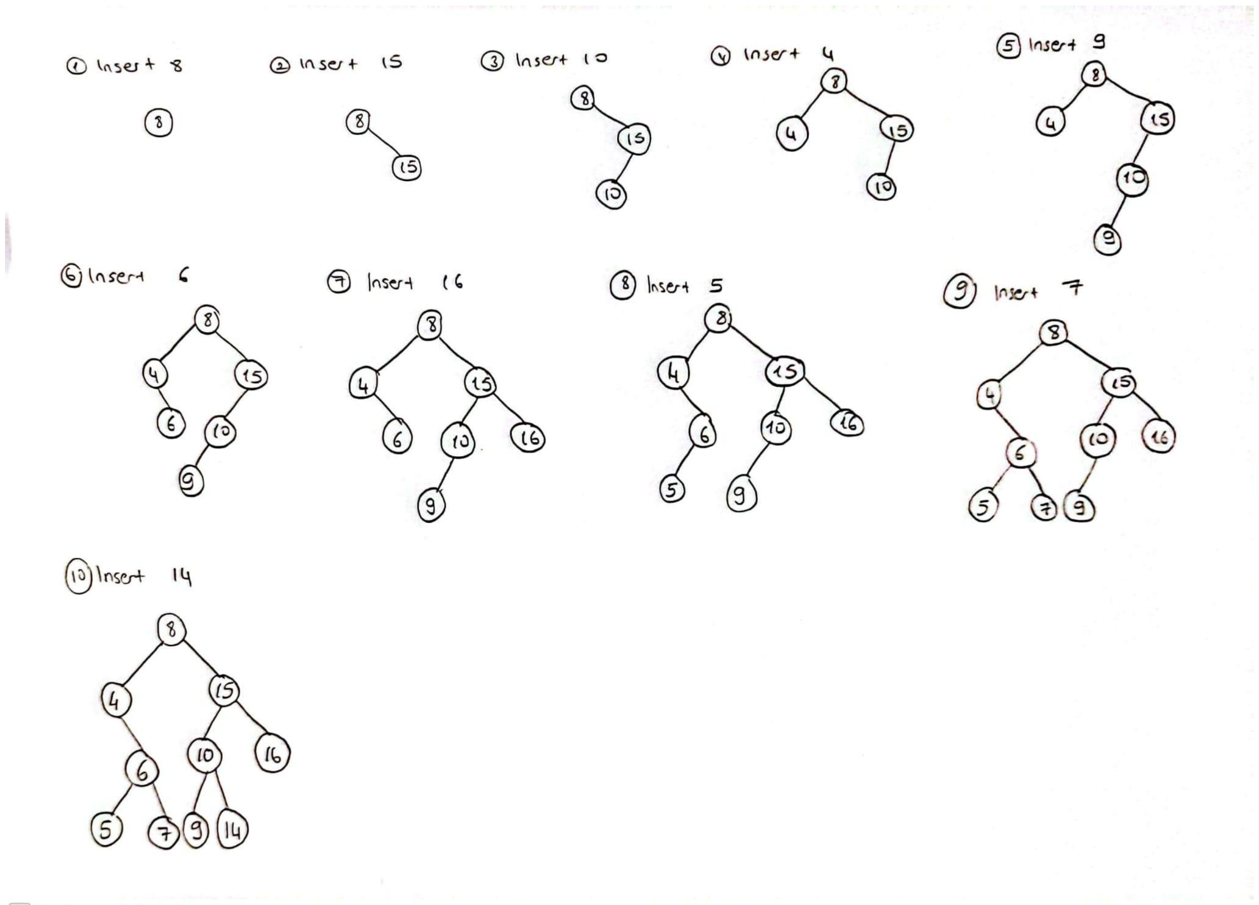
Gizem Gökçe Işık
21803541

22/03/2023

CS202
HOMEWORK 2 - BINARY SEARCH TREES

QUESTION 1

Part a) Insert 8, 15, 10, 4, 9, 6, 16, 5, 7, 14 into an empty binary search tree in the given order. Show the resulting BST after every insertion.



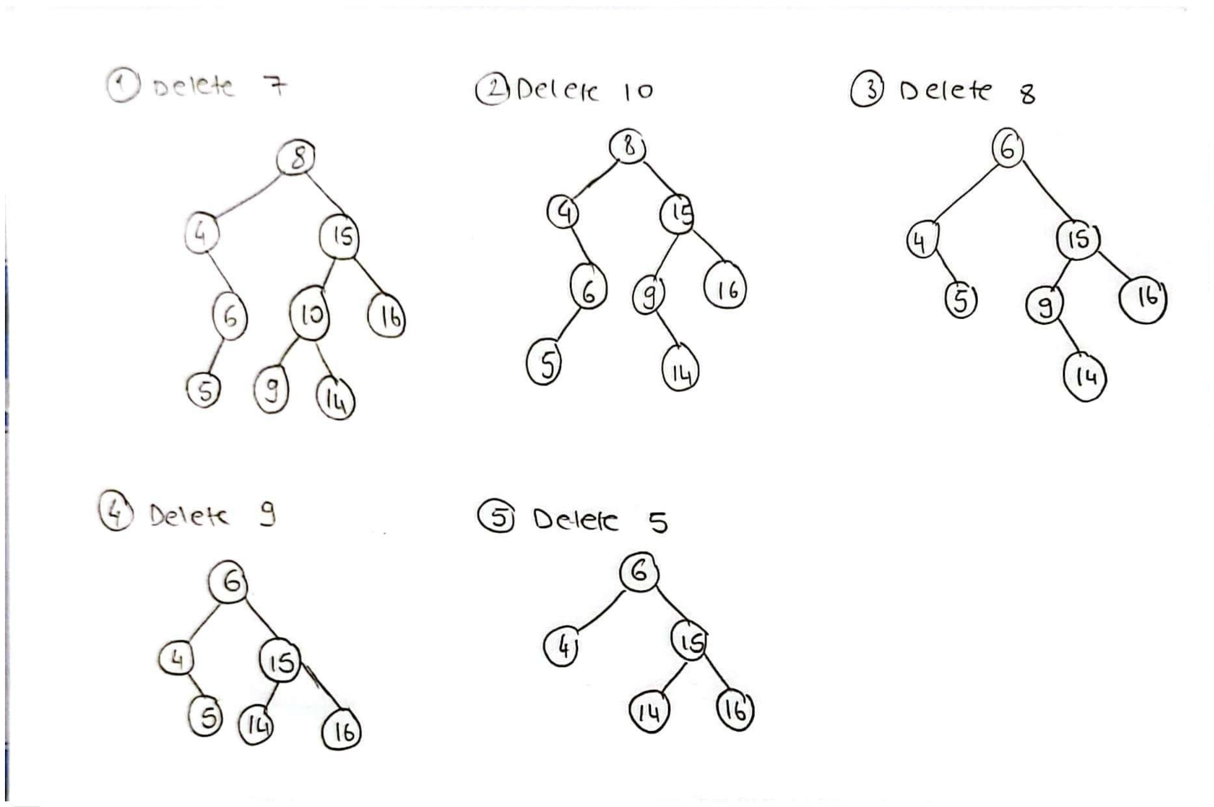
Part b) What are the preorder, inorder, and postorder traversals of the BST you have after (a)?

Preorder Traversal = 8-4-6-5-7-15-10-9-14-6

Inorder Traversal = 4-5-6-7-8-9-10-14-15-16

Postorder Traversal = 5-7-6-4-9-14-10-16-15-8

Part c) Delete 7, 10, 8, 9, 5 from the BST you have after (a) in the given order. Show the resulting BST after every deletion.



Part d) Write a recursive pseudocode implementation for finding the minimum element in a binary search tree.

```
int findMin(Node* root)
        if root == NULL
           return -1
        else if root->left == NULL
            return root->data
        else
            findMin(root->left)
```

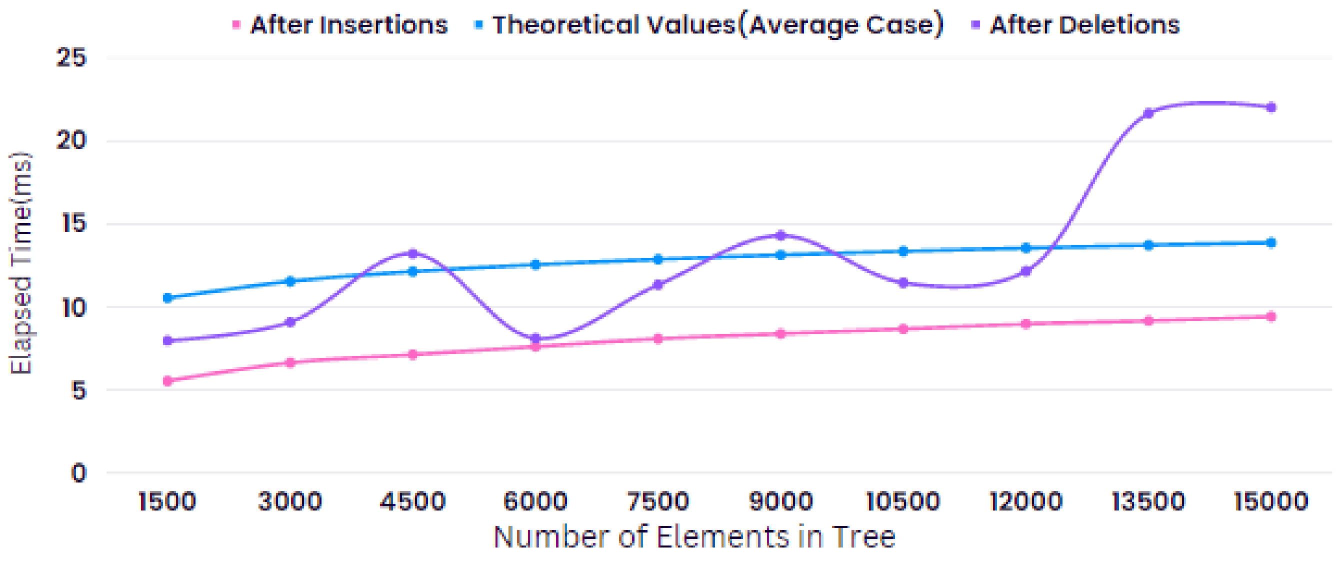
Part e) What is the maximum and minimum height of a binary search tree that contains n items ?

maximum height = n
minimum height = $log_2(n+1))$

QUESTION 3

● Interpret and compare your empirical results with the theoretical ones. Explain any differences between the empirical and theoretical results, if any.

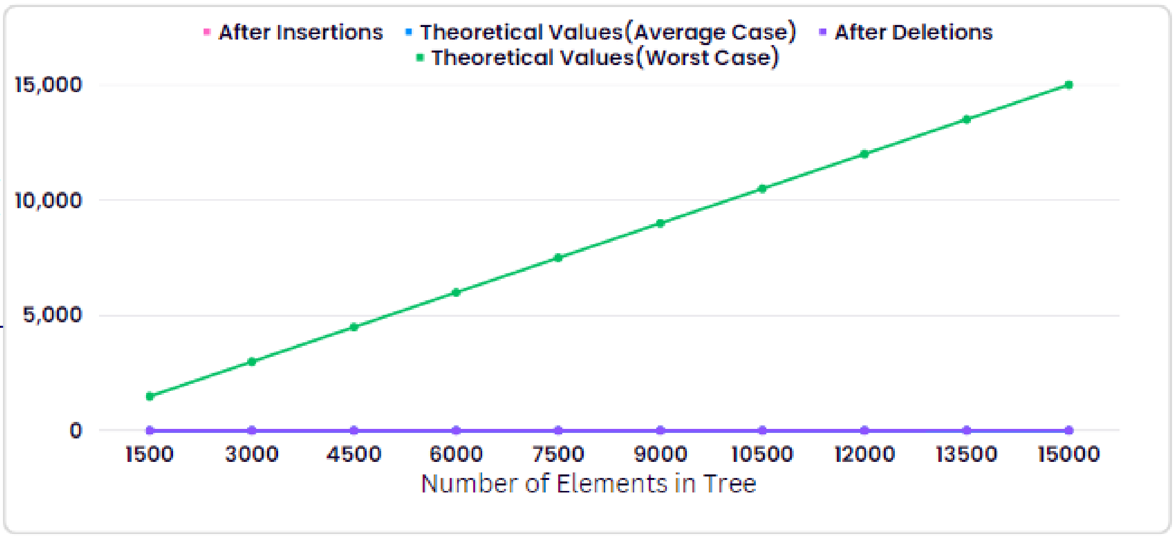
● How would the time complexity of your program change if you inserted sorted numbers into it instead of randomly generated numbers?



Average Case for insertion and deletion operations is : O(logn)
Worst Case for insertion and deletion operations is : O(n)

For both of these case it is expected to have incremental graph. Which means if node count increments the expected elapsed time should also be incremented.

Average case is expected to have less increase compare to worst case. For average case, for n = 1500 time complexity is: 10.5507467854 and for n = 3000 time complexity is 11.5507467854. However, for worst case, n = 1500 time complexity is: 1500 and for n = 3000 time complexity is : 3000. There is a sharp proportional difference in worst case comparing to the changes in average case. If I drew the worst case in my graph it would look like the graph below which I found hard to make inferences, since O(n) is increasing too sharp comparing to my results and average case. This is why I will inference my comment from the graph above.



For the second question, if we insert sorted numbers instead of randomly generated numbers, our binary search tree will be unbalanced and all the nodes will be in the left or right subtree, which force us to visit all nodes and the time complexity will be O(n) which is very high comparing to O(logn). However, in my empirical results, I found similar results to average case time complexity which is O(logn) and which makes them more efficient, because the randomly generated nodes will be more likely to locate balanced comparing to sorted arrays.