Algorithms, Probability, and Computing

Angelika Steger
Institut für Theoretische Informatik

steger@inf.ethz.ch
Random Search Trees

Lemma 2.1. \( S \subseteq \mathbb{R} \), finite. Given a tree in \( B_S \), 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

\[ D_n := \text{depth of smallest key}; \quad d_n := E[D_n] \]

\( n=3: \)

- \( d_1 = 0, \quad d_2 = \frac{1}{2}, \quad d_3 = \frac{5}{6} \)
Depth of Smallest Key

\[ D_n := \text{depth of smallest key}; \quad d_n := E[D_n] \]

\[ d_1 = 0, \quad d_2 = 1/2, \quad d_3 = 5/6 \]

\[
E[D_n] = \sum_{i=1}^{n} \left( E[D_n | \text{rk(root)} = i] \cdot \Pr[\text{rk(root)} = i] \right)
\]

\[ = \begin{cases} 
0, & \text{if } i = 1, \text{ and} \\
1 + E[D_{i-1}], & \text{otherwise.} 
\end{cases} \quad \text{with } \Pr = 1/n \]

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

$$X_n := \text{sum of depths of all keys in tree;} \quad x_n := E[X_n]$$

$n=3$:  

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

$x_1 = 0, \quad x_2 = 1, \quad x_3 = \frac{8}{3}$
Overall Depths of Keys

\[ X_n := \text{sum of depths of all keys in tree;} \quad x_n := E[X_n] \]

\[ x_1 = 0, \quad x_2 = 1, \quad x_3 = \frac{8}{3} \]

\[
E[X_n] = \sum_{i=1}^{n} E[X_n | \text{rk}(\text{root}) = i] \cdot \frac{\Pr[\text{rk}(\text{root}) = i]}{1/n}
\]

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

\[
x_n = \begin{cases} 
0, & \text{if } n = 0, \text{ and} \\
n - 1 + \frac{2}{n} \sum_{i=0}^{n-1} x_i, & \text{otherwise.}
\end{cases}
\]
\( D_n^{(i)} := \text{depth of key of rank } i \)

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

n=3:

\[
\begin{align*}
\frac{1}{6} \cdot 2 & \quad \frac{1}{6} \cdot 2 & \quad \frac{1}{3} \cdot 1 & \quad \frac{1}{6} \cdot 2 & \quad \frac{1}{6} \cdot 2 \\
1 & \quad 2 & \quad 1 \text{ (with height 3)} & \quad 1 \quad 2 & \quad 1 \quad 2 \\
3 & \quad 2 & \quad 3 & \quad 3 & \quad 3
\end{align*}
\]

\( x_1 = 0, \quad x_2 = 1, \quad x_3 = \frac{5}{3} \)
\( D_n^{(i)} := \text{depth of key of rank } i \)

\[ X_n := \max_{1 \leq i \leq n} D_n^{(i)} \]

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

\[ E[X_n] = E[\max_{1 \leq i \leq n} D_n^{(i)}] = \text{? ? ?} \]
Height

\[ D_n^{(i)} := \text{depth of key of rank } i \]

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

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

\[ \mathbb{E}[X_n] \leq \log \mathbb{E}[2^{X_n}] = \log \mathbb{E}[2^{\max_{i=1}^n D_n^{(i)}}] \]

Jensen’s Inequality: If \( f : \mathbb{R} \to \mathbb{R} \) is a convex function, then \( f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \)
$D_n^{(i)} := \text{depth of key of rank } i$

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

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

\[
\leq \log \mathbb{E}
\left[\sum_{i=1, i \text{ is leaf}}^n 2^{D_n^{(i)}}\right] =: Z_n
\]
\(D_n^{(i)} := \text{depth of key of rank } i\)

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

\(Z_n := \sum_{i=1, i \text{ is leaf}}^{n} 2^{D_n^{(i)}}\)

\[z_n := E[Z_n]\]

\(n=3:\)

\[
\begin{align*}
\text{Case 1:} & \quad \frac{1}{6} \cdot 2^2 \\
\text{Case 2:} & \quad \frac{1}{6} \cdot 2^2 \\
\text{Case 3:} & \quad \frac{1}{3} \cdot 2 \cdot 2^1 \\
\text{Case 4:} & \quad \frac{1}{6} \cdot 2^2 \\
\text{Case 5:} & \quad \frac{1}{6} \cdot 2^2 \\
\end{align*}
\]

\(z_1 = 1, \; z_2 = 2, \; z_3 = 4\)
\[ D_n^{(i)} := \text{depth of key of rank } i \]

\[ X_n := \max_{1 \leq i \leq n} D_n^{(i)} ; \quad \mathbb{E}[X_n] \leq \log_2(\mathbb{E}[Z_n]) \]

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

\[ z_n := \mathbb{E}[Z_n] \]

\[ \mathbb{E}[Z_n] = \sum_{i=1}^{n} \frac{\mathbb{E}[Z_n | \text{rk(root) = i}] \cdot \mathbb{P}[\text{rk(root) = i}]}{2(\mathbb{E}[Z_{i-1}] + \mathbb{E}[Z_{n-i}])} = \frac{1}{n} \]

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

\[ D_n^{(i)} := \text{depth of key of rank } i \]

\[ X_n := \max_{1 \leq i \leq n} D_n^{(i)} \]

\[
E[X_n] \leq \log C \prod_{0}^{X_n} = \log E[C_{\max_{i=1}^{n} D_n^{(i)}}] \]

Repeat calculations from before:

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

Optimize C:

\[
E[X_n] \leq 4.311.. \ln(n)
\]

(constant is known to be best possible, cf Devroye'86)
Depth of Key of Rank $i$

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

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

\[
A_i^j := \begin{cases} 
1, & \text{if node } j \text{ is ancestor of node } i, \text{ and} \\
0, & \text{otherwise.}
\end{cases}
\]

\[
E\left[ D_n^{(i)} \right] = \sum_{j=1, j \neq i}^{n} E\left[ A_i^j \right]
\]
Random Search Trees

\[ \mathbb{E}[\text{depth of smallest key}] = H_n - 1 = \ln n + O(1) \]
\[ \mathbb{E}[\text{sum of depths}] = 2(n + 1) H_n - 4n = 2n \ln n + O(n) \]
\[ \mathbb{E}[\text{max depth}] \leq 4.311 \ldots \ln n \]
\[ \mathbb{E}[\text{depth of key of rank } i] = H_i + H_{n-i+1} - 2 \leq 2 \ln n \]
Depth of Key of Rank $i$

$$D_{n(i)} := \text{depth of key of rank } i; \quad d_{i,n} := \mathbb{E}[D_{n(i)}]$$

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

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

$$\mathbb{E}[D_{n(i)}] = \sum_{j=1, j \neq i}^{n} \mathbb{E}[A_i^j] = \sum_{j=1, j \neq i}^{n} \Pr[A_i^j = 1]:$$

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

$$\Pr[A_i^j = 1] = \Pr[\text{node } j \text{ is ancestor of node } i] = \frac{1}{|i-j|+1}.$$
function quicksort(S)
if S = ∅ then return ();
else

    x ← 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 = ∅ then return ();
else
    x ← u.a.r. S;
    split S into $S^{<x}$, {x}, $S^{>x}$;
    return quicksort($S^{<x}$) \circ (x) \circ quicksort($S^{>x}$);

$t_n := \text{expected number of comparisons for } n \text{ keys}$

$$t_n = n - 1 + \sum_{i=1}^{n} \left( t_{i-1} + t_{n-i} \right) \frac{1}{n} = n - 1 + \frac{2}{n} \sum_{i=1}^{n} t_{i-1} = E[\text{sum of depths}]$$
Random Search Trees

\[ S = \{ \text{Tom, Ben, Tim, Leo} \} \]

\[ S = \{ 1, 2, 3, 4 \} \]

\[ \tilde{B}_S \rightarrow \begin{cases} \lambda, & \text{if } S = \emptyset, \text{ and} \\ \chi, & \text{for } \chi \in_{\text{u.a.r.}} S, \text{ otherwise.} \end{cases} \]
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] ≤ 4.311.. ln n
E[depth of key of rank i] = $H_i + H_{n-i+1} - 2$ ≤ 2 ln n

for $x \in u.a.r. S$, otherwise.

if $S = \emptyset$, and

$H_n = \ln n + O(1)$

$\tilde{B}_S \rightarrow \begin{cases} \lambda, \\
\tilde{B}_{S<x} \quad \tilde{B}_{S>x} \end{cases}$
Treap = (search) tree & (min) heap

- defined for sets \( Q \subseteq \mathbb{R} \times \mathbb{R} \)

  keys: \( \text{key}(x) \)     priorities: \( \text{prio}(x) \)

- search tree wrt to keys & min heap wrt to priorities
Treap = (search) tree & (min) heap

- defined for sets $Q \subseteq \mathbb{R} \times \mathbb{R}$
  
  keys: $\text{key}(x)$
  
  priorities: $\text{prio}(x)$

- 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
**Insert**

- 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: \( \forall T \forall x: \text{ expected number of rotations } < 2 \)
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.
Lemma: $\forall T \forall x$: 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)
**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
**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.