EECS 280
Discussion #4

Week of January 28

Outline

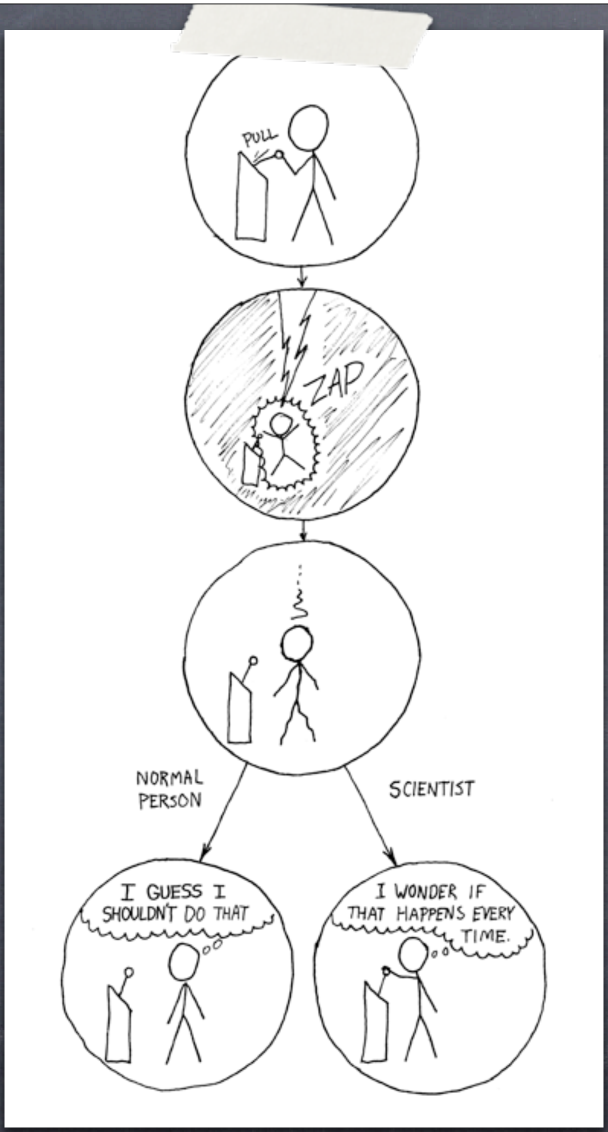
- **Administrivia**
- Testing
- Binary Trees

Administrivia

- Assignment #2
  - Due Thursday @ 11:59 PM
  - Submission Open
  - Test thoroughly, we will
- Assignment #3
  - Out late this week

Outline

- Administrivia
- **Testing**
  - Motivation
  - Process
- Binary Trees


PULL
ZAP
NORMAL PERSON
SCIENTIST
I GUESS I SHOULDN'T DO THAT
I WONDER IF THAT HAPPENS EVERY TIME.

Motivating Humor

Testing

- Motivating Question:
  - Does my program do what it’s supposed to do?
- How would you answer this question?

Testing Example

```
float get_slope(int x1, int y1, int x2, int y2)
// EFFECT: returns the slope of the line defined by
//         points (x1, y1) and (x2, y2)
{
    // rise over run!
    return ( ( y2 - y1 ) / ( x2 - x1 ) );
}
```

Is this function correct?

What happens in the case of a vertical line? Is this correct?

What is Correctness?

- Formal method: compare implementation with specification
  - requires formal (read: mathematical) description of specification and implementation
  - time consuming, complicated, etc.
  - Quis custodiet ipsos custodes?
- Empirical method: testing

The Testing Process

- Develop “testable” code
  - Function decomposition
  - Unit testing, drivers
  - Stubs
- Develop “representative” tests
- Apply tests, evaluate code, rinse and repeat

Function Decomposition

- When faced with a complex problem, break code into reasonably sized “chunks” that lend themselves well to individual testing
  - Avoid “god” functions/classes/programs
  - Single purpose code!

Unit Testing, Drivers

- With well decomposed code, you can write new functions/programs whose sole purpose is to test other functions
  - Unit testing: test a single function
  - Integration testing: test interaction between functions
- A driver provides an automated, isolated environment for running test code

Driver Example

```
int main()
{
    // test simple line
    float result = get_slope( 1, 1, 2, 2 );
    cout << “slope from (1,1) to (2,2) is “ << result;
    cout << “ and should be 1” << endl;

    // test complex line
    ...
}
```

Stubs

- Stub: dummy procedure, module, or unit
  - Display a trace message
  - Display a parameter value
  - Return a value from a table
  - Return table value selected by parameter
- Useful for visualizing flow, tracking bugs

Stub Example

```
float get_slope(int x1, int y1, int x2, int y2)
// EFFECT: returns the slope of the line defined by
//         points (x1, y1) and (x2, y2)
{
    // stub data
    cout << “enter get_slope: (“;
    cout << x1 << “, “ << y1 << “), (“;
    cout << x2 << “, “ << y2 << “)” << endl;

    // rise over run!
    return ( ( y2 - y1 ) / ( x2 - x1 ) );
}
```

Exhaustive Testing

Occasionally we can exhaustively test all possible inputs to a function:

```
string get_month_name( int month_number )
{
    if ( month_number == 1 )
        return “January”;
    else if ...
}
```

Representative Tests

- Primarily we need to choose a set of test inputs to convince ourselves of the correctness of our code, given time/ financial/computational constraints upon us
- This choice may depend upon whether we know how the specification is implemented
  - Black box = code unknown
  - White box = code known

Black Box Example

- Given that we only know the specification of `get_slope`, what set of tests would you run?
  - Positive slope
  - Negative slope
  - Horizontal line
  - Vertical line

White Box Example

Consider the following code, what additional tests might you run given this knowledge:

```
float get_slope(int x1, int y1, int x2, int y2)
{
    int y_diff = ( y2 - y1 );
    int x_diff = ( x2 - x1 );
    int ratio = ( y_diff / x_diff );

    return ratio;
}
```

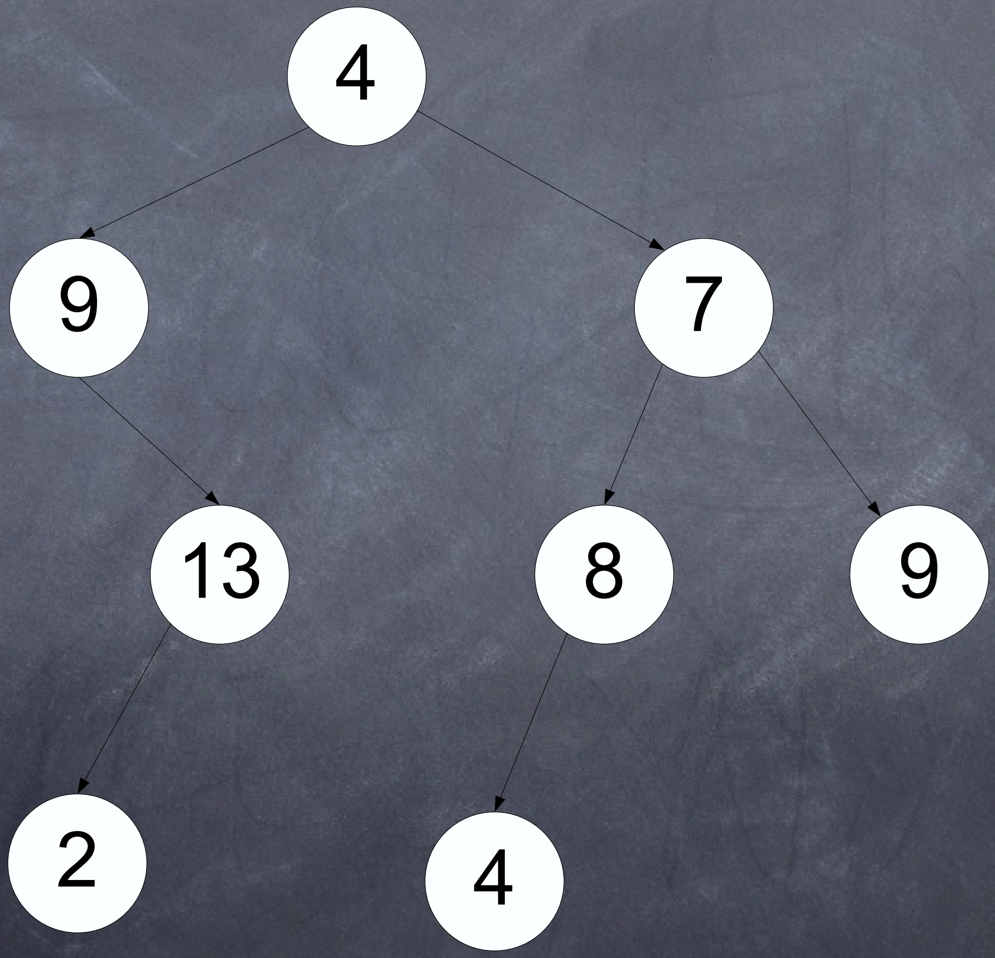
Outline

- Administrivia
- Testing
- **Binary Trees**
    - Terminology
    - Traversal
    - Height

Binary Trees Terminology

- Tree
  - a graph (consisting of nodes and edges) that is connected and acyclic
- Binary Tree
  - a directed tree where each node has at most two children
- Leaf
  - a node in a tree that has no children

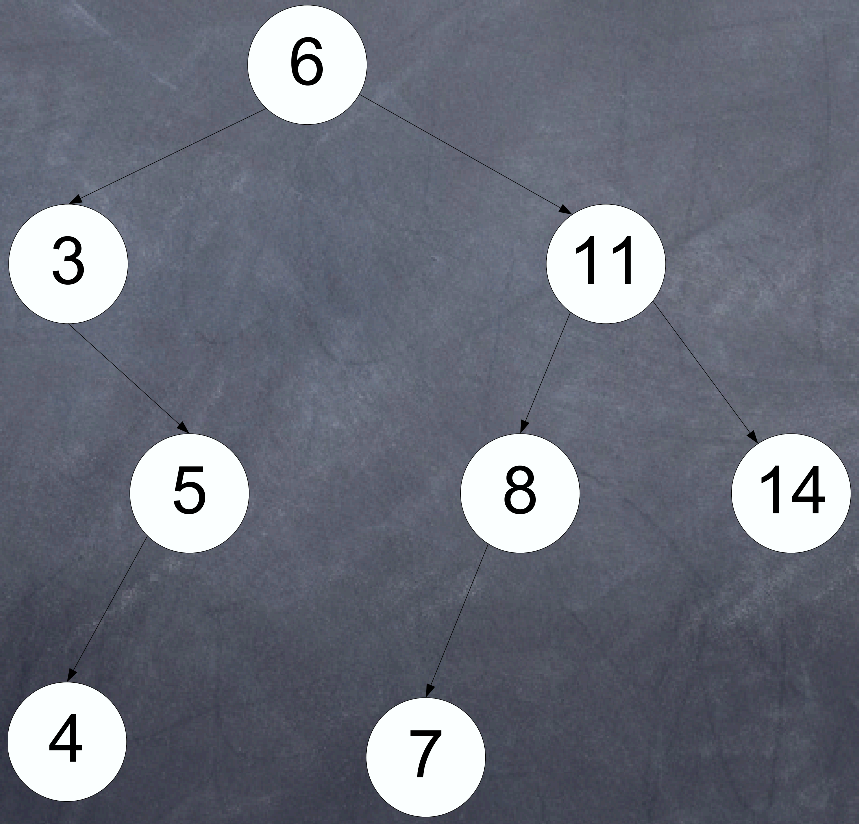
Binary Tree Example



Sorted Binary Trees

- Binary Tree where
  - left subtree is a sorted binary tree and all elements are strictly less than the root
  - right subtree is a sorted binary tree and all elements are greater than or equal to the root

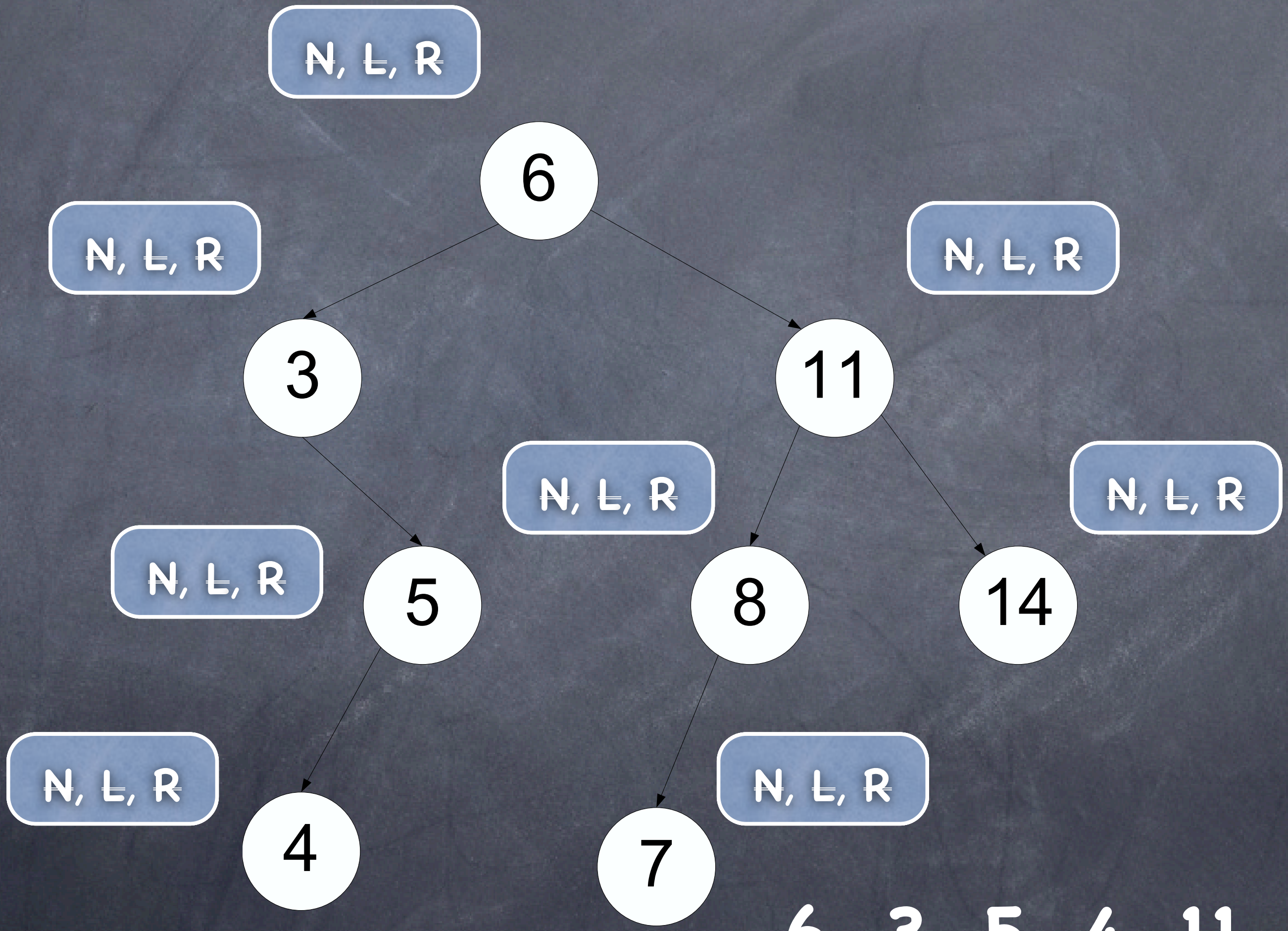
Sorted Binary Tree Example



Binary Tree Traversal

- Traversal
  - the process of visiting each node in a tree structure, exactly once, in a systematic way
- Types of traversal
  - Preorder: **node**, left, right
  - Inorder: left, **node**, right
  - Postorder: left, right, **node**

Preorder Traversal

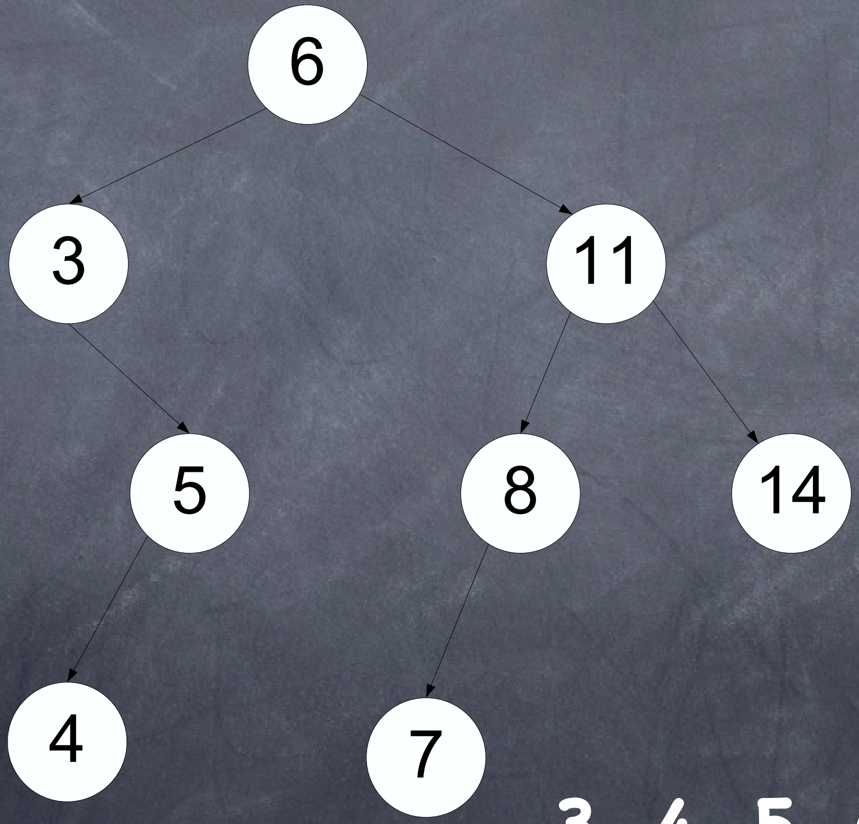


6, 3, 5, 4, 11, 8, 7, 14

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

Inorder Traversal

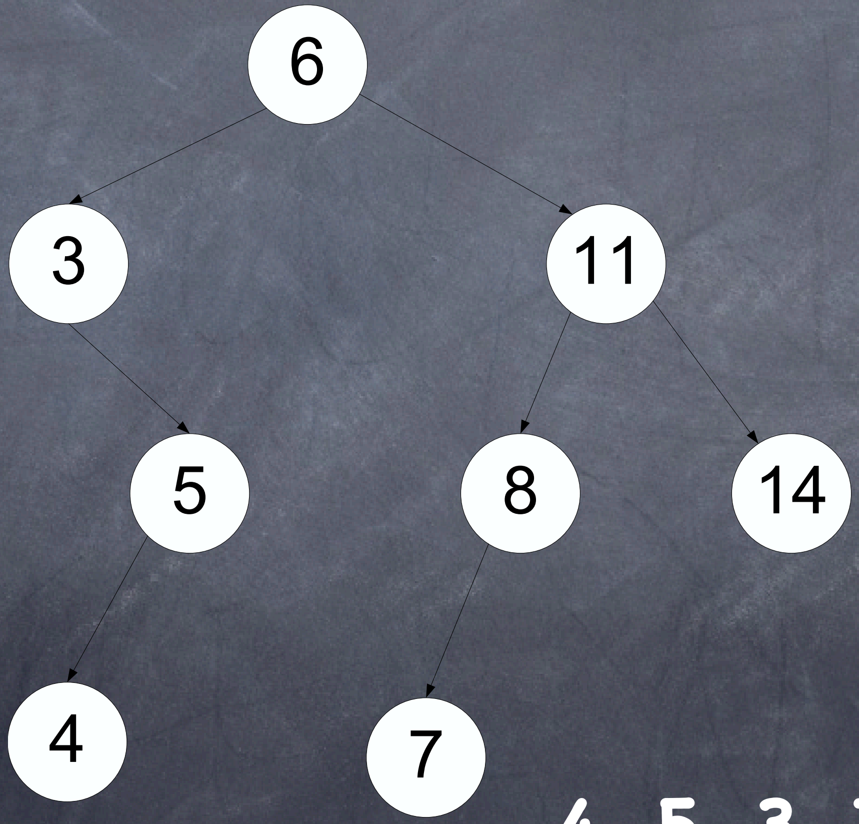
L, N, R



3, 4, 5, 6, 7, 8, 11, 14

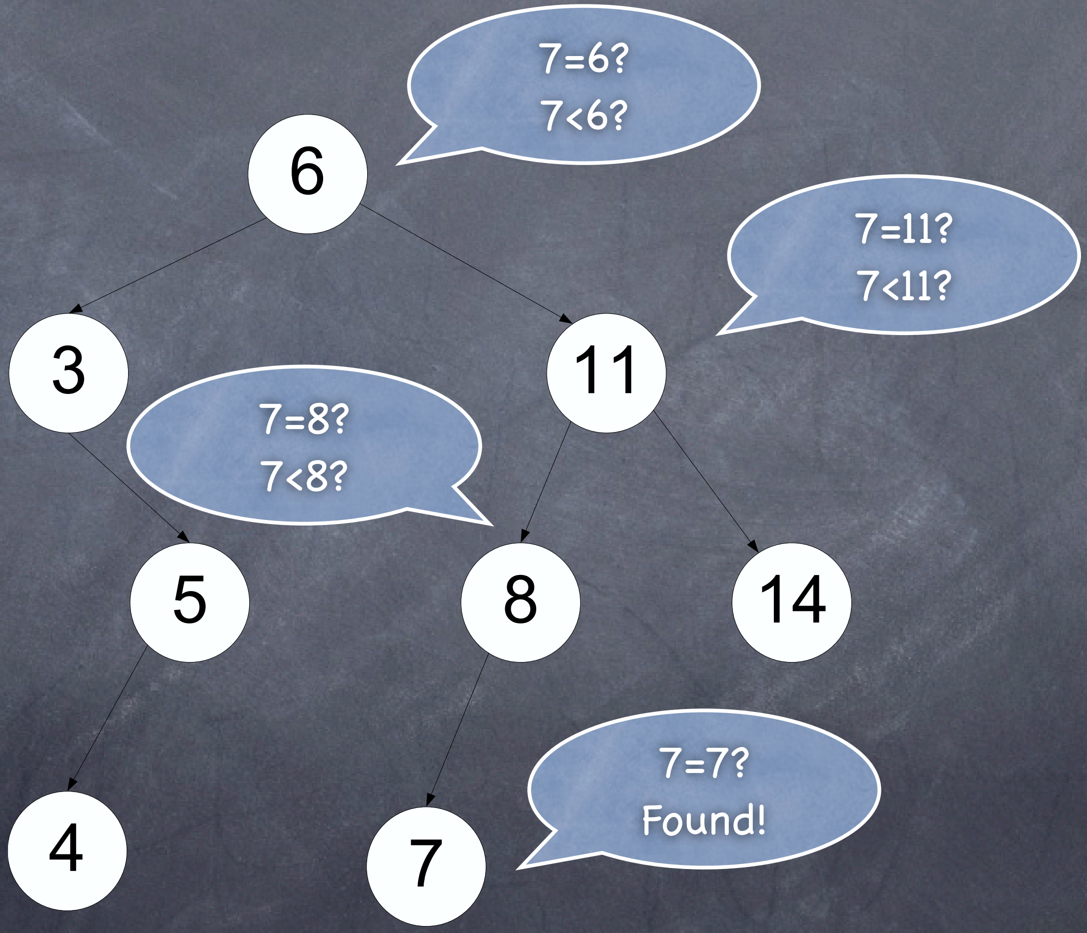
Postorder Traversal

L, R, N



4, 5, 3, 7, 8, 14, 11, 6

Binary Tree Search: 7



Final Thoughts

- Good luck with assignment #2
  - Due Thursday @ 11:59 PM
  - Submit early, backup your code, test thoroughly, sleep :)
- Extra challenge: tree_height
  - See discussion notes