Algorithms, Probability & Computing

Emo Welzl
Ueli Maurer
Angelika Steger
Peter Widmayer
Thomas Holenstein

Contents

- Random(ized) Search Trees
- Point Location
- Network Flows
- Minimum Cut
- Randomized Algebraic Algorithms
- Lovász Local Lemma
- Cryptographic Reductions
- Probabilistically Checkable Proofs

Formalities

web page:

http://www.ti.inf.ethz.ch/ew/courses/APC10/

exercise sessions (starting this week!):

Wed 13-15, Wed 15-17, Fri 14-16 (choose any one)

grade:

final exam (60%): Sessionsprüfung,

midterm exam (20%): November 15 (during class),

two special assignments (20%): tba

lecture notes: yes

Part I: Data Structures

Randomized Search Trees

Randomized Search Trees: Plan

- Definition

  ➔ define an appropriate probability space

- Study Properties

  ➔ learn methods and techniques how to do that

- Revisit: Quicksort & Quickselect

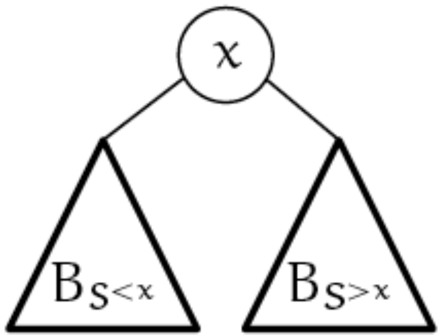
- A new data structure: Treaps

Recall: (Binary) Search Tree

S some (totally ordered) set of elements/keys

B_S search tree for S:

$$\rightarrow \begin{cases} \lambda, & \text{if } S = \emptyset, \text{ and} \end{cases}$$



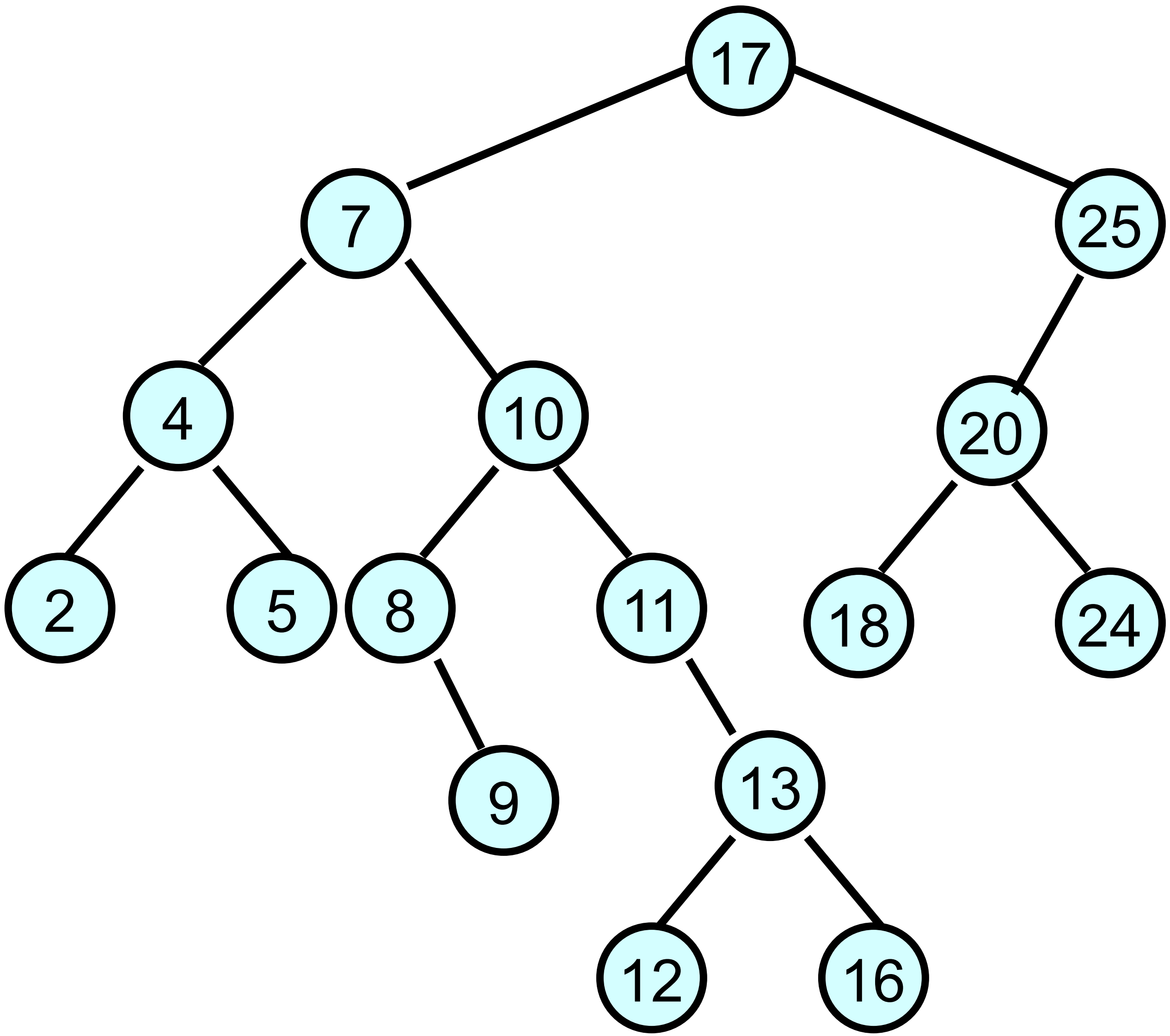
for $x \in S$, otherwise.

where

$$S^{<x} := \{a \in S \mid a < x\}$$

$$S^{>x} := \{a \in S \mid a > x\}$$

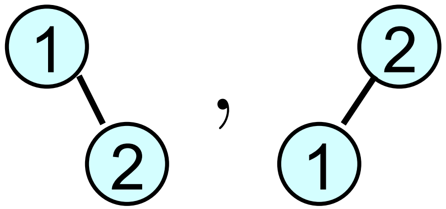
Example

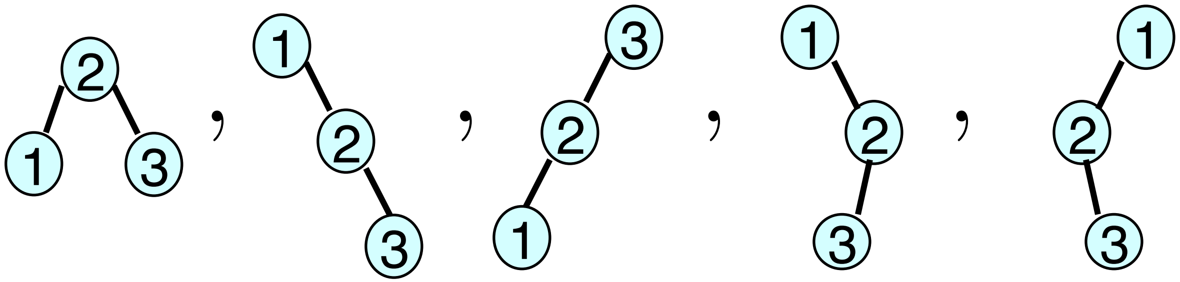


Examples (2)

$$\mathcal{B}_\emptyset = \{\lambda\}$$

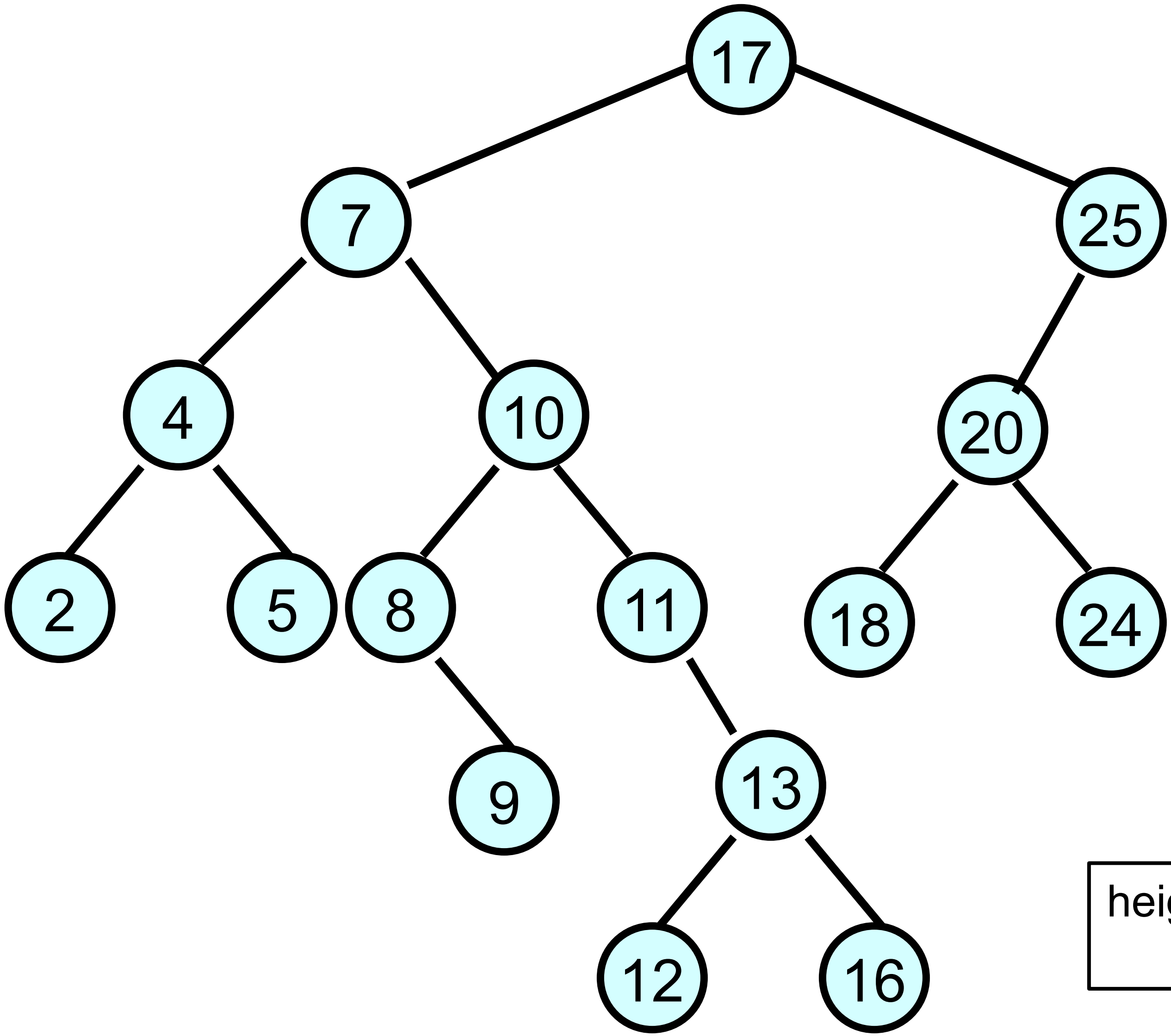
$\mathcal{B}_{\{1\}} = \{$  $\}$

$\mathcal{B}_{\{1,2\}} = \{$  $\}$

$\mathcal{B}_{\{1,2,3\}} = \{$  $\}$

Note: $$|\mathcal{B}_{[n]}| = \frac{1}{n+1}\binom{2n}{n}$$

Depth & Height



root: depth 0

← depth 1

← depth 2

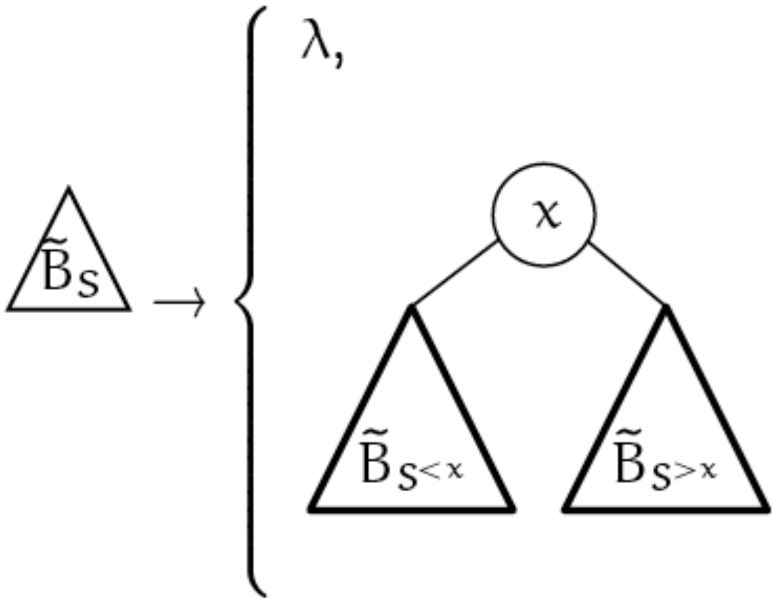
← depth 3

⋮

height := max depth of an element

Random Search Tree

$\widetilde{B}_S$ random search tree for S:

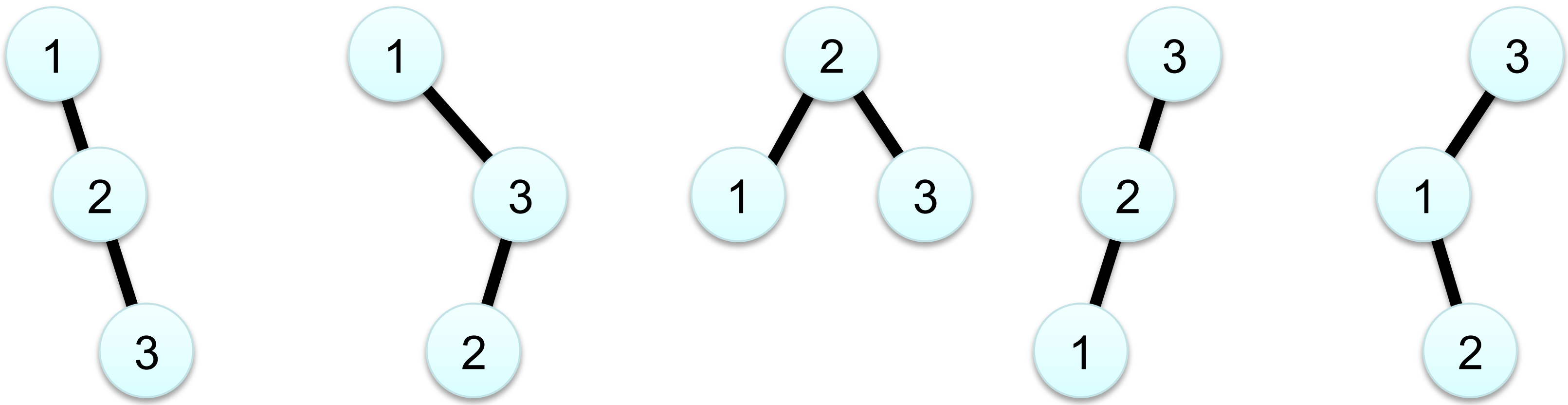


if $S = \emptyset$, and

for $x \in_{\text{u.a.r.}} S$, otherwise.

u.a.r := uniformly at random (= random with respect to uniform distribution)

Example: S={1,2,3}



1/6	**1/6**	**1/3**	**1/6**	**1/6**
= 1/3 * 1/2 * 1	**= 1/3 * 1/2 * 1**	**= 1/3 * 1 * 1**	**= 1/3 * 1/2 * 1**	**= 1/3 * 1/2 * 1**

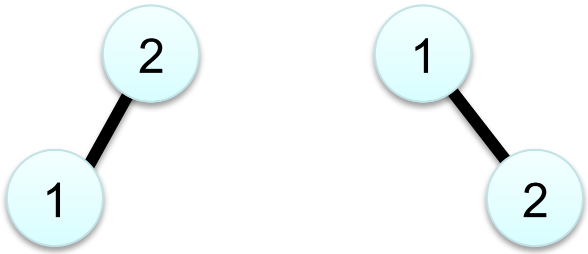
Note: This is **not** the uniform distribution on the set of all binary search trees for S = {1,2,3}

Expected Number of Leaves

$l_n :=$ $\mathbf{E}$[number of leaves in random search tree of size n]

$l_1 = 1$

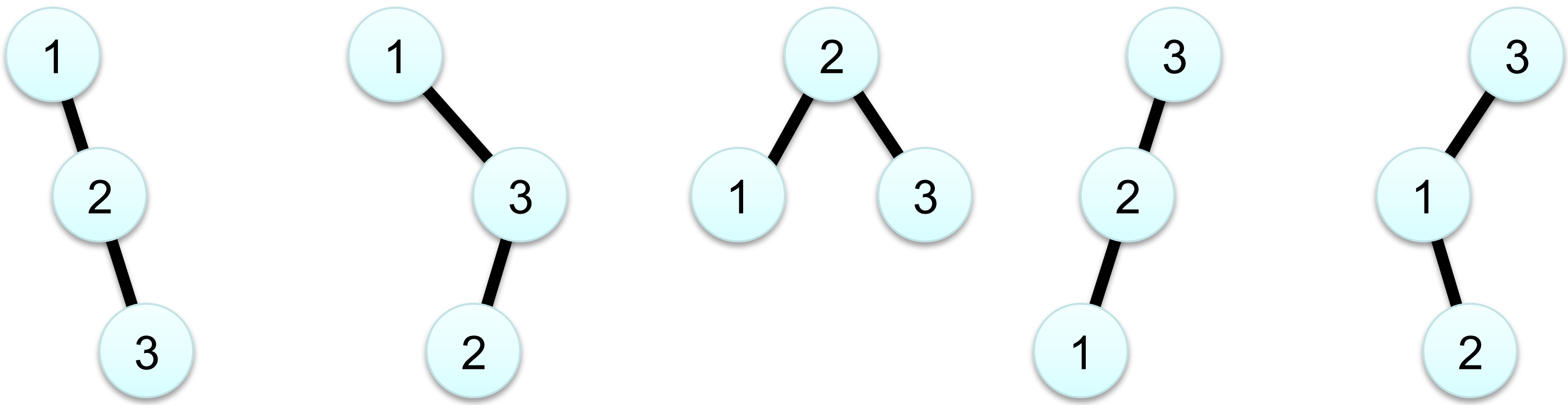
$l_2 = 1$



$l_3 = \ldots$

Expected Number of Leaves

$l_n :=$ $\mathbf{E}$[number of leaves in random search tree for S = [n]]



$l_3 =$ 1/6 * 1 + 1/6 * 1 + 1/3 * 2 + 1/6 * 1 + 1/6 * 1 = 4/3

Expected Number of Leaves

$$\begin{aligned}
\ell_n &= \mathbb{E}[\#\text{ leaves in random search tree for } S = [n]] \\
&= \sum_{i=1}^{n} \mathbb{E}[\#\text{ leaves ... for } S = [n] \mid \text{root} = i] \cdot \Pr[\text{root} = i] \\
&= \sum_{i=1}^{n} (\ell_{i-1} + \ell_{n-i}) \cdot \frac{1}{n} \\
&= \frac{2}{n} \sum_{i=0}^{n-1} \ell_i
\end{aligned}$$

Expected Number of Leaves

Hence, for n ≥ 3:

$$\frac{2}{n}\ell_n = \sum_{i=0}^{n-1} \ell_i, \qquad \text{and}$$

$$\frac{2}{n-1}\ell_{n-1} = \sum_{i=0}^{n-2} \ell_i$$

Subtract both equations: $\frac{2}{n}\ell_n - \frac{2}{n-1}\ell_{n-1} = \ell_{n-1}$ (for n ≥ 3)

I.e. $\frac{2}{n}\ell_n = \frac{n+1}{n-1}\ell_{n-1} = \frac{n+1}{n-1}\frac{n}{n-2}\ell_{n-2} = \frac{n+1}{n-1}\frac{n}{n-2}\cdots\frac{3}{2}\underbrace{\ell_2}_{=1} = \frac{(n+1)\cdot n}{3\cdot 2}$

Hence, $\ell_n = \frac{n+1}{3}$ for all $n \geq 3$

Plan

Properties of Random Search Trees (Sec. 1.2 – 1.4)

- number of leaves (warmup)
- depth of keys:
  - sum of all depths
  - depth of smallest/largest key
  - depth of individual keys (ith smallest, for all $1 \leq i \leq n$)

Notations

rank of $x \in S$:

$$rk(x) = rx_S(x) := 1 + |\{y \in S : y < x\}|$$

$D_n^{(i)}$:= random variable for the depth of the key of rank i

$D_n := D_n^{(1)}$ (= depth of smallest key)

$X_n := \sum_{i=1}^{n} D_n^{(i)}$ (= *overall* depth)

General Scheme

$Z_n :=$ random variable defined for a random search tree of size n

$$\mathbb{E}[Z_n] = \sum_{i=1}^{n} \mathbb{E}[Z_n \mid \text{rk}(\text{root}) = i] \cdot \Pr[\text{rk}(\text{root}) = i]$$

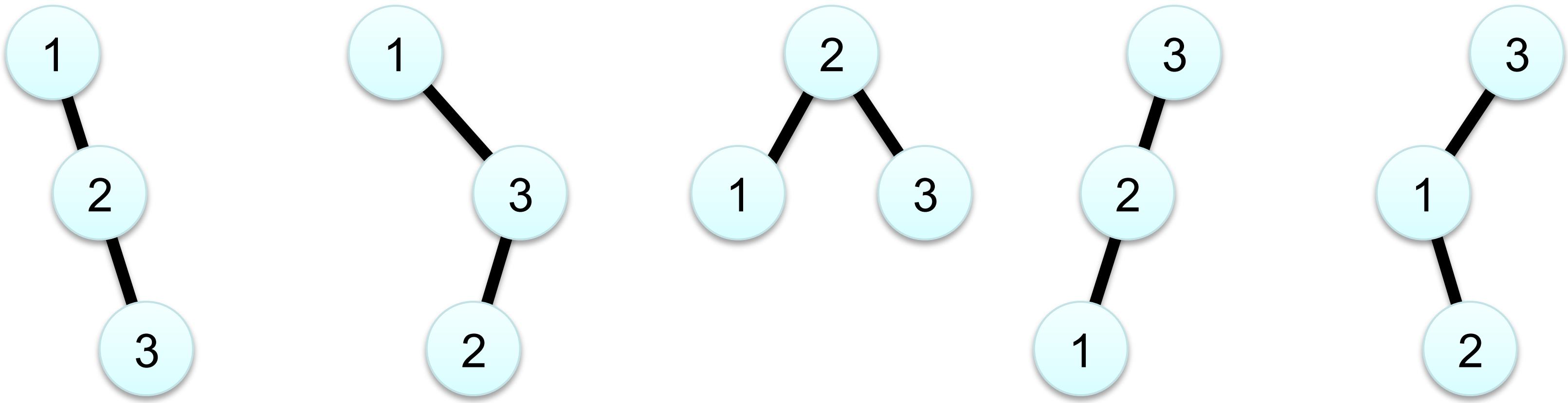
$= \dfrac{1}{n}$

write as function of $\mathbb{E}[Z_{i-1}]$ and $\mathbb{E}[Z_{n-i}]$

Solve recurrence relation; useful trick: subtract equations for n and $n-1$

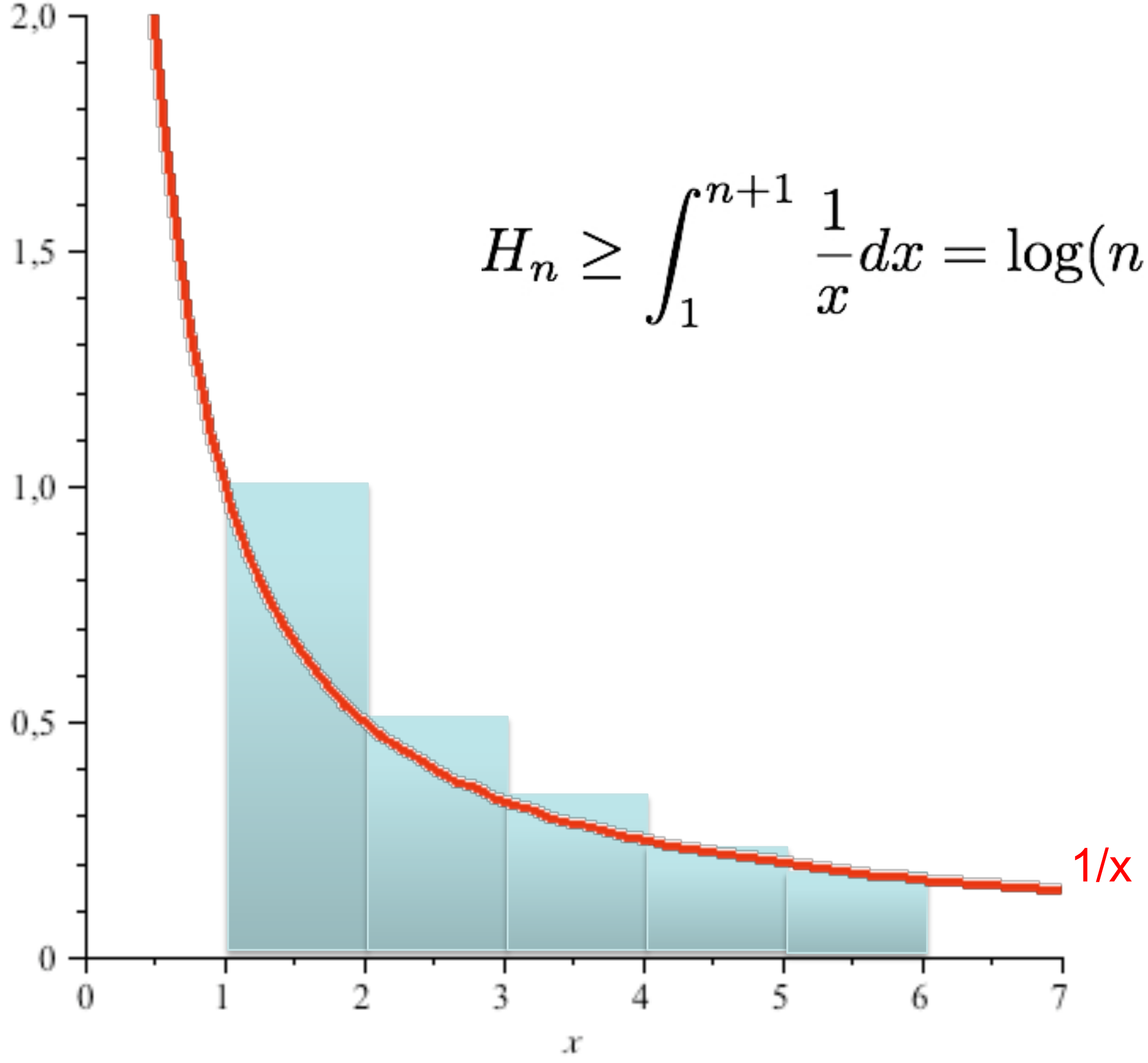
Expected depth of smallest key

$d_n = \mathbb{E}[D_n]$: $d_1 = 0$, $d_2 = 1/2$



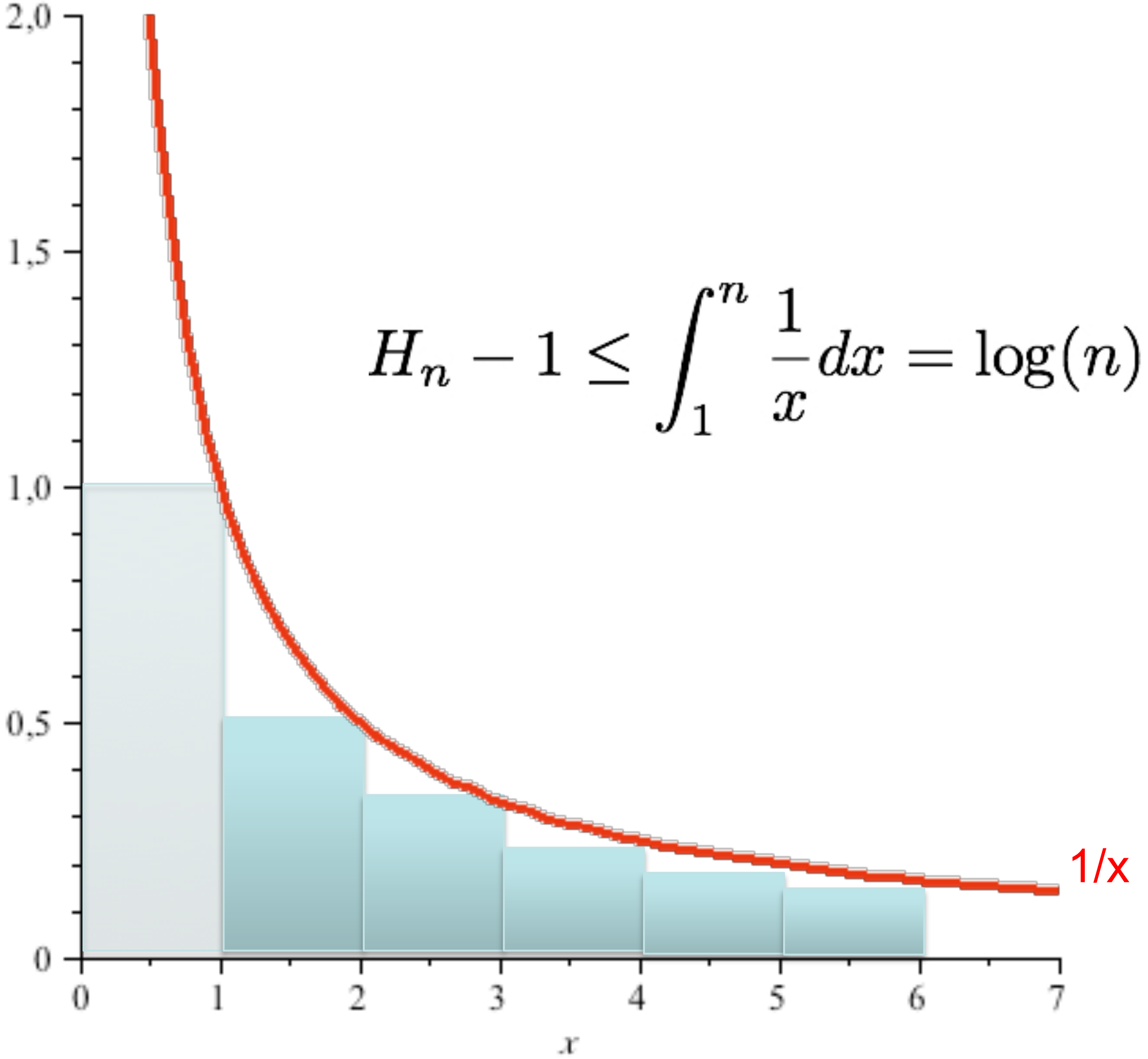
d_3 = **1/6 * 0 + 1/6 * 0 + 1/3 * 1 + 1/6 * 2 + 1/6 * 1 = 5/6**

Bounds for harmonic number



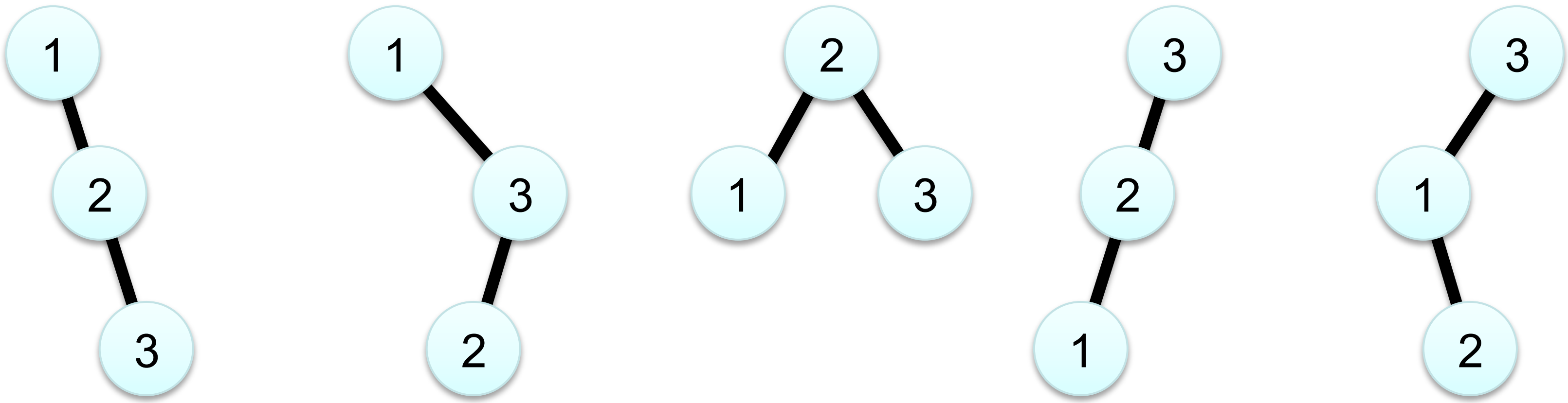
$$H_n \geq \int_1^{n+1} \frac{1}{x} dx = \log(n+1)$$

Bounds for harmonic number



Expected overall depth

$$X_n := \sum_{i=1}^{n} D_n^{(i)}, \qquad x_n := \mathbb{E}[X_n]$$



x_3 = **1/6 * (0+1+2) + 1/6 * (0+2+1) + 1/3 * (1+0+1) + 1/6 * (2+1+0) + 1/6 * (1+2+0)**

= 8/3

General Scheme

$Z_n :=$ random variable defined for a random search tree of size n

$$\mathbb{E}[Z_n] = \sum_{i=1}^{n} \mathbb{E}[Z_n \mid \text{rk(root)} = i] \cdot \Pr[\text{rk(root)} = i]$$

$$= \frac{1}{n}$$

write as function of $\mathbb{E}[Z_{i-1}]$ and $\mathbb{E}[Z_{n-i}]$

Solve recurrence relation; useful trick: subtract equations for n and $n-1$

Results

$\mathbb{E}[\text{depth of smallest key}]$

$$= \quad \mathbb{E}[D_n] = H_n - 1 = \ln n + \mathcal{O}(1)$$

$\mathbb{E}[\text{overall depth}]$

$$= \quad \mathbb{E}[\sum_{i=1}^{n} D_n^{(i)}] = 2(n+1)H_n - 4n = 2n\ln(n) + \mathcal{O}(n)$$

$\mathbb{E}[\text{height}]$

$$= \quad \mathbb{E}[\max_{1\le i\le n} D_n^{(i)}] \le c\ln(n), \quad \text{where } c = 4.311.. \text{ is the unique}$$

$$\text{solution greater 2 of } (2e/c)^c = e$$

Reed'03, Drmota'03: $\quad \mathbb{E}[\max_{1\le i\le n} D_n^{(i)}] = c\ln(n) - \frac{3c}{2(c-1)}\ln\ln(n) + \mathcal{O}(1)$

Recap

Quicksort(S): For a set S of size n:

Expected number of comparisons between elements of S
= Expected overall depth in a random search tree for S
= $2n \ln(n) + O(n)$

Quickselect(S,k): For a set S of size n and any $1 \le k \le n$

Expected number of comparisons between elements of S
$\le 4n$

Note:	best deterministic alg:	$2.95n$	[Dor, Zwick 1999]
	lower bound:	$(2+\varepsilon)n$	[Dor, Zwick 2001]
	best randomized alg:	$3/2\ n$	[folklore]

Recap (2)

(Deterministic) **Quicksort(S):**

for *random* inputs:

Expected number of camparisons ≈ 2 n ln(n)

(Randomized) **Quicksort(S):**

for *all* inputs:

Expected number of camparisons ≈ 2 n ln(n)

Today: Treaps

(Deterministic) **SearchTree(S):**

for *random* inputs:

we can bound expected height, depth, etc

(Randomized) **Treap(S):**

for *all* inputs:

we can bound expected height, depth, etc (as above)

Search Trees & Heaps

(Binary) **Search Tree:**

for every node v: keys in left subtree $<$ key(v) $<$ keys in right subtree

(Binary) **Heap:**

for every node v: key(v) $<$ keys in (both) subtrees

Treap

Treap := Search Tree + Heap

More precisely: every node has *two* keys:

for every node v:

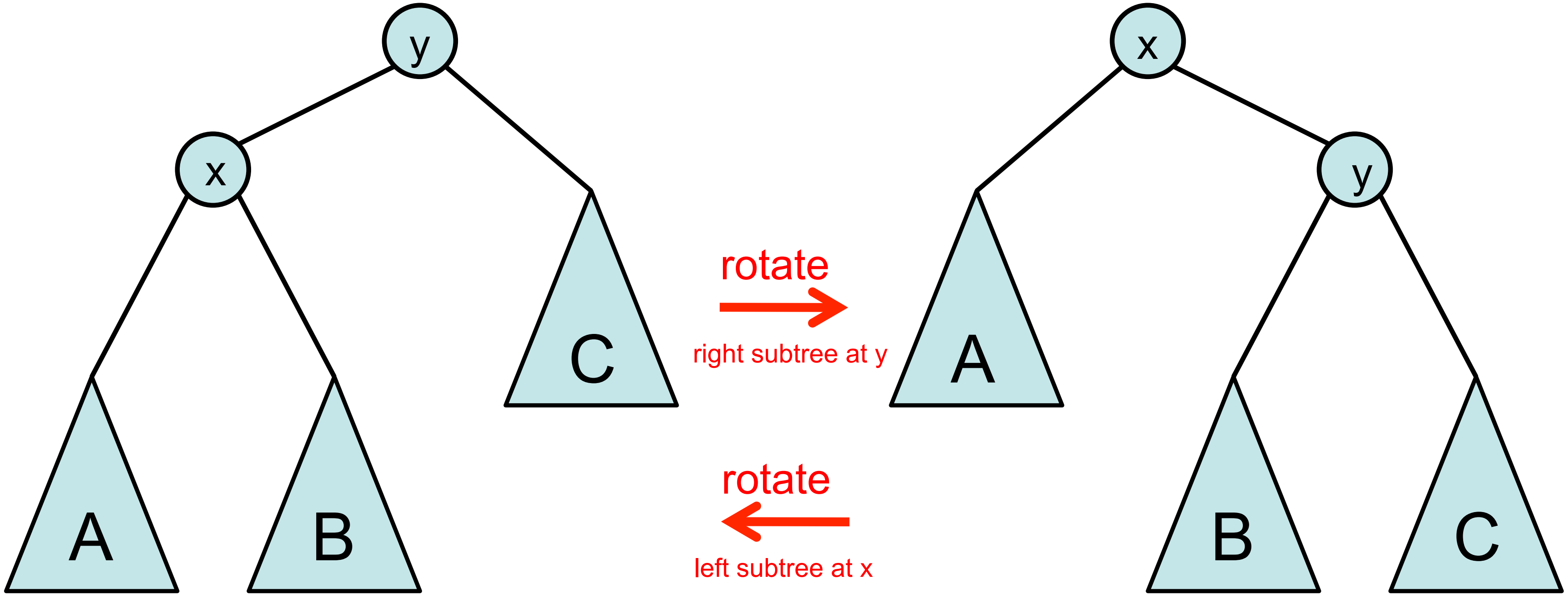
- 1st keys in left subtree < 1st key of v < 1st keys in right subtree
- 2nd key of v < 2nd keys in (both) subtrees

1st key: the real key ...

2nd key: random values drawn u.a.r from [0,1)

Observe: A treap is a *random* search tree - for *all* inputs

Rotations



Note:

1st key: A < x < B < y < C ✓

2nd key: before: everything ok except edge {x,y}
now: everything ok ✓

Insertion

Insert element v into a treap T of size n:

- choose 2nd key of v uar from [0,1)
- insert v into search tree (as a leaf!)
- rotate v upwards until heap property is satisfied

Runtime analysis:

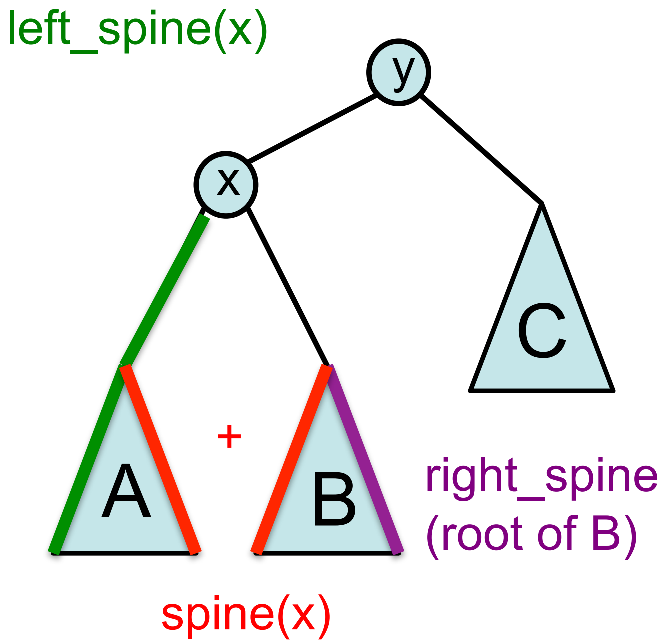
- time to insert v in search tree: O(ln n), as a random search tree has height O(ln n)
- *we will now show:* $\mathbb{E}[\text{ \# of rotations}] \leq 2$

Expected number of rotations

Definitions:

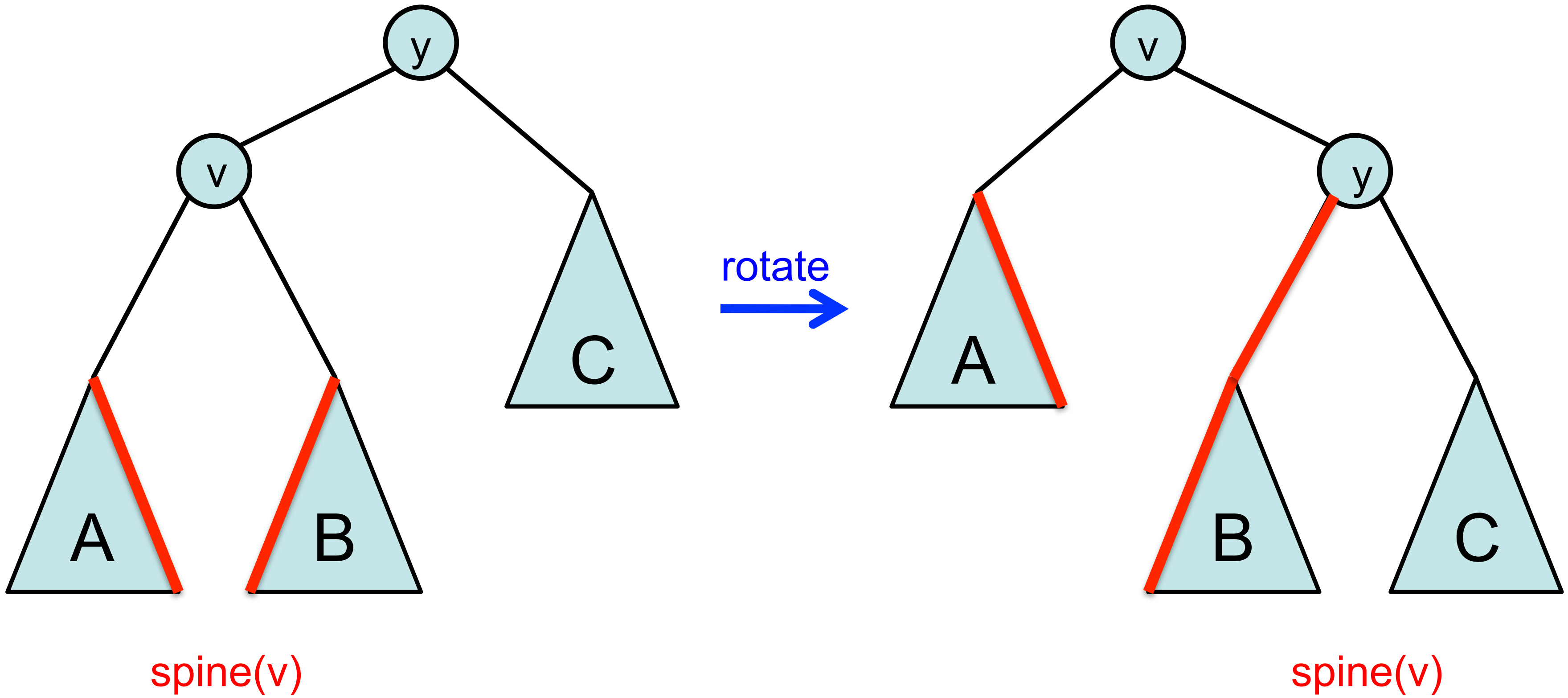
- left_spine(v) := number of nodes on path from v to smallest element in subtree rooted at v
- right_spine(v) := number of nodes on path from v to largest element in subtree rooted at v

- spine(v) := left_spine(right child of v) + right_spine(left child of v)



Lemma:
After inserting v we have: spine(v) = # of performed rotations

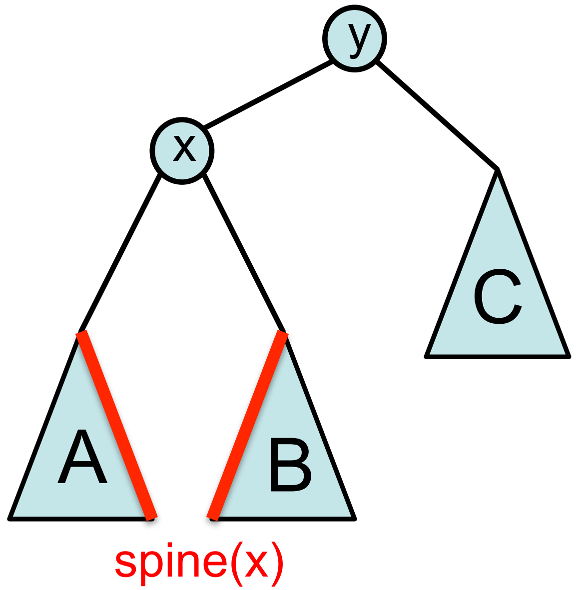
Proof of Lemma



Hence: every rotation increases spine(v) by exactly one

Expected number of rotations

spine(v) := left_spine(root of right subtree of v) +
right_spine(root of left subtree of v)



Lemma: After inserting v we have:

spine(v) = # of performed rotations

Lemma: In a random search tree of size n we have for all nodes v:

$$\mathbb{E}[\,\mathrm{spine}(v)\,] = (1 - \frac{1}{\mathrm{rk}(v)}) + (1 - \frac{1}{n - \mathrm{rk}(v) + 1})$$