Announcements
Binary Trees
Binary Tree Class
Binary Tree Class

class BTree(Tree):
Binary Tree Class

A binary tree is a tree that has a left branch and a right branch

class BTree(Tree):
A binary tree is a tree that has a left branch and a right branch.

```python
class BTree(Tree):
```

![Binary Tree Diagram]
**Binary Tree Class**

A binary tree is a tree that has a left branch and a right branch. 

**Idea:** Fill the place of a missing left branch with an empty tree

```python
class BTree(Tree):
```

![Binary Tree Diagram]

```plaintext
3
  /    
1  7
   /    
5 9
  /    
11
```
Binary Tree Class

A binary tree is a tree that has a left branch and a right branch. 

**Idea:** Fill the place of a missing left branch with an empty tree.

```python
class BTree(Tree):
```

![Binary Tree Diagram]
Binary Tree Class

A binary tree is a tree that has a left branch and a right branch

Idea: Fill the place of a missing left branch with an empty tree

E: An empty tree

class BTree(Tree):
Binary Tree Class

A binary tree is a tree that has a left branch and a right branch.

**Idea:** Fill the place of a missing left branch with an empty tree.

```python
class BTree(Tree):
    empty = Tree(None)
```

![Binary Tree Diagram]

E: An empty tree
Binary Tree Class

A binary tree is a tree that has a left branch and a right branch.

**Idea:** Fill the place of a missing left branch with an empty tree.

**Idea 2:** An instance of BTree always has exactly two branches.

class BTree(Tree):
    empty = Tree(None)
Binary Tree Class

A binary tree is a tree that has a left branch and a right branch.

**Idea:** Fill the place of a missing left branch with an empty tree.

**Idea 2:** An instance of BTree always has exactly two branches.

```python
class BTree(Tree):
    empty = Tree(None)
```

![Diagram of a binary tree with empty branches](image)
Binary Tree Class

A binary tree is a tree that has a left branch and a right branch.

**Idea:** Fill the place of a missing left branch with an empty tree.

**Idea 2:** An instance of BTree always has exactly two branches.

```python
class BTree(Tree):
    empty = Tree(None)
    def __init__(self, root, left=empty, right=empty):
        Tree.__init__(self, root, [left, right])
```

![Binary Tree Diagram](image)
Binary Tree Class

A binary tree is a tree that has a left branch and a right branch.

**Idea:** Fill the place of a missing left branch with an empty tree.

**Idea 2:** An instance of BTree always has exactly two branches.

class BTree(Tree):
    empty = Tree(None)

    def __init__(self, root, left=empty, right=empty):
        Tree.__init__(self, root, [left, right])

    @property
    def left(self):
        return self.branches[0]
**Binary Tree Class**

A binary tree is a tree that has a left branch and a right branch

**Idea:** Fill the place of a missing left branch with an empty tree

**Idea 2:** An instance of BTree always has exactly two branches

```python
class BTree(Tree):
    empty = Tree(None)

    def __init__(self, root, left=empty, right=empty):
        Tree.__init__(self, root, [left, right])

    @property
def left(self):
        return self.branches[0]

    @property
def right(self):
        return self.branches[1]
```

E: An empty tree

```
3
\ /   \ \\
1    7
   \ /   \\
  E   E
  /\   /\ \\
 E   E 5  9
 / \   /  \\
E   E 11
 / \  /  \\
 E  E  E
```

4
**Binary Tree Class**

A binary tree is a tree that has a left branch and a right branch.

**Idea:** Fill the place of a missing left branch with an empty tree.

**Idea 2:** An instance of BTree always has exactly two branches.

```
class BTree(Tree):
    empty = Tree(None)

    def __init__(self, root, left=empty, right=empty):
        Tree.__init__(self, root, [left, right])

    @property
    def left(self):
        return self.branches[0]

    @property
    def right(self):
        return self.branches[1]

t = BTree(3, BTree(1),
          BTree(7, BTree(5),
               BTree(9, BTree.empty, BTree(11))))
```
Binary Tree Class

A binary tree is a tree that has a left branch and a right branch.

**Idea:** Fill the place of a missing left branch with an empty tree.

**Idea 2:** An instance of BTree always has exactly two branches.

![Diagram of a binary tree with nodes 3, 1, 7, 5, 9, 11, and empty nodes E.]
Binary Search Trees
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half.

20 in [1, 2, 4, 8, 16, 32, 64]
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in \([1, 2, 4, 8, 16, 32, 64]\)

\[\uparrow\]

\([1, 2, 4, 8, 16, 32, 64]\)
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]

[1, 2, 4, 8, 16, 32, 64]
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]

[1, 2, 4, 8, 16, 32, 64]

[1, 2, 4, 8, 16, 32, 64]

[1, 2, 4, 8, 16, 32, 64]
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]

[1, 2, 4, 8, 16, 32, 64]

[1, 2, 4, 8, 16, 32, 64]

False
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]  
4 in [1, 2, 4, 8, 16, 32]  
[1, 2, 4, 8, 16, 32, 64]  
[1, 2, 4, 8, 16, 32, 64]  
[1, 2, 4, 8, 16, 32, 64]  
False
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]  4 in [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64]

[1, 2, 4, 8, 16, 32, 64]

[1, 2, 4, 8, 16, 32, 64]

False
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]  4 in [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64]  [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64]  [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64]  [1, 2, 4, 8, 16, 32]

False
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]  4 in [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64]  [1, 2, 4, 8, 16, 32]

False
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64] 4 in [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64] [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64] [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64] [1, 2, 4, 8, 16, 32, 64]

False
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]  4 in [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64]  [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64]  [1, 2, 4, 8, 16, 32, 64]

False
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64] False

[1, 2, 4, 8, 16, 32, 64]

[1, 2, 4, 8, 16, 32, 64]

[1, 2, 4, 8, 16, 32, 64]

False

4 in [1, 2, 4, 8, 16, 32] True

[1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64]

True
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half.

20 in [1, 2, 4, 8, 16, 32, 64]  True
   [1, 2, 4, 8, 16, 32, 64]  True
      [1, 2, 4, 8, 16, 32]  True
         [1, 2, 4, 8, 16]  True
            [1, 2, 4, 8]  True
               [1, 2, 4]  True

4 in [1, 2, 4, 8, 16, 32]  False
   [1, 2, 4, 8, 16, 32]  False
      [1, 2, 4, 8, 16]  False
         [1, 2, 4, 8]  False
            [1, 2, 4]  False
               [1, 2, 4]  False

For a sorted list of length n, what Theta expression describes the time required?
Binary Search

A strategy for finding a value in a sorted list: check the middle and eliminate half

20 in [1, 2, 4, 8, 16, 32, 64]  4 in [1, 2, 4, 8, 16, 32]

[1, 2, 4, 8, 16, 32, 64]  [1, 2, 4, 8, 16, 32]
[1, 2, 4, 8, 16, 32, 64]  [1, 2, 4, 8, 16, 32, 64]

False  True

For a sorted list of length n, what Theta expression describes the time required? $\Theta(\log n)$
Binary Search Trees
A binary search tree is a binary tree where each root value is:
A binary search tree is a binary tree where each root value is:

- Larger than all entries in its left branch and
Binary Search Trees

A binary search tree is a binary tree where each root value is:
• Larger than all entries in its left branch and
• Smaller than all entries in its right branch
Binary Search Trees

A binary search tree is a binary tree where each root value is:

- Larger than all entries in its left branch and
- Smaller than all entries in its right branch
A binary search tree is a binary tree where each root value is:
- Larger than all entries in its left branch and
- Smaller than all entries in its right branch
Binary Search Trees

A binary search tree is a binary tree where each root value is:
• Larger than all entries in its left branch and
• Smaller than all entries in its right branch
Binary Search Trees

A binary search tree is a binary tree where each root value is:
• Larger than all entries in its left branch and
• Smaller than all entries in its right branch

(Demo)
Discussion Questions

What's the largest element in a binary search tree?
Discussion Questions

What's the largest element in a binary search tree?

```python
def largest(t):
    if _________________________:
        return __________________
    else:
        return __________________
```
Discussion Questions

What's the largest element in a binary search tree?

def largest(t):
    if ________________:
        return _____________
    else:
        return ________________
Discussion Questions

What's the largest element in a binary search tree?

```python
def largest(t):
    if ______________________:
        return ________________
    else:
        return ________________
```

![Binary Search Tree Diagram]

The largest element in the binary search tree is 11.
What's the largest element in a binary search tree?

```python
def largest(t):
    if ____________:
        return ____________
    else:
        return ________________
```

[Binary search tree diagram]
Discussion Questions

What's the largest element in a binary search tree?

def largest(t):
    if t.right is BTree.empty:
        return t.root
    else:
        return largest(t.right)
Discussion Questions

What's the largest element in a binary search tree?

```python
def largest(t):
    if t.right is BTree.empty:
        return t.root
    else:
        return largest(t.right)
```

What's the second largest element in a binary search tree?
Discussion Questions

What's the largest element in a binary search tree?

```python
def largest(t):
    if t.right is BTree.empty:
        return t.root
    else:
        return largest(t.right)
```

What's the second largest element in a binary search tree?

```python
def second(t):
    if t.is_leaf():
        return None
    elif ________________:
        return t.root
    elif ________________:
        return _____________
    else:
        return _____________
```
Discussion Questions

What's the largest element in a binary search tree?

```python
def largest(t):
    if t.right is BTree.empty:
        return ___________
t.root
    else:
        return ___________
largest(t.right)
```

What's the second largest element in a binary search tree?

```python
def second(t):
    if t.is_leaf():
        return None
    elif ____________:
        return t.root
    elif ____________:
        return ___________
largest(t.right)
    else:
        return ___________
```

![Binary Search Tree Diagram](image-url)
Discussion Questions

What's the largest element in a binary search tree?

```python
def largest(t):
    if t.right is BTree.empty:
        return t.root
    else:
        return largest(t.right)
```

What's the second largest element in a binary search tree?

```python
def second(t):
    if t.is_leaf():
        return None
    elif t.root:
        return t.root
    elif t.root:
        return largest(t.right)
    else:
        return largest(t.right)
```
Discussion Questions

What's the largest element in a binary search tree?

```python
def largest(t):
    if t.right is BTree.empty:
        return t.root
    else:
        return largest(t.right)
```

What's the second largest element in a binary search tree?

```python
def second(t):
    if t.is_leaf():
        return None
    elif t.right is BTree.empty:
        return t.root
    elif largest(t.right):
        return ______
    else:
        return ______
```
Discussion Questions

What's the largest element in a binary search tree?

```python
def largest(t):
    if t.right is BTree.empty:
        return t.root
    else:
        return largest(t.right)
```

What's the second largest element in a binary search tree?

```python
def second(t):
    if t.is_leaf():
        return None
    elif t.root:
        return largest(t.right)
    elif t.root:
        return second(t.root)
    else:
        return second(t.left)
```
Discussion Questions

What's the largest element in a binary search tree?

```python
def largest(t):
    if t.right is BTree.empty:
        return ______t.root________
    else:
        return ______largest(t.right)______
```

What's the second largest element in a binary search tree?

```python
def second(t):
    if t.is_leaf():
        return None
    elif ______t.right.is_leaf______:
        return t.root
    elif _________________________:
        return ____________
    else:
        return _______
What's the largest element in a binary search tree?

```python
def largest(t):
    if t.right is BTree.empty:
        return t.root
    else:
        return largest(t.right)
```

What's the second largest element in a binary search tree?

```python
def second(t):
    if t.is_leaf():
        return None
    elif t.right.is_leaf():
        return t.root
    elif t.right is BTree.empty:
        return ...
    else:
        return ...
```
Discussion Questions

What's the largest element in a binary search tree?

```python
def largest(t):
    if t.right is BTree.empty:
        return t.root
    else:
        return largest(t.right)
```

What's the second largest element in a binary search tree?

```python
def second(t):
    if t.is_leaf():
        return None
    elif t.right.is_leaf():
        return t.root
    elif t.right is BTree.empty:
        return largest(t.left)
    else:
        return __________
```
**Discussion Questions**

What's the largest element in a binary search tree?

```python
def largest(t):
    if _______________:
        return ___________
    else:
        return ___________
```

What's the second largest element in a binary search tree?

```python
def second(t):
    if t.is_leaf():
        return None
    elif _______________
        return t.root
    elif _______________
        return ___________
    else:
        return ___________
```

```python
t.right is BTree.empty
largest(t.right)
```

```python
t.right is BTree.empty
largest(t.left)
```

```python
t.right is BTree.empty
largest(t.left)
```

```python
t.left is BTree.empty
largest(t.right)
```
Sets as Binary Search Trees
Membership in Binary Search Trees
Membership in Binary Search Trees

`contains` traverses the tree
Membership in Binary Search Trees

*contains* traverses the tree

- If the element is not the root, it can only be in either the left or right branch
Membership in Binary Search Trees

**contains** traverses the tree

- If the element is not the root, it can only be in either the left or right branch
- By focusing on one branch, we reduce the set by the size of the other branch
Membership in Binary Search Trees

**contains** traverses the tree

* If the element is not the root, it can only be in either the left or right branch
* By focusing on one branch, we reduce the set by the size of the other branch
Membership in Binary Search Trees

`contains` traverses the tree

• If the element is not the root, it can only be in either the left or right branch
• By focusing on one branch, we reduce the set by the size of the other branch
Membership in Binary Search Trees

`contains` traverses the tree

• If the element is not the root, it can only be in either the left or right branch
• By focusing on one branch, we reduce the set by the size of the other branch

```python
def contains(s, v):
```
Membership in Binary Search Trees

`contains` traverses the tree

- If the element is not the root, it can only be in either the left or right branch
- By focusing on one branch, we reduce the set by the size of the other branch

```python
def contains(s, v):
    if s is BTree.empty:
        return False
```

```
3 5
 / \
1 7 9
 /   \
7 11
```
Membership in Binary Search Trees

**contains** traverses the tree

- If the element is not the root, it can only be in either the left or right branch
- By focusing on one branch, we reduce the set by the size of the other branch

```python
def contains(s, v):
    if s is BTree.empty:
        return False
    elif s.root == v:
        return True
```

![Binary Search Tree Diagram](diagram.png)
Membership in Binary Search Trees

`contains` traverses the tree

- If the element is not the root, it can only be in either the left or right branch
- By focusing on one branch, we reduce the set by the size of the other branch

```python
def contains(s, v):
    if s is BTree.empty:
        return False
    elif s.root == v:
        return True
    elif s.root < v:
        return contains(s.right, v)
```
Membership in Binary Search Trees

contains traverses the tree
• If the element is not the root, it can only be in either the left or right branch
• By focusing on one branch, we reduce the set by the size of the other branch

```python
def contains(s, v):
    if s is BTree.empty:
        return False
    elif s.root == v:
        return True
    elif s.root < v:
        return contains(s.right, v)
```

If 9 is in the set, it is in this branch
Membership in Binary Search Trees

`contains` traverses the tree
- If the element is not the root, it can only be in either the left or right branch
- By focusing on one branch, we reduce the set by the size of the other branch

```python
def contains(s, v):
    if s is BTree.empty:
        return False
    elif s.root == v:
        return True
    elif s.root < v:
        return contains(s.right, v)
    elif s.root > v:
        return contains(s.left, v)
```

If 9 is in the set, it is in this branch
Membership in Binary Search Trees

contains traverses the tree
• If the element is not the root, it can only be in either the left or right branch
• By focusing on one branch, we reduce the set by the size of the other branch

```python
def contains(s, v):
    if s is BTree.empty:
        return False
    elif s.root == v:
        return True
    elif s.root < v:
        return contains(s.right, v)
    elif s.root > v:
        return contains(s.left, v)
```

Order of growth?

If 9 is in the set, it is in this branch
Membership in Binary Search Trees

**contains** traverses the tree

* If the element is not the root, it can only be in either the left or right branch
* By focusing on one branch, we reduce the set by the size of the other branch

```python
def contains(s, v):
    if s is BTree.empty:
        return False
    elif s.root == v:
        return True
    elif s.root < v:
        return contains(s.right, v)
    elif s.root > v:
        return contains(s.left, v)
```

Order of growth?  $\Theta(h)$ on average
Membership in Binary Search Trees

**contains** traverses the tree
- If the element is not the root, it can only be in either the left or right branch
- By focusing on one branch, we reduce the set by the size of the other branch

```python
def contains(s, v):
    if s is BTree.empty:
        return False
    elif s.root == v:
        return True
    elif s.root < v:
        return contains(s.right, v)
    elif s.root > v:
        return contains(s.left, v)
```

Order of growth? \( \Theta(h) \) on average \( \Theta(\log n) \) on average for a balanced tree
Adjoining to a Tree Set
Adjoining to a Tree Set

```
    8
   / \  
  5   11
 /   /  
3   9
  / 
1  7
```
Adjoining to a Tree Set

Right!
Adjoining to a Tree Set

Right!
Adjoining to a Tree Set

Right!
Adjoining to a Tree Set

```
5
3 9
1 7 11
```

**Right!**

```
9
7 11

8
```

**Left!**
Adjoining to a Tree Set

Right!  
Left!
Adjoining to a Tree Set

Right!  Left!
Adjoining to a Tree Set

Right!  Left!  Right!
Adjoining to a Tree Set

- Right!
- Left!
- Right!
Adjoining to a Tree Set

```
      8
     /\  
    5  9
   / \ /  
  3  7 9  11
```

```
      8
     /\  
    9  7
   /   /  
  7  11 E  E
```

```
      8
     /\  
    E  E
```

Right!  Left!  Right!  Stop!
Adjoining to a Tree Set

Right!  Left!  Right!  Stop!
Adjoining to a Tree Set

Right!  Left!  Right!  Stop!

8

5
3  9
1  7  11

9
7  11

7
E  E

8
Adjoining to a Tree Set

Right!  Left!  Right!  Stop!

11
Adjoining to a Tree Set

Right!  Left!  Right!  Stop!

1 7 11

3 9

5

1

9

7 11

E

E

8

7

8

8

9

7

8

11
Adjoining to a Tree Set

- **Right!**
- **Left!**
- **Right!**
- **Stop!**

```
8
5
3 9
1 7 11
```
```
8
9
7 11
E E
```
```
8
7
8
```
```
8
5
3 9
1 7 11
8
```
Adjoining to a Tree Set

Right!  Left!  Right!  Stop!

(Demo)