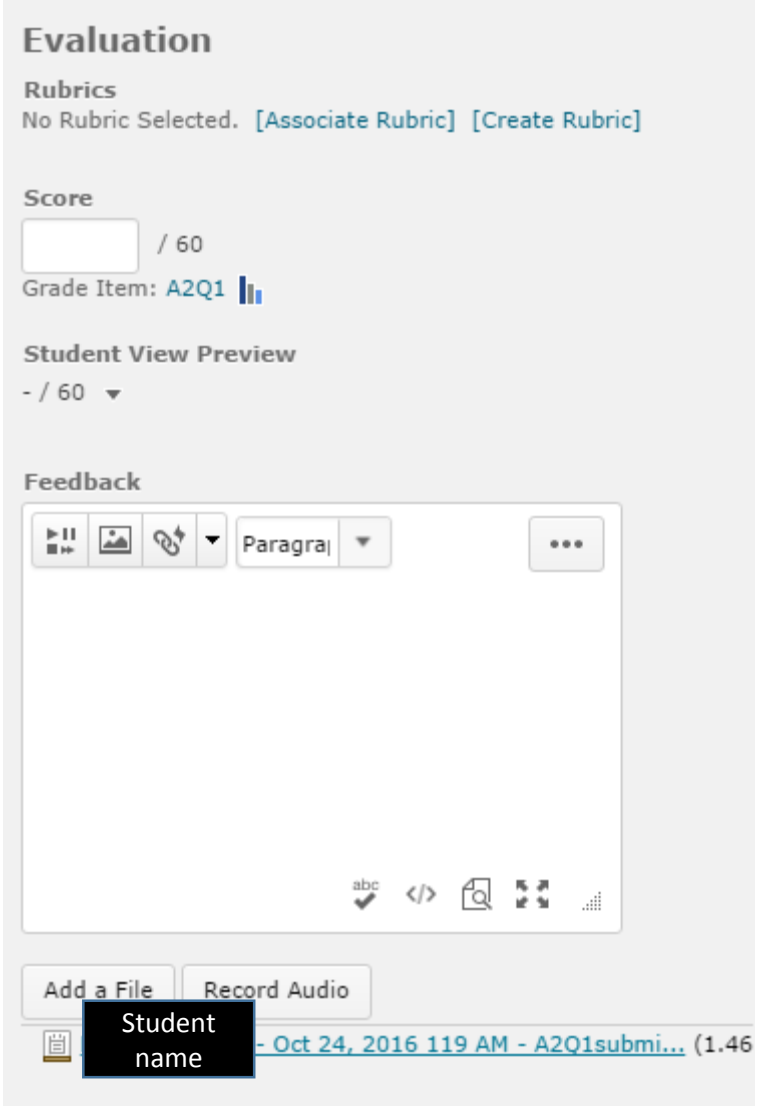
COMP 250

Lecture 20

binary search trees

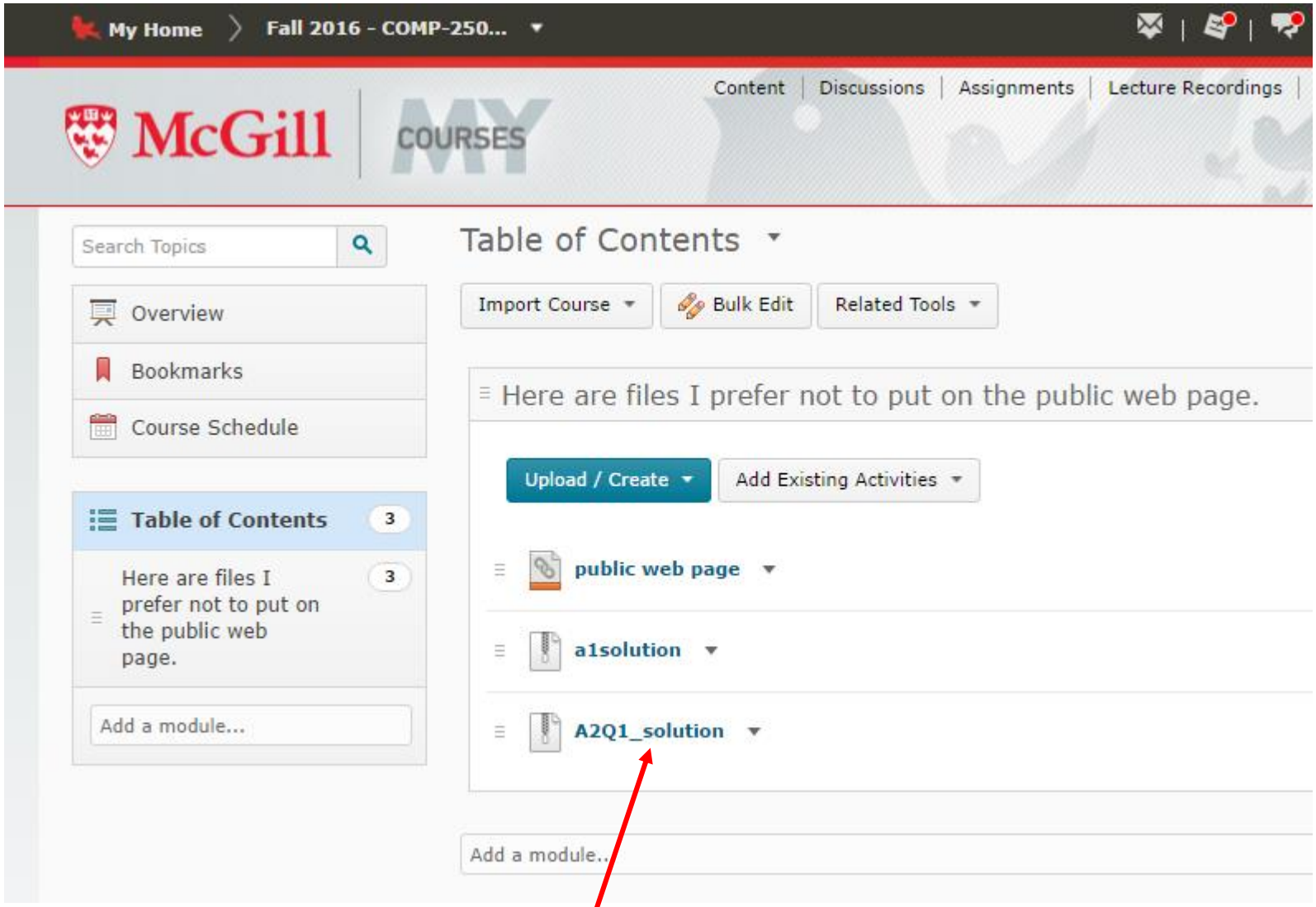
Oct. 26, 2016

Assignment 2 Q1 grading



Please only email Antoine (TA) if necessary and only after you run tester code and you understand my solution.

Comment file



Tester + SOLUTION

~~(binary search) tree~~

binary (search tree)

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

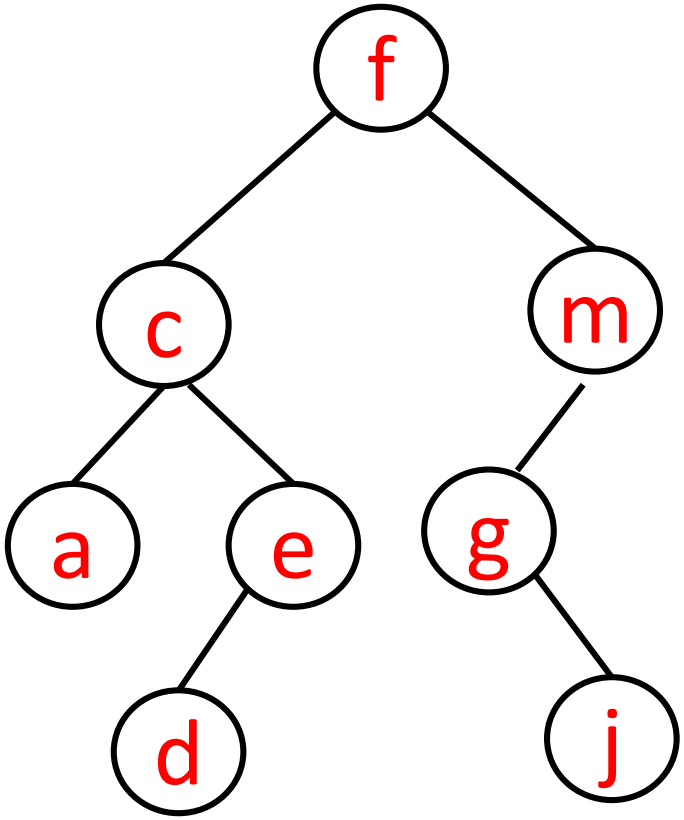
The keys are “comparable”
e.g. numbers, strings.

Binary Search Tree Definition

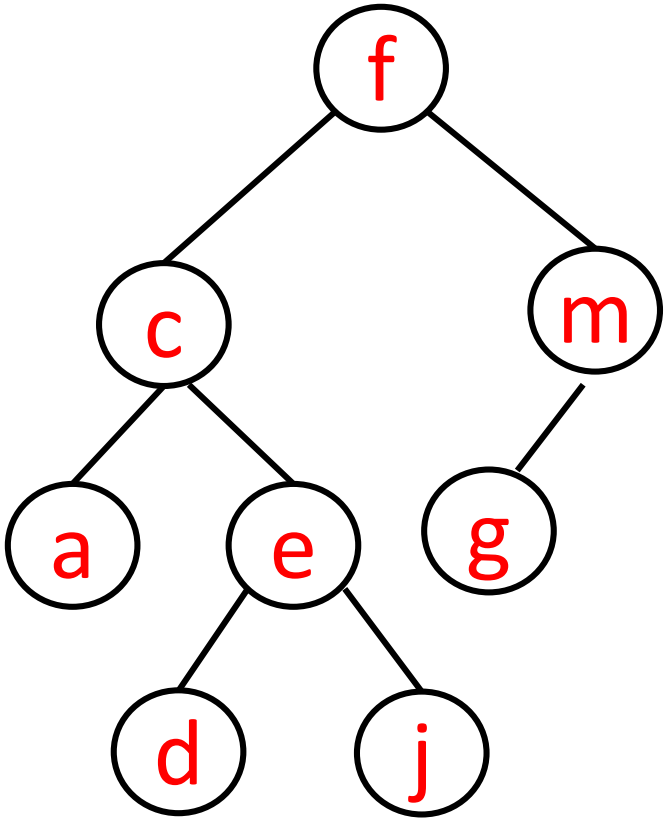
- binary tree
- keys are comparable, unique
- for each node, all descendents in left subtree are less than the node, and all descendents in the node's right subtree are greater than the node

  (comparison is based on node key)

Example

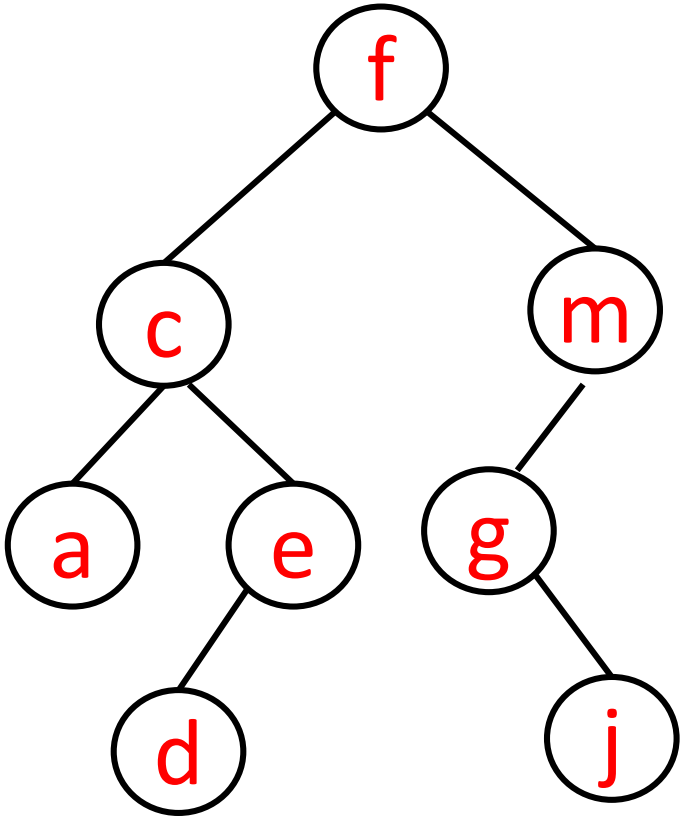


This is not a BST. Why not?



Claim: An in-order traversal on a BST visits the nodes in order.

Proof: Exercise

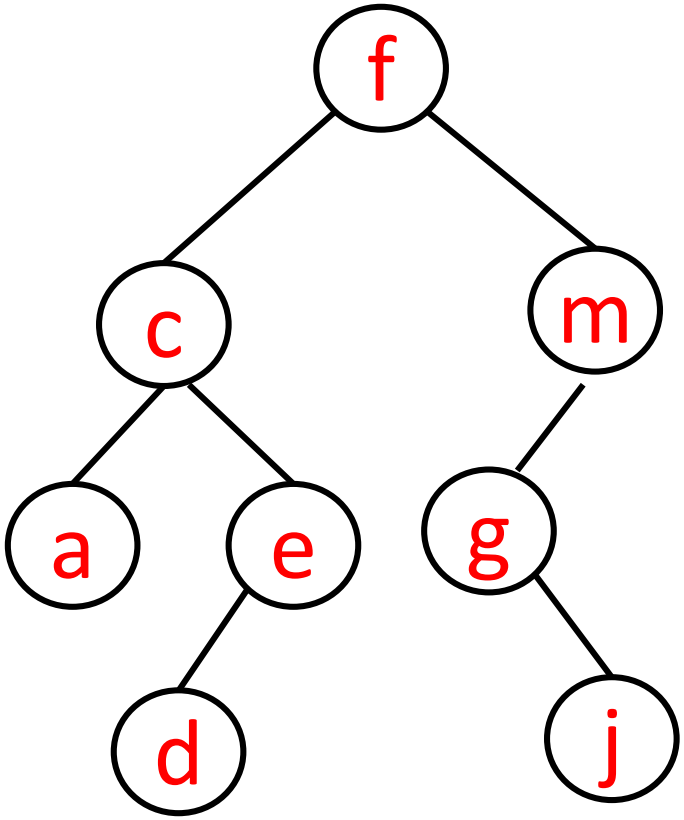


acdefgjm

Binary Search Tree ADT

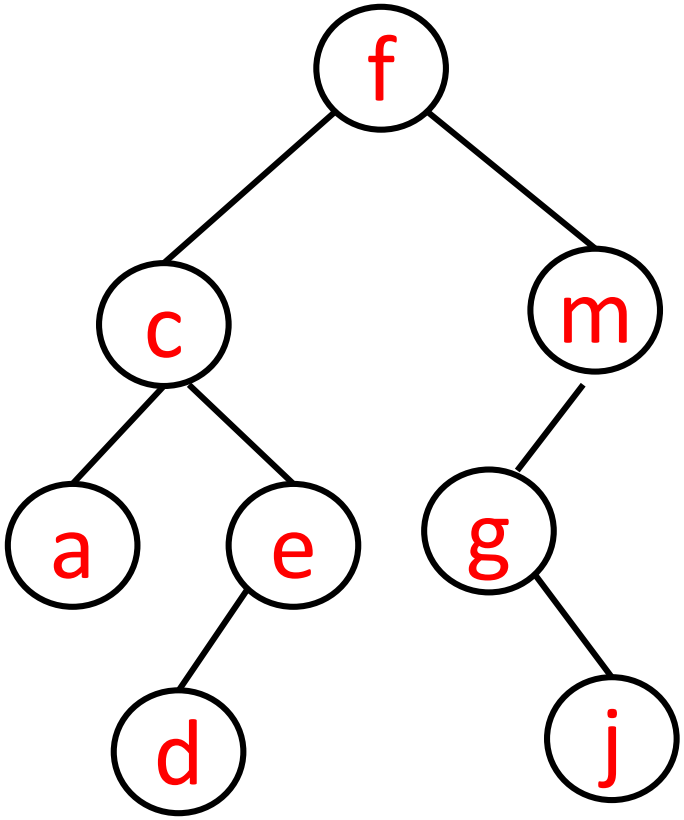
- find(key)
- findMin()
- findMax()
- add(key)
- remove(key)

find(root, g) returns g node
find(root, s) returns null

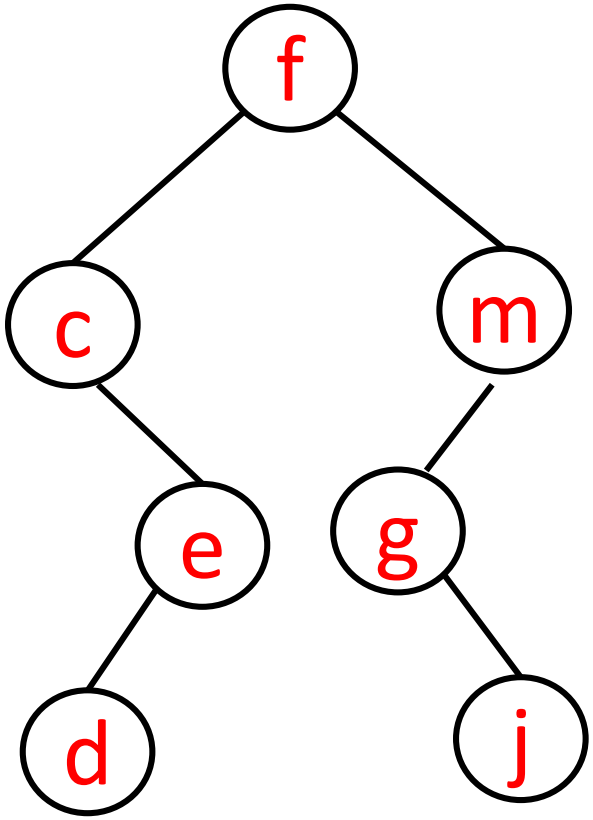


```
find(root,key){                  // returns a node
  if (root == null)
      return null
  else if (root.key == key))
      return root
  else if (key < root.key)
      return  find(root.left, key)
  else
      return  find(root.right, key)
}
```

findMin() returns a node

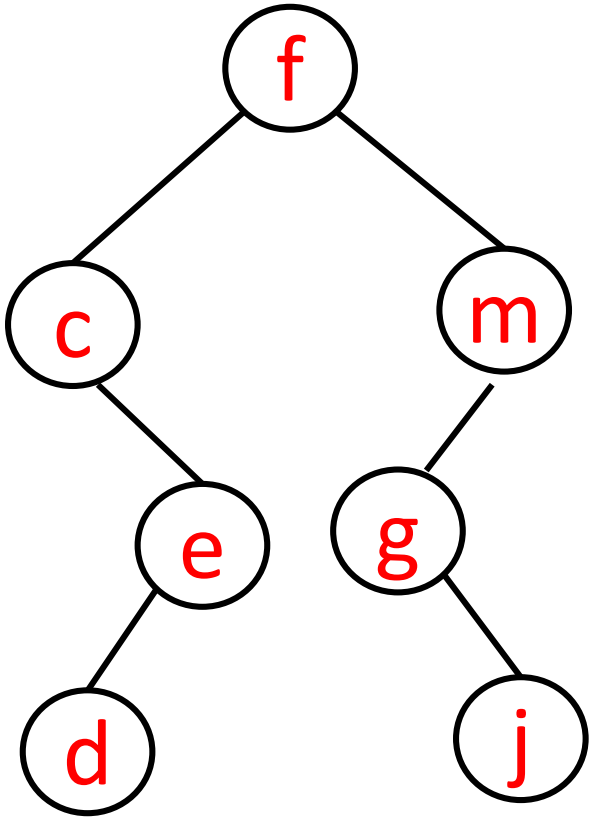


findMin() returns c node

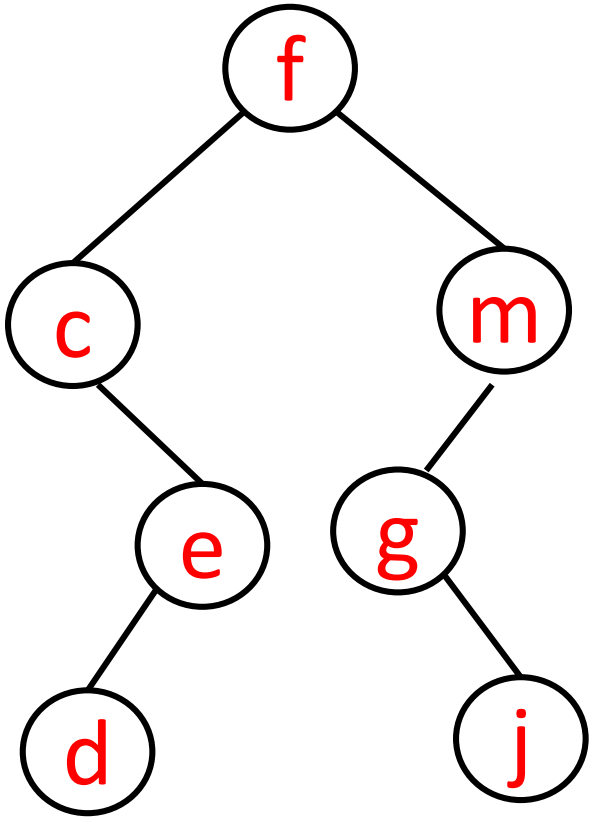


```
findMin(root){
    if (root == null)
        return null
    else if (root.left == null)
        return root
    else
        return findMin( root.left )
}
```

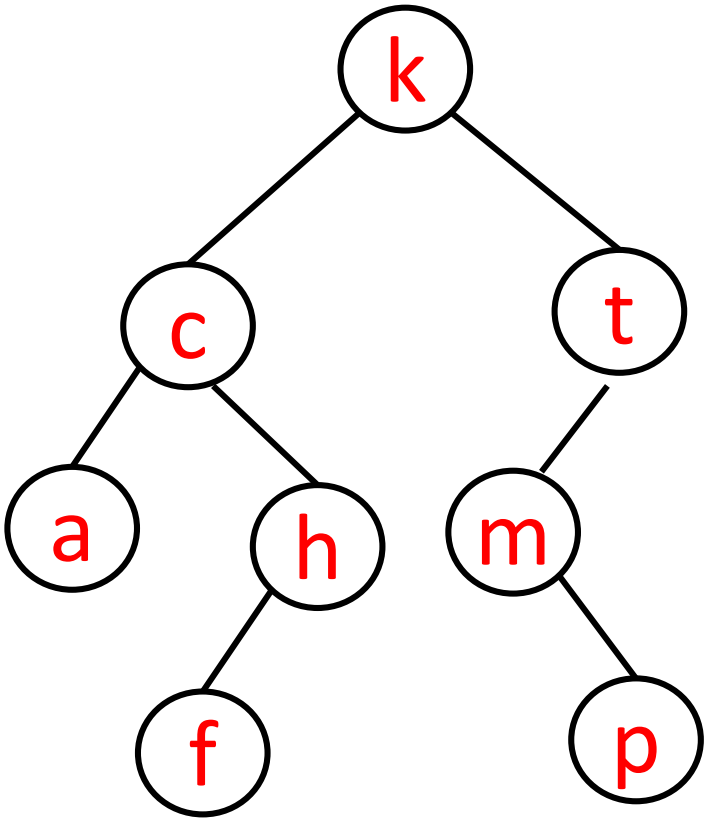
findMax() returns ?



findMax() returns m node



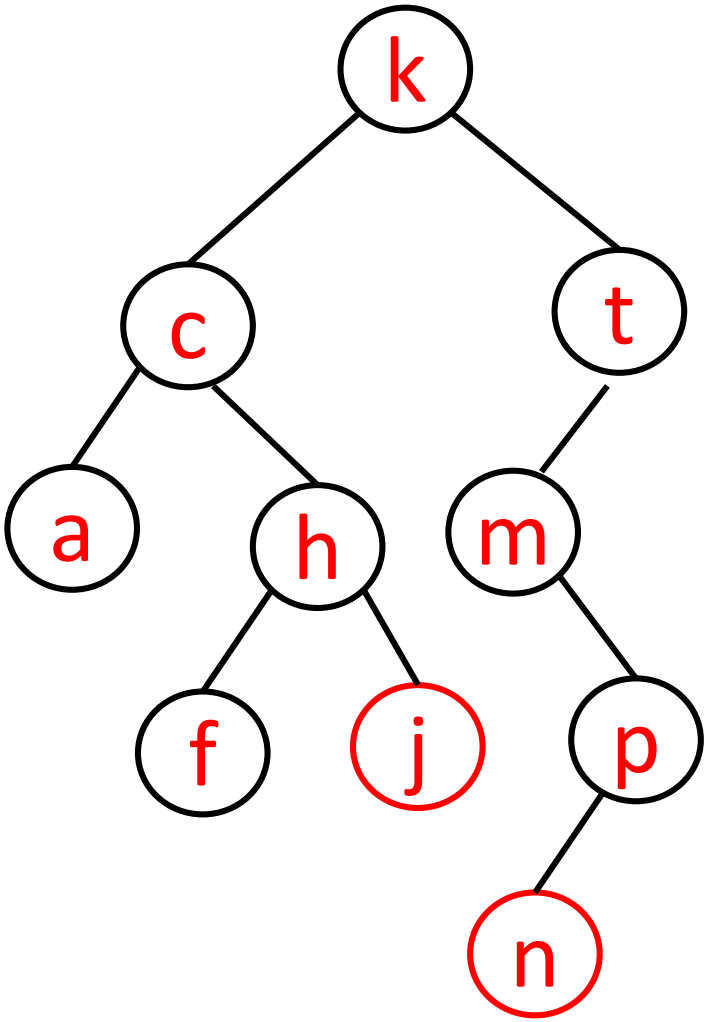
```
findMaximum(root){
   if (root == null)
        return null
   else if (root.right == null))
        return root
   else
        return findMaximum(root.right)
}
```



add(j) ?

add(n) ?

A new key is always a leaf.



add(j) ?

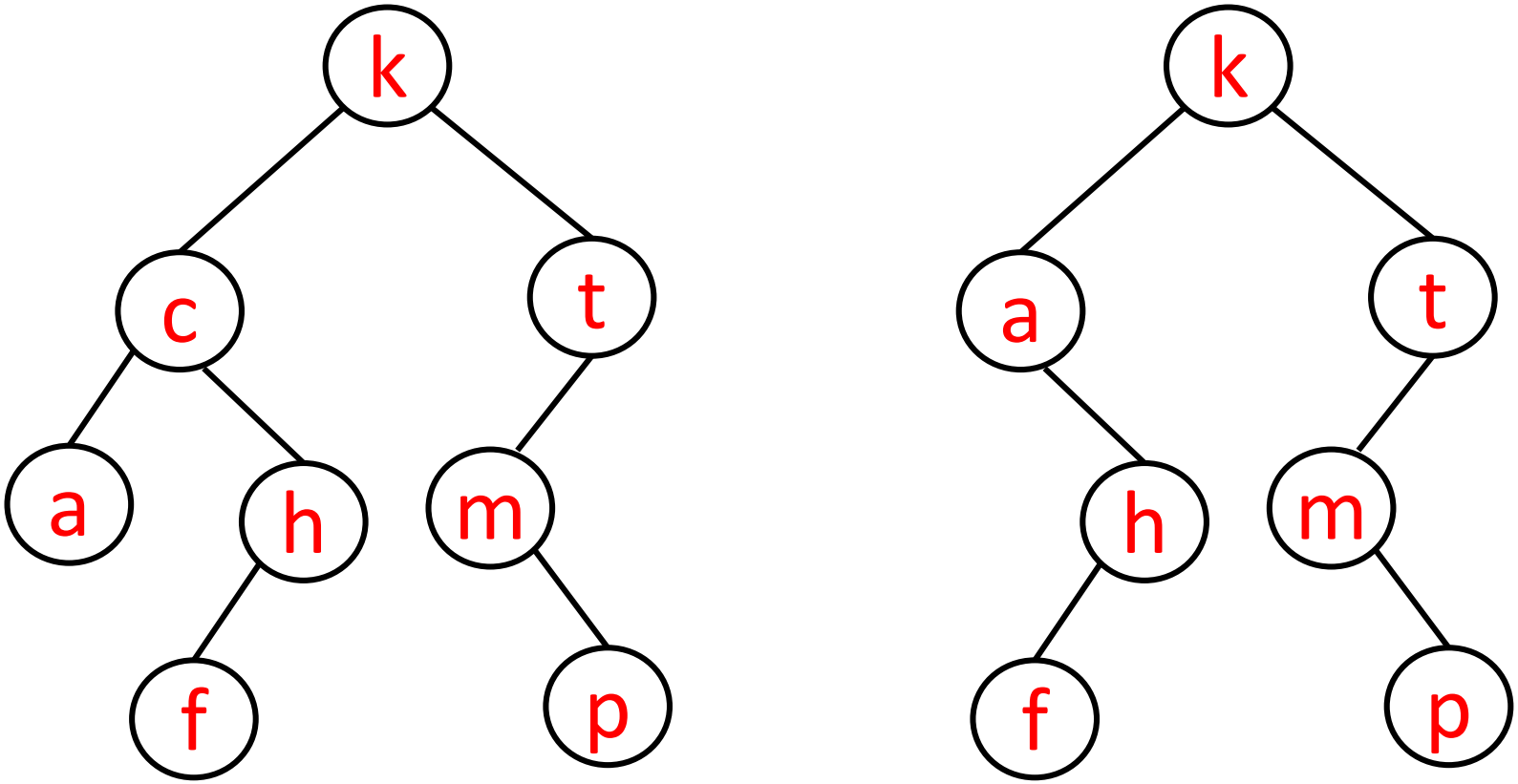
add(n) ?

A new key is always a leaf.

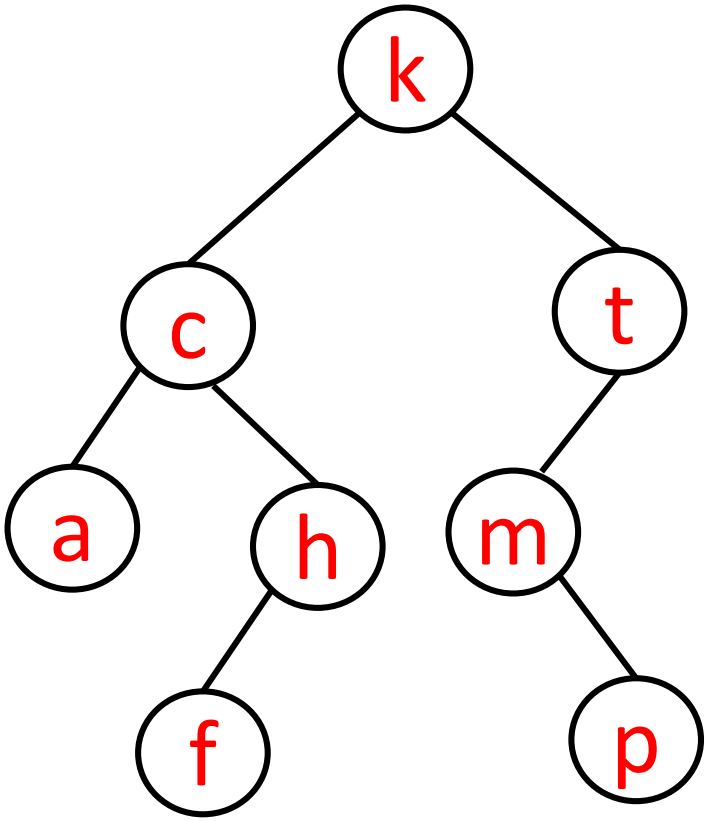
```
add(root, key){
    if (root == null)
            root =  new BSTnode(key)
    else if (key < root.key){
            root.left    =  add(root.left,key)
    else if (key > root.key){
            root.right  =  add(root.right,key)

     return root
 }
```

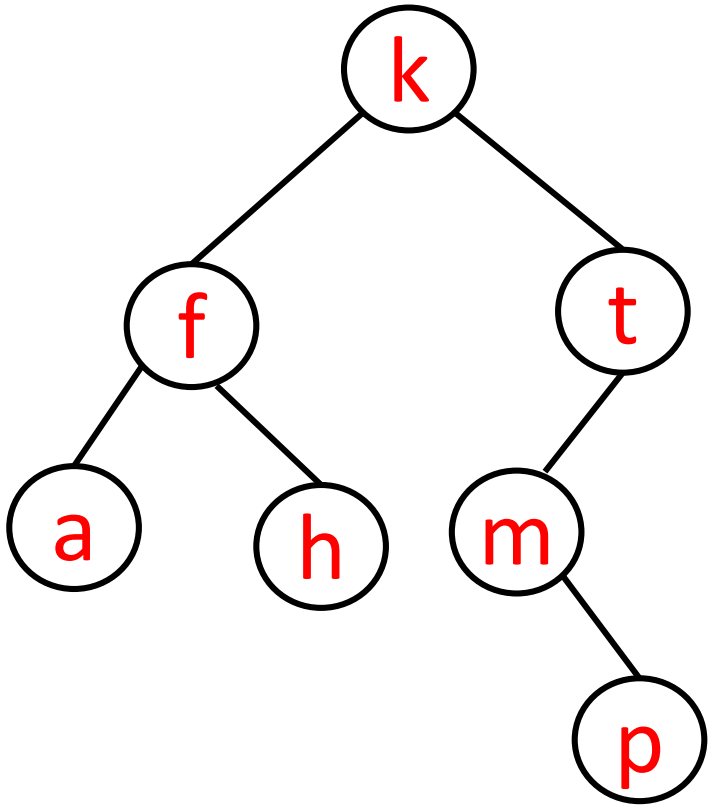
Does this handle base case properly ?
The add calls are a bit weird, no ?
What if root.key == key ?

remove(c)



remove(c)





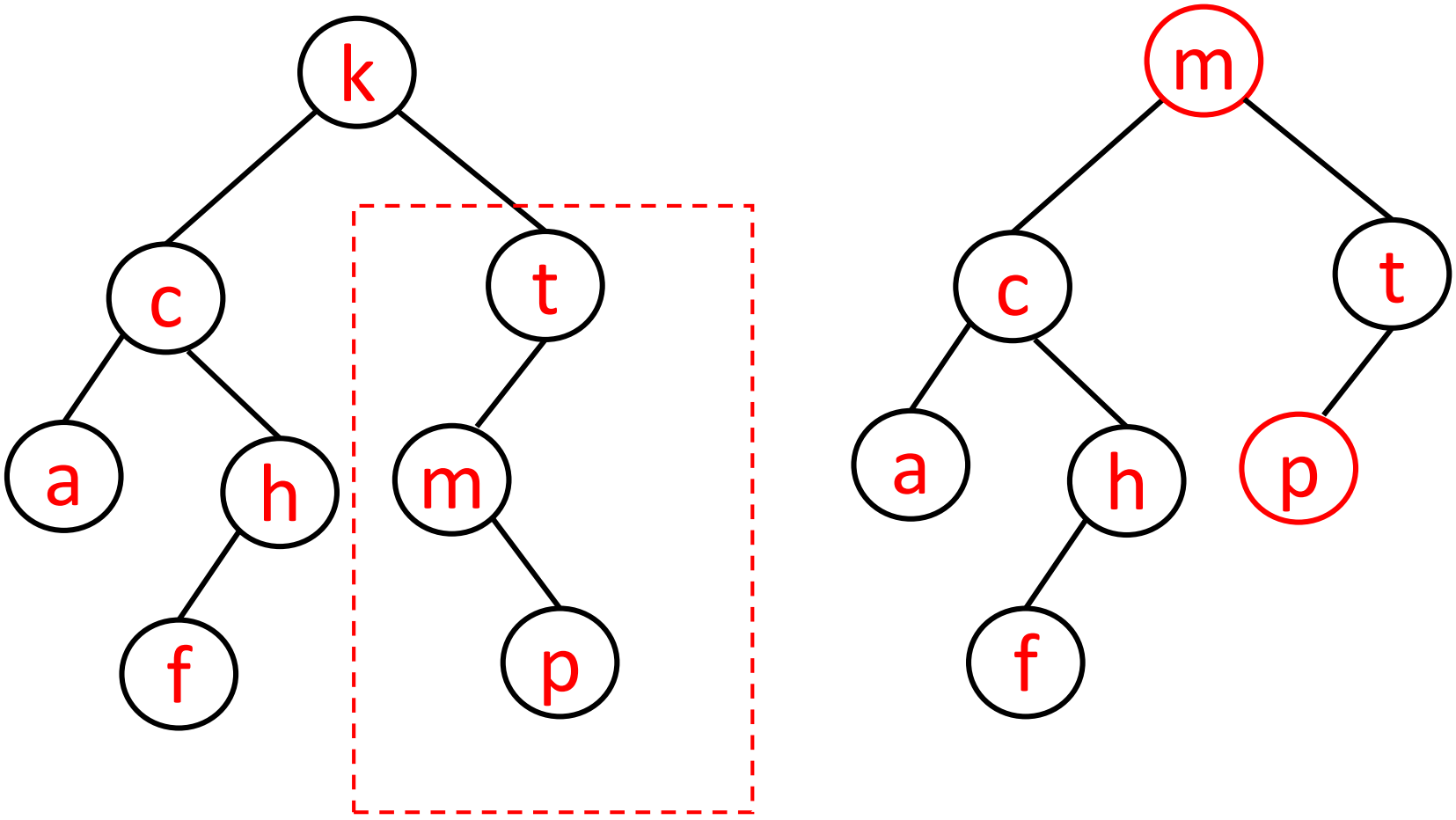
The algorithm I present does it slightly differently (next slide)

```
remove(root, key){
   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;
}
```

```
remove(root, key){
   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;
}
```

```
remove(root, key){
   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;
}
```

remove(k)



Next lecture:

Binary Search Trees vs. Binary Search (with array list)

Best and worst case

Big O, big Omega, big Theta, and “Limits”

Some discussion of Assignment 3