Lecture 2
September 19

Intro to recursion and trees

Recursion

Before we study the tree data structure, we need to understand recursion.

Recursion is a problem solving technique.

The code always has 2 parts:

1. Base case
2. Recursive case:
    a. Make recursive call with a **smaller problem** size.
    b. Combine the results of recursive call.

Base case is usually trivial.

A very simple recursive program

You are a lazy class monitor, and you have been asked to add the marks of every student in class (15 students).

1. Array sum using recursion (Write the code in sublime text).
2. Array sum using recursion (Split the array in 2 halves).

Ways of returning value from a recursive code

Three ways that you can return values from a recursive code:

1. Return a value.
2. Pass an integer by reference and update its value.
3. Use a global variable instead of passing variable by reference.

   (Go back to the sum of marks example).

Let us look at binary search again

Write the code again in sublime text.

What are the recursive calls doing? They are looking at exactly one element (the mid one), and then handing off the rest of the array to someone else for finding that number.

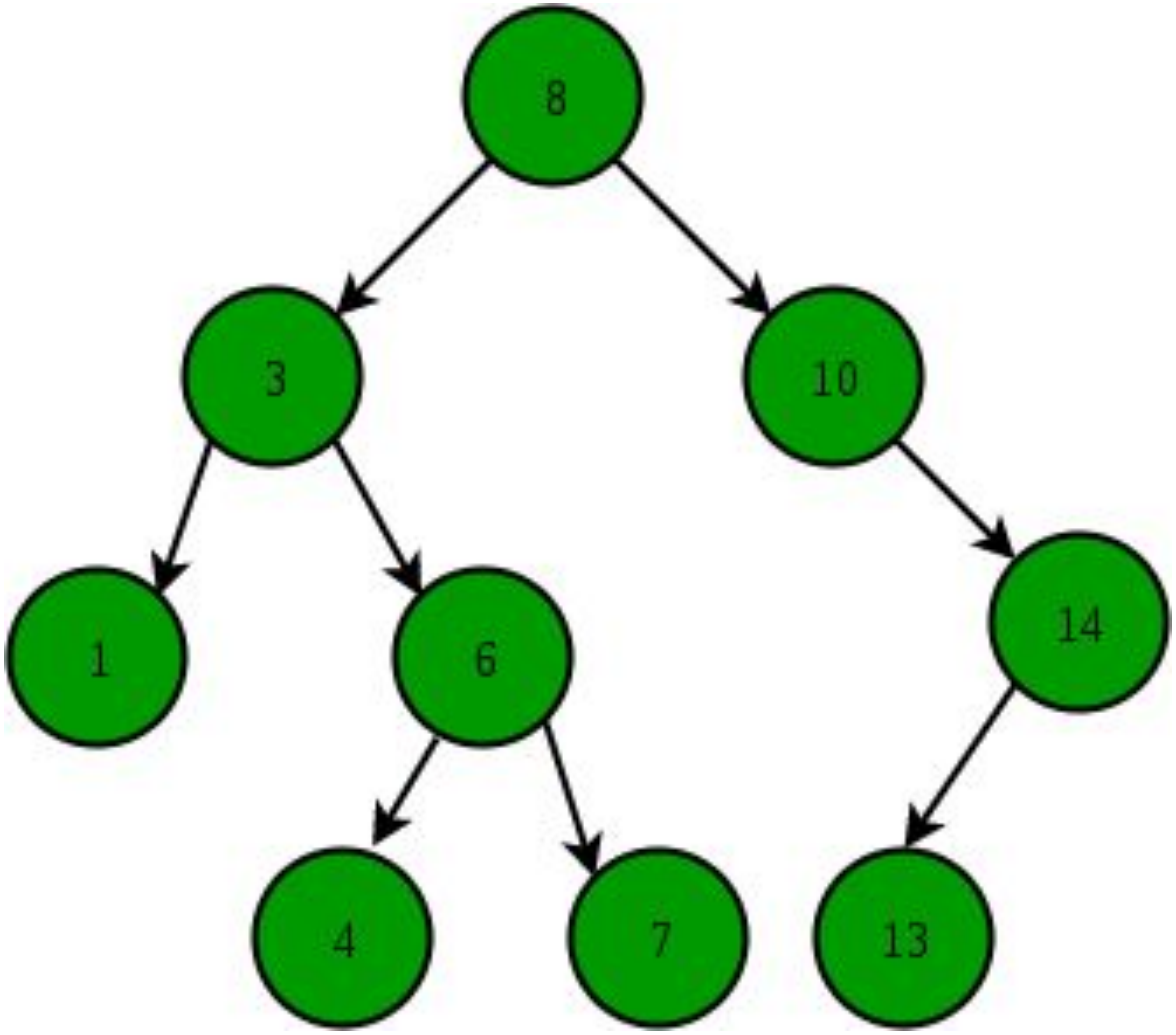
Trees and recursion

Trees lend themselves beautifully to this kind of problem solving.

But before we do that, let us learn some terminology:

1. Binary tree (no restriction on elements).
2. Binary search tree: Everything in the right subtree is larger and left subtree is smaller than the root.
3. Leaf node: A node with no children. Doesn’t necessarily have to be at the last level.
4. Height of a tree: Longest root to leaf path.

Trees and recursion



Trees and recursion

Trees lend themselves beautifully to this kind of problem solving.

Example problems:

1. Find the maximum number in a tree.
2. Find the height of a tree.
3. Tree traversals (basically, convert tree to array).
4. Take a sorted array and construct a binary search tree.

Recursion again

Always has 2 parts:

1. Base case
2. Recursive case:
    a. Make recursive call with a smaller problem size.
    b. Combine the results of recursive call.

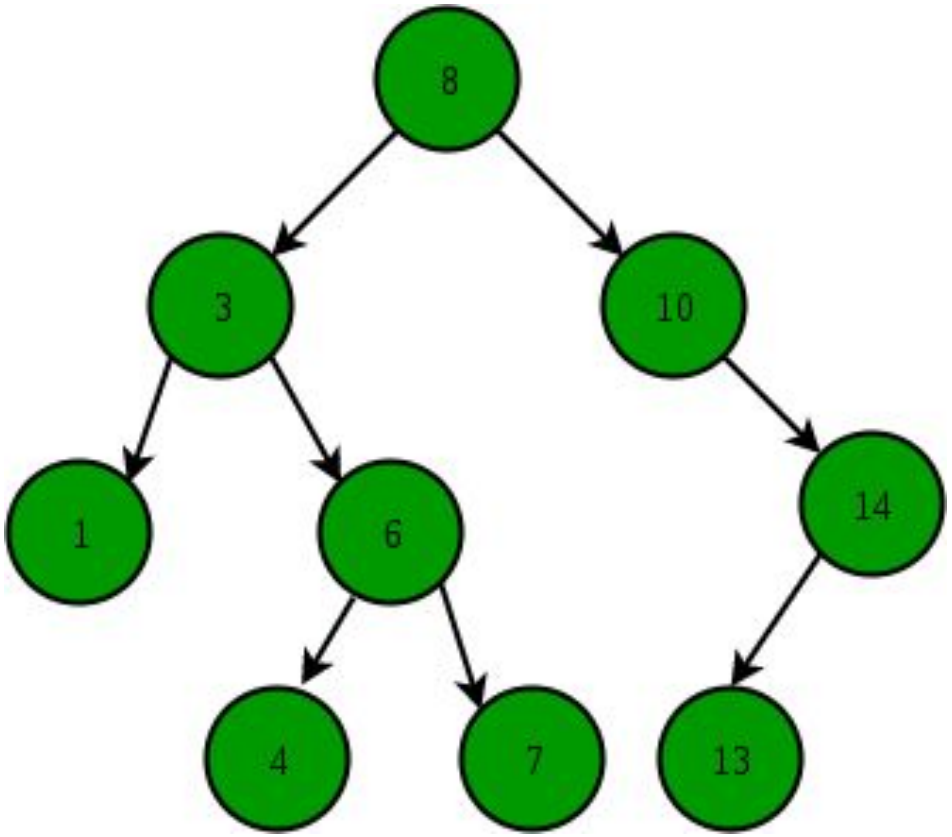
When you are making recursive calls, those calls are solving the exact same problem, but the size is smaller. Go back to the BuildTree example.

Problem: Root to leaf path sum

Root to leaf path:

Any path that goes from the root node to a leaf node.

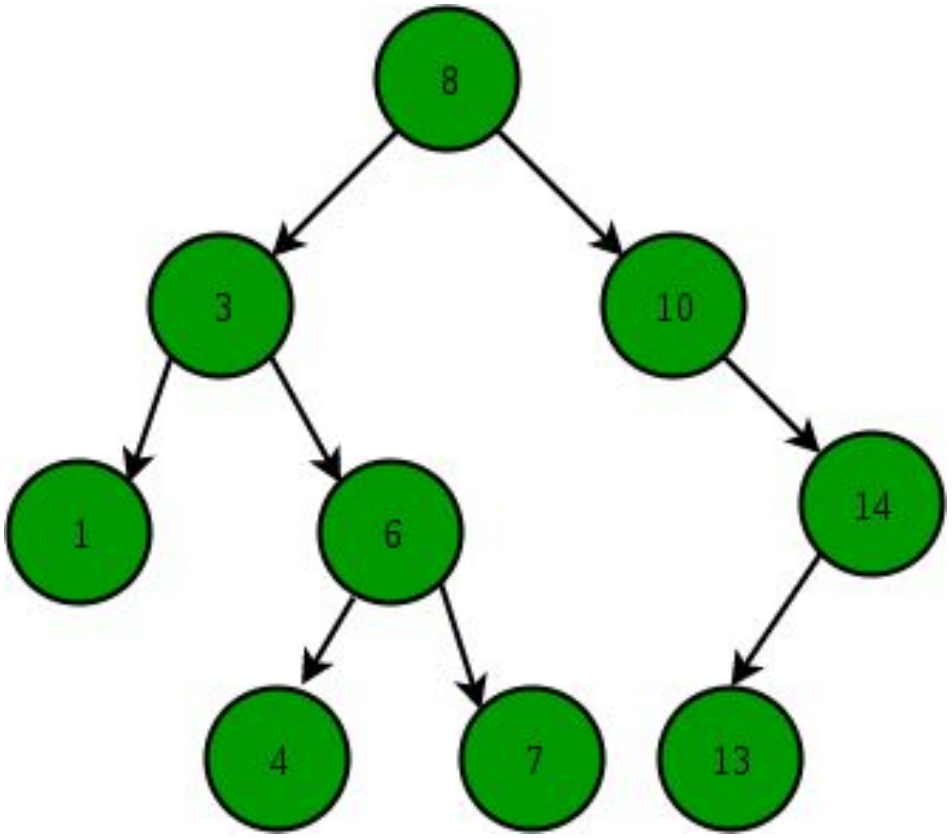
Print the sums of all root to leaf node paths.



Problem: Root to leaf path sum

Code execution is always linear!

At one point, only a given recursion path is being executed, the rest of the paths are waiting.



Homework: Binary Insertion Sort

Homework for you:

You are a lazy class monitor, but this time, you have been asked to sort the students answer sheets by roll number.

```
vec.insert (vec.begin()+5,13);
```