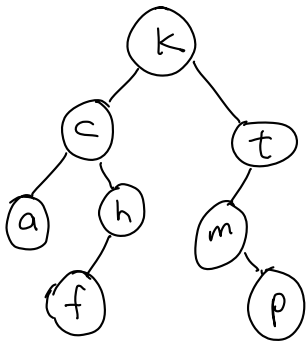


lecture 21
binary search trees (2)

BST ADT

- find (key)
- find Min()
- find Max()
- insert (key)
- remove (key)

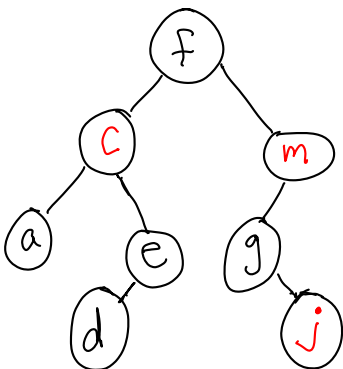


You always insert a leaf (never an internal node)

insert(j)
insert(n)

```
insert (root, key) { // return the root node of modified tree
    if root == null
        root ← new BSTnode(key)
    else {if key < root.key
        root.left ← insert (root.left, key)
        else if key > root.key
            root.right ← insert (root.right, key)
        }
    return root
}
```

remove (key)



remove(j)
remove(m)
remove(c)

```
remove (root, key) { // return the root node of modified tree
    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 if root.left == null
        return root
    else if root.right == null
        return root
    else
        return remove (root.left, key)
    return root
}
```

key does not match
key matches

```

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
  else if root.right == null
  else {
  }
  return root
}

```

key does not match

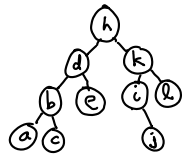
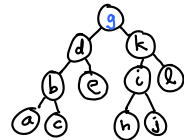
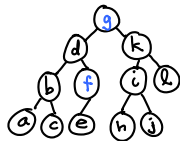
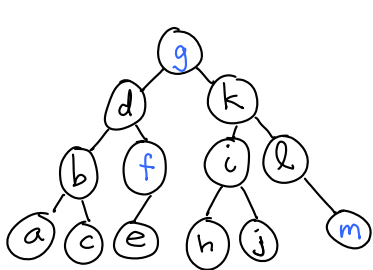
```

remove (root, key) {
  if root == null
  else if key < root.key
  else if key > root.key
  else if root.left == null
    root = root.right
  else if root.right == null
    root = root.left
  else {
    root.key = find min (root.right).key
    root.right =
      remove (root.right, root.key)
  }
  return root
}

```

key matches

Example

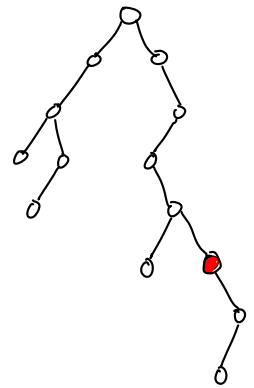


remove (m)
 remove (f)
 remove (g)

Binary search



Binary search tree



Worst Case for BSTs with n nodes

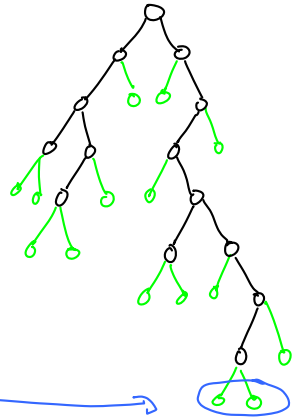
find (key) is $O(n)$
 add (key) is $O(n)$
 remove (key) is $O(n)$

- In COMP 251, you will see data structures and algorithms for more complicated BSTs such that find, add, remove are $O(\log n)$.
 - If you randomly add n elements to a BST, the height of the tree grows (on average) like $\sim c \log n$.
- (The proof is way beyond this course.)

Sketch Only (not on exam)

Why is average height $O(\log n)$?

Think what is necessary for the next insert to increase the height of the tree (probability is $\sim \frac{2}{n}$)



Sketch Only (not on exam)

Let $t(n)$ be the average height of a random binary search tree with n nodes. Then, when n is large,

$$t(n+1) \approx t(n) + \frac{2}{n}$$

$$\approx c + 2 \sum_{i=1}^n \frac{1}{i}$$

is $O(\log n)$

$$\int_a^b \frac{dx}{x} = \ln \frac{b}{a}$$

- A3 should be posted on Friday (more challenging than A1, A2)

- Exercises 6: trees

- Fri. March 11 class in ENGM 304

- Fri. March 18 Quiz 2

- big O tutorial?
(send me email, if interested)