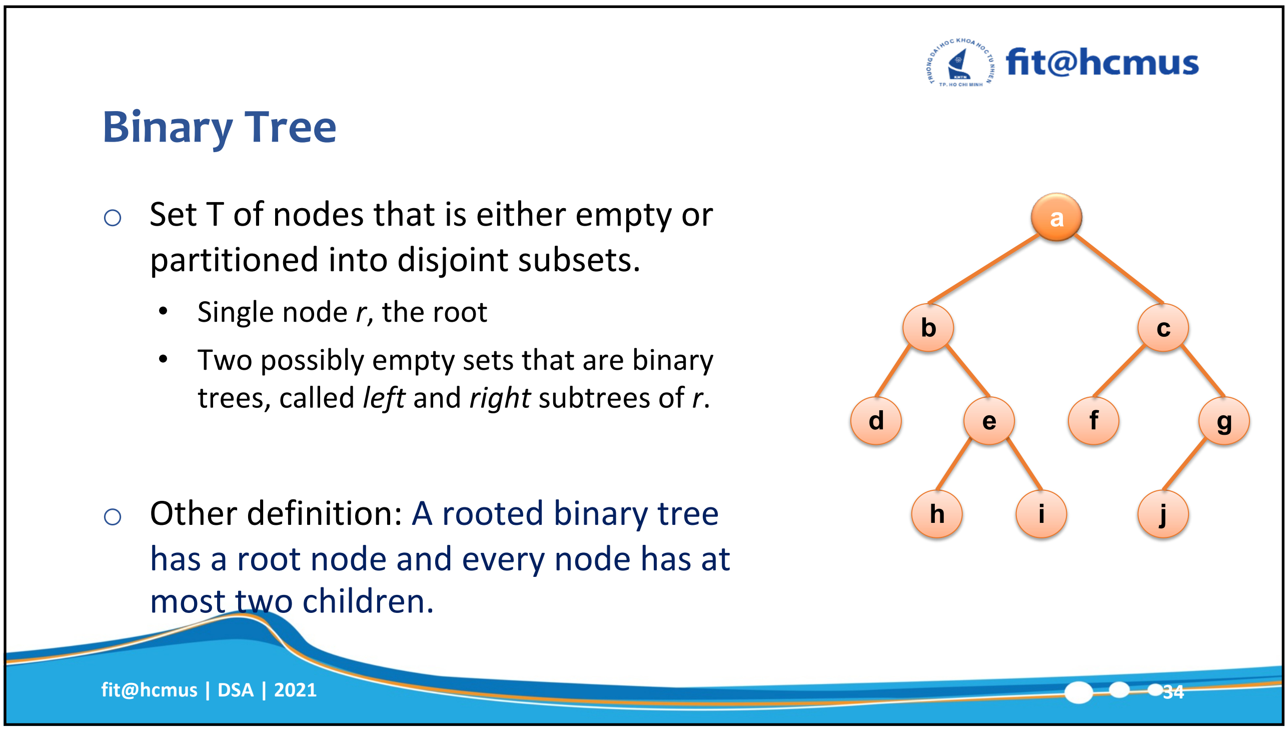
Tree Structures

1

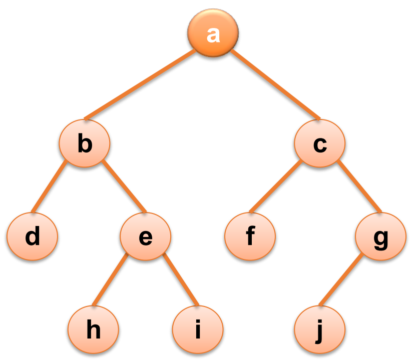
Binary Tree

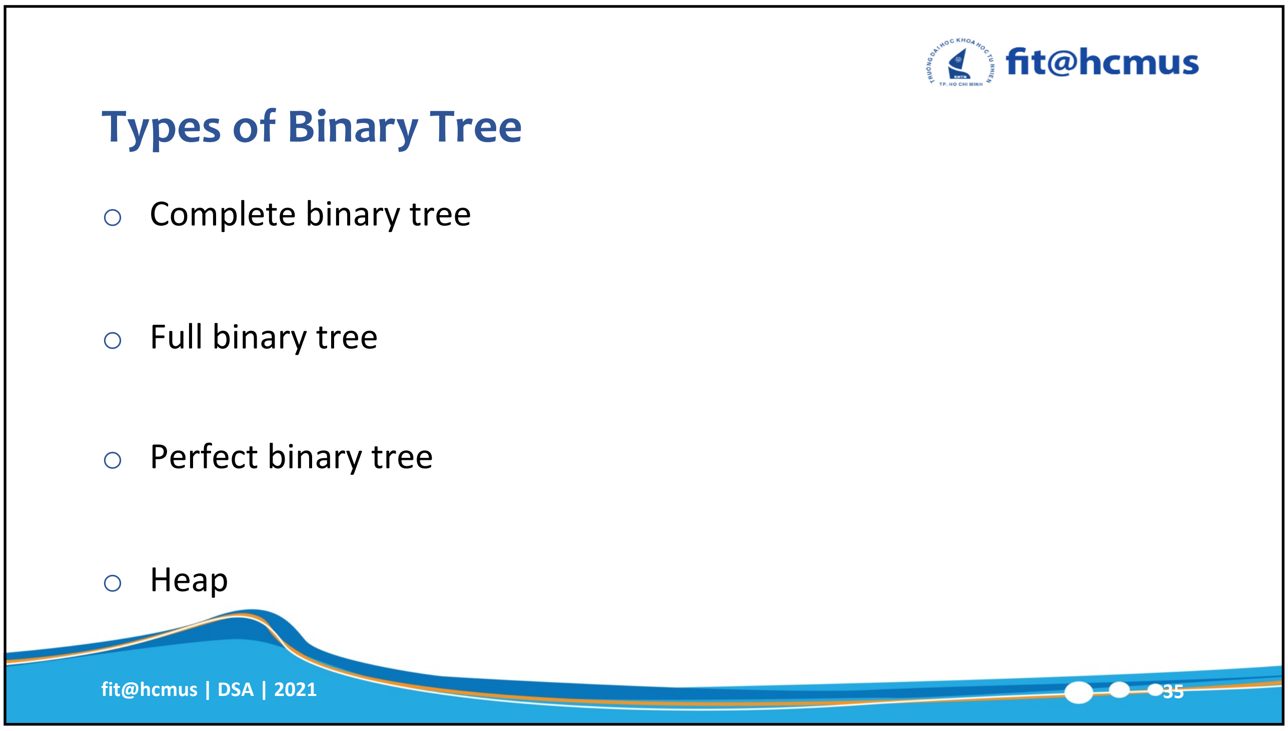
33



Binary Tree

- Set T of nodes that is either empty or partitioned into disjoint subsets.
  - Single node *r*, the root
  - Two possibly empty sets that are binary trees, called *left* and *right* subtrees of *r*.
- Other definition: A rooted binary tree has a root node and every node has at most two children.





Types of Binary Tree

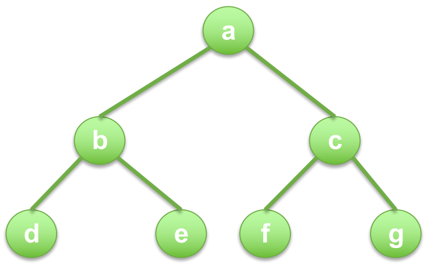
- Complete binary tree
- Full binary tree
- Perfect binary tree
- Heap

Perfect Binary Tree

- A **perfect binary tree** is a binary tree in which
  - all interior nodes have two children
  - and all leaves have the same depth or same level.
- In a perfect binary tree of height *h*, all nodes that are at a level less than *h* have two children each.

Perfect Binary Tree

- If *T* is empty, *T* is a perfect binary tree of height 0.
- If *T* is not empty and has height *h* > 0, T is a perfect binary tree if its root's subtrees are both perfect binary trees of height *h* – 1.



fit@hcmus

Complete Binary Tree

- A **complete binary tree** of height *h* is a binary tree that is **perfect** down to level *h* – 1, with level *h* filled in from left to right.

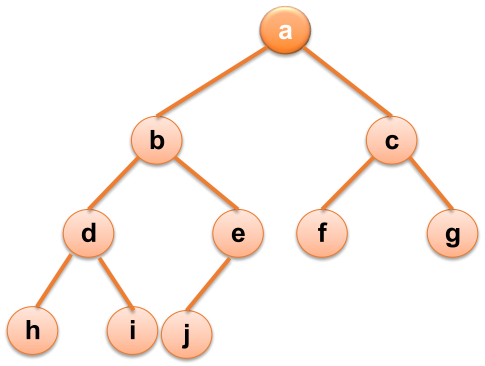
- In a complete binary tree every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible.

- Other definition: A **complete binary tree** is a perfect binary tree whose rightmost leaves (perhaps all) have been removed.

fit@hcmus

Complete Binary Tree

- A binary tree is complete if
  - All nodes at level *h* – 2 and above have two children each, and
  - When a node at level *h* – 1 has children, all nodes to its left at the same level have two children each, and
  - When a node at level *h* – 1 has one child, it is a left child

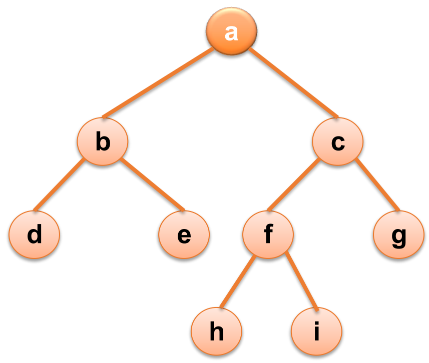


fit@hcmus

Full Binary Tree

- A full binary tree (sometimes referred to as a **proper binary tree** or **a plane binary tree**) is a binary tree in which ***every node has either 0 or 2 children***.

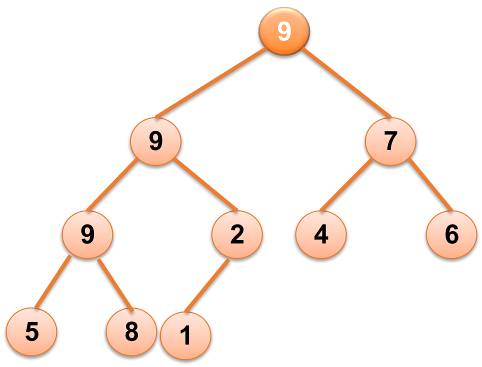
- A full binary tree is either:
  - A single vertex.
  - A tree whose root node has two subtrees, both of which are full binary trees.



fit@hcmus

Heap

- A heap is a ***complete binary tree*** that either is empty or
  - Its root
    - (Max-heap): Contains a value greater than or equal to the value in each of its children, and
    - (Min-heap): Contains a value less than or equal to the value in each of its children, and
    - Has heaps as its subtrees



Number of Nodes

Given a binary tree *T* height of *h*.

- What is the maximum number of nodes?

- What is the minimum number of nodes?

42

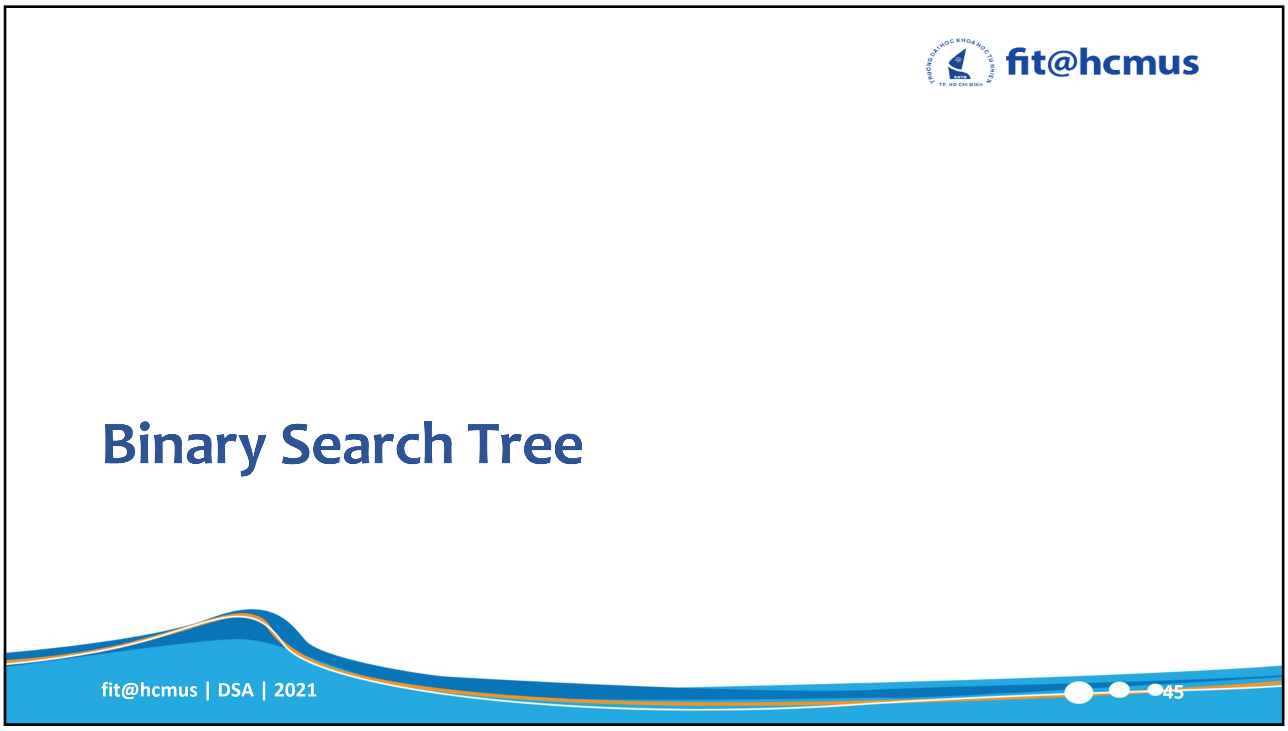
Height of Tree

Given a binary tree *T* with *n* nodes.

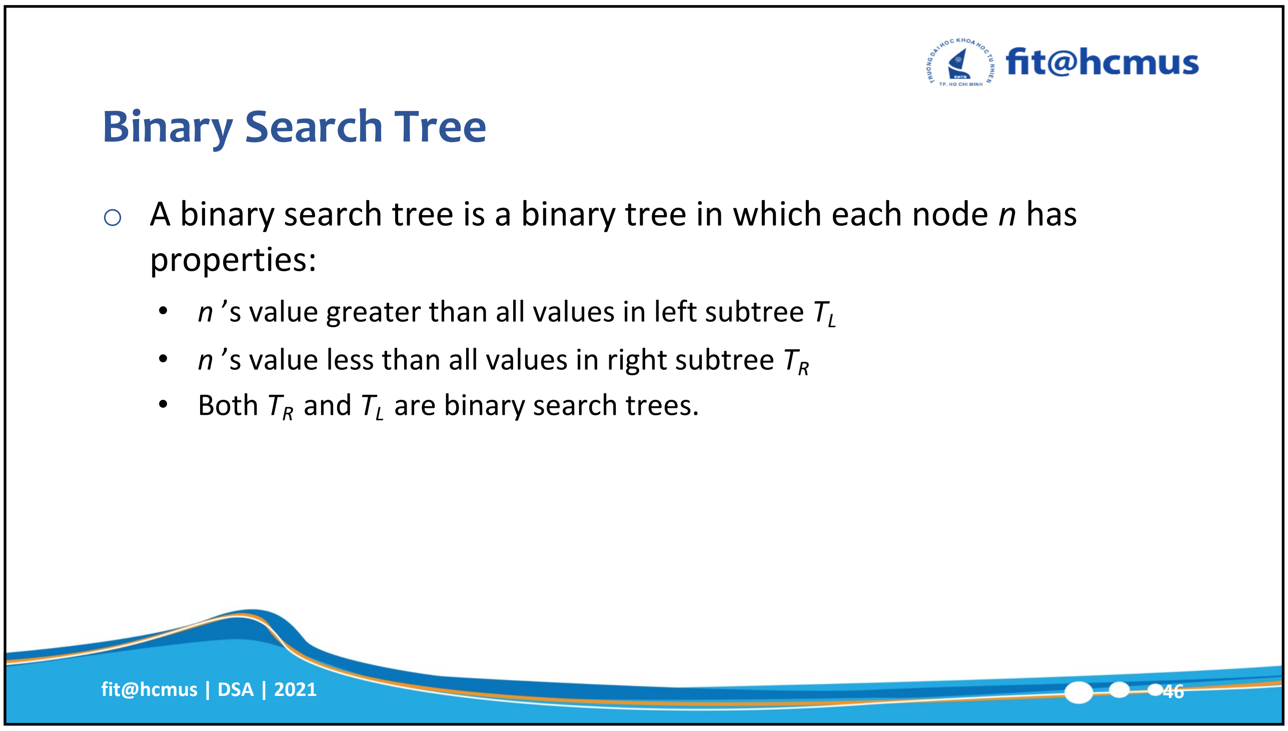
- What is the maximum height of that tree?

- What is the minimum height of that tree?

43

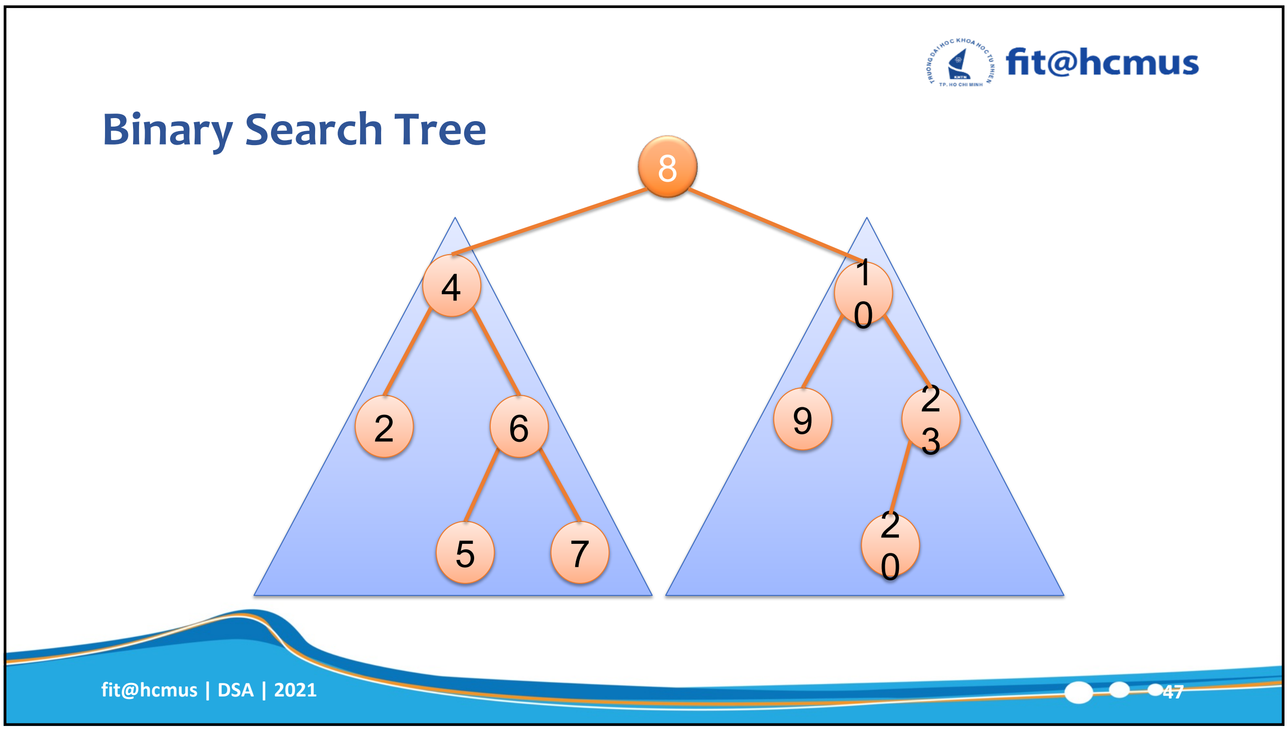

fit@hcmus

Binary Search Tree

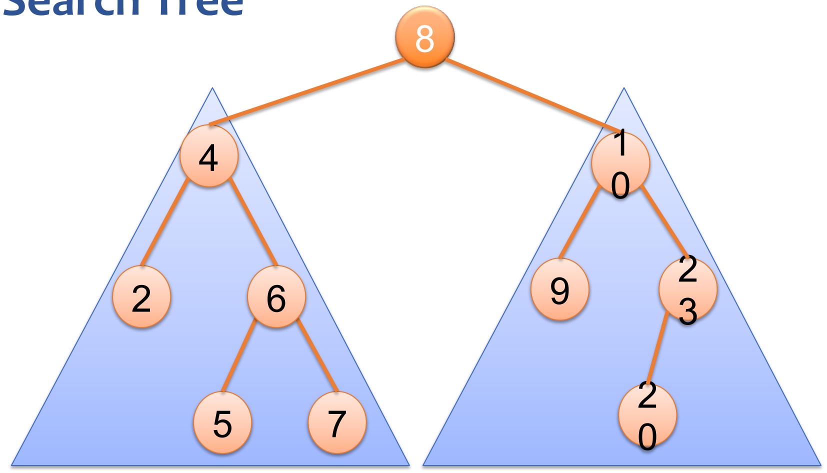

fit@hcmus

Binary Search Tree

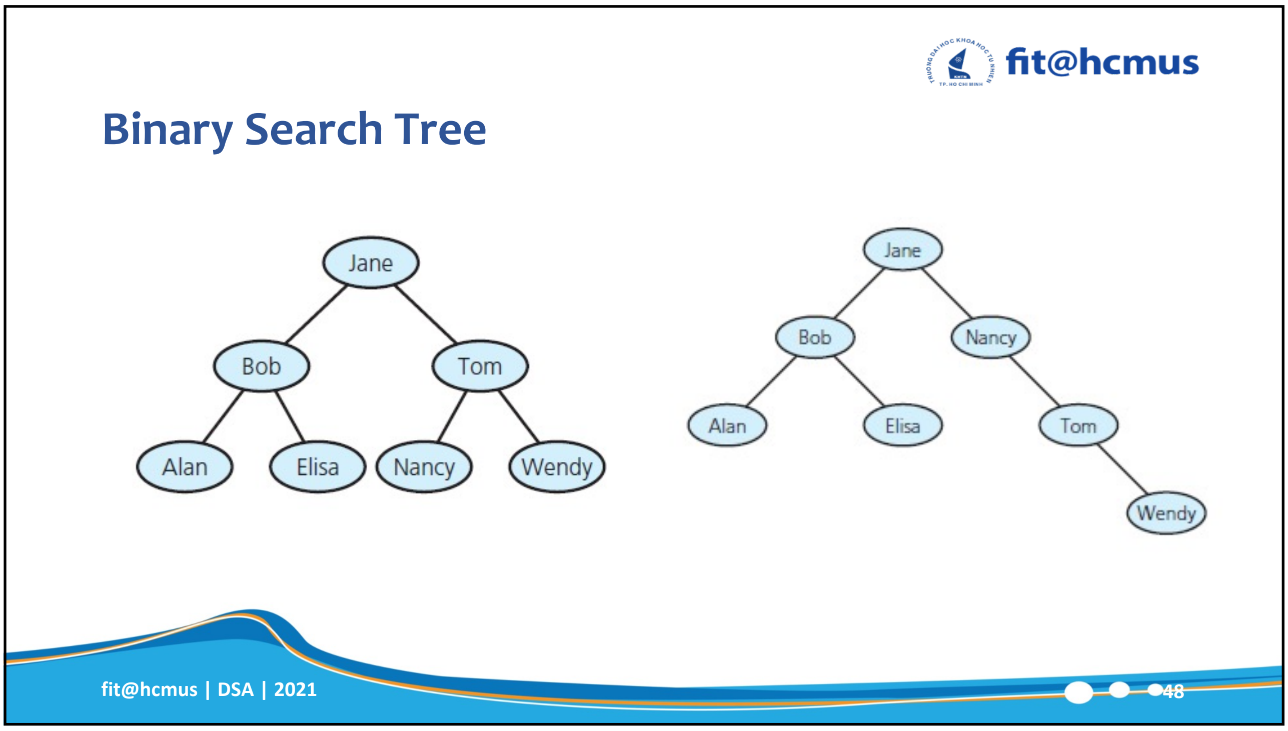
- A binary search tree is a binary tree in which each node *n* has properties:
  - *n* ’s value greater than all values in left subtree T_L
  - *n* ’s value less than all values in right subtree T_R
  - Both T_R and T_L are binary search trees.



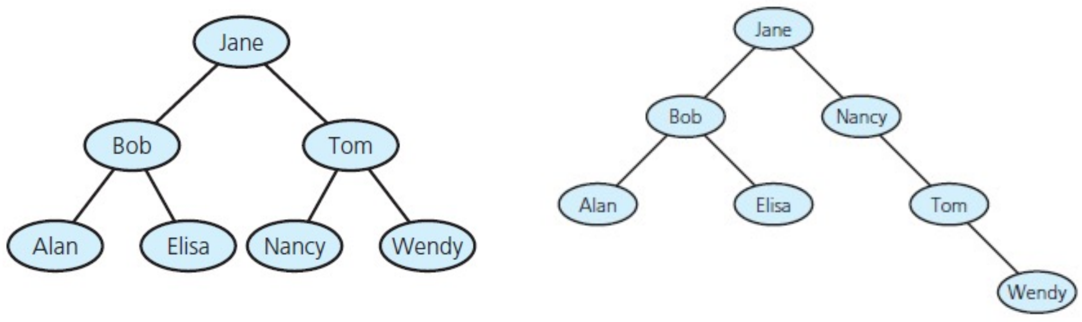
Binary Search Tree



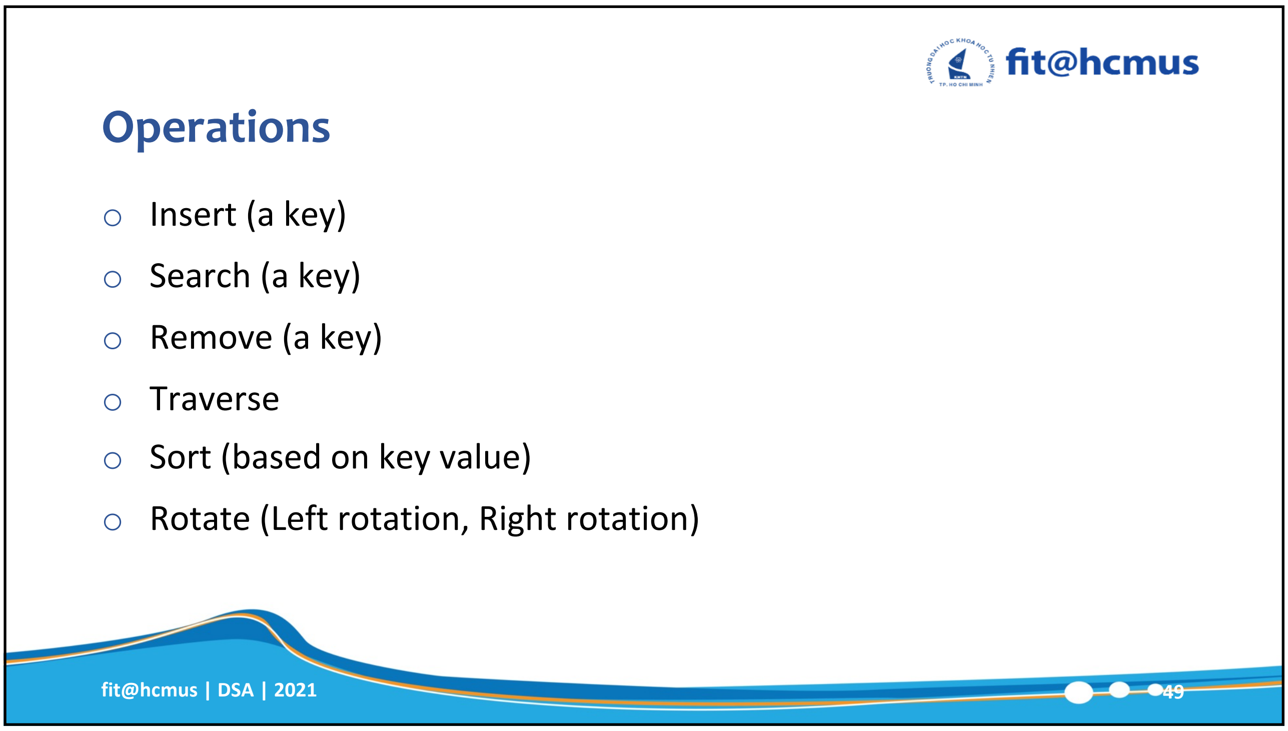
47



Binary Search Tree



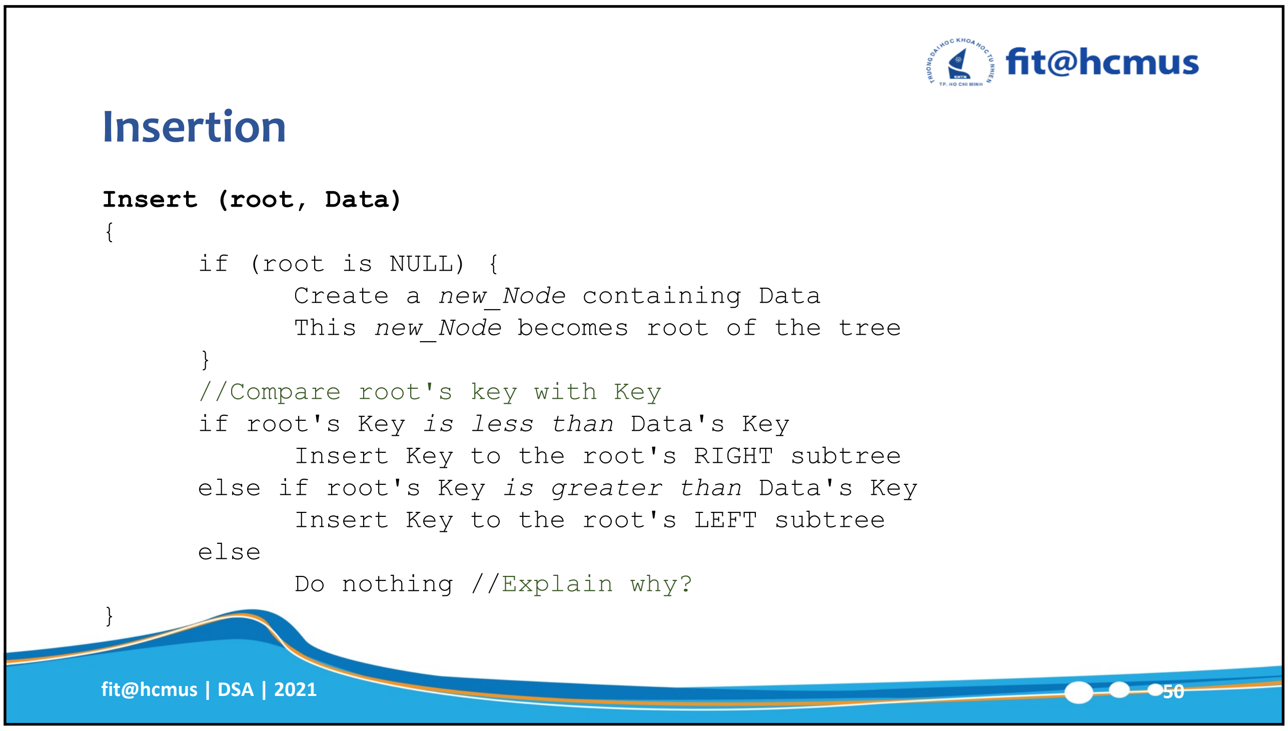
48



Operations

- Insert (a key)
- Search (a key)
- Remove (a key)
- Traverse
- Sort (based on key value)
- Rotate (Left rotation, Right rotation)

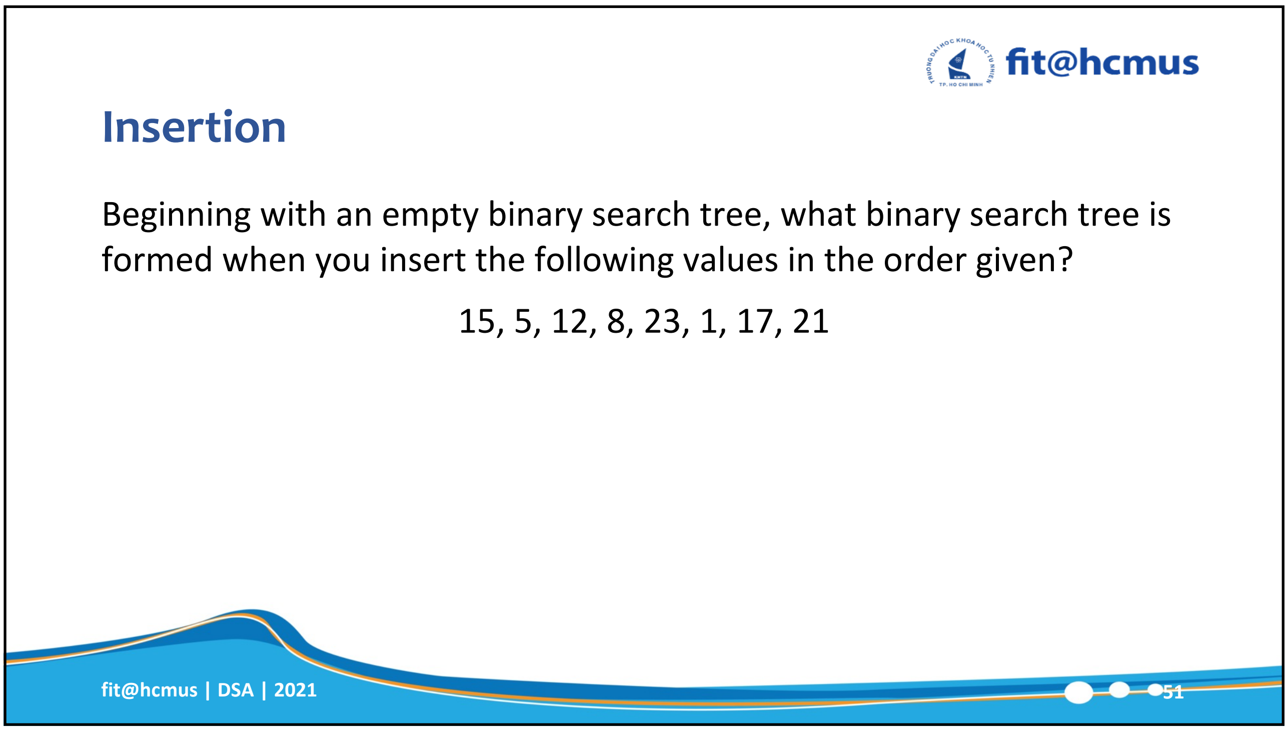
49



Insertion

```
Insert (root, Data)
{
      if (root is NULL) {
            Create a new_Node containing Data
            This new_Node becomes root of the tree
      }
      //Compare root's key with Key
      if root's Key is less than Data's Key
            Insert Key to the root's RIGHT subtree
      else if root's Key is greater than Data's Key
            Insert Key to the root's LEFT subtree
      else
            Do nothing //Explain why?
}
```

50



Insertion

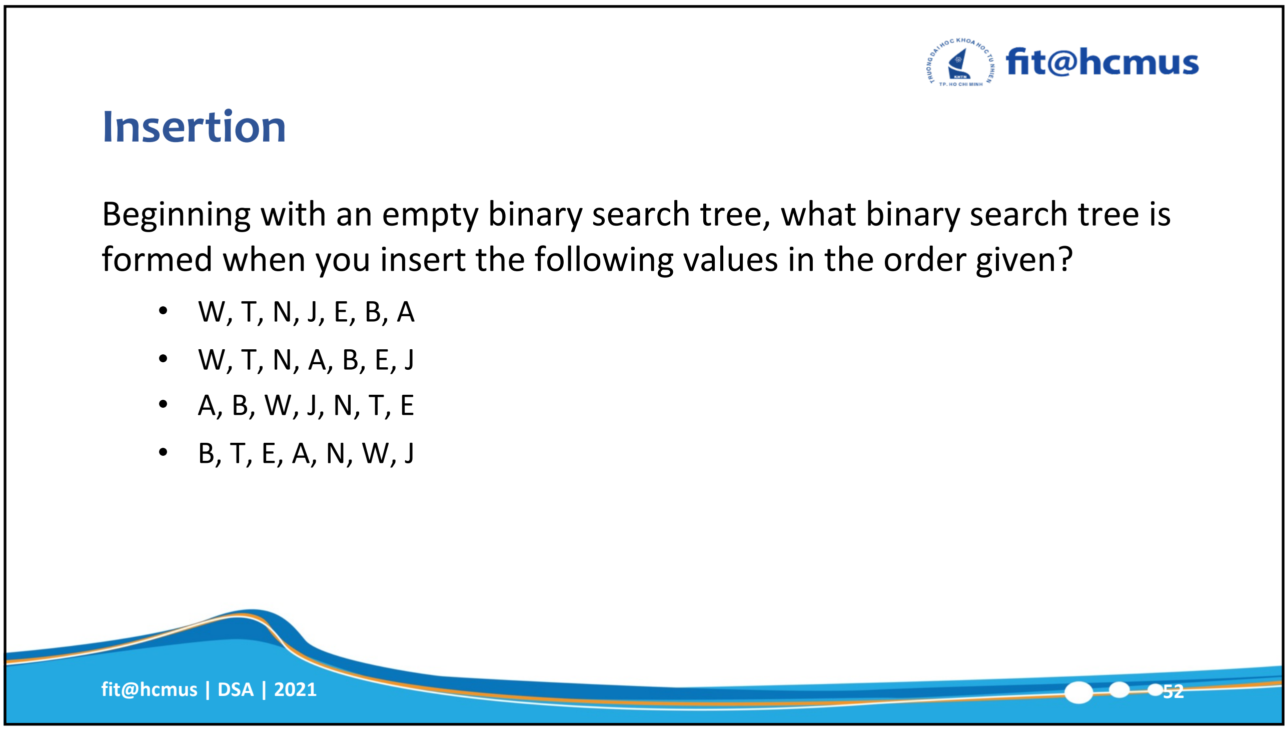
Beginning with an empty binary search tree, what binary search tree is formed when you insert the following values in the order given?

15, 5, 12, 8, 23, 1, 17, 21

fit@hcmus | DSA | 2021

51

51



Insertion

Beginning with an empty binary search tree, what binary search tree is formed when you insert the following values in the order given?

- W, T, N, J, E, B, A
- W, T, N, A, B, E, J
- A, B, W, J, N, T, E
- B, T, E, A, N, W, J

fit@hcmus | DSA | 2021

52

52

fit@hcmus

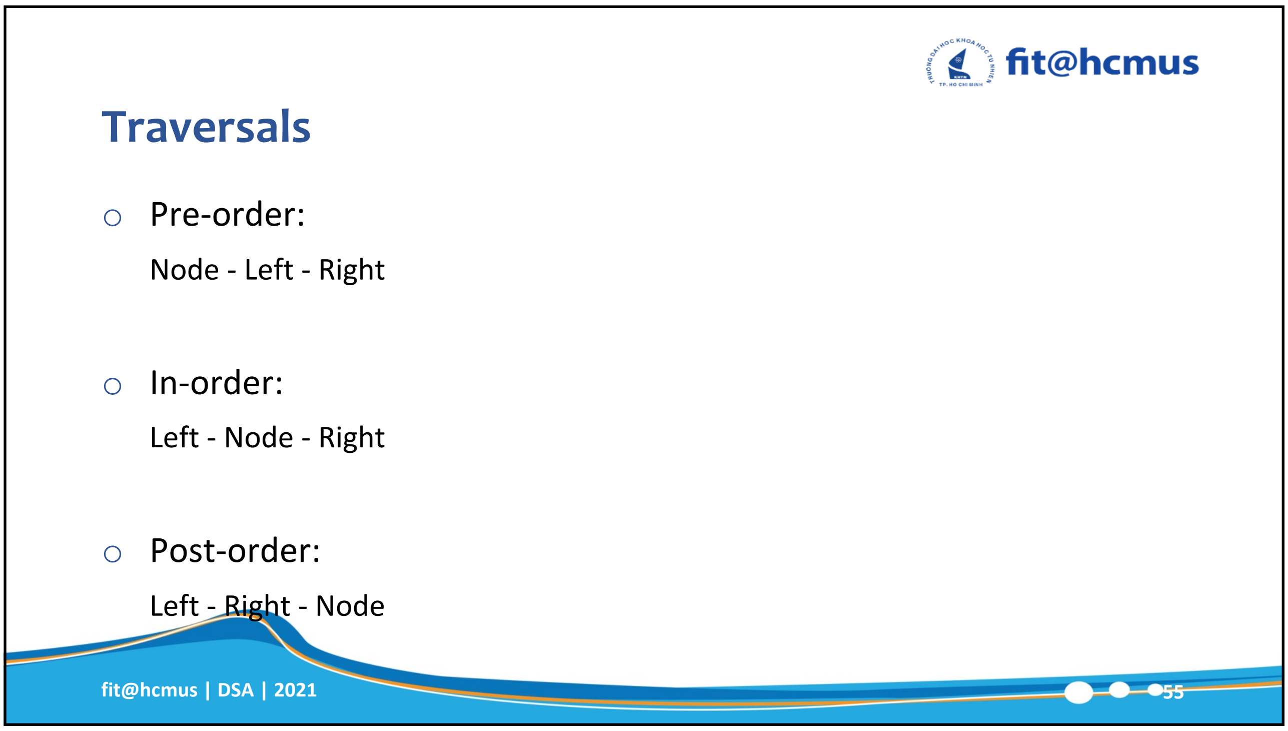
Search

```
Search (root, Data)
{
      if (root is NULL) {
            return NOT_FOUND;
      }
      //Compare root's key with Key
      if root's Key is less than Data's Key
            Search Data in the root's RIGHT subtree
      else if root's Key is greater than Data's Key
            Search Data in the root's LEFT subtree
      else
            return FOUND //Explain why?
}
```

fit@hcmus

Deletion

- When we delete a node, three possibilities arise.
- Node to be deleted:
  - **is leaf:**
    - Simply remove from the tree.
  - has **only one child:**
    - Copy the child to the node and delete the child
  - has **two children:**
    - Find in-order successor (predecessor) ***S_Node*** of the node.
    - Copy contents of ***S_Node*** to the node and delete the ***S_Node***.



Traversals

- Pre-order:

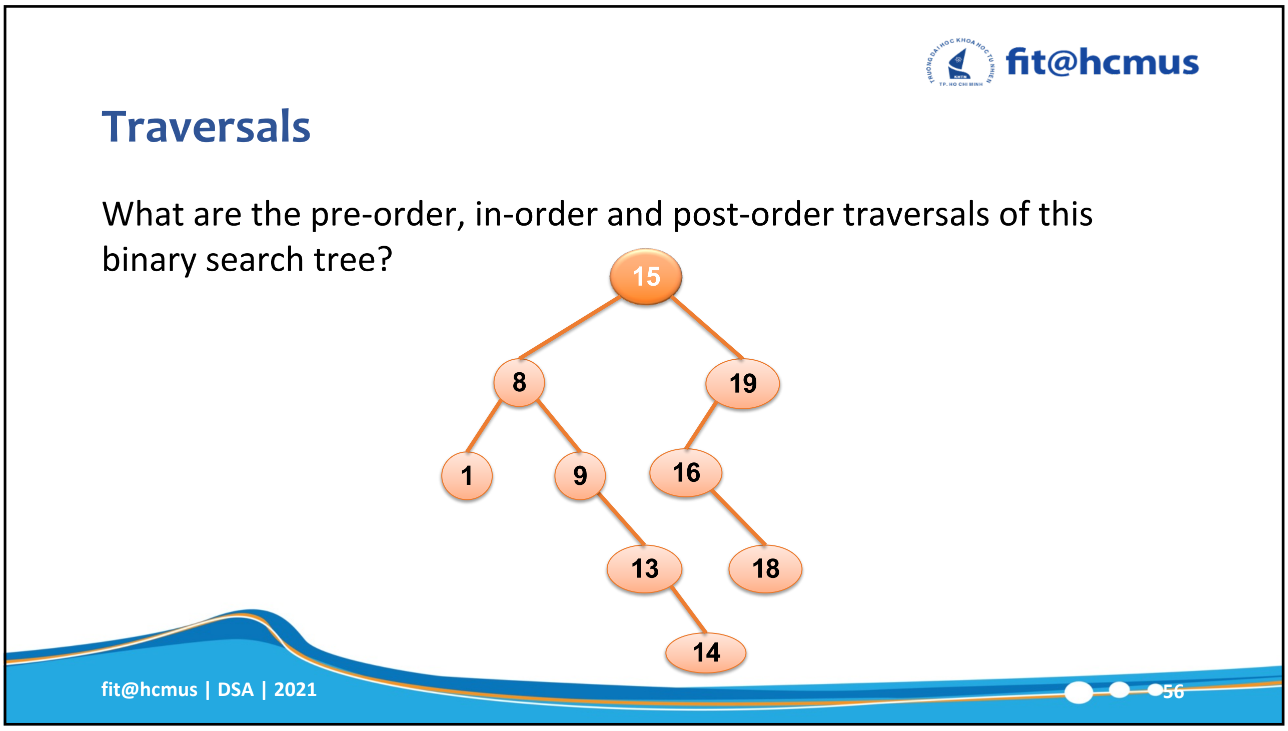
  Node - Left - Right

- In-order:

  Left - Node - Right

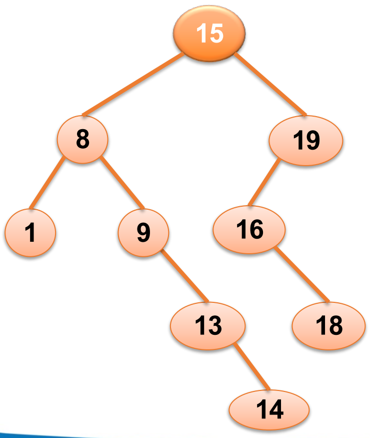
- Post-order:

  Left - Right - Node



Traversals

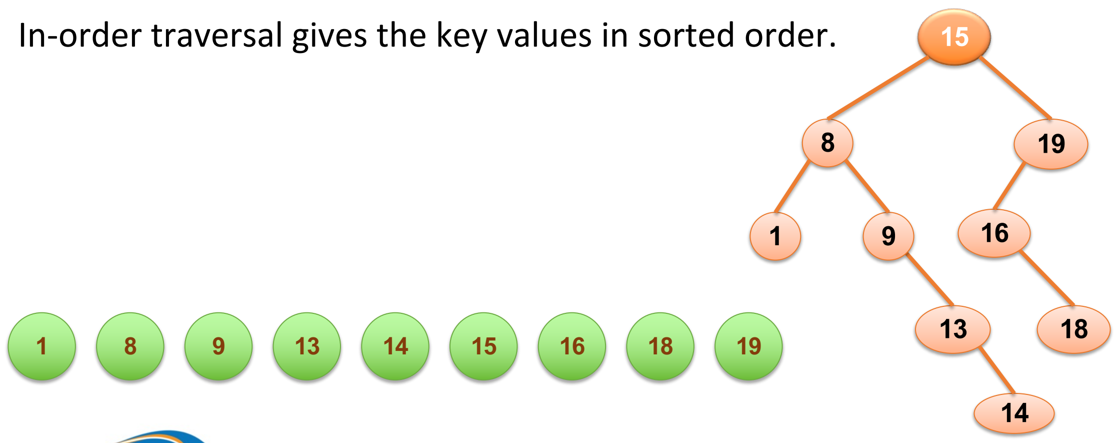
What are the pre-order, in-order and post-order traversals of this binary search tree?



fit@hcmus

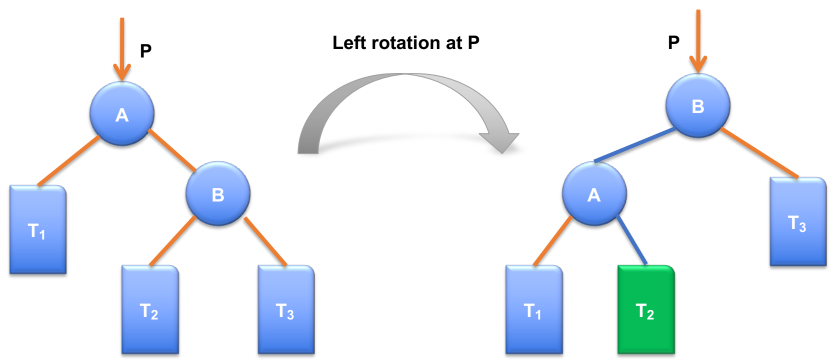
Sort

In-order traversal gives the key values in sorted order.

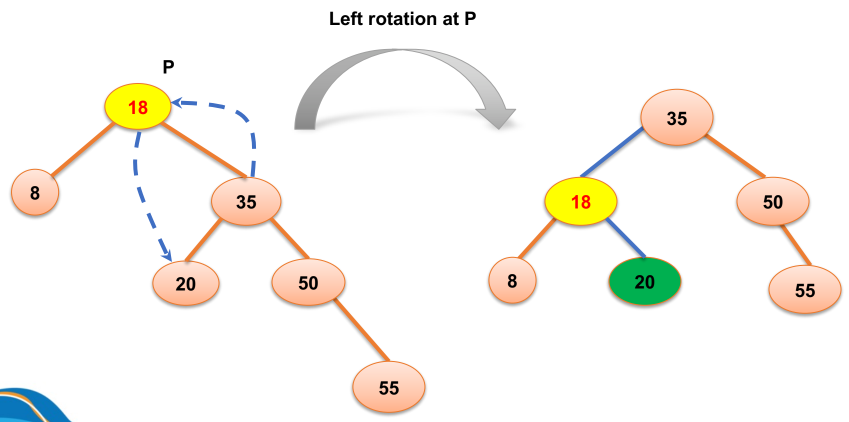


fit@hcmus

Left Rotation

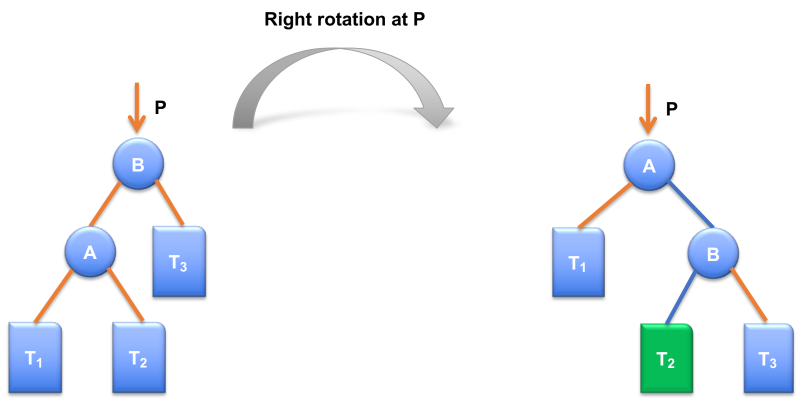


Left Rotation



59

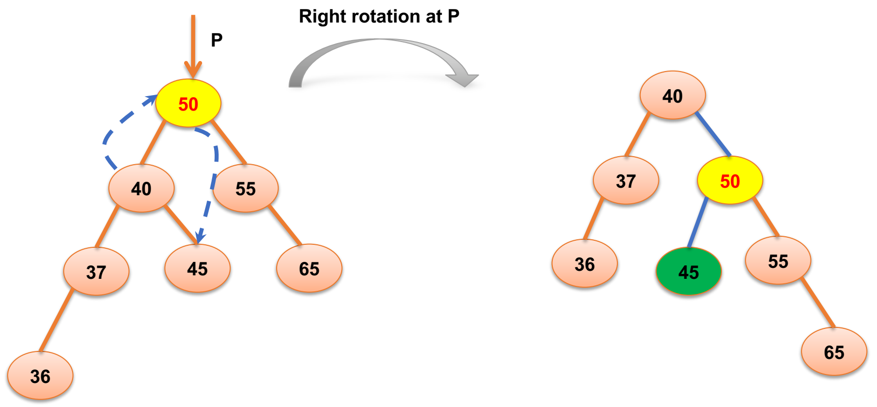
Right Rotation



60

fit@hcmus

Right Rotation



61

fit@hcmus

Efficiency of Binary Search Tree Operations

Operation	Average case	Worst case
Retrieval	O(log *n*)	O(*n*)
Insertion	O(log *n*)	O(*n*)
Removal	O(log *n*)	O(*n*)
Traversal	O(*n*)	O(*n*)

62

Very Bad Binary Search Tree

Beginning with an empty binary search tree, what binary search tree is formed when inserting the following values in the order given?

2, 4, 6, 8, 10, 12, 14, 18, 20