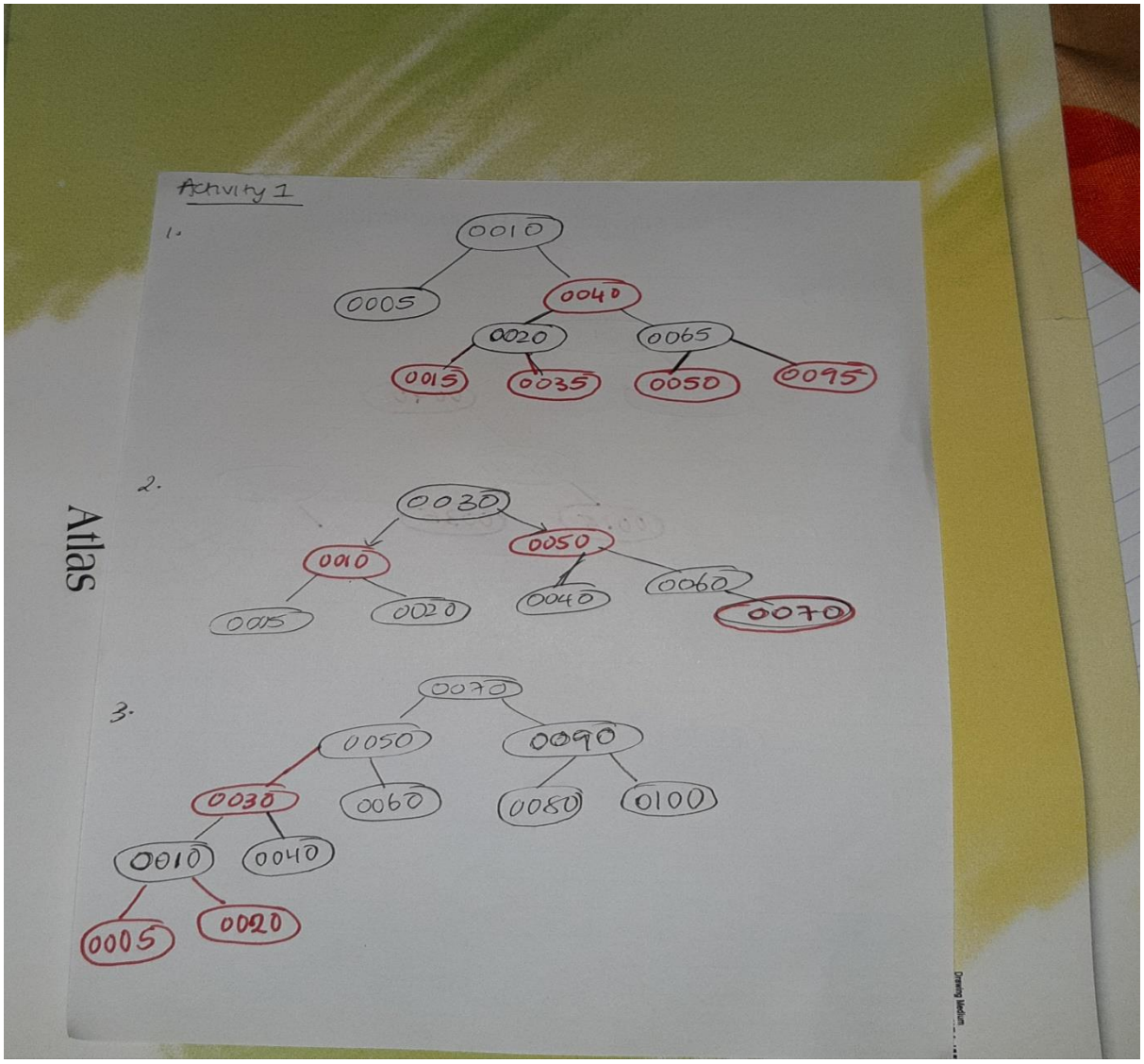
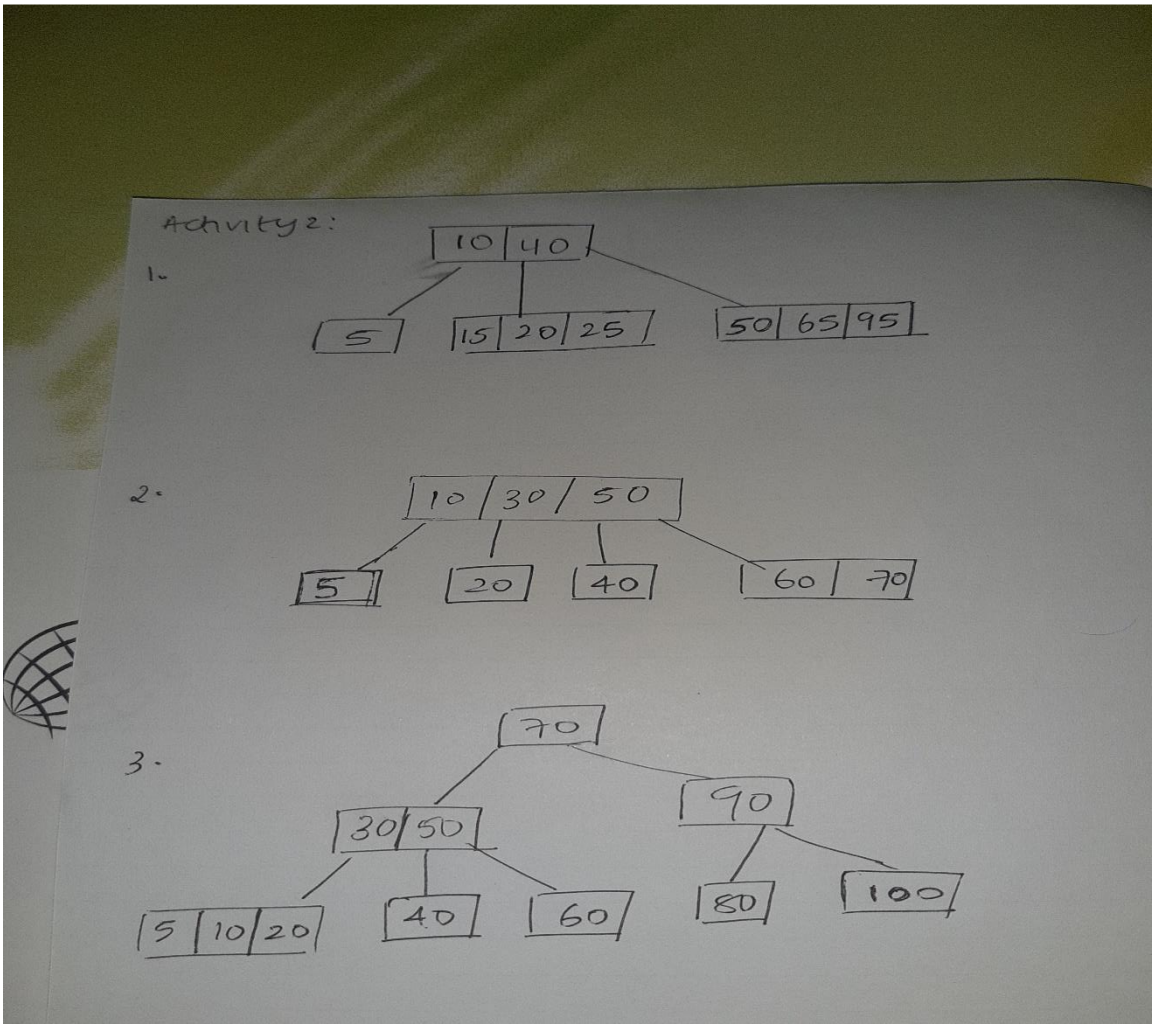
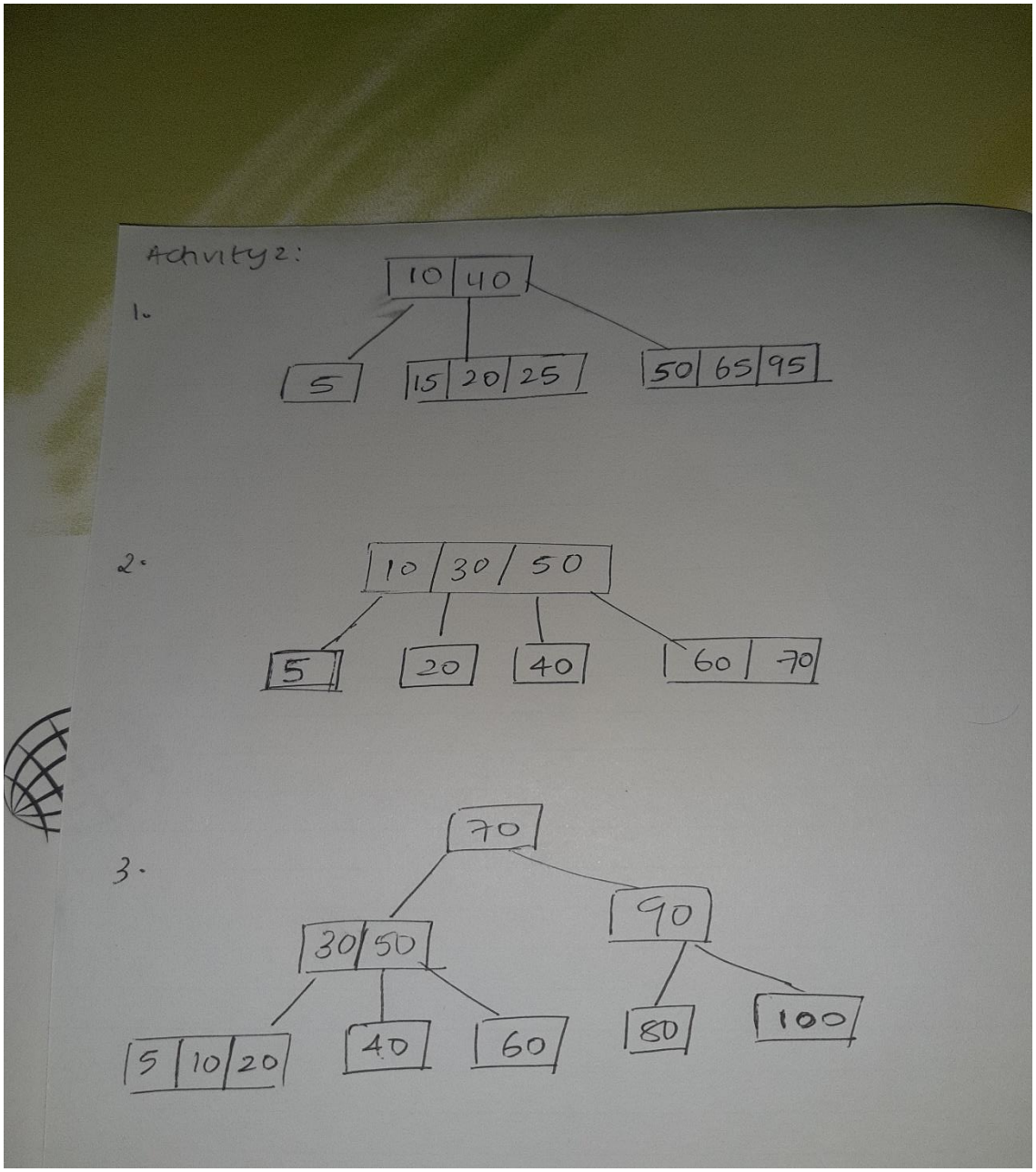
Activity 1


Activity 1
1.
0010
0005
0040
0020
0065
0015
0035
0050
0095
2.
0030
0010
0050
0005
0020
0040
0060
0070
3.
0070
0050
0090
0030
0060
0080
0100
0010
0040
0005
0020
Atlas

Activity 2


Activity 2:
1.
10 40
5
15 20 25
50 65 95
2.
10 30 50
5
20
40
60 70
3.
70
30 50
90
5 10 20
40
60
80
100

Activity 3


Activity 2:
1.
10 40
5
15 20 25
50 65 95
2.
10 30 50
5
20
40
60 70
3.
70
30 50
90
5 10 20
40
60
80
100

Activity 4

- **Compare the heights of the resultant trees – how do they compare with a Binary Search Tree (BST) for the same input values?**

Both the Red-Black trees and the Binary Search Tree are same in terms of their heights . They are considered to have a greater height compared to 2-3-4 Trees and B – Tree . It is because the nodes in the 2-3-4 Trees and B-Trees allow for more than just two other nodes.

Furthermore , the smaller height of the 2-3-4 Trees is because of its nodes that accommodate up to 3 values at a time. Hence, this clearly shows that the height of the 2-3-4 Trees is definitely less than Binary Search Trees.

The B-Trees height vary between the 2-3-4 Trees and that of Red-Black Trees as their heights are less than the Binary Search Trees .The B-Trees can hold more than a value at a time compared to binary search tree, the 2-3-4 Trees , Red-Black Trees and the B-Trees.

- **Compare the complexity of the algorithms, how much work would be required for the main operations: insert | find | delete? Compare this to BST.**

TYPE OF TREES	**INSERT**	**FIND**	**DELETE**
Binary Search Trees	**Best : O(1)** **Average : O(log n)** **Worst : O(log n)**	**Best: O(1)** **Average: O(log n)** **Worst: O(log n)**	**Best: O(1)** **Average: O(log n)** **Worst: O(log n)**
Red-Black Trees	**Best : O(1)** **Average : O(log n)** **Worst : O(log n)**	**Best: O(1)** **Average: O(log n)** **Worst: O(log n)**	**Best: O(1)** **Average: O(log n)** **Worst: O(log n)**
2-3-4 Trees	**Best : O(1)** **Average : O(log n)** **Worst : O(log n)**	**Best: O(1)** **Average: O(log n)** **Worst: O(log n)**	**Best: O(1)** **Average: O(log n)** **Worst: O(log n)**
B-Trees	**Best : O(1)** **Average : O(log n)** **Worst : O(log n)**	**Best: O(1)** **Average: O(log n)** **Worst: O(log n)**	**Best: O(1)** **Average: O(log n)** **Worst: O(log n)**

- **Compare the understand ability of the algorithms, which would be easier to implement?**

  The algorithm for the red-black trees and the binary search trees will definitely be more easier compared to both the 2-3-4 trees and B- trees as there are 3 values to be held in a node. Thus , this may cause a problem while doing the implementation.

- **Describe how an in-order traversal would work on each type of tree.**

1. **Red-Black Trees**

The in-order traversal would work the same as the Binary Search Trees , that is the left-most element first, then the parent ,then the leftmost of the last untraversed level, the parent of that node and then , the right child of that node and finally the right child of the first parent traversed.

2. **2-3-4 Trees**

First ,the last level of the tree and the leftmost node will be traversed, then the parent of that node will be traversed , followed by any child connected to that node , then to the right rightmost element of the parent node, then, till the rightmost child of the parent node . if the parent node has a parent , that will be traversed , moving to the last level of the leftmost untraversed level. And, this will repeat.

3. **B-Trees**

   First of all the nodes have to be traversed from left to right through their elements.

   Assuming a three level B-Tree , the leftmost node of the third level will be traversed , followed by the left element of the parent of this node , then the middle node of the parent node , then the rightmost element in the parent node and then the right node of the parent node. It will be followed by the left element of the root and then, the last node on the right of the root will be traversed right like the leftmost node on the second level.

REFERENCE:

Seenarain.B.(2019).DSA/P09/practical 09.pdf