Verified Analysis of Random Binary Tree Structures

Manuel Eberl, Max W. Haslbeck, Tobias Nipkow

Technische Universität München

6 June 2018

Isabelle

Contributions

- Quicksort

Contributions

- Quicksort
- Random Binary Search Trees

Contributions

- Quicksort
- Random Binary Search Trees
- Treaps

Discrete distributions in Isabelle/HOL

Probability distributions with countable support are modelled as *probability mass function* [Hölzl 2011]:

Probability distributions with countable support are modelled as *probability mass function* [Hölzl 2011]:

- Type α *pmf* represents a probability distribution of values of type α

Probability distributions with countable support are modelled as *probability mass function* [Hölzl 2011]:

- Type α *pmf* represents a probability distribution of values of type α
- Isomorphic to the set of functions $f : \alpha \to \mathbb{R}$ with $f(x) \geq 0$ and $\sum_{x::\alpha} f(x) = 1$

Probability distributions with countable support are modelled as *probability mass function* [Hölzl 2011]:

- Type α *pmf* represents a probability distribution of values of type α
- Isomorphic to the set of functions $f : \alpha \to \mathbb{R}$ with $f(x) \geq 0$ and $\sum_{x::\alpha} f(x) = 1$
- Giry monad allows composing PMFs: $\mathbf{do}\ \{x \leftarrow A;\ y \leftarrow B\ x;\ \mathbf{return}\ (f\ x\ y)\}$

Quicksort

Deterministic quicksort with first element as pivot:

$$\mathsf{qs} :: \alpha\ \mathsf{list} \to \alpha\ \mathsf{list}$$

Deterministic quicksort with first element as pivot:

$$\mathsf{qs} :: \alpha\ \mathsf{list} \to \alpha\ \mathsf{list} \times \mathbb{N}$$

Deterministic quicksort with first element as pivot:

$$\mathsf{qs} :: \alpha\ \mathsf{list} \to \alpha\ \mathsf{list} \times \mathbb{N}$$

Quicksort with random pivot:

$$\mathsf{rqs} :: \alpha\ \mathsf{list} \to (\alpha\ \mathsf{list} \times \mathbb{N})\ \mathsf{pmf}$$

Deterministic quicksort with first element as pivot:

$$\mathsf{qs} :: \alpha\ \mathsf{list} \to \alpha\ \mathsf{list} \times \mathbb{N}$$

Quicksort with random pivot:

$$\mathsf{rqs} :: \alpha\ \mathsf{list} \to (\alpha\ \mathsf{list} \times \mathbb{N})\ \mathsf{pmf}$$

Average-case of det. quicksort:

$$\mathsf{avqs}\ \mathit{xs} = \mathbf{do}\ \{\mathit{xs}' \leftarrow \mathsf{rperm}\ \mathit{xs};\ \mathbf{return}\ (\mathsf{qs}\ \mathit{xs}')\}$$

Deterministic quicksort with first element as pivot:

$$\mathsf{qs} :: \alpha\ \mathsf{list} \to \alpha\ \mathsf{list} \times \mathbb{N}$$

Quicksort with random pivot:

$$\mathsf{rqs} :: \alpha\ \mathsf{list} \to (\alpha\ \mathsf{list} \times \mathbb{N})\ \mathsf{pmf}$$

Average-case of det. quicksort:

$$\mathsf{avqs}\ xs = \mathbf{do}\ \{xs' \leftarrow \mathsf{rperm}\ xs;\ \mathbf{return}\ (\mathsf{qs}\ xs')\}$$

Theorem

- $\mathrm{E}[\mathsf{snd}(\mathsf{rqs}\ \mathsf{xs})] = 2(n+1)H_n - 4n \sim 2n \ln n$

Deterministic quicksort with first element as pivot:

$$\mathsf{qs} :: \alpha\ \mathsf{list} \to \alpha\ \mathsf{list} \times \mathbb{N}$$

Quicksort with random pivot:

$$\mathsf{rqs} :: \alpha\ \mathsf{list} \to (\alpha\ \mathsf{list} \times \mathbb{N})\ \mathsf{pmf}$$

Average-case of det. quicksort:

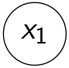
$$\mathsf{avqs}\ xs = \mathbf{do}\ \{xs' \leftarrow \mathsf{rperm}\ xs;\ \mathbf{return}\ (\mathsf{qs}\ xs')\}$$

Theorem

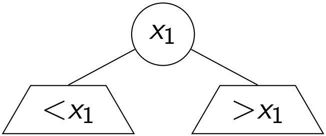
- $E[\mathsf{snd}(\mathsf{rqs\ xs})] = 2(n+1)H_n - 4n \sim 2n \ln n$
- $\mathsf{avqs} = \mathsf{rqs}$

Random Binary Search Trees

What happens when we insert distinct elements $x_1, \ldots, x_n$ into an empty BST?



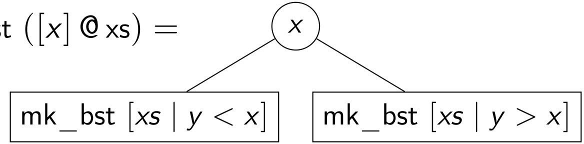
What happens when we insert distinct elements $x_1, \ldots, x_n$ into an empty BST?



mk_bst $[\,] =$ $\bullet$

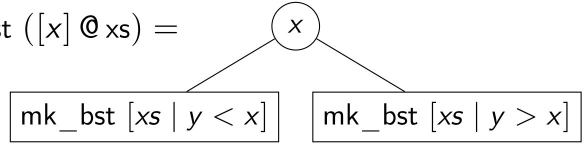
$$\text{mk\_bst } [\,] = \bullet$$

$$\text{mk\_bst } ([x] \mathbin{@} \text{xs}) =$$



$\text{mk\_bst}\ [\,] = \bullet$

$\text{mk\_bst}\ ([x] \mathbin{@} \text{xs}) =$

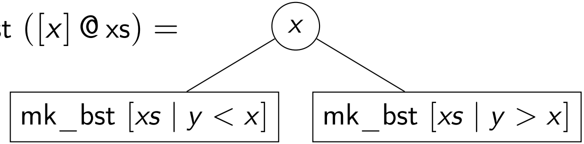


Let us now add elements from a set A in random order:

$$\text{rbst}\ A := \textbf{do}\ \{xs \leftarrow \text{rperm}\ A;\ \textbf{return}\ (\text{mk\_bst}\ xs)\}$$

$\text{mk\_bst}\ [\,] = \bullet$

$\text{mk\_bst}\ ([x] \mathbin{@} \text{xs}) =$

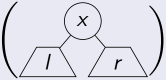


Let us now add elements from a set A in random order:

$$\text{rbst}\ A := \textbf{do}\ \{xs \leftarrow \text{rperm}\ A;\ \textbf{return}\ (\text{mk\_bst}\ xs)\}$$

Lemma

$$\begin{aligned}\text{rbst}\ A = \textbf{do}\ & x \leftarrow \text{uniform}\ A \\ & l \leftarrow \text{rbst}\ \{y \in A \mid y < x\} \\ & r \leftarrow \text{rbst}\ \{y \in A \mid y > x\} \\ & \textbf{return}\end{aligned}$$



Internal Path Length:
Sum of length of all paths from root to a node

Internal Path Length:

Sum of length of all paths from root to a node

$\Longrightarrow$ Average access time $= \frac{1}{n}\text{IPL}$

Internal Path Length:

Sum of length of all paths from root to a node

$\implies$ Average access time $= \frac{1}{n}$IPL

What is the IPL for a random BST?

Internal Path Length:

Sum of length of all paths from root to a node

$\Longrightarrow$ Average access time $= \frac{1}{n}\mathrm{IPL}$

What is the IPL for a random BST?

Exactly the same recurrence as for cost of *rqs*, thus:

$$\textbf{do}\ \{t \leftarrow \mathrm{rbst}\ A;\ \textbf{return}\ (\mathrm{ipl}\ t)\} = \mathrm{rqs\_cost}\ |A|$$

Internal Path Length:

Sum of length of all paths from root to a node

$\Longrightarrow$ Average access time $= \frac{1}{n}\mathrm{IPL}$

What is the IPL for a random BST?

Exactly the same recurrence as for cost of *rqs*, thus:

$$\textbf{do } \{t \leftarrow \mathrm{rbst}\ A;\ \textbf{return}\ (\mathrm{ipl}\ t)\} = \mathrm{rqs\_cost}\ |A|$$

$\Longrightarrow$ Hence average access time is $\sim 2\ln n$.

The height is somewhat more difficult: [CLRS]

The height is somewhat more difficult: [CLRS]

$$\text{eheight\_rbst } A := \textbf{do } \{t \leftarrow \text{rbst } A; \textbf{return } 2^{\text{height } t - 1}\}$$

The height is somewhat more difficult: [CLRS]

$$\text{eheight\_rbst}\ A := \textbf{do}\ \{t \leftarrow \text{rbst}\ A; \textbf{return}\ 2^{\text{height}\ t-1}\}$$

Theorem

- $$\begin{aligned} \text{eheight\_rbst}\ A = \textbf{do}\ & x \leftarrow \text{uniform}\ A \\ & l \leftarrow \text{eheight\_rbst}\ \{y \in A \mid y < x\} \\ & r \leftarrow \text{eheight\_rbst}\ \{y \in A \mid y > x\} \\ & \textbf{return}\ (2 \cdot \max\ l\ r) \end{aligned}$$

The height is somewhat more difficult: [CLRS]

$$\text{eheight\_rbst } A := \textbf{do } \{t \leftarrow \text{rbst } A; \textbf{return } 2^{\text{height } t - 1}\}$$

Theorem

- $\text{eheight\_rbst } A = \textbf{do } x \leftarrow \text{uniform } A$
  $l \leftarrow \text{eheight\_rbst } \{y \in A \mid y < x\}$
  $r \leftarrow \text{eheight\_rbst } \{y \in A \mid y > x\}$
  $\textbf{return } (2 \cdot \max\ l\ r)$
- $\text{E}[\text{eheight\_rbst}(n)] \leq \frac{4}{n} \cdot \sum_{i=0}^{n-1} \text{E}[\text{eheight\_rbst}(i)]$

The height is somewhat more difficult: [CLRS]

$$\text{eheight\_rbst } A := \textbf{do } \{t \leftarrow \text{rbst } A; \textbf{return } 2^{\text{height } t - 1}\}$$

Theorem

- $\text{eheight\_rbst } A = \textbf{do } x \leftarrow \text{uniform } A$
  $l \leftarrow \text{eheight\_rbst } \{y \in A \mid y < x\}$
  $r \leftarrow \text{eheight\_rbst } \{y \in A \mid y > x\}$
  $\textbf{return } (2 \cdot \max\ l\ r)$
- $\mathrm{E}[\text{eheight\_rbst}(n)] \leq \frac{4}{n} \cdot \sum_{i=0}^{n-1} \mathrm{E}[\text{eheight\_rbst}(i)]$
- $\mathrm{E}[\text{eheight\_rbst}(n)] \leq \frac{1}{4}\binom{n+3}{3}$

The height is somewhat more difficult: [CLRS]

$$\text{eheight\_rbst } A := \textbf{do } \{t \leftarrow \text{rbst } A; \textbf{return } 2^{\text{height } t - 1}\}$$

Theorem

- $\text{eheight\_rbst } A = \textbf{do } x \leftarrow \text{uniform } A$
  $\quad l \leftarrow \text{eheight\_rbst } \{y \in A \mid y < x\}$
  $\quad r \leftarrow \text{eheight\_rbst } \{y \in A \mid y > x\}$
  $\quad \textbf{return } (2 \cdot \max\ l\ r)$
- $\mathrm{E}[\text{eheight\_rbst}(n)] \leq \frac{4}{n} \cdot \sum_{i=0}^{n-1} \mathrm{E}[\text{eheight\_rbst}(i)]$
- $\mathrm{E}[\text{eheight\_rbst}(n)] \leq \frac{1}{4}\binom{n+3}{3}$
- $\mathrm{E}[\text{height}(\text{rbst}(n))] \leq \log_2 \binom{n+3}{3} - 1$

The height is somewhat more difficult: [CLRS]

$$\text{eheight\_rbst } A := \textbf{do } \{t \leftarrow \text{rbst } A; \textbf{return } 2^{\text{height } t - 1}\}$$

Theorem

- $\text{eheight\_rbst } A = \textbf{do } x \leftarrow \text{uniform } A$
  $\quad l \leftarrow \text{eheight\_rbst } \{y \in A \mid y < x\}$
  $\quad r \leftarrow \text{eheight\_rbst } \{y \in A \mid y > x\}$
  $\quad \textbf{return } (2 \cdot \max\ l\ r)$
- $\mathrm{E}[\text{eheight\_rbst}(n)] \leq \frac{4}{n} \cdot \sum_{i=0}^{n-1} \mathrm{E}[\text{eheight\_rbst}(i)]$
- $\mathrm{E}[\text{eheight\_rbst}(n)] \leq \frac{1}{4}\binom{n+3}{3}$
- $\mathrm{E}[\text{height}(\text{rbst}(n))] \leq \log_2 \binom{n+3}{3} - 1 \sim 3 \log_2 n$

The height is somewhat more difficult: [CLRS]

$$\text{eheight\_rbst } A := \textbf{do } \{t \leftarrow \text{rbst } A; \textbf{return } 2^{\text{height } t - 1}\}$$

Theorem

- $$\begin{aligned}\text{eheight\_rbst } A = \textbf{do } & x \leftarrow \text{uniform } A \\ & l \leftarrow \text{eheight\_rbst } \{y \in A \mid y < x\} \\ & r \leftarrow \text{eheight\_rbst } \{y \in A \mid y > x\} \\ & \textbf{return } (2 \cdot \max\ l\ r)\end{aligned}$$
- $\mathrm{E}[\text{eheight\_rbst}(n)] \leq \frac{4}{n} \cdot \sum_{i=0}^{n-1} \mathrm{E}[\text{eheight\_rbst}(i)]$
- $\mathrm{E}[\text{eheight\_rbst}(n)] \leq \frac{1}{4}\binom{n+3}{3}$
- $\mathrm{E}[\text{height}(\text{rbst}(n))] \leq \log_2 \binom{n+3}{3} - 1 \sim 3 \log_2 n$

The actual behaviour is $\approx 2.988 \log_2 n$ [Reed 2003].

Treaps

Random BSTs have nice logarithmic height/IPL

Random BSTs have nice logarithmic height/IPL

But BSTs can degenerate if inputs are *not* in random order

Random BSTs have nice logarithmic height/IPL

But BSTs can degenerate if inputs are *not* in random order

A Nice Solution: Treaps [Aragon & Seidel 1989]

A *treap* is a binary tree where every node stores a *key* and a *priority*. It is a BST w. r. t. the keys and a heap w. r. t. the priorities.

A *treap* is a binary tree where every node stores a *key* and a *priority*. It is a BST w. r. t. the keys and a heap w. r. t. the priorities.

- The root of any subtree has lowest priority

A *treap* is a binary tree where every node stores a *key* and a *priority*. It is a BST w. r. t. the keys and a heap w. r. t. the priorities.

- The root of any subtree has lowest priority
- Elements in a left subtree have a smaller key than the root

A *treap* is a binary tree where every node stores a *key* and a *priority*. It is a BST w. r. t. the keys and a heap w. r. t. the priorities.

- The root of any subtree has lowest priority
- Elements in a left subtree have a smaller key than the root
- Elements in a right subtree have a greater key than the root

A *treap* is a binary tree where every node stores a *key* and a *priority*. It is a BST w. r. t. the keys and a heap w. r. t. the priorities.

- The root of any subtree has lowest priority
- Elements in a left subtree have a smaller key than the root
- Elements in a right subtree have a greater key than the root

If priorities are distinct, the shape of a treap is thus uniquely defined by its entries.

If priorities are distinct, the shape of a treap is thus uniquely defined by its entries.

If priorities are distinct, the shape of a treap is thus uniquely defined by its entries.

$\Longrightarrow$ Insert list of elements into a treap in *any* order $\simeq$
Insert elements into a BST by increasing priority

If priorities are distinct, the shape of a treap is thus uniquely defined by its entries.

$\Longrightarrow$ Insert list of elements into a treap in *any* order $\simeq$
Insert elements into a BST by increasing priority

Idea: Choose priority randomly from $[0; 1] \subseteq \mathbb{R}$ upon insertion

If priorities are distinct, the shape of a treap is thus uniquely defined by its entries.

$\Longrightarrow$ Insert list of elements into a treap in *any* order $\simeq$
Insert elements into a BST by increasing priority

Idea: Choose priority randomly from $[0;1] \subseteq \mathbb{R}$ upon insertion
$\Longrightarrow$ Treap behaves like a random BST

If priorities are distinct, the shape of a treap is thus uniquely defined by its entries.

$\implies$ Insert list of elements into a treap in *any* order $\simeq$
Insert elements into a BST by increasing priority

Idea: Choose priority randomly from $[0;1] \subseteq \mathbb{R}$ upon insertion
$\implies$ Treap behaves like a random BST

Randomised Treap

Definition of some operations on treaps

ins :: $(\alpha \times \mathbb{R}) \Rightarrow (\alpha, \mathbb{R})$ treap $\Rightarrow (\alpha, \mathbb{R})$ treap

Definition of some operations on treaps

$\mathsf{ins} :: (\alpha \times \mathbb{R}) \Rightarrow (\alpha, \mathbb{R})\ \mathsf{treap} \Rightarrow (\alpha, \mathbb{R})\ \mathsf{treap}$

$\mathsf{rins} :: \alpha \Rightarrow (\alpha, \mathbb{R})\ \mathsf{treap} \Rightarrow (\alpha, \mathbb{R})\ \mathsf{treap}$
$\mathsf{rins}\ x\ t = \mathbf{do}\ \{p \leftarrow \mathcal{U};\ \mathbf{return}\ (\mathsf{ins}\ (x, p)\ t)\}$

Definition of some operations on treaps

$\text{ins} :: (\alpha \times \mathbb{R}) \Rightarrow (\alpha, \mathbb{R})\ \text{treap} \Rightarrow (\alpha, \mathbb{R})\ \text{treap}$

$\text{rins} :: \alpha \Rightarrow (\alpha, \mathbb{R})\ \text{treap} \Rightarrow (\alpha, \mathbb{R})\ \text{treap}$
$\text{rins}\ x\ t = \textbf{do}\ \{p \leftarrow \mathcal{U};\ \textbf{return}\ (\text{ins}\ (x, p)\ t)\}$

$\text{rinss} :: \alpha\ \text{list} \Rightarrow (\alpha, \mathbb{R})\ \text{treap} \Rightarrow (\alpha, \mathbb{R})\ \text{treap}$
$\text{rinss}\ []\ t = \textbf{return}\ t$
$\text{rinss}\ ([x]\ @\ xs)\ t = \textbf{do}\ \{t' \leftarrow \text{rins}\ x\ t;\ \text{rinss}\ xs\ t'\}$

Proof of the main result on treaps:

$$\text{rinss } xs = \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{treap\_of } [(x, p(x)) \mid x \leftarrow xs]\}$$

Proof of the main result on treaps:

$$\text{rinss } xs = \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{treap\_of } [(x, p(x)) \mid x \leftarrow xs]\}$$

$$\overset{\text{project}}{\simeq} \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{mk\_bst } (\text{sort\_key } p \ xs)\}$$

Proof of the main result on treaps:

$$
\begin{aligned}
\text{rinss } xs &= \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{treap\_of } [(x, p(x)) \mid x \leftarrow xs]\} \\
&\overset{\text{project}}{\simeq} \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{mk\_bst } (\text{sort\_key } p \; xs)\} \\
&= \textbf{do } \{R \leftarrow \text{rel\_from\_prios } \mathcal{U}^{xs}; \\
&\qquad\quad \textbf{return } \text{mk\_bst } (\text{sort\_rel } R \; xs)\}
\end{aligned}
$$

Proof of the main result on treaps:

$$
\begin{aligned}
\text{rinss } xs &= \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return} \text{ treap\_of } [(x, p(x)) \mid x \leftarrow xs]\} \\
&\overset{\text{project}}{\simeq} \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return} \text{ mk\_bst } (\text{sort\_key } p \ xs)\} \\
&= \textbf{do } \{R \leftarrow \text{rel\_from\_prios } \mathcal{U}^{xs}; \\
&\qquad\quad \textbf{return} \text{ mk\_bst } (\text{sort\_rel } R \ xs)\} \\
&= \textbf{do } \{R \leftarrow \text{uniform } (\text{linorder\_on } R); \\
&\qquad\quad \textbf{return} \text{ mk\_bst } (\text{sort\_rel } R \ xs)\}
\end{aligned}
$$

Proof of the main result on treaps:

$$
\begin{aligned}
\text{rinss } xs &= \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{treap\_of } [(x, p(x)) \mid x \leftarrow xs]\} \\
&\overset{\text{project}}{\simeq} \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{mk\_bst } (\text{sort\_key } p\ xs)\} \\
&= \textbf{do } \{R \leftarrow \text{rel\_from\_prios } \mathcal{U}^{xs}; \\
&\qquad\quad \textbf{return } \text{mk\_bst } (\text{sort\_rel } R\ xs)\} \\
&= \textbf{do } \{R \leftarrow \text{uniform } (\text{linorder\_on } R); \\
&\qquad\quad \textbf{return } \text{mk\_bst } (\text{sort\_rel } R\ xs)\} \\
&= \textbf{do } \{xs' \leftarrow \text{rperm } xs;\ \textbf{return } \text{mk\_bst } xs'\}
\end{aligned}
$$

Proof of the main result on treaps:

$$
\begin{aligned}
\text{rinss } xs &= \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{treap\_of } [(x, p(x)) \mid x \leftarrow xs]\} \\
&\overset{\text{project}}{\simeq} \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{mk\_bst } (\text{sort\_key } p \ xs)\} \\
&= \textbf{do } \{R \leftarrow \text{rel\_from\_prios } \mathcal{U}^{xs}; \\
&\qquad\quad \textbf{return } \text{mk\_bst } (\text{sort\_rel } R \ xs)\} \\
&= \textbf{do } \{R \leftarrow \text{uniform } (\text{linorder\_on } R); \\
&\qquad\quad \textbf{return } \text{mk\_bst } (\text{sort\_rel } R \ xs)\} \\
&= \textbf{do } \{xs' \leftarrow \text{rperm } xs; \ \textbf{return } \text{mk\_bst } xs'\} \\
&= \text{random\_bst } xs
\end{aligned}
$$

Proof of the main result on treaps:

$$\begin{aligned}
\text{rinss } xs &= \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{treap_of } [(x, p(x)) \mid x \leftarrow xs]\} \\
&\overset{\text{project}}{\simeq} \textbf{do } \{p \leftarrow \mathcal{U}^{xs}; \textbf{return } \text{mk_bst } (\text{sort_key } p\ xs)\} \\
&= \textbf{do } \{R \leftarrow \text{rel_from_prios } \mathcal{U}^{xs}; \\
&\qquad\quad \textbf{return } \text{mk_bst } (\text{sort_rel } R\ xs)\} \\
&= \textbf{do } \{R \leftarrow \text{uniform } (\text{linorder_on } R); \\
&\qquad\quad \textbf{return } \text{mk_bst } (\text{sort_rel } R\ xs)\} \\
&= \textbf{do } \{xs' \leftarrow \text{rperm } xs;\ \textbf{return } \text{mk_bst } xs'\} \\
&= \text{random_bst } xs \qquad \square
\end{aligned}$$

Measures

Problem: Random treaps are a *continuous* distribution
$\implies$ cannot use PMFs

Measures

Problem: Random treaps are a *continuous* distribution
$\implies$ cannot use PMFs

Instead we have to use general measures (Ω, Σ, μ)

Measures

Problem: Random treaps are a *continuous* distribution
$\Longrightarrow$ cannot use PMFs

Instead we have to use general measures (Ω, Σ, μ)

But what does a suitable Σ-algebra for trees look like?

Measures

Problem: Random treaps are a *continuous* distribution
$\Longrightarrow$ cannot use PMFs

Instead we have to use general measures (Ω, Σ, μ)

But what does a suitable Σ-algebra for trees look like?

- Functor $\mathcal{T}$ that maps a Σ-algebra over a set A to a Σ-algebra of trees with elements from A

Measures

Problem: Random treaps are a *continuous* distribution
$\Longrightarrow$ cannot use PMFs

Instead we have to use general measures (Ω, Σ, μ)

But what does a suitable Σ-algebra for trees look like?

- Functor $\mathcal{T}$ that maps a Σ-algebra over a set A to a Σ-algebra of trees with elements from A
- The 'Node' constructor is a measurable function from $\mathcal{T}(M) \otimes M \otimes \mathcal{T}(M)$ to $\mathcal{T}(M)$

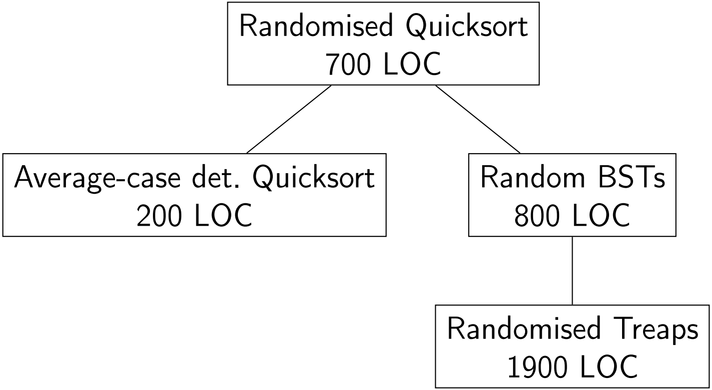
Measures

Problem: Random treaps are a *continuous* distribution
$\Longrightarrow$ cannot use PMFs

Instead we have to use general measures (Ω, Σ, μ)

But what does a suitable Σ-algebra for trees look like?

- Functor $\mathcal{T}$ that maps a Σ-algebra over a set A to a Σ-algebra of trees with elements from A
- The 'Node' constructor is a measurable function from $\mathcal{T}(M) \otimes M \otimes \mathcal{T}(M)$ to $\mathcal{T}(M)$
- Other tree operations (projections, primitive recursion) are similarly measurable


Randomised Quicksort
700 LOC
Average-case det. Quicksort
200 LOC
Random BSTs
800 LOC
Randomised Treaps
1900 LOC

Bonus: Executability

value random_bst $\{1, 2, 3 :: \mathrm{int}\}$

Bonus: Executability

value random_bst $\{1, 2, 3 :: \text{int}\}$

pmf_of_alist

$$[(\langle\langle\langle\langle\rangle, 1, \langle\rangle\rangle, 2, \langle\rangle\rangle, 3, \langle\rangle\rangle, 1/6),$$
$$(\langle\langle\langle\rangle, 1, \langle\langle\rangle, 2, \langle\rangle\rangle\rangle, 3, \langle\rangle\rangle, 1/6),$$
$$(\langle\langle\rangle, 1, \langle\langle\langle\rangle, 2, \langle\rangle\rangle, 3, \langle\rangle\rangle\rangle, 1/6),$$
$$(\langle\langle\langle\rangle, 1, \langle\rangle\rangle, 2, \langle\langle\rangle, 3, \langle\rangle\rangle\rangle, 1/3),$$
$$(\langle\langle\rangle, 1, \langle\langle\rangle, 2, \langle\langle\rangle, 3, \langle\rangle\rangle\rangle\rangle, 1/6)]$$

int tree pmf

Bonus: Executability

value random_bst $\{1, 2, 3 :: \text{int}\}$

$$
\begin{array}{l}
\quad \text{pmf\_of\_alist} \\
\qquad [(\langle\langle\langle\langle\rangle, 1, \langle\rangle\rangle, 2, \langle\rangle\rangle, 3, \langle\rangle\rangle, 1/6), \\
\qquad\ (\langle\langle\langle\rangle, 1, \langle\langle\rangle, 2, \langle\rangle\rangle\rangle, 3, \langle\rangle\rangle, 1/6), \\
\qquad\ (\langle\langle\rangle, 1, \langle\langle\langle\rangle, 2, \langle\rangle\rangle, 3, \langle\rangle\rangle\rangle, 1/6), \\
\qquad\ (\langle\langle\langle\rangle, 1, \langle\rangle\rangle, 2, \langle\langle\rangle, 3, \langle\rangle\rangle\rangle, 1/3), \\
\qquad\ (\langle\langle\rangle, 1, \langle\langle\rangle, 2, \langle\langle\rangle, 3, \langle\rangle\rangle\rangle\rangle, 1/6)] \\
\qquad\qquad \text{int tree pmf}
\end{array}
$$

value measure_pmf.expectation $(\text{random\_bst } \{1..6 :: \text{int}\})$ height

Bonus: Executability

$$
\begin{array}{l}
\textbf{value}\ \text{random\_bst}\ \{1, 2, 3 :: \text{int}\} \\
\quad \text{pmf\_of\_alist} \\
\qquad [(\langle\langle\langle\langle\rangle, 1, \langle\rangle\rangle, 2, \langle\rangle\rangle, 3, \langle\rangle\rangle, 1/6), \\
\qquad\ (\langle\langle\langle\rangle, 1, \langle\langle\rangle, 2, \langle\rangle\rangle\rangle, 3, \langle\rangle\rangle, 1/6), \\
\qquad\ (\langle\langle\rangle, 1, \langle\langle\langle\rangle, 2, \langle\rangle\rangle, 3, \langle\rangle\rangle\rangle, 1/6), \\
\qquad\ (\langle\langle\langle\rangle, 1, \langle\rangle\rangle, 2, \langle\langle\rangle, 3, \langle\rangle\rangle\rangle, 1/3), \\
\qquad\ (\langle\langle\rangle, 1, \langle\langle\rangle, 2, \langle\langle\rangle, 3, \langle\rangle\rangle\rangle\rangle, 1/6)] \\
\qquad\quad \text{int tree pmf}
\end{array}
$$

$$
\begin{array}{l}
\textbf{value}\ \text{measure\_pmf.expectation}\ (\text{random\_bst}\ \{1..6 :: \text{int}\})\ \text{height} \\
\quad 65\ /\ 15 :: \text{real}
\end{array}
$$

Conclusion

- Formalisation of textbook randomised algorithms/data structures is feasible with Isabelle

Conclusion

- Formalisation of textbook randomised algorithms/data structures is feasible with Isabelle
- PMF proofs are nice, high-level, and readable

Conclusion

- Formalisation of textbook randomised algorithms/data structures is feasible with Isabelle
- PMF proofs are nice, high-level, and readable
- Measure proofs can get ugly due to measurability issues

Conclusion

- Formalisation of textbook randomised algorithms/data structures is feasible with Isabelle
- PMF proofs are nice, high-level, and readable
- Measure proofs can get ugly due to measurability issues

Interesting related topics:

Conclusion

- Formalisation of textbook randomised algorithms/data structures is feasible with Isabelle
- PMF proofs are nice, high-level, and readable
- Measure proofs can get ugly due to measurability issues

Interesting related topics:

- Tail bounds [Tassarotti & Harper 2018]

Conclusion

- Formalisation of textbook randomised algorithms/data structures is feasible with Isabelle
- PMF proofs are nice, high-level, and readable
- Measure proofs can get ugly due to measurability issues

Interesting related topics:

- Tail bounds [Tassarotti & Harper 2018]
- Treaps can also use discrete distributions

Conclusion

- Formalisation of textbook randomised algorithms/data structures is feasible with Isabelle
- PMF proofs are nice, high-level, and readable
- Measure proofs can get ugly due to measurability issues

Interesting related topics:

- Tail bounds [Tassarotti & Harper 2018]
- Treaps can also use discrete distributions
- Randomised BSTs [Martinez & Roura 1997]

Conclusion

- Formalisation of textbook randomised algorithms/data structures is feasible with Isabelle
- PMF proofs are nice, high-level, and readable
- Measure proofs can get ugly due to measurability issues

Interesting related topics:

- Tail bounds [Tassarotti & Harper 2018]
- Treaps can also use discrete distributions
- Randomised BSTs [Martinez & Roura 1997]
- Skip Lists (already done, [Haslbeck & E. 2018])