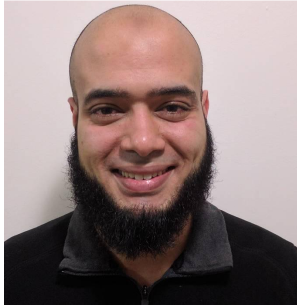
Data Structures

Binary Search Tree

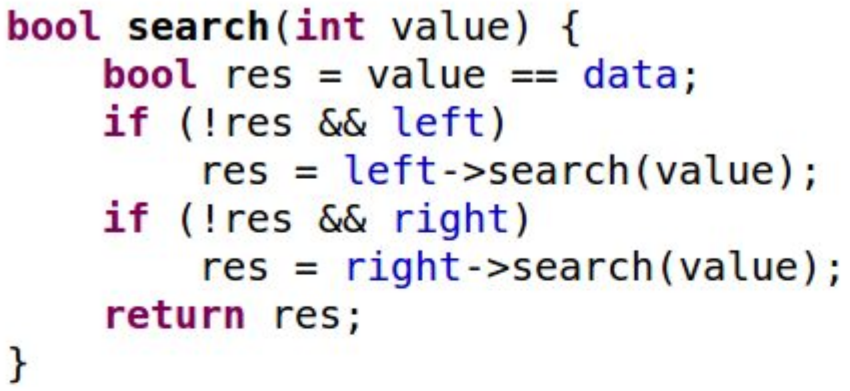
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Searching for an element

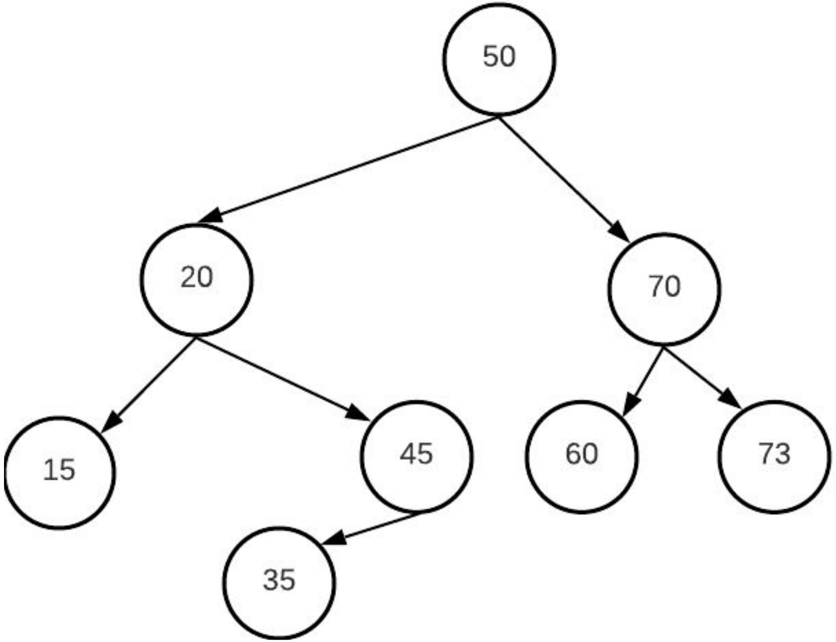
- Given a binary tree, search it to find out whether it contains a specific element or not
- We can simply traverse the whole tree ⇒ O(N) numbers
- Can we arrange the elements in the tree to help speed up the search?



```
bool search(int value) {
    bool res = value == data;
    if (!res && left)
        res = left->search(value);
    if (!res && right)
        res = right->search(value);
    return res;
}
```

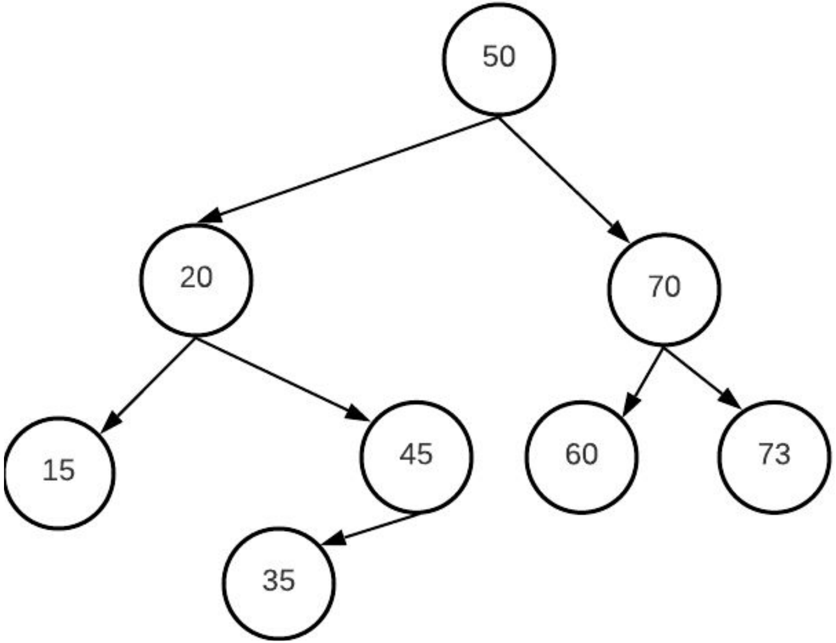
Binary Search Tree (BST)

- Instead of structuring values in a random order, we can use a BST
- BST Tree Properties(**FIX**):
  - Node value > **all** left nodes
  - Node value < **all** right nodes
  - Every-subtree is also BST
- How to search for an element?
  - Think for 5 minutes!



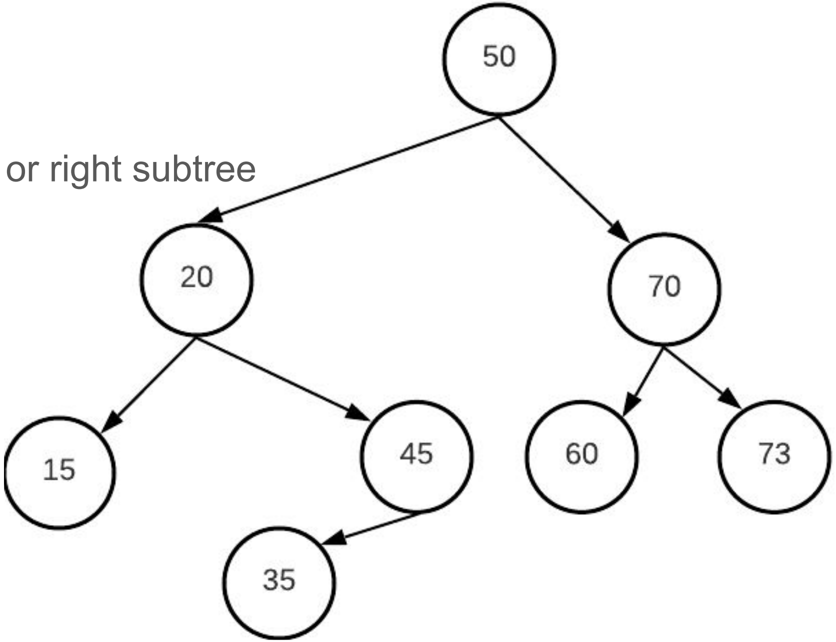
BST properties

- By definition, no duplicates
  - But we can do workarounds
- Inorder traversal: Sorted values
  - 15 20 35 45 50 60 70 73
  - Why? Inorder + BST property
- Given preorder, postorder or level order only, we can build a BST
- The number of BST trees of N nodes is the same as the number of unlabeled binary trees: the Catalan number
  - Why? All of their inorder traversals must be sorted. Values are then useless = the same as an unlabeled tree



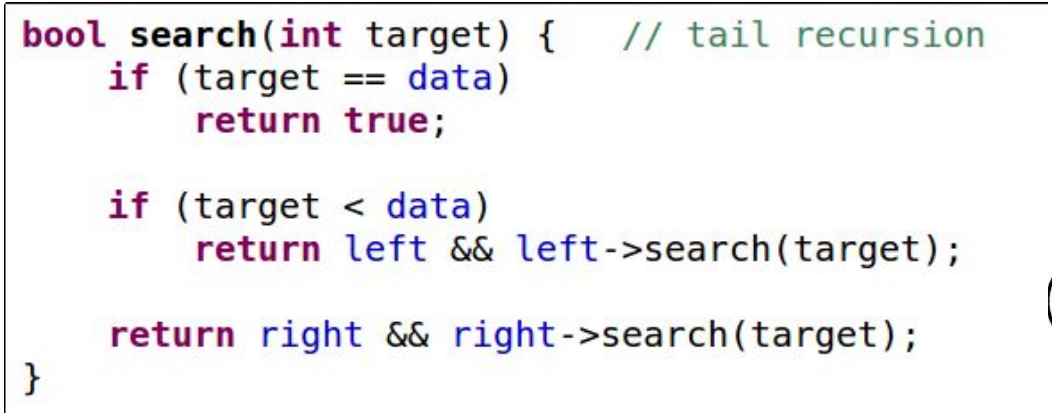
Searching BST

- How to search it?
  - The value is found either in the root, or the left or right subtree
  - This means we always choose a subtree
  - Overall O(h) time
  - O(log(n)) for a balanced tree
- Try to code it



Searching BST

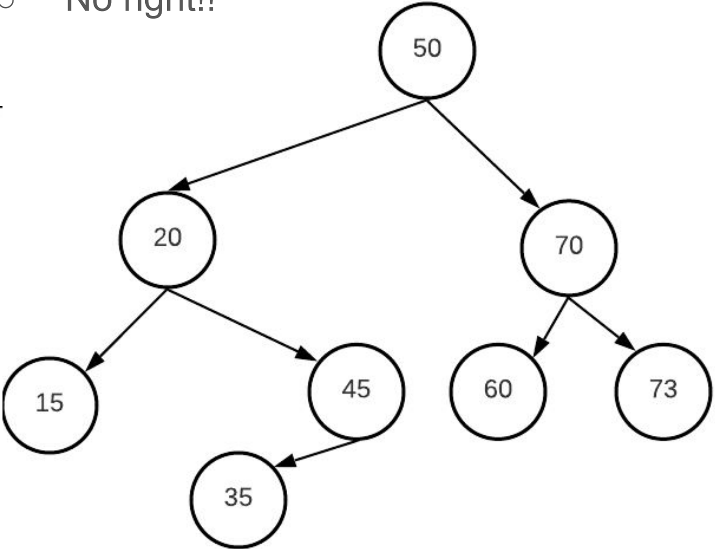
- Search(35)
  - At Node(50)? No, go left as **our target** is smaller
  - At Node(20)? No, go right as bigger.
  - At Node(45)? No, go left as smaller.
  - Found
- Search(17)
  - At Node(50)? No, go left as smaller.
  - At Node(20)? No, go left as smaller.
  - At Node(15)? No, go right as bigger.
  - No right!!



```
bool search(int target) {   // tail recursion
    if (target == data)
        return true;

    if (target < data)
        return left && left->search(target);

    return right && right->search(target);
}
```



"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."