Lesson 25: Recursive Algorithms #2 (W07D4)

Balboa High School

Michael Ferraro

October 2, 2015

Do Now

1. Write down the **two** factors that all recursive procedures *must* have.
2. On paper, determine the output of the recursive procedure below.

```
public class DoNow {
    public static int doNowFcn(int x) {
        if ( x == 1 ) {
            return -1;
        }
        return -1 * x + doNowFcn( x - 1 );
    }

    public static void main(String[] args) {
        int result = doNowFcn(4);
        System.out.println("Result:  " + result);
    }
}
```

Aim

Students will work with recursive versions of *Sum of Squares* and *Euclid's GCF Algorithm*, and practice tracing the execution of recursive procedures.

Stack Overflow!

- You may recall that a missing base case in one of last class' examples produced this error:

```
Exception in thread "main" java.lang.StackOverflowError
```

- Each time a recursive method calls itself again (recurses *deeper*), more memory is used. The data that will be needed later is `pushed` onto a *stack*.
- A stack is a memory structure — items are pushed down onto it, like Pez candies into a Pez dispenser.
- Need data from the stack? `pop` it off — like removing a Pez candy from a dispenser.

Stack Overflow!

- BYOB[1] is a derivative project of MIT Scratch, and Snap! is BYOB in the Cloud.
  - developed at UC Berkeley
  - supports methods/functions/*blocks*
- Save this XML file to your Desktop: `FactorialRecStack.xml`
- Visit Snap!, import the XML file you downloaded
- Let's edit the `factorialRec` block, breaking the base case...

[1] see `http://byob.berkeley.edu/`

Sum of Squares, Part II

- How could you think of `findSumOfSquares()` in recursive terms?

  `findSumOfSquares(n)` $= 1^2 + 2^2 + ... + n^2$

Sum of Squares, Part II

- How could you think of `findSumOfSquares()` in recursive terms?

  `findSumOfSquares(n)` $= 1^2 + 2^2 + ... + n^2$

- First, consider running the sum in reverse:

  `findSumOfSquares(n)` $= n^2 + (n-1)^2 + ... + 1^2$

Sum of Squares, Part II

- How could you think of `findSumOfSquares()` in recursive terms?

  `findSumOfSquares(n)` $= 1^2 + 2^2 + ... + n^2$

- First, consider running the sum in reverse:

  `findSumOfSquares(n)` $= n^2 + (n-1)^2 + ... + 1^2$

- Rewrite the problem in terms of itself:

  `findSumOfSquares(n)` $= n^2 +$ `findSumOfSquares(n-1)`

Sum of Squares, Part II

- How could you think of `findSumOfSquares()` in recursive terms?

  `findSumOfSquares(n)` $= 1^2 + 2^2 + ... + n^2$

- First, consider running the sum in reverse:

  `findSumOfSquares(n)` $= n^2 + (n-1)^2 + ... + 1^2$

- Rewrite the problem in terms of itself:

  `findSumOfSquares(n)` $= n^2 +$ `findSumOfSquares(n-1)`

- Base case: What's the least value of n for which we should compute/evaluate `findSumOfSquares(n)`?

Sum of Squares, Part II

- How could you think of `findSumOfSquares()` in recursive terms?

  `findSumOfSquares(n)` $= 1^2 + 2^2 + ... + n^2$

- First, consider running the sum in reverse:

  `findSumOfSquares(n)` $= n^2 + (n-1)^2 + ... + 1^2$

- Rewrite the problem in terms of itself:

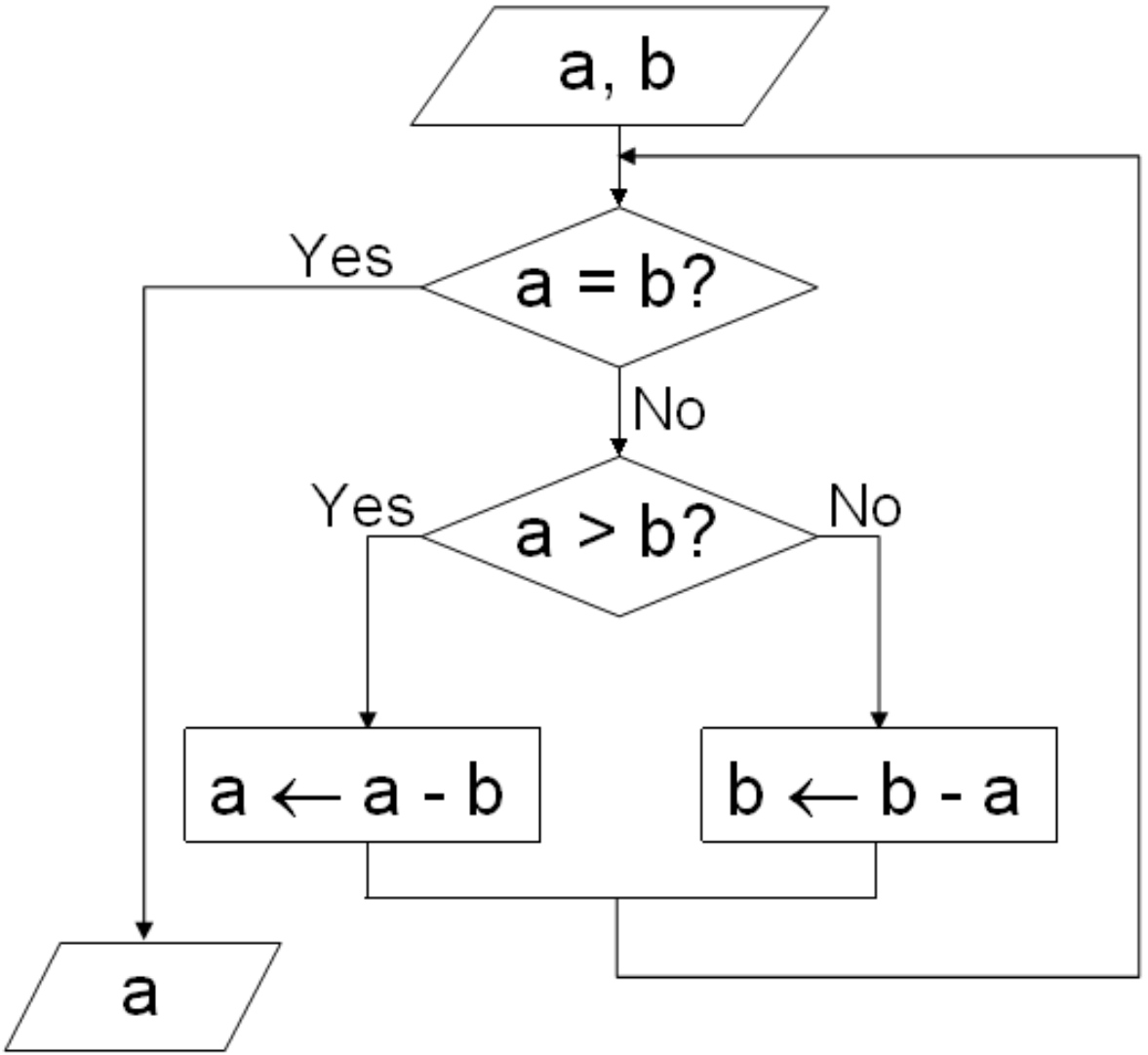
  `findSumOfSquares(n)` $= n^2 +$ `findSumOfSquares(n-1)`

- Base case: What's the least value of n for which we should compute/evaluate `findSumOfSquares(n)`?

- With a partner, write and test a new class in project `Lesson25` called `SumOfSquaresRec`.

Back to Euclid's GCF

Iterative version in a flowchart:



Back to Euclid's GCF

Iterative version in Java:

```
public static int gcf(int a, int b) {

    while ( a != b ) {

        if ( a > b ) {
            a -= b;
        } else {
            b -= a;
        }

    }

    return a;
}
```

Back to Euclid's GCF

Tracing evaluation of iterative `gcf(15,24)`:

a	b	logic	
15	24	since $b > a$,	$b \leftarrow b - a$
15	9	since $a > b$,	$a \leftarrow a - b$
6	9	since $b > a$,	$b \leftarrow b - a$
6	3	since $a > b$,	$a \leftarrow a - b$
3	3	since $a = b$,	return a

Finding GCF Recursively

- To find `gcf(15,24)`, return `gcf(15,9)`...

Finding GCF Recursively

- To find `gcf(15,24)`, return `gcf(15,9)`...
- To find `gcf(15,9)`, return `gcf(6,9)`...

Finding GCF Recursively

- To find `gcf(15,24)`, return `gcf(15,9)`...
- To find `gcf(15,9)`, return `gcf(6,9)`...
- ...

Finding GCF Recursively

- To find `gcf(15,24)`, return `gcf(15,9)`...
- To find `gcf(15,9)`, return `gcf(6,9)`...
- ...
- `gcf(3,3)` is 3.

Finding GCF Recursively

Sketching out the code:

```
gcf(int a, int b) {

    //base case
    if ( ? ) {
        ...
        return ?;
    }

    // oversimplification:
    return gcf( ?, ? );
}
```

With your partner from earlier, write a program that finds GCFs recursively.

HW

- **Reminder**: PS #4a is due at the beginning of next class!
- PS #4b:
  - §5.1, required reading
  - §5.4, trace recursive `mysterySum()`