Randomized Trees and Treaps

June 9, 2020

Description

We prove below that given a set of n random integers, we can insert them in order into a binary search tree and the expected depth of a node will be $O(\log n)$. This is known as a *random tree*. However, in practice, we don't have all the values that we want to put into the tree up front. For both randomized trees and treaps, the goal is to simulate a random tree by inserting the value at a random index in the existing set of values while using rotations to maintain the search constraints.

Treaps randomly generate a priority for each node and maintain the heap property among the priorities in addition to the search constraints. An equivalent way of describing the treap is that it could be formed by inserting the nodes highest-priority-first into a binary search tree without doing any rebalancing. Since, we generate each priority randomly when adding a node, treap insertion simulates putting a value at a random index in the set.

For a randomized tree, we can think of each node being indexed from 1 to n. Given a tree with n nodes, the $n + 1^{th}$ node would have had a $\frac{1}{n+1}$ chance of being at index 1 (the root).

Time Complexity

Average Depth of a Perfectly Balanced Binary Tree

Suppose we have a perfectly balanced tree with $n = 2^{h+1} - 1$ nodes, where h is the height of the tree. Then the sum of the depths of the nodes is

$$\begin{aligned}\sum_{d=0}^{h} d2^d &= h2^{h+1} - 2^{h+1} + 2 \\ &= h(2^{h+1} - 1) - (2^{h+1} - 1) + h + 1\end{aligned}$$

Dividing by the total number of nodes gives us an average depth of

$$\frac{h(2^{h+1} - 1) - (2^{h+1} - 1) + h + 1}{2^{h+1} - 1} = h - 1 + \frac{h+1}{2^{h+1} - 1}$$

In the limit, we have $\lim_{h \to \infty} h - 1 + \frac{h+1}{2^{h+1}-1} = h - 1 \in O(\log n)$.

Average Depth of a Random Binary Tree

Suppose we have a set S of n random numbers that we insert in order into a binary tree. Let X be a random variable representing the number of ancestors of node x, i.e., the depth of x. Let X_i be a Bernoulli random variable equal to

$$\begin{cases} 1 & \text{if node } x_i \text{ is an ancestor of } x \\ 0 & \text{otherwise} \end{cases}$$

Then, $X = \sum_{i=1}^{n} X_i$, so $E[X] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} P(X_i = 1)$. For any i, node x_i is an ancestor of x only when x_i is the first element in the range $[x_i, x]$ (or $[x, x_i]$ if $x < x_i$) to be inserted into the tree; otherwise, if some other node y were inserted before x_i, y would be at the root of some subtree with x and x_i on opposite branches. If x_i's index in the set is 1 position away from x, then there are 2 nodes in the range $[x_i, x]$, so the probability of picking x_i first is $\frac{1}{2}$. Similarly, if x_i's index is 2 positions away, there are 3 nodes in the range, so the probability of picking x_i first is $\frac{1}{3}$. Generalizing, the probability of picking x_i first when x_i is k steps away from x is

$\frac{1}{1+k}$. Suppose x is at position j in the set. Then,

$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n} P(X_i = 1) \\
&= \sum_{i=1}^{j-1} \frac{1}{i+1} + \sum_{i=j+1}^{n} \frac{1}{(i-j)+1} \\
&= \sum_{i=1}^{j-1} \frac{1}{i+1} + \sum_{i=1}^{n-j} \frac{1}{i+1} \\
&\leq 2\sum_{i=1}^{n} \frac{1}{i+1} \\
&\leq 2\sum_{i=1}^{n} \frac{1}{i} \\
&\leq 2H_n \\
&\in O(\log n)
\end{aligned}
$$

The sum of the depths of all the nodes would be

$$
\sum_{j=1}^{n} \left(\sum_{i=1}^{j-1} \frac{1}{i+1} + \sum_{i=j+1}^{n} \frac{1}{(i-j)+1} \right) = 2n(H_n - 1)
$$

which agrees with the result above.