CS 209
Data Structures and Mathematical Foundations

03 / 20 / 2024

Instructor: Michael Eckmann

Today's Topics

- Questions/Comments?
- Recursion

Recursion

- 1. have at least one base case that is not recursive
- 2. recursive case(s) must progress towards the base case
- 3. trust that your recursive call does what it says it will do (without having to unravel all the recursion in your head.)
- 4. try not to do redundant work. That is, in different recursive calls, don't recalculate the same info.

Recursion

- Need to decide if the recursive function will return some value, or not return a value.
- If it is to return a value, then
  - *Every* call to it needs to **capture the returned value**
  - e.g. result = funrec(x)

    or return funrec(x)
  - And a **return statement** must occur for any inputs

Recursion

- The last example showed that recursion didn't really simplify our lives, but there are times when it does.
- e.g. If given an integer and you wanted to print the individual digits in order, but you didn't have the ability to easily convert an int >10 to a string.
- e.g. n=35672
- If we wanted to print 3 first then 5 then 6 then 7 then 2, we need to come up with a way to extract those digits via some mathematical computation.
- It's easy to get the last digit n%10 gives us that.
- Notice: 35672 % 10 = 2 also 2 % 10 = 2.
- Any ideas on a recursive way to display all the numbers in order?

Recursion

```
def print_digits(n):
    if n < 10:
        print(str(n), end='')
    else:
        print_digits((n//10))
        print(str(n%10), end='')
```

// what's the base case here?

// what's the recursive step here? Will it always approach the base case?

Recursion

- Now that last problem was "made up", because python (and most languages) allow us to print ints.
- However what if we wanted to print the int in a different base? Say base 2, 3, 4, 5, or some other base?

Recursion

Let’s go back to the fibonacci code from last time.

Any problems with that code?

Yes – it makes too many calls. And further, these calls are redundant.

It violates that 4th idea of recursion stated earlier: in different recursive calls, don't recalculate the same info.

Recursion

- We know what a tree is.
- Here's a recursive definition of a tree:
  - A tree is empty or it has a root connected to 0 or more subtrees.
  - Note a subtree, taken on its own, is a tree. Because a tree is being defined in terms of other trees, it is a recursive definition.

Recursion

- Let me write insert recursively in the BinarySearchTree code.
- Let me also write find_max iteratively and then again recursively

Recursion

- Let’s use an idea called memoization to make the fibonacci numbers code much more efficient runtime

- Idea is:
  - save computed fibonacci numbers in a table when computed
  - when need a fibonacci number, check table first to see if it has been computed already, if so use it, if not, make recursive call