# Data Structures and Algorithms

# INFO 6205

# Homework 5

# Due: February 17, 2019

Put all your java, compiled class files and documentation files into a zip file named Homework5.zip and submit it via the drop box on the blackboard before the END of due date. Put your name on all .java files. There will be a short quiz on this homework.

1. What is the Balanced Tree, Complete Tree and Non-Complete Tree?

**Ans:**

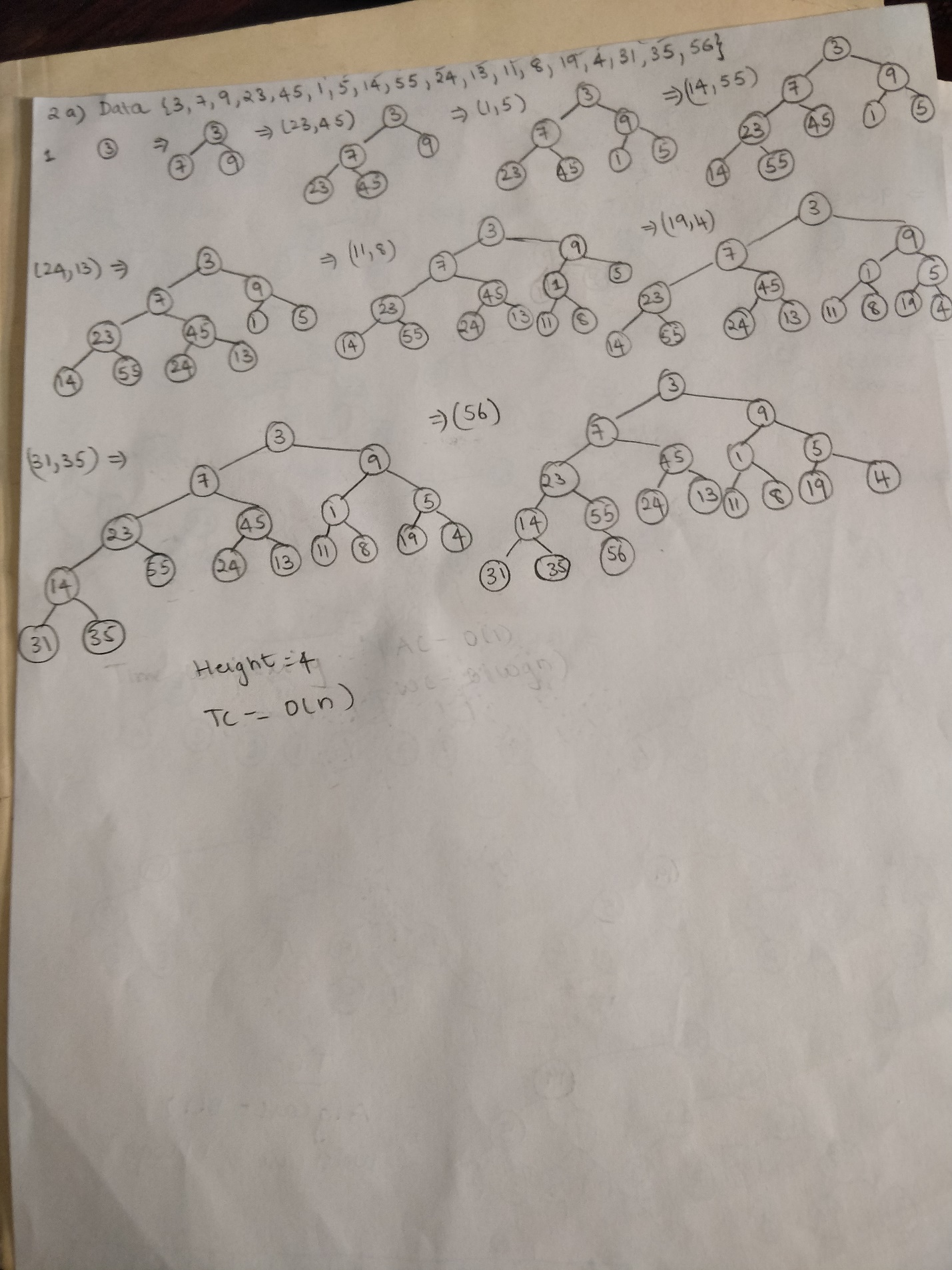
A **balanced** binary tree is the binary tree where the depth of the two subtrees of every node never differ by more than 1.

A **complete** binary tree is a binary tree whose all levels except the last level are filled completely and all the leaves in the last level are all to the left side.

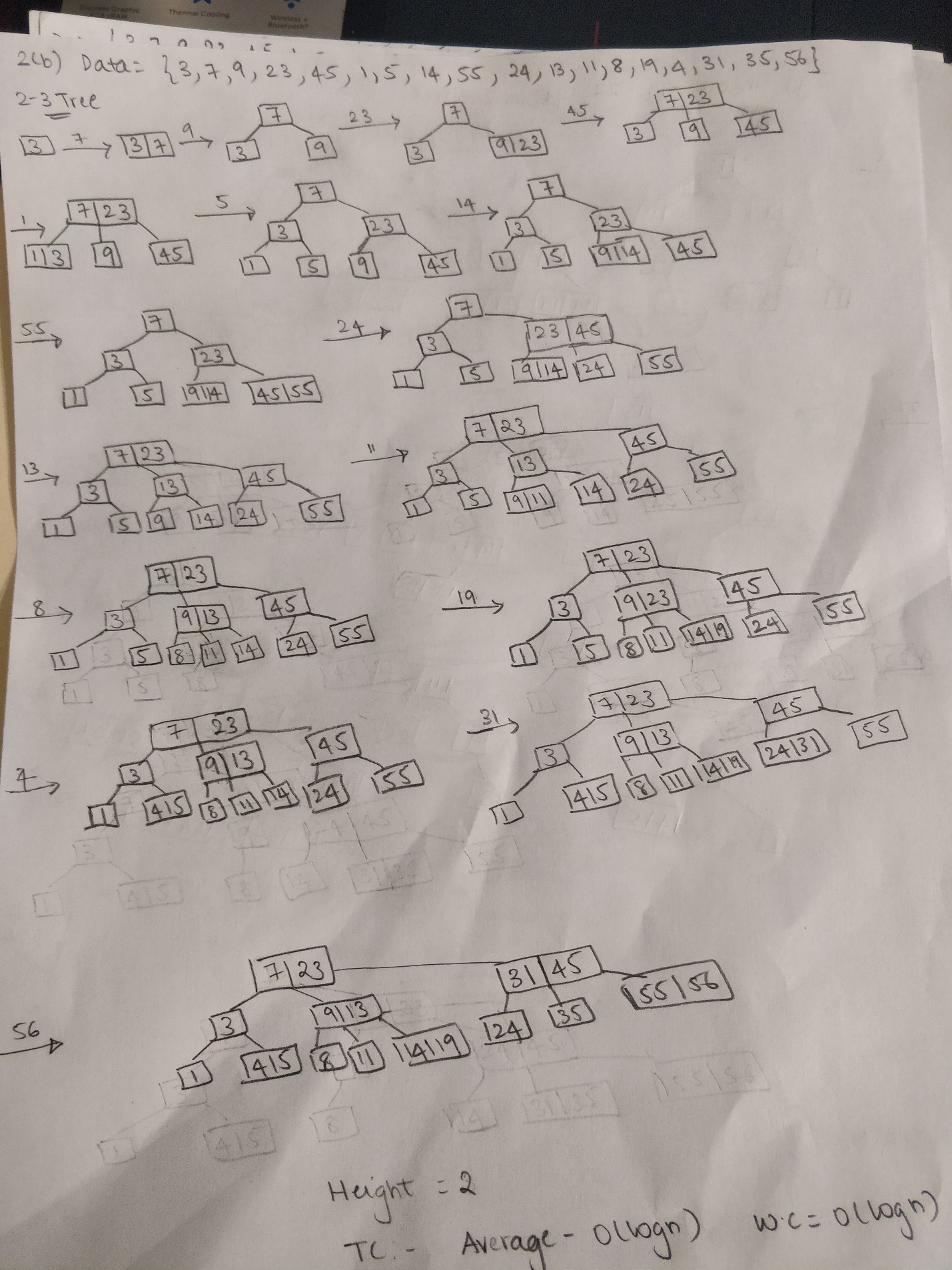
A binary tree where the difference between the heights of left and right subtree cannot be more than one for all nodes is a **non-complete tree**

2. Consider following data: {3,7,9,23,45,1,5,14,55,24,13,11,8,19,4,31,35,56}

1. Construct Binary Tree

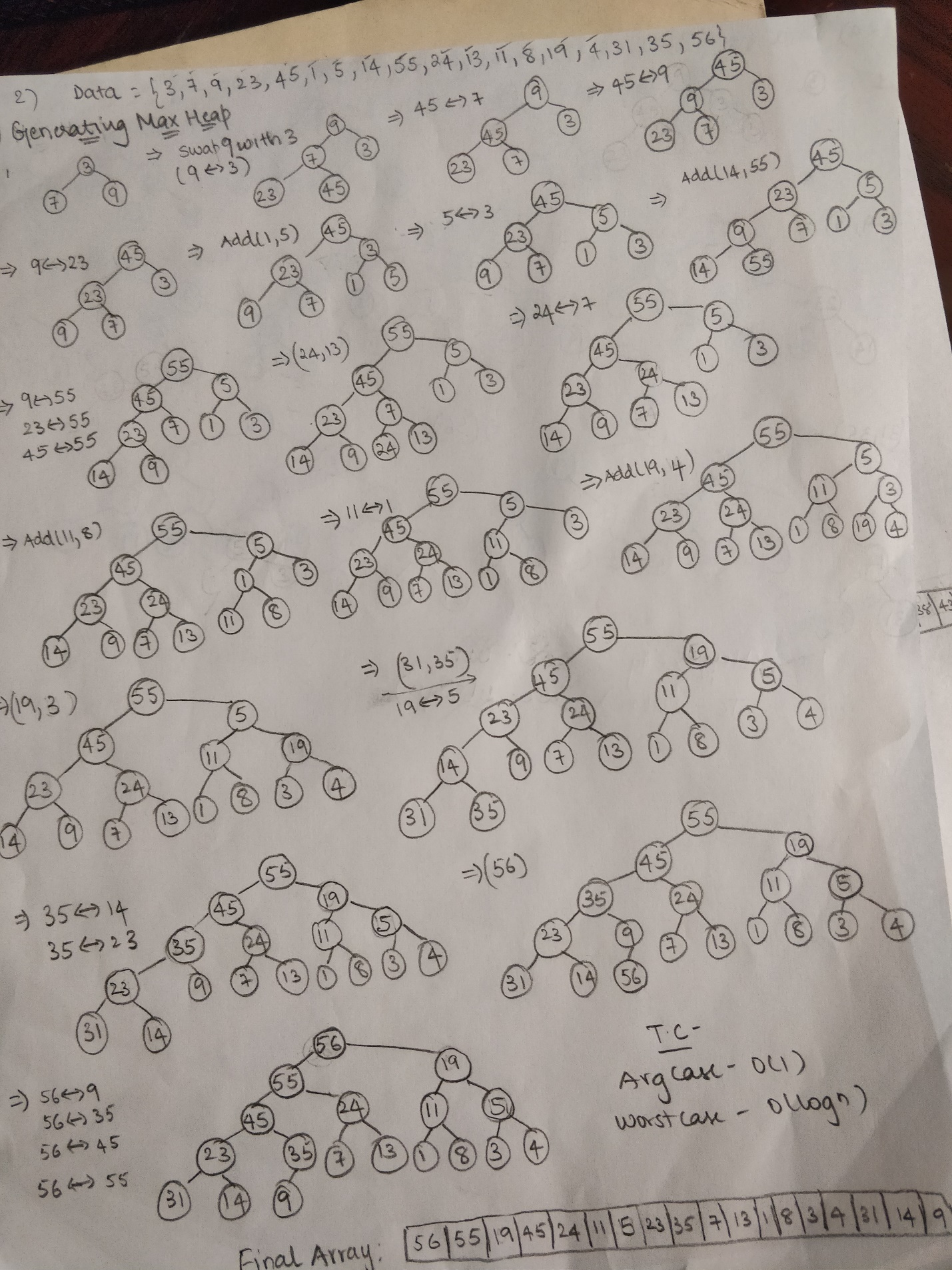


1. Construct 2-3 Tree



c) Construct 2-3-4 Tree

d) Construct Binary Heap Tree



e) What is Time complexity of each case, and Why would you use one versus the other

(**Check drawings**)

**Time Complexity:**

**Binary Tree: O(n)**

**2-3 Tree: O(log n)**

**2-3-4 Tree: O(log n)**

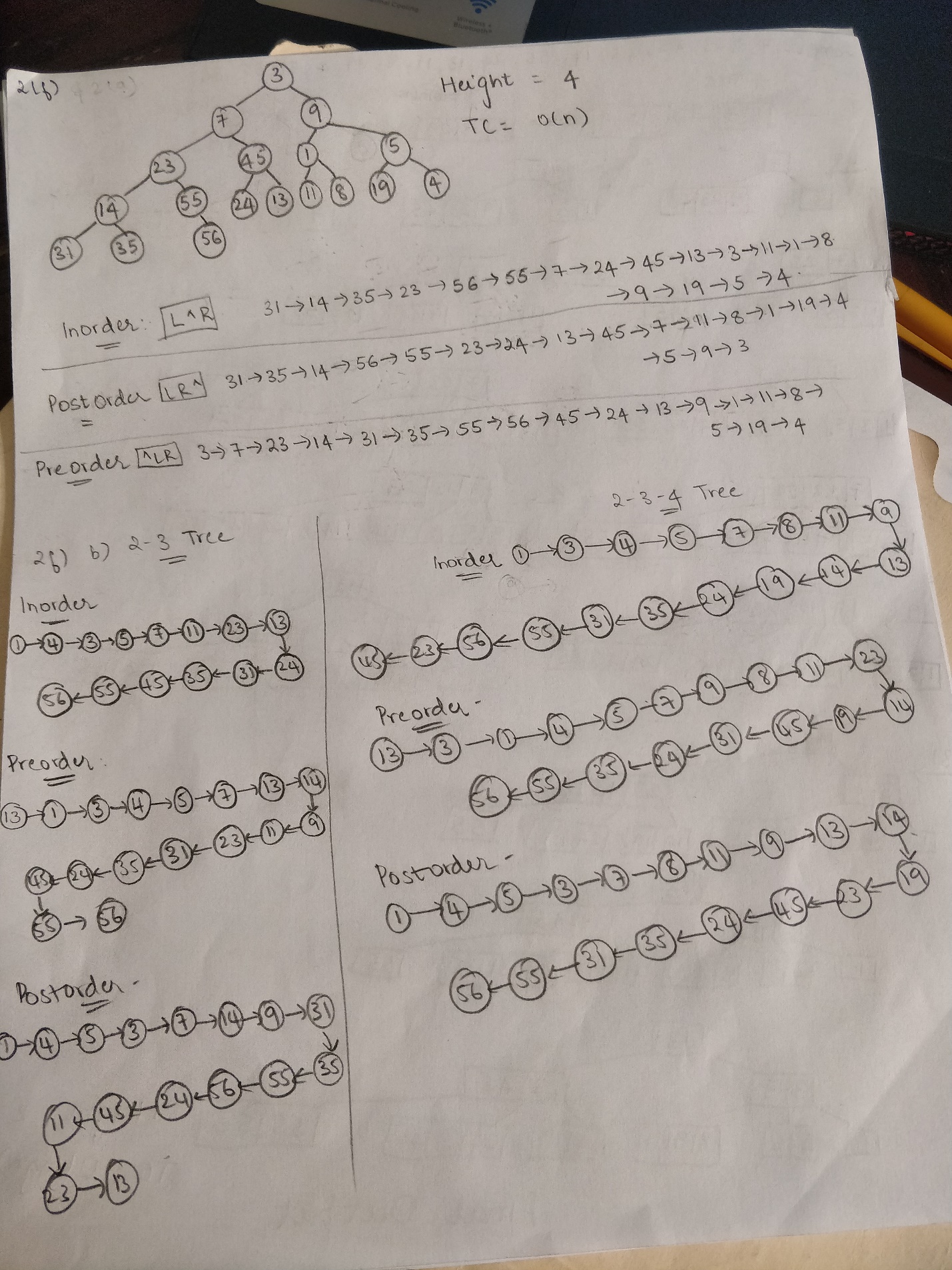
**Unlike Binary tree 2-3 and 2-3-4 tree are balanced tree**

**Reasons to use either tree:**

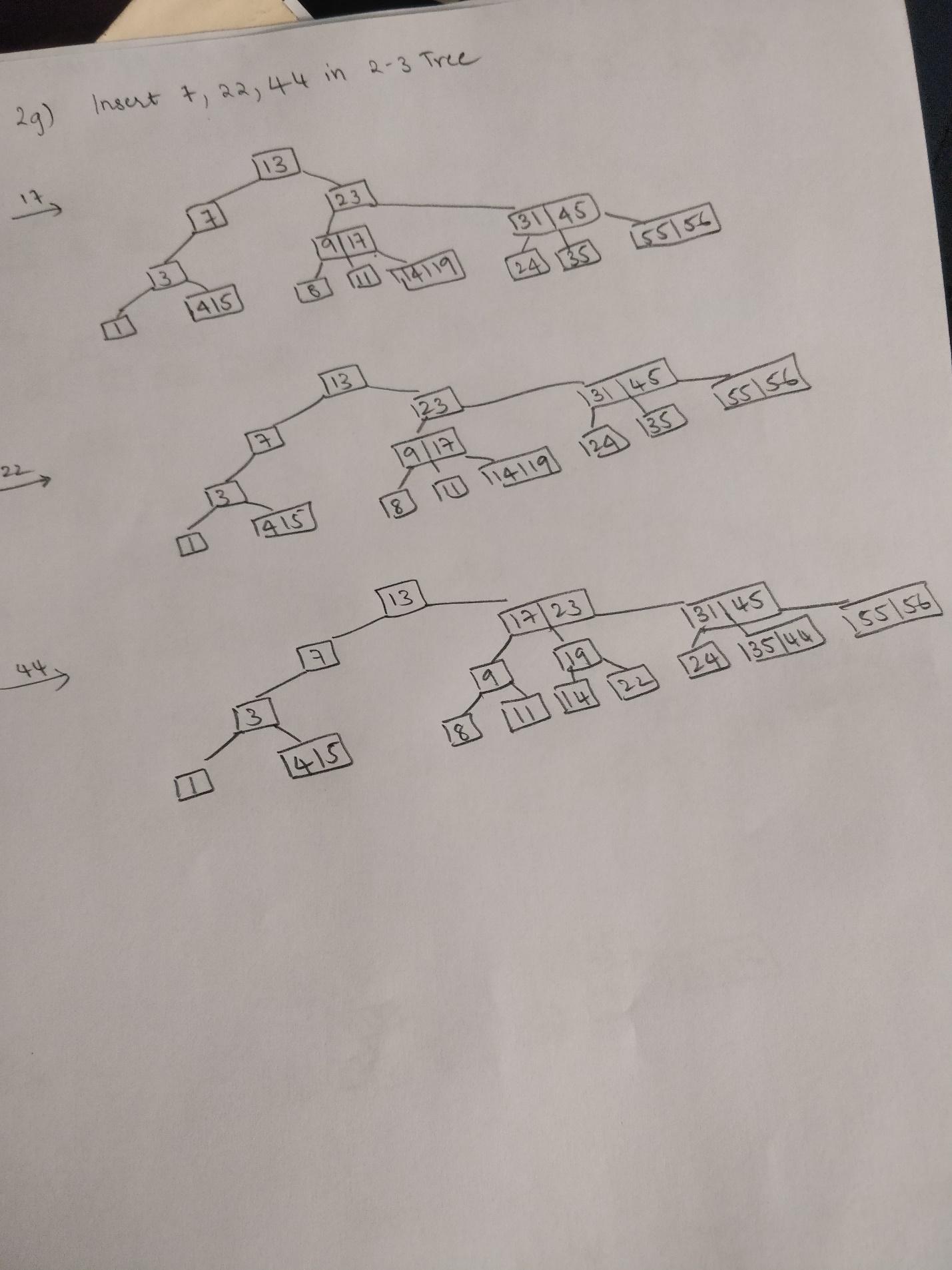
**1. Binary Tree: Easy to insert data but data traversal has the worst possible time complexity based on the number of nodes**

**2. 2-3 and 2-3-4 Trees: Best time complexity from the mentioned trees. Easy for traversal however insertion and deletion can be complicated**

f) Perform Inorder, Preorder, and Postorder traversal for (a), (b), (c)



g) Insert 17, 22, 44 in (b)



h) Delete 24, 19 in (a), (b), (c)

i) What is the Height of (a), (b), (c)?

j) Write Java code for Search and Insert in (a), (b), (c)

k) Write Java code for DeleteMin() and Rank() Algorithms for (a), provide an example

3. Class Record is described below. Write Java code to build Binary Tree, 2-3 Tree and B-Tree

where “key” is the value in data presented in problem-2.

public class Record {

private int key

private Node leftNode;

private Node rightNode;

public Record(int key, Node leftNode, Node rightNode) {

this.key = key;

this.leftNode = leftNode;

this.rightNode = rightNode;

}

public Record(int key){

this.key = key;

}

public int getKey() {

return key;

}

public Node getLeftNode() {

return leftNode;

}

public Node getRightNode() {

return rightNode;

}

public void setKey(int key) {

this.key = key;

}

public void setLeftNode(Node leftNode) {

this.leftNode = leftNode;

}

public void setRightNode(Node rightNode) {

this.rightNode = rightNode;

}

}