Binary Search Tree

Binary Search Tree is a binary tree in which a value (comparable) at each node is:

- greater than all the value in the node's left sub-tree
- less than all the values in the node's right sub-tree

A binary tree is a tree such that each nodes has at most 2 children. Height of a binary tree is defined as the maximum number of edges from a root to a leaf. For a binary tree with N nodes, height is bounded as,

$$logN \leq d \leq N-1$$

Assuming all insertion sequences are equally likely, average height over all nodes in a BST is $O(logN)$

We proceed by first calculating average Internal Path Length. Internal path length is the height of all nodes in a tree. Let B be a binary tree of $P(N)$ internal path length containing N nodes. The root node has left sub-tree of say L nodes such that its internal path length is $P(L)$. Consequently, the right sub-tree contains $N-L-1$ nodes and has internal path length $P(N-L-1)$. Since each $N-1$ nodes in left and right sub-tree is one level deeper with respect to the root, we get the following recurrence relation,

$$P(1) = P(0) = 0$$

$$P(N) = P(L) + P(N-L-1) + N - 1$$

We wish to calculate the expected size of each sub-trees. If smallest element was inserted first, then left sub-tree size would be 0. If second smallest element was inserted first, then left sub-tree size would be 1 and so on. Since we assumed that any insertion sequence is equally likely, it follows that any sub-tree size $\in [0, N-1]$ is equally likely. Thus,

$$E[P(L)] = E[P(N-L-1)] = \frac{1}{N}\sum_{i=0}^{N-1} P(i)$$

And the recurrence relation becomes,

$$P(N) = \frac{2}{N}\sum_{i=0}^{N-1} P(i) + N - 1$$

$$NP(N) = 2\sum_{i=0}^{N-1} P(i) + N(N-1)$$

$$(N-1)P(N-1) = 2\sum_{i=0}^{N-2} P(i) + (N-1)(N-2)$$

Subtracting above two equations and ignoring constants and scaling factors,

$$NP(N) = (N+1)P(N-1) + N$$

Rearranging terms,

$$\frac{P(N)}{N+1} = \frac{P(N-1)}{N} + \frac{1}{N+1}$$

$$\frac{P(N-1)}{N} = \frac{P(N-2)}{N-1} + \frac{1}{N}$$

$$...$$

$$\frac{P(1)}{2} = \frac{P(0)}{1} + \frac{1}{2}$$

Telescoping the sum,

$$\frac{P(N)}{N+1} = \frac{P(0)}{1} + \sum_{i=2}^{N+1} \frac{1}{i}$$

Thus,

$$P(N) = O(NlogN)$$

Hence, the expected height of a node in a binary tree is $\frac{O(NlogN)}{N} = O(logN)$.

However, this does not imply that average run-time for all operations in BST is $O(logN)$. This is because deletion, which finds a replacement node from the right sub-tree, tends to make left sub-tree deeper. Thus after multiple deletions, the size of a sub-tree is not equally likely as we assumed. In fact, it is shown that after $\Theta(N^2)$ $insert/delete$ operations, the expected height of a node in the binary search tree becomes $\Theta(\sqrt{N})$. This could be avoided if replacement strategy is randomized between left and right sub-trees.