Data Structures & Algorithms

07: Binary Search Trees; Part-I

Dr Ram Prasad Krishnamoorthy

Associate Professor
School of Computing and Data Science

ram.krish@saiuniversity.edu.in

SAI UNIVERSITY

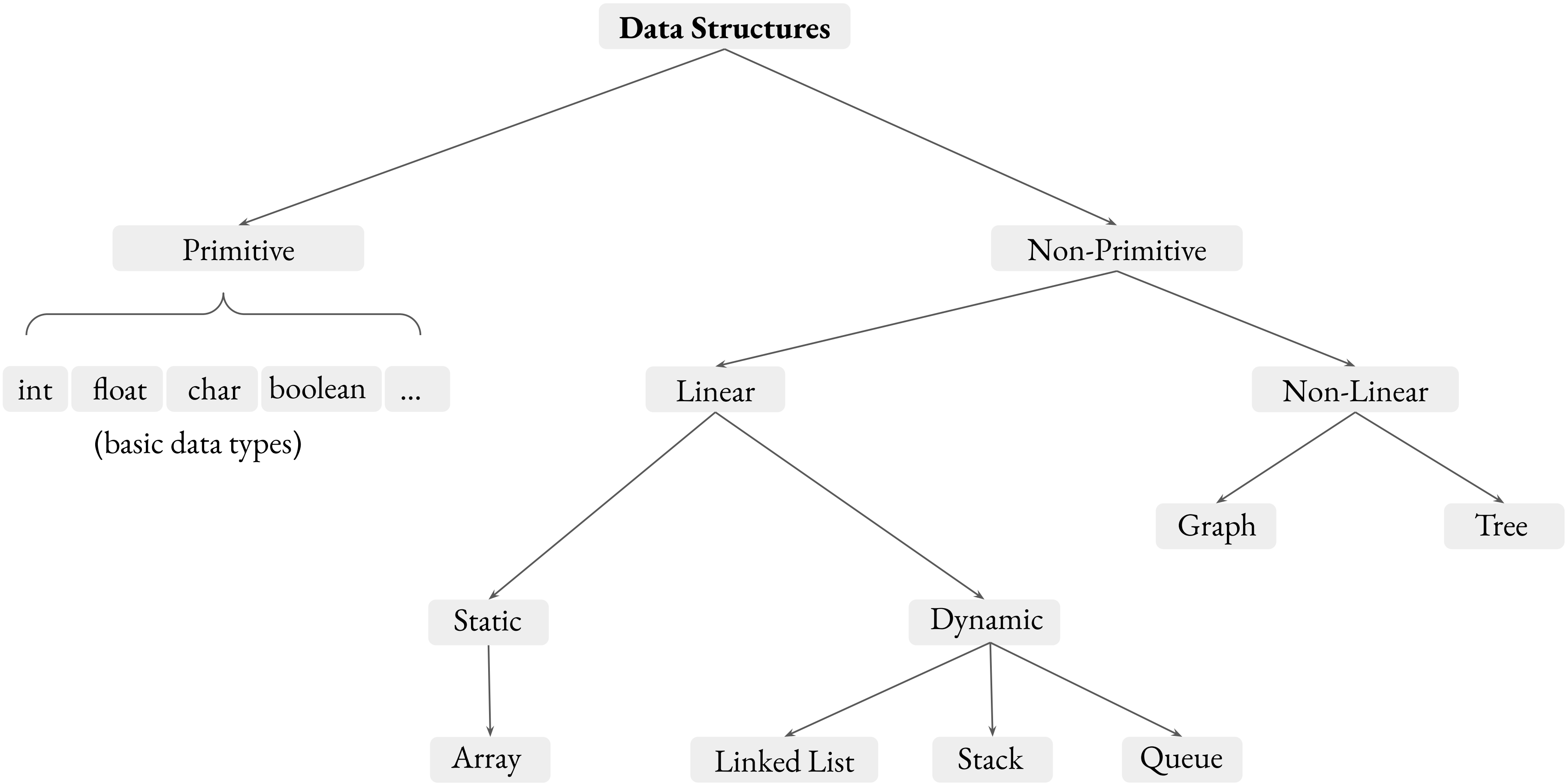
DSA Class test schedules:

- 05-April-2024, Friday
- 19-April-2024, Friday

- 10-May-2024, Friday
- 24-May-2024, Friday

Tutorial support from Turingites

- Anish Sriram
- Gayanthika Shankar
- Shagun Shukla
- Vidhyakshaya Kannan

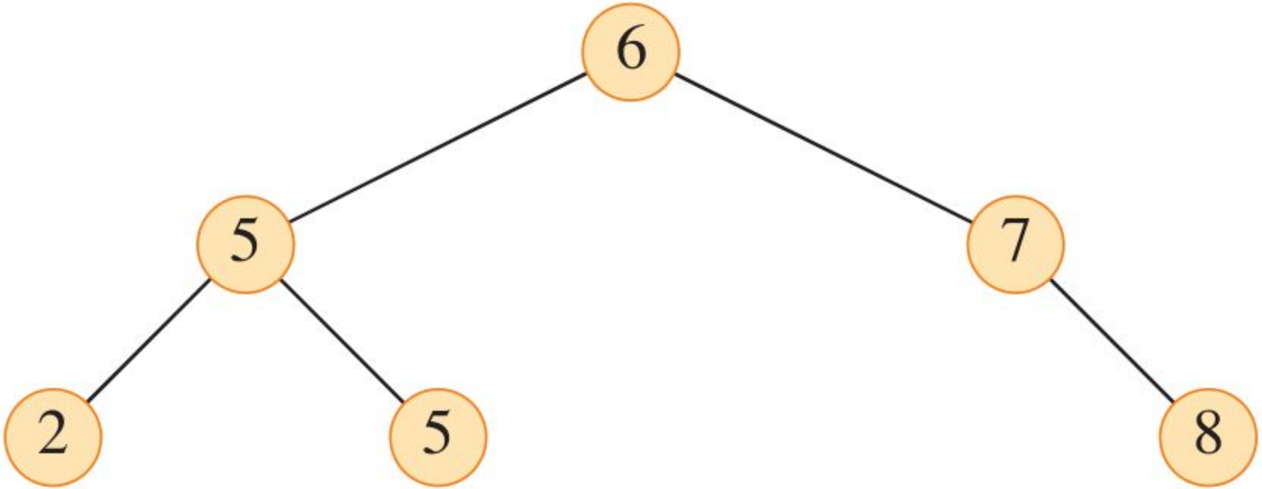

Data Structures
Primitive
Non-Primitive
int
float
char
boolean
...
(basic data types)
Linear
Non-Linear
Graph
Tree
Static
Dynamic
Array
Linked List
Stack
Queue

BINARY SEARCH TREES

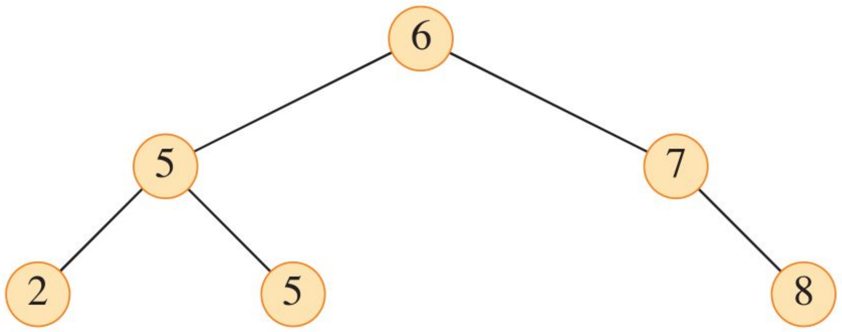
Binary Search Trees (BST) are an important data structure for dynamic sets.

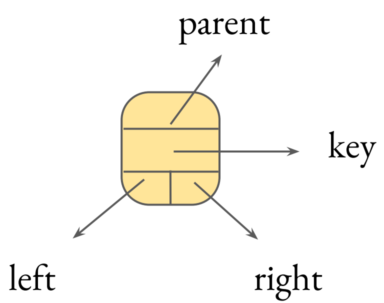
It represent a binary tree by a linked data structure in which each node is an object.

BST is also referred to as an **ordered** or **sorted binary tree**.

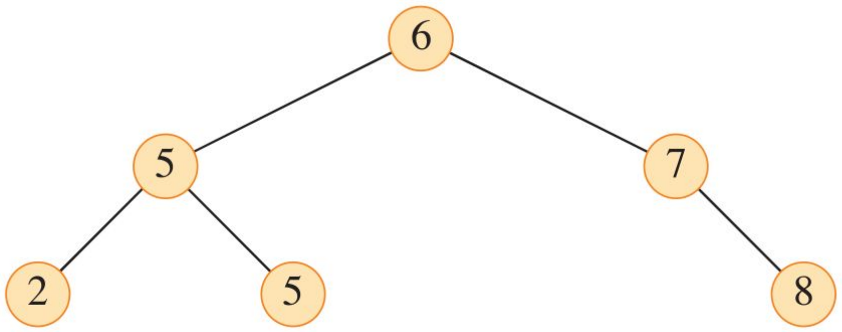


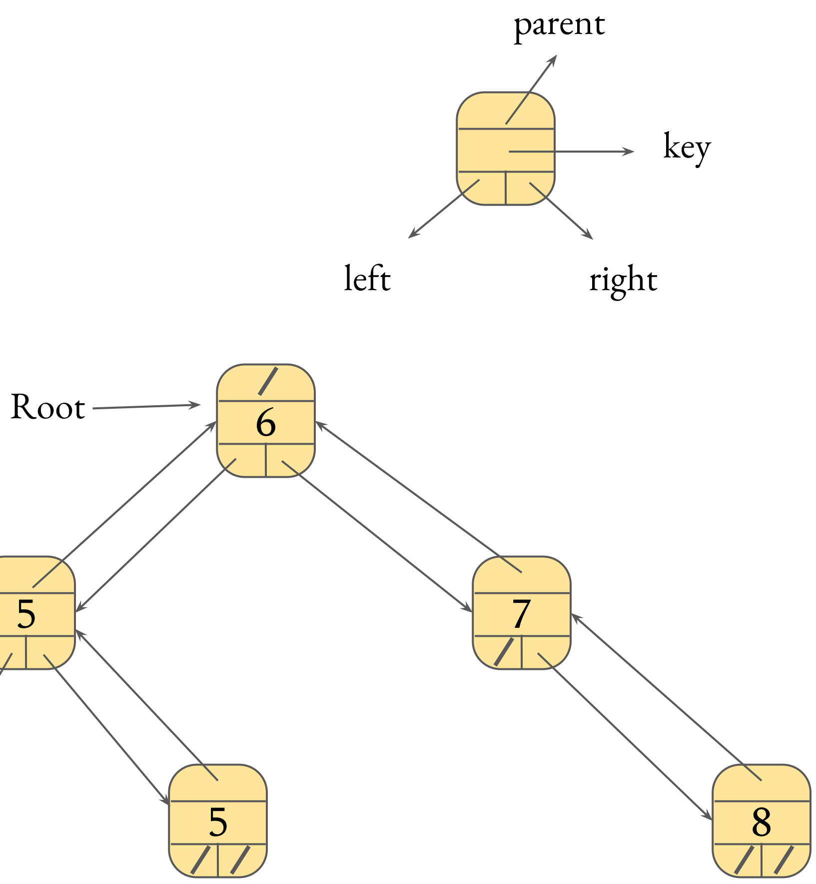
Binary Search Trees





Binary Search Trees



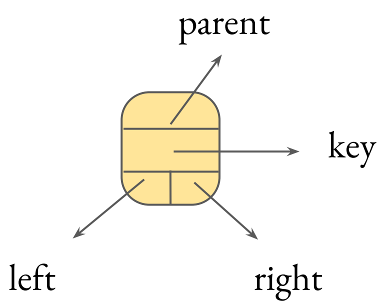


Binary Search Trees

Each node contains the attributes

- key: data value.
- left: points to left child.
- right: points to right child.
- parent: points to parent.

For Root, the parent is **NULL**.

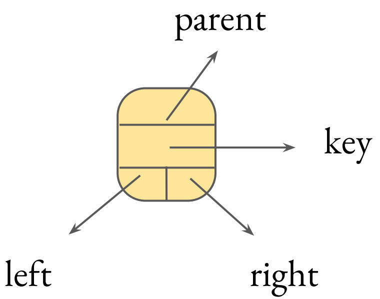


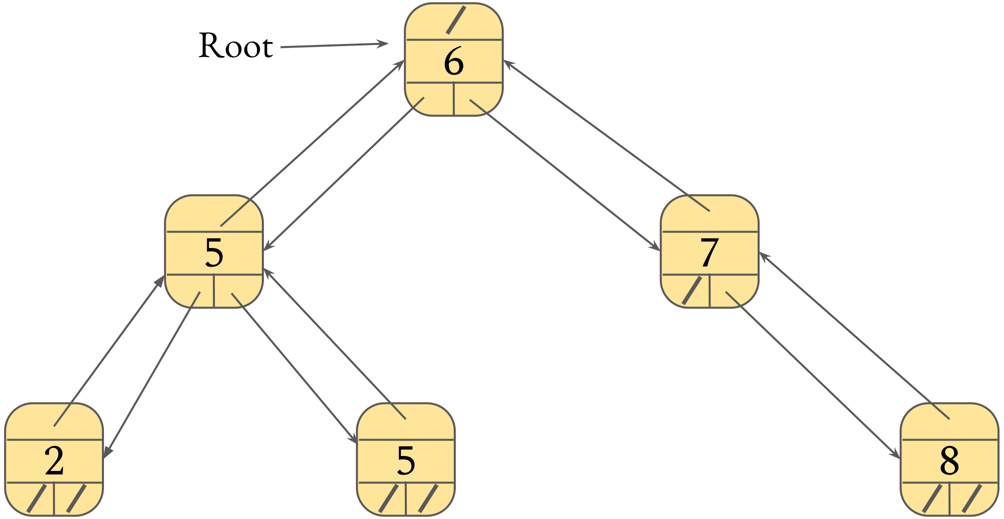
Binary Search Trees

Stored keys must satisfy the binary-search-tree property.

If y is in left subtree of x,
then **y→key** < **x→key**.

If y is in right subtree of x,
then **y→key** >= **x→key**.





Binary Search Trees

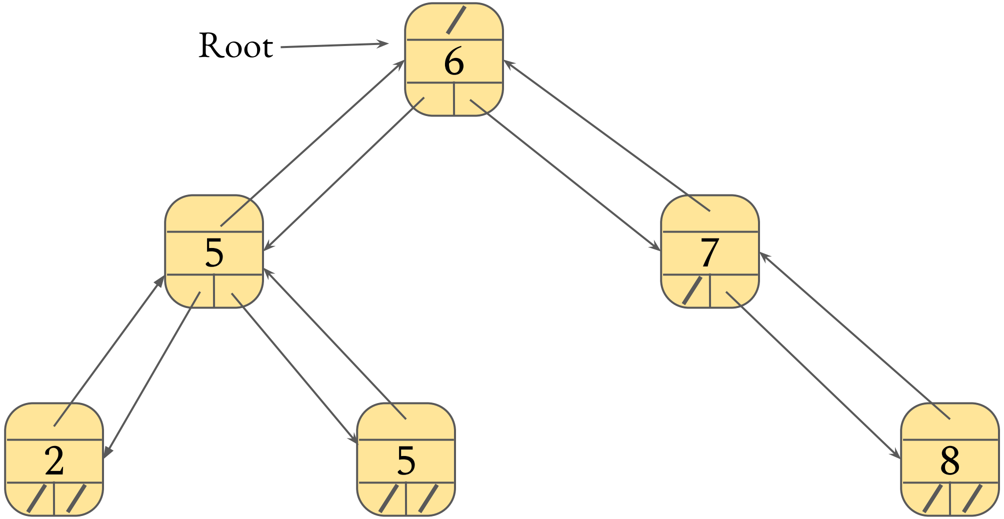
BST Insert Operation

```
TREE-INSERT(T, z)
  x = T.root                    // node being compared with z
  y = NIL                       // y will be parent of z
  while x ≠ NIL                 // descend until reaching a leaf
      y = x
      if z.key < x.key
          x = x.left
      else x = x.right
  z.p = y                       // found the location—insert z with parent y
  if y == NIL
      T.root = z                // tree T was empty
  elseif z.key < y.key
      y.left = z
  else y.right = z
```

Binary Search Trees

BST Inorder Traversal

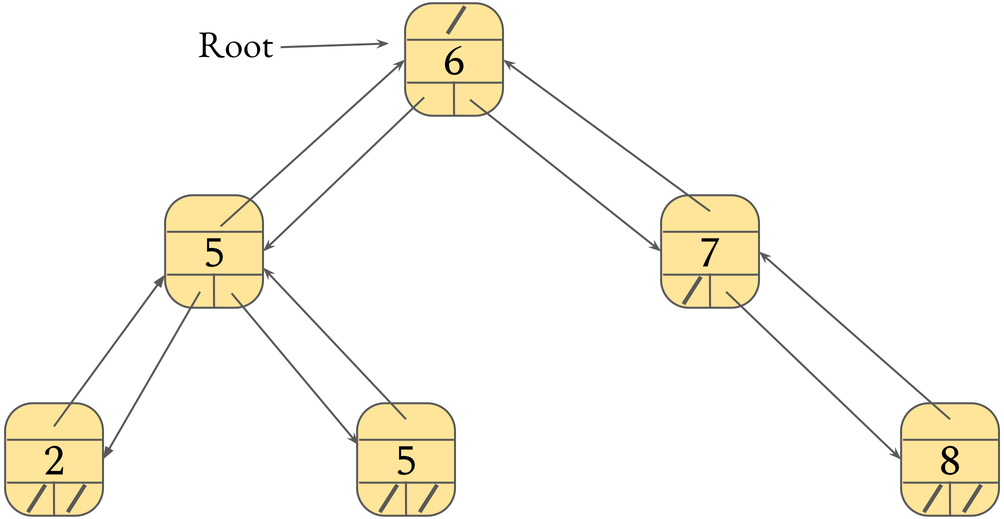
INORDER-TREE-WALK(x)
  if $x \neq$ NIL
    INORDER-TREE-WALK$(x.left)$
    print $key[x]$
    INORDER-TREE-WALK$(x.right)$



BINARY SEARCH TREES

BST Inorder Traversal

INORDER-TREE-WALK(x)
  if $x \neq$ NIL
    INORDER-TREE-WALK$(x.\mathit{left})$
    print $\mathit{key}[x]$
    INORDER-TREE-WALK$(x.\mathit{right})$



How INORDER-TREE-WALK works:

- Check to make sure that x is not NIL.
- Recursively print the keys of the nodes in x's left subtree.
- Print x's key.
- Recursively print the keys of the nodes in x's right subtree.