Diagram-First Template

The most common approach to teach a complex topic is what we call the "Diagram-First" approach.

We start with a brief introduction (a paragraph or two).

Then we move right into describing a scenario problem to be solved. (Sometimes referred to as a "concrete example".)

5

Tree Structures

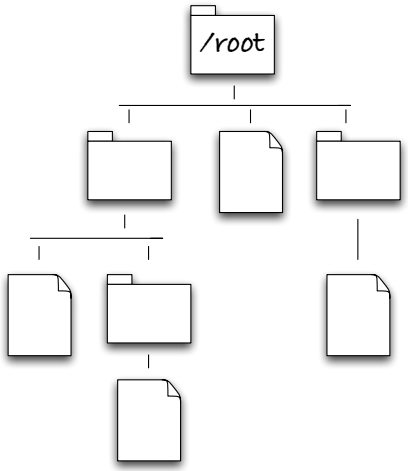
In this chapter we will discuss one of the most frequently used data structures, blah, blah, blah, Tree Structures.

Suppose you wanted to write a program to search through your computer and find all of the the .pdf files.

Scenario Diagram(s)

Once we've described the scenario, it's powerful to reinforce and clarify it with a diagram or two…

Introduce the graphic with a little text…



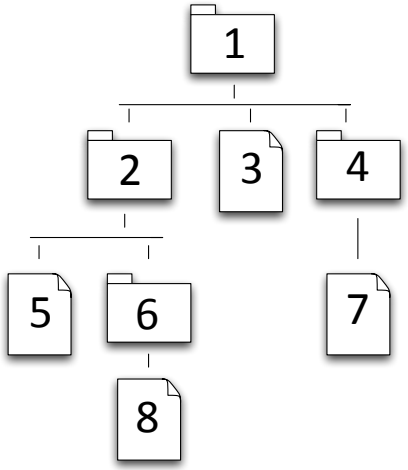
Discuss the graphic, e.g.,

Notice that the directory structure of your computer is basically a tree structure. If we had to do this search manually with a pencil and paper, how might we search through the tree systematically?

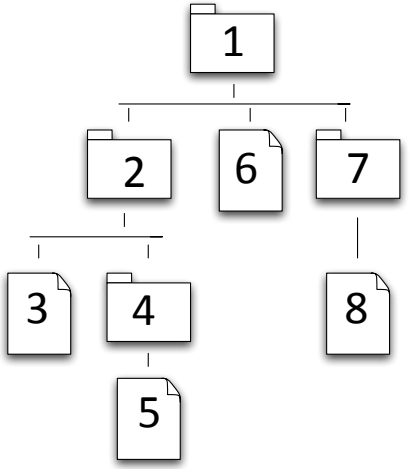
First Step Diagram(s)

Next we might start to describe solutions with a few diagrams…

We could search the tree row by row, which is known as a “breadth-first” search.



Or we could search entire branches as we discover them, a so-called “depth-first” search.



Solution code, version 1

Once we've described the scenario, it's powerful to reinforce and clarify it with a diagram or two…

Introduce the code with a little text…

```
Import java.util.*;
public class DepthSearcher {
  p.s.v. main() {
    Tree t = new Tree();   // #a
    t.go();
  }
  void go() {
    // open root dir 'd'
    myRecurse(d);
  }
  boolean myRecurse(File f) {
    // do recursive stuff
    if(blah) myRecurse(newF);  // #b
   }
 }

#a More discussion of line a
#b More discussion of recursion
```

Discuss the code, e.g.,

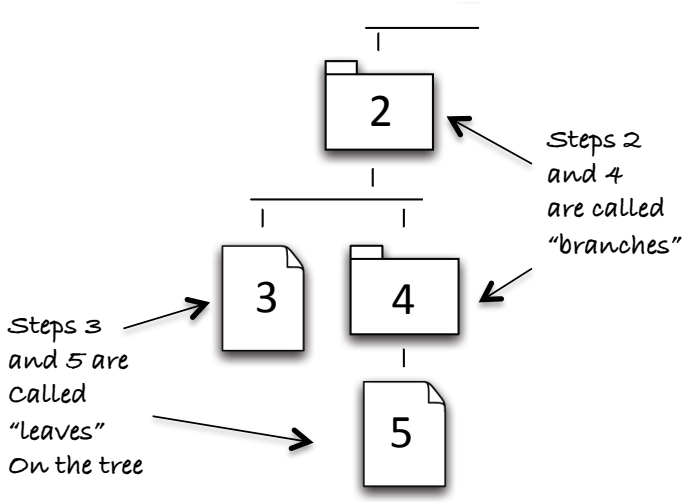
Notice that this code represents the directory structure using the Tree class from java.util. The recursive method is doing some tricky stuff , so let's look at it more closely…

Discuss v1 code

Since we're teaching a complex topic, it's common that we'll need more than one page to discuss the code…

Another common technique – using the "forward flow" authoring pattern – is to zoom in on a bit of the problem, perhaps both the diagram and the code.

Introduce the zoomed-in graphic with a little text…



Zoom in on the code…

```
boolean myRecurse(File f) {
  // blah
  // blah      #a
  if(blah) myRecurse(newF);
}

#a And here we add a long annotation
that discusses branches and leaves
and how the code notices them and deals
with them differently… blah, blah..
```

More D.T.C. Iterations

Once we've made a first pass, we might want to look at another aspect of this topic and we will iterate through another set of:

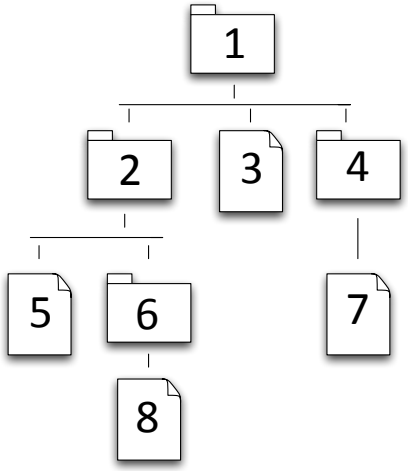
-Diagrams

-Text

-Code

-In this example we might go back and look at "breadth first" searches…

Intro: "As you recall, earlier we mentioned "breadth-first" searches….



Discuss the differences, e.g.,

Unlike in depth-first searches, the breadth first search will remain on the current level of the tree until all of the nodes on that level have been visited…

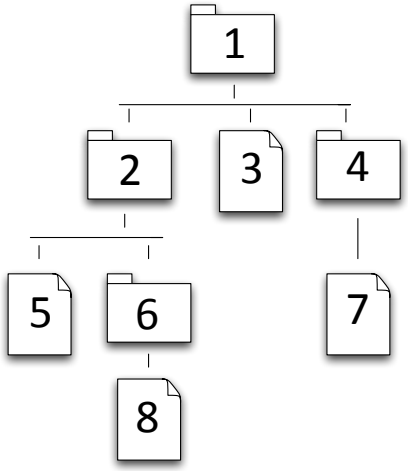
Blah,
Blah,
Blah…

Add an Exercise

Exercises can be formal elements in a chapter or they can be suggestions.

It's powerful to use exercises to reinforce ideas you've just taught and you can also use exercises to "slightly" expand into new territory…

Let's look at the breadth-first search one more time…



Here are a few tips for coding a breadth-first search:
1 – remember to…
2 – notice that leaves don't have the same meaning…
3 – sdfsdfsdf

With those tips in mind, try changing the code we've been working on to do a breadth-first search. Our solution is on the next page.

Exercise solution

And then we show our solution, with some explanations

Introduce the solution code with a little text…

```
Import java.util.*;
public class DepthSearcher {
  p.s.v. main() {
    Tree t = new Tree();
    t.go();
  }
  void go() {
    // open root dir 'd'
    myRecurse(d);
  }
  boolean myRecurse(File f) {
    // do recursive stuff
    // do breadth stuff        // #a
    if(blah) myRecurse(newF);
   }
 }

#a Here we discuss where this code
differs from the earlier version
```

Discuss the code, e.g.,

Notice the differences in the myRecurse() method…

Use D.T.C. to Generalize

Once we've taught a useful subset of the topic (as in the first 8 pages here), we should end the topic by teaching broader, more general, and/or more abstract ideas.

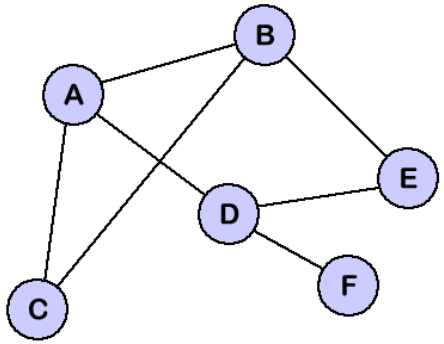
After this page we might go on to discuss topics like performance characteristics, common uses for each traversal approach and so on.

Advanced Tree Structures, Uses and Performance Concerns

We've discussed normal trees and looked at several traversal approaches.

Trees can become more complex when we add symbolic links….

Another important subset of Tree data structures are graphs. In graphs, the notions of branches and leaves are not used because graphs allow for circular connections as shown below.



In the example above, notice how A, B, and C form a closed loop. None can said to be the leaves of another…