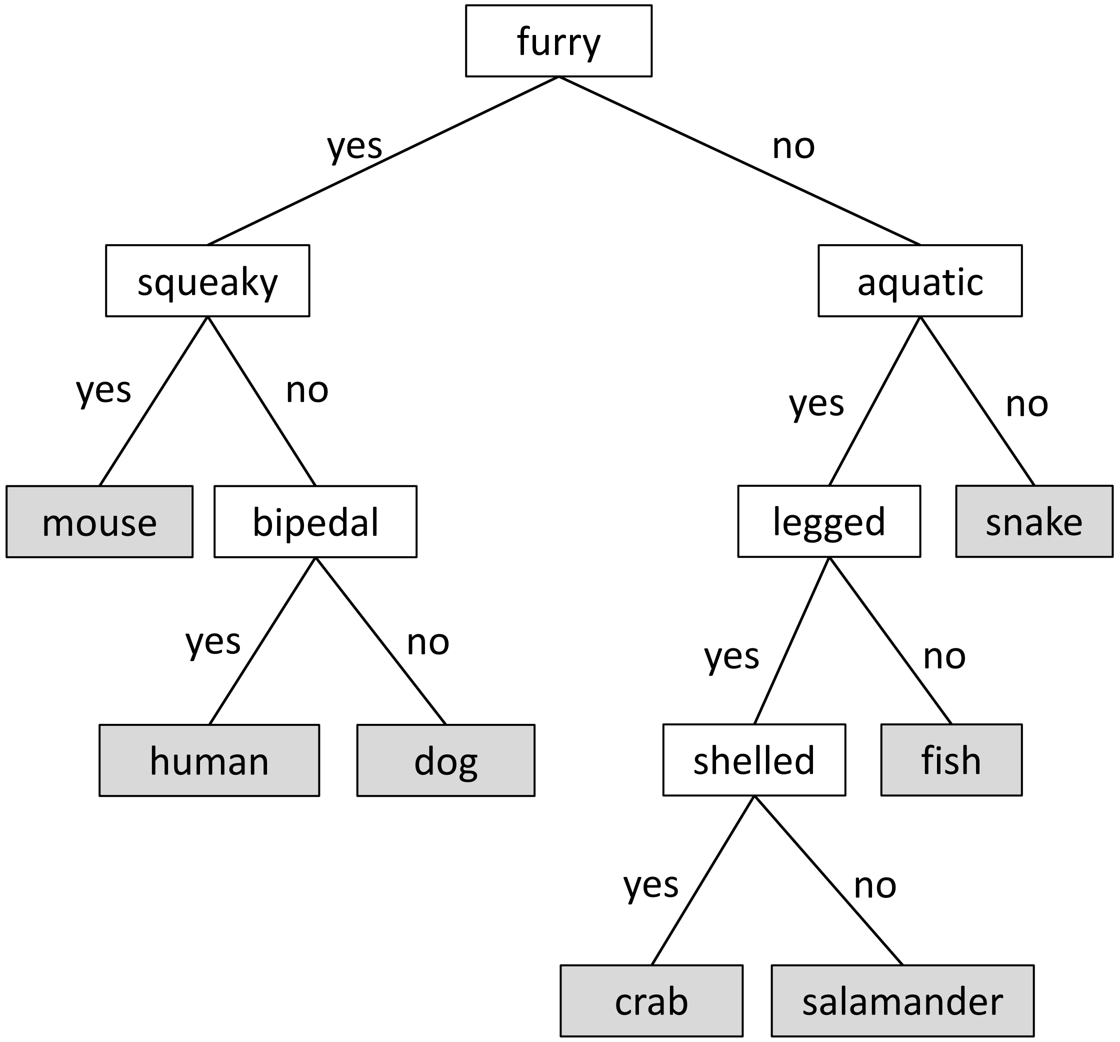
CSCI 232:
Data Structures and Algorithms

Program 1, More BST

Reese Pearsall

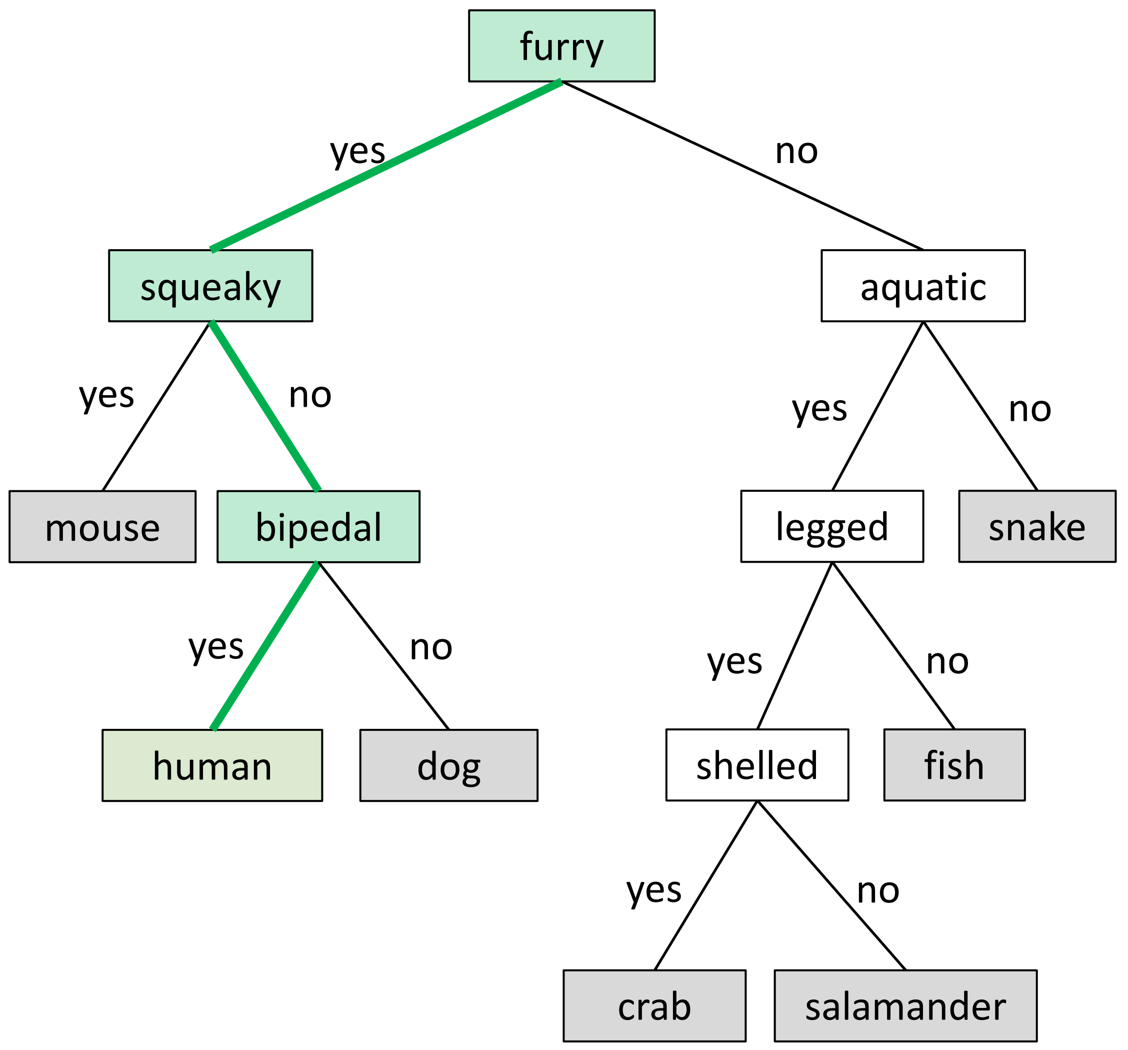
Spring 2025



```
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:

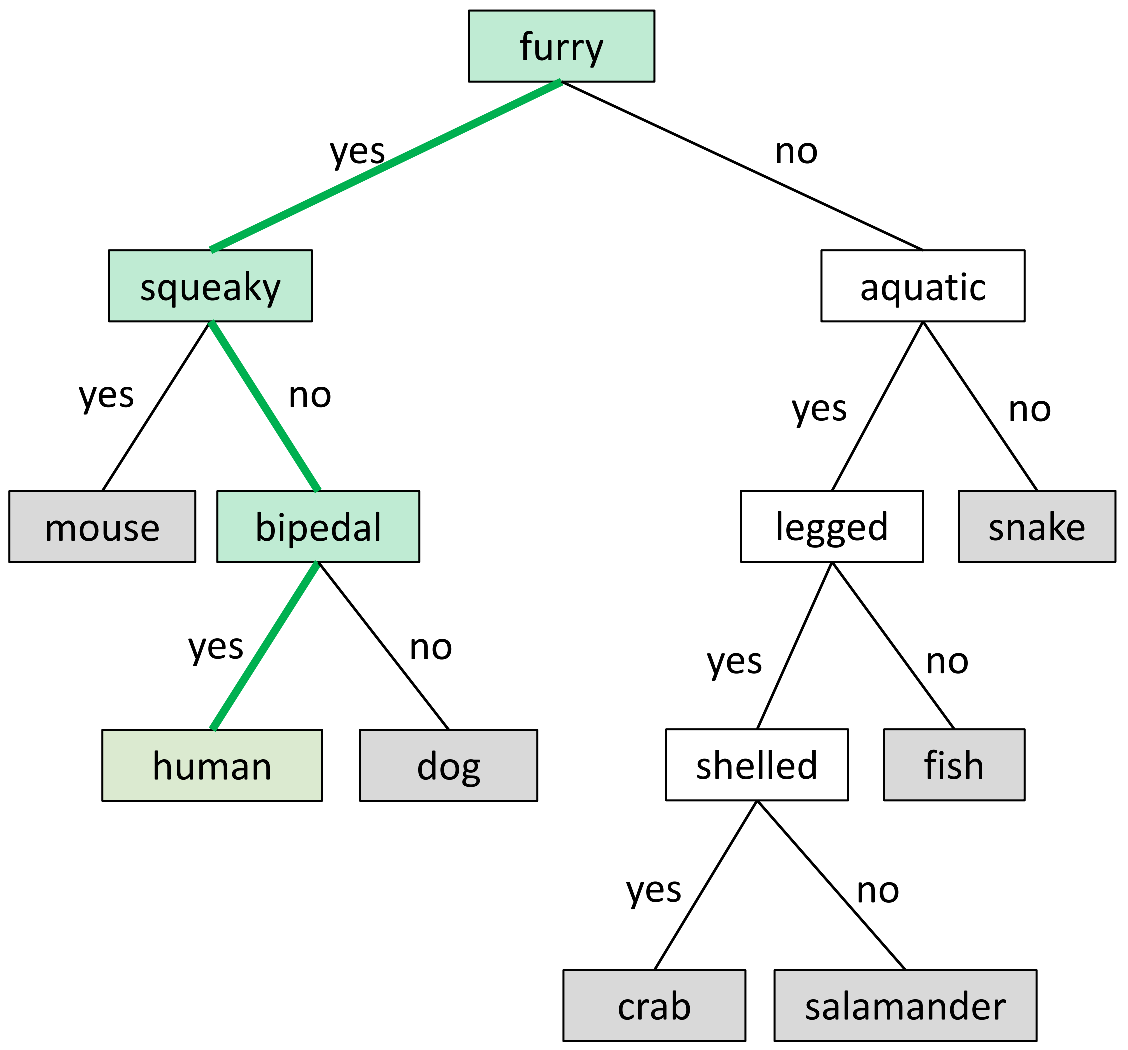
1.



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:

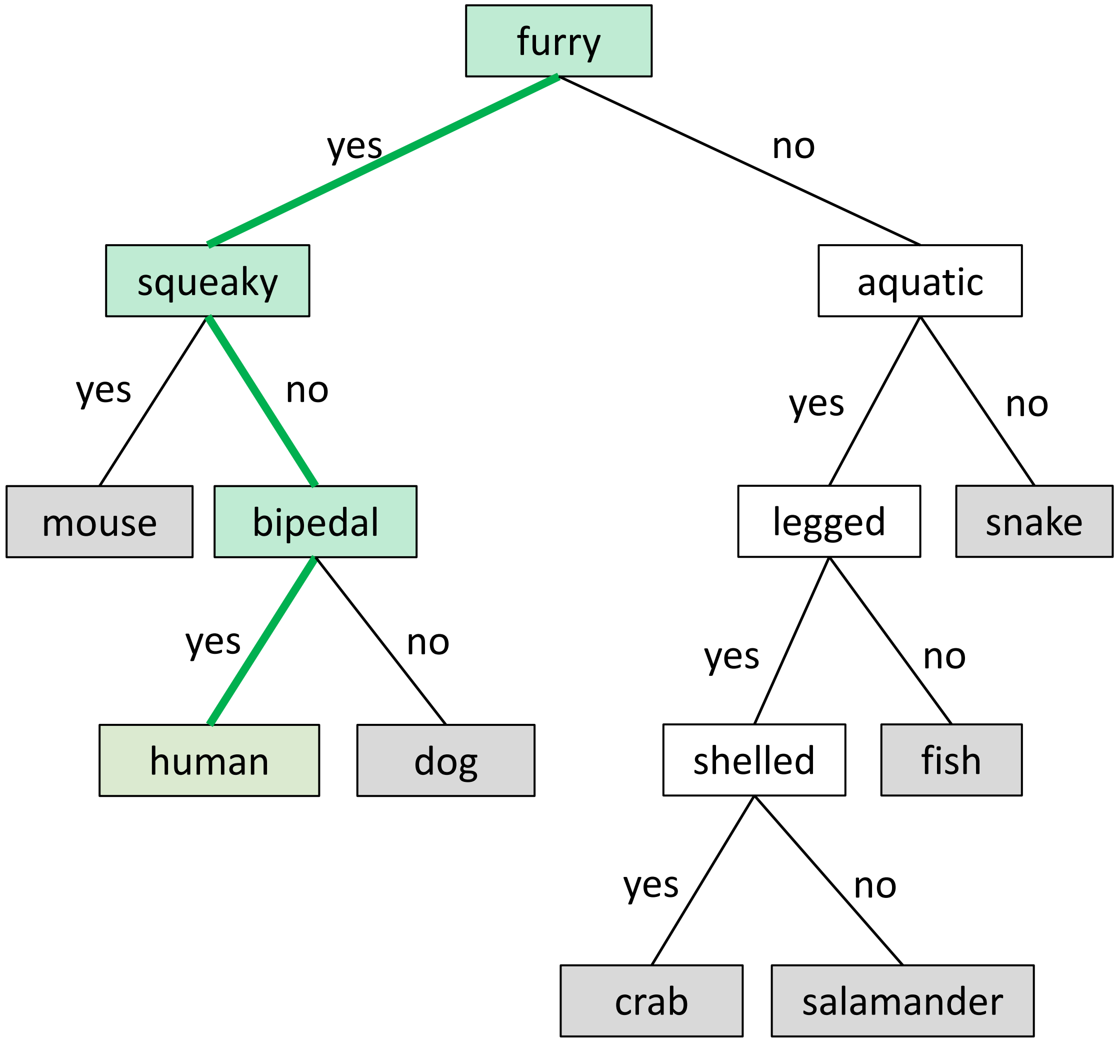
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3.



```
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:

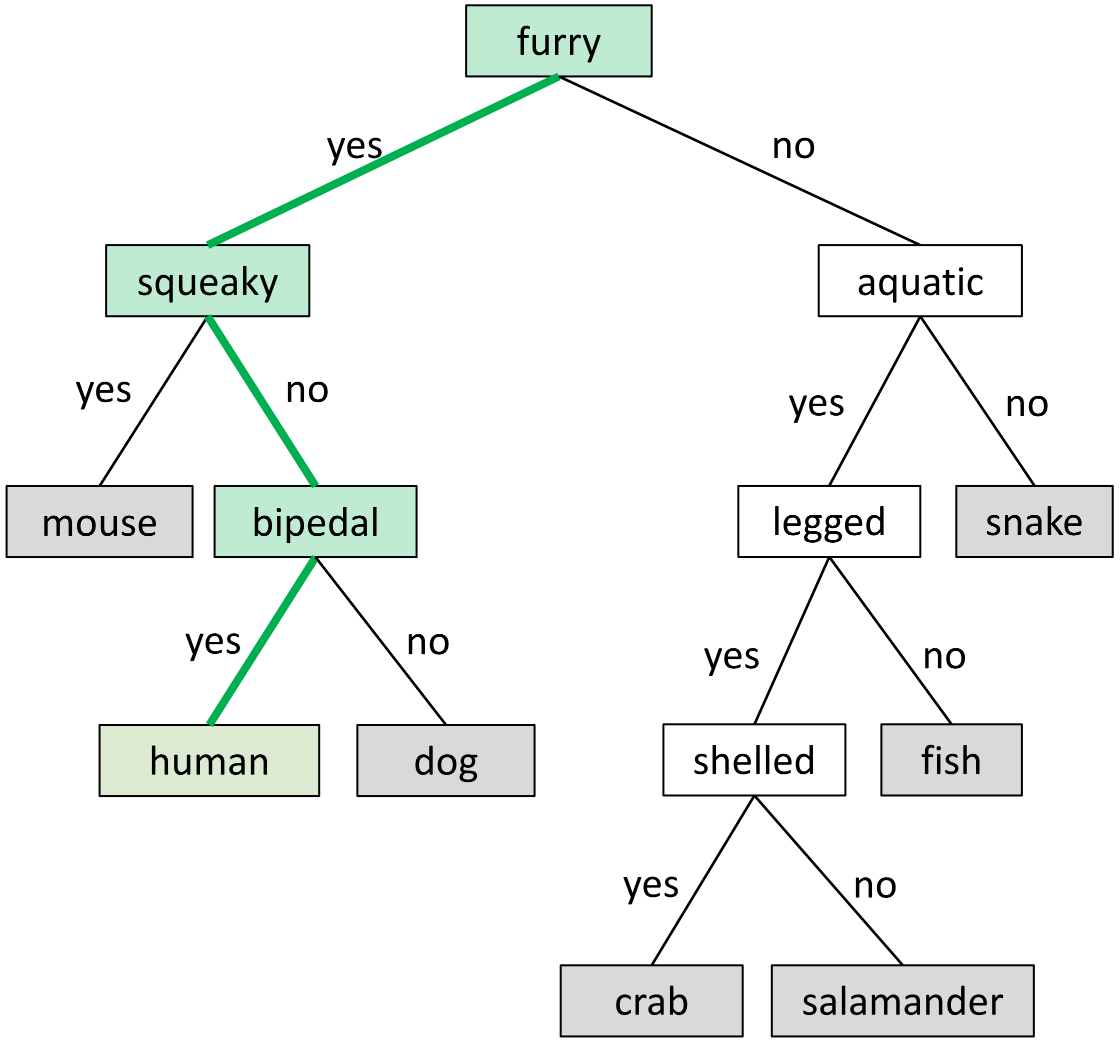
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   3.1. Print location in tree.
   3.2.



```
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:

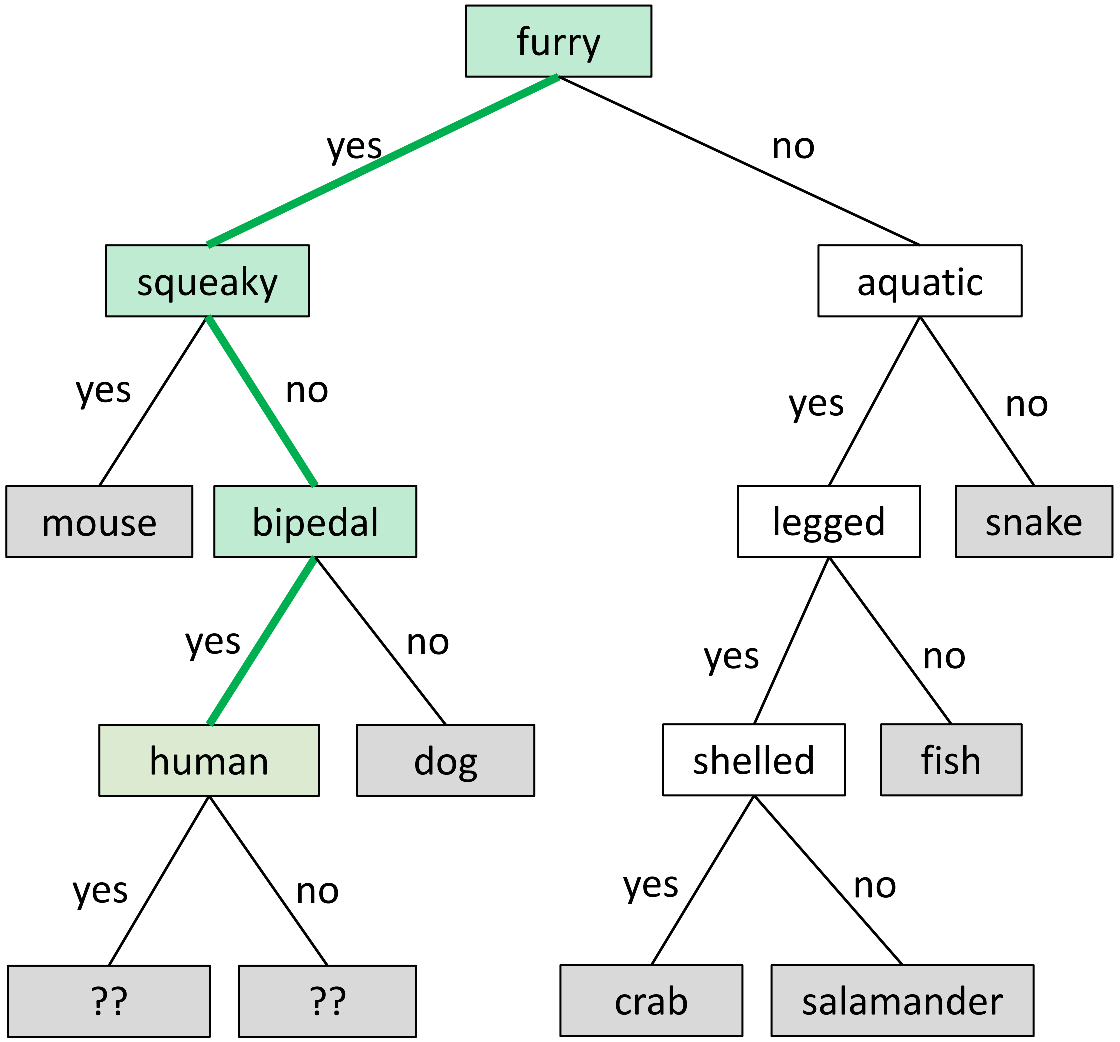
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
    - 3.1. Print location in tree.
    - 3.2. Get name of new animal.
    - 3.3. Get distinguishing characteristic.
    - 3.4.



```
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:

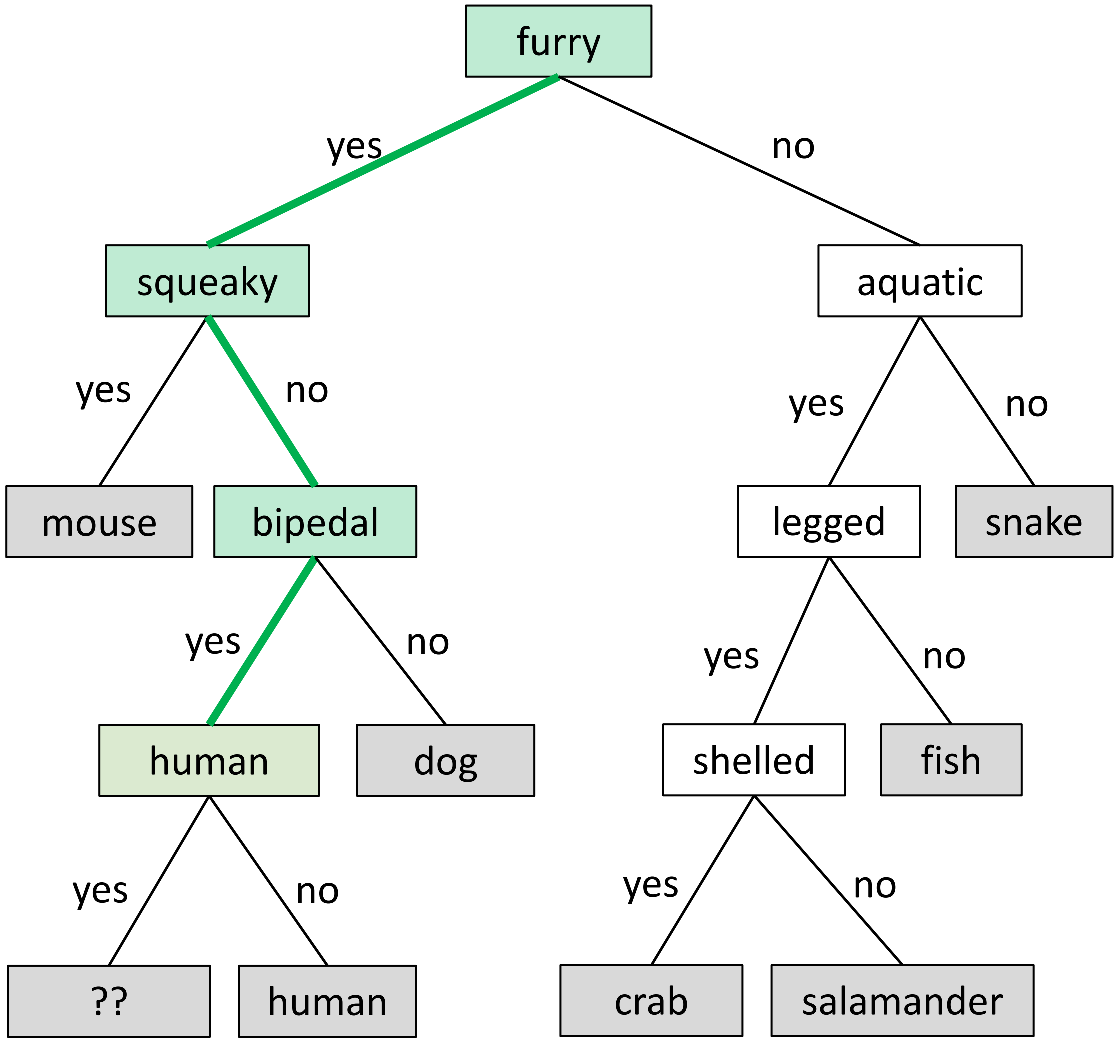
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   3.1. Print location in tree.
   3.2. Get name of new animal.
   3.3. Get distinguishing characteristic.
   3.4. Modify tree:
      3.4.1.



```
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:

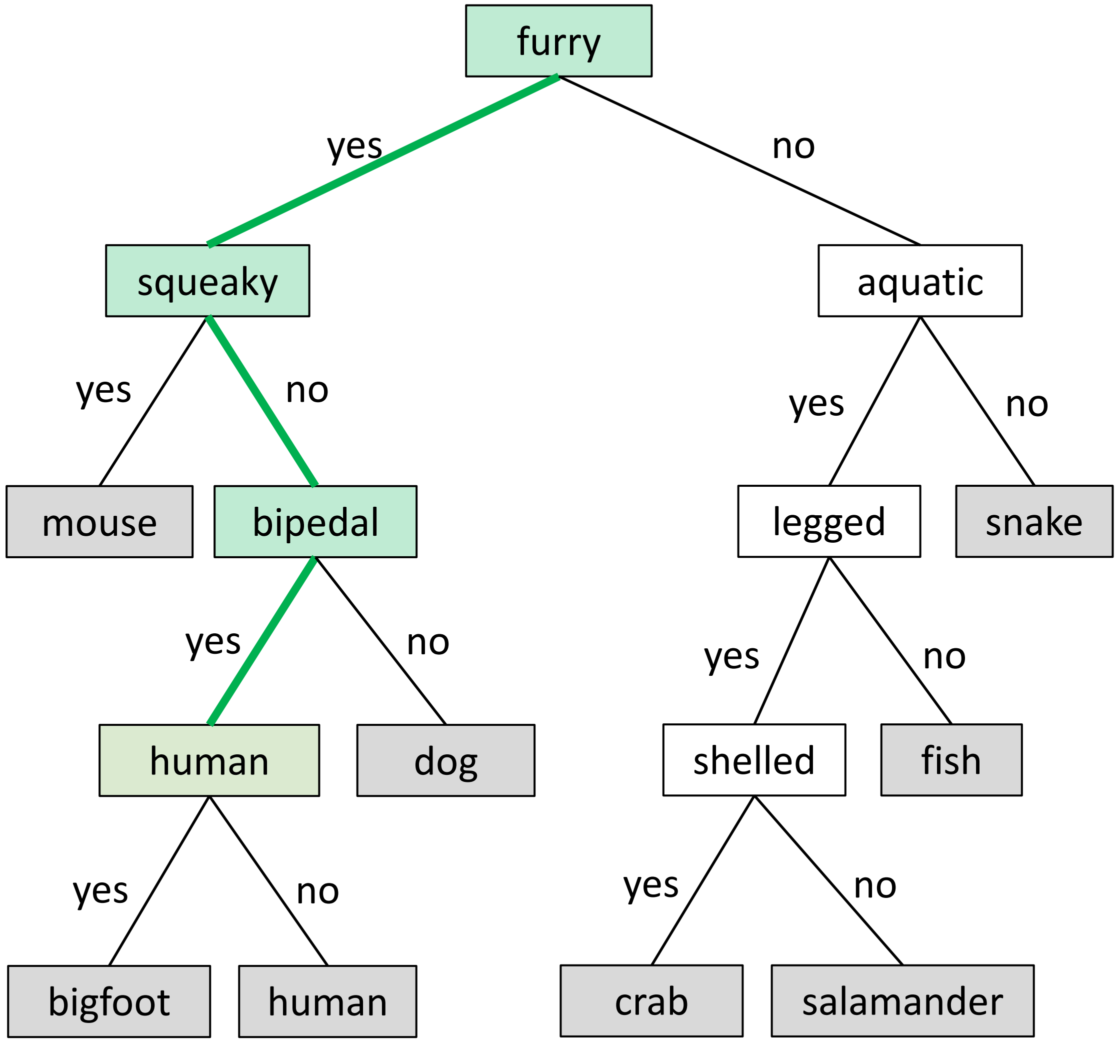
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   - 3.1. Print location in tree.
   - 3.2. Get name of new animal.
   - 3.3. Get distinguishing characteristic.
   - 3.4. Modify tree:
     - 3.4.1. Create two new child nodes at current leaf.
     - 3.4.2.



```
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:

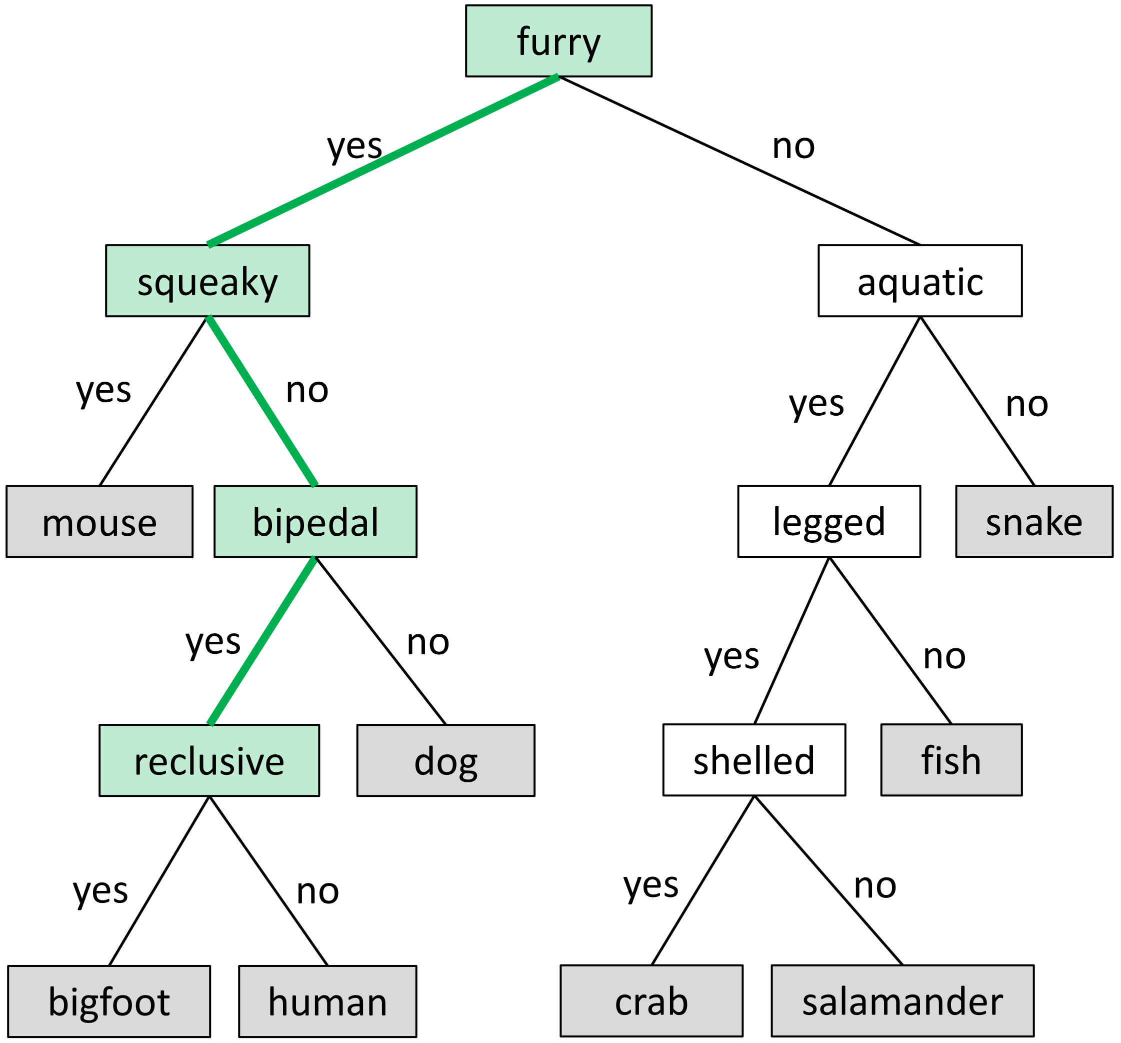
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
    - 3.1. Print location in tree.
    - 3.2. Get name of new animal.
    - 3.3. Get distinguishing characteristic.
    - 3.4. Modify tree:
        - 3.4.1. Create two new child nodes at current leaf.
        - 3.4.2.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

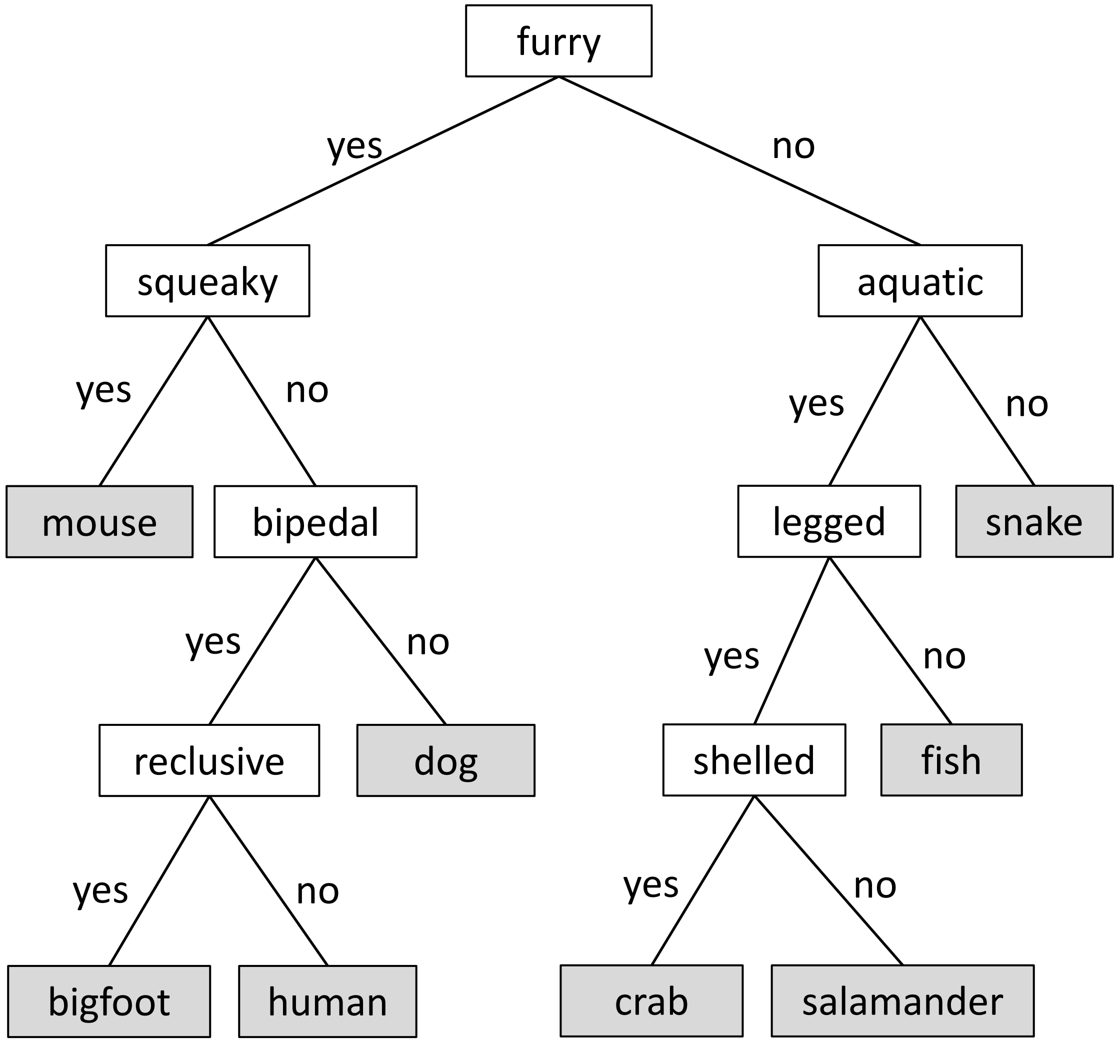
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   - 3.1. Print location in tree.
   - 3.2. Get name of new animal.
   - 3.3. Get distinguishing characteristic.
   - 3.4. Modify tree:
     - 3.4.1. Create two new child nodes at current leaf.
     - 3.4.2. Make "no"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

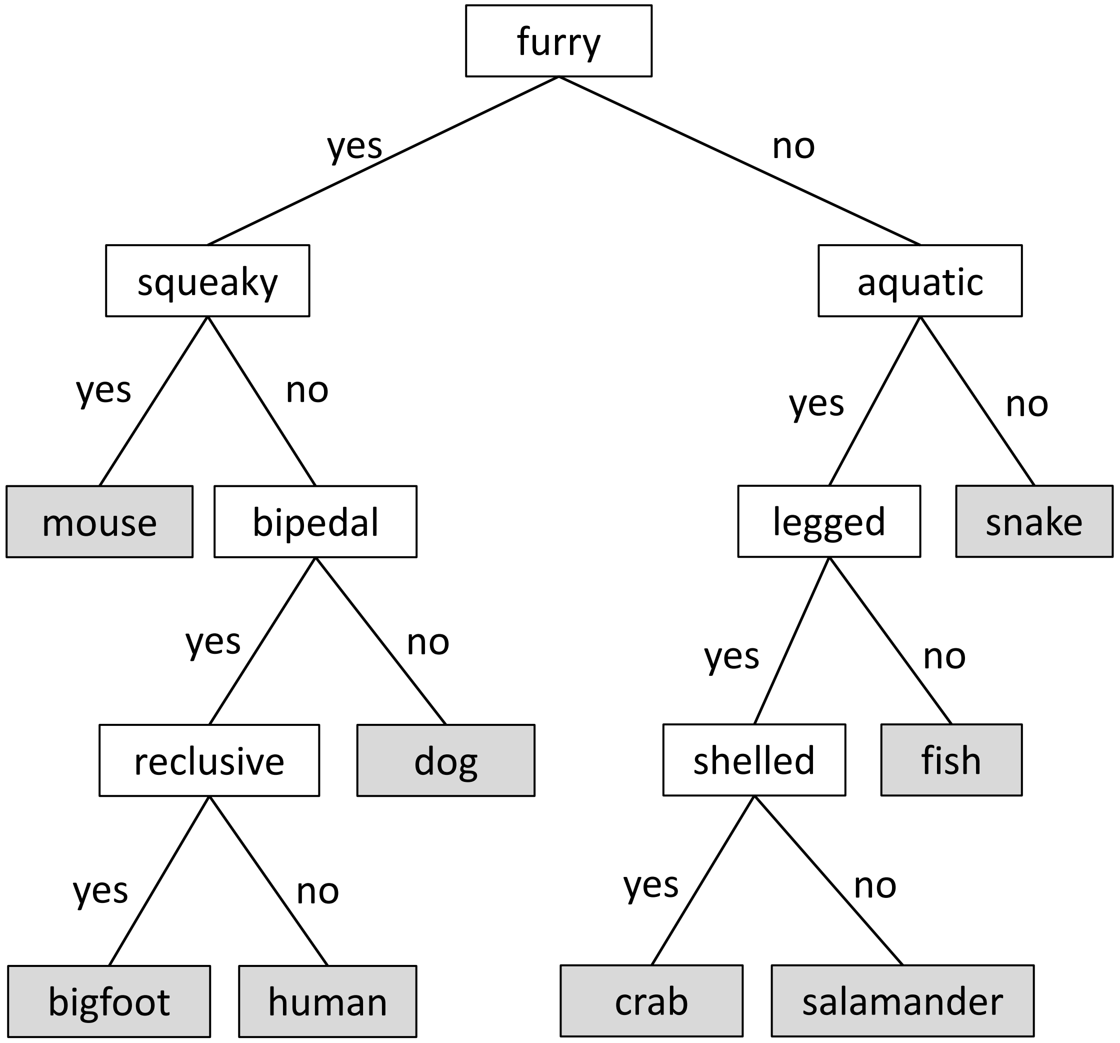
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
    - 3.1. Print location in tree.
    - 3.2. Get name of new animal.
    - 3.3. Get distinguishing characteristic.
    - 3.4. Modify tree:
        - 3.4.1. Create two new child nodes at current leaf.
        - 3.4.2.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

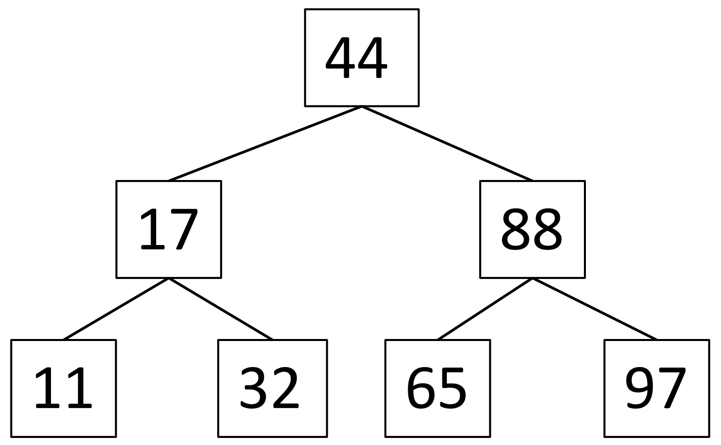
1. Yes/No questions to navigate to a leaf (animal).
2. Is animal correct?
3. If not:
   - 3.1. Print location in tree.
   - 3.2. Get name of new animal.
   - 3.3. Get distinguishing characteristic.
   - 3.4. Modify tree:
     - 3.4.1. Create two new child nodes at current leaf.
     - 3.4.2. Make "no"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



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


    ...
}
```

File read/writing

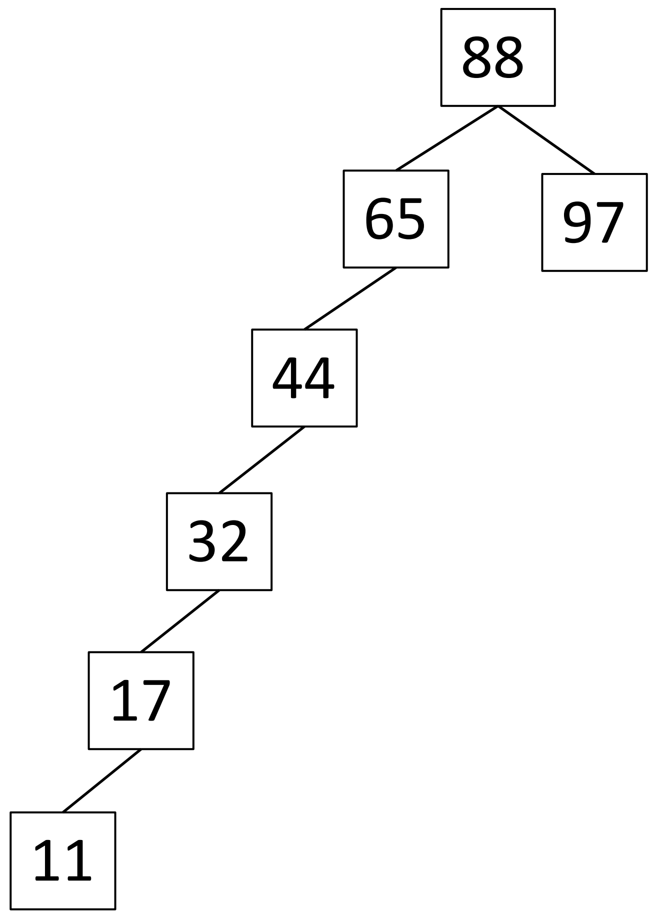
Order Matters



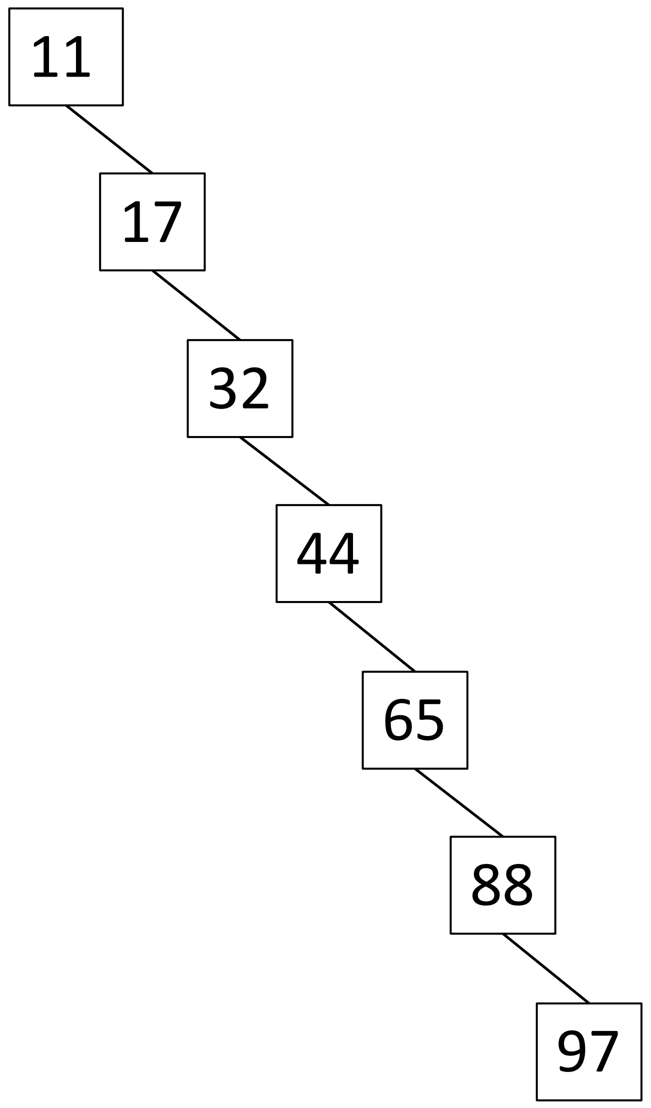
44, 17, 88, 11, 32, 65, 97

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

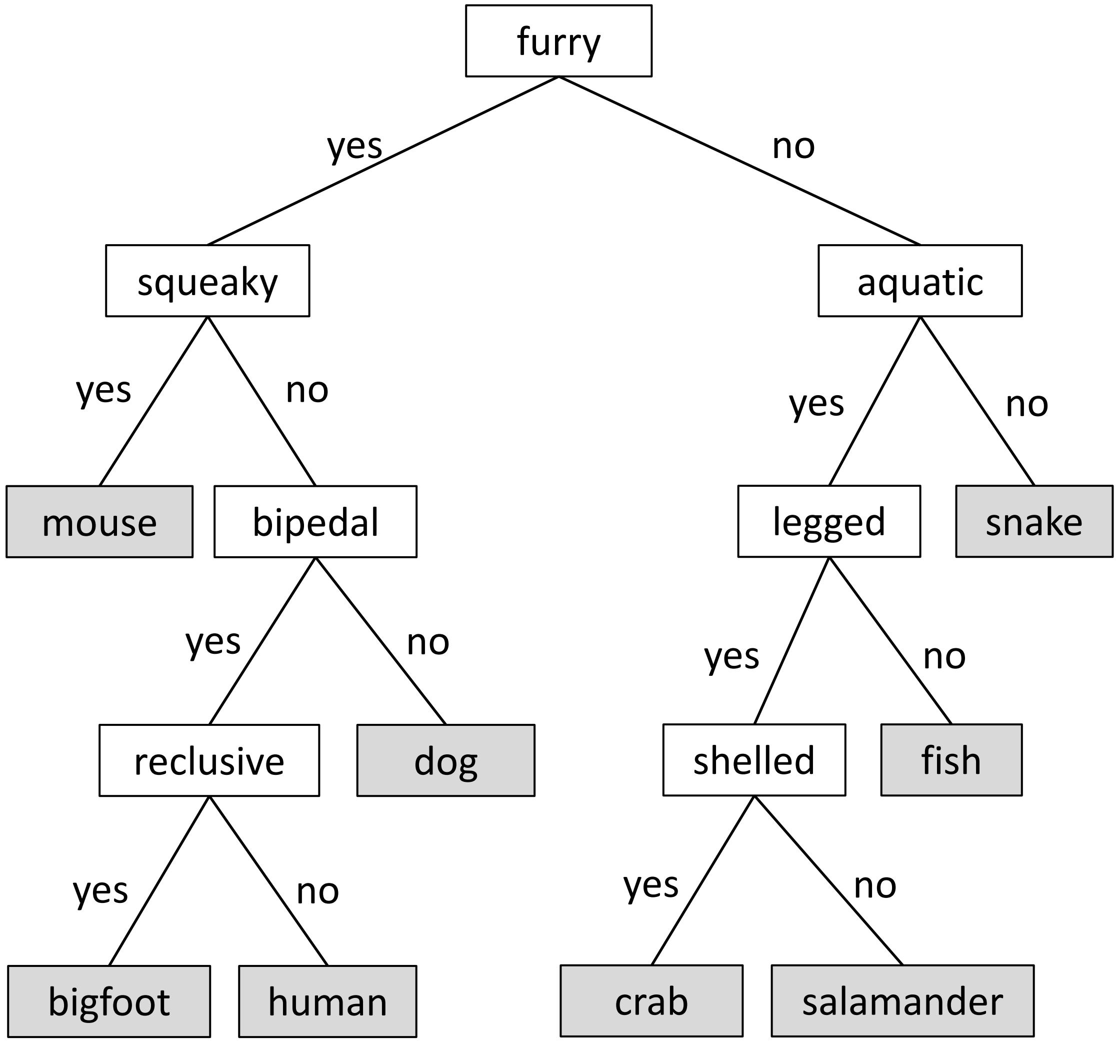
44, 88, 65, 97, 17, 32, 11



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

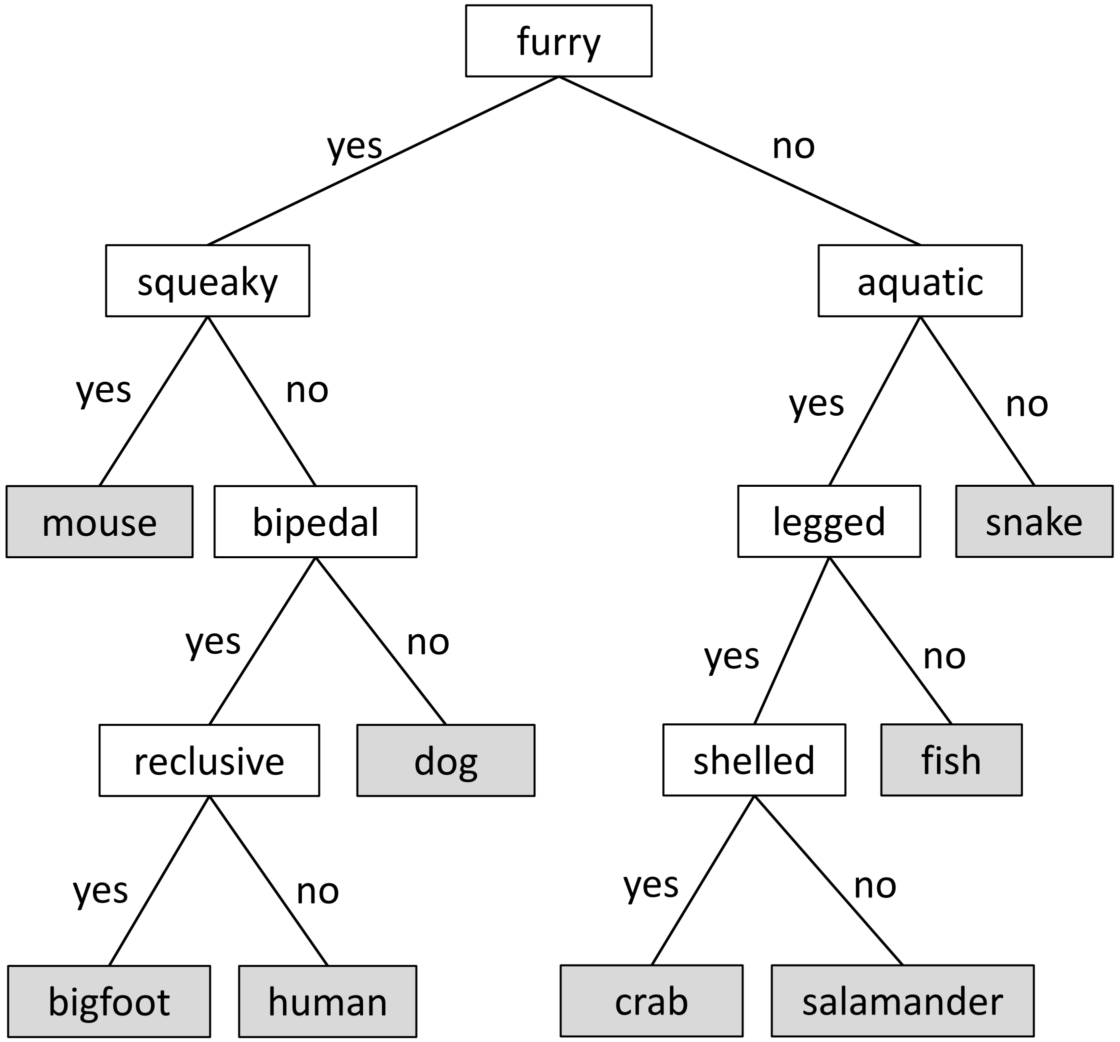


11, 17, 32, 44, 65, 88, 97



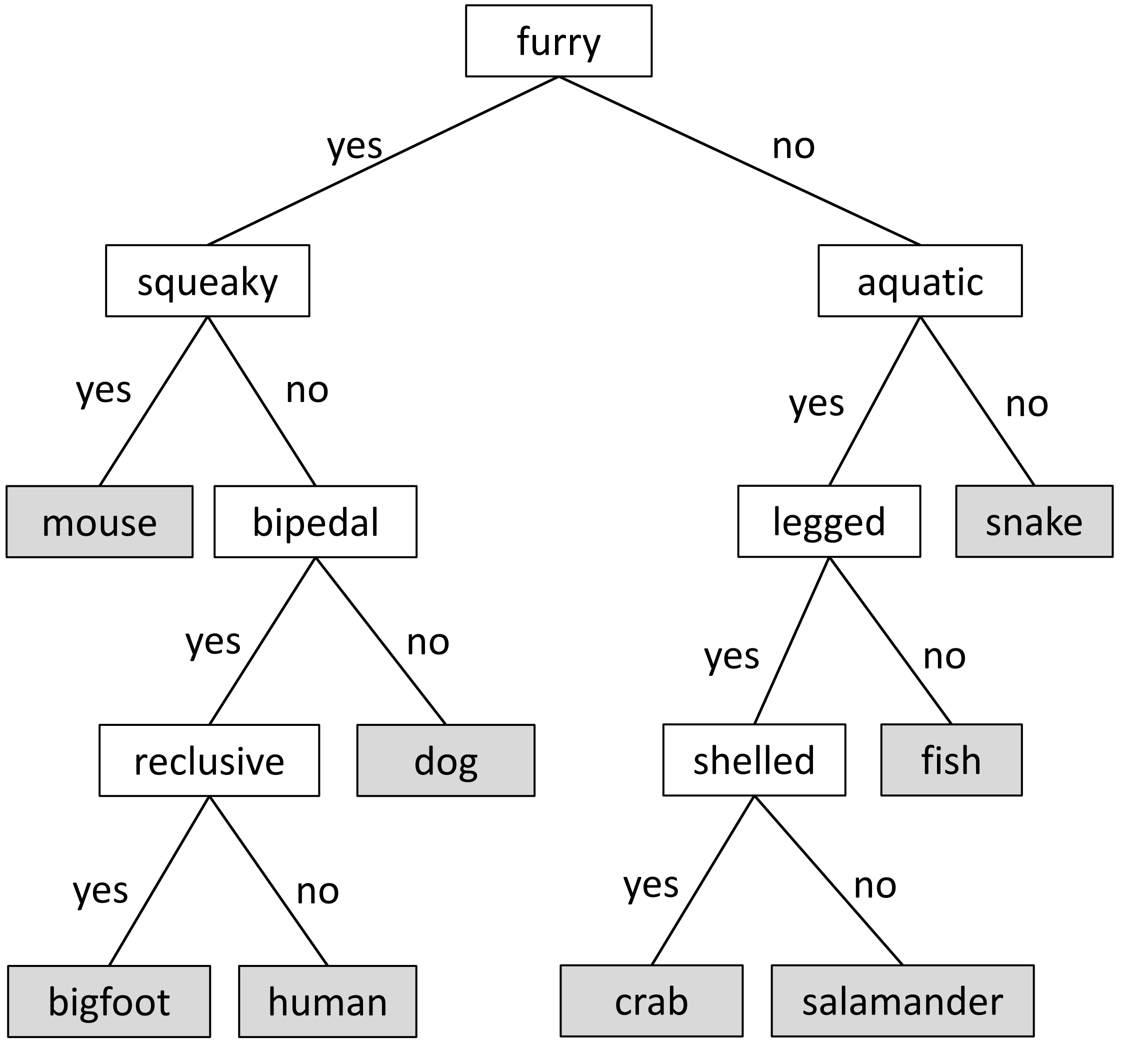
```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;


    ...
}
```

File read/writing



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

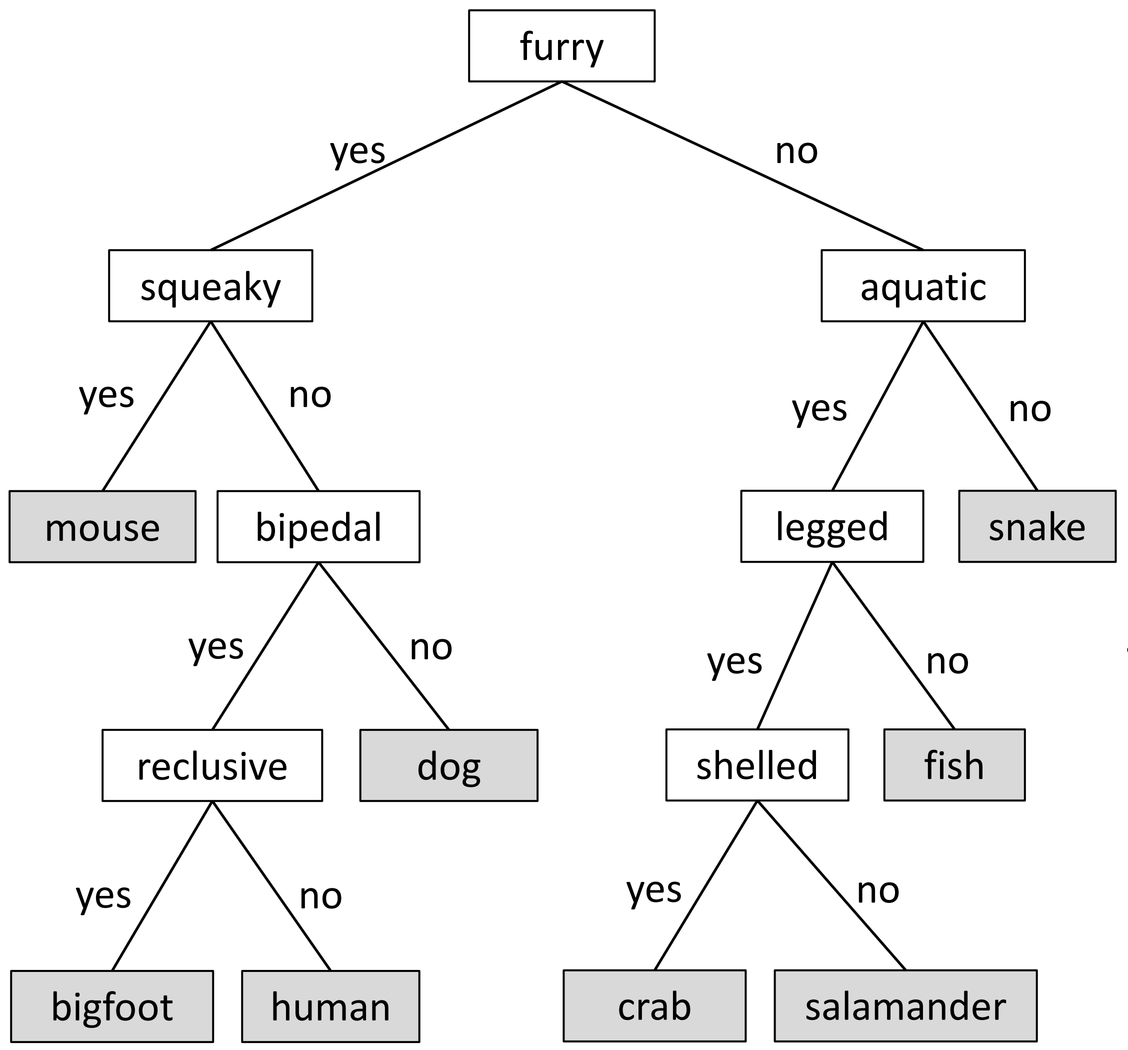
File read/writing



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

Save to file:

File read/writing

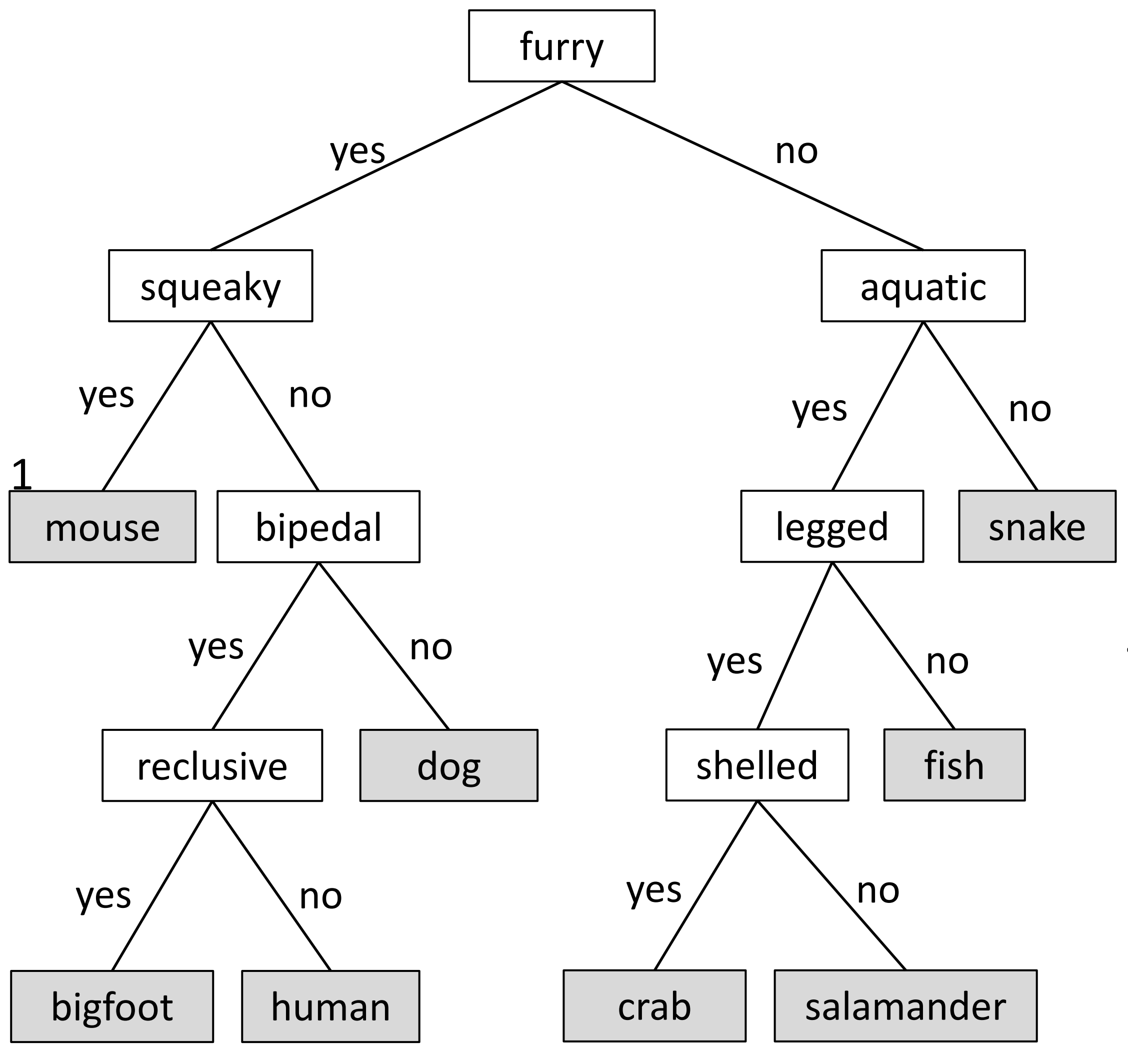


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

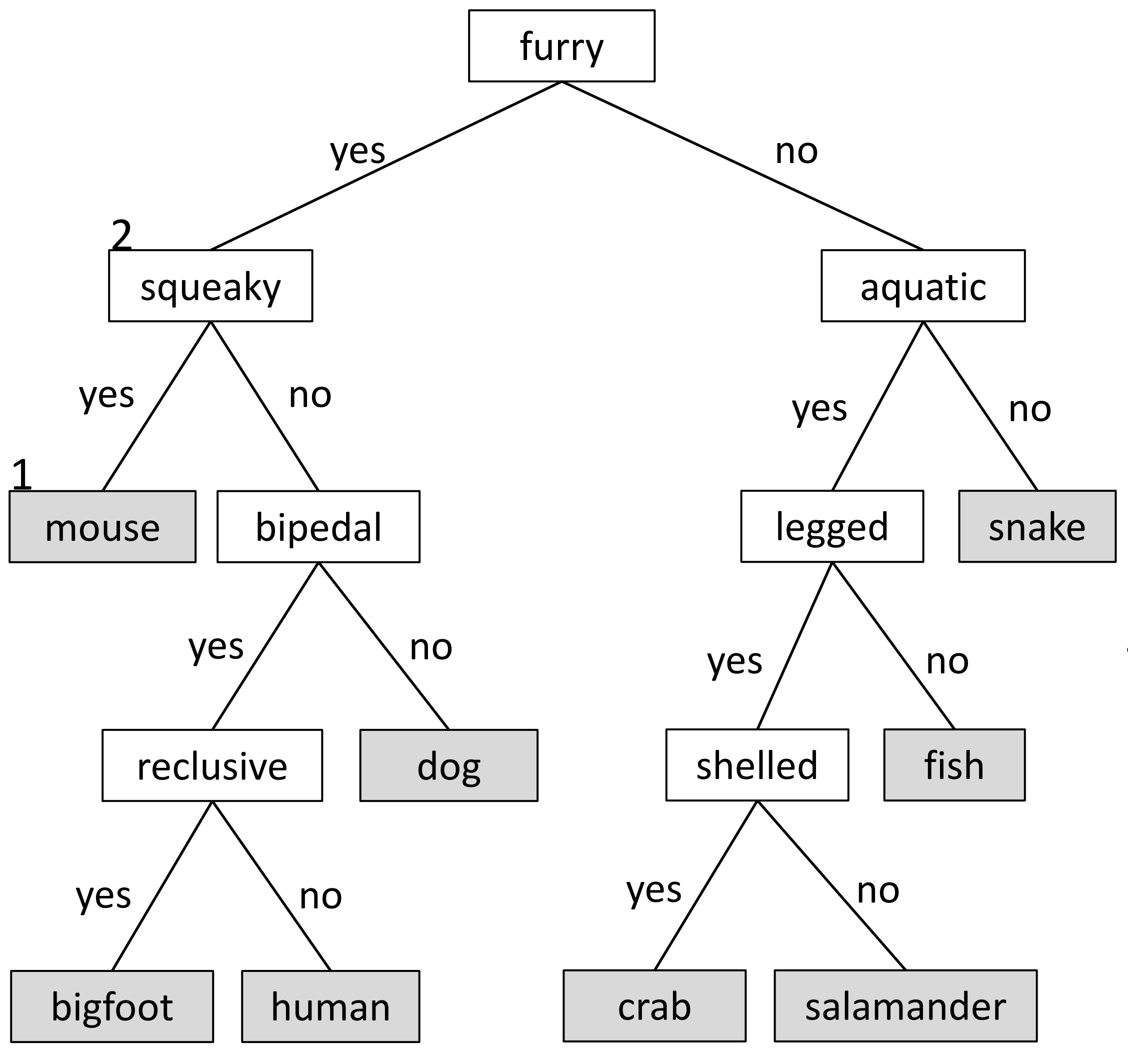


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

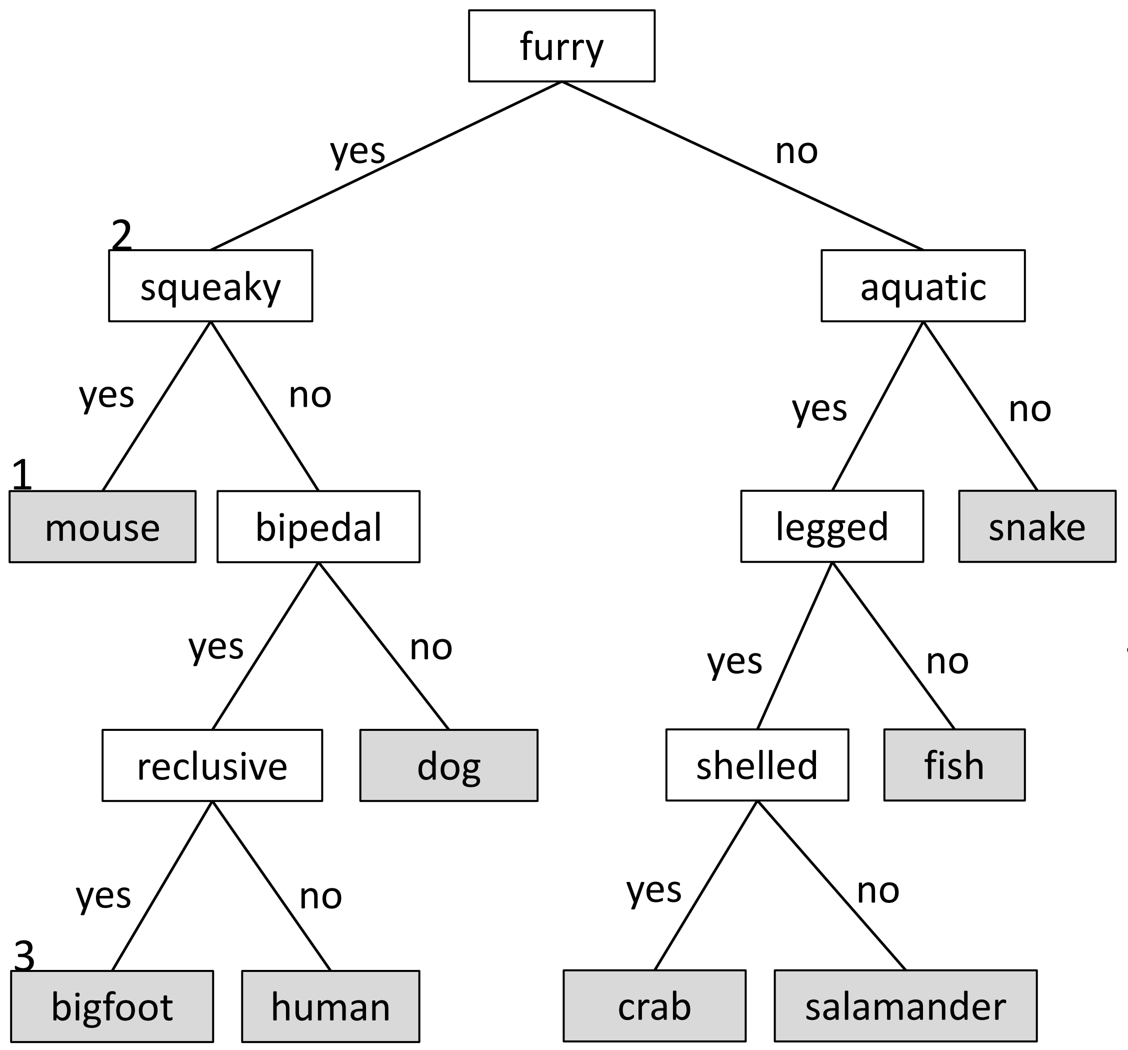


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

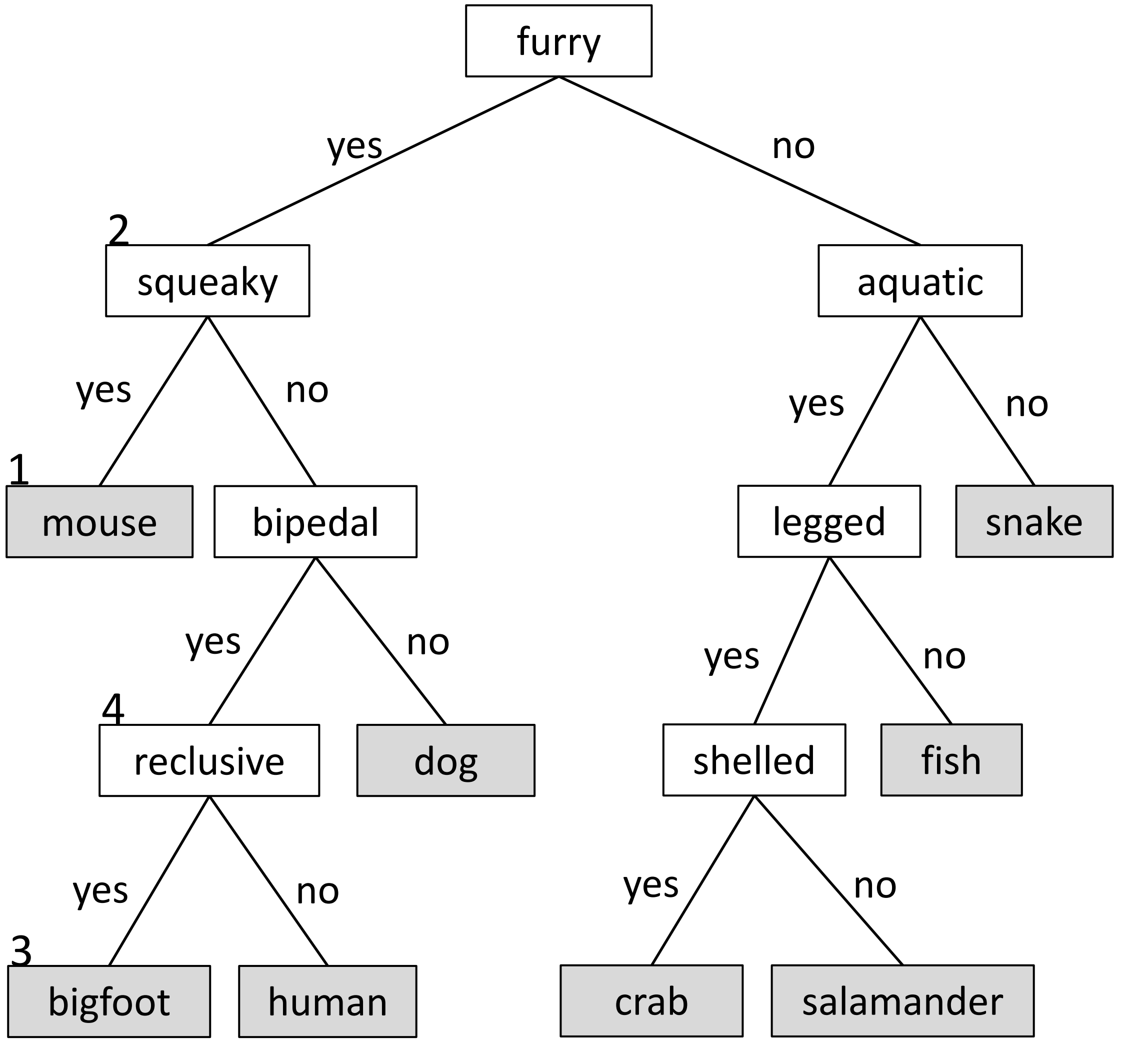


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

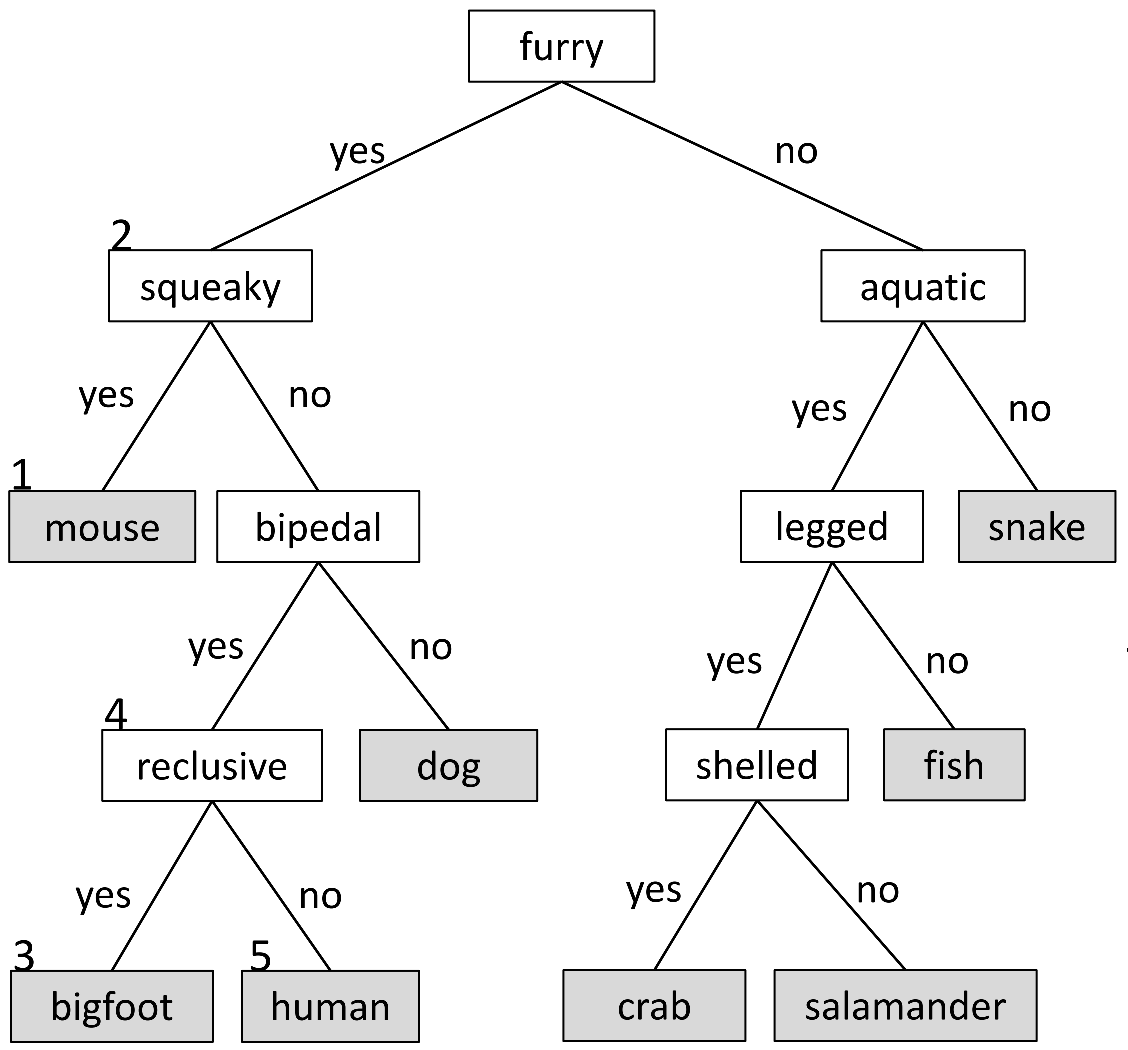


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

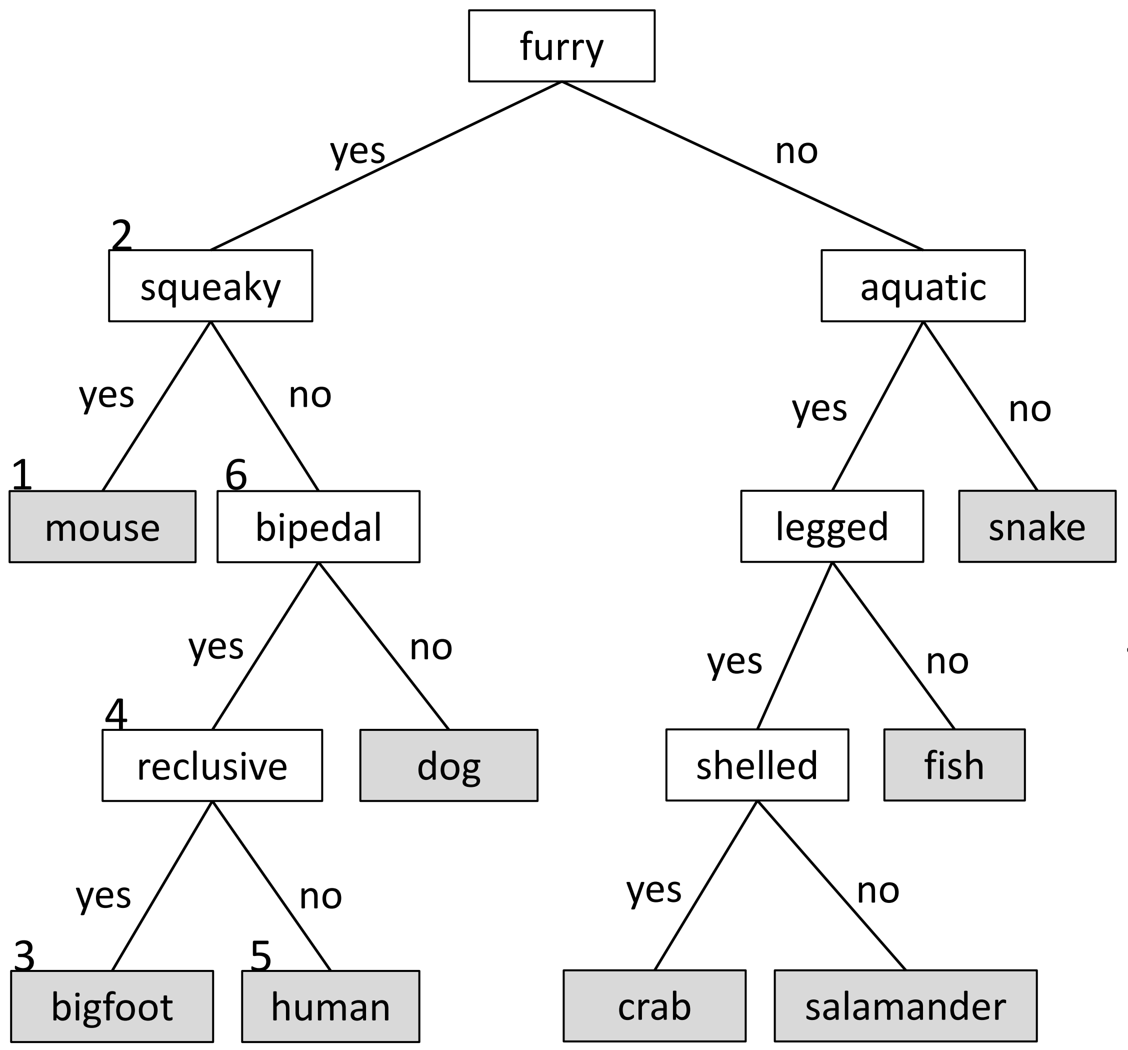


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

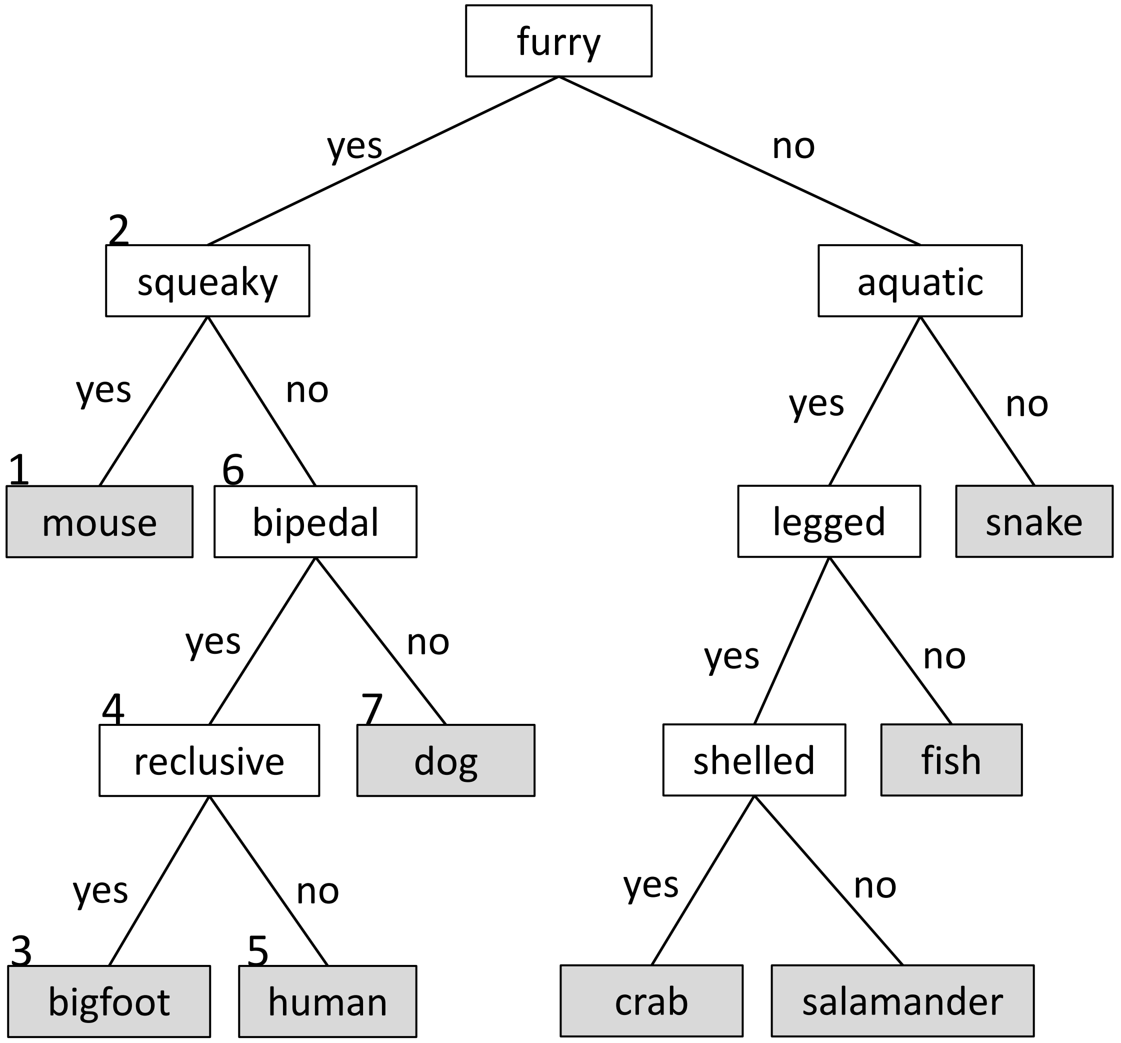


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

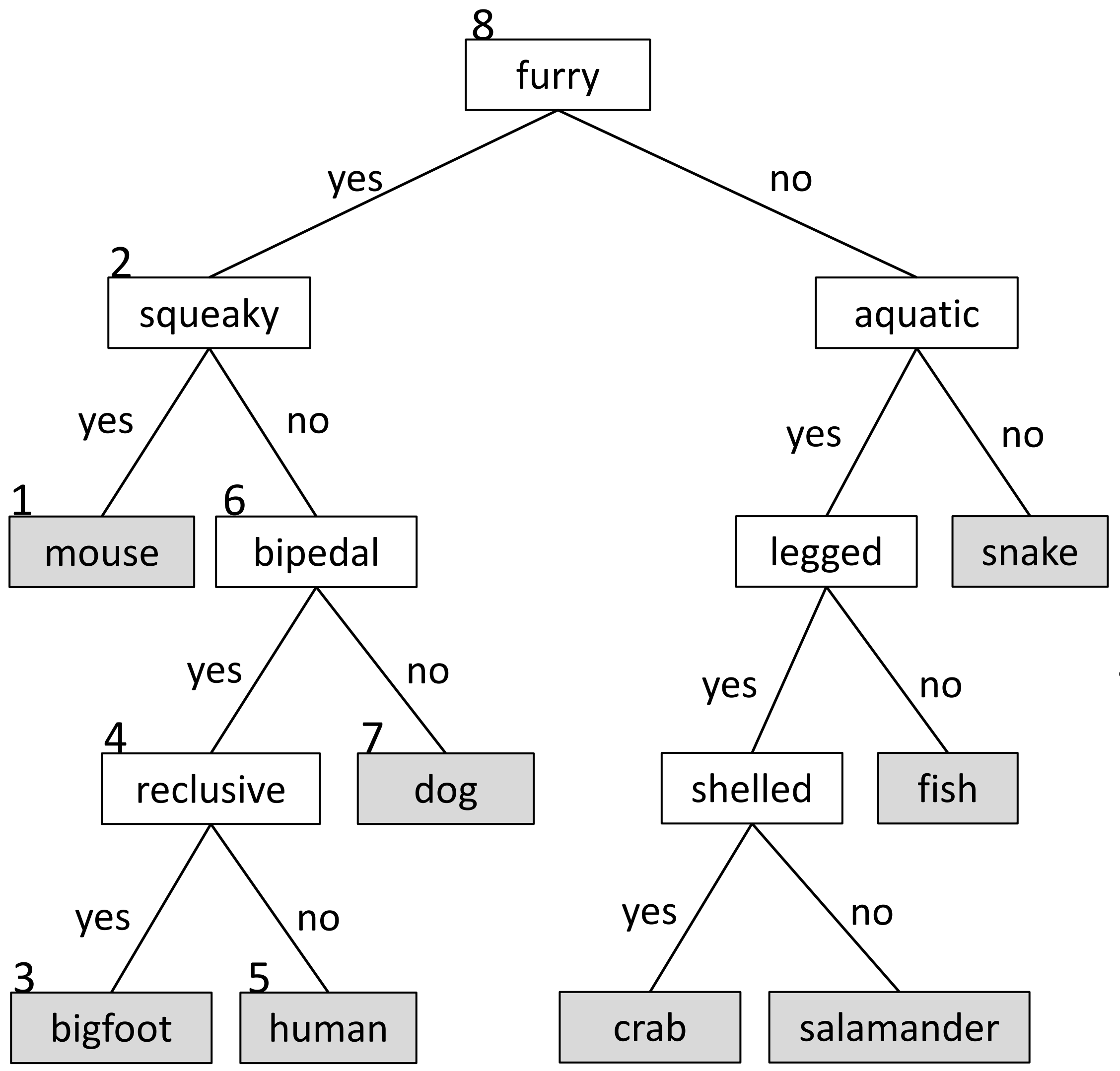


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

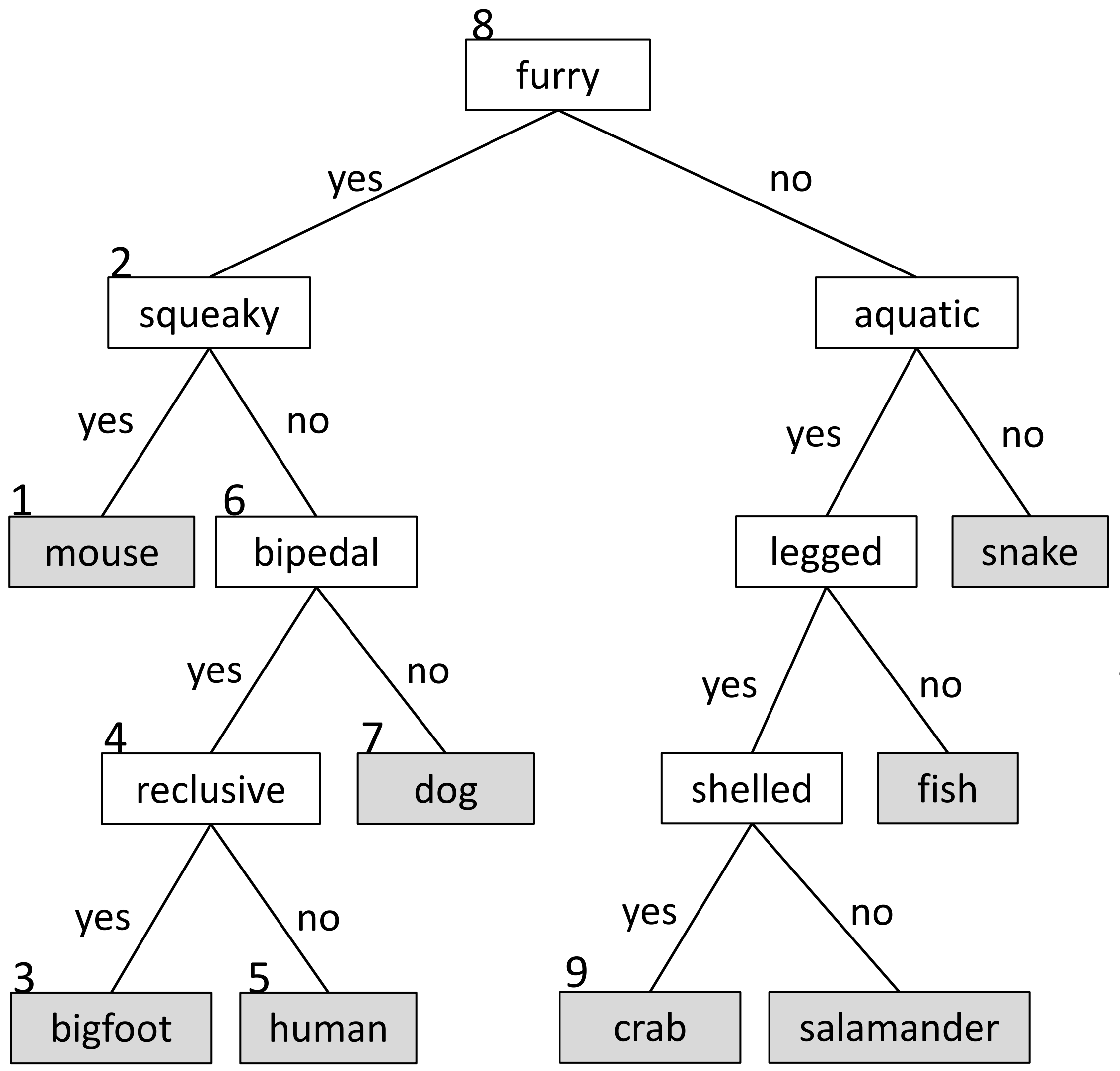


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

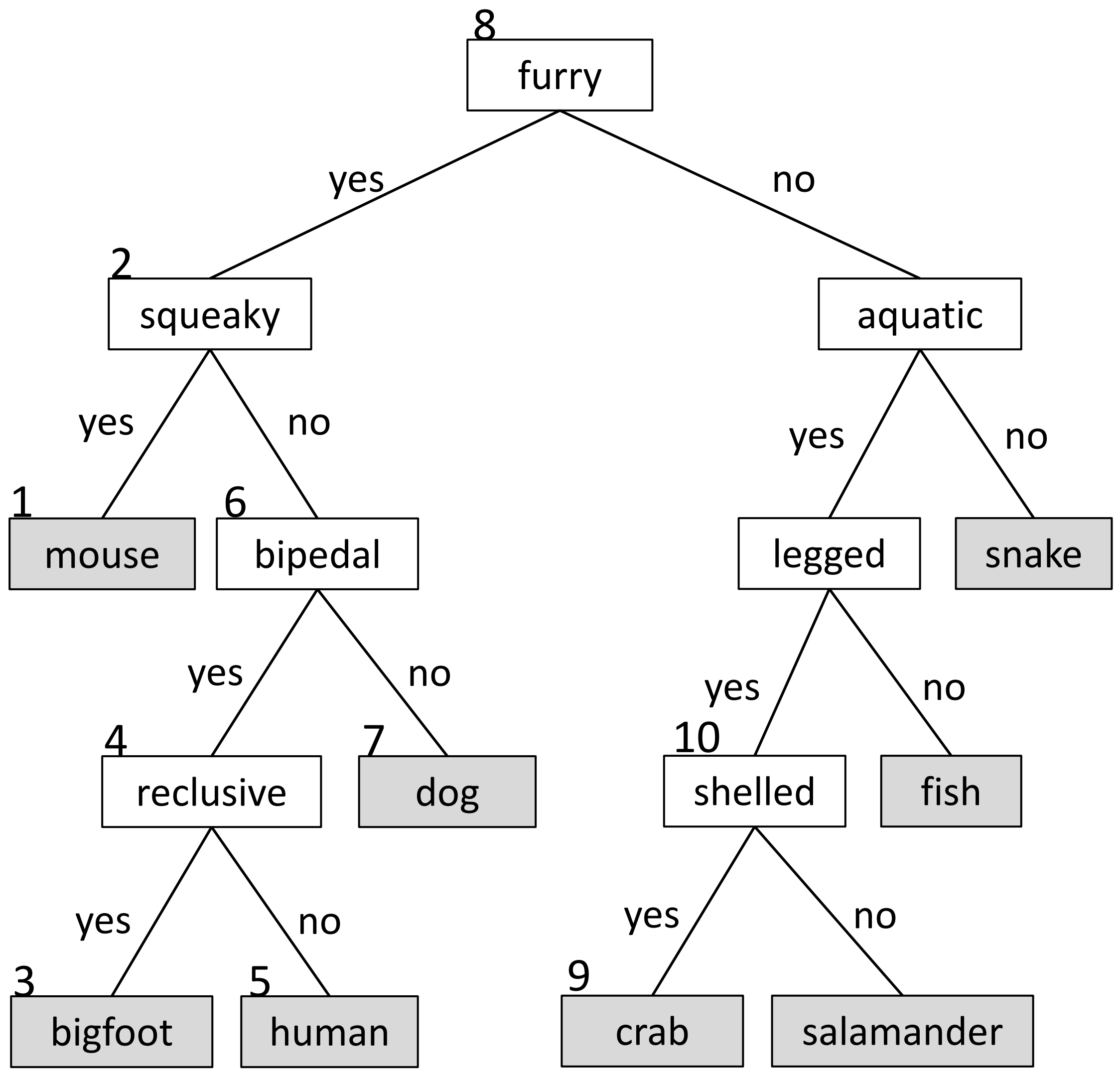


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

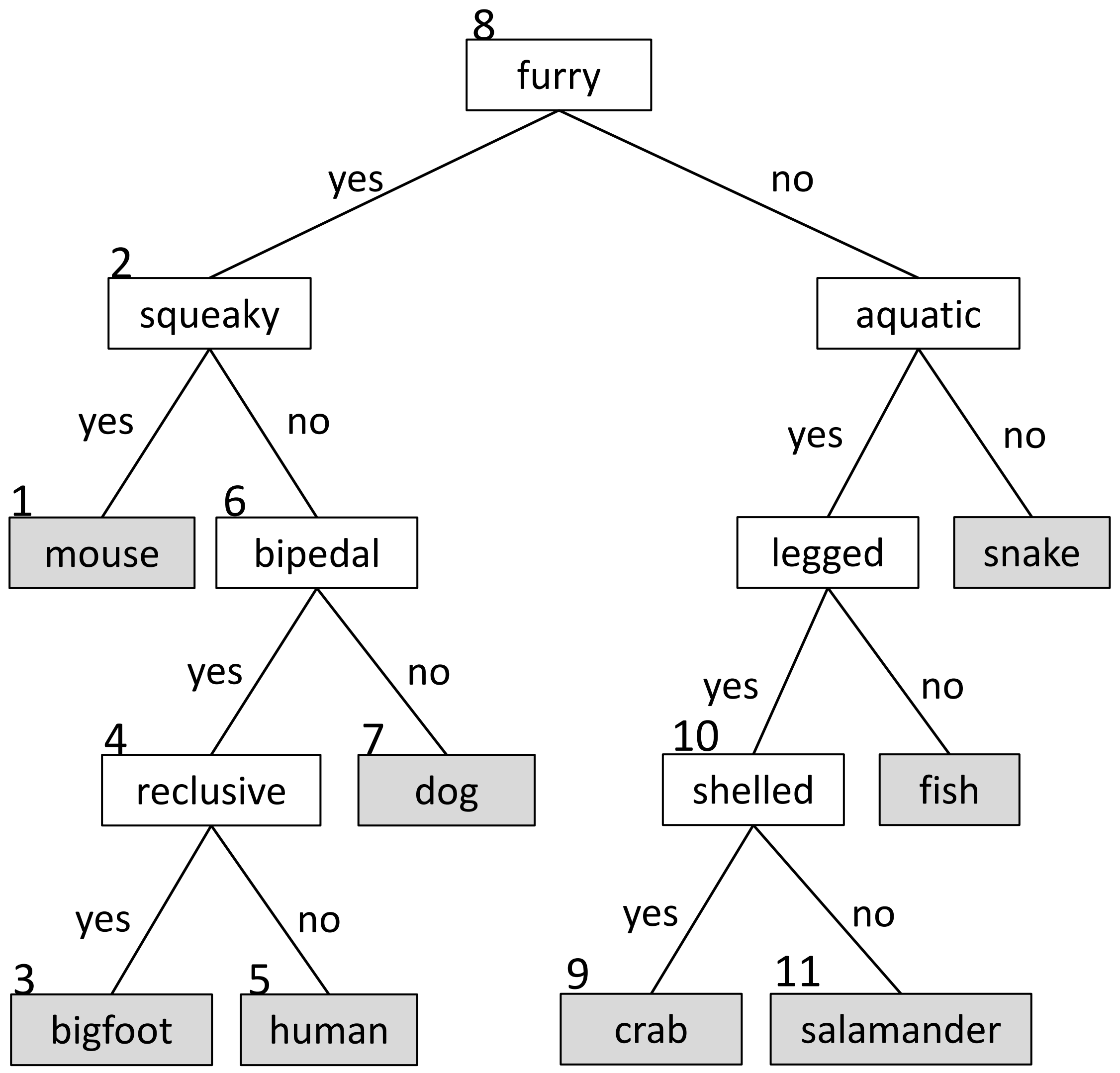


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

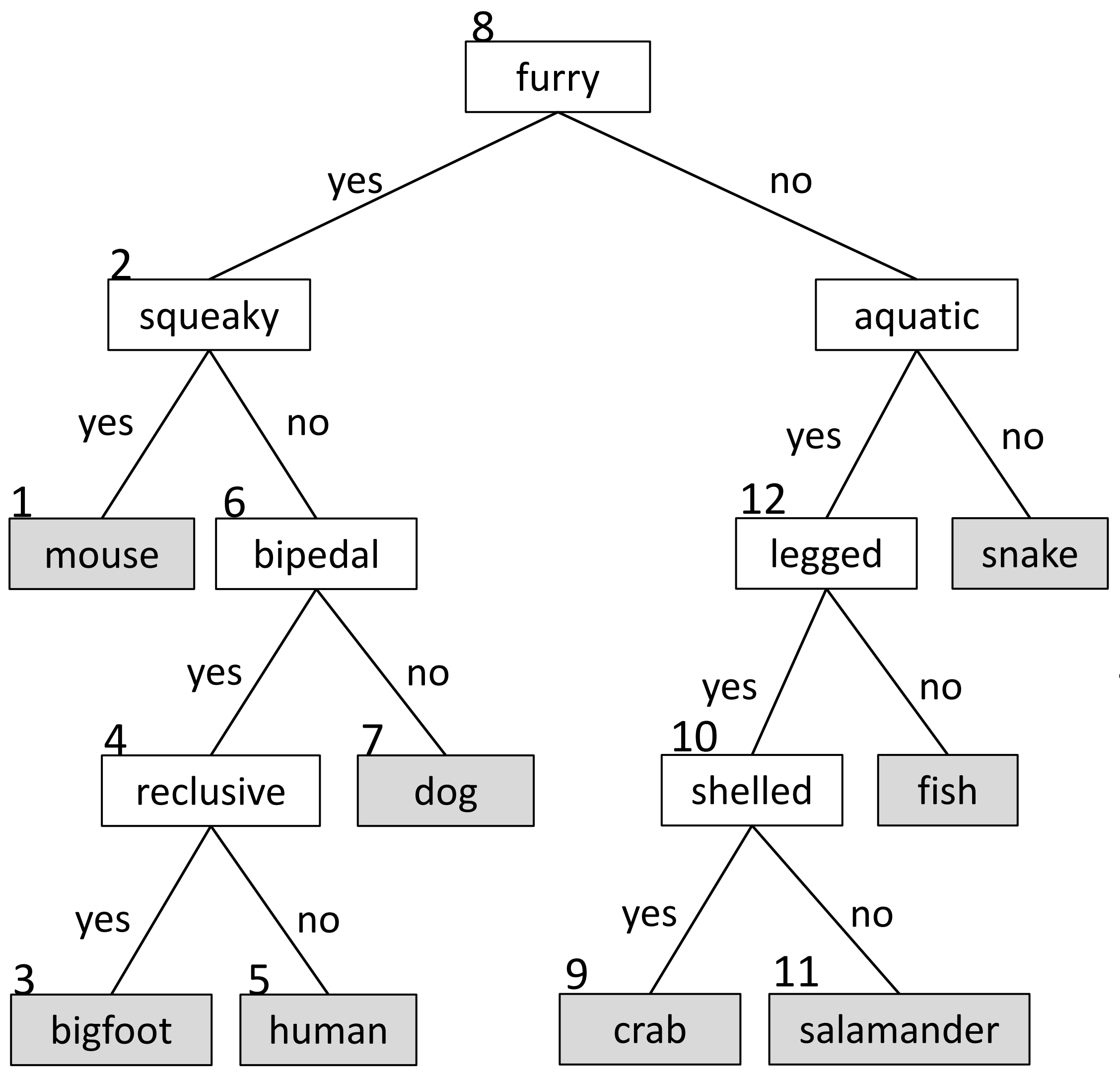


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

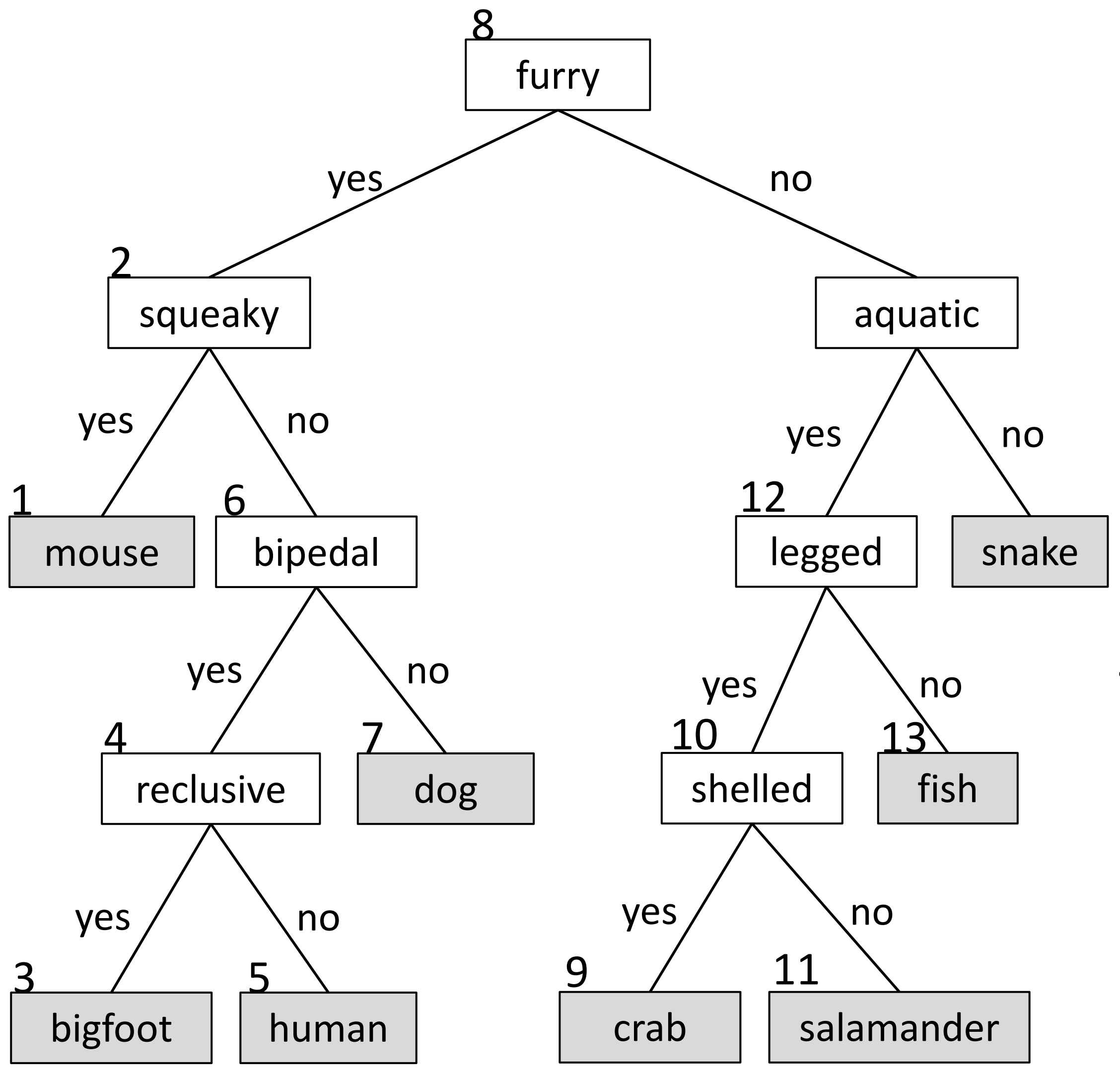


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

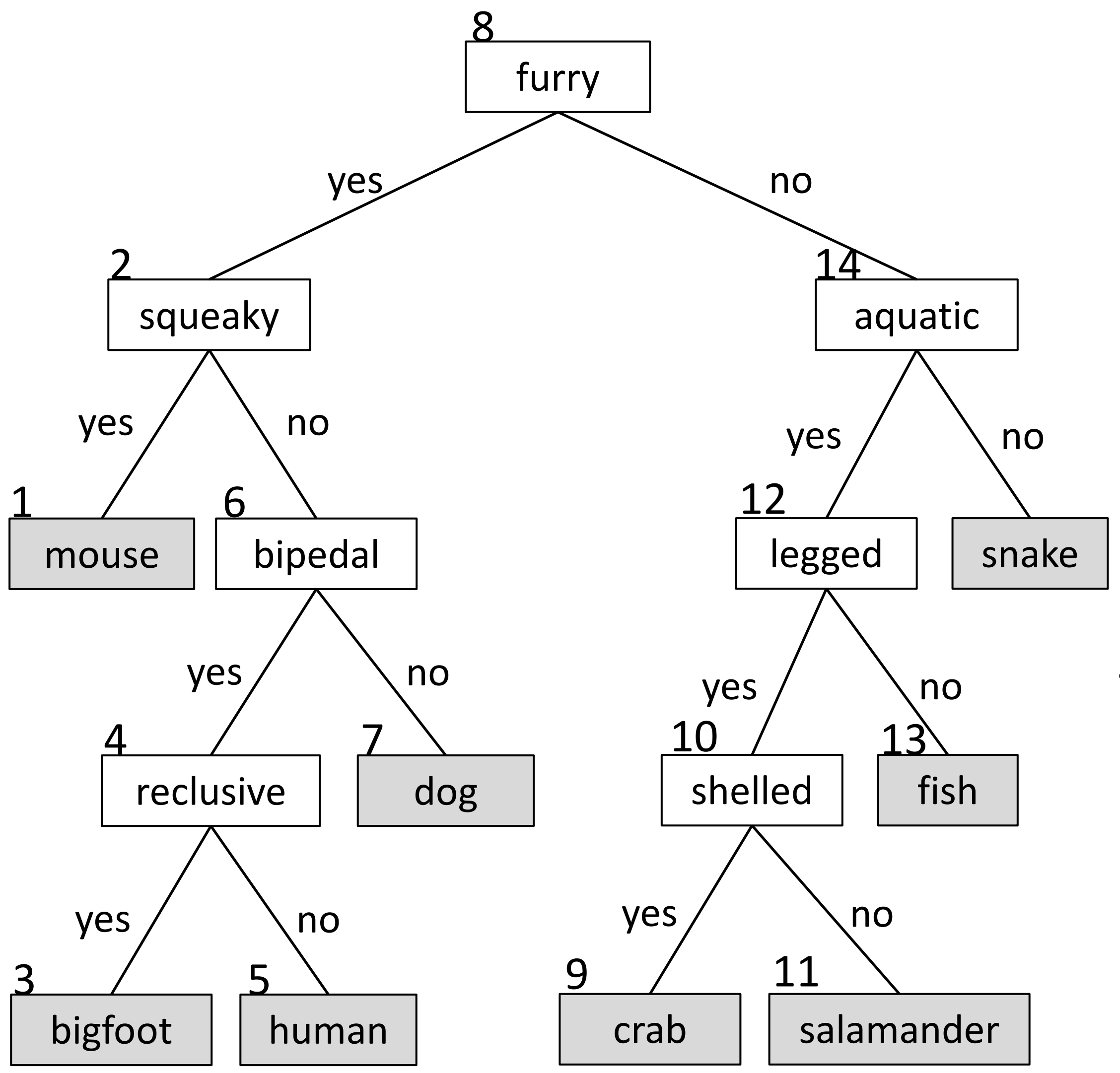


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

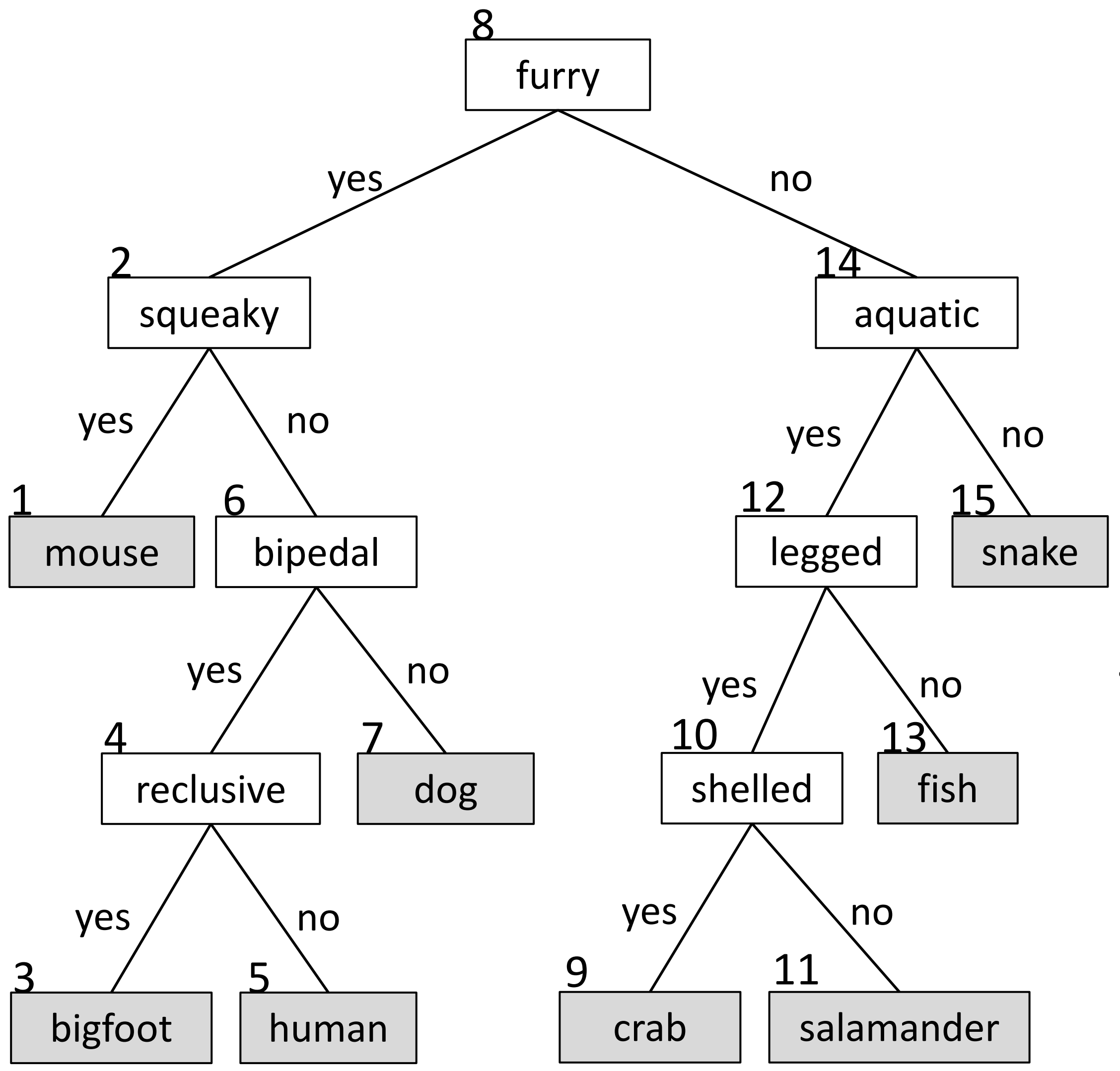


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

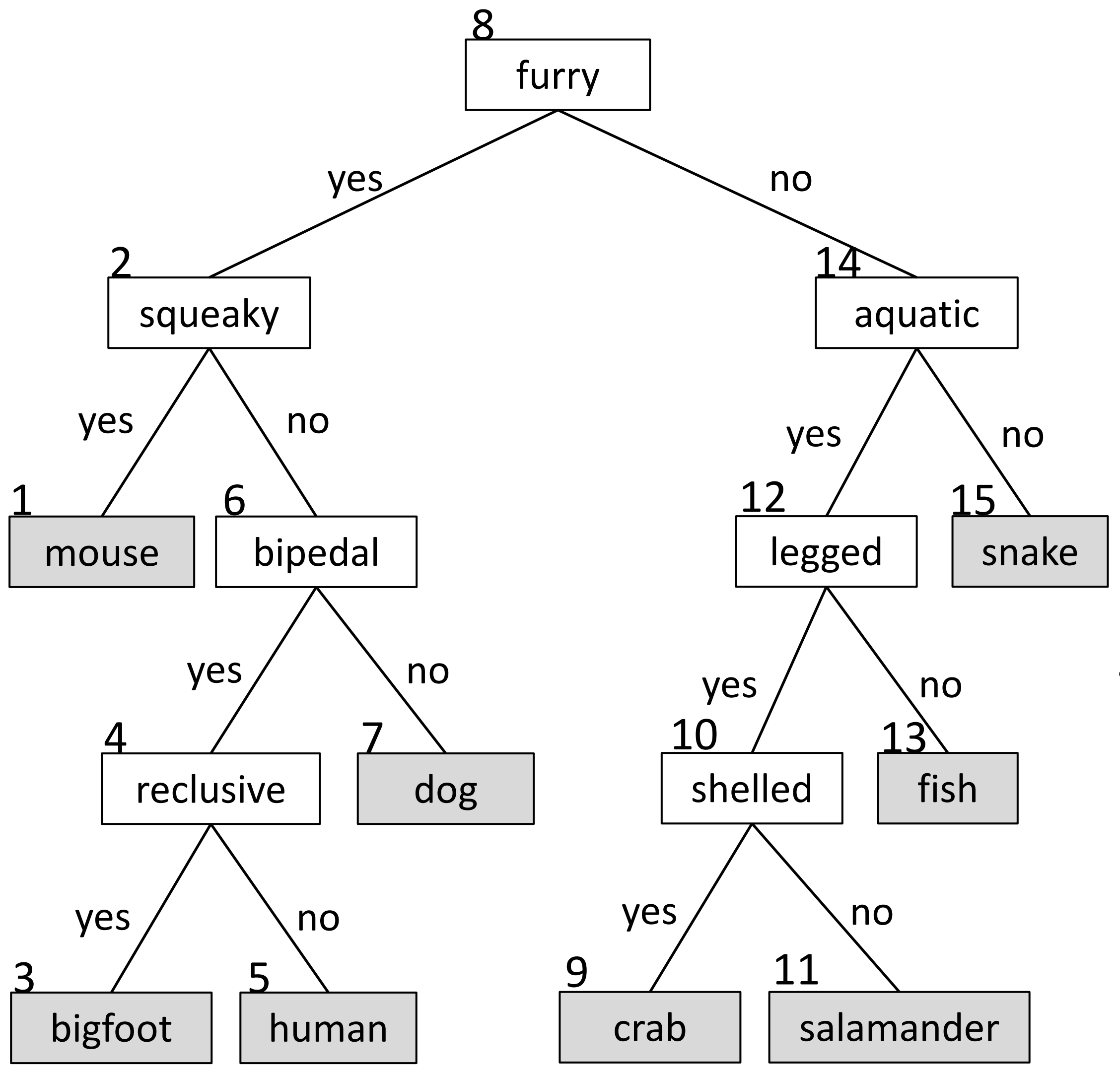


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

Save to file:

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

File read/writing

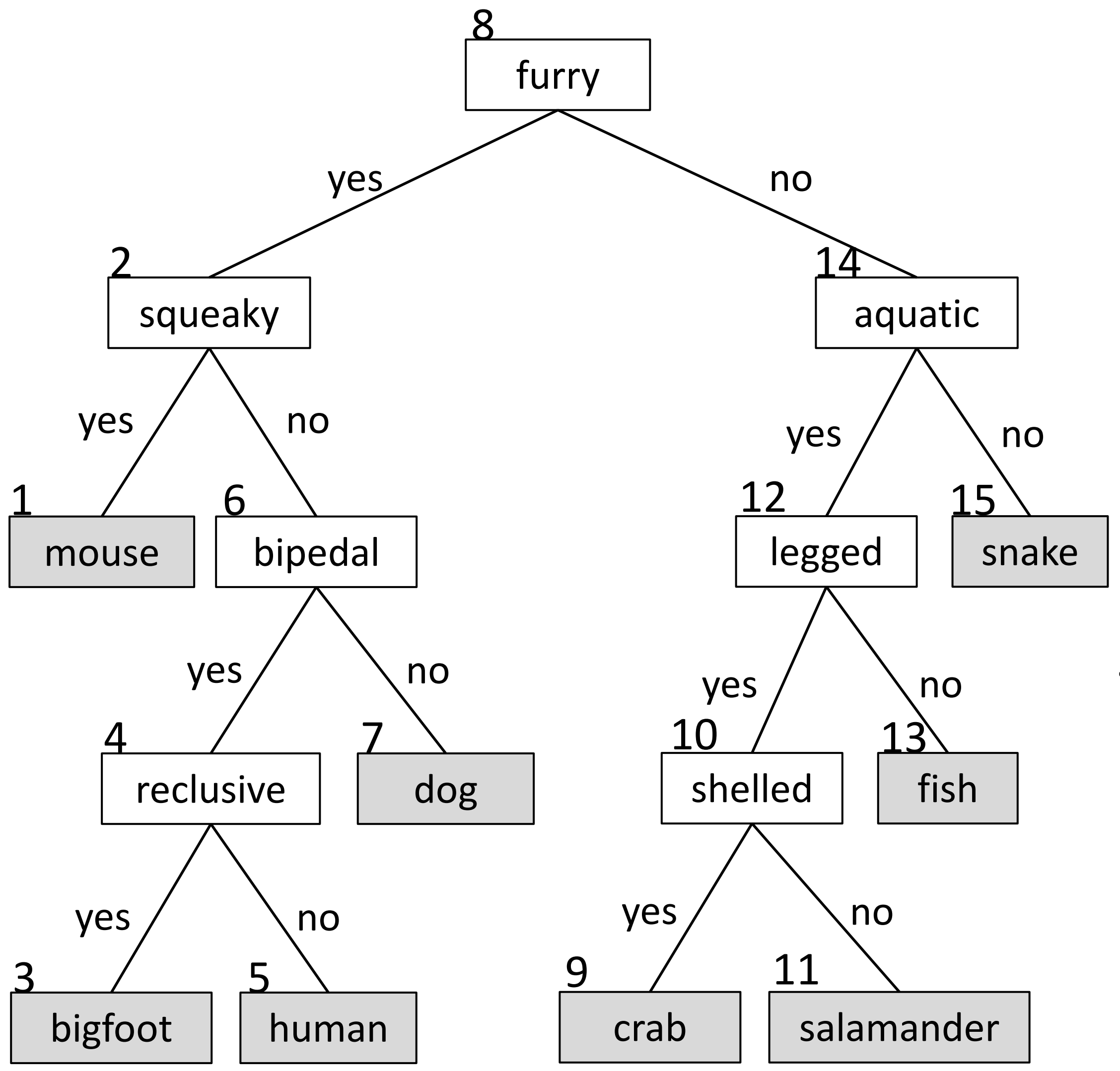


```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

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



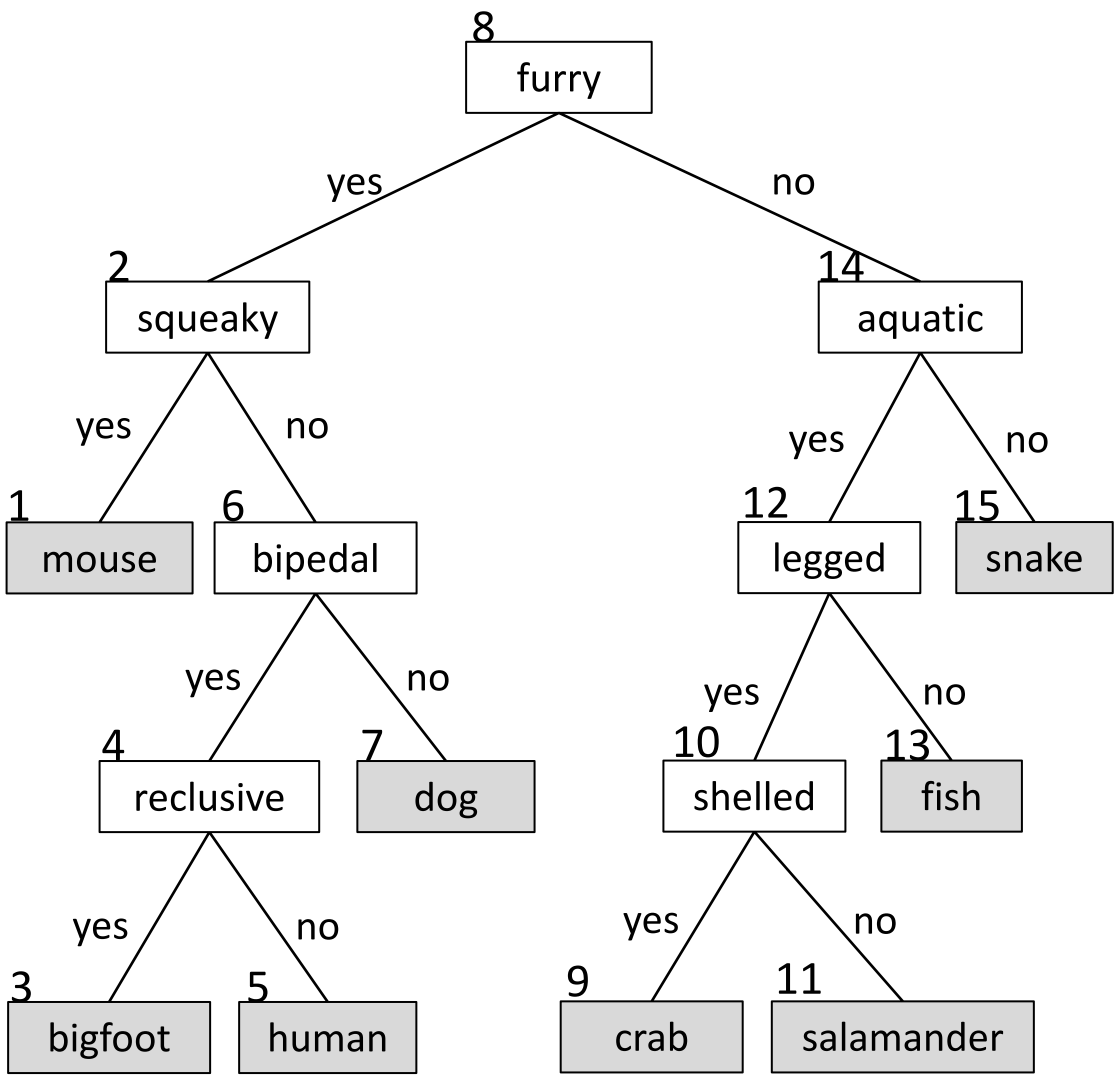
```
public class Node {
    private String text;
    private Node yesChild;
    private Node noChild;
    private Node parent;
    private int tag;
    ...
}
```

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;
    ...
}
```

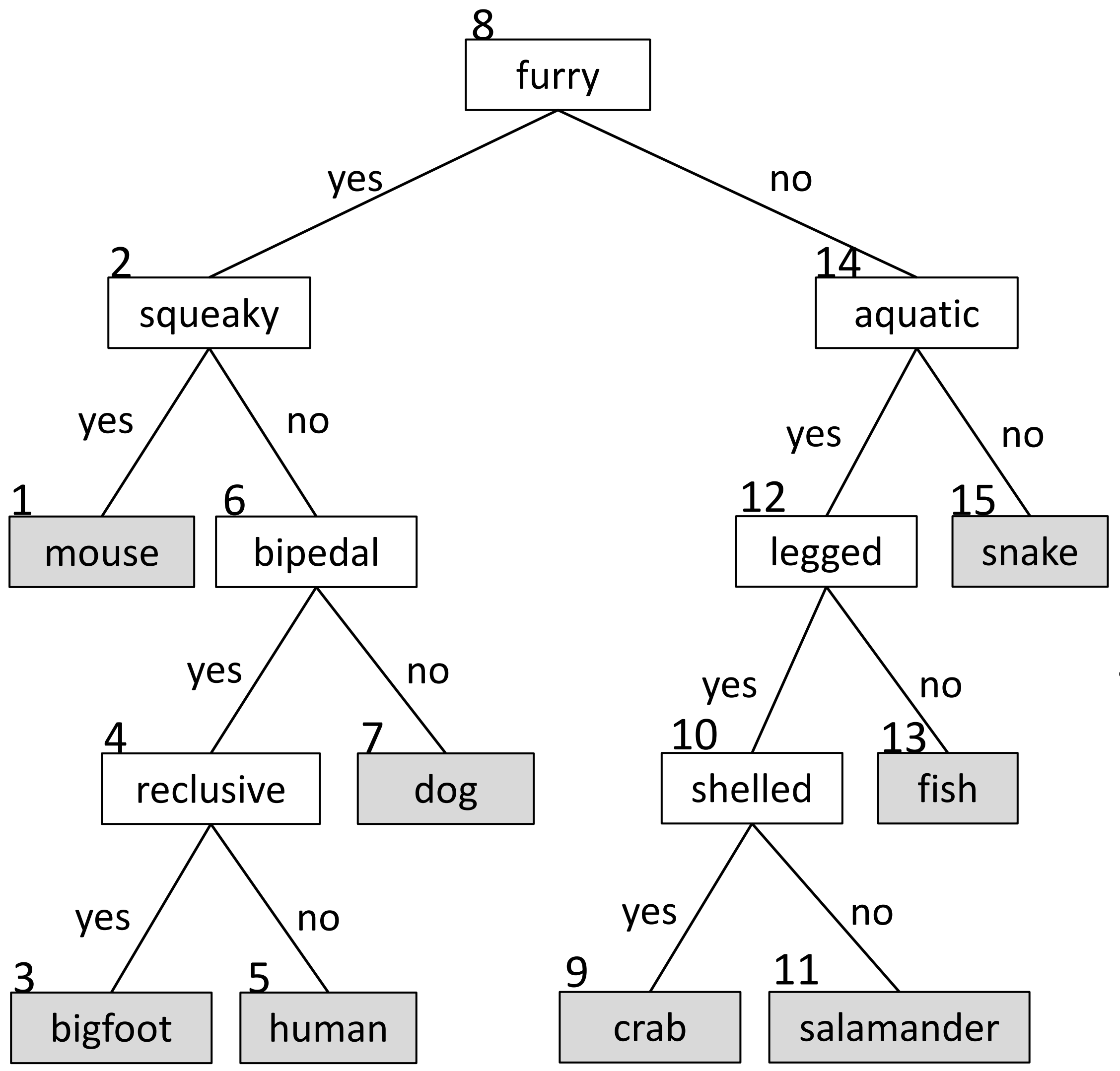
Save to file:

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



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

Save to file:

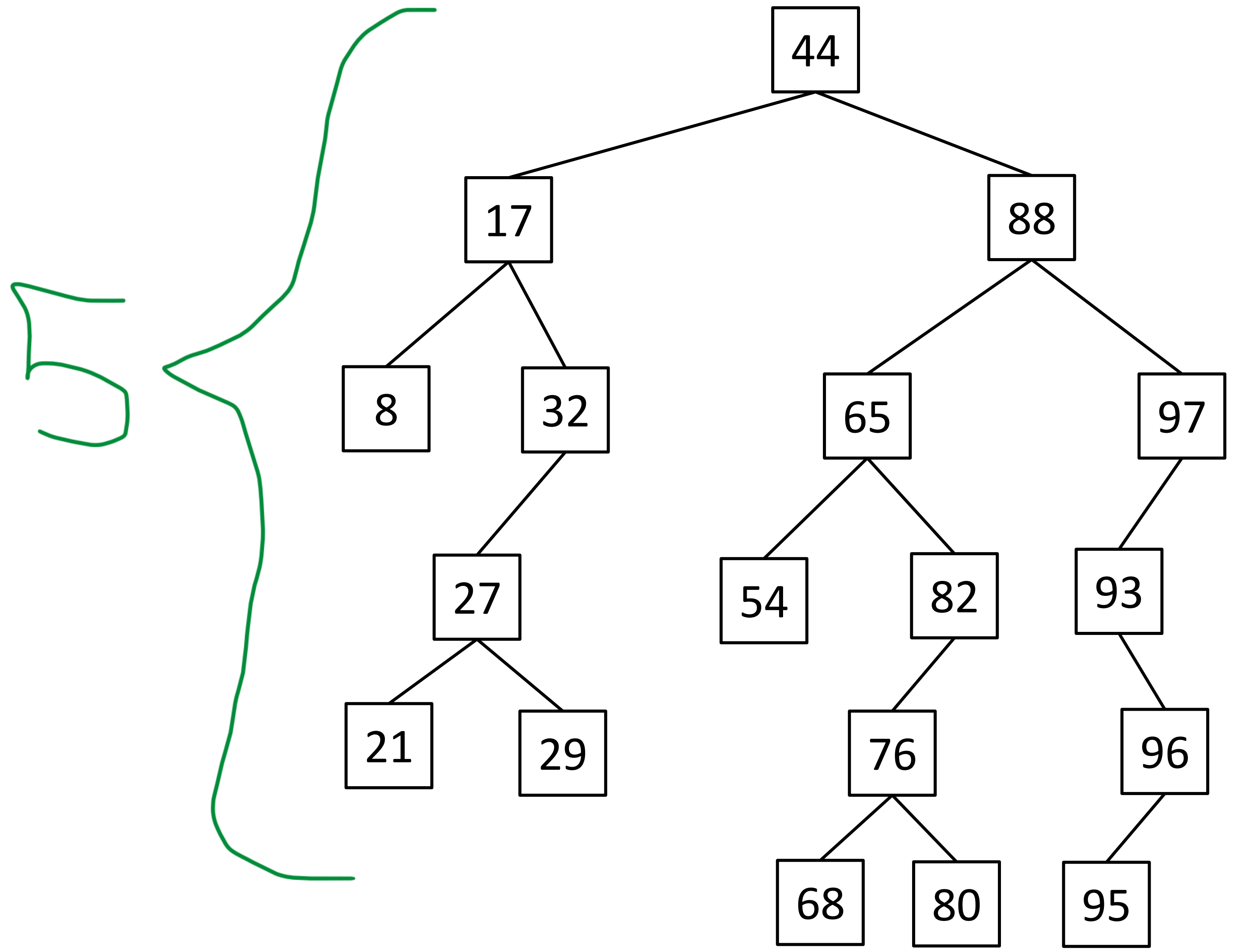
1. Do inorder traversal of tree and assign sequential integer tag values.
2. Do breadth first traversal and write tag and text values to file. E.g. 8-furry,2-squeaky,14-aquatic,1-mouse,6-bipedal,…

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.

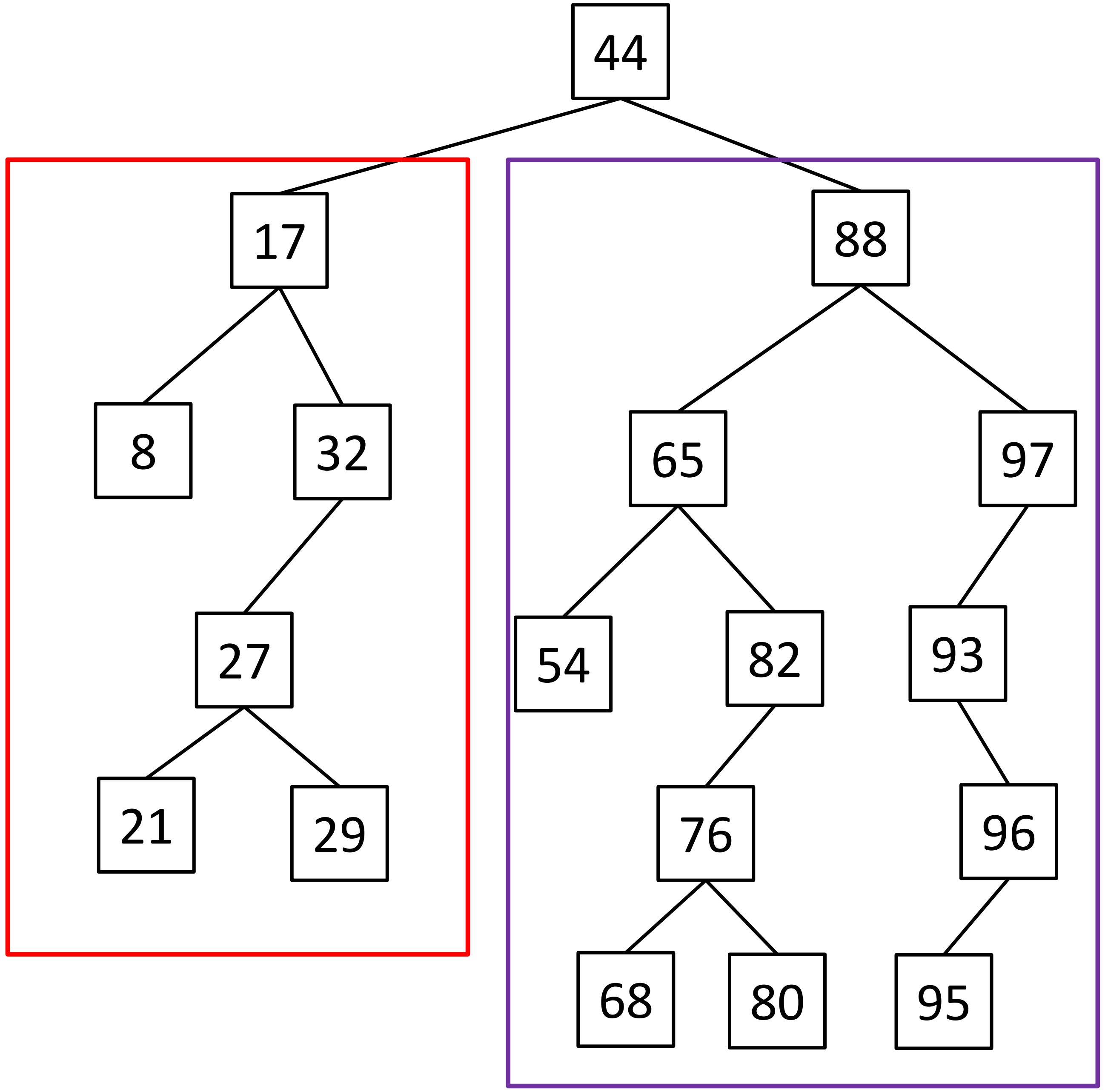
File read/writing

Find height of tree



Find height of tree

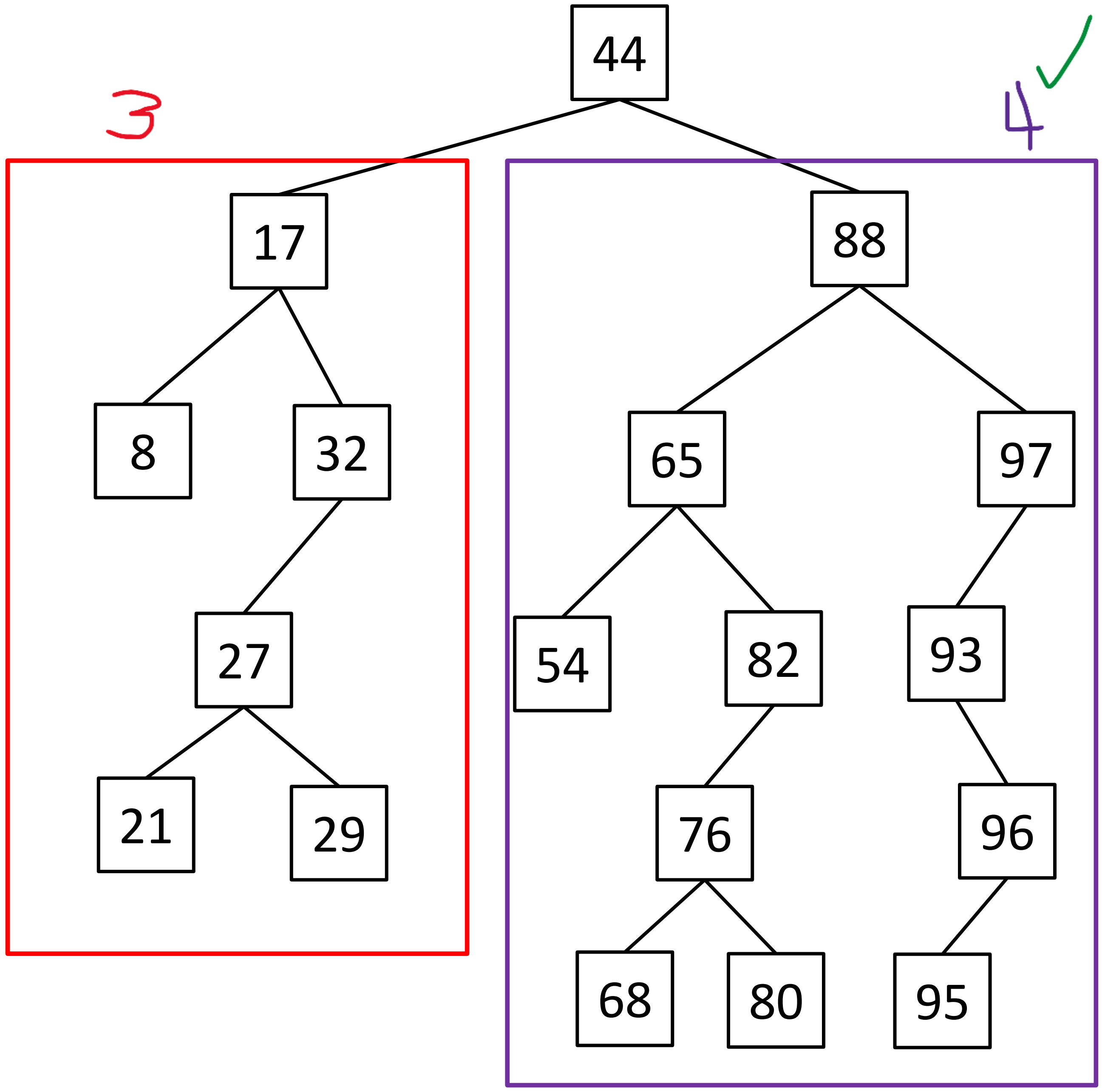
1. Recursively find height of left subtree and right subtree



Find height of tree

1. Recursively find height of left subtree and right subtree

2. Select maximum value

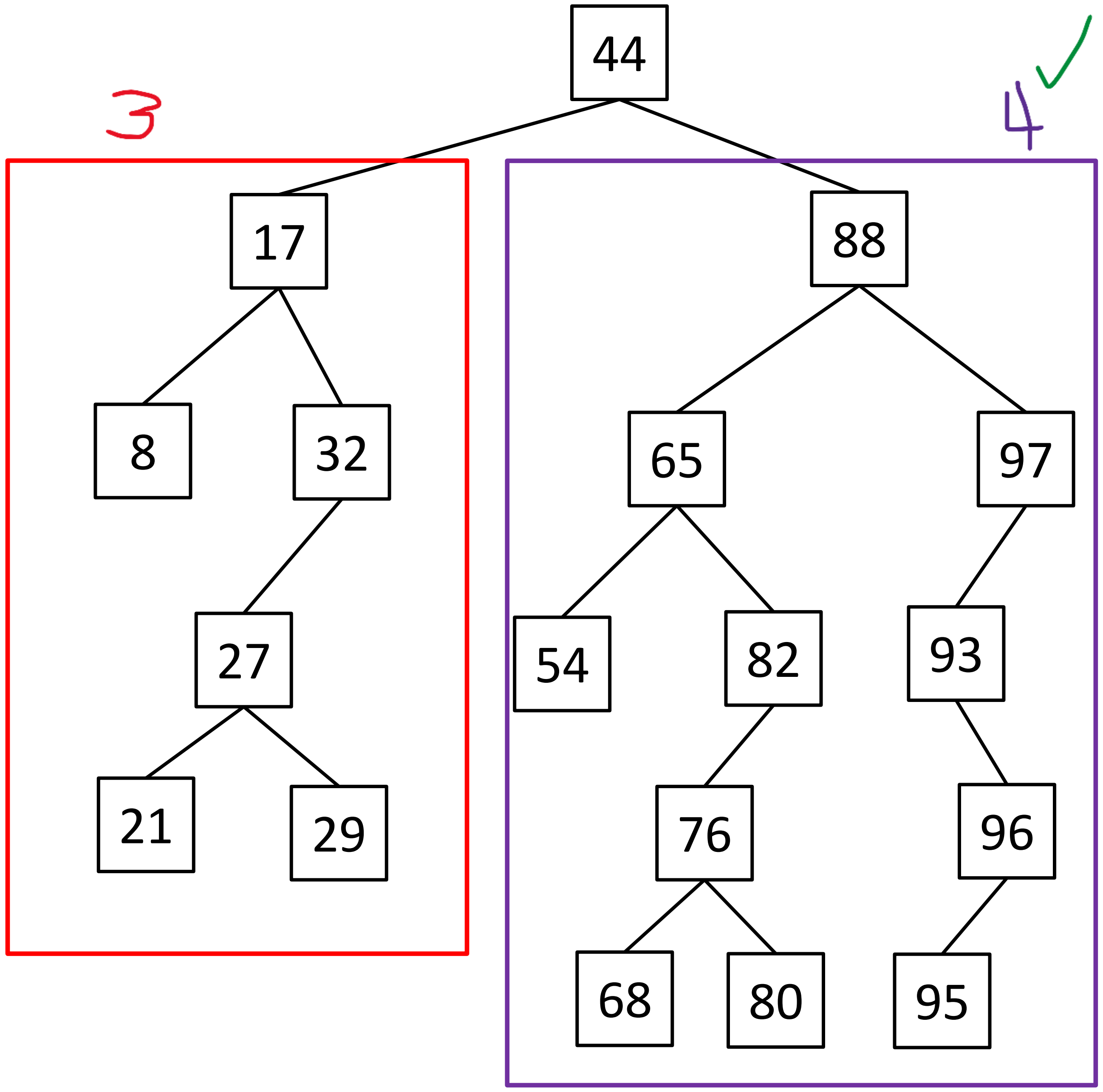


Find height of tree

1. Recursively find height of left subtree and right subtree

2. Select maximum value

3. Return 1 + maximum value



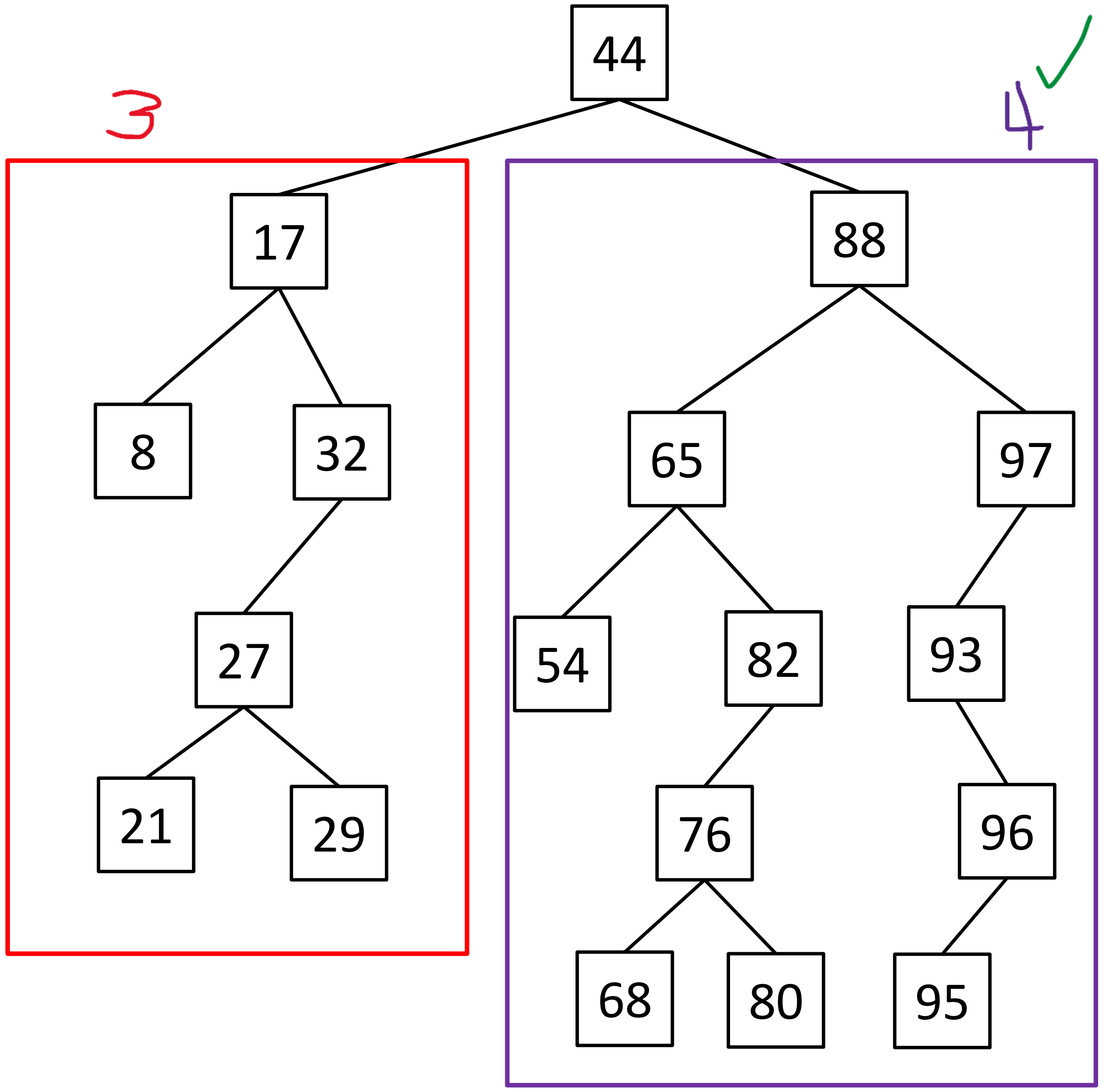
Find height of tree

1. Recursively find height of left subtree and right subtree

2. Select maximum value

3. Return 1 + maximum value

```
public int findHeight(Node current) {

   if(current == null) {
      return -1;
   }
   else {

      int rightHeight = findHeight(current.getRight());
      int leftHeight = findHeight(current.getLeft());

      return 1 + Math.max(rightHeight, leftHeight);
   }
}
```



Finding Cousins (LeetCode #993)

Nodes 54 and 93 **are** cousins because:

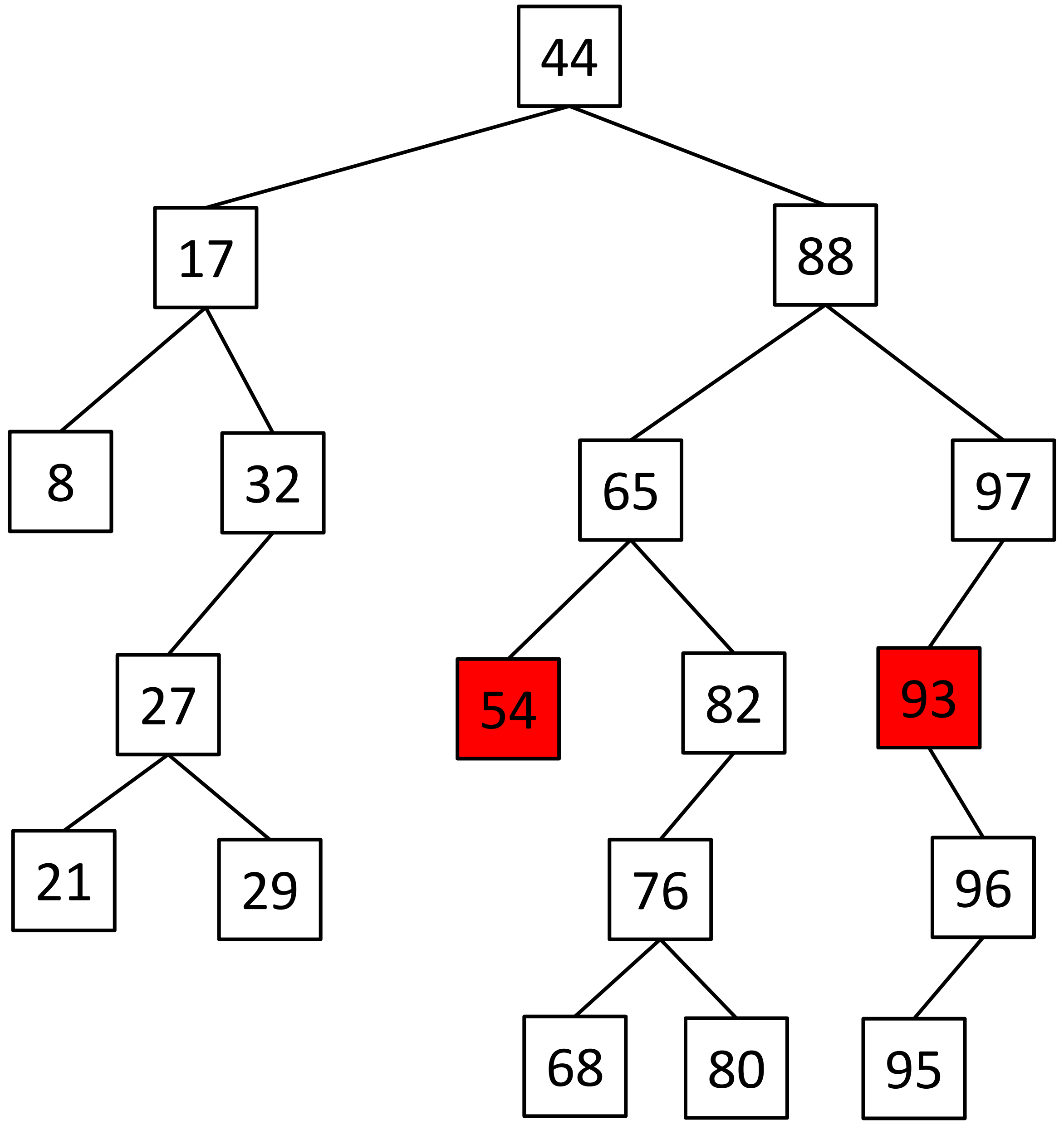
- They are at the same depth level
- They do not have the same parent

Nodes 54 and 82 **are not** cousins because:

- They have the same parent

Nodes 54 and 76 **are not** cousins because:

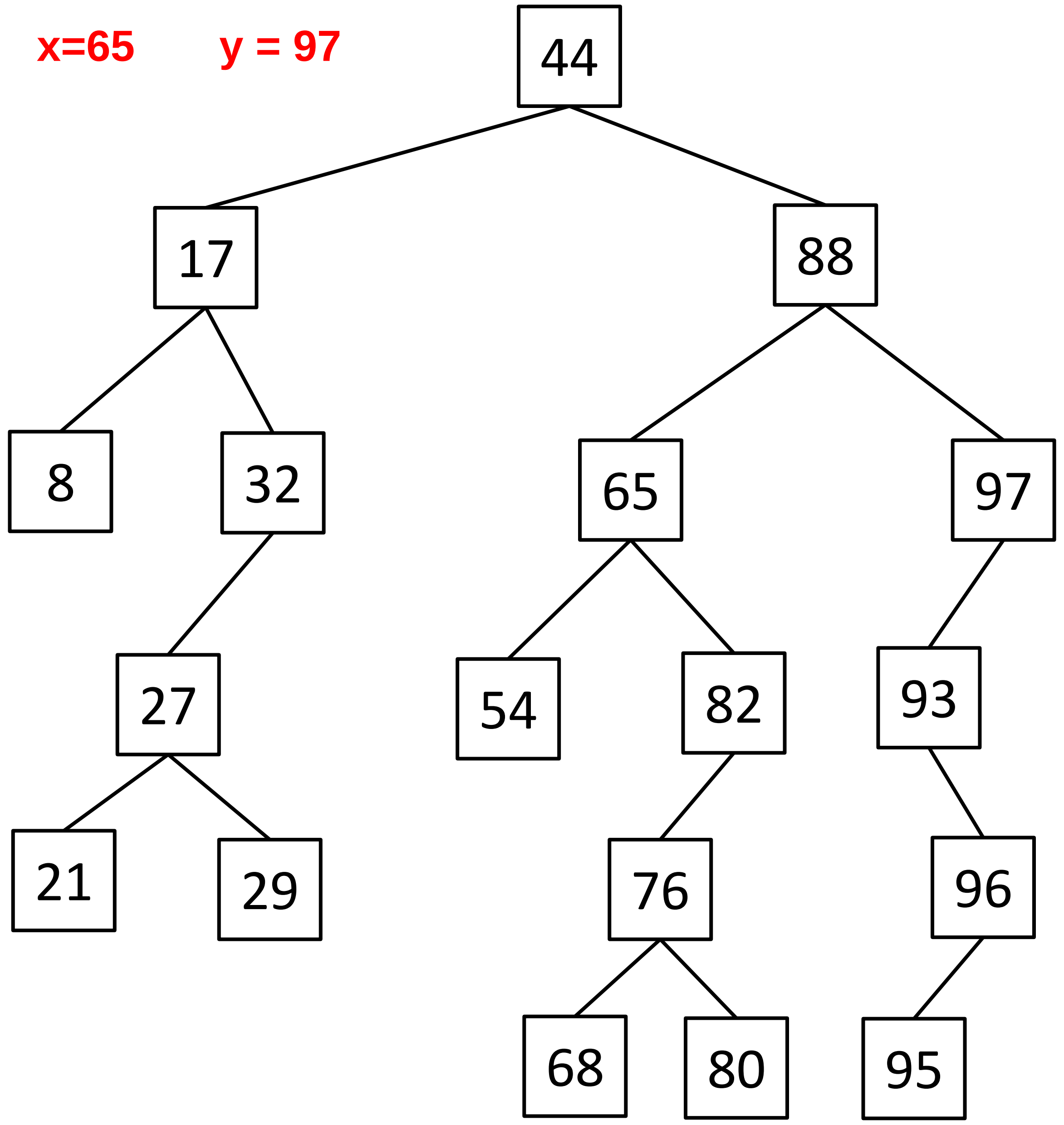
- They are on different depth levels



Finding Cousins (LeetCode #993)

1. Do breadth-first to find x and y

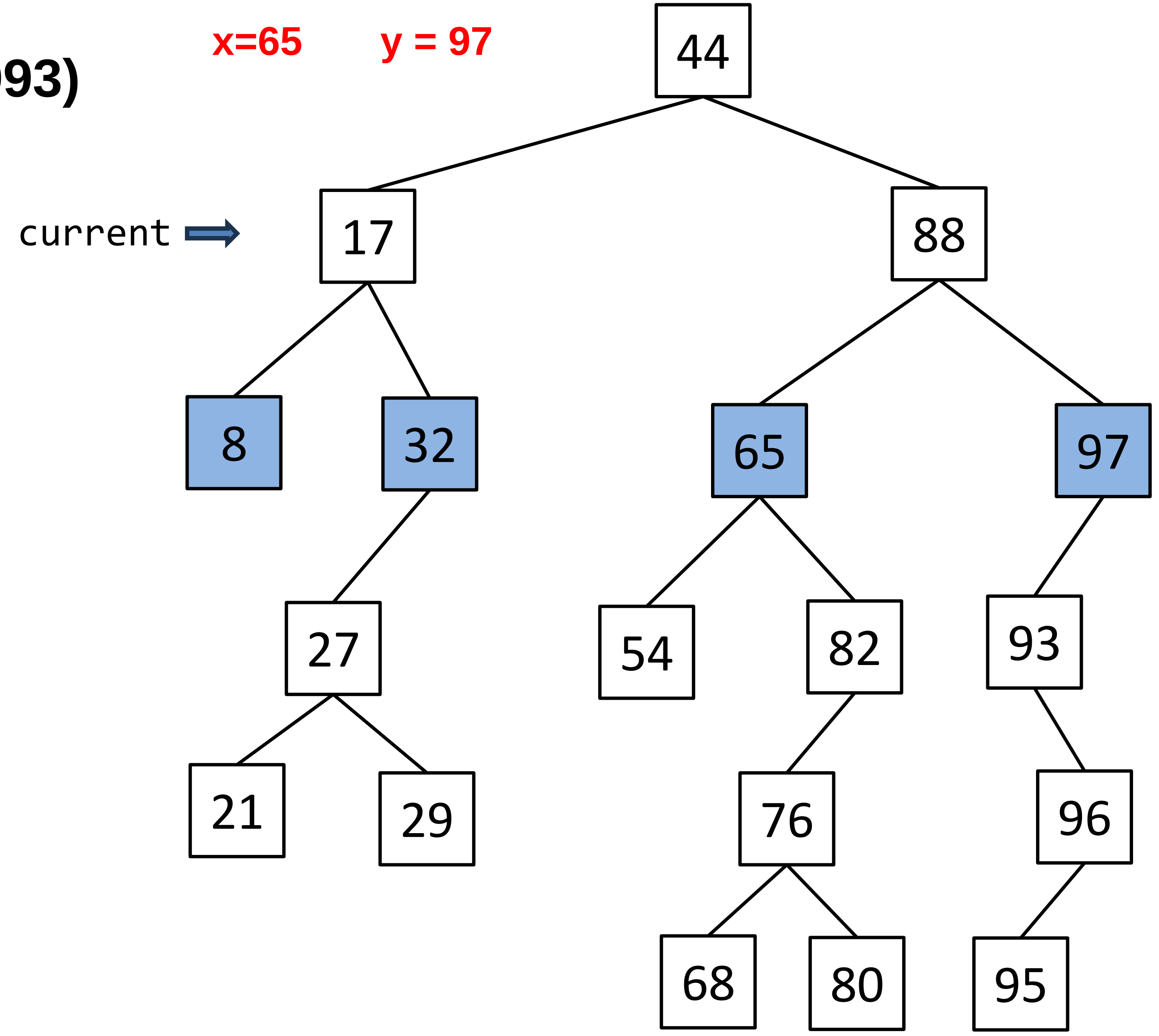
x=65 y = 97



Finding Cousins (LeetCode #993)

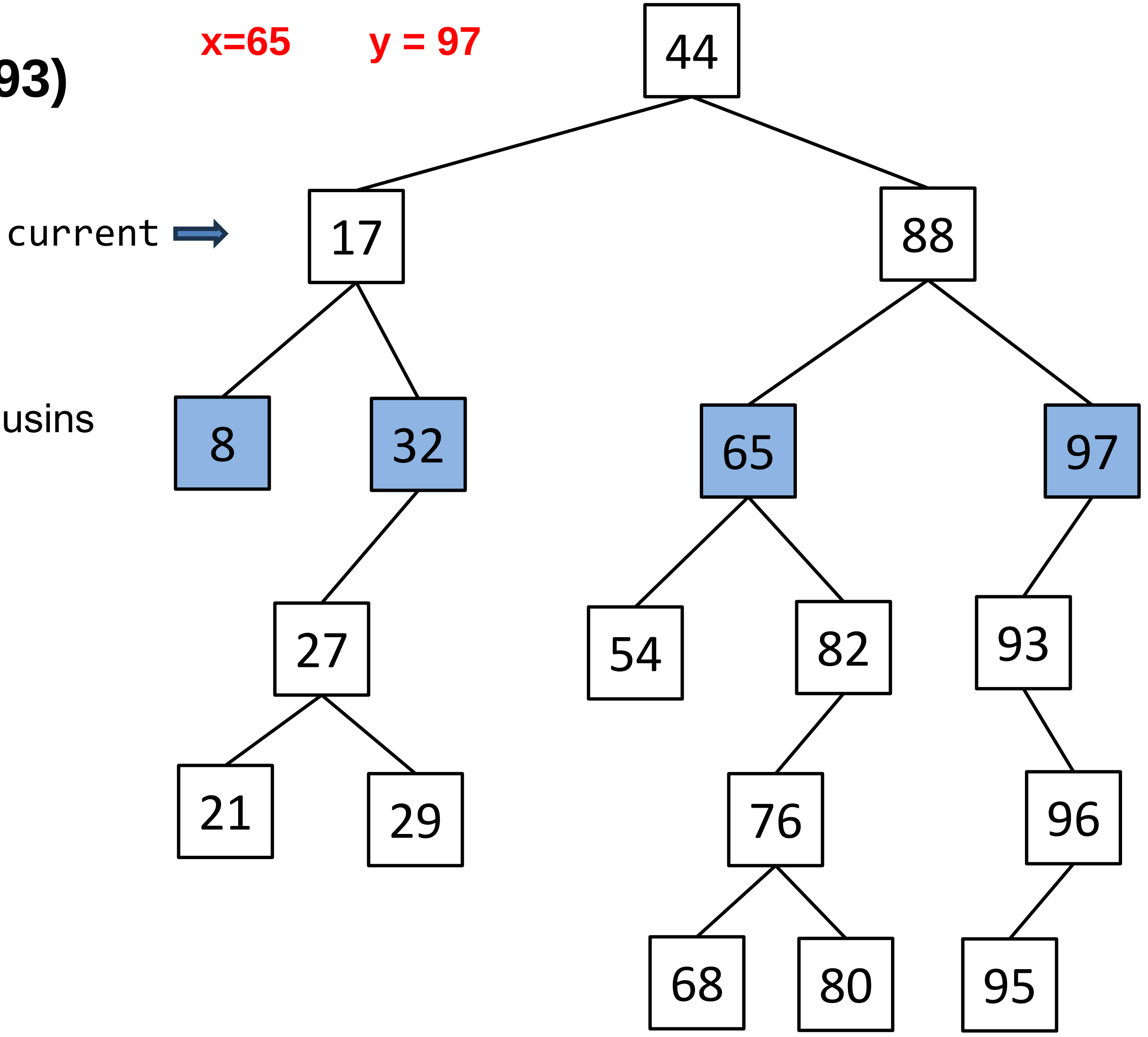
1. Do breadth-first to find x and y

2. Look at nodes in row below to find a match



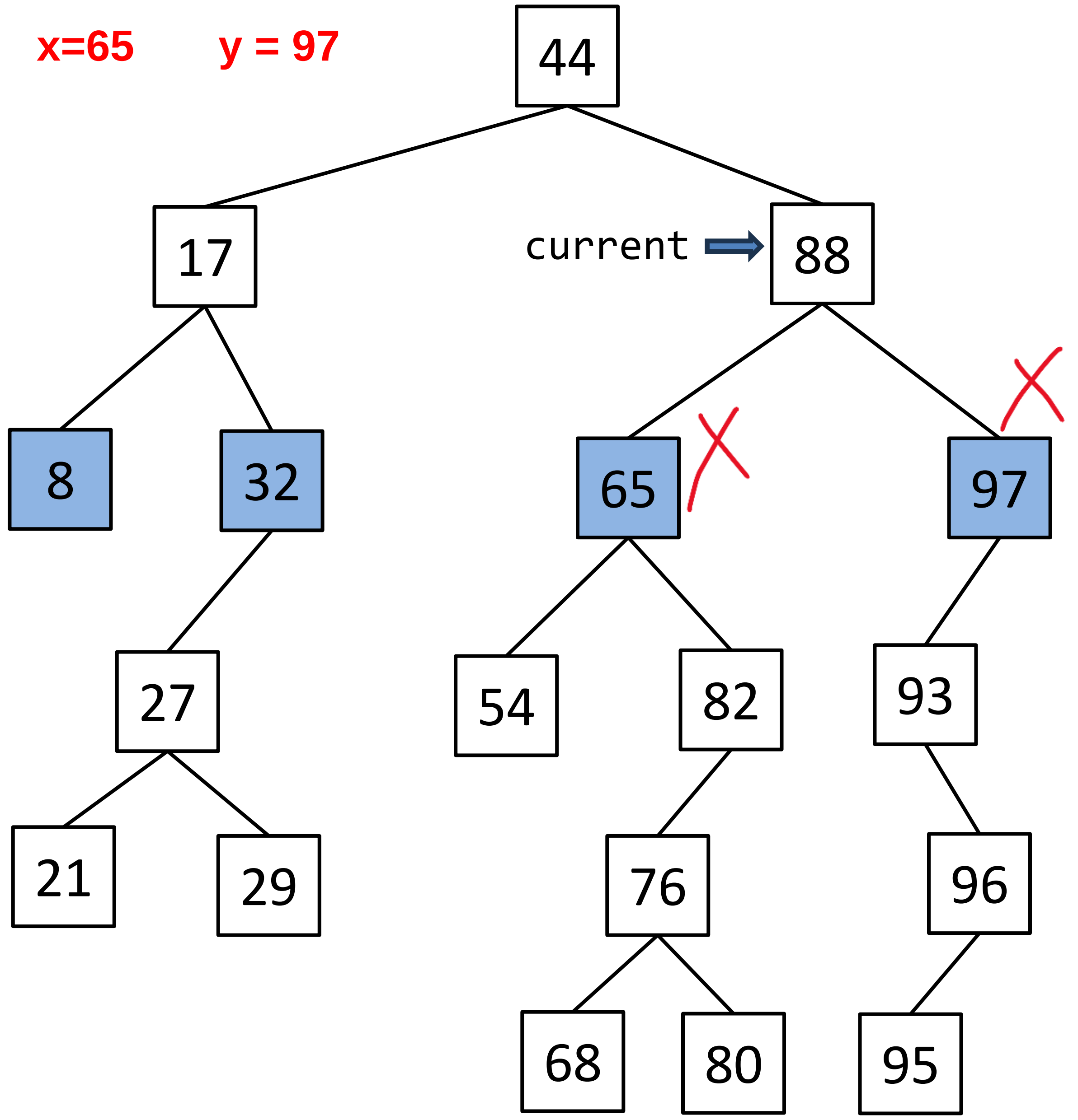
Finding Cousins (LeetCode #993)

1. Do breadth-first to find x and y
2. Look at nodes in row below to find a match
3. If current's children are x and y, they cannot be cousins



Finding Cousins (LeetCode #993)

1. Do breadth-first to find x and y
2. Look at nodes in row below to find a match
3. If current’s children are x and y, they cannot be cousins



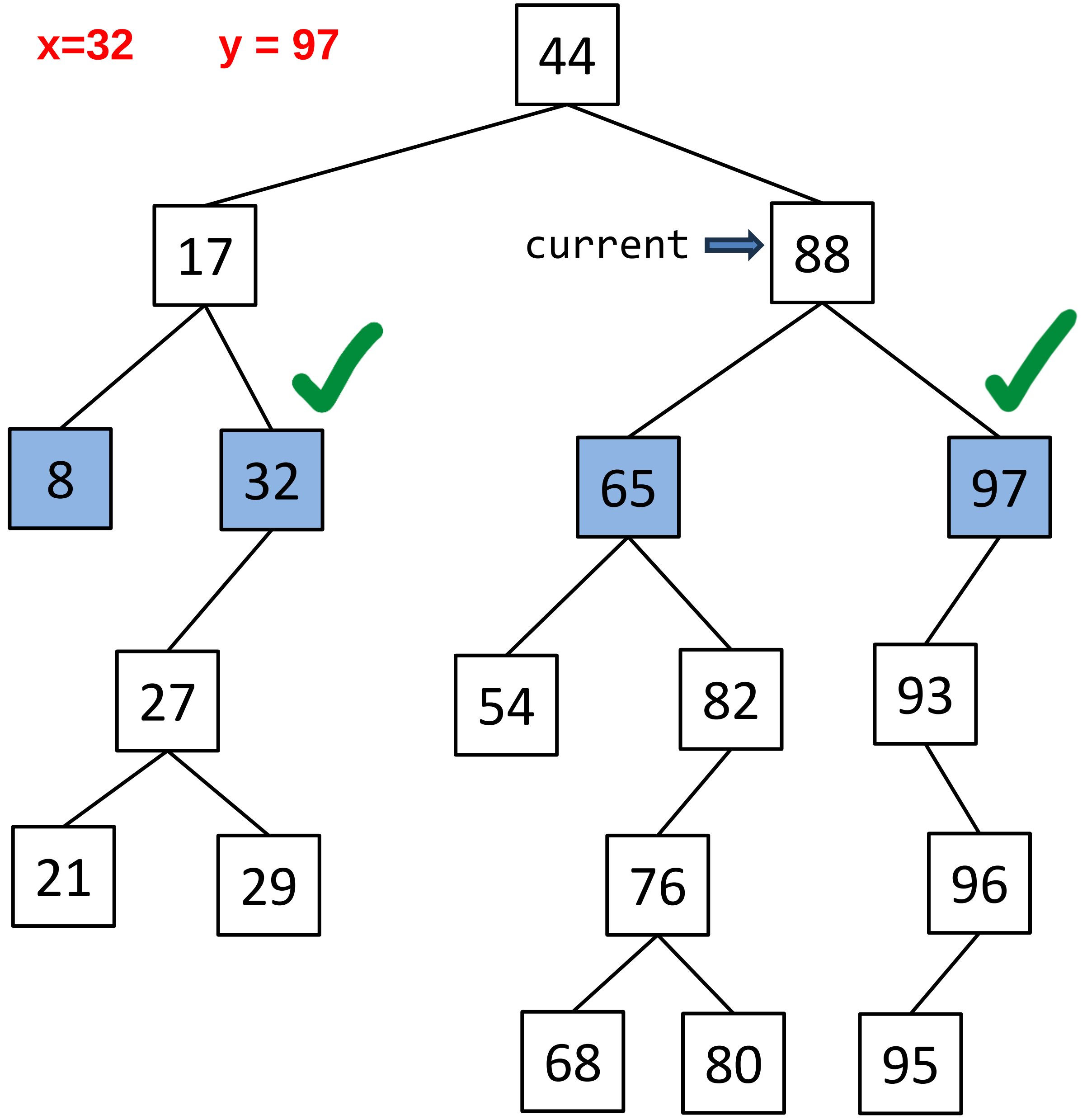
Finding Cousins (LeetCode #993)

1. Do breadth-first to find x and y

2. Look at nodes in row below to find a match

3. If current's children are x and y, they cannot be cousins

4. If nodes are found in the same row, and have different parents, return true


x=32
y = 97
current
44
17
88
8
32
65
97
27
54
82
93
21
29
76
96
68
80
95