CSCI 2270

Data Structures & Algorithms

Gabe Johnson

Lecture 7 Jan 30, 2013

Binary Search Trees
Computational Complexity

Upcoming Homework Assignment

HW #2 **Due: Friday, Feb 1**

Binary Search Trees (Recursion)

Tests are up on GitHub in C++ and Python. I will try my best to get the Python one up later today. The driver file has changed since I initially released it. The new version (heavily based on something Alec Thilenius wrote) has color and is arguably easier to use. *It is also what the server uses to grade your assignment, so you may try it out on your machine first using the --retrograde command line flag.*

Lecture Goals

1. Announcements
2. About Recursion
3. RetroGrade tests
4. Coding BSTs
5. Intro to Comp. Complexity

Announcements

1. **Command Line Tutorial**
   Host: Andrew Sayler
   Date: Jan 30
   Time: 7:30PM
   Where: ECCR 105
   (this will be recorded and made available later)

Recursion

Recursion is a ubiquitous concept in computer science, programming, and math.

For programming, it is typically used in problems that can be either predictably subdivided, or if a solution can be transformed into a simpler case for which the solution is known.

Uses of Recursion

Recursion is used in lots of situations: data structures, parsing programming languages, data mining and machine learning, not to mention all the mathematical uses.

Here's an example of using recursion to parse a programming language...

Recursive Language Parsing

y = 3
x = (4 + y) + (8 * y + 10)

1. Evaluate Expression (x = (4 + y) + (8 * y + 10))
   (recurse)
2. Evaluate Expression ((4 + y) + (8 * y + 10))
   (recurse)
3. Evaluate Expression ((4 + y))
   (return 7)
4. Evaluate Expression ((8 * y + 10))
   (return 34)
5. Evaluate Expression (7 + 34)
   (return 41)
6. Evaluate Expression x = 41
   (return 41)

Hints at Coding BSTs

Fake code.

```
insert(top, new_node)
  if top is empty, set top := new_node
  otherwise, insert_recursively(top, new_node)
```

The `insert` function just calls another version of `insert` that uses recursion. So what's that look like?

Insert Recursively

More fake code.

```
insert_recursively(top, new_node)
  if new_node's data < top's data,
    if top has no left child, set it to new_node
    otherwise, insert_recursively(top.left, new_node)
  otherwise,
    if top has no right child, set it to new_node
    otherwise, insert_recursively(top.right, new_node)
```

This is just one way of doing it. I can think of a couple, but this is the way I did it.

An heuristic for recursion

The trick is to look at the situation, and determine one of three things:

1) We're in a state where we can complete a task and return,
2) We are not yet ready but the answer might be 'downstream', where we can use information we have to recurse deeper, or
3) We are not yet ready and we're sure we won't be able to do it based on what we see.

Transform/Simplify

In other words, can we transform the problem into a simpler problem for which the solution is (possibly) known or achievable?

E.g. inserting:

“I can’t insert a node here because I have two children already. Ask one of them to insert your node.”

For more recursion info:

The Shaffer text has a short chapter on recursion. In my copy (C++ version) it is page 36, section 2.5.

If you are still baffled, he has a different way of explaining it in slightly more mathy terms.

The new RG driver

Alec made a really cool C++ framework for unit testing that I like more than the Google Testing Framework.

I replaced the driver I wrote with his.

And: this is the same thing RetroGrade uses to test your code. This means you can test on your own machine.

Use ./binary_search_tree_driver --retrograde to see how your code will do on the server.

Java and Python

The Java version is up already.

Python will be up later today, since I am finally (mostly) caught up.

Thanks for reducing the volume of email.

Coding Example

Here is the part where I fire up Emacs and we hack out a few functions together. It will be fun.

I will put the output up on GitHub. To find out where, check the UPDATES.md file in the root, or watch the commit log.

Computational Complexity

Shaffer Chapter 3 (page 55 of C++ version)

This is a mathematical way of reasoning about how much *time* (or *operations)* or *space (memory)* an algorithm or data structure requires.

Thought Experiment

Say I have a gigantic list of numbers. Millions of them. Seventeen digit beasts. They are not really in any sort of sensible order.

How many numbers do you expect to look at before finding the number 37438432029 34?

Or concluding that it isn't in there?

Big Oh

With an unsorted list of numbers, you might have to look through all of them to find it. You'll certainly have to look through all of them to determine it isn't there, and that I've been wasting your time.

In other words:

Given an unsorted array of size **n**, the algorithm to find a specific value is written as **O(n)** and pronounced 'big oh of n', or just 'oh of n'.

What's the Complexity?

Say you are confronted with the following code:

```
for (int i=0; i < N; i++) {
  for (int j=0; j < N; j++) {
    product[i][j] = i * j;
  }
}
```

What's the computational complexity of this algorithm? How many times will we run the inner loop?

Survey Says...

```
for (int i=0; i < N; i++) {
 for (int j=0; j < N; j++) {
  product[i][j] = i * j;
 }
}
```

The runtime complexity of the above is **$O(n^2)$**. Stare at this and convince yourself I am not lying. (I'm not, this time.)

Complexity of sorted things

Say we have a *sorted* data structure containing a billion numbers.

How many slots *on average* do we have to look in to determine if it has a particular value?

How many slots *in the worst case scenario* do we have to look in? The *best case*?

Complexity of BSTs

If the sorted data structure is a plain binary search tree, the best/average/worst-case scenarios are:

Best: **O(1).** The droid we are looking for is on top.

Average: **O(log n).** Happens when the tree is balanced.

Worst: **Survey the audience?**

Complexity of BSTs

(answer from previous: **O(n)**)

How about the insert function?

Remove?

Size?

Consider this an ungraded homework assignment. Wikipedia is *wonderful* in this regard.