CSCI 532 – Algorithm Design
Assignment 5

Name: Badarika Namineni **CWID:** 50233279

Question 1: What is the difference between the binary-search-tree property and the min-heap property? Can the min-heap property be used to print out the keys of an n-node tree in sorted order in O(n) time?

Solution:

Binary search tree is an efficient structure for storing an ordered data. The Binary tree is having a root node which is greater than or equal to it left child node and less than or equal to its right child node which means all the nodes on the left sub-tree are smaller and all the nodes on the right sub-tree are larger. Binary node can have utmost of 2 child nodes.

Min-heap is having a root node which is smaller than or equal to 2 of its child nodes. It only talks about the root node having being the smaller node than the children nodes but does not decide on which child is larger or sorted.

It is impossible to know which sub-tree contains the next smallest number by using min-heap property.

Heap sort takes Ω (n logn) in the worst case to sort the array and **it can't be used to print out the keys of an n-node** tree in sorted order in O(n) time. Whereas it is clear from binary search tree that left node is smaller and right node is larger and we get an idea of where to go next to fetch the next smallest number.

Question 2: Give recursive algorithms that perform preorder and post order tree walks in Θ (n) time on a tree of n nodes.

Solution:
Considering x as a root node
For preorder tree walk:

```
Preorder(x)
return root node value
Traverse left subtree by calling Preorder(x)
Traverse right subtree by calling Preorder(x)
```

It has running time of Θ (n)

For post order tree walk:

```
Postorder(x)
Traverse left subtree by calling Preorder(x)
Traverse right subtree by calling Preorder(x)
Return root node value
```

It also has running time of Θ (n)

Question 3: Suppose that we have numbers between 1 and 1000 in a binary search tree and want to search for the number 363. Which of the following sequences could not be the sequence of nodes examined?

a. 2, 252, 401, 398, 330, 344, 397, 363.

b. 924, 220, 911, 244, 898, 258, 362, 363.

c. 925, 202, 911, 240, 912, 245, 363.

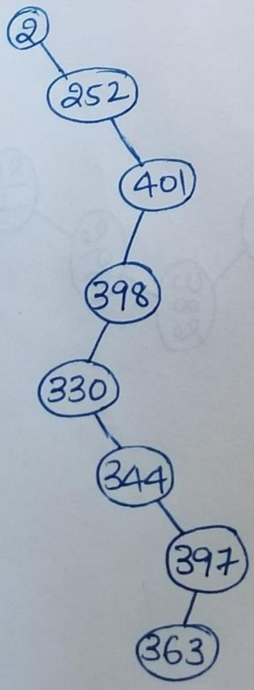
d. 2, 399, 387, 219, 266, 382, 381, 278, 363.

e. 935, 278, 347, 621, 299, 392, 358, 363.

Solution:

a. 2, 252, 401, 398, 330, 344, 397, 363

a) The sequence is drawn as
2, 252, 401, 398, 330, 344, 397, 363

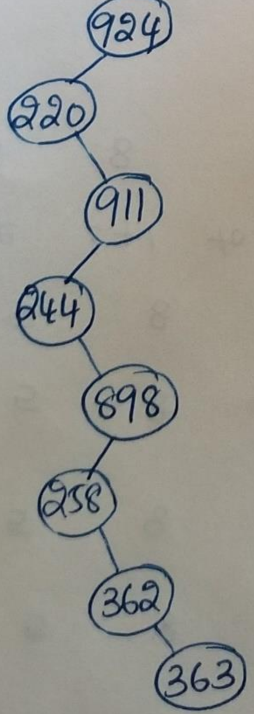


- Binary search tree is drawn from the sequence

- 363 can be examined from this sequence of nodes.

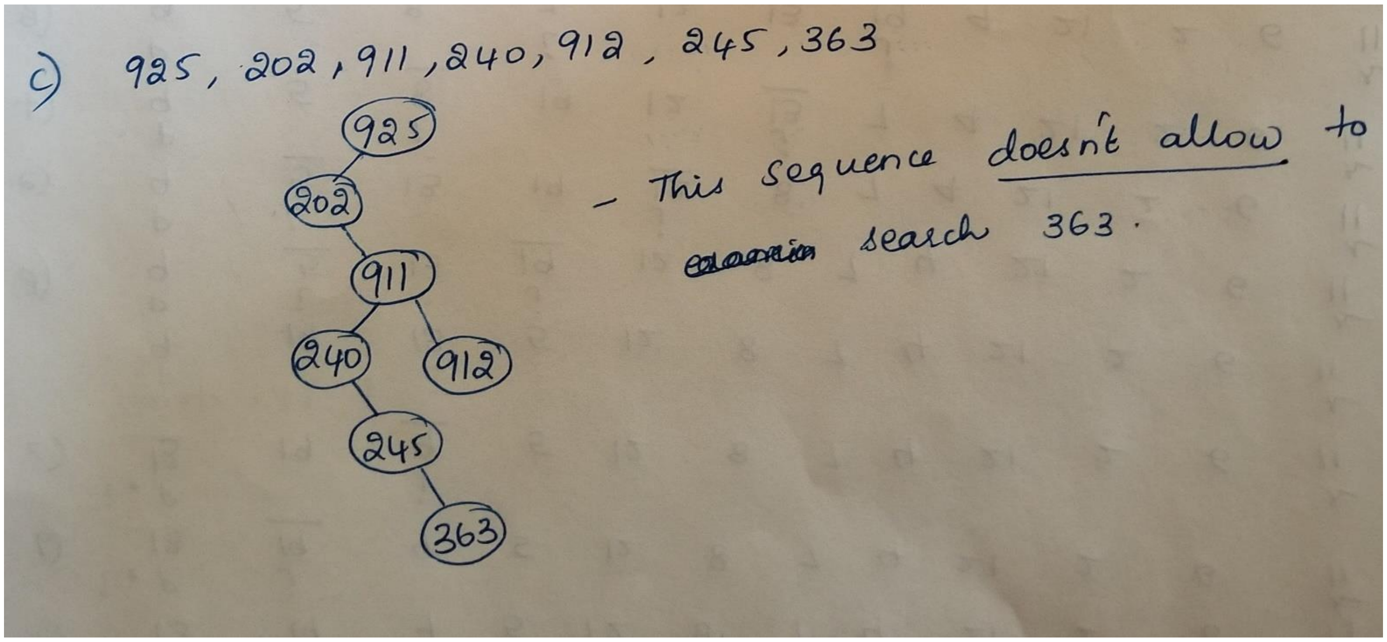
b. 924, 220, 911, 244, 898, 258, 362, 363.

b) 924, 220, 911, 244, 898, 258, 362, 363



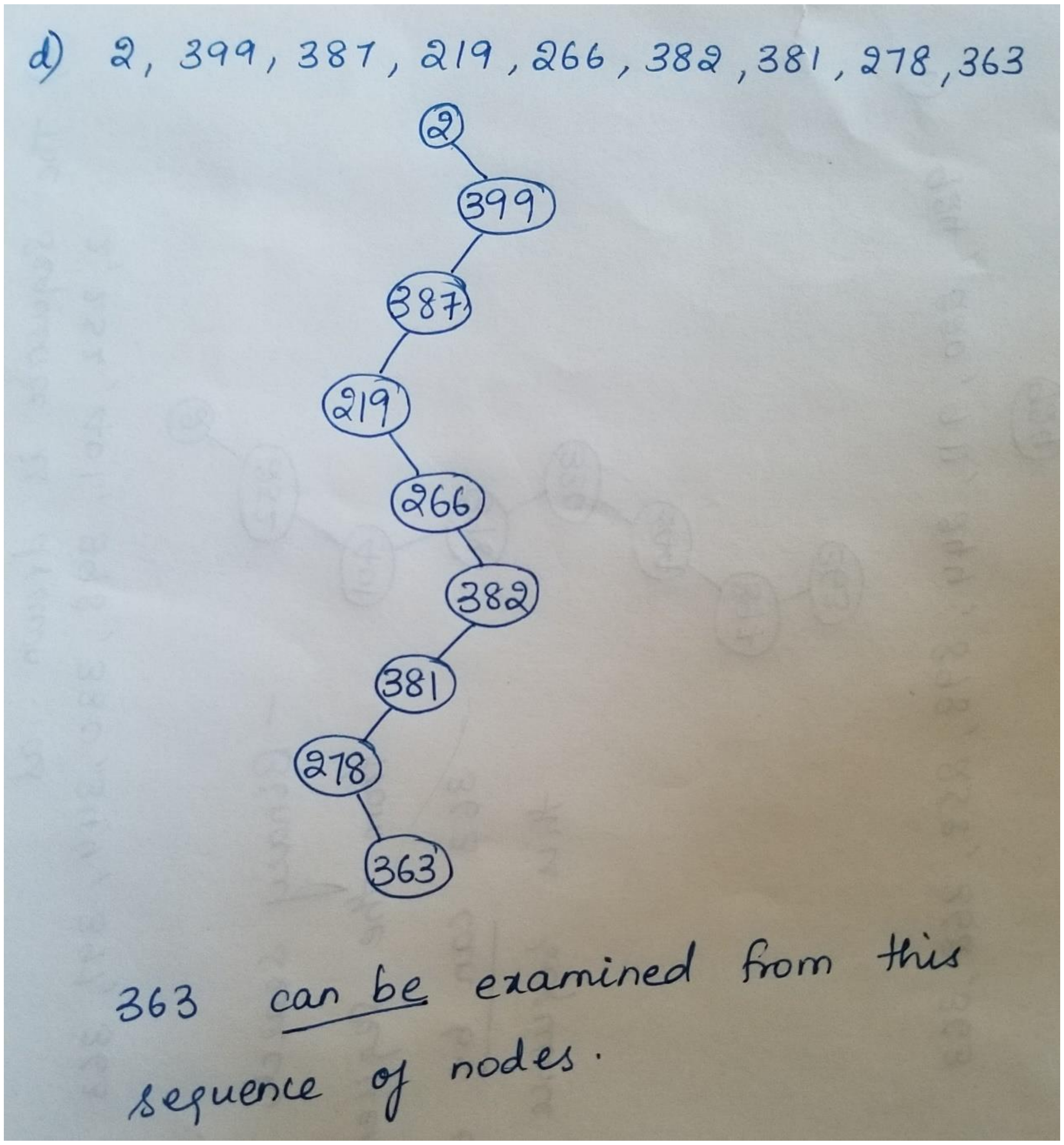
- 363 can be examined from this sequence of nodes.

c. 925, 202, 911, 240, 912, 245, 363


c) 925, 202, 911, 240, 912, 245, 363

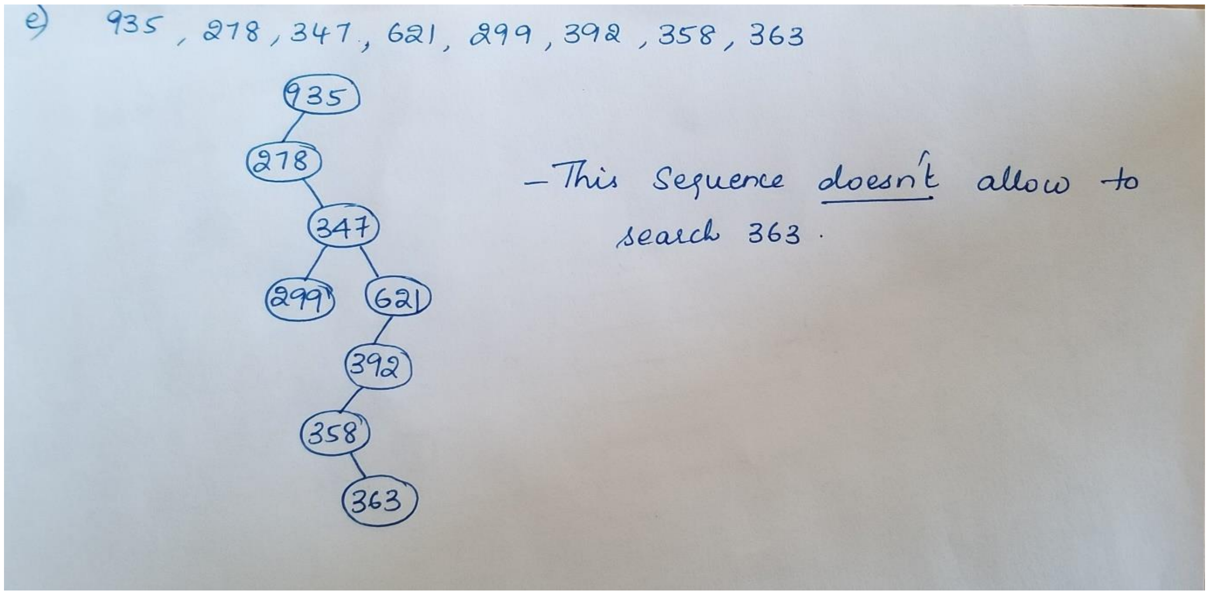
– This sequence doesn't allow to search 363.

d. 2, 399, 387, 219, 266, 382, 381, 278, 363


d) 2, 399, 387, 219, 266, 382, 381, 278, 363

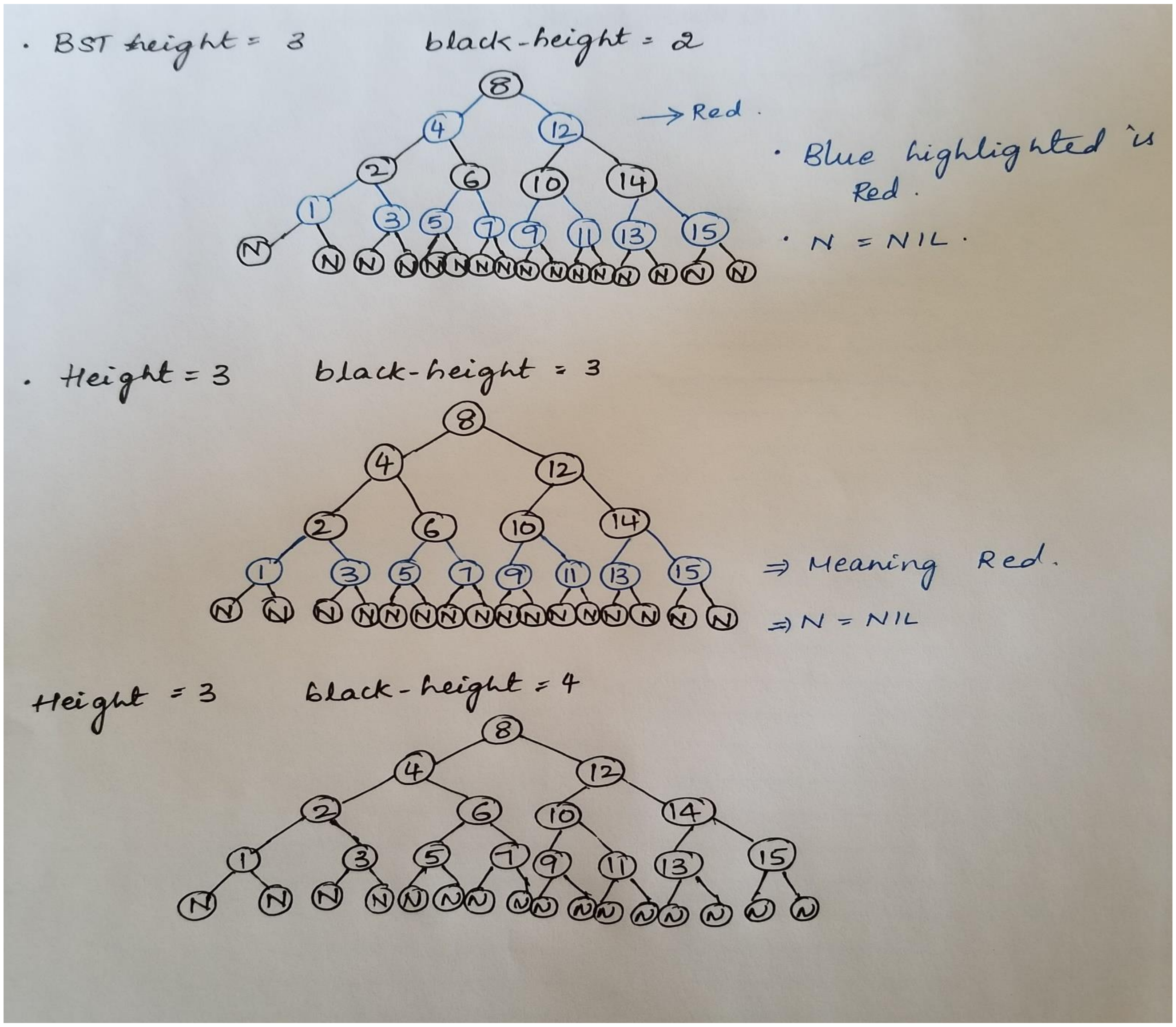
363 can be examined from this sequence of nodes.

e. 935, 278, 347, 621, 299, 392, 358, 363


e) 935, 278, 347, 621, 299, 392, 358, 363
935
278
347
299
621
392
358
363
– This Sequence doesn't allow to search 363.

Question 4: In the style of Figure 13.1(a), draw the complete binary search tree of height 3 on the keys {1, 2, ..., 15}. Add the NIL leaves and color the nodes in three different ways such that the black-heights of the resulting red-black trees are 2, 3, and 4.

Solution:



Question 5: Let us define a relaxed red-black tree as a binary search tree that satisfies red- black properties 1, 3, 4, and 5. In other words, the root may be either red or black. Consider a relaxed red-black tree T whose root is red. If we color the root of T black but make no other changes to T, is the resulting tree a red-black tree?

Solution:

Yes, the resulting tree will still be a red-black tree even after changing the root of T to black. The black heights remain same and since the root has 2 black children, it doesn't violate any property of the re-black tree.

Question 6: Show that the longest simple path from a node x in a red-black tree to a descendant leaf has length at most twice that of the shortest simple path from node x to a descendant leaf.

Solution:

The shortest simple path from any node x will be the black height of the tree with x as a root. Let's consider a Red-Black Tree with black-height of 3. The shortest simple path would be 3 black nodes in line. The longest simple path would be 3 black nodes + 3 red nodes. Every red node must have a black child, and we give every black node a red child to maximize the red children we have on the path.

In total, the length of the shortest possible path for black-height 3, is 3 nodes. The longest possible path is 6 nodes, which is twice, possibly less but can never be more than the number of nodes on the shortest simple path.