FIND WHETHER 2 BINARY TREES ARE EXACTLY THE SAME

FIND WHETHER 2 BINARY TREES ARE EXACTLY THE SAME

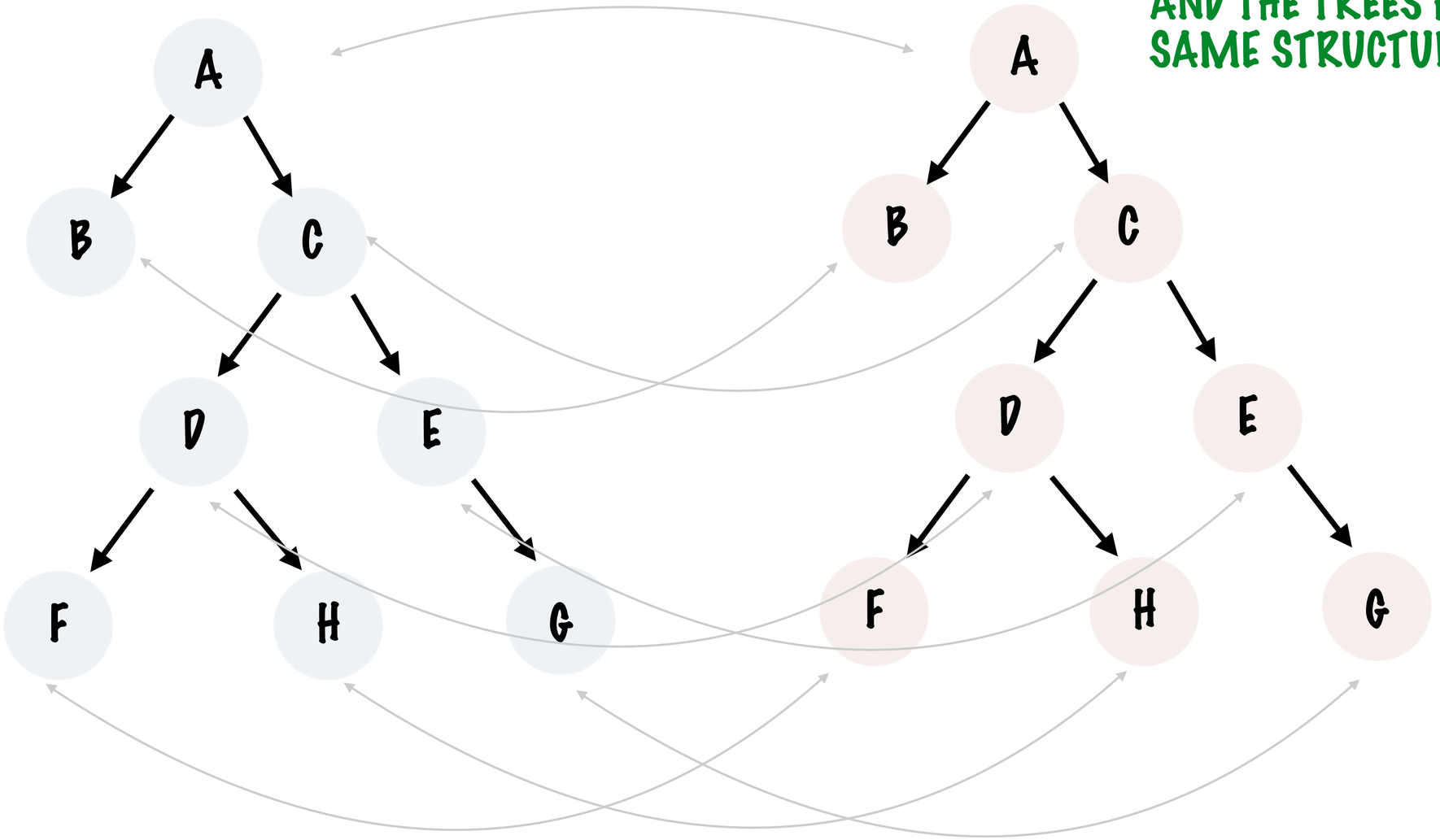
A BINARY TREE IS ONE WHERE EVERY NODE CAN HAVE A MAXIMUM OF TWO CHILDREN - A LEFT CHILD AND RIGHT CHILD

TWO BINARY TREES ARE THE SAME IF

1. EVERY CORRESPONDING NODE HAS THE SAME VALUE
2. THE STRUCTURE OF THE TREE AT EVERY CORRESPONDING NODE IS THE SAME

HERE ARE SOME SAMPLE TREES...

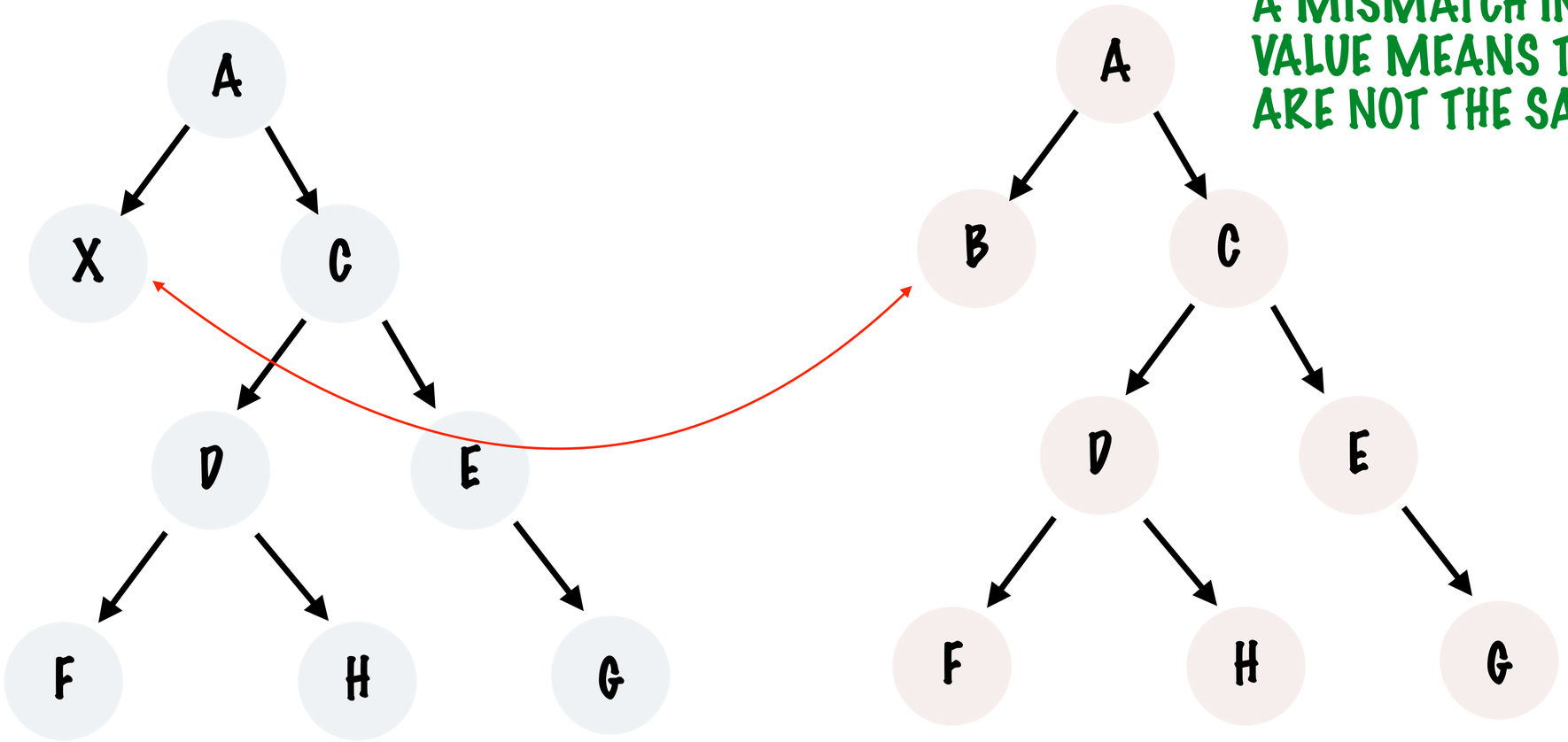
EVERY CORRESPONDING NODE HAS THE SAME VALUE AND THE TREES HAVE THE SAME STRUCTURE



SOME MORE...

TREES ARE NOT THE SAME

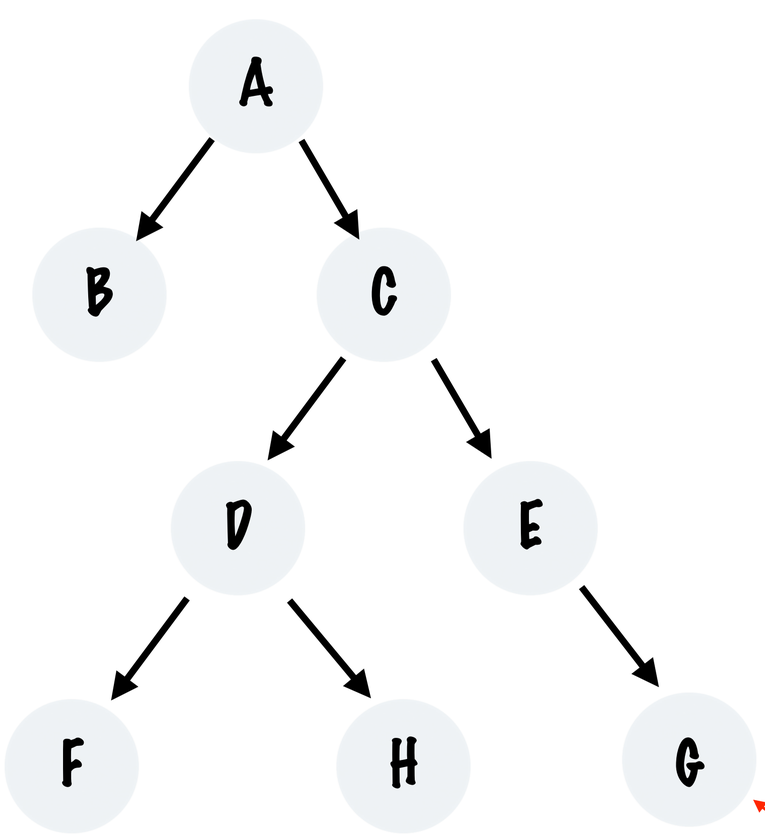
A MISMATCH IN A SINGLE VALUE MEANS THE TREES ARE NOT THE SAME

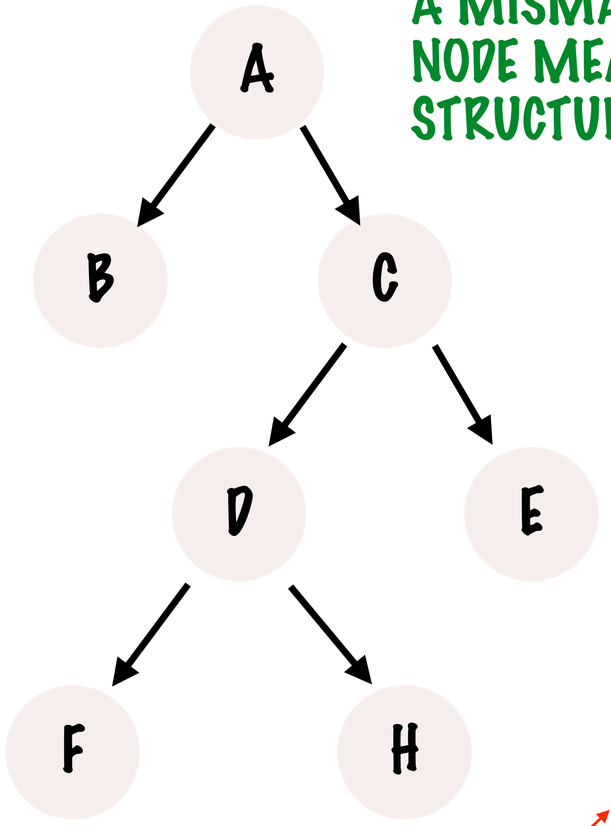


LAST ONE...

TREES ARE NOT THE SAME

A MISMATCH IN A SINGLE NODE MEANS THE TREE STRUCTURES ARE DIFFERENT





COMPARE THE NODE VALUES AT EVERY LEVEL

CHECK THAT NODES ARE PRESENT FOR EVERY CORRESPONDING NODE IN BOTH TREES

REPEAT TILL THE ENTIRE TREE HAS BEEN TRAVERSED

WHAT IS THE BASE CASE?

1. A NULL CURRENT NODE ON ONE TREE SHOULD CORRESPOND TO A NULL IN THE OTHER TREE

2. NODE VALUES SHOULD BE THE SAME AT EVERY NODE

WHAT IS THE RECURSIVE CASE?

CHECK THAT THE SUB-TREE ROOTED AT EVERY NODE IS THE SAME

A SINGLE NODE

```java
public static class Node {
    private int id;
    private Node left;
    private Node right;

    public Node(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public Node getLeft() {
        return left;
    }

    public void addChildren(Node left, Node right) {
        this.left = left;
        this.right = right;
    }

    public Node getRight() {
        return right;
    }

}
```

EACH NODE HAS A LEFT AND RIGHT CHILD, EITHER CHILD CAN BE NULL

THE VALUE STORED IN THE NODE

HELPER METHODS TO ADD CHILDREN AND ACCESS THE LEFT AND RIGHT CHILD

SAME TREE

```
public static boolean sameTree(Node head1, Node head2) {
    if (head1 == null && head2 == null) {
        return true;
    }
    if (head1 == null) {
        return false;
    } else if (head2 == null) {
        return false;
    }


    if (sameTree(head1.getLeft(), head2.getLeft())
            && sameTree(head1.getRight(), head2.getRight())) {
        return head1.getId() == head2.getId();
    }

    return false;
}
```

THE HEAD NODES OF BOTH THE TREES ARE THE ARGUMENTS

IF BOTH ARE NULL THE TREE IS THE SAME, RETURN TRUE

IF ONE IS NULL AND THE OTHER ISN'T THE STRUCTURE OF THE NODES ARE NOT THE SAME, SO IT'S NOT THE SAME TREE

CHECK IF THE TREES ROOTED AT THE LEFT AND RIGHT CHILD ARE THE SAME TREES

CHECK THE VALUES ON THE CURRENT NODE

WE HAVE TO VISIT EVERY NODE IN BOTH TREES SO 2N NODES, THE COMPLEXITY IS O(N)