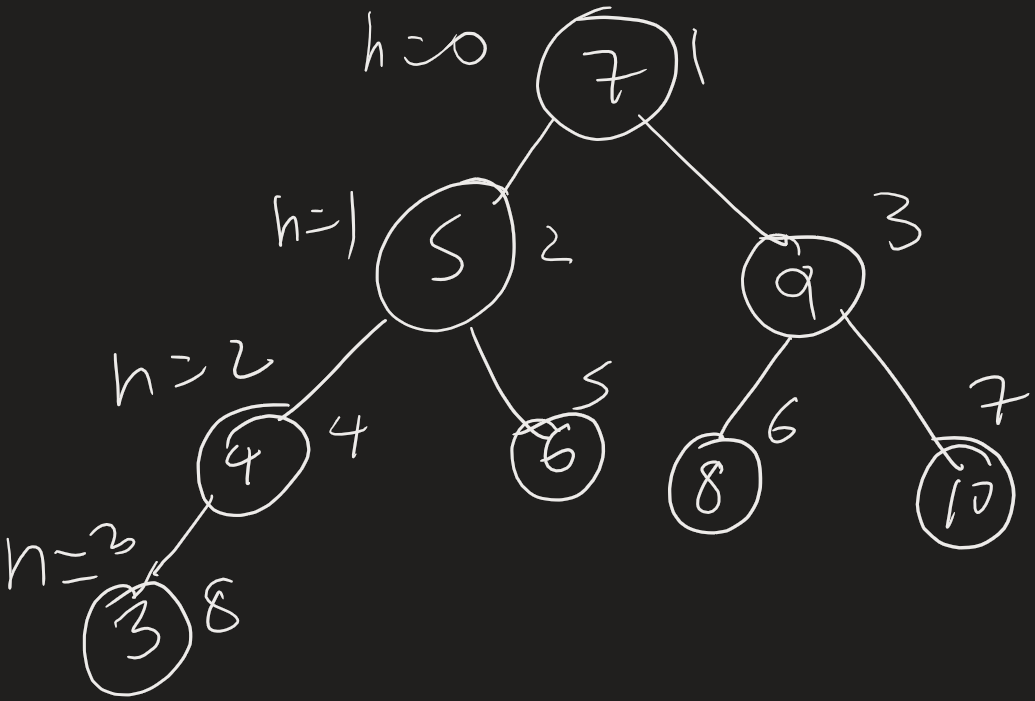
1. (\*) = the number of nodes in a complete tree is at least 2h

Base case:

Let h = 3 for 2h  = 23 = 8 nodes



Therefore (\*) works for h = 3

Induction step:

Let’s assume (\*) works for h = k

Let h = k +1

The two sub trees from the root have a height of k, thus they have at least 2k nodes because of the inductive hypothesis. Since there are 2 sub-trees, we can assume the number of nodes for the whole tree would be 2k + 2k = 2k+1

Therefore, the number of nodes in a complete tree is at least 2h.

1. A) This runs on O(log n) because t = n and t is continuously being divided by 2 until it reaches 1.

B) This runs in O(n \* log n) because the for loop runs on O(n) time and the nested while loop runs on O(log n) and they are multiplied since they are nested.

C) This has a run time of O((log n)2) because both loops run on log n and they are multiplied because the loops are nested.

D) This has a run time of O(1) because the condition of the while loop never be true, so the loop never runs.

3) There would need to be two more functions, one function counts the total number of nodes and the other finds the median. The first function will use any traversal a use a counter to find the total number of nodes. The function returns the counter at the end. This function is called in the find median function. The find median function takes the result from the other function and does (count % 2). If the result is 1 then we know there are odd number of nodes and there is just one median in the middle. Then the function does in order traversal (count / 2) + 1 times. The node it’s at when the loop ends is the median. If the result is 0 then the function will have to take the middle two nodes and take the average of their keys. The loop runs the same amount of times but saves the nodes when count is count / 2 and (count / 2) +1. Take the average of the keys of those nodes and that is your median.