Complexity

Jay

February 2020

1 Q1

Let n be the size of the tree, m be the height of the tree. $m \approx \log n$ for a balanced tree.

1.1 (a) search

$\mathcal{O}(n)$, $\Omega(1)$
Worst case is when one has to go through all nodes to find the value;
Best case is when node to find is at the root.

1.2 (b) insert

$\mathcal{O}(m)$, $\Omega(1)$
Insertion for a normal binary tree is to insert when we hit a leaf. Worst case is when one has to go all the way through the longest path.
One best case scenario is when the root.left is None, we can insert to the left of the root.

1.3 (c) delete

$\mathcal{O}(m)$, $\Omega(1)$
(What is the general rule of deleting a node from a normal binary tree?)
Worst case is to delete root, then substitute its value with a leaf on the longest root-to-leaf path.
One best case is when the value to delete is root.left, and at the same time a leaf.

2 Q2

Let n be the size of the tree. If the tree is generated randomly, for a full tree ($n = 2^k - 1$ for some k, or alternatively, $k \approx \log n$), the distribution of m is given by:

The analysis of complexity is the same as Q1, when n and m is fixed.

2.1 (a) search

$\mathcal{O}(n)$, $\Omega(1)$

2.2 (b) insert

$\mathcal{O}(m)$, $\Omega(1)$

2.3 (c) delete

$\mathcal{O}(m)$, $\Omega(1)$

3 Q3

Let n be the size of the tree, m be the height of the tree. $m \approx \log n$ for a balanced tree.

3.1 (a) search

$\mathcal{O}(m)$, $\Omega(1)$
Worst case is when one has to go all the way through the longest root-to-leaf path;
Best case is when node to find is at the root.

3.2 (b) insert

$\mathcal{O}(m)$, $\Omega(1)$
Worst case similar to (a).
One best case scenario is when the value < root.value and root.left is None.

3.3 (c) delete

$\mathcal{O}(m)$, $\Omega(1)$
Worst case can be either:
1. to delete root and run a min search;
2. to delete a leaf on a longest root-to-leaf path.
Both cases are $\mathcal{O}(m)$.

One best case is when the value to delete is root.left, and at the same time a leaf.

4 Q4

Let n be the size of the tree. For a binary tree, let $f(n)$ be the average height of a size n random bst. Then an inductive formula for $n > 2$ can be given by:

$$f(n) = \begin{cases} \frac{2}{2k}(\sum_{i=k}^{2k-1}(1+f(i))) & \text{if } n = 2k \\ \frac{2}{2k+1}(\sum_{i=k+1}^{2k}(1+f(i))) + \frac{1}{2k+1}(1+f(k)) & \text{if } n = 2k+1 \end{cases} \tag{1}$$

With base $f(1) = 1$ and $f(2) = 2$. Solve this we get $f(n) =$

The analysis of complexity is the same as Q3, when n and m is fixed.

4.1 (a) search

$\mathcal{O}(n)$, $\Omega(1)$

4.2 (b) insert

$\mathcal{O}(m)$, $\Omega(1)$

4.3 (c) delete

$\mathcal{O}(m)$, $\Omega(1)$