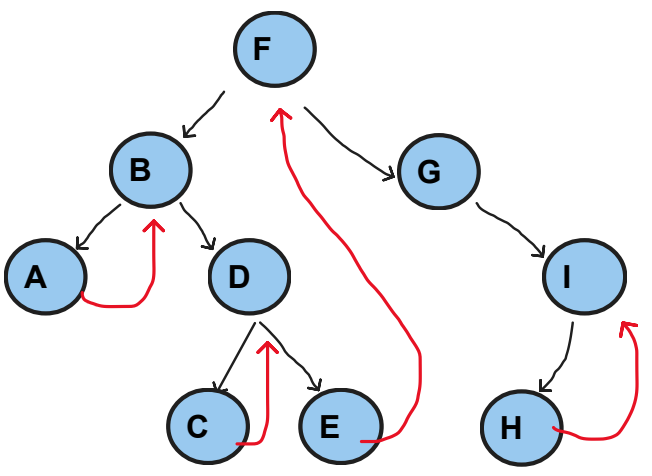
Morris Traversal

INORDER

↳ right->if left not exist or either left subtree is completely processed

↳ something mark so that we get to knowwhether left subtree is completely processed of not



↳ print
a. if left is null
b. if thread is cut down then print current node
mark of left subtree is completely processed or not :
if thread already exists

INORDER

CODE

```
ArrayList<Integer> inOrder(Node root) {
        // Code
        ArrayList<Integer>ans=new ArrayList<>();
        Node curr=root;
        while(curr!=null){
            Node leftnode=curr.left;
            if(leftnode==null){
                ans.add(curr.data);
                curr=curr.right;
            }else{
                Node rightmostnode=rightmostnode(leftnode,curr);
                if(rightmostnode.right==null){
                    rightmostnode.right=curr;
                    curr=curr.left;
                }else{
                    rightmostnode.right=null;
                    ans.add(curr.data);
                    curr=curr.right;
                }
            }
        }
        return ans;
    }
    Node rightmostnode(Node leftnode,Node curr){
        Node itr=leftnode;
        while(itr.right!=null && itr.right!=curr){
            itr=itr.right;
        }
        return itr;
    }
```

Pre order

CODE

```
ArrayList<Integer> inOrder(Node root) {
        // Code
        ArrayList<Integer>ans=new ArrayList<>();
        Node curr=root;
        while(curr!=null){
            Node leftnode=curr.left;
            if(leftnode==null){
                ans.add(curr.data);
                curr=curr.right;
            }else{
                Node rightmostnode=rightmostnode(leftnode,curr);
                if(rightmostnode.right==null){
                    ans.add(curr.data);
                    rightmostnode.right=curr;
                    curr=curr.left;
                }else{
                    rightmostnode.right=null;
                    curr=curr.right;
                }
            }
        }
        return ans;
    }
    Node rightmostnode(Node leftnode,Node curr){
        Node itr=leftnode;
        while(itr.right!=null && itr.right!=curr){
            itr=itr.right;
        }
        return itr;
    }
```