

Assignment Project Exam Help

Add WeChat powcoder

# COMP 250

## INTRODUCTION TO COMPUTER SCIENCE

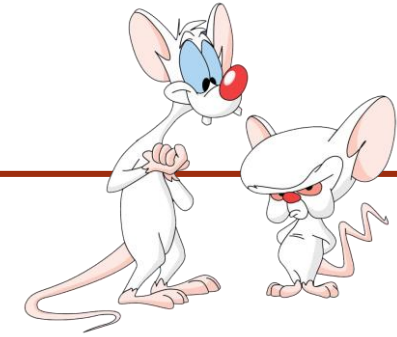
Week 12-1: Binary Search Trees

Giulia Alberini, Fall 2020

Slides adapted from Michael Langer's

# Assignment Project Exam Help

WHAT ARE WE GOING TO DO IN THIS VIDEO?



- Binary Search Trees

<https://powcoder.com>

Add WeChat powcoder

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help  
BINARY SEARCH  
<https://powcoder.com>  
TREES

Add WeChat powcoder

# Assignment Project Exam Help

## ~~BSTNode Add WeChat powcoder~~

- The keys are “comparable”  $<, =, >$   
e.g. numbers, strings.

Assignment Project Exam Help

<https://powcoder.com>

```
class BSTNode<K>{  
    K key;  
    BSTNode<K> leftchild;  
    BSTNode<K> rightchild;  
    :  
}
```

# Assignment Project Exam Help

## BINARY SEARCH TREE DEFINITION

- binary tree

Assignment Project Exam Help

- keys are comparable, and unique (no duplicates)

<https://powcoder.com>

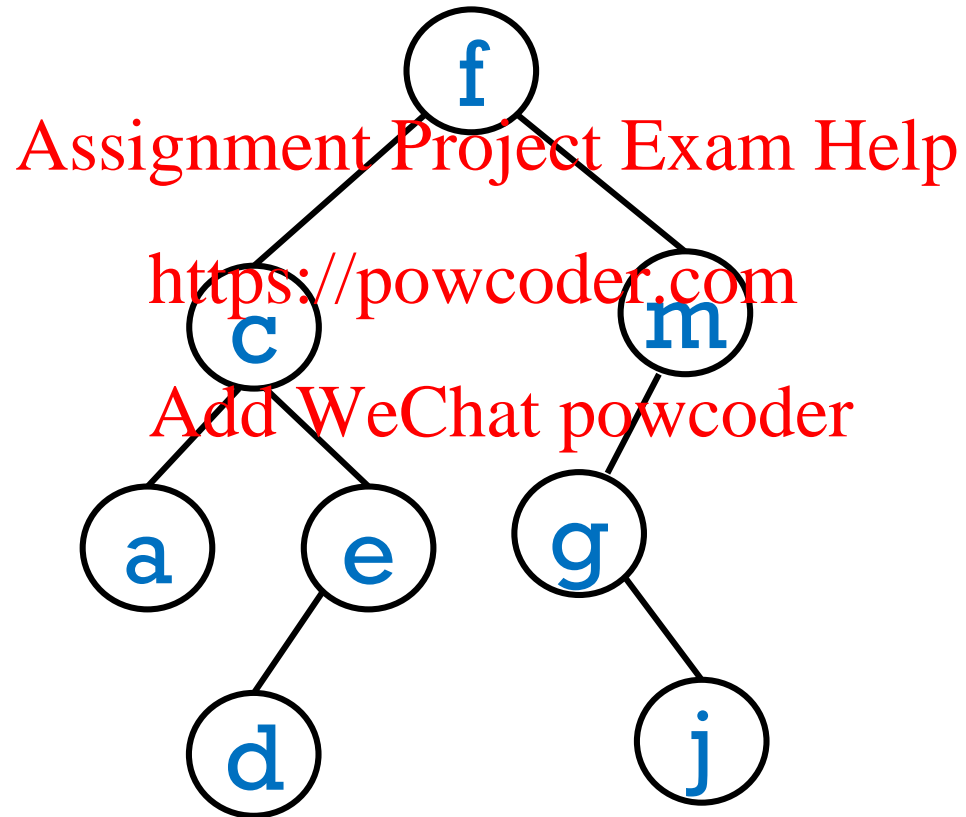
Add WeChat powcoder

- for each node, all descendents in left subtree are less than the node, and all descendents in the node's right subtree are greater than the node  
(comparison is based on node key)

# Assignment Project Exam Help

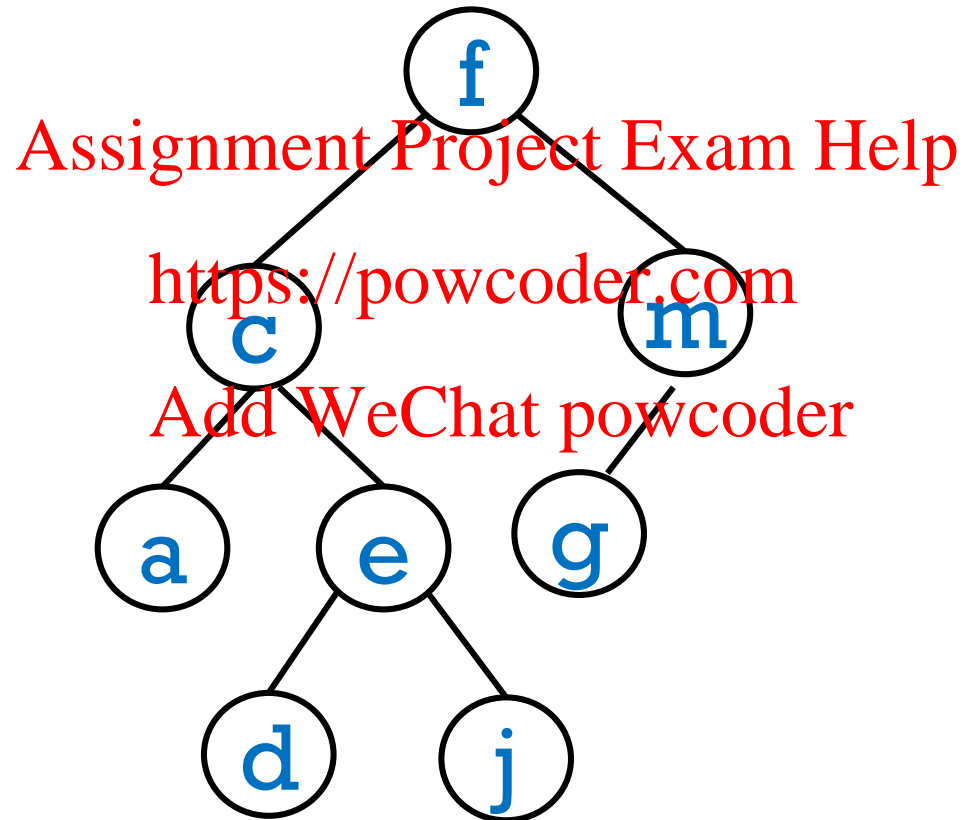
EXAMPLE Add WeChat powcoder

---



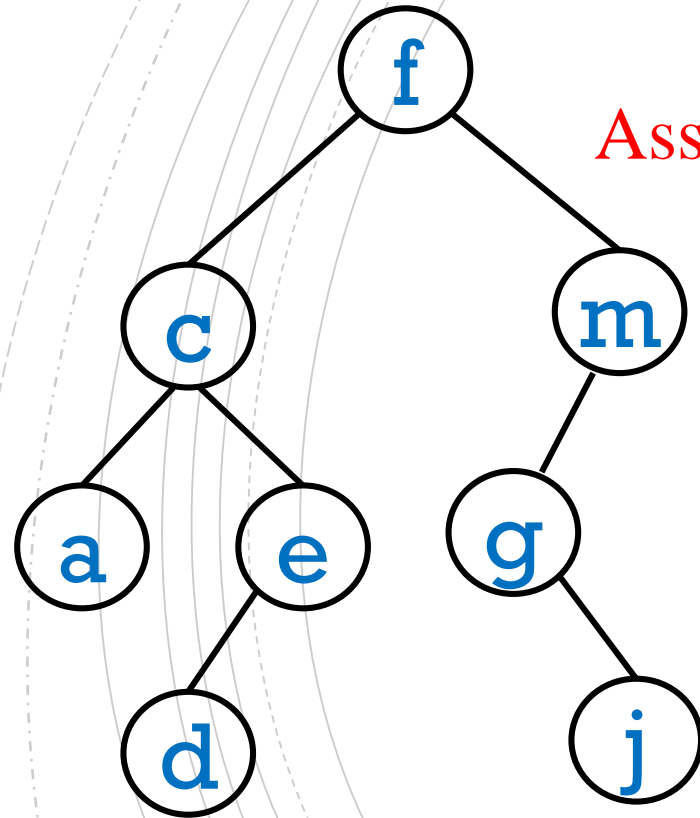
# Assignment Project Exam Help

THIS IS NOT A BST. WHY NOT?



# Assignment Project Exam Help

BST - TRAVERSALS



An in-order traversal on a BST visits the nodes in the natural order defined by the key.

https://powcoder.com

Add WeChat powcoder

a c d e f g j m



# Assignment Project Exam Help

## BINARY SEARCH TREE ADT

- find( key )
- findMin()
- findMax()
- add(key)
- remove(key)

Assignment Project Exam Help

<https://powcoder.com>

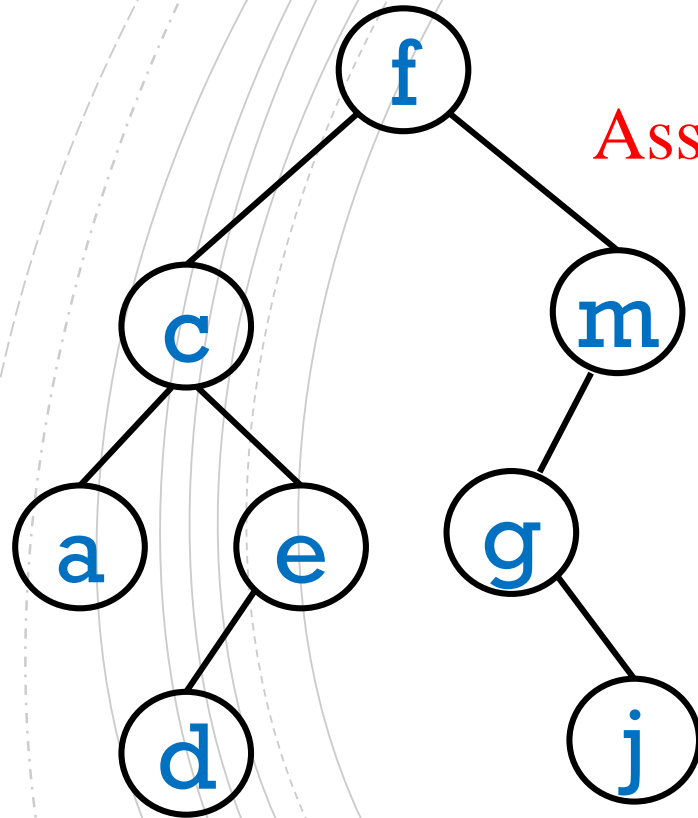
Add WeChat powcoder

We can define the operations of a BST without knowing how they are implemented. (ADT)

Let's next look at some recursive algorithms for implementing them.

# Assignment Project Exam Help

FIND() Add WeChat powcoder



## Assignment Project Exam Help

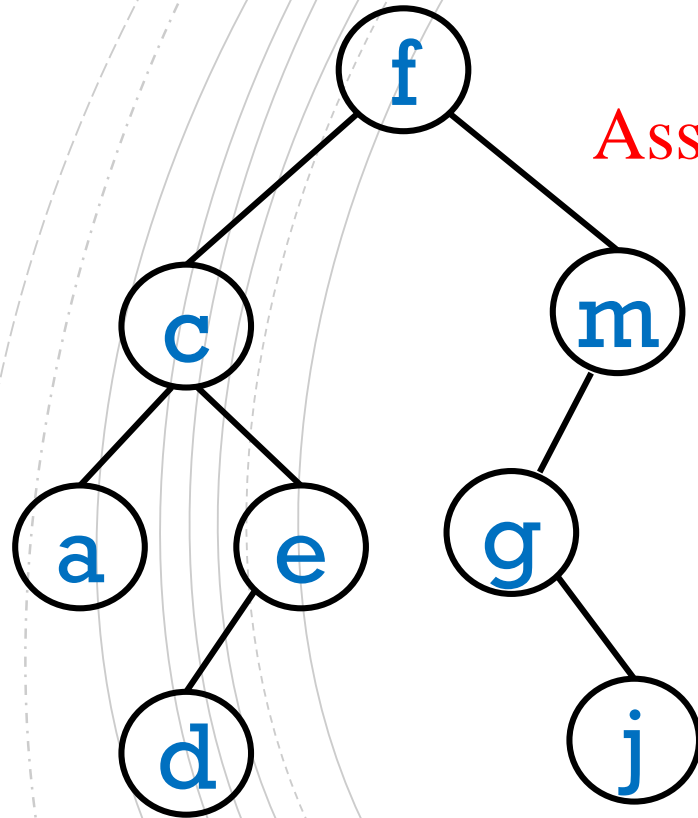
Desired behaviour:

<https://powcoder.com>

- `find(root, g)` returns the `g` node
- `find(root, s)` returns null.

# Assignment Project Exam Help

FIND() — IMPLEMENTATION



```
find(root, key) { // returns a node
    if (root == null)
        return null
    else if (root.key == key)
        return root
}
```

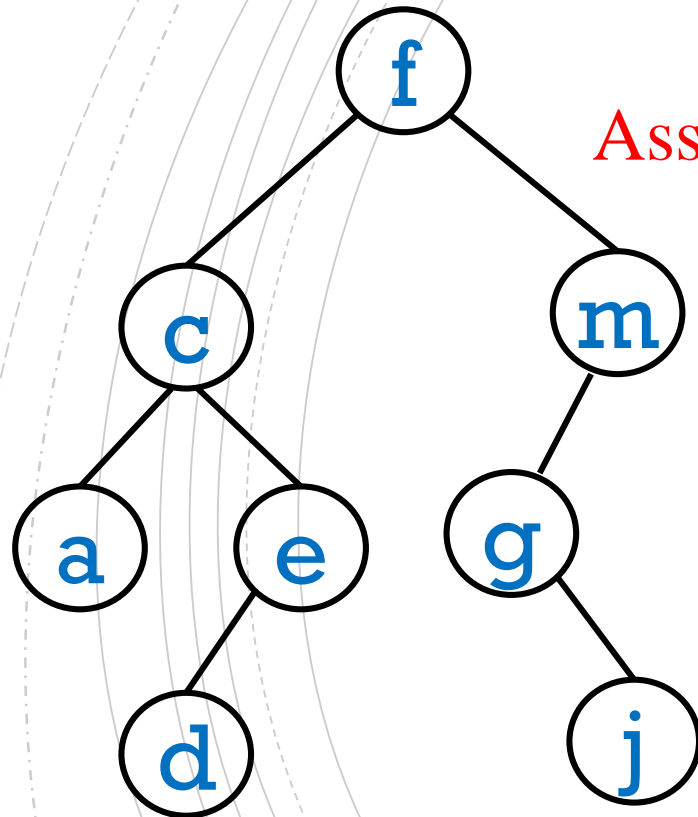
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

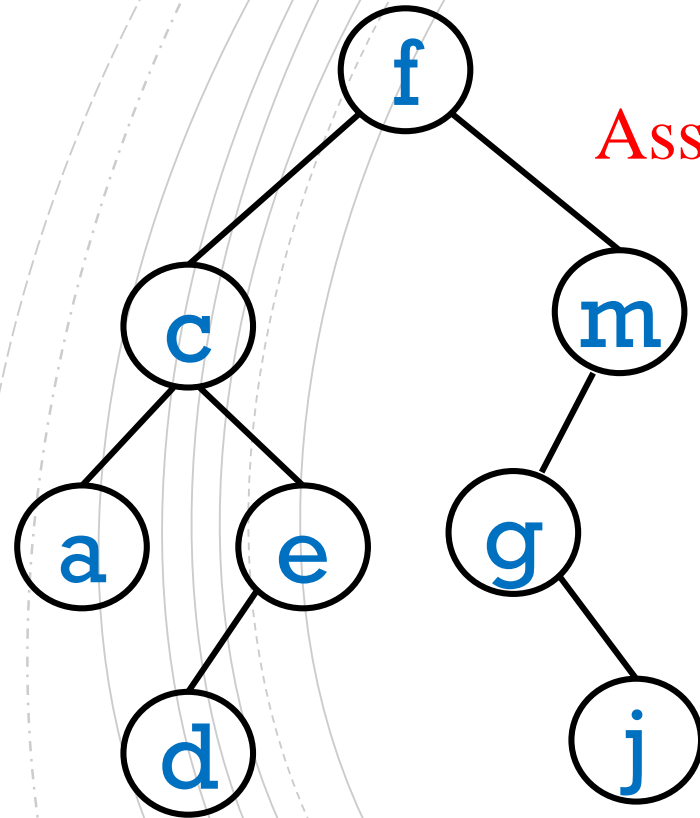
## FIND() — IMPLEMENTATION



```
find(root, key) { // returns a node
    if (root == null)
        return null
    else if (root.key == key)
        return root
    else if (key < root.key)
        return find(root.left, key)
    else
        return find(root.right, key)
}
```

# Assignment Project Exam Help

~~FINDMIN Add WeChat powcoder~~



Assignment Project Exam Help

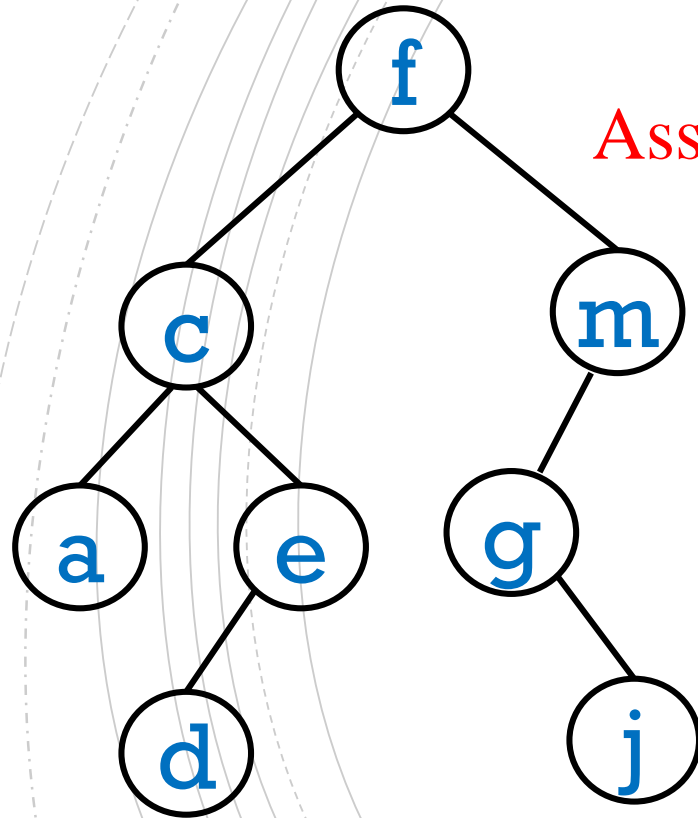
<https://powcoder.com>  
findMin() should return the node with the smallest key. So, for example given

Add WeChat powcoder  
the BST on the left,

- `findMin(root)` returns ... ?

# Assignment Project Exam Help

~~FINDMIN Add WeChat powcoder~~



## Assignment Project Exam Help

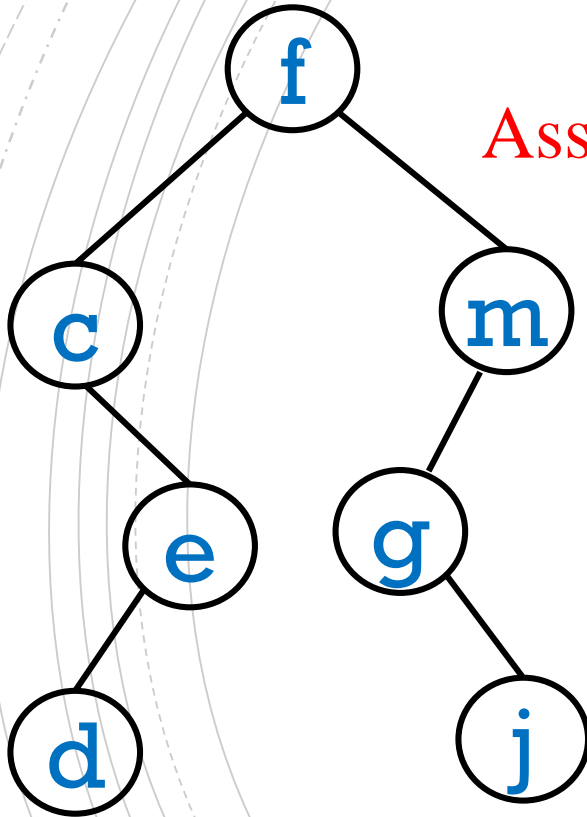
<https://powcoder.com>  
findMin() should return the node with the smallest key. So, for example given

~~Add WeChat powcoder~~  
the BST on the left,

- `findMin(root)` returns the **a** node

# Assignment Project Exam Help

~~FINDMIN~~ Add WeChat powcoder



Assignment Project Exam Help

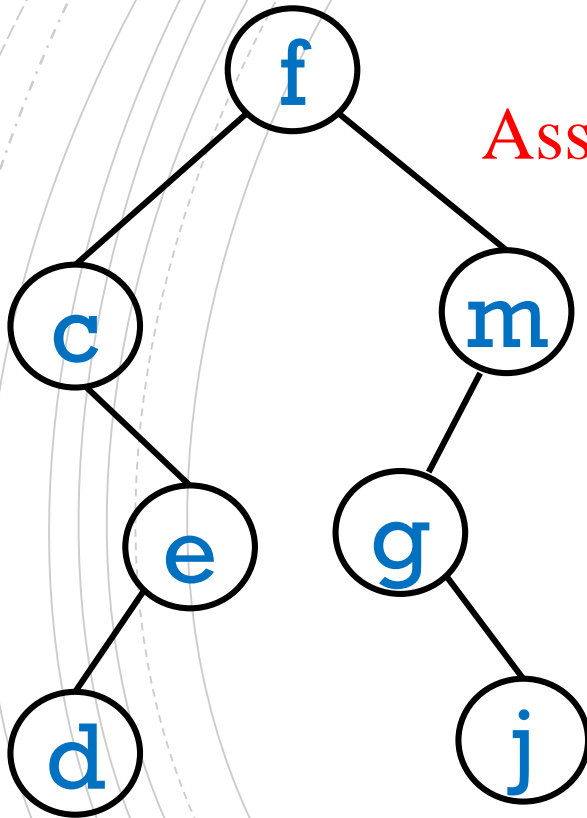
<https://powcoder.com>  
findMin() should return the node with the smallest key. So, for example given

Add WeChat powcoder  
the BST on the left,

- `findMin(root)` returns ... ?

# Assignment Project Exam Help

~~FINDMIN Add WeChat powcoder~~



## Assignment Project Exam Help

<https://powcoder.com>  
findMin() should return the node with the smallest key. So, for example given

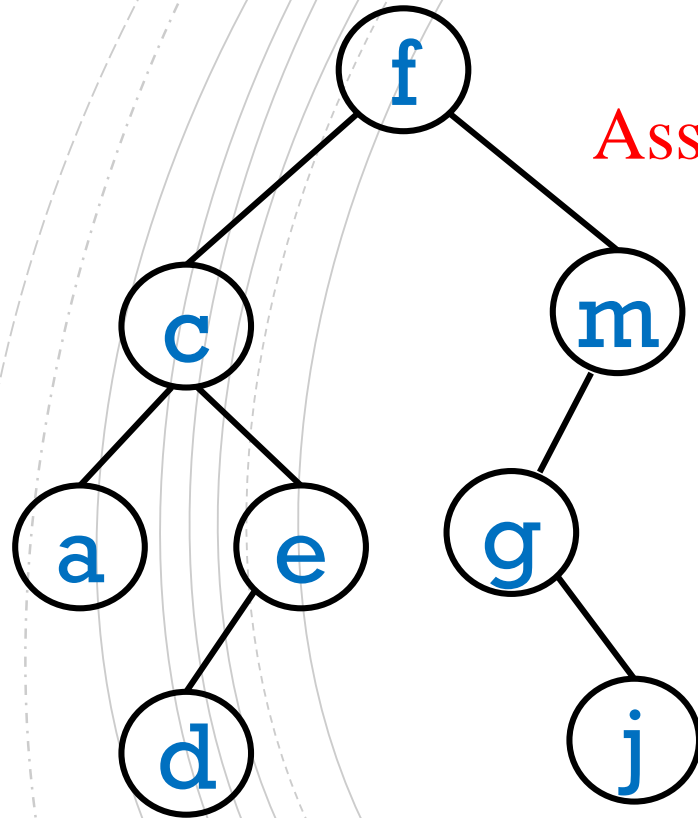
~~Add WeChat powcoder~~  
the BST on the left,

- `findMin(root)` returns the **c** node



# Assignment Project Exam Help

FINDMIN And Implementation



Assignment Project Exam Help

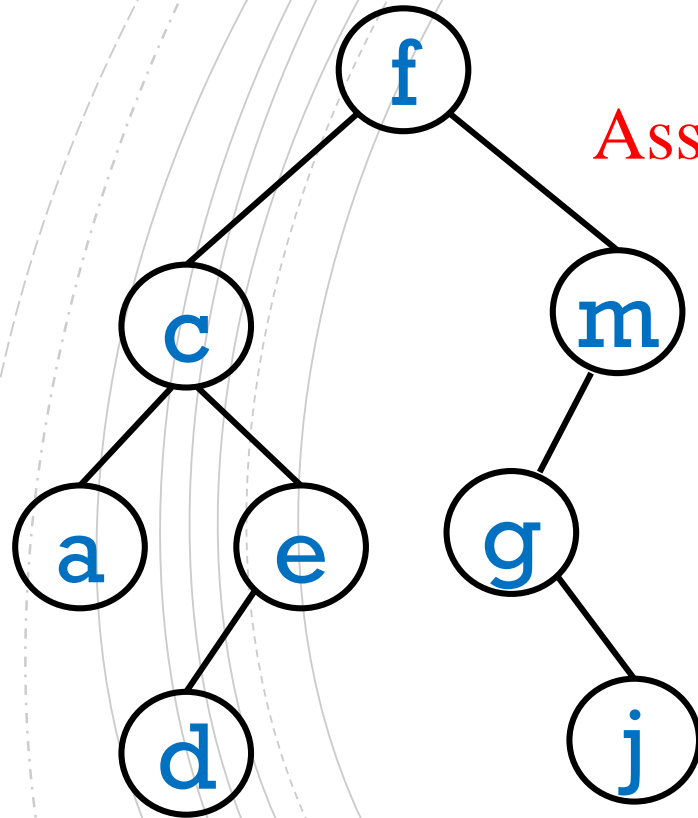
<https://powcoder.com>

Add WeChat powcoder

```
findMin(root){ // returns a node
    if (root == null)
        return null
}
```

# Assignment Project Exam Help

## FINDMIN AND IMPLEMENTATION



Assignment Project Exam Help

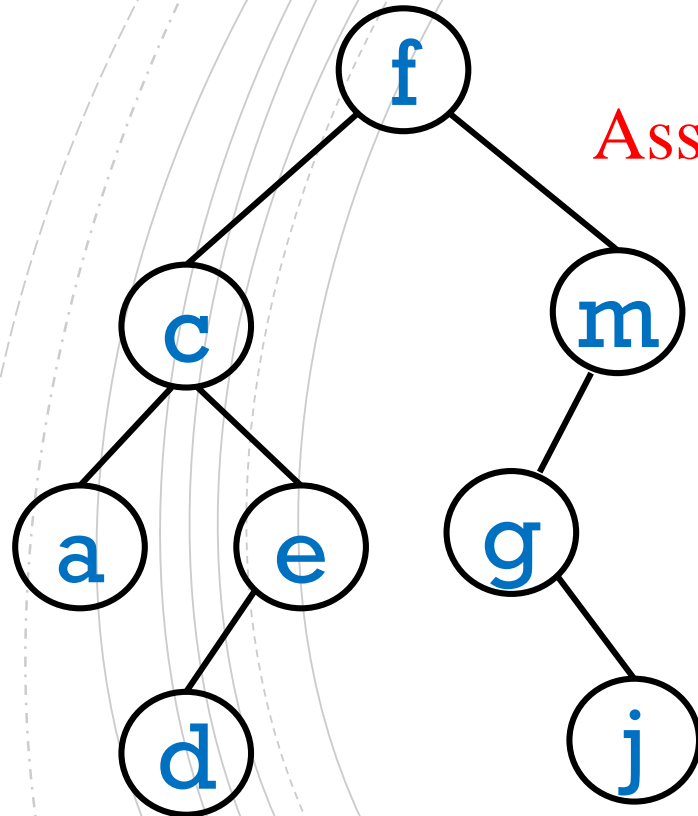
<https://powcoder.com>

Add WeChat powcoder

```
findMin(root){ // returns a node
    if (root == null)
        return null
    else if (root.left == null)
        return root
    else
        return findMin( root.left )
}
```

# Assignment Project Exam Help

~~FINDMAX()~~ Add WeChat powcoder



## Assignment Project Exam Help

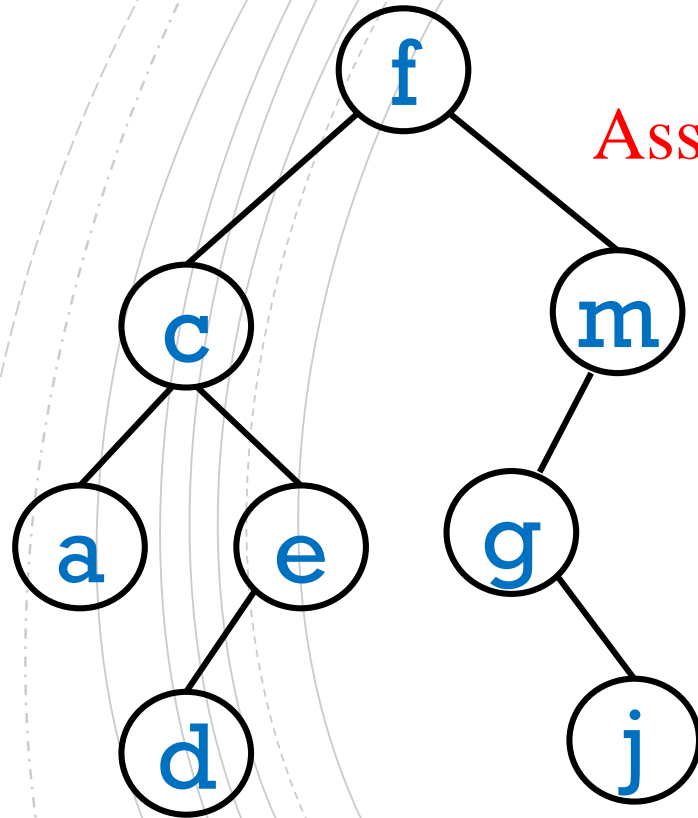
<https://powcoder.com>  
findMax() should return the node with the greatest key. So, for example given

Add WeChat powcoder  
the BST on the left,

- `findMax(root)` returns ... ?

# Assignment Project Exam Help

~~FINDMAX()~~ Add WeChat powcoder



## Assignment Project Exam Help

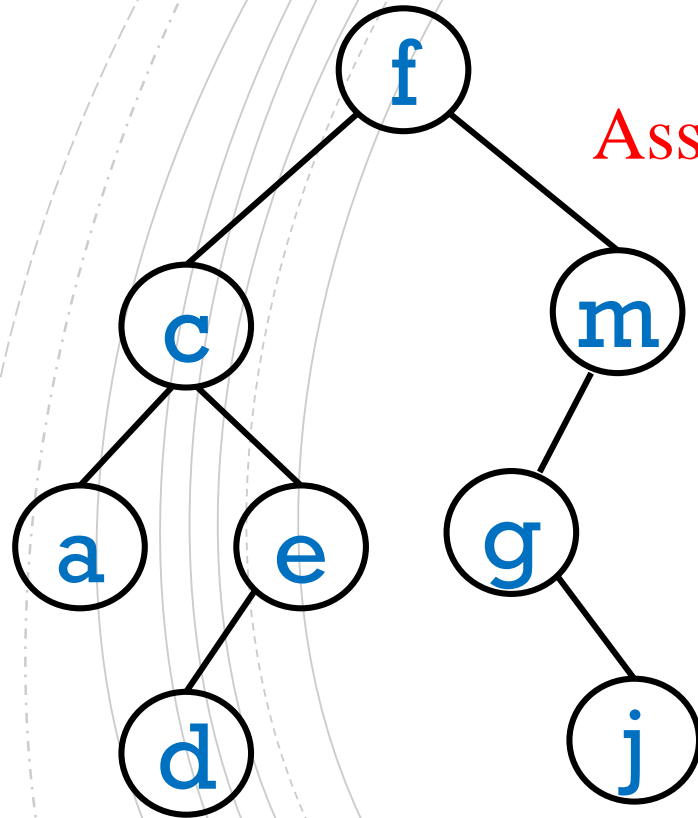
<https://powcoder.com>  
findMax() should return the node with the greatest key. So, for example given

Add WeChat powcoder  
the BST on the left,

- `findMax(root)` returns the **m** node.

# Assignment Project Exam Help

## FINDMAX() IMPLEMENTATION



Assignment Project Exam Help

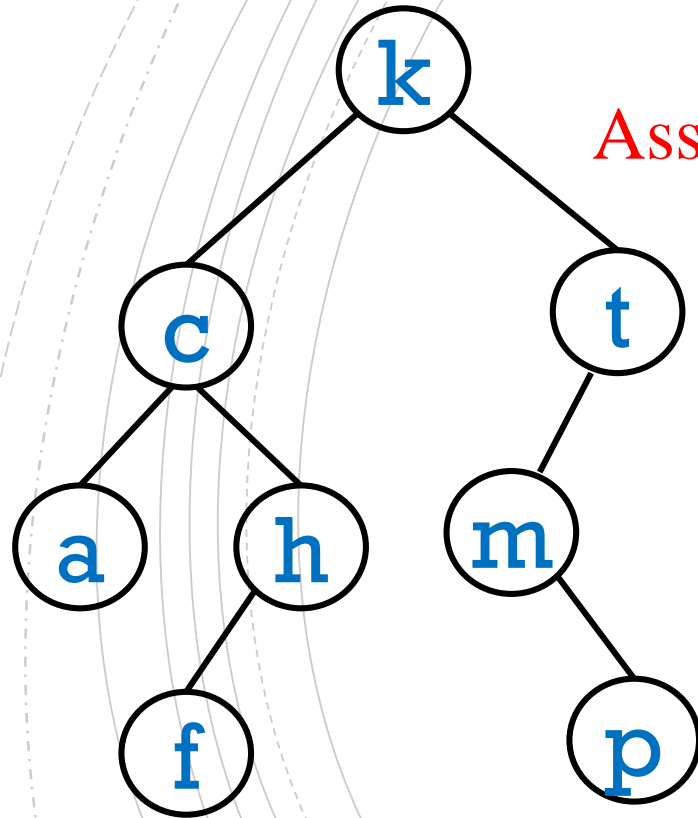
<https://powcoder.com>

Add WeChat powcoder

```
findMax(root) { // returns a node
    if (root == null)
        return null
    else if (root.right == null)
        return root
    else
        return findMax (root.right)
}
```

# Assignment Project Exam Help

ADD() Add WeChat powcoder



Assignment Project Exam Help

add(key) should add a BSTNode to the tree.

<https://powcoder.com>

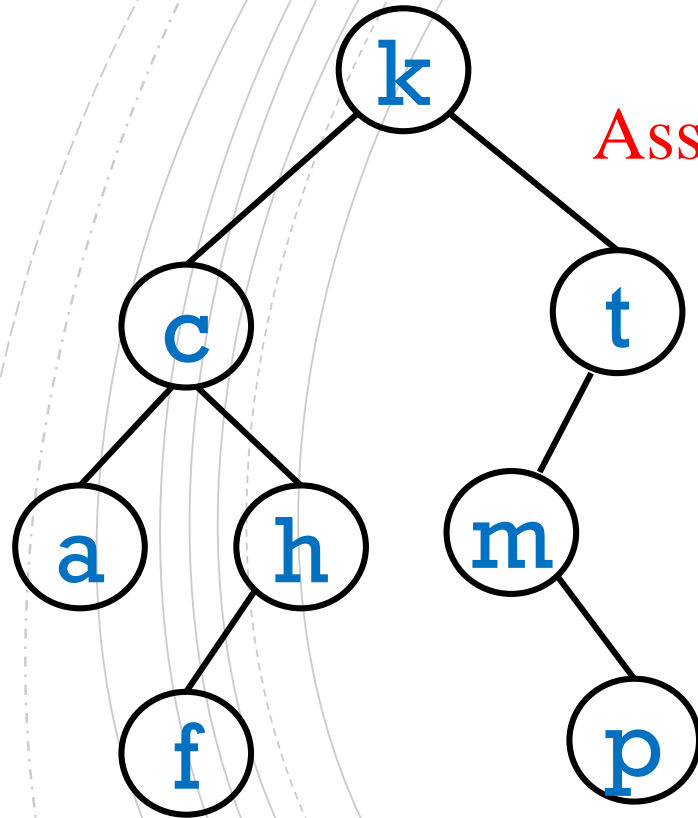
- Where?

Add WeChat powcoder

- How?

# Assignment Project Exam Help

ADD() Add WeChat powcoder



Assignment Project Exam Help

add(key) should add a BSTNode to the tree.

<https://powcoder.com>

- add(j) ?

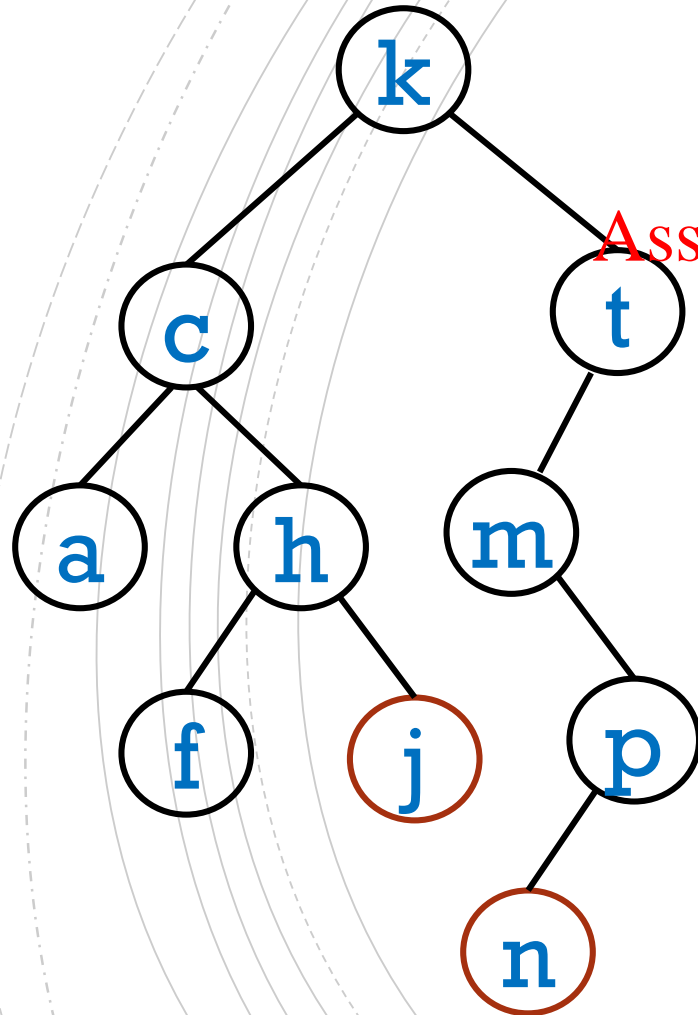
Add WeChat powcoder

- add(w) ?

A new node is always a leaf.

# Assignment Project Exam Help

ADD() Add WeChat powcoder



Assignment Project Exam Help

add(key) should add a BSTNode to the  
<https://powcoder.com>  
tree.

Add WeChat powcoder

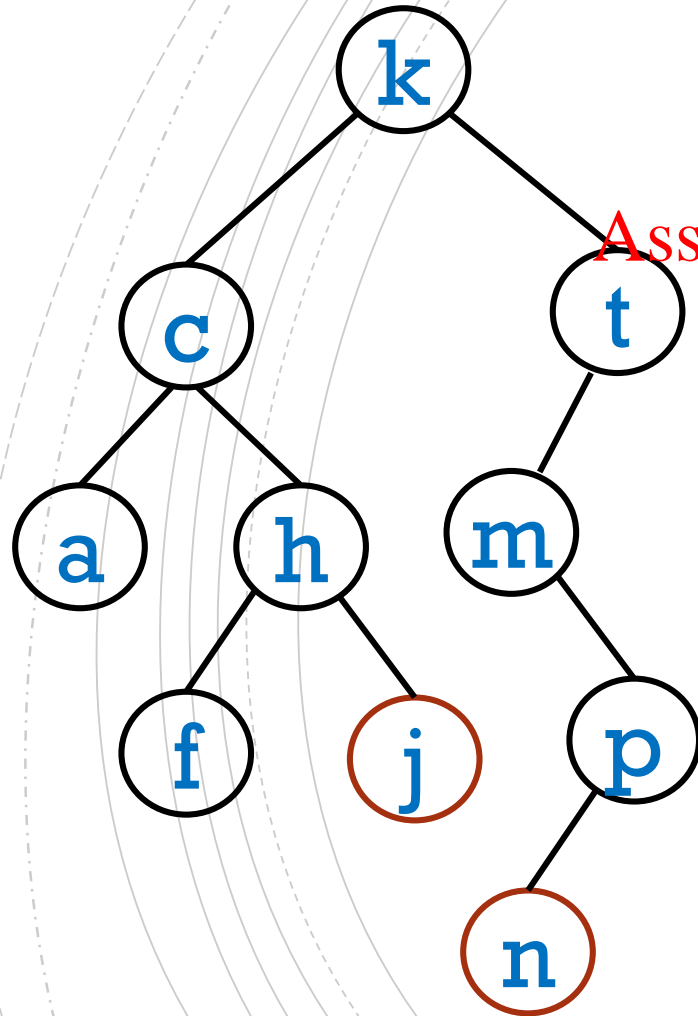
- add(j) ?
- add(n) ?

A new node is always a leaf.



# Assignment Project Exam Help

ADD() - IMPLEMENTATION



`add(root, key) { // returns root node`

<https://powcoder.com>

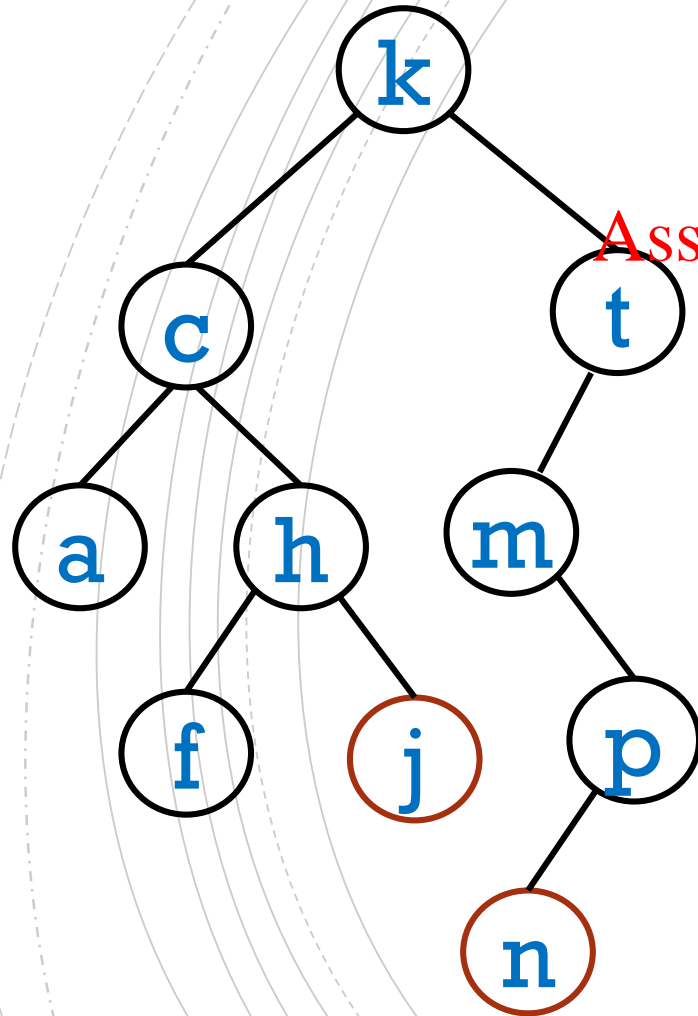
Add WeChat powcoder

`return root`

`}`

# Assignment Project Exam Help

## ADD() - IMPLEMENTATION



Assignment Project Exam Help  
`add(root, key) { // returns root node`  
`if (root == null)`

<https://powcoder.com>  
`root = new BSTnode(key)`

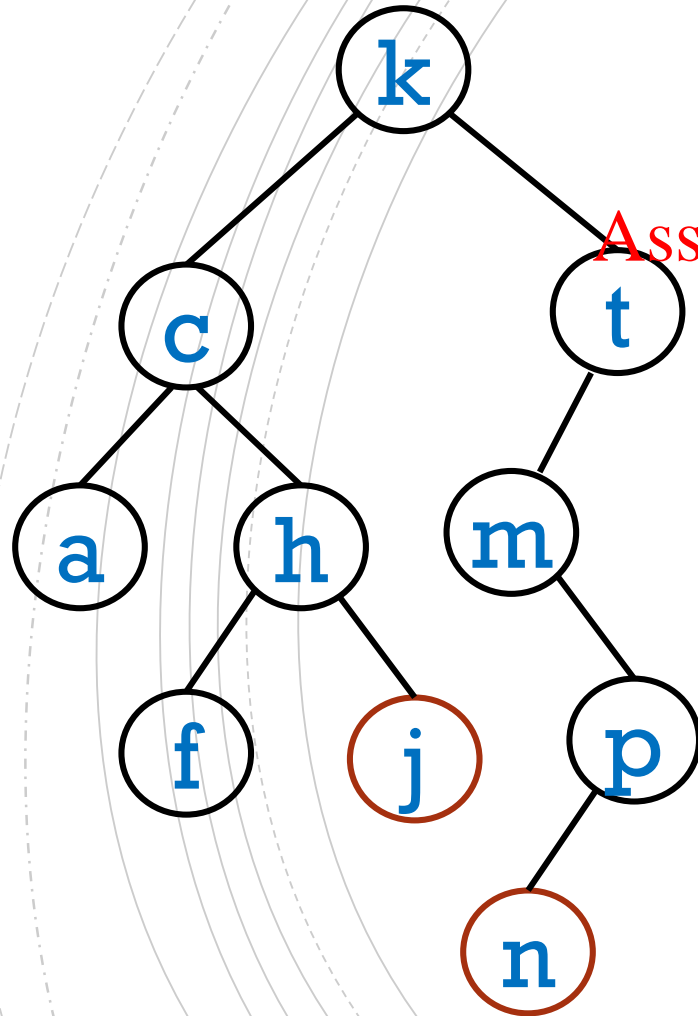
Add WeChat powcoder

`return root`

`}`

# Assignment Project Exam Help

## ADD() - IMPLEMENTATION

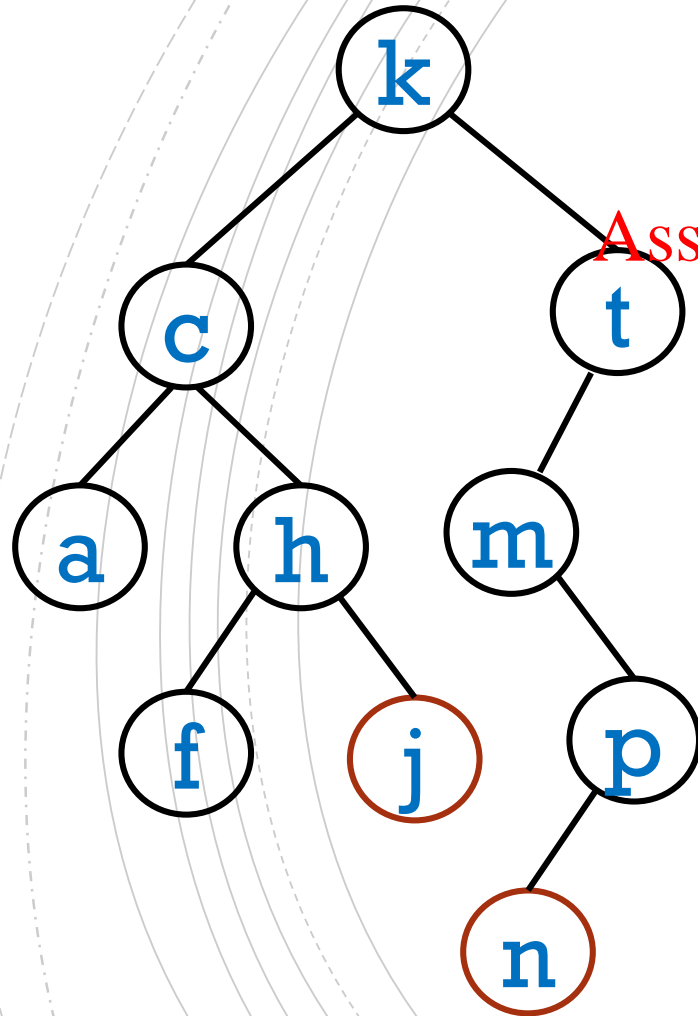


```
add(root, key) { // returns root node
    if (root == null)
        root = new BSTnode(key)
    else if (key < root.key) {
        root.left = add(root.left, key)
    }

    return root
}
```

# Assignment Project Exam Help

## ADD() - IMPLEMENTATION



```
add(root, key) { // returns root node
    if (root == null)
        root = new BSTnode(key)
    else if (key < root.key) {
        root.left = add(root.left, key)
    }
    else if (key > root.key) {
        root.right = add(root.right, key)
    }
    return root
}
```

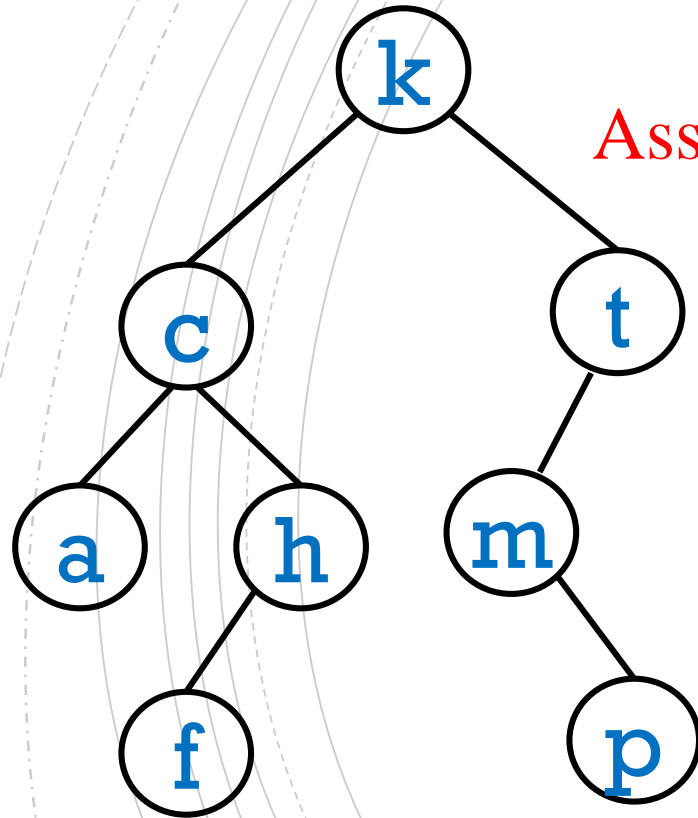
Q: What happens if `root.key == key`?

A: Nothing!

# Assignment Project Exam Help

~~REMOVE~~ Add WeChat powcoder

remove(**c**) →



Assignment Project Exam Help

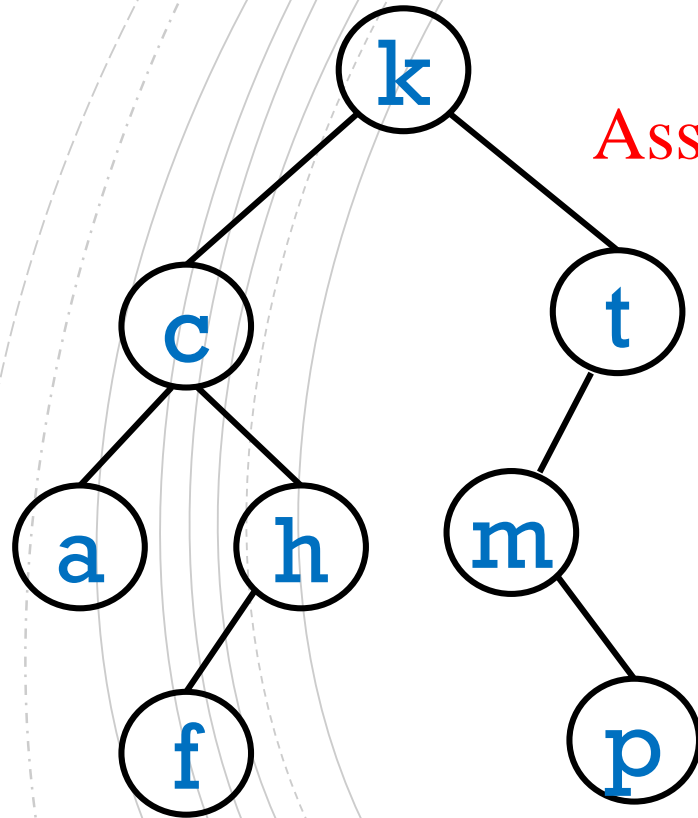
<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

~~REMOVE~~ Add WeChat powcoder

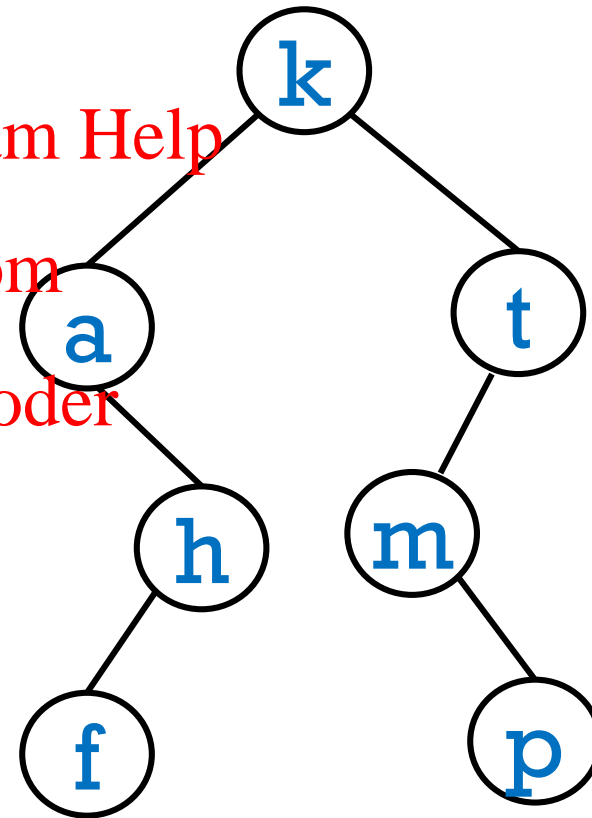
remove (c) → this is one way to do it



Assignment Project Exam Help

<https://powcoder.com>

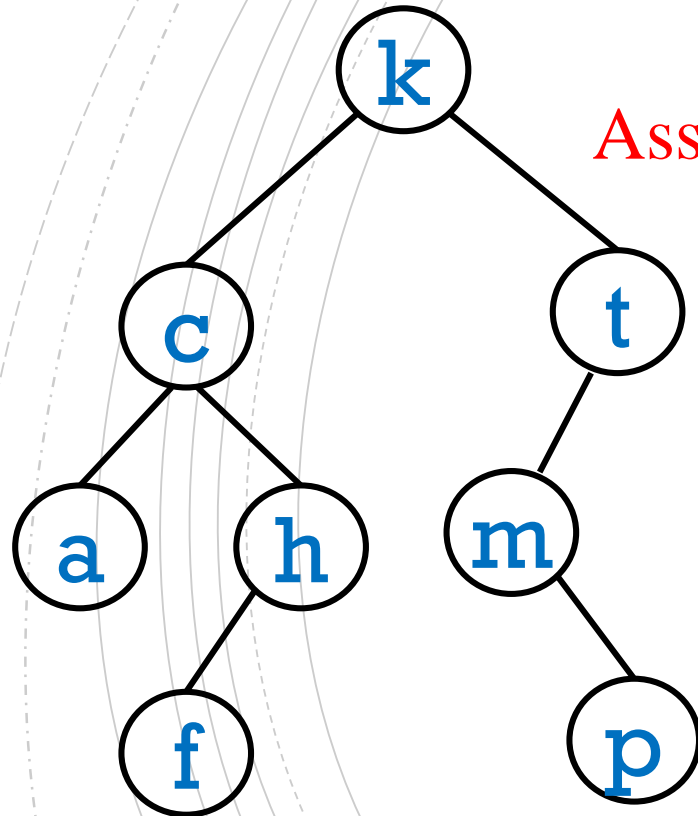
Add WeChat powcoder



# Assignment Project Exam Help

~~REMOVE Add WeChat powcoder~~

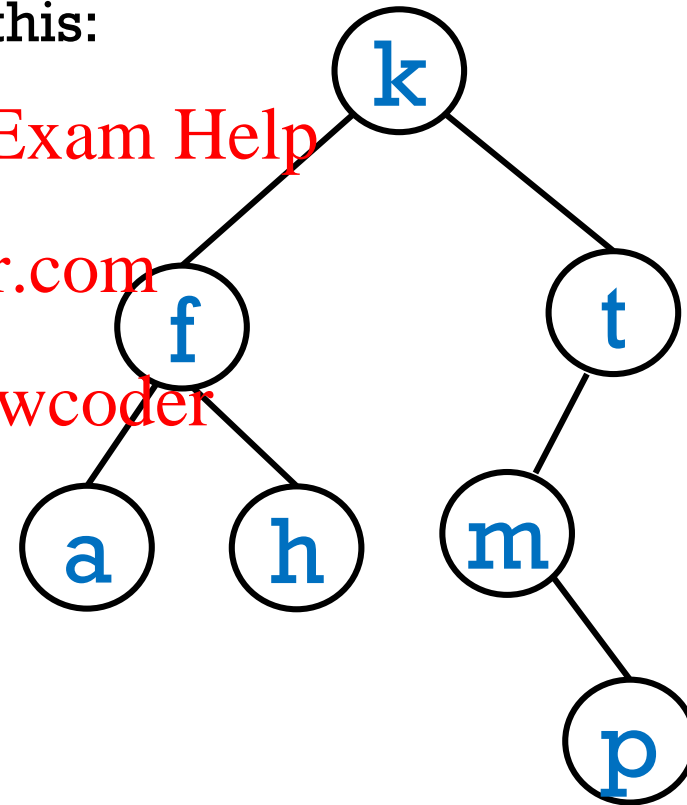
remove(**c**) → the following algorithm  
does this:



Assignment Project Exam Help

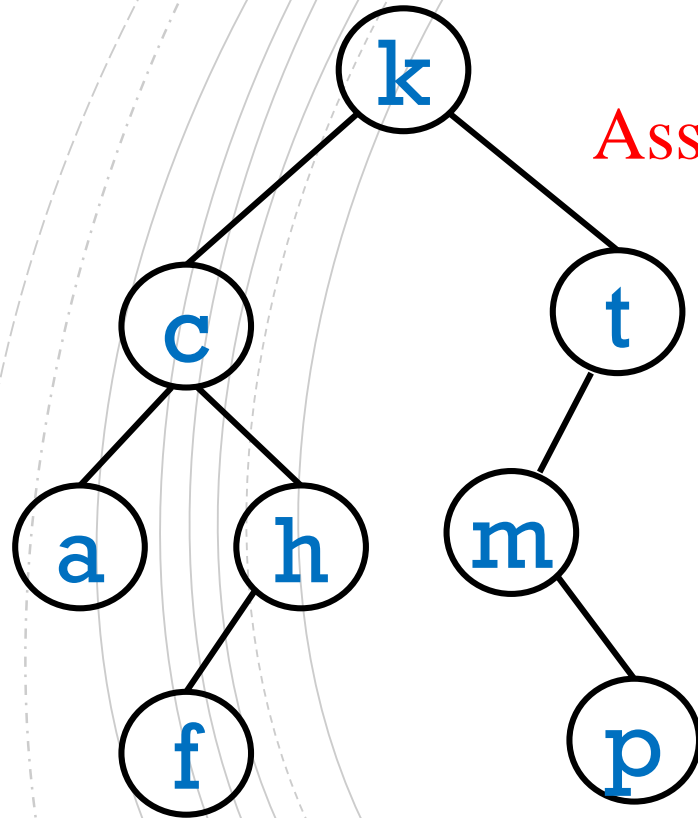
<https://powcoder.com>

Add WeChat powcoder



# Assignment Project Exam Help

REMOVE Add WeChat powcoder



```
remove (root, key) { // returns root node
    if( root == null )
        return null
    else if ( key < root.key )
    else if ( key > root.key )
    else
    }
    return root
}
```

Assignment Project Exam Help

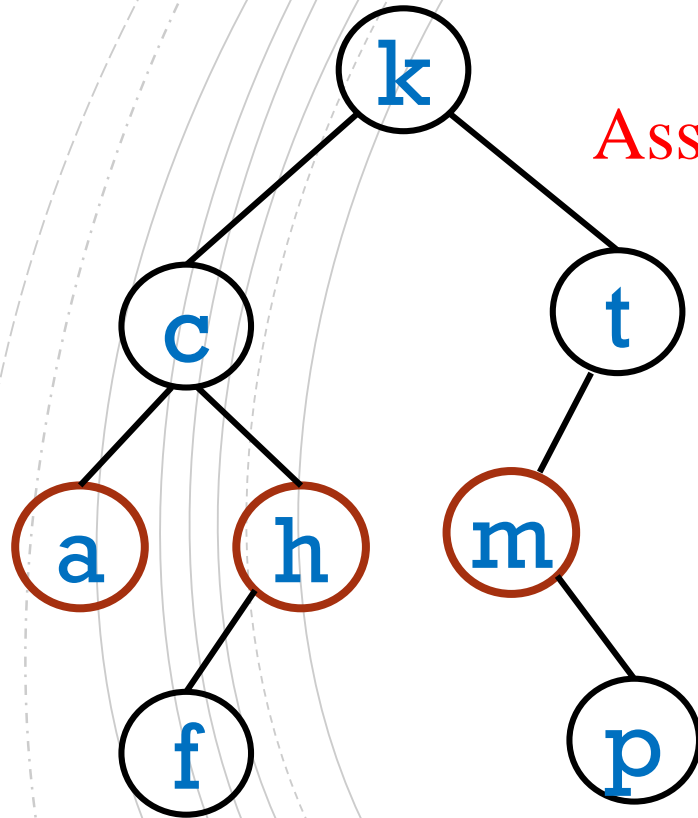
<https://powcoder.com>

Add WeChat powcoder



# Assignment Project Exam Help

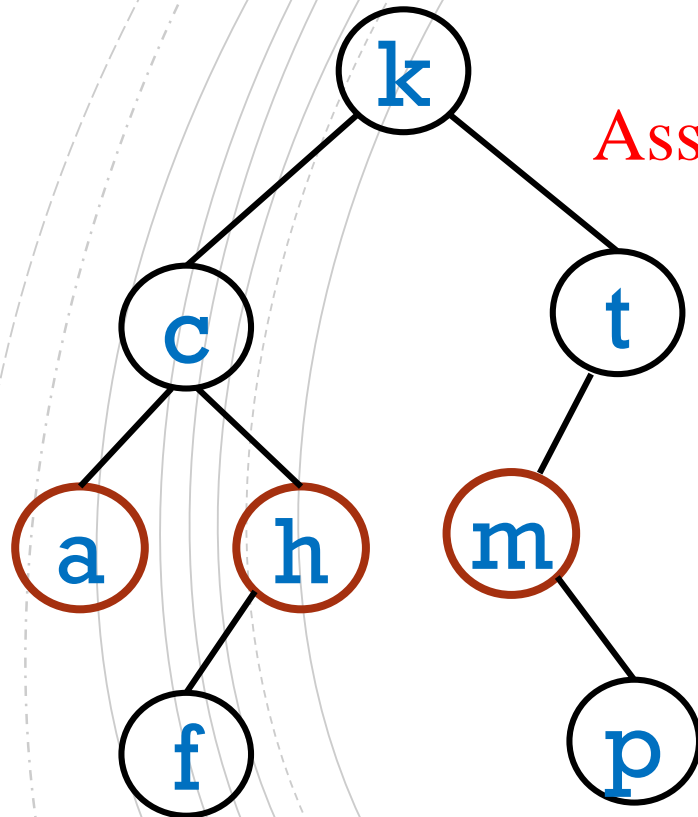
REMOVED IMPLEMENTATION



```
remove (root, key) { // returns root node
    if( root == null )
        return null
    else if ( key < root.key )
        root.left = remove (root.left, key)
    else if ( key > root.key )
        root.right = remove (root.right, key)
    else
        // ... (implementation for finding the node to remove)
    }
    return root
}
```

# Assignment Project Exam Help

REMOVED IMPLEMENTATION

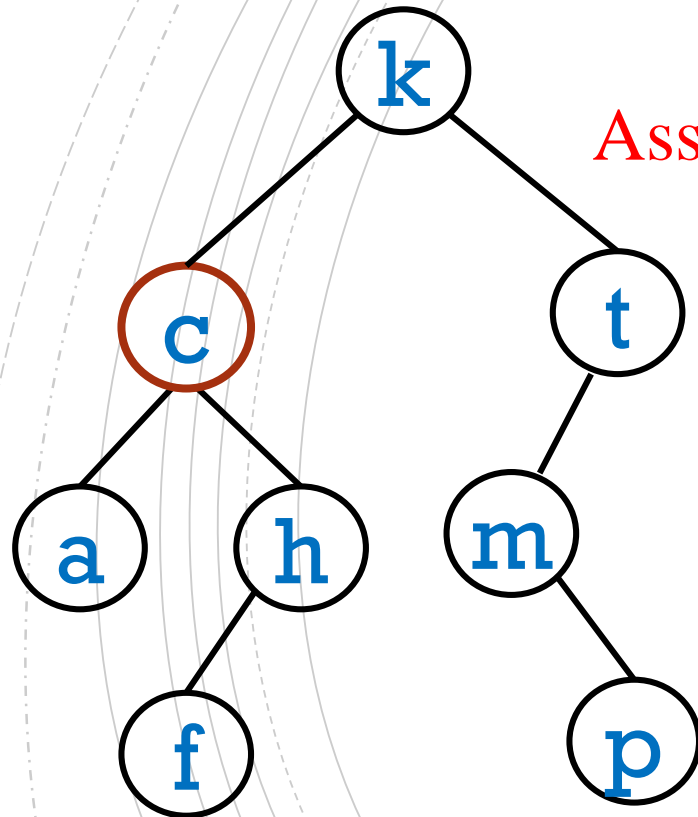


```
remove (root, key) { // returns root node
    if ( root == null )
        return null
    else if ( key < root.key )
        root.left = remove (root.left, key)
    else if ( key > root.key )
        root.right = remove (root.right, key)
    else if (root.left == null)
        root = root.right
    else if (root.right == null)
        root = root.left

    }
    return root
}
```

# Assignment Project Exam Help

REMOVED IMPLEMENTATION

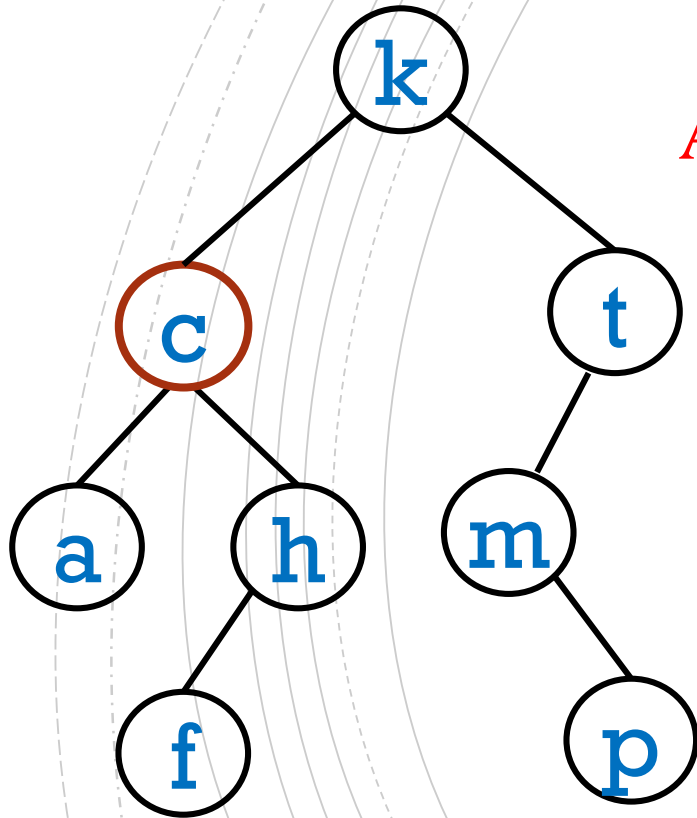


```
remove(root, key){ // returns root node
    if( root == null )
        return null
    else if ( key < root.key )
        root.left = remove(root.left, key)
    else if ( key > root.key )
        root.right = remove(root.right, key)
    else if (root.left == null)
        root = root.right
    else if (root.right == null)
        root = root.left

    }
    return root
}
```

# Assignment Project Exam Help

REMOVED IMPLEMENTATION

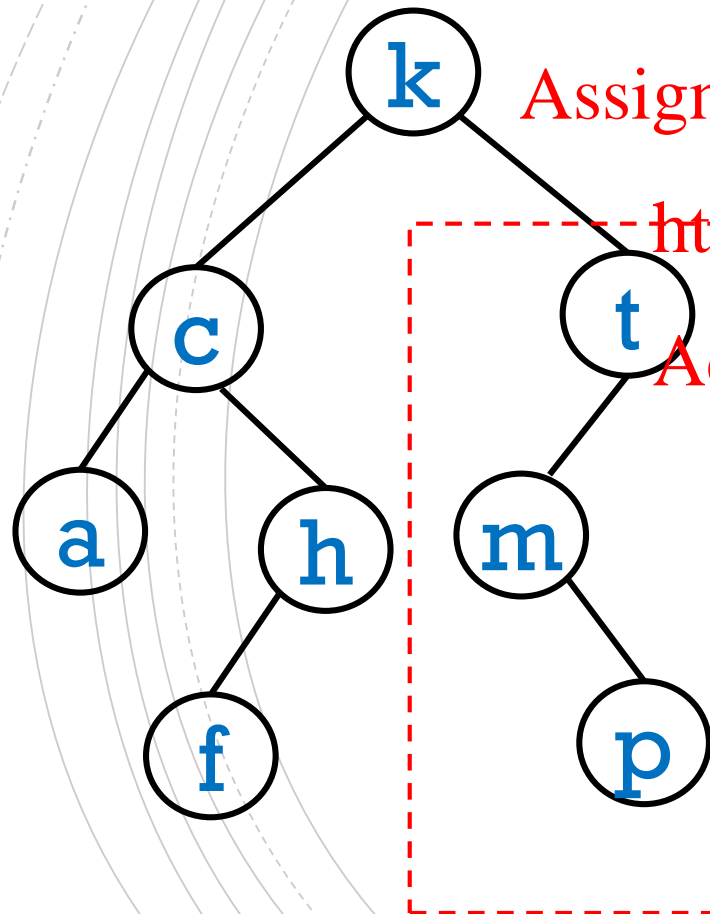


```
remove(root, key){ // returns root node
    if( root == null )
        return null
    else if ( key < root.key )
        root.left = remove(root.left, key)
    else if ( key > root.key )
        root.right = remove(root.right, key)
    else if (root.left == null)
        root = root.right
    else if (root.right == null)
        root = root.left
    else {
        root.key = findMin(root.right).key
        root.right = remove(root.right, root.key)
    }
    return root
}
```

# Assignment Project Exam Help

REMOVE EXAMPLE powcoder

remove (**k**) →



Assignment Project Exam Help

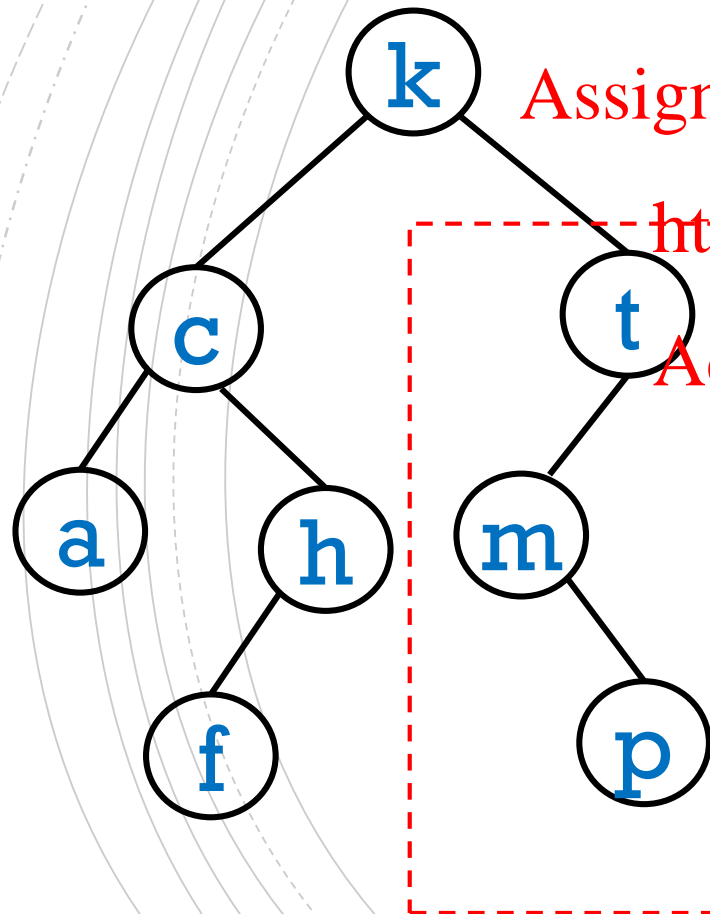
<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

REMOVE Add WeChat powcoder

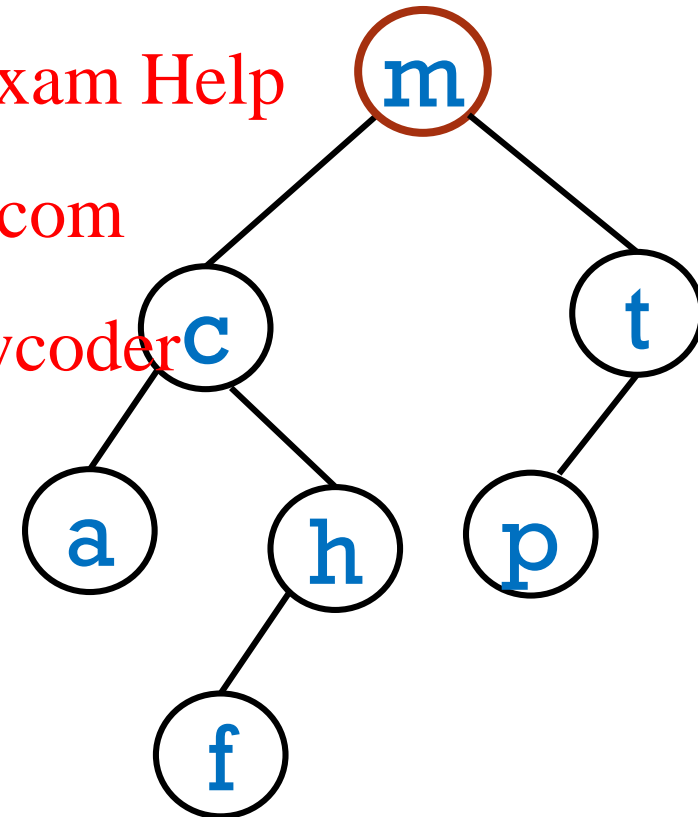
remove(**k**) →



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



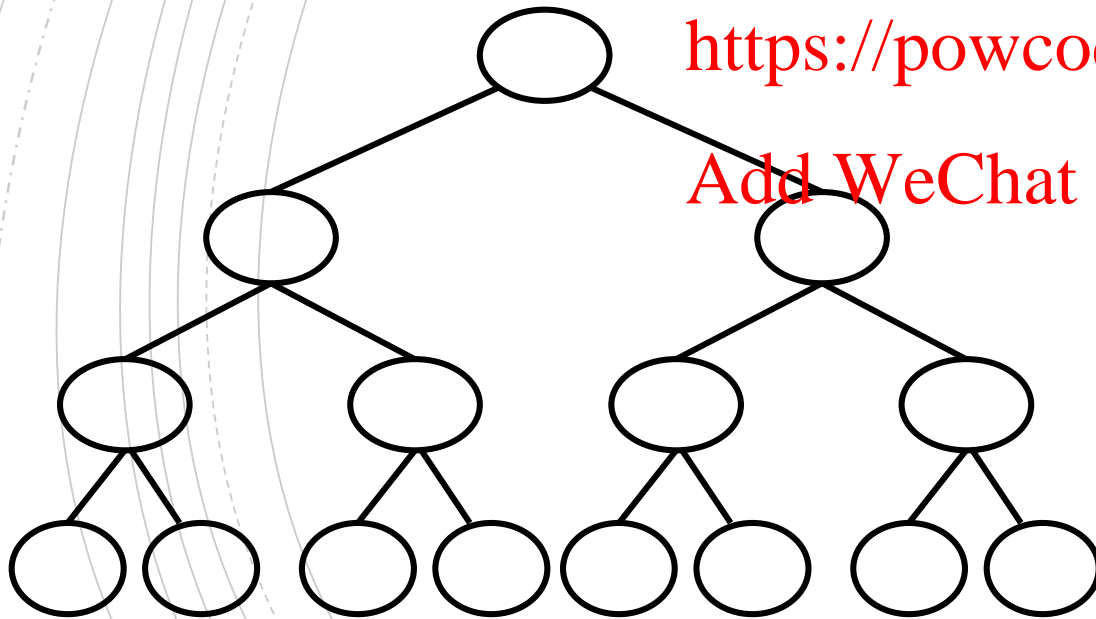
# Assignment Project Exam Help

BALANCED VS UNBALANCED

balanced

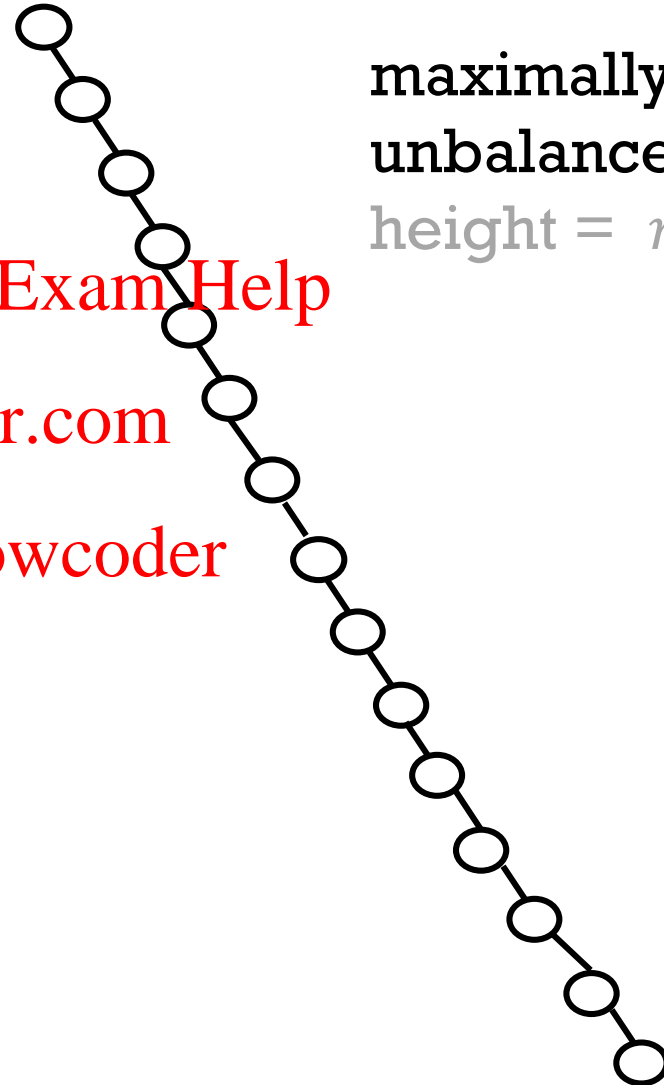
$$\text{height} = \log(n + 1) - 1$$

$$n = 2^{h+1} - 1$$



maximally  
unbalanced

$$\text{height} = n - 1$$



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

## BEST VS WORST CASE SCENARIO

best case

worst case

**findMin()**

**findMax()**

**find( key )**

**add(key)**

**remove(key)**

<https://powcoder.com>

Add WeChat powcoder



# Assignment Project Exam Help

## BEST VS WORST CASE SCENARIO

best case

worst case

findMin()  $O(1)$  findMax()  $O(n)$



<https://powcoder.com>

Add WeChat powcoder

findMax()

find( key )

add(key)

remove(key)

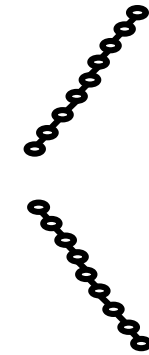
# Assignment Project Exam Help

## BEST VS WORST CASE SCENARIO

	best case	worst case
findMin()	$O(1)$	$O(n)$
findMax()	$O(1)$	$O(n)$
find( key )		
add(key)		
remove(key)		

<https://powcoder.com>

Add WeChat powcoder



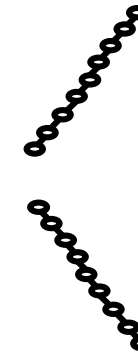
# Assignment Project Exam Help

## BEST VS WORST CASE SCENARIO

	best case	worst case
findMin()	$O(1)$	$O(n)$
findMax()	$O(1)$	$O(n)$
find( key )	$O(1)$	$O(n)$ could be zigzag
add(key)		
remove(key)		

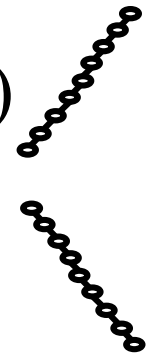
<https://powcoder.com>

Add WeChat powcoder



# Assignment Project Exam Help

## BEST VS WORST CASE SCENARIO

	best case	worst case	
findMin()	$O(1)$	$O(n)$	 Could be zigzag
findMax()	$O(1)$	$O(n)$	
find( key )	$O(1)$	$O(n)$	
add(key)	$O(1)$	$O(n)$	
remove(key)	$O(1)$	$O(n)$	

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

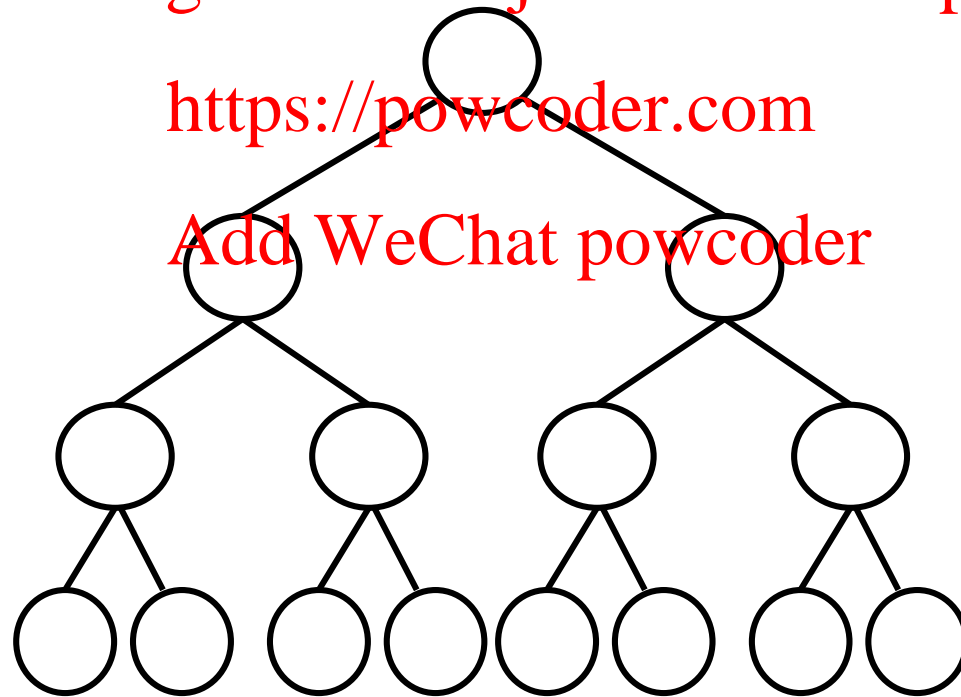
## BINARY SEARCH TREES

When a binary search tree is balanced, then finding a key is very similar to a binary search.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# Assignment Project Exam Help

## BALANCED BINARY SEARCH TREES

(COMP 251: AVL TREES, RED-BLACK TREES)

	best case	worst case
findMin()	$O(\log n)$	$O(\log n)$
findMax()	$O(\log n)$	$O(\log n)$
find( key )	$O(1)$	$O(\log n)$
add(key)	$O(\log n)$	$O(\log n)$
remove(key)	$O(\log n)$	$O(\log n)$

Assignment Project Exam Help

Coming Soon  
Add WeChat powcoder

Assignment Project Exam Help

In the next videos:

- Heaps

<https://powcoder.com>

Add WeChat powcoder