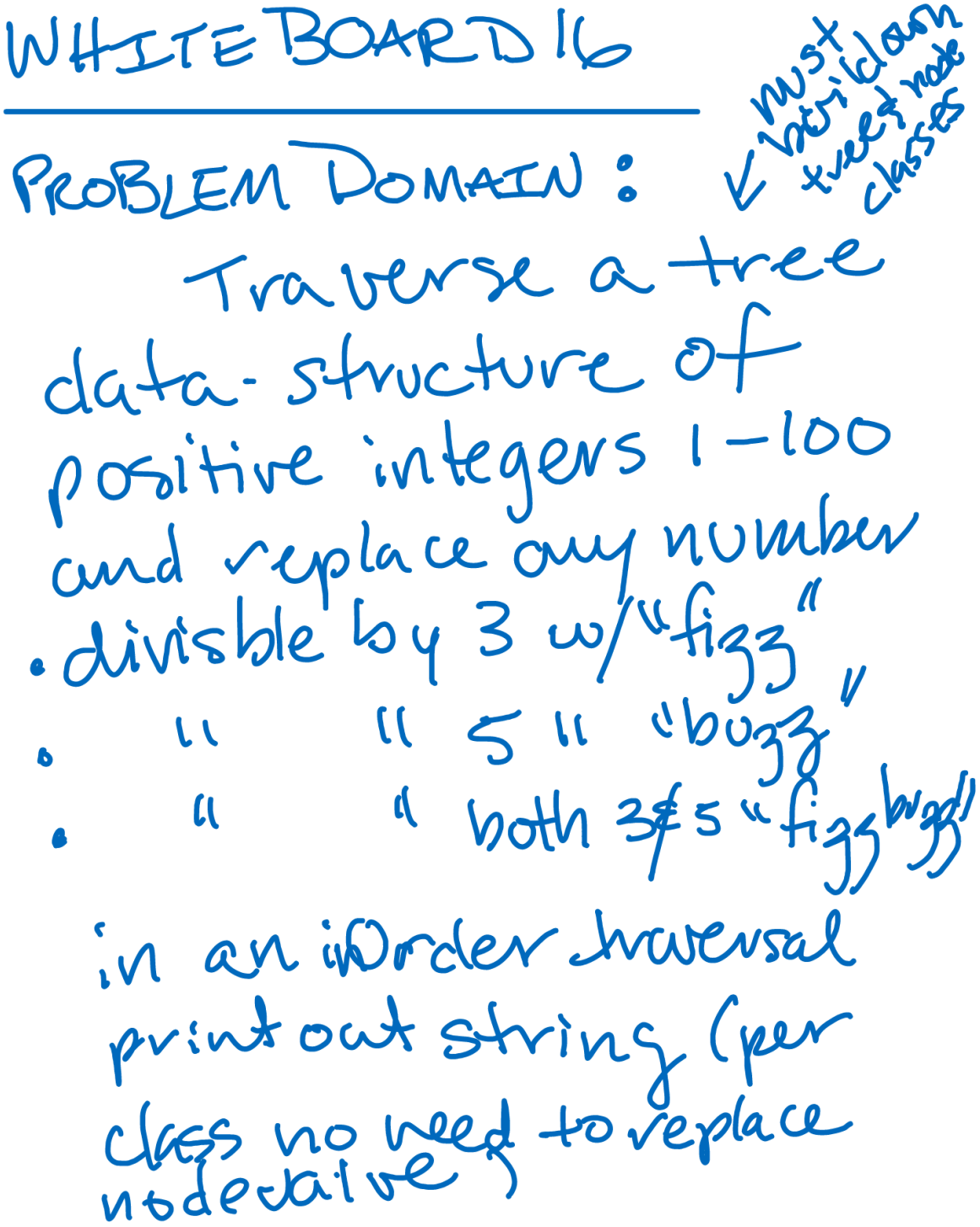
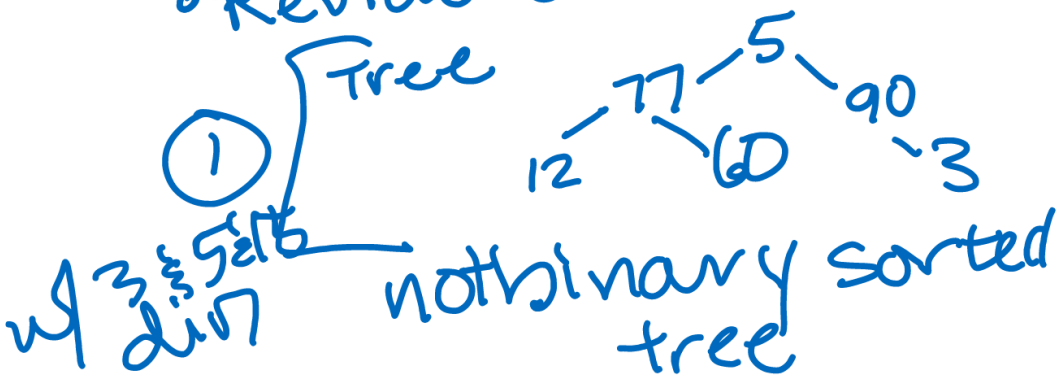
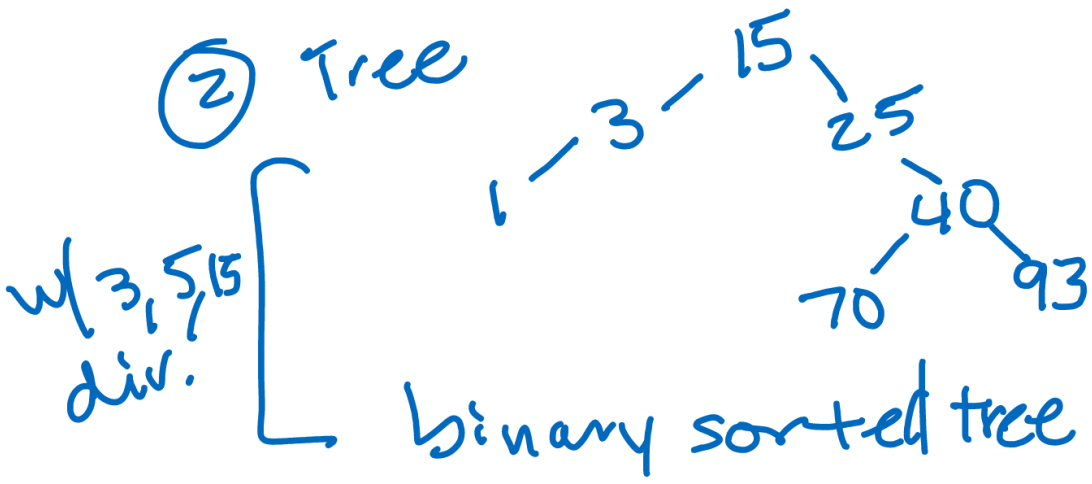
Whiteboard 16

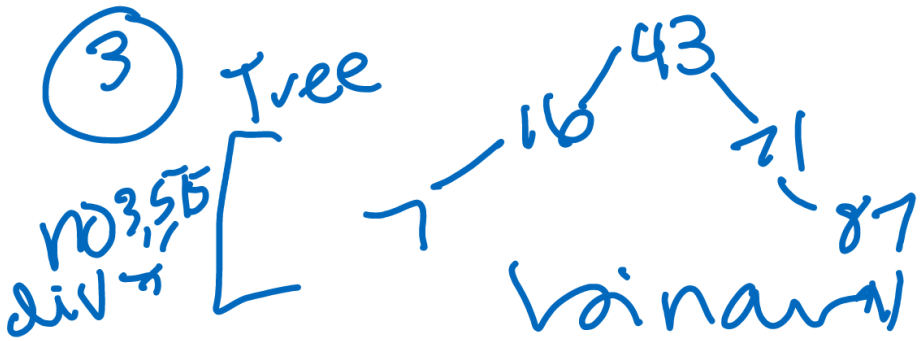


Visual

- Review test cases







(4) Edge Case empty tree

Algorithm

Build TreeNode Class
w/ root, left & right
properties values

Build Tree class
method → Build to String StringBuilder
Build inOrder Method

Build fizzbuzz method
to evalute values
in method for
↑ fizzbuzz
Return string after FizzBuzz End

Big O

no nested loops
but there are loops - while/for
so Big O^n time

not a lot of net new variables of great size to evaluate trees w/ new classes, methods,
so Big O 'space

Pseudo Code

- Build TreeNode constructor Class
- Build tree class w/ TreeNode & add properties
- Build StringBuilder method attached to Tree Class
- Build inOrder traversal method for tree class
- Build FizzBuzz evaluator to apply to String

after 'inOrder' & StringBuilder methods applied return String w/ fizzBuzz req.

Code

1st Tree Node class

```
public class TreeNode {
    protected int data;
    protected TreeNode left;
    protected TreeNde right;
```

```
package tree;
public TreeNode(int value){
    this.data = value;
}
public int getData(){
    return this.data;
}
public TreeNode getLeft()
{ return this.left; }
public TreeNode getRight
(){
    return this.right; }
public String toString(){
    return "" + this.data;
} }
```

② package tree;

public class BinaryTree {

private TreeNode root;

public void add(int value) {

if (this.root == null) {

this.root = new TreeNode(value);

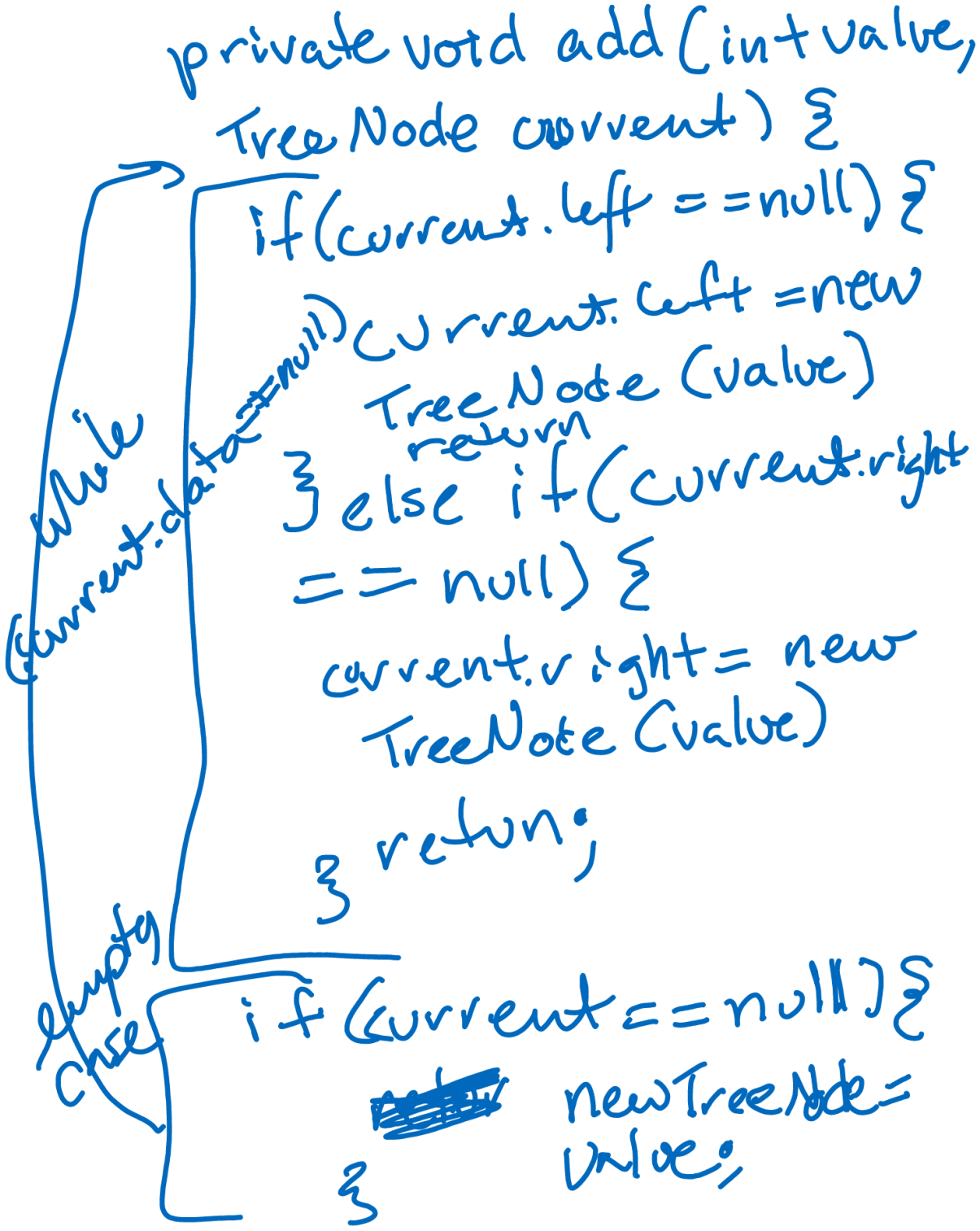
return;

}

this.add(value, this.root);

}

// recursive add method that is private & helps public add method


private void add(int value,
TreeNode current) {
if(current.left == null) {
current.left = new
TreeNode(value)
return
} else if(current.right
== null) {
current.right = new
TreeNode(value)
return;
}
while
(current.data != null)
if(current == null) {
newTreeNode =
value;
}
Empty
Case

```
public String toString(){
  StringBuilder builder=
    new StringBuilder();
  return inOrder(builder,
    this.root).toString();

}
private StringBuider inOrder
  (StringBuilder builder,
  TreeNode current) {
    if (current==null) {
      return builder;
    }
```

```
inOrder(builder, current.left);
builder.append(current.data
    + " ");
inOrder(builder, current.
    right + " ");
return builder;
}
```

(3)

```
package tree;
public class FizzBuzzTree{
```

```
public String fizzBuzzTree
(String string){
for(int i=0; i > string.
length; i++){
if(i%3==0 && i%5
== 0){
String.atChar(i)=
"fizzbuzz".
}else if(i%5 == 0){
String.atChar(i)=
"buzz".
}else if(i%3 == 0){.
```

```
            string.atChar(i)="fizz";
        } else string.atchar(i)
            = i;

        }
    }
    return String;
}
```

④ main

```
public static void main
(String[] args) {
```

```
BinaryTree bt1=
    bt1.add(6);
    bt1.add(7);
    bt1.add(11);
    bt1.add(4);
    bt1.add(22);
                stringBuilder.fizz
return      buzzTree(bt1);

}
// end program
```