{1}{6} \cdot 0$ $\frac{1}{3} \cdot 1$ $\frac{1}{6} \cdot 1$ $\frac{1}{6} \cdot 2$

$d_1 = 0$, $d_2 = 1/2$, $d_3 = 5/6$

Depth of Smallest Key

D_n := depth of smallest key; $d_n := E[D_n]$

$d_1 = 0$, $d_2 = 1/2$, $d_3 = 5/6$

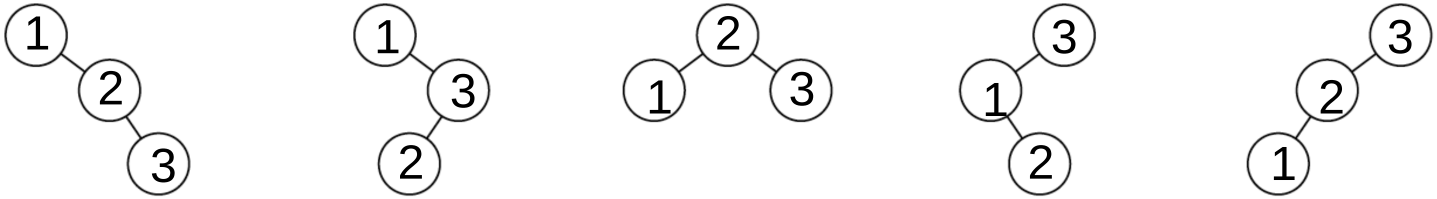
$$\mathsf{E}[D_n] = \sum_{i=1}^{n} \underbrace{\mathsf{E}[D_n \,|\, rk(root) = i]}_{= \begin{cases} 0, & \text{if } i = 1 \text{, and} \\ 1 + \mathsf{E}[D_{i-1}], & \text{otherwise.} \end{cases}} \cdot \underbrace{\Pr[rk(root) = i]}_{=1/n}$$

$$d_n = \begin{cases} 0, & \text{if } n = 1 \text{, and} \\ \frac{1}{n} \sum_{i=2}^{n} (1 + d_{i-1}), & \text{otherwise.} \end{cases}$$

Overall Depths of Keys

$X_n :=$ sum of depths of all keys in tree; $x_n := E[X_n]$

n=3:



$\frac{1}{6} \cdot (0+1+2)$ $\frac{1}{6} \cdot (0+1+2)$ $\frac{1}{3} \cdot (0+1+1)$ $\frac{1}{6} \cdot (0+1+2)$ $\frac{1}{6} \cdot (0+1+2)$

$x_1 = 0$, $x_2 = 1$, $x_3 = 8/3$

Overall Depths of Keys

$X_n :=$ sum of depths of all keys in tree; $x_n := E[X_n]$

$x_1 = 0$, $x_2 = 1$, $x_3 = 8/3$

$$E[X_n] = \sum_{i=1}^{n} \underbrace{E[X_n \,|\, rk(root) = i]}_{n-1+E[X_{i-1}]+E[X_{n-i}]} \cdot \underbrace{\Pr[rk(root) = i]}_{=1/n}$$

$$= n - 1 + \frac{1}{n} \cdot 2 \cdot \sum_{i=1}^{n} E[X_{i-1}] \,,$$

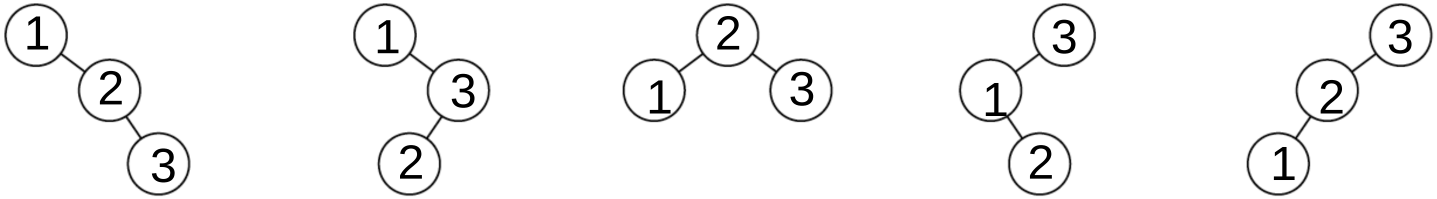
$$x_n = \begin{cases} 0, & \text{if } n = 0, \text{ and} \\ n - 1 + \frac{2}{n} \sum_{i=0}^{n-1} x_i, & \text{otherwise.} \end{cases}$$

Height

$D_n^{(i)}$:= depth of key of rank i

$X_n := \max_{1 \leqq i \leqq n} D_n^{(i)}$

n=3:



$\frac{1}{6} \cdot 2$ $\quad$ $\frac{1}{6} \cdot 2$ $\quad$ $\frac{1}{3} \cdot 1$ $\quad$ $\frac{1}{6} \cdot 2$ $\quad$ $\frac{1}{6} \cdot 2$

$x_1 = 0$, $x_2 = 1$, $x_3 = 5/3$

Height

$D_n^{(i)}$:= depth of key of rank i

$X_n := \max_{1 \leqq i \leqq n} D_n^{(i)}$

$x_1 = 0$, $x_2 = 1$, $x_3 = 5/3$

$E[X_n] = E[\max_{1 \leqq i \leqq n} D_n^{(i)}] =$? ? ?

Height

$D_n^{(i)}$:= depth of key of rank i

$X_n := \max_{1 \leqq i \leqq n} D_n^{(i)}$; $\quad x_n := E[X_n]$

$x_1 = 0, \quad x_2 = 1, \quad x_3 = 5/3$

$$\mathbf{E}[X_n] \leq \log \mathbf{E}\left[2^{X_n}\right] = \log \mathbf{E}\left[2^{\max_{i=1}^{n} D_n^{(i)}}\right]$$

Jensen's Inequality: If $f : \mathbf{R} \to \mathbf{R}$ is a convex function, then $f(\mathbf{E}[X]) \leq \mathbf{E}[f(X)]$

Height

$D_n^{(i)}$:= depth of key of rank i

$X_n := \max_{1 \leqq i \leqq n} D_n^{(i)}$; $x_n := E[X_n]$

$$\mathbf{E}[X_n] \leq \log \mathbf{E}\left[2^{X_n}\right] = \log \mathbf{E}\left[2^{\max_{i=1}^{n} D_n^{(i)}}\right]$$

$$\leq \log \mathbf{E}\left[\boxed{\sum_{i=1,\, i \text{ is leaf}}^{n} 2^{D_n^{(i)}}}\right]$$

$=: Z_n$

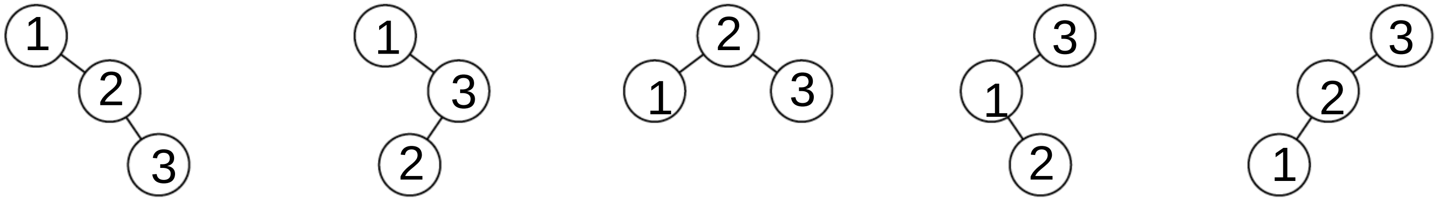
Height

$D_n^{(i)}$:= depth of key of rank i

$X_n := \max_{1 \leqq i \leqq n} D_n^{(i)}$; $\qquad E[X_n] \leqq \log_2(E[Z_n])$

$$Z_n := \sum_{i=1,\, i \text{ is leaf}}^{n} 2^{D_n^{(i)}} \qquad\qquad z_n := E[Z_n]$$

n=3:



$\frac{1}{6} \cdot 2^2 \qquad \frac{1}{6} \cdot 2^2 \qquad \frac{1}{3} \cdot 2 \cdot 2^1 \qquad \frac{1}{6} \cdot 2^2 \qquad \frac{1}{6} \cdot 2^2$

$z_1 = 1,\quad z_2 = 2,\quad z_3 = 4$

Height

$D_n^{(i)}$:= depth of key of rank i

$X_n := \max_{1 \leqq i \leqq n} D_n^{(i)}$; $\qquad E[X_n] \leqq \log_2(E[Z_n])$

$$Z_n := \sum_{i=1,\, i \text{ is leaf}}^{n} 2^{D_n^{(i)}} \qquad z_n := E[Z_n]$$

$$\mathbf{E}[Z_n] = \sum_{i=1}^{n} \underbrace{\mathbf{E}[Z_n \mid rk(root) = i]}_{2(\mathbf{E}[Z_{i-1}] + \mathbf{E}[Z_{n-i}])} \cdot \underbrace{\Pr[rk(root) = i]}_{=1/n}$$

$$z_n = \begin{cases} 0, & \text{if } n = 0, \\ 1, & \text{if } n = 1, \text{ and} \\ \frac{4}{n}\sum_{i=1}^{n} z_{i-1}, & \text{otherwise.} \end{cases}$$

Improving the Constant

$D_n^{(i)}$:= depth of key of rank i

$X_n :=\ \max_{1 \leqq i \leqq n} D_n^{(i)}$

$$\mathbf{E}[X_n] \leq \log_C \mathbf{E}\left[C^{X_n}\right] = \log \mathbf{E}\left[C^{\max_{i=1}^{n} D_n^{(i)}}\right]$$

Repeat calculations from before:

$$\mathbf{E}[X_n] < \frac{2C-1}{\ln C} \ln n \quad \text{for } n \geq 3 \text{ and any real } C > 1.$$

Optimize C:

$$\mathbf{E}[X_n] \leqq\ 4.311..\ \ln(n)$$

(constant is known to be best possible, cf Devroye'86)

Depth of Key of Rank i

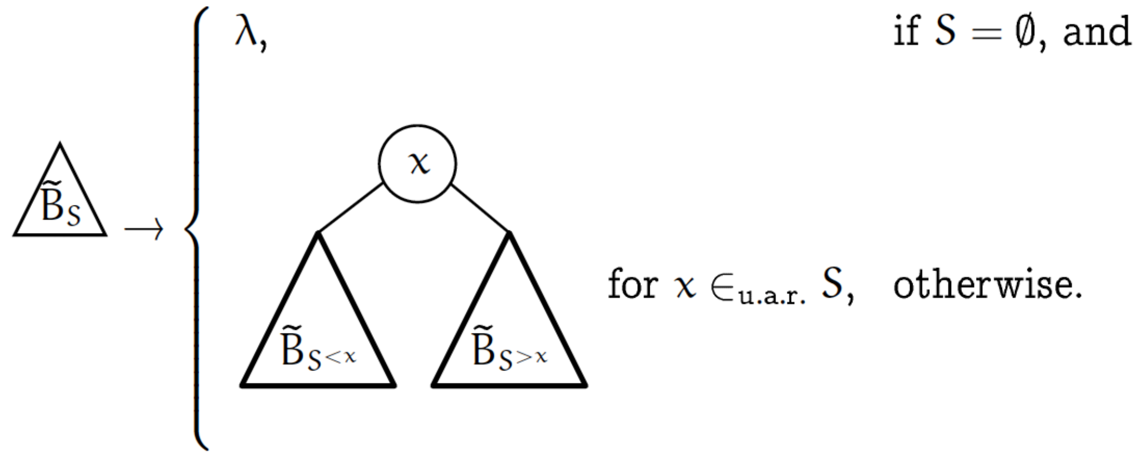
$D_n^{(i)}$:= depth of key of rank i; $d_{i,n} := E[D_n^{(i)}]$

$$A_i^j := [\text{node } j \text{ is ancestor of node } i]$$

$$= \begin{cases} 1, & \text{if node } j \text{ is ancestor of node } i, \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

$$\boxed{\mathbf{E}\left[D_n^{(i)}\right] = \sum_{j=1,\, j\neq i}^{n} \mathbf{E}\left[A_i^j\right]}$$

Random Search Trees



E[depth of smallest key] $= H_n - 1 = \ln n + O(1)$

E[sum of depths] $= 2(n+1)H_n - 4n = 2n \ln n + O(n)$

E[max depth] $\leq 4.311..\ln n$

E[depth of key of rank i] $= H_i + H_{n-i+1} - 2 \leq 2 \ln n$

Depth of Key of Rank i

$D_n^{(i)}$:= depth of key of rank i; $d_{i,n}$:= $E[D_n^{(i)}]$

$$A_i^j := [\text{node } j \text{ is ancestor of node } i]$$

$$= \begin{cases} 1, & \text{if node } j \text{ is ancestor of node } i, \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

$$\mathbf{E}\left[D_n^{(i)}\right] = \sum_{j=1,\, j\neq i}^{n} \mathbf{E}\left[A_i^j\right] = \sum_{j=1,\, j\neq i}^{n} \Pr\left[A_i^j = 1\right]:$$

Lemma 2.5. $i, j \in \mathbf{N}$. *In a random search tree for* $n \geq \max\{i, j\}$ *keys*

$$\Pr\left[A_i^j = 1\right] = \Pr[\textit{node } j \textit{ is ancestor of node } i] = \frac{1}{|i-j|+1}\;.$$

QuickSort

function quicksort(S)

if $S = \emptyset$ then return $()$;

else

$x \leftarrow_{\text{u.a.r.}} S$;

split S into $S^{<x}$, $\{x\}$, $S^{>x}$;

return quicksort$(S^{<x}) \circ (x) \circ$ quicksort$(S^{>x})$;

QuickSort

function quicksort(S)

if $S = \emptyset$ **then return** $()$;

else

$x \leftarrow_{\text{u.a.r.}} S$;

split S **into** $S^{<x}$, $\{x\}$, $S^{>x}$;

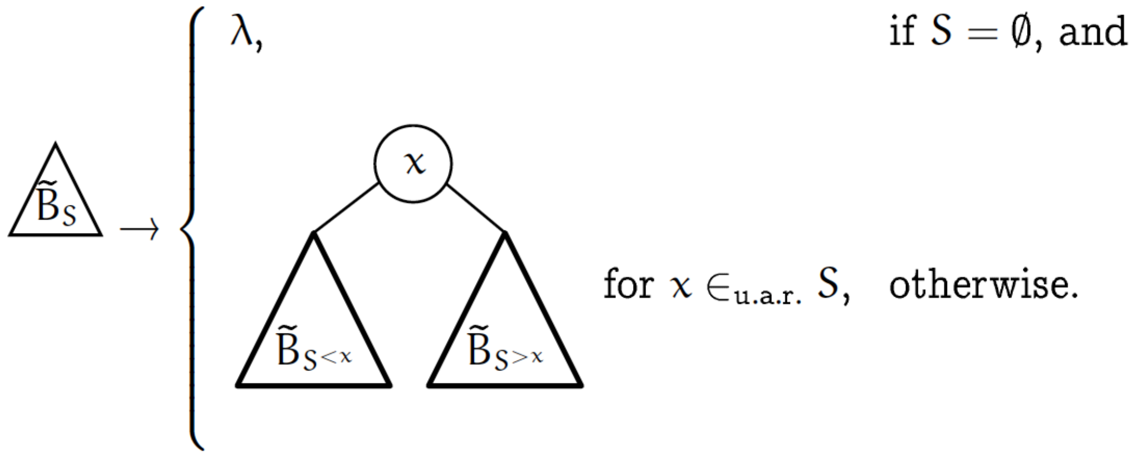
return quicksort$(S^{<x}) \circ (x) \circ$ quicksort$(S^{>x})$;

t_n := expected number of comparisons for n keys

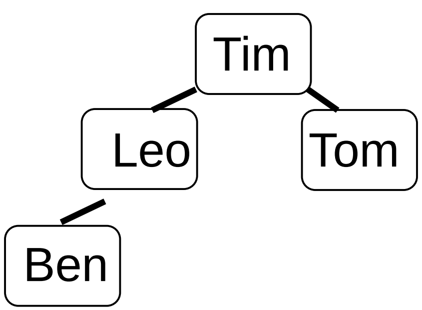
$$t_n = n - 1 + \sum_{i=1}^{n} (t_{i-1} + t_{n-i}) \frac{1}{n} = n - 1 + \frac{2}{n} \sum_{i=1}^{n} t_{i-1}$$

= E[sum of depths]

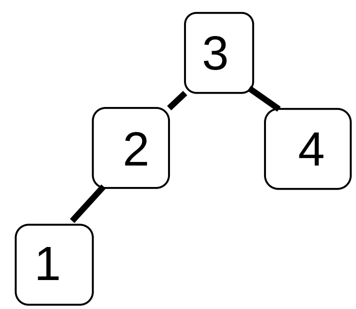
Random Search Trees



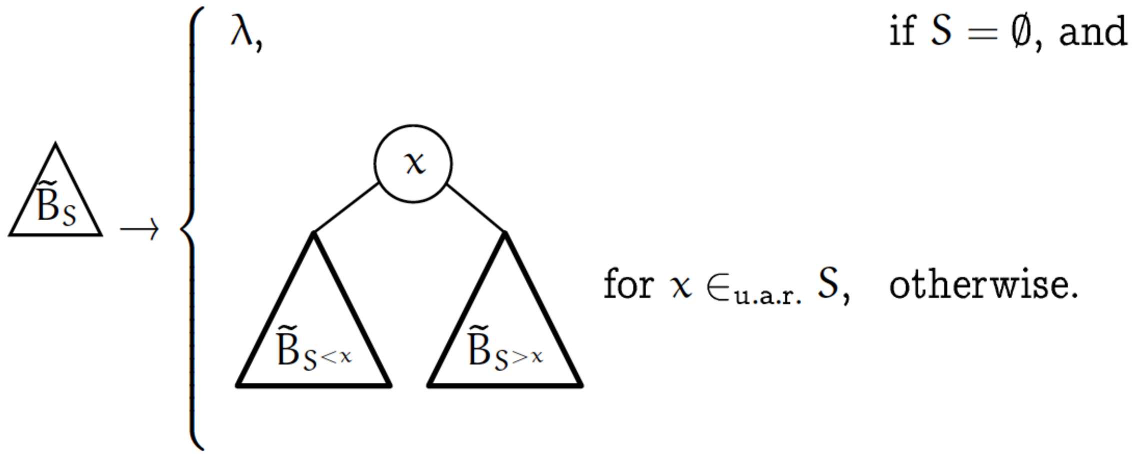
S={Tom,Ben,Tim,Leo}



S={1,2,3,4}



Random Search Trees



E[depth of smallest key] = $H_n - 1$ $= \ln n + O(1)$

E[sum of depths] = $2(n + 1) H_n - 4n = 2n \ln n + O(n)$

E[max depth] $\leq 4.311.. \ln n$

E[depth of key of rank i] = $H_i + H_{n-i+1} - 2$ $\leq 2 \ln n$

Treap

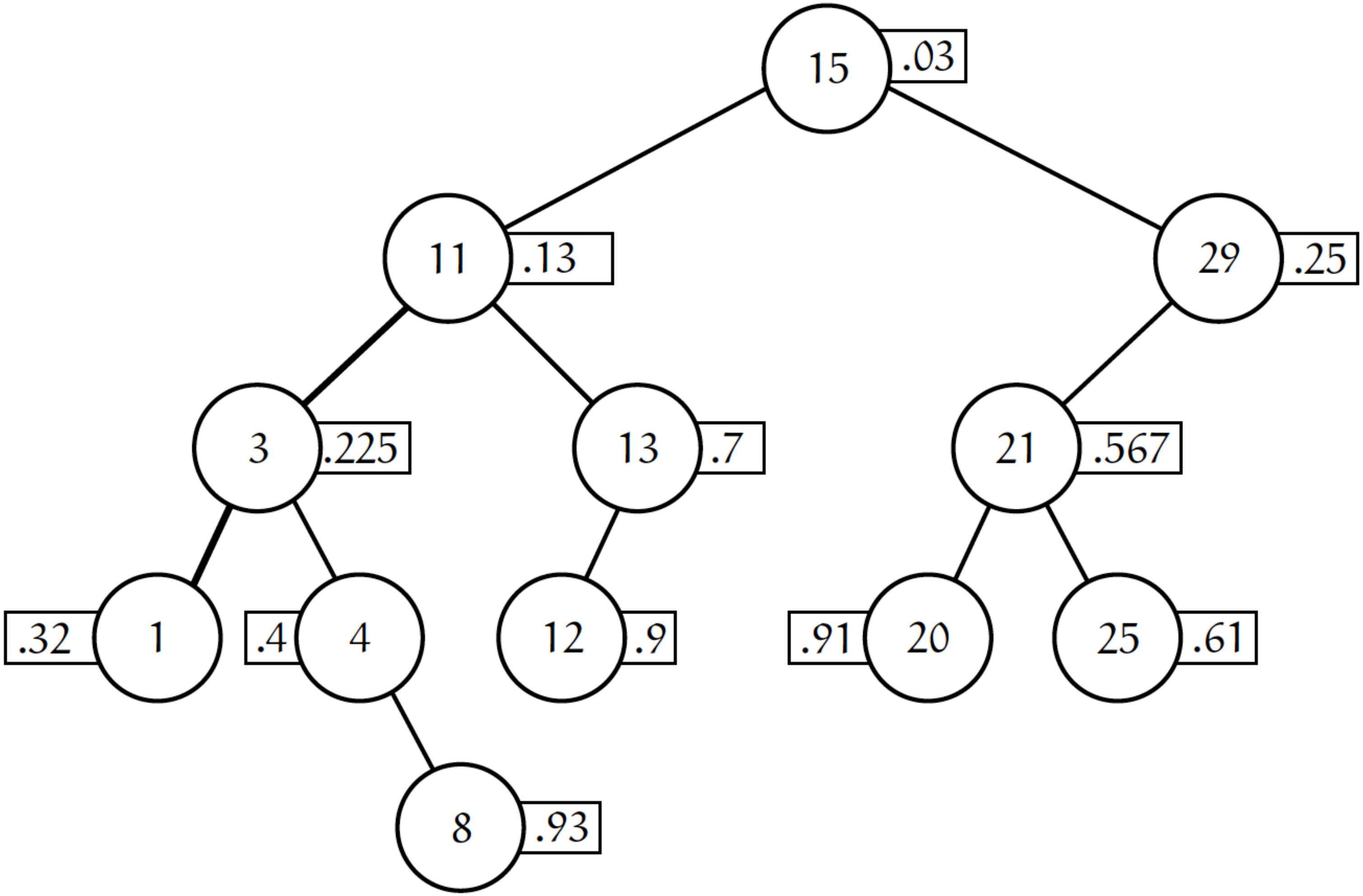
Treap = (search) tree & (min) heap

- defined for sets $Q \subseteq \mathbf{R} \times \mathbf{R}$

keys: key(x) priorities: prio(x)

- search tree wrt to keys & min heap wrt to priorities

Treap



Treap

Treap = (search) tree & (min) heap

- defined for sets $Q \subseteq \mathbf{R} \times \mathbf{R}$

  keys: key(x) priorities: prio(x)

- search tree wrt to keys & min heap wrt to priorities

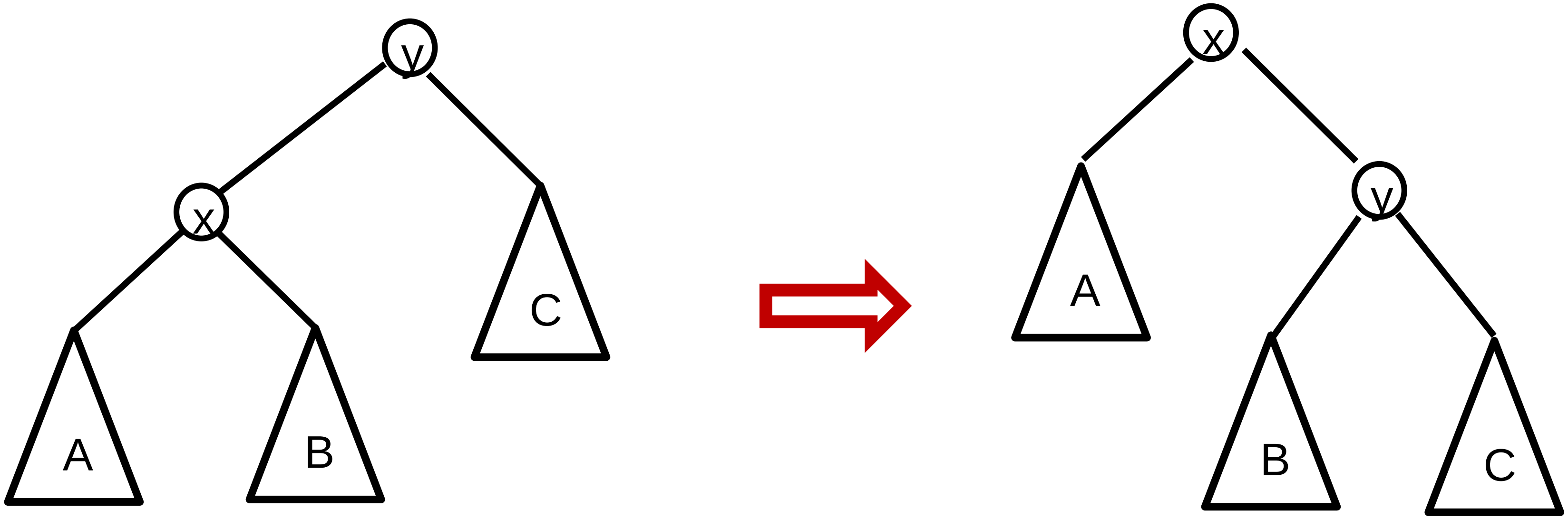
Idea: choose priorities uar from [0,1]

$\Rightarrow$ the constructed search tree will be a **random** search tree

Treap - Insert

Insert(x)

- insert x as a leaf according to rules of a search tree
- rotate x up until at correct position wrt to its priority

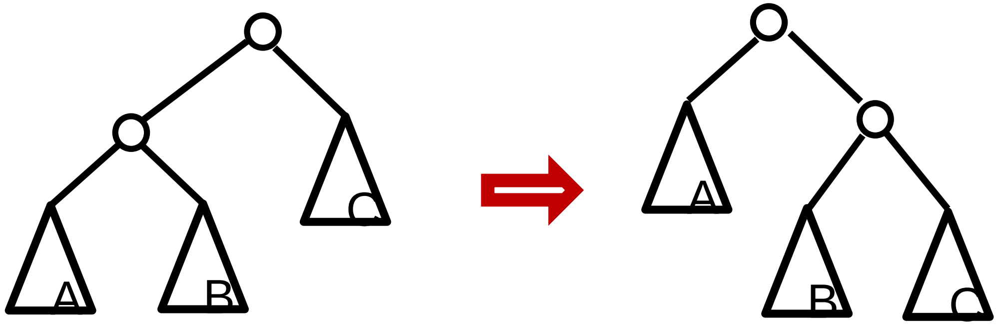


Treap: Insertions

Insert(x)

- insert x as a leaf according to rules of a search tree
- rotate x up until at correct position wrt to its priority

Lemma: $\forall\, T\ \forall\, x$: expected number of rotations < 2

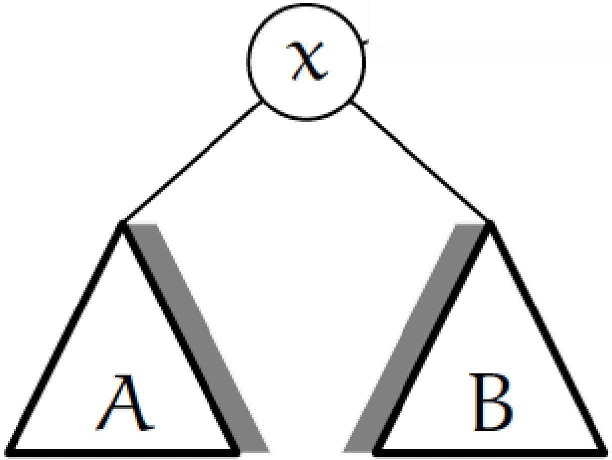


Spines

left (right) spine of a node x:

- sequence of nodes on the path from x to largest (smallest) node in left (right) subtree rooted at x (excluding x)

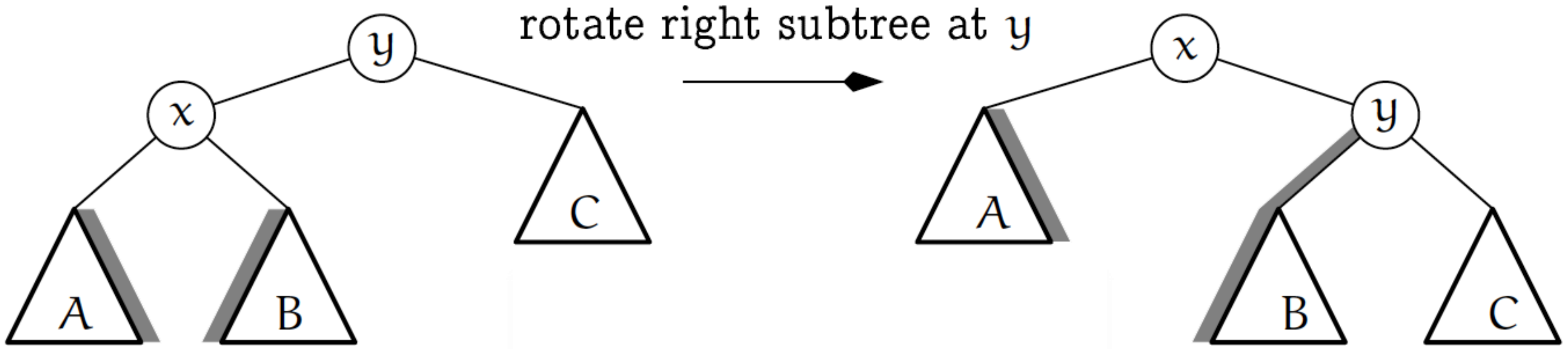
Note: The above definition is a shortcut of the definition in the lecture notes: there we define the left and right spine of a *tree*, as the path from the root to the smallest resp largest node in the tree. Then we associate with a node the two spines mentioned in the above definition, cf. picture below.



Spines

Lemma: ∀ T ∀ x:
Each rotation of x increases
length of left spine + length of right spine
by exactly one.

Proof:



Lemma: ∀ T ∀ x:
Each rotation of x increases
length of left spine + length of right spine
by exactly one.

It suffices to show:

Lemma: ∀ n: in a random search tree for [n] we have:
∀ j:
expected length of left spine = $1 - 1 / j$
expected length of right spine = $1 - 1 / (n-j+1)$

Spines

Lemma: ∀ n: in a random search tree for [n] we have:
∀ j:
expected length of left spine = 1 - 1 / j
expected length of right spine = 1 - 1 / (n-j+1)

Proof:

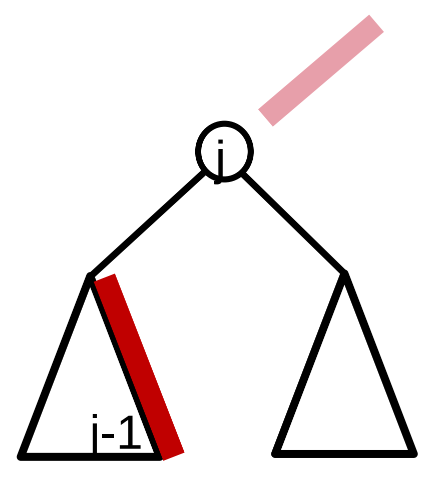
$$A_i^j := [\text{node } j \text{ is ancestor of node } i]$$

$$C_{i,j}^k := [\text{node } k \text{ is ancestor of nodes } i \text{ and } j]$$

We have:

length of left spine of j =

$$\sum_{k=1}^{j-1} \left(A_{j-1}^k - C_{j-1,j}^k \right)$$



Note: largest node in left subtree is either j-1 or left subtree is empty

Randomized Search Trees

Theorem 2.10. *In a* randomized search tree *(a treap with priorities independently and u.a.r. from* $[0, 1)$*) operations find, insert, delete, split and join can be performed in expected time* $O(\log n)$*,* n *the number of keys currently stored. The expected number of rotations necessary for an insertion or a deletion is always less than* 2*.*