**Assignment 5.2**

**Question 2 :**

Inorder to create a tree which allows us to search operation in log n time we need to create a balance binary search tree using all the nodes of the given unbalanced tree, whose height will be log n

Algorithm :

1. Sort the elements of the tree in the increasing order by doing inorder traversal on the binary search tree.
2. Once the elements are sorted we can create a balanced binary search tree by

Inserting middle element into the tree first.

1. We recursively call the same function on the left and right part part of the sorted array. This allows to insert in the order which creates an balanced binary search tree.

Time Complexity :

Sorting elements which involves inorder traversal takes linear time as it involves visiting all nodes in the tree recursively. As mentioned in class, we can think of it as edge traversals. Each edge in the binary search tree is traversed exactly twice.

Number of edges in the binary search tree with n nodes = (n -1)

Therefore maximum time taken = 2(n -1) = 2n- 2

Time taken by the recursive call to insert the mid element of the array to create a balanced tree = 2T(n/2) + C

T(n) = T(n/4) +2C

= T(n/8) +3C

:

:

* T(n) = T(1) + i\*C -> Equation 1

By using above equations this can be generalized as

T(n) = T(n/2^i) + i\*C -> Equation 2

By using Equation 1 and 2, we have:

n = 1 => i = log2 n -> Equation 3

2i

Substituting Equation 3 in 1, we have:

T(n) = T(1) +C ( log2 n)

Since T(1) is running time when n=1 , it is considered as some constant Q

* T(n) = Q+ C ( log2 n)

We can ignore constants as they are insignificant when n is large

Total time complexity T(n) = log2 n + 2n -2

By ignoring smaller term we have ,

Time complexity **T(n) = O(n)**

**Space Complexity :**

Since this involves first sorting the algorithm using inorder traversal, the space complexity of this traversal is = h (height of the tree) as max size of the recursive stack can be “h” at any point of time

Space taken to create another balanced binary search tree using the existing nodes = n

Total space complexity = h + n

Ignoring the smaller terms, we have

**Space complexity = O(n)**