Tree Traversals Part 1

By the end of this video you will be able to…

- Explain the need to visit data in different orderings
- Author a preorder traversals

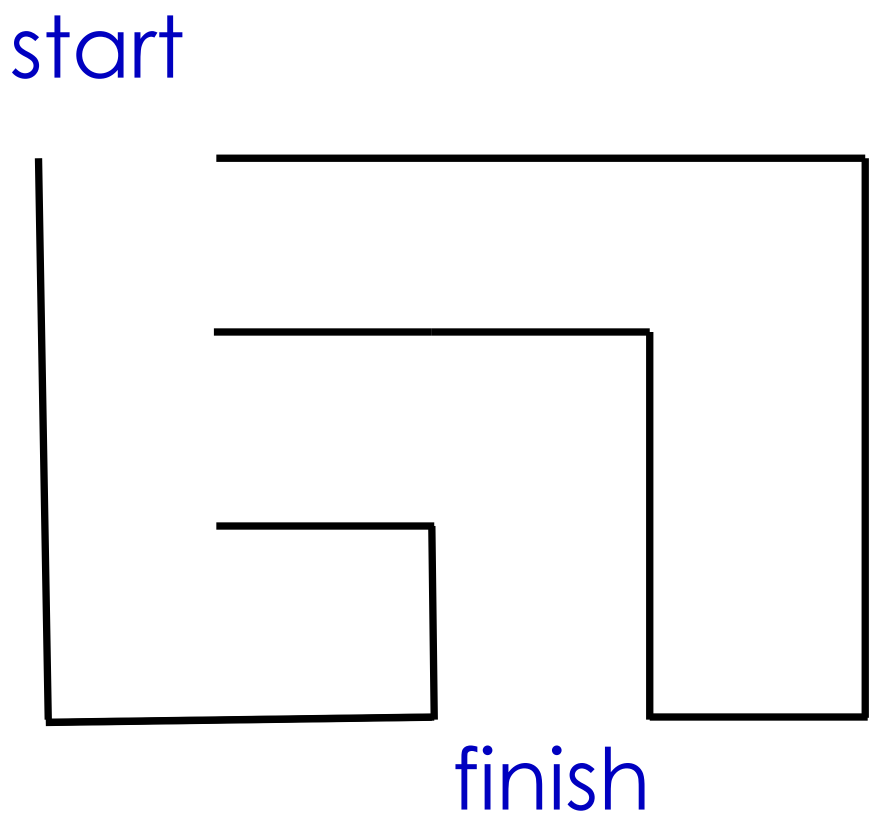
Traversals

Traversals

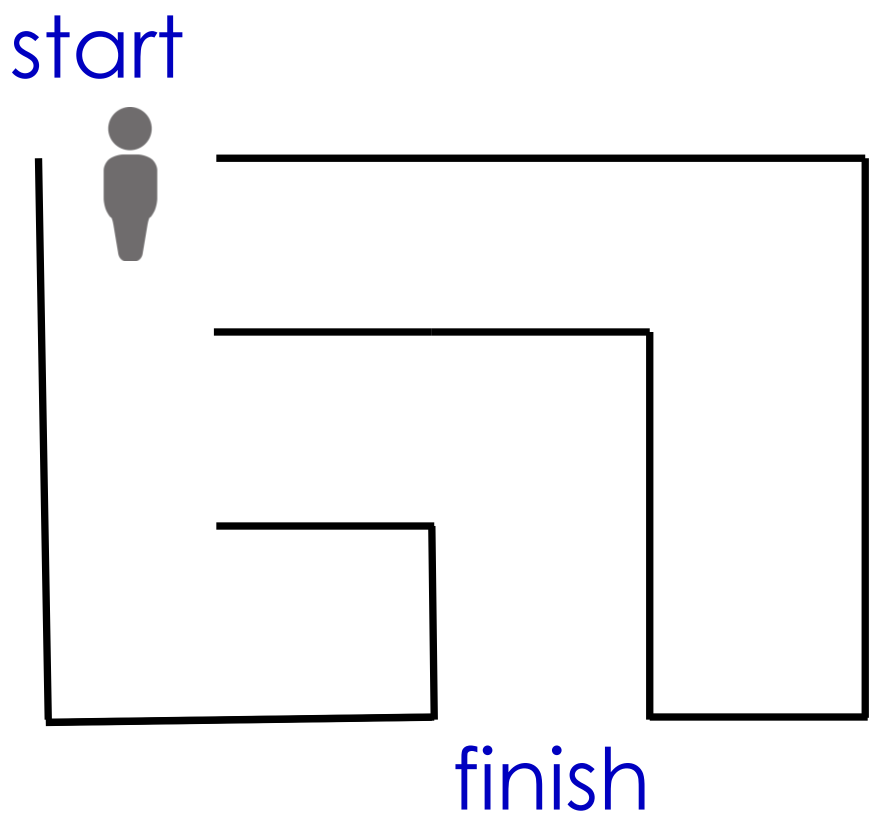
Warning: These first examples are really graphs. We'll visit graphs in detail in the next course.

Maze Traversal

Imagine this is a hedge maze

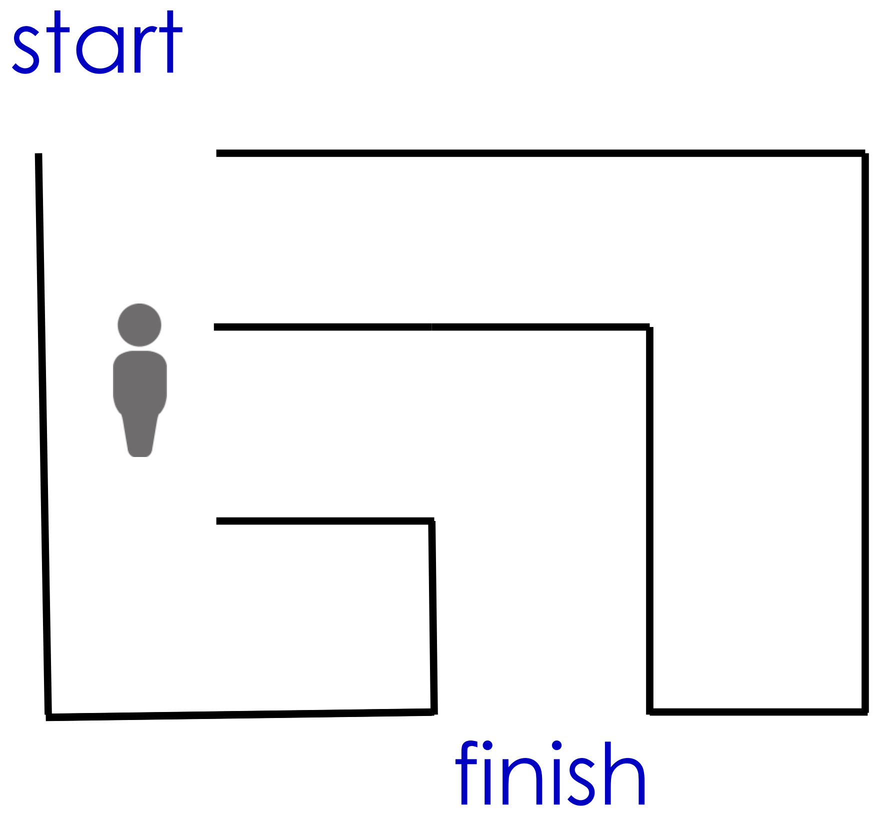


Maze Traversal



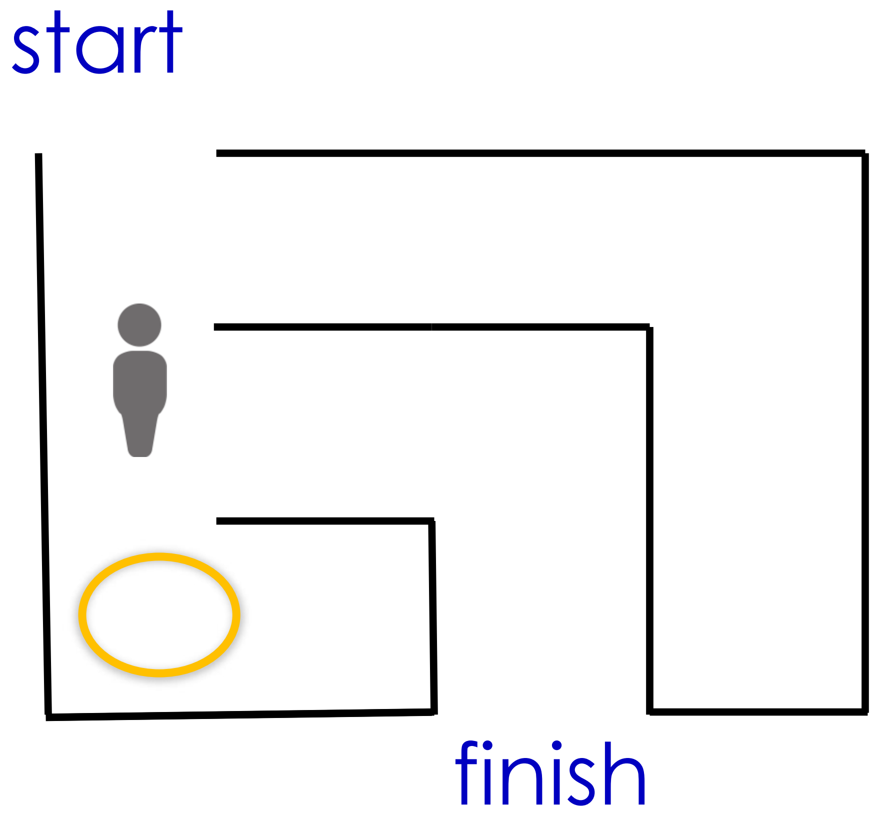
Maze Traversal

What's my next step?



Maze Traversals

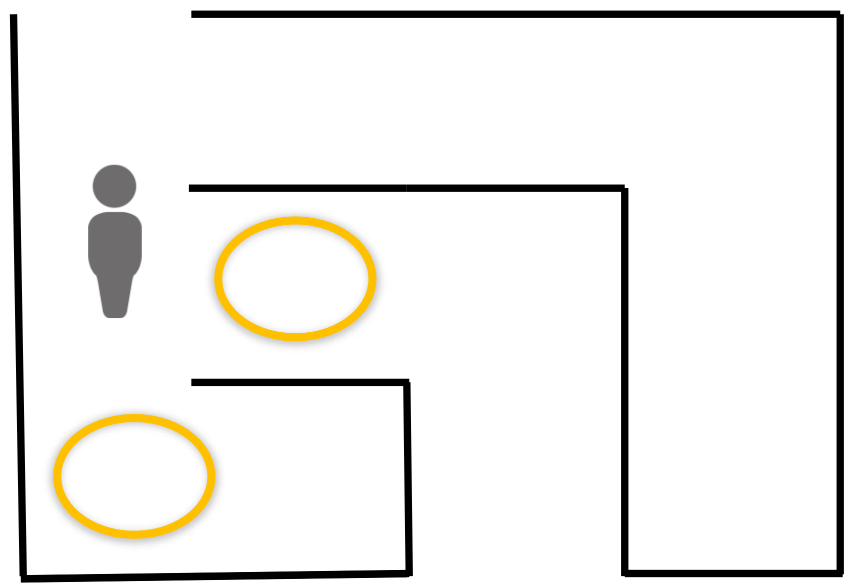
What's my next step?



Maze Traversal

What's my next step?

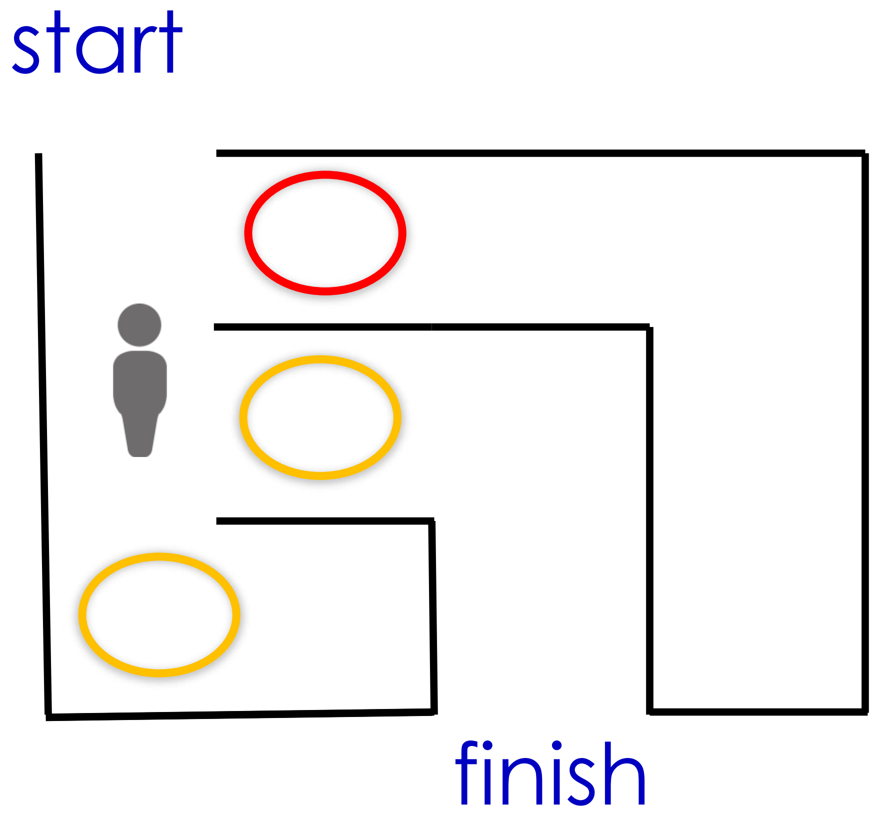
start



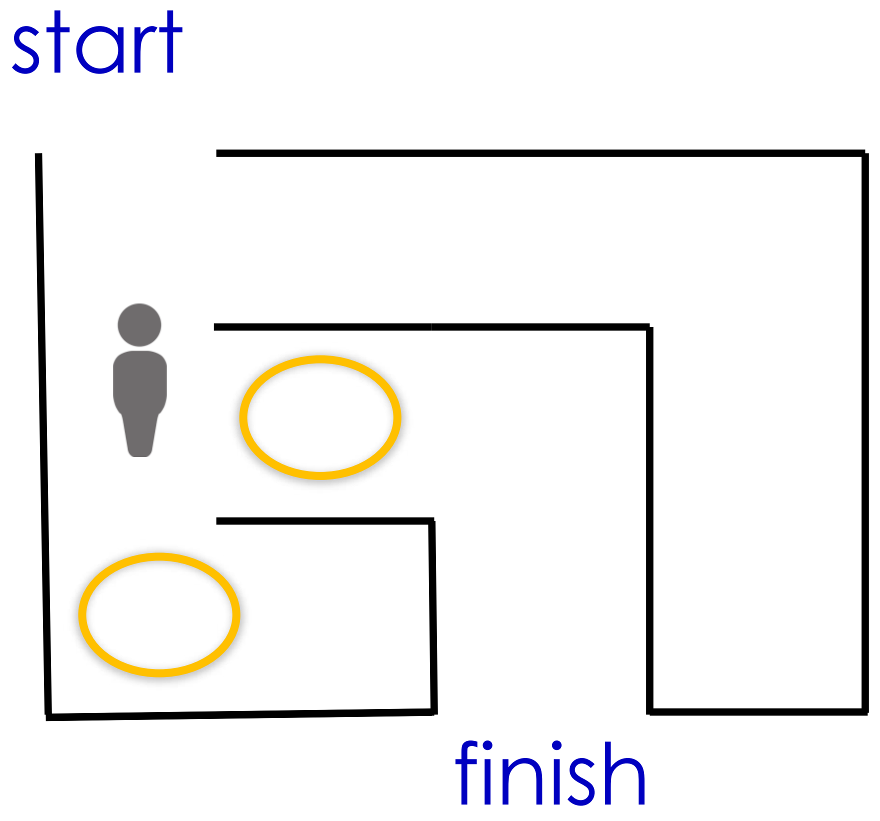
finish

Maze Traversal

What's my next step?



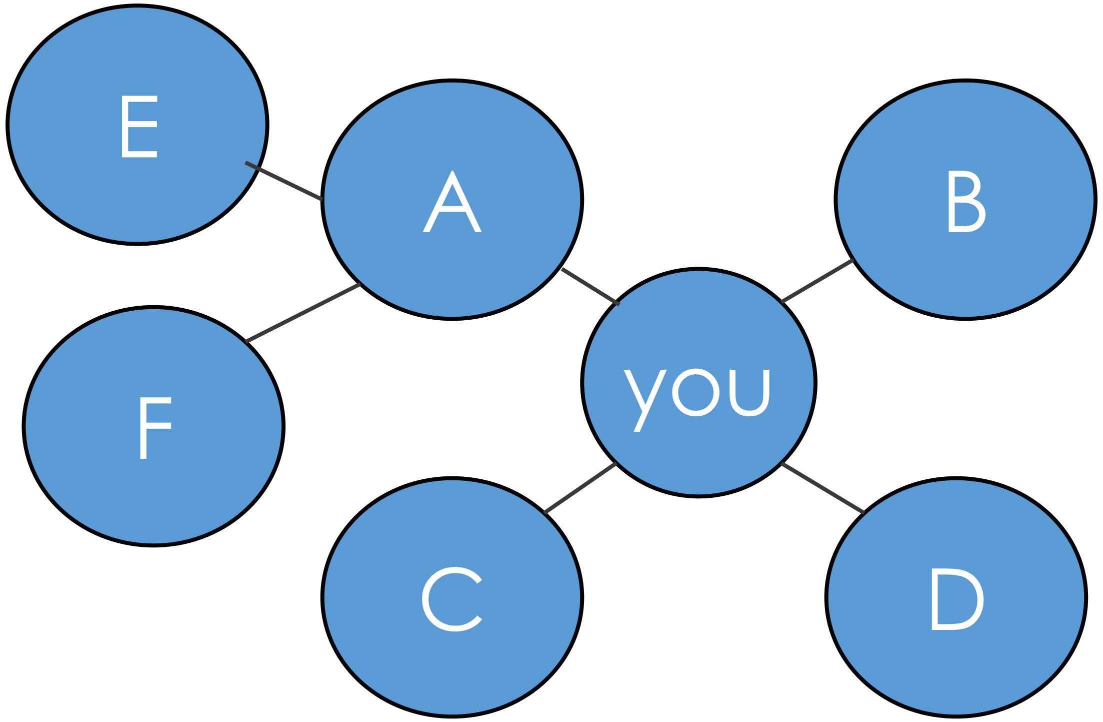
Maze Traversal



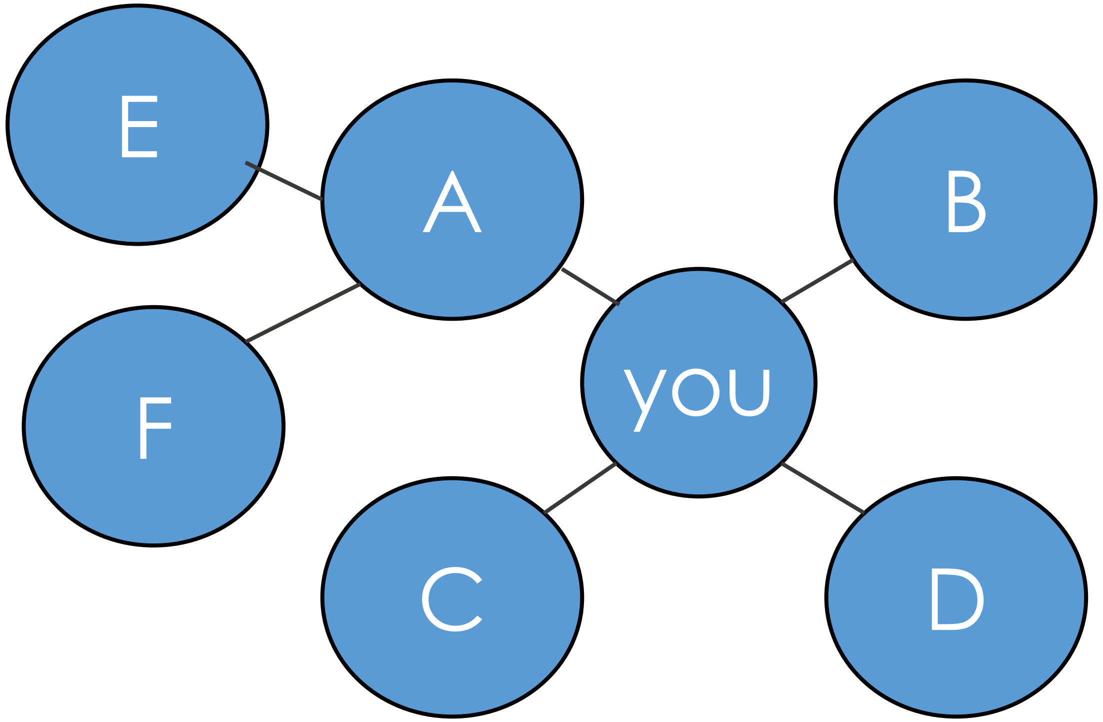
Mazes benefit from "Depth First Traversals"

Social Network

How closely are you connected with D?



Social Network

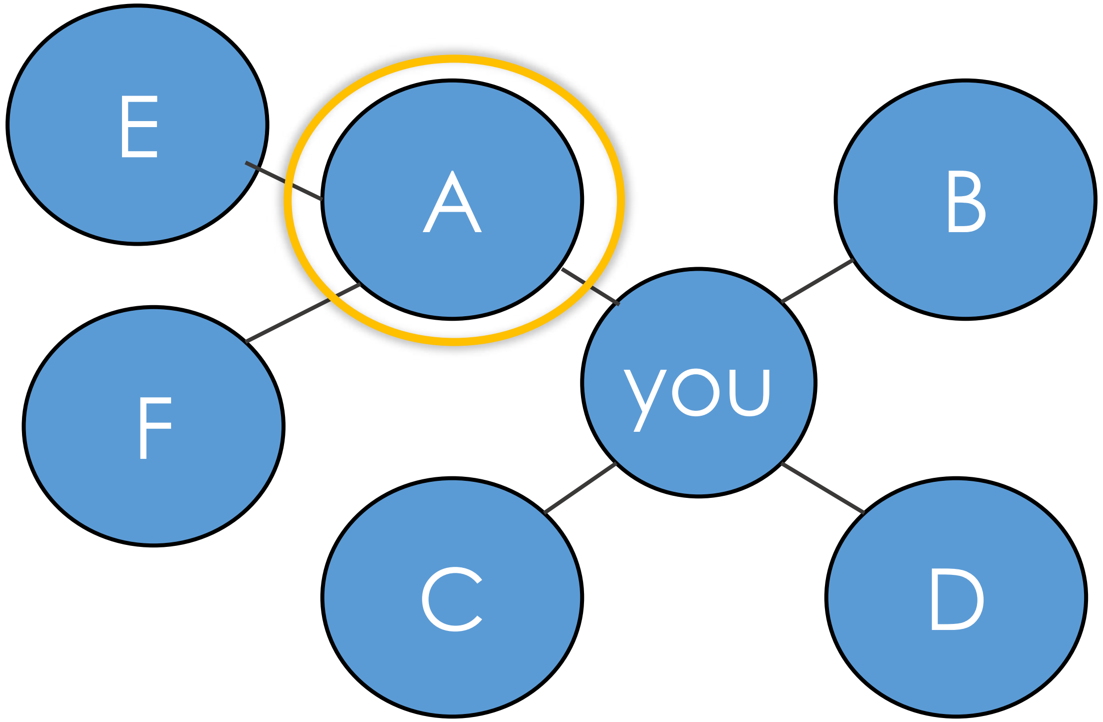


How closely are you connected with D?

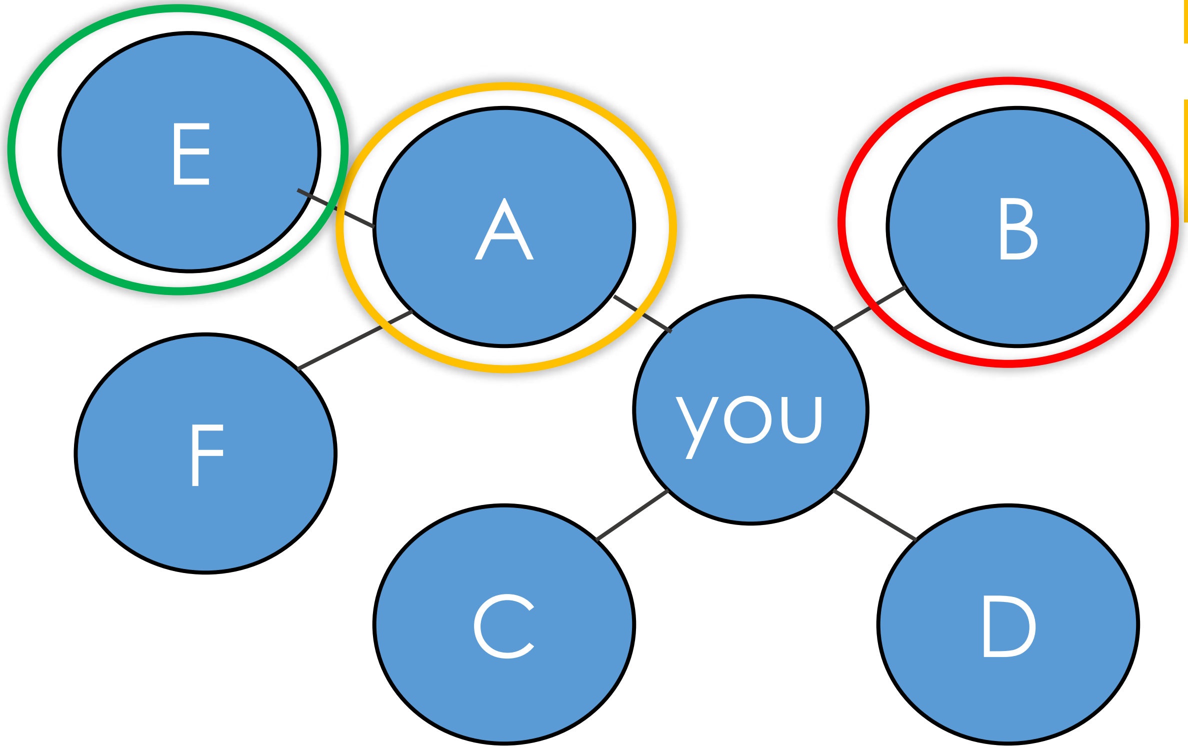
Suppose you have a list of your friends and each of your friends have lists

Social Network

How closely are you connected with D?



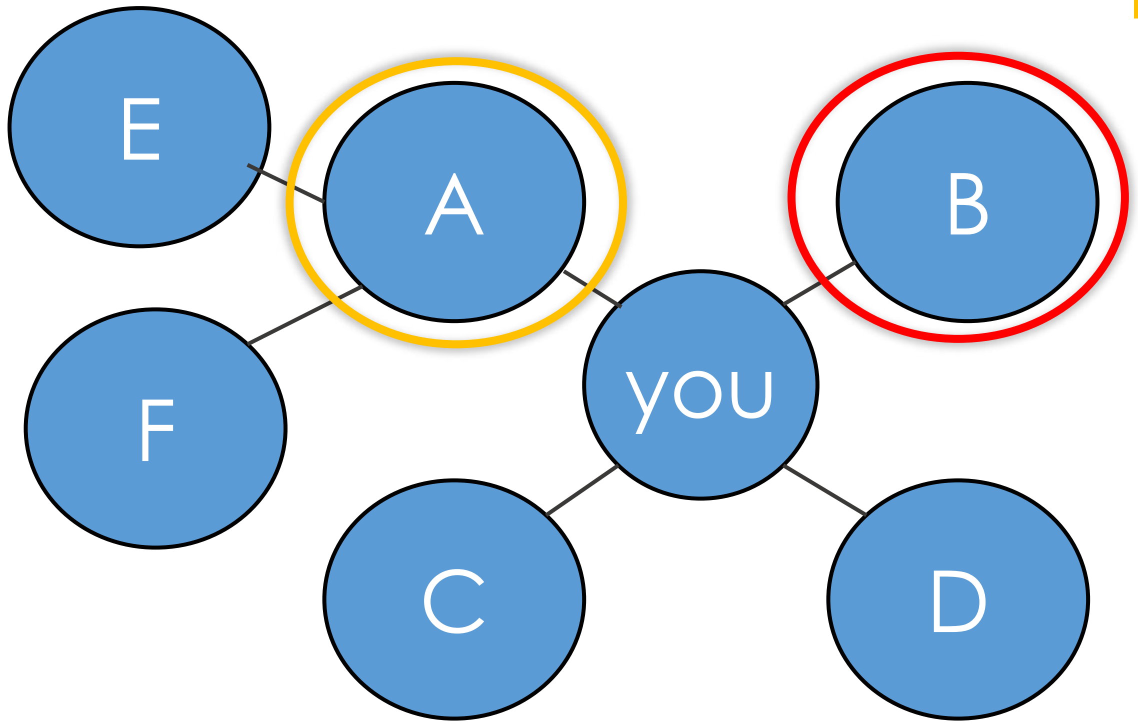
Social Network



How closely are you connected with D?

What's my next step?

Social Network



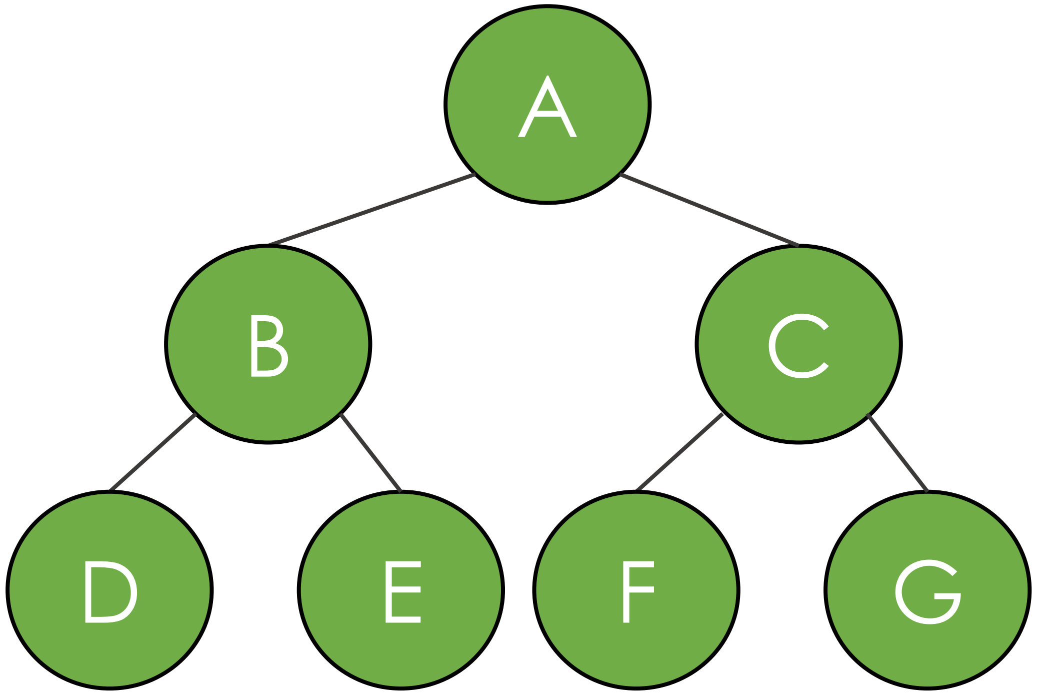
How closely are you connected with D?

This problem benefits from "Breadth First Traversals"

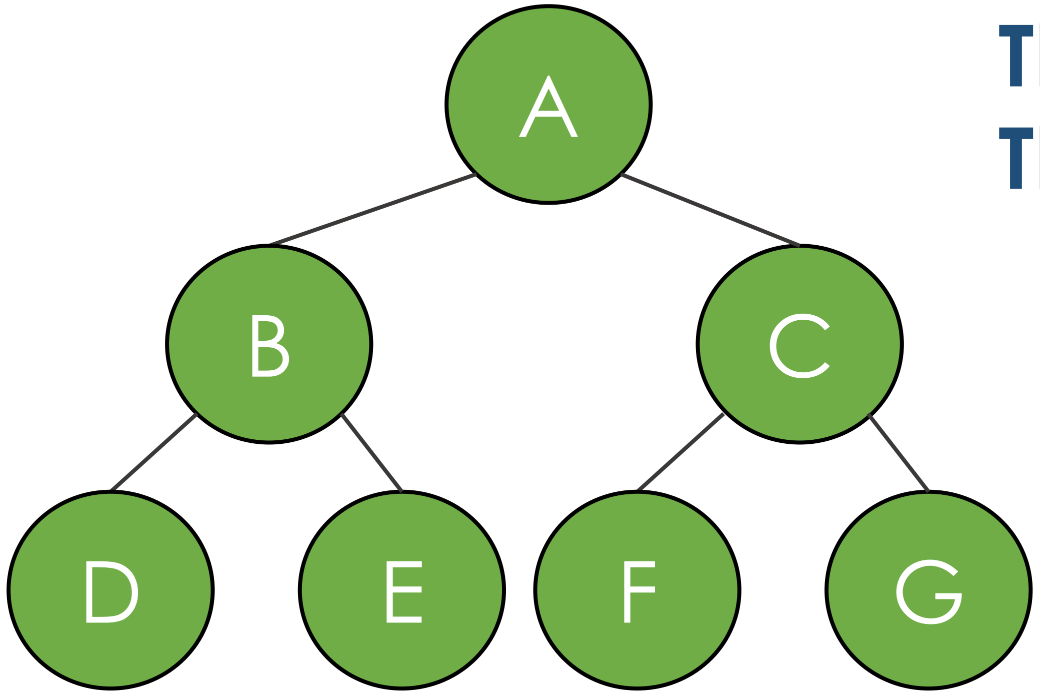
Traversals

Bottom line: Order we visit matters and we'll make choices based on our needs

Tree Traversals



Preorder Traversal



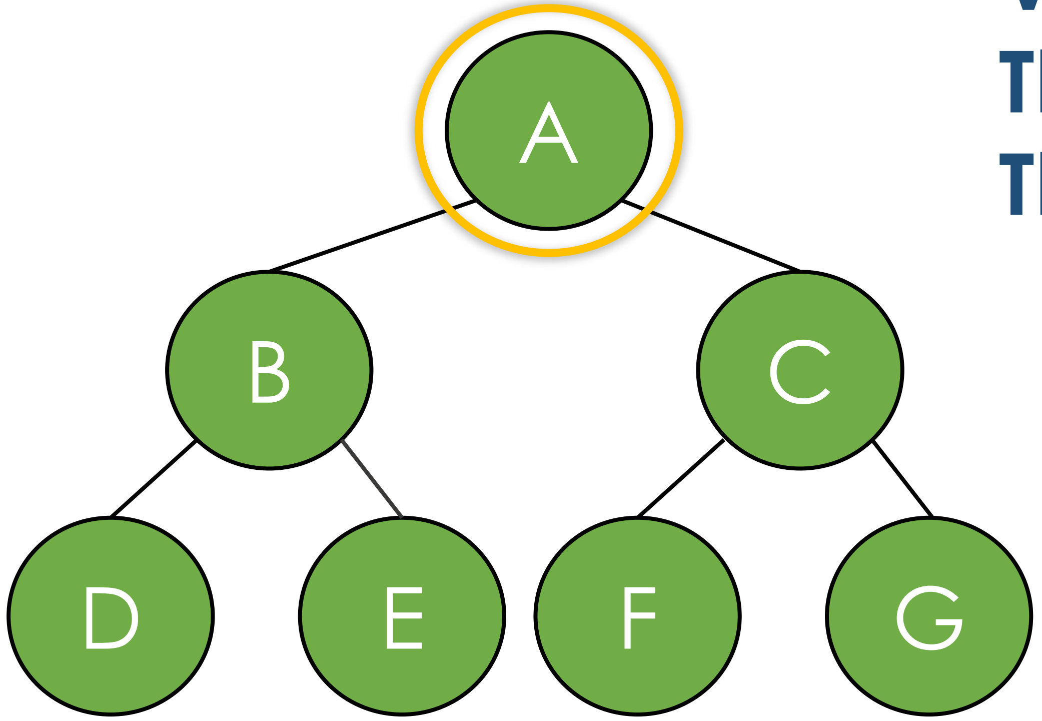
Idea:

Visit yourself

Then visit all your left subtree

Then visit all your right subtree

Preorder Traversal



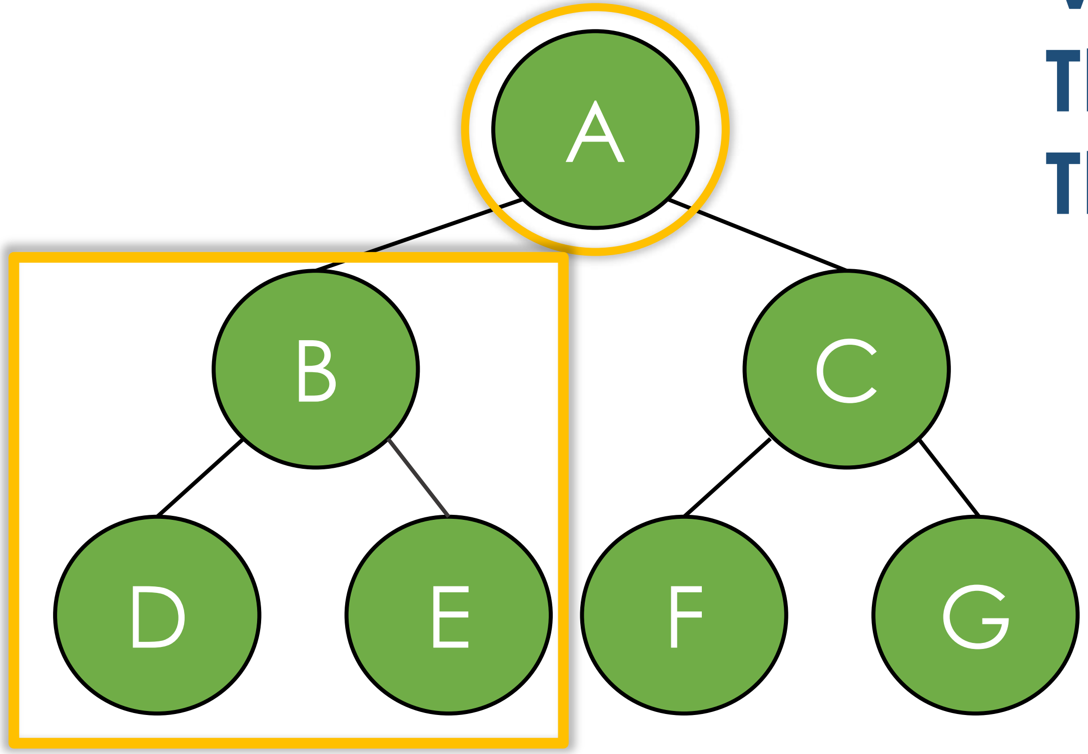
Idea:

Visit yourself

Then visit all your left subtree

Then visit all your right subtree

Preorder Traversal



Idea:

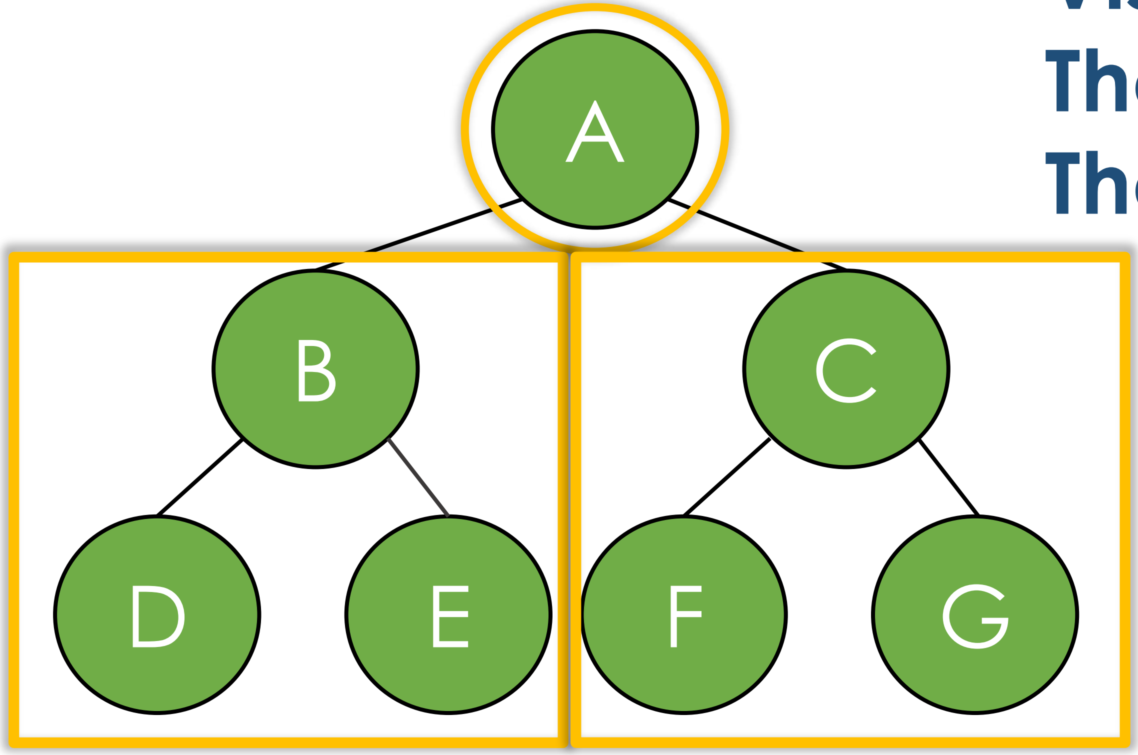
Visit yourself

Then visit all your left subtree

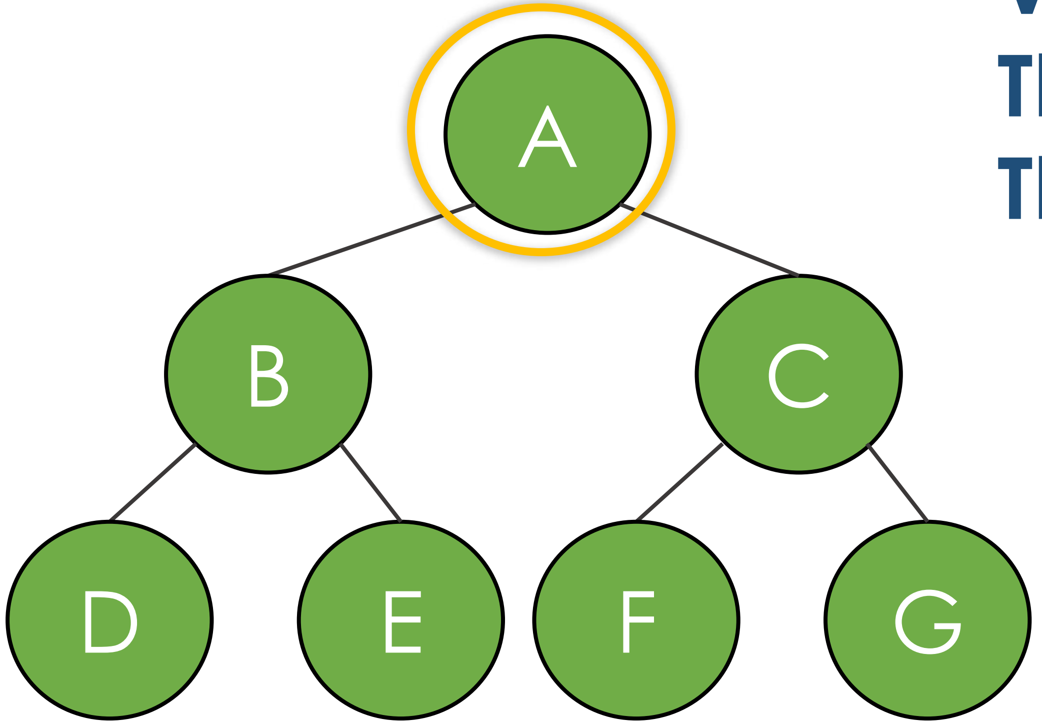
Then visit all your right subtree

Preorder Traversal

Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

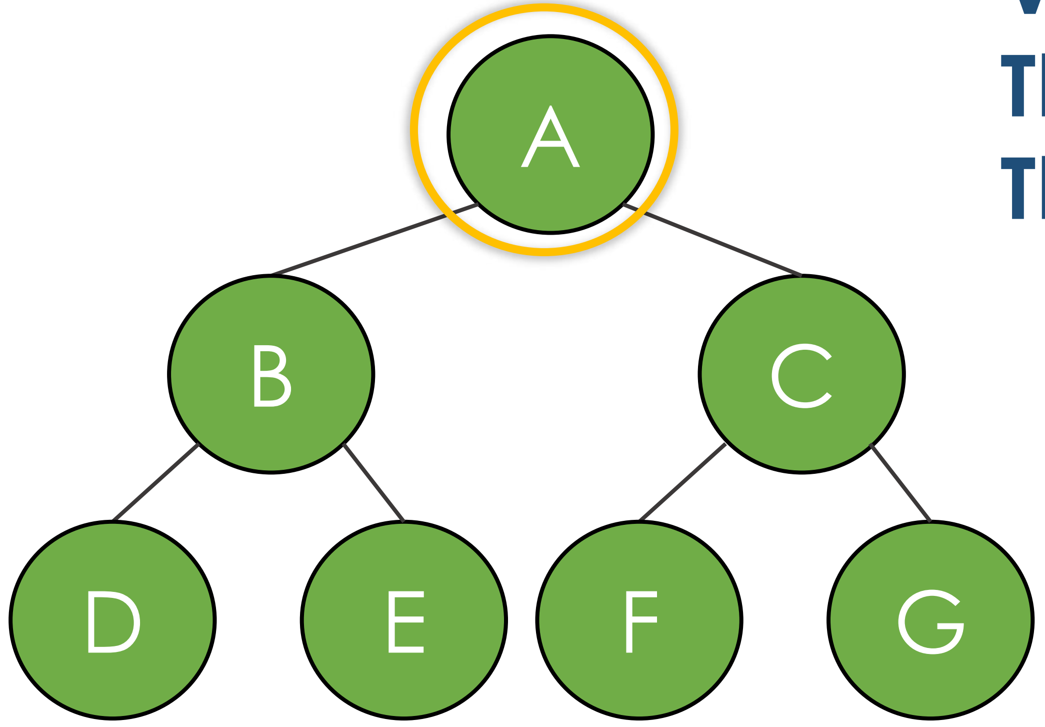


Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

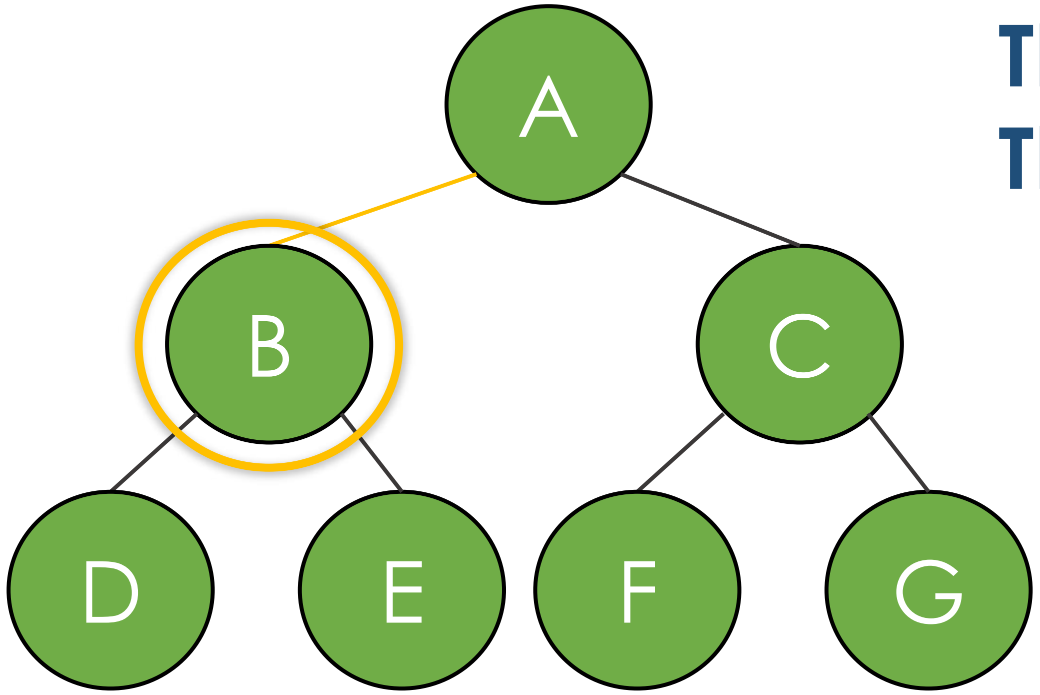
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A

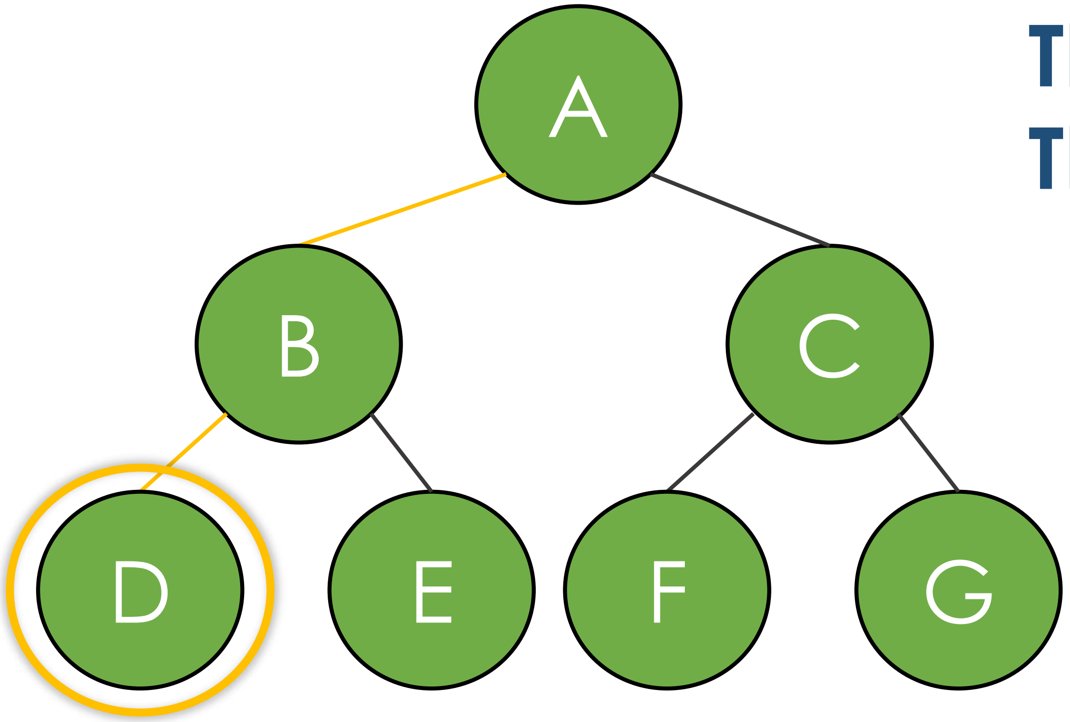
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B

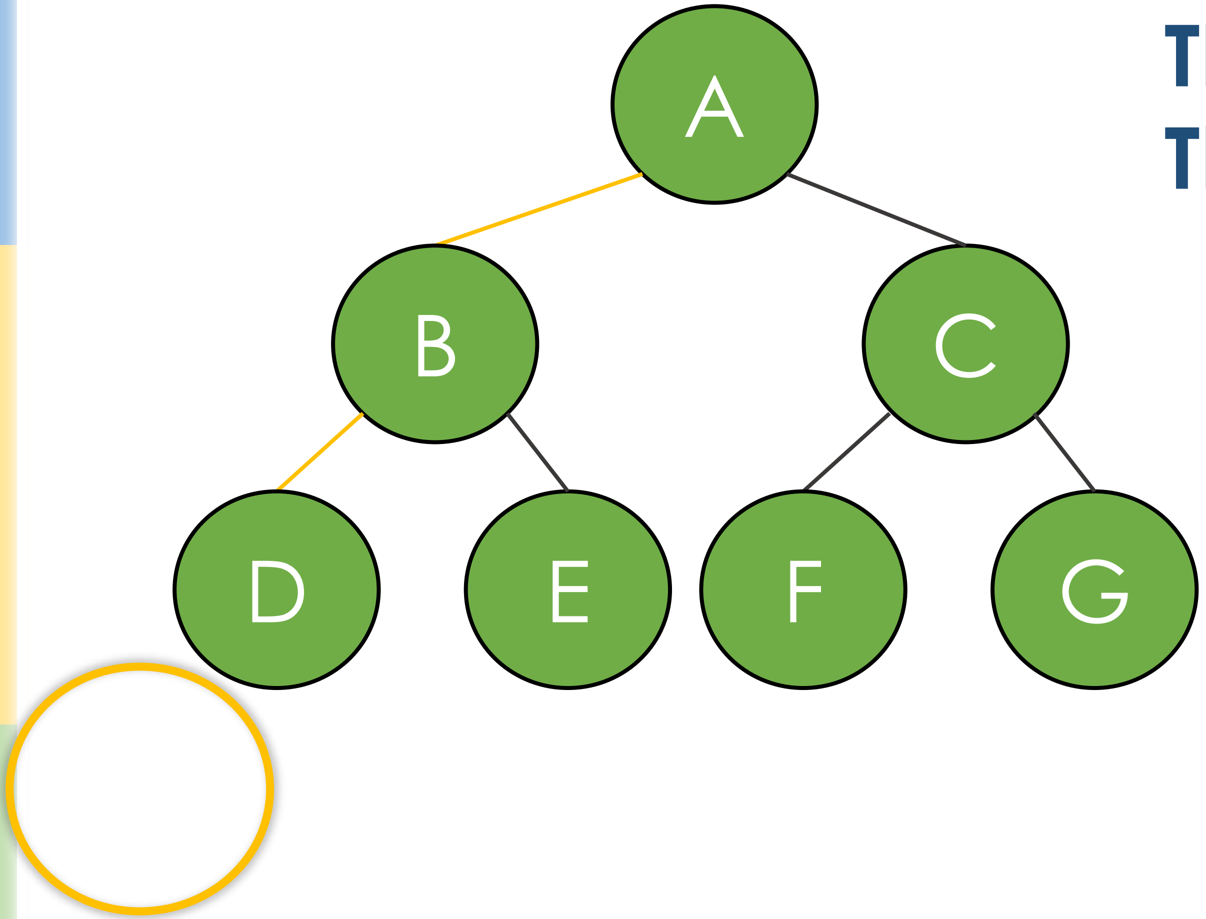
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D

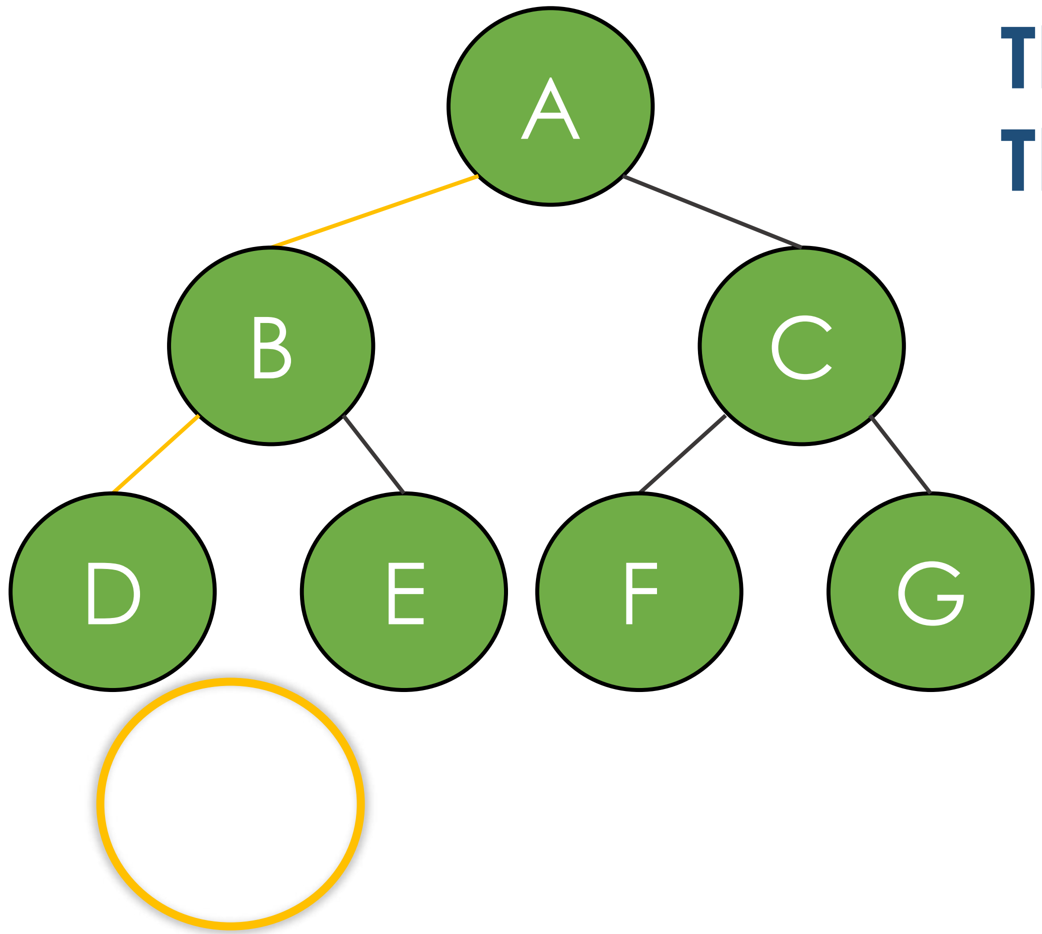
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D

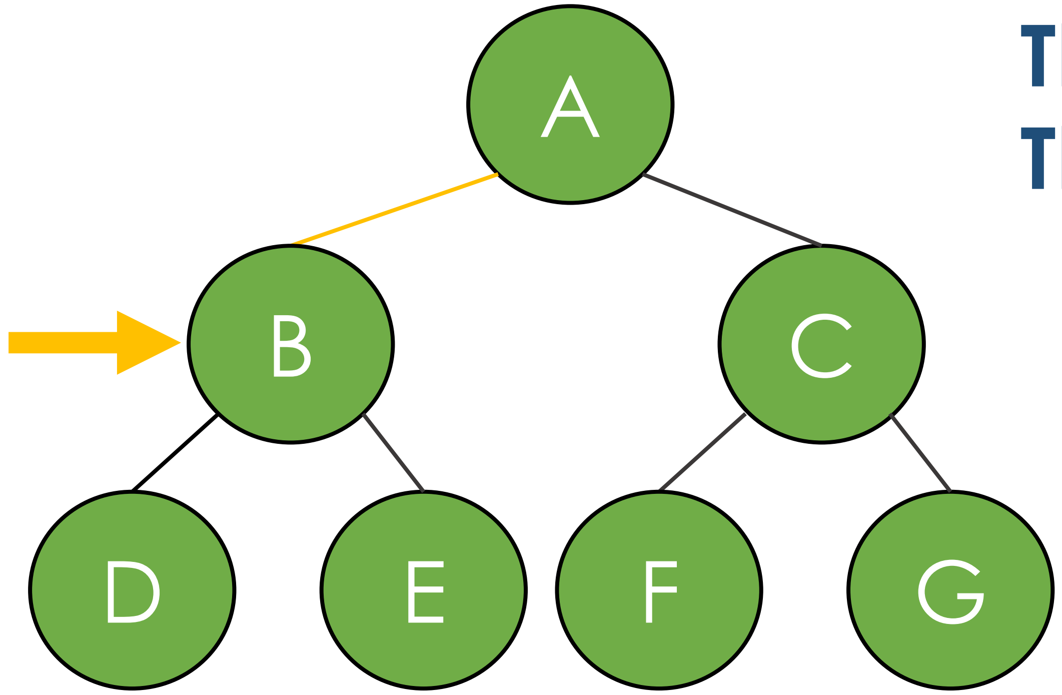
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D

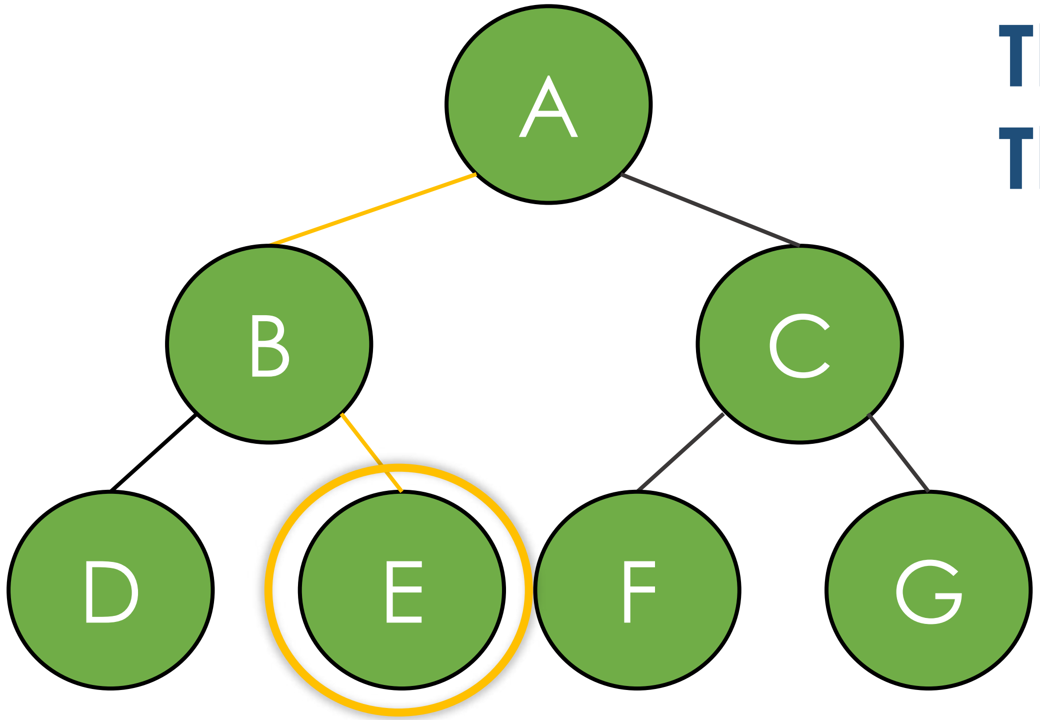
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D

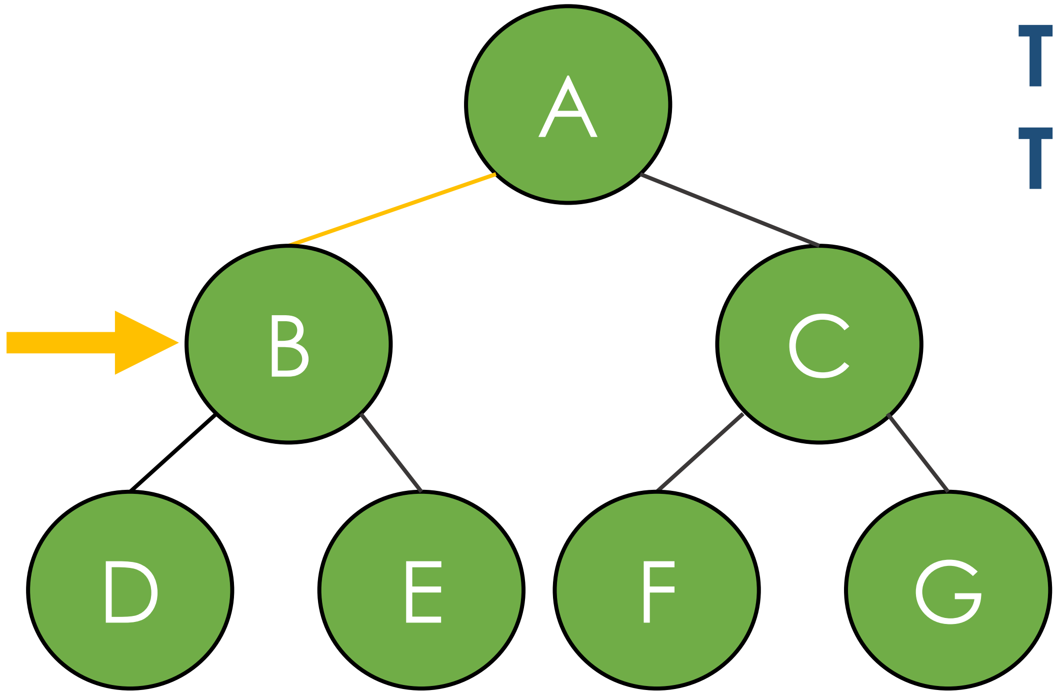
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D E

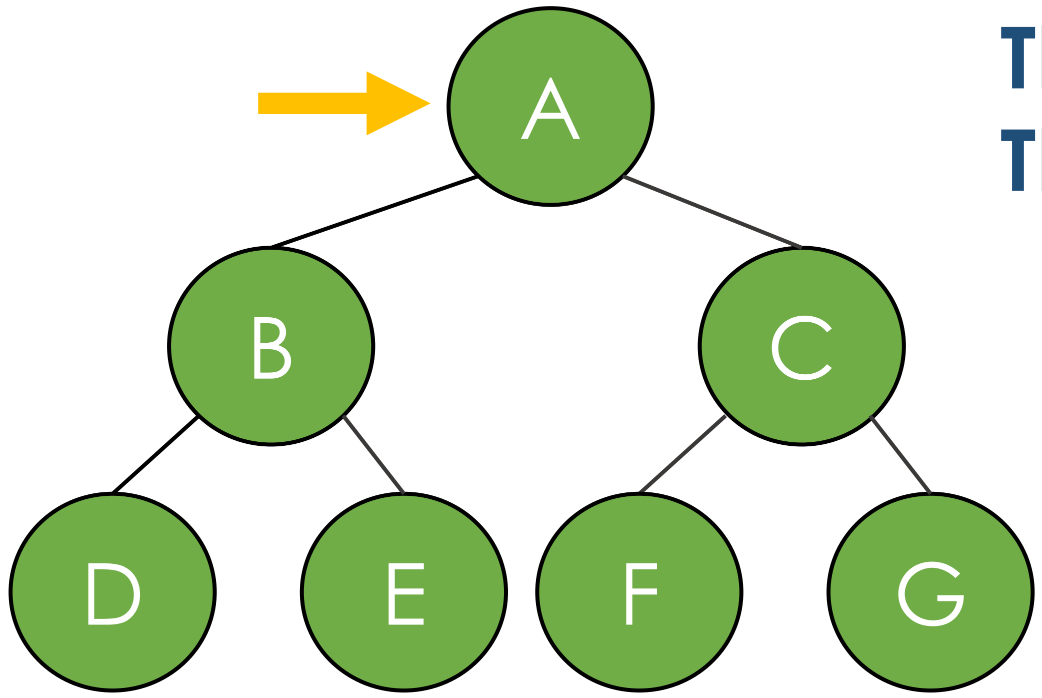
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D E

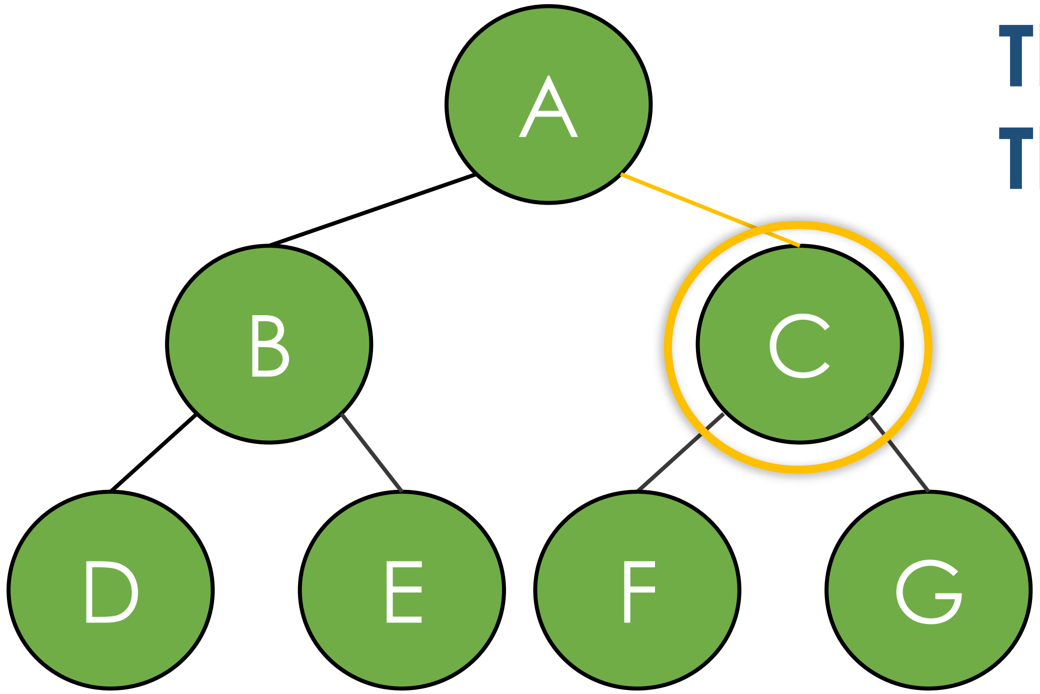
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D E

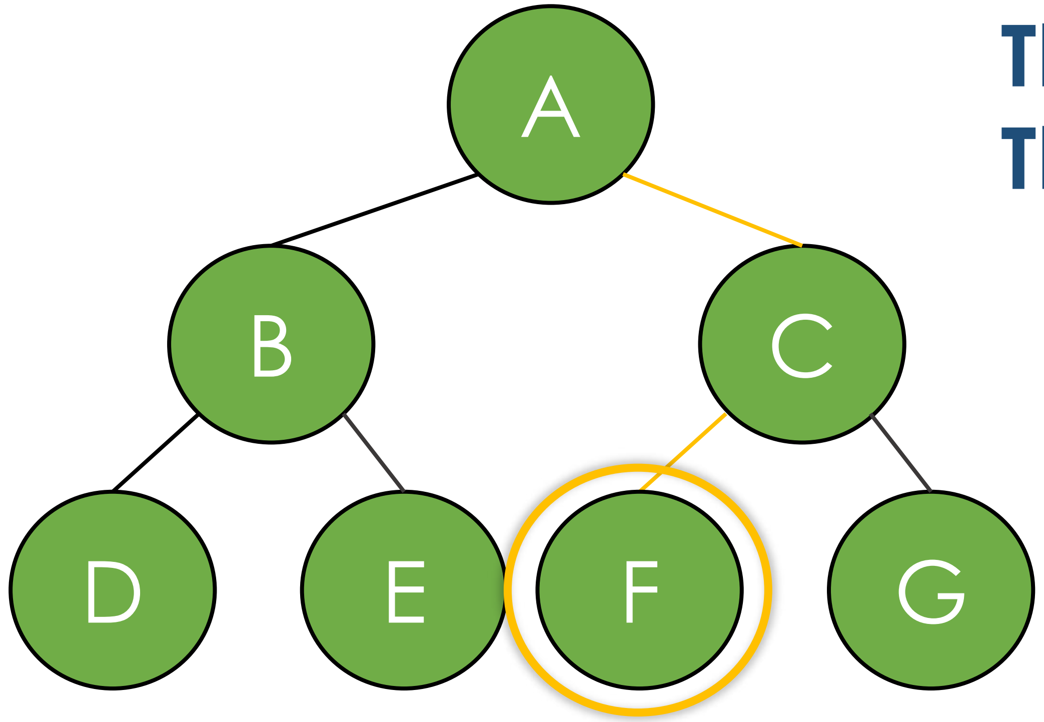
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D E C

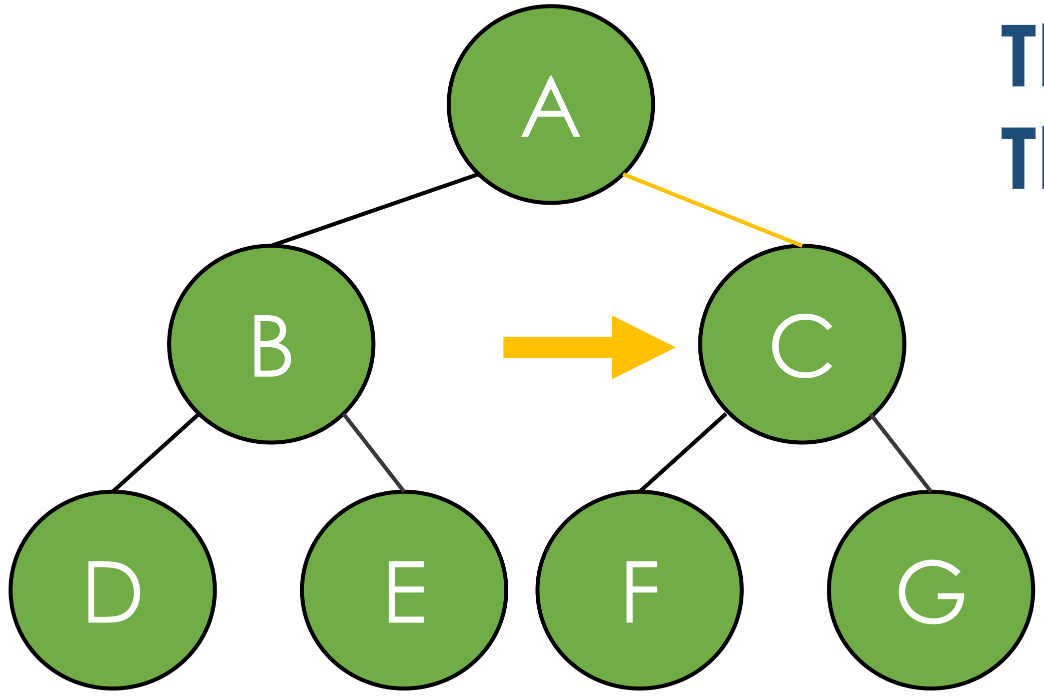
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D E C F

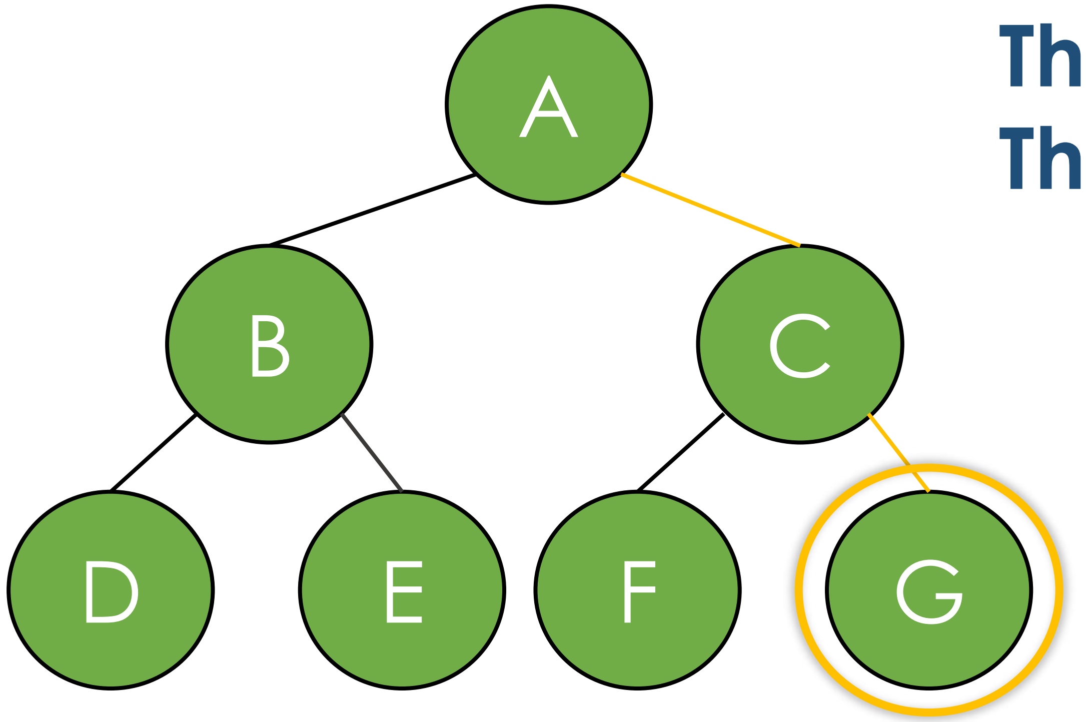
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D E C F

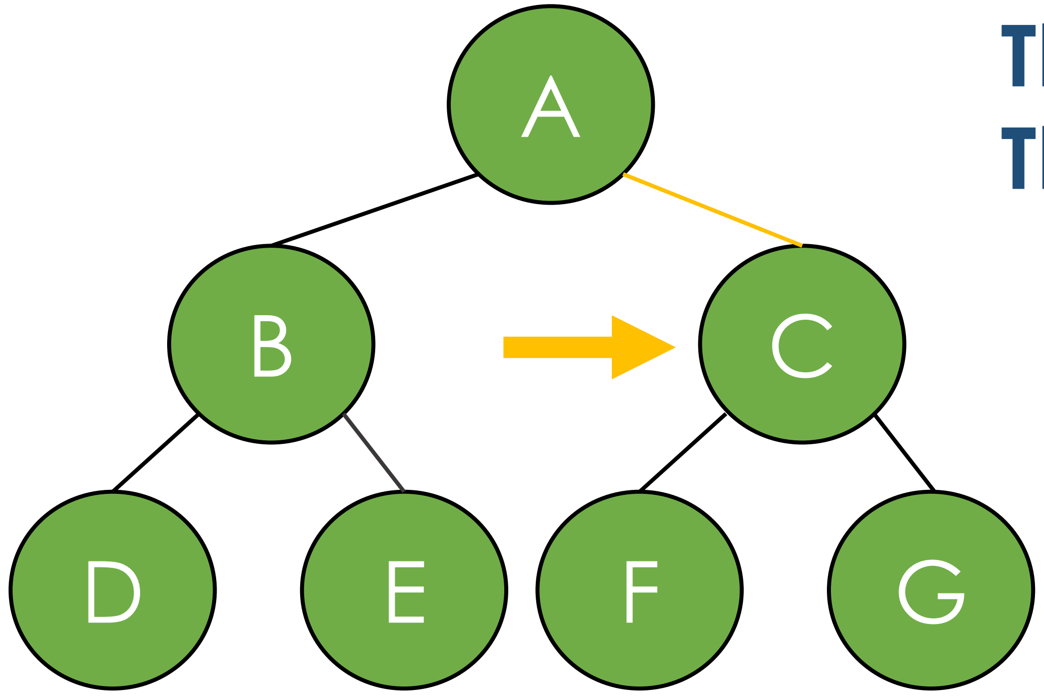
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D E C F G

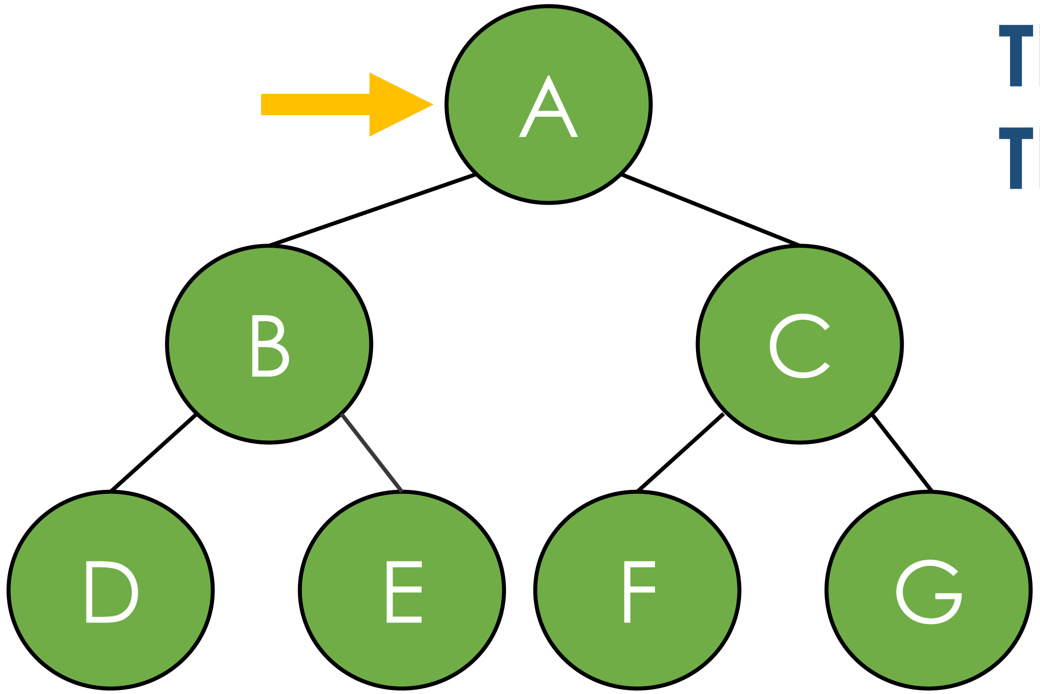
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D E C F G

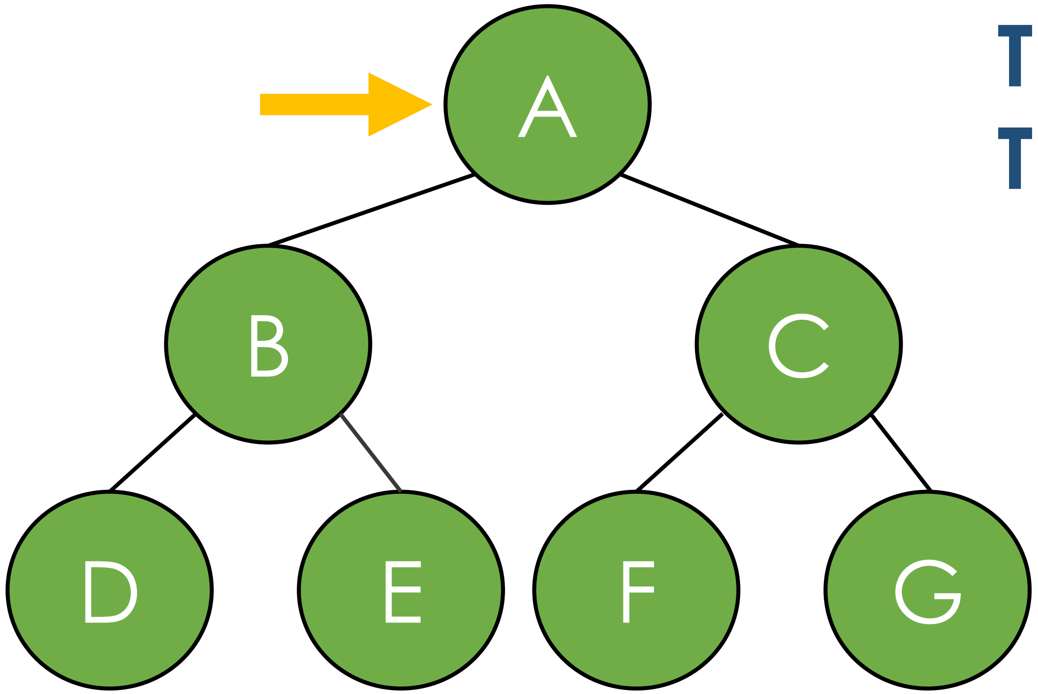
Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D E C F G

Preorder Traversal

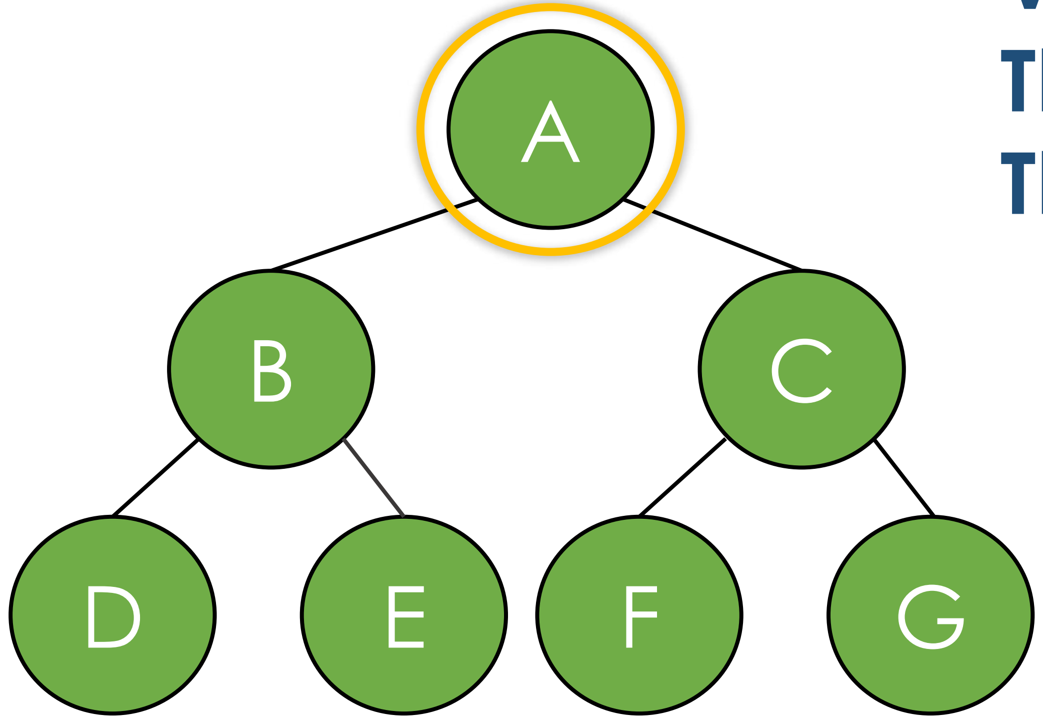


Recursion will help us do this!

Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Visited:
A B D E C F G

Preorder Traversal



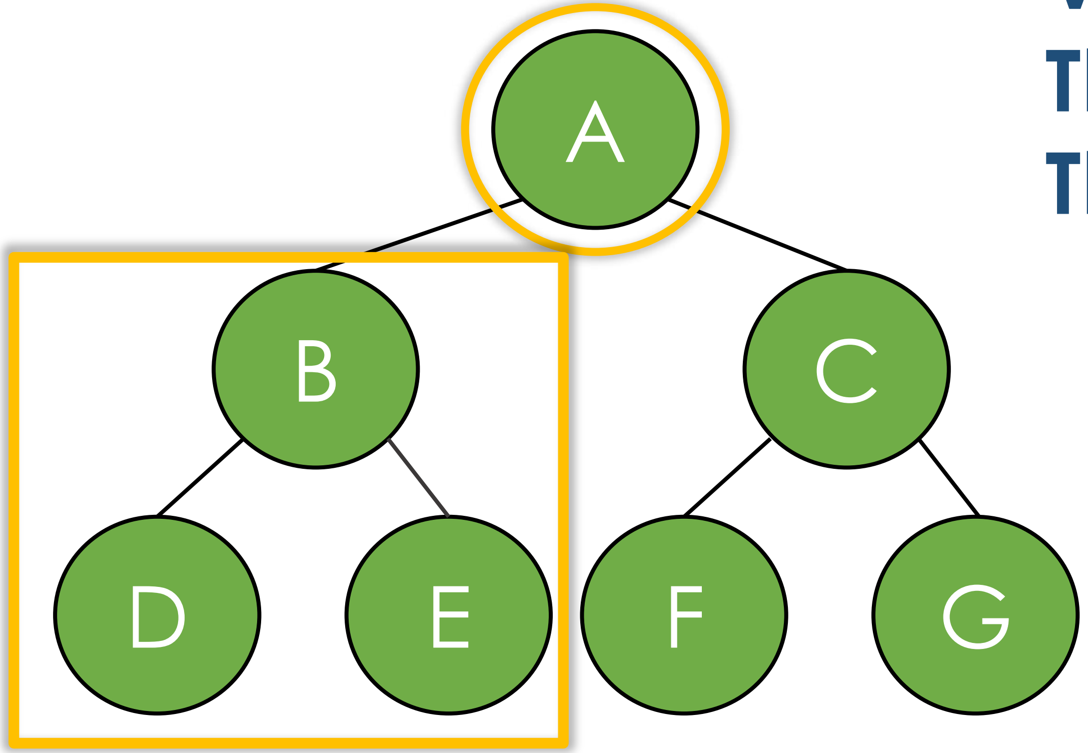
Idea:

Visit yourself

Then visit all your left subtree

Then visit all your right subtree

Preorder Traversal



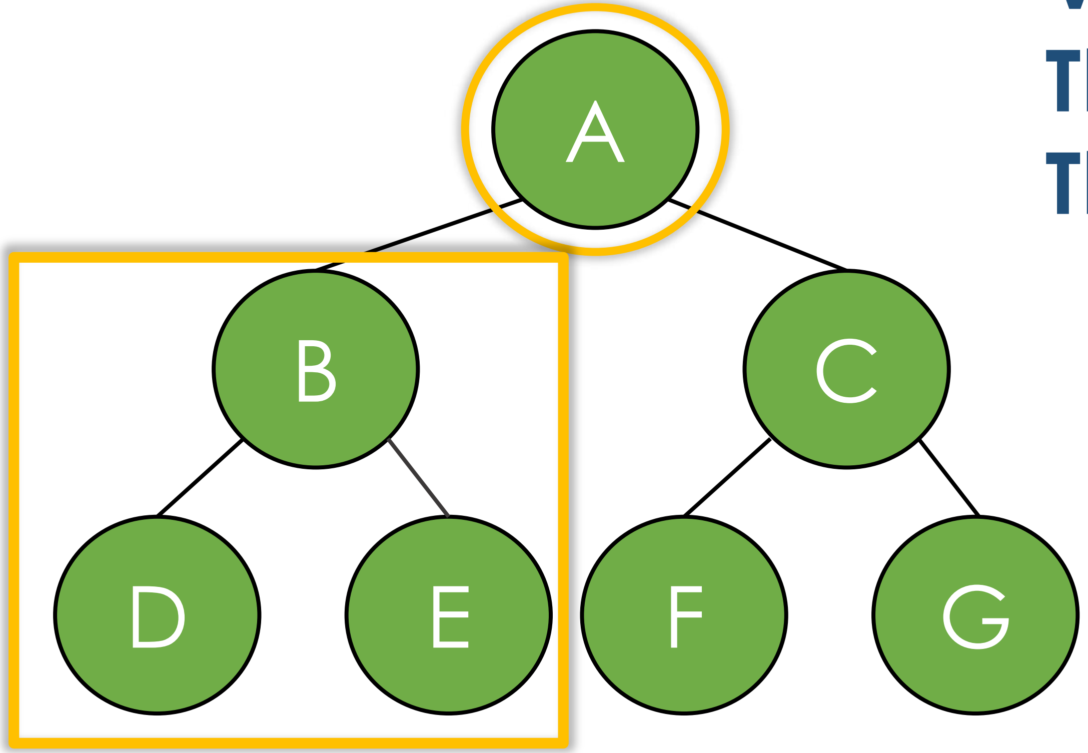
Idea:

Visit yourself

Then visit all your left subtree

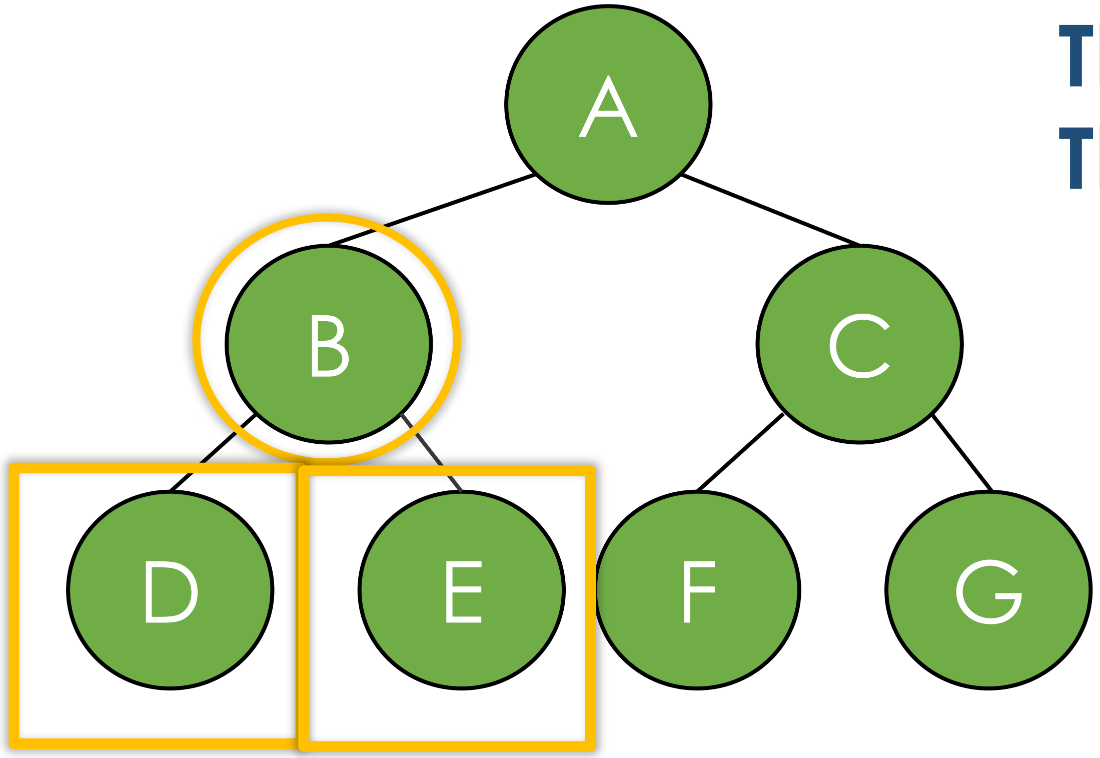
Then visit all your right subtree

Preorder Traversal



Idea:
Visit yourself
Then visit all your left subtree
Then visit all your right subtree

Preorder Traversal



Idea:

Visit yourself

Then visit all your left subtree

Then visit all your right subtree

```java
public class BinaryTree<E> {
  TreeNode<E> root;
  //...
  private void preOrder(TreeNode<E> node) {
    if(node!= null) {
      node.visit();
      preOrder(node.getLeftChild());
      preOrder(node.getRightChild());
    }
  }



}
```

```
public class BinaryTree<E> {
  TreeNode<E> root;
  //...
  private void preOrder(TreeNode<E> node) {
    if(node!= null) {
      node.visit();
      preOrder(node.getLeftChild());
      preOrder(node.getRightChild());
    }
  }




}
```

```
public class BinaryTree<E> {
  TreeNode<E> root;
  //...
  private void preOrder(TreeNode<E> node) {
    if(node!= null) {
      node.visit();
      preOrder(node.getLeftChild());
      preOrder(node.getRightChild());
    }
  }



}
```

```
public class BinaryTree<E> {
   TreeNode<E> root;
   //...
   private void preOrder(TreeNode<E> node) {
      if(node!= null) {
         node.visit();
         preOrder(node.getLeftChild()); ⬅
         preOrder(node.getRightChild());
      }
   }



}
```

```
public class BinaryTree<E> {
   TreeNode<E> root;
   //...
   private void preOrder(TreeNode<E> node) {
      if(node!= null) {
         node.visit();
         preOrder(node.getLeftChild());
         preOrder(node.getRightChild()); ←
      }
   }



}
```

```
public class BinaryTree<E> {
  TreeNode<E> root;
  //...
  private void preOrder(TreeNode<E> node) {
    if(node!= null) {
      node.visit();
      preOrder(node.getLeftChild());
      preOrder(node.getRightChild());
    }
  }
  public void preOrder() {
    this.preOrder(root);
  }
}
```