

Assignment Project Exam Help

Add WeChat edu\_assist\_pro

# COMP 250

## INTRODUCTORY SCIENCE

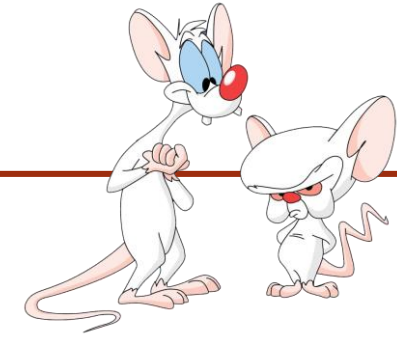
Week 12-1: Binary

Giulia Alberini, Fall 2020

Slides adapted from Michael Langer's

# Assignment Project Exam Help

WHAT ARE WE DOING THIS VIDEO?



- Binary Search Trees

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

Assignment Project Exam Help

Add WeChat edu\_assist\_pro

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# Assignment Project Exam Help

~~BSTNode Add WeChat edu\_assist\_pro~~

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

Assignment Project Exam Help

<https://eduassistpro.github.io/>

```
class
    K Key;
    BSTNode<K> leftchild;
    BSTNode<K> rightchild;
    :
}
```

# Assignment Project Exam Help

BINARY SEARCH Tree Chat edu\_assist\_pro

- binary tree

## Assignment Project Exam Help

- keys are comparable, a

<https://eduassistpro.github.io/>

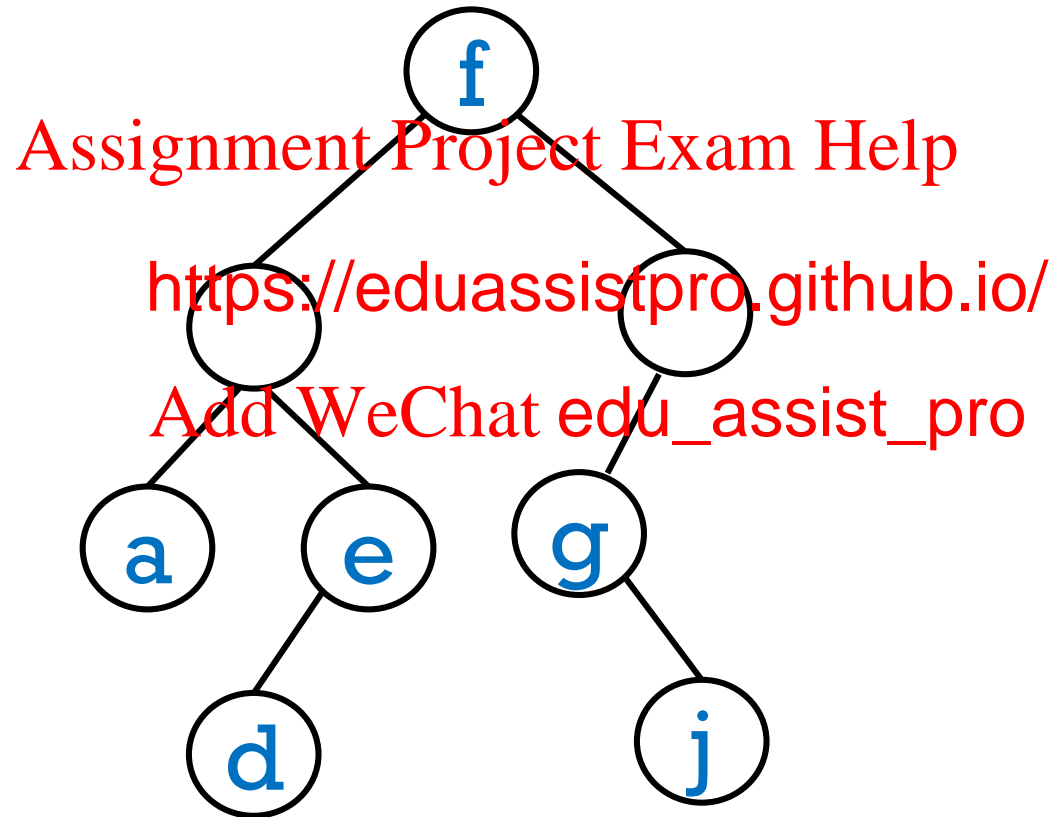
Add WeChat edu\_assist\_pro

- 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

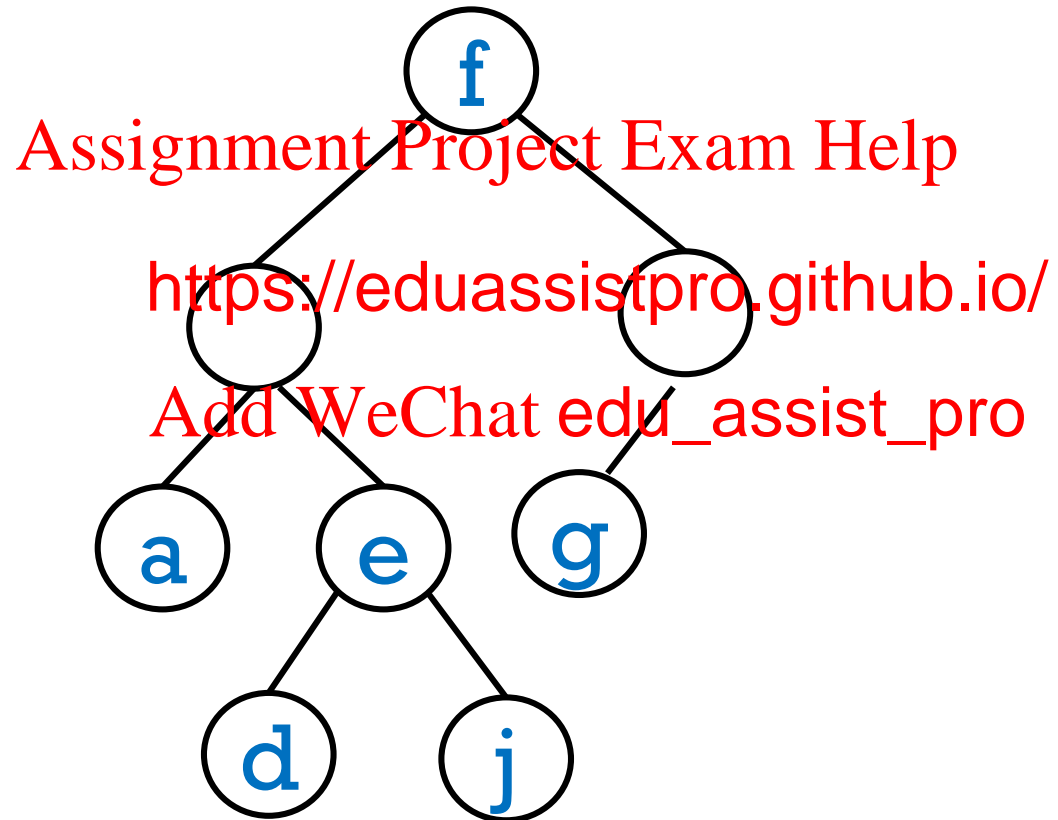
EXAMPLE Add WeChat edu\_assist\_pro

---



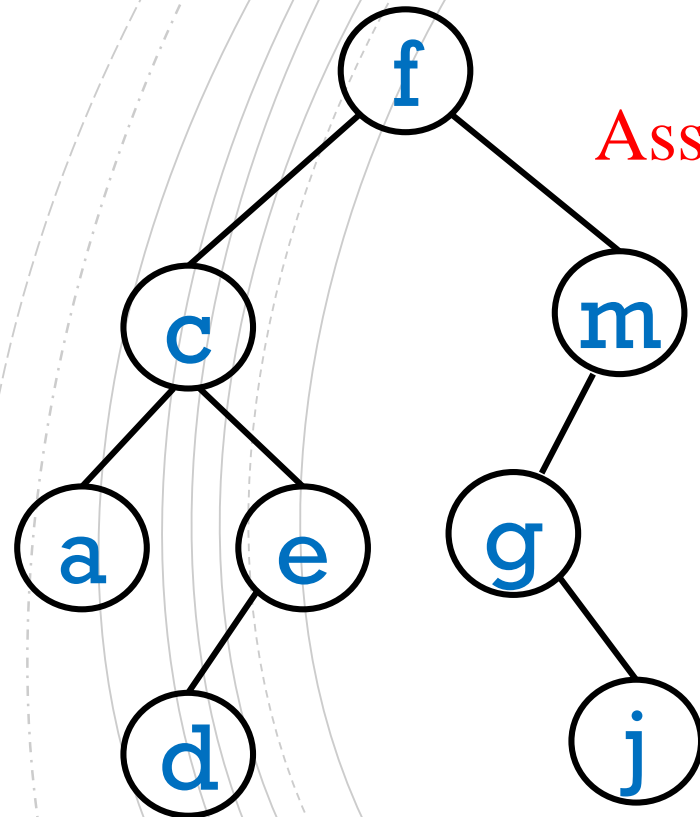
# Assignment Project Exam Help

THIS IS NOT A WeChat edu\_assist\_pro



# Assignment Project Exam Help

BST - TRAVERSALS



Assignment Project Exam Help  
An in-order traversal on a BST visits the natural order

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



# Assignment Project Exam Help

BINARY SEARCH WeChat edu\_assist\_pro

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

Assignment Project Exam Help

<https://eduassistpro.github.io/>

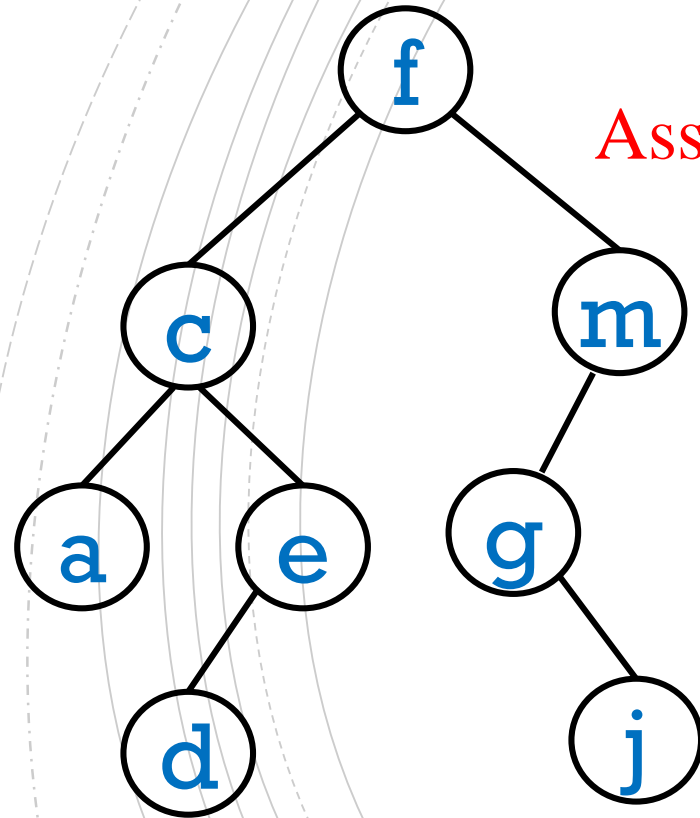
Add WeChat edu\_assist\_pro

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

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

# Assignment Project Exam Help

~~FIND()~~ ~~Add WeChat edu\_assist\_pro~~



Assignment Project Exam Help

viour:

<https://eduassistpro.github.io/>

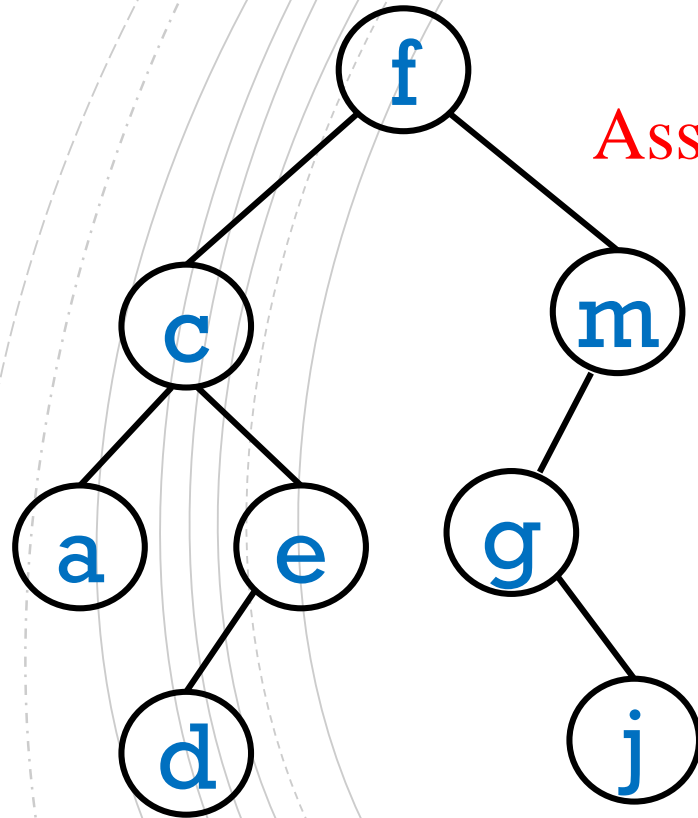
~~Add WeChat edu\_assist\_pro~~

, **g**) returns the **g** node

**s**) returns null.

# Assignment Project Exam Help

FIND() — Add WeChat edu\_assist\_pro



Assignment Project Exam Help

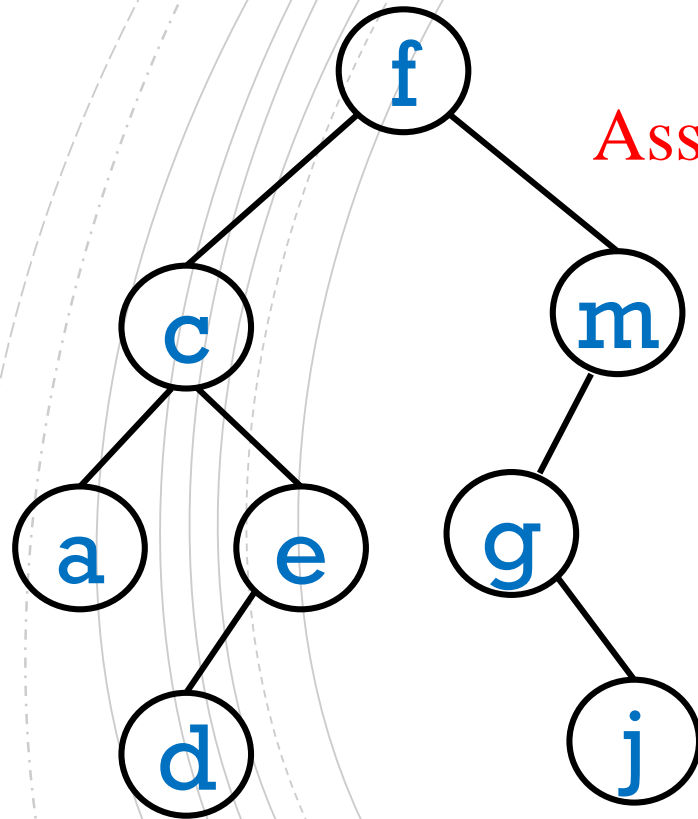
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

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

# Assignment Project Exam Help

FIND() — Add WeChat edu\_assist\_pro



Assignment Project Exam Help

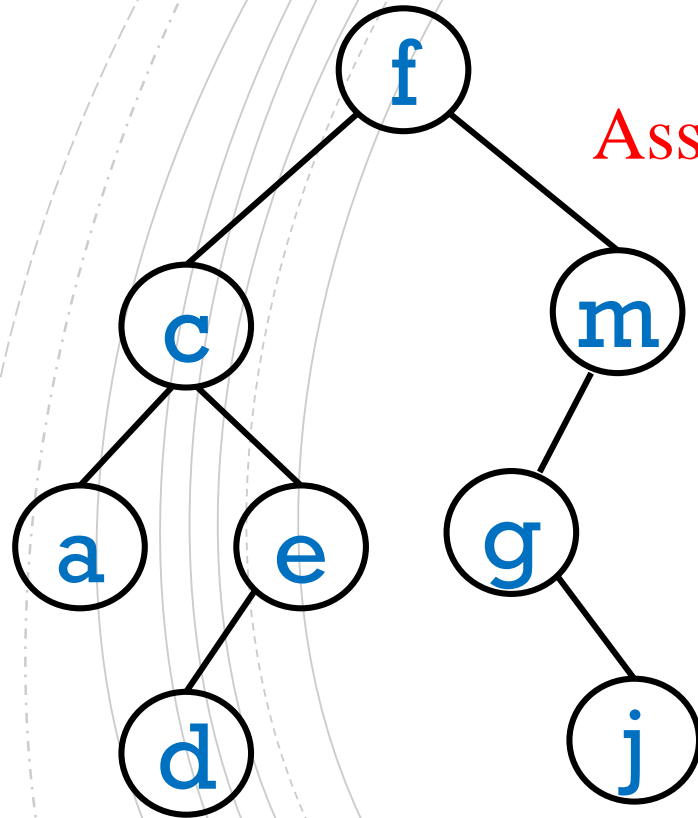
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

```
find(root, key) { // returns a node
    if (root == null)
        return null
    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 edu\_assist\_pro



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

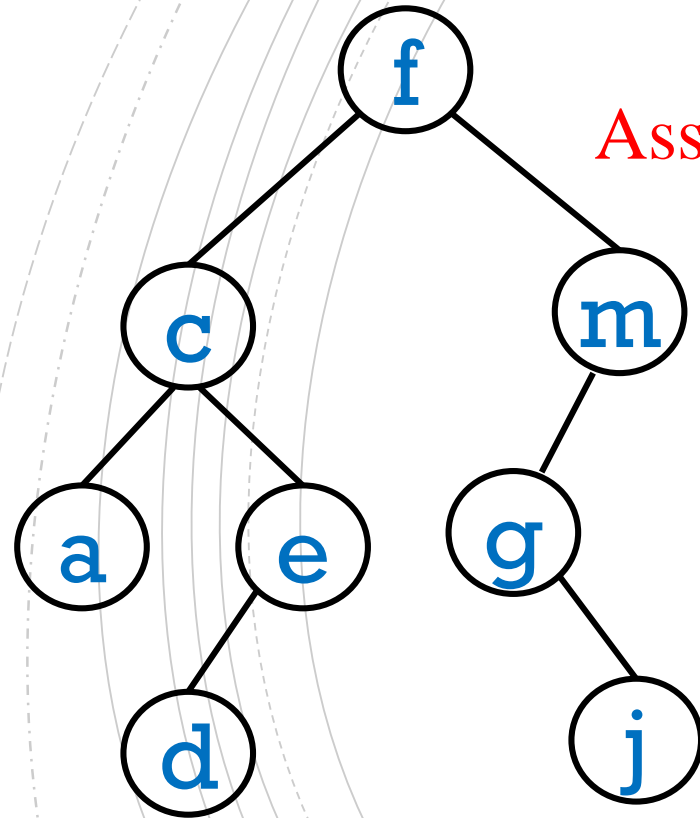
uld return the node with

ey. So, for example given

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

# Assignment Project Exam Help

~~FINDMIN Add WeChat edu\_assist\_pro~~



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat ~~edu\_assist\_pro~~

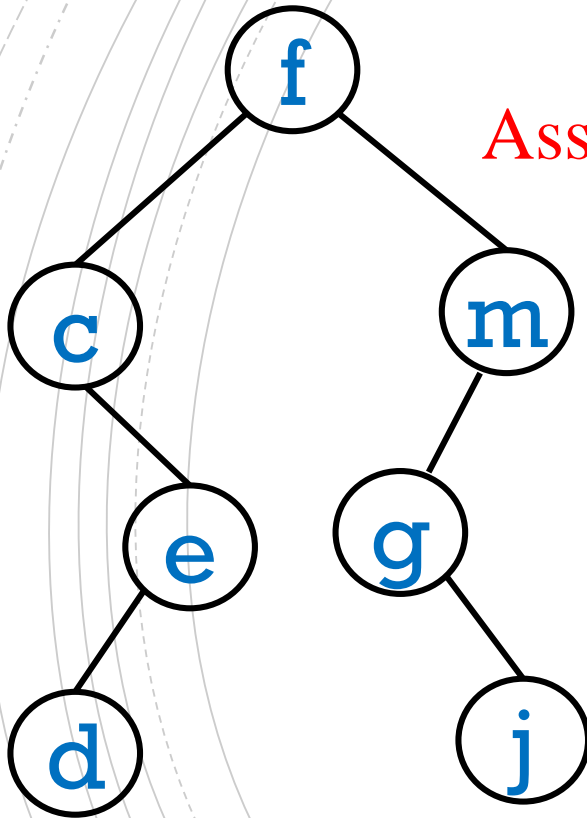
uld return the node with

ey. So, for example given

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

# Assignment Project Exam Help

~~FINDMIN Add WeChat edu\_assist\_pro~~



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat ~~edu\_assist\_pro~~

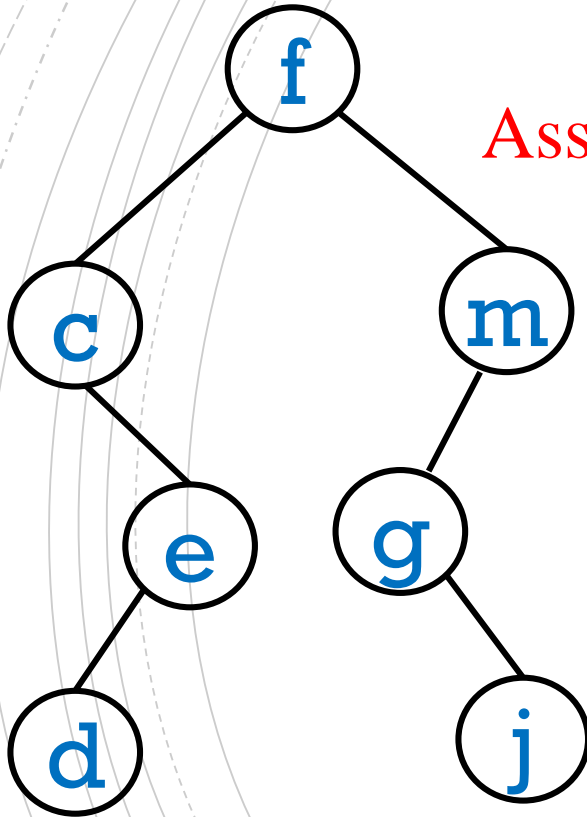
uld return the node with

ey. So, for example given

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

# Assignment Project Exam Help

~~FINDMIN Add WeChat edu\_assist\_pro~~



## Assignment Project Exam Help

<https://eduassistpro.github.io/>

~~Add WeChat edu\_assist\_pro~~

uld return the node with

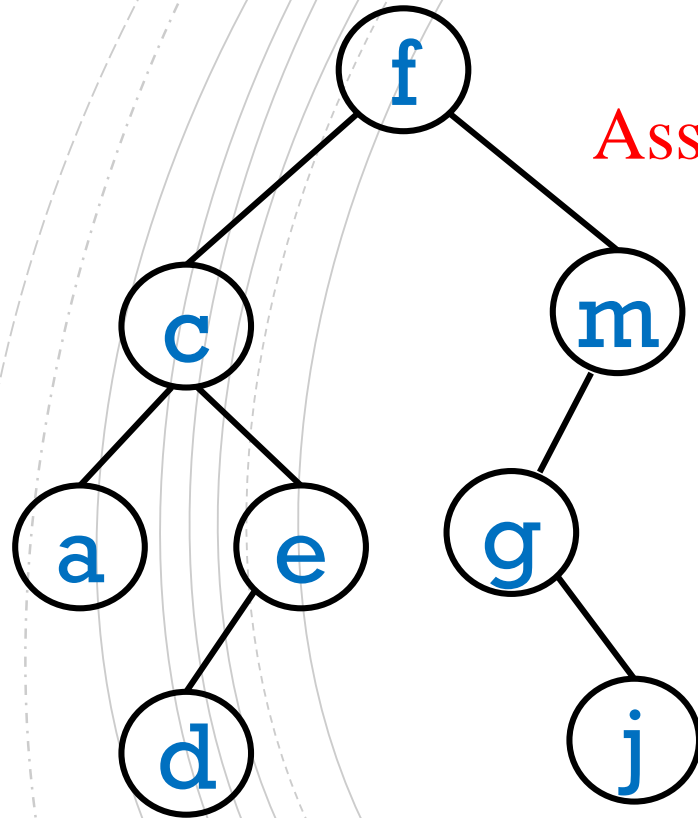
ey. So, for example given

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



# Assignment Project Exam Help

FINDMIN Add WeChat edu\_assist\_pro



Assignment Project Exam Help

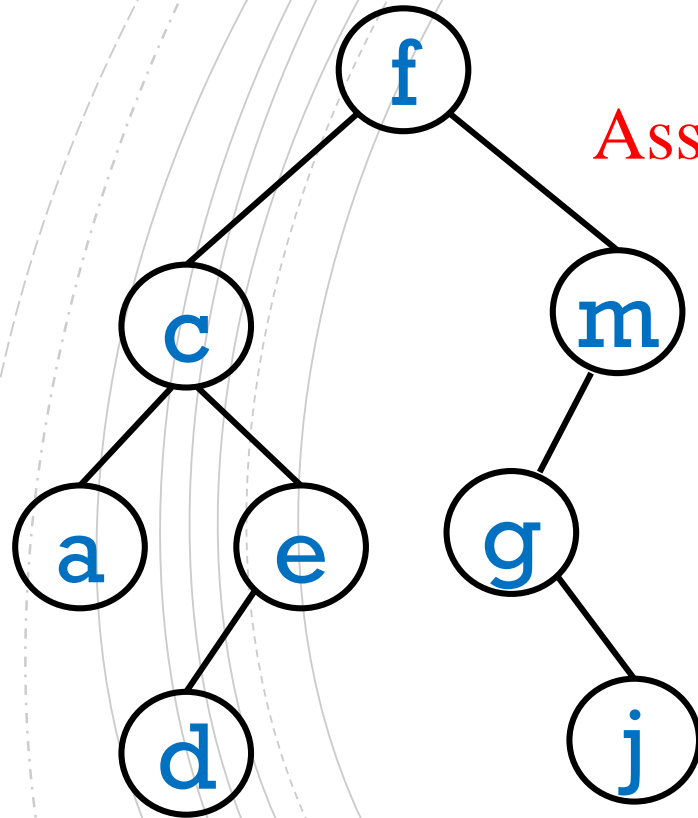
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

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

# Assignment Project Exam Help

FINDMIN Add WeChat edu\_assist\_pro



Assignment Project Exam Help

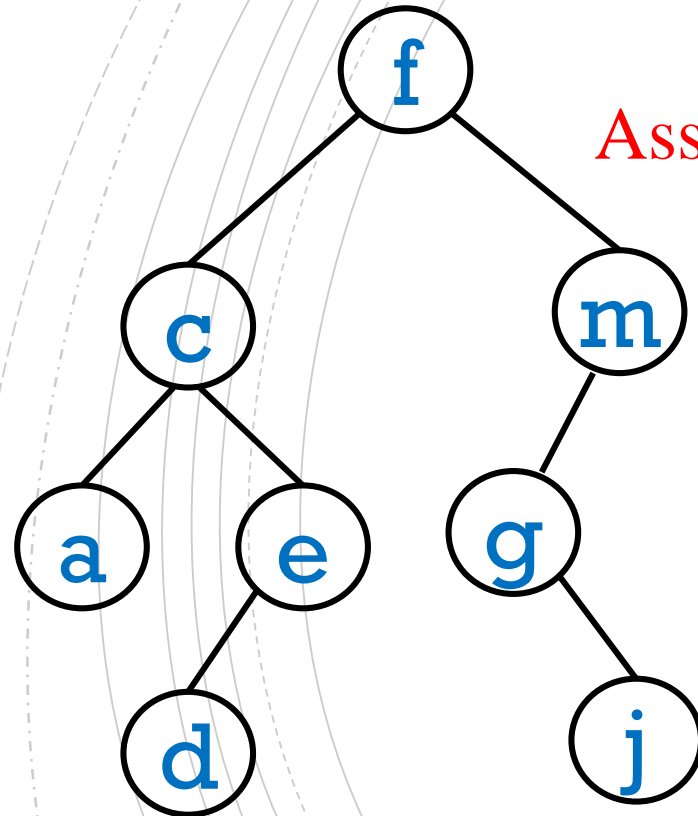
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

```
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 edu\_assist\_pro



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

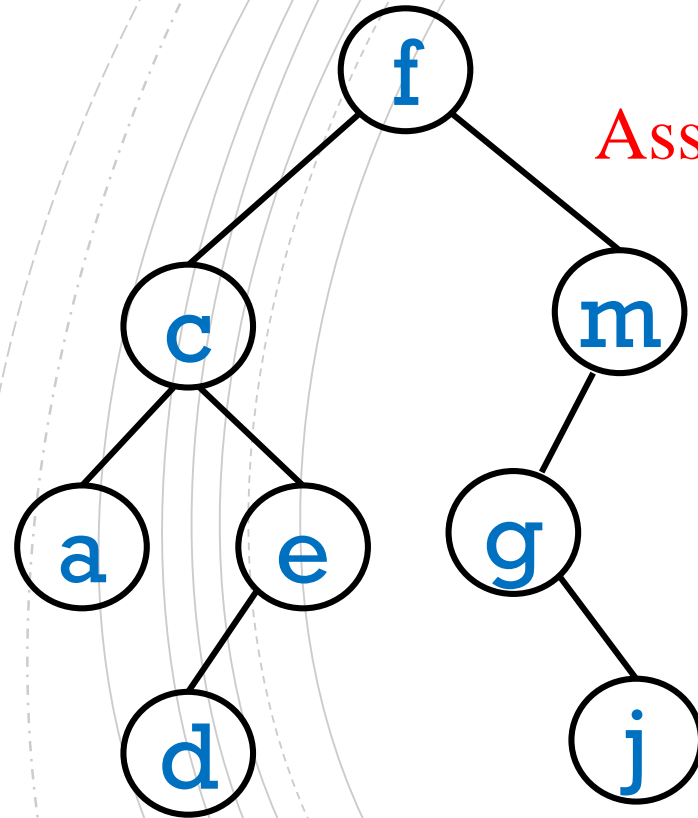
uld return the node with

ey. So, for example given

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

# Assignment Project Exam Help

~~FINDMAX()~~ Add WeChat edu\_assist\_pro



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

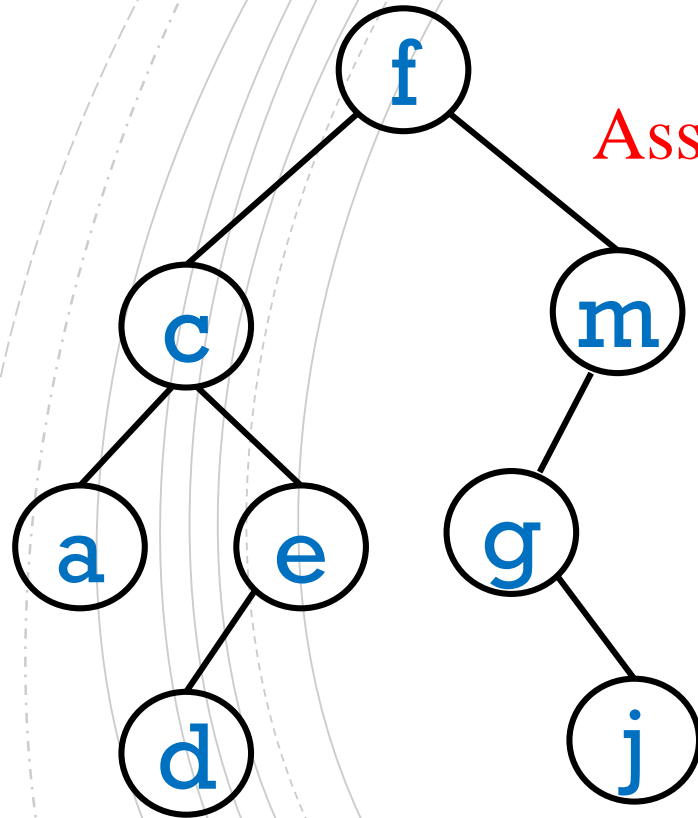
uld return the node with

ey. So, for example given

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

# Assignment Project Exam Help

FINDMAX() IMPLEMENTATION



Assignment Project Exam Help

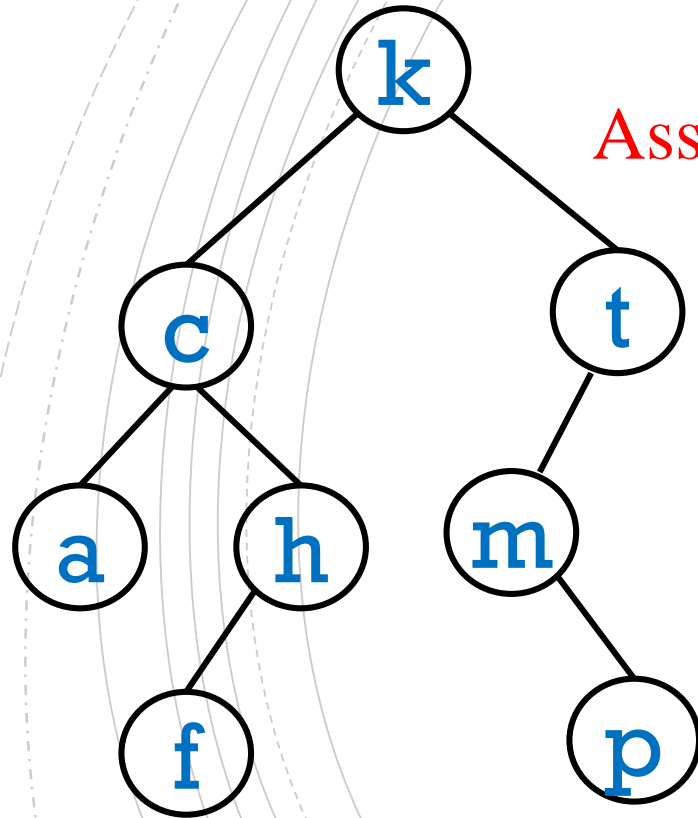
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

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

# Assignment Project Exam Help

ADD() Add WeChat edu\_assist\_pro



Assignment Project Exam Help

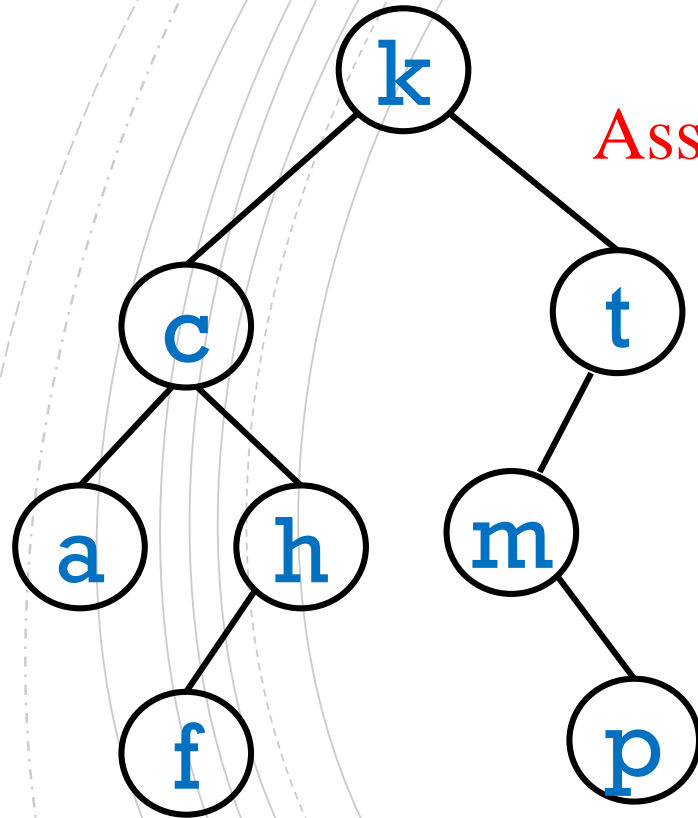
ld add a BSTNode to the tree.

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# Assignment Project Exam Help

ADD() Add WeChat edu\_assist\_pro



Assignment Project Exam Help

ld add a BSTNode to the tree.

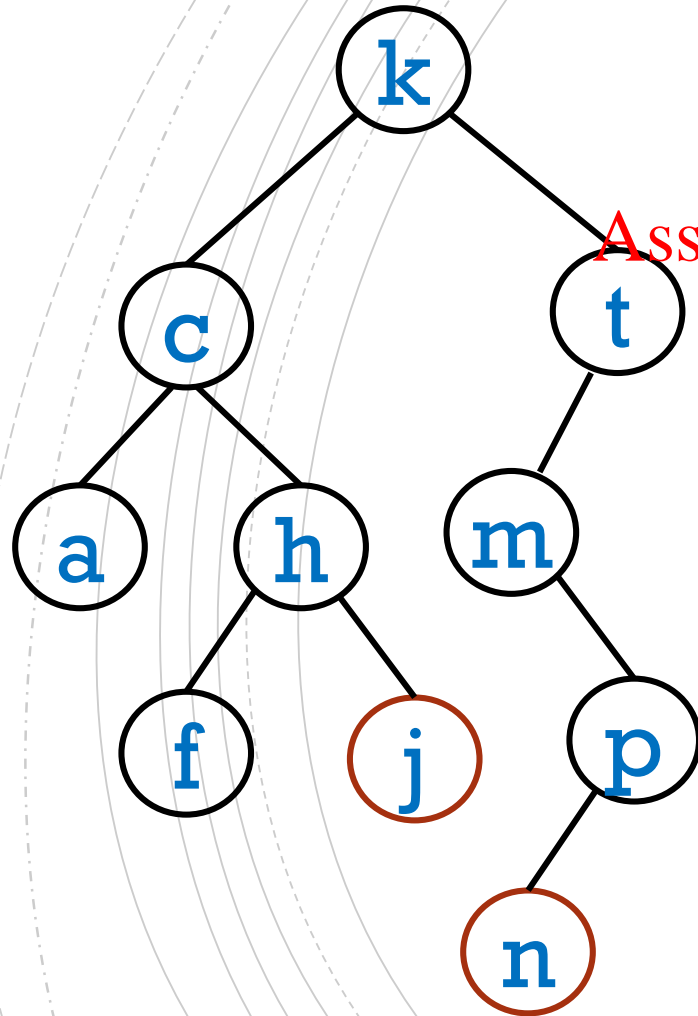
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

A new node is always a leaf.

# Assignment Project Exam Help

ADD() Add WeChat edu\_assist\_pro



Assignment Project Exam Help

uld add a BSTNode to the

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

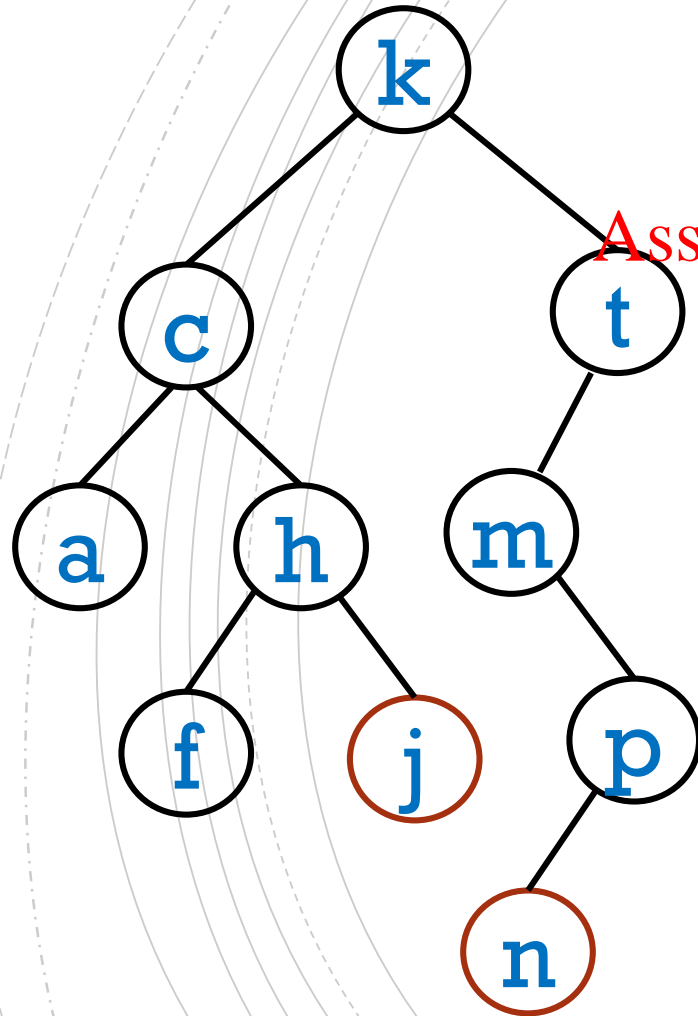
- add(**n**) ?

A new node is always a leaf.



# Assignment Project Exam Help

ADD() - Add WeChat edu\_assist\_pro



Assignment Project Exam Help

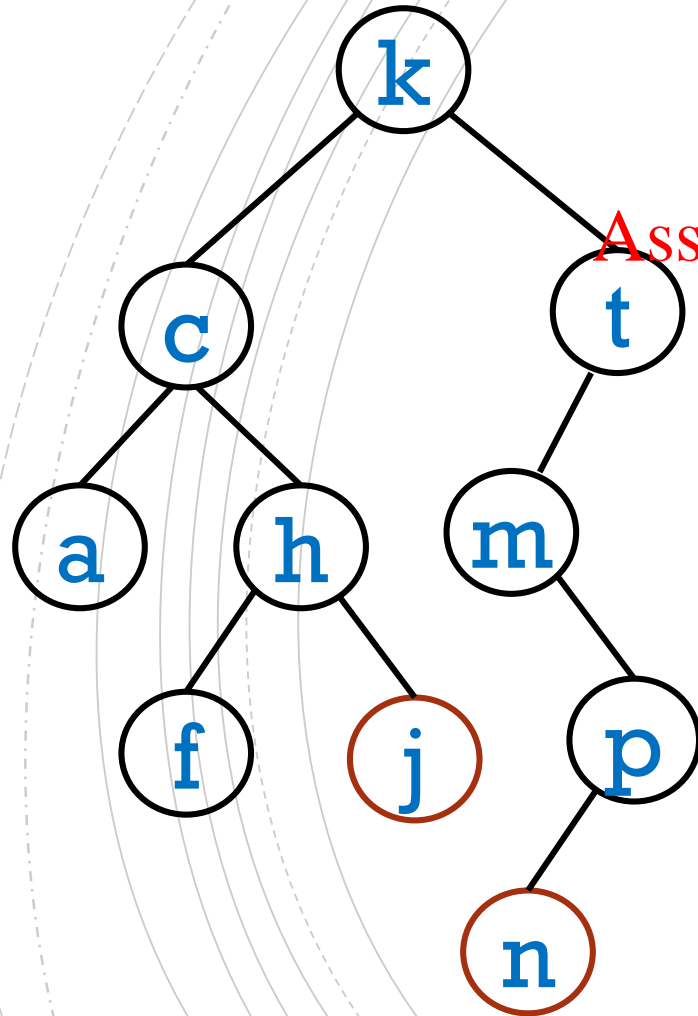
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

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

# Assignment Project Exam Help

ADD() - Add WeChat edu\_assist\_pro



Assignment Project Exam Help

<https://eduassistpro.github.io/>

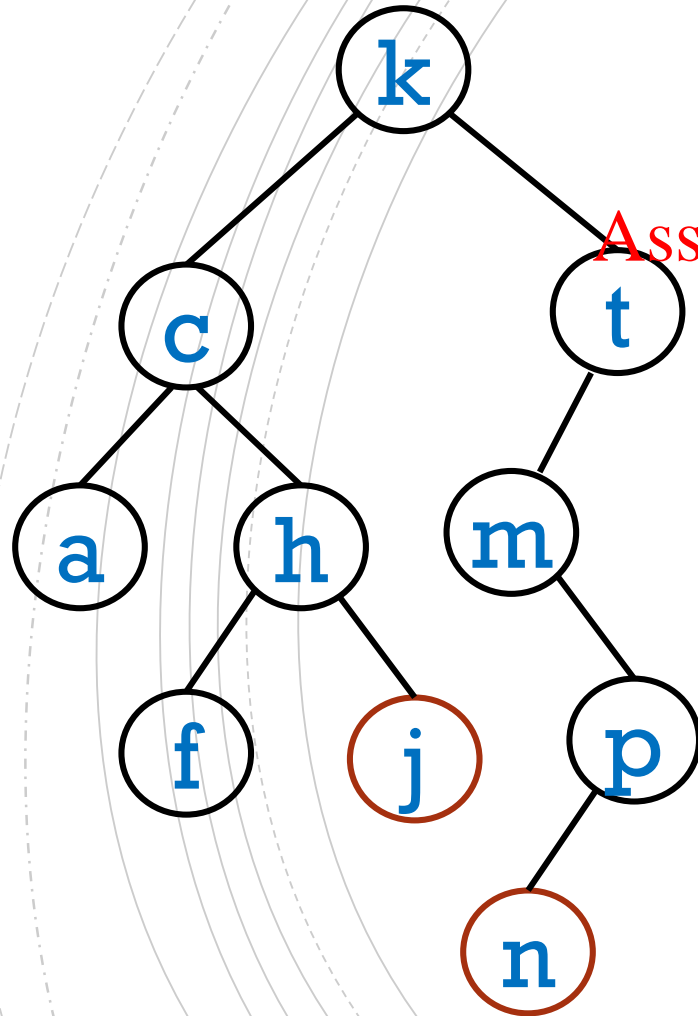
Add WeChat edu\_assist\_pro

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

    return root
}
```

# Assignment Project Exam Help

ADD() - Add When at edu\_assist\_pro



Assignment Project Exam Help

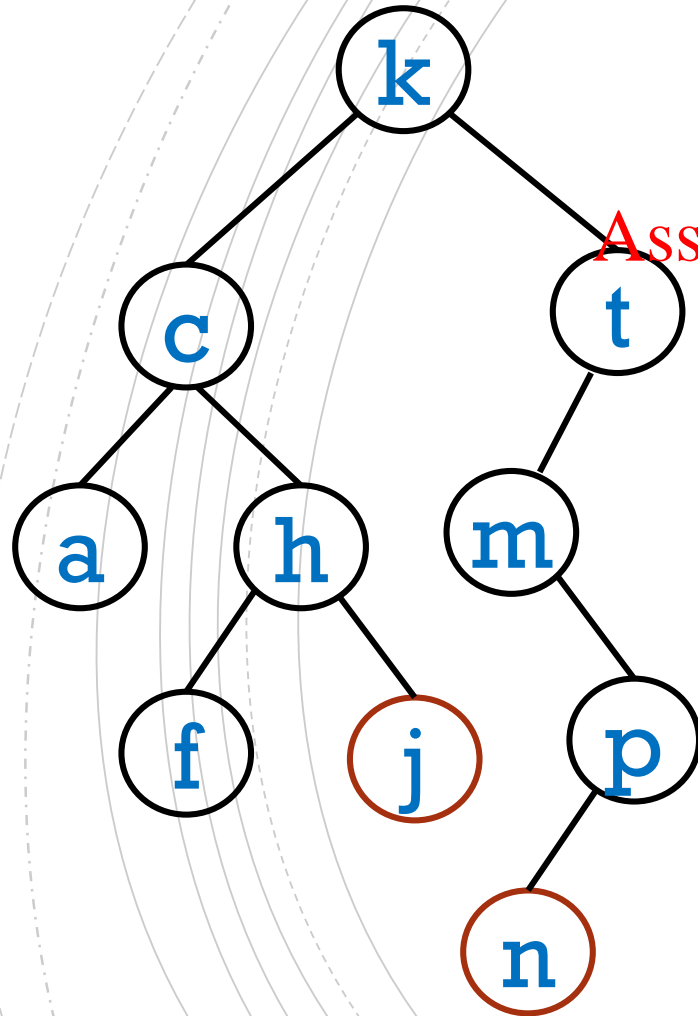
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

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

# Assignment Project Exam Help

ADD() - Add When



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

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

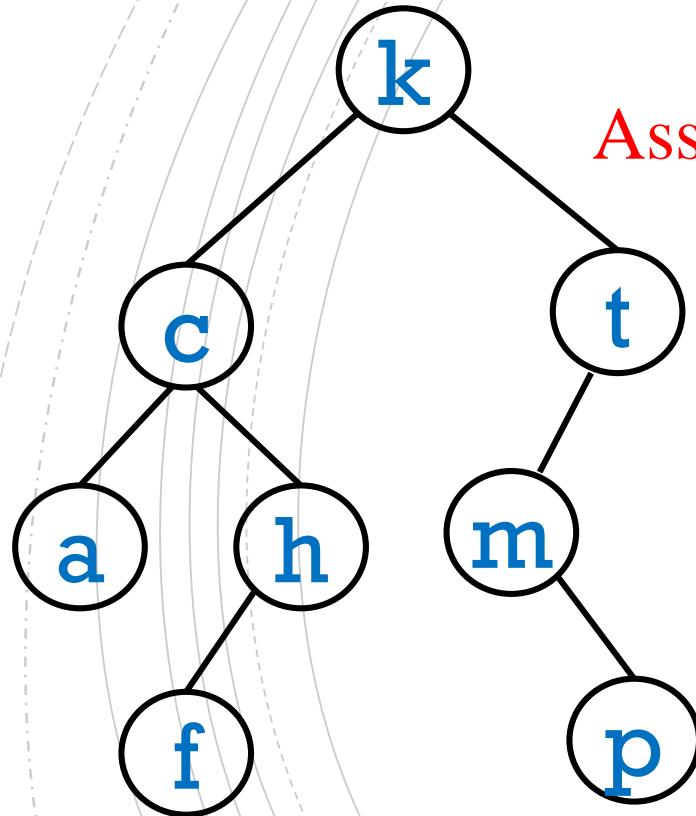
Q: What happens if root.key == key?

A: Nothing!

# Assignment Project Exam Help

~~REMOVE~~ Add WeChat edu\_assist\_pro

remove(**c**) →



Assignment Project Exam Help

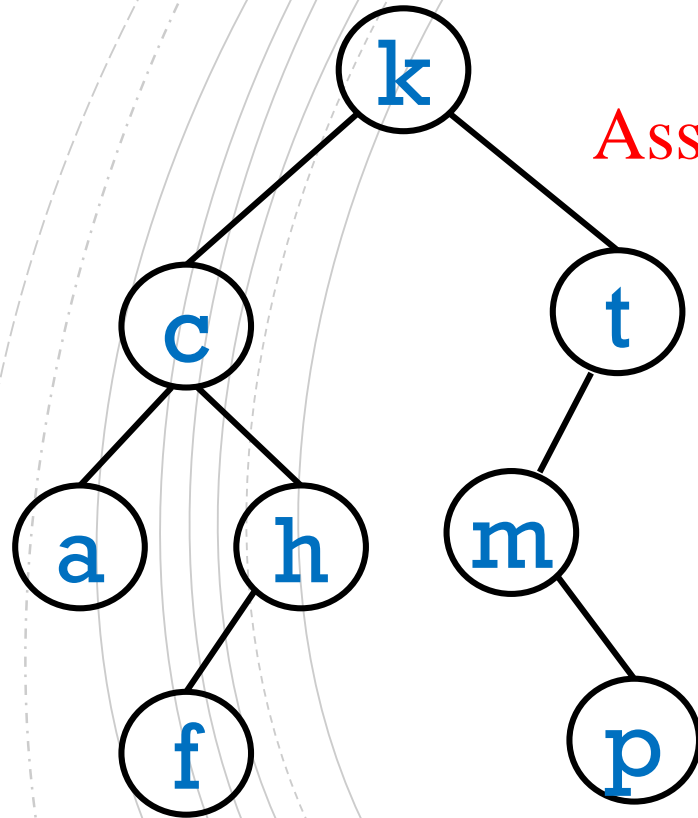
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# Assignment Project Exam Help

~~REMOVE Add WeChat edu\_assist\_pro~~

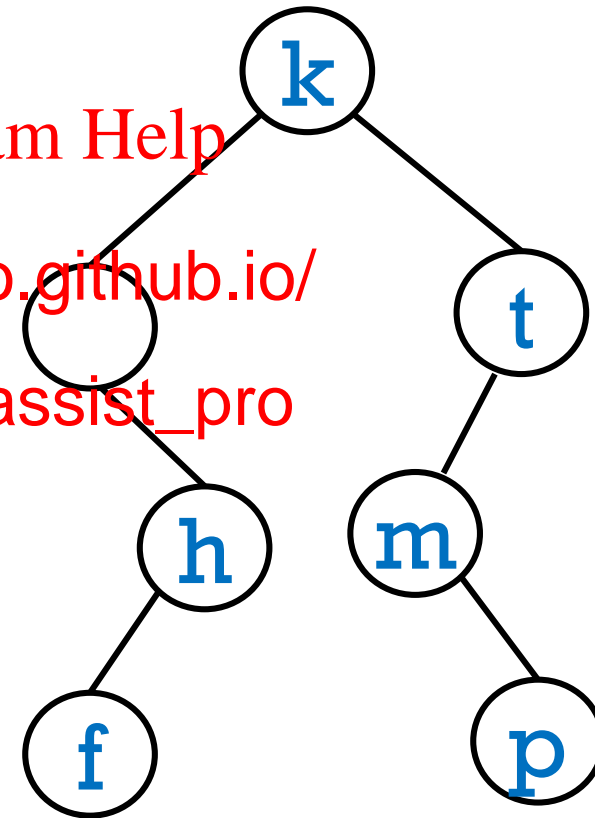
remove(**c**) → this is one way to do it



Assignment Project Exam Help

<https://eduassistpro.github.io/>

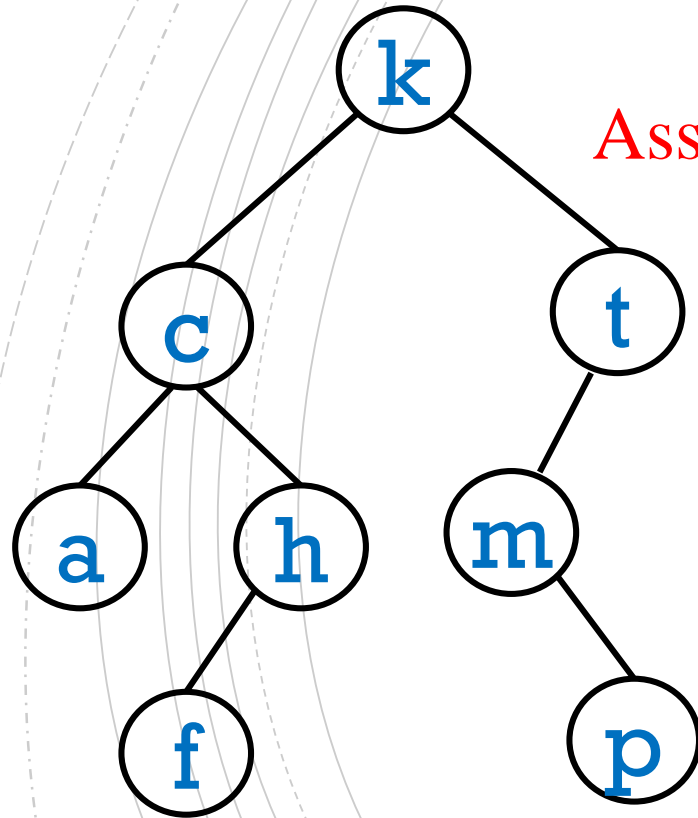
Add WeChat edu\_assist\_pro



# Assignment Project Exam Help

~~REMOVE Add WeChat edu\_assist\_pro~~

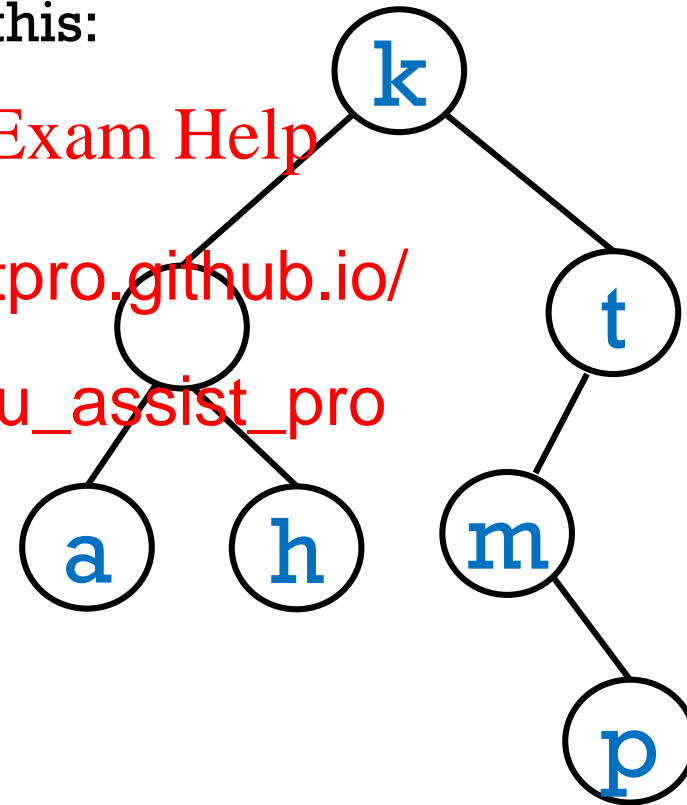
remove(**c**) → the following algorithm  
does this:



Assignment Project Exam Help

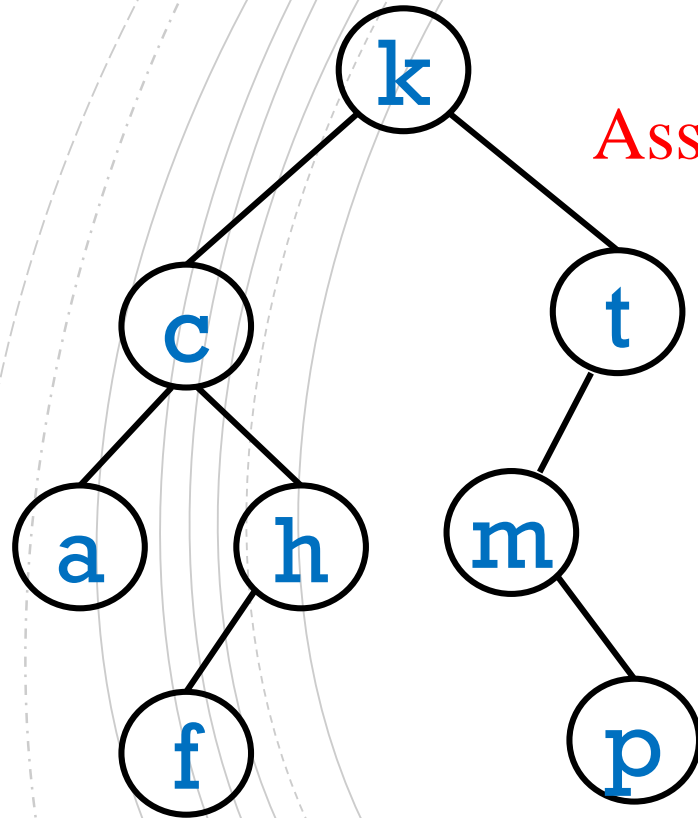
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



# Assignment Project Exam Help

REMOVE Add WeChat edu\_assist\_pro



Assignment Project Exam Help

<https://eduassistpro.github.io/>

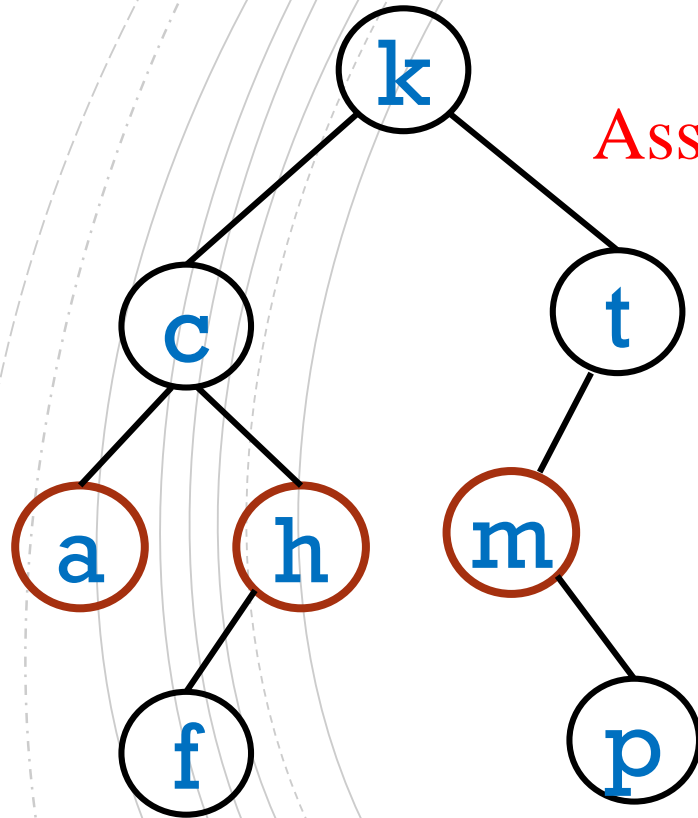
Add WeChat edu\_assist\_pro

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



# Assignment Project Exam Help

REMOVE Add WeChat edu\_assist\_pro



```
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
        // Node to be removed
    }
    return root
}
```

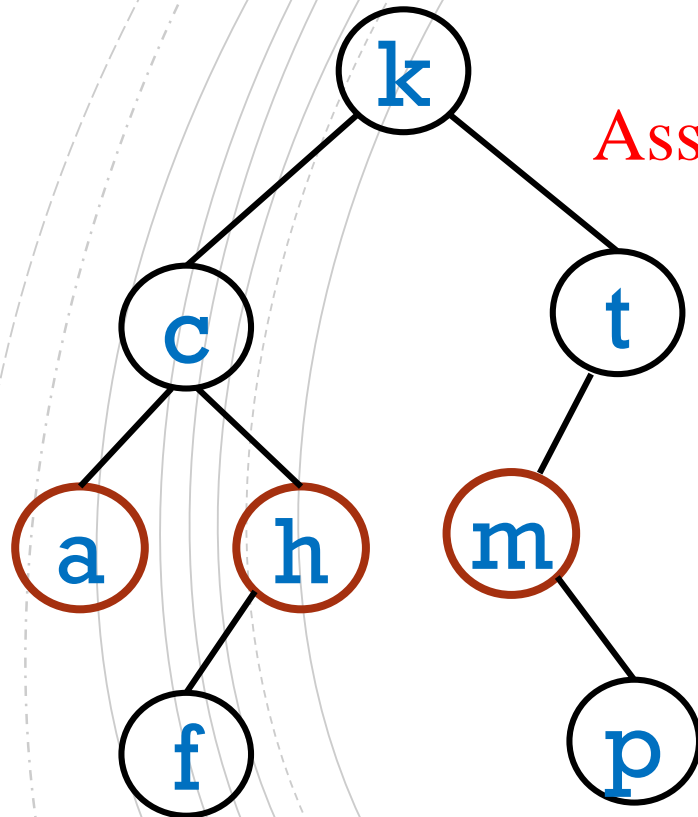
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# Assignment Project Exam Help

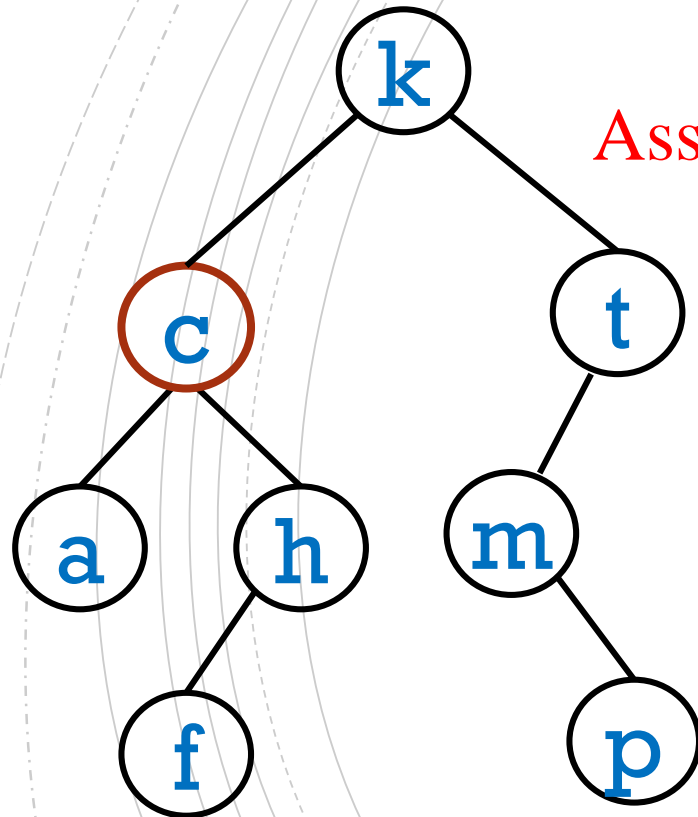
REMOVE Add WeChat edu\_assist\_pro



```
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 ( key == root.key )
        if ( root.left == null )
            return root.right
        else if ( root.right == null )
            return root.left
        else
            // replace with in-order successor
            // ...
    }
    return root
}
```

# Assignment Project Exam Help

REMOVE Add WeChat edu\_assist\_pro



Assignment Project Exam Help

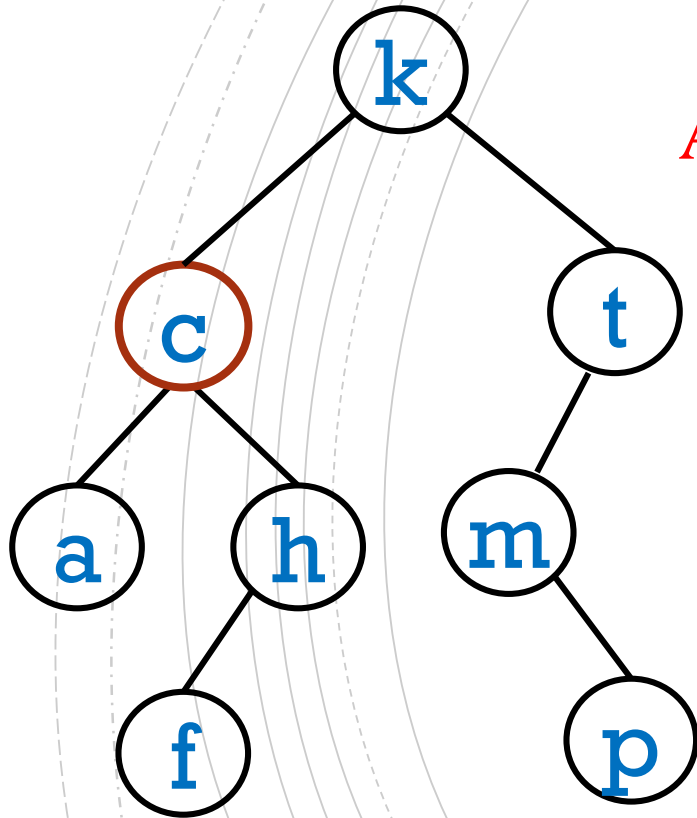
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

```
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 ( key == root.key )
        if ( root.left == null )
            return root.right
        else if ( root.right == null )
            return root.left
        else
            // find the inorder successor
            // ...
    }
    return root
}
```

# Assignment Project Exam Help

REMOVE Add WeChat edu\_assist\_pro



Assignment Project Exam Help

```
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.key == key )
        root = findMin(root.right)
        root.key = findMin(root.right).key
        root.right = remove(root.right, root.key)
    }
    return root
}
```

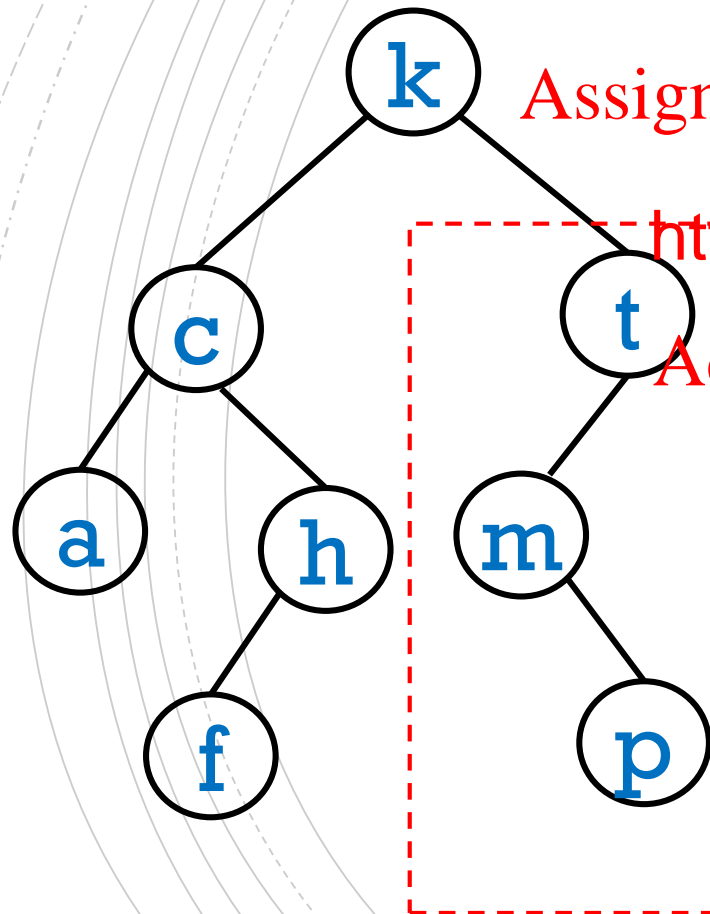
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# Assignment Project Exam Help

REMOVE EXAM Add WeChat edu\_assist\_pro

remove(**k**) →



Assignment Project Exam Help

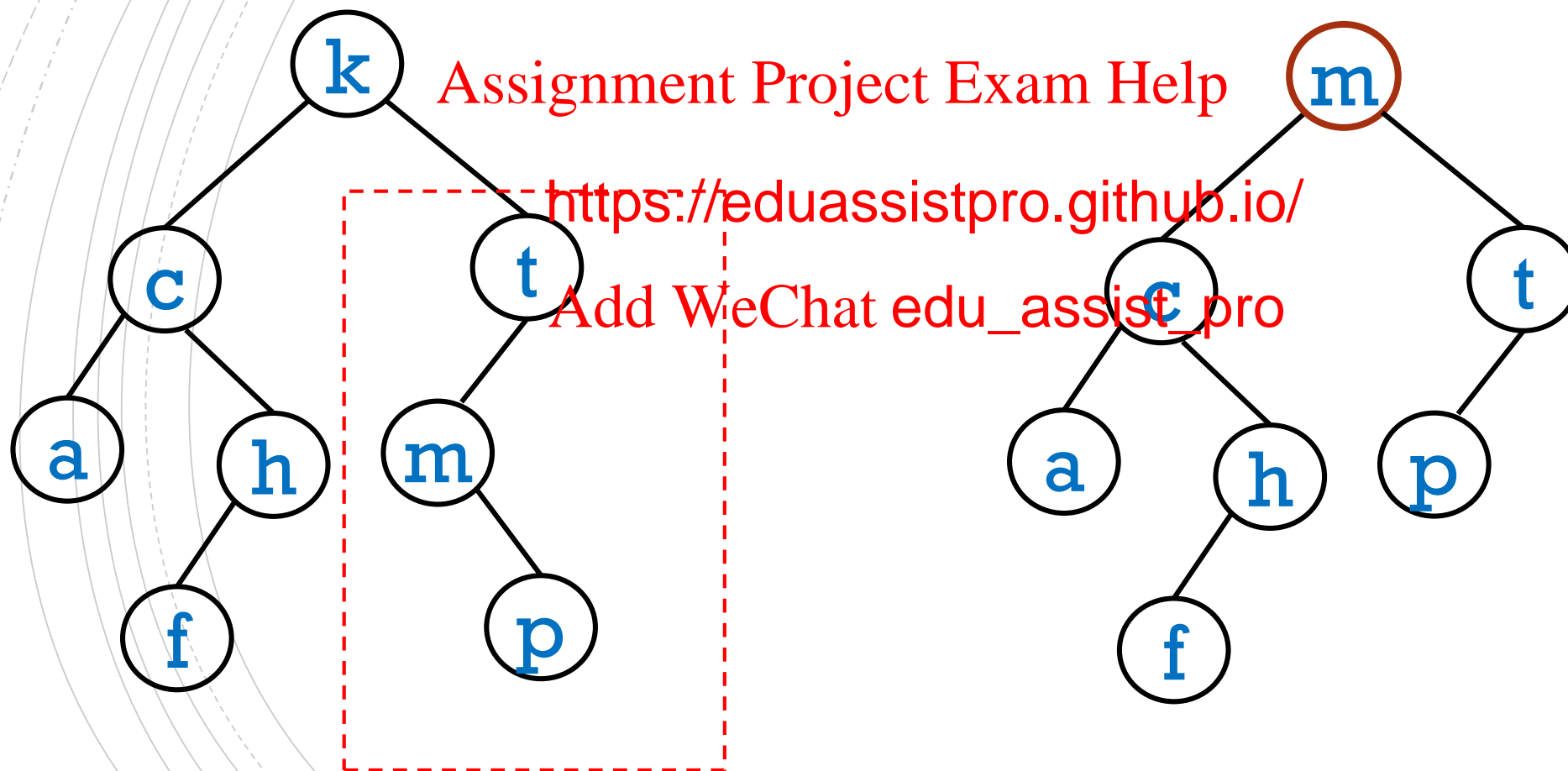
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# Assignment Project Exam Help

REMOVE Add WeChat edu\_assist\_pro

remove(**k**) →



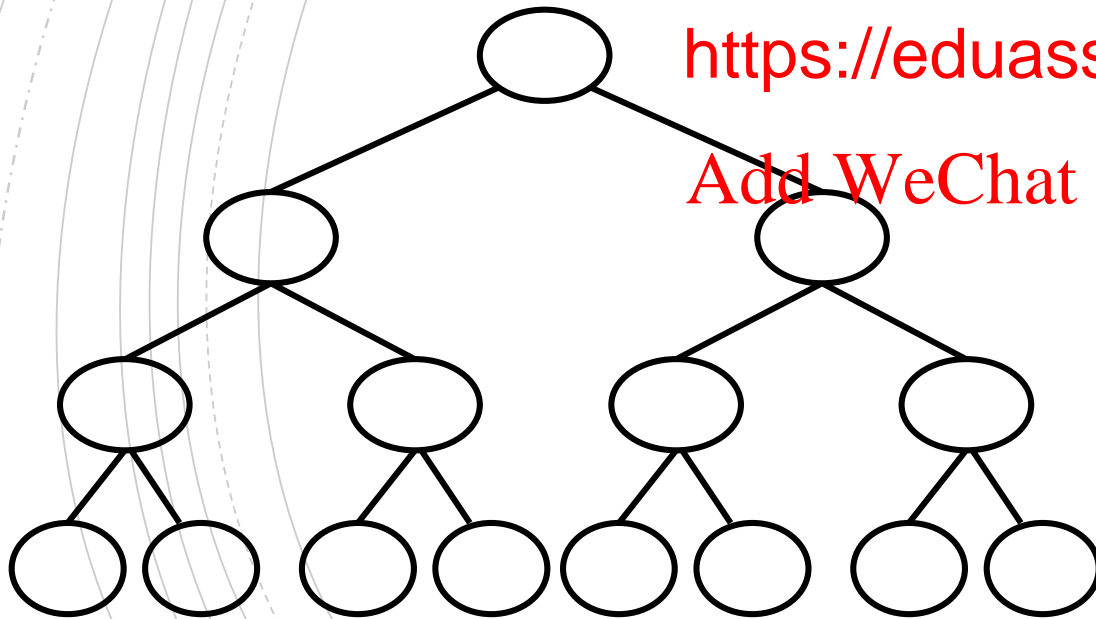
# Assignment Project Exam Help

BALANCED Add WeChat edu\_assist\_pro

balanced

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

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

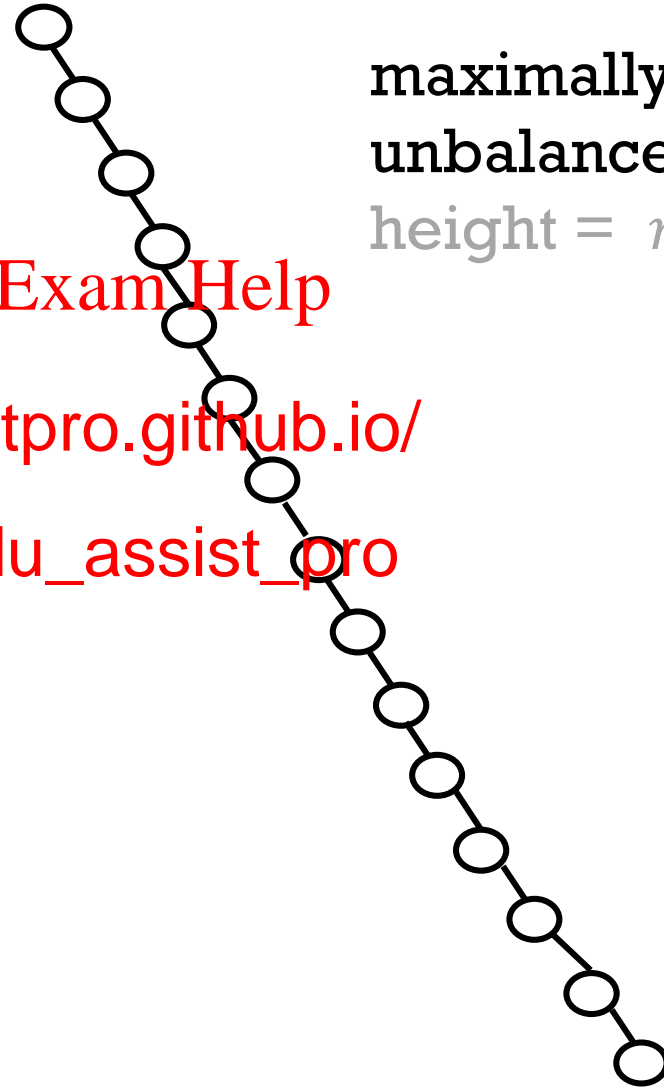


<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

maximally  
unbalanced

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



# Assignment Project Exam Help

BEST VS WORST CASE

best case

worst case

**findMin()**

**findMax()**

**find( key )**

**add(key)**

**remove(key)**

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



# Assignment Project Exam Help

BEST VS WORST CASE

best case

worst case

`findMin()`  $O(1)$   $O(n)$



`findMax()`

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

`find( key )`

`add(key)`

`remove(key)`

# Assignment Project Exam Help

BEST VS WORST CASE

best case

worst case

findMin()  $O(1)$

$O(n)$

findMax()

$O(n)$

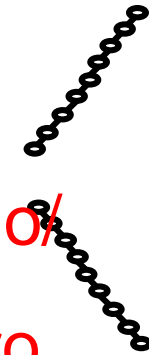
find( key )

add(key)

remove(key)

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



# Assignment Project Exam Help

BEST VS WORST CASE


	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://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# Assignment Project Exam Help

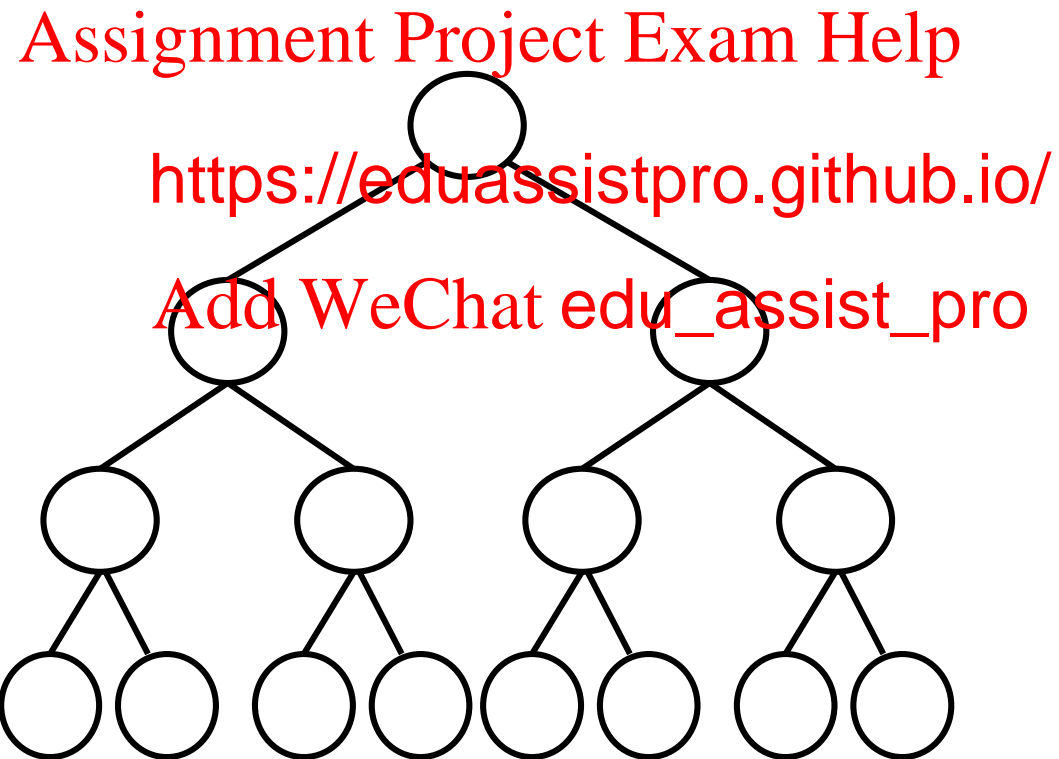
BEST VS WORST CASE

	best case	worst case	
findMin()	$O(1)$	$O(n)$	 <a href="https://eduassistpro.github.io/">https://eduassistpro.github.io/</a> Add WeChat edu_assist_pro
findMax()	$O(1)$	$O(n)$	
find( key )	$O(1)$	$O(n)$	
add(key)	$O(1)$	$O(n)$	
remove(key)	$O(1)$	$O(n)$	
			Could be zigzag

# Assignment Project Exam Help

BINARY SEARCH Tree Chat edu\_assist\_pro

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



# Assignment Project Exam Help

BALANCED BINARY TREES

BALANCED BINARY TREES

(COMP 251: AVL TREES)

(COMP 251: AVL TREES)

Add WeChat edu\_assist\_pro

best case

worst case

findMin()  $O(\log n)$

findMax()  $O(\log n)$

find( key )  $O(1)$

add(key)  $O(\log n)$

remove(key)  $O(\log n)$

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# Assignment Project Exam Help

~~Add WeChat edu\_assist\_pro~~

---

## Assignment Project Exam Help

In the next

■ Heaps

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro