**Week 8: Data Structures**

1. **insert\_bst**:

* **Guide**:
  + For inserting into a BST, if the new key is less than the current node’s key, go to the left subtree. Otherwise, go to the right subtree. Follow this process recursively.
* **Pseudocode**:
* FUNCTION insert\_bst(root, key):  
   IF root is None:  
   RETURN TreeNode(key)  
   ELSE IF key < root.val:  
   root.left = insert\_bst(root.left, key)  
   ELSE:  
   root.right = insert\_bst(root.right, key)  
   RETURN root

1. **find\_max\_heap**:

* **Guide**:
  + In a max-heap represented as an array, the maximum element is always at the root, which is the first element of the array.
* **Pseudocode**:
* FUNCTION find\_max\_heap(arr):  
   RETURN arr[0]

1. **is\_full\_binary\_tree**

* **Guide**:
  + A full binary tree is defined as a binary tree in which every node has either two or zero children.
  + Every node in the tree will either have both a left child and a right child or have no children at all.
  + This function will recursively check every node to see if it adheres to this definition.
* **Pseudocode**:
* FUNCTION is\_full\_binary\_tree(root):  
   IF root does not exist:  
   RETURN True  
   IF root exists but has no children:  
   RETURN True  
   IF root has both left and right children:  
   CHECK if both left and right subtrees are also full binary trees  
   IF they are, RETURN True  
   RETURN False

1. **get\_tree\_height**

* **Guide**:
  + The height of a binary tree is the length of the longest path from the root node to any leaf node.
  + This function computes the height by recursively calculating the height of both the left and right subtrees. It will then return the maximum of these two heights plus one to account for the current node.
  + An empty tree or a null node has a height of 0.
* **Pseudocode**:
* FUNCTION get\_tree\_height(root):  
   IF root does not exist:  
   RETURN 0  
   COMPUTE height of left subtree  
   COMPUTE height of right subtree  
   RETURN the maximum of the two heights plus 1