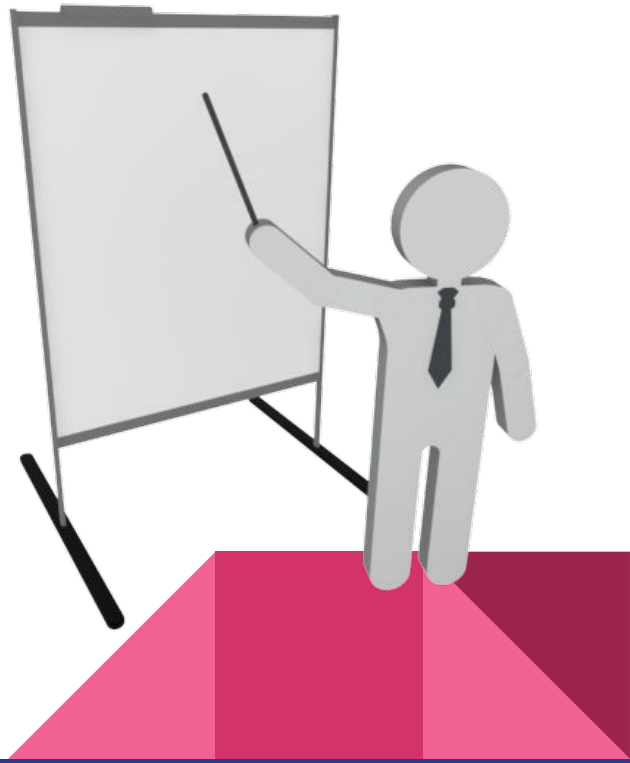
Treeviz

By Conner Caprio, Moira Pelton, Austin Zensen

Dr. Fox: CS4624 Multimedia , Hypertext, and Information Access
Virginia Tech Blacksburg VA 24061
5/2/2019

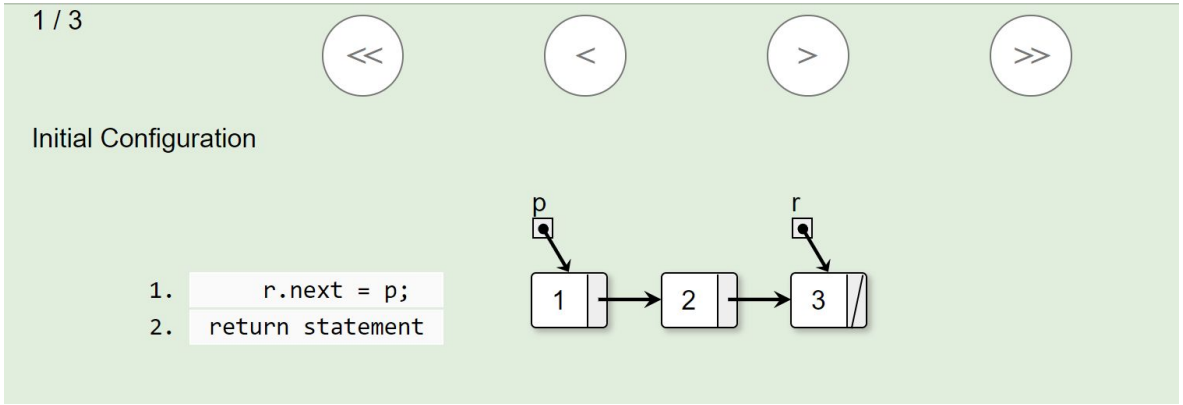
Outline

1. Project Overview Recap
2. Project Walkthrough
3. Project Completed
4. Deep Dive
5. End of Semester Timeline Comparison
6. Lessons Learned
7. Future Work
8. Acknowledgements

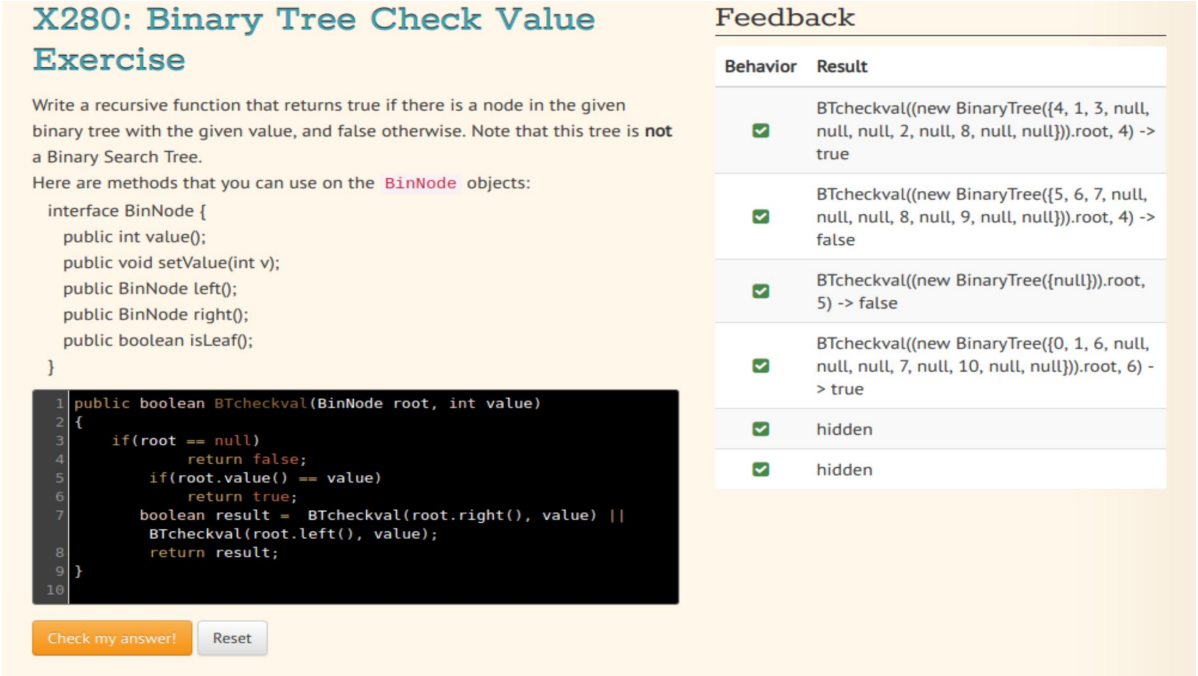


Brief Overview

- Create visualization of student code from CodeWorkout exercises to help students find bugs more easily and increase understanding of data structures



Project Walkthrough



X280: Binary Tree Check Value Exercise

Write a recursive function that returns true if there is a node in the given binary tree with the given value, and false otherwise. Note that this tree is **not** a Binary Search Tree.

Here are methods that you can use on the `BinNode` objects:

```
interface BinNode {
  public int value();
  public void setValue(int v);
  public BinNode left();
  public BinNode right();
  public boolean isLeaf();
}
```

```
1  public boolean BTcheckval(BinNode root, int value)
2  {
3      if(root == null)
4              return false;
5          if(root.value() == value)
6              return true;
7         boolean result =  BTcheckval(root.right(), value) ||
           BTcheckval(root.left(), value);
8          return result;
9  }
10
```

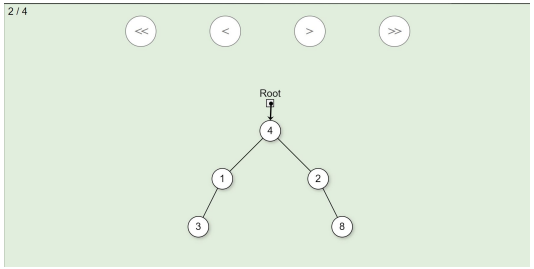
Check my answer! Reset

Feedback

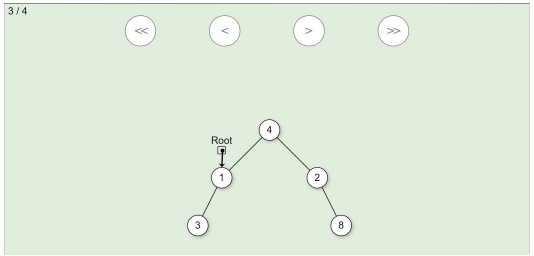
Behavior	Result
✓	BTcheckval((new BinaryTree({4, 1, 3, null, null, null, 2, null, 8, null, null})).root, 4) -> true
✓	BTcheckval((new BinaryTree({5, 6, 7, null, null, null, 8, null, 9, null, null})).root, 4) -> false
✓	BTcheckval((new BinaryTree({null})).root, 5) -> false
✓	BTcheckval((new BinaryTree({0, 1, 6, null, null, null, 7, null, 10, null, null})).root, 6) -> true
✓	hidden
✓	hidden

https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/BinaryTreeInfFlw.html

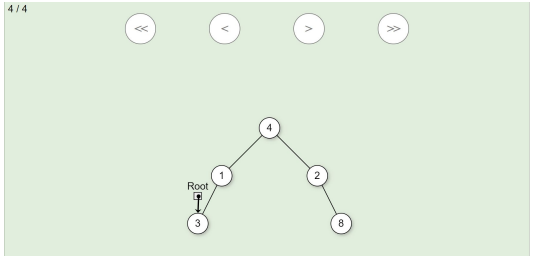
Project Walkthrough (cont.)



Pointing to root node, value of 4



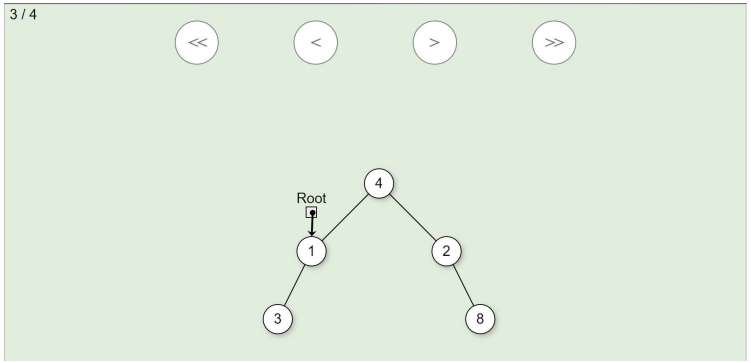
Pointing to first left child, value 1

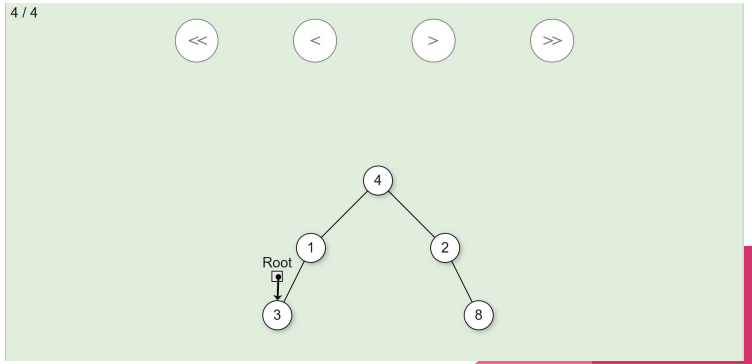


Pointing to second left child, value 3

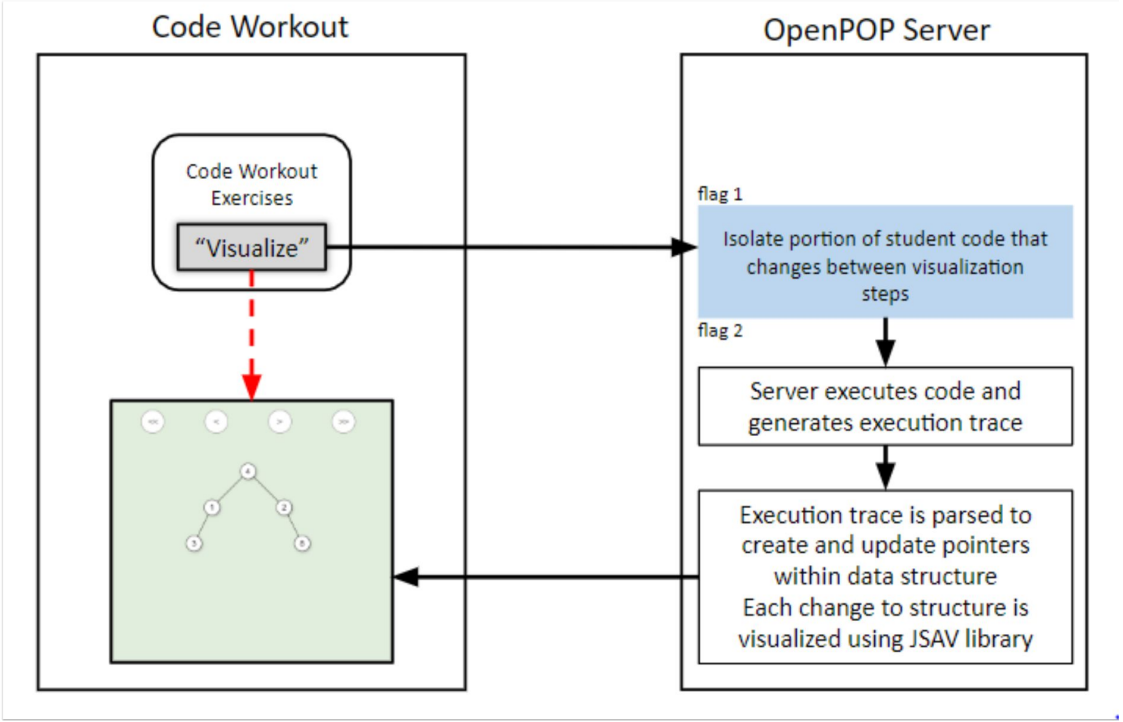
Completed Project Work

- Implemented binary tree visualizations
  - All code is compatible with existing code
- Used JSAV library to visualize binary tree
- Included JSAV pointers to show recursive value checking





Quick Deep Dive



System Diagram of interaction between CodeWorkout and OpenPOP Server (OPEN Pointer OPerations)

"heap":{"172":["INSTANCE","BinNode",["elem",4],["left",["REF",173]],["right",["REF",175]]],

```
for (var key in trace[step].heap) { //this for loop goes through each node in a single heap / tree
    //this grabs all the data for a single node and notes is there are any changes to the tree to add a jsav step
    var data = trace[step].heap[key][2][1];
    var leftRef = trace[step].heap[key][3][1];
    var rightRef = trace[step].heap[key][4][1];
    if (leftRef != null) {
        leftRef = leftRef[1];
    }
    if (rightRef != null) {
        rightRef = rightRef[1];
    }
    if (nodeIDs.includes(key)) { //this is if the node already exists
        let node = nodes.find(node => node.nodeReference == key);
        if (node.getLeft() != leftRef) {
            node.setLeft(leftRef);
            valueChanged = true;
        }
        if (node.getRight() != rightRef) {
            node.setRight(rightRef);
            valueChanged = true;
        }
        if (node.getData() != data) {
            node.setData(data);
            valueChanged = true;
        }
    }
    else { //this is if it is a new node and adds it to the array of nodes
        let treeNode = new TreeNode(data, key, leftRef , rightRef);
        nodes.push(treeNode);
        nodeIDs.push(key);
        if (first) {
            roots = treeNode;
            first = false;
        }
        valueChanged = true;
    }
```

Grabs all data for a single node

Checks if any values have changed if the node already existed

Adds the node to an array of nodes if it doesn't already exist

```
//connecting nodes with real pointers
for(let node of nodes) {
    var leftNum = node.getLeft();
    var rightNum = node.getRight();
    if (node.getReference() == rootPointerID) {
        node.setIsRoot(true);
    }
    else {
        node.setIsRoot(false);
    }
    if (leftNum == null && rightNum == null) {
        continue; //skips looking for pointers if both child are should be null
    }
    for (let nodeTwo of nodes) {
        if (leftNum != null && nodeTwo.getReference() == leftNum) {
            node.setLeftNode(nodeTwo);
        }
        if (rightNum != null && nodeTwo.getReference() == rightNum) {
            node.setRightNode(nodeTwo);
        }
    }
}
```

Goes through all nodes and sets their child nodes with pointers to tree node objects

```
if (firstPointer) { //this is making the jsav tree but only the first time
    firstPointer = false;
    bt.root(roots.getData());
    bt.layout();
    pointer = jsav.pointer("Root", bt.root(), {top: -20, arrowAnchor: "center top"});
}
//given something has changed on this iteration this adds a jsav step and visualizes it recurssively
if (valueChanged) {
    recursiveJsavVisualize(roots, bt.root(), jsav, pointer);
    bt.layout();
    jsav.step();
}
```

Creates JSAV binary tree, pointer to the root, and sets first visualization step

```
//this method creates the jsav tree recurssively and also sets the pointer to the correct node
function recursiveJsavVisualize(node, btnode, jsav, pointer) {
    var leftNum = node.getLeft();
    var rightNum = node.getRight();
    if (node.getIsRoot() == true) {
        pointer.target(btnode);
    }
 if (leftNum != null) {
    var newBtNode = btnode.left(node.getLeftNode().getData());
    recursiveJsavVisualize(node.getLeftNode(), newBtNode, jsav, pointer);
 }
 if (rightNum != null) {
     var newBtNode = btnode.right(node.getRightNode().getData());
     recursiveJsavVisualize(node.getRightNode(), newBtNode, jsav, pointer);
 }
 return;
}
```

Makes the tree recursively for the JSAV object

Project Timeline

- Create tree from trace ✓
- Write code for basic tree visualization ✓
- Visualize multiple steps of students code ✓
- User Testing ✗

Lessons Learned

- Constant communication with client will help project stay on deadline
- Having Windows vs. Linux made the project more difficult
- The JSAV api is very easy to work with when visualizing data structures
- Analyzing binary tree data from a stack trace introduces timing errors when creating nodes

Future Work

- Implement project work in all OpenDSA Tree exercises
- Expand project to include other data structures
  - Hash table
  - M-nary tree
- Optimize current code

Acknowledgements

Mostafa Mohammed

PHD Student @ VT

Dr. Cliff Shaffer

CS Professor @ VT

OpenDSA

CodeWorkout

Questions?