Data Structures and Algorithms

Lecture 15

Aniket Basu Roy

BITS Pilani Goa Campus

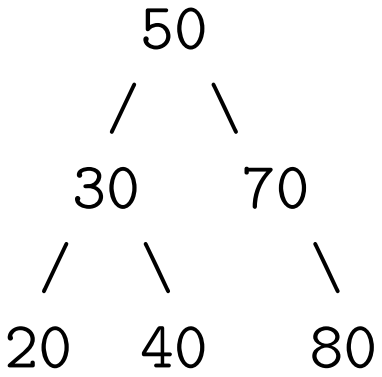
2026-02-18 Wed

Agenda

Data Structures

Binary Search Trees

Binary Search Trees



Left Subtree $<$ Node $<$ Right Subtree

Binary Search Trees

Functions	Time Complexity
Insert(x)	$O(h)$
Delete(x)	$O(h)$
Find(x)	$O(h)$
ListAllElements()	$\Theta(n)$

Deleting from BST

Pseudocode for BST Deletion

```
ALGORITHM: DELETE(root, key)

INPUT:
root - pointer to root of BST,
key - value to delete

OUTPUT:
root - pointer to root of modified BST
```

Pseudocode for BST Deletion

```
1. // Base case: empty tree
2. IF root == NULL THEN
3.     RETURN NULL
4. END IF
5.
6. // Recursive search for the node
7. IF key < root.data THEN
8.     root.left = DELETE(root.left, key)
9. ELSE IF key > root.data THEN
10.     root.right = DELETE(root.right, key)
11. ELSE
12.     // Node found! Now handle three cases
13.
```

```
14.      // Case 1: Leaf node (no children)
15.      IF root.left == NULL AND root.right == NULL
THEN
16.          DELETE root
17.          RETURN NULL
18.
```

```
19.     // Case 2a: Only right child exists
20.     ELSE IF root.left == NULL THEN
21.         temp = root.right
22.         DELETE root
23.         RETURN temp
24.
25.     // Case 2b: Only left child exists
26.     ELSE IF root.right == NULL THEN
27.         temp = root.left
28.         DELETE root
29.         RETURN temp
30.
```

```
31.     // Case 3: Both children exist
32.     ELSE
33.         // Find inorder successor (minimum in
right subtree)
34.         temp = FIND_MIN(root.right)
35.
36.         // Copy successor's value to current node
37.         root.data = temp.data
38.
39.         // Delete the successor
40.         root.right = DELETE(root.right, temp.data)
41.     END IF
42. END IF
43.
44. RETURN root
```

Helper Function: Find Minimum

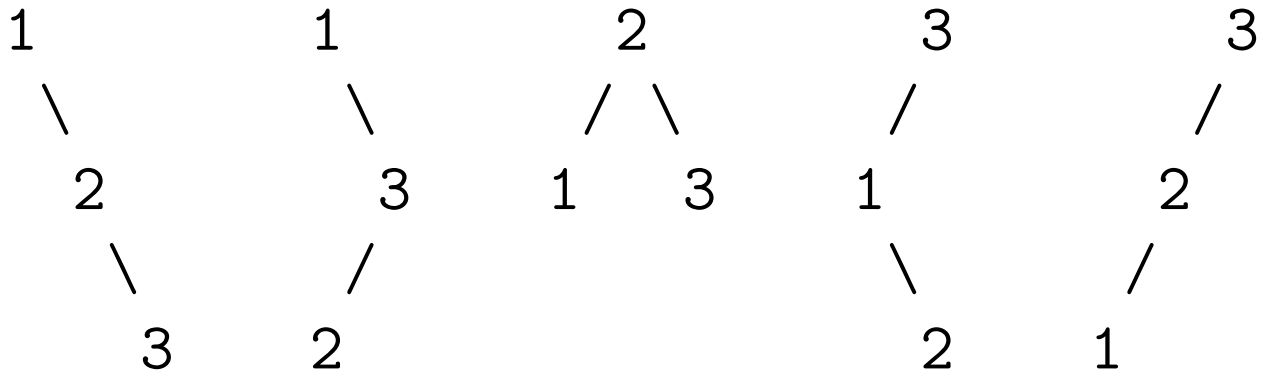
```
ALGORITHM: FIND_MIN(node)
INPUT: node - pointer to a node in BST
OUTPUT: pointer to node with minimum value

1. WHILE node.left != NULL DO
2.     node = node.left
3. END WHILE
4. RETURN node
```

Binary Search Trees

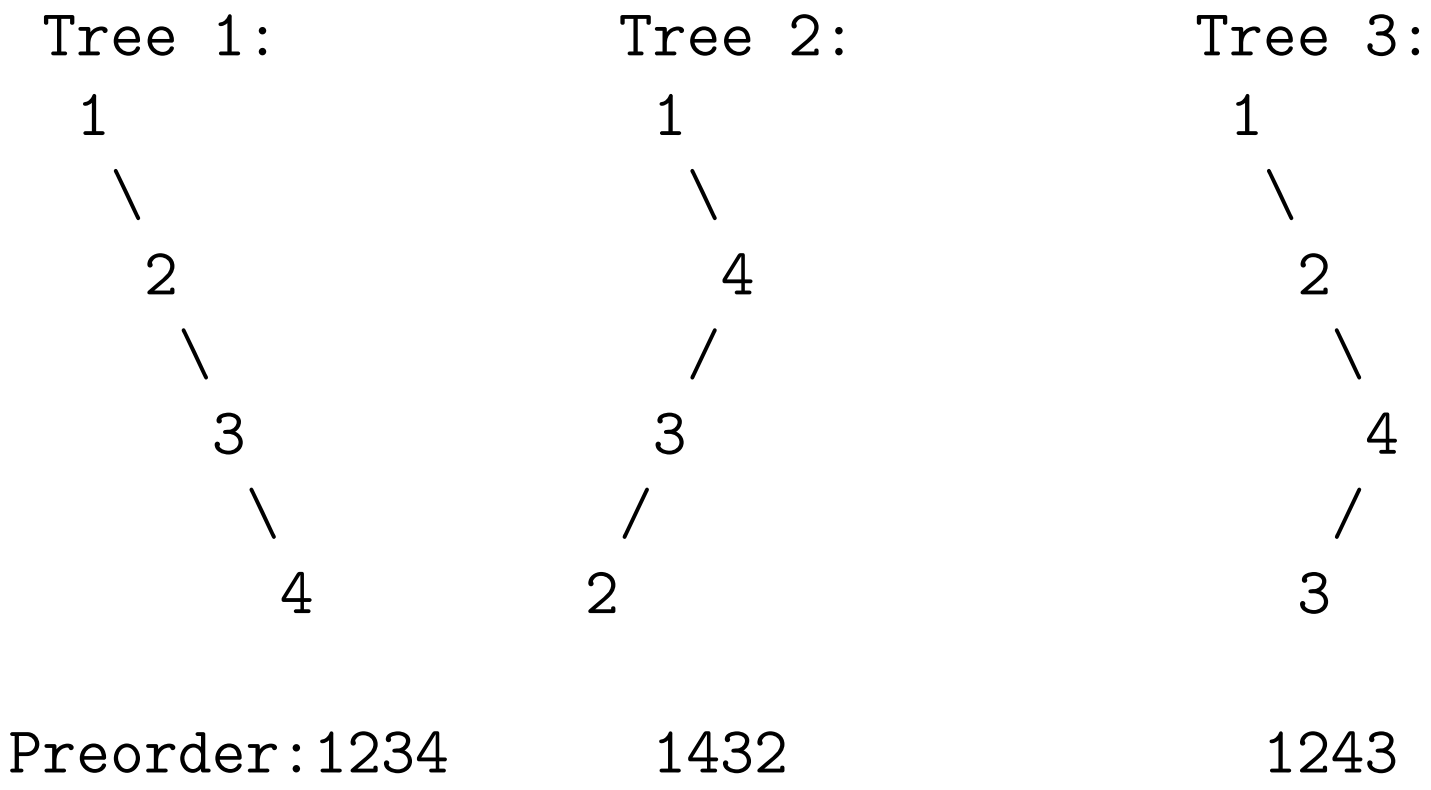
List all elements / Traversals

- Preorder
- Inorder
- Postorder

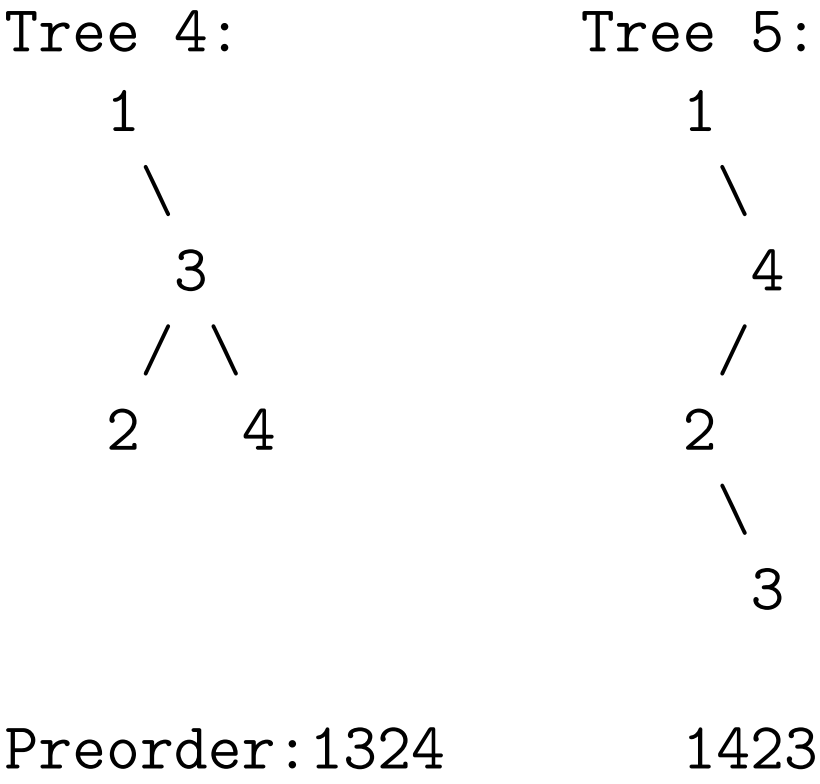
BST with 1, 2, 3



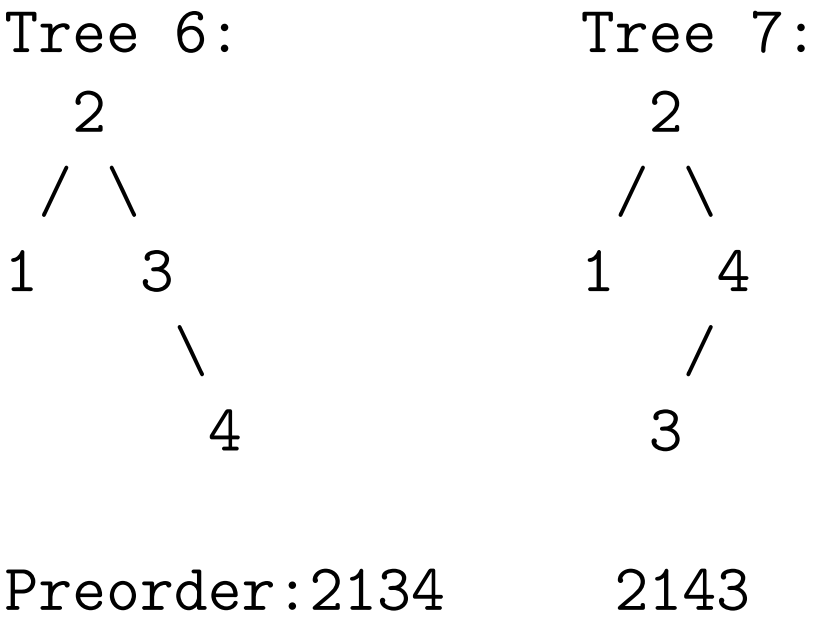
Trees with Root = 1 (5 trees)



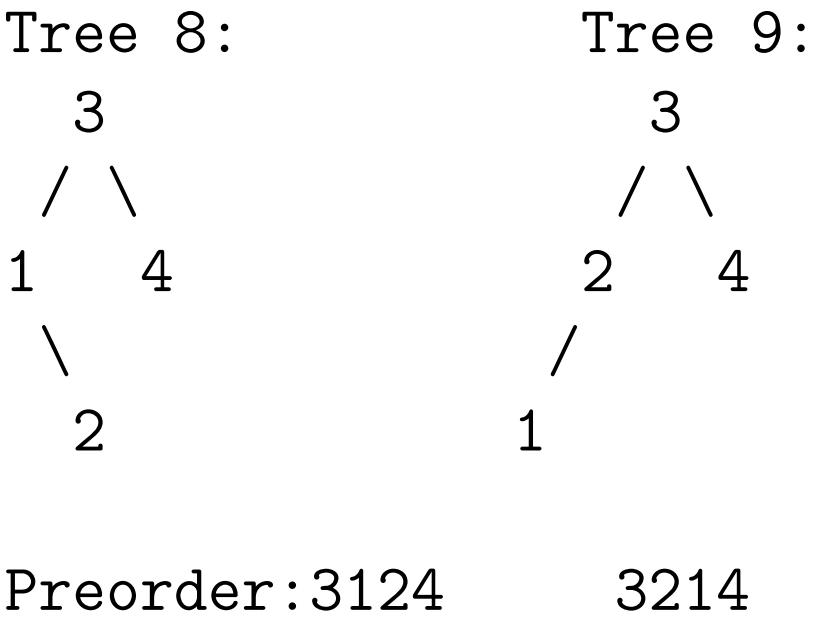
Trees with Root = 1 (5 trees) [contd.]



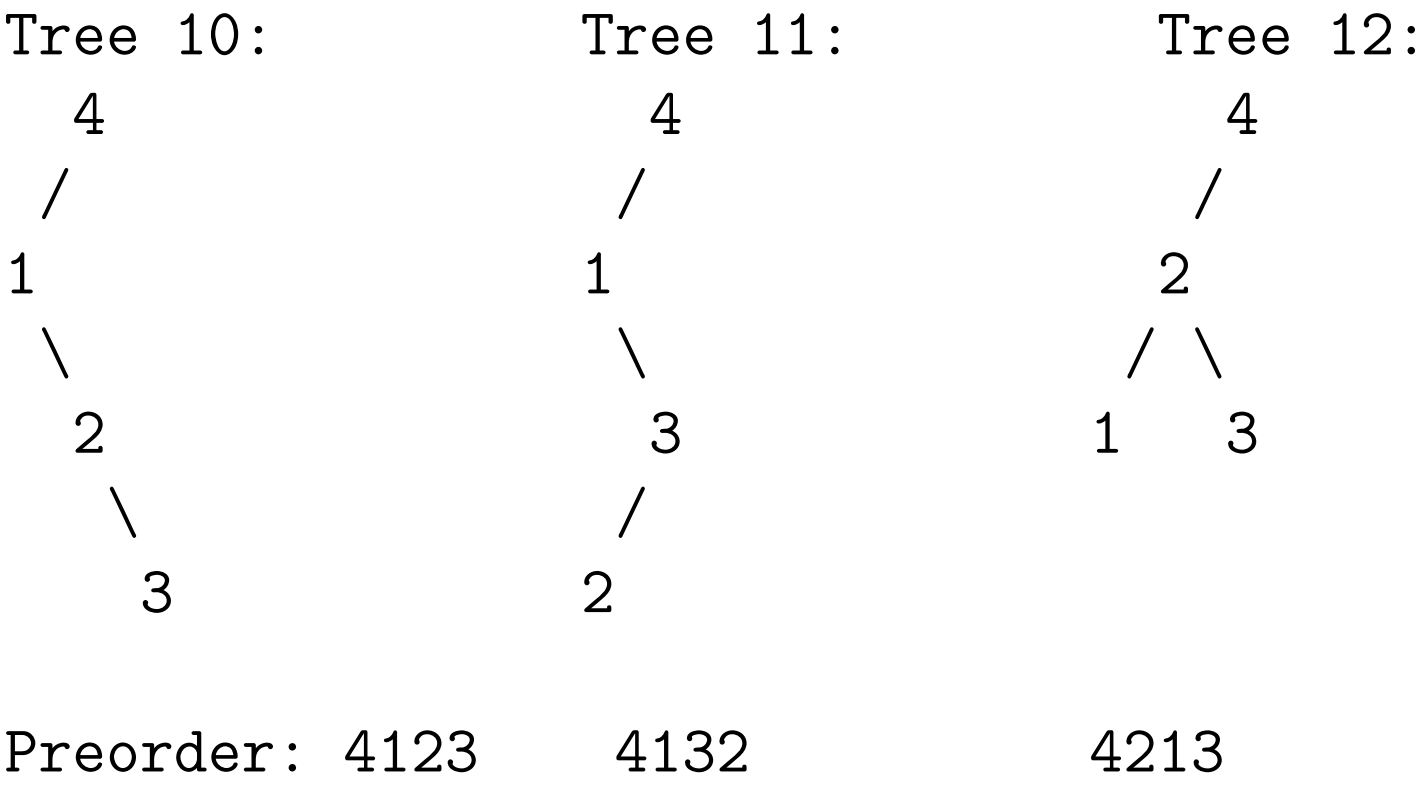
Trees with Root = 2 (2 trees)



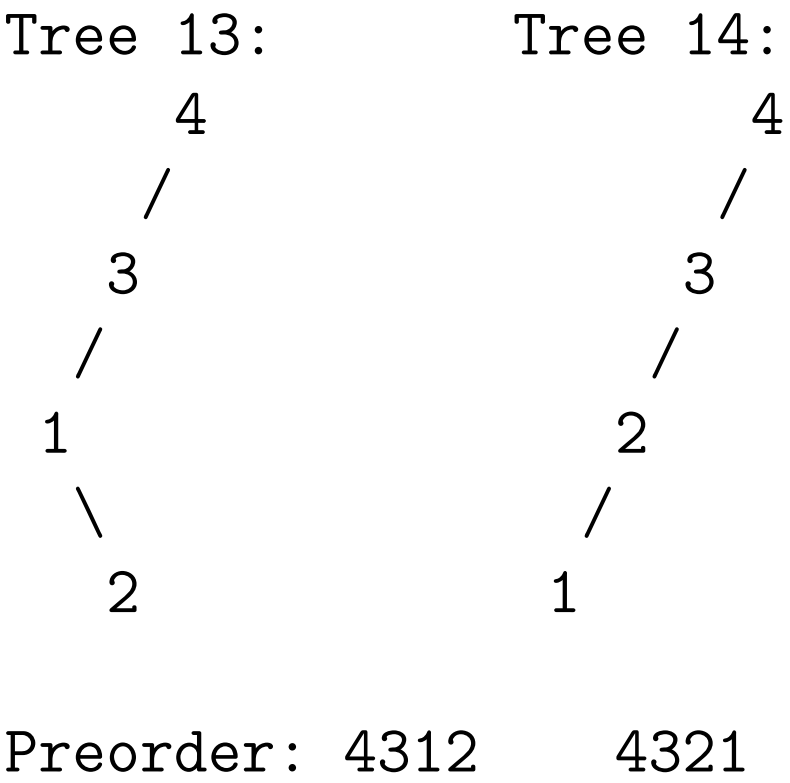
Trees with Root = 3 (2 trees)



Trees with Root = 4 (5 trees)



Trees with Root = 4 (5 trees) [contd.]



Q. Given a preorder sequence, can we uniquely construct the corresponding BST?

Q. Show bijection between preorder sequences and balanced parantheses?