CS202- HW 1 Report

Halil Arda Özongun- 22202709- Section 1

20 October 2024

Question 1

a) Which of the following sequences can represent the pre-order traversal of a binary search tree with 12 nodes? For the correct sequence, draw the corresponding binary search tree and write the post-order and in-order traversals of the tree.

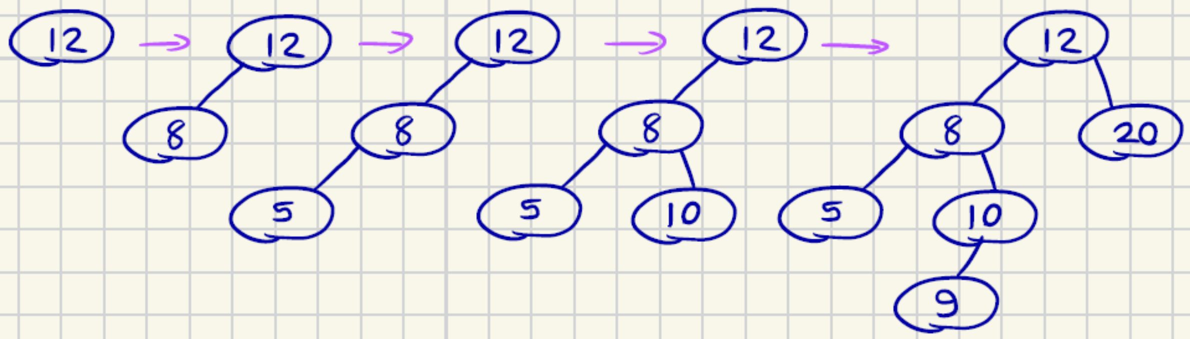
< 12, 8, 5, 10, 9, 20, 14, 15, 18, 13, 16, 19>

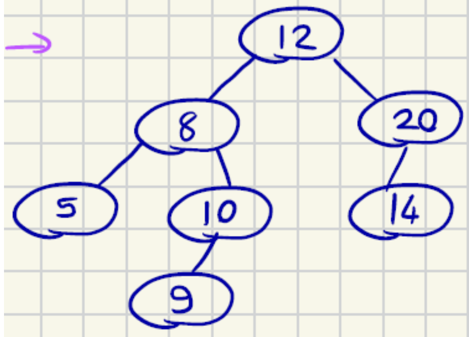
< 12, 8, 5, 10, 9, 20, 14, 13, 18, 15, 16, 19 >

Question 1

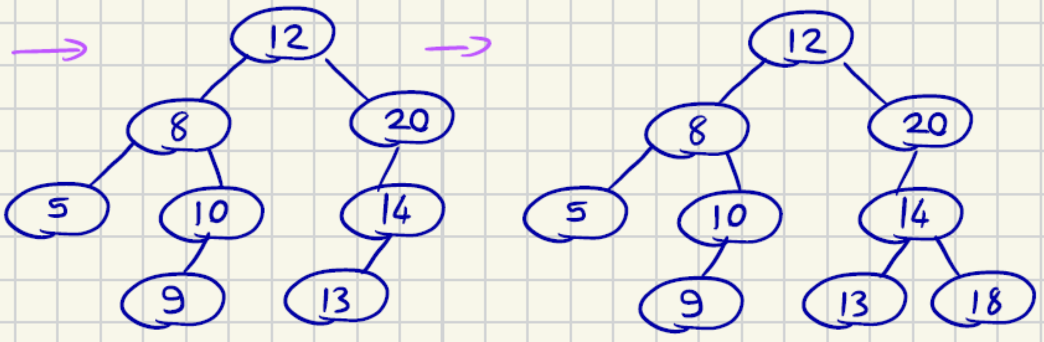
a- Second one is correct since, while you are adding you can follow the root→ left→right structure.

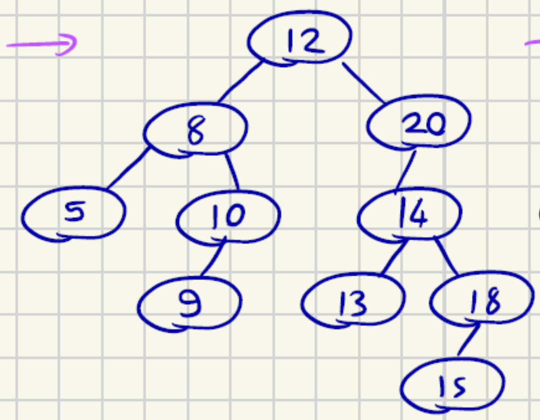
First one is wrong: It adds 13 after 18 and 15, but since it is in the left subtree of 14 it must be before

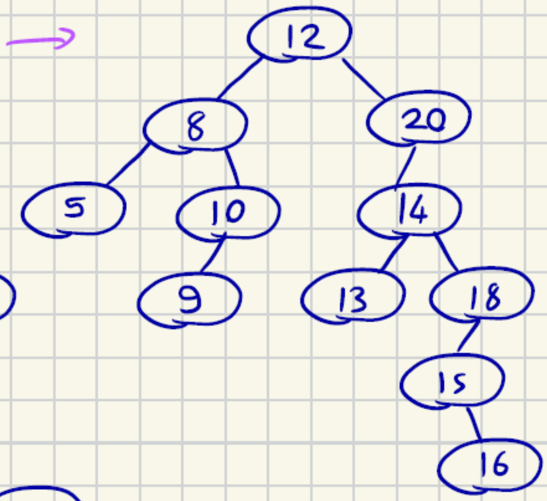


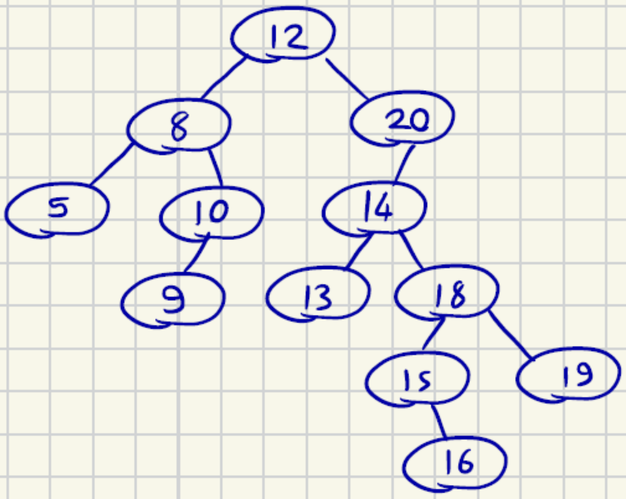


First one wants to insert 15 now. But since it is pre-order it must finish left of root 14. But 15 is bigger than 14 so it should be inserted after 13.







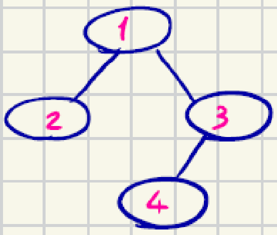


Post order Traversal: < 5, 9, 10, 8, 13, 16, 15, 19, 18, 14, 20, 12 >

In order Traversal: < 5, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19, 20 >

b) In how many different ways can we insert the elements a < b < c < d into an empty binary search tree to form the given structure? Explain.

b-

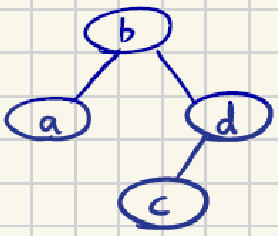


Node number 2 is the smallest one since it is on left side of root, and every other node is bigger than root

Also Node 3 is bigger than Node 1 since it is on right side of root and bigger than Node 4 since it is on left of Node 3

So: Node 3 > Node 4 > Node 1 > Node 2

d > c > b > a



Since it is root, you have to insert b first. Also d is before c since c is a child of it.

We can't say anything about a since it is on left so it is independent from c and d

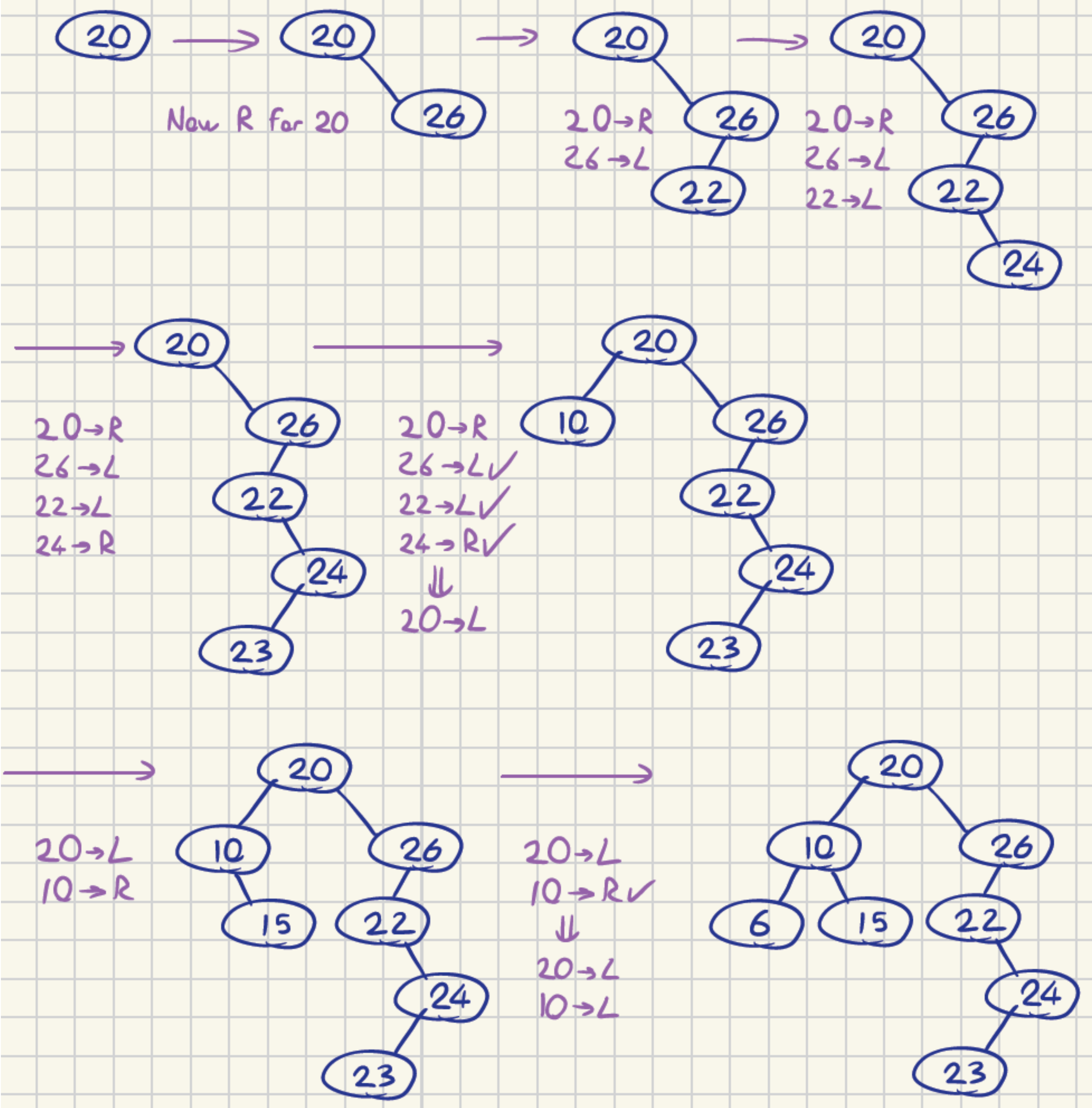
So
$$\left.\begin{array}{l} b \to a \to d \to c \\ b \to d \to a \to c \\ b \to d \to c \to a \end{array}\right\} 3 \text{ ways}$$

c) What is the 7th key in the preorder traversal of a binary search tree if its postorder traversal is as follows?

Postorder: 5, 6, 15, 10, 23, 24, 22, 26, 20

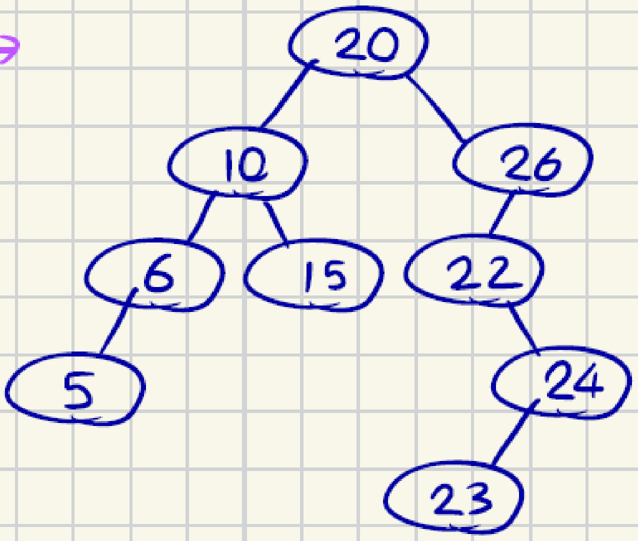
c- Post order: 5, 6, 15, 10, 23, 24, 22, 26, 20 → L, R, Root

Reading from end to beginning: And going as Root, Right, Left



$\longrightarrow$

20 → L
10 → L
6 → L



After inserting 5 all recursive calls ends.

Preorder is Root → L → R

So it's preorder : < 20, 10, 6, 5, 15, 26, 22, 24, 23>

So 7th element is 22

d) Suppose we have inserted 100 keys, numbered 1 to 100, into a binary search tree. The order in which the keys were inserted is unknown, but all possible insertion orders are equally likely. What is the probability that keys 4 and 9 will be compared during the insertion process, given that the first key inserted is 43? Explain.

When 43 is inserted first, the remaining numbers in 1-42 range are placed on left subtree. Our numbers 4 and 9 both are in this side. So we care about on this new tree which is left child of 43.

Let x be the number we are going to insert, it is in [1, 42]. Suppose this number is not between 4 and 9. Since the number is bigger or smaller than both, if we add 4 and 9, they will be on same subtree so they still be able to compared. So we need a number which is greater than one and smaller than other.

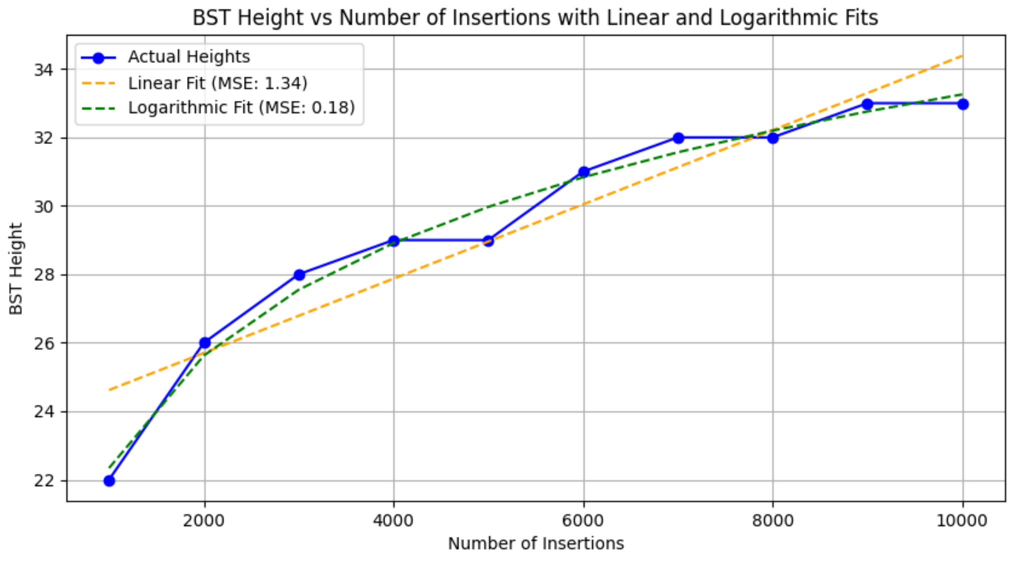
$9 > x > 4 \Rightarrow x \in \{5, 6, 7, 8\}$ So, if we choose a number from this set before we choose both 4 and 9, they can't come across. Choosing other numbers have no effect

We wonder the probability to encounter so it is same as choosing 4 and 9 first from $\{4, 5, 6, 7, 8, 9\}$

$$\frac{S(\{4,9\})}{S(\{4,5,6,7,8,9\})} = \frac{2}{6} = \frac{1}{3}$$

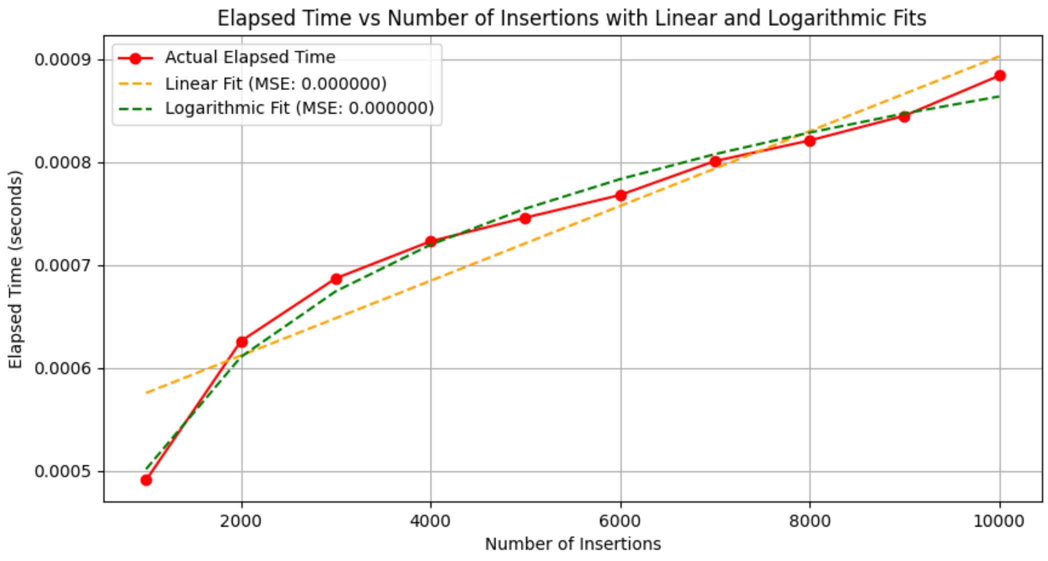
Question 3

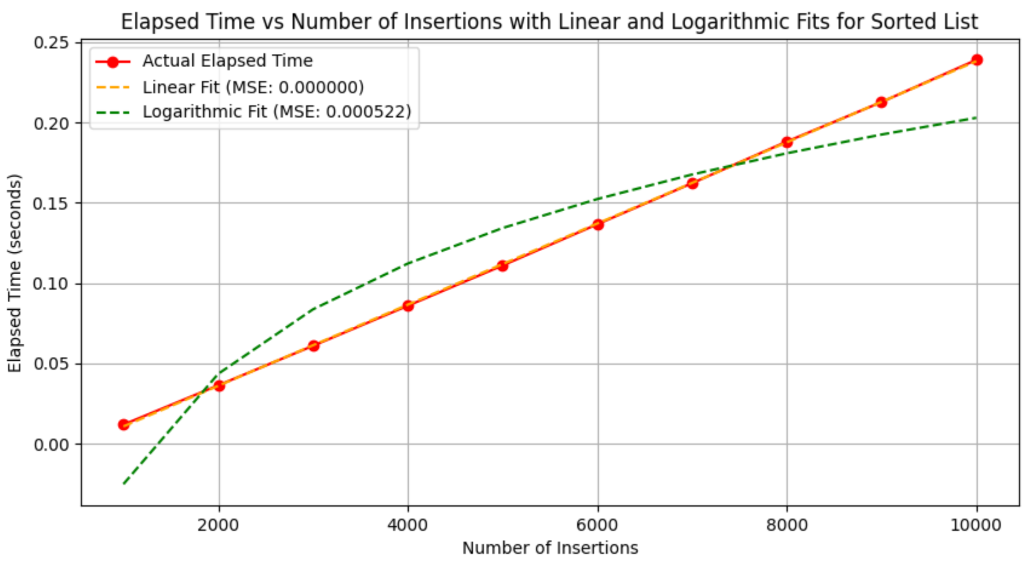
A-



This function is similar to the logarithmic function. This is because the insertion into the BST is O(height). Each time a number is inserted, we follow a path from root to leaf to find where the incoming number is, and this path is equal to height. This height, if the tree is balanced, satisfies the equality logn = height where n is number of nodes. This is similar to the result we get in the experiment. Still, it is not exactly logarithmic because the tree is not really balanced. Randomly added numbers get a result that approximates the balanced tree, but it is not perfect. This is the reason for the fluctuations.

B-





This graph, like the above graph, is supposed to be logarithmic with n (when n is the number of nodes). The reason for that, the element which will be added be directly proportional to the height as it will be added to the leaves. Since it is proportional to the height and since the height is approximately equal to log n, it is a logarithmic function. There are some fluctuations here due to the fact that the tree is not a balanced tree.

If the list were sorted, we would add each element to the under of the bottom element, so the time would be increased cumulatively for each addition. To get to the node we want to add, we would have to go as far as height, but this time height = n (number of nodes). So adding sorted would be O(h) = O(n) for each node. (Just considered for addind a node)

C-

First we have to sort the list of nodes that will be added. Then, we need to put the node in the middle of the list to the root. After that, we need to add the middle elements of the left and right halves of the list, adding and adding recursively until the list is complete. We can think of it like this, the root should be such a number that it divides the numbers to its left and right into equal parts so that tree gets balanced.

If we compare this solution with other solutions, the sorted version would be the worst because O(n) for each insertion. The random version would be close to O(logn) but worse than O(logn) since it is not perfectly balanced tree. In the optimal case we have developed, the case is O(logn) but this solution is faster than the random solution since it is perfectly balanced tree. The bounds given here are not tightest bounds.