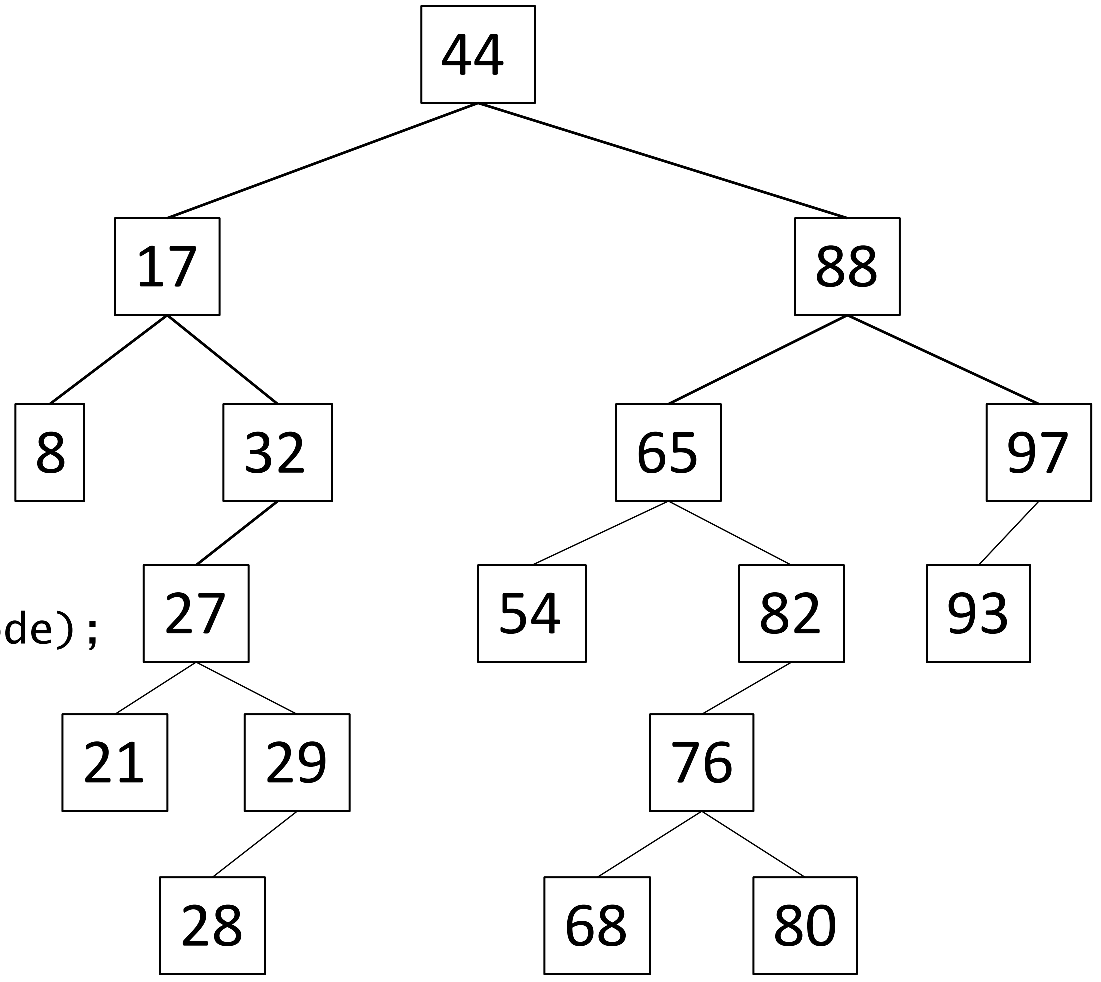
Binary Search Trees
CSCI 232

Binary Search Tree - Insertion

`insert(28);`

```
public void insert(int newValue) {
  if (root == null) {
    root = new Node(newValue);
  } else {
    Node currentNode = root;
    boolean placed = false;
    while (!placed) {
      if (newValue < currentNode.getValue()) {
        if (currentNode.getLeft() != null) {
          currentNode = currentNode.getLeft();
        } else {
          currentNode.setLeft(new Node(newValue));
          currentNode.getLeft().setParent(currentNode);
          placed = true;
        }
      } else {
        if (currentNode.getRight() != null) {
          currentNode = currentNode.getRight();
        } else {
          currentNode.setRight(new Node(newValue));
          currentNode.getRight().setParent(currentNode);
          placed = true;
}
```

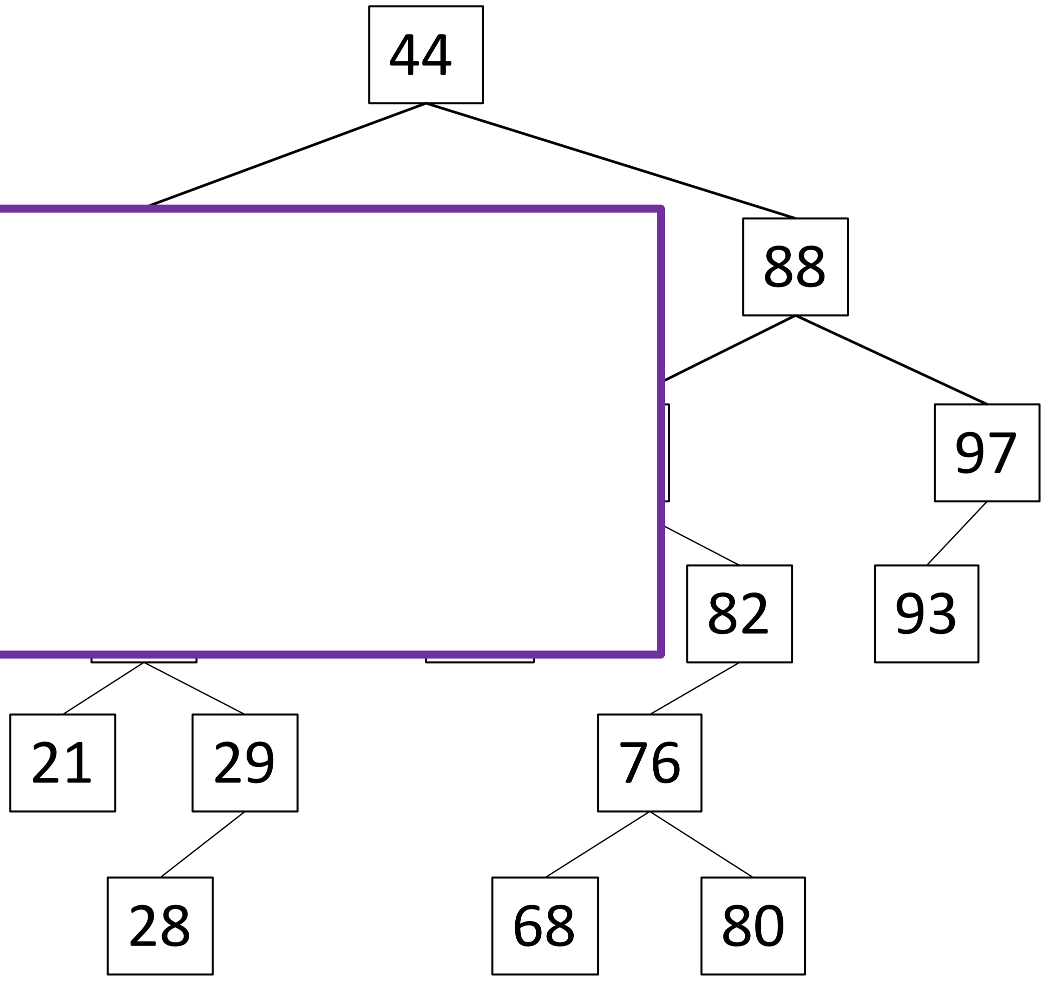


Binary Search Tree - Insertion

insert(28);

```
public void insert(int newValue) {
  if (root == null) {
    root = new Node(newValue);
  } else {
    Node currentNode = r
    boolean placed = fal
    while (!placed) {
      if (newValue < cur
        if (currentNode.
          currentNode = 
        } else {
          currentNode.se
          currentNode.ge
          placed = true;
        }
      } else {
        if (currentNode.getRight() != null) {
          currentNode = currentNode.getRight();
        } else {
          currentNode.setRight(new Node(newValue));
          currentNode.getRight().setParent(currentNode);
          placed = true;
}
```

Running time?



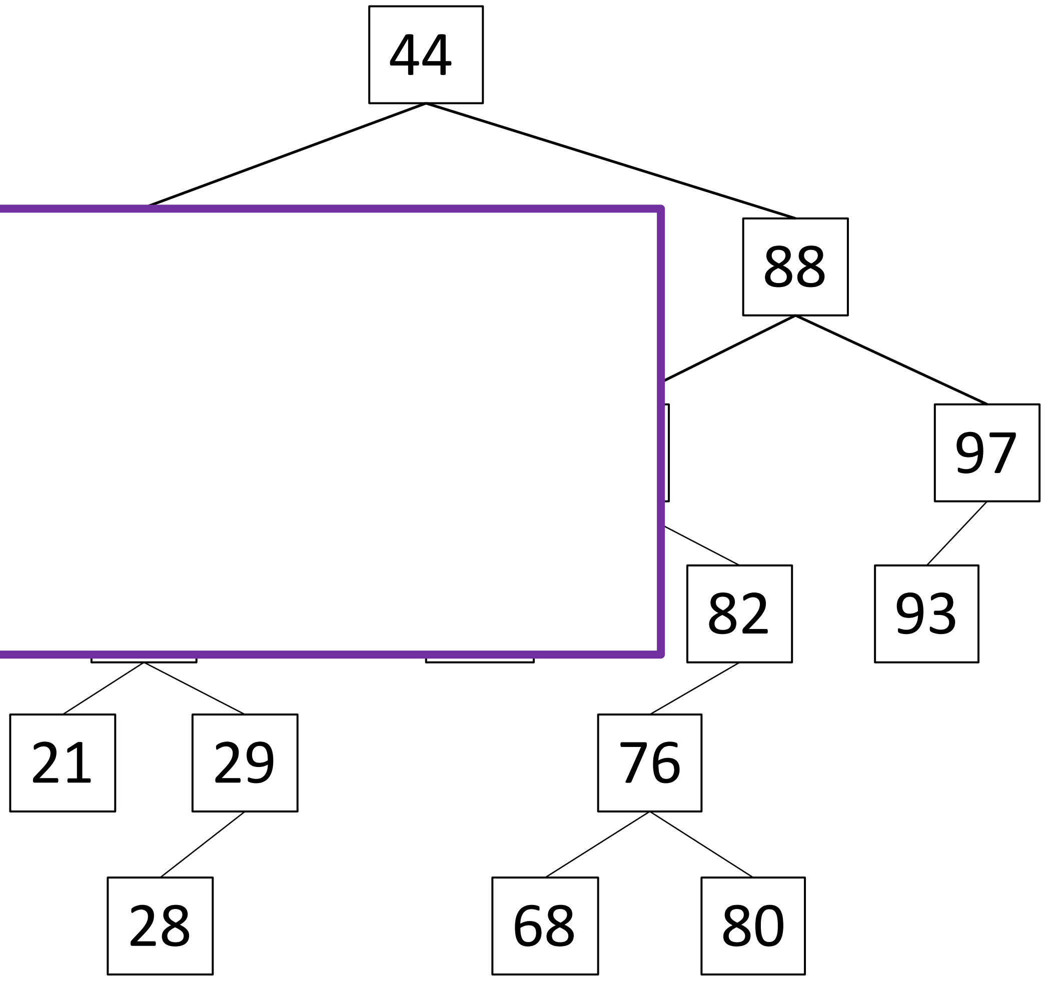
Binary Search Tree - Insertion

insert(28);

```
public void insert(int newValue) {
  if (root == null) {
    root = new Node(newValue);
  } else {
    Node currentNode = r
    boolean placed = fal
    while (!placed) {
      if (newValue < cur
        if (currentNode.
          currentNode =
        } else {
          currentNode.se
          currentNode.ge
          placed = true;
        }
      } else {
        if (currentNode.getRight() != null) {
          currentNode = currentNode.getRight();
        } else {
          currentNode.setRight(new Node(newValue));
          currentNode.getRight().setParent(currentNode);
          placed = true;
}
```

Running time?

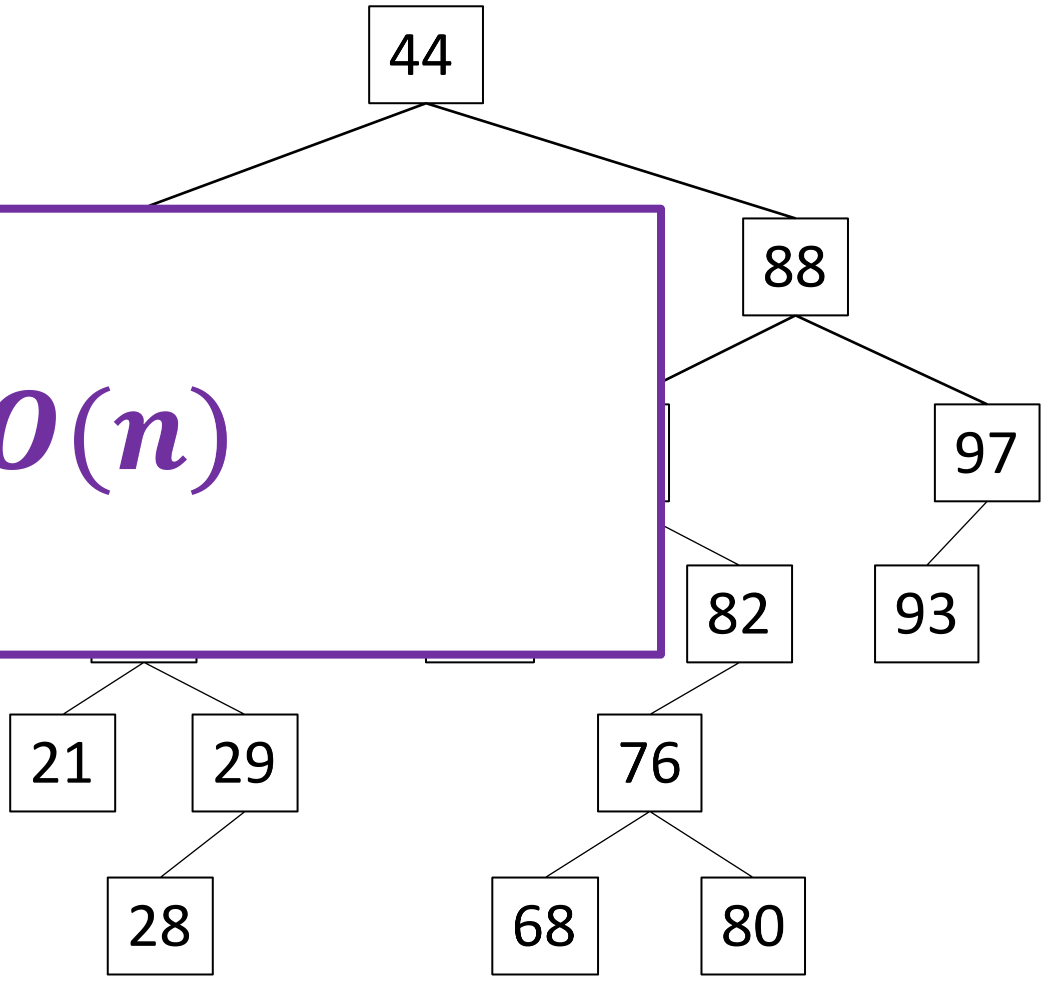
"Bad" tree?



Binary Search Tree - Insertion

insert(28);

```
public void insert(int newValue) {
  if (root == null) {
    root = new Node(newValue);
  } else {
    Node currentNode = r
    boolean placed = fal
    while (!placed) {
      if (newValue < cur
        if (currentNode.
          currentNode =
        } else {
          currentNode.se
          currentNode.ge
          placed = true;
        }
      } else {
        if (currentNode.getRight() != null) {
          currentNode = currentNode.getRight();
        } else {
          currentNode.setRight(new Node(newValue));
          currentNode.getRight().setParent(currentNode);
          placed = true;
}
```



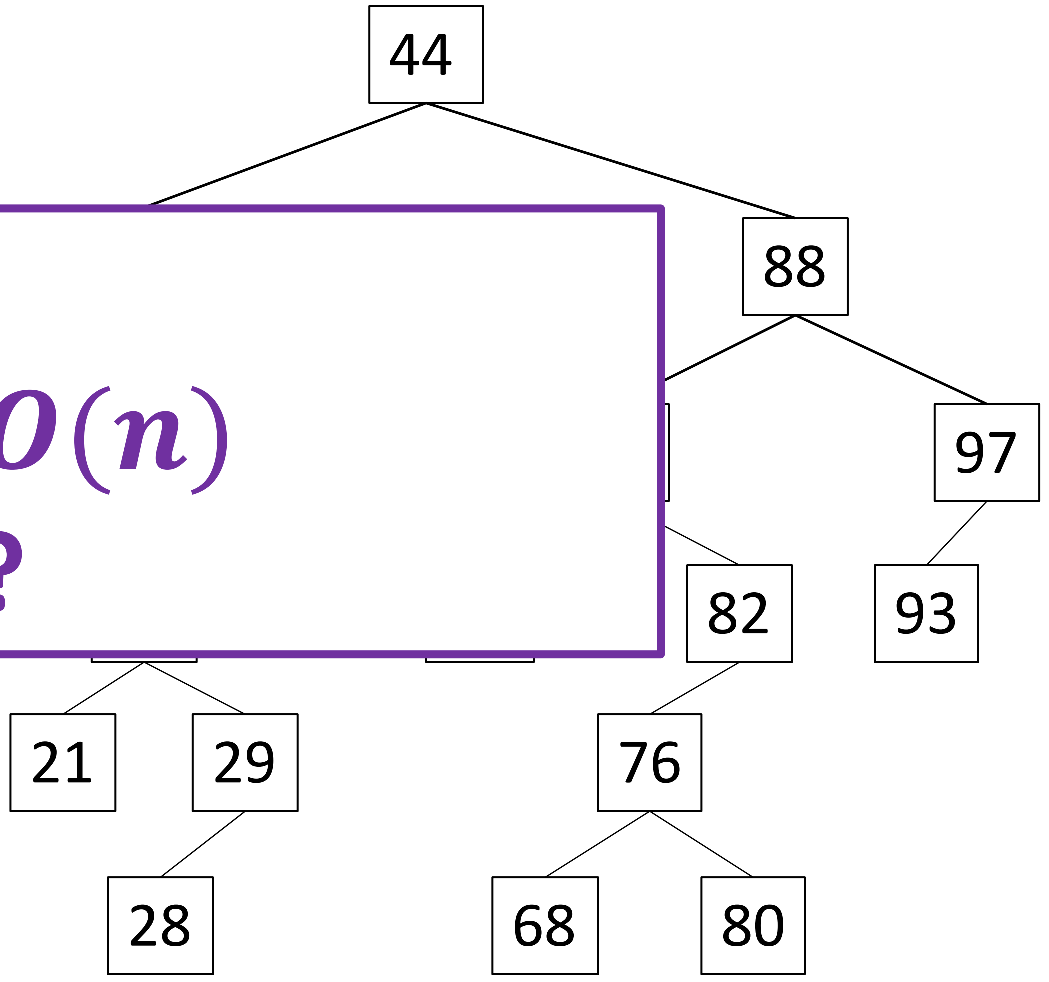
Running time?

"Bad" tree? $O(n)$

Binary Search Tree - Insertion

`insert(28);`

```
public void insert(int newValue) {
  if (root == null) {
    root = new Node(newValue);
  } else {
    Node currentNode = r
    boolean placed = fal
    while (!placed) {
      if (newValue < cur
        if (currentNode.
          currentNode =
        } else {
          currentNode.se
          currentNode.ge
          placed = true;
        }
      } else {
        if (currentNode.getRight() != null) {
          currentNode = currentNode.getRight();
        } else {
          currentNode.setRight(new Node(newValue));
          currentNode.getRight().setParent(currentNode);
          placed = true;
}
```



Running time?

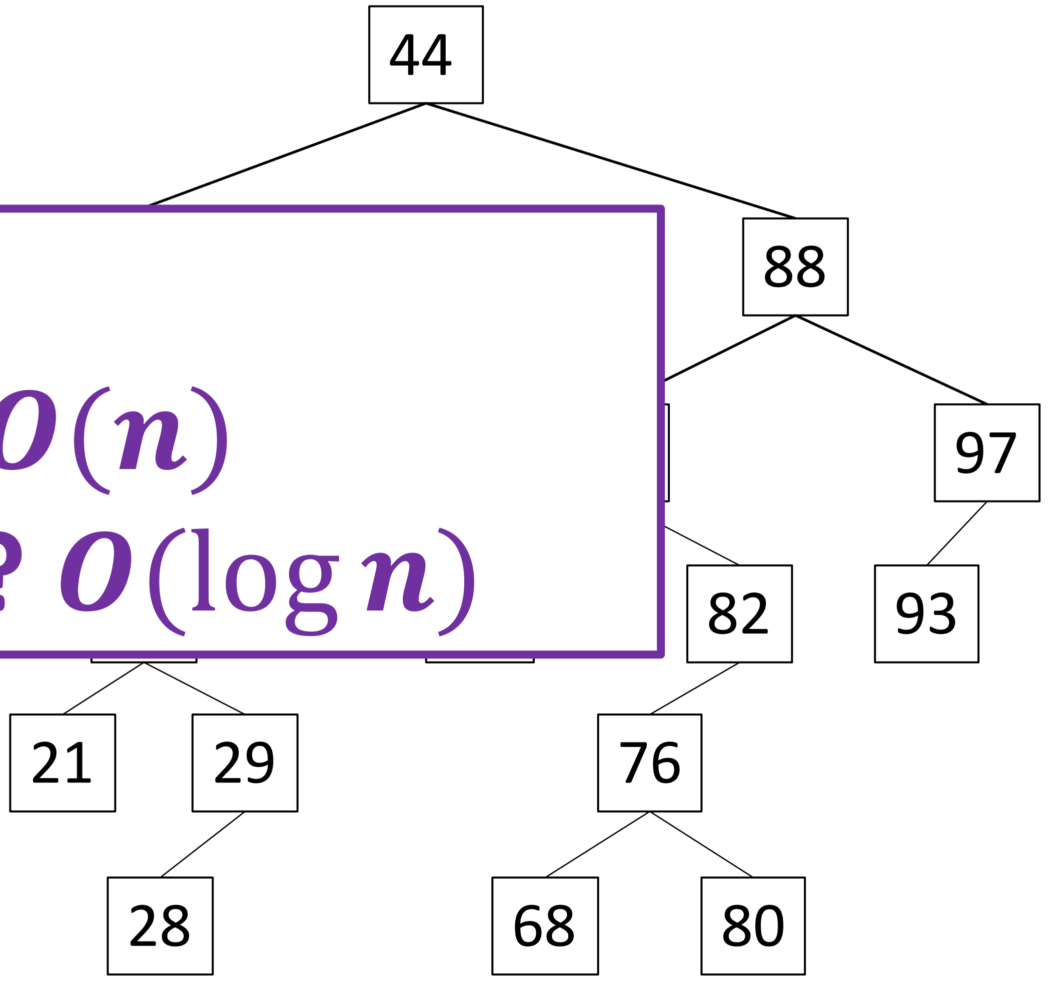
"Bad" tree? $O(n)$

"Good" tree?

Binary Search Tree - Insertion

`insert(28);`

```
public void insert(int newValue) {
  if (root == null) {
    root = new Node(newValue);
  } else {
    Node currentNode = r
    boolean placed = fal
    while (!placed) {
      if (newValue < cur
        if (currentNode.
          currentNode =
        } else {
          currentNode.se
          currentNode.ge
          placed = true;
        }
      } else {
        if (currentNode.getRight() != null) {
          currentNode = currentNode.getRight();
        } else {
          currentNode.setRight(new Node(newValue));
          currentNode.getRight().setParent(currentNode);
          placed = true;
}
```



Running time?

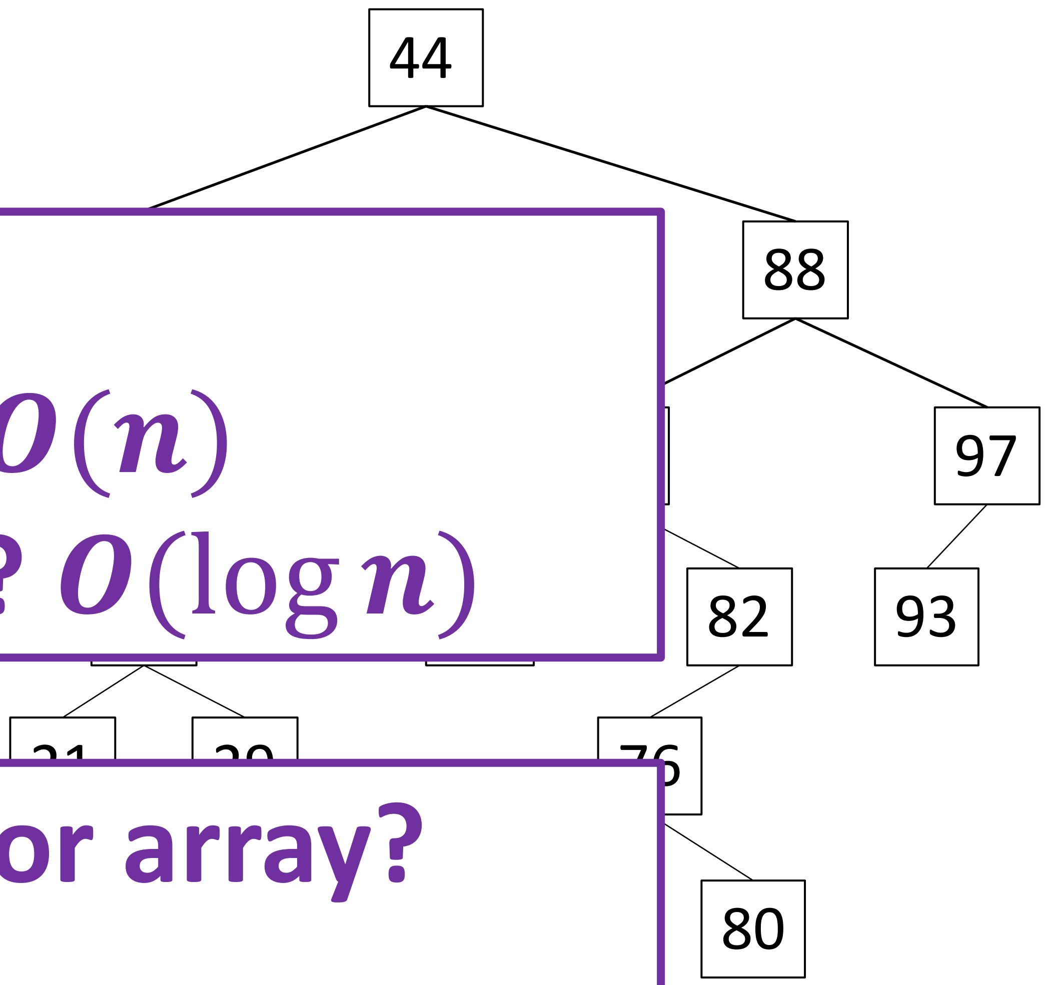
“Bad” tree? $O(n)$

“Good” tree? $O(\log n)$

Binary Search Tree - Insertion

`insert(28);`

```
public void insert(int newValue) {
  if (root == null) {
    root = new Node(newValue);
  } else {
    Node currentNode = r
    boolean placed = fal
    while (!placed) {
      if (newValue < cur
        if (currentNode.
          currentNode =
        } else {
          currentNode.se
          currentNode.ge
          placed = true;
        }
      } else {
        if (currentNode.
          currentNode =
        } else {
          currentNode.se
          currentNode.ge
          placed = true;
}
```



Running time?

“Bad” tree? $O(n)$

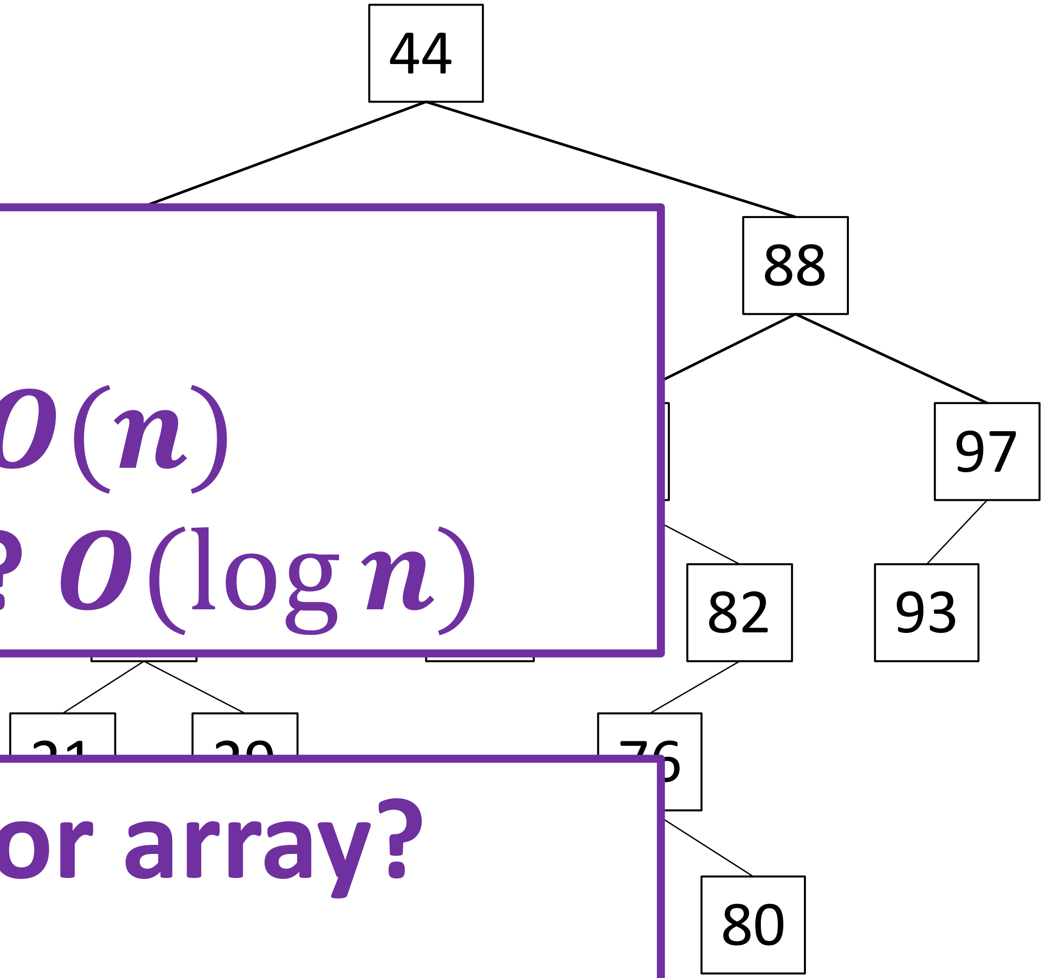
“Good” tree? $O(\log n)$

Running time for array?

Binary Search Tree - Insertion

`insert(28);`

```
public void insert(int newValue) {
  if (root == null) {
    root = new Node(newValue);
  } else {
    Node currentNode = r
    boolean placed = fal
    while (!placed) {
      if (newValue < cur
        if (currentNode.
          currentNode =
        } else {
          currentNode.se
          currentNode.ge
          placed = true;
        }
      } else {
        if (currentNode.
          currentNode =
        } else {
          currentNode.se
          currentNode.ge
          placed = true;
}
```



Running time?

"Bad" tree? $O(n)$

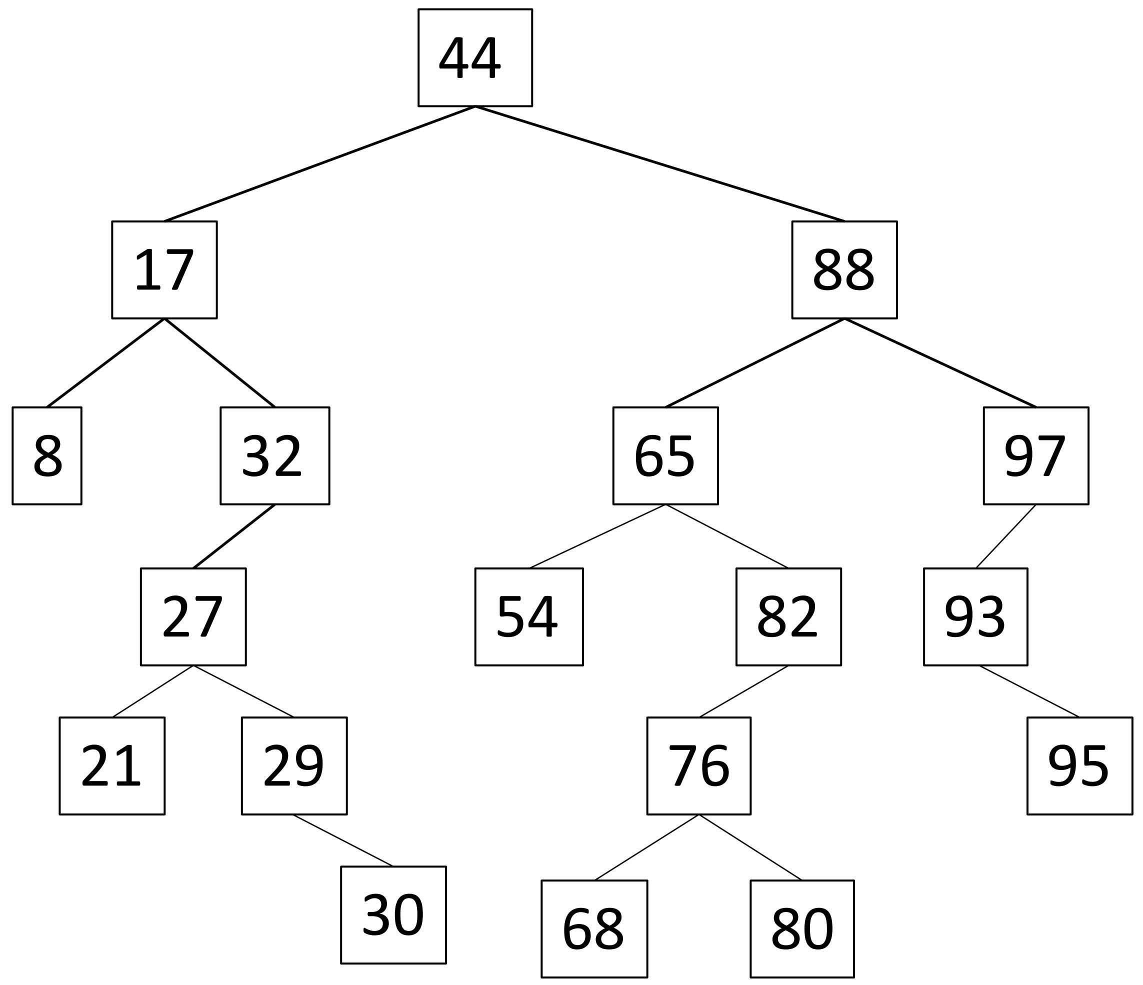
"Good" tree? $O(\log n)$

Running time for array?

$O(n)$

Binary Search Tree - Removal

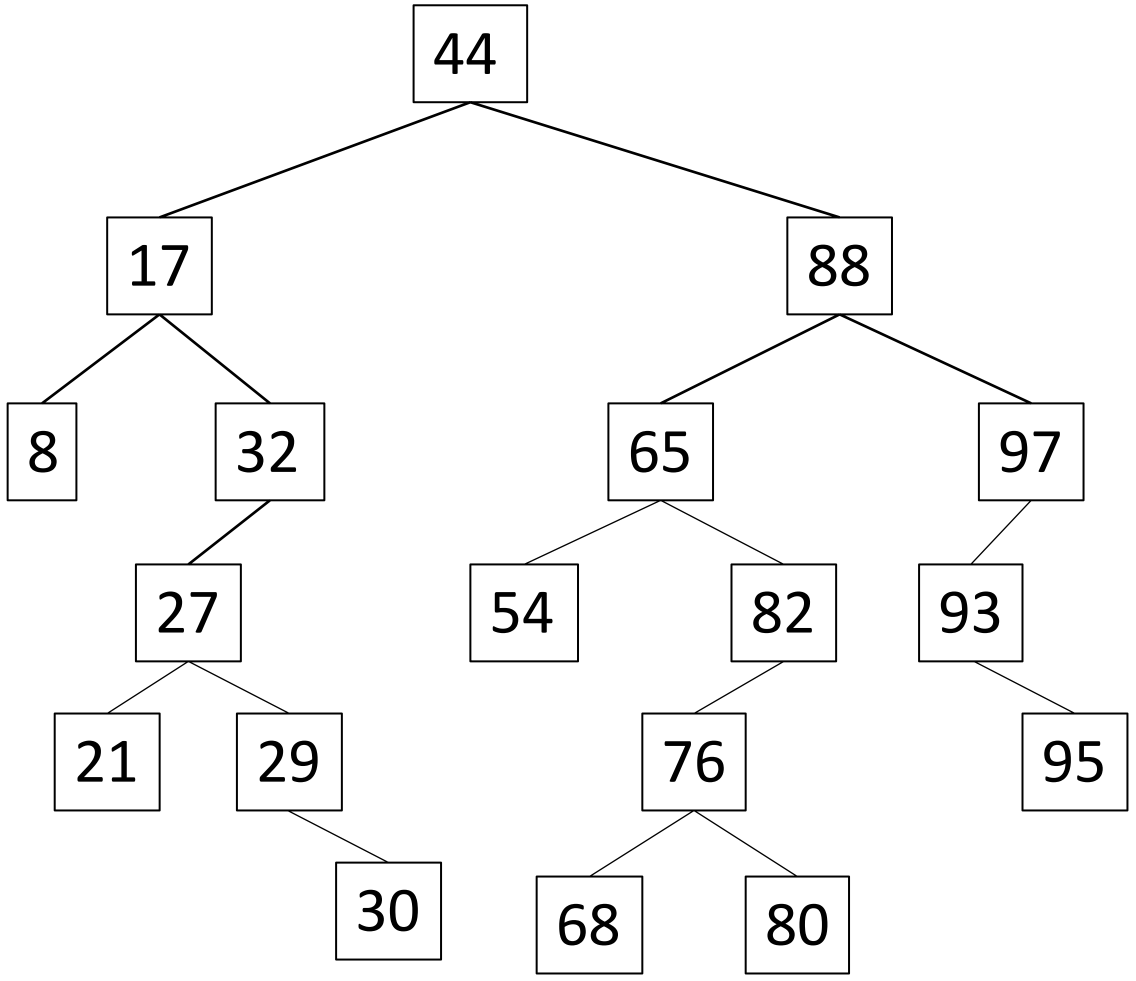
```
remove(68);
```



Binary Search Tree - Removal

```
remove(68);
```

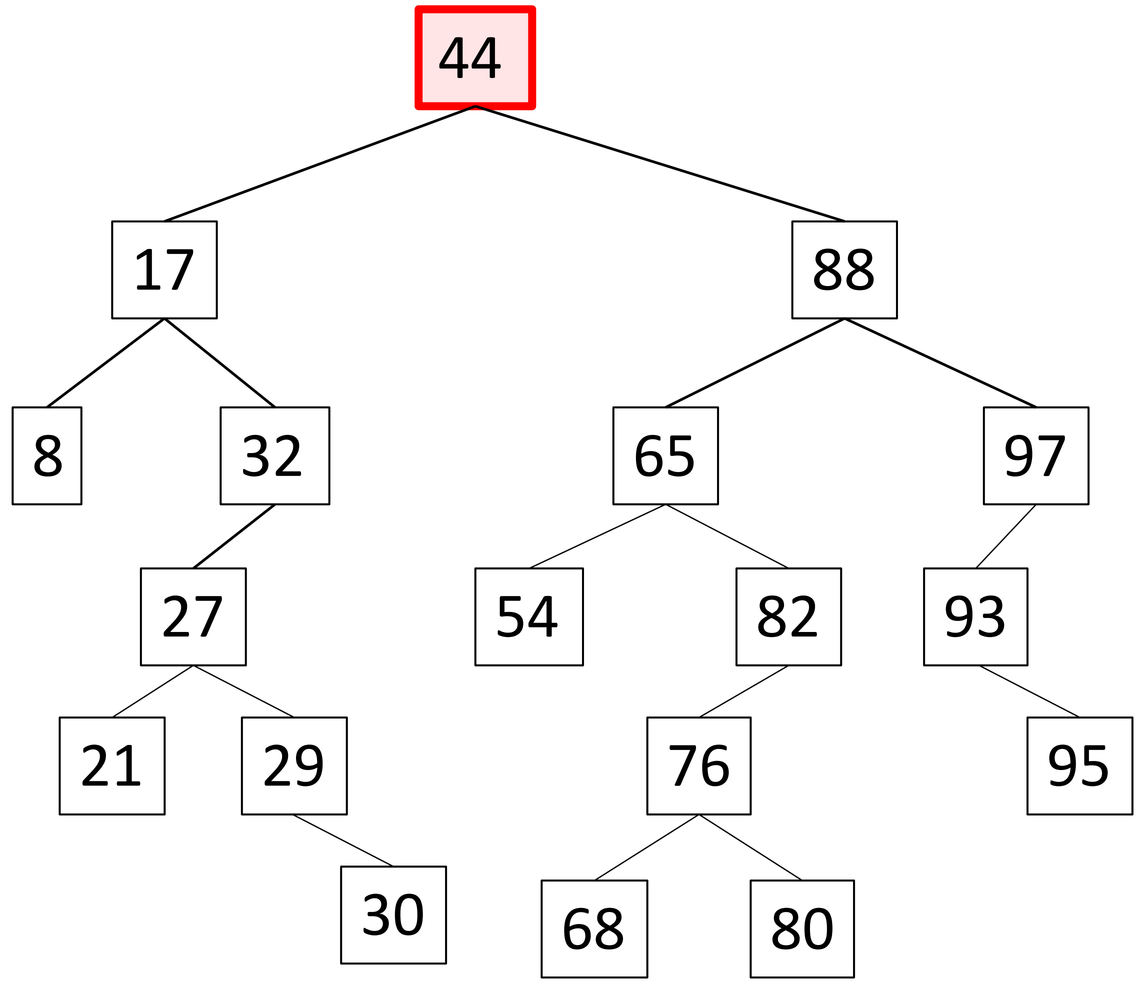
Step 1: Find the Node in the tree.



Binary Search Tree - Removal

```
remove(68);
```

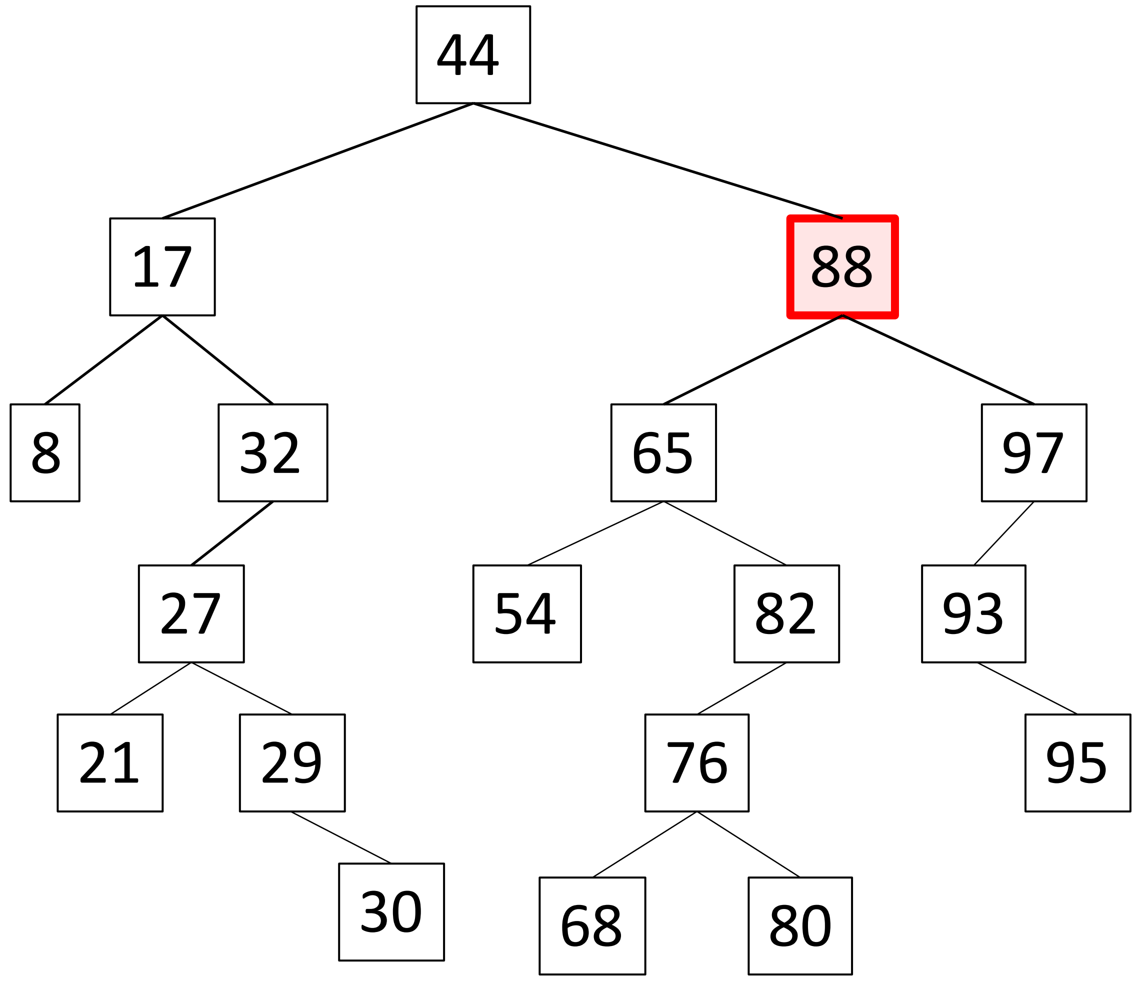
Step 1: Find the Node in the tree.



Binary Search Tree - Removal

`remove(68);`

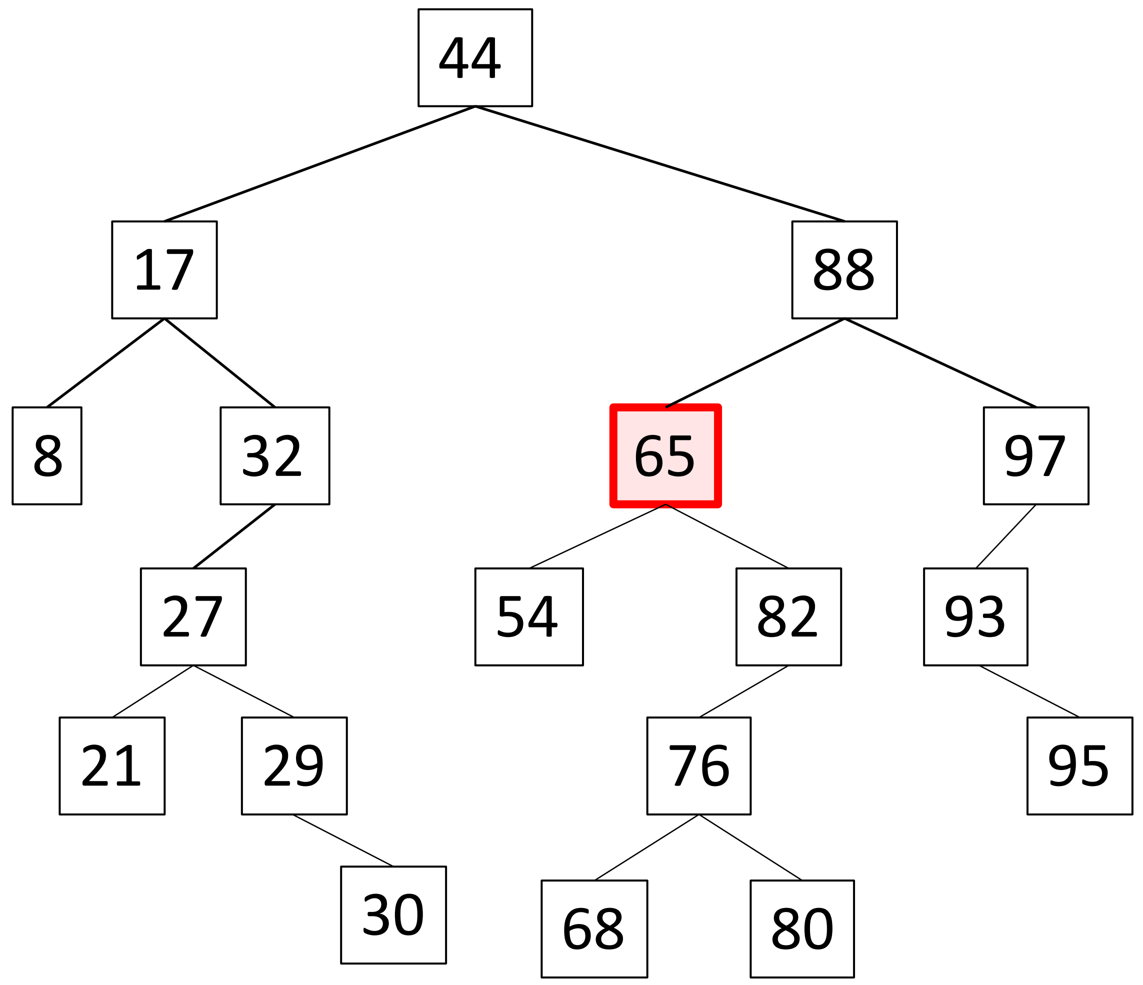
Step 1: Find the Node in the tree.



Binary Search Tree - Removal

```
remove(68);
```

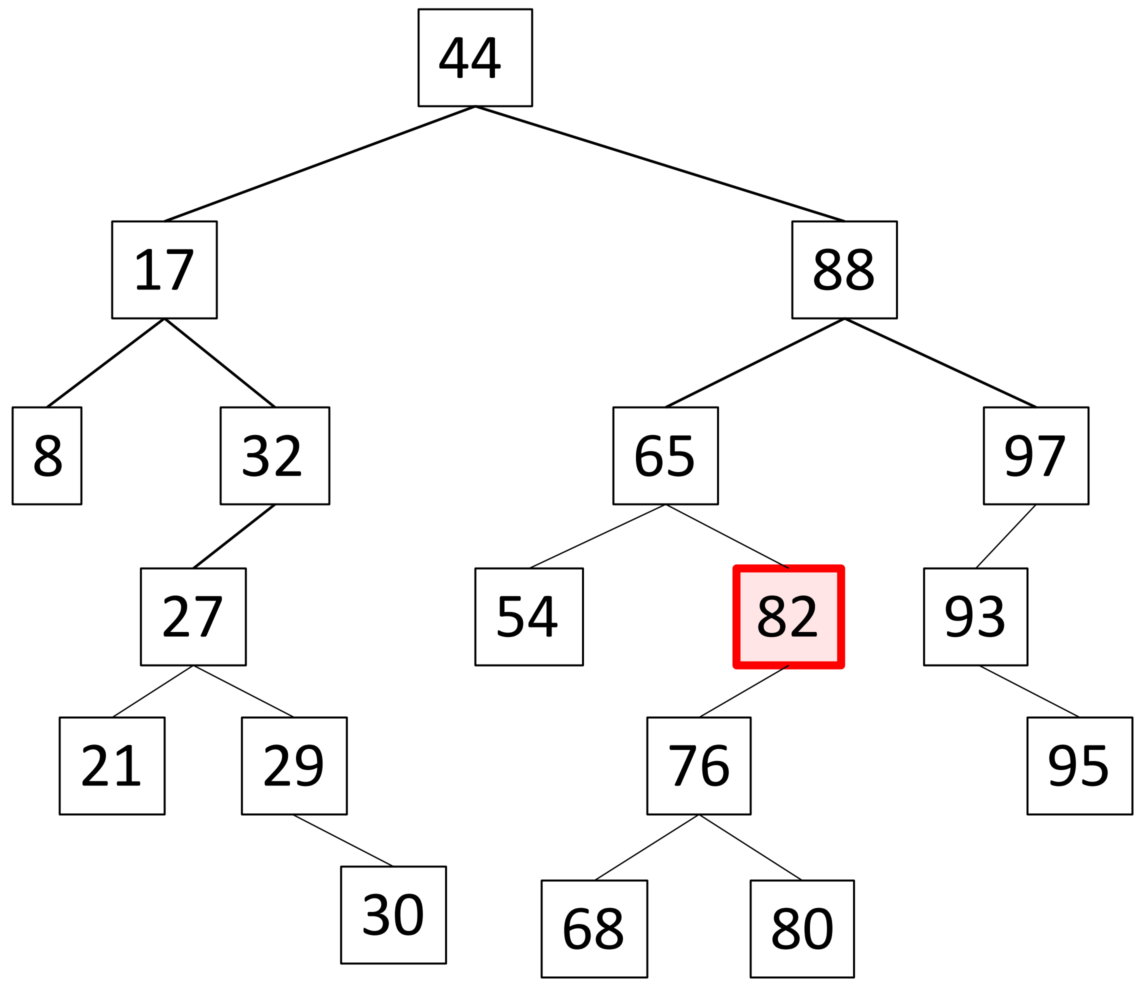
Step 1: Find the Node in the tree.



Binary Search Tree - Removal

`remove(68);`

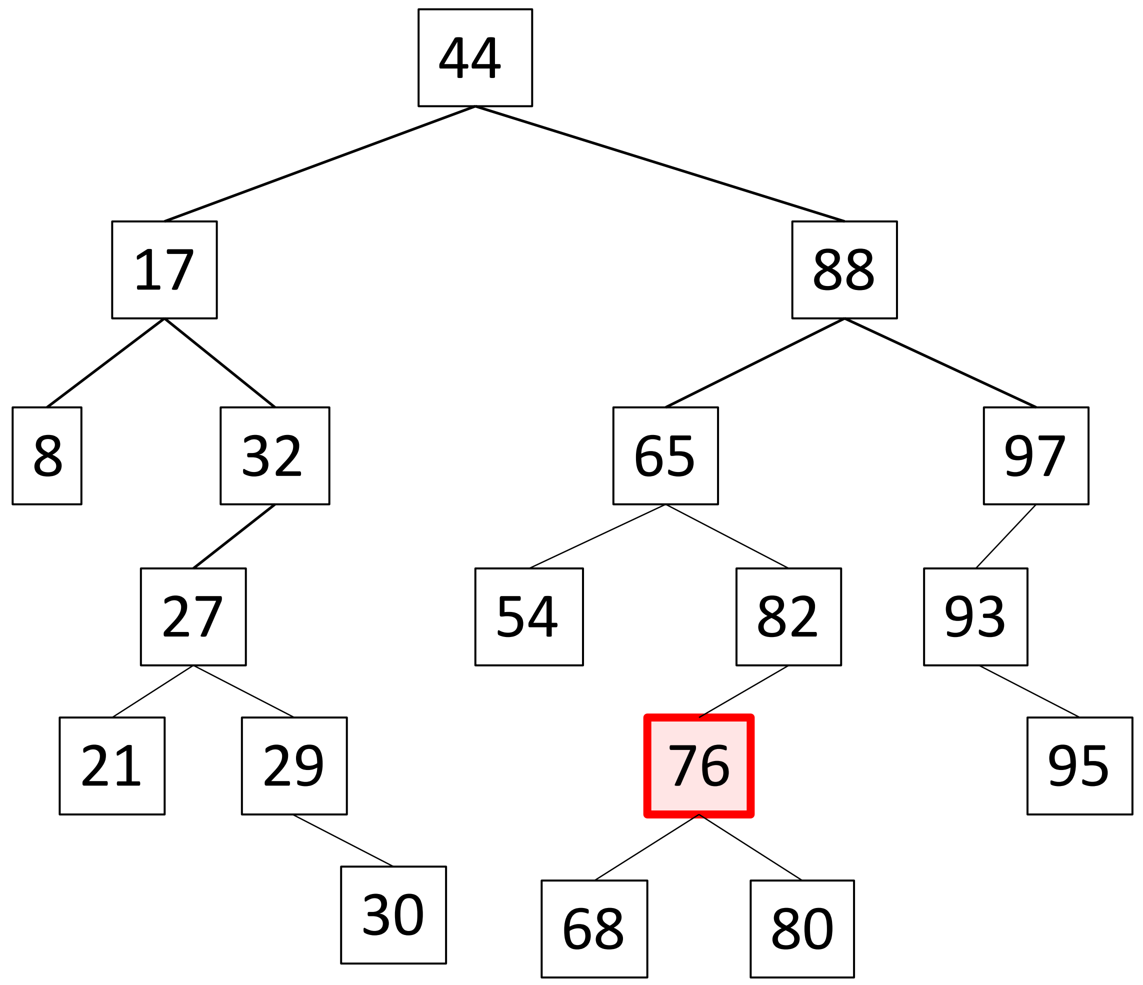
Step 1: Find the Node in the tree.



Binary Search Tree - Removal

```
remove(68);
```

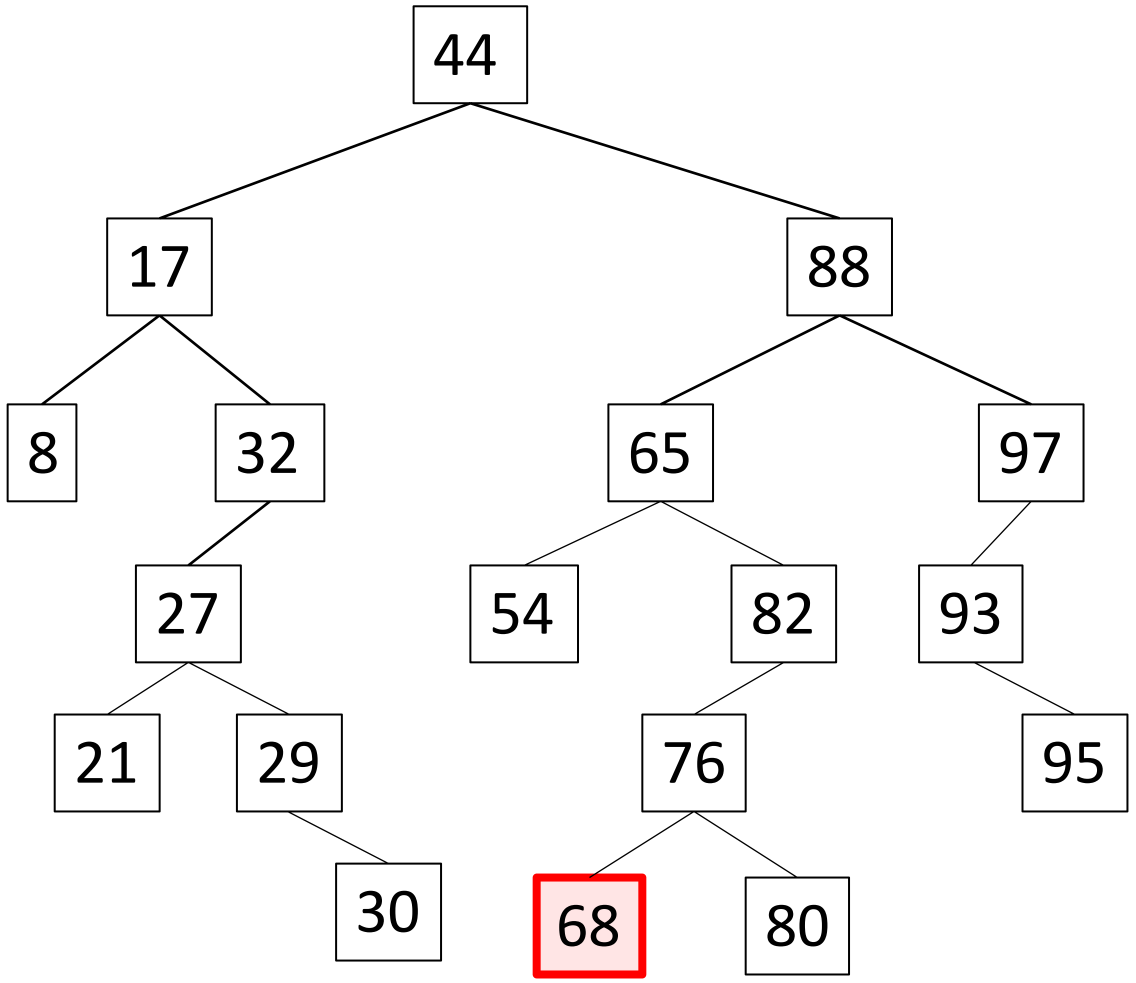
Step 1: Find the Node in the tree.



Binary Search Tree - Removal

```
remove(68);
```

Step 1: Find the Node in the tree.

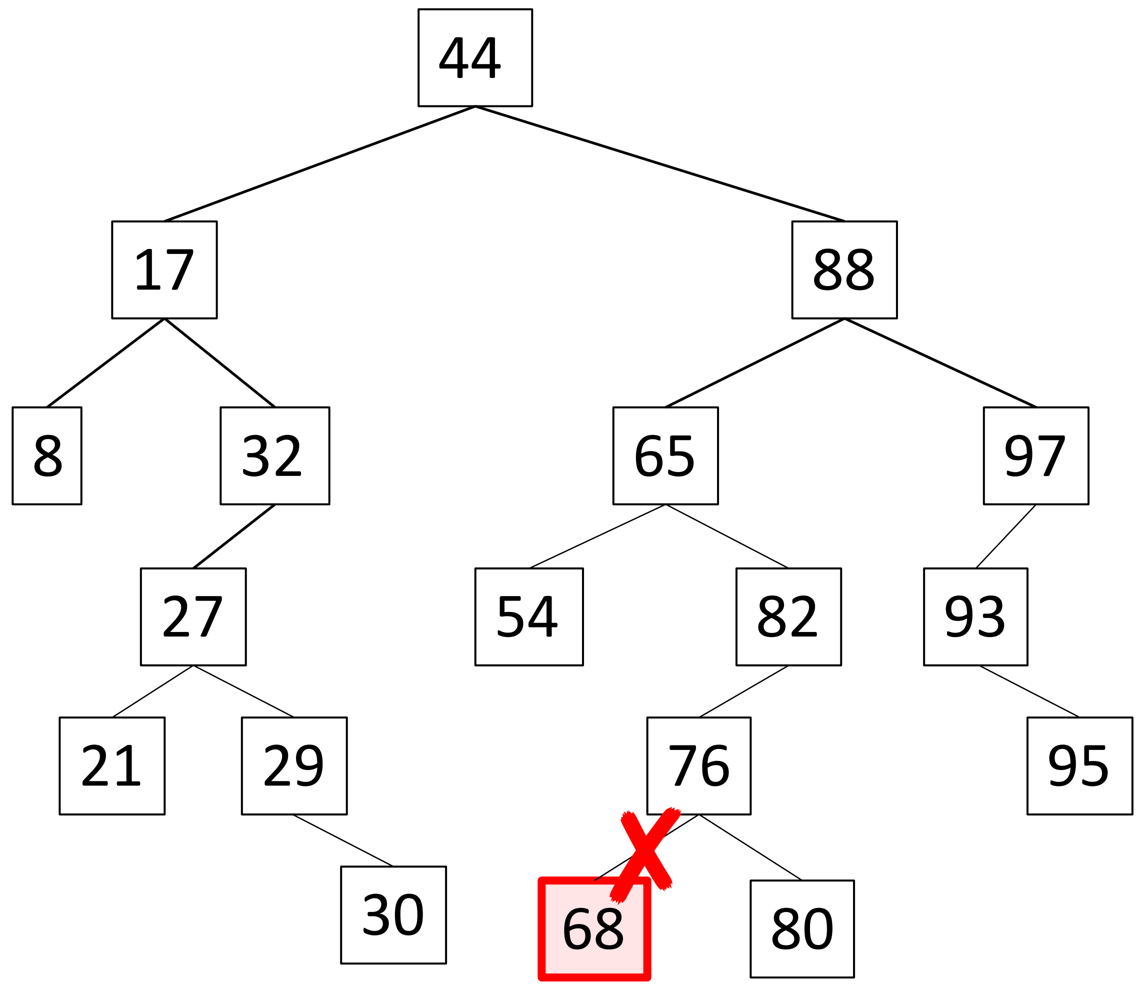


Binary Search Tree - Removal

```
remove(68);
```

Step 1: Find the Node in the tree.

Step 2: Change parent to point to null.



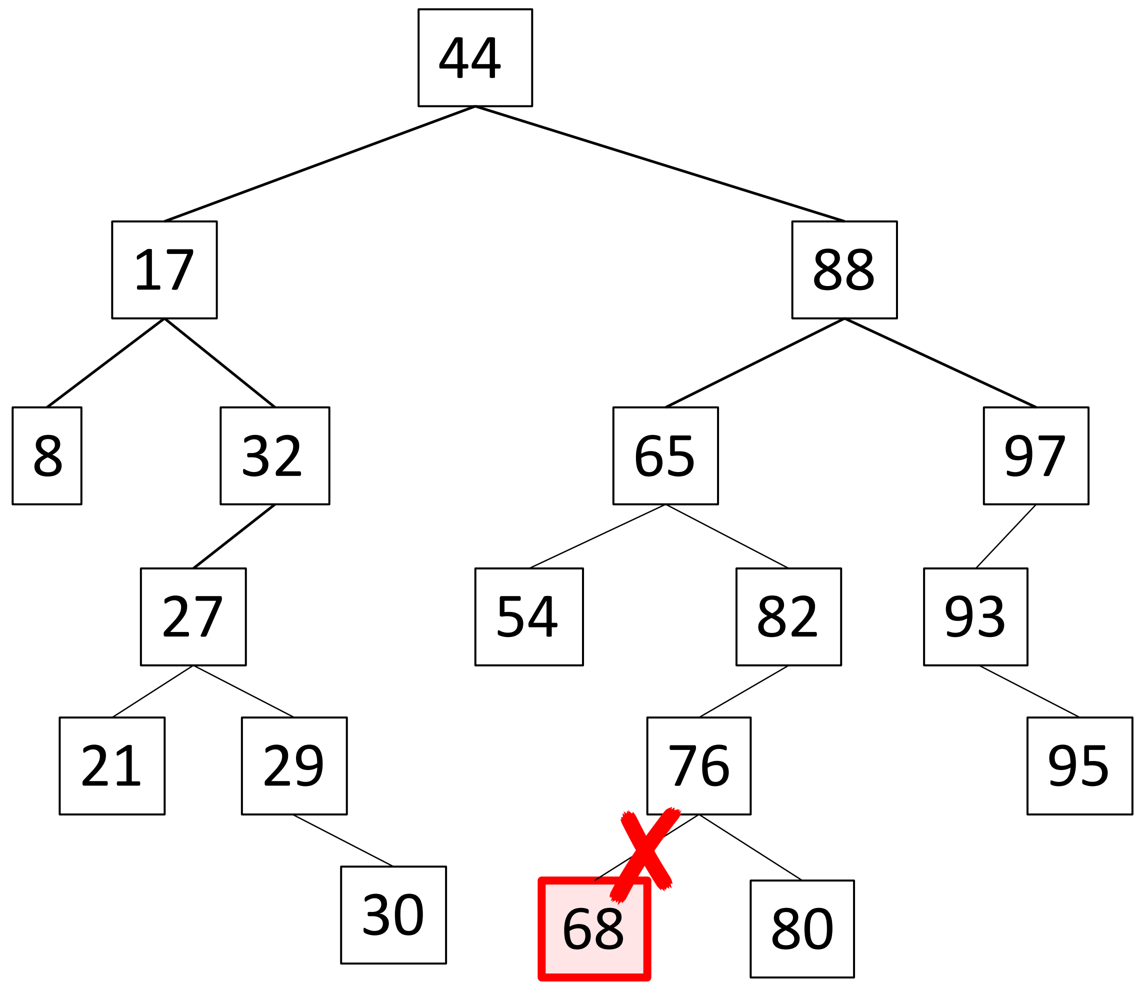
Binary Search Tree - Removal

```
remove(68);
```

Step 1: Find the Node in the tree.

Step 2: Change parent to point to null.

Does this always work?



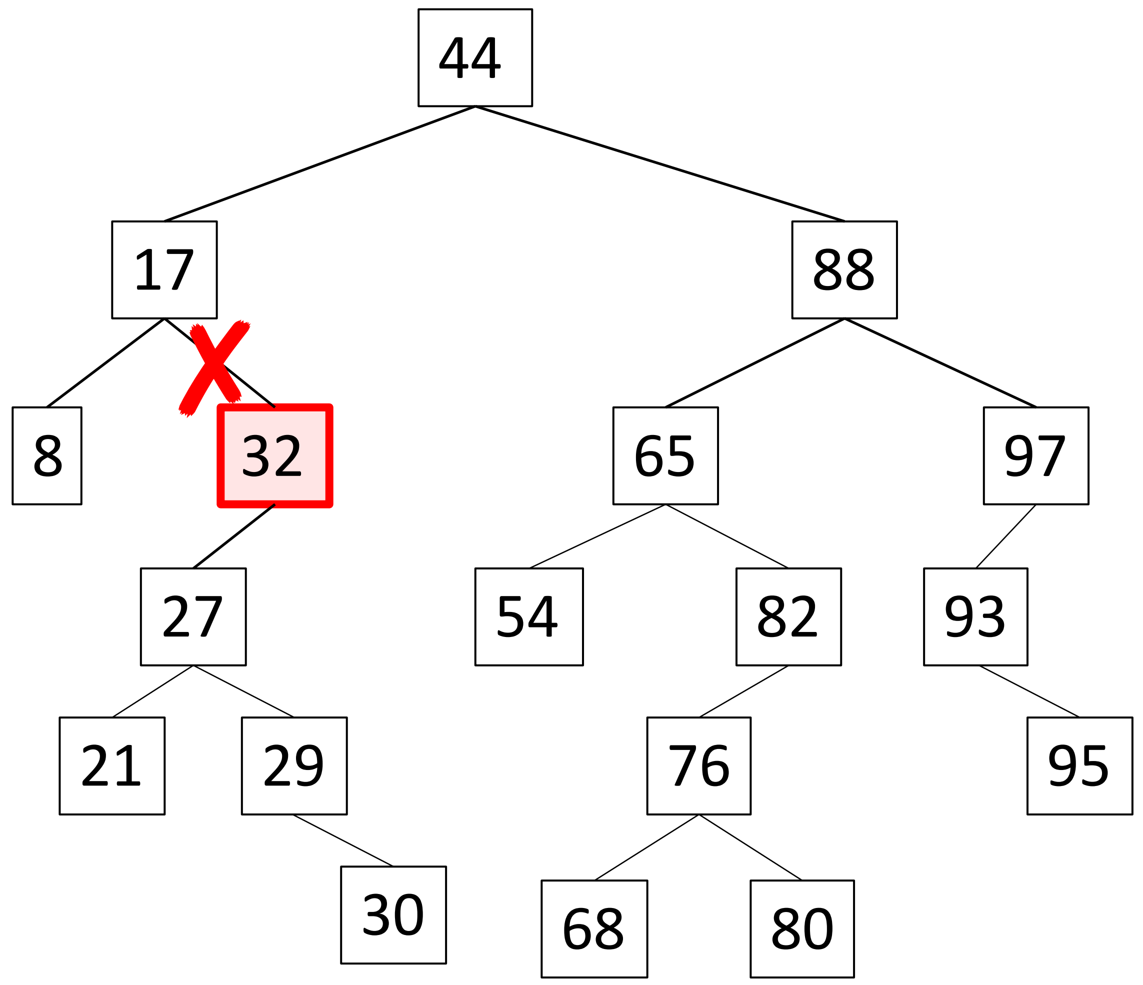
Binary Search Tree - Removal

```
remove(32);
```

Step 1: Find the Node in the tree.

Step 2: Change parent to point to null.

Does this always work?



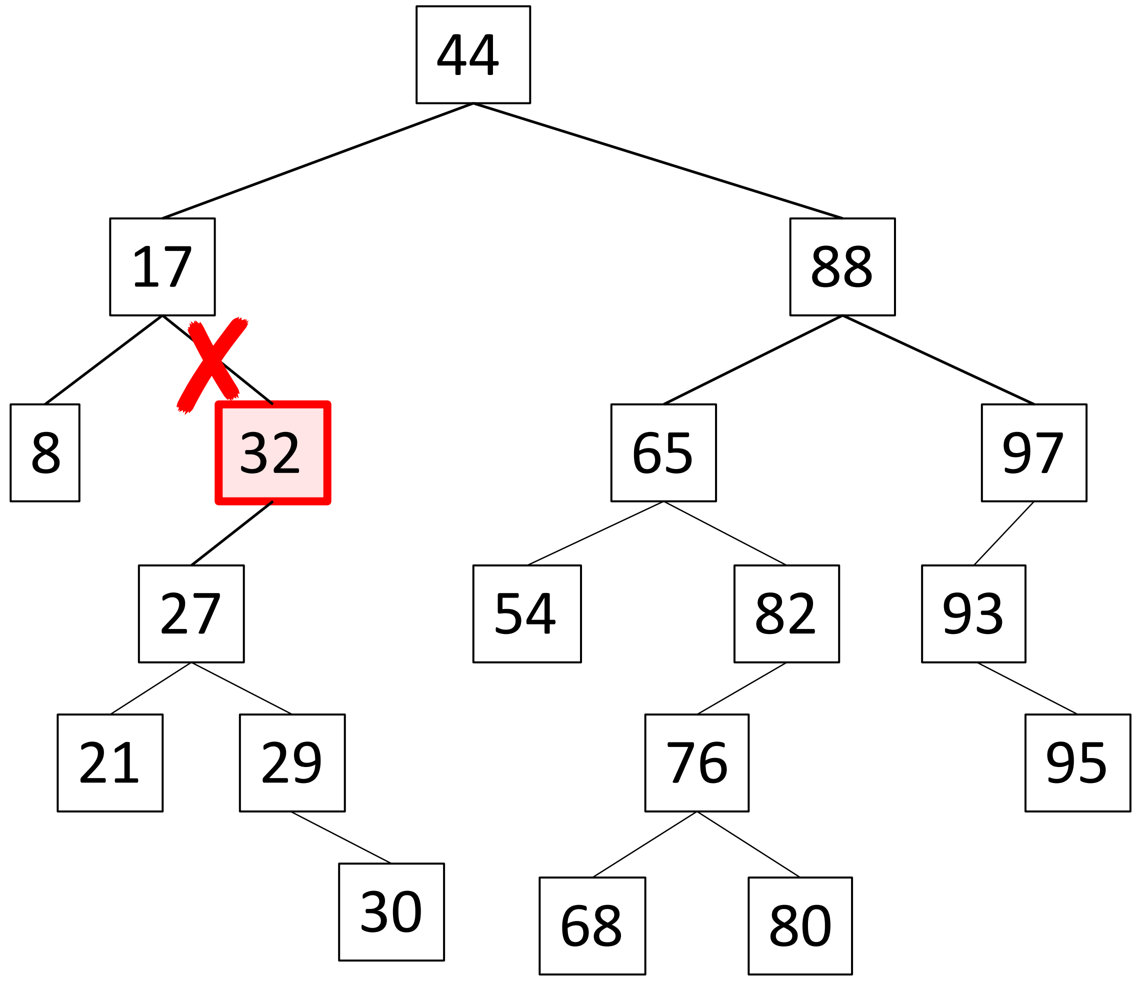
Binary Search Tree - Removal

```
remove(32);
```

Step 1: Find the Node in the tree.

Step 2: Change parent to point to null.

~~**Does this always work?**~~



Binary Search Tree - Removal

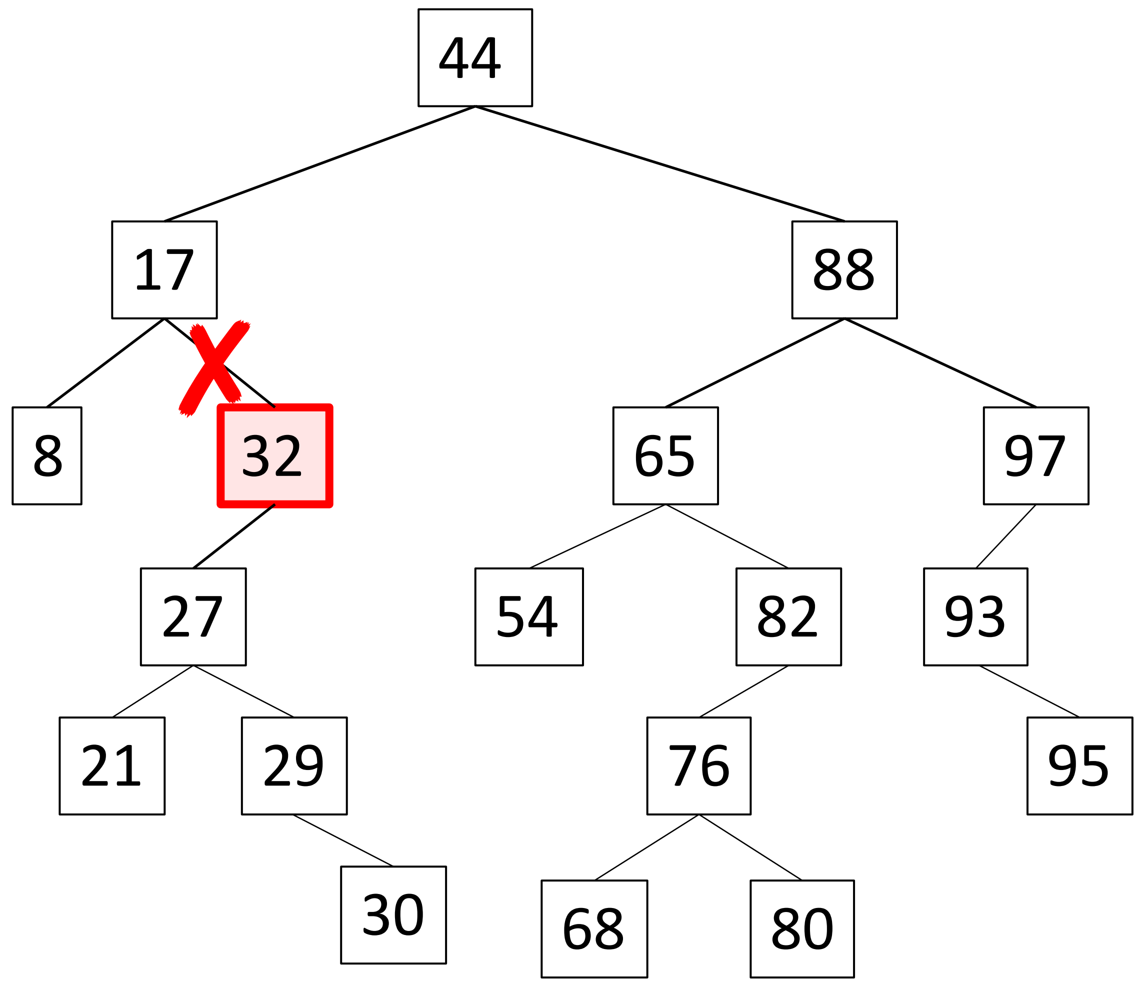
Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(32);
```

Step 1: Find the Node in the tree.

Step 2: Change parent to point to null.

~~**Does this always work?**~~



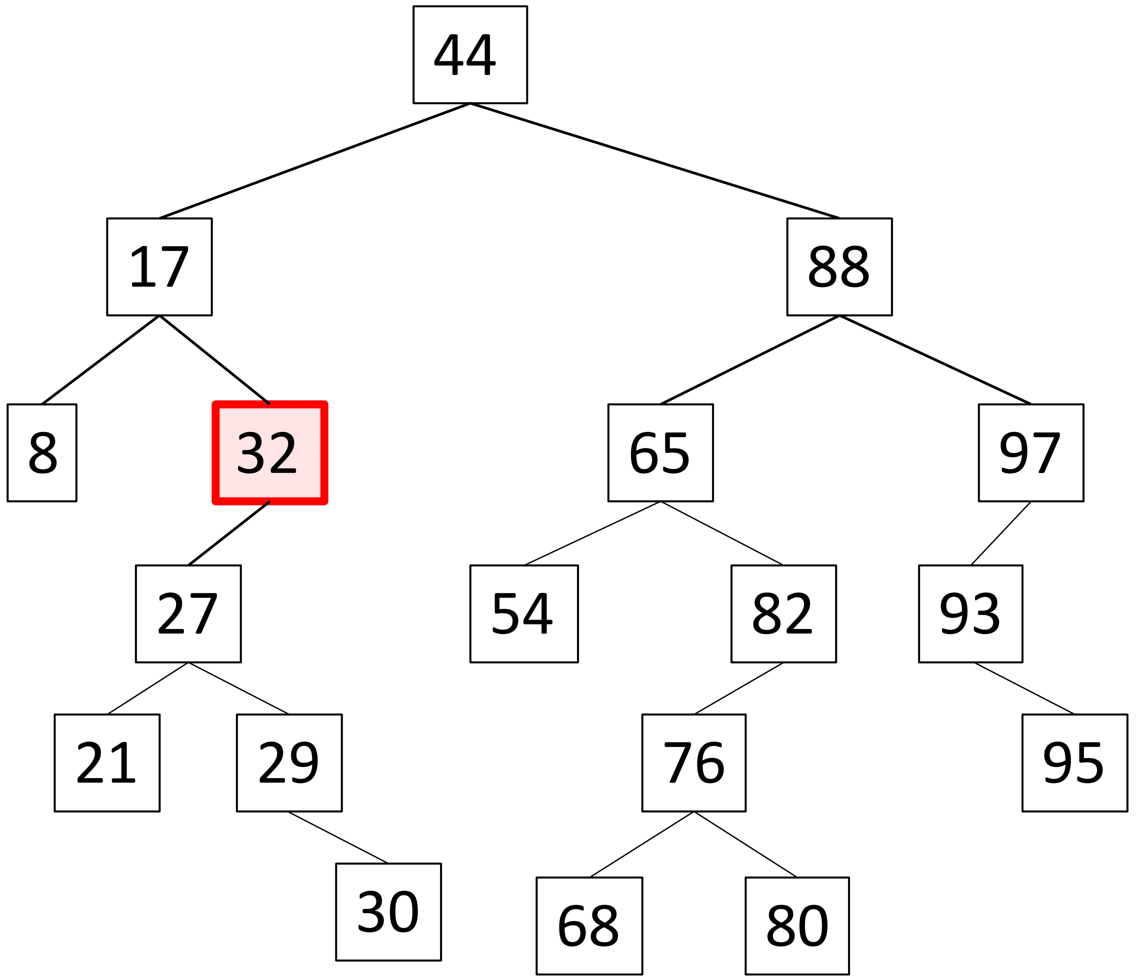
Binary Search Tree - Removal

Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(32);
```

Step 1: Find the Node in the tree.

Step 2: ???



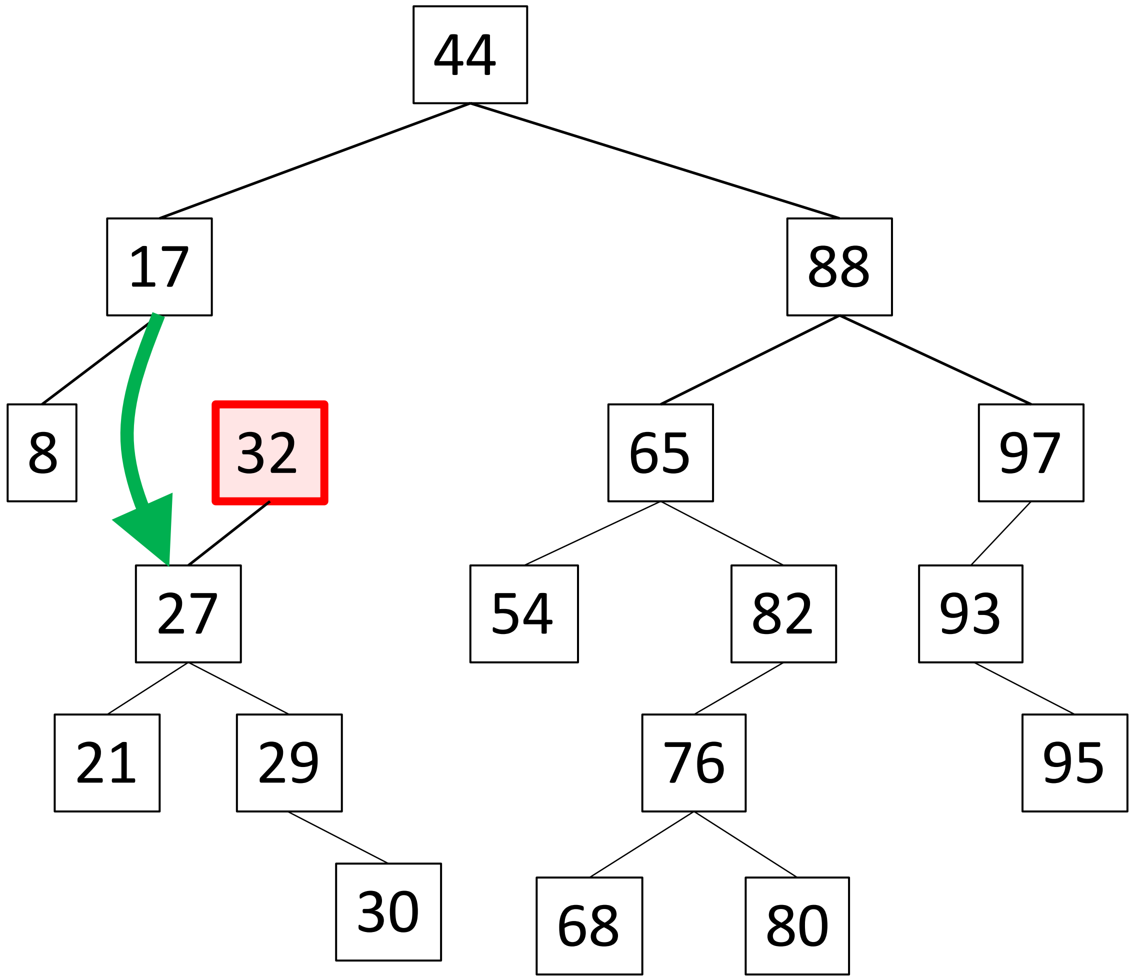
Binary Search Tree - Removal

Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(32);
```

Step 1: Find the Node in the tree.

Step 2: Change parent to point to child.



Binary Search Tree - Removal

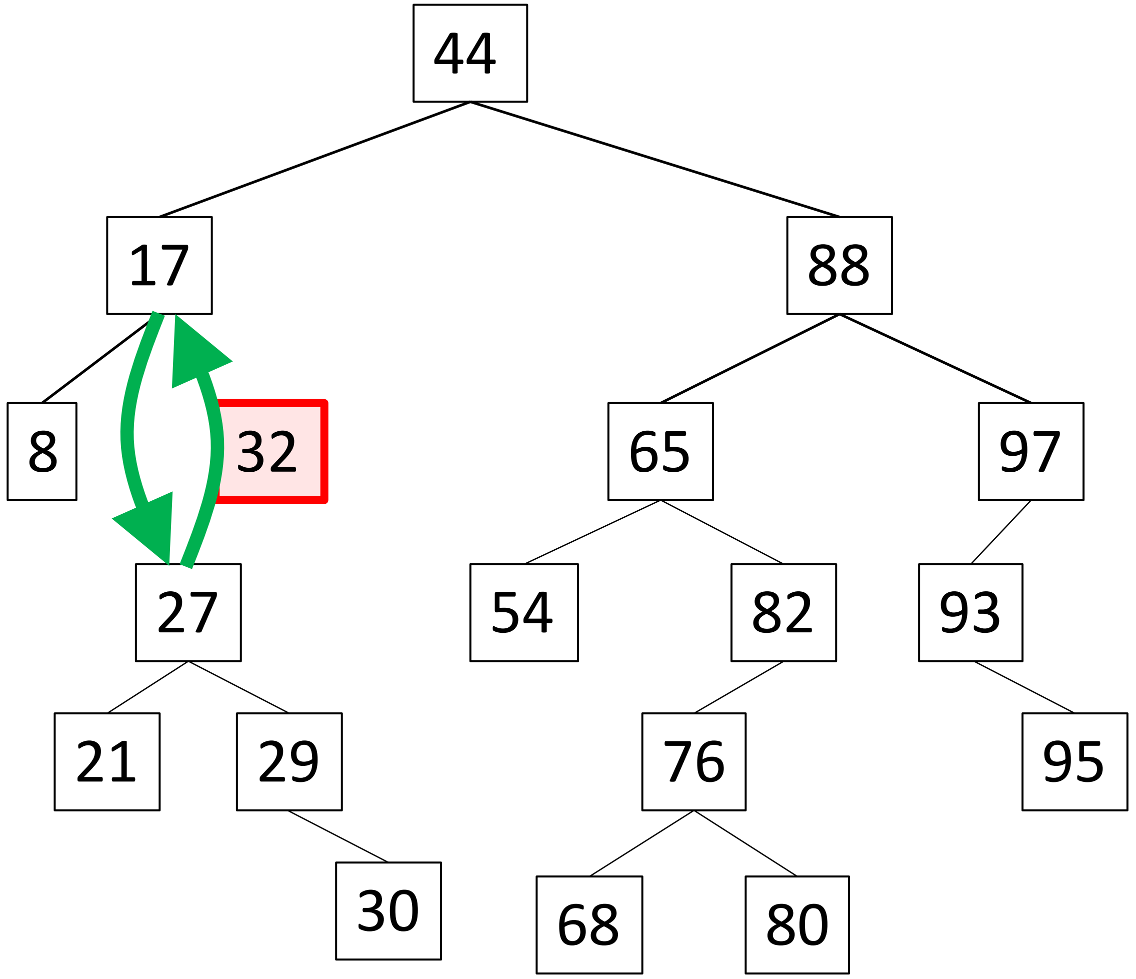
Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(32);
```

Step 1: Find the Node in the tree.

Step 2: Change parent to point to child.

Step 3: Change child to point to parent.



Binary Search Tree - Removal

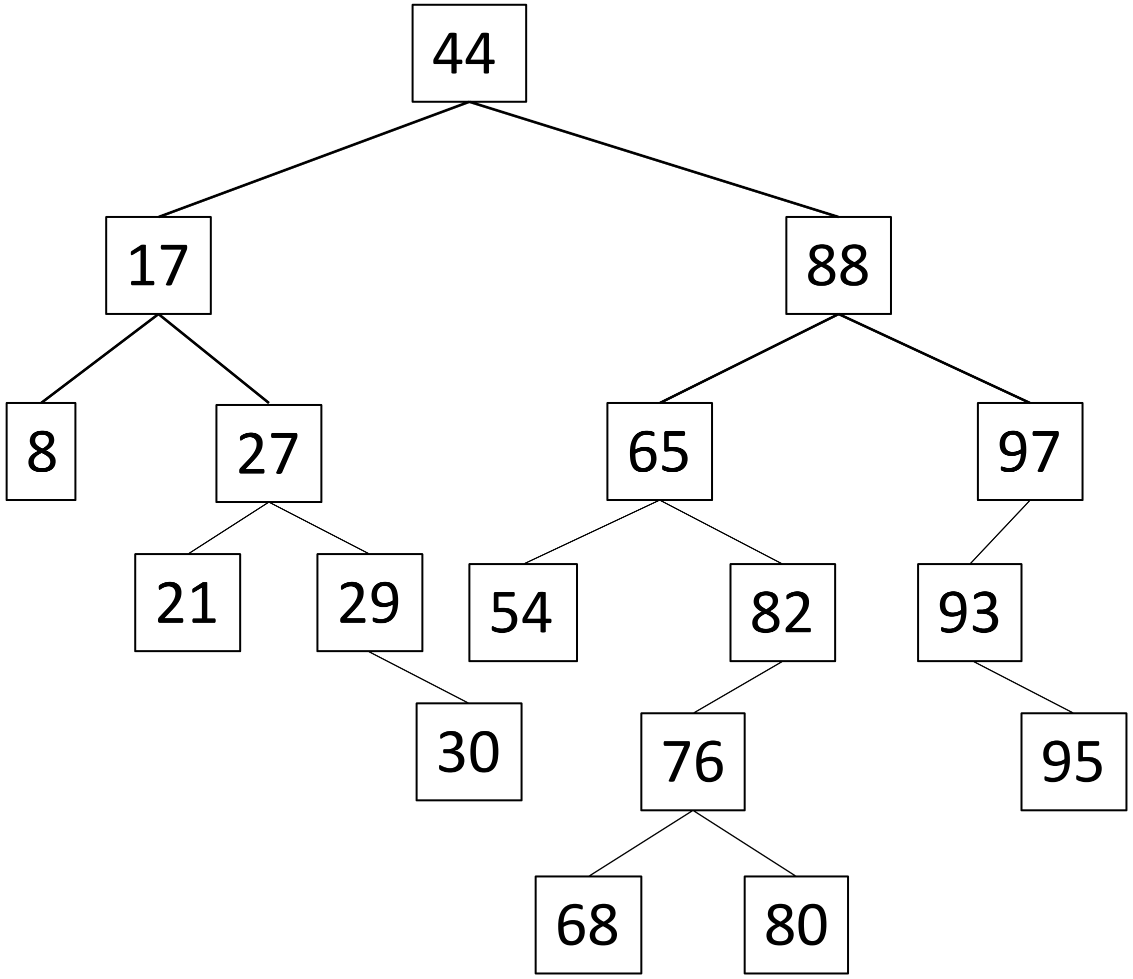
Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(32);
```

Step 1: Find the Node in the tree.

Step 2: Change parent to point to child.

Step 3: Change child to point to parent.

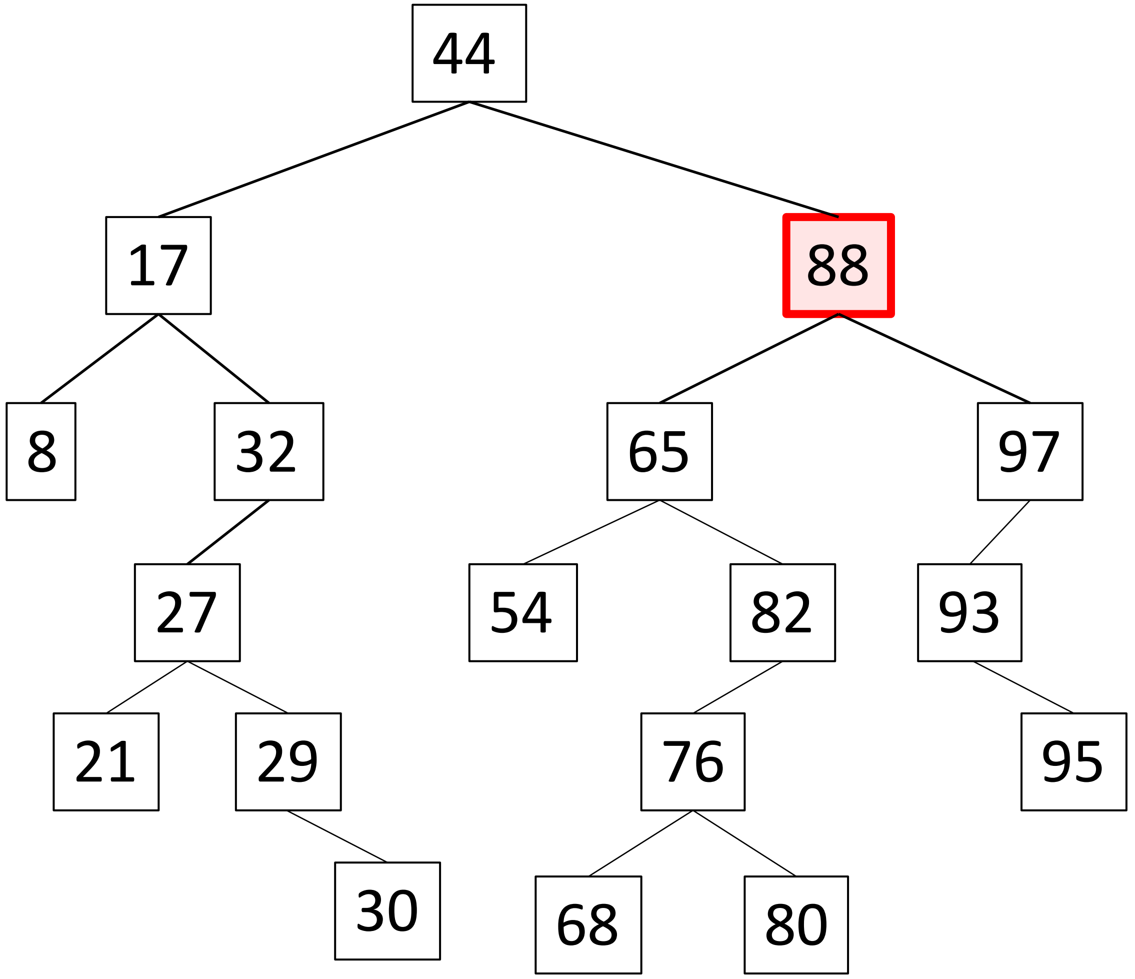


Binary Search Tree - Removal

Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(88);
```

Step 1: Find the Node in the tree.



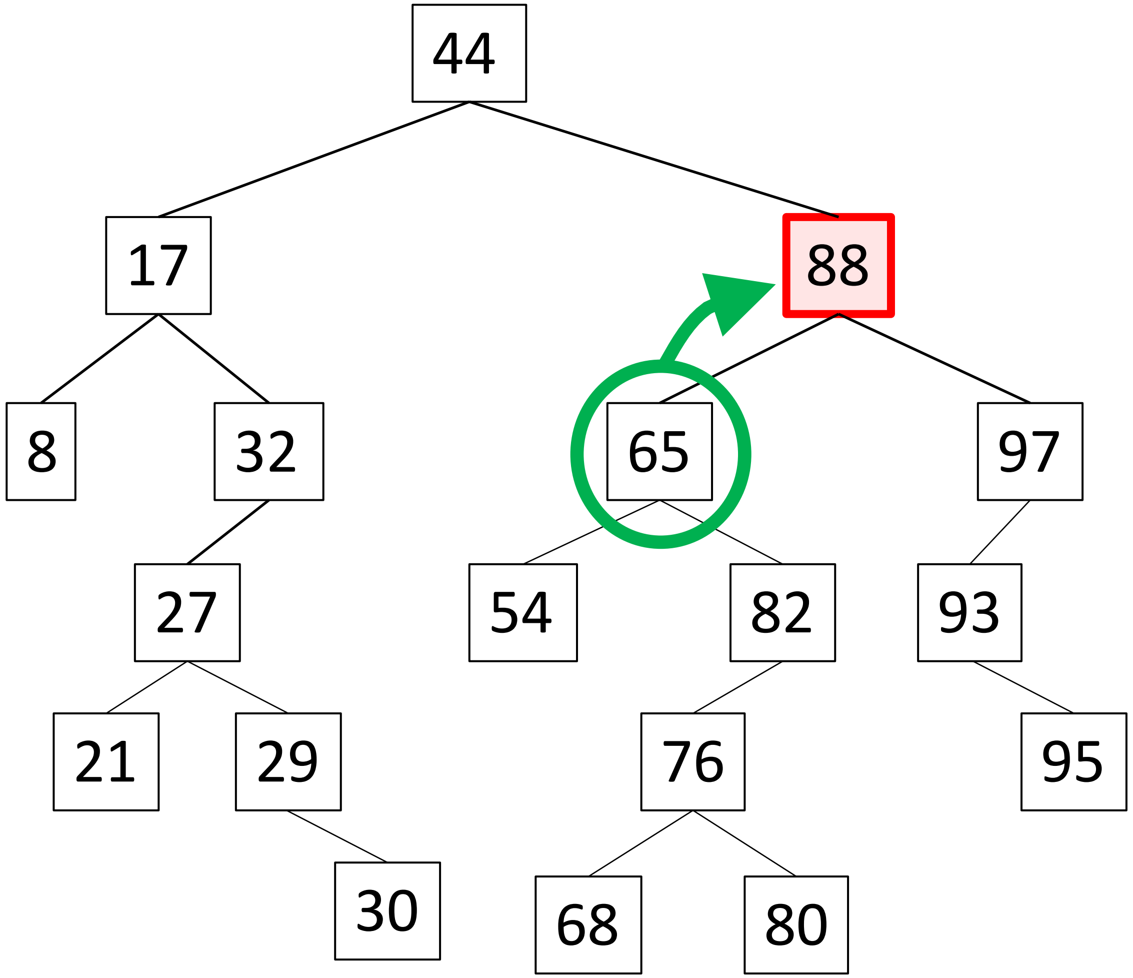
Binary Search Tree - Removal

Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(88);
```

Step 1: Find the Node in the tree.

Can I just move one of the children nodes into the parent spot?



Binary Search Tree - Removal

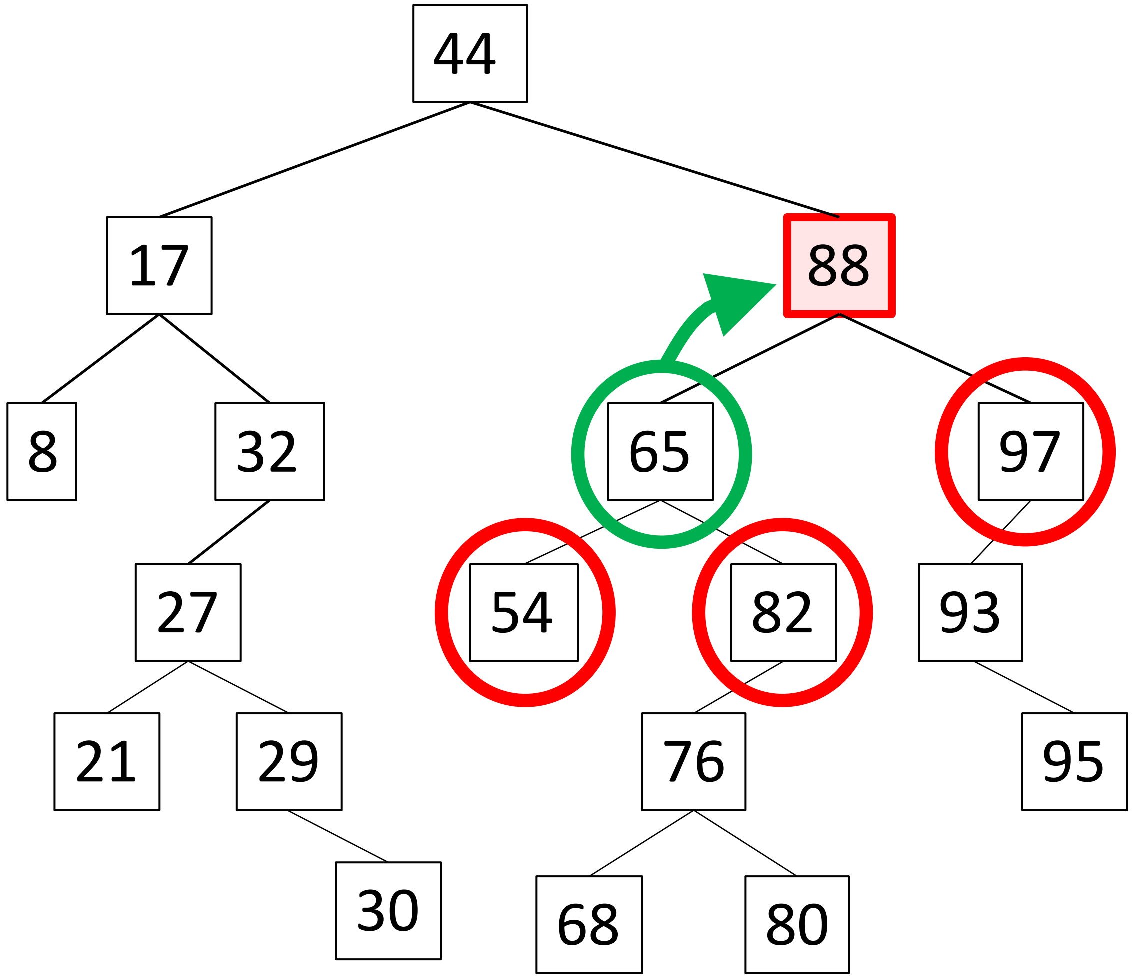
Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(88);
```

Step 1: Find the Node in the tree.

Can I just move one of the children nodes into the parent spot?

No. It already has two children. What happens to 97?



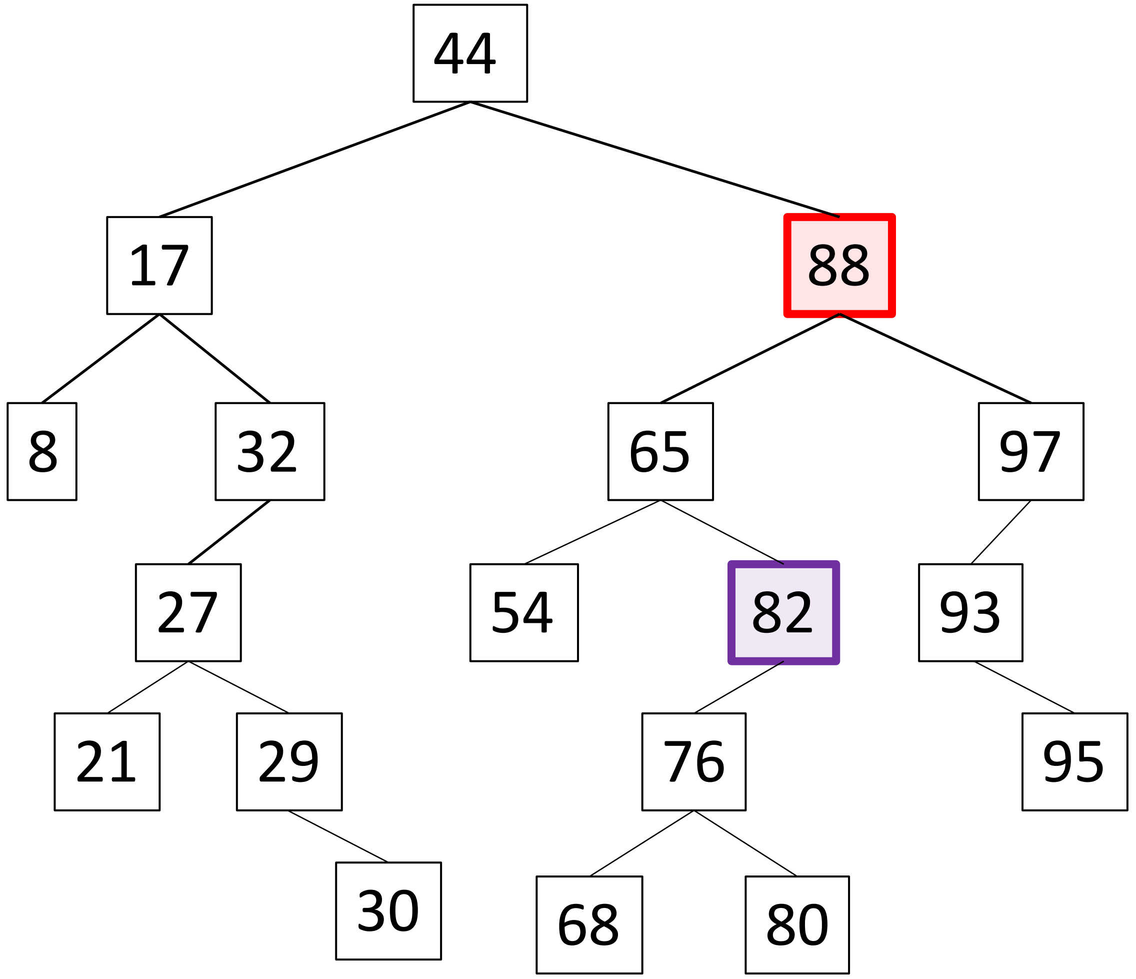
Binary Search Tree - Removal

Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(88);
```

Step 1: Find the Node in the tree.

Step 2: Find the largest node on the left-hand side.



Binary Search Tree - Removal

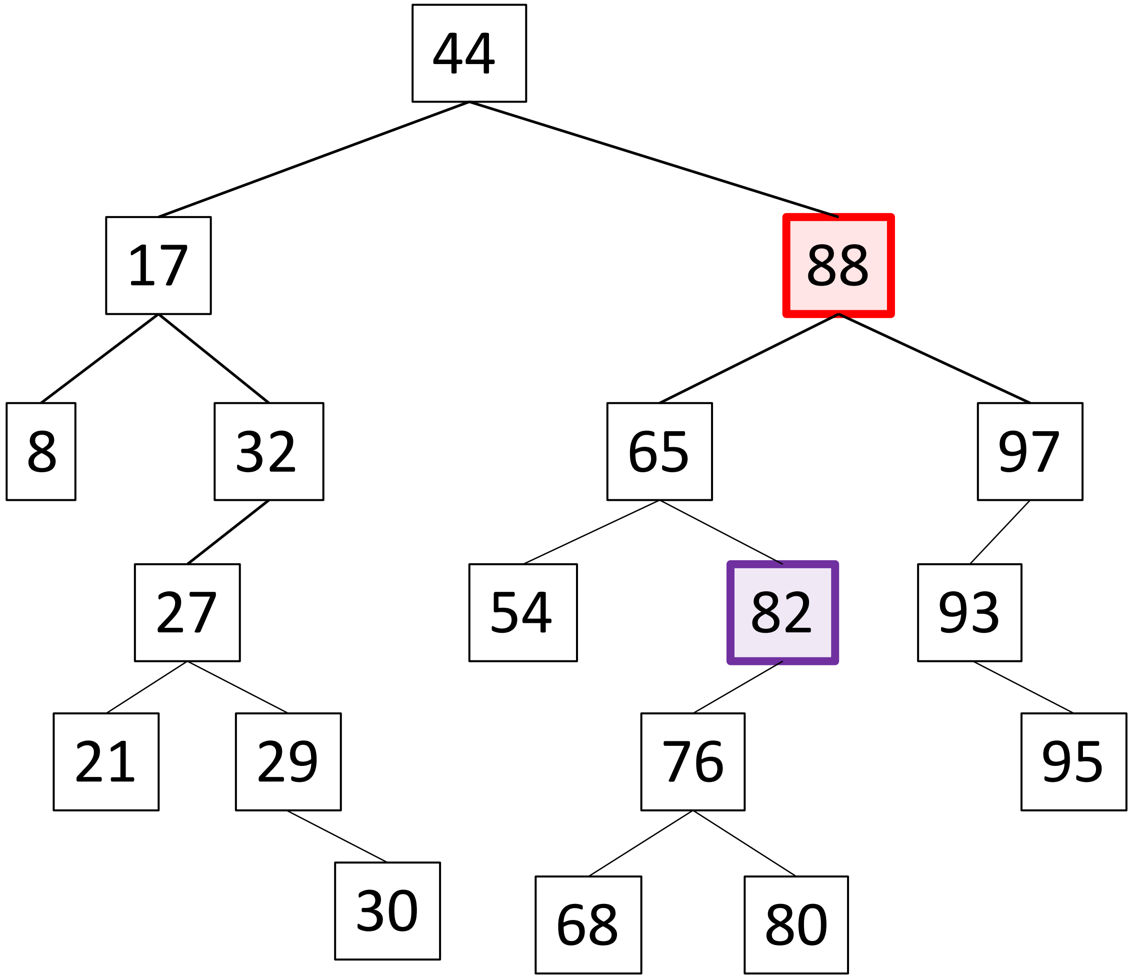
Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(88);
```

Step 1: Find the Node in the tree.

Step 2: Find the largest node on the left-hand side.

The largest node on the left-hand side cannot have two children, because ???



Binary Search Tree - Removal

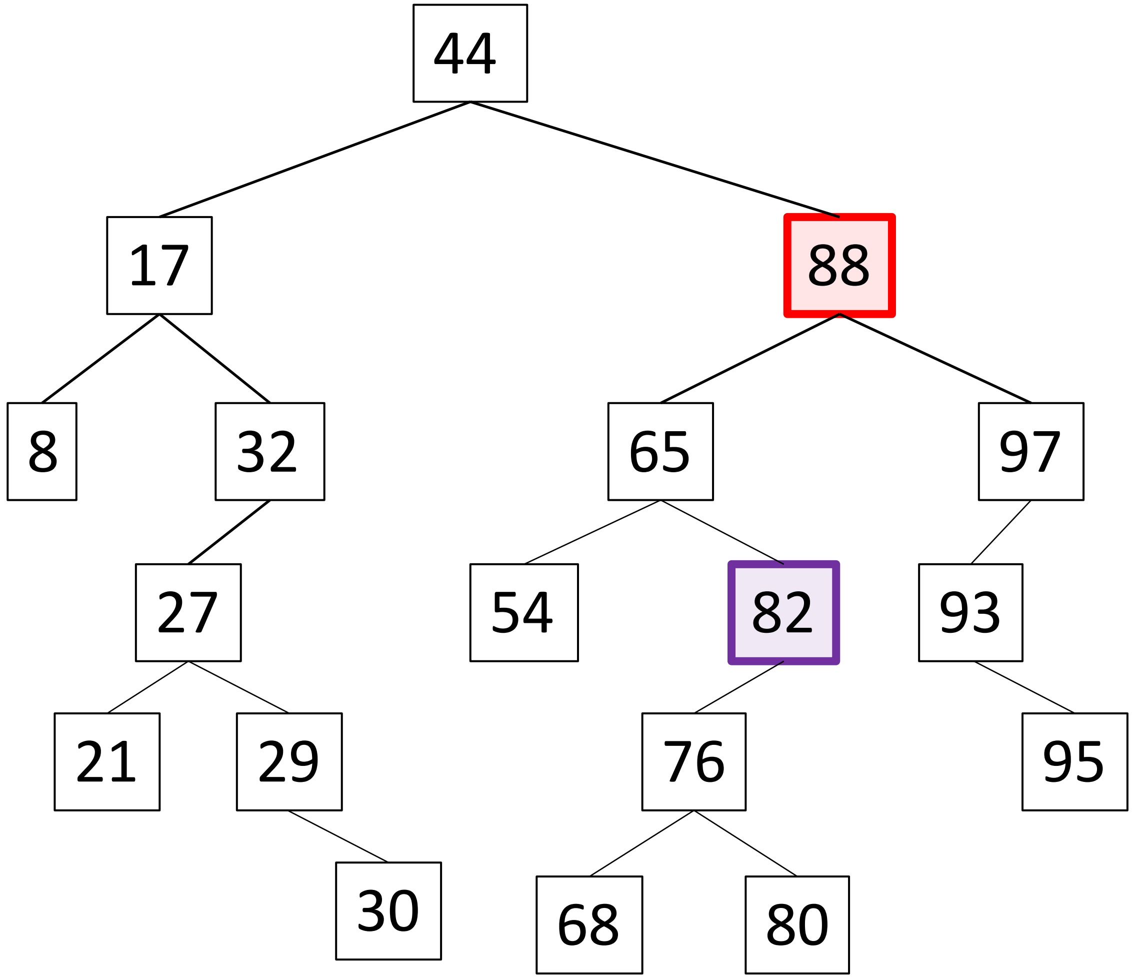
Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(88);
```

Step 1: Find the Node in the tree.

Step 2: Find the largest node on the left-hand side.

The largest node on the left-hand side cannot have two children, because if it did, one would be larger.



Binary Search Tree - Removal

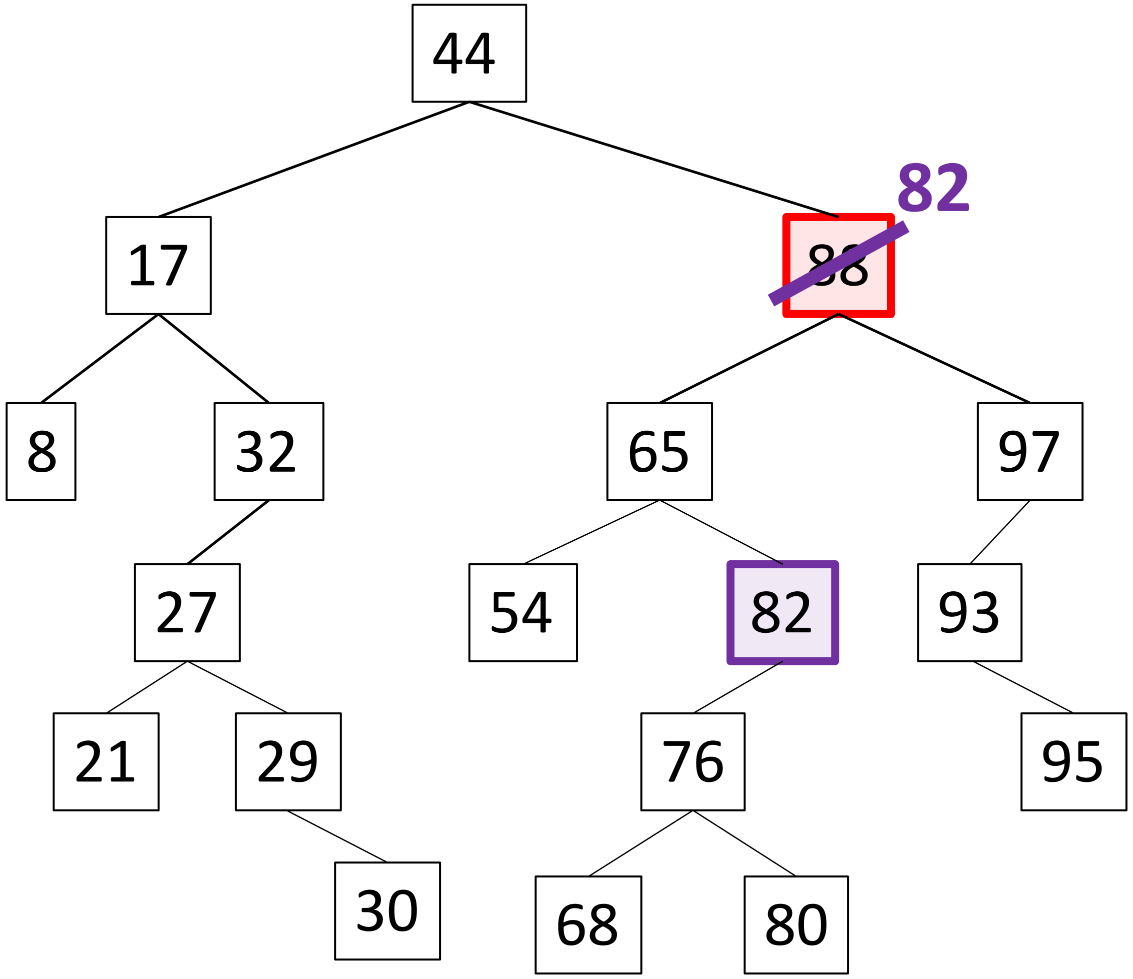
Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(88);
```

Step 1: Find the Node in the tree.

Step 2: Find the largest node on the left-hand side.

Step 3: Put that value in the Node being removed.



Binary Search Tree - Removal

Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

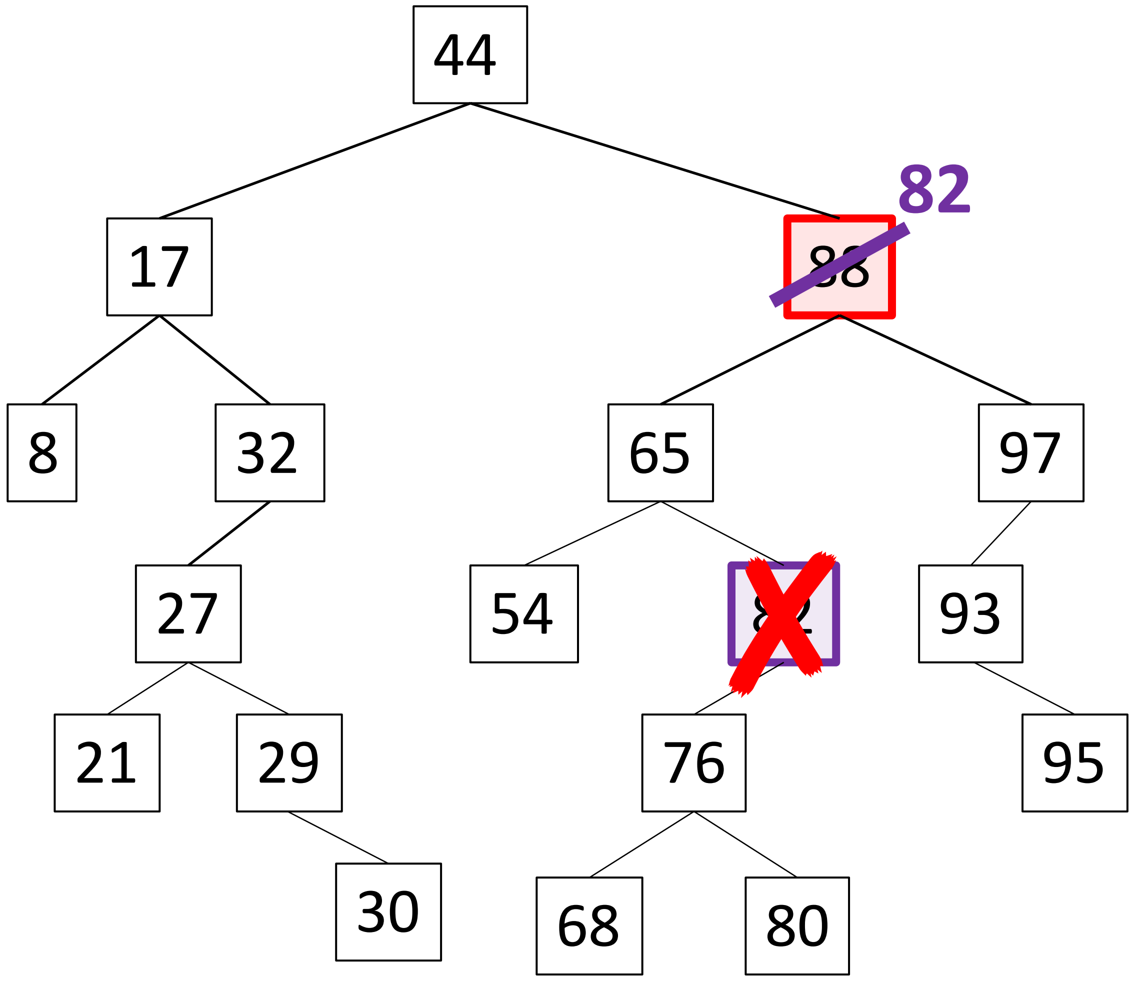
```
remove(88);
```

Step 1: Find the Node in the tree.

Step 2: Find the largest node on the left-hand side.

Step 3: Put that value in the Node being removed.

Step 4: Remove the largest node on the left-hand side.



Binary Search Tree - Removal

Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(88);
```

Step 1: Find the Node in the tree

Step 2: Find the

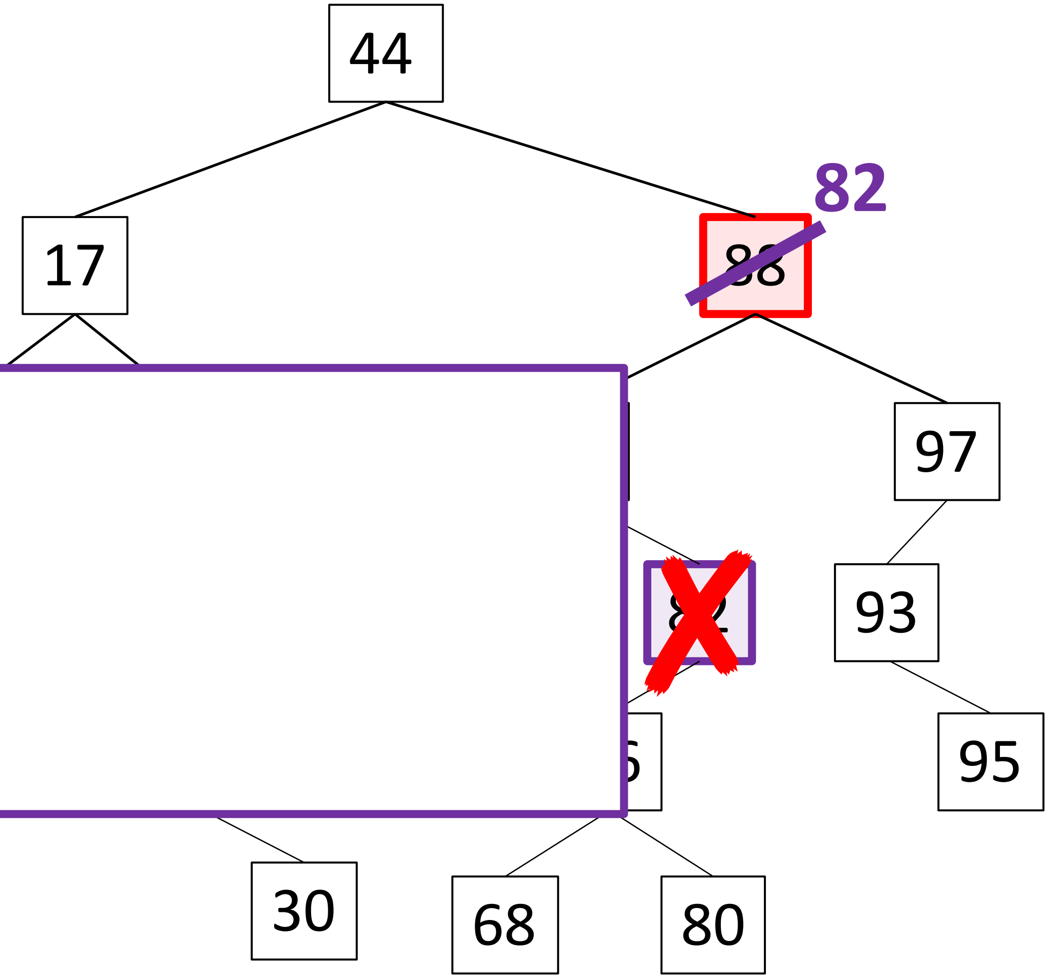
left-hand

Step 3: Put that

being re

Step 4: Remove the largest node on the left-hand side.

Running time?



Binary Search Tree - Removal

Case 1: Node has no children
Case 2: Node has one child
Case 3: Node has two children

```
remove(88);
```

Step 1: Find the Node in the tree

Step 2: Find the

left-hand

Step 3: Put that v

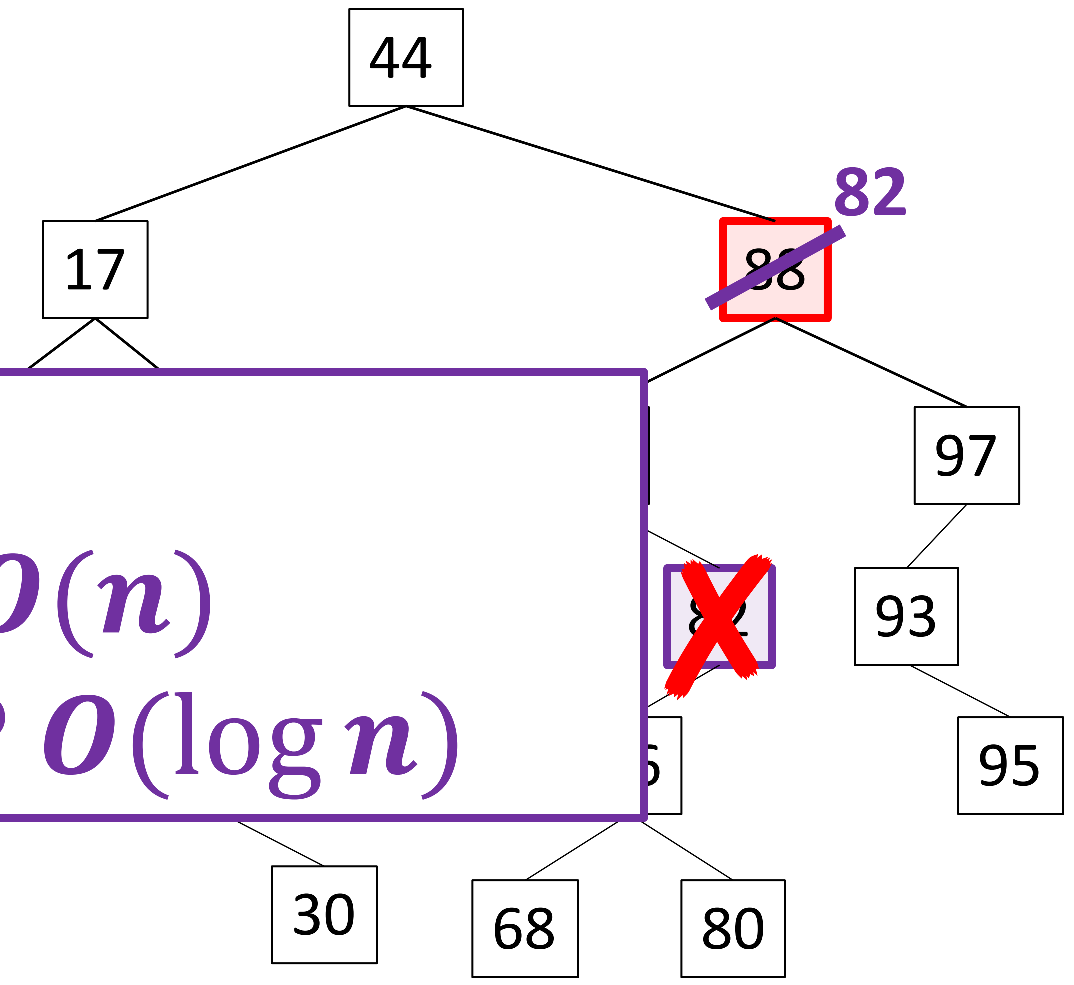
being rem

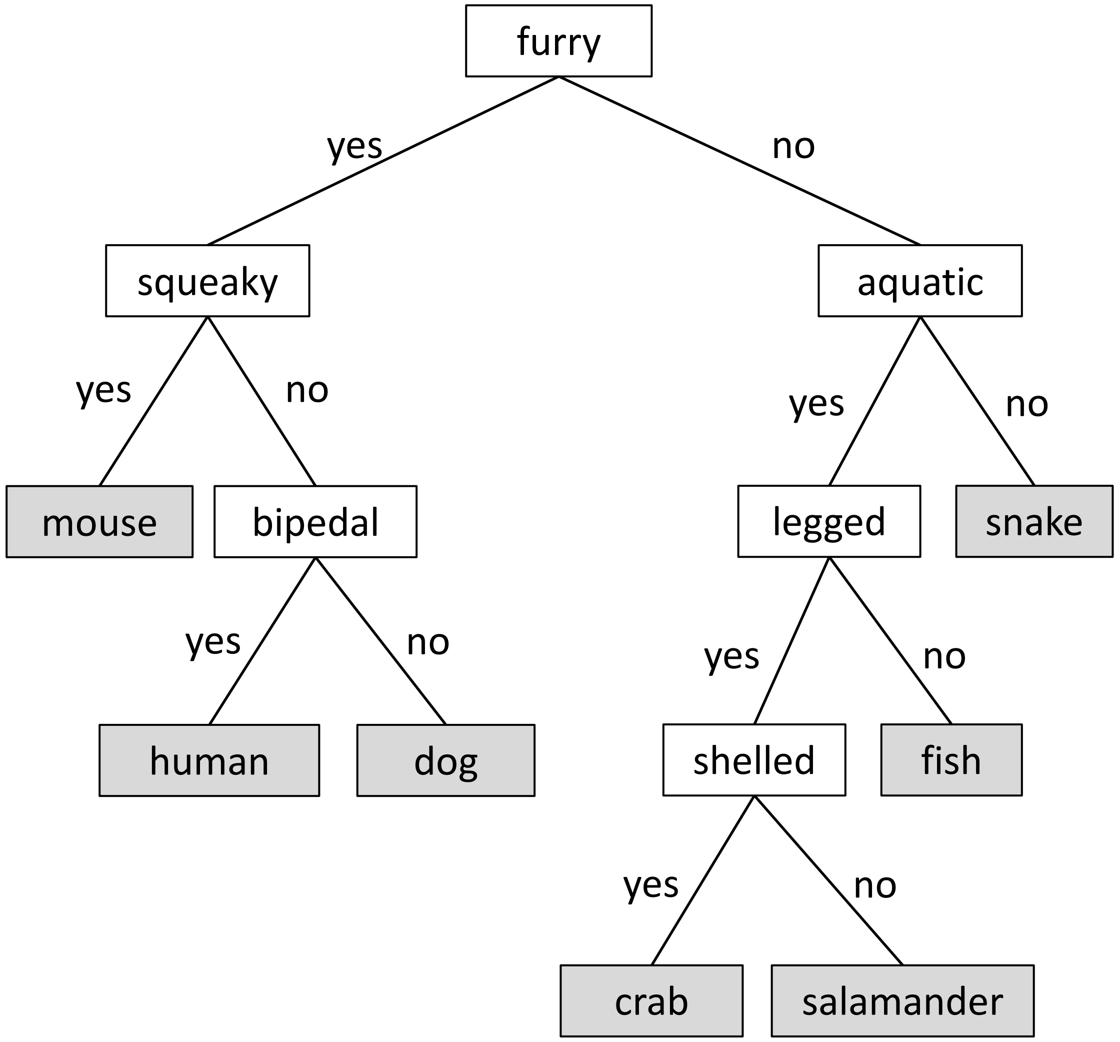
Step 4: Remove the largest node on the left-hand side.

Running time?

"Bad" tree? $O(n)$

"Good" tree? $O(\log n)$

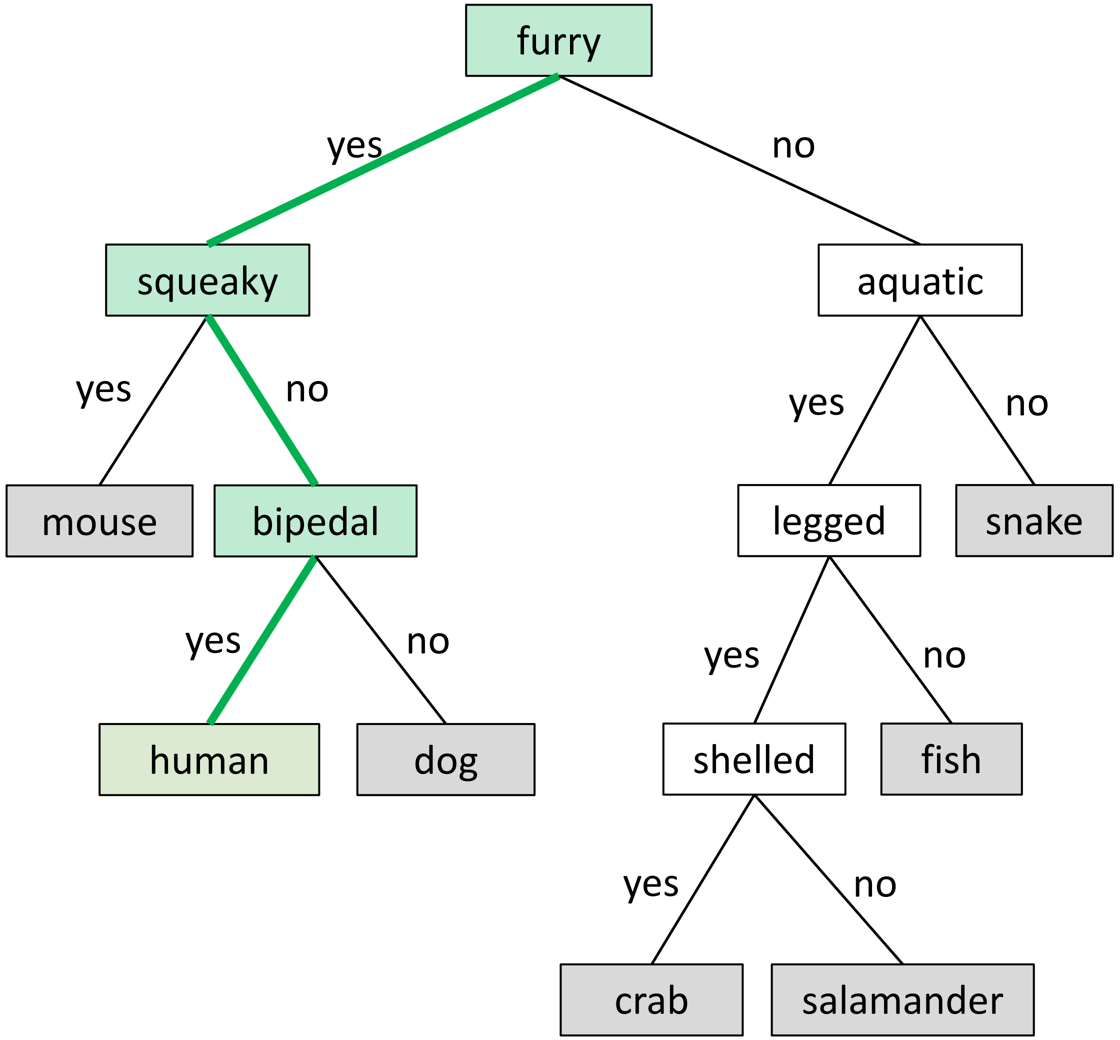




```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I don't know any furry, not squeaky, bipedal animals that aren't a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive
```

Program Execution:

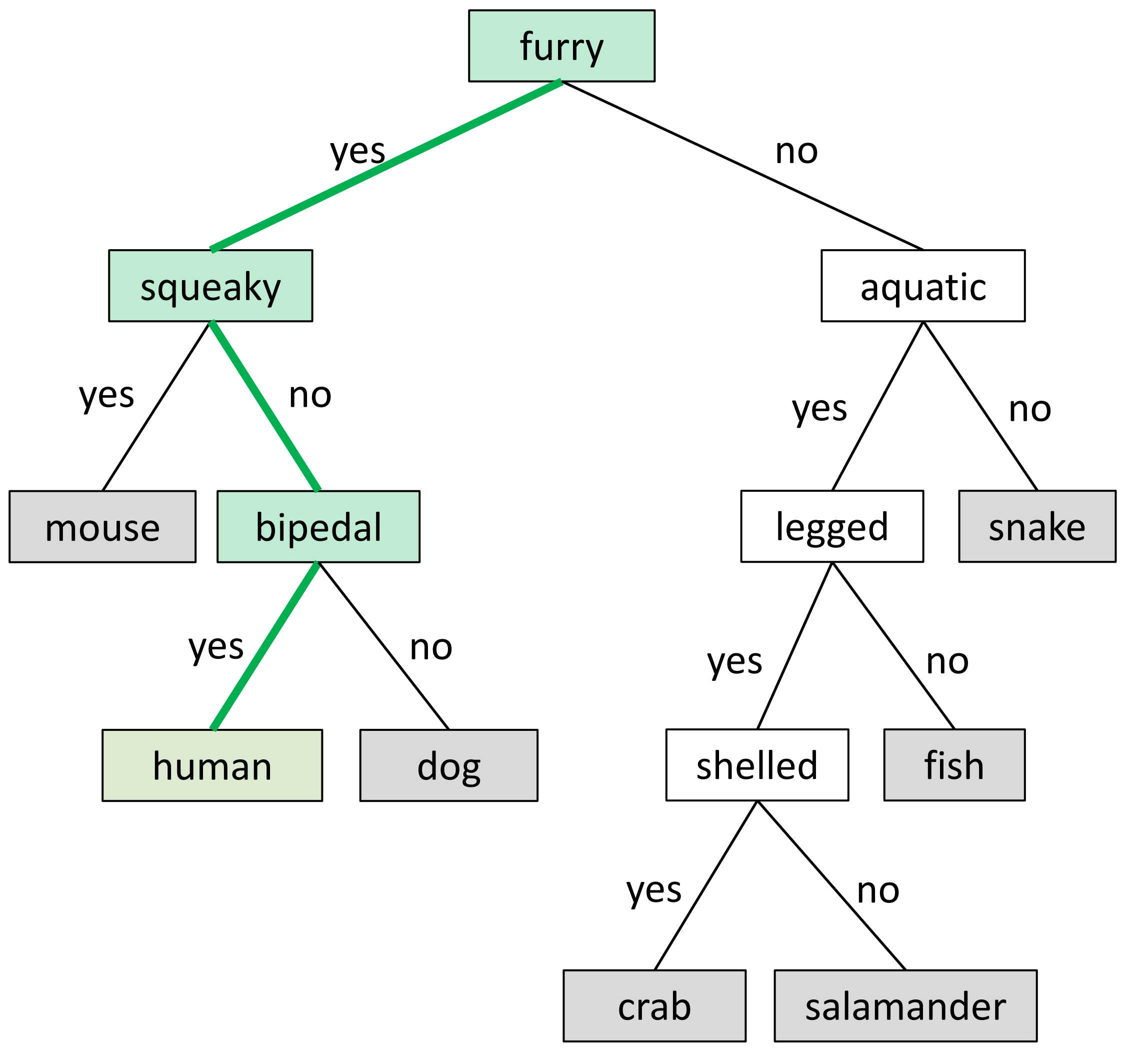
1.



```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I don't know any furry, not squeaky, bipedal animals that aren't a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive
```

Program Execution:

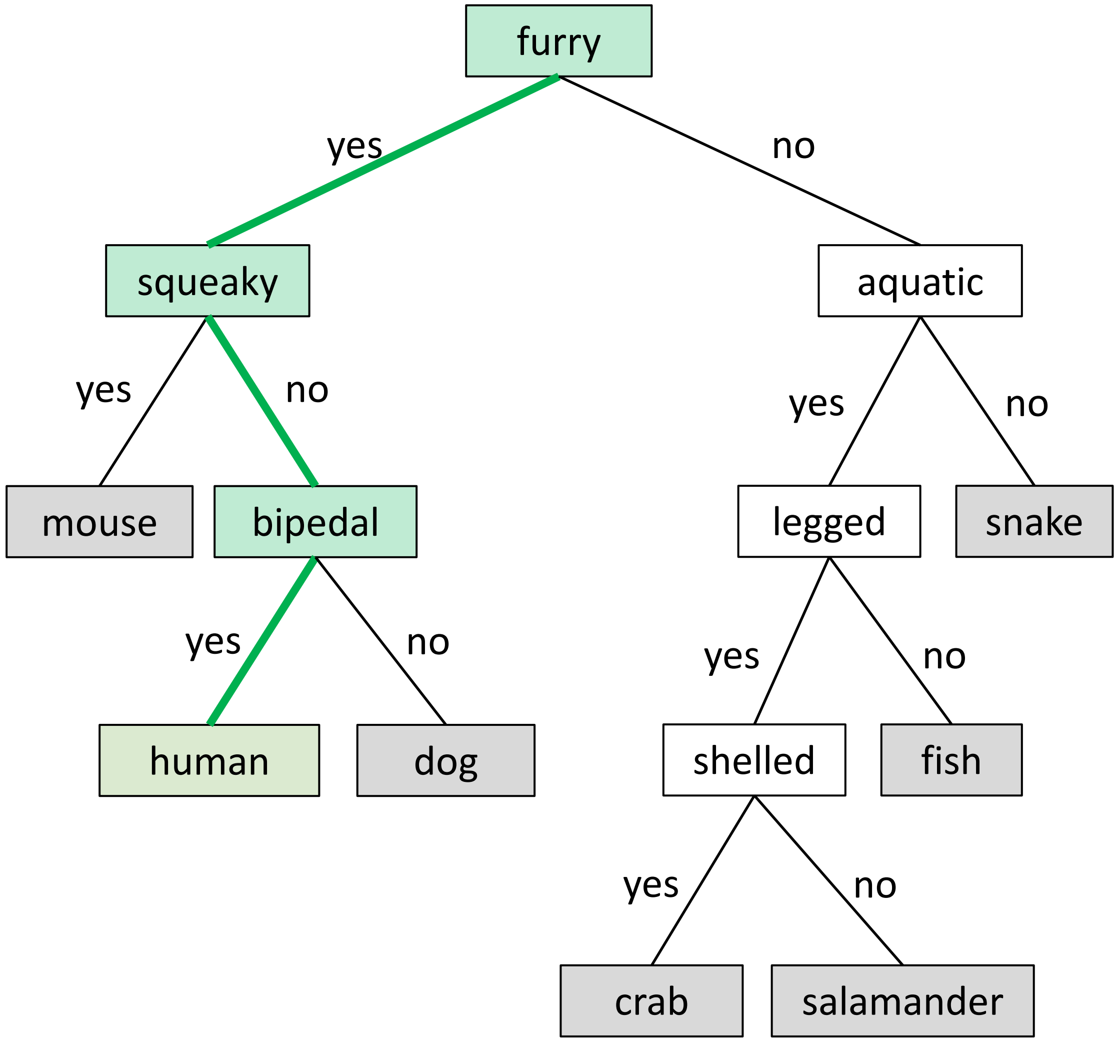
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3.



```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I don't know any furry, not squeaky, bipedal animals that aren't a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive
```

Program Execution:

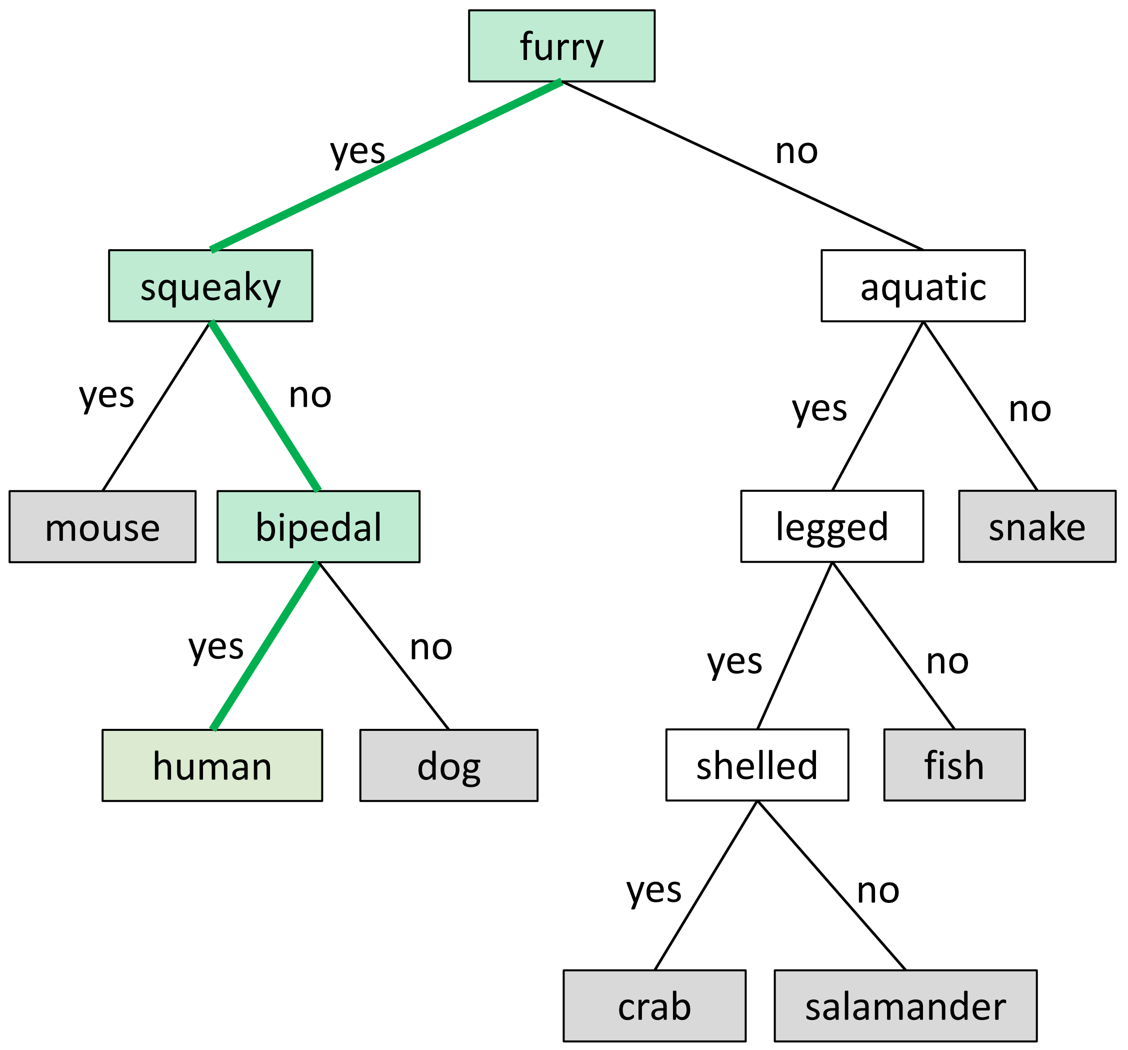
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
    3.1. Print location in tree.
    3.2.



```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I don't know any furry, not squeaky, bipedal animals that aren't a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive
```

Program Execution:

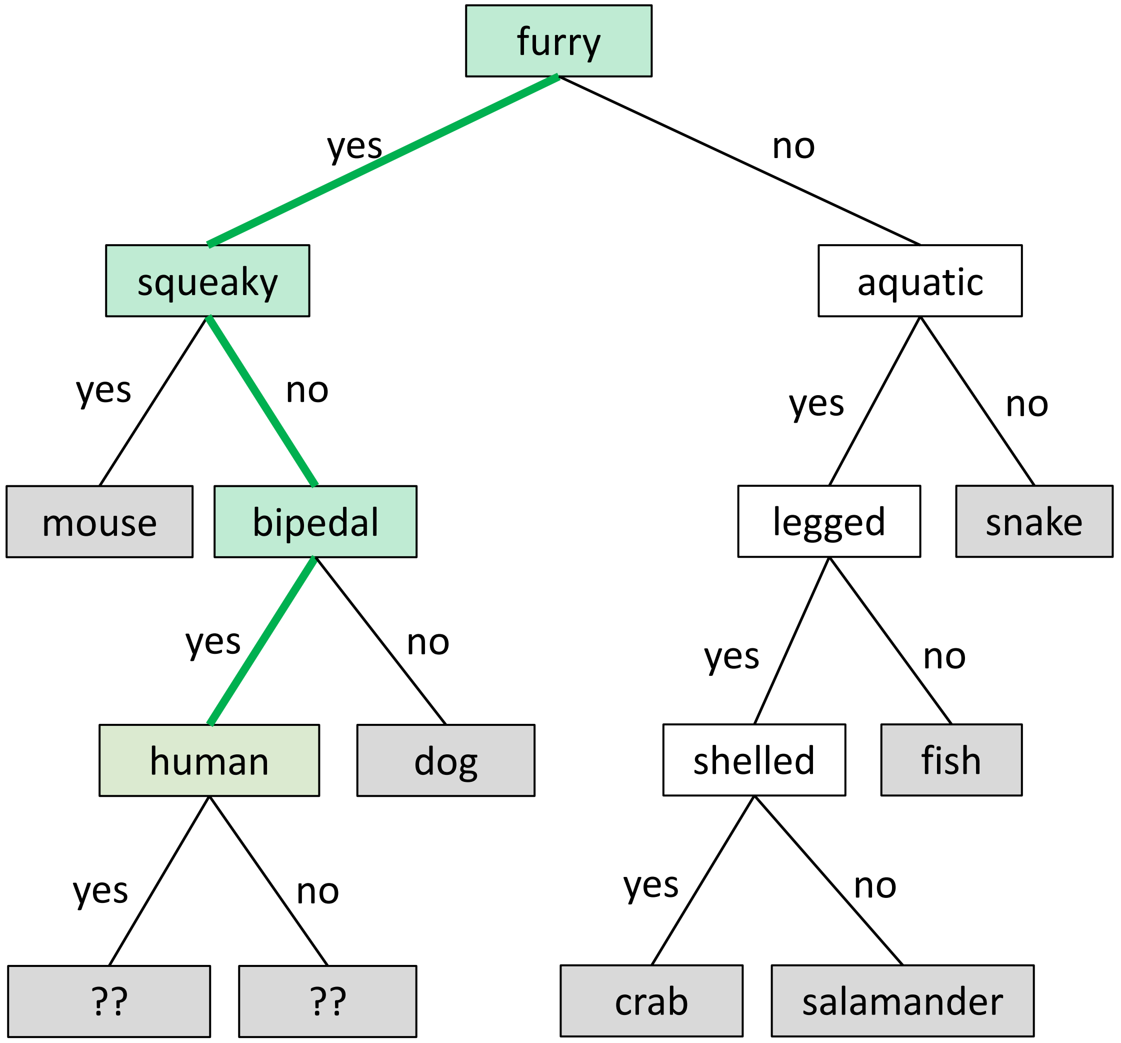
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   - 3.1. Print location in tree.
   - 3.2. Get name of new animal.
   - 3.3. Get distinguishing characteristic.
   - 3.4.



```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I don't know any furry, not squeaky, bipedal animals that aren't a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive
```

Program Execution:

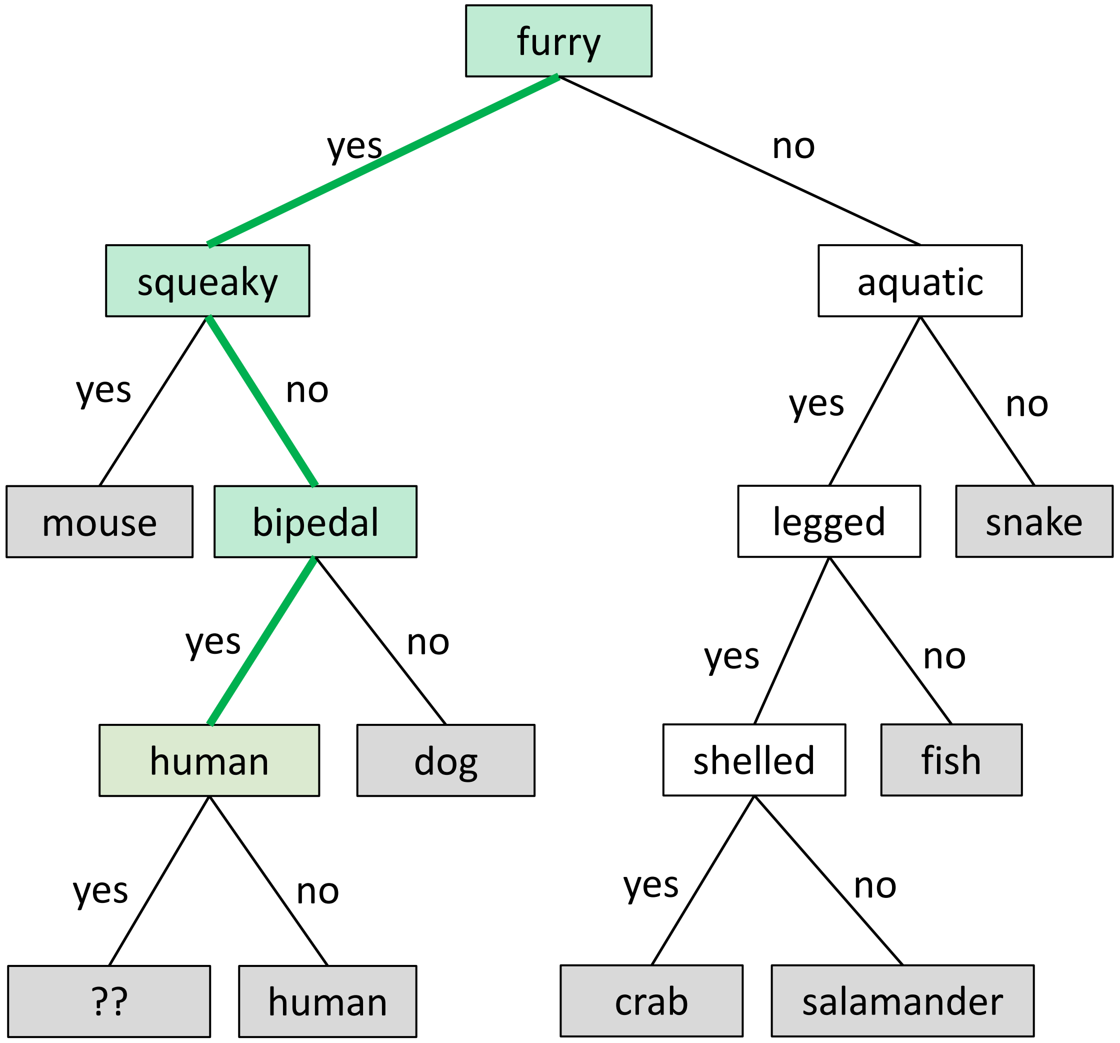
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   - 3.1. Print location in tree.
   - 3.2. Get name of new animal.
   - 3.3. Get distinguishing characteristic.
   - 3.4. Modify tree:
     - 3.4.1.



```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I don't know any furry, not squeaky, bipedal animals that aren't a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive
```

Program Execution:

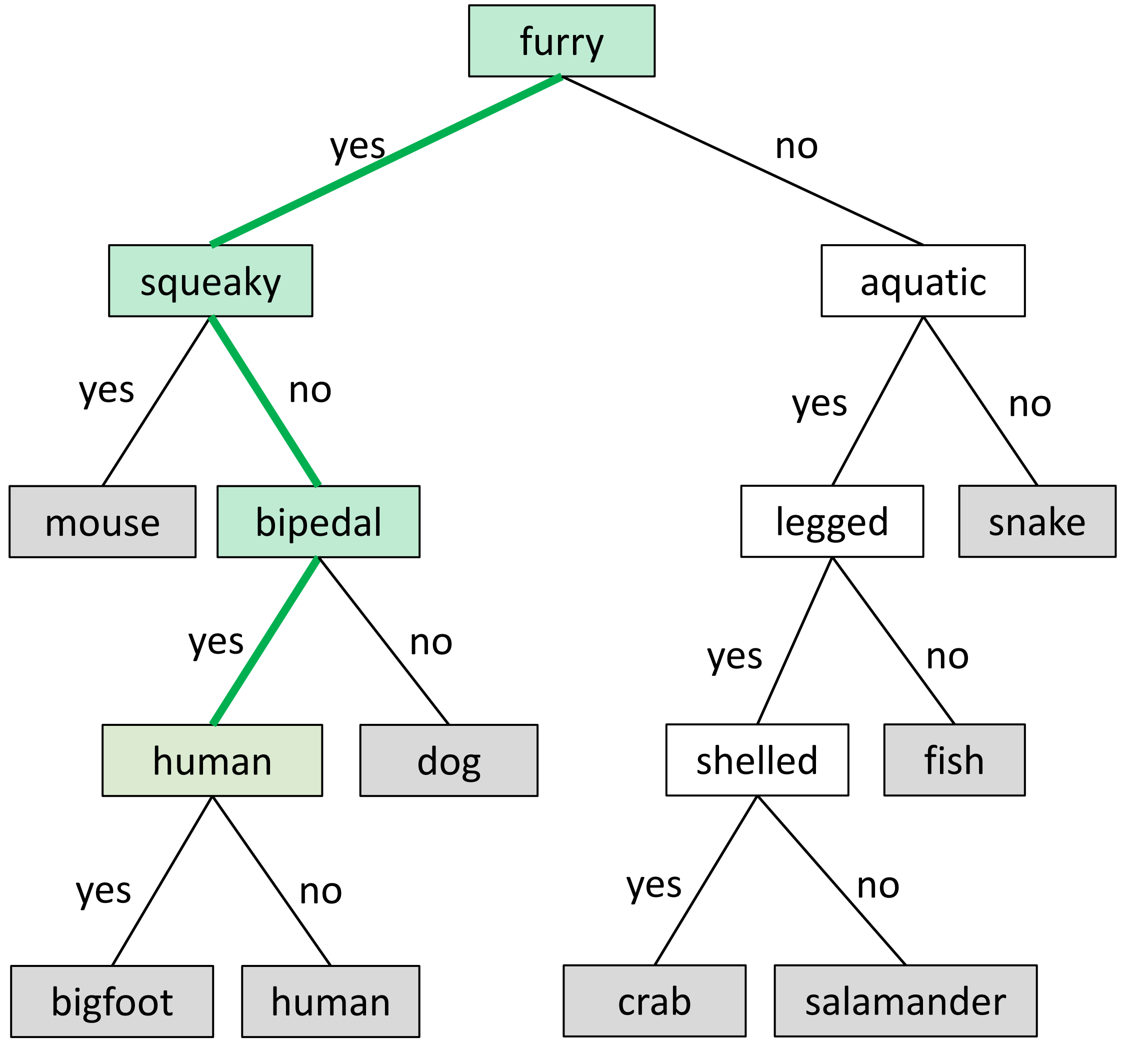
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   - 3.1. Print location in tree.
   - 3.2. Get name of new animal.
   - 3.3. Get distinguishing characteristic.
   - 3.4. Modify tree:
     - 3.4.1. Create two new child nodes at current leaf.
     - 3.4.2.



```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I don't know any furry, not squeaky, bipedal animals that aren't a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive
```

Program Execution:

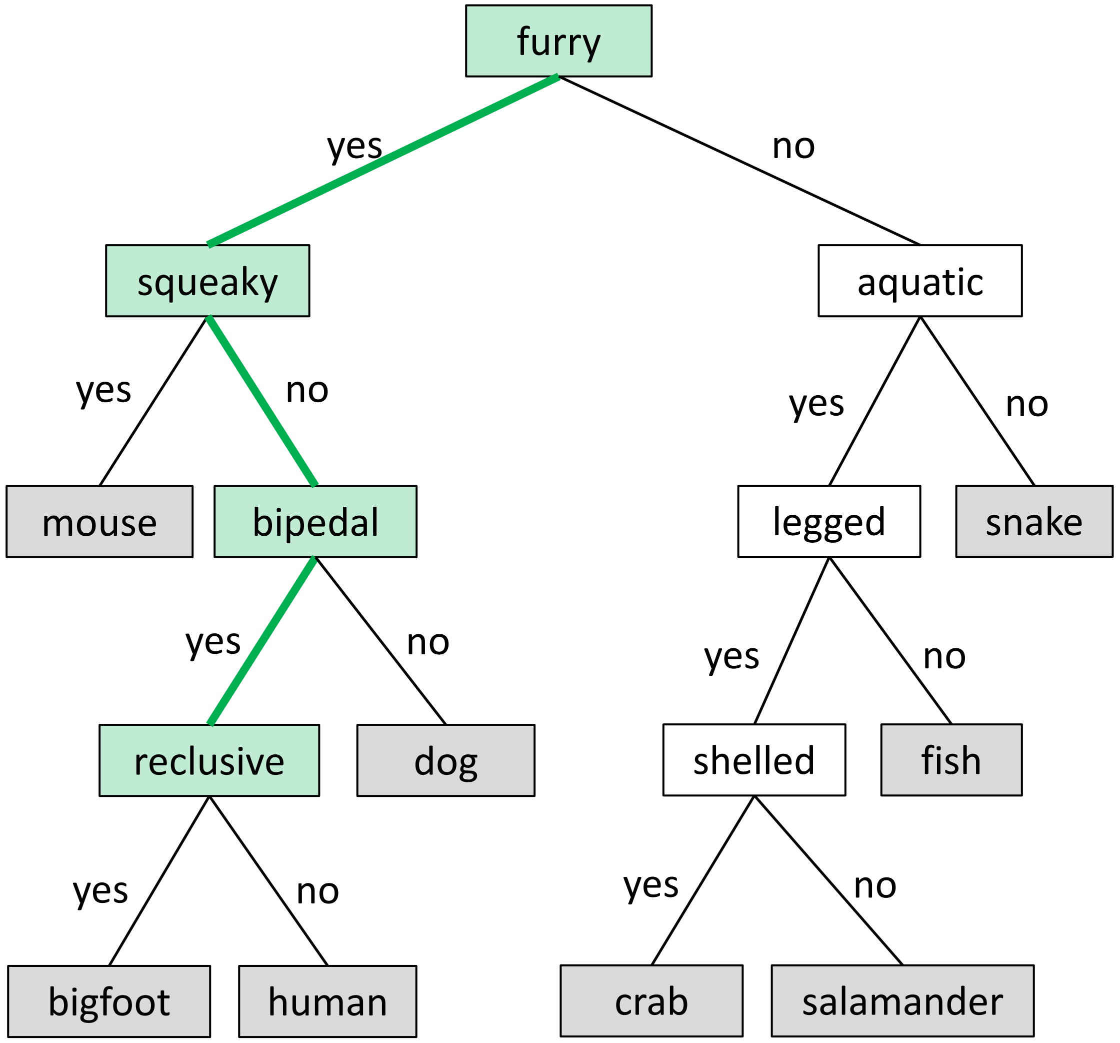
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   - 3.1. Print location in tree.
   - 3.2. Get name of new animal.
   - 3.3. Get distinguishing characteristic.
   - 3.4. Modify tree:
     - 3.4.1. Create two new child nodes at current leaf.
     - 3.4.2. Make "no" child node animal be old leaf.
     - 3.4.3.



```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I don't know any furry, not squeaky, bipedal animals that aren't a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive
```

Program Execution:

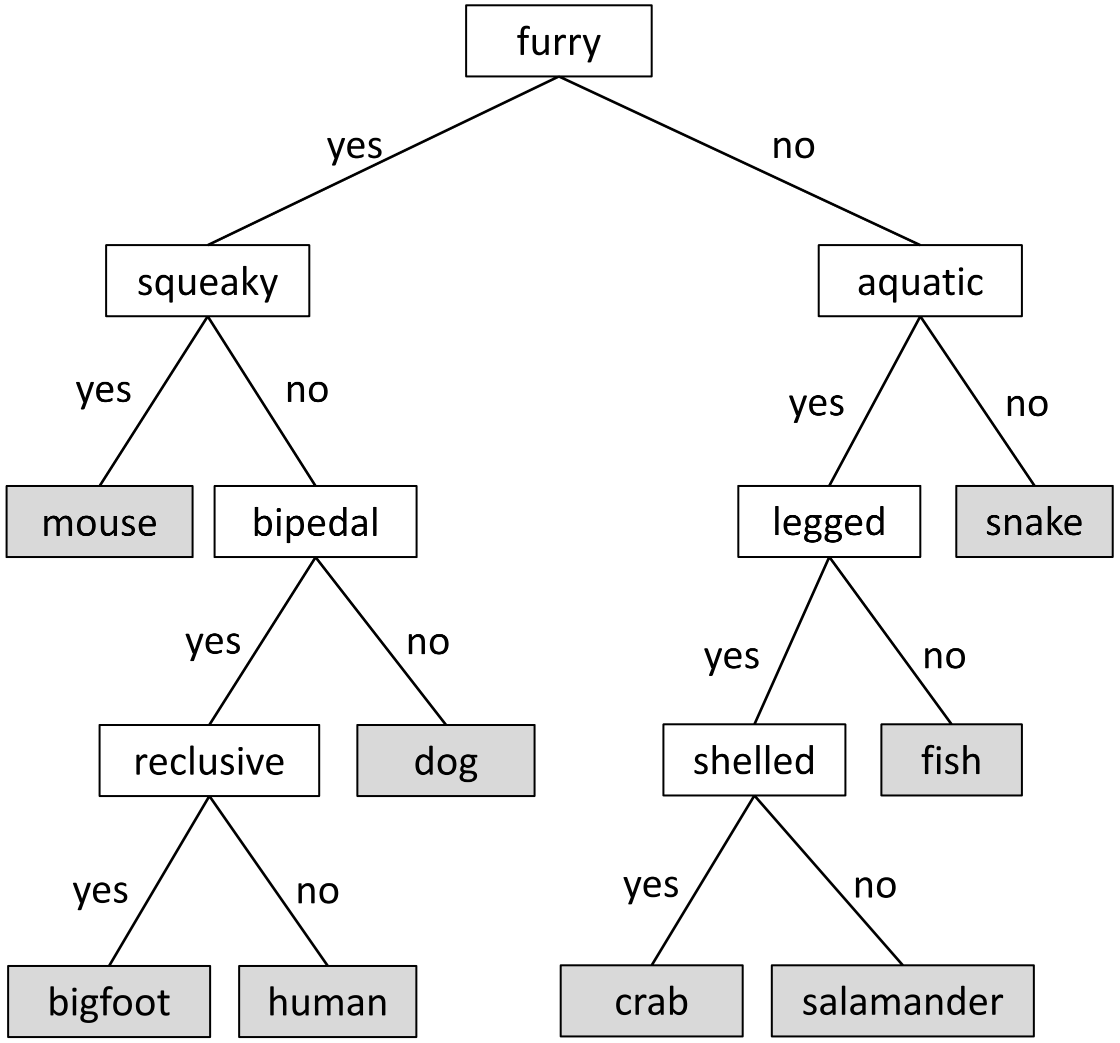
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   - 3.1. Print location in tree.
   - 3.2. Get name of new animal.
   - 3.3. Get distinguishing characteristic.
   - 3.4. Modify tree:
     - 3.4.1. Create two new child nodes at current leaf.
     - 3.4.2. Make "no" child node animal be old leaf.
     - 3.4.3. Make "yes" child node animal be new animal.
     - 3.4.4.



```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I don't know any furry, not squeaky, bipedal animals that aren't a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive
```

Program Execution:

1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   - 3.1. Print location in tree.
   - 3.2. Get name of new animal.
   - 3.3. Get distinguishing characteristic.
   - 3.4. Modify tree:
     - 3.4.1. Create two new child nodes at current leaf.
     - 3.4.2. Make "no" child node animal be old leaf.
     - 3.4.3. Make "yes" child node animal be new animal.
     - 3.4.4. Make old leaf be distinguishing characteristic.



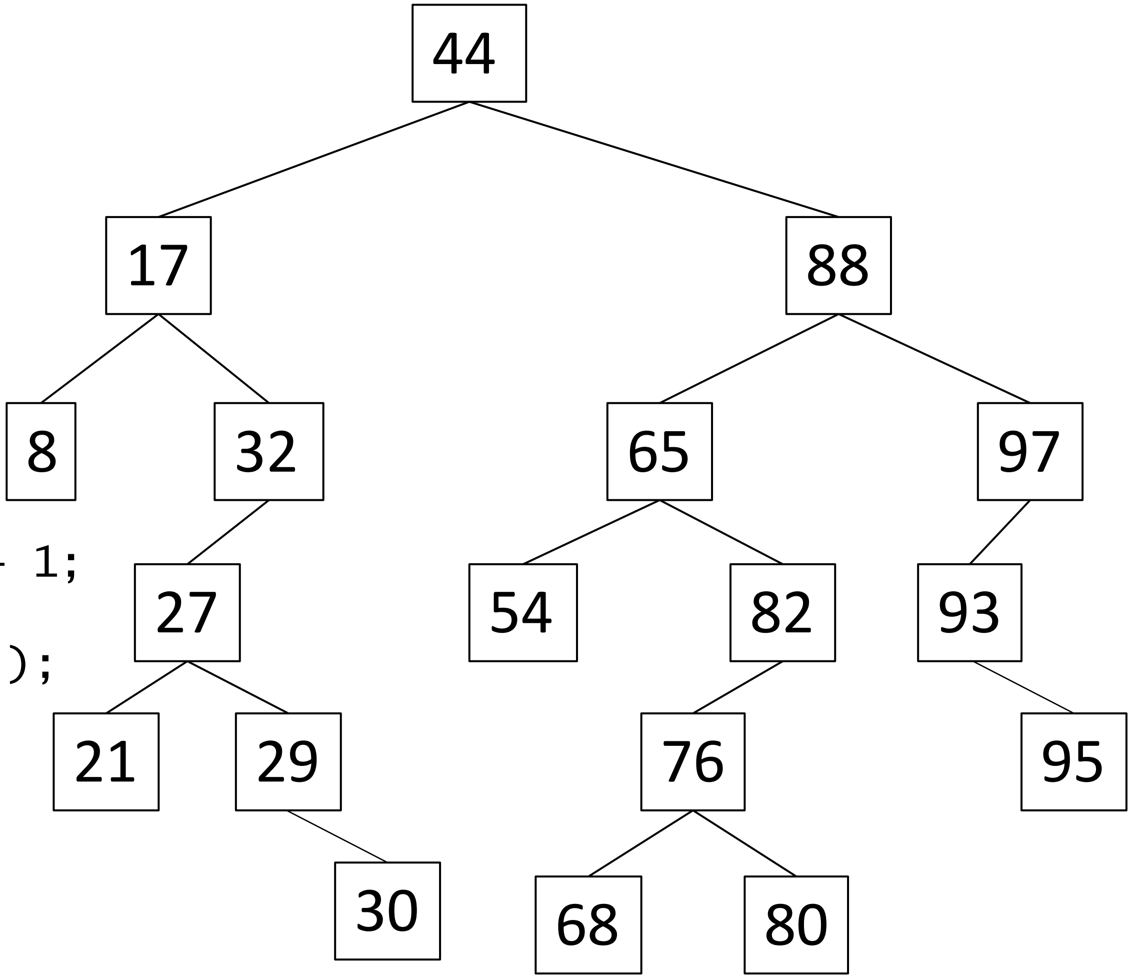
```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I don't know any furry, not squeaky, bipedal animals that aren't a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive
```

Program Execution:

1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
    - 3.1. Print location in tree.
    - 3.2. Get name of new animal.
    - 3.3. Get distinguishing characteristic.
    - 3.4. Modify tree:
        - 3.4.1. Create two new child nodes at current leaf.
        - 3.4.2. Make "no" child node animal be old leaf.
        - 3.4.3. Make "yes" child node animal be new animal.
        - 3.4.4. Make old leaf be distinguishing characteristic.

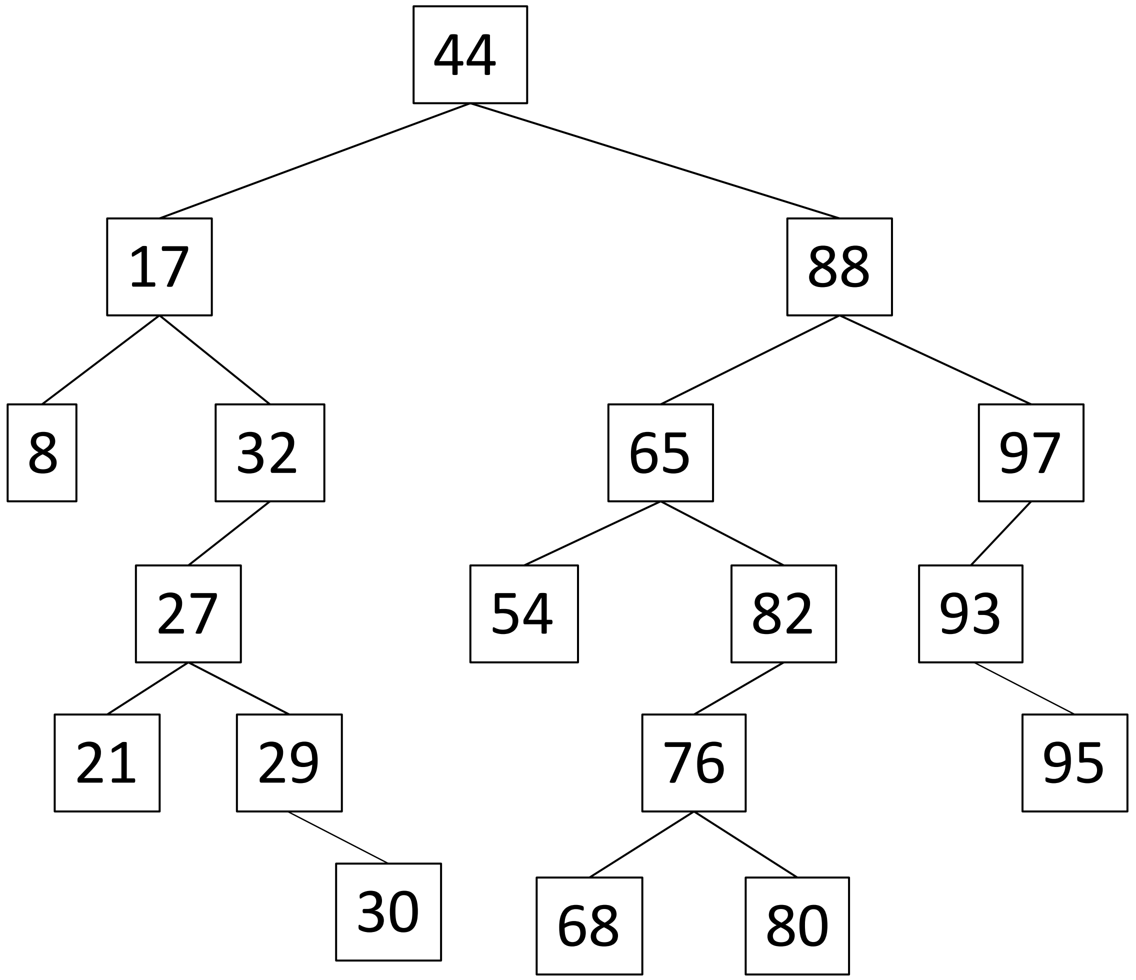
Binary Search Tree - Traversal

```
public void depthFirst() {
    Stack<Node> stack = new Stack<>();
    if (root != null) {
        stack.add(root);
        while(!stack.isEmpty()) {
            Node node = stack.pop();
            System.out.println(node.getName());
            for (int i = node.getChildren().size() - 1;
                 i >= 0; i--) {
                stack.push(node.getChildren().get(i));
            }
        }
    }
}
```



Binary Search Tree - Traversal

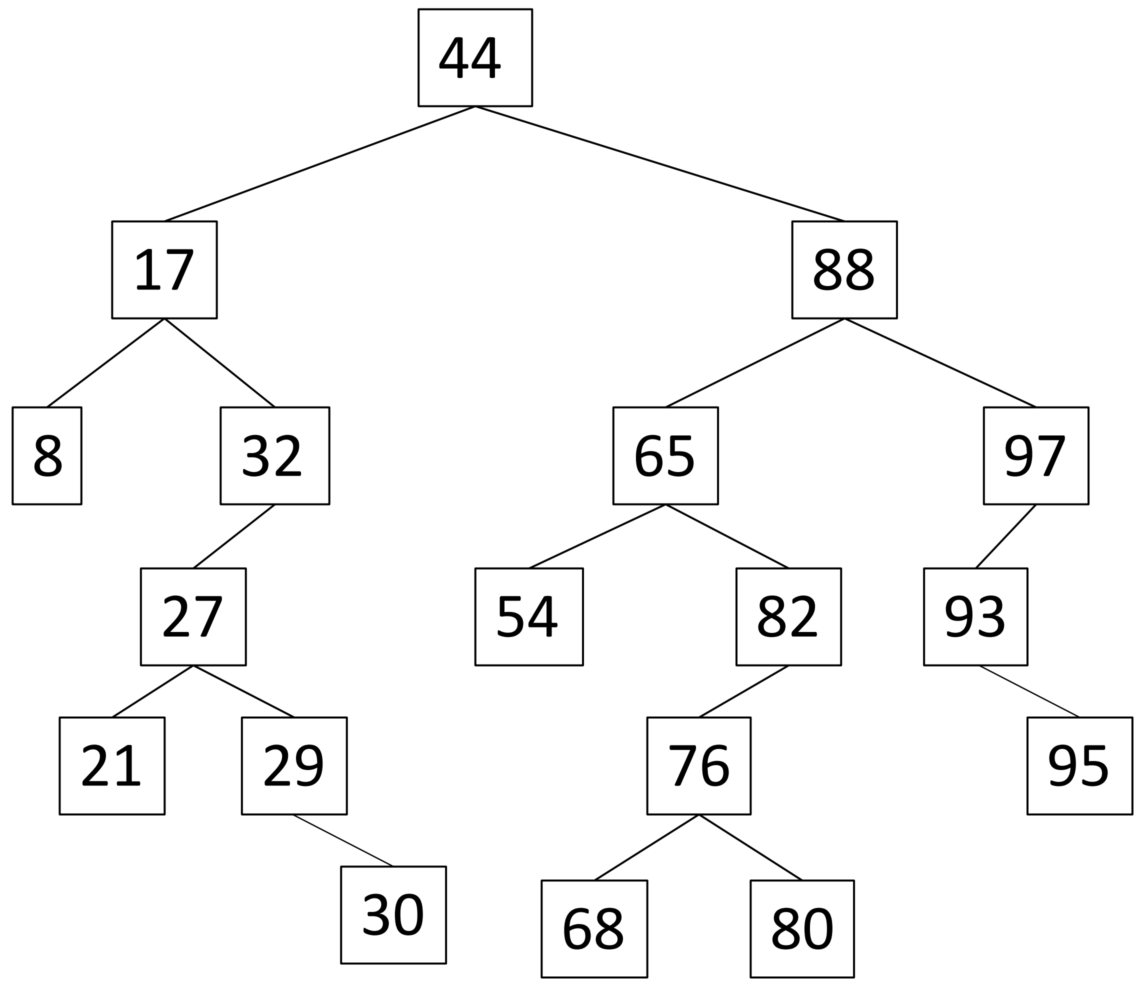
Recursion:



Binary Search Tree - Traversal

Recursion:

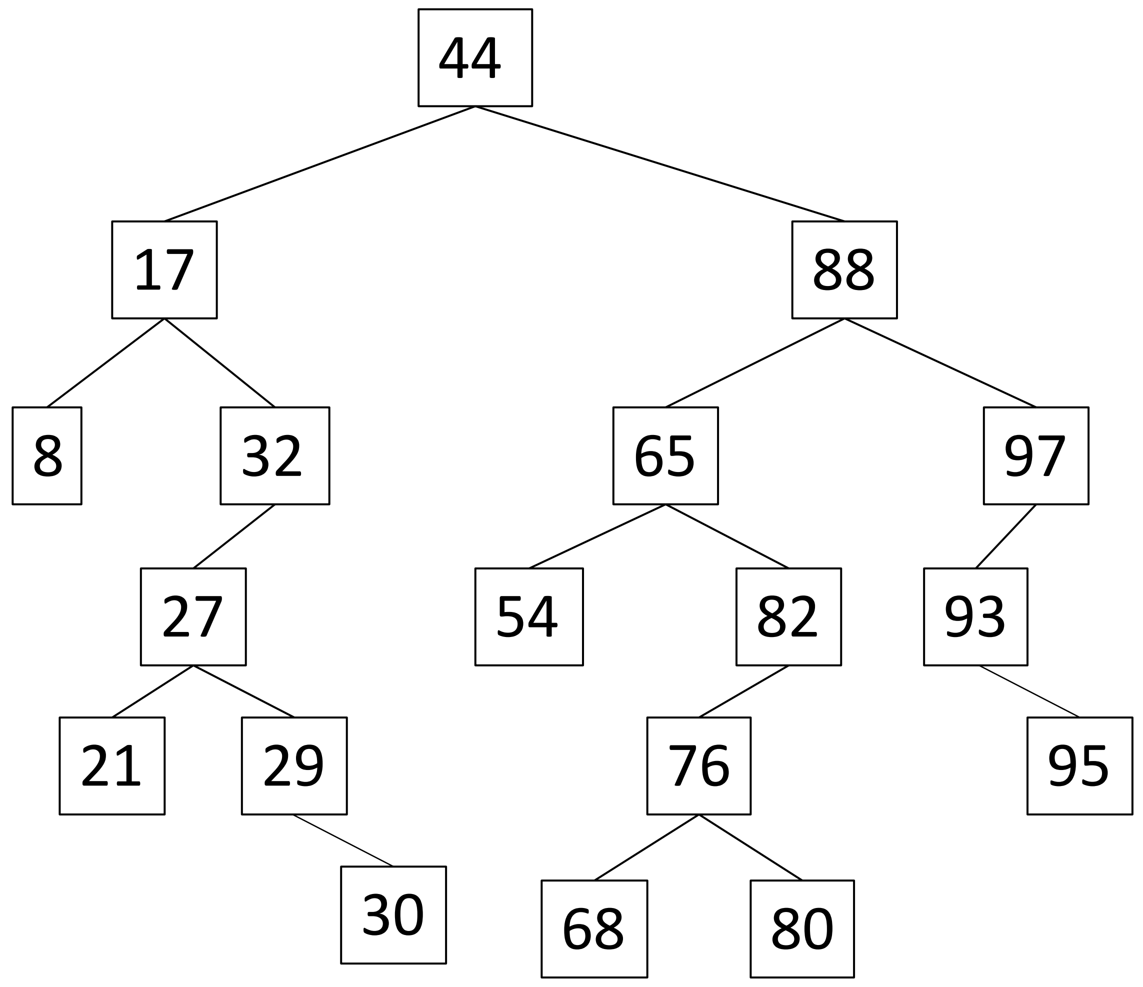
- **Calling a method from inside itself.**



Binary Search Tree - Traversal

Recursion:

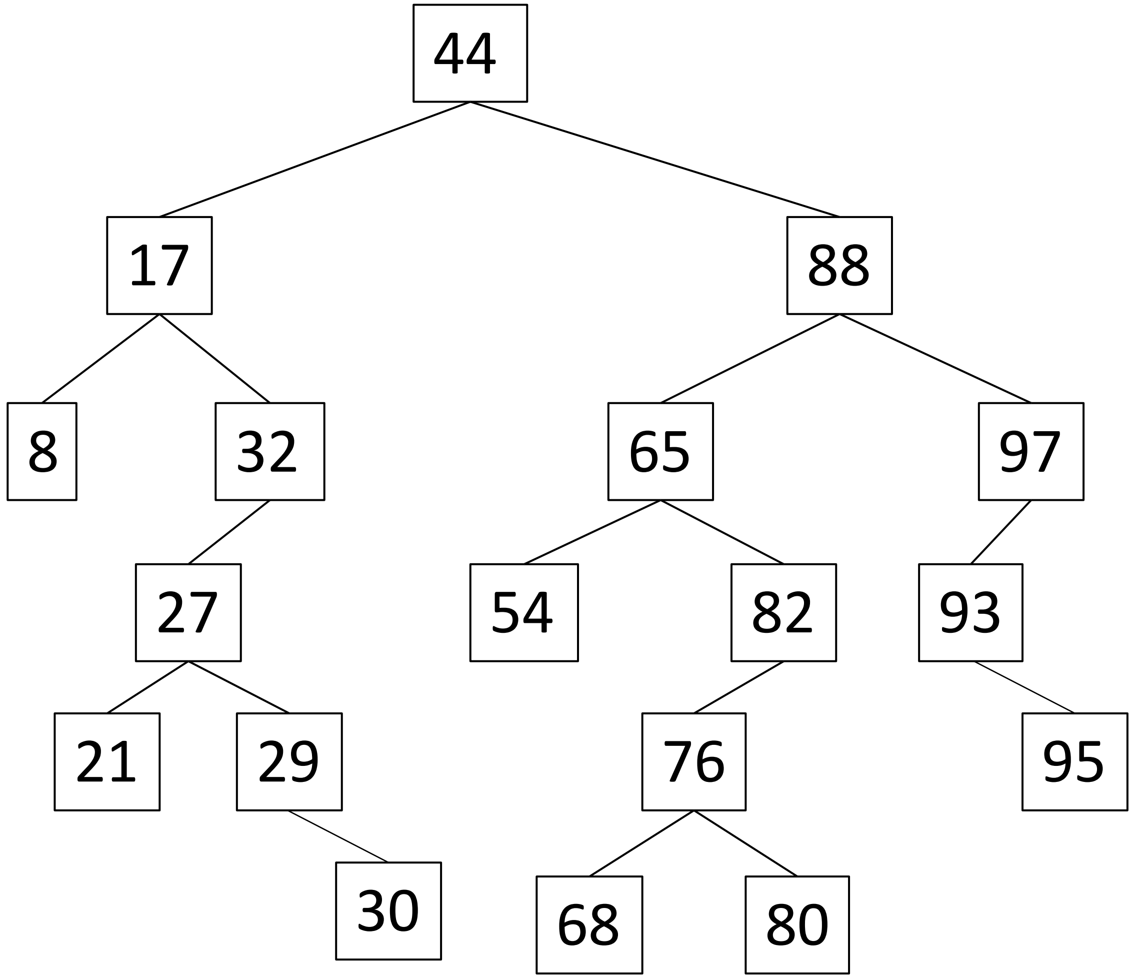
- **Calling a method from inside itself.**
- **Solve the problem by solving identical smaller problems.**



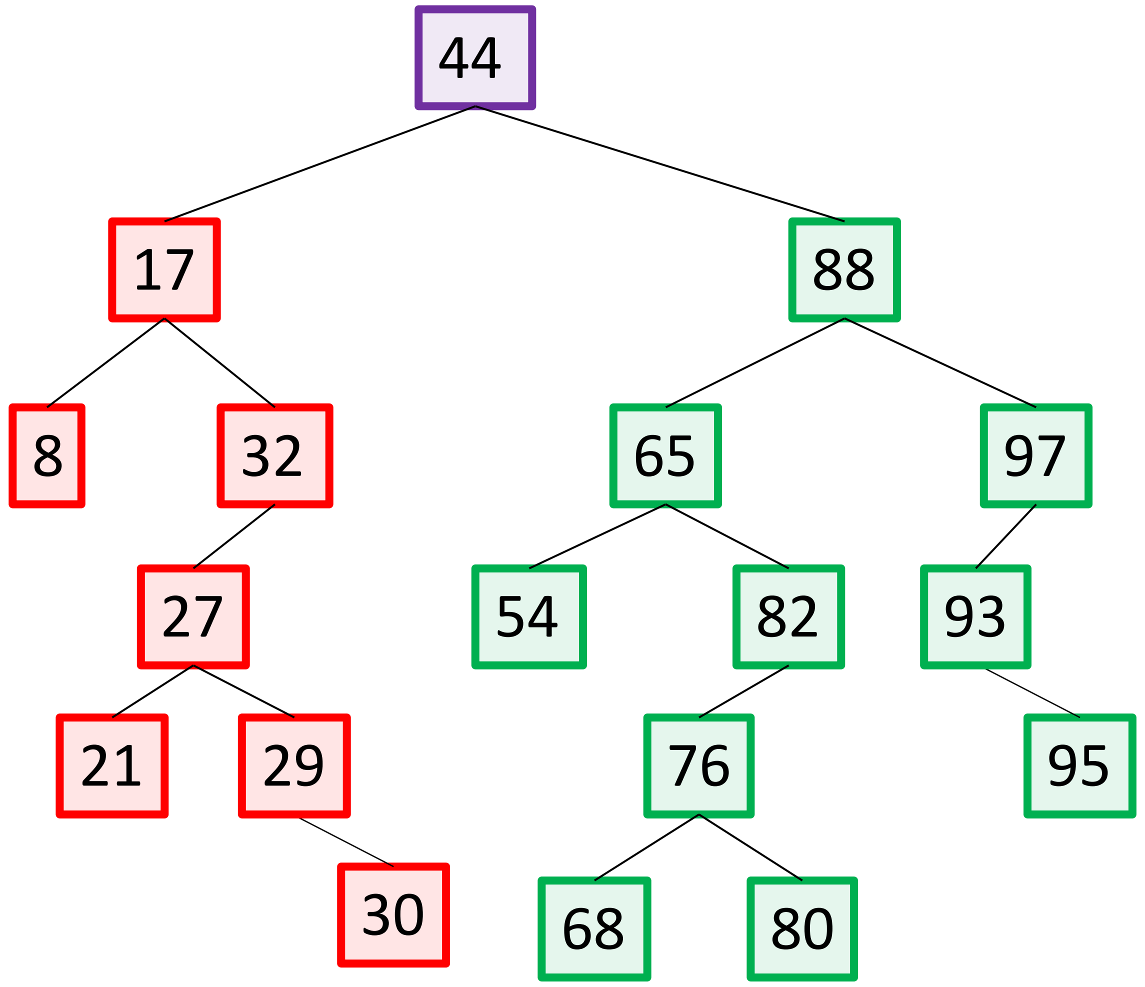
Binary Search Tree - Traversal

Recursion:

- **Calling a method from inside itself.**
- **Solve the problem by solving identical smaller problems.**
- **What is the “smaller problem”?**



Binary Search Tree - Traversal



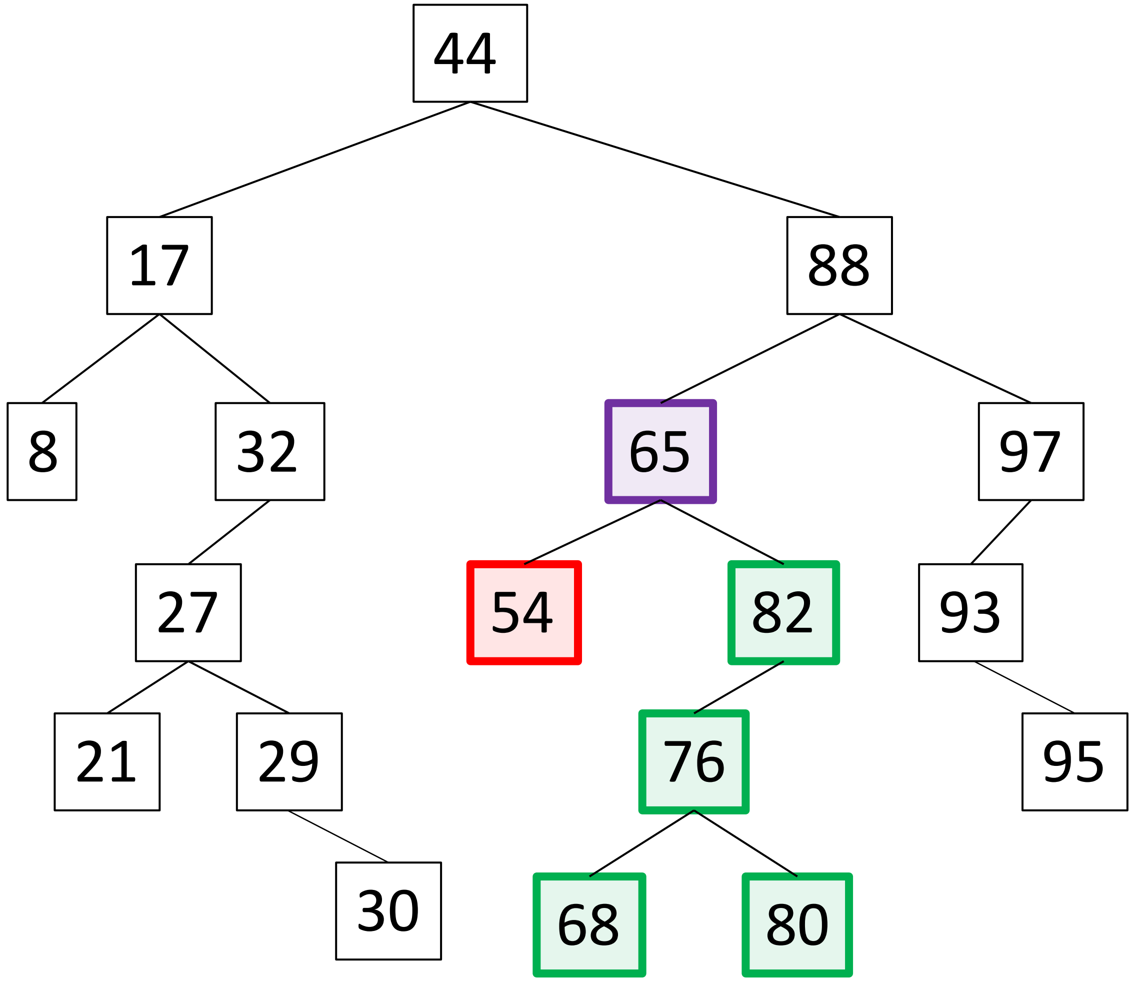
Recursion:

- **Calling a method from inside itself.**
- **Solve the problem by solving identical smaller problems.**
- **What is the "smaller problem"?**
    - **Process the left side, then process the right side.**

Binary Search Tree - Traversal

Recursion:

- **Calling a method from inside itself.**
- **Solve the problem by solving identical smaller problems.**
- **What is the "smaller problem"?**
  - **Process the left side, then process the right side.**

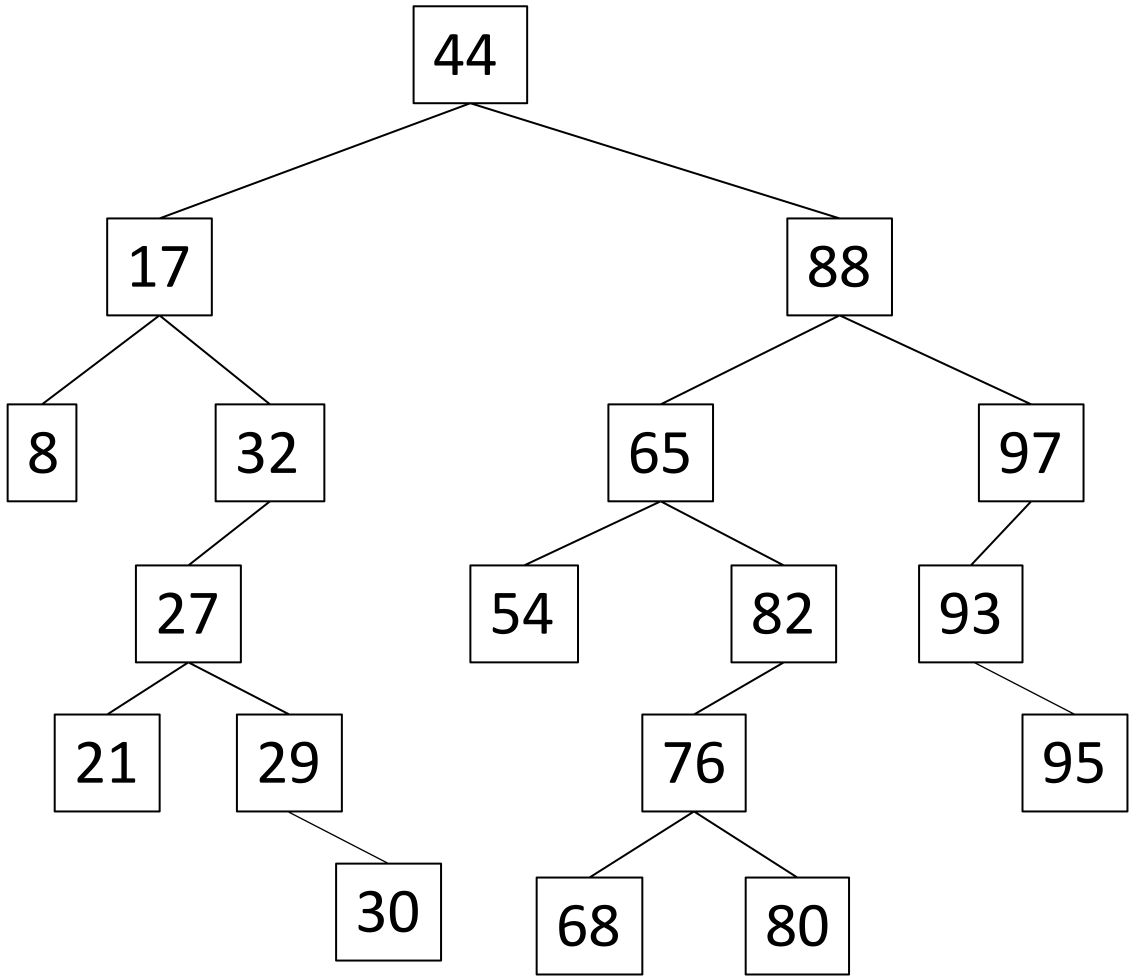


Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

Recursion:

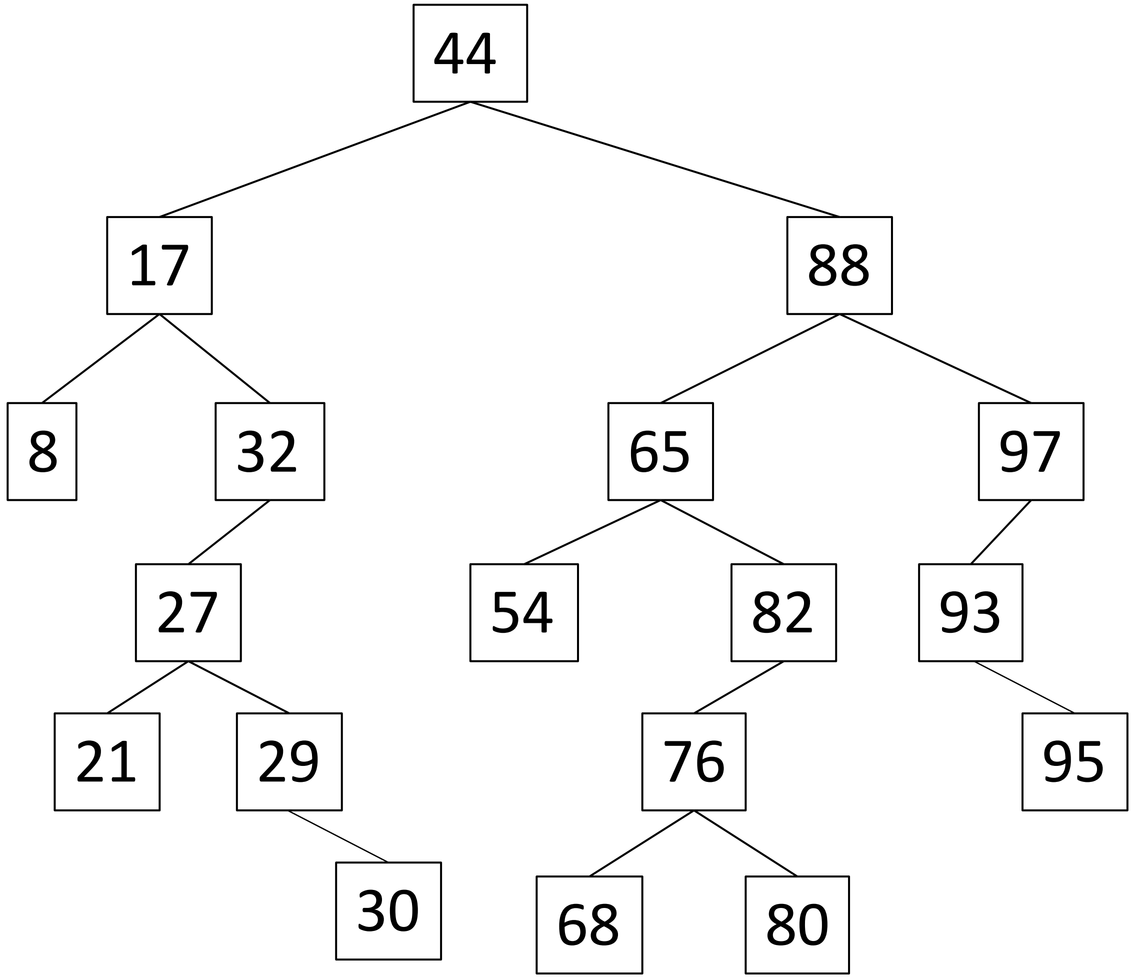
- **Calling a method from inside itself.**
- **Solve the problem by solving identical smaller problems.**
- **What is the “smaller problem”?**
  - **Process the left side, then process the right side.**



Output:

Binary Search Tree - Traversal

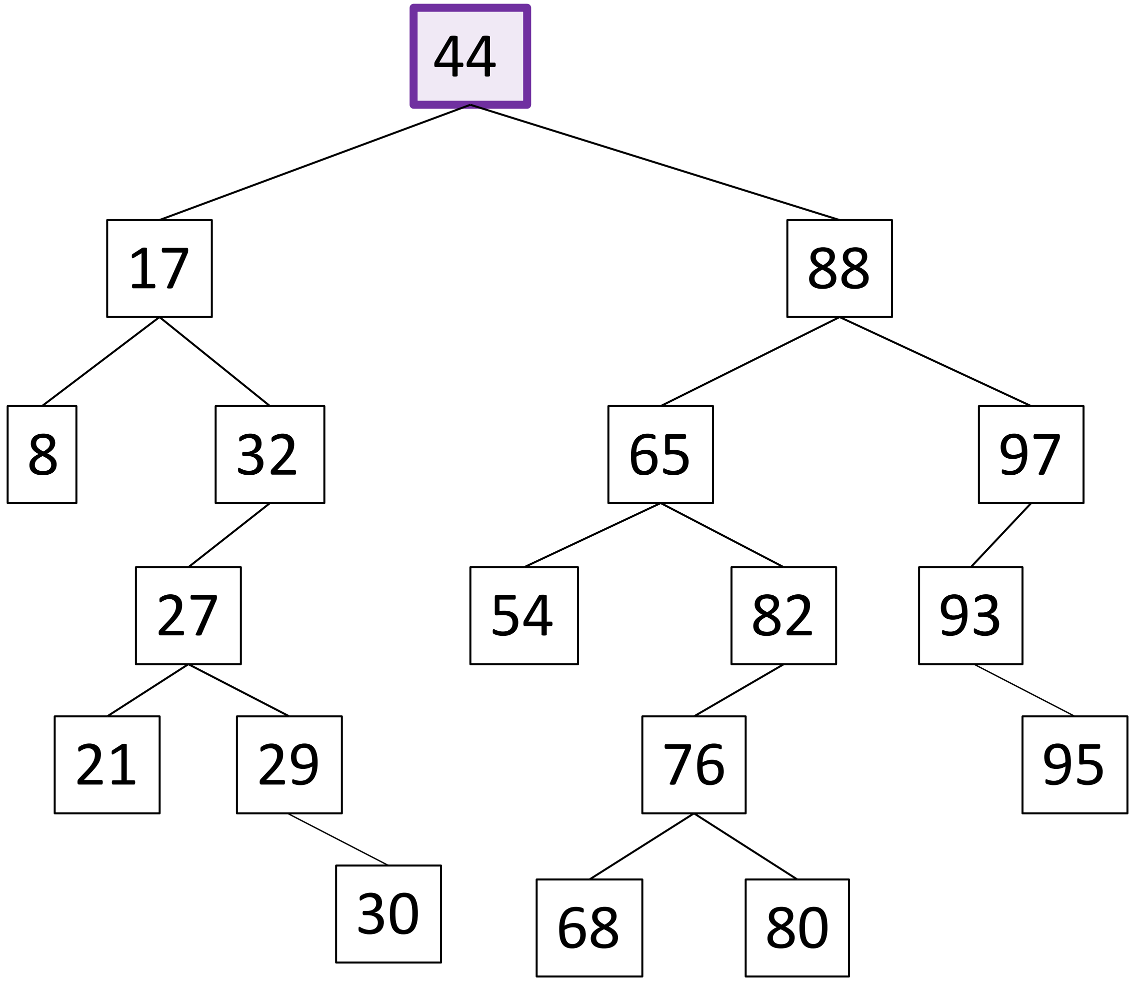
```
public void depthFirst(Node n) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:

Binary Search Tree - Traversal

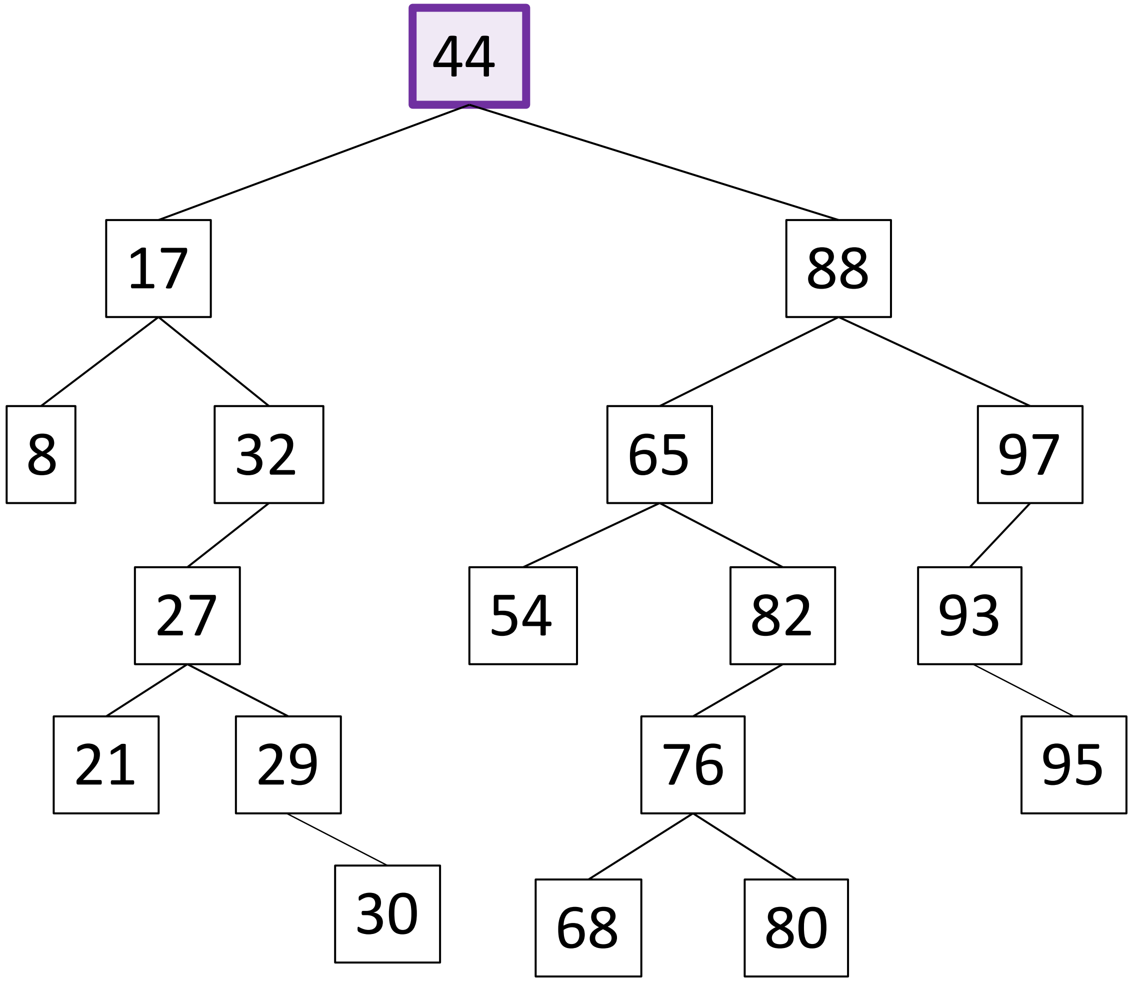
```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:
44

Binary Search Tree - Traversal

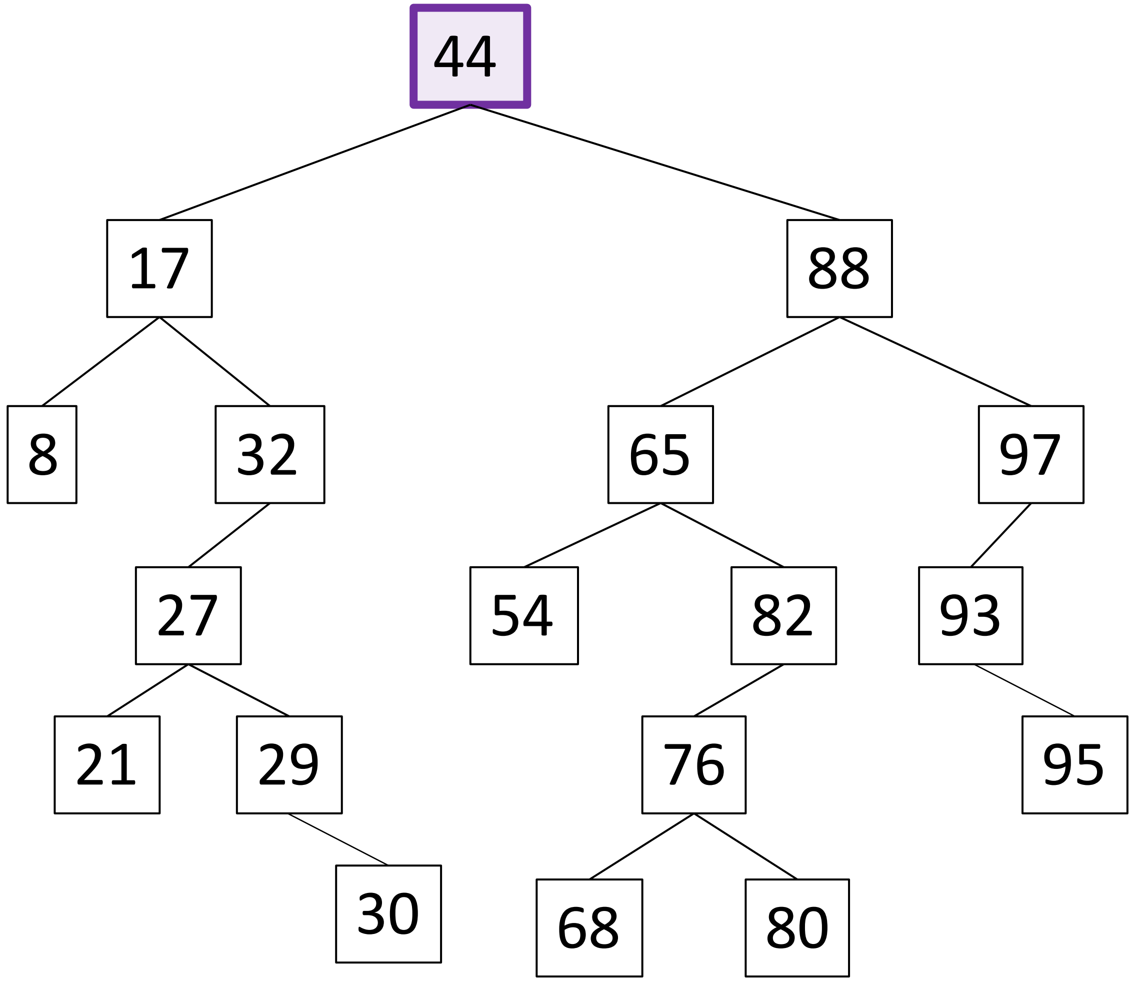
```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:
44

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

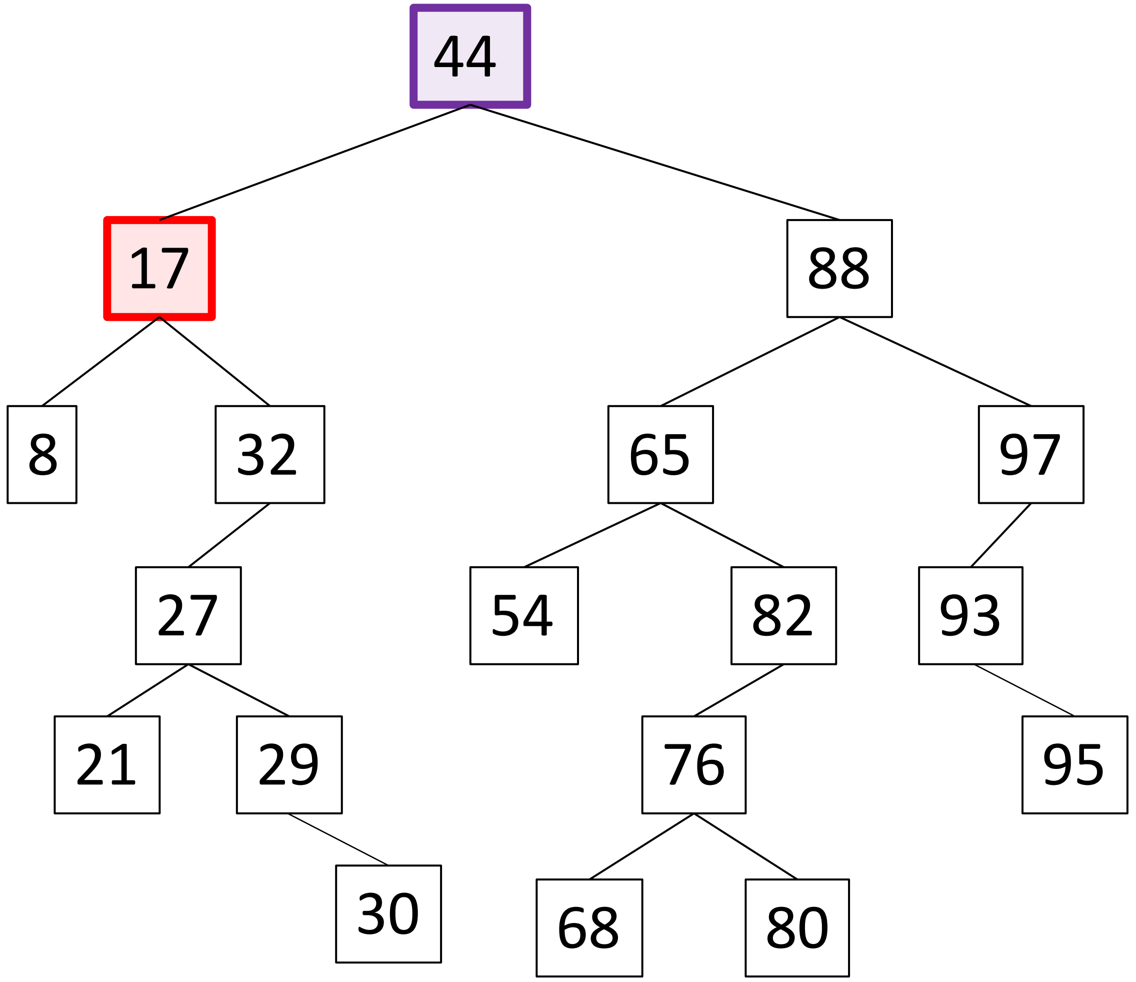


Output:
44

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

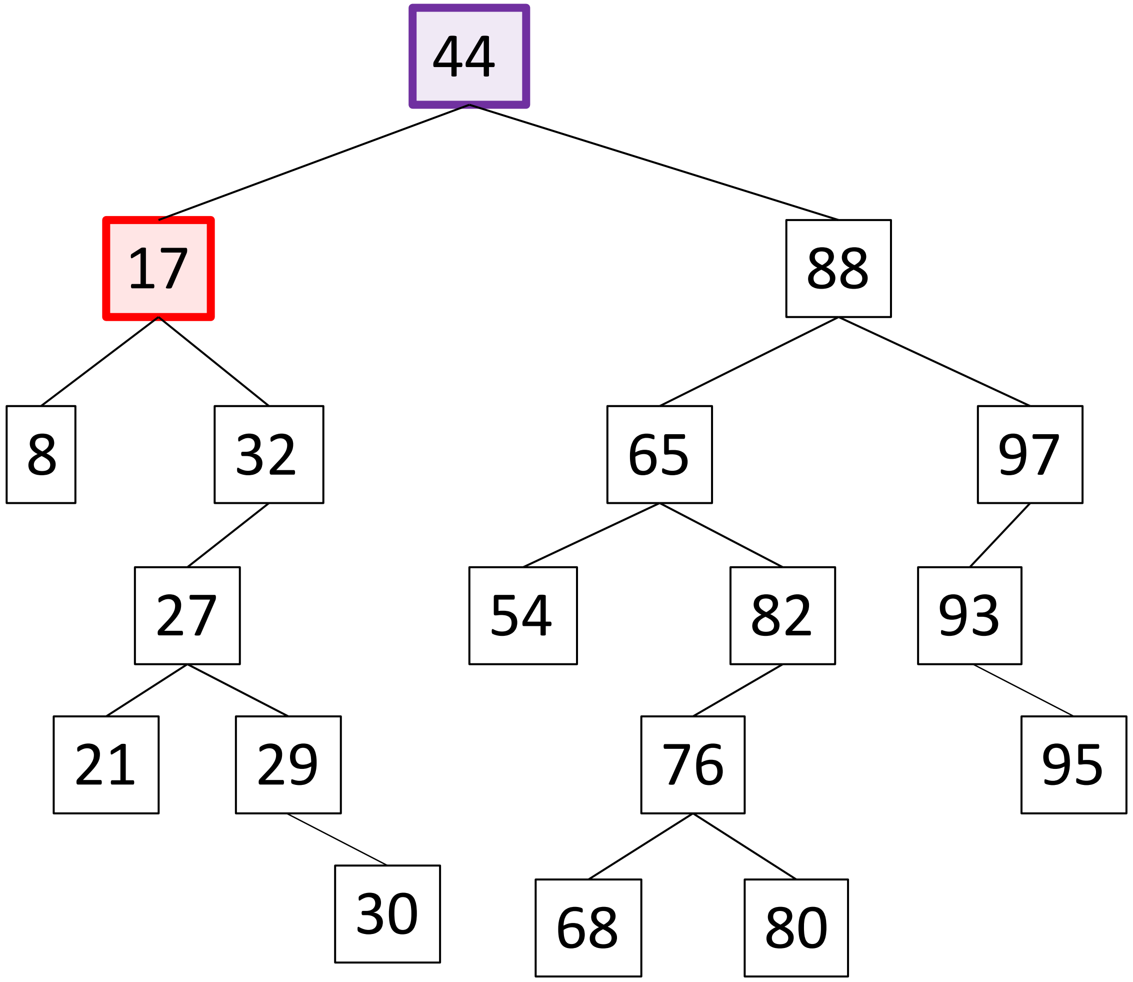


Output:
44
17

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:

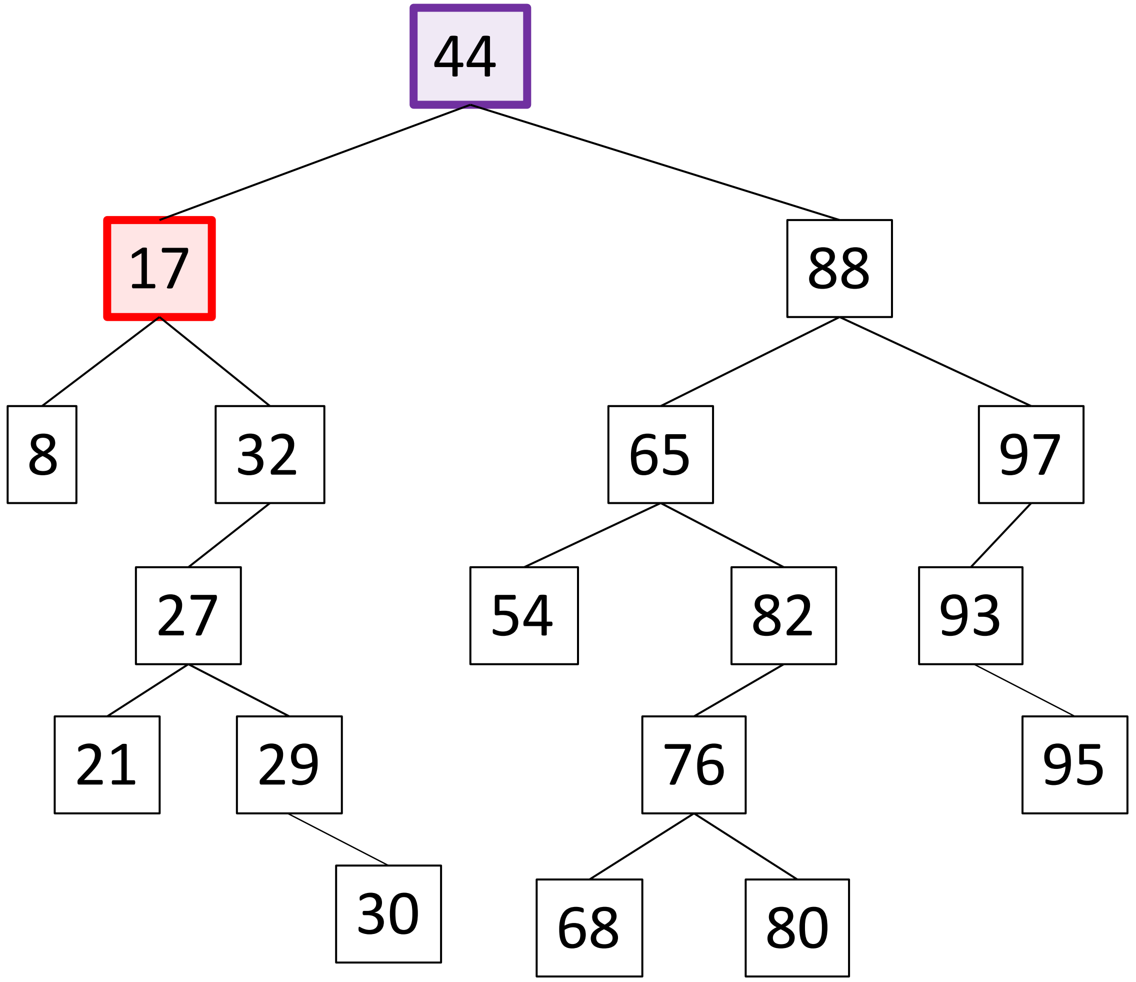
44

17

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



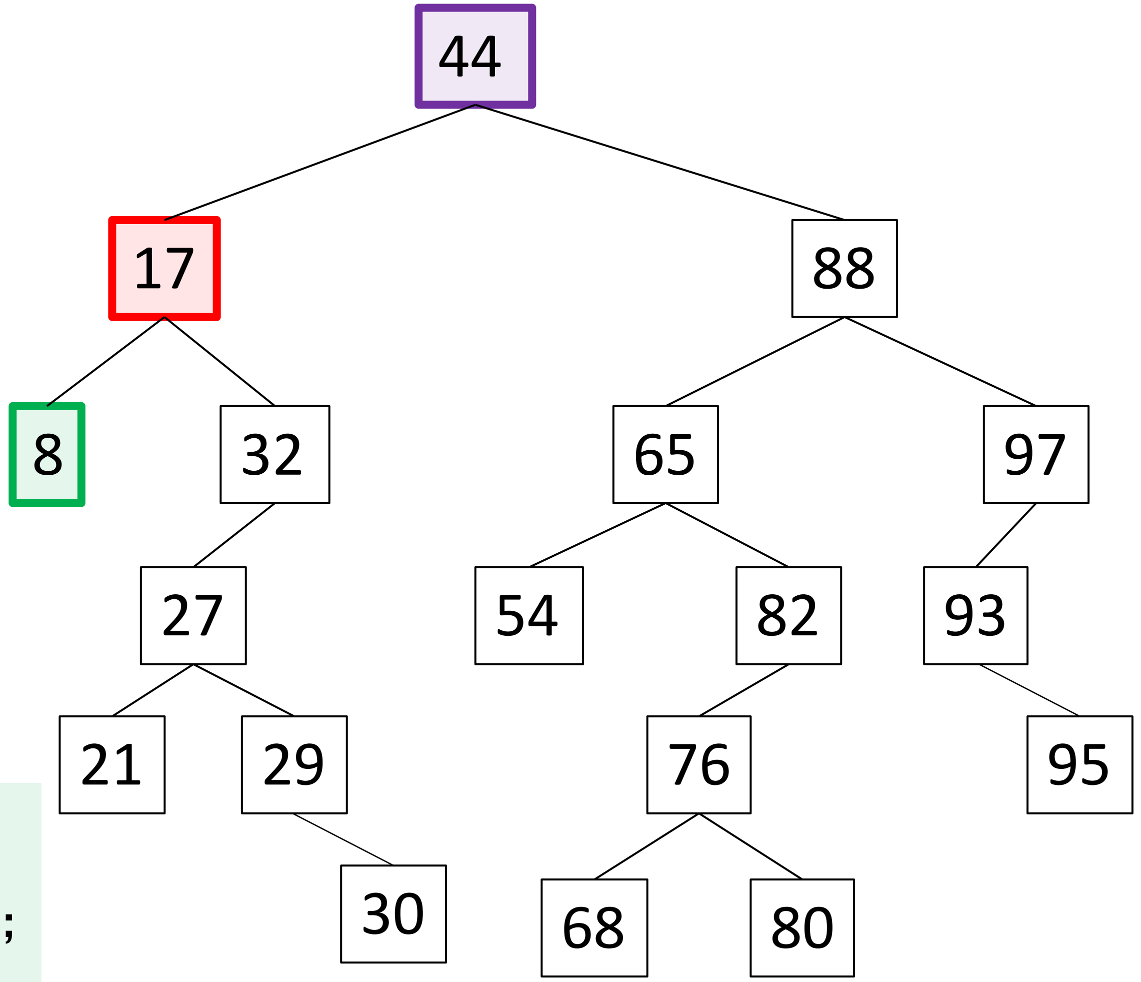
Output:
44
17

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



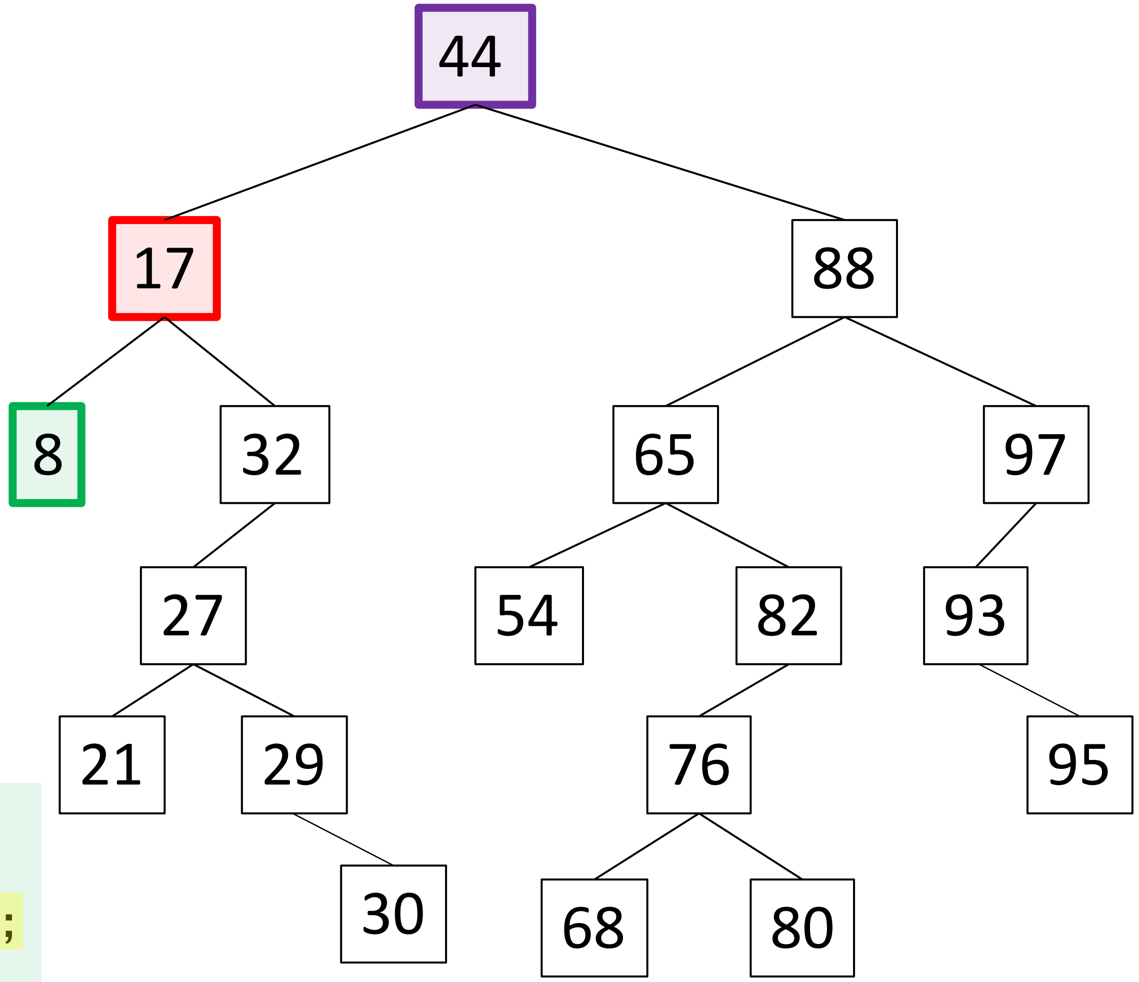
Output:
44
17
8

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Binary Search Tree - Traversal

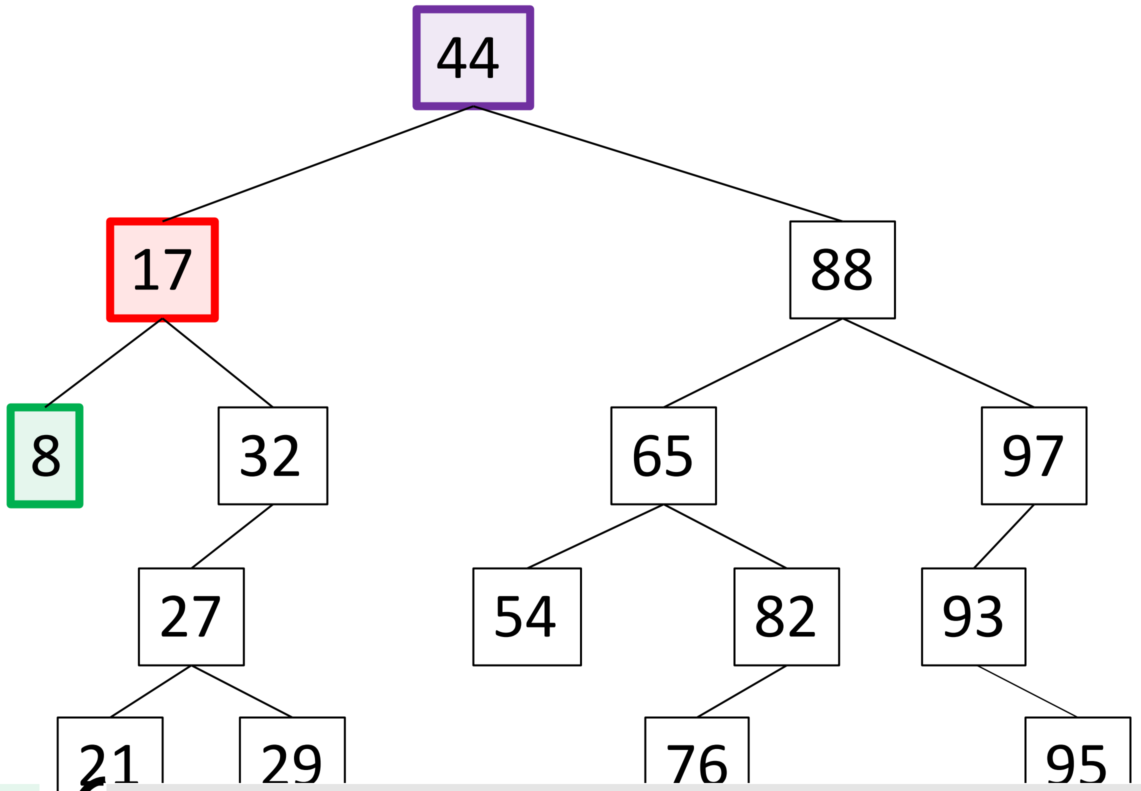
Output:
44
17
8

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(null) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Binary Search Tree - Traversal

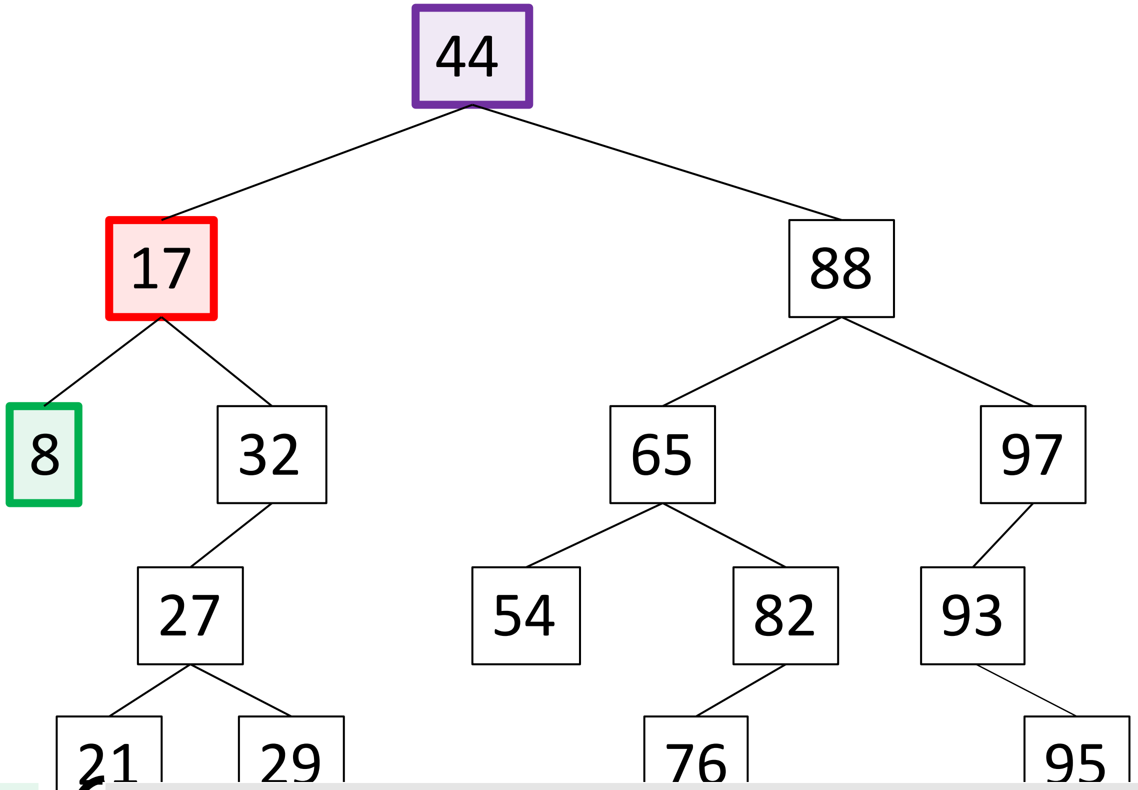
Output:
44
17
8

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(null) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:
44
17
8

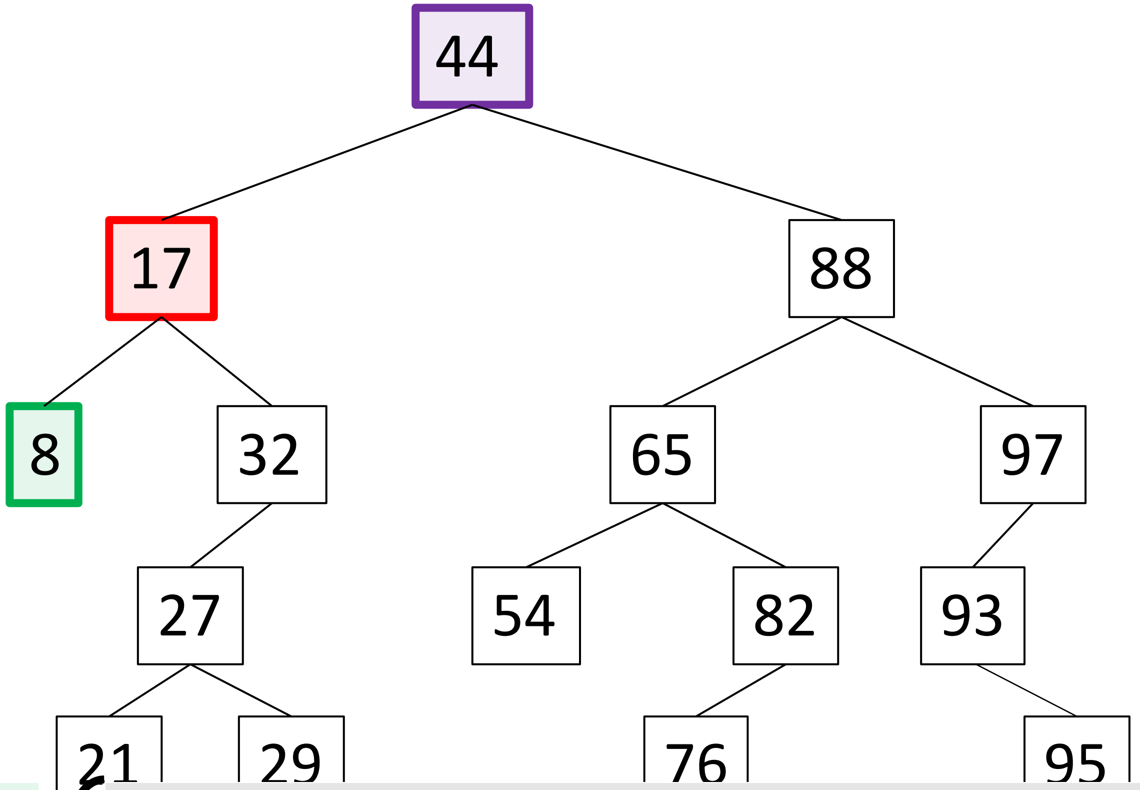
Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(null) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



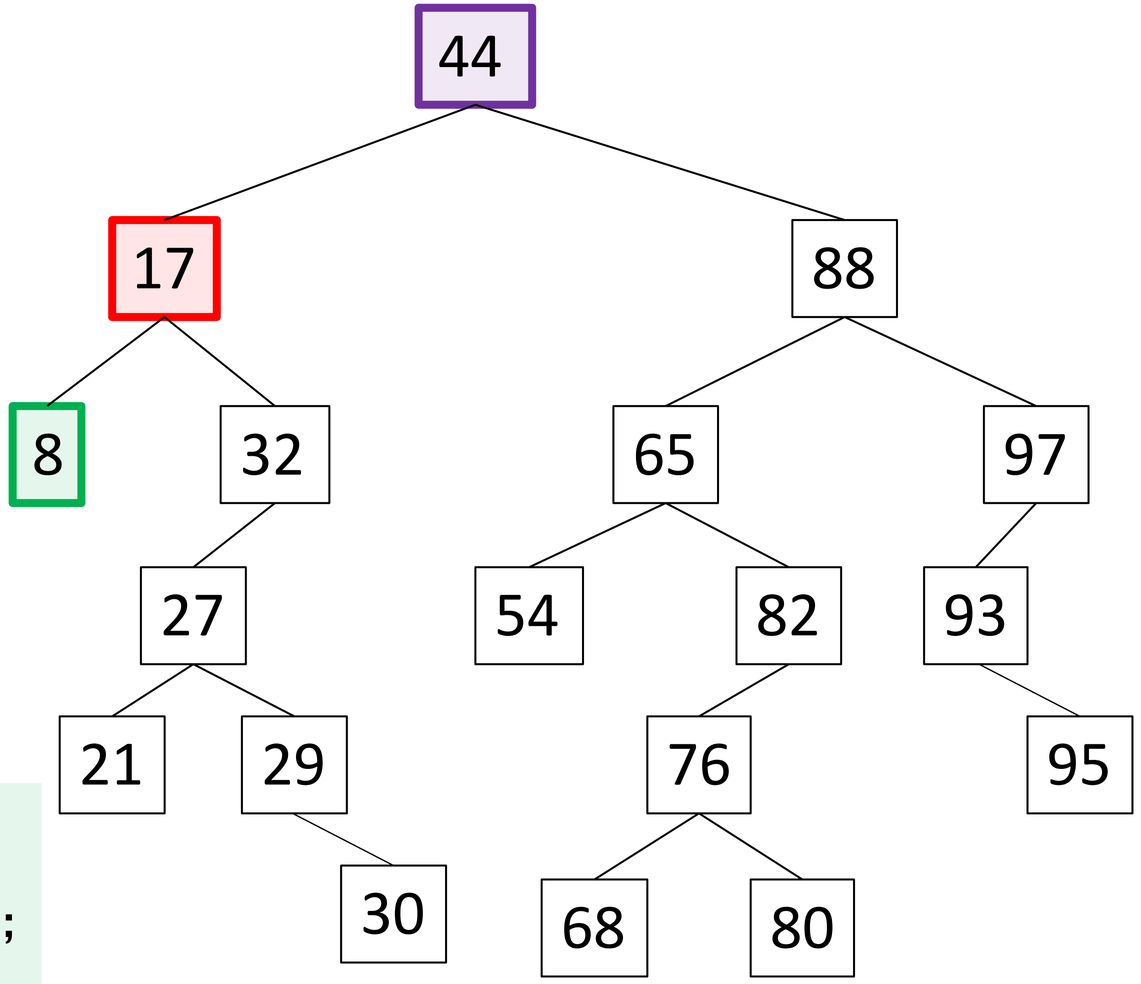
Output:
44
17
8

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:

44

17

8

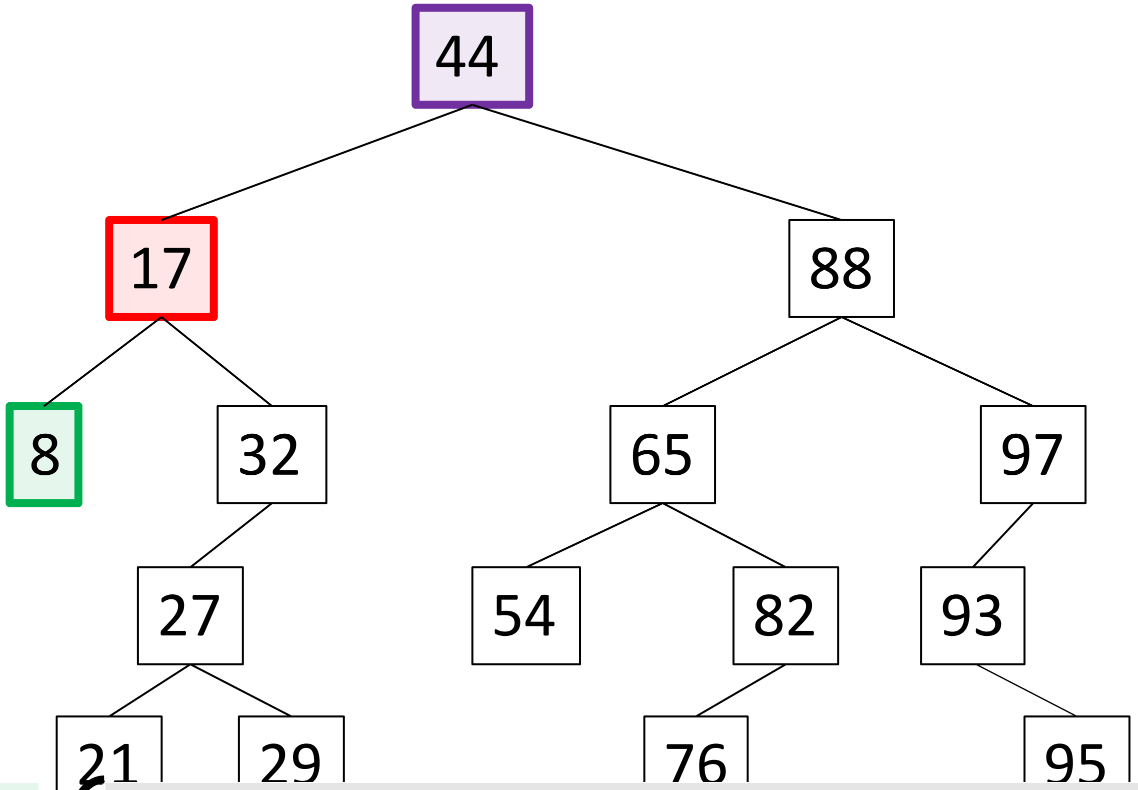
Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(null) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:

44

17

8

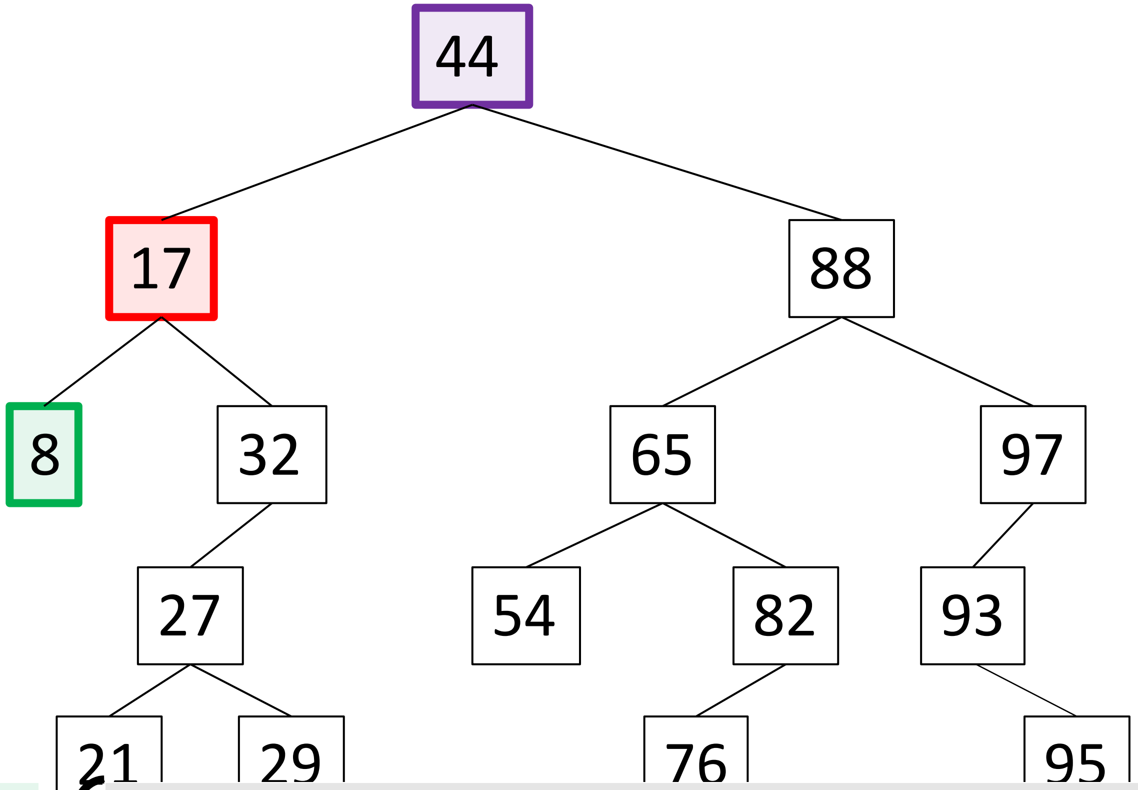
Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(null) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:
44
17
8

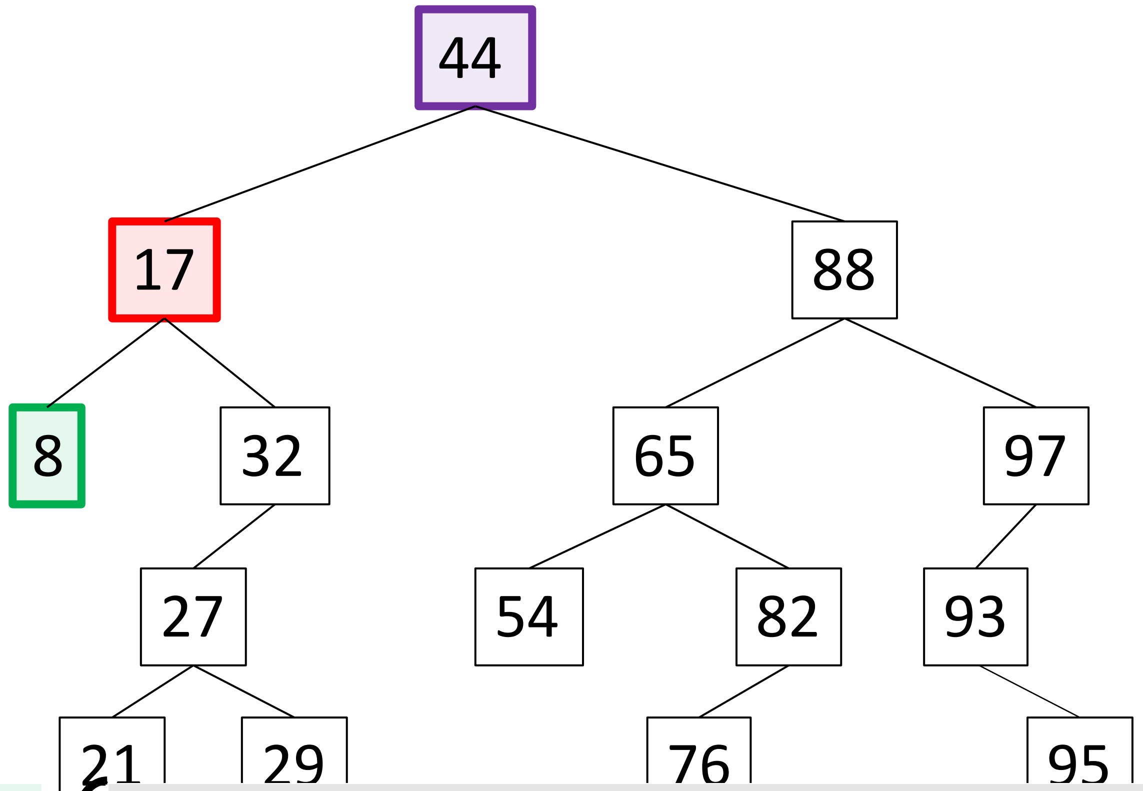
Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(null) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



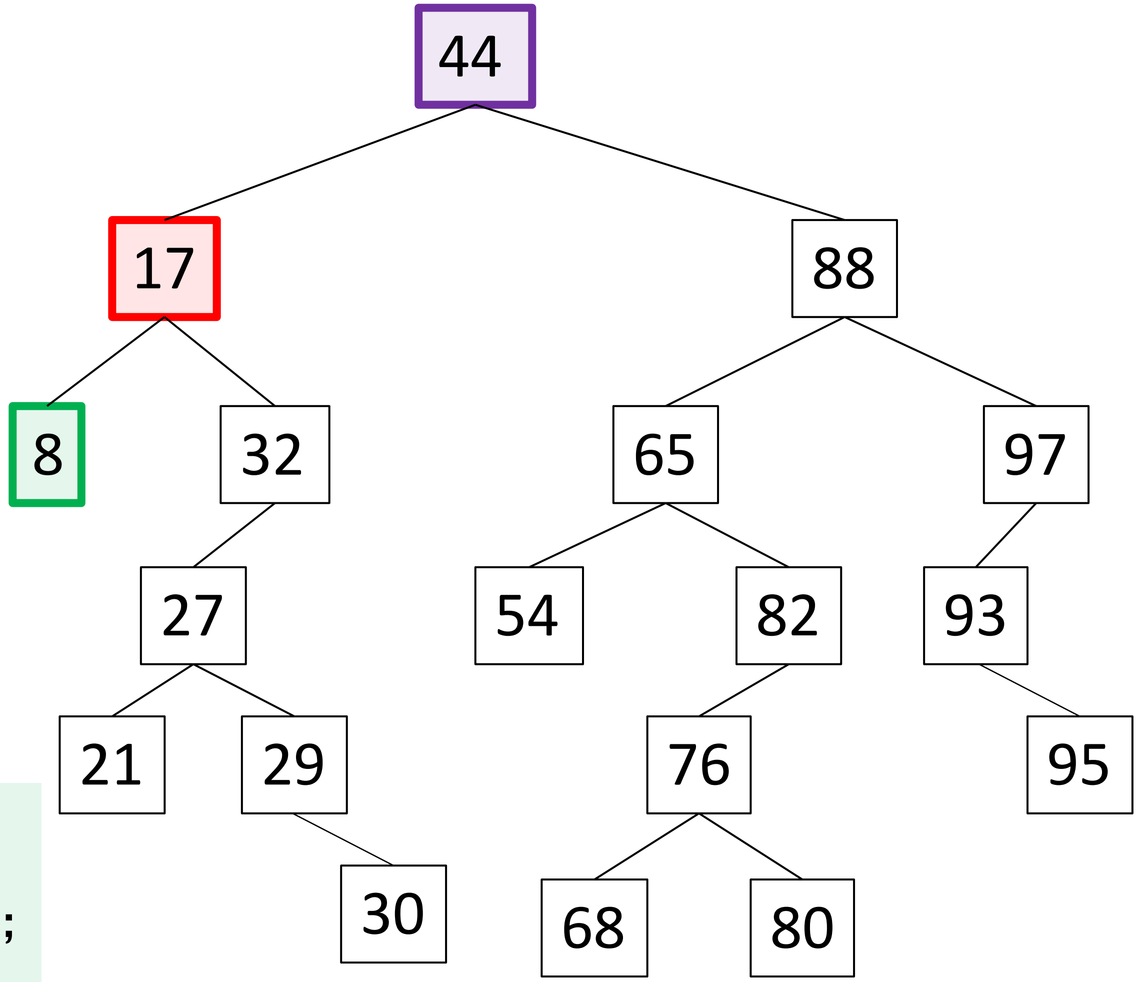
Output:
44
17
8

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



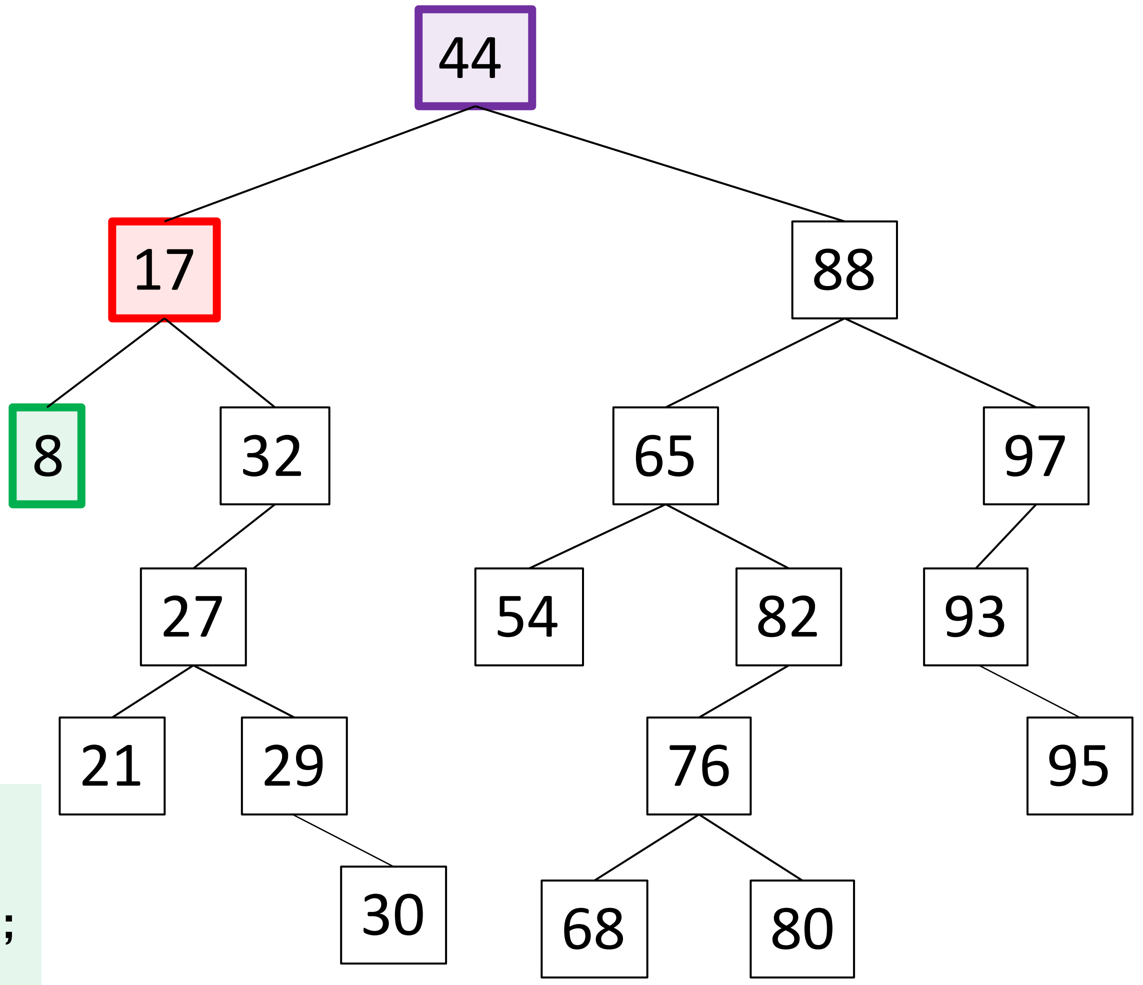
Binary Search Tree - Traversal

Output:
44
17
8

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



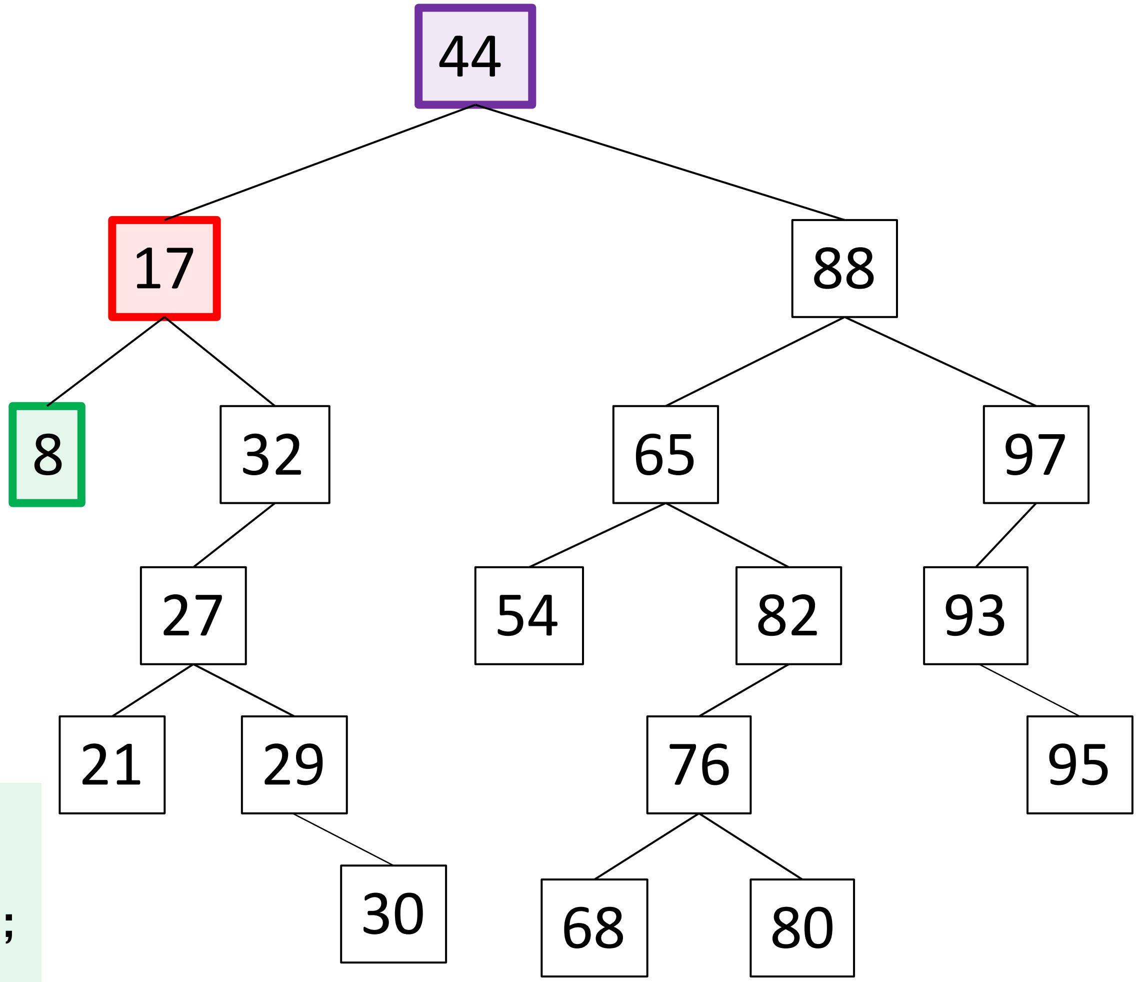
Output:
44
17
8

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:

44

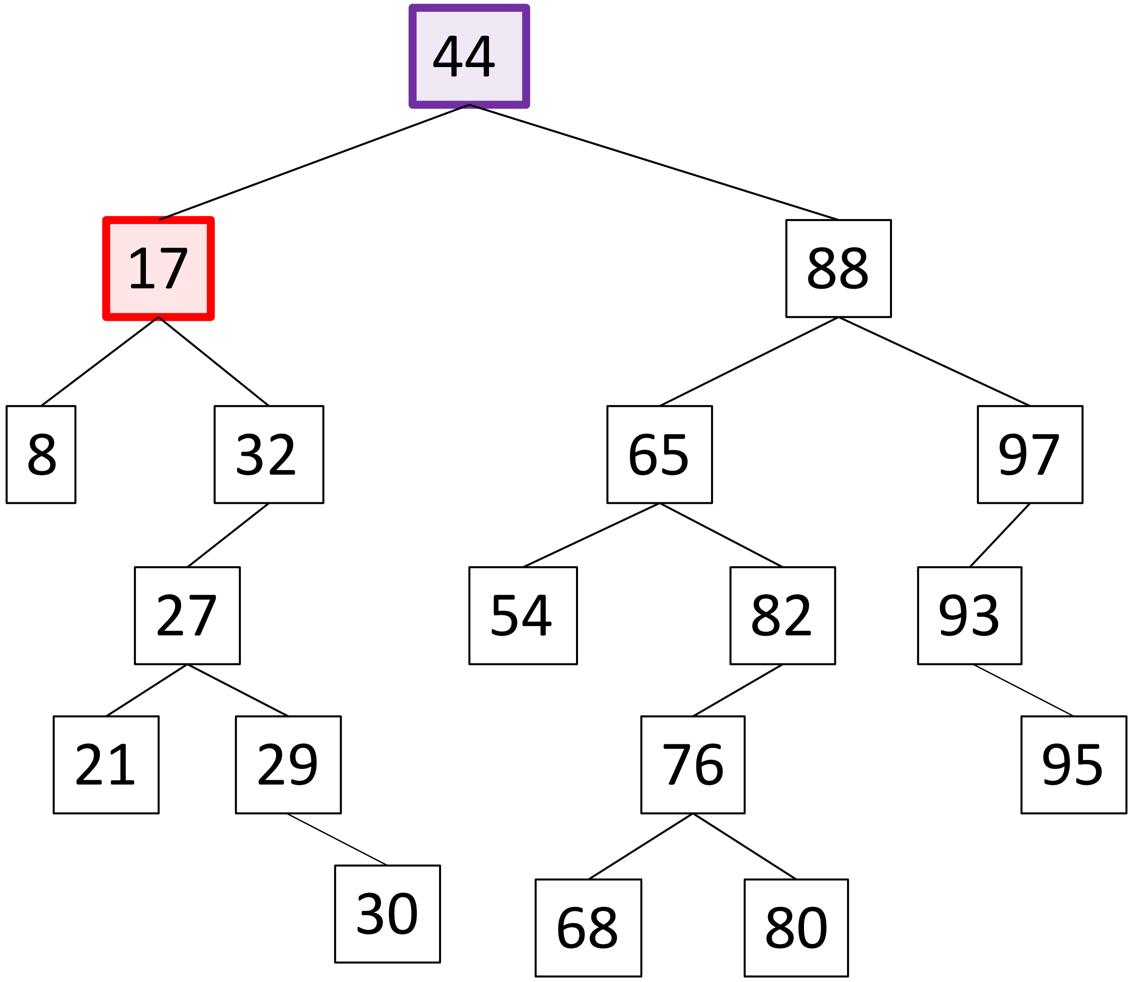
17

8

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



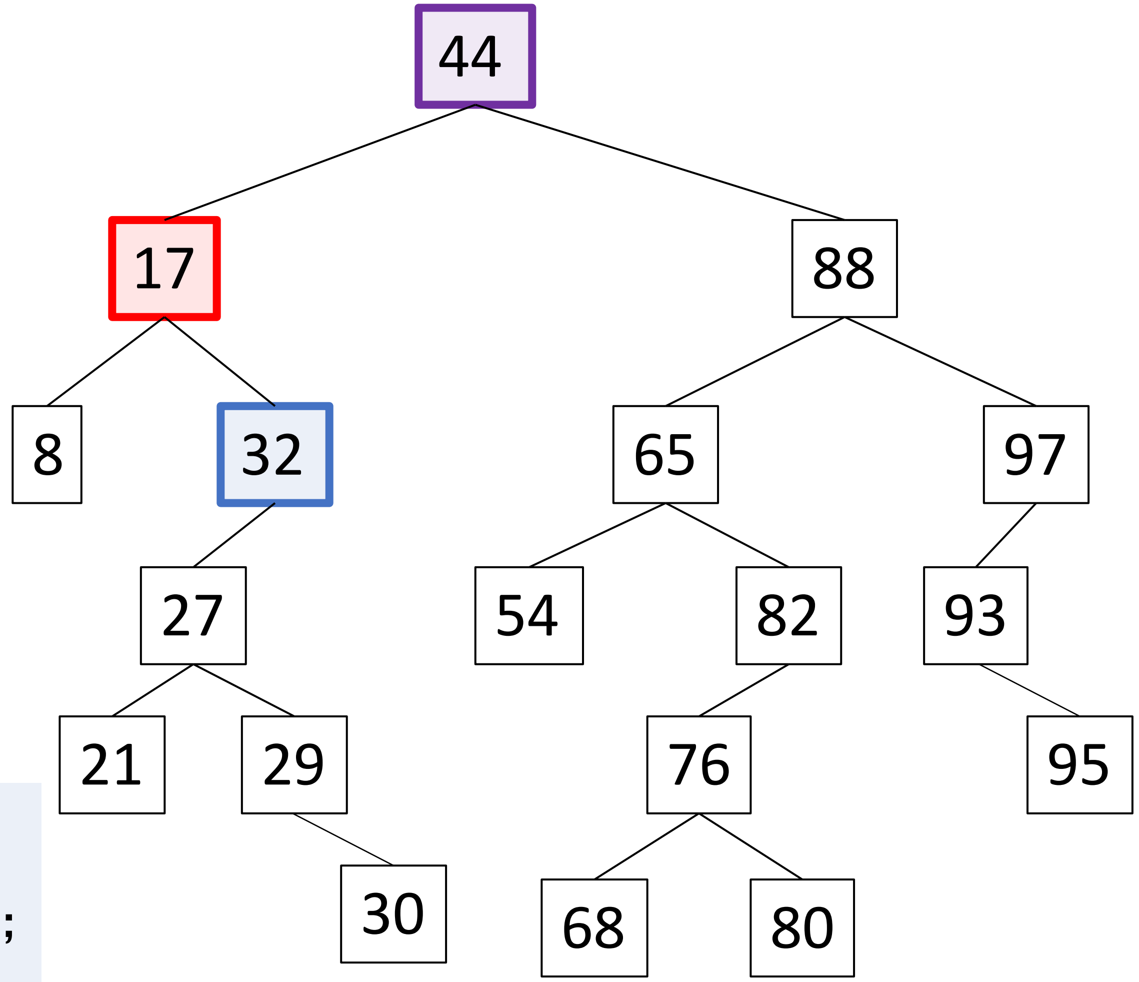
Output:
44
17
8

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(32) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



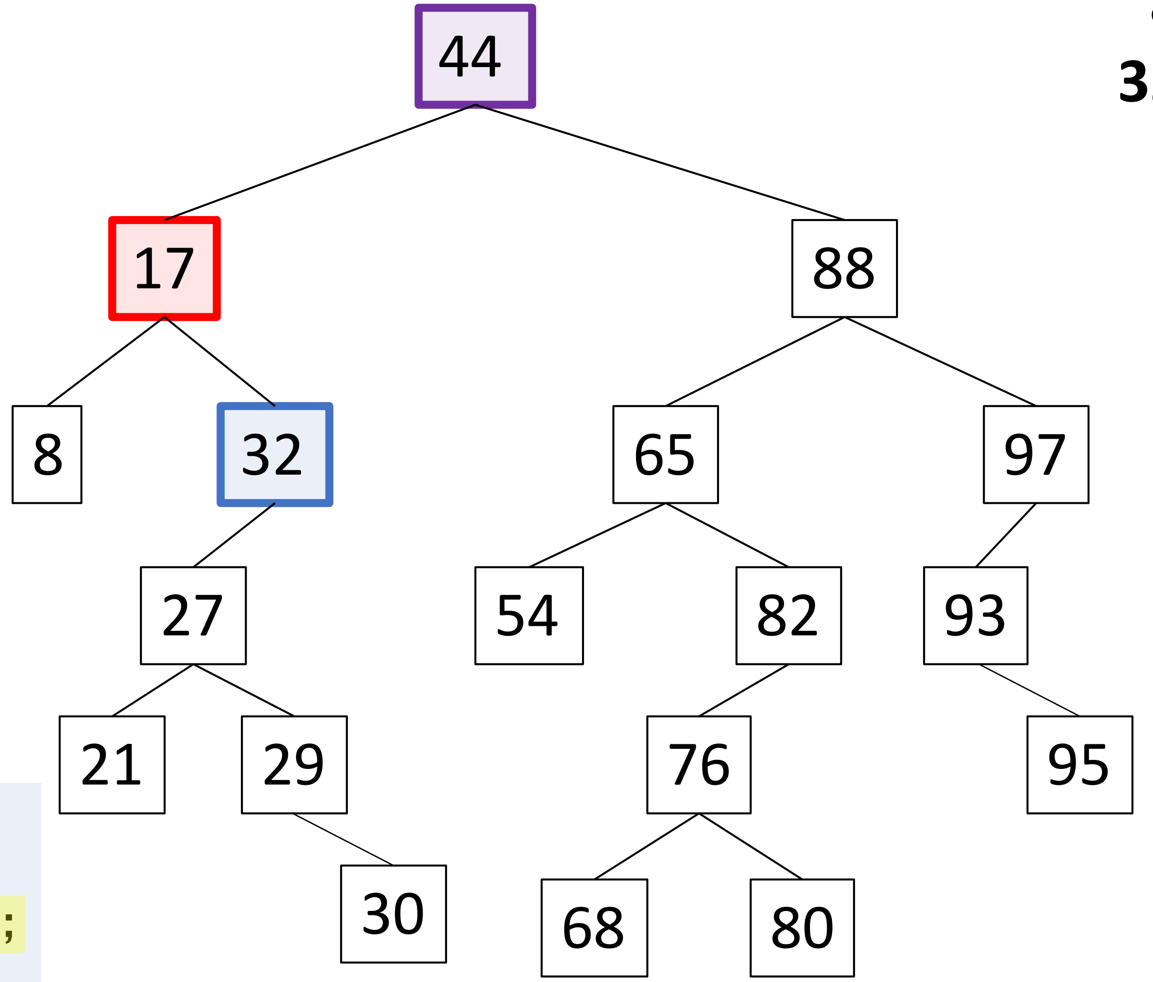
Output:
44
17
8
32

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(32) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



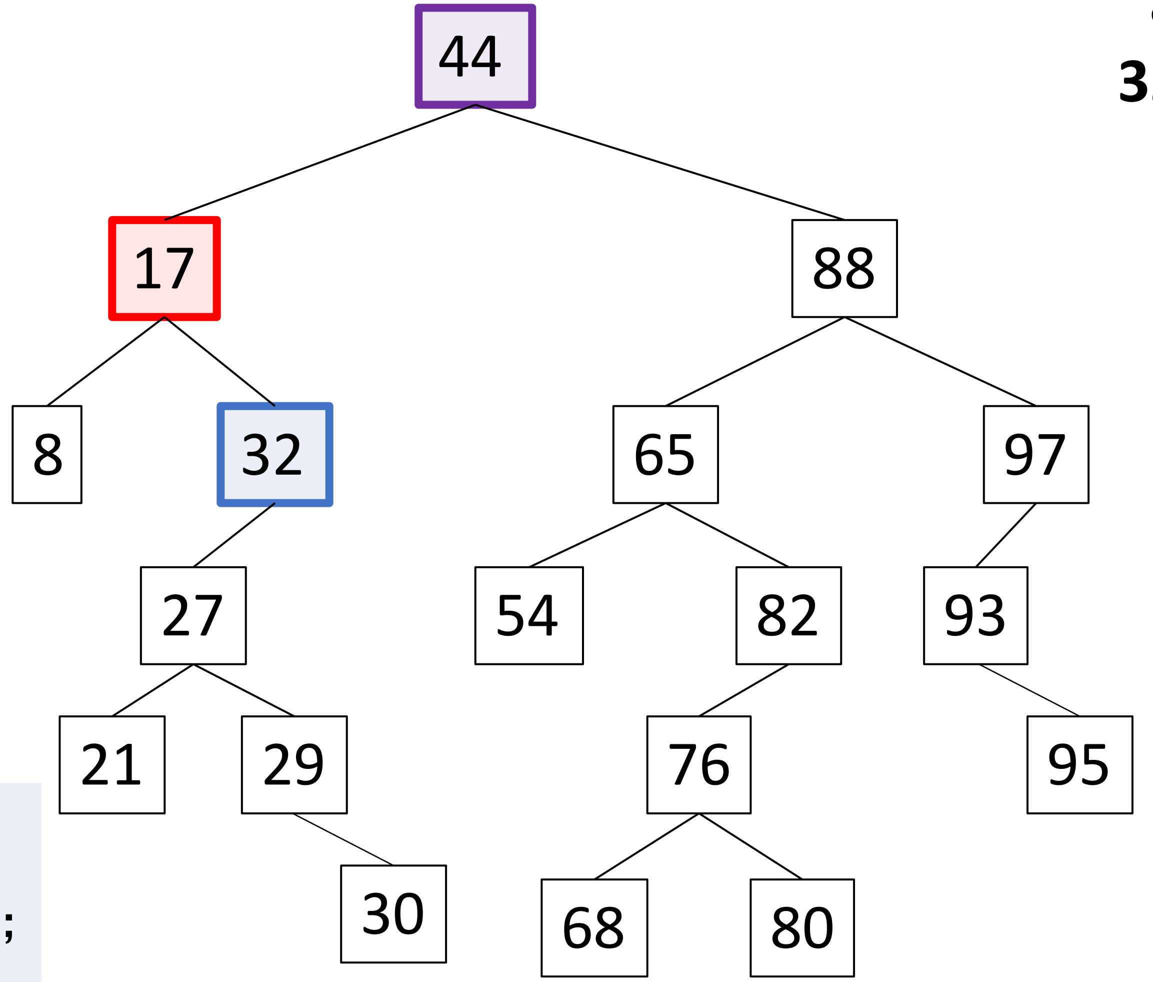
Binary Search Tree - Traversal

Output:
44
17
8
32

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(32) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



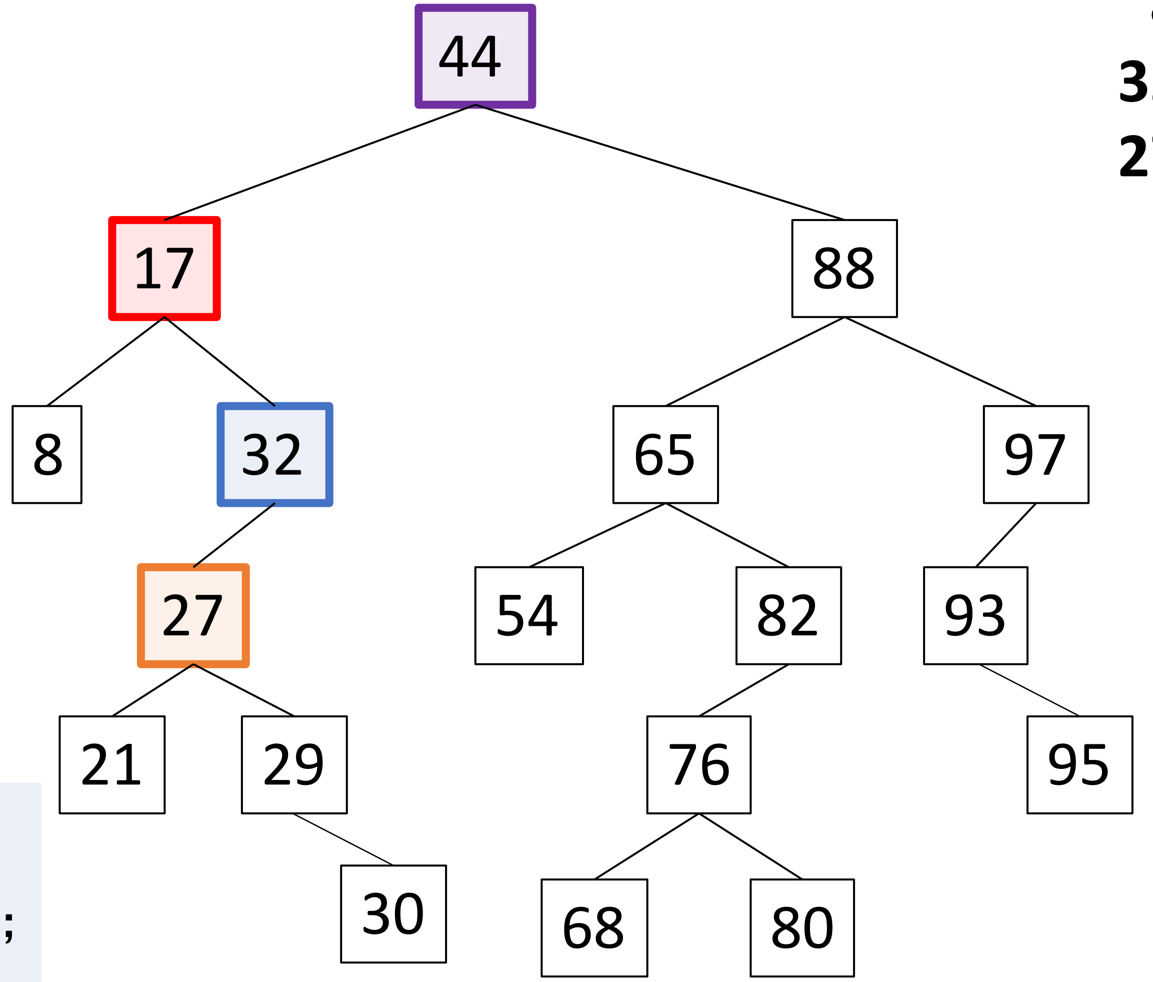
Output:
44
17
8
32
27

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(32) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Binary Search Tree - Traversal

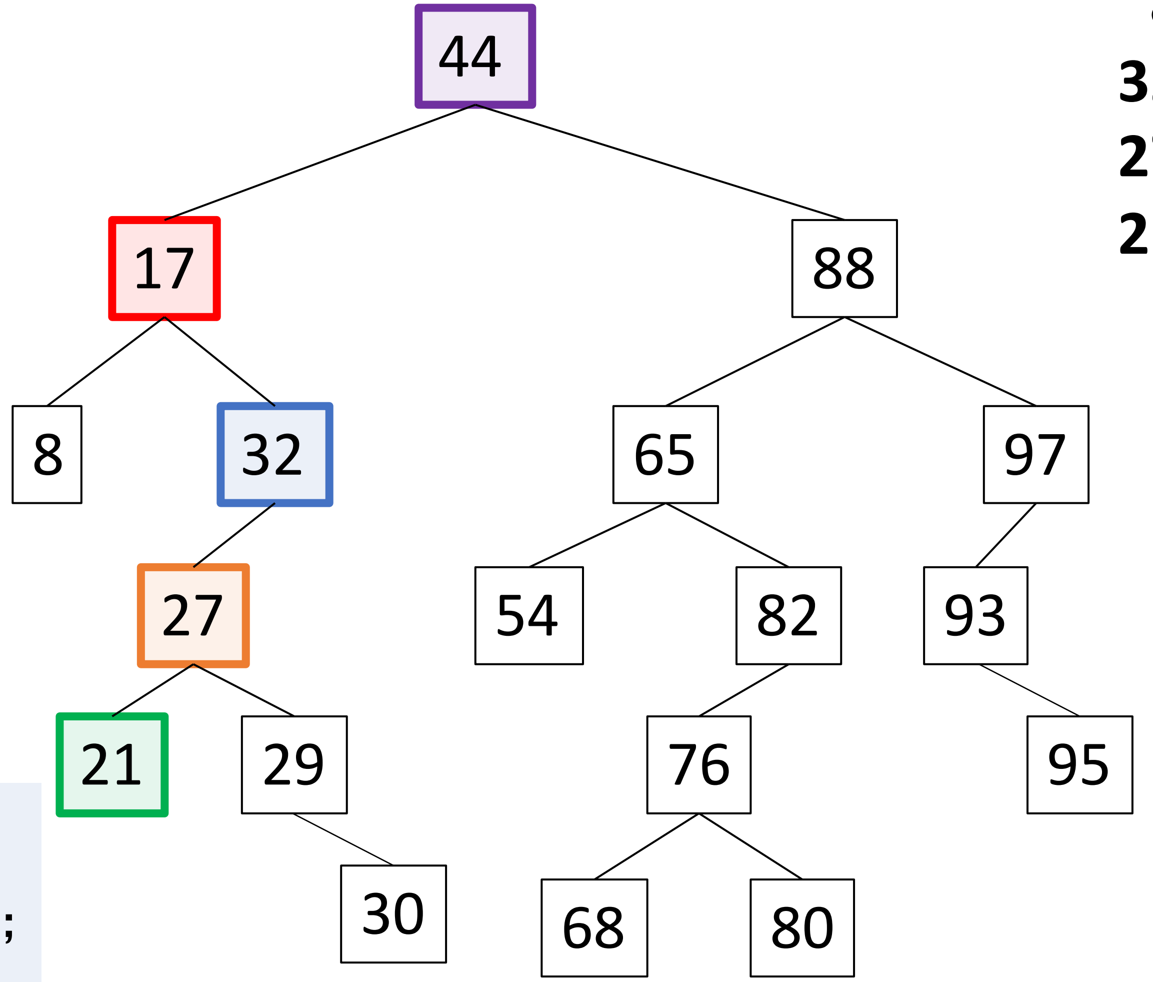
```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(32) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

Output:

```
44
17
8
32
27
21
```

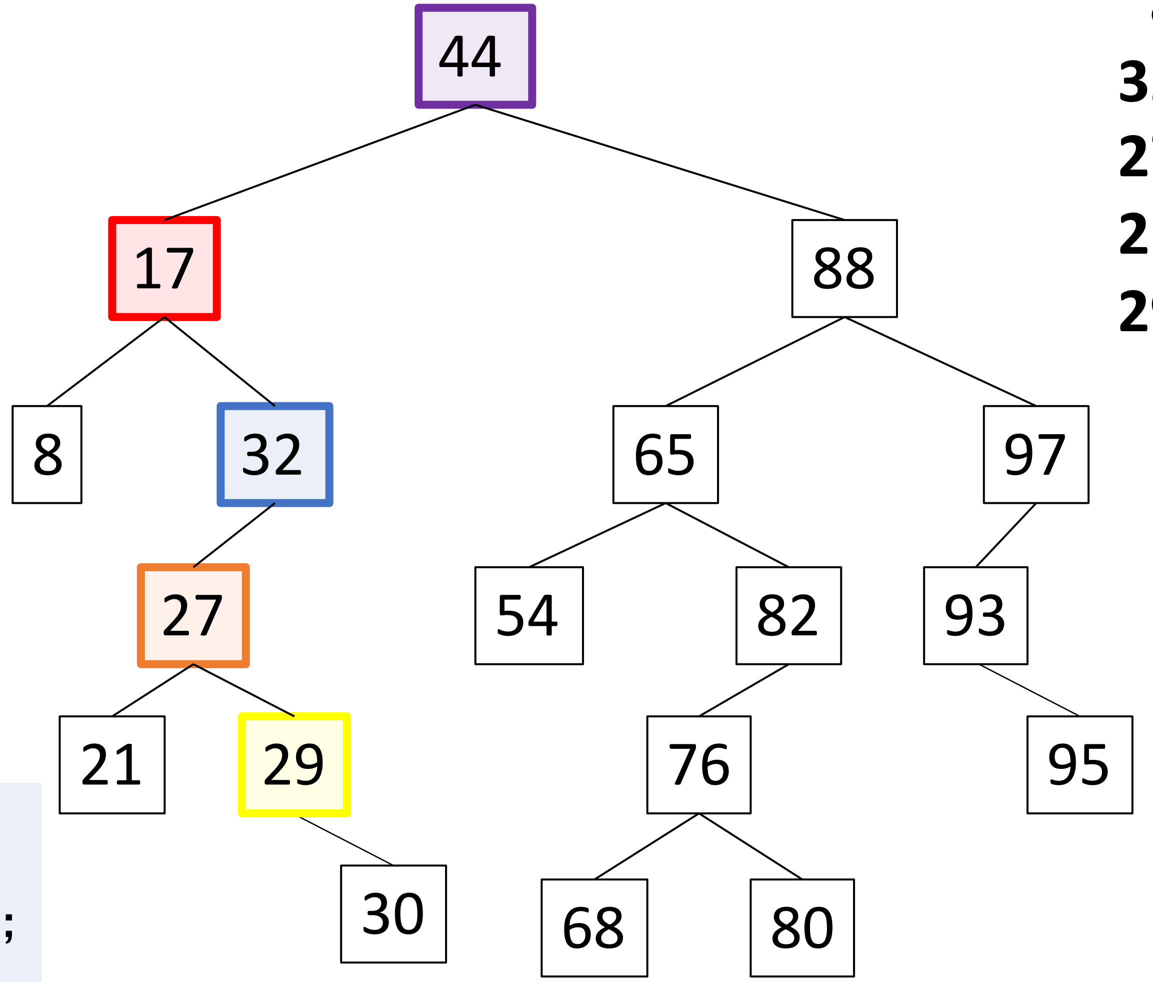


Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(32) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:

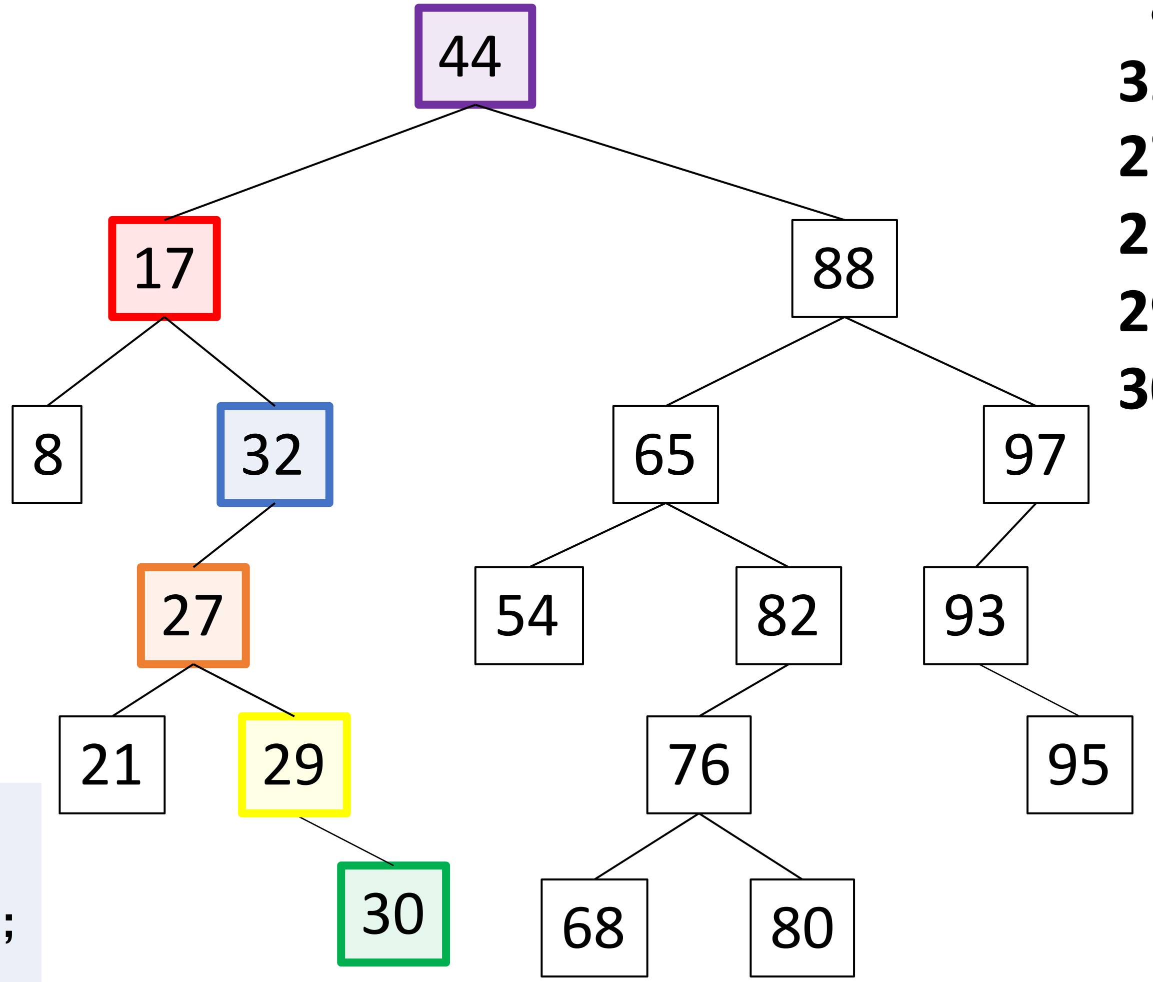
```
44
17
 8
32
27
21
29
```

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(32) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



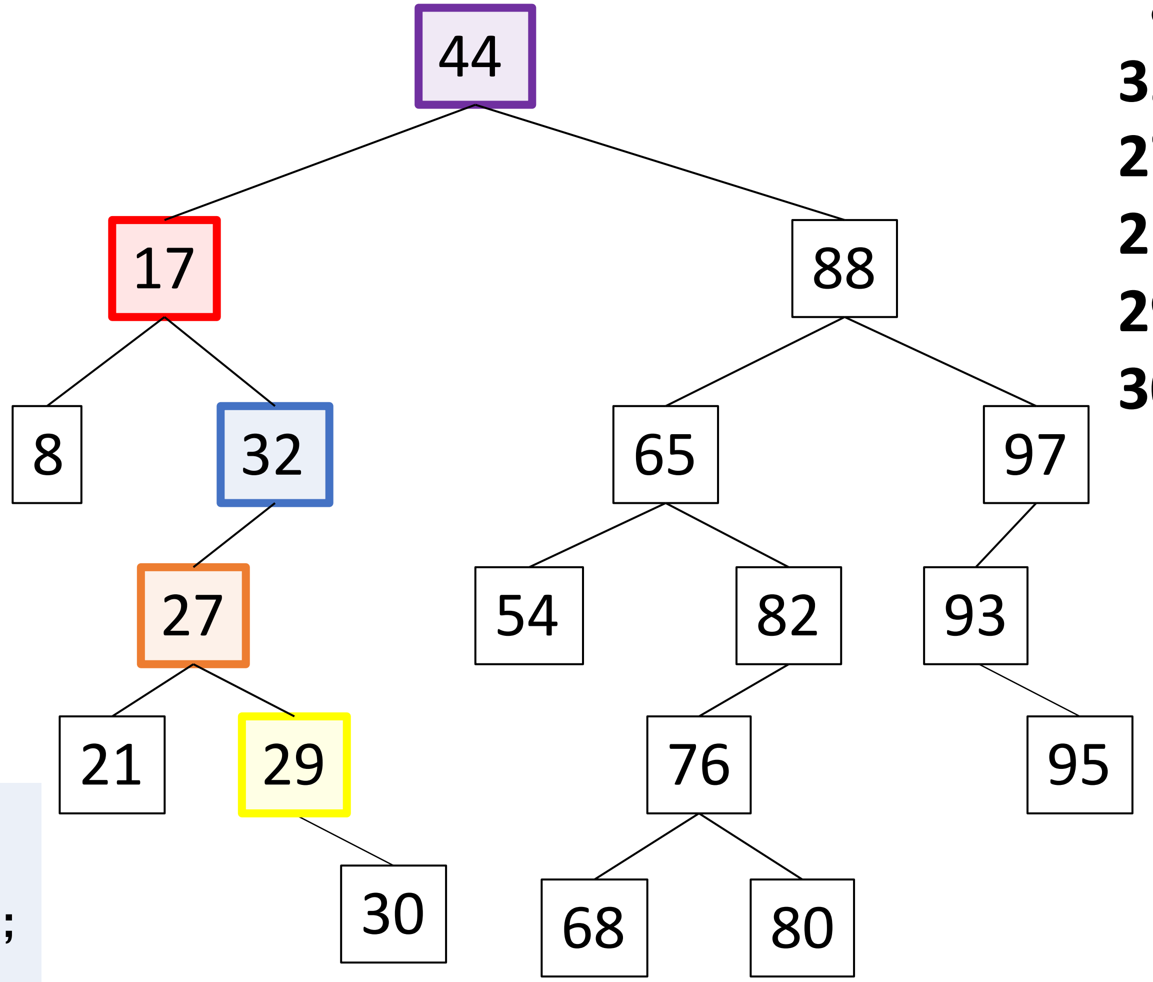
Output:
44
17
8
32
27
21
29
30

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(32) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:

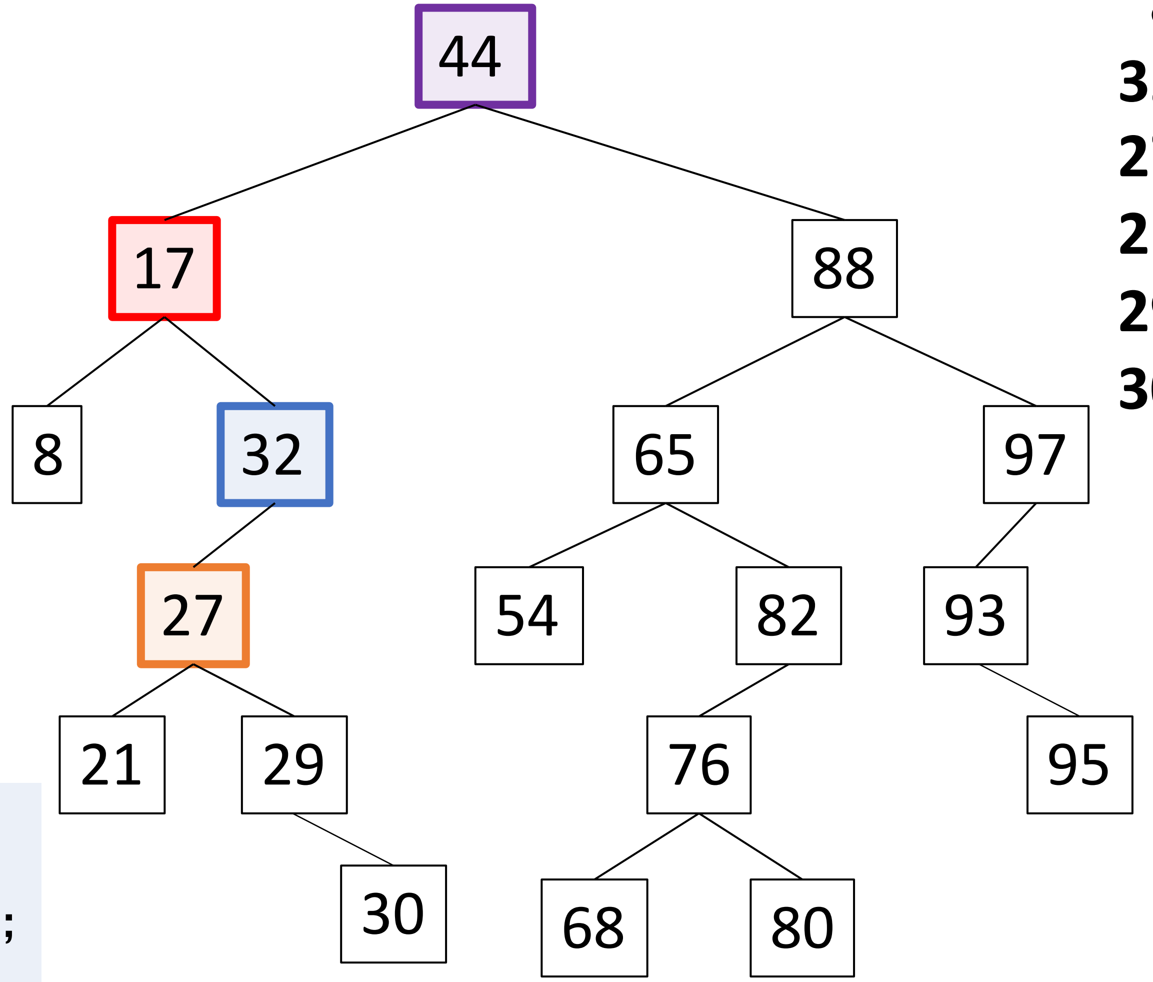
```
44
17
8
32
27
21
29
30
```

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(32) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:

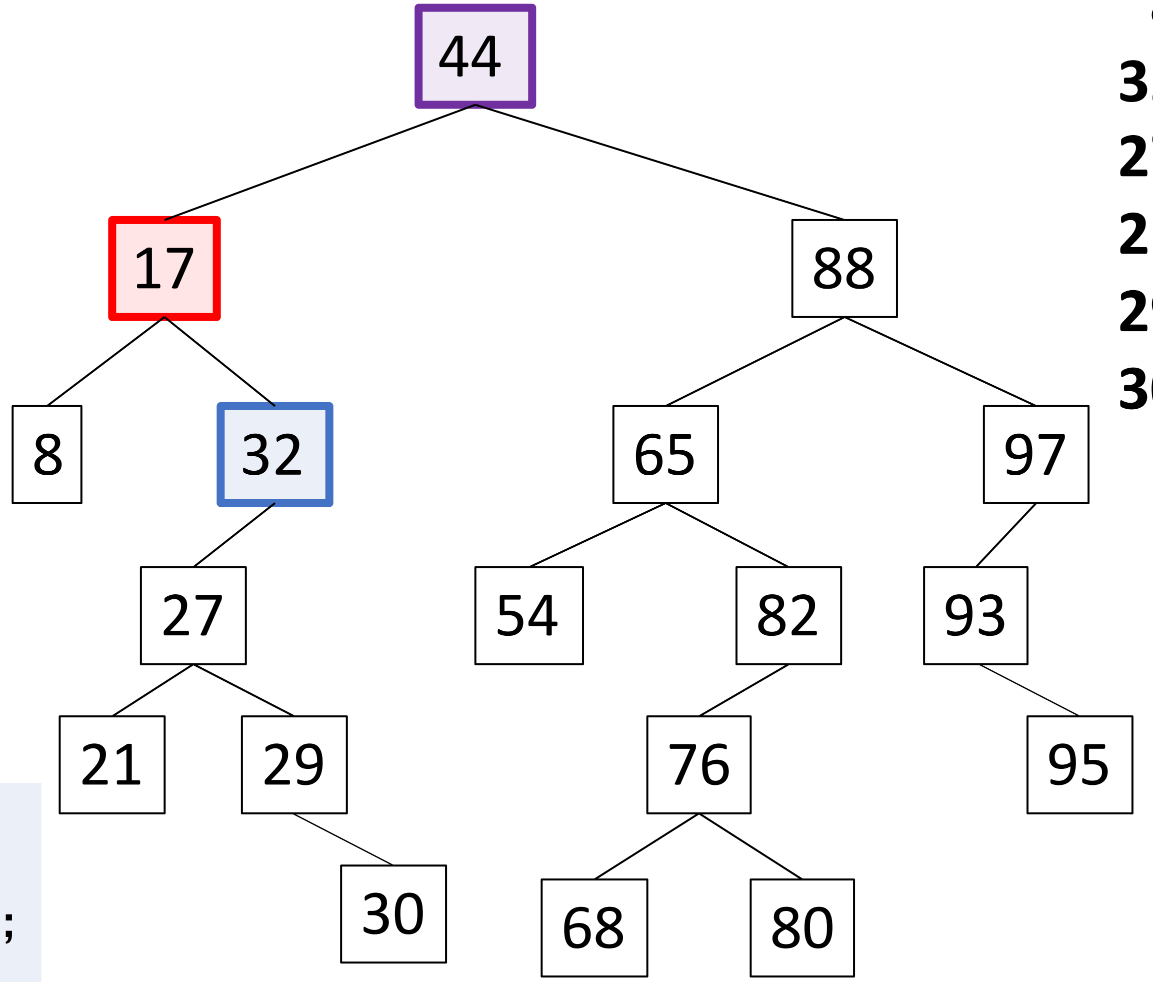
```
44
17
 8
32
27
21
29
30
```

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(32) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



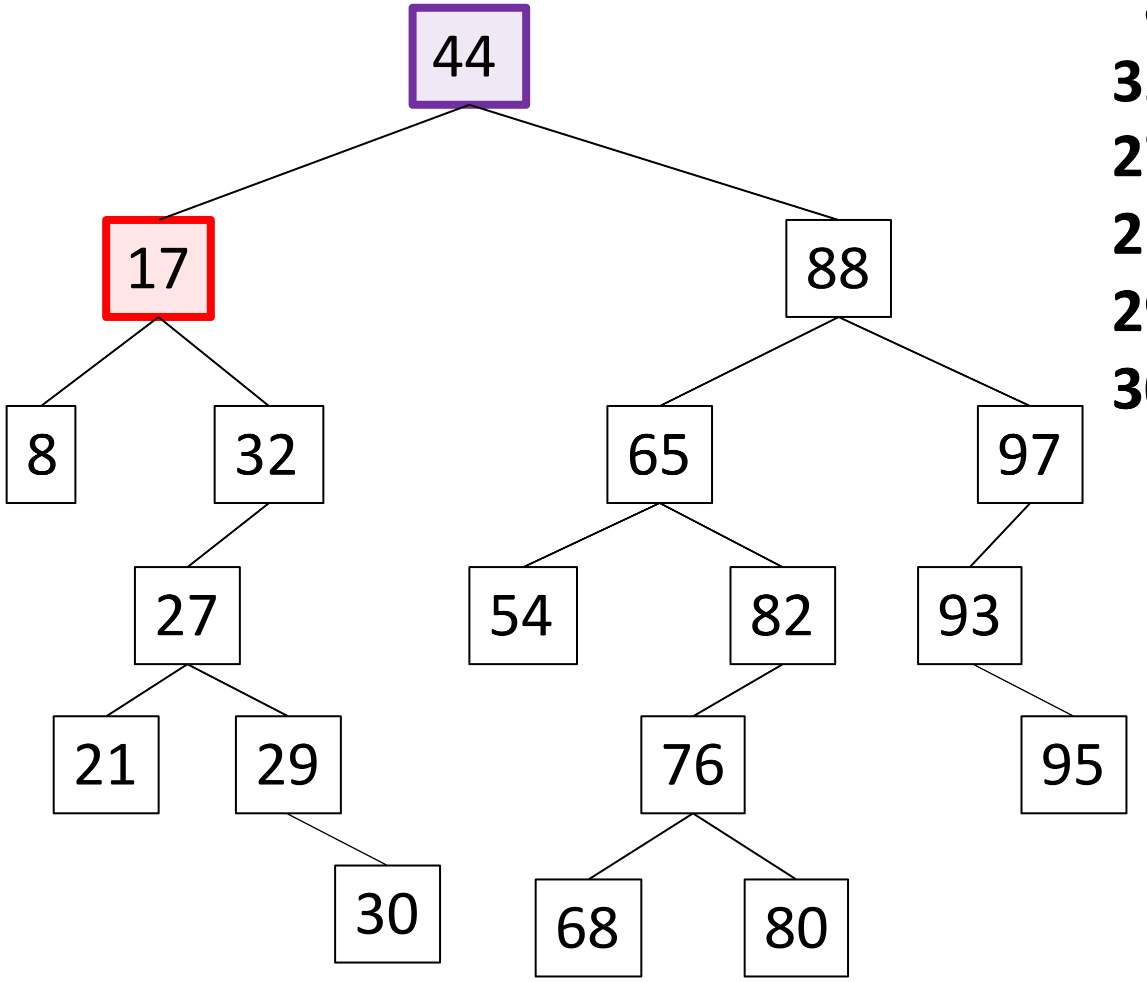
Output:

```
44
17
 8
32
27
21
29
30
```

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

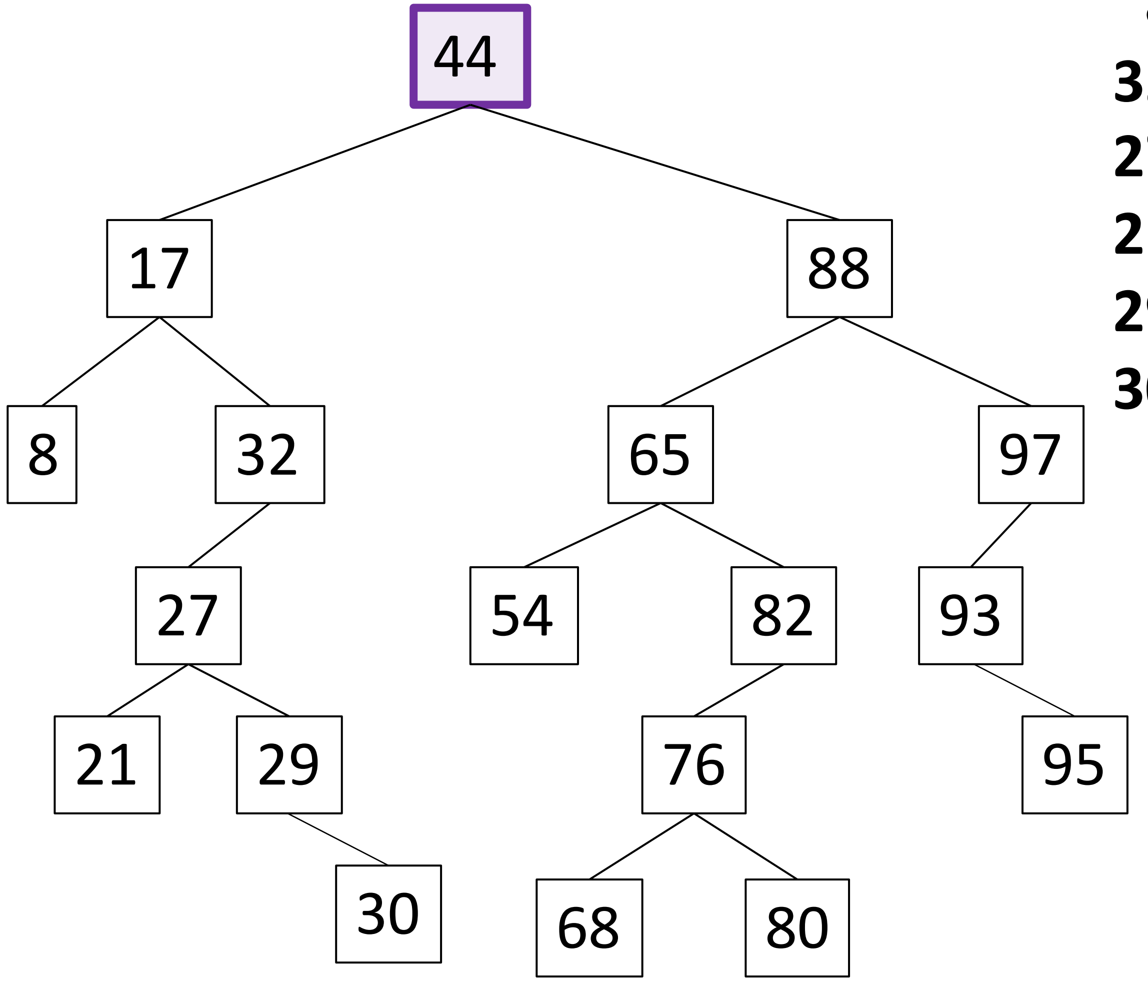


Output:

```
44
17
 8
32
27
21
29
30
```

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

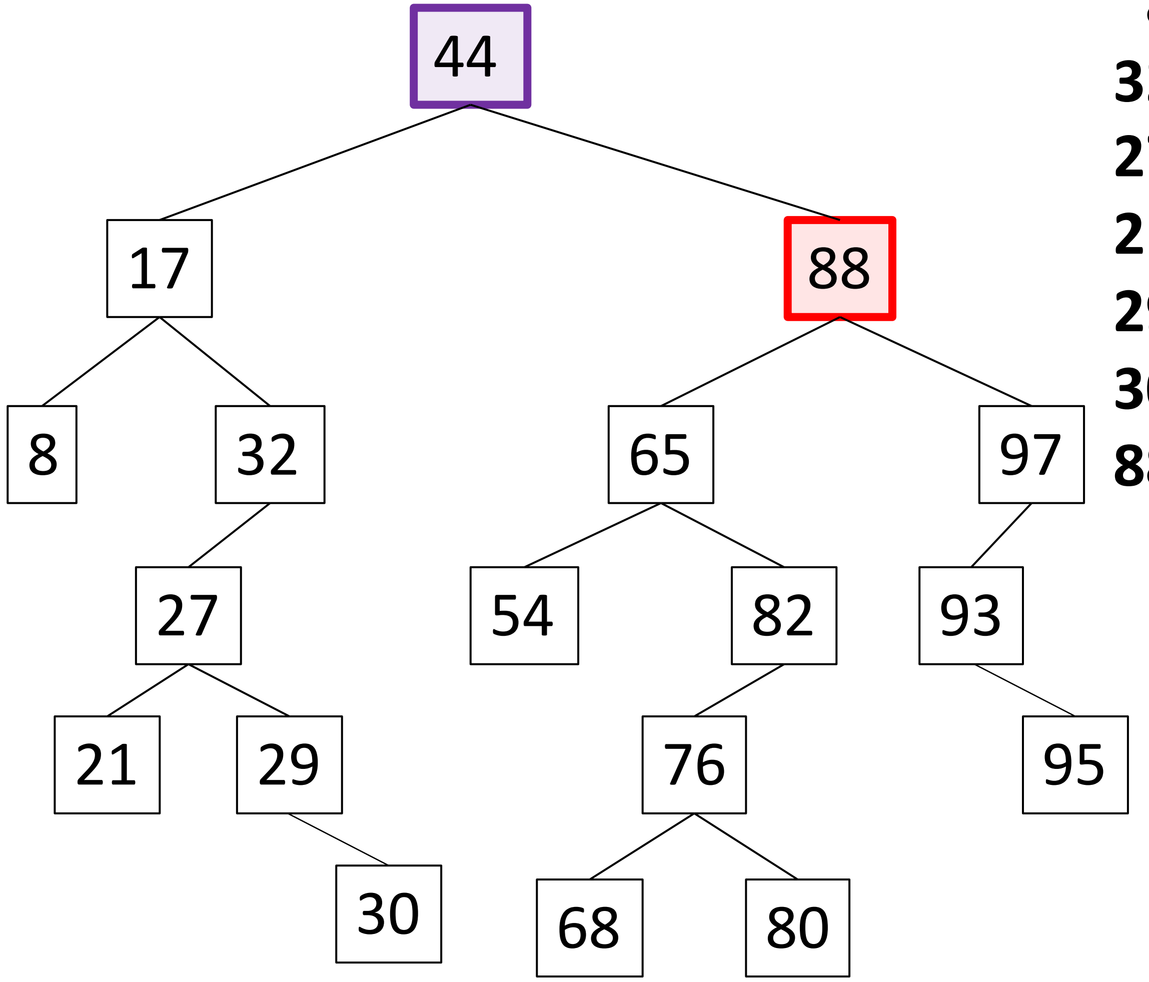


Output:

```
44
17
 8
32
27
21
29
30
```

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

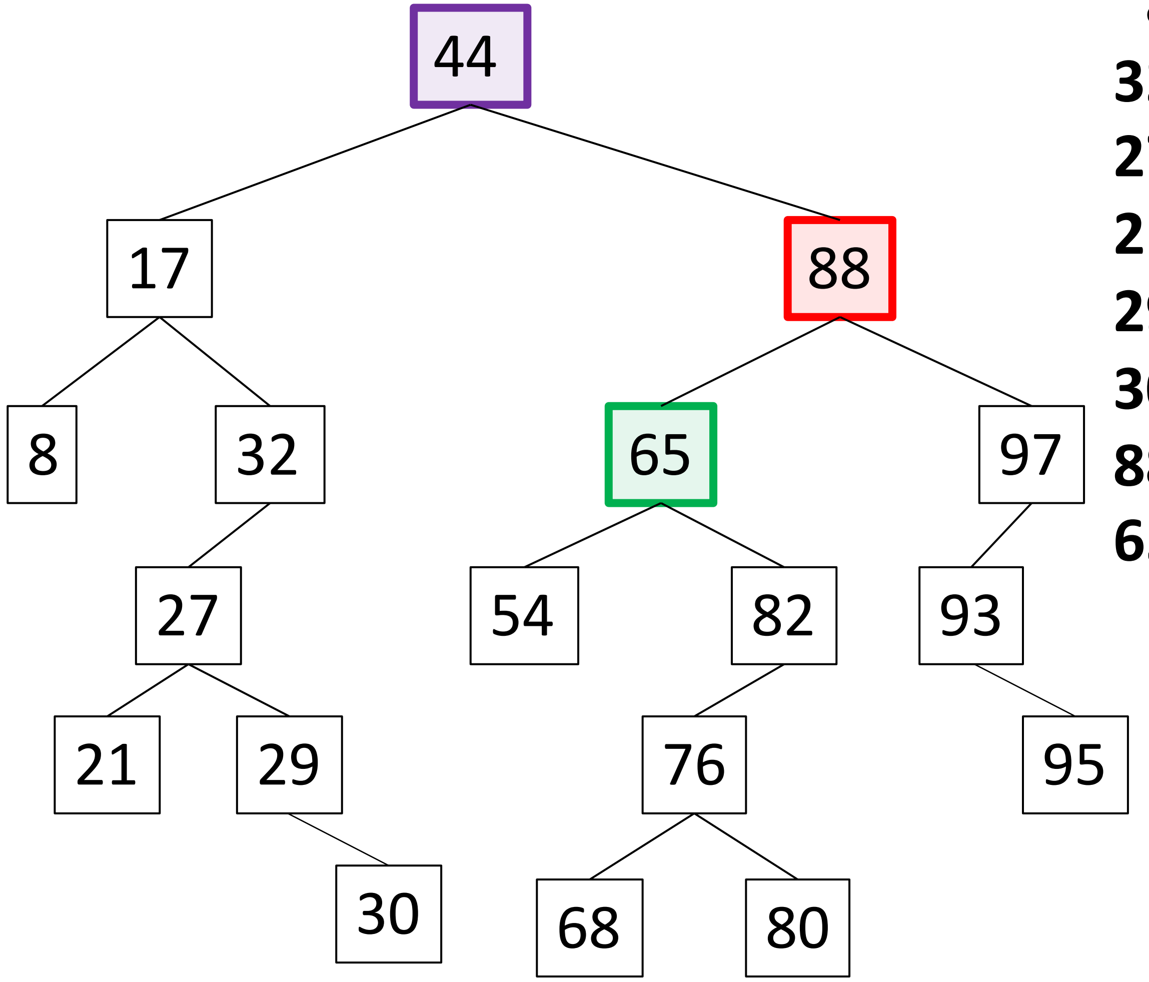


Output:

```
44
17
 8
32
27
21
29
30
88
```

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

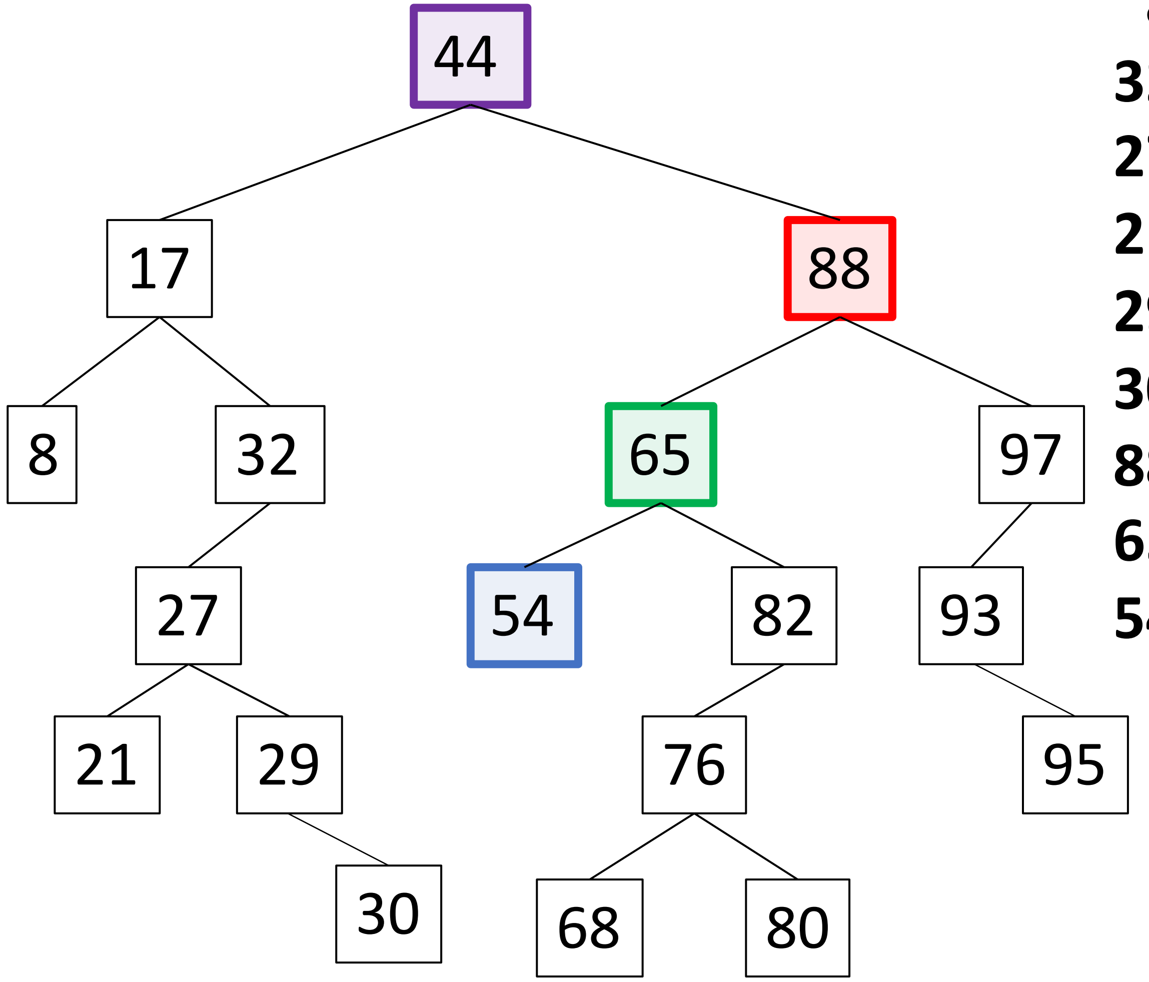


Output:

```
44
17
 8
32
27
21
29
30
88
65
```

Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Output:

44
17
8
32
27
21
29
30
88
65
54

Binary Search Trees
CSCI 232

Output:
44
17
8

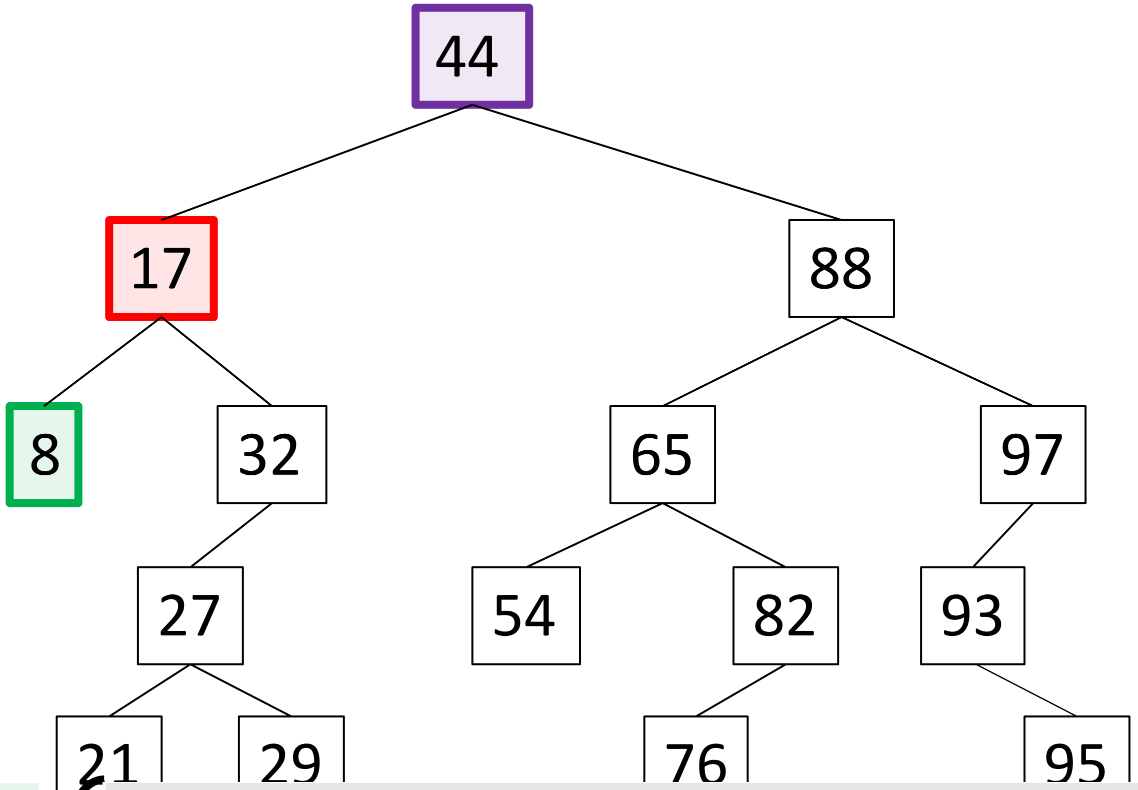
Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(17) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

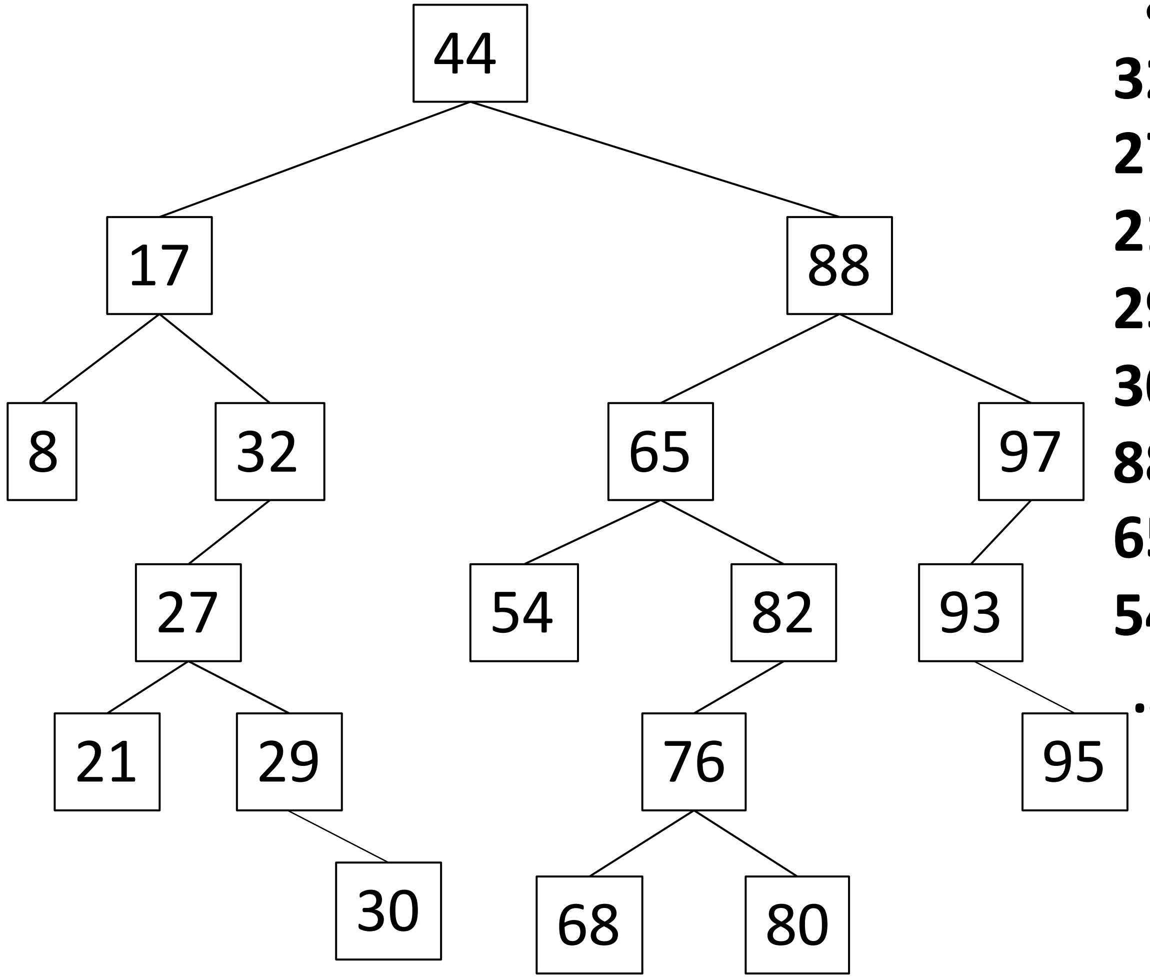
```
public void depthFirst(8) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

```
public void depthFirst(null) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```



Binary Search Tree - Traversal

```
public void depthFirst(44) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

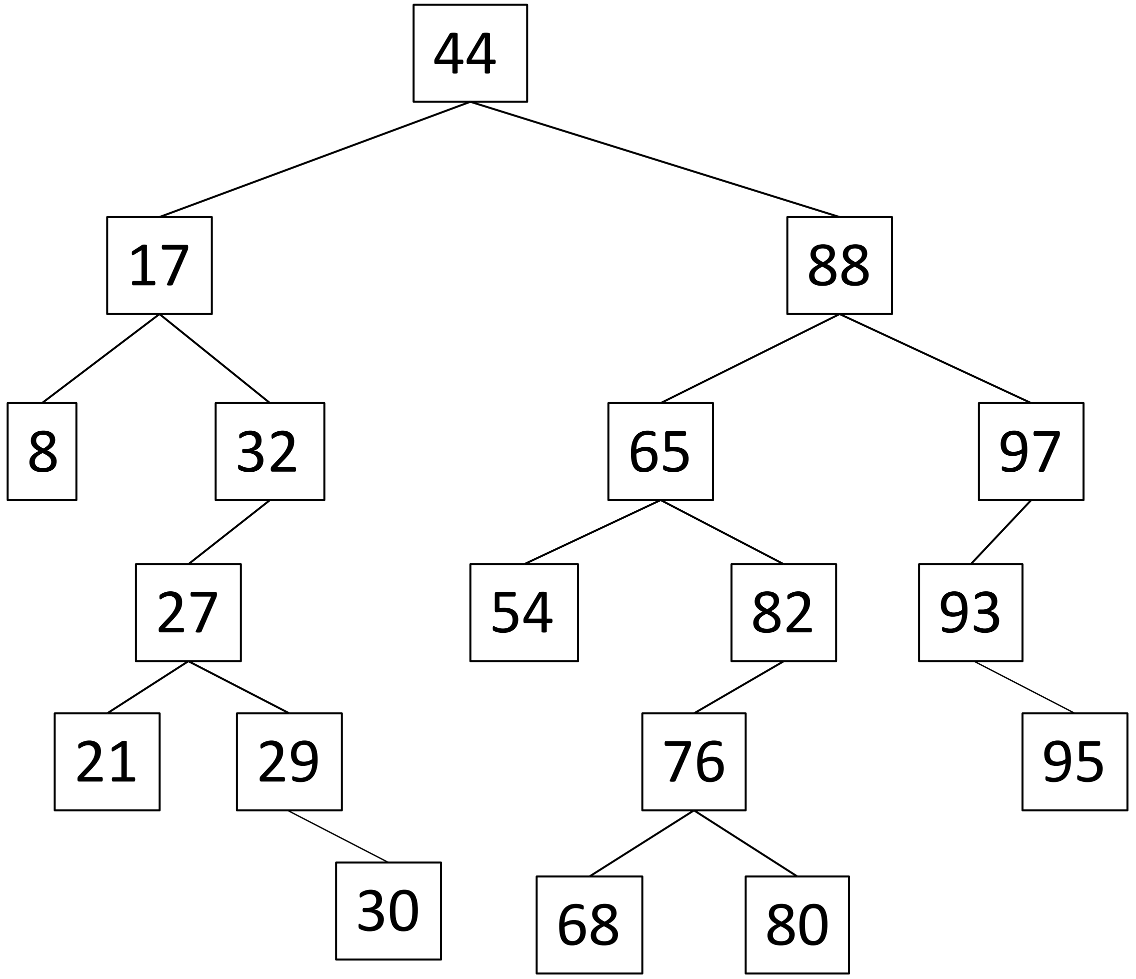


Output:
44
17
8
32
27
21
29
30
88
65
54
…

Output:

Binary Search Tree - Traversal

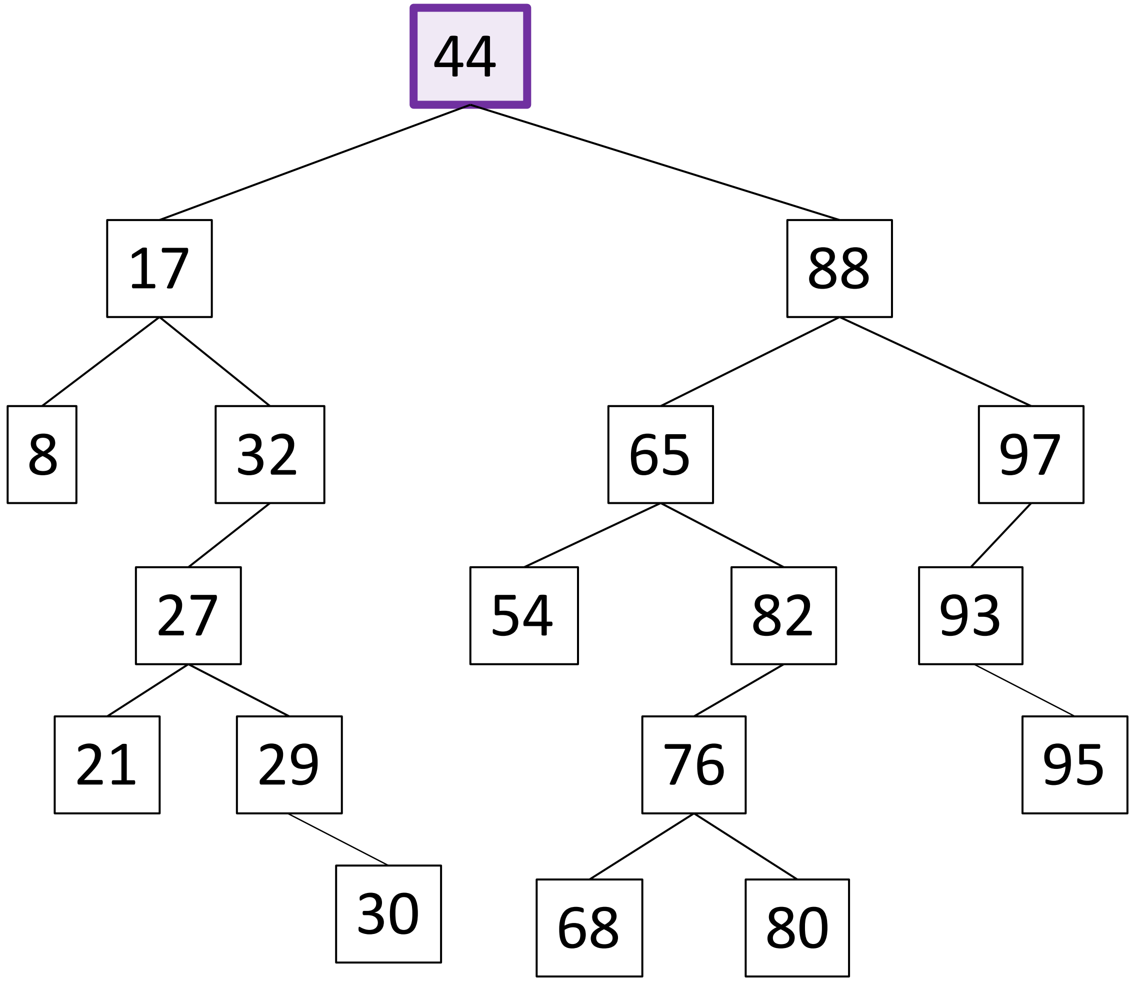
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:

Binary Search Tree - Traversal

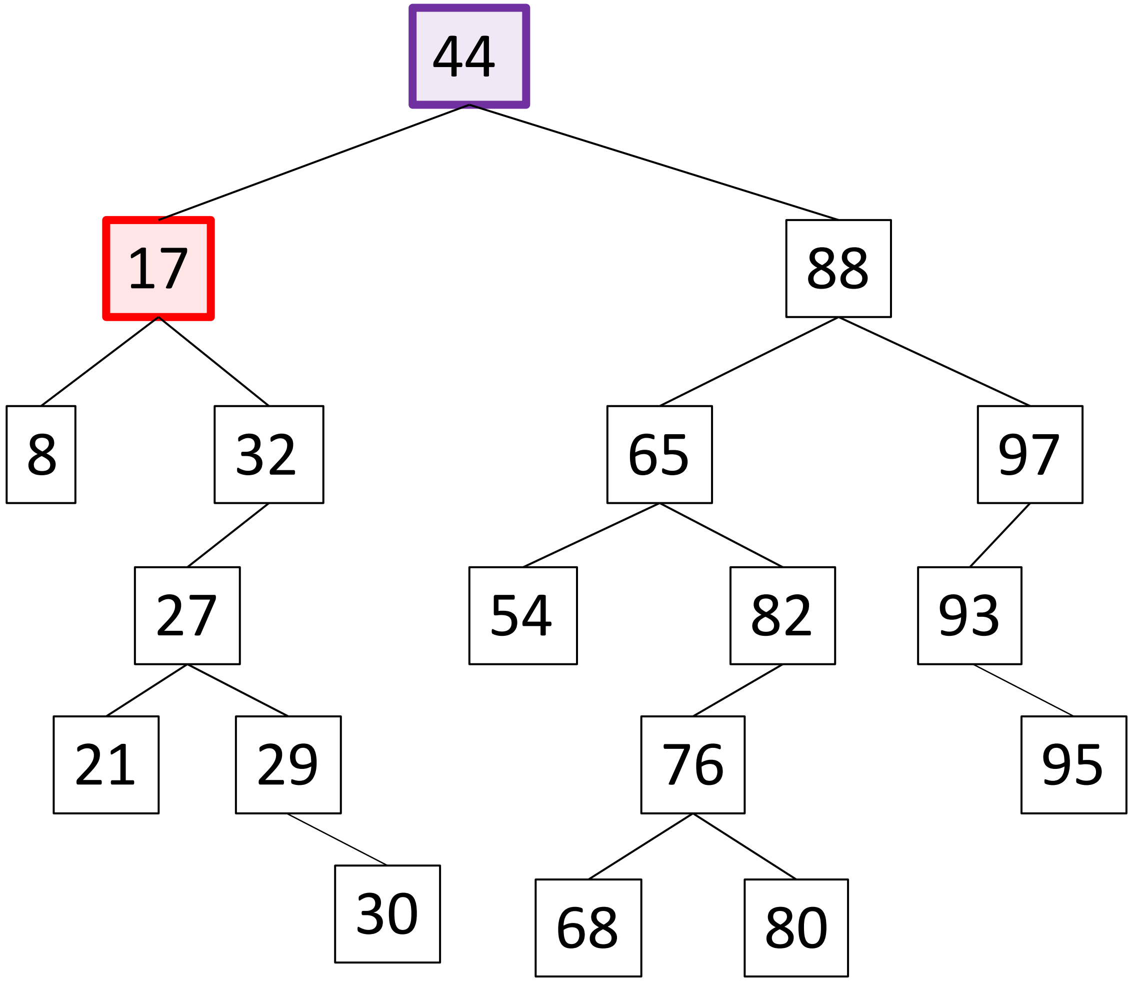
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:

Binary Search Tree - Traversal

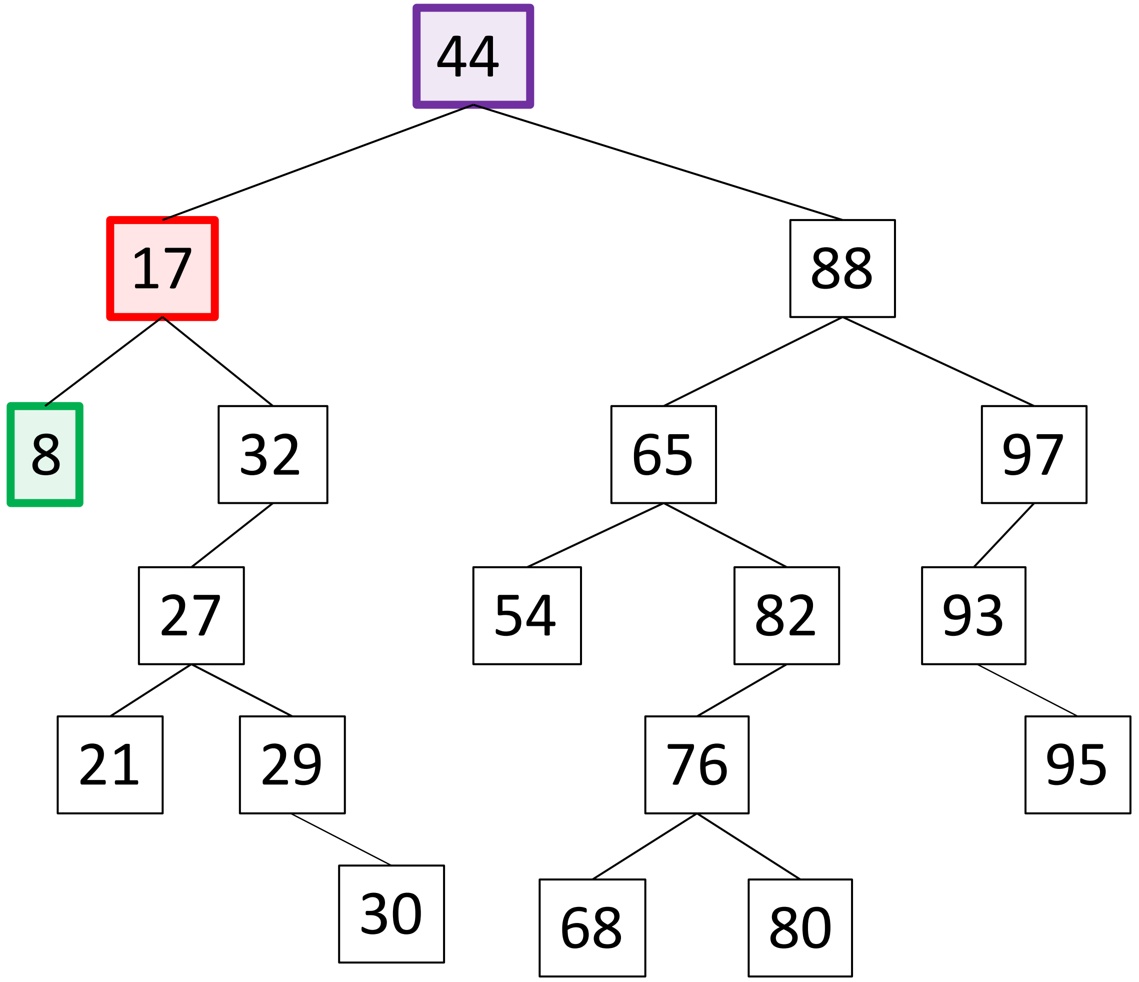
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());

        System.out.println(n.getValue());
    }
}
```



Output:

Binary Search Tree - Traversal

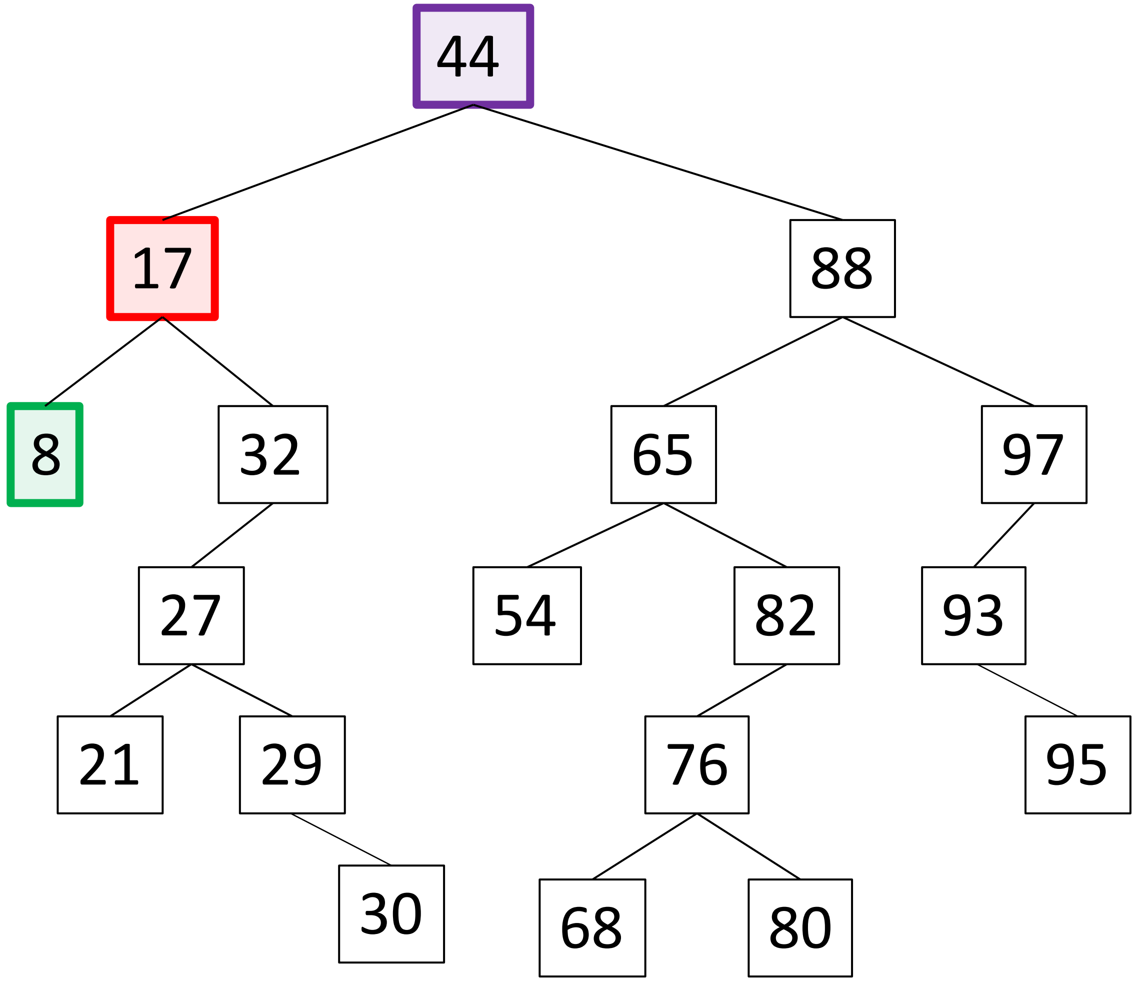
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8

Binary Search Tree - Traversal

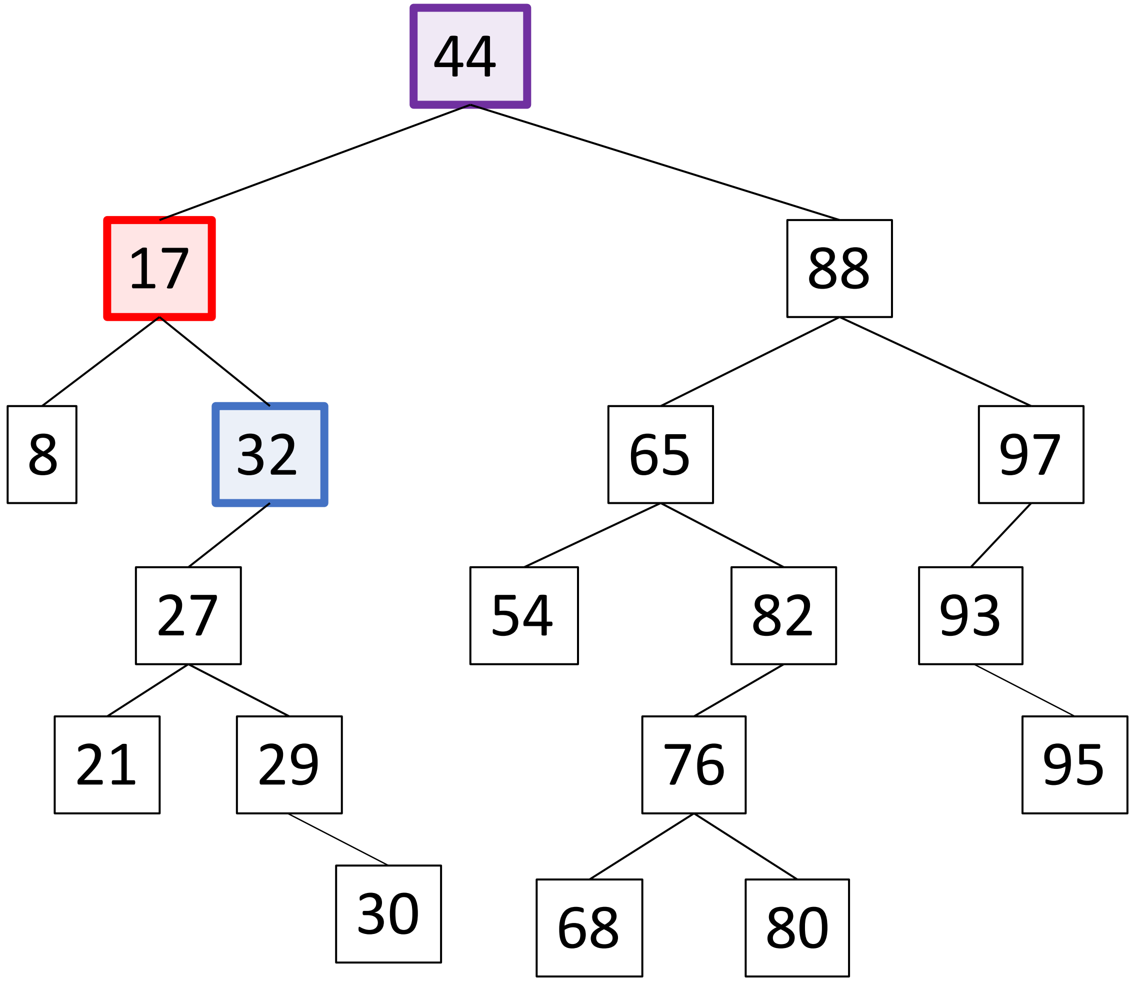
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8

Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```

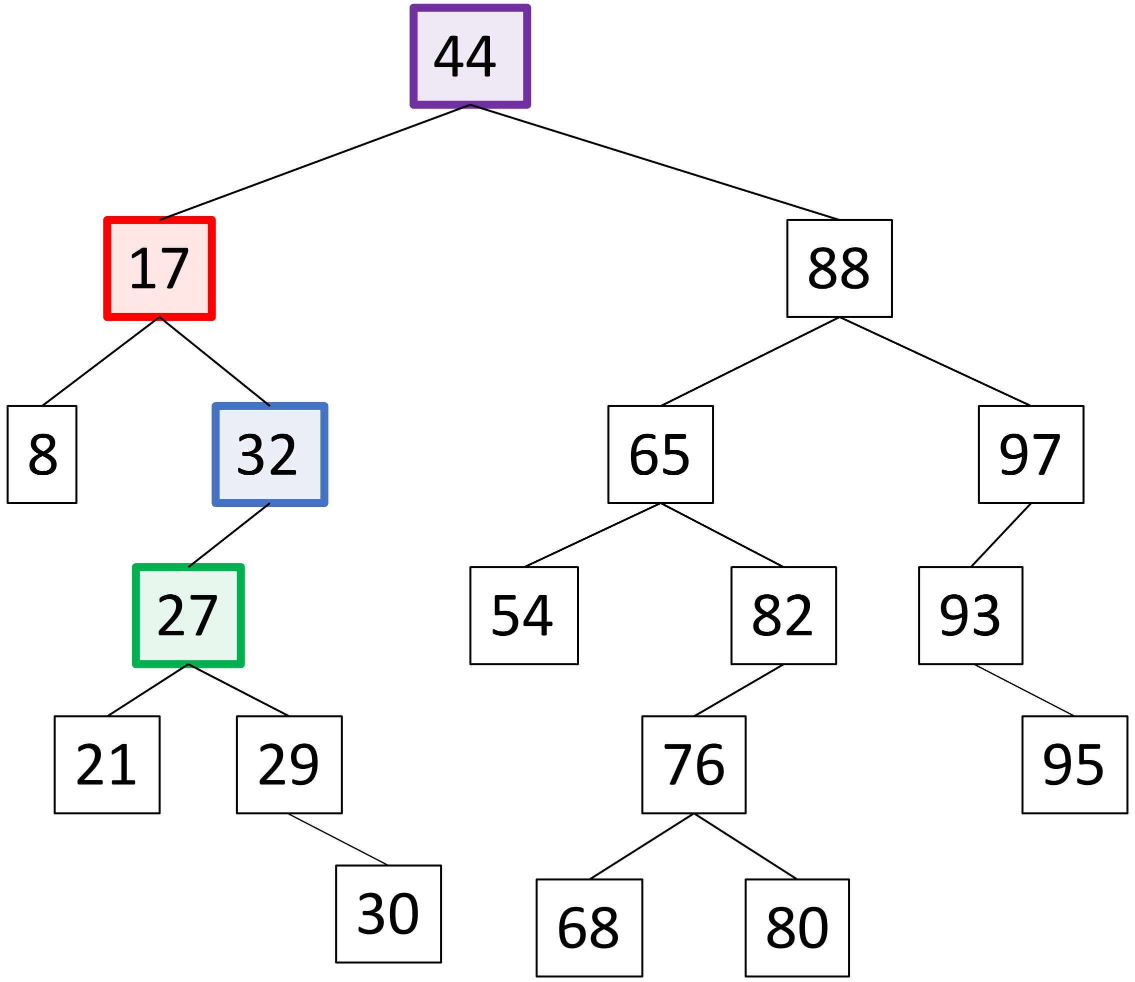


Output:

8

Binary Search Tree - Traversal

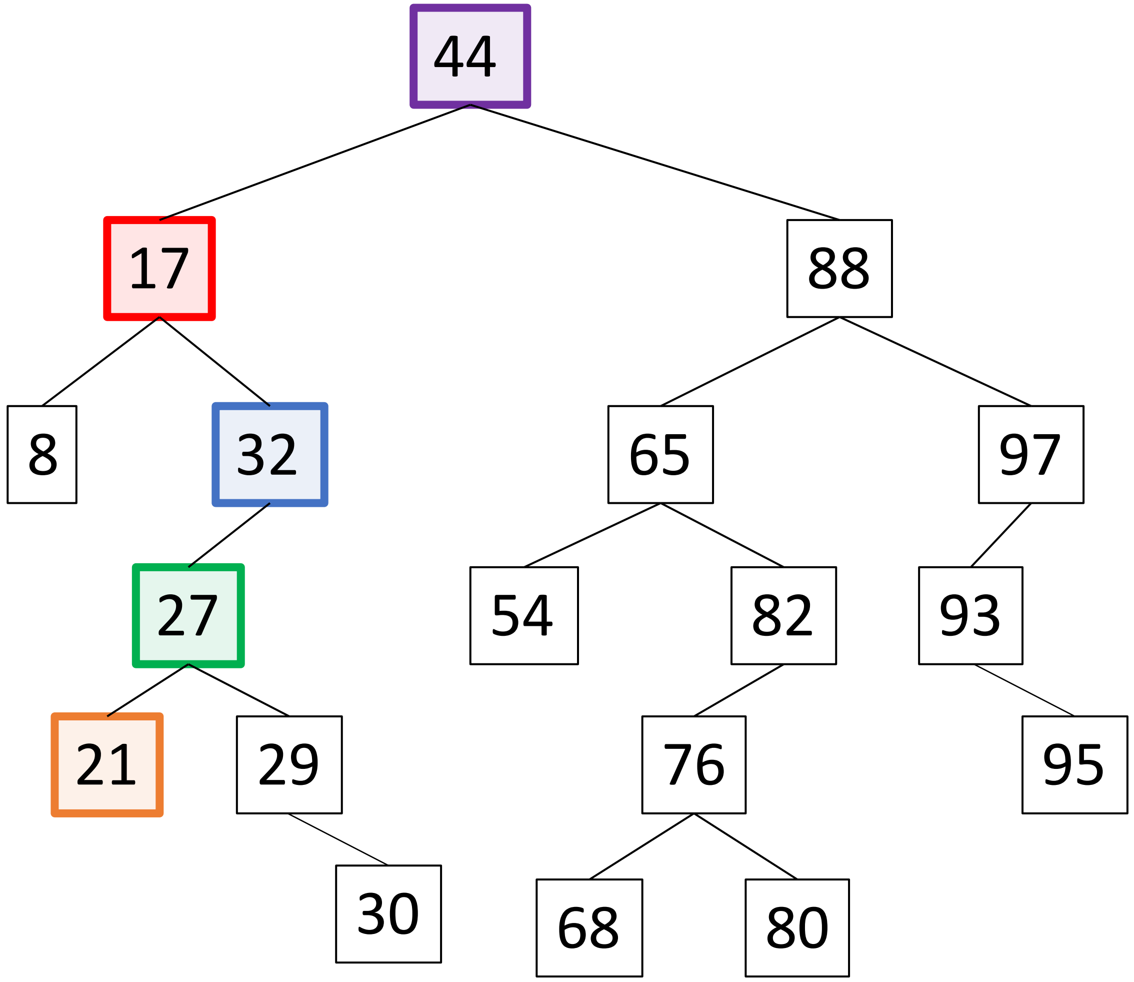
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8

Binary Search Tree - Traversal

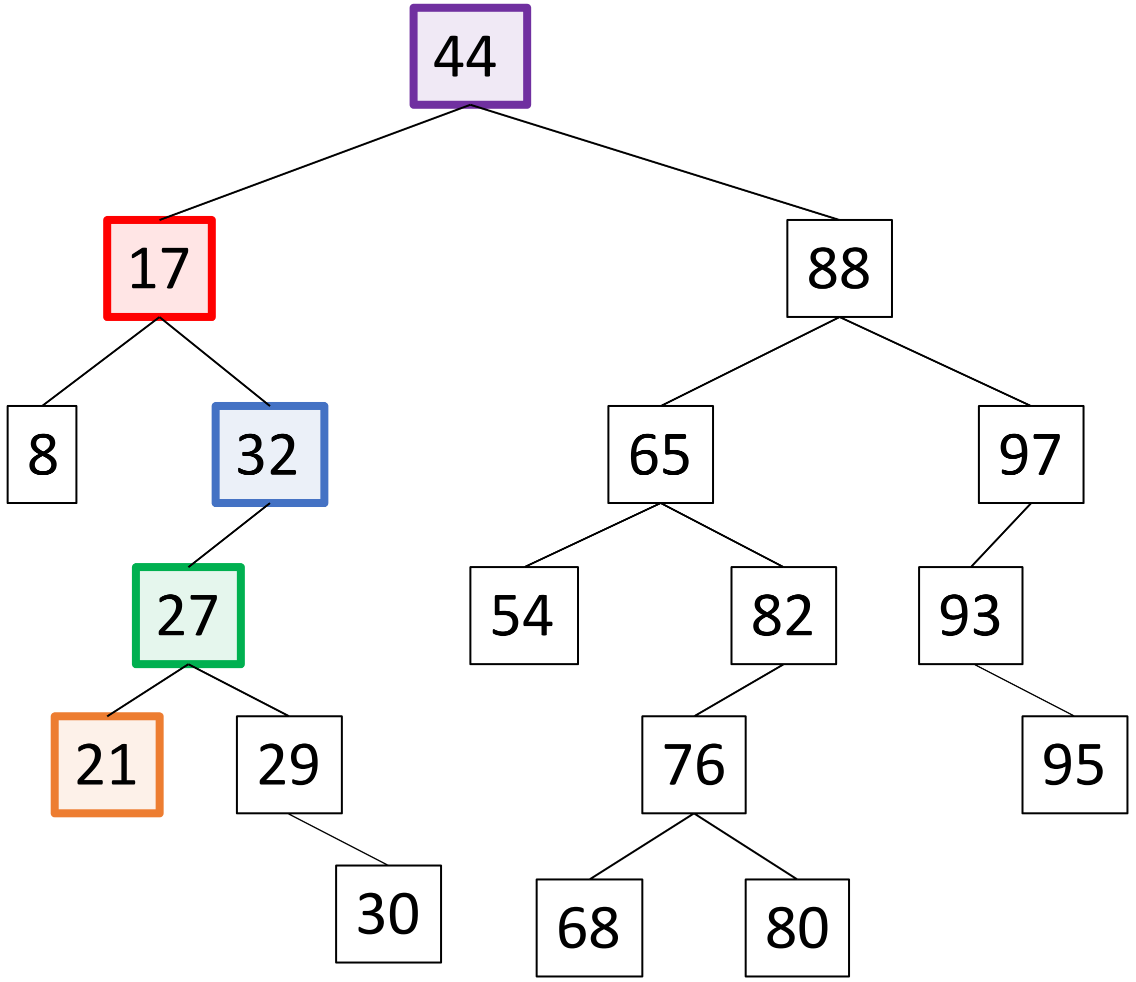
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8
21

Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



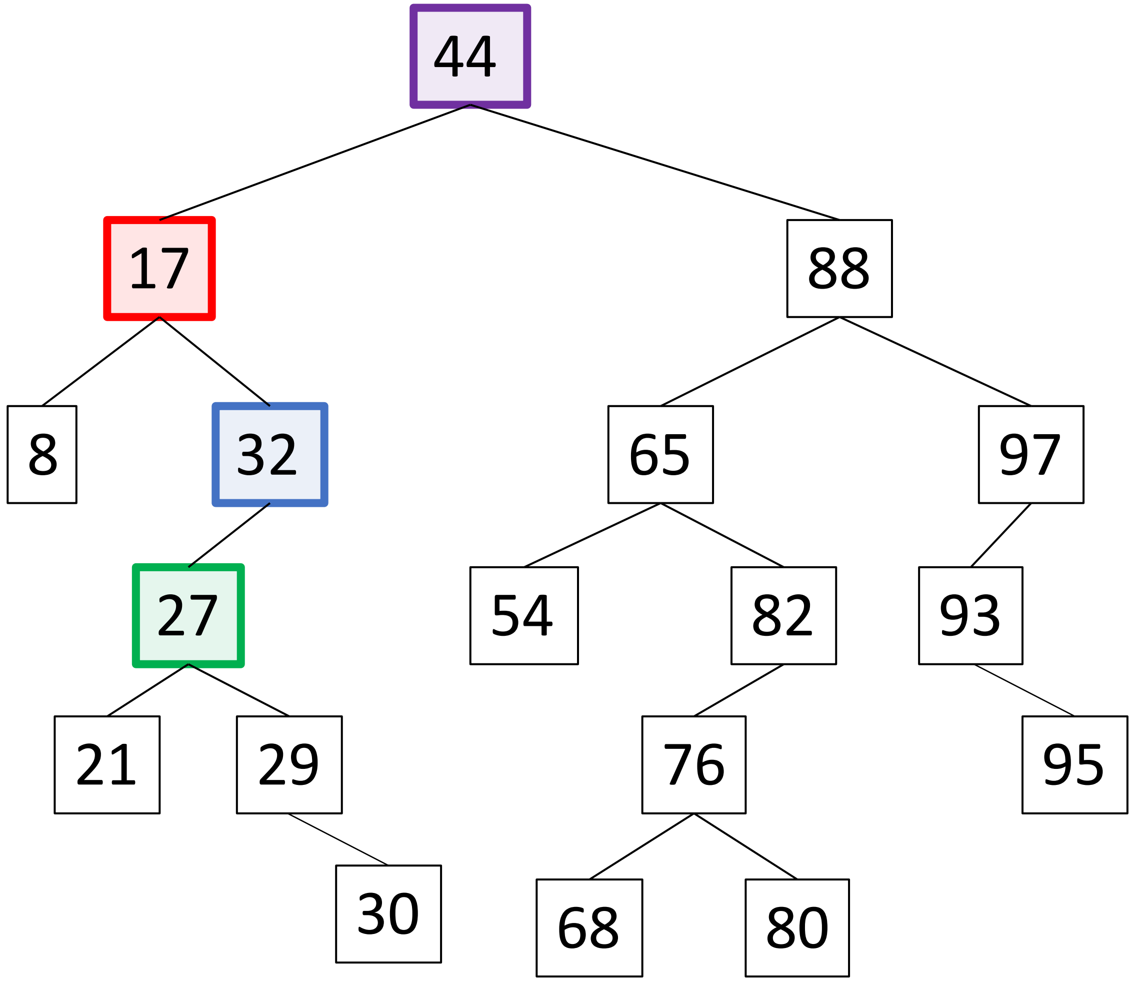
Output:

8

21

Binary Search Tree - Traversal

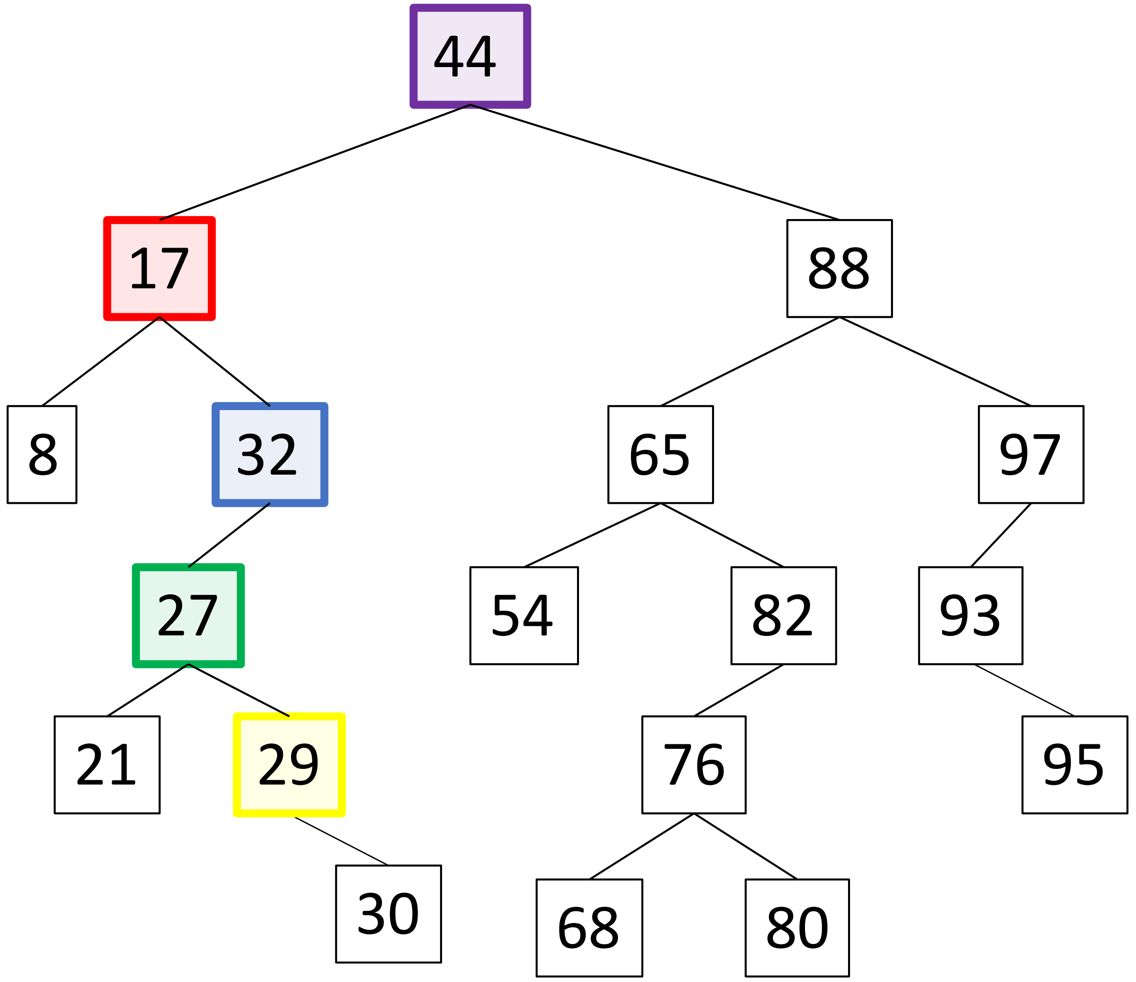
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8
21

Binary Search Tree - Traversal

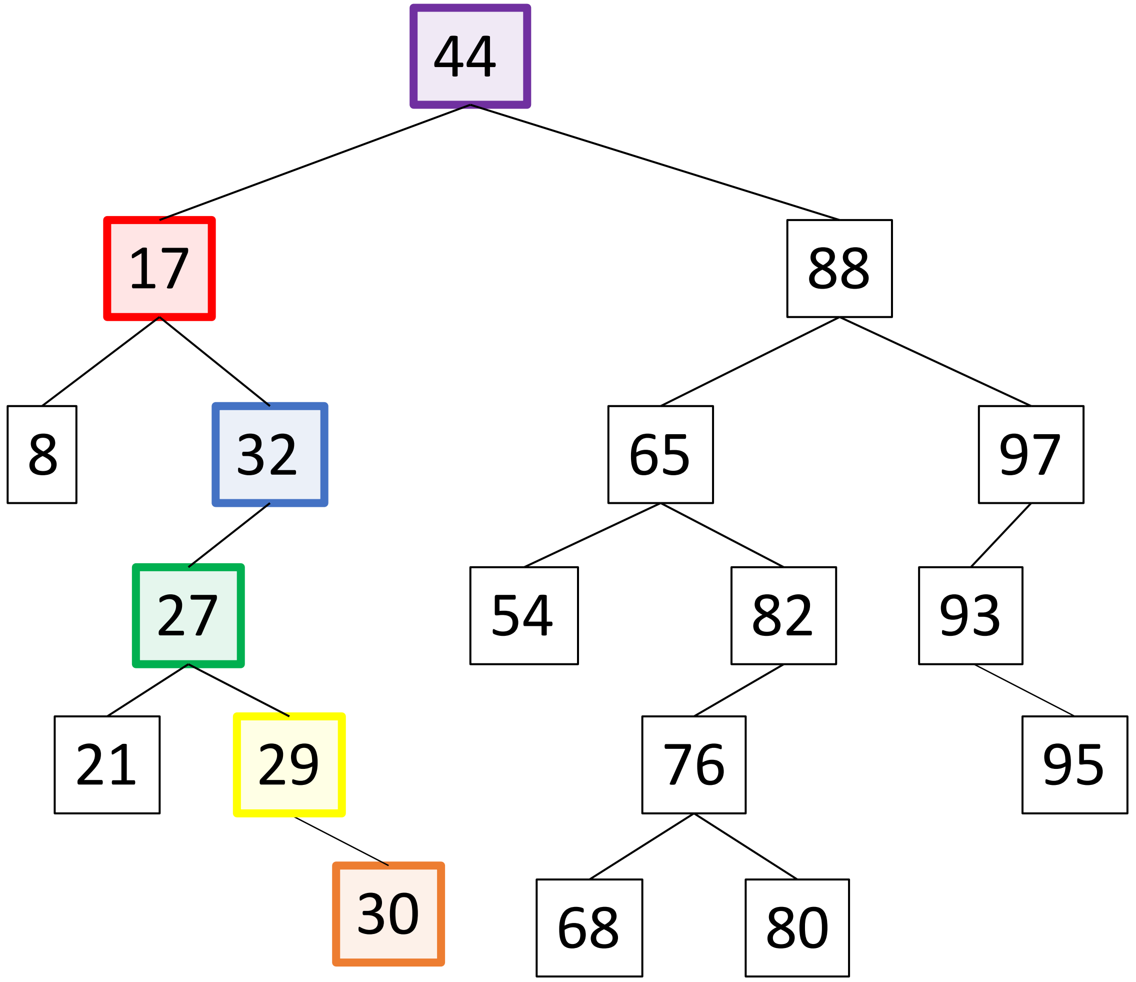
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8
21

Binary Search Tree - Traversal

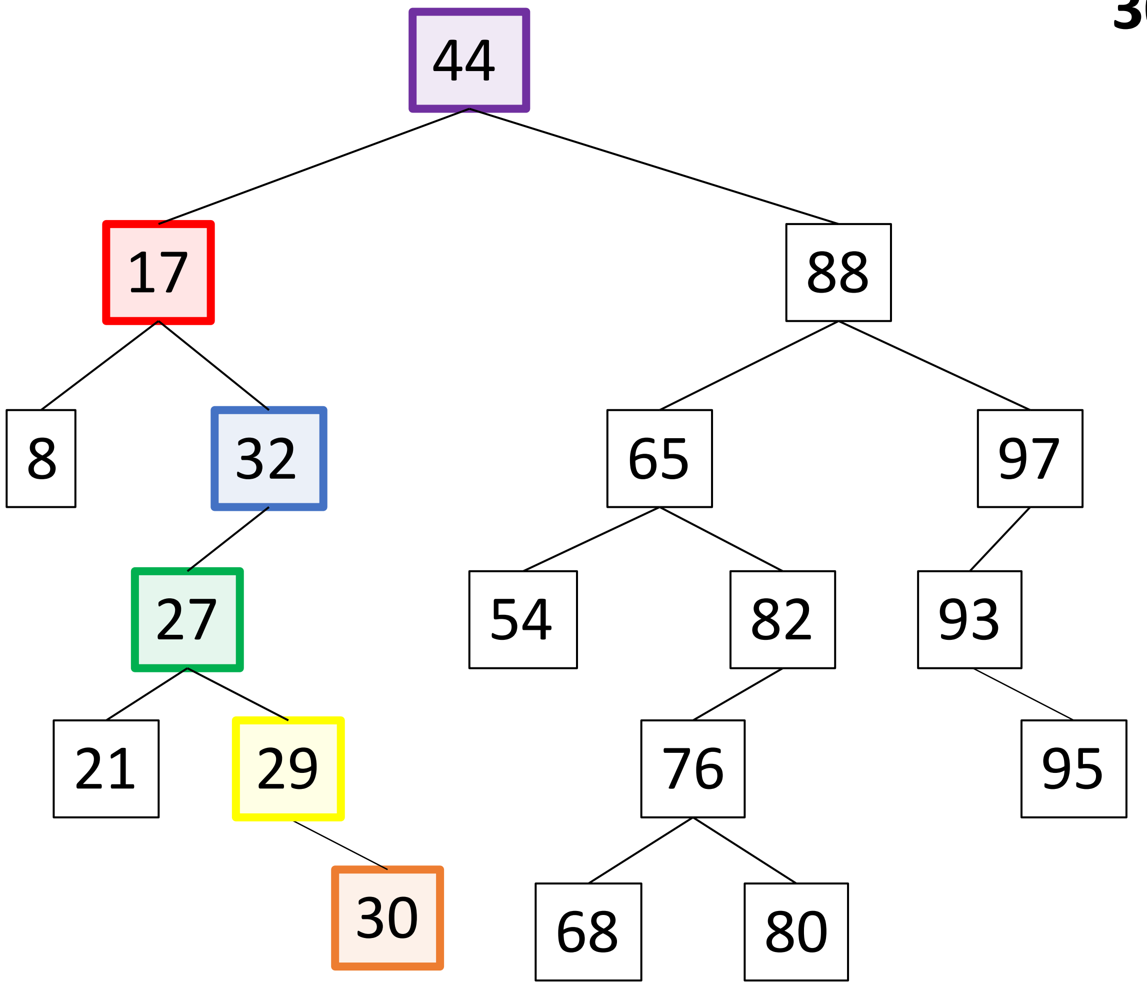
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8
21
30

Binary Search Tree - Traversal

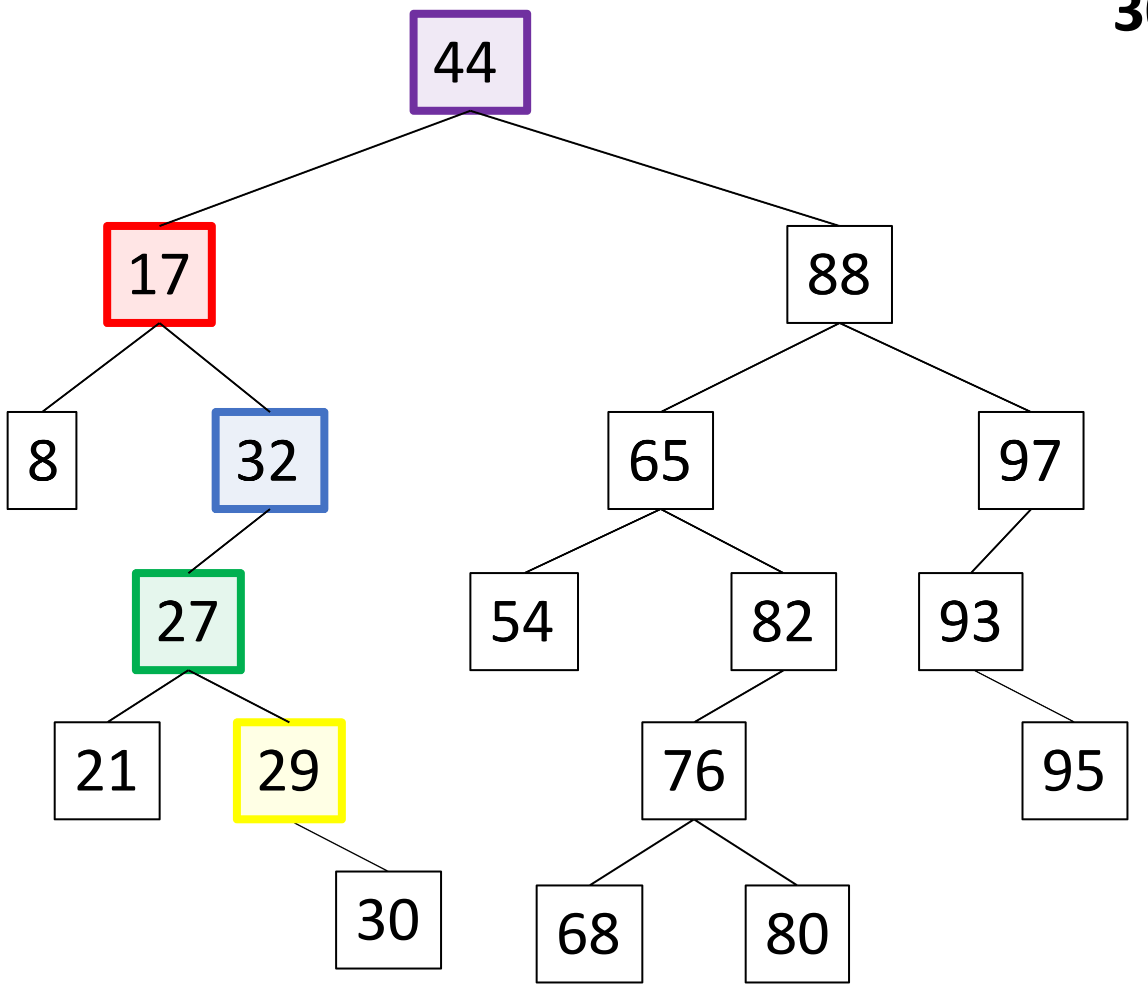
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8
21
30

Binary Search Tree - Traversal

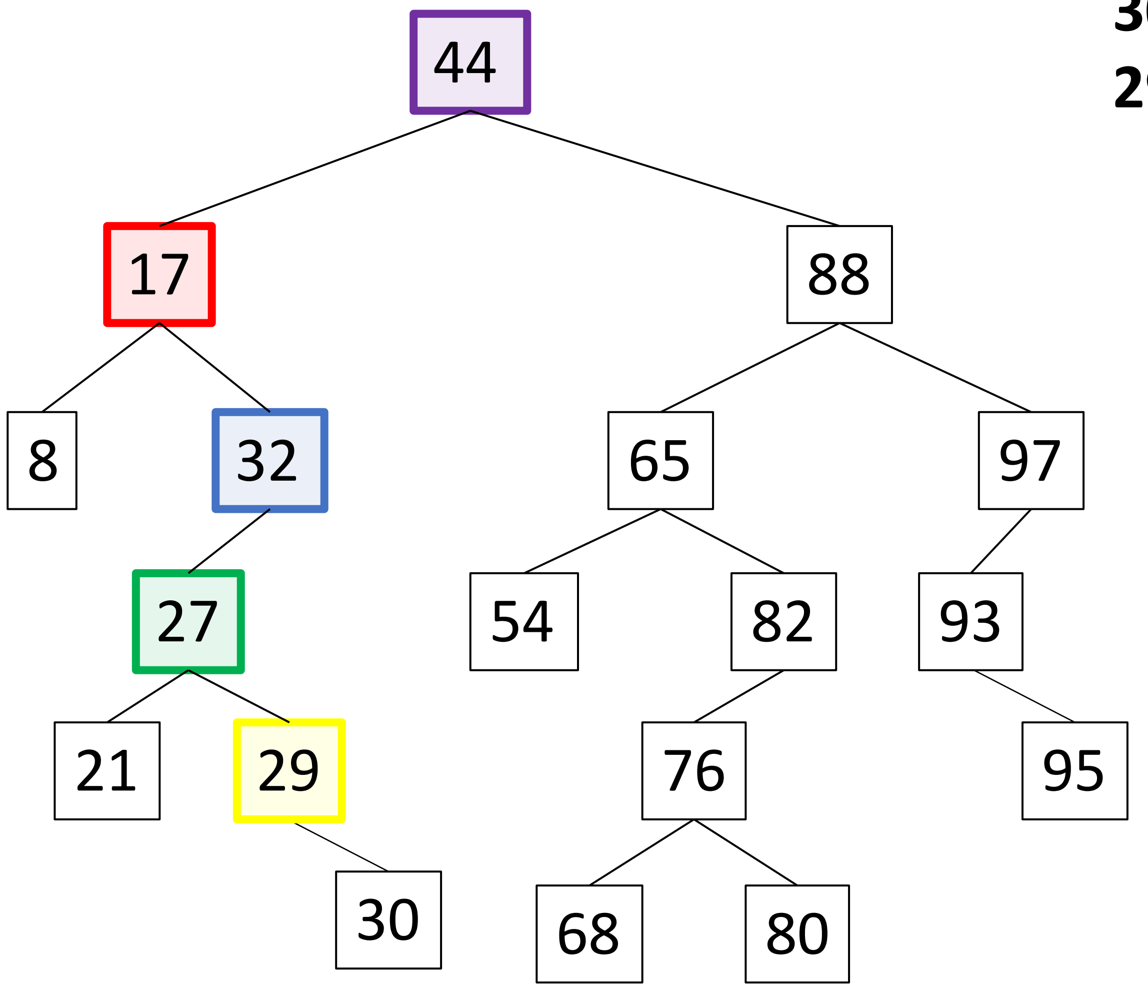
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8
21
30
29

Binary Search Tree - Traversal

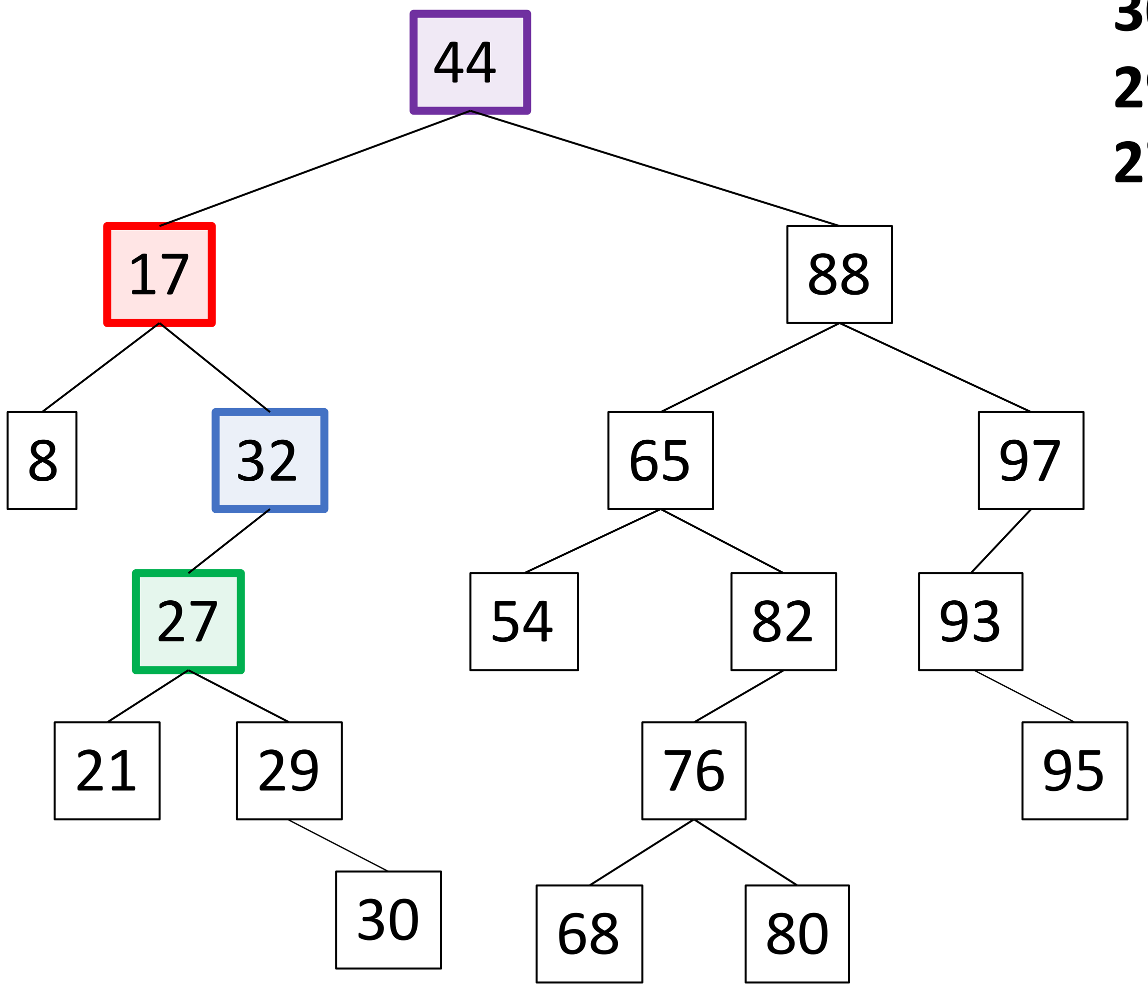
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8
21
30
29
27

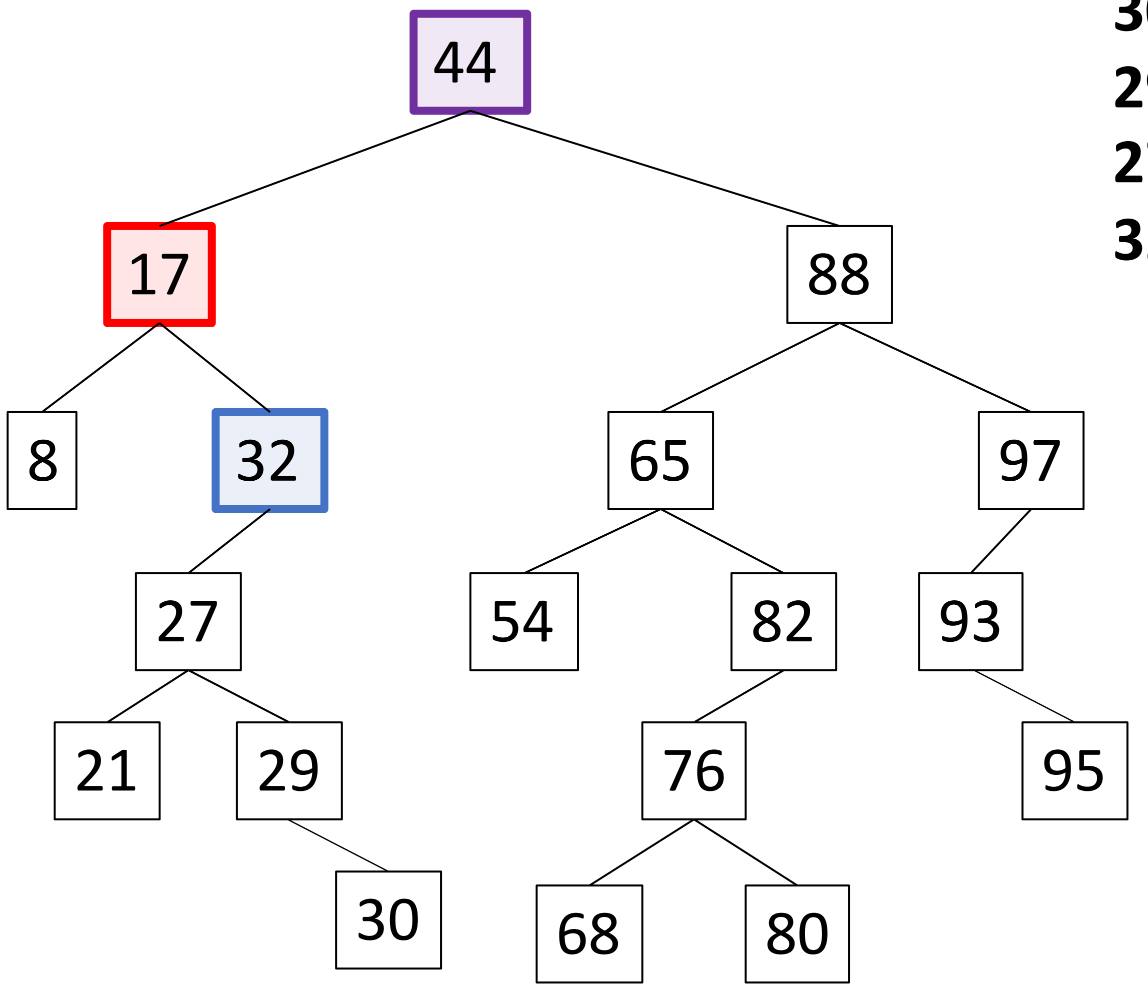
Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:

8

21

30

29

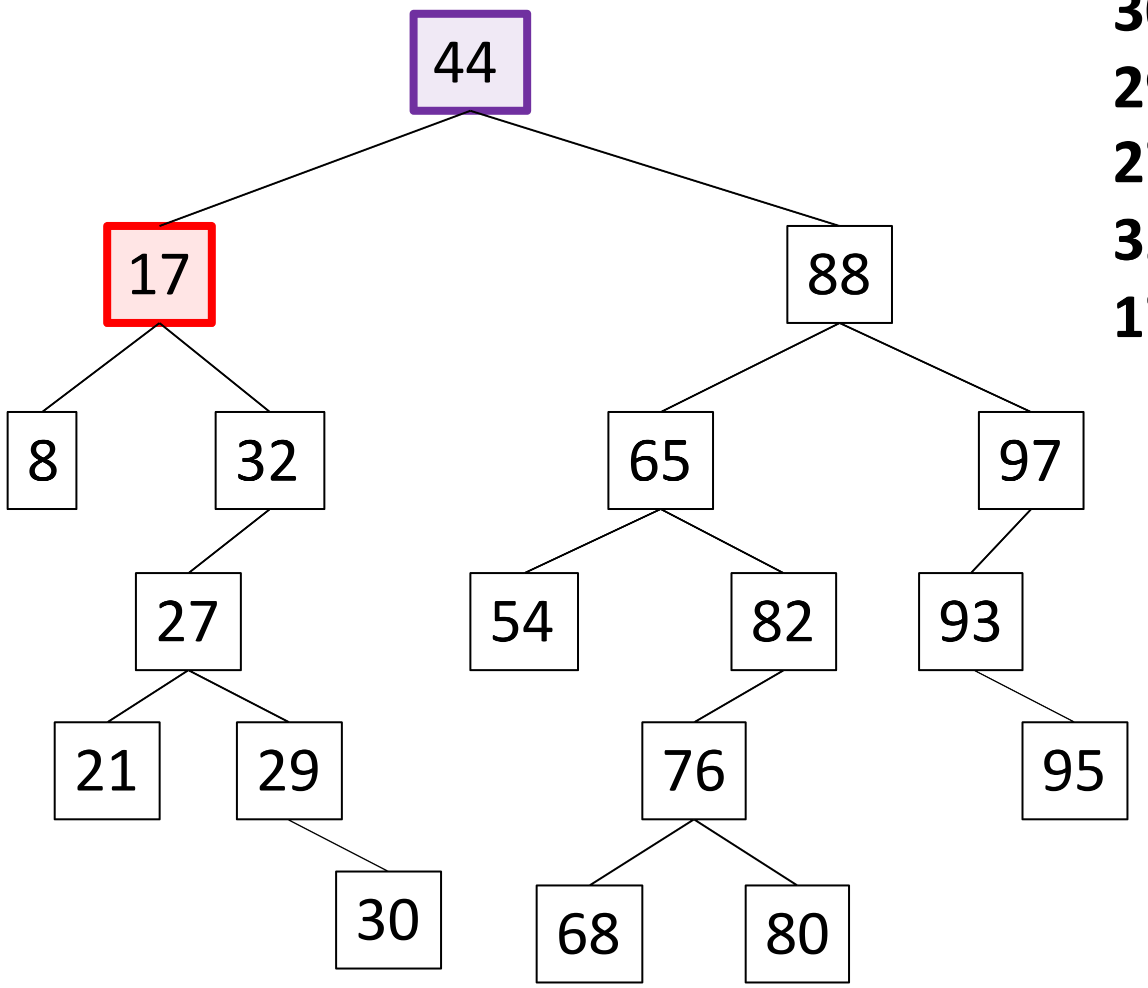
27

32

Output:
8
21
30
29
27
32
17

Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:

8

21

30

29

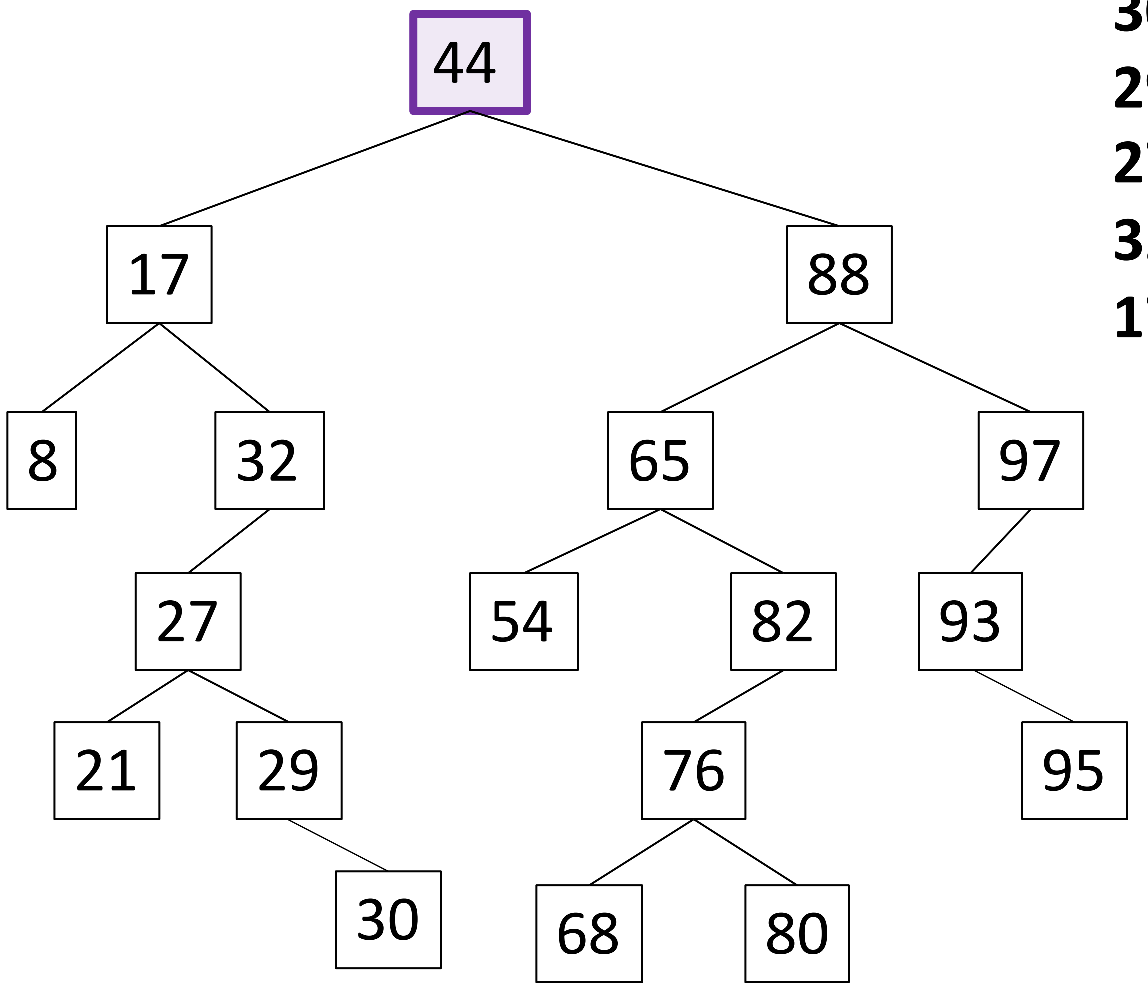
27

32

17

Binary Search Tree - Traversal

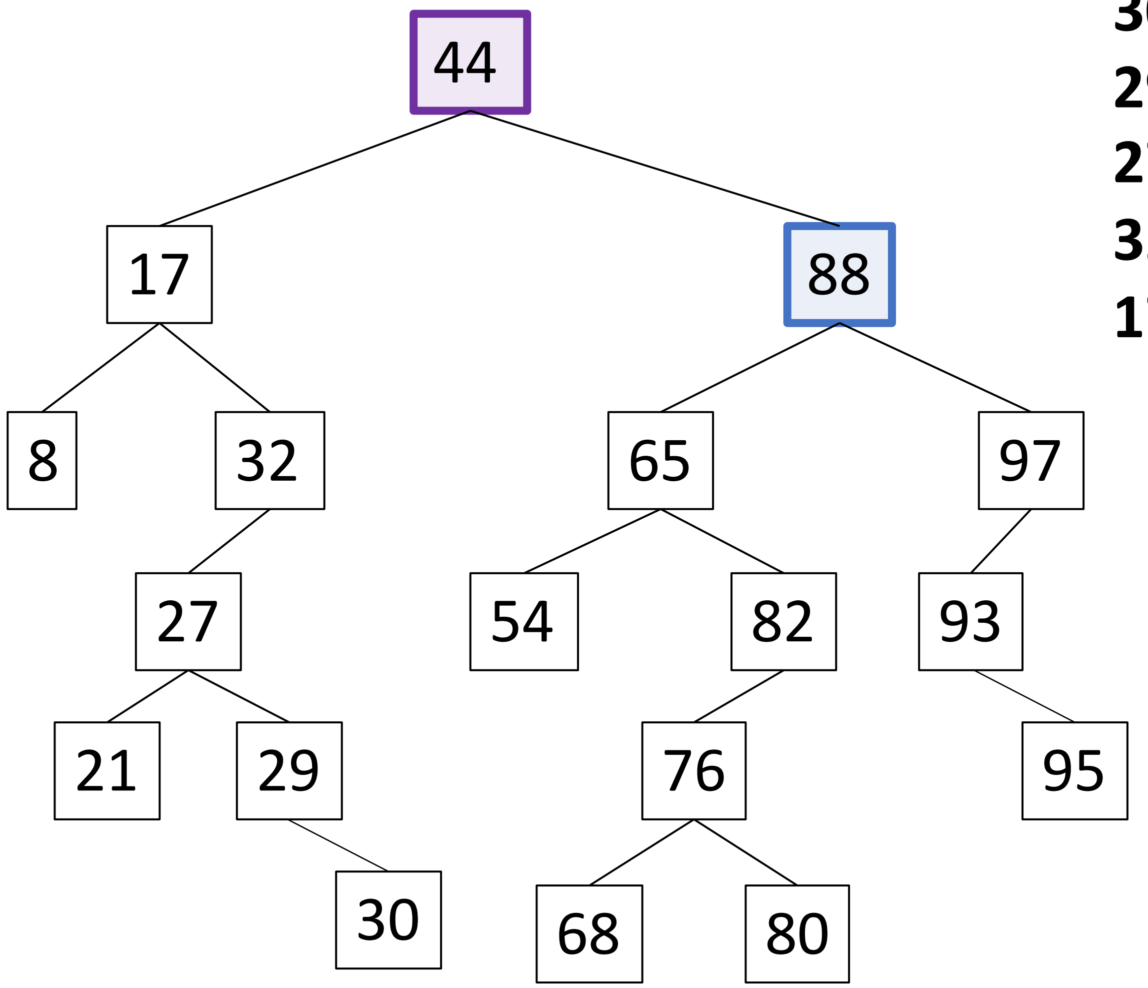
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8
21
30
29
27
32
17

Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:

8

21

30

29

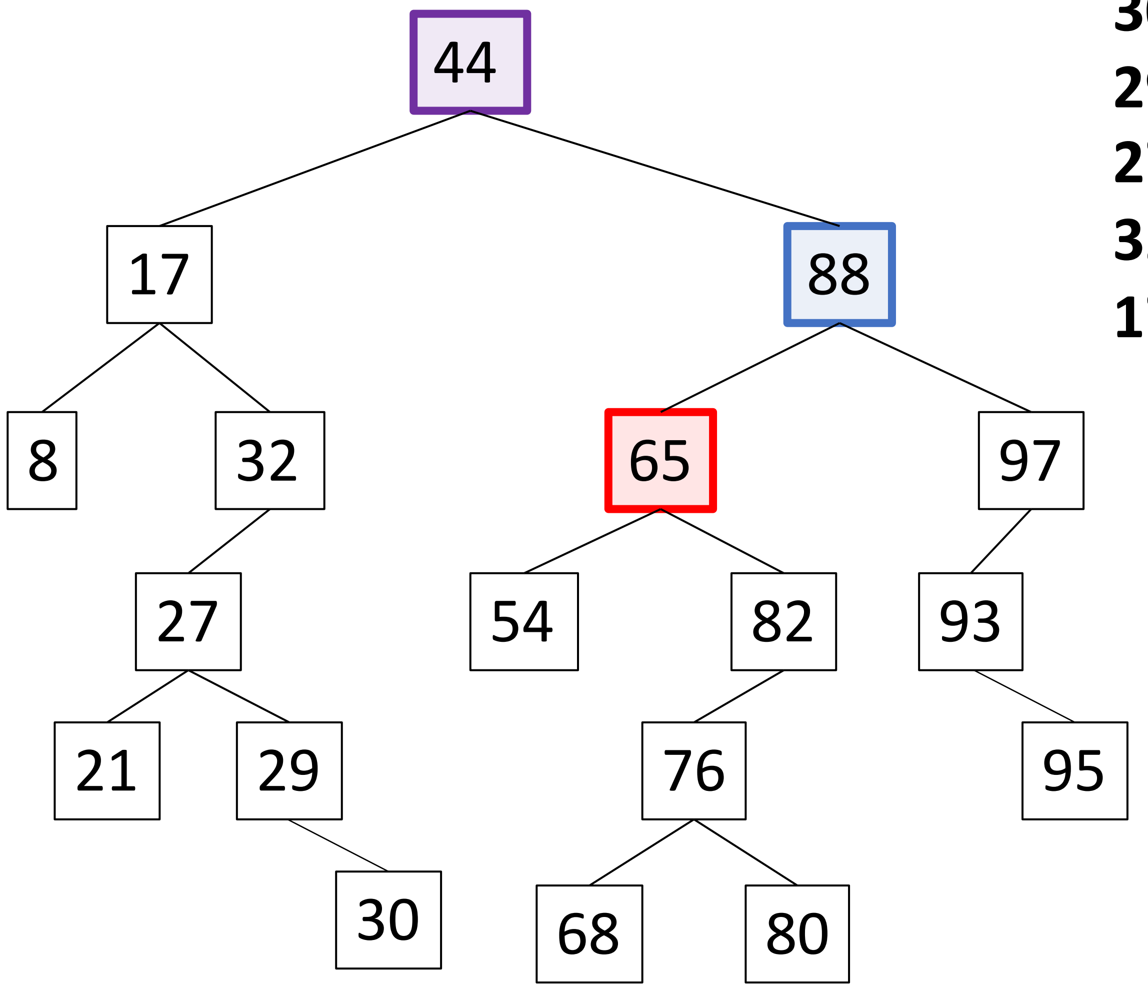
27

32

17

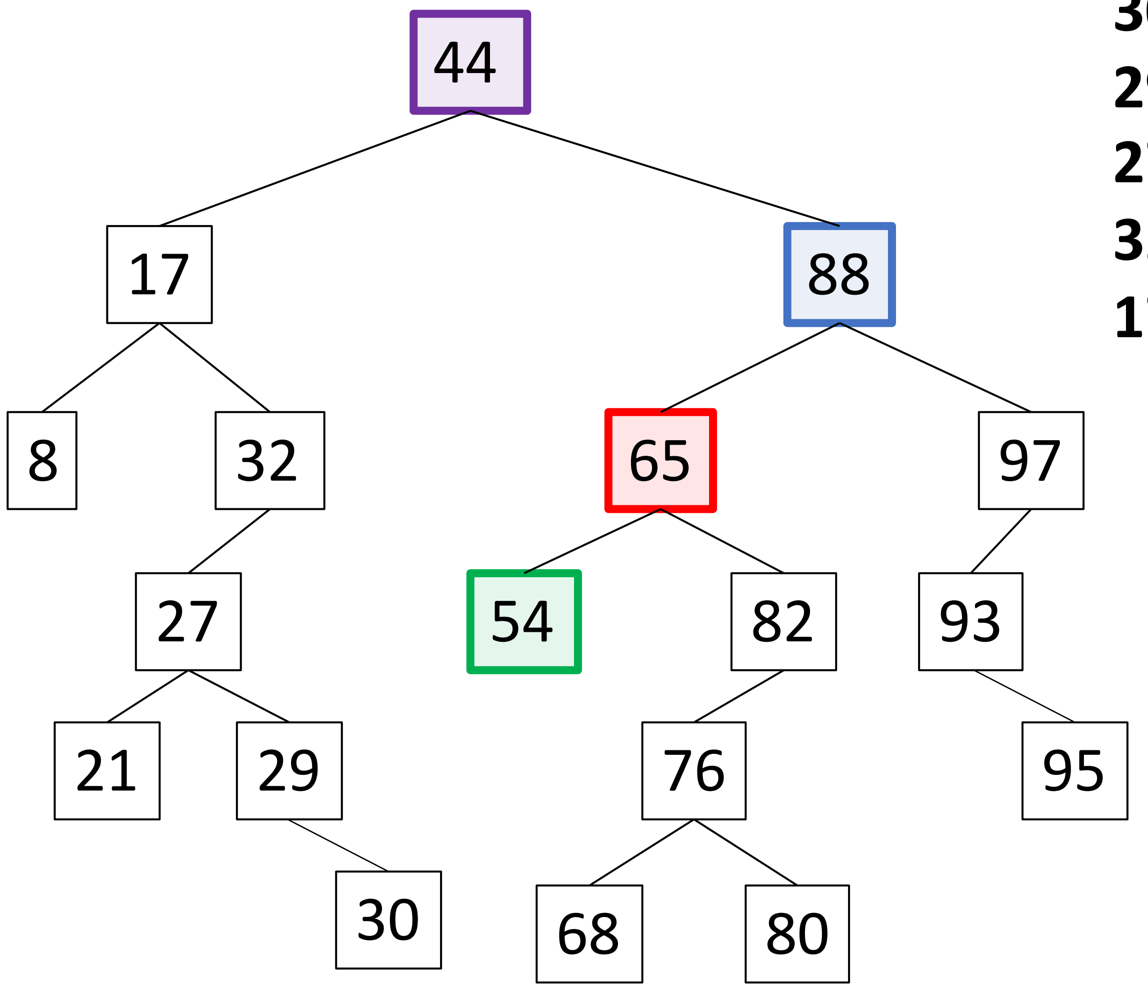
Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



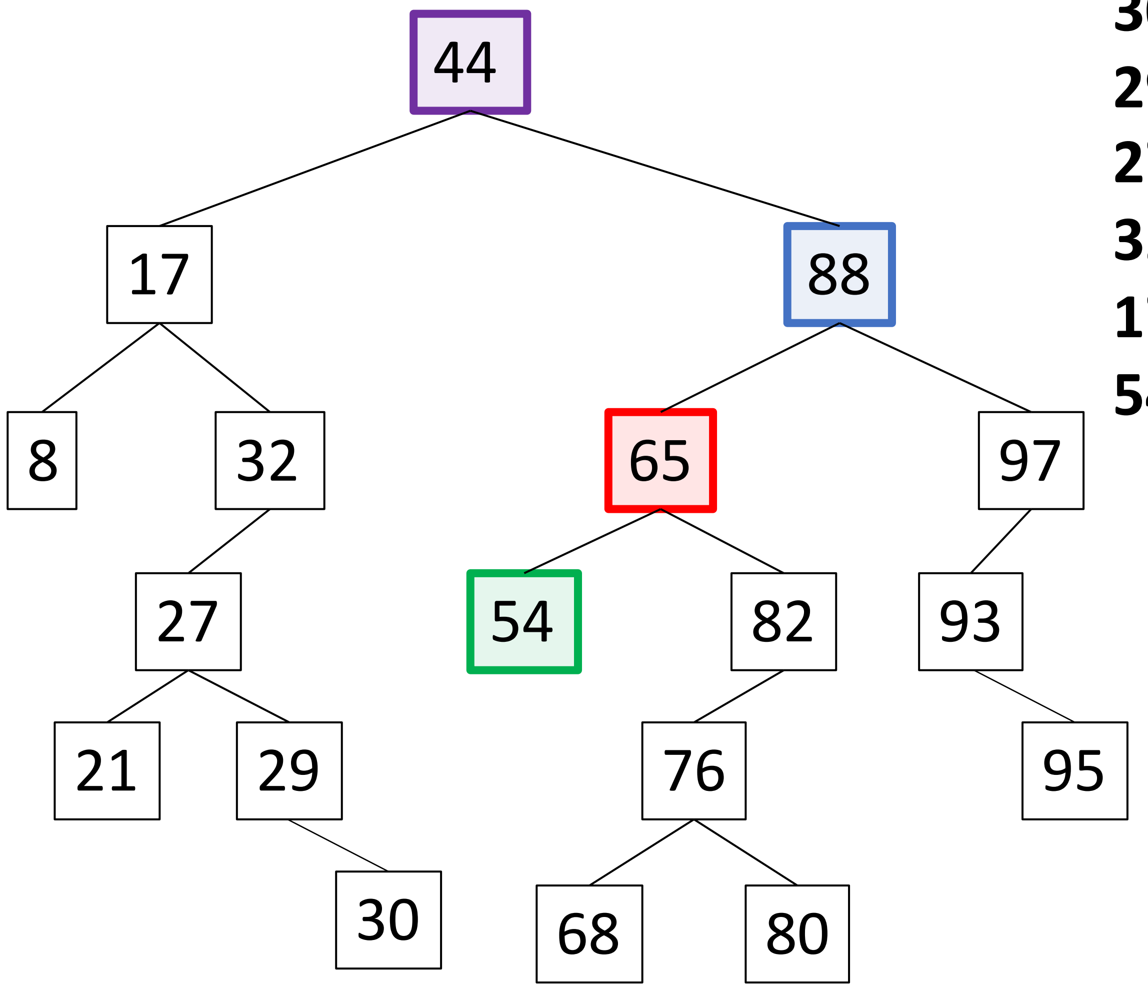
Output:

```
 8
21
30
29
27
32
17
```

Output:
8
21
30
29
27
32
17
54

Binary Search Tree - Traversal

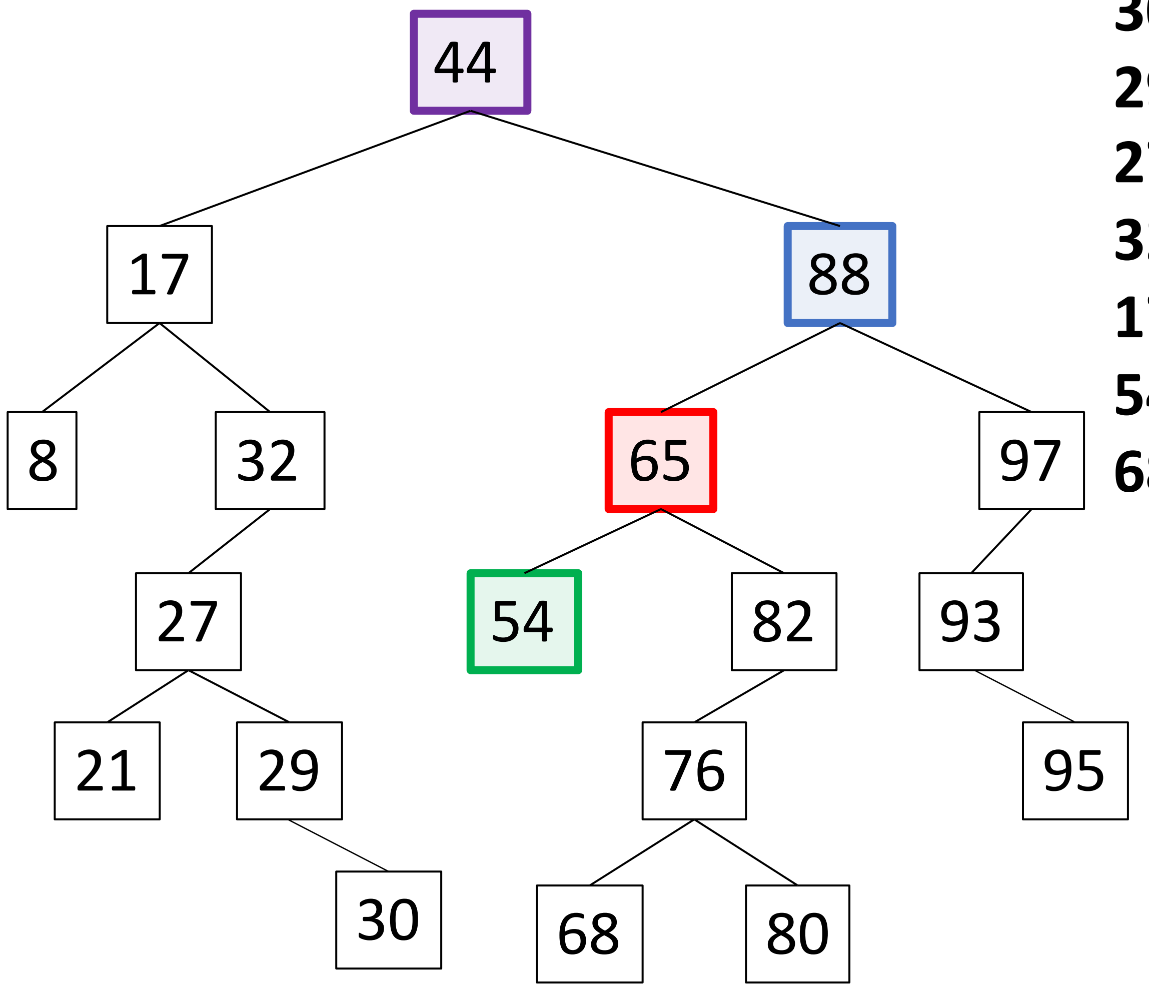
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:
8
21
30
29
27
32
17
54
68

Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Output:

8

21

30

29

27

32

17

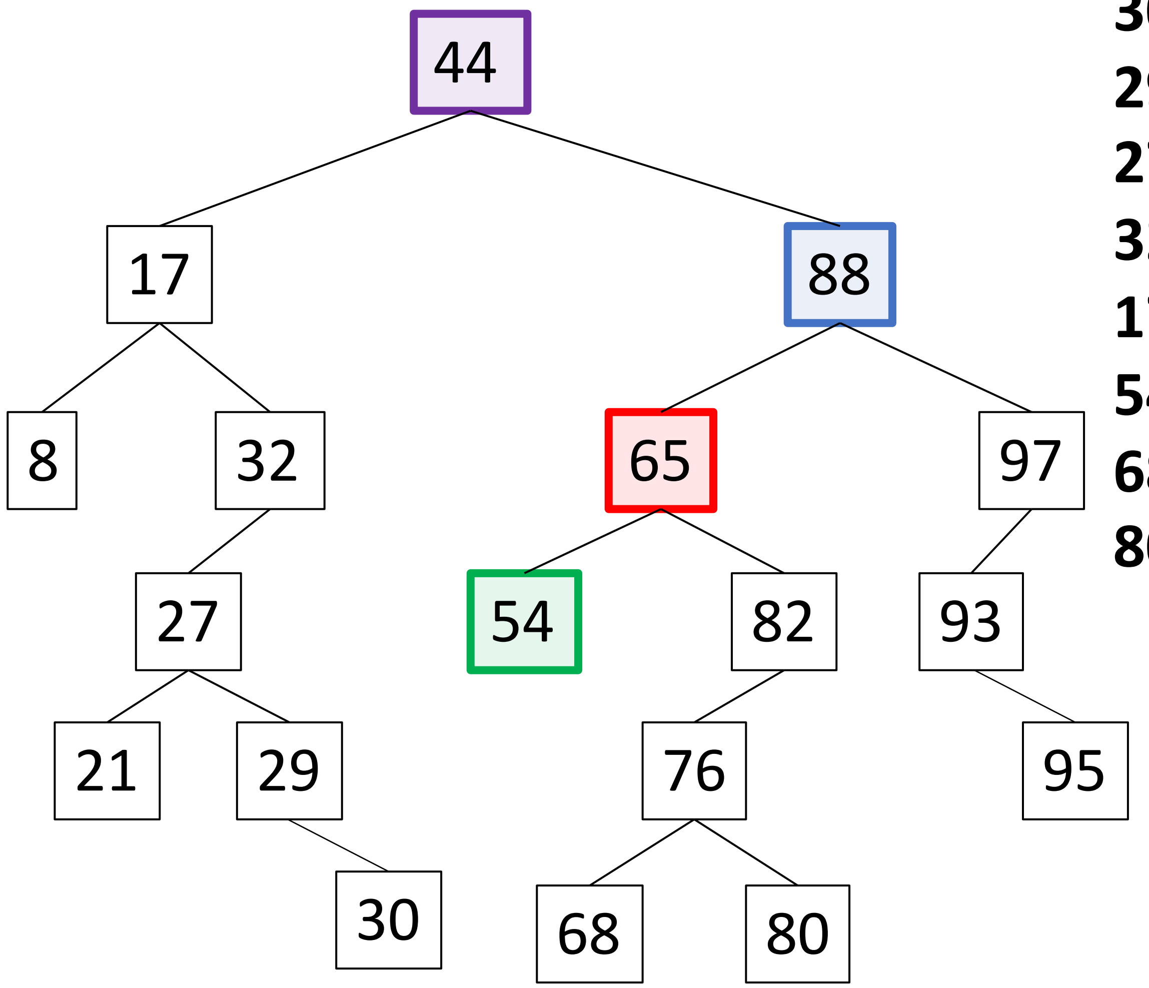
54

68

80

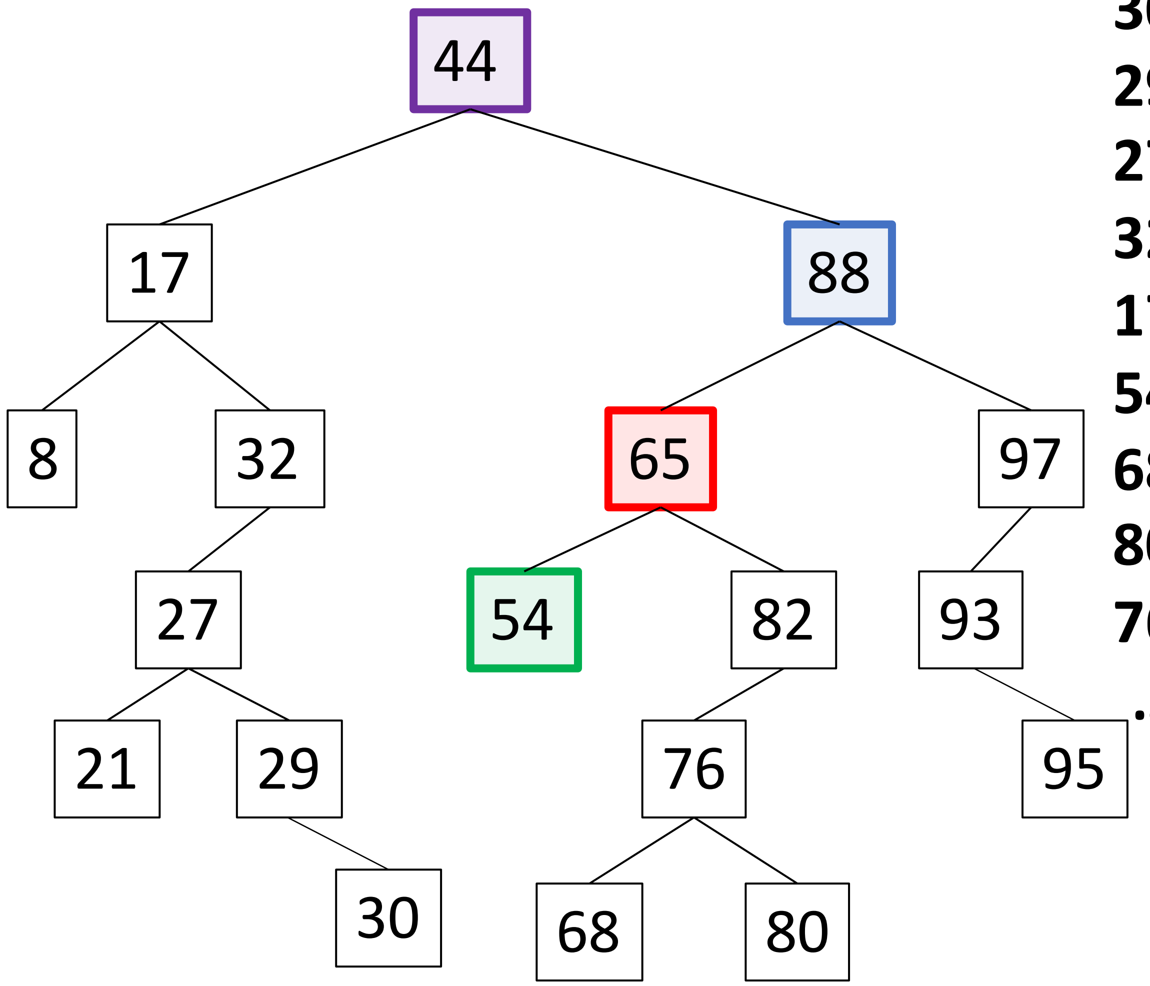
Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```



Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```

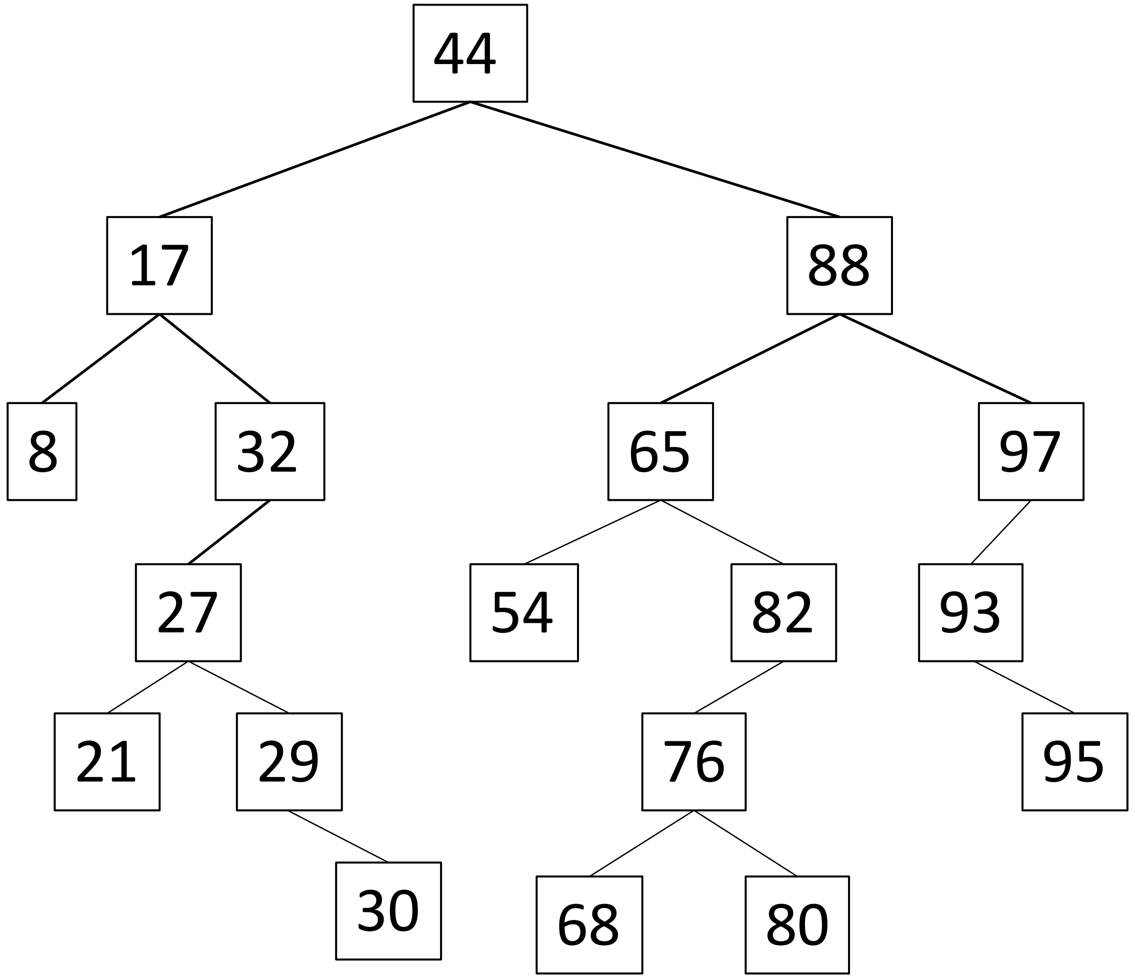


Output:
8
21
30
29
27
32
17
54
68
80
76
…

Output:

Binary Search Tree - Traversal

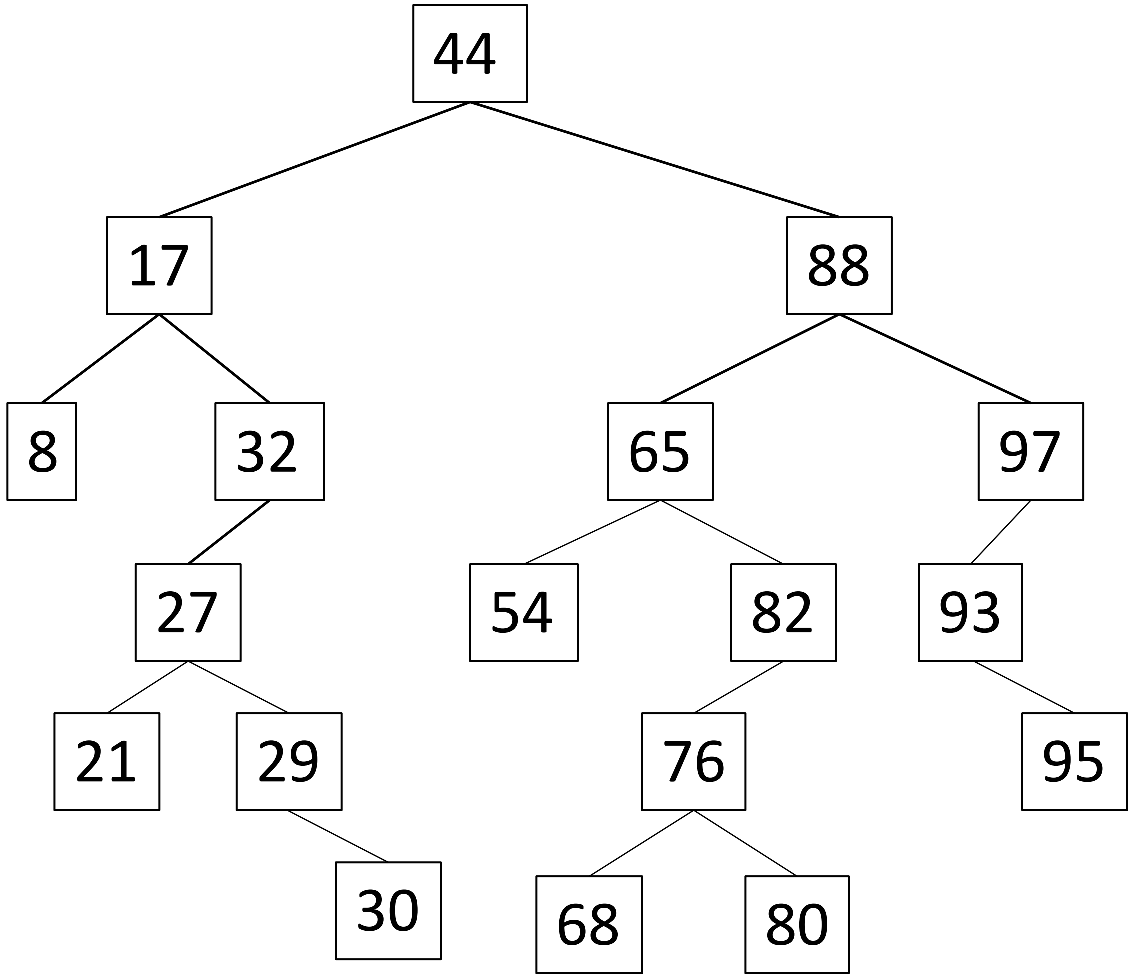
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Output:
8

Binary Search Tree - Traversal

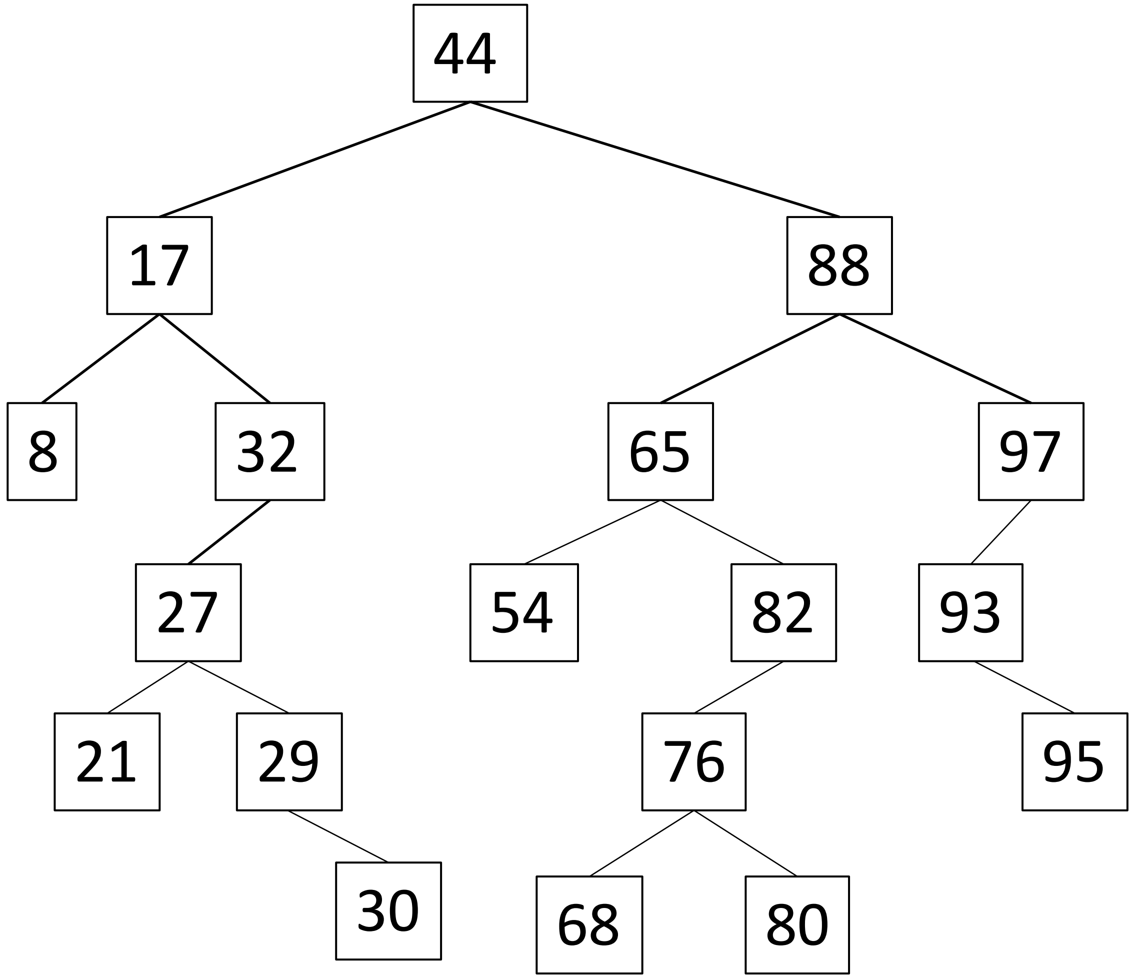
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Output:
8
17

Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Output:

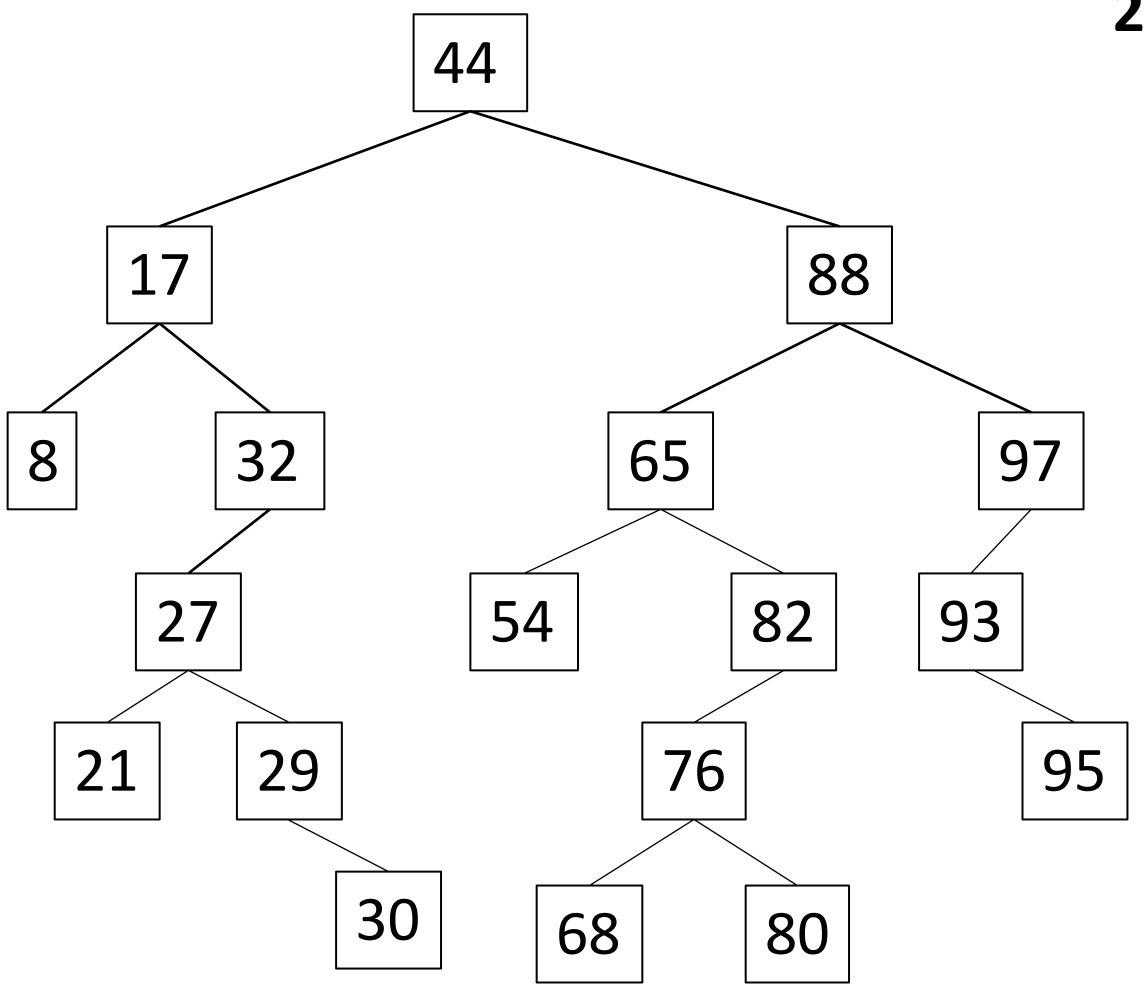
8

17

21

Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Output:

8

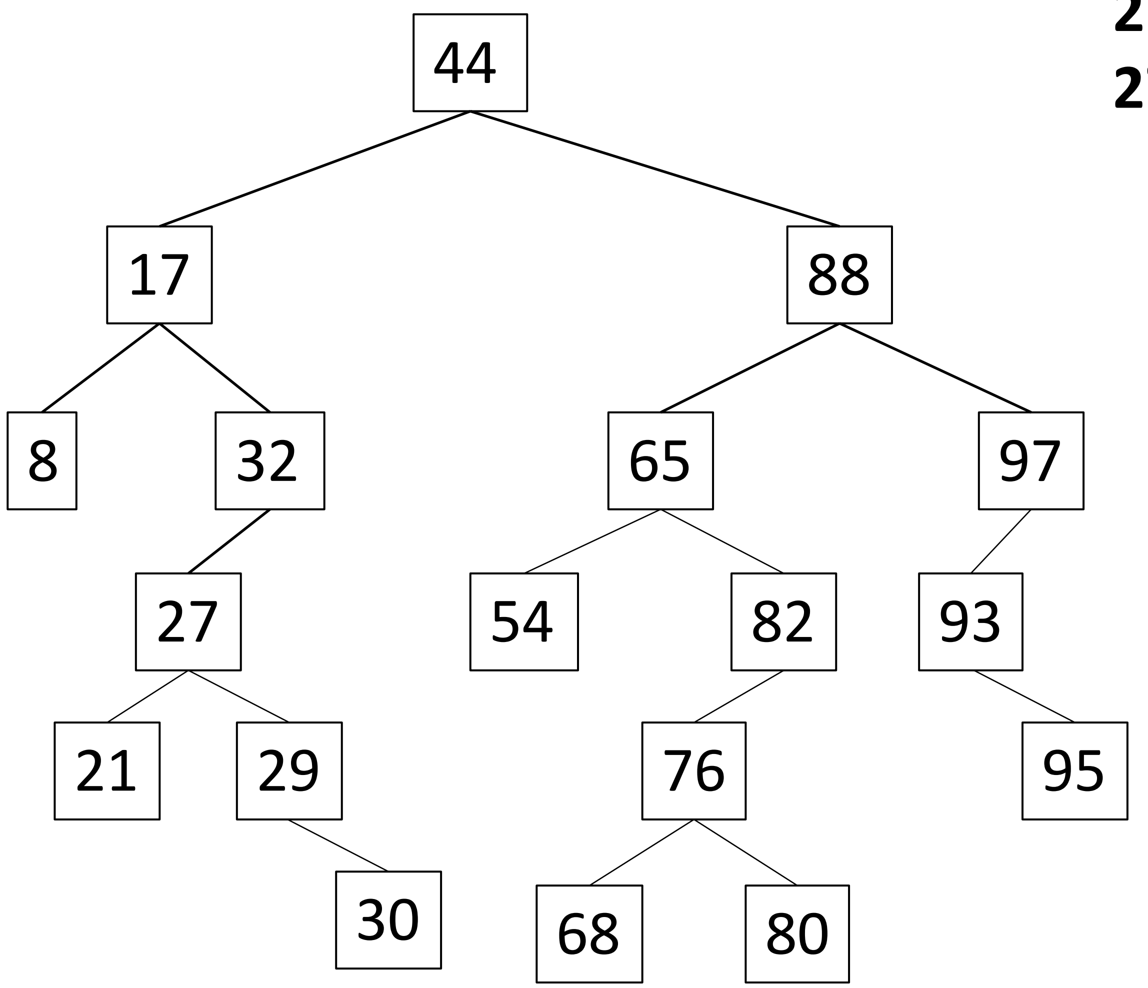
17

21

27

Binary Search Tree - Traversal

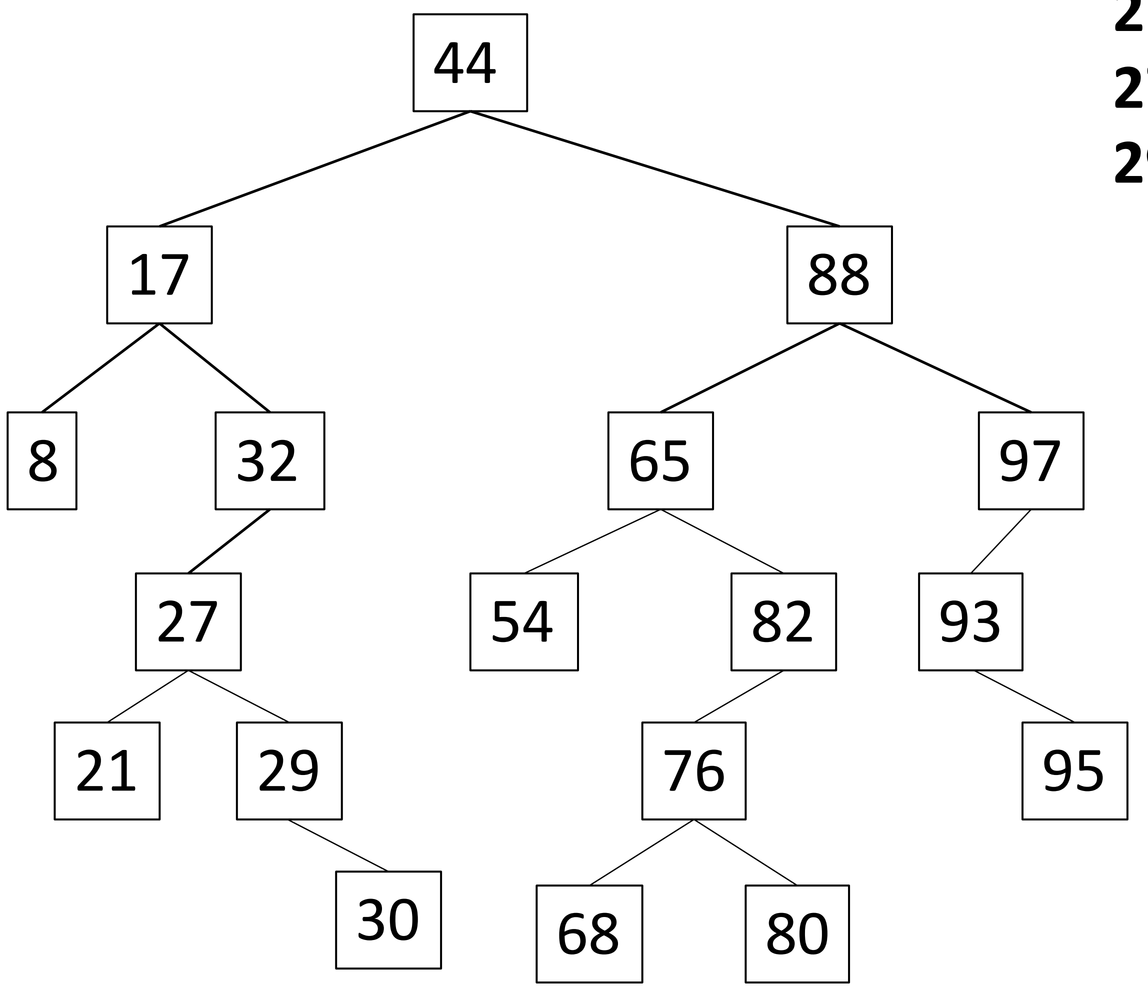
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Binary Search Tree - Traversal

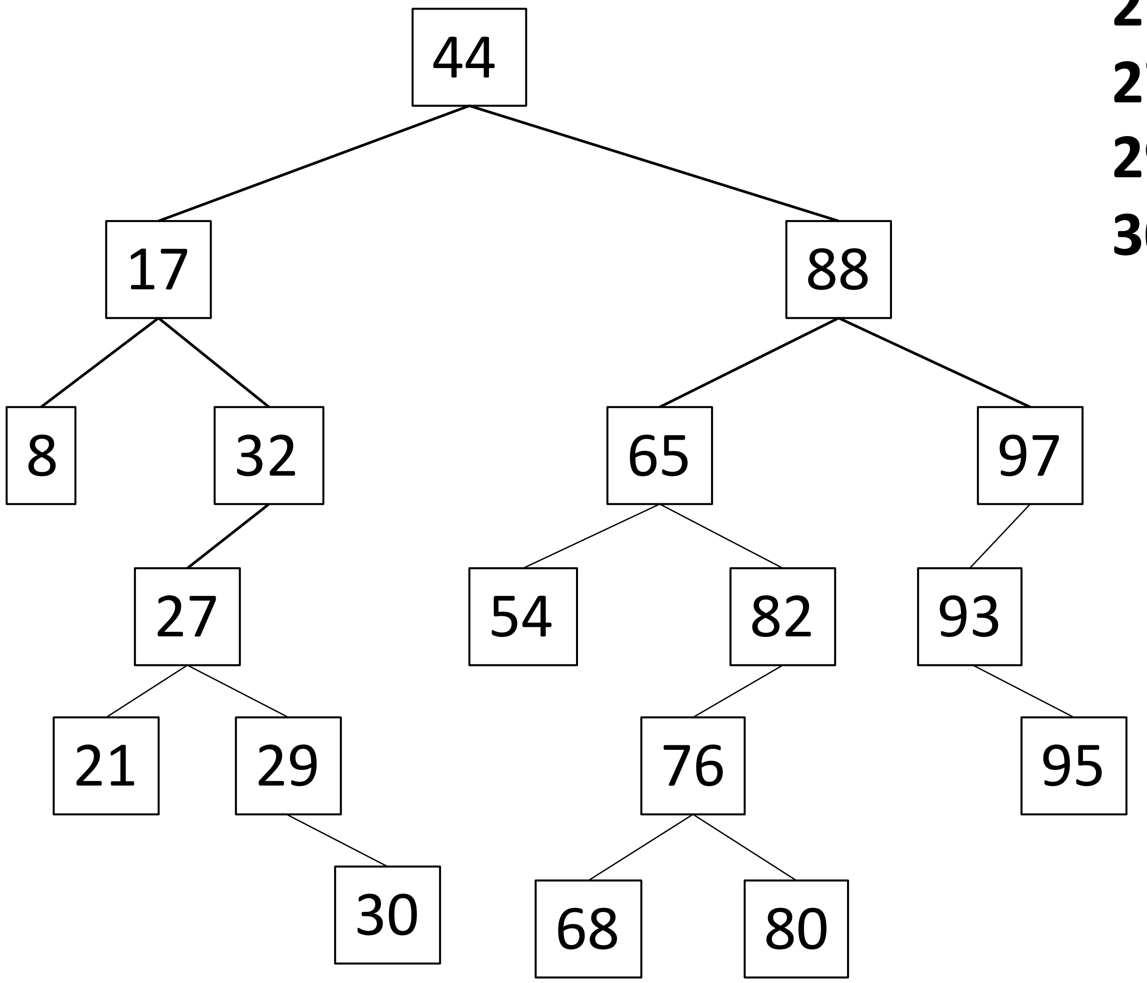
Output:
8
17
21
27
29

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Output:

8

17

21

27

29

30

Output:

8

17

21

27

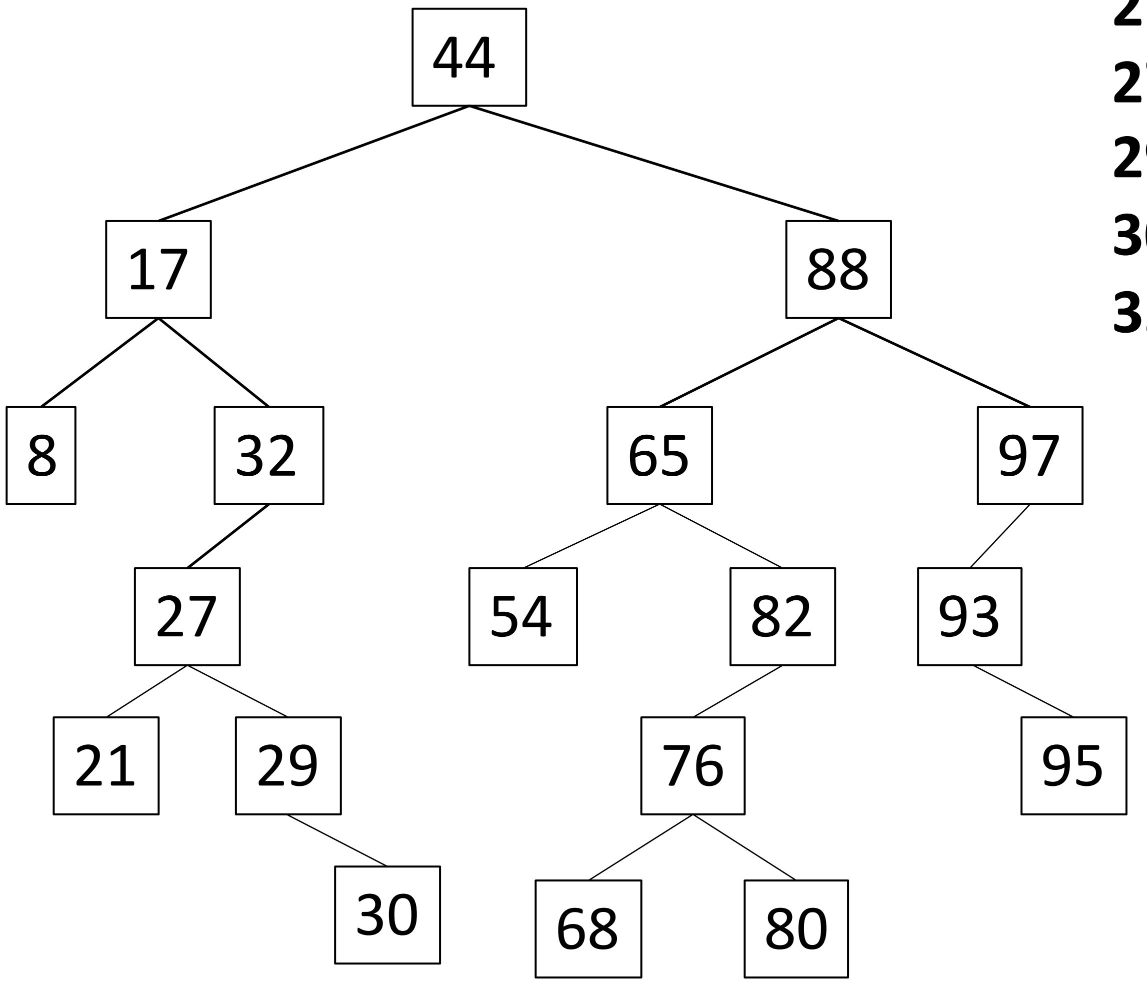
29

30

32

Binary Search Tree - Traversal

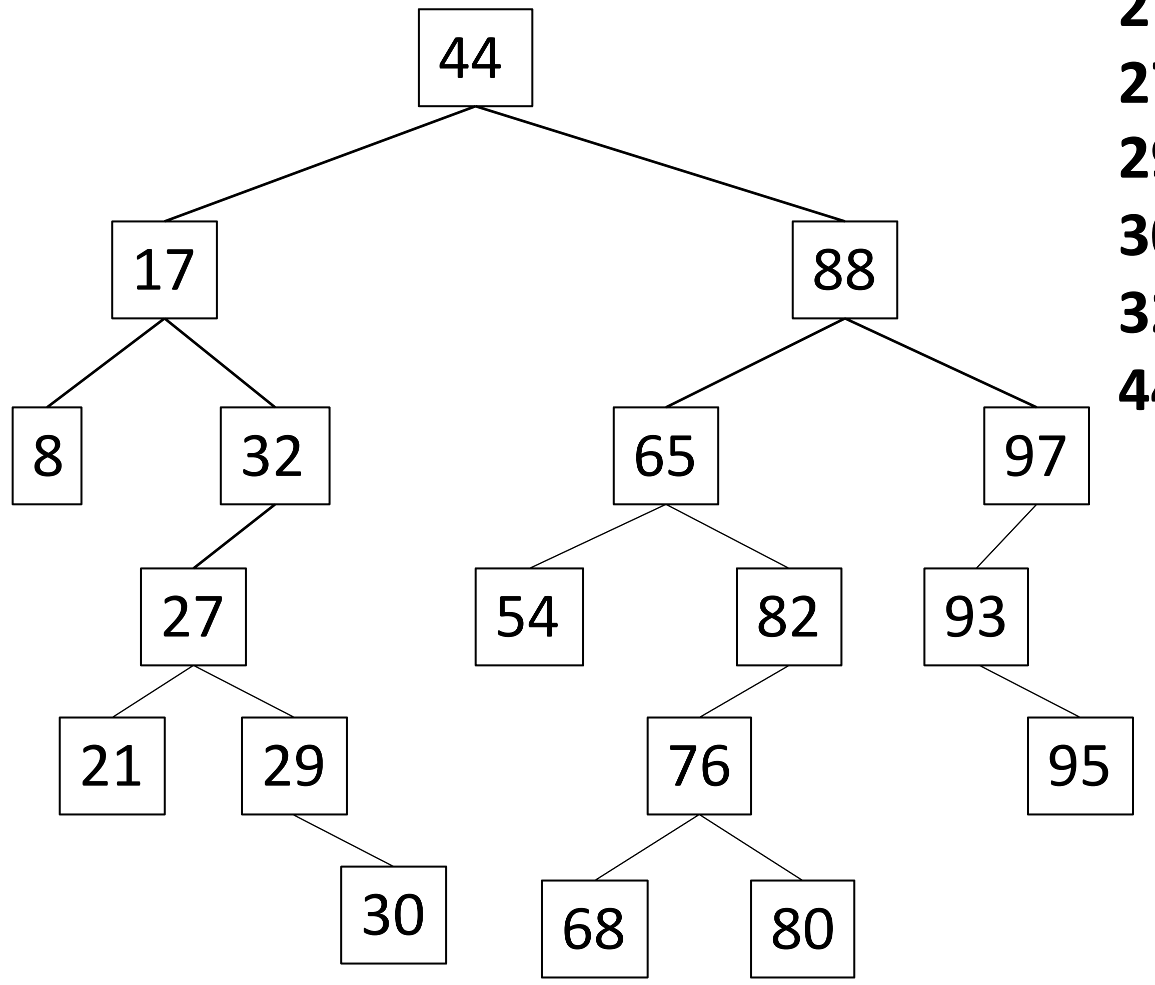
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Output:
8
17
21
27
29
30
32
44

Binary Search Tree - Traversal

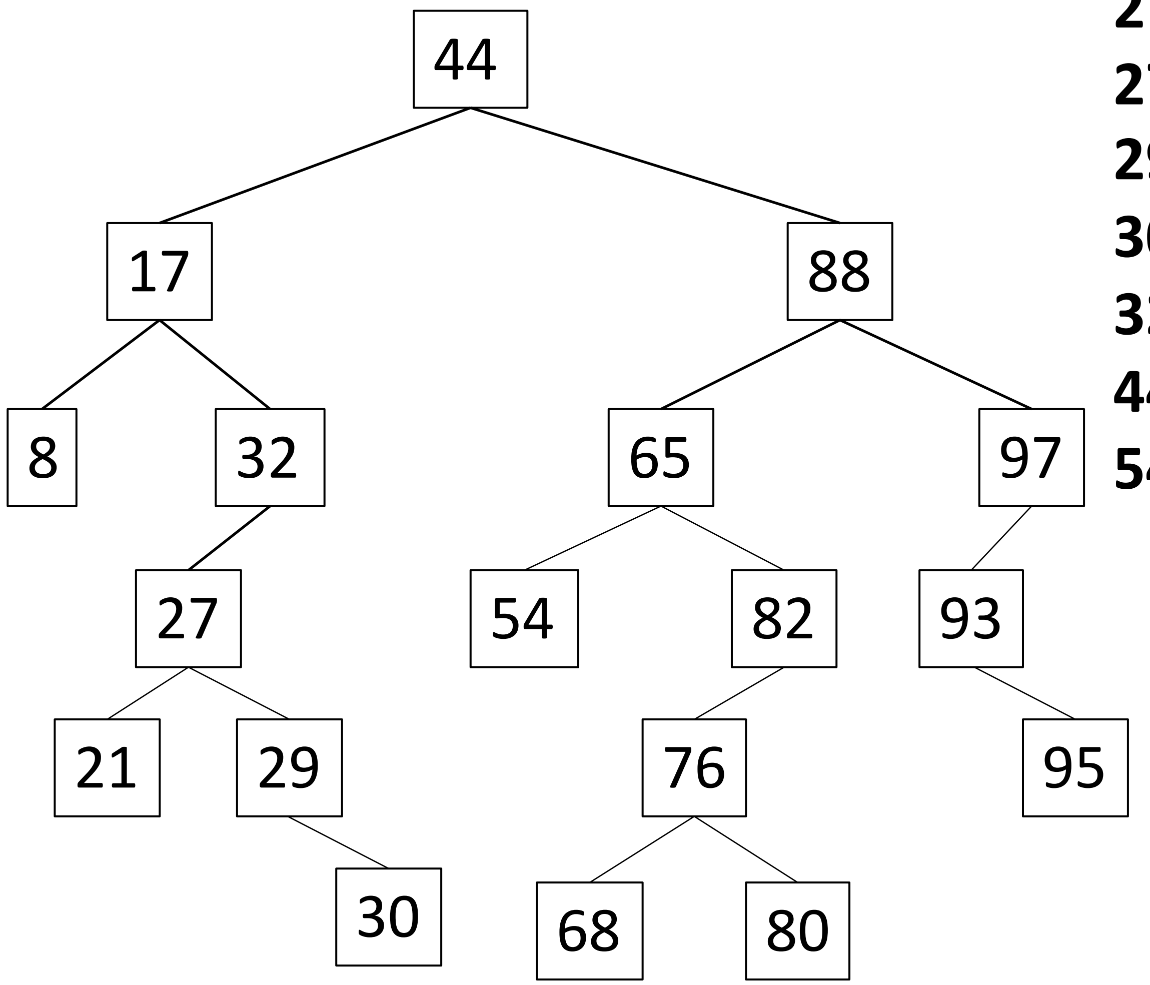
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Output:
8
17
21
27
29
30
32
44
54

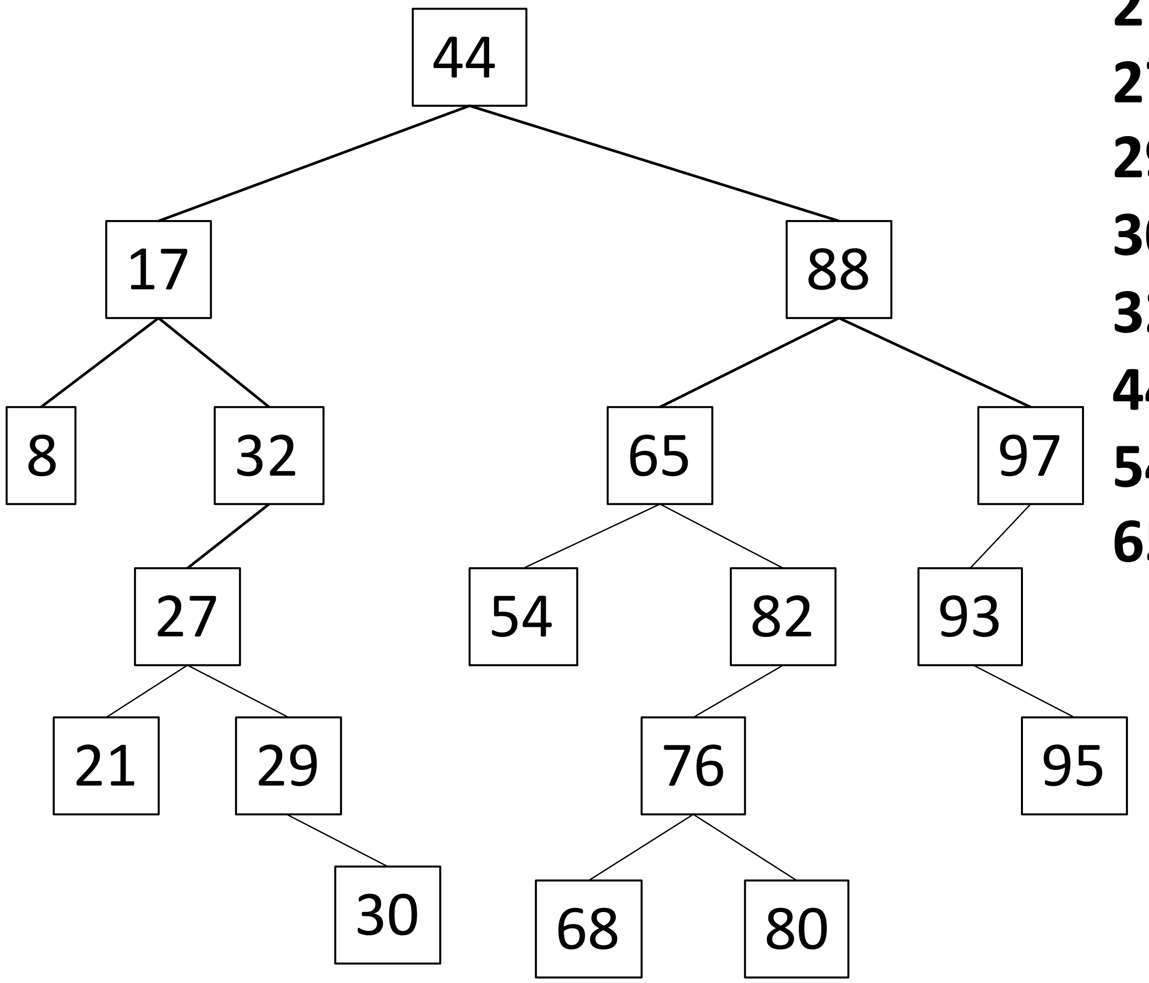
Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```

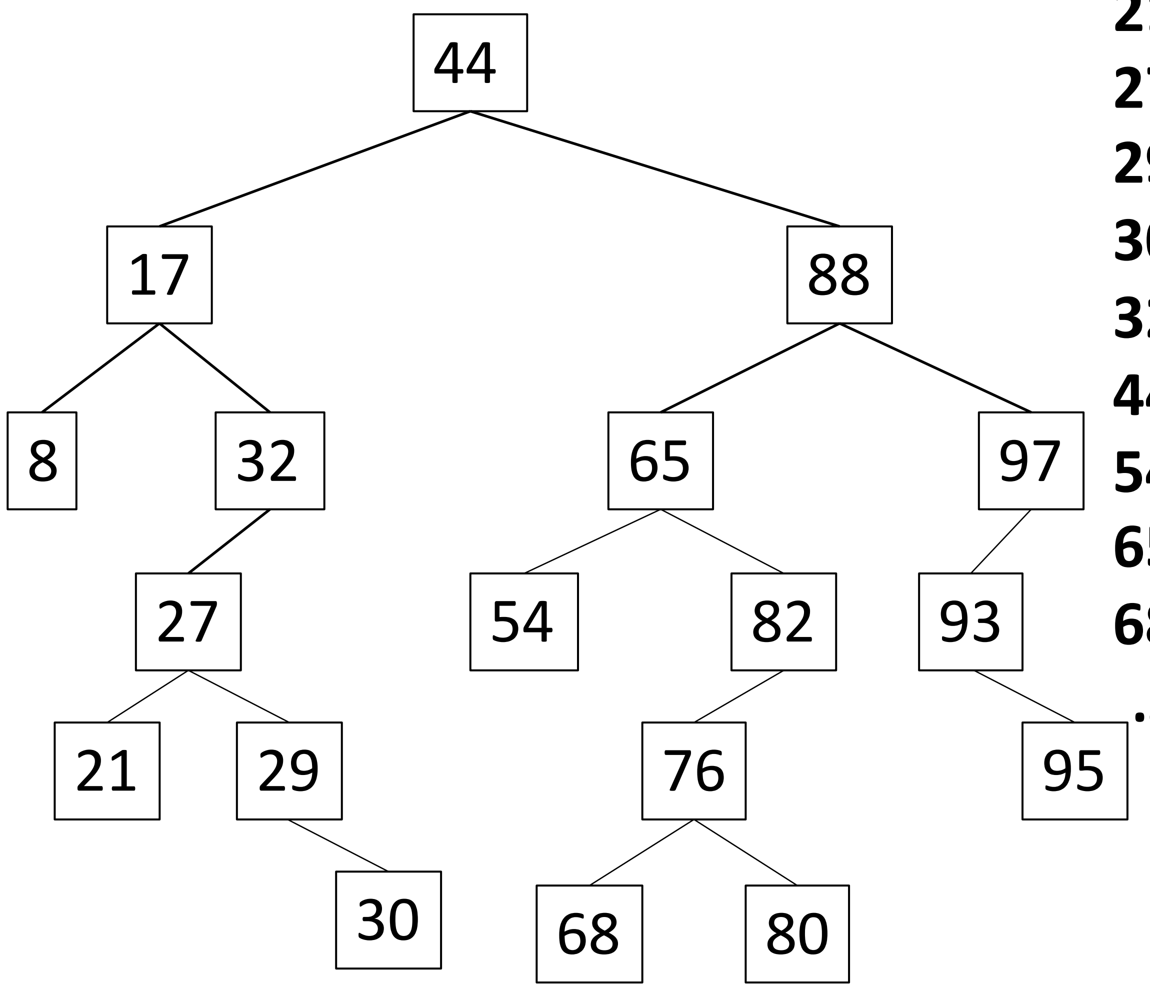


Output:

```
8
17
21
27
29
30
32
44
54
65
```

Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```



Output:

```
8
17
21
27
29
30
32
44
54
65
68
…
```

Binary Search Tree - Traversal

```
public void depthFirst(Node n) {
    if (n != null) {
        System.out.println(n.getValue());
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
    }
}
```

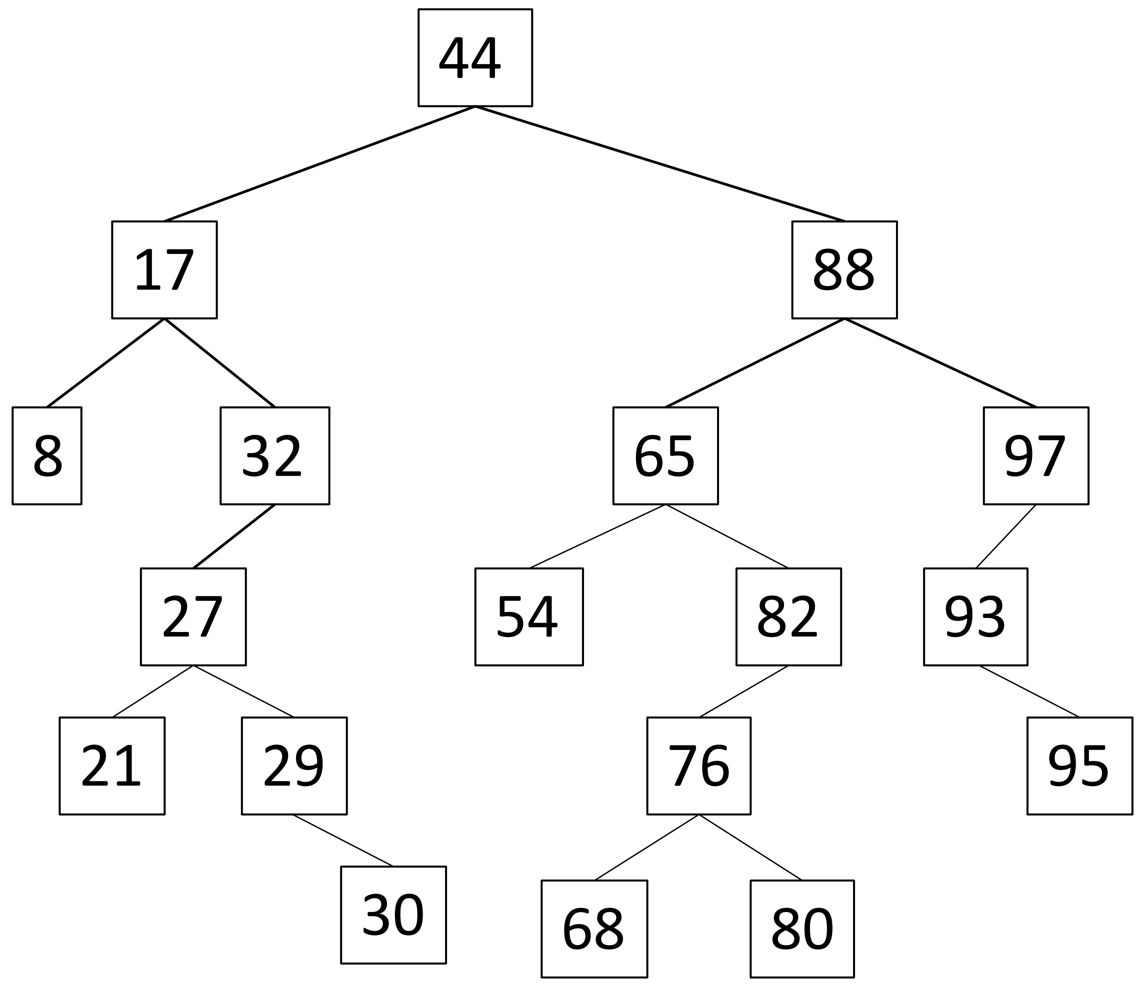
Preorder

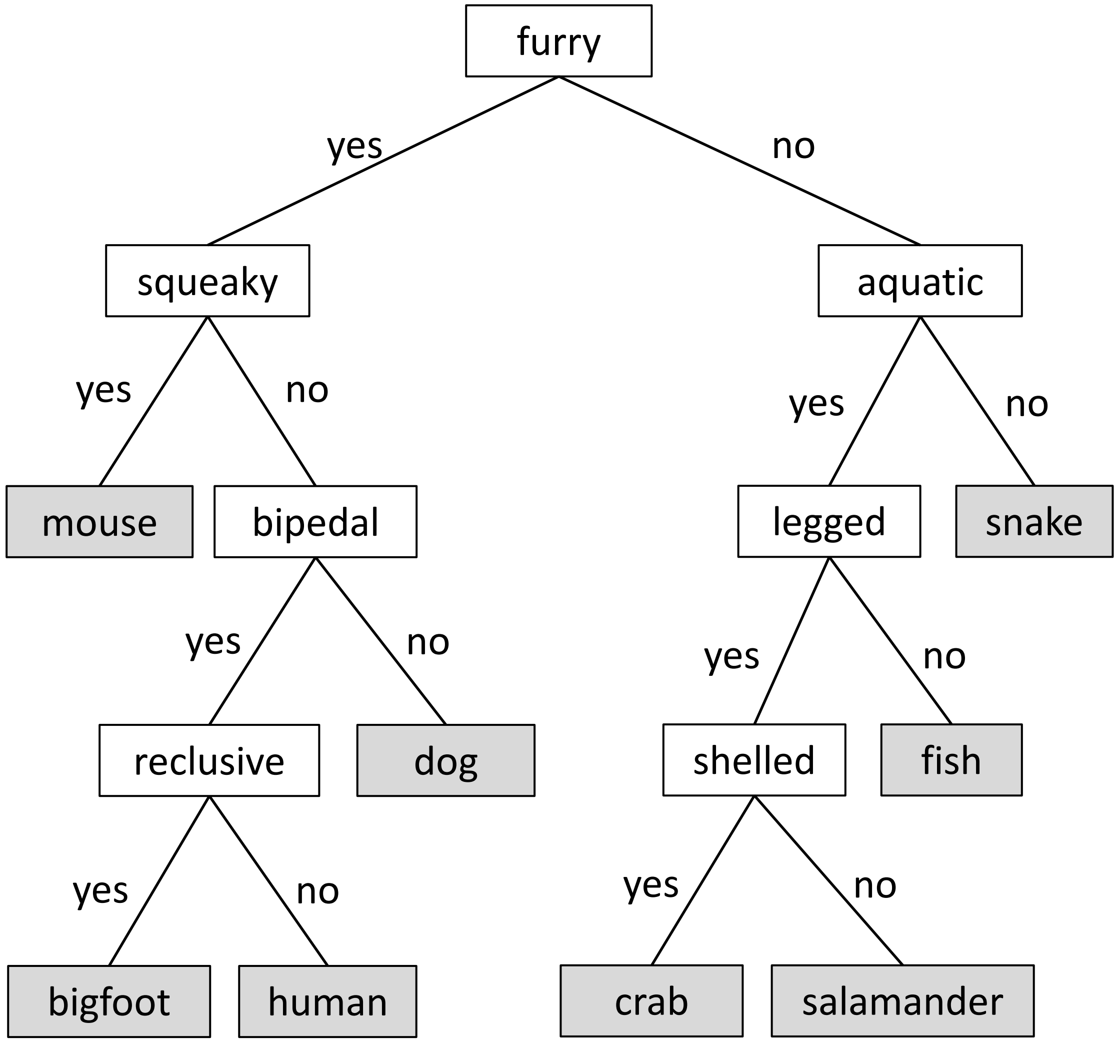
```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        depthFirst(n.getRight());
        System.out.println(n.getValue());
    }
}
```

Postorder

```
public void depthFirst(Node n) {
    if (n != null) {
        depthFirst(n.getLeft());
        System.out.println(n.getValue());
        depthFirst(n.getRight());
    }
}
```

Inorder

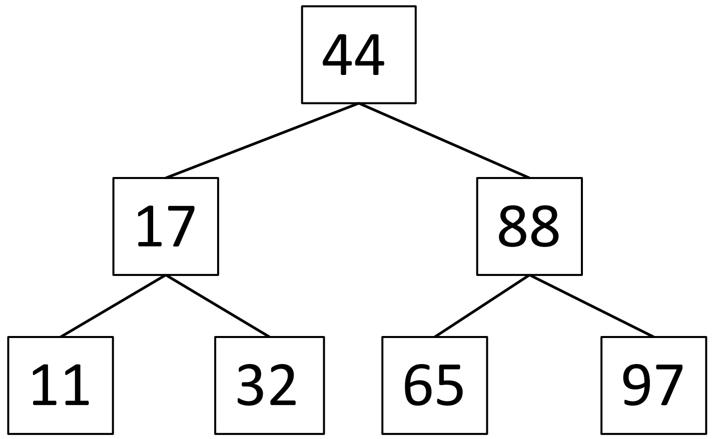




```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;


    ...
}
```

File read/writing

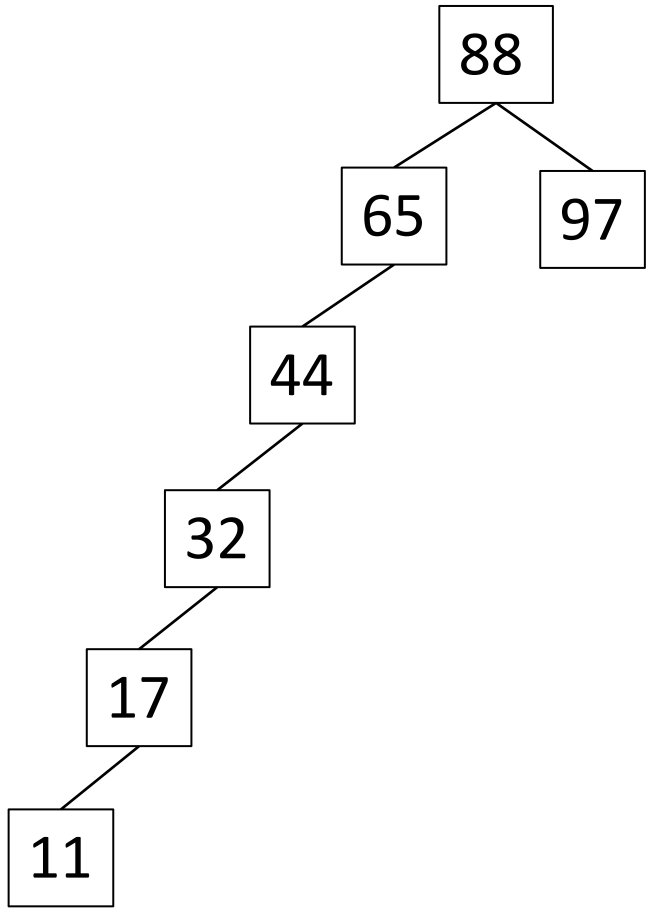
Order Matters



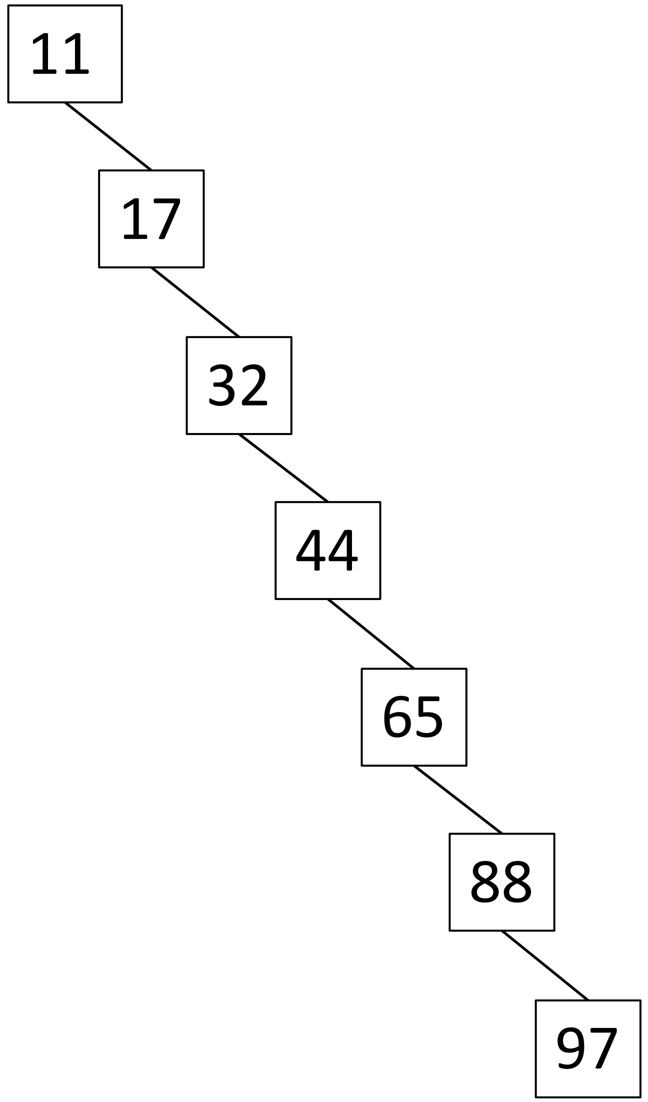
44, 17, 88, 11, 32, 65, 97

44, 17, 32, 88, 11, 97, 65

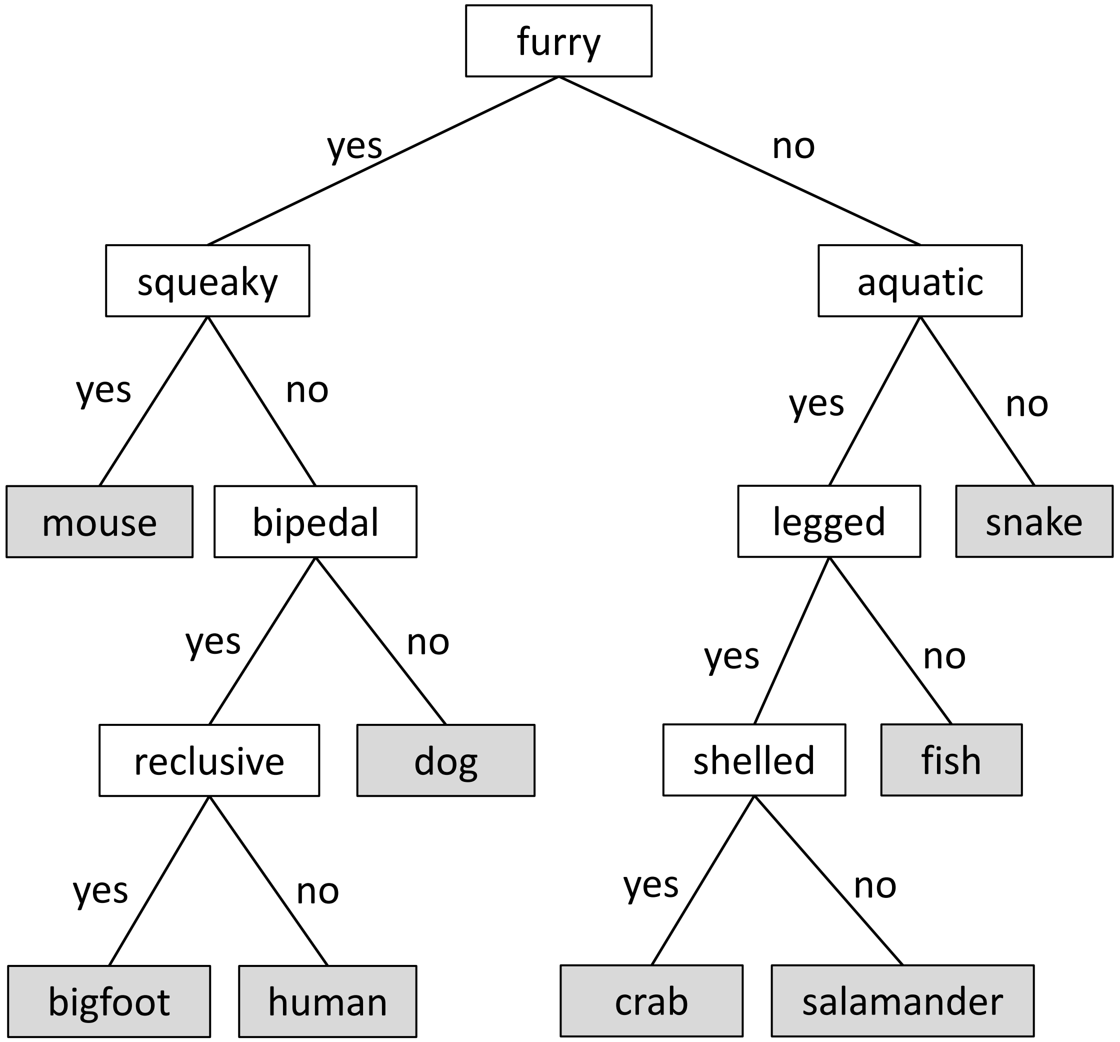
44, 88, 65, 97, 17, 32, 11



88, 65, 44, 32, 97, 17, 11

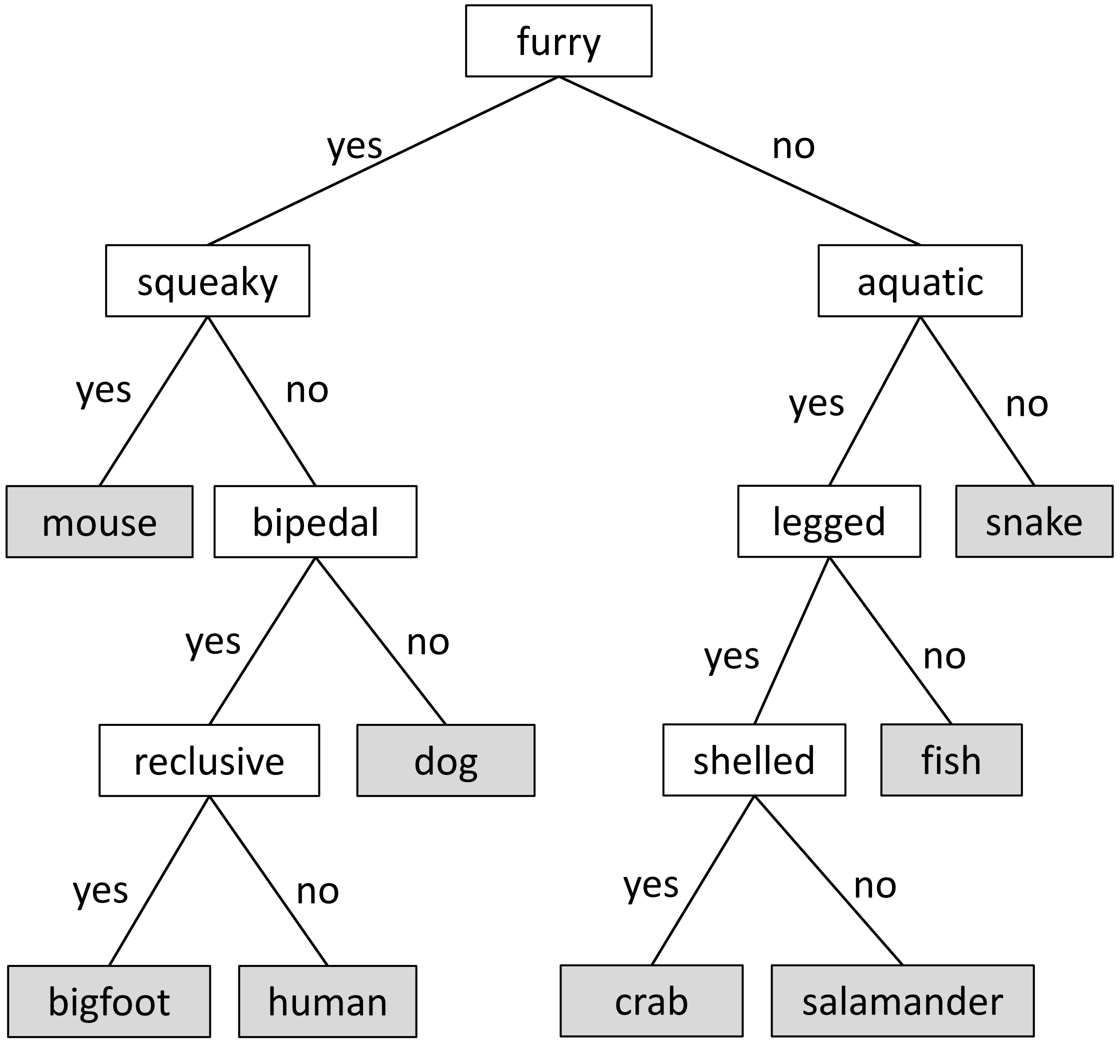


11, 17, 32, 44, 65, 88, 97



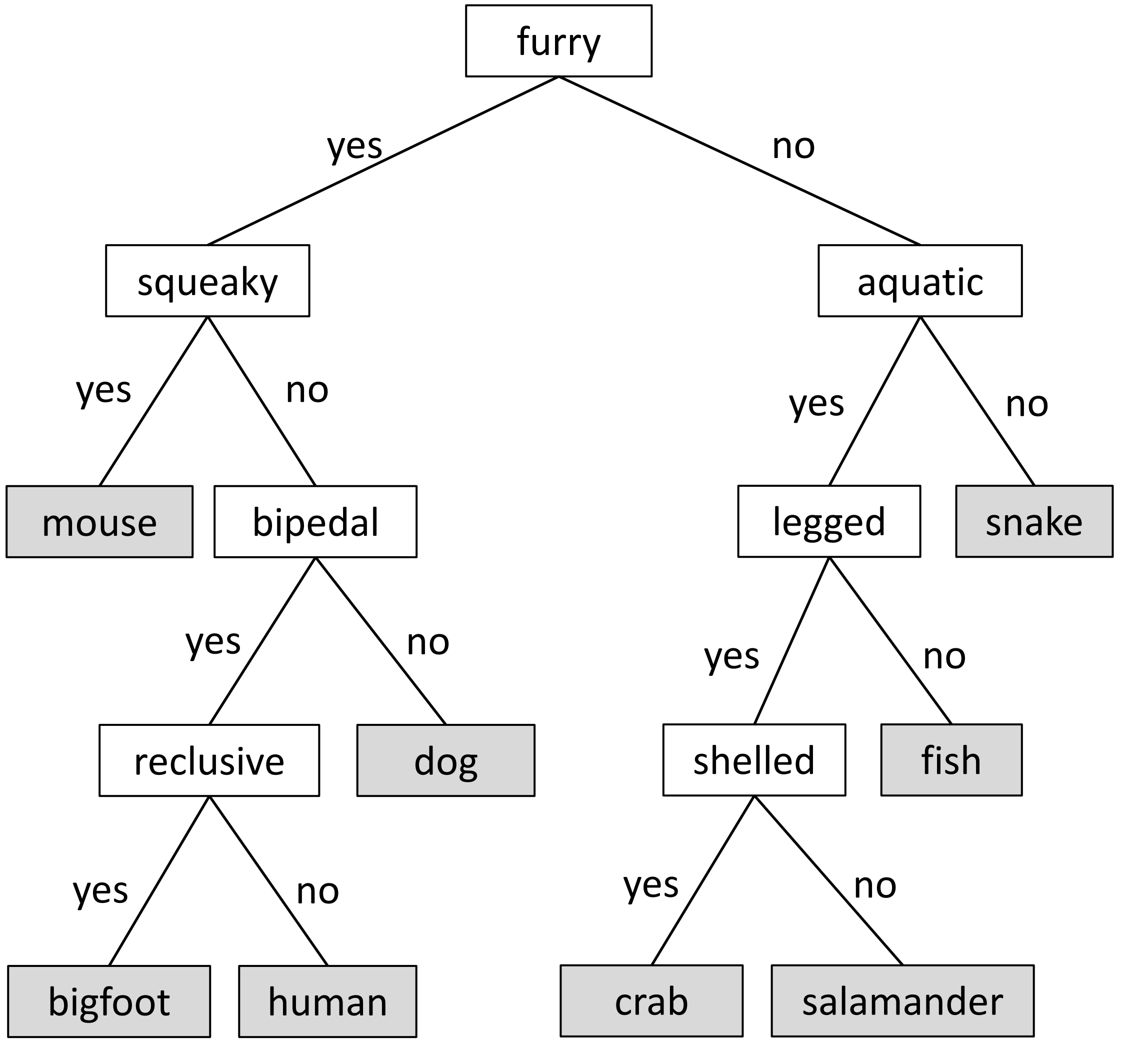
```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;


    ...
}
```

File read/writing



```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

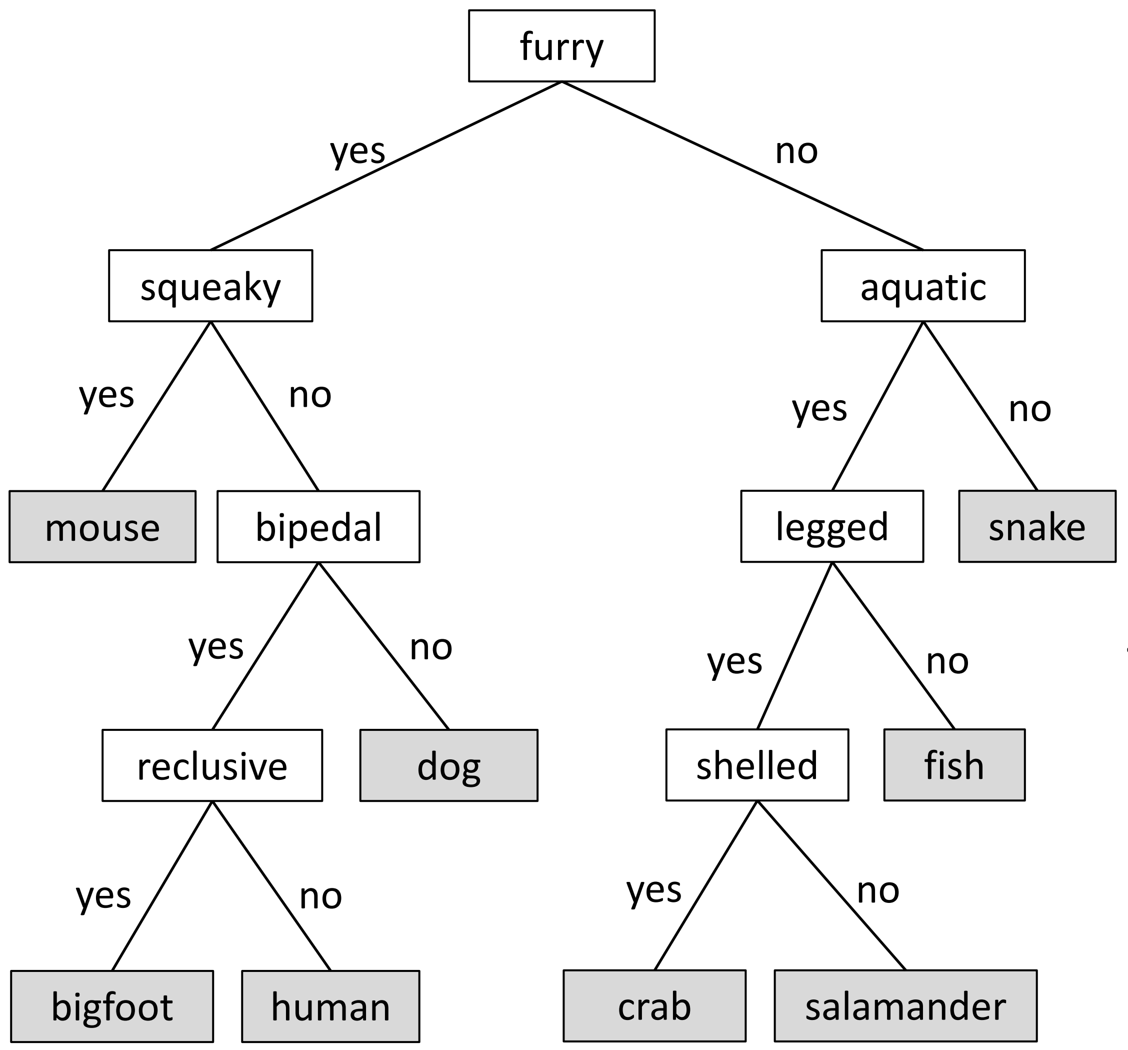
File read/writing



```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

File read/writing

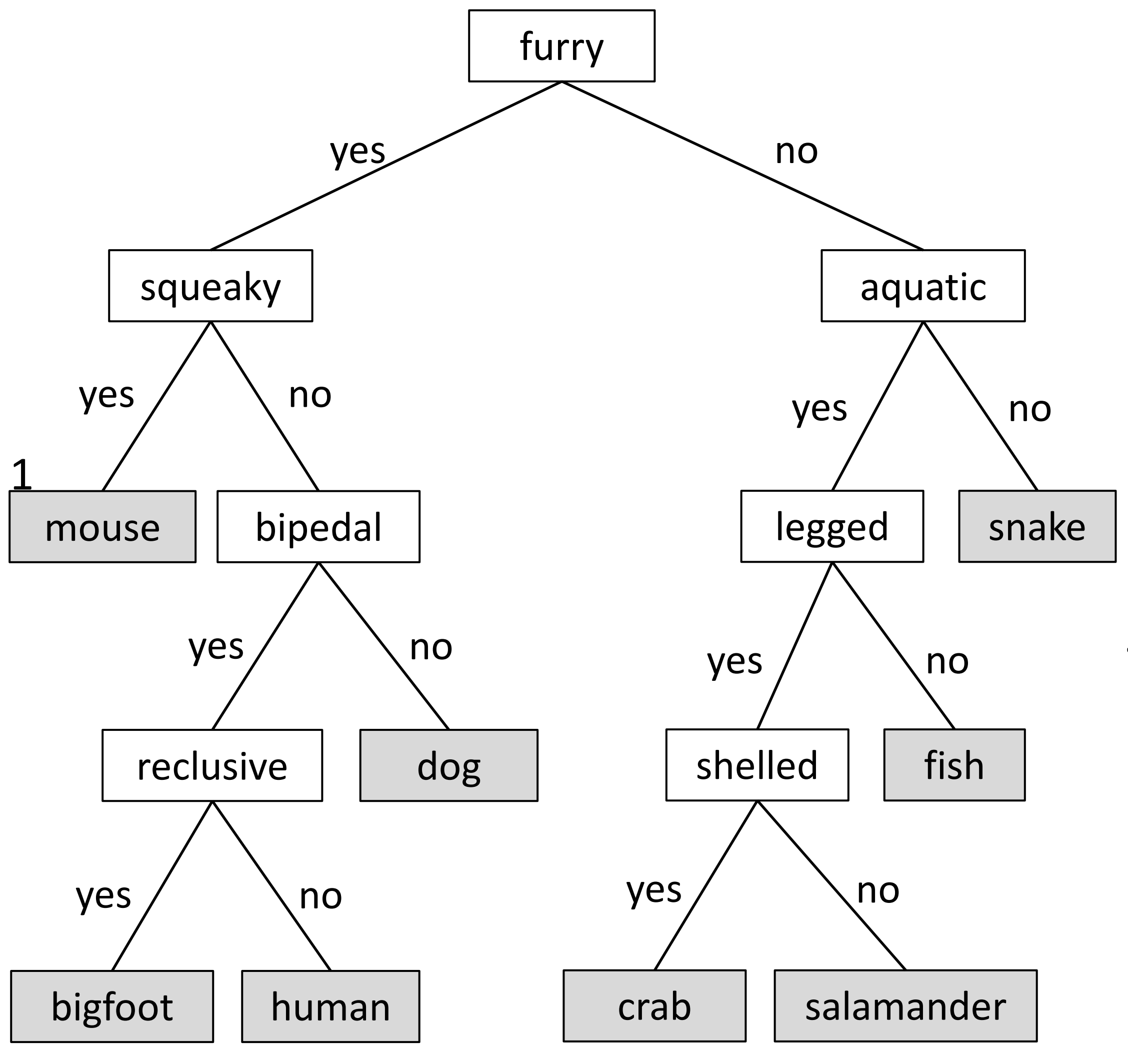


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

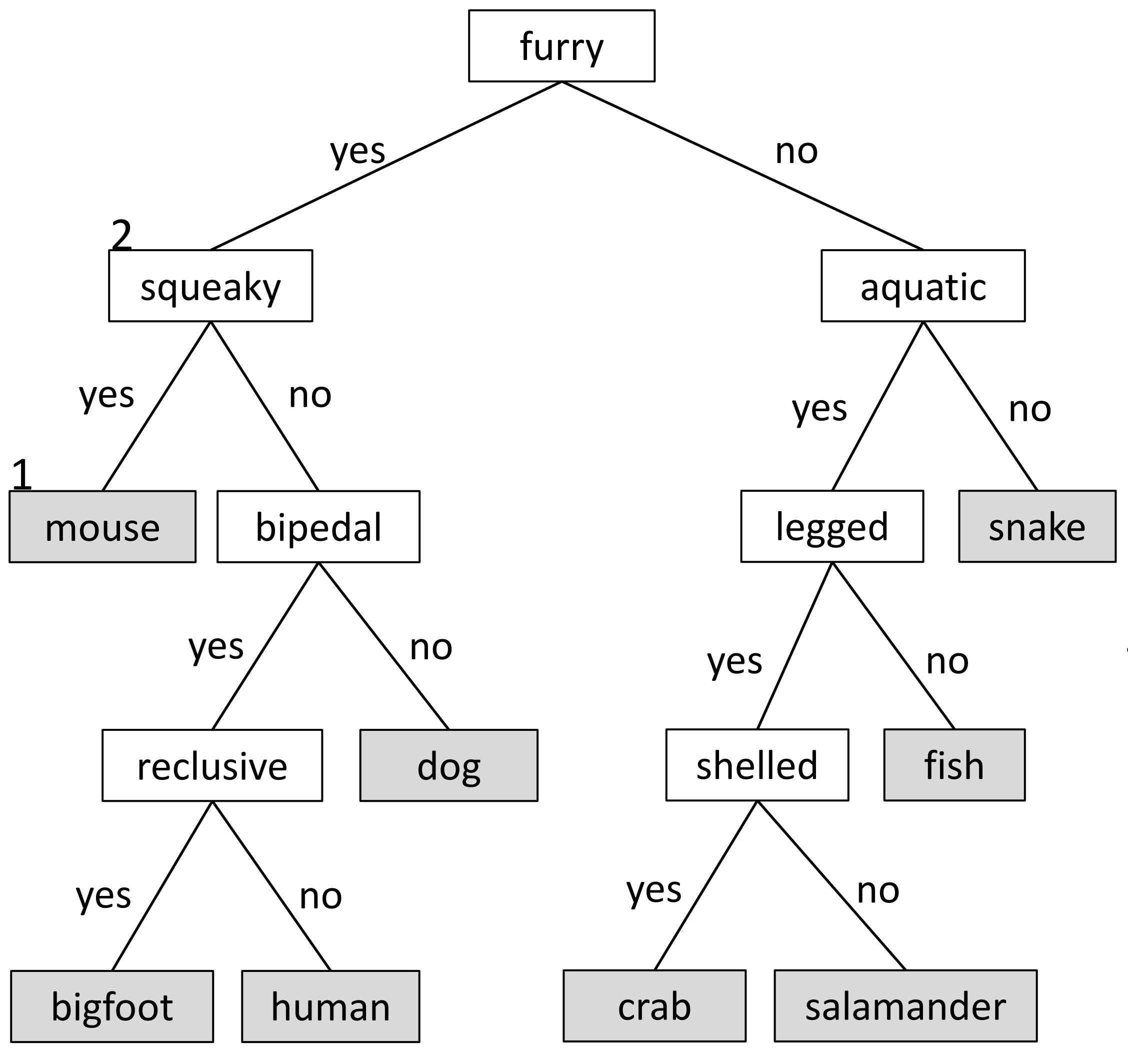


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

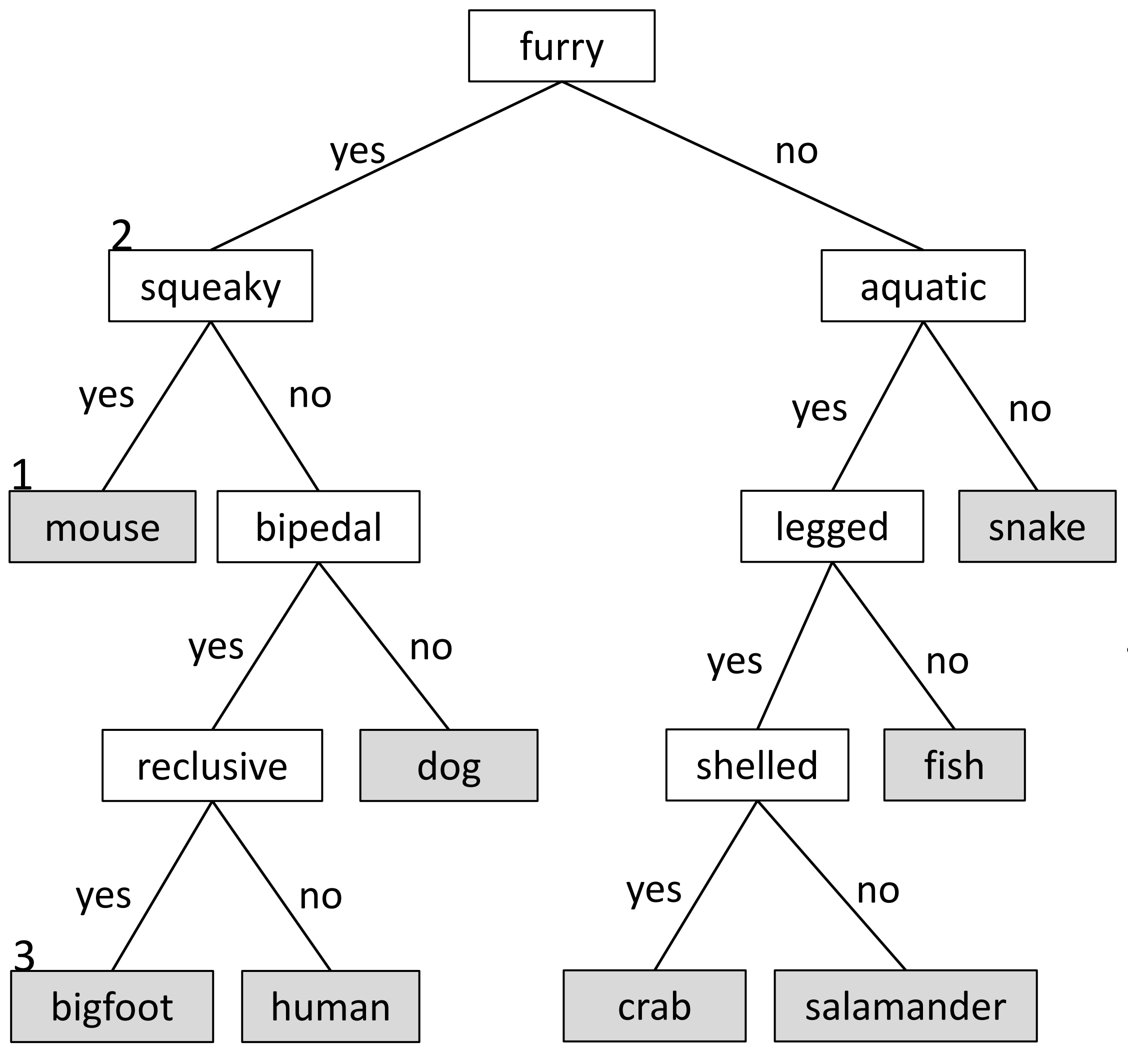


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

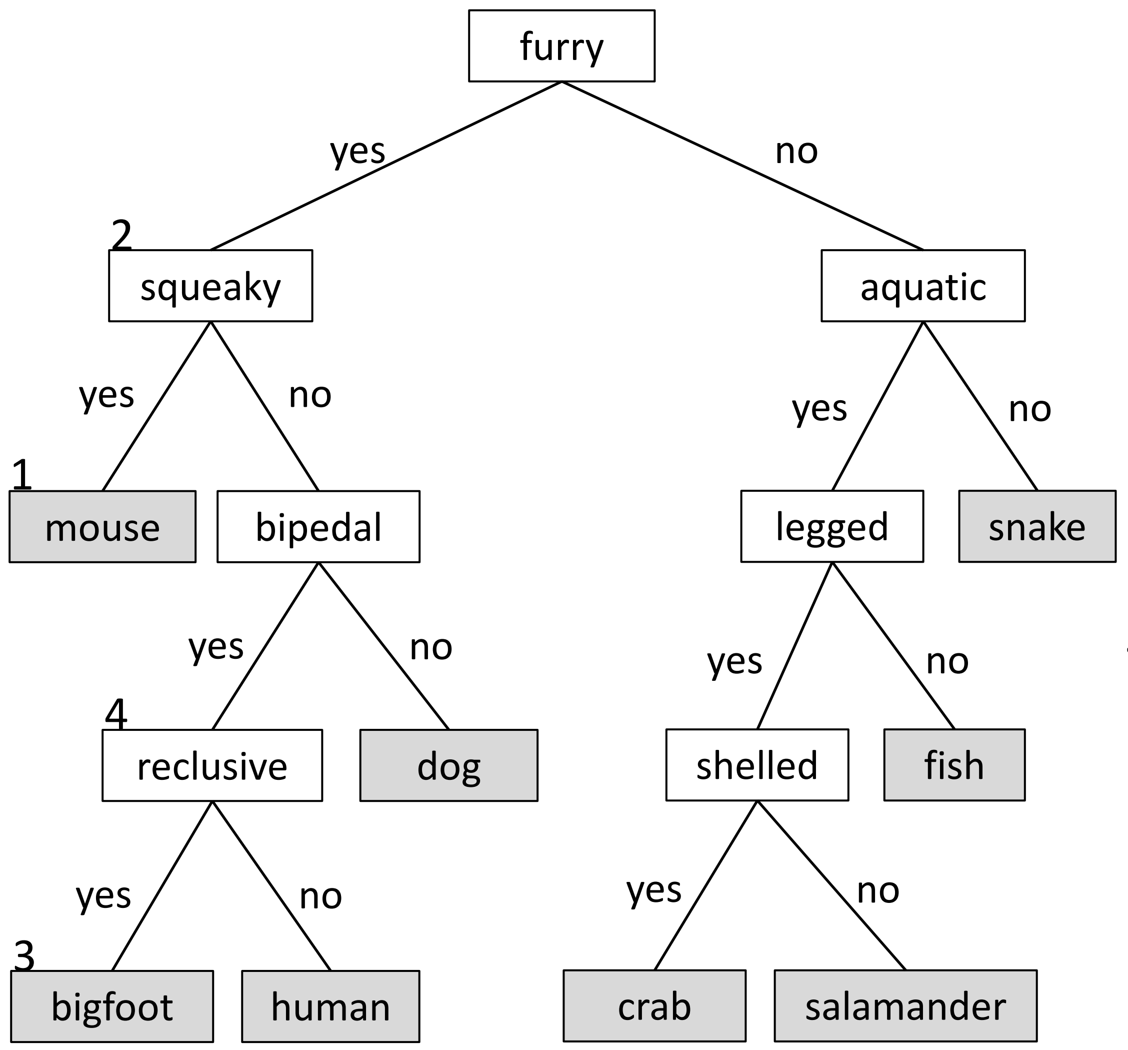


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

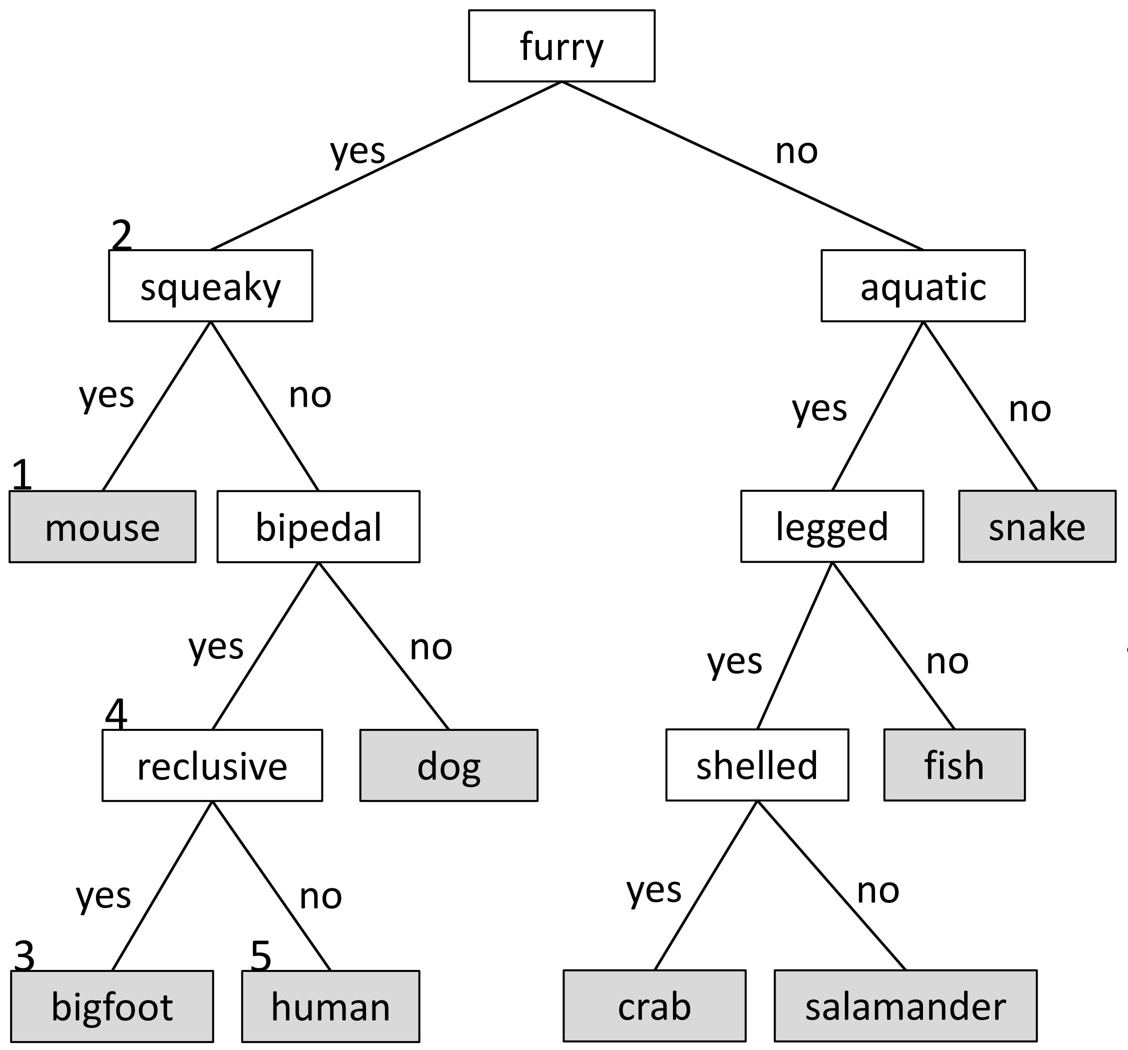


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

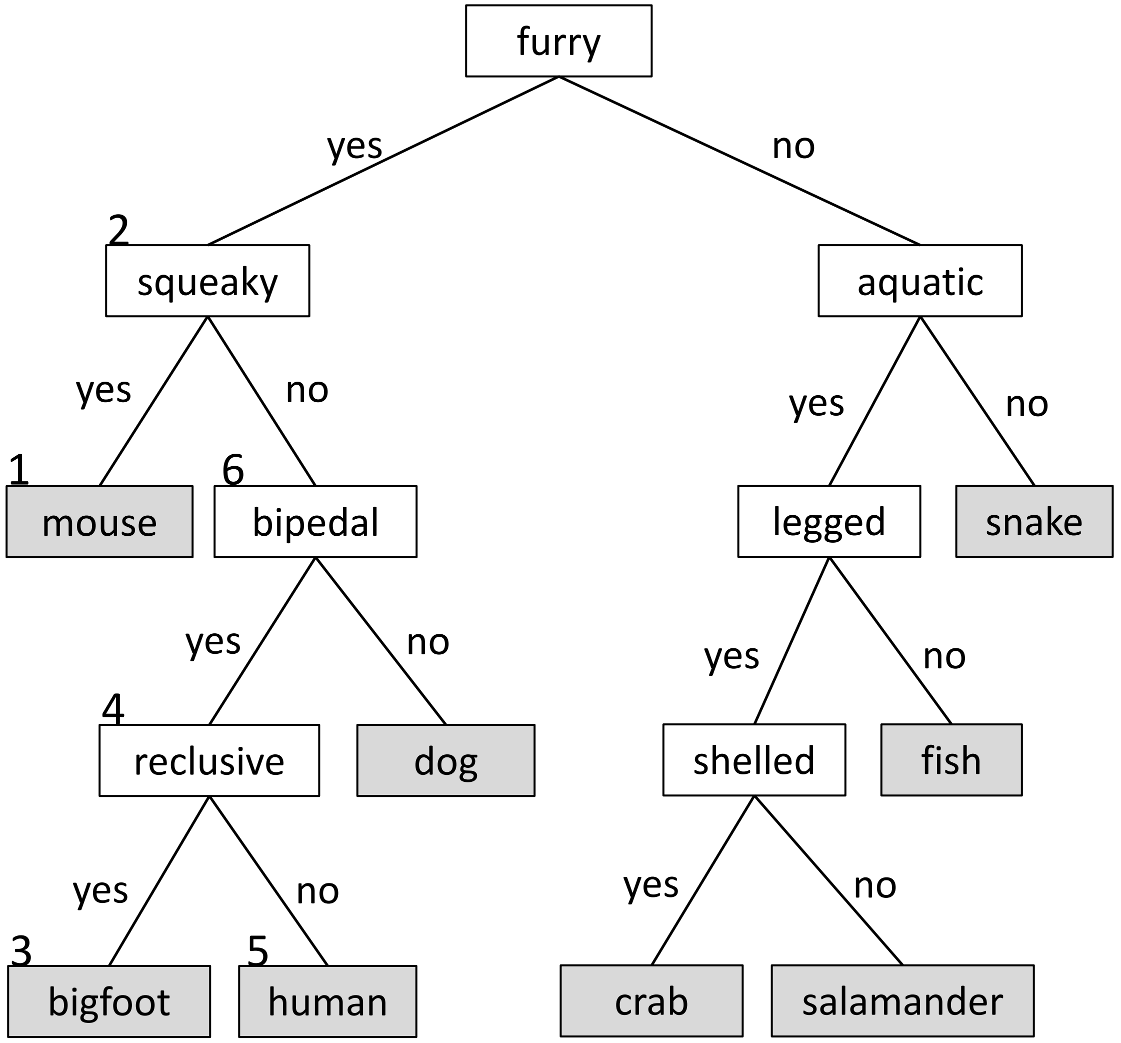


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

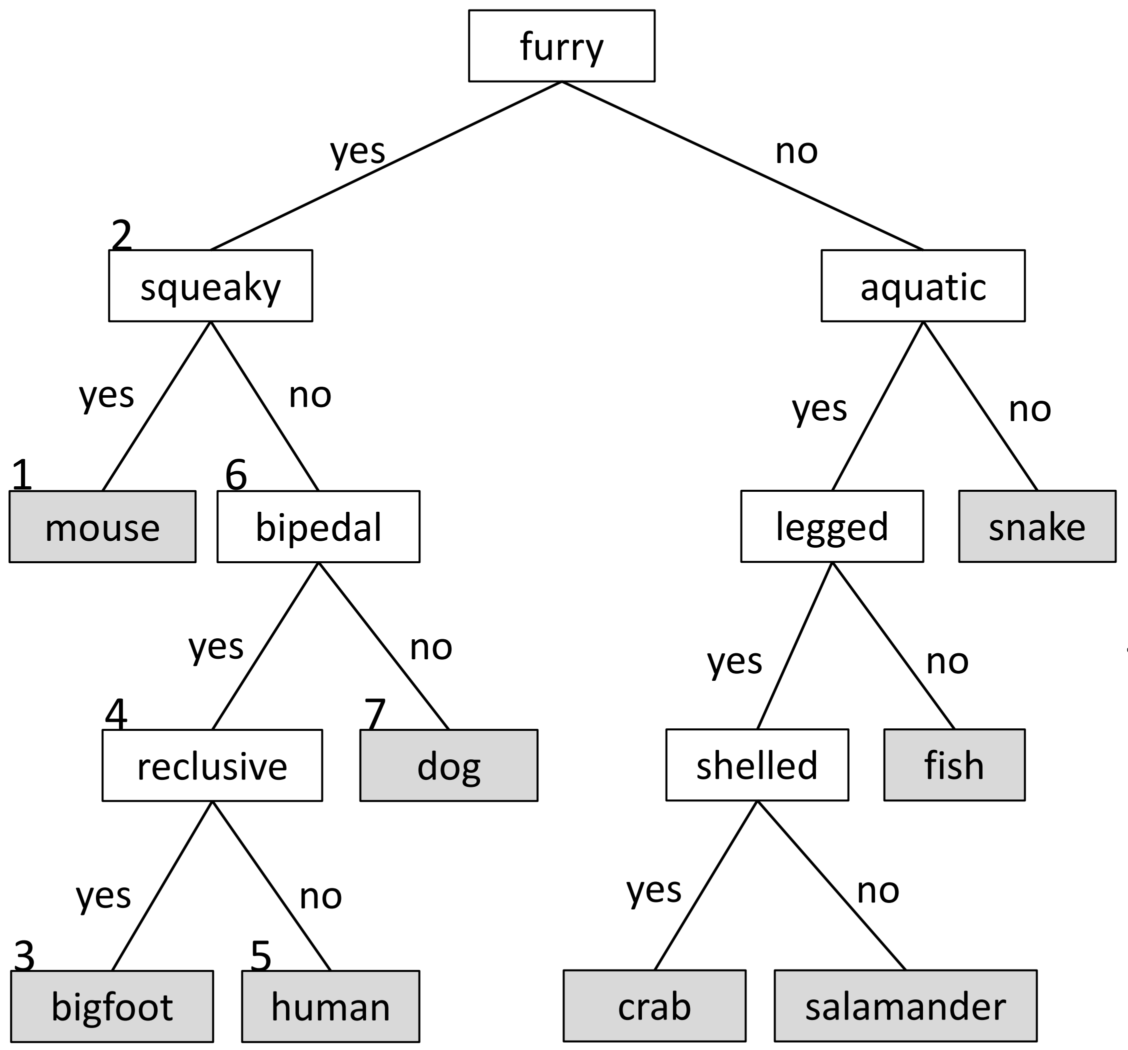


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

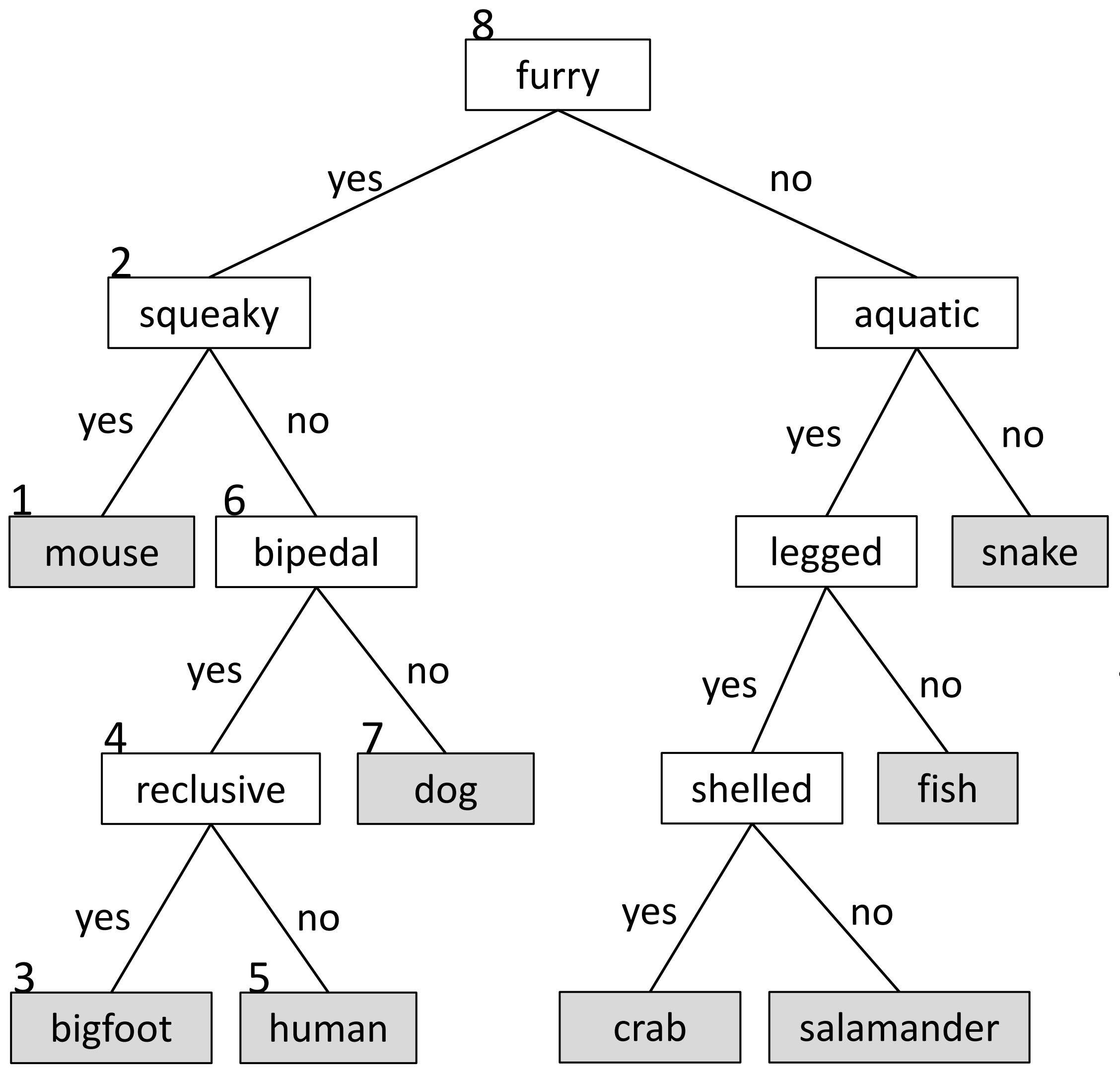


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

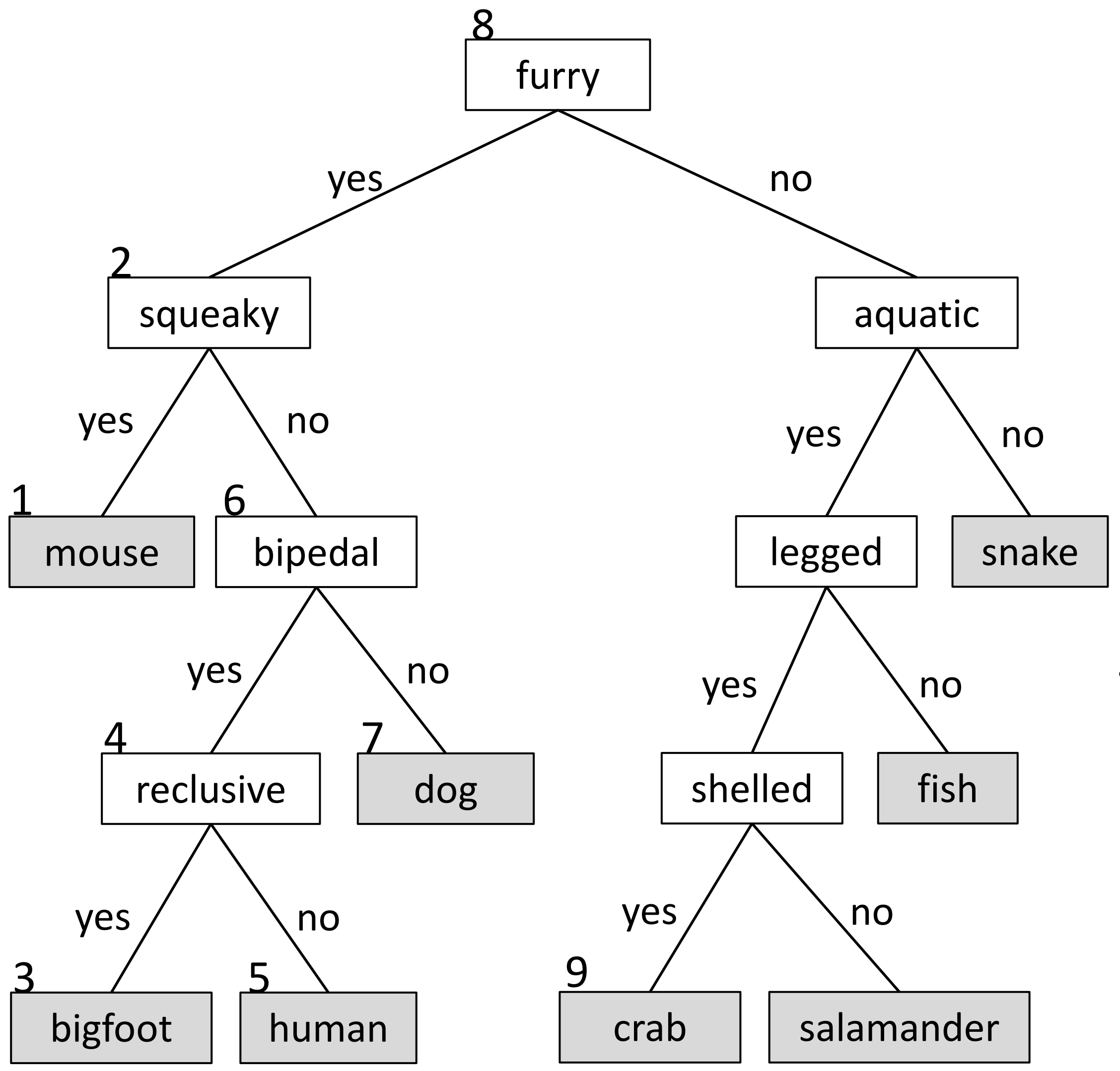


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

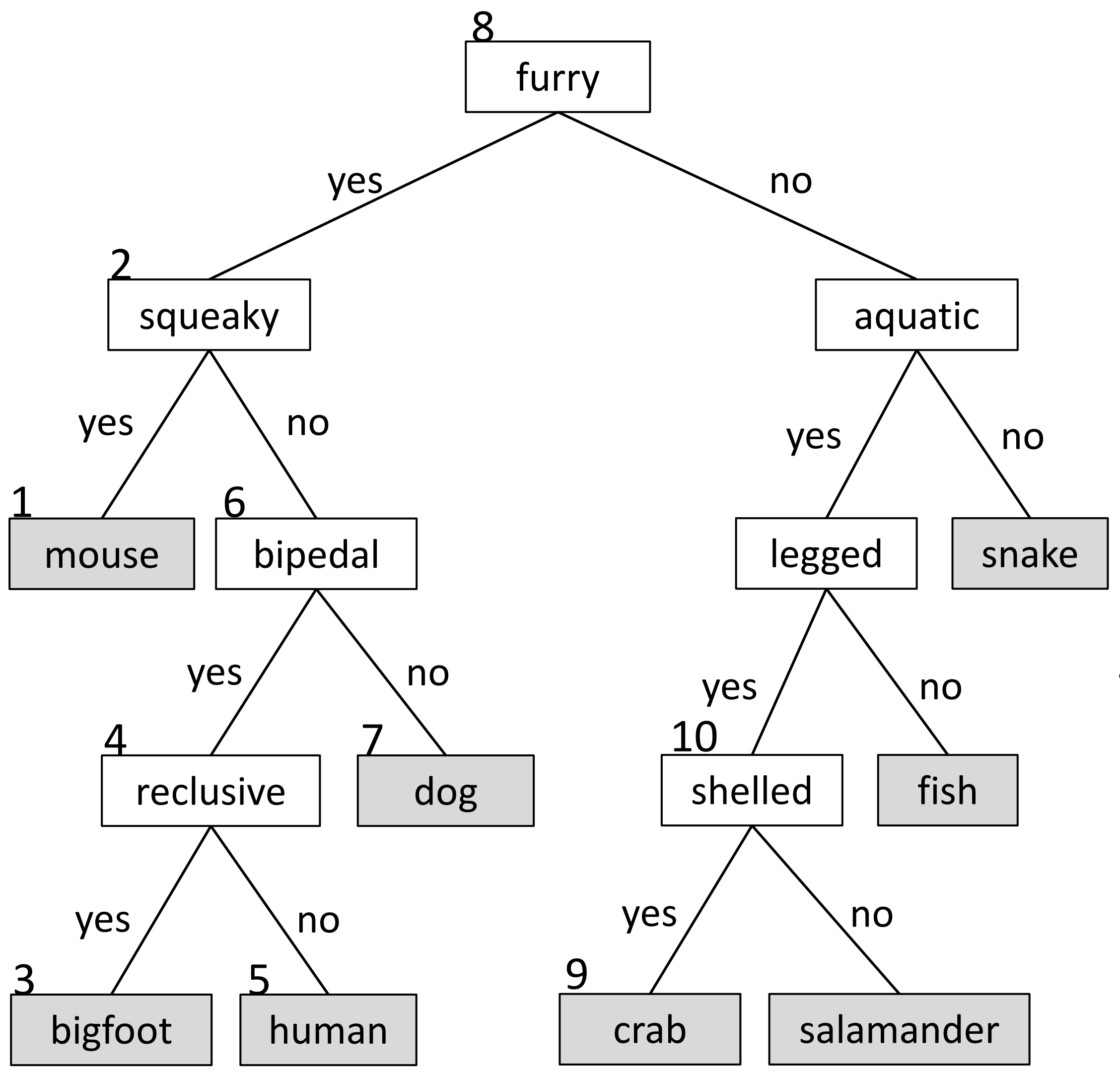


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

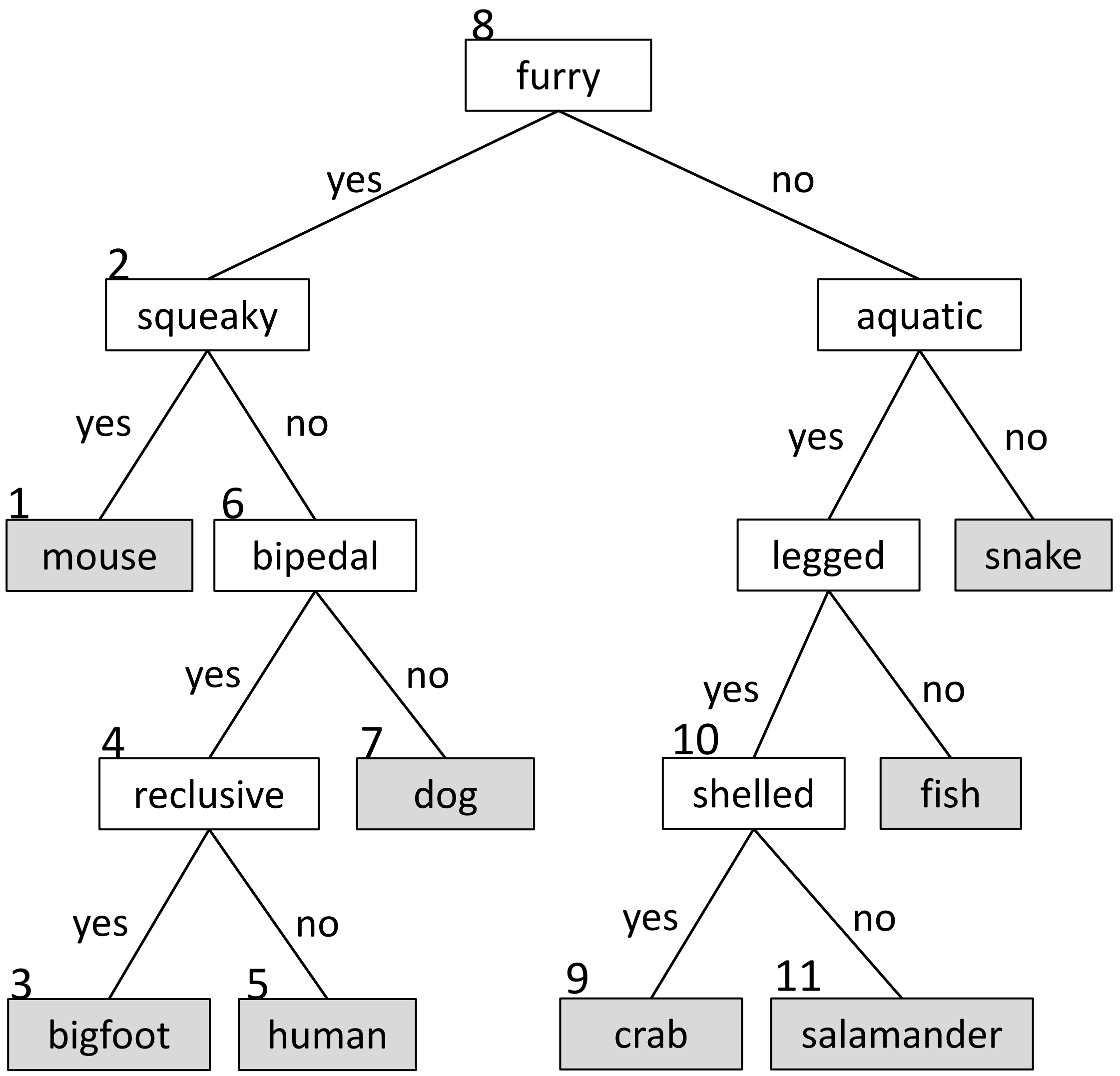


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

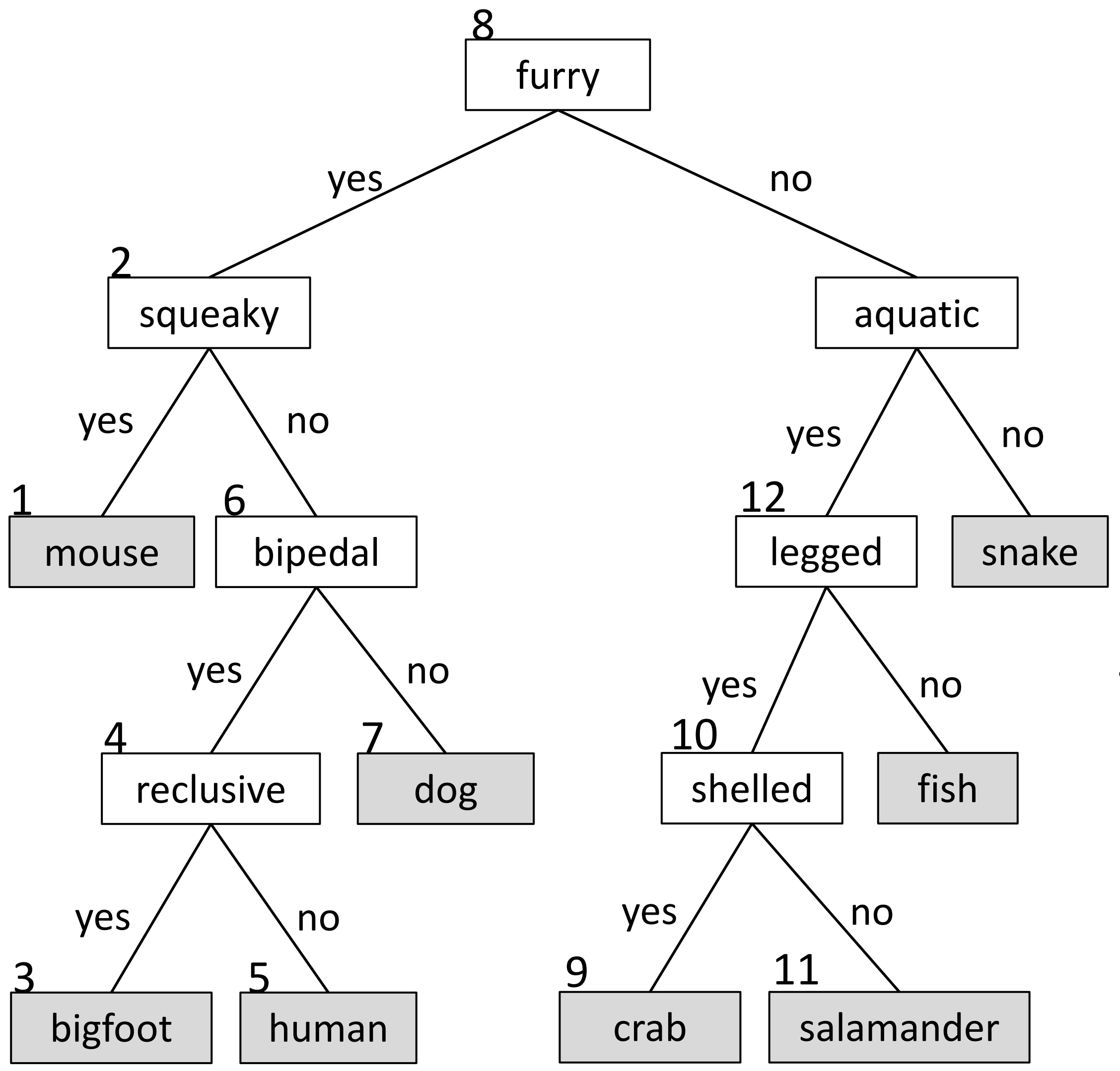


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

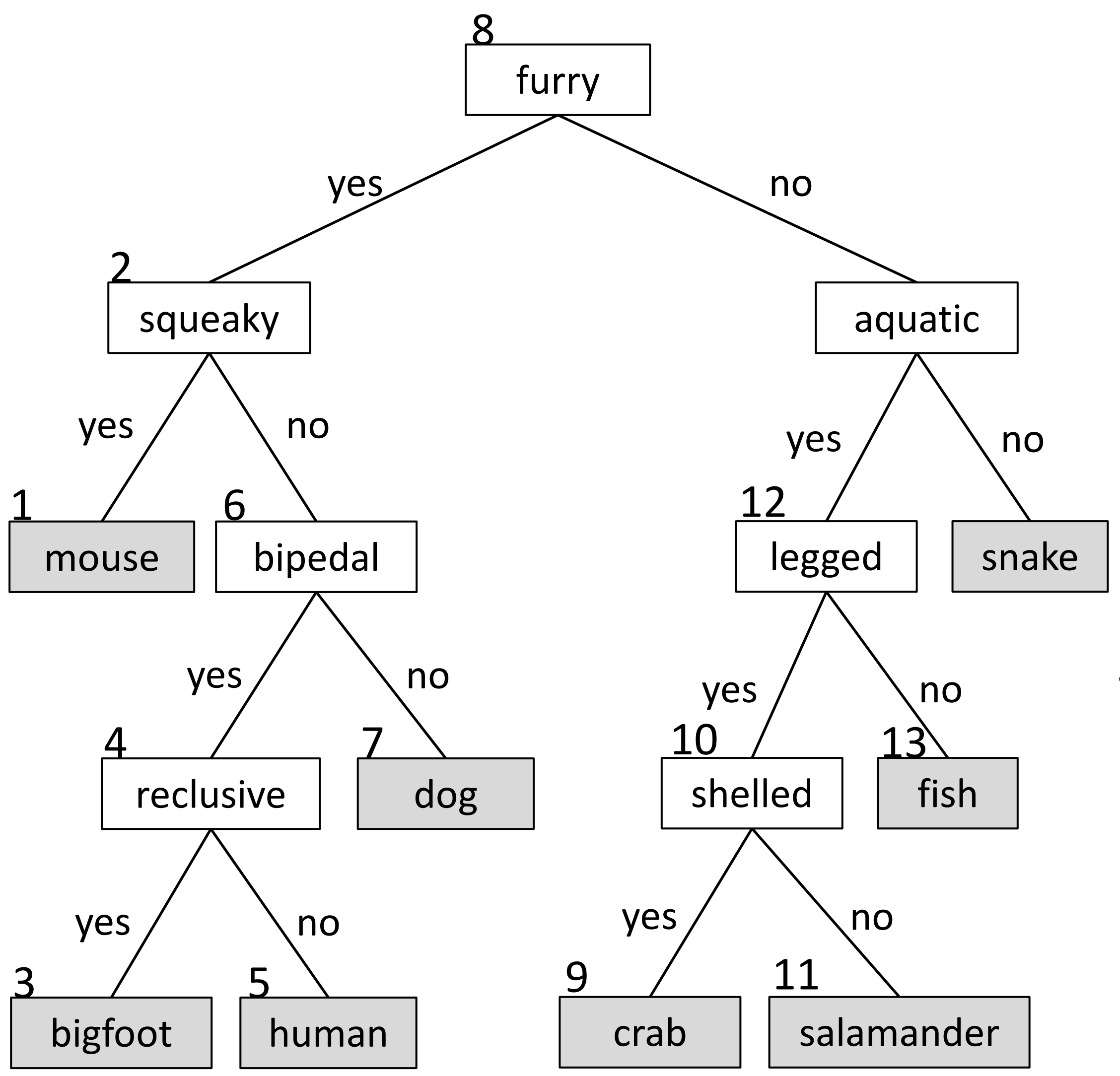


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

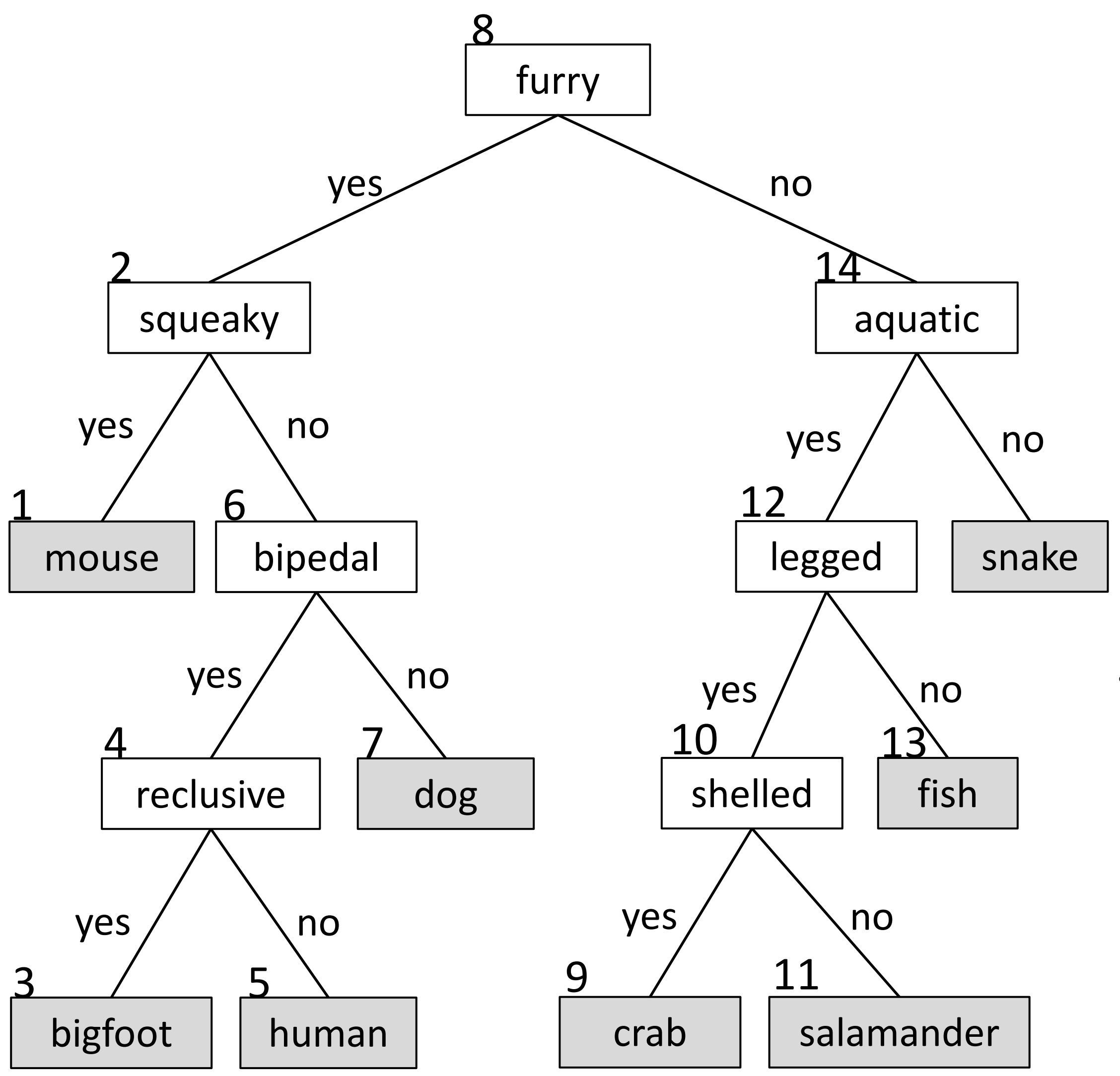


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

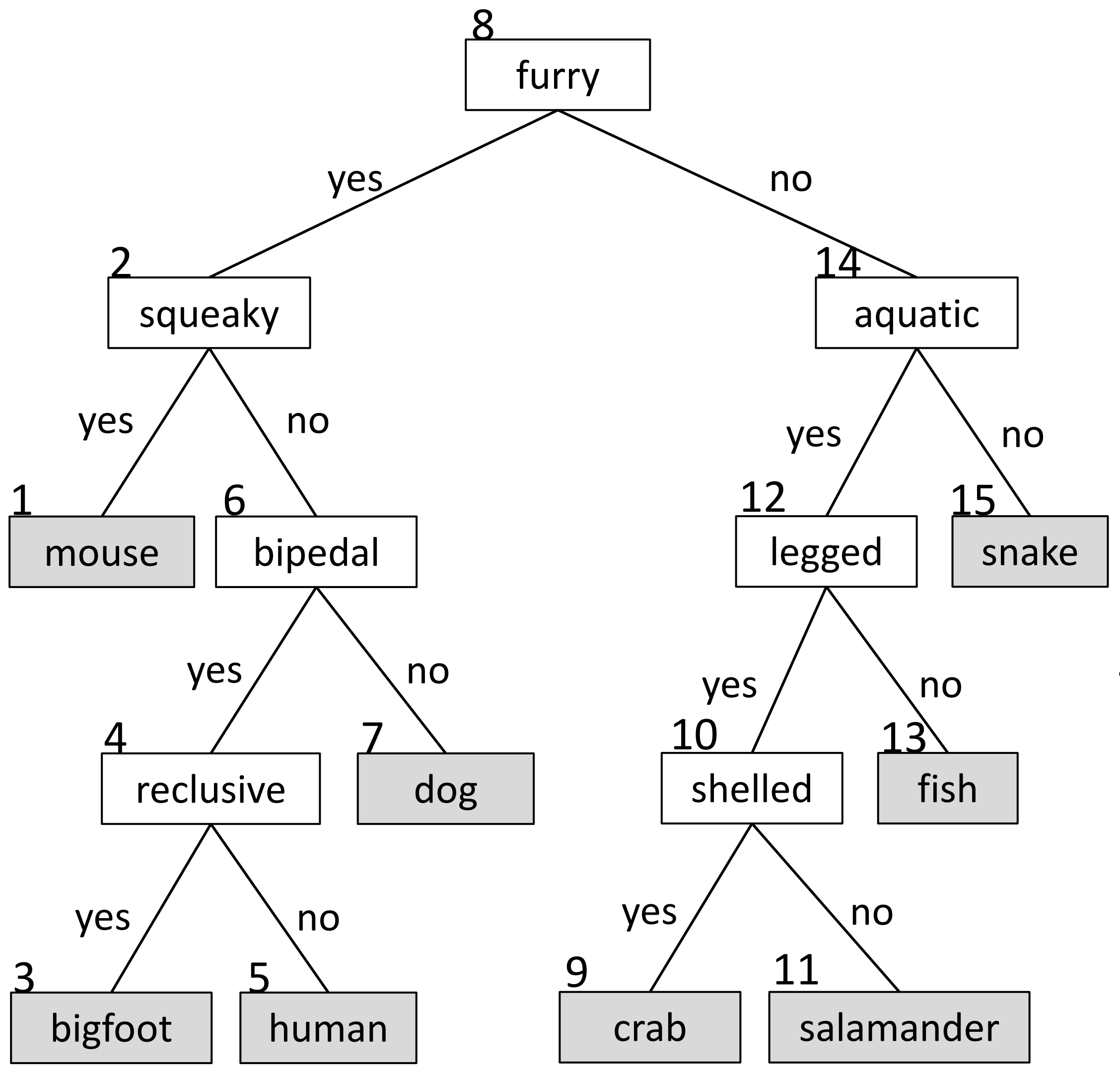


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

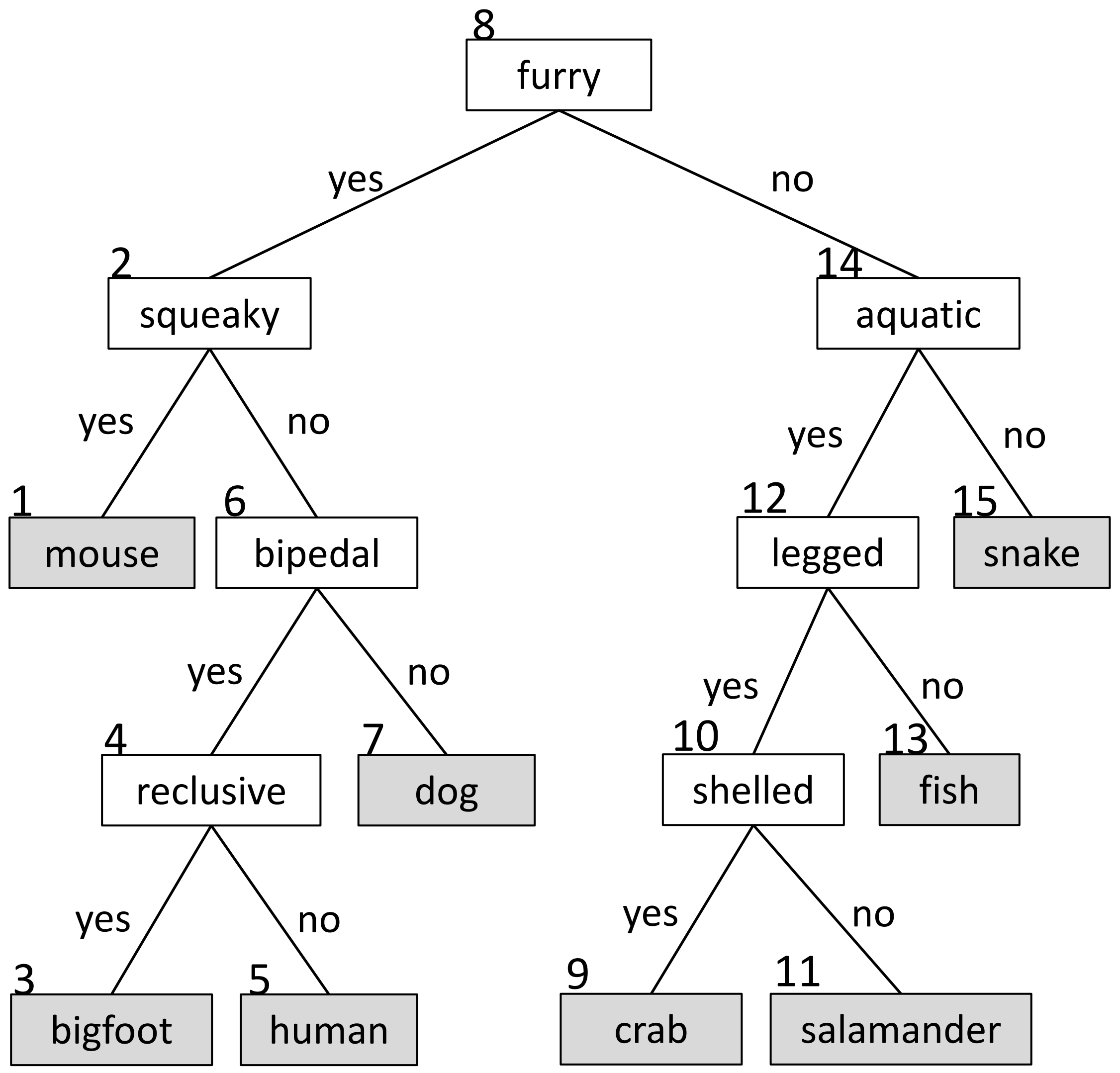


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.

File read/writing

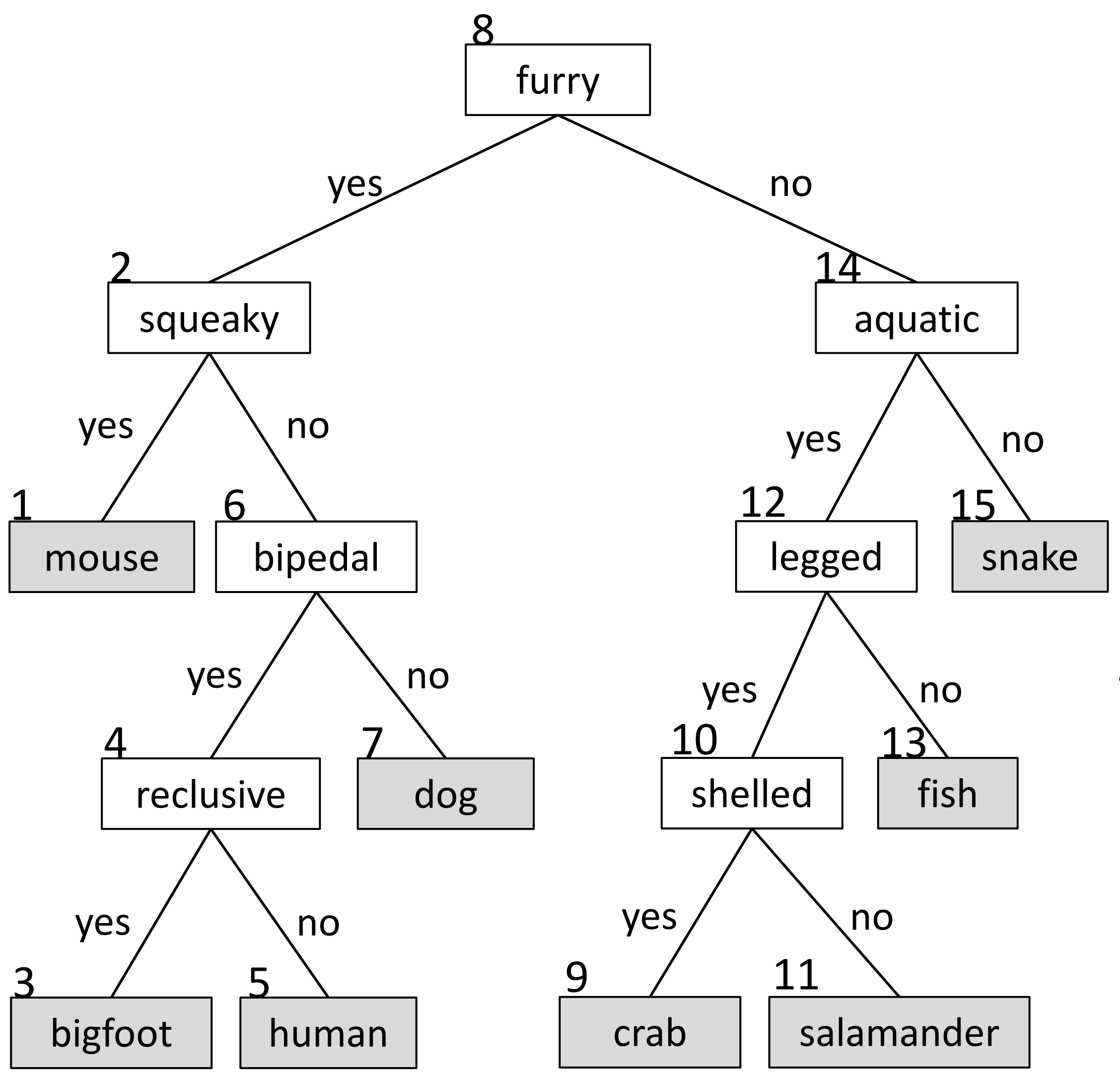


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.
2. Do breadth first traversal and write tag and text values to file. E.g. 8-furry,2-squeaky,14-aquatic,1-mouse,6-bipedal,…

File read/writing



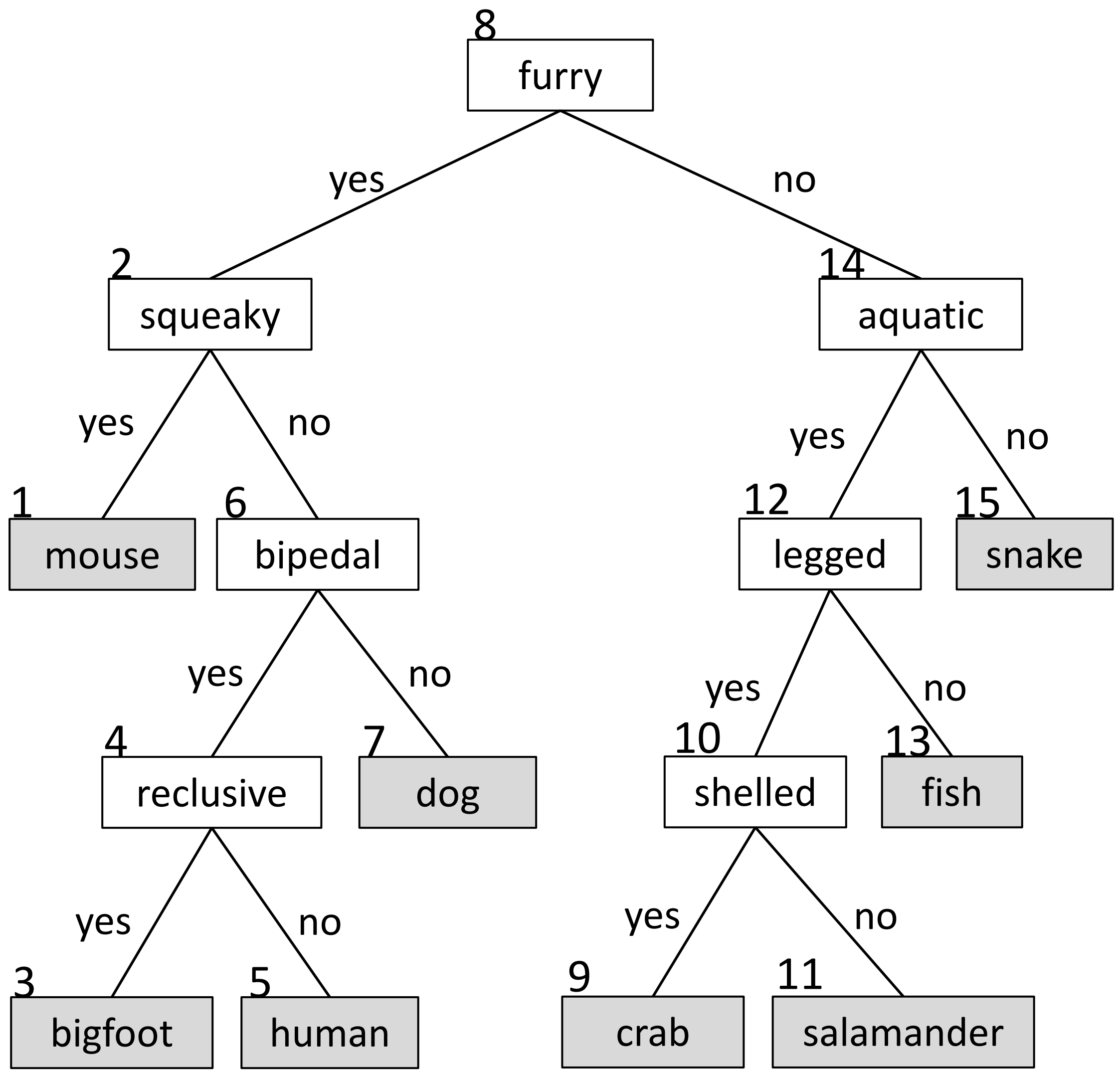
```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.
2. Do breadth first traversal and write tag and text values to file. E.g. 8-furry,2-squeaky,14-aquatic,1-mouse,6-bipedal,…

Build from file:

File read/writing



```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

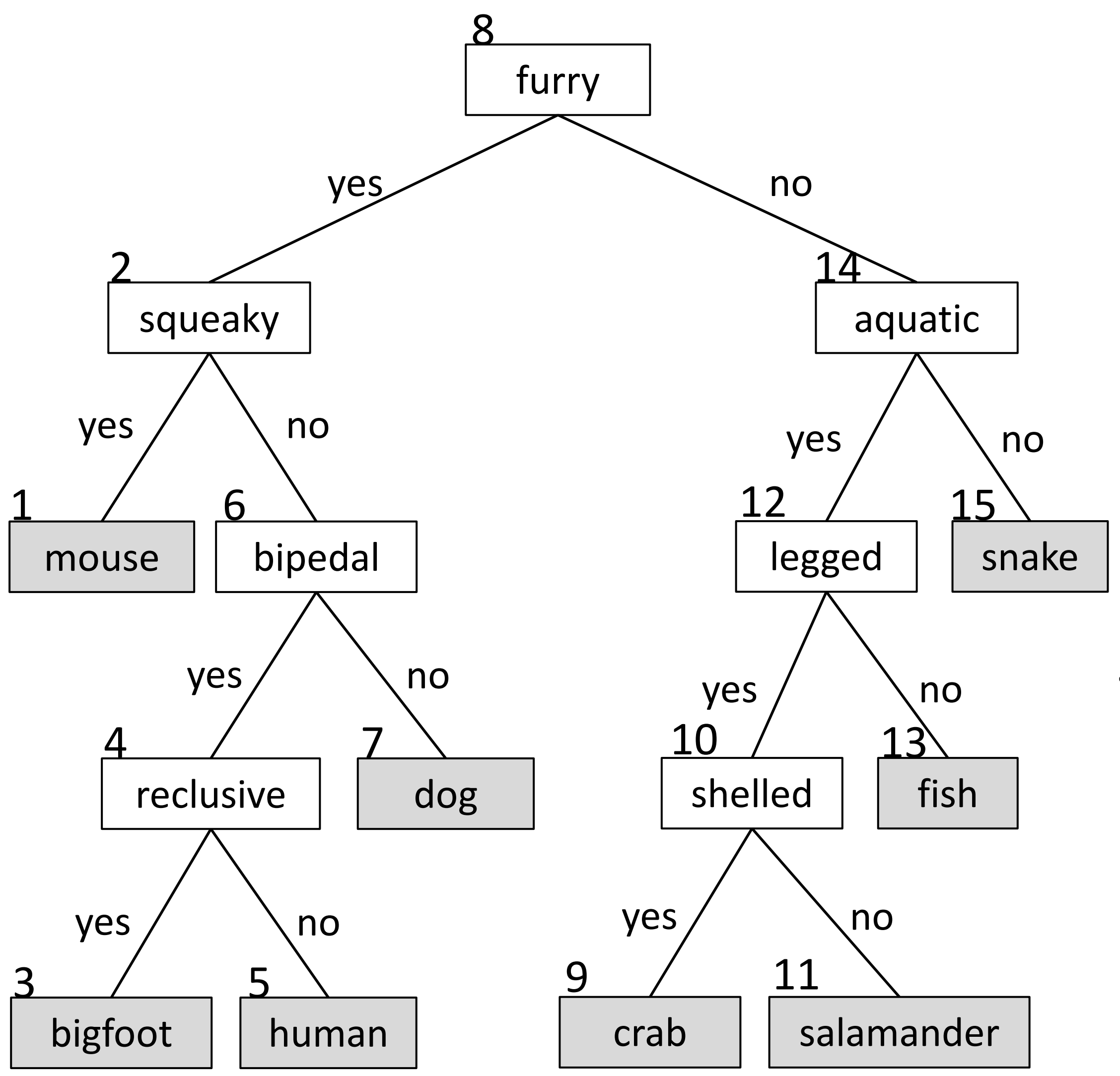
Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.
2. Do breadth first traversal and write tag and text values to file. E.g. 8-furry,2-squeaky,14-aquatic,1-mouse,6-bipedal,…

Build from file:

1. Parse input on commas to get each entry.
2. Parse each entry on dash to get tag value and text value.

File read/writing



File read/writing

```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

1. Do inorder traversal of tree and assign sequential integer tag values.
2. Do breadth first traversal and write tag and text values to file. E.g. 8-furry,2-squeaky,14-aquatic,1-mouse,6-bipedal,…

Build from file:

1. Parse input on commas to get each entry.
2. Parse each entry on dash to get tag value and text value.
3. Use BST insert method to put tag/text where it should be.