THE INTRODUCTION TO

TREES

DATA STRUCTURE - TREE

- **Trees** are abstract data structures, used to manage data in a hierarchical way, making data retrieving much more efficient than other data structure methods.
- A tree is a collection of **nodes** that are linearly connected and don't have any cyclic relations. These nodes contain information and are usually arranged in a **key-value** structure. The key serves as an identifier for the data, and value is the actual data that you want to store.

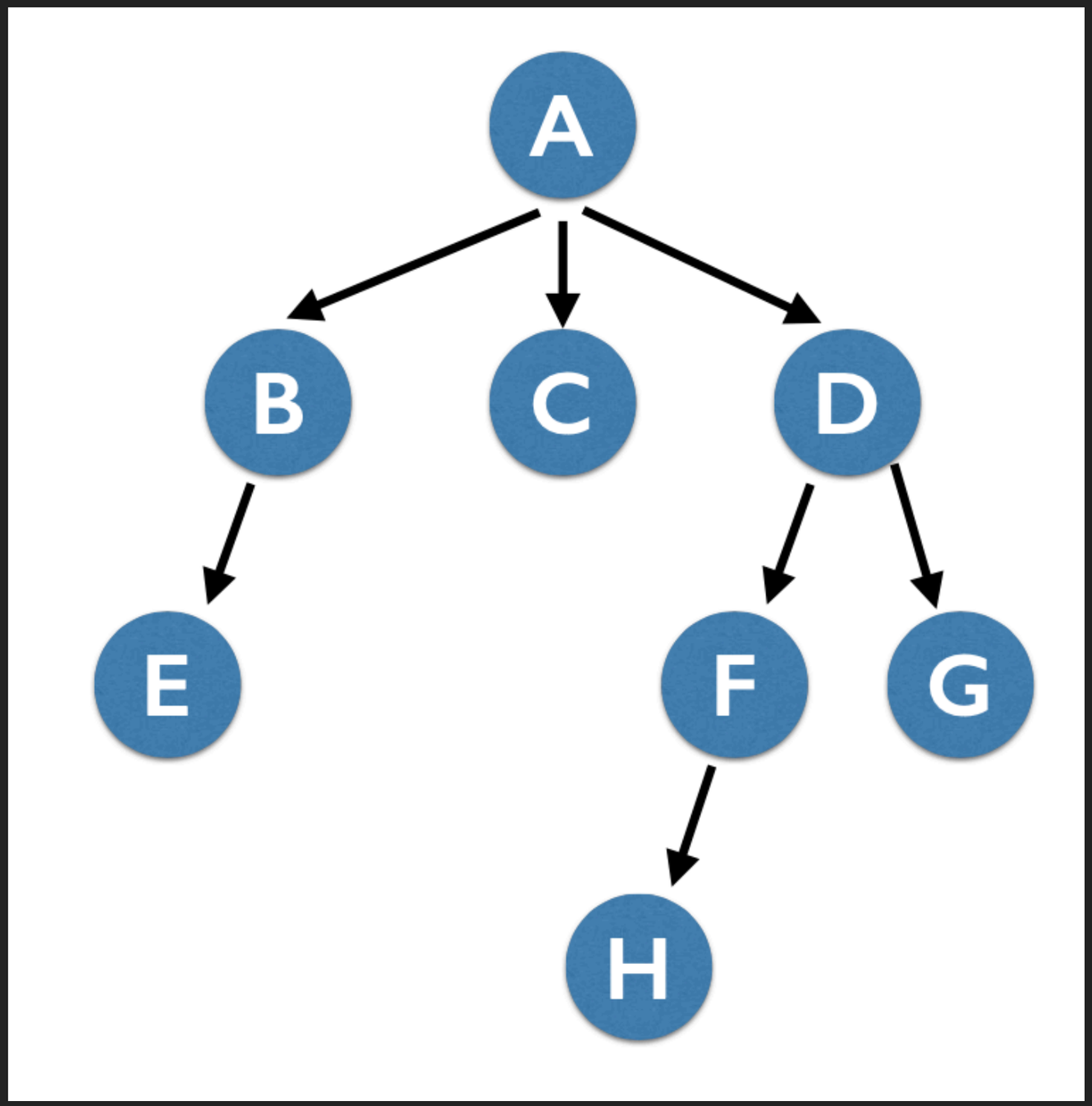
- ADS explanation

DATA STRUCTURE - TREES

DATA STRUCTURE - TREES

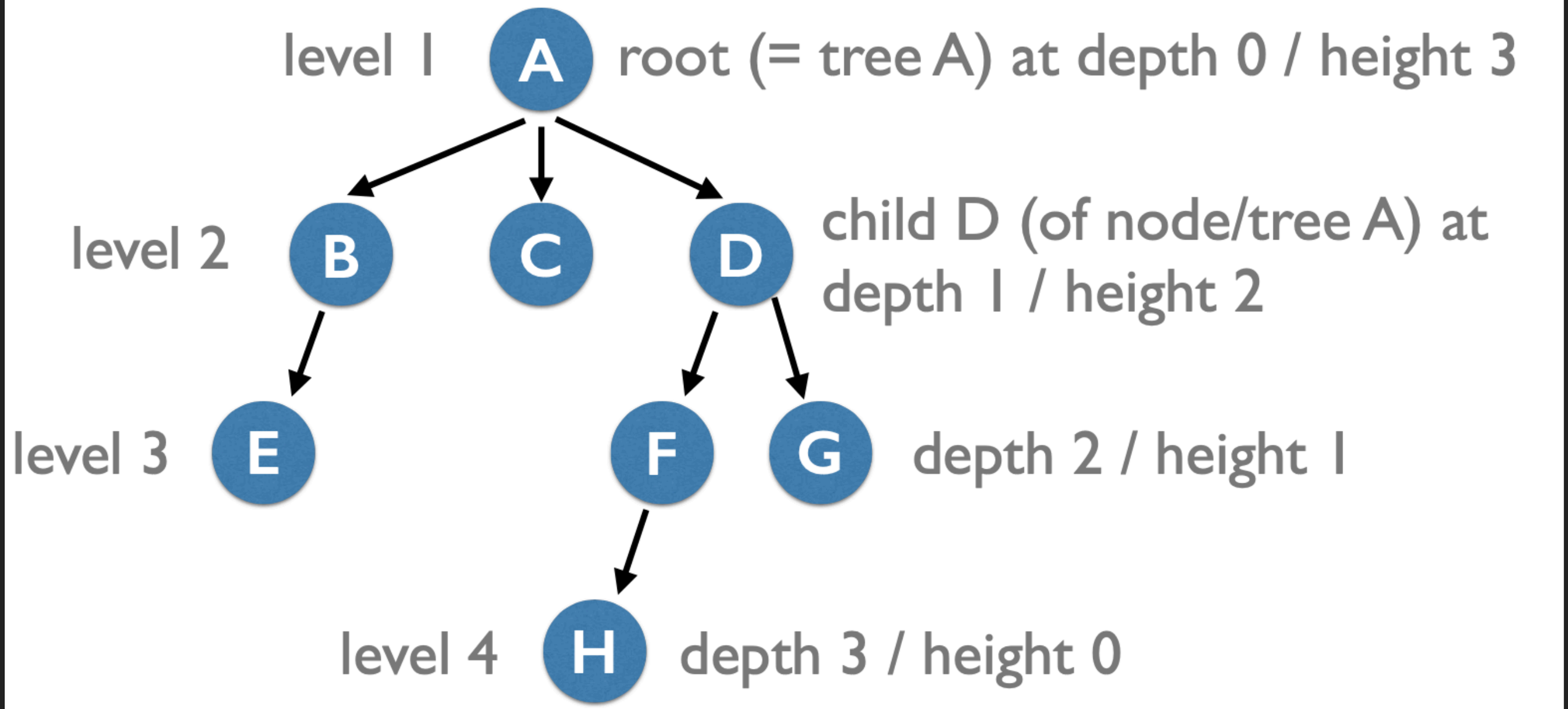
- Tree Terminology
  When working with trees, there are some terms that we need to know
  - Root: The initial node of the tree.
  - Node: Any single item in the tree.
  - Child: A node directly connected to another node when moving away from the root.
  - Parent: the converse notion of a child.

DATA STRUCTURE - LINKED LIST

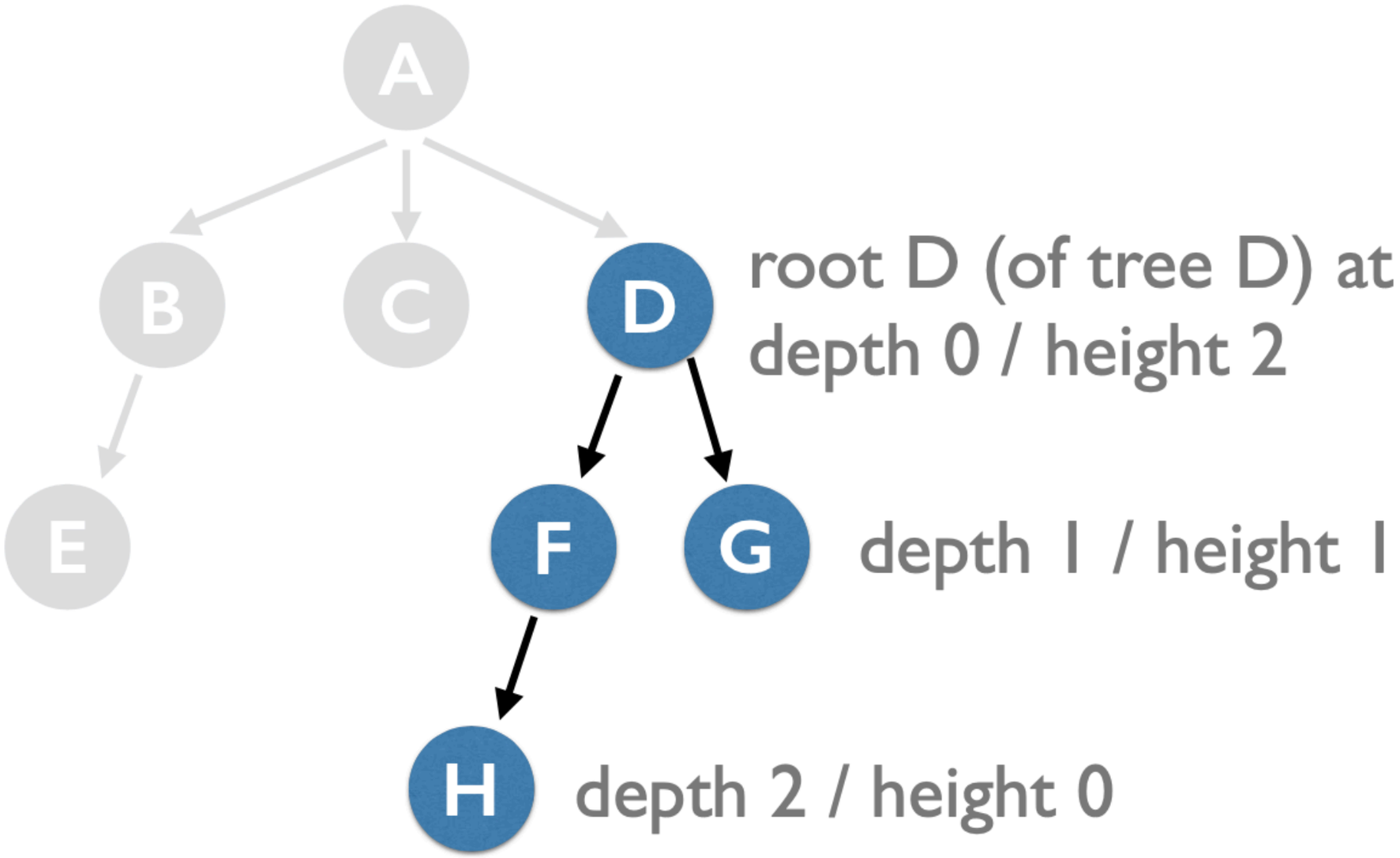


- Nodes contain values
- There's a primary 'root' node
- Children subtrees
- No duplicated children
- Trees can branch but never converge
- Final nodes are called 'leaves'
- Height of a tree is the longest path to a leaf

DATA STRUCTURE - LINKED LIST

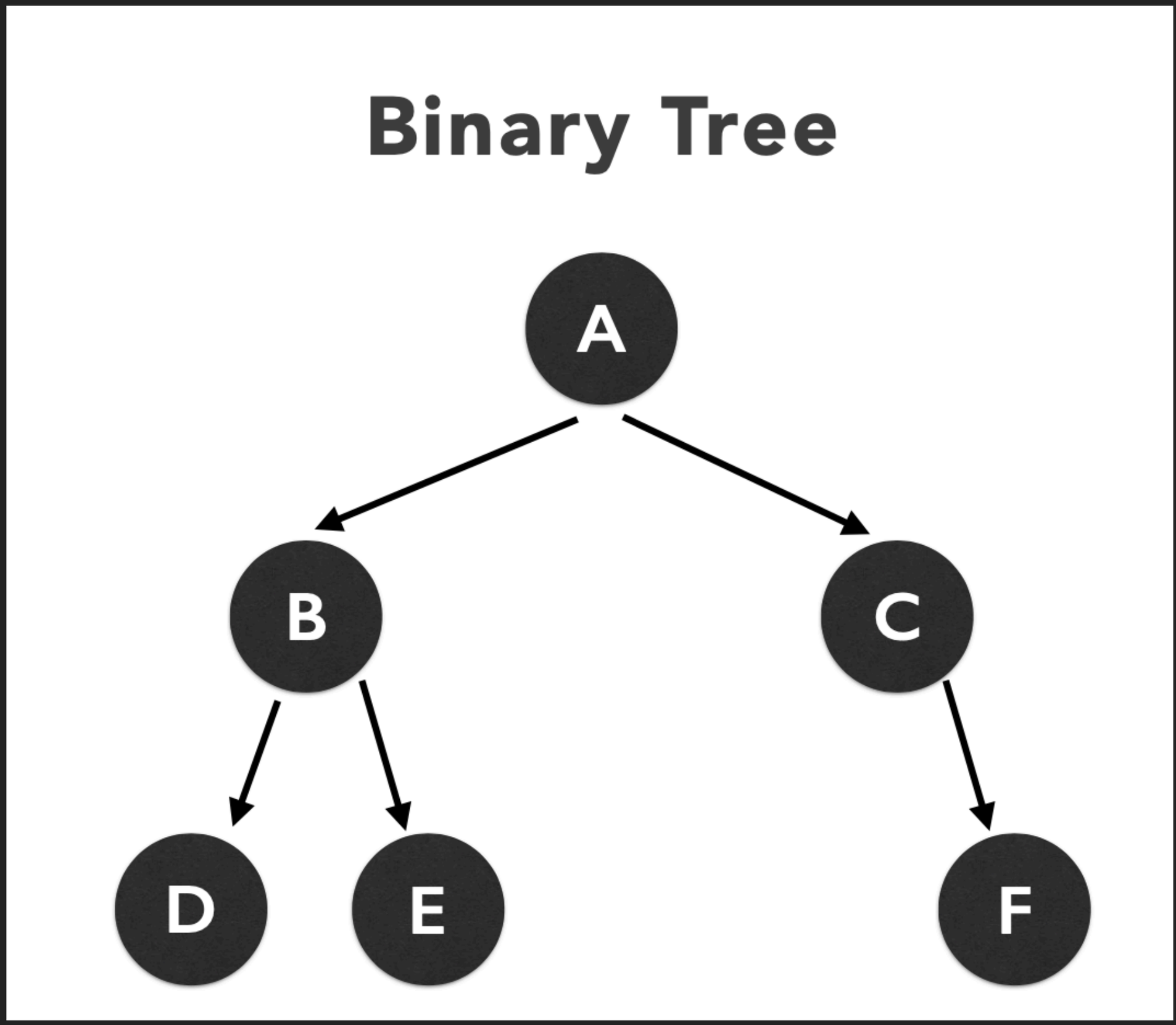


Every node is the root of a tree.
You might even say a node *represents* a tree.

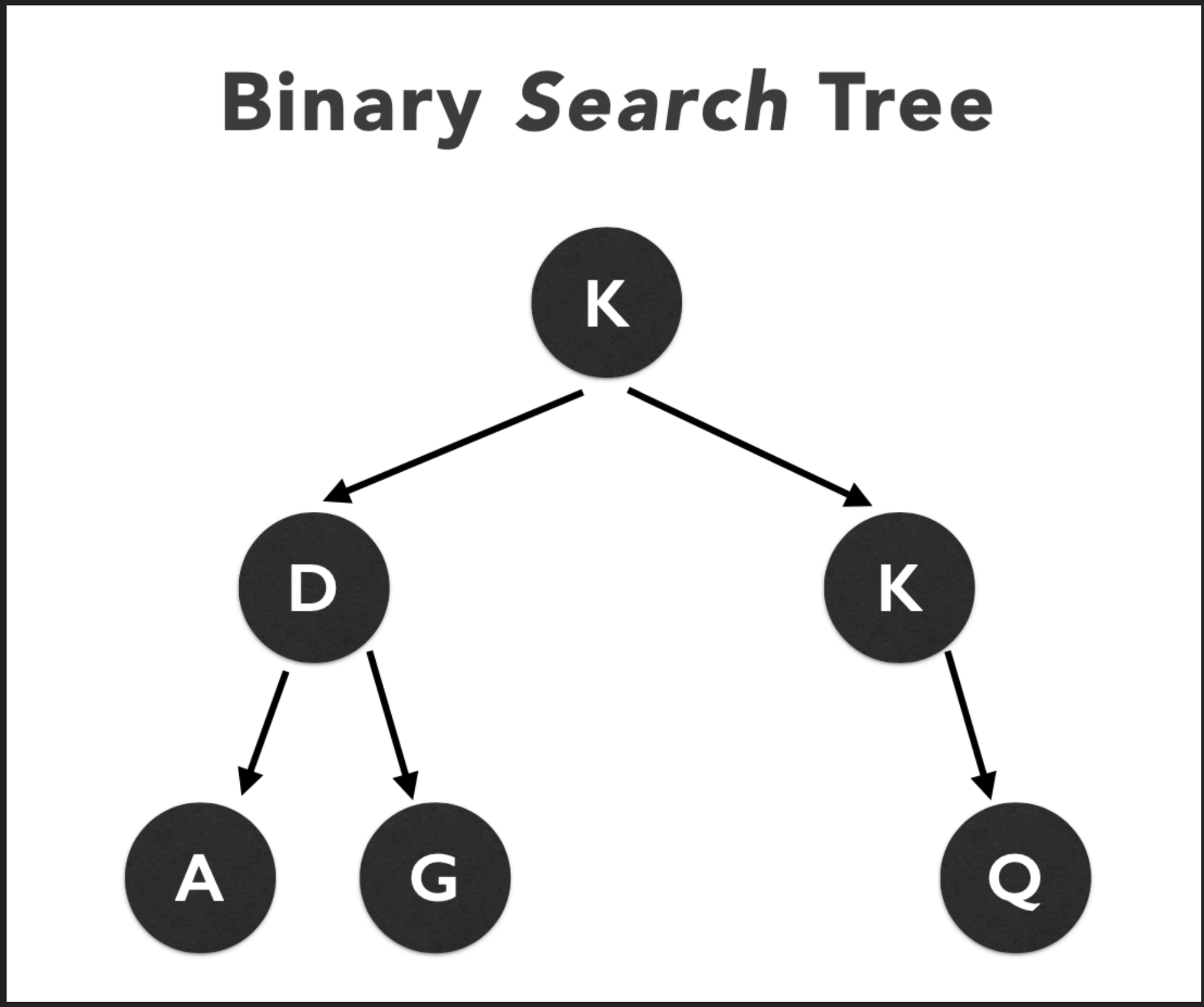


- Types of trees:
- There’re different types of trees that you can work with. The most common type of tree is a binary tree. This type of tree is so named because each parent node can only have two children. Other types of trees consist of a parent node being able to have more than 2 children.
- The deciding factor of which tree type to use is performance. Since trees are data structures, performance is measured in terms of inserting and retrieving data. In this case, the binary tree is the most efficient when it comes to these operations, and therefore the most used.

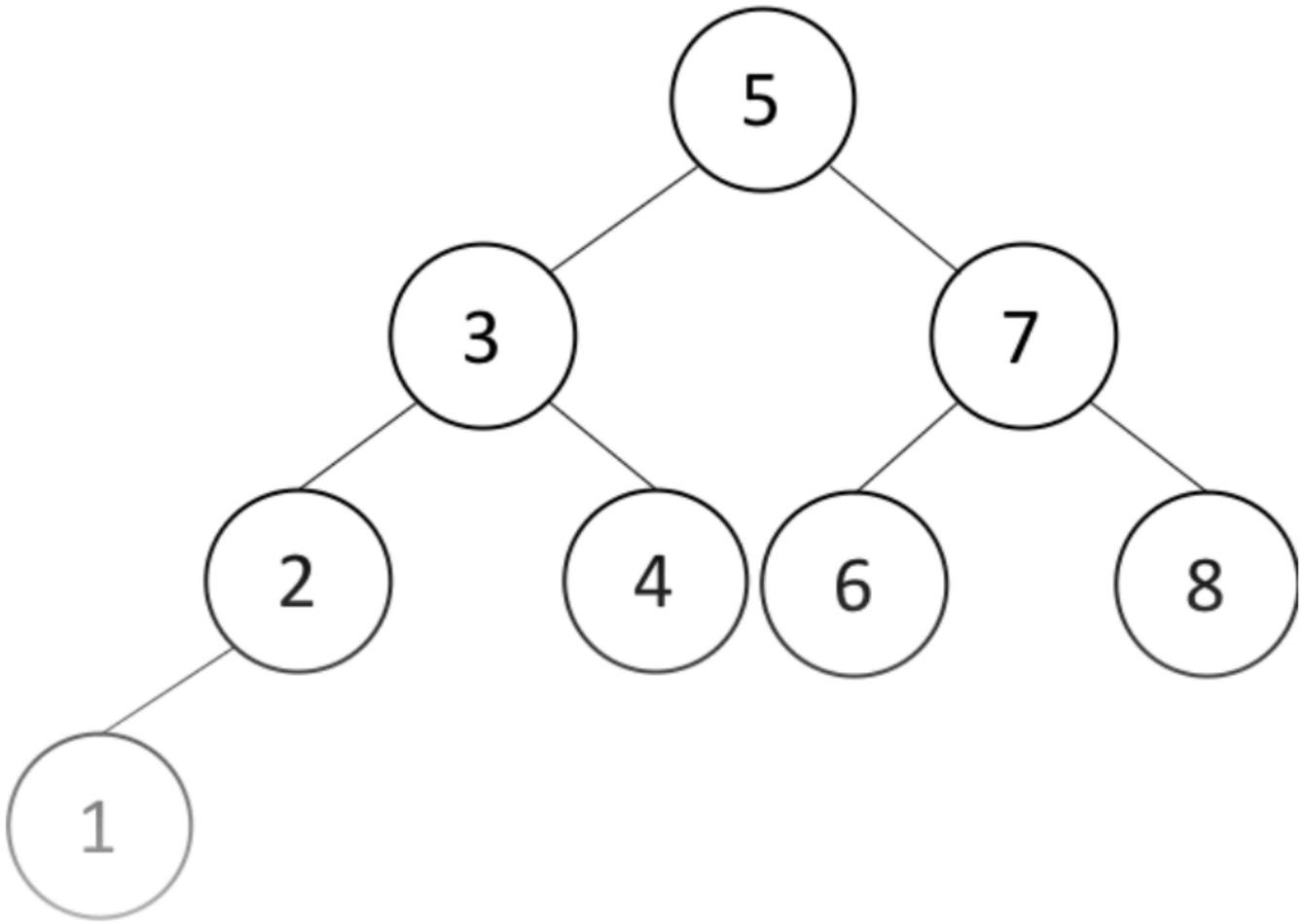
DATA STRUCTURE - TREE



DATA STRUCTURE - TREE



DATA STRUCTURE - TREE



The way our BST is sorted, is the leafs on the left side are less than the parent node. The value on the right are greater than or equal to.

Left descendants < root value <= right descendants

- Implementing a BST

```
class Node
{
    constructor(data)
    {
        this.data = data;
        this.left = null;
        this.right = null;
    }
}
```

- Implementing a BST

```
class BinarySearchTree
{
    constructor()
    {
        // root of a binary seach tree
        this.root = null;
    }

    // Methods we need to build
    // insert(data)
    // remove(data)

}
```

- Inserting into a BST

```
insert(data) {
  // Creating a node and initailising
  // with data
  var newNode = new Node(data);

  // root is null then node will
  // be added to the tree and made root.
  if (this.root === null)
    this.root = newNode;
  else

    // find the correct position in the
    // tree and add the node
    this.insertNode(this.root, newNode);
}
```

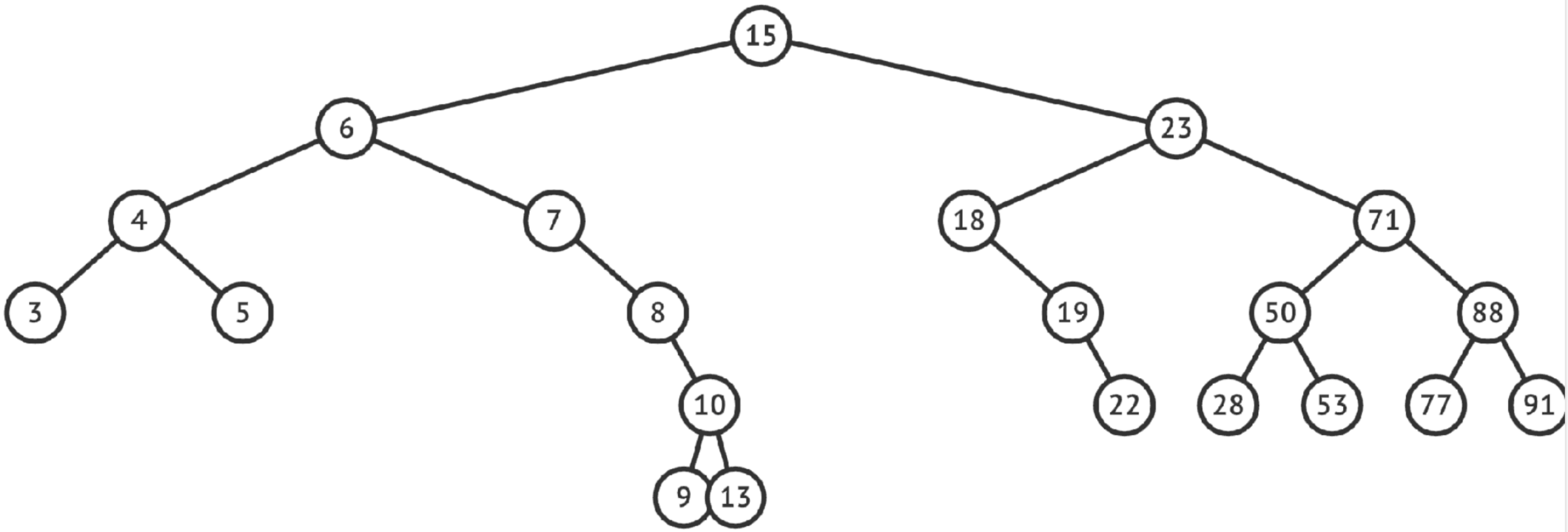
DATA STRUCTURE - TREE

```
insertNode(node, newNode) {
  // if the data is less than the node
  // data move left of the tree
  if (newNode.data < node.data) {
    // if left is null insert node here
    if (node.left === null)
      node.left = newNode;
    else

      // if left is not null recurr until
      // null is found
      this.insertNode(node.left, newNode);
  }

  // if the data is more than the node
  // data move right of the tree
  else {
    // if right is null insert node here
    if (node.right === null)
      node.right = newNode;
    else

      // if right is not null recurr until
      // null is found
      this.insertNode(node.right, newNode);
  }
}
```

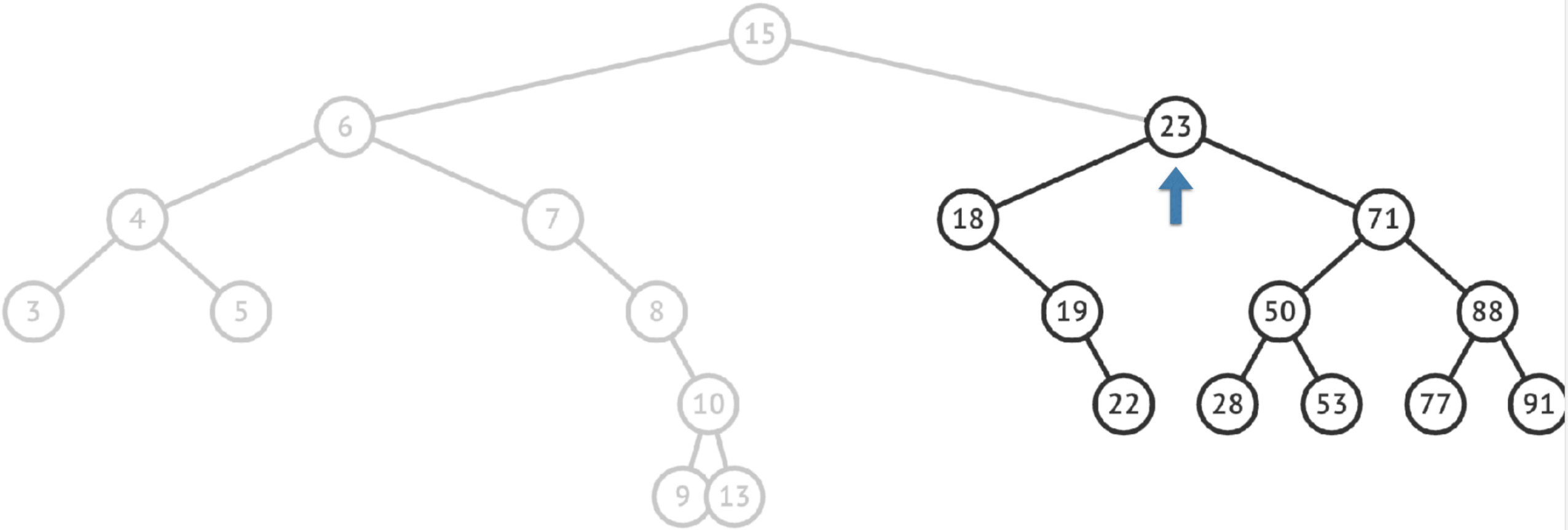
DATA STRUCTURE - TREE

- Now its up to you to build out the rest
  - Methods to Add:
    - Remove a specified value
    - Find the smallest value
    - Search the tree for a value (return the node if found)
    - Get the root node

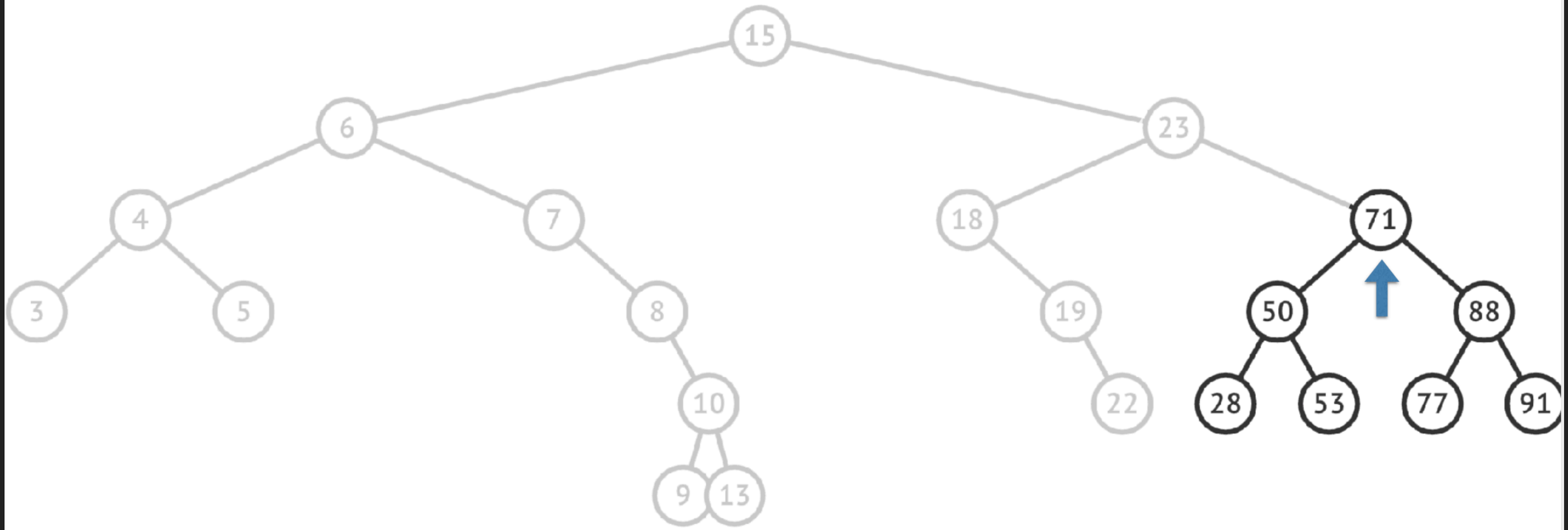
Does this BST contain the value 28?



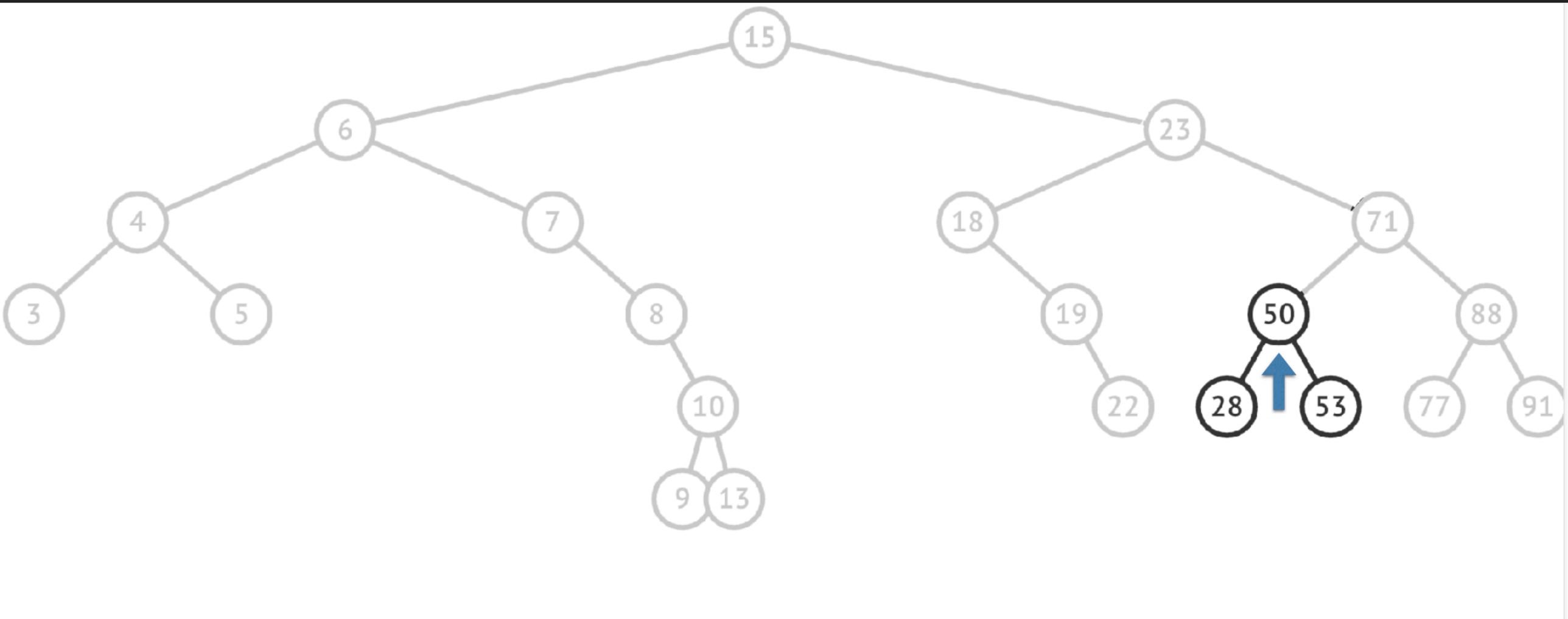
Does this BST contain the value 28?



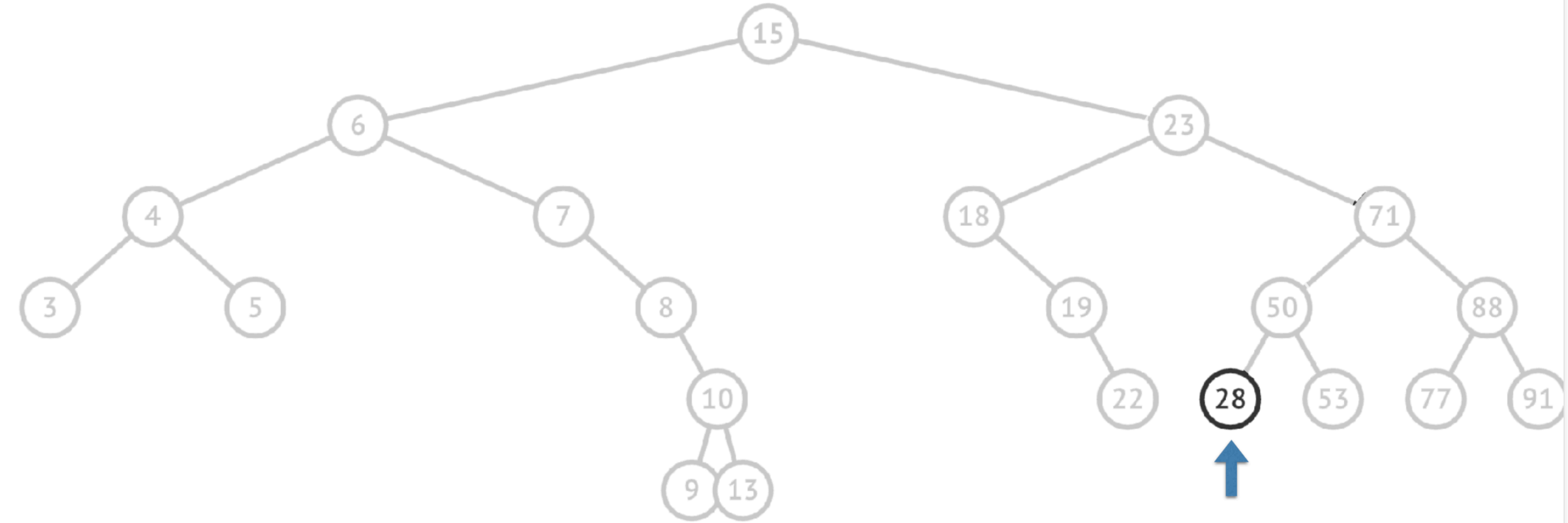
DATA STRUCTURE - TREE



DATA STRUCTURE - TREE



DATA STRUCTURE - TREE



- Now its up to you to build out the rest
  - Extra: Build out the different types of searches for a BST
    - Breadth-first search
    - Depth-first search