CS202: Fundamentals of Computer Science II

HOMEWORK ASSIGNMENT - 2

Binary Search Trees

Zeynep Begüm Kara
ID: 22003880
CS202 - 01

November 7, 2022

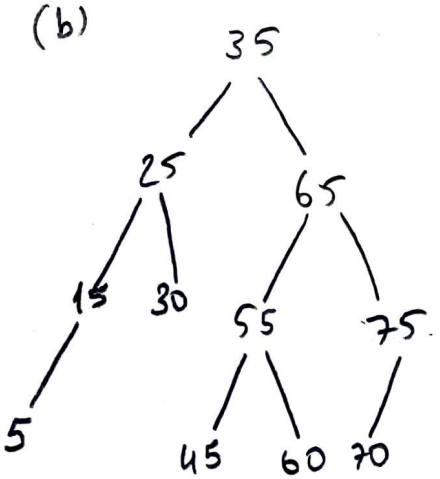
Question -1

(a) Preorder: K N P T C O R S A

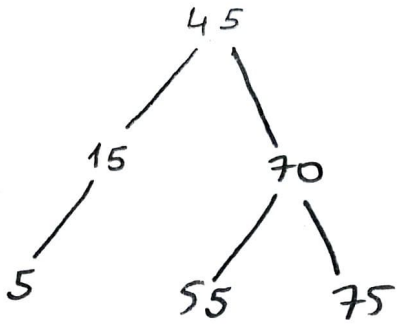
Inorder: P T N C K R O S A

Postorder: T P C N R A S O K

(b)

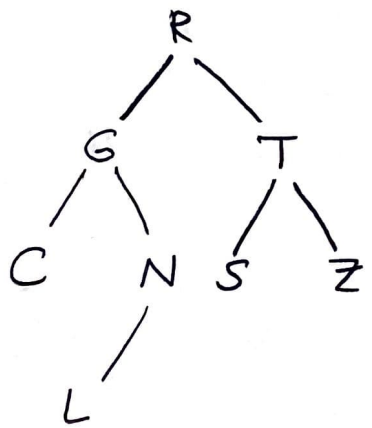


— insertion —



— after deletion —

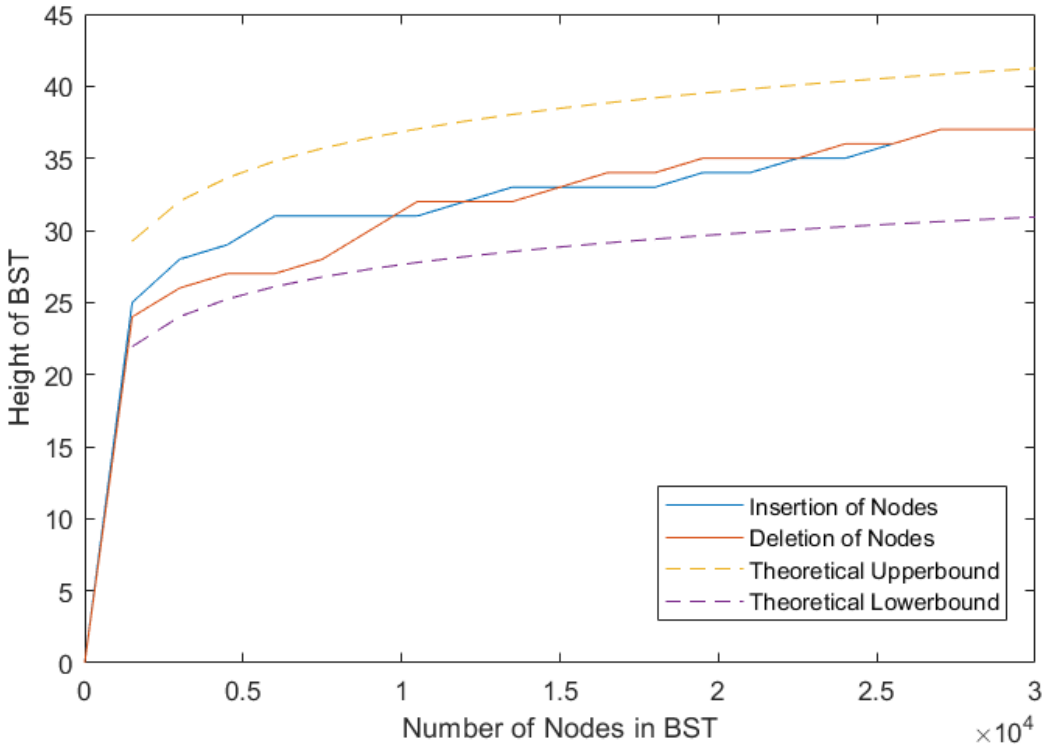
(c)



preorder: R G C N L T S Z

Postorder: C L N G S Z T R

Figure 1: Binary Search Tree: Tree Height vs. Number of Elements



In this study, it is aimed to observe the relationship between the number of nodes with the height of a binary search tree. To analyse this relation a random integer sequence with $3 * 10^4$ nodes were generated and a binary search tree was constructed accordingly due to search property. During that process, the height of the tree were found in each 1500 insertions by calling getHeight(), which founds the height by calculating the height of each node recursively and picks the largest one by definition. We theoretically expected the height of the tree to be directly proportional to $logn$ in average cases where n is the number of nodes. The experimental results i.e. the height of the tree throughout insertions, which was shown in blue graph on Figure 1, turned out to be satisfied since the graph fits $\Theta(logn)$ complexity. Observe that purple dashed graph is a lower bound or $O(logn)$ for our results when $C = 3$, $n > n_0 = 0.5 * 10^4$ and $f(n) = logn$. Also, observe that yellow dashed graph is a upper bound or $\Omega(n)$ for our results when $C = 4$, $n > n_0 = 0.5 * 10^4$ and $f(n) = logn$. After array of randomly generated integers are shuffled, tree is destructed accordingly. During that process the height of the tree were also found in each 1500 insertions. One can easily identify that deletion of nodes, which was shown in red graph on Figure 1, has the same complexity by the same observation done previously. Therefore, height of a binary tree has $\Theta(logn)$ complexity when there are n nodes and the nodes are consist of random integers. One can found more strict upper bounds and lower bounds by choosing appropriate C and n_0 for $logn$. Also decreasing the size of the intervals can help to the graph to be more look like $logn$ due to sensitivity of data alignment. We see that the difference between blue and red graphs is greater in when $n < 10^4$ than compared to the rest of the graph this might have resulted from the configuration of randomly generated numbers. Since the randomly generated integers are not completely random and shuffleArray() also utilizes from pseudo-random numbers, the experimental data have fluctuations. Especially the relation of height and node number can be easily disrupted if the random number generator produces numbers in order. Hence, our fluctuations can be resulted from partially sets of number sequences that are in order. Overall, the graph suits $\theta(logn)$ complexity as we expected.

The worst case of height complexity of a binary search tree is when the sequence of nodes is in sorted either in ascending or descanting ways. In such cases, since we must preserve the search property, when we are constructing the tree we will always going to insert the new items to only left subtree or only the right subtree according to whether we have an increasing or decreasing sequence. Therefore, the height of the tree will be equal to the number of nodes. This clearly implies that the worst case complexity of height is $\Theta(n)$ where n is the number of nodes in the tree.