22.2 Tree Traversals

Tree ~~Iteration~~ Traversal

Remember how we learned to iterate through lists? There was a way to iterate through lists that felt natural. Just start at the beginning... and keep going.

Or maybe we did some things that were a little strange, like iterate through the reverse of the list. Recall we also wrote iterators in discussion to skip over students who didn't write descriptions on the Office Hours queue.

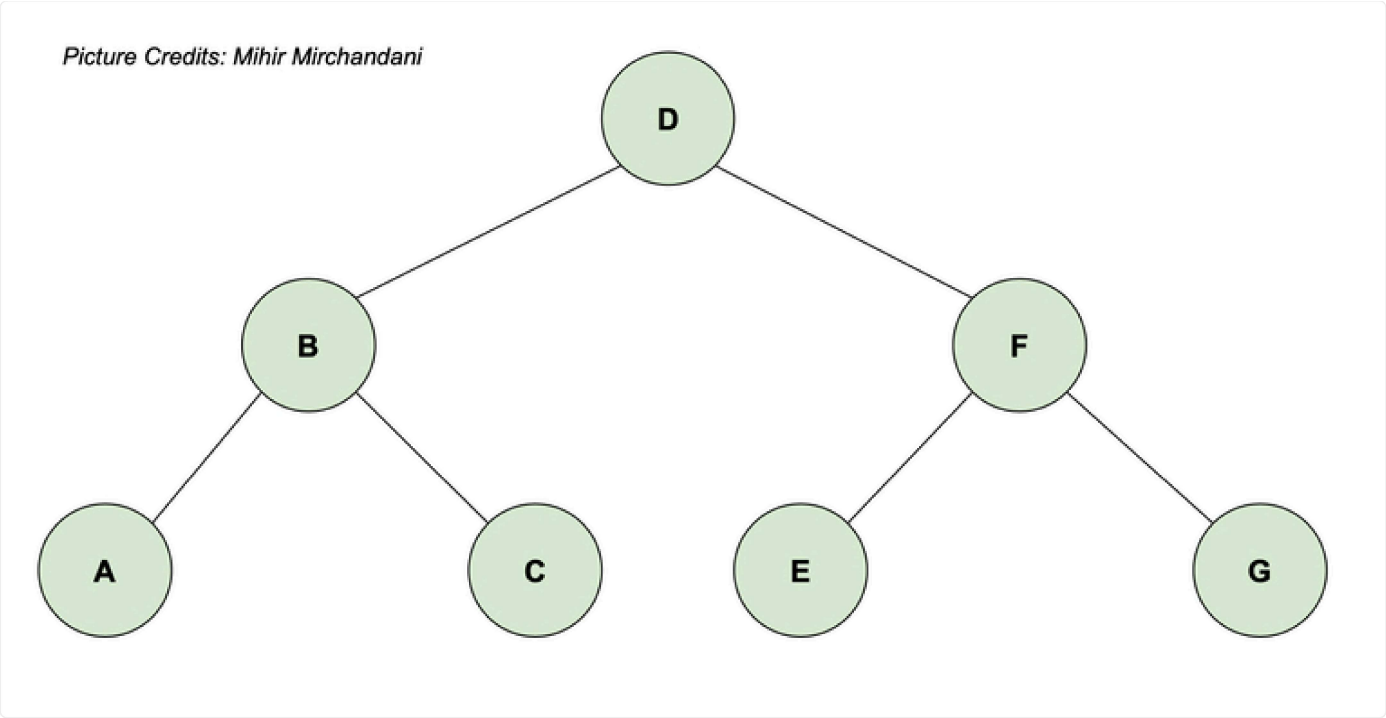
Now how do you iterate over a tree? What's the correct 'order'?

Before we answer that question, we must not use the word iteration. Instead, we'll call it 'traversing through a tree' or a 'tree traversal'. Why? No **real** reason, except that everyone calls iteration through trees 'traversals'. Maybe it's because the world likes alliterations.

So what are some natural ways to 'traverse' through a tree? As it turns out, there are a few — unlike a list which basically has one natural way to iterate through it:

1. Level order traversal. Breadth First Search (BFS)
2. Depth-First traversals — of which there are three: pre-order, in-order and post-order. (DFS)

Let's test out these traversals mentioned above on the tree below.



Different Traversals on One Tree...

Level Order Traversal

We'll iterate by levels, left to right. Level 0? D. Level 1? B and then F. Level 2? A, C, E, and G.

This gives us `D B F A C E G`.

Imagine each level was a sentence in an english book, and we just read it off line by line.

Exercise 17.2.1. Write the code for performing a level order traversal (warning, this is more difficult than the writing the other traversals). *Hint*: You will want to keep track of what level you are at.

Pre-order Traversal

Here's the idea behind pre-order traversal. Start at the root. **Visit the root** (aka, do the **action** you want to do.) The action here is "print".

So, we'll print the root. D. There we go.

Now, go left, and recurse. Then, go right and recurse.

So now we've gone left. We're at the B node. Print it. B. We'll go left after printing. (Remember, after we're done with B's left branch, we'll come back up and visit B's right.)

Keep following this logic, and you get `D B A C F E G`.

```
preOrder(BSTNode x) {
    if (x == null) return;
    print(x.key)
    preOrder(x.left)
    preOrder(x.right)
}
```

In-order Traversal

Slightly different, but same big-picture idea. Here, instead of **visiting** (aka **printing**) first, we'll first visit the left branch. Then we'll print. Then we'll visit the right branch.

So we start at D. We don't print. We go left.

We start at B. We don't print. We go left.

We start at A. We don't print. We go left. We find nothing. We come back up, and print A.

Then go to A's right. Find nothing. Go back up. Now we're at B. Remember, we print after visiting left and before visiting right, so now we'll print B, then we'll visit right.

Keep following this and you get `A B C D E F G`.

An alternative way to think about this is as follows:

First, we're at D. We know we'll print out the items from left, then D, then items from right.

[items from left] D [items from right].

Now what's [items from left] equal to? We'll start at B, print out left, then print B, then print stuff from right of B.

[items from left] = [items from B's left] B [items from B's right] = A B C.

A B C D [stuff from right] = A B C D E F G.

```
inOrder(BSTNode x) {
    if (x == null) return;
    inOrder(x.left)
    print(x.key)
    inOrder(x.right)
}
```

Post-order Traversal

Again, same big-picture idea, but now we'll print left branch, then right branch, then ourselves.

Using the method we devised earlier, the result looks like:

[items from left] [items from right] D.

What's [items from left]? It's the output from the B subtree.

If we're at B, we'd get [items from left of B] [items from right of B] B, which is equal to A C B.

Following this through, we get: `A C B E G F D`

```
postOrder(BSTNode x) {
    if (x == null) return;
    postOrder(x.left)
    postOrder(x.right)
    print(x.key)
}
```

22.1 Tree Recap

Next
22.3 Graphs

Last updated 1 year ago